UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA

Departamento: INGENIERÍA ELECTRÓNICA Y AUTOMÁTICA

Programa de Doctorado: INGENIERÍA DE TELECOMUNICACIÓN AVANZADA

Título de la Tesis

METODOLOGÍA PARA LA VERIFICACIÓN DE SISTEMAS DE CONMUTACIÓN DE PAQUETES

Tesis Doctoral presentada por D. Carlos Javier Sosa González

Dirigida por el Dr. D. Juan Antonio Montiel Nelson

El Director, **El Doctorando,**

Las Palmas de Gran Canaria, a 23 de Junio de 2006

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

A Minerva, Aday,
Nita y Pepe.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

Agradecimientos

A Minerva, que ha sido el contrapunto de mi vida. Yo que si, y ella que no. Si, no no no. Pues al final no me queda claro y triunfa tu no. Como novia, ahora como esposa y madre de un tal Aday alias 'el pollito del cruce', chapó.

No tengo que olvidar a Nita y Pepe. Que han tragando mucho y en seco, pensando ¿Pero mi niño cuándo vas a presentar este parto? Mama, ya puedes dar el grito. Sólo tengo que alegar a mi favor, que esto no hubiese pasado si no me hubiesen repetido tanto la frase 'estudiar es bueno' ;) Juanita, Ima, Juani y Eloina tampoco me olvido de ustedes que apoyando, apoyando siempre empujando.

Tampoco puedo olvidar a Ana y a Armando que a pesar de no entender que es lo que hago, han sabido estar en esos momentos que hacen falta. También tengo que agradecer a Lourdes y a Alicia todo el cariño que me han ofrecido desinteresadamente como aquellas hermanas de sangre que no tengo.

A Dr. D. Juan A. Montiel Nelson, como Catedrático de Universidad que es, le doy mi más sincero agradecimiento por su inestimable ayuda a lo largo de esta Tesis Doctoral. Nelson como compañero, gracias por escucharme, alentarme y pararme los pies cuando ves algo raro. Aunque está mal que lo diga yo, pero algunas veces las ideas esas locas que tengo funcionan. No creas que por terminar esta Tesis te vas a librar de seguir escuchando esas 'ideitas'.

Héctor, José Carlos no se como decirlo, pero creo que lo mas sencillo es decir gracias por recorrer este camino pasito a pasito a mi lado y con su desinteresada ayuda en todo momento. Nunca olvidaré aquel código en C con sólo variable globales o los ronquidos de München.

A todo aquel que mostró su sorpresa por ver a un elemento como yo diciendo que no podía asistir a un asadero, o celebración porque 'tengo que hacer tesis'. Y una palmadita en la espalda a aquellos usuarios a los que trajinaba los ciclos de cpu y mientras ellos comentaban que les iba más rápido en su PC. Igualmente, gracias a sysadmin por la quota casi infinita del /var/tmp.

Supongo que me dejo a alguien por nombrar, pero tengo prisa para ir a encuadernar estos tomos. Esto ha sido posible en mayor o menor medida a gracias a ustedes.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

Resumen

Esta Tesis Doctoral aporta una metodología y un entorno para la verificación y validación de sistemas integrados de última generación, basados en la exploración del espacio de diseño y la generación guiada mediante diversas métricas de cobertura.

El entorno desarrollado en C++ permite comprobar elementos sencillos, como son los módulos de lógica combinacional o secuencial, o tan complejos, como un sistema integrado de última generación; bien de forma automática o especificando casos críticos. La metodología y el entorno propuesto, son aplicados a la verificación de elementos en fase de desarrollo en cualquier nivel y donde existan descripciones hardware y/o software.

Con el objeto de reducir el número de vectores y el tiempo requerido para comprobar un sistema integrado, el entorno se complementa con diversas técnicas — heurísticas y deterministas — de generación de vectores guiados por métricas de cobertura, basadas en potencia de consumo o ejercitación de rutas.

Se provee una metodología eficiente para explorar el espacio de diseño del sistema bajo verificación con el objeto de obtener una estimación precisa del retardo de cada funcionalidad. Como resultado de la exploración del espacio de diseño no sólo se obtiene el retardo a anotar en cada funcionalidad, sino que se logra el rango de funcionamiento del conjunto de soluciones circuitales óptimas en retado–área o retardo–potencia de consumo.

Tanto la metodología presentada como el entorno que la implementa han sido utilizados ampliamente en la verificación de circuitos integrados comerciales y no comerciales. La consecución de dichos sistemas es una buena garantía de que el entorno de verificación funciona correctamente y que la metodología propuesta es práctica en la verificación de sistemas de integrados.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

Índice general

1. Verificación de Sistemas Integrados **1**
- 1.1. Introducción . . . 3
- 1.2. Diseño de Sistemas de Conmutación de Próxima Generación . . . 5
- 1.3. Verificación Basada en Simulación . . . 14
- 1.4. Entorno de Verificación Propuesto . . . 28
- 1.5. Organización de la Memoria . . . 28

2. Entorno de Verificación **31**
- 2.1. Introducción . . . 33
- 2.2. Trabajos Previos . . . 34
- 2.3. Priorización . . . 38
- 2.4. Modelos de Verificación . . . 38
- 2.5. Visión General del Entorno . . . 43
- 2.6. Interfaces . . . 43
- 2.7. El Sistema de Ficheros . . . 52
- 2.8. Características Avanzadas . . . 55
- 2.9. Implementación Interna . . . 58
- 2.10. Resultados Experimentales . . . 63
- 2.11. Conclusiones . . . 66

3. Generación de Vectores de Máxima Cobertura **69**
- 3.1. Introducción . . . 71
- 3.2. Definiciones de Teoría de Grafos . . . 75
- 3.3. Cobertura Basada en Rutas (CBR) . . . 76
- 3.4. Cobertura Basada en Potencia de Consumo . . . 107

4. Optimización del Retardo **121**
- 4.1. Introducción . . . 124
- 4.2. Trabajos Previos . . . 126
- 4.3. Sinopsis . . . 128
- 4.4. Formulación del Problema de Optimización de la Ruta Crítica . . . 129
- 4.5. Optimización de un Circuito Empleando el Conjunto Ruta Crítica 131
- 4.6. Curva de Prestaciones de un Circuito . . . 132
- 4.7. RTL versus Circuito de Lógica Combinacional . . . 146
- 4.8. Conclusiones . . . 151

5. Exploración del Espacio de Diseño Potencia de Consumo–Retardo **155**
- 5.1. Introducción . . . 157
- 5.2. Trabajos Previos . . . 158
- 5.3. Modelo de Puerta . . . 168

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

5.4. Formulación del Problema de Optimización de la Ruta Crítica . . 169
5.5. Optimización de un Circuito Empleando el Conjunto Ruta Crítica 170
5.6. Curva de Prestaciones de un Circuito 171
5.7. El Problema del Punto de Arranque 177
5.8. Formulación del Dimensionado de Puerta Empleando Programación Lineal 188
5.9. Resultados Experimentales 195
5.10. Conclusiones 203

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

Índice de figuras

1.1. Impacto de la metodología de diseño en el costo de implementación de sistemas. . . . 4
1.2. Densidad de integración frente a productividad. . . . 4
1.3. Arquitectura de un conmutador. . . . 6
1.4. El ciclo de diseño. . . . 8
1.5. Ejemplo de un sistema de conmutación en fase de desarrollo. . . . 21
1.6. Las capas de software para la interfaz entre C++ y el código RTL. En la parte superior el Dispositivo Bajo Verificación (DBV) y en la inferior el modelo de referencia (*Golden Reference*). . . . 22
1.7. Refinamiento del modelo de comportamiento hasta nivel RTL en C++. . . . 23
1.8. Verificación respecto al *Golden Reference*. . . . 24
1.9. Representación gráfica del registro de simulación C++ o RTL. . . . 25
1.10. Verificación con especificación de reglas. . . . 26
1.11. Distintas formas de generar la especificación de reglas. . . . 27
1.12. Entorno de verificación propuesto. . . . 29

2.1. Verificación de un sistema integrado. . . . 37
2.2. Esquema del modo de verificación de casos críticos. . . . 39
2.3. Esquema del modo automático de verificación. . . . 41
2.4. Entorno de desarrollo de bancos de prueba. . . . 44
2.5. Diagrama de bloques de la interfaz paralela. . . . 45
2.6. Diagrama de bloques de la interfaz serie. . . . 48
2.7. Diagrama de bloques de la interfaz de CPU. . . . 51
2.8. Generación de células de datos, relleno y ráfaga en la interfaz serie. 54
2.9. Sincronización de la interfaz paralela. . . . 56
2.10. Detalle del módulo de generación de *jitter* en el diagrama de bloques de la interfaz serie. . . . 57
2.11. Esquema a nivel de bloques de las librerías implementadas. . . . 58
2.12. Comparativa de la evolución de la cobertura alcanzada en función del lenguaje empleado. . . . 64

3.1. Ejemplo de cobertura basada en rutas: (a) circuito combinacional, (b) grafo del circuito, (c) ejercitación del $path_4$. . . . 78
3.2. Ejemplo de cobertura basada en rutas con ejercitación múltiple: (a) circuito combinacional a verificar, (b) verificación simultánea de múltiples rutas. . . . 81
3.3. Modelo básico de la puerta AND para la propagación de transición. 83
3.4. Modelo básico de la puerta OR para la propagación de transición. . 84
3.5. Modelo básico de la puerta INV para la propagación de transición. . 85

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

3.6. Ejemplo de la expansión de las condiciones de propagación: (a) esquema del subcircuito, (b) grafo del subcircuito, y (c) expansión de la condición de propagación de la ruta $path_{16}$ y contradicción sobre la ruta $path_7$. 86
3.7. Algoritmo para la generación del patrón de entrada en la verificación de una ruta. 88
3.8. Algoritmo para la generación de patrones de máxima cobertura. . . 90
3.9. Algoritmo de generación de patrones de máxima cobertura empleando Algoritmos Genéticos. 92
3.10. Cálculo del coste de un individuo: (a) individuo con 3 rutas válidas, (b) individuo con 4 rutas válidas y una de ellas duplicada, (c) individuo con tan sólo 2 rutas válidas. 92
3.11. Ratios del tiempo de CPU para el conjunto de circuitos MCNC'91. 95
3.12. Convergencia de la función de coste de los mejores individuos en la generación de patrones de verificación para el circuito 5xp1. . . . 96
3.13. Convergencia de la función de coste medio de los individuos en la generación de patrones de verificación para el circuito 5xp1. 96
3.14. Convergencia de la función de coste de los peores individuos en la generación de patrones de verificación para el circuito 5xp1. 97
3.15. Diversos modelos de descripción para la puerta NOR: a) completo, b) suponiendo y' y y'' variables enteras, c) con las entradas codificadas, y d) PROLOG. 106
3.16. Algoritmo para evaluar la actividad de conmutación producida por dos vectores de entrada consecutivos. 111
3.17. Ejemplo de actividad de conmutación. a) Circuito de lógica combinacional bajo estudio representado como grafo, b) estado interno estable de la red Booleana cuando el vector de entrada es el 0000, c) estado interno estable de la red Booleana cuando el vector de entrada es el 1111, d) detección de las transiciones cuando el vector de entrada cambia de 0000 a 1111, e) estado interno estable cuando el vector de entrada es el 0101, y f) detección de las transiciones cuando el vector de entrada cambia de 0000 a 0101. 112
3.18. Algoritmo para obtener el estado interno de un circuito; dado un vector de entrada, actualiza y evalúa la actividad de conmutación basado en el estado anterior de la red Booleana. 114
3.19. Algoritmo de cálculo del costo para evaluar la actividad de conmutación en un circuito de lógica combinacional basado en el último vector aceptado. 114
3.20. Algoritmo final de coste para evaluar la actividad de conmutación en un circuito de lógica combinacional. 115
3.21. Convergencia de la función de coste del circuito C6288 para la búsqueda de la secuencia de estímulos de entrada con un conjunto de 5 búsquedas simultáneas. 118

4.1. Curva de prestaciones área–retardo. La recta C representa el punto de partida de la exploración, mínima área, máximo retardo del circuito. La recta D representa el punto de mínimo retardo del circuito y, por tanto, el de máxima área. 126

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

4.2. Conjunto ruta crítica como subred del circuito: a) conjunto ruta crítica (zona sombreada), b) ruta dependiente del conjunto ruta crítica (área punteada), y c) rutas dependientes del conjunto ruta crítica. 130
4.3. Algoritmo de actualización del conjunto ruta crítica. 135
4.4. Algoritmo para la generación de la curva completa de prestaciones. 136
4.5. Curva de retardo de una NAND bajo diferentes cargas frente al factor de velocidad. 139
4.6. Algoritmo de generación de la curva completa de prestaciones empleando la técnica de Programación Lineal. 141
4.7. Algoritmo de dimensionado de puerta empleando técnicas de Programación Lineal. 142
4.8. Número de variables frente al número de puertas. 145
4.9. Variaciones en el número de variables frente al número de puertas. . 145
4.10. Tiempo de CPU frente al número de puertas. 146
4.11. Curva de prestaciones completa de área activa frente a retardo para el circuito bw. 147
4.12. Retardo del circuito bw bajo diferentes cargas frente a la suma de factores de velocidad. El símbolo Ξ representa la carga de un inversor con 15 fF de capacidad de cableado parásita. 147
4.13. Algoritmo para la generación de la curva completa de prestaciones. 148
4.14. Exploración del espacio de diseño completo en mapeado tecnológico atendiendo a la ruta crítica del circuito: a) metodología tradicional de mapeado tecnológico, b) exploración del espacio de diseño atendiendo a la ruta crítica. 150

5.1. Puerta CMOS con las cuatro corrientes empleadas para definir su corriente de fuga. 161
5.2. Puerta sometida a una alta carga de *fanout*. 164
5.3. Diagrama de bloques del modelo de retardo empleado en el cálculo de la densidad de transiciones aplicando el filtrado. 167
5.4. Ejemplo de cono lógico de entrada y cono lógico de salida de una puerta lógica. 170
5.5. Algoritmo de actualización del conjunto ruta crítica. 175
5.6. Algoritmo para la generación de la curva completa de prestaciones. 175
5.7. Curva de prestaciones retardo–potencia de consumo. La recta C representa el punto de partida de la exploración, mínima potencia de consumo, máximo retardo del circuito. La recta D representa el punto de mínimo retardo del circuito y, por tanto, de máxima potencia de consumo. 177
5.8. Representación y codificación de un circuito de lógica combinacional mediante técnicas basadas en algoritmos genéticos: (a) circuito a optimizar, (b) representación mediante un grafo, y (c) codificación del individuo de la población del algoritmo genético. 178
5.9. Ejemplo de individuo propuesto por el algoritmo genético. 179
5.10. Convergencia de la función de coste en la optimización del circuito i8 para la obtención del punto de mínima potencia. 183
5.11. Algoritmo de generación de la curva completa de prestaciones empleando la técnica de Programación Lineal. 191
5.12. Algoritmo de dimensionado de puerta empleando técnicas de Programación Lineal. 192

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

5.13. Número de variables frente a número de puertas. 198
5.14. Variaciones en el número de variables frente al número de puertas. . 198
5.15. Tiempo de CPU frente a número de puertas. 199
5.16. Curva completa de prestaciones (potencia de consumo, retardo de propagación y área activa) empleada el circuito C6288 optimizando la red Booleana $\vec{G}$ y el conjunto ruta crítica $path_{cp,i}$. 201
5.17. Curva completa de prestaciones (consumo de potencia, retardo de propagación y área activa) empleada el circuito apex6 optimizando la red Booleana $\vec{G}$ y el conjunto ruta crítica $path_{cp,i}$. 201
5.18. Curva completa de la prestación consumo de potencia frente a retardo para el circuito apex6. 202

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

Índice de tablas

2.1. Descripción de los comandos permitidos en la interfaz serie . . 53
2.2. Descripción de los comandos permitidos en la interfaz paralela 55
2.3. Tipos de datos y operandos soportados por el nivel 2 60
2.4. Descripción de las operaciones permitidas sobre listas 62
2.5. Comparativa de la metodología basada en desarrollo a medida frente a la metodología propuesta en este trabajo 64
2.6. Principales características de la metodología propuesta frente a las técnicas a medida 65

3.1. Rutas y condiciones para la cobertura basada en rutas del circuito del ejemplo 77
3.2. Comparativa de los experimentos realizados entre la metodología basada en Algoritmos Genéticos y el Algoritmo de Búsqueda Exhaustiva de Rutas para la verificación empleando la métrica de cobertura de rutas 94
3.3. Resultados experimentales y comparativas de la generación de vectores funcionales para la verificación mediante MILP y cobertura de rutas de datos 102
3.4. Resultados experimentales y comparativas de la generación de vectores funcionales para la verificación mediante MILP y cobertura de rutas de datos (continuación I) 103
3.5. Resultados experimentales y comparativas de la generación de vectores funcionales para la verificación mediante MILP y cobertura de rutas de datos (continuación II) 104
3.6. Comparativas entre diversas formas de codificar el problema con MILP y PROLOG 105
3.7. Comparativa de los resultados experimentales empleando Algoritmos Genéticos y una técnica probabilística para obtener el consumo de potencia máximo, medido como actividad de conmutación 117

4.1. Resultados, complejidad, número de variables, tiempos de ejecución y ganancias de la metodología propuesta. Ejemplos de lógica combinacional multinivel 152
4.2. Resultados, complejidad, número de variables, tiempos de ejecución y ganancias de la metodología propuesta. Ejemplos de lógica combinacional multinivel, continuación 153
4.3. Resultados, complejidad, número de variables, tiempos de ejecución y ganancias de la metodología propuesta. Ejemplos de lógica combinacional de dos niveles 154

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

5.1. Resultados experimentales empleando Algoritmos Genéticos para la obtención del punto de mínima potencia 182
5.2. Resultados experimentales empleando Programación Lineal para la obtención del punto de mínima potencia 186
5.3. Comparativa en términos de tiempo de CPU para la obtención del punto de mínima potencia mediante Algoritmos Genéticos frente a Programación Lineal 187
5.4. Comparativa, en términos de área, retardo y potencia optimizada, para obtener el punto de mínima potencia de consumo de los circuitos de dos niveles y multinivel con mas de 1.000 puertas del MCNC'91 189
5.5. Comparativa del parámetro de influencia del tiempo de transición de las señales en la potencia de corto circuito para diversas tecnologías actuales y futuras en términos de tamaño mínimo de puerta, tensión de alimentación, tensión umbral y fabricante 195
5.6. Comparaciones de resultados experimentales empleando Programación Lineal para obtener la curva de mínima potencia de consumo de los circuitos de dos niveles y multinivel con mas de 1.000 puertas del MCNC'91 197
5.7. Comparaciones de potencia y área activa necesaria, en términos de error porcentual cometido, de las curvas de prestaciones potencia de consumo–retardo y área–retardo de la metodología de exploración del espacio de diseño potencia de consumo–retardo, empleando exclusivamente la ruta crítica o la red Booleana completa de los circuitos de dos niveles y multinivel con más de 1000 puertas del banco de pruebas MCNC'91 . . . 200
5.8. Comparaciones, en términos de error porcentual cometido, de las curvas de prestaciones potencia de consumo–retardo de la metodología de exploración del espacio de diseño área–retardo basado en la ruta crítica y la exploración del espacio de diseño potencia de consumo–retardo empleando la red Booleana completa sobre los circuitos de dos niveles y multinivel con más de 1000 puertas del banco de pruebas MCNC'91 203

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

Capítulo 1

Verificación de Sistemas Integrados

Índice General

1.1. Introducción **3**

1.2. Diseño de Sistemas de Conmutación de Próxima Generación **5**

1.2.1. El Ciclo de Diseño 6

1.2.2. Unicidad entre Lenguajes 13

1.3. Verificación Basada en Simulación **14**

1.3.1. Trabajos Previos 14

Los Métodos Formales o Semi–Formales 15

La Verificación Basada en Simulación 16

1.3.2. El Entorno para la Verificación 19

La Verificación de Sistemas: Unicidad 20

1.3.3. Verificación Automática Respecto al Golden Reference 23

1.3.4. La Verificación Basada en Reglas 25

1.3.5. Análisis de Prestaciones con Exploración del Espacio de Diseño 27

1.4. Entorno de Verificación Propuesto **28**

1.5. Organización de la Memoria **28**

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

Glosario de Términos

ATM	Asynchronous Transfer Mode.
ATPG	Advanced Test Pattern Generation.
BDD	Binary Decision Diagram.
CAD	Computer Aided Design.
CI	Circuito Integrado.
CSIX	Common Switch Interface.
DBV	Dispositivo Bajo Verificación.
DSL	Digital Subscriber Line.
DUT	Device Under Test.
FPGA	Field Programmable Gate Array.
Gb/s	Gigabits por segundo.
HDL	Hardware Description Language.
ITRS	International Technology Roadmap for Semiconductors.
OBDD	Ordered Binary Decision Diagram.
PLI	Programming Language Interface.
OC–48	Optical Carrier–48, Interfaz de comunicaciones CSIX a 2,5 Gb/s.
OC–192	Optical Carrier–192, Interfaz de comunicaciones CSIX a 10 Gb/s.
OC–768	Optical Carrier–768, Interfaz de comunicaciones CSIX a 40 Gb/s.
QoS	Quality of Service.
RTL	Register Transfer Level/Language.
RTPG	Random Test Pattern Generation.
TR	Transceiver o Transceptor.
UML	Unified Modeling Language.
VCD	Value Change Dump.
VHDL	VHSIC HDL, Very High Speed Integrated Circuit HDL.
VLSI	Very Large Scale Integration.
VOQ	Virtual Output Queue.

Este primer capítulo es de carácter introductorio. Se plantea la necesidad e importancia de la verificación de sistemas integrados. Se particulariza con un ejemplo de verificación sobre un sistema avanzado de conmutación de paquetes. El objetivo de este capítulo es esbozar las características y metodologías implementadas en el entorno de verificación desarrollado en esta Tesis Doctoral.

Tras una introducción a la verificación de sistemas integrados, este capítulo, comienza con una descripción somera de los sistemas de conmutación de paquetes. A continuación, se presenta el Ciclo de Diseño de los sistemas integrados con énfasis en las tareas de verificación y análisis de prestaciones. Se continúa con el estado del arte de la verificación de sistemas integrados, la verificación basada en simulación, la generación de vectores de verificación y las métricas de cobertura. También se aborda la importancia del análisis de prestaciones con exploración del espacio de diseño, como pieza clave en la verificación de sistemas integrados. Por último se introducen las metodologías propuestas de verificación con análisis de prestaciones y exploración del espacio de diseño, así como el entorno desarrollado en esta Tesis Doctoral.

El capítulo comienza en el apartado 1.1 introduciendo la importancia de la verificación en el diseño de sistemas integrados. A continuación, en el apartado 1.2 se presenta la arquitectura típica de un Sistema de Conmutación de Próxima Generación y su Ciclo de Diseño, poniendo especial énfasis en la verificación. En el apartado 1.3.1 se muestran los diversos trabajos previos en el área de la verificación funcional. En los apartados 1.3.2, 1.3.3 y 1.3.4 se presenta el entorno propuesto en esta Tesis Doctoral para la verificación y los dos modelos de verificación propuestos — automático o basado en casos críticos. El apartado 1.3.5 justifica la necesidad del análisis de prestaciones con exploración del espacio de diseño del sistema integrado. Las principales características del entorno de verificación propuesto e implementado en esta Tesis Doctoral se presentan en el apartado 1.4. Por último, el apartado 1.5 comenta la organización de esta memoria.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

1.1. Introducción

Un parámetro fundamental en el diseño de cualquier sistema es el *time–to–market*[1]. Este parámetro es de vital importancia en un entorno cada vez más competitivo, donde muchos fabricantes se disputan el mercado, siendo la tecnología empleada similar para todos ellos. En el caso de los sistemas integrados, el *time–to–market* se encuentra totalmente determinado por el tiempo de verificación. Actualmente, el tiempo y el esfuerzo empleados en el proceso de verificación de un sistema, supera en más de tres veces el tiempo de desarrollo del mismo [1]–[3].

Time–To–Market

En el diseño de sistemas integrados, el desarrollo temporal de un proyecto se planifica minuciosamente para cumplir con una fecha de lanzamiento del producto al mercado. Con esta finalidad, las prestaciones del mismo son calibradas para que su verificación no se extienda más allá del plazo de tiempo previsto. De esta forma, las prestaciones del producto final no se encuentran acotadas por la tecnología de fabricación, sino por la capacidad del equipo de verificación, tanto desde el punto de vista de disponibilidad de recursos humanos como tecnológicos. Esta disponibilidad de recursos tiene un fuerte impacto en el costo del diseño. De hecho, es imposible reducir el coste por debajo de un cierto umbral (véase la figura 1.1).

Bottleneck

La figura 1.2 refleja la evolución de las prestaciones de los sistemas integrados a lo largo del tiempo frente al crecimiento potencial previsto de acuerdo con la tecnología de fabricación. Se puede apreciar como existe un claro desajuste entre el potencial ofertado por la tecnología y el nivel de complejidad que se alcanza actualmente.

Como muestra la figura 1.2, el aumento de complejidad debido a mejoras en los procesos de fabricación — medido en términos de número de transistores que se pueden integrar en un mismo circuito integrado (CI) — es de un $58\,\%$ anual; mientras que la tasa de crecimiento real de los CIs fabricados es de, tan solo, un $21\,\%$ anual. A partir de 1997, aparecen incrementos constantes respecto a ésta tasa de crecimiento, debido al uso de las herramientas CAD. Actualmente y en el futuro, se precisan de herramientas de verificación y validación del diseño si se quiere mantener las tasas de complejidad.

Proceso de Fabricación

Los análisis en detalle del sector[2] indican que son necesarias nuevas metodologías y herramientas para automatizar el proceso de verificación, como paso fundamental para reducir la diferencia entre potencial tecnológico y productividad.

Para aumentar las prestaciones de los sistemas integrados de forma significativa es necesario una mejora notable en las metodologías de verificación. En el caso de los sistemas de conmutación, esta mejora permitirá incrementar la complejidad de los procesadores de red, lo que conllevará, indudablemente, un mayor

Metodología de Verificación

[1] Se define el *time–to–market* como el tiempo que transcurre entre que se define un producto hasta que el producto está, finalmente, en el mercado.

[2] Consúltese el "Technology Roadmap for Semiconductors" en sus ediciones desde el año 2001 hasta la actualidad. [http://public.itrs.net/].

[1] Sedar Tasiran and Kurt Keutzer. Coverage Metrics for Functional Validation of Hardware Designs. *IEEE Design and Test of Computers*, 18(4):36–45, August 2001.

[2] EDA today. Electronic Design Automation User Study, 1999.

[3] Edmund M. Clarke and Jeannette M. Wing. Formal Methods: State of the Art and Future Directions. *ACM Computing Surveys*, 28(4):626–643, 1996.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

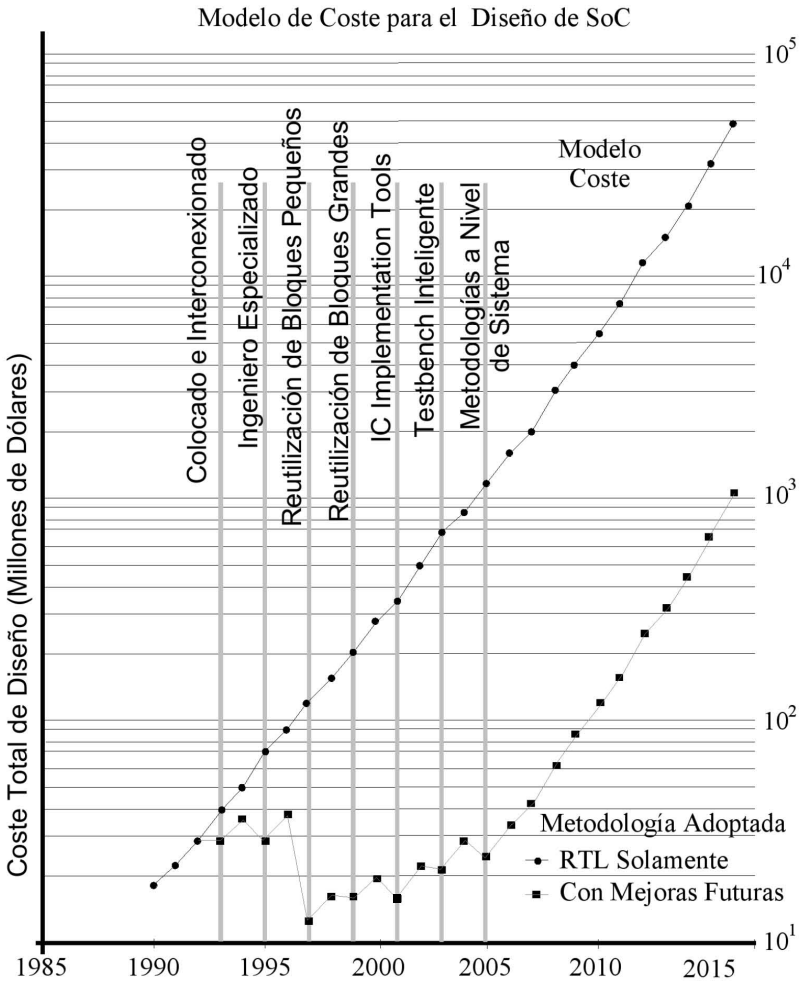


Figura 1.1: Impacto de la metodología de diseño en el costo de implementación de sistemas.

Fuente: "Technology Roadmap for Semiconductors", 2001.

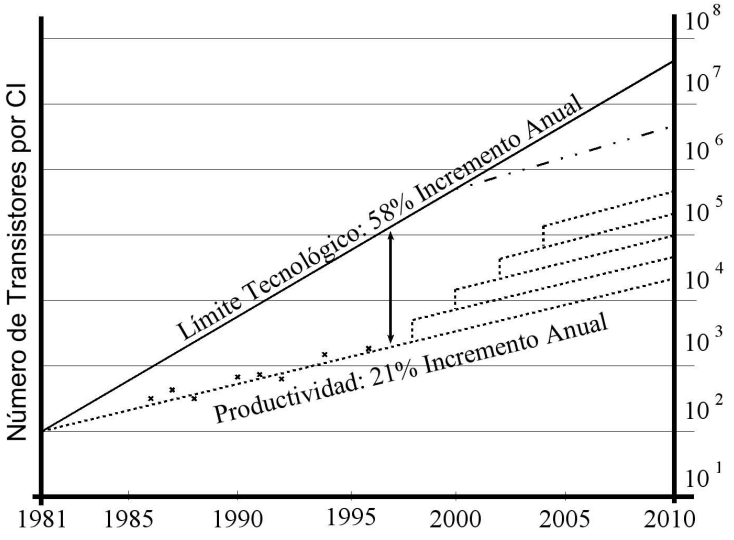


Figura 1.2: Densidad de integración frente a productividad.

Fuente: "Technology Roadmap for Semiconductors", 2001.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

aprovechamiento del ancho de banda en las futuras redes de comunicación, y una mejora en la calidad de servicio, en adelante, QoS[3].

El objetivo fundamental que se pretende cubrir en esta Tesis Doctoral es la de aportar soluciones que permitan una reducción efectiva del tiempo y del esfuerzo en la etapa de verificación. Para lograr este objetivo, es necesario proveer de un entorno que permita generar estímulos, comprobar respuestas, y analizar prestaciones explorando el espacio de diseño del sistema integrado a verificar. Tanto la generación de estímulos como la comprobación de respuestas ha de permitir el empleo de modelos de referencia y métricas de cobertura, los cuales guiarán el proceso de verificación. El análisis de prestaciones con exploración del espacio de diseño permitirá ajustar los retardos anotados de las unidades funcionales implementadas en el sistema integrado a verificar, para así acometer una verificación más realista. La anotación se corresponderá con un rango de valores válidos — solución de la exploración del espacio de diseño — y no con un único valor puntual como se hace en la actualidad. Esto incrementará el grado de libertad a la hora de determinar el rango de implentaciones que superan la verificación o no — en la actualidad sólo se plantea una única solución circuital que supera o no la verificación.

En resumen, el entorno propuesto y las metodologías implementadas en él ha de ser capaz de analizar las prestaciones y verificar sistemas integrados complejos de una manera más eficiente a la actualmente empleada, cubriendo desde la etapa más temprana del diseño hasta la de validación del primer prototipo.

1.2. Diseño de Sistemas de Conmutación de Próxima Generación

Desarrollo de Internet

El desarrollo de Internet, cada vez, demanda más ancho de banda y más funcionalidades a los equipos de comunicación. Aunque, la clave de estos sistemas es la estructura de transmisión (paralela o serie) para transferir paquetes de datos desde un puerto de entrada a uno o más puertos de salida, en la práctica resulta imprescindible disponer de arquitecturas de altas prestaciones para la planificación y la conmutación.

Las redes de comunicaciones no son meras arbitradoras de ancho de banda entre tráfico de datos y de voz. De hecho, una partición estricta del ancho de banda conlleva un uso ineficiente del ancho de banda total. A medida que el ancho de banda del sistema crece, también se incrementa el costo relativo por el desaprovechamiento de un porcentaje del mismo, incluso, aunque sea posible una reducción del costo absoluto medido en Gb/s. Se necesitan funciones sofisticadas de QoS, que aseguren el uso eficiente del ancho de banda, y por tanto, reduzcan el costo total del sistema.

Calidad de Servicio

El hecho de aprovechar, eficientemente, el ancho de banda del canal, y distinguir tráfico de distintas prioridades para distintas clases de aplicaciones y/o usuarios; determina en última instancia y en gran medida, la QoS del sistema, a costa de un incremento notable en la complejidad de los sistemas de conmutación. Con esta finalidad, los sistemas de conmutación han de disponer de mecanismos para:

- La gestión de tráfico con distintos niveles de prioridad.

[3] Acrónimo del término anglosajón *Quality of Service*. Calidad de servicio de la red.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

- Decidir la configuración óptima de la matriz de conmutación.
- Permitir una escalabilidad del sistema (aumentar o reducir el número de puertos de entrada/salida) sin que se degraden las prestaciones.
- Controlar el flujo de datos.

Arquitectura Conmutador VOQ

En la figura 1.3, se muestra un ejemplo de la arquitectura de estos conmutadores de altas prestaciones. Básicamente, existen dos tipos de circuitos: el transceptor bidireccional y la matriz de conmutación.

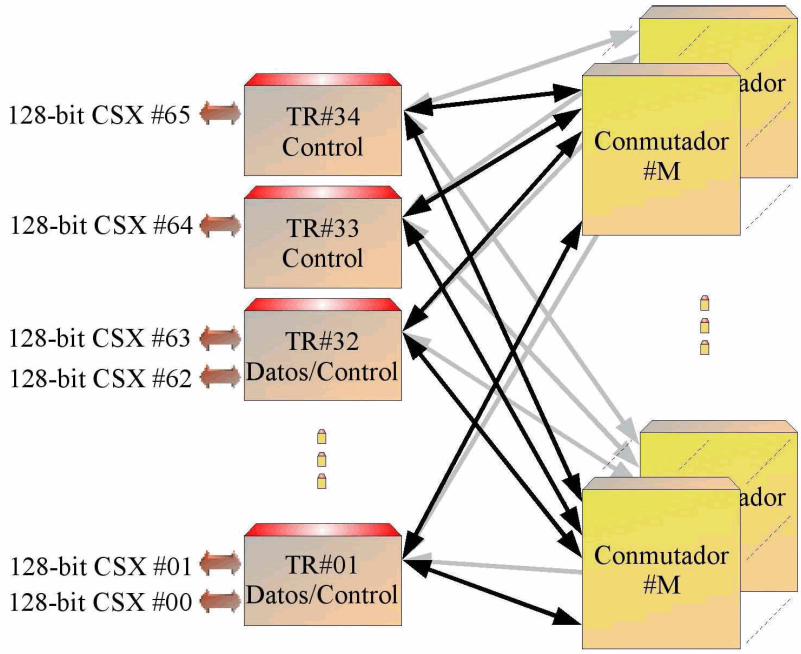


Figura 1.3: Arquitectura de un conmutador.

El transceptor, convierte las tramas de un formato estándar como por ejemplo CSIX[4], a un formato propio — interno del fabricante — para la matriz de conmutación. El número de transceptores depende del número de puertos del sistema. Además, existe al menos un par de transceptores dedicados a la información de control de flujo. El tipo de entrada es paralelo y requiere de 32 ó 128 bits, en función de la tasa de entrada, que puede ser OC–48 ($2'5\ Gb/s$) ó OC–192 ($10\ Gb/s$).

La conexión con el conmutador está compuesta por un conjunto de 2 a 4 enlaces serie de alta velocidad, en función de la tasa de entrada, por lo que cada circuito conmutador requiere 2 o 4 CIs conmutadores. El número de circuitos conmutadores depende del número de puertos del sistema final, y puede ser 1, 2 ó 3; ya que los transceptores no suelen aceptar más de 3 entradas serie por puerto CSIX.

1.2.1. El Ciclo de Diseño

Tradicionalmente, para poder desarrollar un sistema integrado — como es el caso de un sistema de conmutación para *GigabitEthernet* — deben alcanzarse los

[4]CSIX es un estándar de hecho, desarrollado por el Common Switch Interface Consortium. Este consorcio incluye a más de 25 fabricantes de equipos de conmutación en redes ATM, *FastEthernet* y *GigabitEthernet*.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

siguientes objetivos, tal y como se ilustra en la figura 1.4. Estas etapas se describen a continuación.

Definición de la Arquitectura.– En esta etapa se ha de obtener un modelo algorítmico o descripción de la arquitectura del sistema completo[5] — *Golden Reference*. Este modelo se escribe en un lenguaje de alto nivel, generalmente en C/C++. La elección de un lenguaje se debe a varias razones, de entre ellas se destacan las siguientes:

Modelo de Alto Nivel

- Obtener las prestaciones del sistema conmutador.– La forma habitual de reflejar las prestaciones de los sistemas de conmutación, es por medio de curvas de carga. Estas curvas reflejan el comportamiento del sistema completo o de un subsistema concreto, en función de distintos valores de la carga. Este comportamiento, se suele expresar mediante el *throughput*[6], además de utilizar otro tipo de medidas como la latencia, el porcentaje de paquetes rechazados, el tiempo medio de espera en cola, el tiempo máximo de servicio, el número medio de tramas en cola, entre otros.
- Servir como patrón o referencia del comportamiento del sistema conmutador.– Define como deberá comportarse, finalmente, el diseño implementado, tanto funcionalmente, como en prestaciones. Este patrón de referencia, es el *Golden Reference*.
- Ajustar las variables *trade–off* y analizar el comportamiento de las prestaciones finales del conmutador de cara a la aplicación final.– Las variables *trade–off* son aquellos parámetros del diseño con dependencias entre los distintos parámetros y las prestaciones finales. Algunas de estas variables son el tamaño de las colas, el número de colas por destino y por prioridad, la política de gestión de prioridades, y el algoritmo de planificación, entre otras. Existen herramientas específicas para la obtención y el análisis de las prestaciones finales del sistema en función de éste tipo de decisiones en el ámbito de arquitectura [4].

Variables Trade–off

- Estudiar las prestaciones latencia, *throughput*, entre otras, del sistema conmutador, respecto a escalabilidad en cuanto a número de puertos y número de matrices conmutadoras.– Las prestaciones del sistema se obtienen como el resultado de una serie estadística de simulaciones del comportamiento de la arquitectura frente a paquetes entrantes. Aquellos parámetros relacionados con la llegada de paquetes al conmutador, como el tiempo entre llegadas, la prioridad, el puerto de destino, el tipo de trama, se modelan mediante variables pseudo–aleatorias[7]. Estas fuentes pseudo–aleatorias generan eventos en las entradas del sistema. La evolución de determinados parámetros estadísticos se representa gráficamente conformando las curvas de prestaciones del sistema.
- Evaluar las prestaciones del sistema en comparación con otros productos existentes en el mercado.– En general, con cada mejora de un producto o especificidad del mismo se suministran datos relativos a las prestaciones que se esperan obtener de forma comparativa. Aunque, un fabricante no suele disponer de modelos completos de productos de la competencia, sí dispone de modelos completos de productos propios.

Particionado.– Particionar la arquitectura en módulos independientes, y definir la funcionalidad de todos y cada uno de los módulos integrantes del sistema.

[5] Modelo de muy alto nivel del sistema, que recoge los aspectos fundamentales de la especificación del sistema. En verificación automática, se contrasta la funcionalidad del sistema con la del *Golden Reference*.

[6] Tasa de salida del sistema completo. Expresándose normalmente en Gb/s.

[7] Números aleatorios generados por ordenador.

[4] Israel Cidon, Amit Gupta, Tony Hsiao, Asad Khamisy, Abhay Parekh, Raphael Rom, and Moshe Sidi. OPENET: An Open and Efficient Control Platform for ATM Networks. *IEEE INFOCOM*, pages 824–831, April 1998.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

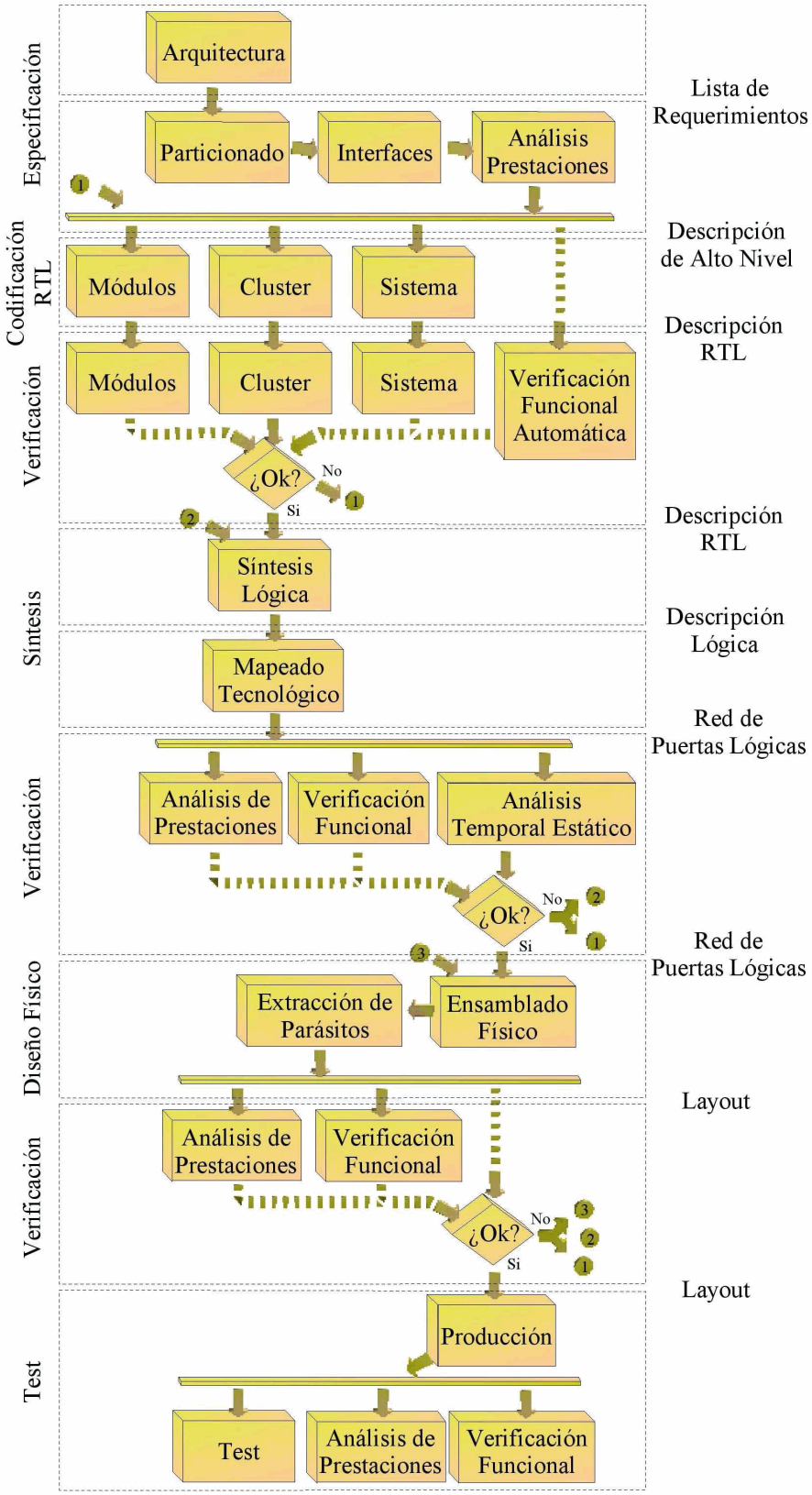


Figura 1.4: El ciclo de diseño.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

Las particiones se realizan en al menos dos niveles denominados módulos y sus agrupaciones o *clusters*[8].

Especificación de las Interfaces.– Se identifican y analizan las necesidades de comunicación entre las distintas funcionalidades de los diversos módulos definidos en el particionado de la arquitectura. Una vez particionado el diseño, se procede a la especificación de las interfaces entre los módulos. El no disponer de un modelo algorítmico a éste nivel, implica la existencia de múltiples imprecisiones en dichas especificaciones. Los errores en este nivel son muy comunes y una vez detectados, provocan rehacer las especificaciones y volver a codificar los módulos.

Análisis de Prestaciones.– Una vez se ha finalizado el modelo de alto nivel del sistema a implementar, se somete a diversas condiciones de trabajo con el fin de obtener múltiples resultados en función de los diversos parámetros de configuración del sistema. Es decir, se trata de conocer las prestaciones del sistema, a través del modelo de alto nivel, en varias situaciones de funcionamiento. De estas simulaciones a muy alto nivel, se obtienen diversas gráficas, que en el caso de un sistema de conmutación son: el número de paquetes medio en cada cola, ordenadas por prioridad, frente a la carga del sistema o con una dirección de destino determinada. En los sistemas de conmutación esta información es muy útil a la hora de seleccionar la profundidad de memoria en función de la carga máxima o media a soportar. Una vez definidos los parámetros de configuración de la arquitectura, las simulaciones obtenidas con dichos parámetros definirán el comportamiento del mismo, con lo cual dichas simulaciones pueden ser tomadas como referencia del normal comportamiento del diseño a implementar.

Simulación de Alto Nivel

Esta etapa, también, permite conocer el impacto de las etapas previas en las prestaciones especificadas del sistema. El particionado y la incorporación de interfaces, pueden delimitar las prestaciones del sistema.

Codificación RTL de los Módulos, *Clusters*[9] y Sistema.– Es el proceso de descripción hardware (comportamiento o estructural) de cada uno de los módulos, *clusters* y sistema definidos en el particionado. La codificación hardware consiste en crear un programa escrito en un subconjunto del lenguaje HDL elegido. Este subconjunto recibe el nombre de HDL sintetizable. Por ejemplo, para el lenguaje *Verilog* este subconjunto se denomina en la literatura *Verilog*–RTL o *Verilog* sintetizable. Por tanto, el objetivo de esta etapa no es sólo desarrollar cada uno de los módulos, *clusters* y sistema que componen el diseño, sino que dicho desarrollo se describa de forma tal que sea posible su implementación física.

Verificación de los Módulos.– Se trata de comprobar que la descripción hardware del módulo tenga la misma funcionalidad que la definida en la etapa de particionado. Un gran parte de la verificación, se efectúa en base a un conjunto de simulaciones. Se suele utilizar el mismo lenguaje HDL para realizar esta etapa de la verificación. El lenguaje puede calificarse de más o menos apto para esta tarea en función de la complejidad del módulo y de

Verificación de Funcionalidades Básicas

[8]Esta nomenclatura suele ser confusa. A veces se intercambian los términos módulos por *clusters* y viceversa. Lo fundamental es que exista una partición jerárquica en al menos dos niveles del sistema completo.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

la rigurosidad de la verificación. Los fallos detectados a nivel de módulo implican volver a codificar el módulo de nuevo para que éste se ajuste a la especificación correctamente, y por supuesto, volver a verificarlo.

Verificación de los Clusters, y del Sistema.– Una vez que todos los módulos integrantes del *cluster* o del sistema han sido comprobados funcionalmente, se enlazan y se comprueba la funcionalidad de los *clusters* que forman. De igual forma que con los módulos, se agrupan los *clusters* para componer el circuito, y así verificar su funcionalidad. Esta verificación detecta errores cometidos en las etapas de particionado y/o en la especificación de interfaces. Por tanto, los fallos en este nivel implican redefinir la interfaz de los módulos afectados, volver a codificarlos y a verificarlos. Una vez corregidos, es necesario volver a verificar los *clusters* y el CI, hasta comprobar que la funcionalidad de todo el sistema coincide con la del *Golden Reference*.

Verificación de Interfaces y Sistema

Verificación de las Prestaciones del Sistema.– Del análisis de prestaciones del sistema se desprenden un conjunto de características que ha de cumplir el código implementado. La verificación de las prestaciones del sistema consiste en comprobar que dichas características extraídas del análisis de prestaciones, se cumplen. Si el sistema a comprobar no fuese determinista o la función que mide las prestaciones es muy compleja, se necesitaría verificar las prestaciones del sistema frente a las obtenidas con el *Golden Reference*. Este paso se hace impracticable mediante simulaciones con lenguajes HDL convencionales, pues implicaría simular una estadística suficientemente amplia sobre una descripción a nivel hardware o RTL del sistema completo.

Verificación Automática.– La verificación automática del sistema es la comparación, de una forma automatizada, entre la descripción hardware final a nivel RTL del sistema completo y el *Golden Reference*. Un entorno que permita la verificación automática del sistema evitaría tener que generar un *testbench*[10] y una multitud de simulaciones, con el consiguiente ahorro de tiempo. Estas simulaciones son necesarias para asegurar que el sistema se comporta exactamente como indica el modelo.

Golden Reference

Para llevar a cabo la verificación automática, existen diversas soluciones dentro de las técnicas de verificación formal. Las técnicas de verificación formal persiguen una verificación certera. En sistemas con un alto nivel de complejidad, como los sistemas de conmutación, cubrir todos los casos posibles es impracticable. Se buscan soluciones como el *Model–Checking*[11] que aseguren el sistema para una determinada cobertura de fallos, a pesar de existir un grado de incertidumbre en el proceso. El *Model–Checking* permite verificar el sistema de dos formas diferentes: mediante un modelo del sistema o frente a un conjunto de reglas que deban cumplirse.

Síntesis Lógica.– Las herramientas de diseño actuales permiten tiempos de codificación muy reducidos para diseños de un nivel de complejidad medio. Sin embargo, para diseños más complejos no hay muchas herramientas que sinteticen los diseños desde el nivel algorítmico. La programación orientada a

[10] Es el entorno necesario para efectuar la simulación y la verificación del sistema.

[11] Consiste en verificar la validez de una implementación comparando su comportamiento frente a un modelo de referencia.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

objeto es un complemento muy adecuado para estos lenguajes de especificación.

La síntesis es el proceso por el cual se transforma la descripción RTL en una descripción circuital basada en puertas e interconexiones entre estas[12]. La síntesis consiste en aplicar dos procesos bien diferenciados. El primero de ellos es la Síntesis Lógica y el segundo el Mapeado Tecnológico. La Síntesis Lógica procesa la descripción RTL y la convierte a una descripción puramente lógica. Este proceso posee un extenso estado del arte, con muchas aportaciones teóricas y avances en la industria.

Todas las fases que se desarrollan dentro de la Síntesis Lógica están relacionadas con la transformación, manipulación y optimización de lógica combinacional. Además de convertir el código RTL en una descripción lógica, otro objetivo primordial en esta etapa es reducir las funciones lógicas finales que describen el circuito sintetizado.

Mapeado Tecnológico.– Este segundo proceso que se aplica en la síntesis de un circuito requiere una librería tecnológica de puertas. Esta librería permitirá convertir la descripción lógica de cada módulo del sistema, en una descripción de puertas de dicha librería e interconexiones entre ellas. El objetivo final del Mapeado Tecnológico es obtener un circuito que cumpla un conjunto de restricciones de retardo, área y/o potencia. Este conjunto de restricciones definen el llamado punto de trabajo del espacio de diseño de la implementación del circuito sintetizado. La literatura existente en este otro área de la síntesis de circuitos es también muy amplia. La definición de un punto del espacio de diseño es muy importante para la verificación, pues permite realizar las anotaciones[13] correspondientes de retardo a los diversos módulos implementados. Esto permite comprobar la funcionalidad implementada con un temporización muy aproximada[14] a la real del circuito.

Diseño Físico.– Una vez se posee una red de puertas asociadas a una librería tecnológica, se procede a realizar la planificación, colocado e interconexionado físico de las mismas, dentro del área especificado para ello. El desarrollo de este proceso aplica diversas políticas de colocado e interconexionado en función de factores como la disposición de los buses de alimentación dentro o fuera del *layout*[15] de la puerta, la disposición y acceso a los terminales[16] de estas, entre otras. Por lo general, el área final del circuito está limitada por el área máxima del encapsulado seleccionado para el producto final y por tanto cada módulo y *cluster* posee a su vez un área asignada, la cual no ha de superar. El resultado de este laborioso paso es un trazado físico en el que ya no sólo tienen interpretación física los polígonos que definen a las

Layout

[12]En la literatura anglosajona se denomina a la vista circuital, compuesta por descripciones físicas de puertas e interconexiones entre estas, como *net–list*.

[13]Este proceso se conoce tradicionalmente en la literatura anglosajona como *back–annotation*.

[14]En esta fase del flujo de diseño se dispone del valor correcto de los retardos de puerta, pero puesto que no se ha realizado la interconexión física entre las puertas, módulos o incluso los *clusters*, se ha de hacer uso de estimadores de cableado para obtener un valor del retardo aproximado al real.

[15]La vista *layout* de una puerta define su trazado físico. El trazado físico especifica los polígonos de cada capa que son necesarios para construir la puerta.

[16]En la literatura se denomina *pin*.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

puertas y sus componentes, sino que cada interconexión posee su representación en base a polígonos, o lo que es o mismo, se dispone de un *layout* del dispositivo.

Parámetros Parásitos

Del *layout* se pueden deducir las características parásitas (capacidades, inductancias, resistencias, etc.) que va a poseer la implementación física del mismo. La extracción de dichos parámetros permite aproximar aún más los valores de retardo de las simulaciones realizadas al real.

El carácter tardío de la obtención de esta vista *layout* es un duro inconveniente. Un error descubierto en esta etapa final del flujo de diseño puede suponer un retraso temporal elevado sobre el *time–to–market*, puesto que la modificación necesaria puede afectar a las etapas tempranas del flujo de diseño.

Igualmente, la naturaleza tardía de la obtención de los parámetros parásitos fuerza a realizar las simulaciones con estimadores tanto de retardo de puerta como de interconexionado desde las etapas tempranas del diseño. Una vez alcanzada esta etapa y finalizada la mayor parte de la verificación del circuito, sólo queda tiempo para ejecutar unas pocas simulaciones de casos críticos con el objeto de comprobar la validez de las suposiciones realizadas.

Realizar el Test de un Primer Prototipo.– Se trata de un paso previo a la fabricación de la serie, donde se realiza la verificación sobre un dispositivo hardware (test) utilizando un emulador (FPGA, equipo de emulación, etc). La verificación a este nivel, ha de ser una verificación a nivel hardware del prototipo. Sin embargo, las posibilidades del equipo de test limitan el alcance y profundidad de esta verificación. Un inconveniente adicional, es que el código escrito para la verificación del sistema no se suele aprovechar para esta tarea.

El tiempo y esfuerzo que se invierte en el proceso de verificación, es normalmente muy superior al invertido en el diseño del mismo, tal y como ha quedado reflejado en el flujo de diseño descrito; lo cual impone una fuerte limitación en el *time–to–market*, quedando éste prefijado, según la complejidad del sistema, por el tiempo invertido en las simulaciones.

Cabe destacar que para dar comienzo a las simulaciones de los *clusters*, o del CI, todos los módulos deben haber sido descritos a nivel RTL y su funcionalidad debe haber sido verificada. Esta dependencia supone, en la práctica, un cuello de botella muy importante, pues implica no cometer errores en las etapas de particionado y definición de las interfaces, para poder concluir, simultáneamente, el diseño de todos los bloques.

Punto de Trabajo

Una carencia importante, hoy día, es la definición del punto de trabajo del circuito. Las herramientas de síntesis física actuales no exploran el espacio de diseño del circuito a implementar, se limitan a generar una versión del CI en un punto del espacio de prestaciones, que incluso puede no cumplir las restricciones de área, retardo y/o potencia de consumo exigidas en las especificaciones. Ello implica reejecutar la herramienta de síntesis una y otra vez hasta alcanzar el objetivo marcado. En última instancia, en caso de no alcanzar dichos requisitos, es necesaria la recodificación parcial de alguno de los módulos desarrollados. Lo que significa que se ha de ejecutar nuevamente todas las pruebas de verificación en todos los niveles en los que se involucra a dicho módulo.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

Sería deseable que dichas herramientas de síntesis física provean más que un único punto del espacio de diseño, un rango de funcionamiento. Ello permitiría elegir el punto de diseño para no sólo cumplir con las restricciones de área, retardo y potencia de consumo, sino que permitiría en muchos casos también culminar la verificación si errores debidos a la existencia de un retardo incorrecto en alguna funcionalidad.

La exploración del espacio de diseño de los circuitos dotaría a la herramienta de verificación de un gran margen de decisión, puesto que los retardos de cada módulo se sustituirían por un rango de posibles retardos. Dicho margen permitiría a la herramienta de verificación determinar cuál es el subconjunto de retardos que cumplen la especificación dada, y por tanto, el rango de puntos del espacio de diseño que cumplen o no con la especificación verificada. Hoy en día, las prestaciones temporales del CI quedan fijadas por el punto del espacio de diseño que determina la herramienta de síntesis física, la fase de verificación sólo puede determinar si dicho punto es válido o no. Es deseable disponer de las prestaciones antes de la etapa de síntesis física, incluso antes de la síntesis lógica. Es en este punto donde quedan encuadradas las aportaciones de esta tesis relativas al análisis de prestaciones.

1.2.2. Unicidad entre Lenguajes

Los lenguajes HDL utilizados para el desarrollo de sistemas integrados han demostrado ser ineficientes e inadecuados para abordar la codificación y verificación de los complejos diseños actuales.

Entre los múltiples inconvenientes, se destaca que dichos lenguajes no soportan descripciones en el ámbito de las especificaciones que sirvan de patrón de referencia. Esto obliga a tener que emplear dos o más lenguajes distintos en el ciclo de diseño. Por ejemplo, se utiliza un lenguaje de alto nivel para la extracción de las prestaciones del modelo, y un lenguaje HDL para el ciclo de diseño. Realizar estas dos especificaciones en lenguajes distintos es un grave inconveniente. En la mejor de las situaciones, se ha de realizar una verificación cruzada entre ambas descripciones.

Unicidad en la Descripción

Actualmente, las especificaciones, tanto del sistema hardware como del *testbench*, se realizan con lenguajes HDL que son de muy bajo nivel que no ayudan al diseñador en todas las fases de diseño. De aquí que, el esfuerzo y el tiempo invertido en la verificación de esta clase de sistemas es excesivo.

Existen herramientas comerciales como *Specman*[17], *FoCs*[18], *Active HDL* o *Riviera*[19], *FormalPro*[20], entre otras, que ofrecen un entorno integrado para la generación y verificación del *testbench*. Sin embargo, obligan al diseñador a aprender un nuevo lenguaje que es exclusivo para el uso de dicha herramienta.

Por otro lado, y a pesar de que existen lenguajes de muy alto nivel como C, C++, o UML, entre otros, para abordar eficientemente el diseño de sistemas altamente complejos, el diseñador hardware sigue empleando lenguajes HDL para esta labor, ya que muchos de los anteriores lenguajes no soportan características propias de sistemas digitales, como son, la especificación del conexionado basada en buses, la simulación a cuatro estados por señal, la simulación en el dominio

[17] *Specman* es un producto de Verisity Design Inc, ahora Cadence Design Systems Inc.
[18] *FoCs* es un entorno de verificación formal desarrollado por IBM.
[19] *Active HDL* o *Riviera* son herramientas de verificación de la compañía ALDEC Inc.
[20] *FormalPro* entorno de Mentor Graphics que permite realizar *Equivalence Checking*.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

del tiempo, la especificación de retardos, y por último, no son compatibles con la herramienta de síntesis lógica.

1.3. Verificación Basada en Simulación

Potencial Tecnológico

A pesar del espectacular desarrollo tecnológico en fabricación de circuitos, no hay un incremento significativo en las prestaciones de los sistemas integrados — en comparación con el potencial que cabe esperar de dichas tecnologías. Este desaprovechamiento de recursos radica en que el proceso de fabricación ha dejado de ser un factor restrictivo en el diseño de sistemas integrados, para quedar relegado a un segundo plano frente al paradigma de la complejidad. La causa fundamental de esta situación es la falta de una metodología de verificación de sistemas, que ofrezca soluciones eficientes ante el aumento de complejidad de los sistemas integrados. En la presente línea de investigación, se propone la creación de un conjunto de herramientas para la verificación de sistemas de comunicación, basados en una aportación metodológica propia.

1.3.1. Trabajos Previos

En estos últimos años se ha producido un cambio determinante en el panorama de las tecnologías de la información y de las comunicaciones. A medida que Internet se ha ido convirtiendo en un medio de masas para la comunicación, la educación, el trabajo y el entretenimiento; la cantidad de tráfico transmitida a través de las redes ha pasado de aumentar una media de un 100 % cada año, a multiplicarse por un factor de cuatro, a partir de 1997.

Este hecho, junto con el desarrollo a gran escala de las tecnologías de acceso de alta velocidad como DSL[21], han llevado consigo una creciente demanda de ancho de banda. Esta demanda insaciable de ancho de banda conlleva un aumento exponencial en la complejidad de los sistemas de conmutación de las actuales redes de comunicación.

El incremento de la complejidad de los circuitos integrados en general, y de los sistemas de conmutación en particular, es patente hoy día si observamos nuestro entorno — ordenadores, multimedia, telefonía, hogar, routers, entre otros. Dicho crecimiento en complejidad es impulsado tanto por las nuevas tecnologías como la mejora e introducción de técnicas innovadoras de diseño.

A la par que los sistemas y subsistemas digitales se vuelven más y más complejos, la dificultad para comprobar su funcionamiento se ve aumentada en igual o superior medida. Por otro lado, la verificación de sistemas siguiendo metodologías formales o semi–formales tienen un fuerte impacto en el costo final del sistema. De hecho y de acuerdo con el *Technology Roadmap for Semiconductor*, desde su edición de 2001 hasta la actualidad, se recoge textualmente: "... Technological progress depends on developing rigorous and efficient methods for analyzing a design (the bulk of formal and semi–formal verification research today)..." donde se expresa que una de las claves para progresar en el diseño se encuentra en los métodos formales y semi–formales de verificación. Finalmente, un punto de inflexión importante en la verificación de sistemas es la reutilización del esfuerzo

[21] Acrónimo de *Digital Subscriber Line*.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

realizado durante la fase de diseño para verificar el sistema fabricado. Respecto a ese último punto no hay soluciones industriales[22].

Actualmente, el esfuerzo necesario para comprobar las funcionalidades básicas de cualquier circuito comercial supone más del 60 % del tiempo total del desarrollo del prototipo. La imperante necesidad de asegurar la confiabilidad del producto ha dirigido las miradas, tanto de los diseñadores como de la industria, hacia la investigación sobre las técnicas de verificación y la creación de herramientas que faciliten la misma.

Esfuerzo en la Verificación

Las técnicas existentes, hoy día, son el fruto de muchos trabajos independientes de la comunidad científica, que han ido conformando varias metodologías que permiten en ciertas ocasiones abordar total, o parcialmente, la verificación funcional de los circuitos de forma total y parcial.

Los Métodos Formales o Semi–Formales

Los métodos formales o semi–formales entienden la verificación como un problema puramente matemático. En este tipo de metodología se propone una aritmética, que con un conjunto de axiomas permiten determinar la validez del diseño [3], [5]. Estos métodos son:

Theorem Proving.– Este método fue de los primeros propuestos en verificación formal [6]. La especificación del sistema y su implementación se realiza a nivel lógico. Para ello, se define un conjunto finito de operaciones, y de operandos, mediante un alfabeto también finito. El sistema descrito a nivel lógico, se contrasta con la especificación de origen. La principal ventaja de esta metodología es el grado de abstracción que adquiere la verificación. La principal desventaja es la complejidad de la descripción lógica del sistema.

Alto Grado de Abstracción

Model–Checking.– La aproximación *Theorem Proving* se fundamenta en la descripción estructural para realizar la verificación [6]. El método de *Model Checking* se fundamenta en la existencia de un modelo que describe la funcionalidad — a nivel de comportamiento — a verificar, lo cual es completamente opuesto al método *Theorem Proving*. En general, el método se basa en utilizar un algoritmo que determine la validez de la descripción del sistema a nivel temporal con respecto al comportamiento esperado por una descripción de referencia del mismo.

Equivalence Checking.– Este método se fundamenta en la comparación de redes lógicas, la de referencia y la del sistema a verificar [7]. La exactitud de

[22] En el 20th Anniversary of the IEEE VLSI Test Simposium, Monterey, CA, 2002, se discutió ampliamente la necesidad de realizar hardware para verificación que reutilice toda la infraestructura desarrollada durante la verificación del diseño.

[3] Edmund M. Clarke and Jeannette M. Wing. Formal Methods: State of the Art and Future Directions. *ACM Computing Surveys*, 28(4):626–643, 1996.

[5] Carl-Johan Seger. *An Introduction to Formal Hardware Verification*. PhD thesis, Deparment of Computer Science, Univ. British Columbia, 1992.

[6] Elliot Mendelson. *Introduction to Mathematical Logic*. Chapman & Hall/CRC, 4th edition, June 1997.

[7] G.V. Bochmann. Hardware Specification with Temporal Logic: An Example. *IEEE Transactions on Computers*, C-31(3):223–231, March 1982.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

los algoritmos de extracción de las equivalencias entre redes es el principal escollo de esta metodología. En particular se precisa de un analizador de patrones a fin de obtener las equivalencias entre redes. Dentro de este tipo de verificación se encuadran todas las técnicas de análisis de diagramas de decisión Booleanas (BDD, OBDD, entre otros), que permiten reducir la complejidad de los analizadores de patrones por medio de la fijación de una estructura de datos común para la descomposición de las funcionalidades del sistema.

Symbolic Simulation.– Consiste en la simulación simbólica de las funciones a verificar [8], [9]. Las variables o conjunto de valores lógicos que se introducen en el simulador simbólico se propagan por la descripción del sistema haciéndolas evolucionar hasta que se generen variables de salida. Es evidente que este tipo de simulación no puede ser realizada con los simuladores tradicionales. La complejidad del simulador requerido se hace patente a medida que crece la funcionalidad del circuito a verificar.

Evaluación Abstracta del Comportamiento

La simulación simbólica se centra en descomponer, adecuadamente, la funcionalidad del sistema, con el fin de reducir su complejidad en la verificación. Para ello, se emplean árboles de decisión binaria (Binary Decision Tree), árboles ordenados BDD (OBDD), que junto a técnicas de evaluación de la trayectoria simbólica — propagar las "X", o estados desconocidos, en las funciones Booleanas — reduzcan de forma notoria la complejidad del simulador simbólico.

La Verificación Basada en Simulación

La industria también plantea soluciones pragmáticas. La premisa industrial es que no es posible verificar el diseño al 100 %, debido al reducido tiempo disponible para poner en el mercado el producto (*time–to–market*). La determinación y posterior verificación de la funcionalidad mínima aceptable del producto permite acotar este tiempo, eso sí, a costa de sacrificar la integridad total de la funcionalidad del circuito o sistema.

time–to–market

El lanzamiento del producto, no supone la finalización de la verificación. Por lo general, la primera versión del producto posee una funcionalidad básica verificada. En paralelo a la producción en masa de unidades para la comercialización, la verificación del producto se extiende a las funcionalidades no verificadas con anterioridad. En base a la respuesta del mercado ante el nuevo producto, se plantea la inclusión de funcionalidades extras en las sucesivas revisiones del producto o la parálisis de cualquier ampliación y/o verificación sobre el mismo. De alguna forma, los entornos de verificación basados en simulación son específicos del sistema hardware a verificar. Por tanto, sacrificando cierta generalidad en el proceso de verificación se gana en reducción del *time–to–market*.

La planificación de la verificación en base a distintos niveles de confiabilidad, hace posible acortar el tiempo, a pesar de no garantizar la integridad de toda la

[8] M. Browne, E. Clarke, D. Dill, and B. Mishra. Automatic Verification of Synchronous Circuits using Symbolic Logic Simulation an Temporal Logic. In *Proc. of IMEC IFIP International Workshop on Applied Formal Methos for Correct VLSI Design*, 1989.

[9] H. Navarro, J. A. Montiel-Nelson, J. Sosa, and José C. García. A Symbolic Solver to Obtain the Entire Solution Space of the Boolean Satisfiability Problem using Genetic Algorithms. In *Evolutionary and Deterministic Methods for Design, Optimization and Control with Applications to Industrial and Societal Problems*, September 2005.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

funcionalidad. La verificación de dicha funcionalidad se puede realizar mediante alguno de los métodos formales o semi–formales descritos anteriormente, pero por lo general los casos o esquinas a verificar (*corners cases*) son comprobados mediante verificación basada en simulación. Se trata de configurar y estimular el circuito para que se dé el estado a verificar, y comprobar su comportamiento en ese estado. Para ello, ha de escribirse una porción de código específico que configura, estimula y comprueba el circuito. Por razones económicas, dichos casos han de reducirse al conjunto mínimo que asegure la funcionalidad. Además de la planificación de esos casos, existen otros puntos clave en la verificación basada en simulación [10]–[12] que se describen a continuación:

Casos Críticos

RTPG/ATPG.– Una vez configurado el circuito a verificar, la generación de los vectores de estímulos y verificación es uno de los puntos clave de esta metodología [10]. Los RTPG/ATPG[23] realizan la función de estimular aleatoriamente a los circuitos bajo verificación a fin de cubrir el mayor número de situaciones con el menor número de estímulos. En principio, los generadores aleatorios son elementos sencillos que se limitan a seguir unas especificaciones de entrada predefinidas antes de comenzar la simulación. Los generadores avanzados, son capaces de modificar su comportamiento en base a los resultados anteriores en la verificación.

Inyección de Patrones

Métrica de Cobertura.– Definida una funcionalidad a verificar, es necesario simular un conjunto de puntos de trabajo dentro de esa funcionalidad [13]. El conjunto de puntos a simular suele ser inmenso — espacio de simulación — e independiente del empleo de cualquier generador. Por otro lado, la necesidad de acelerar la ejecución de la fase de verificación restringe el conjunto de puntos del espacio de verificación. Se entiende que existe un conjunto de puntos alrededor de los llamados casos críticos del sistema. La extracción de estos casos de verificación hace posible la reducción del conjunto total de puntos a simular.

Confiabilidad de la Verificación

Como se ha expuesto, la cobertura al 100 % en la verificación del diseño suele ser un reto imposible en un plazo aceptable, por lo que se precisa de una métrica para estimar la fiabilidad de la verificación. Por lo general, esta métrica se extrae del número de casos a verificar, de las variaciones y de la complejidad de los mismos.

Otro punto clave de la verificación es la elección del lenguaje en el que se va a implementar el entorno de verificación. La existencia de herramientas, la curva de aprendizaje de las mismas y la reutilizabilidad del código son, básicamente, los factores que intervienen en la decisión de emplear uno u otro lenguaje. Según [2], son tres los lenguajes empleados en la verificación de circuitos integrados.

[23] Acrónimos de los términos anglosajones *Random Test Pattern Generator* y *Advanced Test Pattern Generator*, respectivamente.

[10] Dinos Moundanos, Jacob A. Abraham, and Yatin-Vasant Hoskote. Abstraction Techniques for Validation Coverage Analysis and Test Generation. *IEEE Transactions on Computers*, 47(1):2–14, 1998.

[11] Gale Morrison. Cadence Open–Sources T–Builder. *Electronics News*, August 2000.

[12] Qualis Design Corporation. Empresa Dedicada a la Verificación de Sistemas (Adquirida Recientemente por Synopsys Inc.).

[13] John R. Wallack and Ramaswami Dandapani. Coverage Metrics for Functional Test. *Proc. of the 12th IEEE VLSI Test Symposium*, pages 176–181, 1994.

[2] EDA today. Electronic Design Automation User Study, 1999.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

Lenguaje de Descripción Hardware

HDL (*Verilog* y *VHDL*).– Desarrollar la verificación en el lenguaje el cual se ha descrito el diseño parece reducir el tiempo de la verificación, pues no hay necesidad de aprender un nuevo lenguaje. Además, la existencia de múltiples simuladores HDL hace atractiva la idea de emplear el propio lenguaje de diseño en la verificación. La reusabilidad del código generado se hace altamente dependiente del estilo de programación de la verificación. Sin embargo, al aumentar el nivel de complejidad del circuito, los protocolos entre las interfaces se vuelven más complejos y los datos tienden a ser estructuras complejas, difíciles de implementar en lenguajes HDL.

Lenguaje Propietario

lenguaje–e.– Se definió en sus comienzos como un lenguaje orientado a la especificación de sistemas a alto nivel [14]. Posteriormente, se ha empleado para la verificación de sistemas debido a su gran versatilidad y a las facilidades que presenta a la hora de acometer el desarrollo de los casos de verificación. Es un lenguaje propietario y de no libre distribución, únicamente soportado por Specman. Su principal ventaja es la facilidad en la creación de generadores de datos y la representación gráfica de resultados, además de disponer de un sistema integrado de desarrollo. Si se opta a implementar la verificación en este lenguaje, su curva de aprendizaje es el principal problema. Además, al ser un lenguaje propietario, han de pagarse las licencias del software para poder ejecutar la verificación, suponiendo un coste adicional.

C/C++.– Hay diversas iniciativas que propugnan el empleo de C/C++ para realizar la verificación de los diseños. Entre todas ellas cabe destacar T–Builder de Cadence [11], OpenVera de Synopsys [15]. En cualquier caso, el acceso a los simuladores se realiza con llamadas en C/C++ de alto nivel, como es el caso de PLI para Verilog. La principal ventaja radica en que durante la etapa de estudio de la arquitectura del diseño, se desarrollan modelos de alto nivel para obtener las prestaciones del sistema bajo ciertas condiciones. Dichos modelos son en su mayoría programas escritos en C/C++ que pueden ser reutilizados en la verificación para obtener el correcto funcionamiento del diseño — por ejemplo, un algoritmo de selección de células en un conmutador con colas a la entrada. Si tenemos en cuenta la reusabilidad del código, la encapsulación y distribución del mismo, las posibles desventajas de emplear C/C++ son mínimas.

Lenguaje de Alto Nivel

Hay que tener en cuenta también el esfuerzo realizado por la Open SystemC Initiative, la cual es impulsora del sistema de diseño y verificación SystemC [16]. Este entorno de diseño se centra en la descripción y verificación de sistemas desde muy alto nivel. Incorpora su propio núcleo de ejecución y un amplio conjunto de librerías que permiten desarrollar sistemas descritos en C++ en tiempo récord. El empleo de C++ como plataforma de diseño permite realizar desde alto nivel las tareas de modelado y diseño. Por contra, este entorno no está ideado para abordar la descripción de los sistemas, *clusters* y módulos a bajo nivel.

[14] Verisity Inc. (Recientemente Adquirida por Cadence Inc.). Empresa Desarrolladora del e–lenguaje y del Entorno de Verificación Specman.

[11] Gale Morrison. Cadence Open–Sources T–Builder. *Electronics News*, August 2000.

[15] Synopsys Inc. *Synopsys User Manual*, v1.0 edition.

[16] Open SystemC Comunity. SystemC: Entorno de Desarrollo y Verificación de Sistemas, 2000.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

Es evidente que ninguna de las técnicas de verificación expuestas abarca todos los tipos de diseño, y en función de su implementación unas se presentan más idóneas que otras. El empleo de unas u otras es, en la actualidad, una decisión a tomar en las etapas tempranas del desarrollo del producto donde influyen factores como el costo de las licencias, la experiencia del grupo de verificación, entre otros. Recientemente, IEEE ha puesto en marcha un grupo de trabajo para el desarrollo de las especificaciones de un lenguaje de verificación [17]. La industria emplea, puntualmente, las técnicas formales cuando la explosión de estados es simple y la sencillez de estos permite su empleo. Sin embargo, el empleo de la verificación basada en simulación es la más extendida por la industria, hasta el momento.

Existen infinidad de simuladores HDL[24] (Verilog, VHDL, entre otros), tanto comerciales como de dominio público. Sin embargo, son muy pocas las herramientas comerciales que permiten el desarrollo de entornos de verificación [11], [12], [14]. En general, casi todas carecen de técnicas de generación de patrones para verificación optimizada por la cobertura. Por ello, cada vez es más importante el estudio de la métrica de cobertura.

Simulación del Diseño

El futuro no se torna en contra de la verificación formal. La verificación formal ha sido, es y será el campo de las contribuciones teóricas en verificación. La aplicación de la verificación formal a la industria dependerá del esfuerzo computacional que se precise en los sistemas complejos. En sistemas de cierta simplicidad, como las máquinas de estados, la verificación formal es adecuada. En sistemas más complejos, la verificación basada en simulación es la única alternativa.

1.3.2. El Entorno para la Verificación

Cada vez se hacen más necesarias nuevas herramientas de verificación que acepten descripciones mixtas hardware/software[25] (en adelante, Hw/Sw) para manejar, eficientemente, la complejidad de esta clase de sistemas.

Las descripciones mixtas Hw/Sw permiten que coexistan descripciones de alto nivel, como el *Golden Reference*, con descripciones hardware, a nivel RTL. La simulación y, por tanto, la verificación de este tipo de sistemas encierra cierta complejidad ya que el hardware es, inherentemente, paralelo en su ejecución; mientras que el software se ejecuta de manera secuencial.

En la presente línea de investigación, se describe, detalladamente, una herramienta para la especificación y verificación de sistemas de conmutación de gran ancho de banda. La herramienta verifica automáticamente cualquier nivel de descripción del sistema (módulo, *cluster*, CI). Permite la especificación de un conjunto de reglas de verificación *Model–Checking*, además de realizar especificaciones basadas en simulación con *Golden Reference*. La herramienta se puede utilizar con la descripción HDL del sistema a verificar o con un prototipo hardware del mismo.

[24] Acrónimo de Hardware Description Languages. Los más empleados son $Verilog$ (CADENCE Design Systems Inc) y $VHDL$ (VHDL International).

[25] Estas herramientas deben ser capaces de simular hardware y al mismo tiempo proveer al usuario de un lenguaje de alto nivel sobre el que especificar el entorno de verificación (o *testbench*). De lo contrario, resulta inviable la verificación de sistemas complejos.

[17] IEEE 1647 Working Group. e Functional Verification Language Working Group (eWG), 2005.

[12] Qualis Design Corporation. Empresa Dedicada a la Verificación de Sistemas (Adquirida Recientemente por Synopsys Inc.).

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

Claves del Entorno de Verificación

Alcanzar este objetivo global implica, básicamente, la consecución de tres objetivos parciales. La verificación de sistemas integrados basada en reglas, la verificación automática de sistemas integrados empleando modelos de referencia de alto nivel (*Golden Reference*), y el análisis de prestaciones con exploración del espacio de diseño. Estos aspectos fundamentales se analizan con más detalle en los apartados que siguen a continuación. Previamente, a éste análisis, las siguientes ventajas derivadas de unificar todas las descripciones en un único lenguaje C/C++, merecen ser destacadas.

La Verificación de Sistemas: Unicidad

Una primera e interesante consecuencia, es que al permitir la verificación Hw/Sw de código a cualquier nivel, podremos iniciar en cualquier momento las verificaciones en los ámbitos de módulo, *cluster* o ; sin necesidad de esperar a que todos los módulos estén codificados y simulados para dar inicio a las simulaciones de los *cluster*; ni tener que esperar a que todos los *clusters* estén verificados para comenzar con la simulación del sistema.

La figura 1.5 muestra la arquitectura de un sistema de conmutación que se encuentra en fase de desarrollo; en donde se distinguen sistemas que ya han sido implementados en RTL, con sistemas que aún están por codificar — de los que se dispone un modelo C/C++ de referencia a alto nivel.

Especificación mixta

La naturaleza de esta especificación es mixta, C/C++ y RTL. Gracias a una capa de software C/C++ que actúa de interfaz, se mezcla el código RTL de sistemas implementados con el código C/C++ de alto nivel de sistemas aún sin desarrollar. Esta especificación mixta conforma una descripción compacta del sistema en C/C++, que puede ser usada para la verificación a nivel de *cluster* o de sistema.

Esta importante ventaja permitirá reducir el *time–to–market* drásticamente, ya que se permite tener asegurada la funcionalidad del sistema en todo momento, tanto a nivel de módulo, como a nivel de *cluster* desde el comienzo del proyecto. Para ello, al codificar un nuevo módulo bastará verificar el mismo frente al modelo C++ de alto nivel. Una vez verificado, se ha de reemplazar el modelo de alto nivel C++ por el nuevo código RTL y continuar con el flujo de diseño. De esta forma, si hubiese fallos en la verificación a nivel de *cluster* o sistema, el fallo deberá estar acotado en el nuevo módulo o en su interfaz con el exterior.

La figura 1.6 muestra que son necesarias tres capas de software, denominadas *Capa de Comunicación*, *Capa de Verificación* y *Capa de Referencia*, para permitir acceder al código RTL desde C++. Esta interfaz C++ permite acceder al código RTL del DUT[26] o DBV[27] directamente en C++, integrando éste con el resto del sistema de verificación C++. El código RTL se ejecuta sobre el simulador HDL externo, y se comunica a través de un interfaz estándar PLI.

Organización del Entorno por Capas

La Capa de Comunicación es la más próxima al DBV. Se trata de una interfaz PLI que controla la comunicación entre C/C++ y el proceso de simulación de código RTL. La Capa de Verificación contiene funciones para comandar la interfaz PLI. Básicamente, se encarga de convertir los estímulos al DBV en estructuras de datos.

[26] Acrónimo del término anglosajón Device Under Test.

[27] Acrónimo del término español Dispositivo Bajo Verificación. No confundir el término DUT con el de DBV. Aunque en la literatura anglosajona se alude siempre al término DUT, éste ha de ser sustituido por DBV cuando se refiere, en español, al proceso de verificación.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

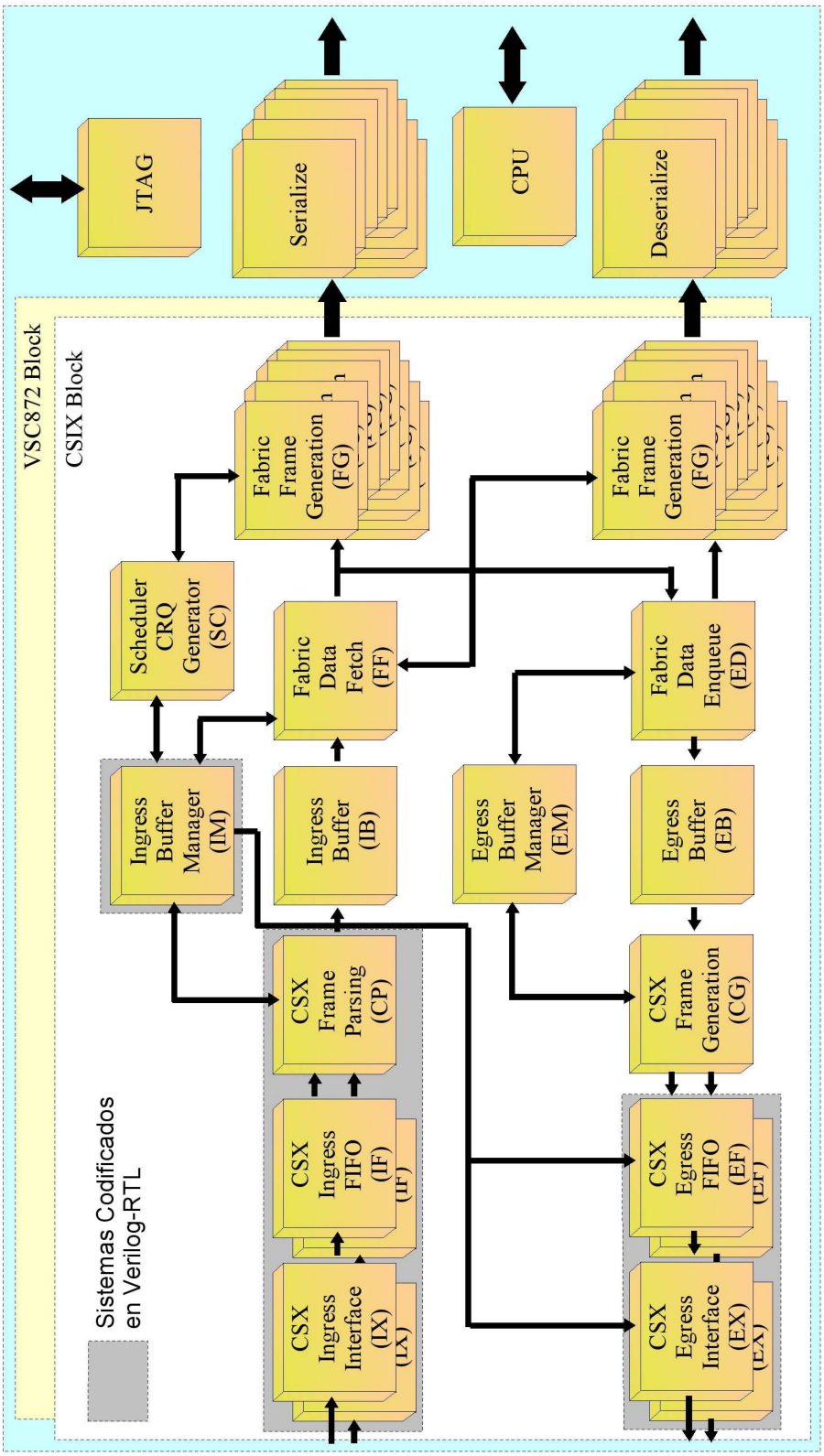


Figura 1.5: Ejemplo de un sistema de conmutación en fase de desarrollo.

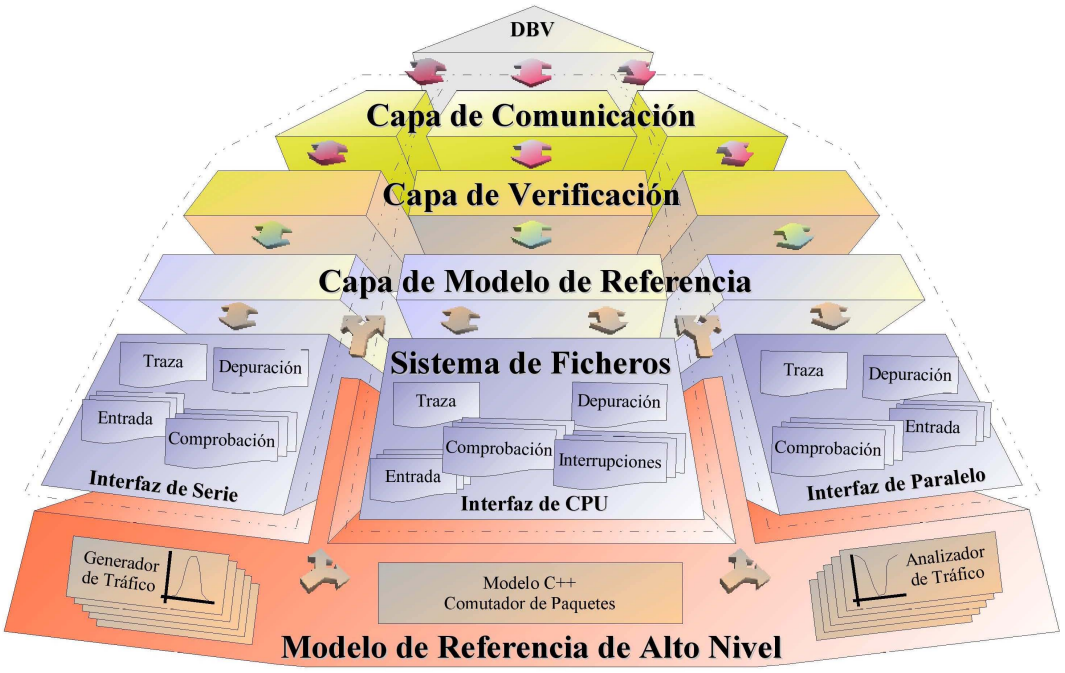


Figura 1.6: Las capas de software para la interfaz entre C++ y el código RTL. En la parte superior el Dispositivo Bajo Verificación (DBV) y en la inferior el modelo de referencia (*Golden Reference*).

La capa que interactúa con C/C++ directamente se denomina *Golden Reference Layer*. Esta capa es la encargada de sincronizar los eventos de simulación que se producen en el entorno C/C++ con los eventos temporales del simulador RTL. En general, es la que comanda el avance de la simulación y genera eventos ante cambios en la interfaz de comunicación del DBV.

Refinamiento del Código

Desde el *Golden Reference* hasta los Módulos, sin Errores.– Otra importante ventaja que se deriva de la posibilidad de especificar formalmente los módulos y sus interfaces mediante un código C++ de alto nivel, es que pueden ser verificados en todo momento frente al *Golden Reference*. De aquí que se podrán eliminar los fallos de especificación que son detectados posteriormente en las simulaciones a nivel de *cluster* y a nivel de sistema.

Refinamiento Tradicional

Codificación a Nivel RTL desde el Modelo C++.– Para la codificación hardware, la posibilidad de generar un conjunto de especificaciones independientes del sistema, a distinto nivel Hw/Sw, permite una gran disminución en el tiempo de diseño del sistema. Esto se debe a que el salto entre la descripción algorítmica inicial de una arquitectura y la descripción hardware final, puede ser abordado en numerosos pasos intermedios, pudiéndose comprobar la validez de cada paso. Este proceso se denomina refinamiento sucesivo.

Con las herramientas tradicionales, el diseñador se ve forzado a acometer este paso de una sola vez, pues los simuladores hardware tradicionales no admiten especificaciones mixtas, con lo que toda la dificultad se afronta en un único paso. Esto genera un círculo vicioso en el que el diseñador genera una especificación y ha de comprobarla una y otra vez hasta que obtiene los resultados esperados.

La figura 1.7 pone de manifiesto esta notable ventaja. A partir de un modelo inicial algorítmico, y un código de verificación se puede generar un

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

simulador ejecutable destinado a asegurar tanto la descripción funcional como las prestaciones del sistema completo. Una vez comprobada su correctitud, el modelo inicial es desgranado. Siguiendo una metodología *top–down*, se mezcla con el código común de verificación, y se vuelve a generar el simulador y comprobar la correctitud. De esta forma, y en sucesivos pasos, el modelo algorítmico queda transformado en una descripción hardware C++/RTL. En ese momento, la descripción del módulo, se encuentra a nivel C++/RTL; y puede ser fácilmente convertida a código Verilog–RTL de forma automática y libre de errores.

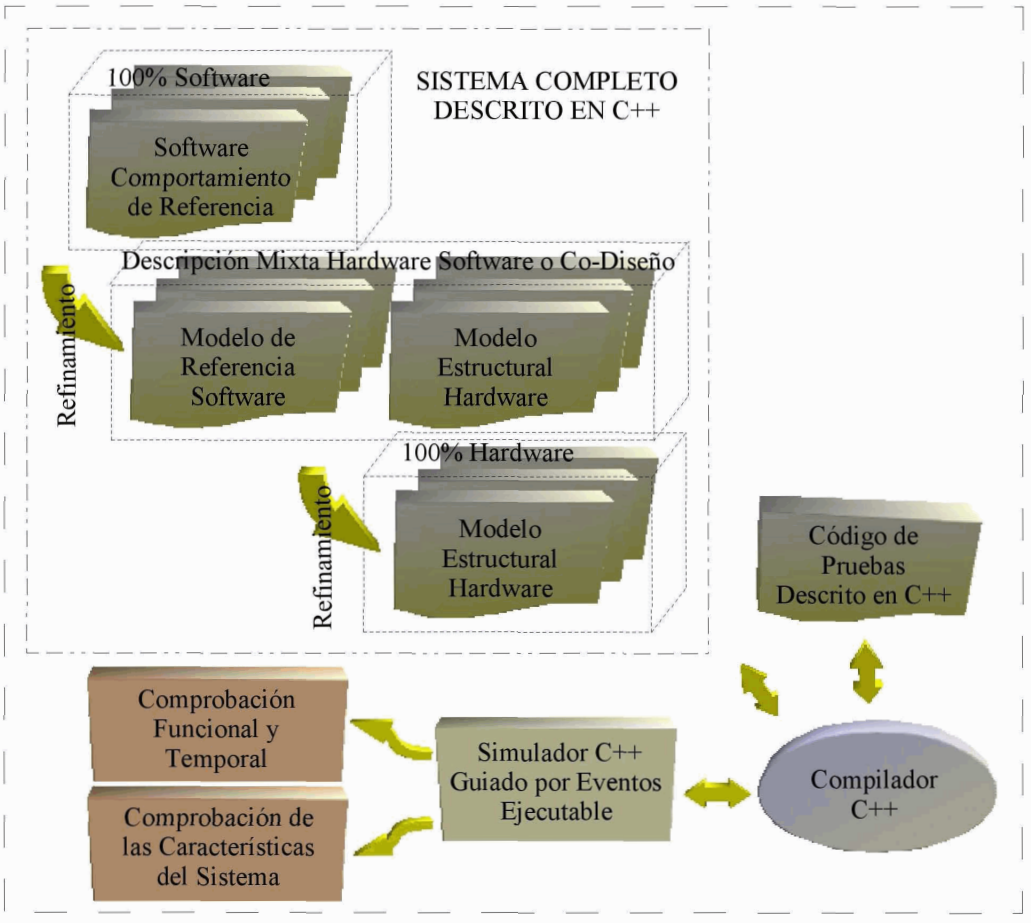


Figura 1.7: Refinamiento del modelo de comportamiento hasta nivel RTL en C++.

Una vez se han expuesto las ventajas derivadas del uso de descripciones mixtas C++ para el entorno de verificación, y la forma de aprovecharlas convenientemente; se procede a analizar las distintas formas de verificación que ofrece el entorno propuesto.

1.3.3. Verificación Automática Respecto al Golden Reference

El problema que se plantea es comparar una descripción del sistema a nivel mixto Hw/Sw del DBV frente a otra especificación de referencia. Esta clase de problemas, es del tipo *Model–Checking*. La figura 1.8 muestra la configuración del entorno para la verificación respecto al *Golden Reference*.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

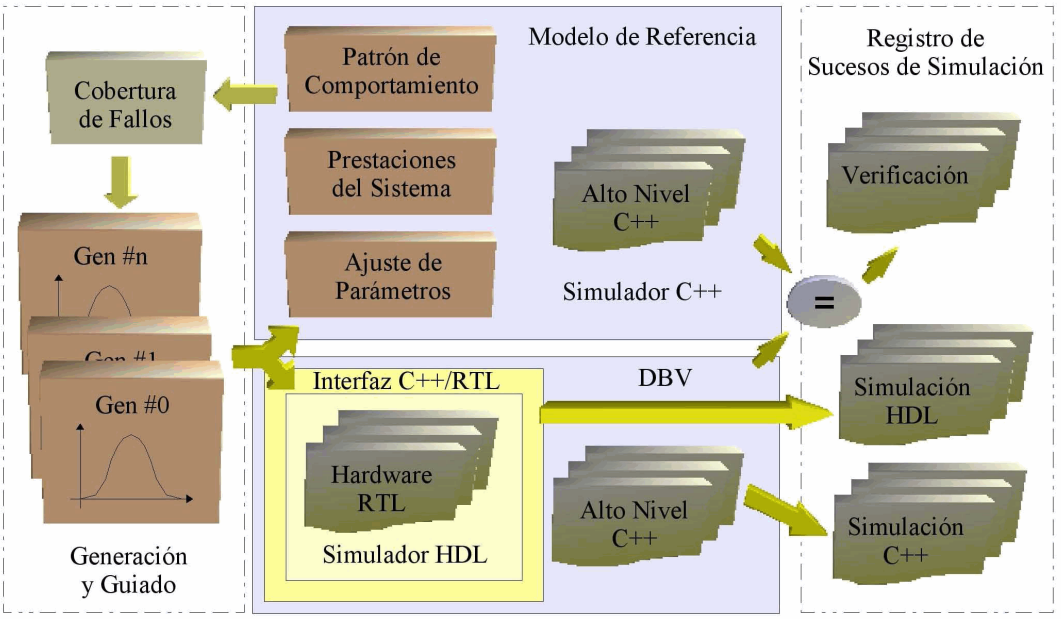


Figura 1.8: Verificación respecto al *Golden Reference*.

Metodología de Verificación Automática

Tal y como se muestra en la figura 1.8, tanto el DBV como el *Golden Reference* reciben las entradas de un generador de estímulos. La salida generada por ambas descripciones es comparada continuamente, con lo que se asegura que ambos sistemas presentan la misma respuesta ante los mismos estímulos. El generador de estímulos crea patrones pseudo–aleatorios[28], en primera instancia.

En segundo lugar, los esquemas pseudo–aleatorios se sustituyen por un generador basado en sistemas ATPG[29]. En este tipo de sistemas, se busca maximizar la cobertura de fallos de la etapa de verificación mediante un generador de vectores de test que es controlado por un medidor de la cobertura de fallos del sistema. El sistema para el guiado del generador de estímulos es complejo y requiere de técnicas evolutivas[30].

Generalmente, el generador de estímulos está basado en algoritmos genéticos. La función que se ha de optimizar, es una medida de la cobertura de fallos del sistema. Para ello, es necesario definir una métrica de dicha cobertura en función del modelo de referencia (*Golden Reference*).

Es decir, la métrica de cobertura dirige la generación de vectores de verificación. Esta configuración determina un sistema realimentado negativamente que tiende a maximizar la cobertura de fallos del proceso de verificación, de una manera automática. El mismo necesita de un ajuste y dimensionamiento del sistema completo para que sea convergente, junto con la definición de la métrica que se determine.

La figura 1.9 muestra una representación gráfica del registro de verificación. La generación del fichero de formas de onda se define en el DBV, donde se ha de

[28] Un generador pseudo–aleatorio es un algoritmo iterativo que proporciona una sucesión de valores cuyo comportamiento aparentemente es aleatorio, pero conocido el algoritmo de generación y sus parámetros iniciales, dicha secuencia puede ser reproducida.

[29] Acrónimo del término inglés Advanced Test Pattern Generation (Generador de Patrones de Test Avanzado).

[30] Las técnicas evolutivas son algoritmos que simulan la evolución de las especies y la aplican a la optimización multivariable.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

especificar una traza total — de todas las señales — o parcial — mediante especificación jerárquica — de las señales presentes en el sistema y su evolución en el tiempo. Dicha traza, está conforme al formato VCD[31]; ampliamente empleado por los simuladores digitales. De esta forma, cualquier herramienta externa de representación gráfica de formas de onda digitales puede ser empleada para su visualización y/o análisis.

Registro de Verificación

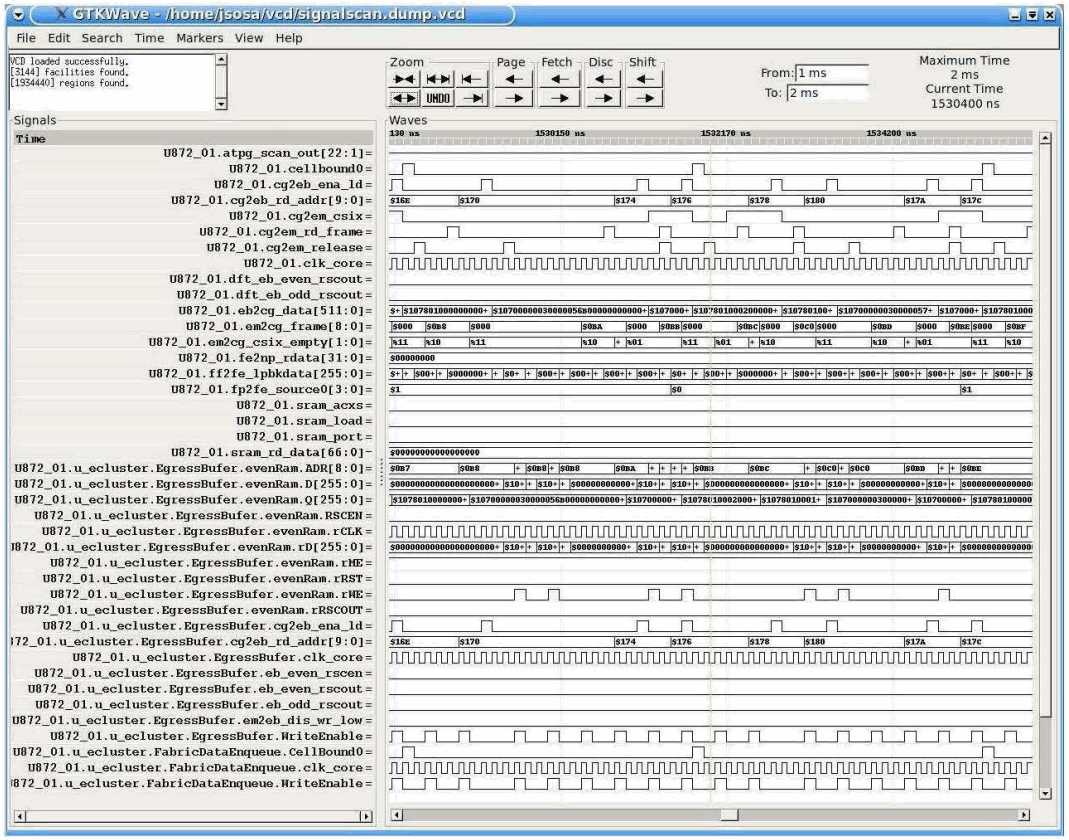


Figura 1.9: Representación gráfica del registro de simulación C++ o RTL.

El registro de verificación consta de un conjunto de ficheros de texto, visibles con cualquier editor de texto estándar, en los que se anotan todas las incidencias ocurridas durante la verificación, externas a los simuladores hardware. El registro de verificación se divide en el registro de comparación, el registro de error, y el registro de mensajes. El registro de comparación contiene todos los vectores de entrada, junto con la salida generada tanto por el DBV como por el modelo de referencia. El registro de error contiene aquellos vectores de entrada que han generado distinta respuesta en el DBV y en el modelo de referencia, junto con la respuesta de ambos. El fichero de mensajes contiene aquellos mensajes y/o alertas generadas por el entorno de verificación.

1.3.4. La Verificación Basada en Reglas

La figura 1.10 muestra el escenario para este otro tipo de verificación. Aquí, la especificación basada en reglas comanda la entrada al DBV y genera aquellas salidas que deban ser comparadas. En este caso, no existe una comparación continua

Metodología Basada en Casos Específicos

[31] Acrónimo del término inglés Value Change Dump.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

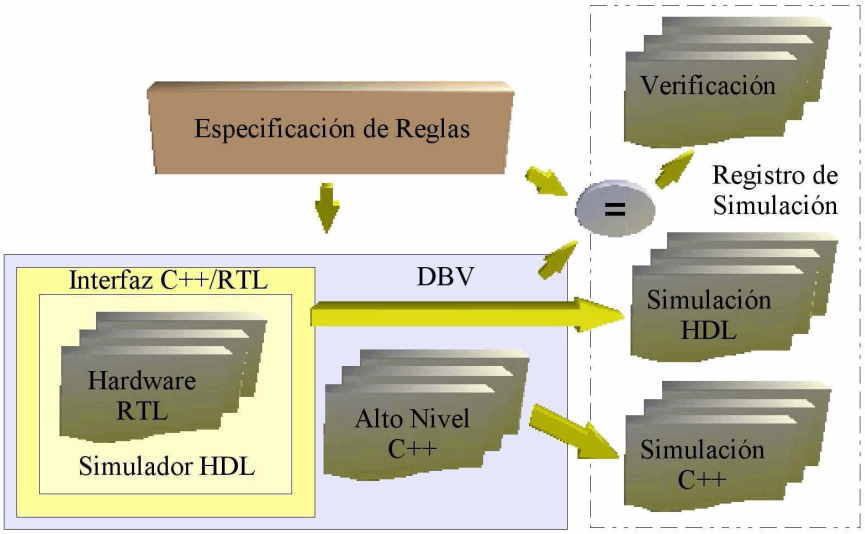


Figura 1.10: Verificación con especificación de reglas.

entre la salida del DBV y la especificación; sino que la comparación es puntual dependiente de las reglas definidas.

Al igual que en el caso anterior, los simuladores hardware C/C++ o el simulador HDL externo para código RTL, pueden generar el registro de simulación bajo demanda del usuario. El registro de verificación es generado automáticamente, y su contenido es similar al caso anterior.

Este tipo de verificación, está orientado a sistemas pequeños, como por ejemplo, la verificación de protocolos o de interfaces entre módulos, *clusters* o sistemas. Para verificaciones más complejas, es conveniente contrastar automáticamente respecto al *Golden Reference*, tal y como se describe en el apartado anterior.

La finalidad de la especificación de reglas, es obtener una lista de elementos a verificar — *assertion rules* — partiendo de la especificación más simple para el usuario. De esta forma, el usuario podrá escribir el fichero de reglas directamente desde un *Verification Report*[32]. Este método queda encuadrado dentro de la técnica formal de *Model Checking*.

El *Verification Report* debe contener una lista de reglas que modelan el funcionamiento básico del circuito — comúnmente denominadas *basic sanity tests* — un conjunto de reglas que verifiquen todas las funcionalidades definidas, y una lista de casos extremos — *corner cases*. Cada regla debe especificar el estado del cual se parte, la entrada al sistema y las acciones que deben ser verificadas.

La especificación basada en reglas puede combinar distintos tipos de especificaciones, según se muestra en la figura 1.11. A diferencia de los lenguajes de especificaciones, las reglas son un conjunto de expresiones aritmético–lógicas y de relaciones temporales que se compilan, creando un código C/C++. Para facilitar la sintaxis y la programación al usuario final de la herramienta, se suele disponer de los siguientes elementos:

- Un lenguaje sencillo para la entrada de las reglas. De esta forma, el usuario especifica las reglas usando un lenguaje propio que es transcompilado[33] a C/C++.

[32]Es un documento que contiene como debe hacerse la verificación a nivel de módulo, *cluster* o sistema y que debe ser comprobado exactamente en cada caso.

[33]Convertir una especificación de un lenguaje a otro distinto, de forma automática.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

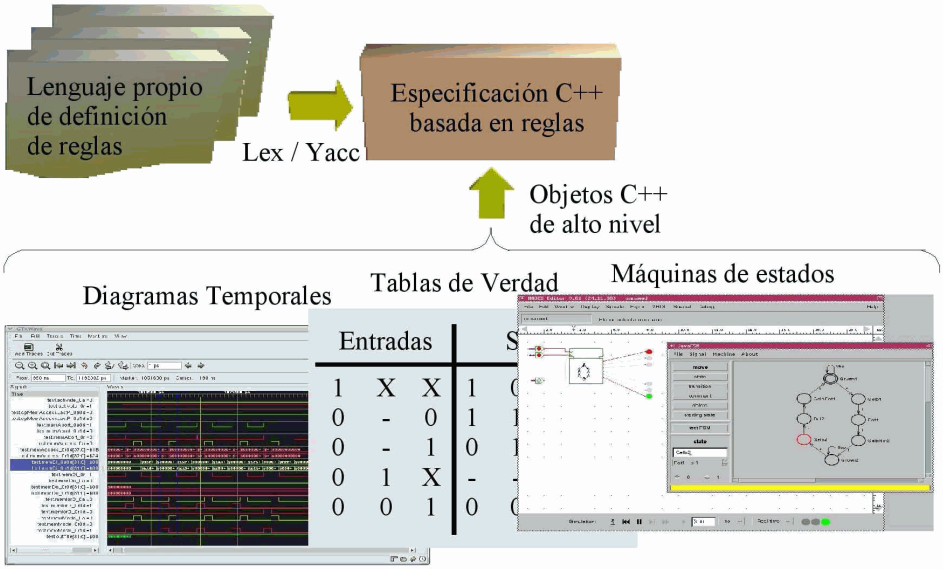


Figura 1.11: Distintas formas de generar la especificación de reglas.

El objetivo que debe cumplir dicho lenguaje propio, es simple: facilitar la especificación de las reglas lo máximo posible. Las herramientas de transcompilación, no sólo permiten al usuario definir su propio lenguaje mediante un conjunto de *tokens*[34], reglas de sintaxis y reglas de semántica; sino que además permiten especificar como se convierte la especificación del usuario del lenguaje propio a lenguaje C/C++.

- Objetos C/C++ de alto nivel que permiten especificar las reglas de distinta forma. El usuario puede mezclar en su especificación de reglas diagramas de tiempo, tablas de verdad, especificaciones de máquinas de estado, etc.

1.3.5. Análisis de Prestaciones con Exploración del Espacio de Diseño

La verificación funcional se puede realizar con una precisión de ciclo de reloj o, si existen estimadores de retardo, con una precisión cercana al comportamiento real. Como ya se avanzó en el apartado 1.2.1, el no disponer de dichos datos hasta acometer la etapa de síntesis, hace en la mayoría de los casos inviable la ejecución de los casos de verificación con valores de retardo aproximados.

Obtención del Punto de Trabajo del Circuito

En la actualidad las herramientas de síntesis procesan el RTL para obtener un punto del espacio de diseño que cumple con ciertas características, por ejemplo retardo–área activa o retardo–potencia de consumo. Fijado ese punto, los requisitos temporales de cada módulo, *cluster* y sistema han de ser satisfechos. Puesto que sólo se oferta un único punto de trabajo, el diseño puede o no cumplir con las especificaciones iniciales. En ese caso sólo quedan dos opciones, se repite la síntesis del elemento discordante para obtener un punto de trabajo mejor, o se recodifica el módulo, *cluster* o sistema hasta que se obtenga una solución temporal válida.

Si se optase por explorar el espacio de diseño de forma óptima, en vez de buscar un único punto del espacio de diseño, se obtendría el conjunto de estos puntos que presentan las mejores prestaciones, lo cual permitiría al diseñador poder es-

[34] Término anglosajón de símbolo.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

coger el punto de espacio de diseño que cumple las especificaciones y, por tanto, la verificación.

Puesto que el paso de síntesis lógica es casi inmediato, el sistema de verificación estaría en disposición de realizar simulaciones basadas en retardos aproximados una vez que cada módulo, *cluster* y sistema han sido codificados.

1.4. Entorno de Verificación Propuesto

La figura 1.12 presenta una diagrama que refleja las principales características del entorno de verificación que propone esta Tesis Doctoral.

Metodología de Verificación

El entorno de verificación permite realizar la verificación de cada uno de los casos críticos especificados por el verificador o acometer la verificación automáticamente. En ambos escenarios el entorno requiere como entrada una implementación del sistema bajo verificación y el conjunto de especificaciones a comprobar.

Métrica de Cobertura

Un punto clave en todo proceso de verificación es la métrica de cobertura. La métrica ha de ser aportada por el verificador. Un ejemplo de métrica es la enumeración de los casos a verificar en función de su importancia. Por ejemplo, casos básicos, intencionales o avanzados. Los casos a verificar se describen en un lenguaje cercano al humano, en base a reglas. El lenguaje de reglas permite por ejemplo, generar un determinado patrón de entrada y describir el patrón de salida esperado o patrón a comprobar.

El entorno de verificación propuesto tendrá implementadas varias técnicas para generar automáticamente patrones de entrada al DBV y al *Golden Reference*, además de comprobar que la respuesta del DBV es coherente con la respuesta del *Golden Reference*. El verificador puede aquí optar por emplear sus propias métricas de cobertura o emplear alguna de las métricas genéricas implementadas en el entorno. Por ejemplo, métricas genéricas como las basadas en: rutas, actividad circuital o potencia de consumo.

Exploración del Espacio de Diseño

La verificación se puede realizar con una precisión de ciclo de reloj, con temporización especificada por el usuario o empleando el módulo de análisis de prestaciones que incorpora el entorno de verificación. El módulo de análisis de prestaciones explora el espacio de diseño del circuito bajo verificación resolviendo la curva completa de prestaciones Área–Retardo o la curva completa de Potencia de Consumo–Retardo. Es decir, una vez el diseño ha superado la fase de síntesis lógica, el módulo de análisis de prestaciones estudia su ruta crítica para explorar su curva de prestaciones de forma óptima.

Finalizado el proceso de análisis de prestaciones y verificación del dispositivo a comprobar, se dispone de una implementación en forma de red de puertas lógicas dispuesta para el ensamblado del *layout* del circuito.

1.5. Organización de la Memoria

Así como en este capítulo 1 se hace una introducción a los sistemas de conmutación de última generación, su funcionamiento y verificación, en el capítulo 2 se describen los detalles de la implementación del entorno de verificación propuesto en esta Tesis Doctoral. El enfoque de dicho capítulo está orientado a describir la metodología propuesta para desarrollar la verificación. Se discutirá en detalle la

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

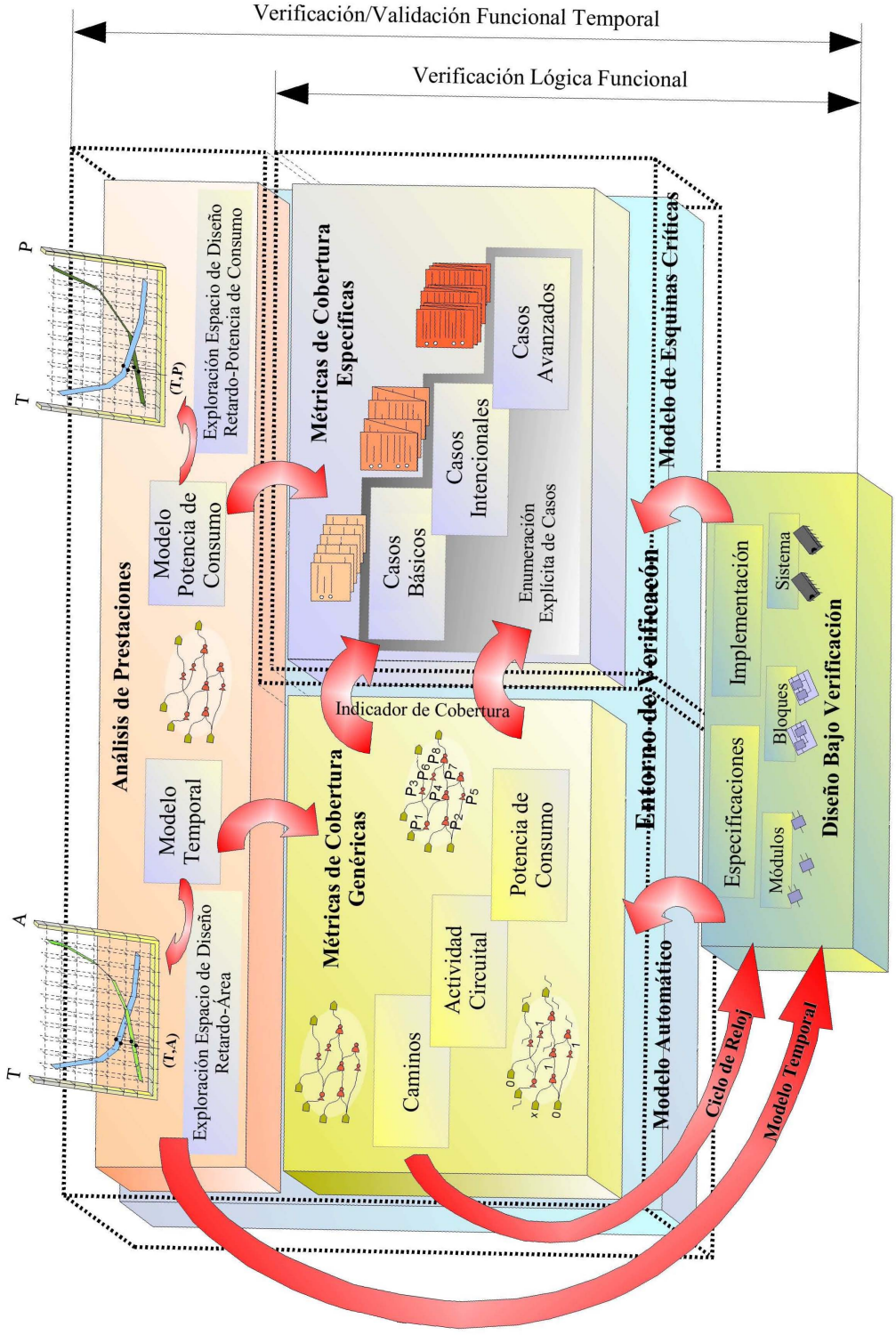


Figura 1.12: Entorno de verificación propuesto.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

implementación de cada una de las partes que componen el entorno de verificación y en particular las interfaces de comunicación entre el DBV, el verificador y el *Golden Reference*.

El capítulo 3 describe varias técnicas de generación de vectores para la verificación basada en distintas métricas de cobertura desarrolladas en el entorno de verificación. Dichos generadores se desarrollan empleando tanto técnicas heurísticas (Algoritmos Genéticos) como técnicas deterministas (Programación Lineal). En ese capítulo se presentarán varios generadores guiados por métricas de cobertura genéricas como la cobertura basada en rutas, actividad circuital o potencia de consumo.

En el capítulo 4 se describe en detalle la exploración del espacio de diseño de un circuito atendiendo a los parámetros retardo y área. Se presenta una metodología novel que se demuestra óptima tanto en tiempo de CPU como en la calidad de los puntos del espacio de diseño obtenidos.

Finalmente, en el capítulo 5 se extiende la metodología propuesta anteriormente a la exploración del espacio de diseño atendiendo a los parámetros retardo y potencia, demostrándose que la metodología de exploración sigue siendo válida para el parámetro potencia de consumo.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

Capítulo 2

Entorno de Verificación

Índice General

2.1. Introducción 33

2.2. Trabajos Previos 34

2.3. Priorización 38

2.4. Modelos de Verificación 38

2.4.1. Casos Críticos 39

2.4.2. Automático 40

2.5. Visión General del Entorno 43

2.6. Interfaces 43

2.6.1. Interfaz Paralela 45

Capa de Comunicación 46

Capa de Verificación 46

2.6.2. Interfaz Serie 47

Capa de Comunicación 47

Capa de Verificación 49

2.6.3. Interfaz de CPU 50

Capa de Comunicación 50

Capa de Verificación 50

2.7. El Sistema de Ficheros 52

2.7.1. Ficheros de la Interfaz de Enlace Serie 53

Fichero ACK 53

Fichero CRQ 53

Fichero de Entrada y Fichero Comprobación 54

2.7.2. Interfaz Paralela 54

Fichero de Paquetes de Entrada y Comprobación . . . 54

2.7.3. Interfaz de CPU 55

2.8. Características Avanzadas 55

2.8.1. Inyección de Errores 55

2.8.2. Sincronización de Interfaces 56

2.8.3. Jitter de Recepción Serie 57

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

2.8.4. Inserción de Nuevos Casos en Tiempo de Ejecución . 57
2.9. Implementación Interna 58
2.9.1. Nivel 0 58
2.9.2. Nivel 1 59
2.9.3. Nivel 2 59
Tipo Booleano: bool_t 61
Tipo Entero Con Signo: int_t 61
Tipo Entero Sin Signo: uint_t 61
Tipo Real: float_t 61
Tipo Imaginario: complex_t 61
2.9.4. Nivel 3 62
2.10. Resultados Experimentales 63
2.11. Conclusiones 66

Glosario de Términos

ACK	Acknowledgment, señal de respuesta de aceptación a una petición.
ASIC	Application–Specific Integrated Circuit.
BEM	Big Endian Mode.
CPU	Central Processing Unit.
CRC	Código de Redundancia Cíclica.
CRQ	Connection ReQuest, petición de conexión.
CSIX	Common Switch Interface.
DBV	Dispositivo Bajo Verificación.
DUT	Device Under Test, dispositivo bajo test.
DUV	Device Under Verification, dispositivo bajo verificación.
FIFO	First In First Out.
gcc	GNU C Compiler.
GMP	GNU Multiple Precision Arithmetic Library.
GNU	GNU is Not Unix.
HDL	Hardware Description Language.
ITU	International Telecommunication Union.
LIFO	Last In First Out.
LEM	Little Endian Mode.
$MGates$	Millones de Puertas.
OC–48	Optical Carrier–48, interfaz CSIX a 2'5 Gb/s.
OC–192	Optical Carrier–192, interfaz CSIX a 10 Gb/s.
OC–768	Optical Carrier–768, interfaz CSIX a 40 Gb/s.
PLI	Programming Language Interface, interfaz de lenguaje de programación.
RTL	Register Transfer Language, lenguaje de transferencia de registros.
Rx	Recepción.
Tx	Transmisión.
Verilog	VERIfying LOGic in general, lenguaje de descripción hardware.
VHDL	VHDIC HDL, lenguaje de descripción hardware de circuitos de muy alta velocidad.
VLSI	Very Large Scale Integration.

Este capítulo presenta una metodología novel orientada a verificar código RTL, a nivel de modulo, *cluster* y sistema, y con aplicación al diseño y verificación de sistemas de conmutación. La verificación del código se realiza mediante el entorno de desarrollo implementado en este trabajo, y probado en la fase de diseño y verificación de circuitos comerciales. Dicho entorno de desarrollo se descompone en diversos niveles, que permiten independizar el conjunto de pruebas de la implementación final del DUV. Este entorno dispone de tres capas o niveles de interfaz con el sistema a verificar. El primer nivel es la Capa de Comunicación,

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

el segundo es la Capa de Verificación y por último la Capa de Referencia. La Capa de Comunicación es la interfaz de bajo nivel con el código RTL. La Capa de Verificación es la interfaz entre los casos a comprobar y la interfaz de bajo nivel. La Capa de Referencia provee una interfaz entre una descripción de alto nivel descrita en C/C++ del sistema a verificar y la Capa de Verificación. También se ha desarrollado un generador de casos de verificación automática, sobre los casos críticos especificados por un usuario. El modo automático requiere un modelo de patrón referencia del sistema. Además, para dar soporte a la verificación automática, se han creado funciones de generación y comprobación de células de entrada y salida, funciones de sincronización entre interfaces, funciones de comprobación de ordenación y diversas características de control de flujo. El entorno de verificación ha sido completamente probado con circuitos integrados industriales, por ejemplo con el chip set GIGAStrem (con más de $10MGates$ por circuito). La metodología propuesta ha demostrado ser eficiente ya que ha permitido completar todo el banco de pruebas especificado. El capítulo está organizado como sigue.

En primer lugar, en el apartado 2.1, se realiza una introducción a la verificación de sistemas integrados. El apartado 2.2 desglosa los principales entornos de trabajo desarrollados para realizar la verificación funcional de un diseño. A continuación, en los apartados 2.3 y 2.4 respectivamente, se presenta la importancia de la priorización de las tareas de verificación así como los modelos de verificación funcional previstos en el entorno de verificación.

El apartado 2.5 se da una visión general del entorno desarrollado en esta Tesis Doctoral. Para posteriormente, en los apartados 2.6 y 2.7, mostrar en detalle las implementaciones de las interfaces con el dispositivo bajo verificación y con el verificador. En el apartado 2.8 se describen el conjunto de aportaciones novedosas que incorpora el entorno desarrollado. Y a continuación se exponen los detalles más significativos de la implementación interna del entorno (véase el apartado 2.9). El conjunto de experimentos a los que ha sido sometido son mostrados en el apartado 2.10. Finalizando el capítulo con las conclusiones alcanzadas con su implementación y los resultados de las pruebas a las que se ha sometido el entorno propuesto.

2.1. Introducción

La verificación es una de las tareas más importantes en el diseño de sistemas integrados. En esta etapa, se intenta asegurar el correcto funcionamiento del sistema integrado según lo especificado. La verificación, como tarea particular del flujo de diseño de sistemas integrados, absorbe cada vez más y más tiempo a medida que su complejidad aumenta. En la actualidad, el periodo de verificación de un diseño toma al menos la mitad del esfuerzo necesario para realizar un sistema integrado. Describir e implementar el conjunto de pruebas que ha de superar el circuito diseñado en un entorno de verificación realizado a medida, con lenguajes como Verilog, VHDL o en C/C++ y empleando PLI directamente, ha demostrado ser tedioso, poco productivo y de baja reusabilidad.

Lenguaje de Verificación

Si se añade la presión que supone cumplir con el *time–to–market*, se torna crítica la etapa de verificación. Por otro lado, debido a los cortos ciclos de vida de los productos — la mayoría de las ocasiones, los estándares implementados se encuentran en etapas tempranas de definición y por tanto están sujetos a las variaciones en su especificación final — el diseño y la verificación de los sistemas

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

integrados precisa de nuevas herramientas.

En ese sentido, se ha de tener en cuenta las sucesivas y continuas mejoras en cada dispositivo implementado y en un periodo de tiempo muy corto. Por ejemplo, en un conmutador de red, las sucesivas versiones de los dispositivos integrados siempre han de incluir interfaces más rápidos y confiables (OC–3, OC–12, OC–48, OC–192, etc), sistemas de colas complejos, nuevas políticas de encaminamiento y prioridades de servicio, entre otros.

En resumen, actualizar el diseño de un circuito integrado implica añadir una nueva característica funcional y mejorar sus interfaces (ancho de banda, protocolo, etc). Con el objeto de mantener la compatibilidad con las versiones previas, la funcionalidad principal ha de mantenerse inalterada. En otras palabras, a fin de reusar la mayoría del código de verificación y los bancos de prueba[1] desarrollados en productos previos, se ha de proponer una metodología que permita reconfigurar cada una de las interfaces implementadas, a medida que dichas interfaces cambien. Dicha actualización se ha de realizar de forma que la modificación afecte en menor medida al resto de código y bancos de prueba desarrollados.

time–to–market

Todo ello refuerza la necesidad de crear un entorno que reduzca la complejidad de la etapa de verificación, y más importante aún, que permita reducir el tiempo de desarrollo de un sistema integrado. Además, dicho entorno ha de permitir cambiar rápidamente sus interfaces y volver a ejecutar la verificación una vez el diseño sea actualizado.

Este capítulo presenta una metodología y un entorno que permite describir y verificar sistemas integrados, en general; y en particular sistemas de comunicaciones de alta complejidad de una forma amigable y rápida con C++, lenguaje–e o cualquier lenguaje HDL. La descomposición y posterior descripción de las interfaces del sistema en diversos niveles permite implementar todas sus interfaces de una forma natural permitiendo su posterior actualización instantáneamente. Esa facilidad radica en la diferenciación de las diversas funcionalidades a implementar en cada nivel y a la especificación de las interfaces entre niveles.

Además, se presenta la implementación de un entorno basado en unas librerías desarrolladas en C++ que permiten desarrollar descripciones hardware, disponer de todas las entidades de verificación, además de contar con las potencialidades de un lenguaje orientado a objeto. Las librerías resultantes proveen una interfaz sencilla para describir sistemas e implementar sus bancos de pruebas. Como valor añadido, se ha incluido una conexión transparente al simulador HDL. Los resultados alcanzados han presentado una ganancia significativa en la creación de bancos de prueba reutilizables y en la depuración de simulaciones en ejecución.

2.2. Trabajos Previos

Una vez la funcionalidad básica de un diseño ha sido definida, se realiza un plan de verificación. Este plan es descrito a diferentes niveles. Por ejemplo, a nivel de módulo, se comprueba la conformidad de cada característica funcional del módulo. A nivel de bloques, el objetivo principal es cotejar los protocolos entre módulos. El objeto de la verificación a nivel de circuito integrado es comprobar el funcionamiento global.

Verificación en Diversos Niveles

A nivel de módulo, la temporización de los eventos es crítica, mientras que el procesado de datos se encuentra en un segundo plano. En este nivel de verifica-

[1] Traducción directa del término anglosajón *testbench*.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

ción puede ser empleada una metodología basada en transacciones [18]–[19]. Esta metodología está integrada en una plataforma de verificación como la presentada en [11].

La verificación a nivel de bloque o circuito integrado, implica trabajar con estructuras de datos complejas, mientras que los protocolos son menos relevantes. Por ello se requiere en general un lenguaje de verificación de alto nivel. Hoy día están disponibles varios entornos de verificación comerciales como [20] y no comerciales como [21], [22] los cuales se basan en lenguajes como Verilog/VHDL, C/C++, Superlog [23] o lenguaje–e [20].

El tiempo para producir un diseño y verificarlo es muy reducido de acuerdo con las restricciones impuestas por el *time–to–market*. La adecuada elección de los casos a comprobar, en unión con la elección correcta del entorno de verificación en cada nivel de verificación, es un factor clave que permite reducir significativamente el tiempo de la etapa de verificación [24]–[26].

time–to–market

La escritura de bancos de prueba en HDL dentro de un entorno de verificación hecho a medida es tedioso e improductivo. De entre todos los lenguajes empleados en la verificación, el C++ [27] es una solución potente para desarrollar los conceptos de encapsulación, clases y plantillas. Con el entorno descrito en este documento, el esfuerzo necesario para escribir bancos de pruebas en C++ se reduce significativamente. C++ es un lenguaje estándar, y muchos ingenieros ya han tenido experiencias de trabajo con él. En [2] se concluye que más del 80 % de los diseñadores encuestados ya emplean o tienen intención de utilizar C/C++ en sus proyectos.

Cuando un banco de pruebas es realizado en C++, existen modelos C/C++ y

[18] Dhananjay S. Brahme, Steven Cox, Jim Gallo, Mark Glasser, William Grudmann, C. Norris Ip, William Paulsen, John L. Pierce, John Rose, Dean Shea, and Karl Whiting. Creating a C++ Library for Test Bench Authoring. Technical Report CDNL-TR-2000-0820, Cadence Berkeley Labs, August 2000.

[19] Steve Forde, Steve Bishop, and Rammath S. Velu. Streamlining HDL Code Coverage Analysis. *Integrated System Design*, December 1998.

[11] Gale Morrison. Cadence Open–Sources T–Builder. *Electronics News*, August 2000.

[20] T. Kuhn, T. Oppold, M Witerholer, W. Rosenstiel, Marc Edwards, and Yaron Kashai. A Framework for Object Oriented Hardware Specification, Verification, and Synthesis. *IEEE Design Automation Conference*, pages 18–22, June 2001.

[21] H. Navarro, Juan A. Montiel-Nelson, J. Sosa, and R. Sarmiento. DEMETER: A Novel Framework for Hardware and Software System Specification, Simulation and Verification in C++. *XVI Conference on Design of Circuits and Integrated Systems*, pages 20–23, November 2001.

[22] Matthias Bauer and Wolfgang Ecker. Hw. Sw. Co–Simulation in a VHDL–Based Test Bench Approach. *Proc. of the 34th Conference on Design Automation*, pages 70–75, June 1987.

[23] Peter L. Flake and Simon J. Davidmann. Superlog, a Unified Design Language for System–on–Chip. *IEEE Design Automation Conference, Asia and South Pacific*, pages 25–28, January 2000.

[24] Hong Peng and Sofiene Tahar. Verification and Validation of Complex Digital Designs - A Practical Perspective. *IEEE VTS Tutorial*, June 2001.

[25] Hong Peng and Sofiene Tahar. Compositional Verification of an ATM Bit Error Rate Monitor Circuit. Technical report, Dept. of Electrical & Computer Eng., Concordia University, March 1999.

[26] Hong Peng and Sofiene Tahar. Hardware Modeling and Verification of an ATM Ring MAC Protocol. *The 12th International Conference on Microelectronics*, November 2000.

[27] Bjarne Stroustrup. *The C++ Programming Language*. Addisson-Wesley, 3rd edition, 1998.

[2] EDA today. Electronic Design Automation User Study, 1999.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

otros scripts que pueden ser incorporados fácilmente. C++ es un lenguaje potente en el cual se puede crear una librería que contenga los conceptos específicos que son necesarios en verificación. Esta librería puede ser reutilizada durante un proyecto y también en varios proyectos en desarrollo.

Habilidad de Codificación

La habilidad para emplear programación orientada a objetos y los otros paradigmas de C++ facilitan la reusabilidad de cada componente de un banco de pruebas, en el ciclo de diseño de un proyecto, o entre proyecto y proyecto en diferentes simuladores, y diferentes etapas del desarrollo. La habilidad para usar hilos de ejecución[2] es también favorable en la escritura de bancos de prueba.

El lenguaje C++ posee sus limitaciones. Con una librería no diseñada correctamente para ayudar al usuario, la potencia del C++ podría ser difícil de encajar. Un programador novel en C++ podría no ser capaz de crear una clase correcta y eficiente, pero es muy fácil para una persona emplear una clase diseñada e implementada de propósito específico. Con el empleo de los componentes ya implementados en la librería C++, los usuarios pueden crear y adaptar sus bancos de prueba simplemente mediante la utilización de las clases apropiadas de la librería [18].

Algunas de las ventajas de emplear C++ pueden ser:

Comprobación Funcional.– Los diseñadores usualmente comienzan a trabajar con un modelo C++ que emplean como descripción inicial. Esta descripción inicial puede ser dividida en diversos bloques, módulos y demás. Una implementación hardware correcta de esos bloques y módulos es necesaria.

Reutilización del Código.– las descripciones de modelos diseñados y sus bancos de prueba pueden ser reutilizados en todo el ciclo de diseño. Parte de los bancos de prueba escritos para un sistema pueden ser reutilizados para desarrollar nuevos sistemas hardware.

Encapsulación y Distribución.– Un diseñador podría desarrollar sus propios módulos en C++, y encapsularlos en sus módulos desarrollados en un fichero de librería y un fichero de cabeceras con la interfaz de entrada/salida del módulo de más alto nivel [21].

Especificación Hardware

En el pasado, se han presentado varias soluciones que permiten especificar sistemas hardware en base a objetos. Por ejemplo, se han propuesto extensiones de lenguajes de descripción hardware para incluir los conceptos de programación

[2] Término identificado en la literatura inglesa como *threads*.

[18] Dhananjay S. Brahme, Steven Cox, Jim Gallo, Mark Glasser, William Grudmann, C. Norris Ip, William Paulsen, John L. Pierce, John Rose, Dean Shea, and Karl Whiting. Creating a C++ Library for Test Bench Authoring. Technical Report CDNL-TR-2000-0820, Cadence Berkeley Labs, August 2000.

[21] H. Navarro, Juan A. Montiel-Nelson, J. Sosa, and R. Sarmiento. DEMETER: A Novel Framework for Hardware and Software System Specification, Simulation and Verification in C++. *XVI Conference on Design of Circuits and Integrated Systems*, pages 20–23, November 2001.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

orientada a objetos en [28], [22], y [29]. Existe también la tendencia inversa donde los diseñadores cambian el lenguaje HDL para desarrollar sus productos, en lenguajes orientados a objeto propietarios como el e–lenguaje [20] o C++. Actualmente existen diversas herramientas comerciales y no comerciales que explotan el desarrollo y verificación en C++, y como ejemplo de ello está Superlog [23] de la empresa Co–Design Automation, System–C [16] de CoWare o CynLib [30] de CynApps.

C++ Frente HDL

En contraste con la mayoría de los simuladores HDL, C++ es un lenguaje compilado[3], y obviamente, un modelo de sistema compilado se ejecuta mucho más rápido que un modelo de hardware interpretado; incluso si este es orientado a objeto, además, esta característica no aporta ningún beneficio en tiempo de ejecución. Un código escrito en C++ podría ir hasta 3 veces más rápido que el mismo código escrito en HDL [21].

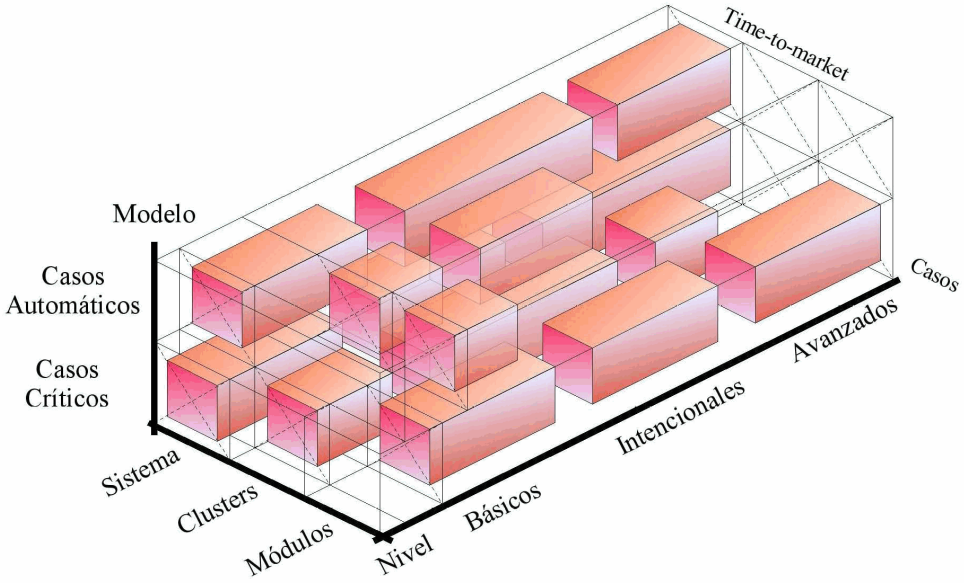


Figura 2.1: Verificación de un sistema integrado.

[3]No es así con NC–Verilog, de Cadence, que se compila para ser ejecutado.

[28] Sowmitri Swami, Arthur Molin, and Burt Counot. Object Oriented Extensions to VHDL. *IEEE Computers*, 28(10), October 1995.

[22] Matthias Bauer and Wolfgang Ecker. Hw. Sw. Co–Simulation in a VHDL–Based Test Bench Approach. *Proc. of the 34th Conference on Design Automation*, pages 70–75, June 1987.

[29] M. Radetzki, W. Putzke-Roming, and W. Nebel. Objective VHDL: The Object–Oriented Approach to Hardware Reuse. *Advanced ins Information Technologies*, The Business Chalenge, 1997.

[20] T. Kuhn, T. Oppold, M Witerholer, W. Rosenstiel, Marc Edwards, and Yaron Kashai. A Framework for Object Oriented Hardware Specification, Verification, and Synthesis. *IEEE Design Automation Conference*, pages 18–22, June 2001.

[23] Peter L. Flake and Simon J. Davidmann. Superlog, a Unified Design Language for System–on–Chip. *IEEE Design Automation Conference, Asia and South Pacific*, pages 25–28, January 2000.

[16] Open SystemC Comunity. SystemC: Entorno de Desarrollo y Verificación de Sistemas, 2000.

[30] CynApps. *System Modelling with Cynlib*, v1.1 edition, December 1999.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

2.3. Priorización

Tomando como premisa que el tiempo necesario para desarrollar y comprobar todas las funcionalidades de un dispositivo, y así asegurar su conformidad con su especificaciones iniciales es muy corto; se plantea la necesidad de priorizar tanto el desarrollo como la ejecución de las diversas funcionalidades especificadas, de acuerdo a algún criterio.

Identificación de la Importancia de los Casos a Verificar

Por lo general, dicha priorización atiende al grado de importancia de la función bajo verificación. En los primeros momentos de la etapa de verificación se ha de realizar una clasificación de las funciones a implementar, asignando un nivel de prioridad en relación a su nivel de importancia sobre el conjunto del sistema. Se proponen tres grupos de priorización. Estos son: el grupo básico, intencional y avanzado.

El conjunto de casos básicos define aquellas situaciones que han de ser comprobadas al comienzo de la verificación. Las funcionalidades agrupadas en este nivel de priorización implementan los requisitos denominados imprescindibles/mínimos del sistema a verificar. Por ejemplo, dentro de la verificación de un microprocesador, las operaciones de lectura y/o escritura. Dentro de un sistema de conmutación de paquetes, en este grupo entrarían las funcionalidades de configuración del sistema o los sincronismos de y entre diversas interfaces.

El conjunto de casos intencionales comprenden aquellas situaciones que han de ser verificadas en segundo lugar, tras la comprobación de todos los casos básicos. La complejidad y la cantidad de los casos intencionales es importante y por lo general muy superior al de los casos básicos. Un ejemplo de casos intencionales, en la verificación de un sistema de conmutación de paquetes, son los errores de CRC y paridad vertical de las interfaces, entre otros.

Casos Avanzados

Finalmente, el grupo de los casos avanzados contiene aquellas funcionalidades que plantean situaciones a comprobar, y que por razones de tiempo no son posible de comprobar si se desea cumplir con el *time–to–market* del producto. Las funcionalidades incluidas en este grupo de priorización son aquellas que pueden ser obviadas de la funcionalidad final dispuesta en el circuito. No todas las especificaciones iniciales son incluidas en el producto final. En la actualidad, el conjunto de especificaciones iniciales incluyen funcionalidades avanzadas que se añaden al producto final con el objetivo de evaluar su eficacia en futuras versiones. Dichas funcionalidades permanecen ocultas al usuario final del sistema integrado y dan la oportunidad al diseñador de comprobarlas en tiempo real.

La figura 2.1 presenta el tiempo requerido para comprobar las especificaciones de un diseño, siguiendo una metodología basada en casos críticos y en casos automáticos, para todos los niveles en la verificación. El tiempo requerido si todo el proceso se realizase en serie sería extraordinariamente superior al *time–to–market* (plano derecho superior en la gráfica).

2.4. Modelos de Verificación

Independientemente del número de casos, propiedades o funcionalidades a comprobar en el sistema bajo verificación, esta se puede realizar en base a las llamadas esquinas críticas[4] o mediante un sistema de generación de casos automáticos. Es importante tener en cuenta que no se ha de limitar la comprobación del

Metodología

[4]Referenciado en la literatura anglosajona como *corner cases* o casos críticos.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

sistema integrado, bloque o *cluster* a uno de estos dos métodos, exclusivamente, pues se ha demostrado, fehacientemente, que a pesar de superar alguno de los dos métodos, puede permanecer encubierto algún error [1]. En general, en la etapa de verificación se emplean ambos métodos con el objeto de reducir la probabilidad de no comprobar la existencia de un error.

2.4.1. Casos Críticos

Son necesarios para verificar los casos especiales. El número de casos críticos ha de ser reducido al mínimo. En particular, la complejidad de especificar e implementar casos críticos aumenta a medida que el diseño gana complejidad. Por ello, el número de casos críticos a implementar a nivel de sistema ha de ser muy inferior al número de casos críticos en el nivel de bloques, y este muy inferior al número de casos críticos en el nivel de módulo.

En este modelo de verificación, el usuario ha de proveer los estímulos de entrada y, opcionalmente, las salidas esperadas. El objetivo de este modelo es comprobar un comportamiento determinista del circuito. Los casos particulares son escritos manualmente en los ficheros de entrada y de verificación, para ser comprobados en un proceso de corta duración. Este modo asegura la correctitud de ciertas funcionalidades.

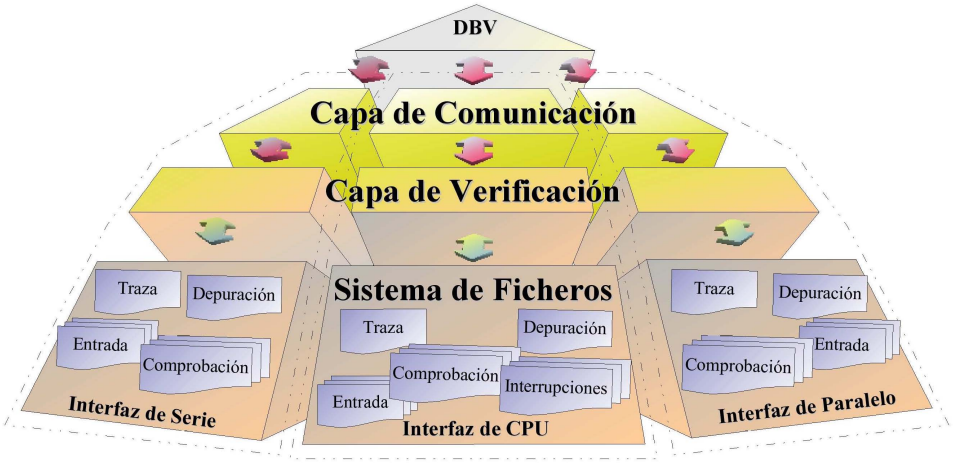


Figura 2.2: Esquema del modo de verificación de casos críticos.

La figura 2.2 ilustra el entorno de verificación propuesto con una configuración siguiendo el modelo de casos críticos, donde:

Dispositivo Bajo Verificación (DBV)[5]**.–** Es la descripción del dispositivo a nivel de circuito integrado, bloque o módulo del conmutador de paquetes, escrita en RTL.

Circuito a Verificar

Capa de Comunicación.– Provee un mecanismo basado en eventos, para enlazar el código RTL y el entorno de verificación. En esta etapa se incluyen todos los mecanismos de sincronismo y conversión de tipos entre el simulador

Nivel Hardware

[5]Traducción del termino anglosajón *Device Under Verification o sus siglas DUV*.

[1] Sedar Tasiran and Kurt Keutzer. Coverage Metrics for Functional Validation of Hardware Designs. *IEEE Design and Test of Computers*, 18(4):36–45, August 2001.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

HDL empleado y el lenguaje de programación de alto nivel utilizado. Por ejemplo, se traducen las estructuras de datos complejas de C/C++ a bits y bytes de Verilog XL mediante las librerías PLI — en caso de describir el entorno de verificación con C/C++ y seleccionar el simulador HDL Verilog XL de Cadence.

Capa de Verificación.– Contiene diversas funciones directas e inversas para traducir los comandos de entrada en lenguaje natural en cadenas de estímulos dispuestas para administrarse a la Capa de Comunicación. Además, esta capa posee diversas rutinas de comprobación. Estos procedimientos de verificación realizan comparaciones entre la respuesta del sistema a un conjunto de estímulos y su respuesta esperada. Las funcionalidades implementadas en esta capa del entorno de verificación se corresponden por ejemplo con diversos compiladores que permiten al verificador implementar un lenguaje cercano al humano a la hora de especificar sus casos críticos. Además se incluyen las funciones típicas de cálculo de CRC con diversos polinomios o paridad.

Comprobaciones Básicas

Ficheros de Estímulos.– Son el conjunto de ficheros que indican a la Capa de Verificación como generar la cadena de datos a aplicar al dispositivo bajo verificación. Existe al menos un fichero de estímulos por cada interfaz del sistema. Los estímulos especificados dentro de cada fichero se corresponden con una sintaxis y léxico definido en la Capa de Verificación por el verificador. Además, hay diversos ficheros de configuración que definen el modo de operación de la interfaz. Estos ficheros de configuración también siguen la especificación expresada por el verificador en la Capa de Verificación.

Especificación del Caso a Verificar

Ficheros de Comprobación.– Están formados por un conjunto de ficheros que definen el comportamiento de las salidas de las interfaces. Se pueden definir, opcionalmente, en cada interfaz del dispositivo bajo verificación. Poseen una estructura interna parecida a la de los ficheros de estímulos, salvo que en estos se incluyen estructuras condicionales que permiten tomar decisiones donde sean necesarias. Por último, dentro del sistema de ficheros de comprobación se encuentra el fichero de traza de cada interfaz. En este último fichero se reflejan todos los datos y eventos acaecidos en la interfaz.

Comprobación de Resultados

Las capas de comunicación y verificación proveen una primitiva de comunicación para cada interfaz física del sistema, e independiente. Esto permite realizar actualizaciones de cualquier parte de la interfaz sin afectar al resto de las interfaces del DBV. Esas interfaces son básicamente un enlace entre el verificador y el simulador hardware empleado.

2.4.2. Automático

El modelo automático se basa en el llamado modelo de referencia o patrón[6] que, normalmente, está codificado en un lenguaje de alto nivel como C/C++. Esta descripción a nivel de circuito integrado, *cluster* o módulo provee el comportamiento de referencia del circuito bajo verificación.

Modelo Automático

El objetivo en este modelo de funcionamiento, es emplear el sistema de referencia para verificar casos generales y comprobar las prestaciones del sistema,

[6] En la literatura anglosajona referenciado como *Golden Reference*.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

cluster o módulo RTL. La figura 2.3 ilustra la configuración del entorno de casos automáticos.

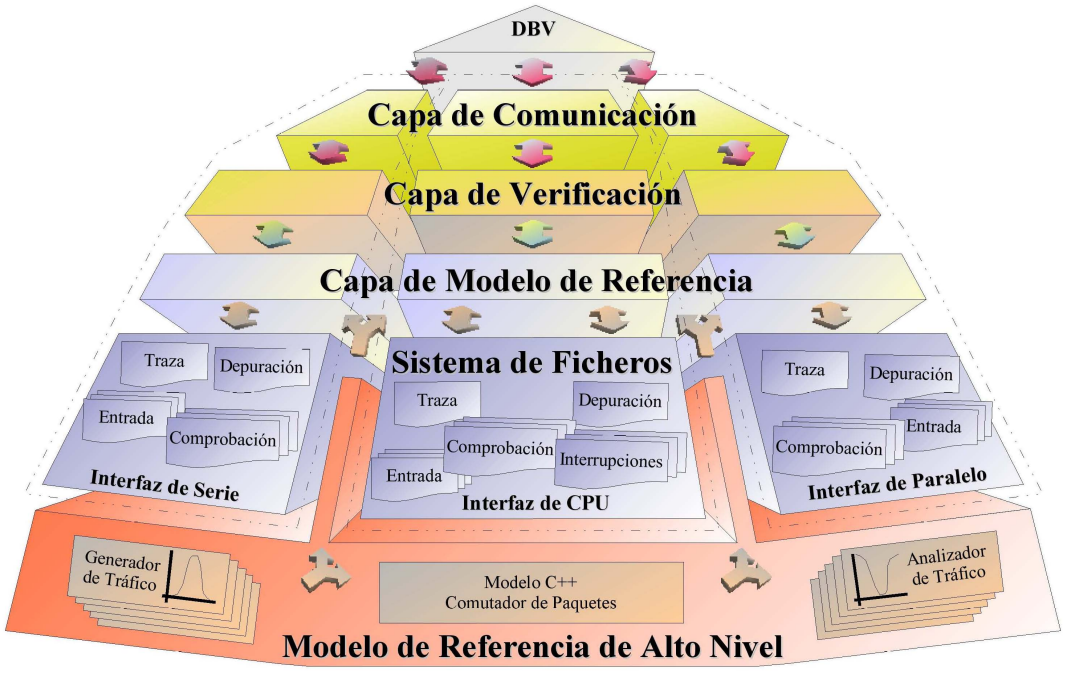


Figura 2.3: Esquema del modo automático de verificación.

En el modelo automático, la Capa de Comunicación y la Capa de Verificación no se ven alteradas con respecto al modelo de casos críticos. Sin embargo, en el modelo automático se añade una capa extra llamada Capa de Referencia. A su vez, los estímulos y los ficheros de entrada cambian su finalidad y son empleados como ficheros de traza.

La Capa de Referencia.– Se emplea para comunicar el modelo de alto nivel con el entorno de verificación y, por tanto, con el dispositivo bajo verificación (en general, codificado en RTL). Esta capa se emplea con dos propósitos. Primero, introducir los estímulos al dispositivo de forma automática, esto es, reemplazar los ficheros de estímulos y comprobación del verificador en el modelo de casos críticos. Y segundo, sincronizar los eventos basados en franjas temporales o *slots* de tiempo[7] del modelo C/C++ con los eventos hardware (RTL).

Sincronización entre Modelos

Los Ficheros de Estímulos y Comprobación.– Son empleados como ficheros de traza y, por tanto, son de salida. En cada simulación, la Capa de Referencia genera los estímulos y los ficheros de comprobación. Esos ficheros poseen la misma sintaxis y estructura que poseen en el modelo de casos críticos. Ello permite reproducir una situación particular generada por el modelo automático mediante el modelo de casos críticos. En ese caso particular, se podrían entonces aplicar todas las técnicas de depuración existentes, lo cual sería inviable siguiendo el modelo automático debido a su alto coste computacional y de almacenamiento.

Registro de Sucesos

[7]En la bibliografía se identifica este término como *slot–time*.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

La idea básica es reutilizar el entorno de simulación de la arquitectura, bloques o módulos para generar las salidas esperadas, definida una configuración de los mismos. Ello implica configurar, ejecutar y sincronizar el modelo de referencia y la implementación del DBV, con el objeto de comprobar que ambos modelos presentan salidas idénticas.

Filosofías de Ejecución

Atendiendo a la naturaleza de ejecución de los simuladores desarrollados hasta ahora, estos se pueden clasificar en dos tipos. Los guiados por eventos y los basados en *slot* temporales[8]. A pesar de parecer dos filosofías completamente independientes, nada más lejos de la realidad, todos ellos emplean eventos. Las ocurrencias de sucesos define el concepto de evento. Lo que los diferencia es la temporización que ejecutan.

Simuladores por Eventos

Los llamados simuladores guiados por eventos se emplean cuando la carga temporal del sistema a simular es baja con respecto al tiempo de ocurrencia de los mismos, es decir, cuando los tiempos entre eventos es grande. La aplicación que desarrolla este tipo de metodología se basa en crear una lista temporal de eventos. La cabeza o primer elemento de la lista representa el evento más cercano a ejecutar. Y la cola o último elemento, el evento más lejano a ejecutar. La aparición de un evento, implica que éste sea insertado ordenadamente en la lista de eventos, para que el núcleo de ejecución pueda seguir la cronología de eventos o sucesos del sistema. La variable que representa al tiempo en este tipo de simuladores es incrementada con la ejecución de cada evento de la lista. Dichos incrementos no son constantes, pues la distancia temporal entre eventos es variable.

Cuando la carga temporal de eventos es muy alta, o dicho de otra forma, existe una muy alta probabilidad de ejecutar uno o más eventos en cada instante de tiempo, se demuestra, que los simuladores basados en técnicas *slot–time* son más eficientes que los guiados, puramente, por eventos. Esto es debido, principalmente, a que si se tiene la certeza de la existencia de al menos un evento en cada instante de tiempo, la cola de eventos es inoperativa, pues el núcleo de simulación crea y destruye eventos en casi todos los instantes de reloj. El modelo basado en *slot–time* ejecuta en cada instante de reloj al menos un evento. La variable que representa el tiempo en este tipo de simuladores se incrementa en una unidad cada vez que se ejecutan todos los eventos de un *slot–time*.

Simuladores slot–time

En general, los simuladores de conmutadores de paquetes están basados en núcleos de ejecución *slot–time*, pues para obtener las prestaciones del mismo se ha de simular éste en alta carga, y por tanto, hay una muy alta probabilidad de ocurrencia de suceso en cada instante de tiempo. Pero, por otro lado, hay que tener en cuenta que los simuladores de lenguajes como Verilog/VHDL poseen núcleos de ejecución guiados por eventos, fomentado este hecho por la propia naturaleza del RTL.

En resumen, la Capa de Referencia se añade para sincronizar los eventos de la Capa de Verificación con los eventos el modelo de referencia. Es decir, para unir los mundos de simulación basada en eventos del código RTL con el mundo de los eventos del modelo de referencia, pudiendo ser este último basado en eventos o *slot–time*, según se halla desarrollado el modelo de referencia con uno u otro núcleo de simulación.

[8] Término referenciado en la literatura anglosajona como *slot–time*.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

2.5. Visión General del Entorno

Implementación Interna

Aunque C++ es un lenguaje de propósito general y por sí mismo no está diseñado para modelar hardware, los conceptos importantes del hardware se pueden encapsular con un sistema de clases y los métodos apropiados [31]. Mientras que C++ es un lenguaje predominantemente secuencial la ilusión de concurrencia y las estructuras estáticas son logradas mediante diversas utilidades de multihilo[9], detalles permanecen ocultas al usuario [32].

La figura 2.4 presenta el entorno propuesto, que permite la verificación de dispositivos mediante la ejecución de bancos de prueba escritos en C++, controla a un simulador HDL, como por ejemplo Cadence XL/NC. La comunicación con el simulador HDL se realiza mediante una Capa de Comunicación, la cual realiza funciones de acceso a la capa física del HDL. La Capa de Comunicación también actúa con la interfaz de la Capa de Verificación. La Capa de Verificación es donde se describen los bancos de prueba [33].

En las etapas iniciales del desarrollo de un circuito integrado, solamente se encuentra disponible una descripción del dispositivo en lenguaje HDL, los bancos de prueba se ejecutan entonces sobre una estación de trabajo donde reside el simulador. Poco a poco, esta descripción pasa a ser un ASIC. Con el fin de mantener una interfaz entre la descripción HDL (software) y la parte de la descripción mapeada en el ASIC (hardware), una sección de la capa de acceso físico ha de ser transferida a memoria y ejecutada en un microprocesador. El resto de esta capa comunica el microprocesador y la estación de trabajo — donde reside el simulador. Las otras dos capas — Capa de Comunicación y de Verificación — continuarán ejecutándose en la estación de trabajo, permaneciendo sin alteraciones durante el ciclo de diseño.

2.6. Interfaces

Protocolos de Comunicación

Atendiendo a los protocolos de comunicación empleados en la actualidad tanto entre módulos, *clusters* o sistemas integrados, se puede concluir que hay tres tipos básicos bien diferenciados. A medida que dichos tipos básicos se emplean en un nivel de integración superior, su refinamiento es mayor. Además, sobre dichos tipos básicos se realizan diversas aportaciones que suponen la inclusión de alguna novedad sobre estos tres tipos básicos.

Estos tres tipos básicos son: las interfaces serie de alta velocidad, las interfaces de gran ancho de palabra paralelas y las interfaces de control o CPU.

Las interfaces serie pueden ser síncronas o asíncronas. Por lo general, son asíncronas, con lo cual se ha de conocer la velocidad de bit, velocidad de muestreo,

[9] Traducción directa del término anglosajón *Muti–thread.*

[31] M. V. Shahdadpuri, J. Sosa, H. Navarro, Juan A. Montiel-Nelson, and R. Sarmiento. A New Framework for Hardware System Verification Based on C++. *XVI Conference on Design of Circuits and Integrated Systems*, pages 20–23, November 2002.

[32] M. V. Shahdadpuri, J. Sosa, H. Navarro, J.A. Montiel-Nelson, and R. Sarmiento. VESTA: A System Level Verification Environment Based on C++. In *Proc. of SPIE of VLSI Circuits and Systems*, May 2003.

[33] J. Sosa, J. A. Montiel-Nelson, H. Navarro, M. V. Shahdadpuri, and R. Sarmiento. System–Level Verification Methodology for Advanced Switch Fabrics. In *Proc. of SPIE of VLSI Circuits and Systems*, May 2003.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

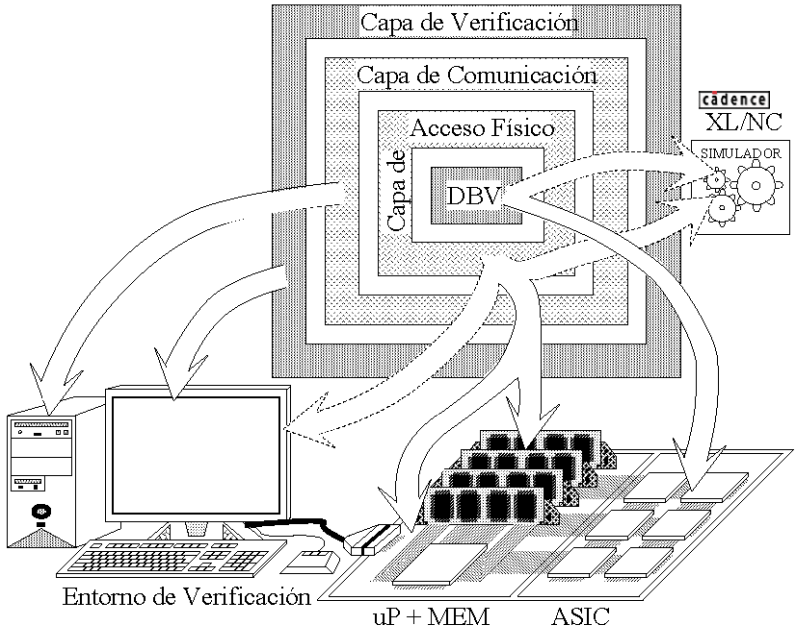


Figura 2.4: Entorno de desarrollo de bancos de prueba.

la marca de comienzo y fin de transmisión. Este tipo de interfaz se diseña en la mayoría de los casos en el límite de la tecnología seleccionada para obtener una alta tasa de transmisión/recepción. En particular, en los sistemas de comunicación basados en paquetes se emplean para comunicar las distintas unidades origen y destino.

Las interfaces paralelas nacen como evolución de la tecnología serie, realizando la transmisión paralela de varios bits a un mismo tiempo. El ancho de palabra es el que realmente determina la tasa de bits transmitido. Estas interfaces pueden ser síncronas o no con un reloj. Y cuando son síncronos, dicha señal de reloj puede estar omitida en el bus. Un ancho de palabra superior al bit transmitido por un puerto serie permite reducir la velocidad de transmisión/recepción del puerto paralelo. No obstante, se ha de tener en cuenta que a mayor ancho de bit, el dispositivo empleará un número mayor de terminales y superficie de interconexionado. Ello implica grandes problemas de interconexionado a nivel de sistema y circuito integrado.

Por último y no por ello menos importante, se incluye una interfaz de control o CPU. Esta no es una interfaz dedicada a la transmisión masiva de datos como es el caso de la serie o paralela, su utilidad se centra en controlar el dispositivo y/o permitir el acceso a la información de depuración en el mismo. Suele tener una estructura parecida a la interfaz paralela, al que se han añadido diversas señales con múltiples propósitos, como por ejemplo, las señales de interrupción[10] o *reset*[11].

A continuación se describe en detalle cada una de las interfaces implementadas en el entorno de verificación. En el modelo basado en casos críticos, el usuario ha de proveer los estímulos de entrada, los ficheros de configuración y opcional-

[10]Una interrupción es una señal que indica al dispositivo que la recibe, que ha de salvar el contexto de ejecución en que se encuentra y pasar a ejecutar una rutina de servicio específica al evento acaecido.

[11]El *reset* no deja de ser una señal de interrupción que indica al dispositivo que comience su ejecución desde el inicio de la rutina de funcionamiento normal.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

mente los de comprobación de los estímulos de salida para cada interfaz. Es decir, al menos un fichero de estímulos y configuración por cada interfaz serie, paralela y/o de CPU presentes en el dispositivo bajo verificación. Opcionalmente, el verificador puede incluir el fichero de respuestas esperadas por cada interfaz.

En el modelo automático, la Capa de Referencia emplea el dispositivo patrón para obtener los estímulos de entrada y los estímulos esperados en cada interfaz. Los ficheros de configuración de cada interfaz han de seguir siendo especificados por el verificador.

En los próximos apartados la Capa de Referencia será obviada con el objeto de aclarar correctamente la descripción de cada interfaz. Esta capa sería tan sólo un traductor de los eventos del simulador basado en *slot–time* al simulador guiado por eventos.

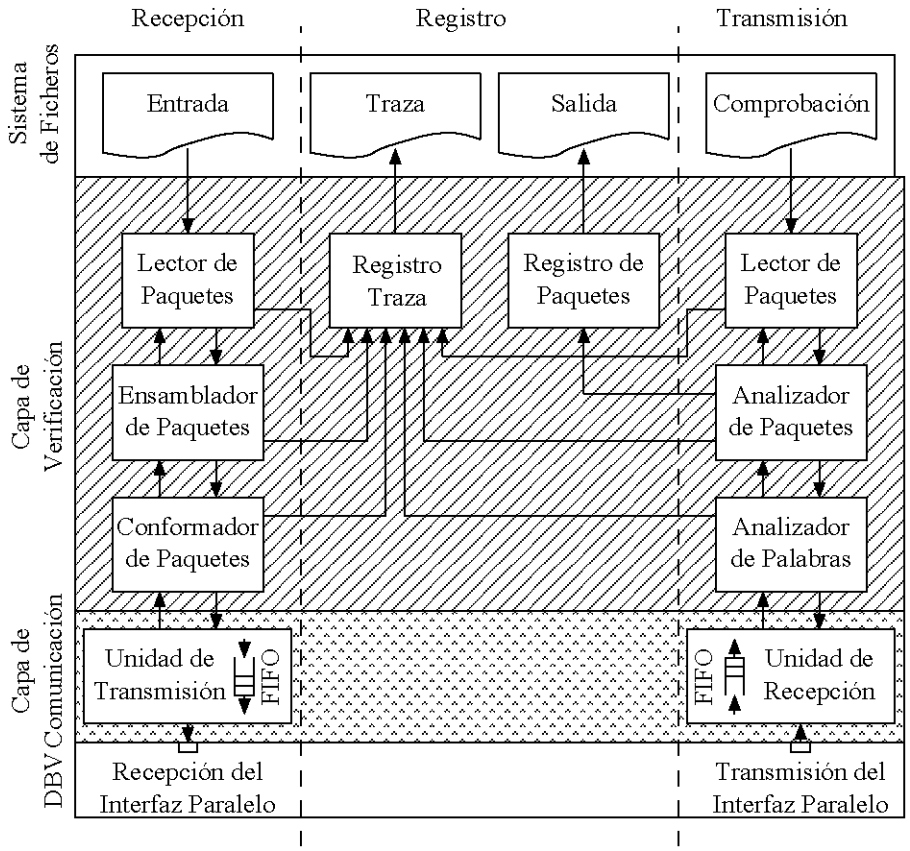


Figura 2.5: Diagrama de bloques de la interfaz paralela.

2.6.1. Interfaz Paralela

La Capa de Comunicación paralela es altamente dependiente de la interfaz física RTL paralela. La Capa de Verificación es un grupo de funciones y procedimientos que permiten al entorno acceder a la interfaz paralela RTL y comprobarla. La interfaz paralela implementada se corresponde con el estándar CSIX [34]. Esta interfaz se corresponde con un puerto paralelo de ancho de palabra variable y

Estándar CSIX

[34] Common Switch Interface Consortium. *CSIX–L1: Common Switch Interface Specification–L1*, v1.0 edition, May 2000.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

múltiplo de 32 bits. Las transmisiones poseen sincronismo a nivel de palabra, y para mantener dicho sincronismo, la interfaz se mantiene, constantemente, transmitiendo datos o información de sincronismo. Los datos se encapsulan dentro de las llamadas células de información. Con cada palabra transmitida se adjunta una paridad horizontal de la misma. Al final de cada célula se adjunta una palabra de paridad vertical calculada sobre toda la célula.

Implementación por Capas

Esta unidad está compuesta de tres secciones: Recepción, Transmisión y Registro de datos. Las secciones de Recepción y Transmisión operan simultánea e independientemente. La sección de Registro actúa exclusivamente bajo petición expresa de las dos anteriores. La secciones de Recepción y Transmisión están presentes en todas las capas del entorno, mientras que la sección de Registro sólo posee implementación en las Capa de Verificación y Sistema de Ficheros. La sección de Recepción se encarga de transmitir a la interfaz los paquetes que el verificador desea que se reciban en el dispositivo bajo verificación. La sección de Transmisión se encarga de recibir los paquetes que dispensa el dispositivo bajo verificación. Por último, la sección de Registro traza el funcionamiento de la interfaz.

A continuación se detallará el funcionamiento de las unidades funcionales contenidas en cada una de las capas.

Capa de Comunicación

Bajo Nivel

Esta capa contiene una única Unidad de Transmisión y una única Unidad de Recepción. La Unidad de Transmisión emite una palabra a la velocidad de trabajo de la interfaz. La información a transmitir la extrae de un *buffer* tipo FIFO que se alimenta desde la Capa de Verificación. La información a transmitir se compone de la palabra paralela y su correspondiente bit de paridad. La unidad de recepción toma la palabra y bit de paridad de las líneas de entrada de la interfaz paralela. Los datos recibidos se almacenan en un *buffer* FIFO de entrada.

Como ya se ha indicado anteriormente, estas unidades de comunicación trabajan simultáneamente. Además, esta capa provee la sincronización de la interfaz cuando no existen datos que transmitir. Por defecto, esta sincronización se inicia automáticamente una vez se ha iniciado la ejecución del entorno de verificación.

Capa de Verificación

Comprobaciones Básicas

Una vez el *buffer* de transmisión, localizado en la unidad de transmisión en la Capa de Comunicación, detecta un nivel mínimo de palabras a transmitir, la Unidad de Transmisión requiere al Conformador de Paquetes un nuevo paquete. El Conformador de Paquetes solicita al Ensamblador de Paquetes una nueva unidad de información. El Ensamblador de Paquetes obtiene las características del mismo mediante el Lector de Paquetes. El Ensamblador de Paquetes, una vez obtenidas las características de la unidad de información, monta el nuevo paquete a transmitir. A su vez, el Conformador de Paquetes aplica al paquete aquellas reglas definidas en su estructura como por ejemplo el CRC, paridades o codificación. Además, encapsula el paquete de información generado y conformado en el formato de la interfaz paralela empleada.

Por otro lado, cuando las palabras capturadas por la Capa de Comunicación de la interfaz alcanzan un nivel determinado, se ordena desde la Capa de Comunicación, el análisis de dichos datos recibidos. Dicho análisis se lleva a cabo en el Analizador de palabras. Éste determina si la palabra recibida es correcta o no.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

El analizador de palabras, también rastrea los datos recibidos con el objeto de obtener un paquete entrante completo. Una vez este analizador determina que se ha recibido un paquete, éste se pasa al Analizador de Paquetes con el objeto de compararlo con los especificados por el verificador, si procede.

Toda la interfaz de verificación se encuentra monitorizada y se realiza una traza de lo acaecido con diversas llamadas tanto al Registro de Traza, como al Registro de Paquetes. El Registro de Traza guarda la información de monitorización del sistema y el Registro de Paquetes almacena la información de los paquetes recibidos.

2.6.2. Interfaz Serie

La interfaz serie propuesta se muestra en la figura 2.6. La interfaz se ha dividido en cuatro secciones bien diferenciadas. Estas son: Transmisión, Recepción, Gestión y Registro. Transmisión, Recepción y Registro poseen el mismo propósito que las correspondientes secciones de igual nombre de la ya presentada interfaz paralela. La sección de Gestión tiene como propósito añadir la capacidad de administrar la utilización de la interfaz cuando se produce un conflicto de acceso.

Interfaz Asíncrono

Sea cual sea la interfaz de transmisión de datos, cuando esta se comparte con más de un elemento que accede a la misma, se produce un conflicto de recurso. Entonces, se aplica una política de solicitud y asignación de recurso. La interfaz implementada soporta tanto la recepción de las peticiones y asignación de recurso, como realizar peticiones y esperar asignación de recurso. Ello permite verificar tanto al recurso como a la entidad solicitante del mismo.

Al igual que la interfaz paralela, las diversas secciones que componen la interfaz serie se han implementado siguiendo la política de capas expuesta con anterioridad. Las secciones de Recepción y Transmisión poseen implementación en todos los niveles. Sin embargo, las secciones de Registro y Gestión poseen implementación en todas las capas salvo en la Capa de Comunicación.

Capa de Comunicación

Esta capa está compuesta por una Unidad de Transmisión y una Unidad de Recepción. Ambas unidades trabajan simultánea e independientemente. Además, tras el estado inicial de arranque del sistema bajo verificación, y una vez la interfaz comienza a operar, estos módulos dan soporte a los posibles protocolos de sincronización de la interfaz serie.

Bajo Nivel

Al igual que en la implementación de estas unidades en la interfaz paralela, también es necesaria una FIFO por unidad. Cuando llega un bit a la interfaz serie, éste se captura en la Unidad de Recepción. Este bit capturado es almacenado en la FIFO de la Unidad de Recepción. Al mismo tiempo, si existe al menos un bit en la FIFO de la Unidad de Transmisión, la Unidad de Transmisión extrae dicho bit y lo envía a la interfaz.

Cuando la FIFO de la Unidad de Transmisión desciende hasta un nivel mínimo, la Unidad de Transmisión informa a la Capa de Verificación para que inserte nuevos datos. Por otro lado, cuando el número de bits recibidos desde la interfaz alcanzan el tamaño de una palabra de la interfaz serie, la Unidad de Recepción advierte a la Capa de Verificación de este hecho.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

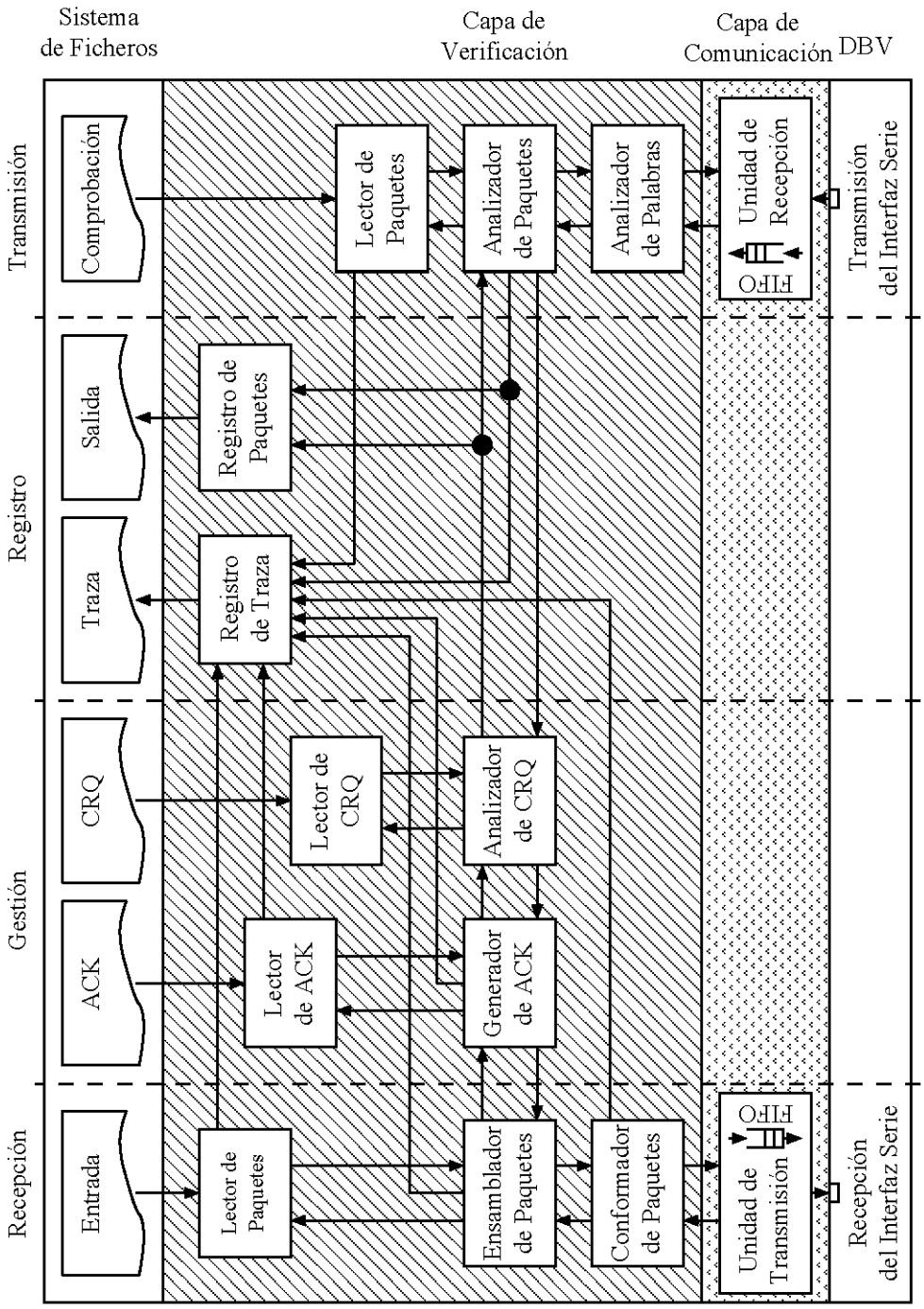


Figura 2.6: Diagrama de bloques de la interfaz serie.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

Capa de Verificación

Cuando se detecta un nivel bajo de bits en el *buffer* de transmisión, la Capa de Verificación hace uso del Lector de Paquetes para extraer las características del próximo elemento a enviar. Con dicha información, el Ensamblador de Paquetes construye una nueva unidad de información a enviar. Entonces, el paquete ensamblado se encapsula en el formato de nivel de enlace mediante el Conformador de Paquetes. Finalmente, se introduce en la FIFO de la Unidad de Transmisión.

Comprobaciones Básicas

Simultáneamente, cuando se recibe la advertencia de la posible existencia de una palabra serie en la FIFO de la Unidad de Recepción, el Analizador de Palabras comprueba la existencia o no de dicha palabra. En caso negativo, ordena a la Capa de Comunicación desalojar de la FIFO de la Unidad de Recepción los bits que procedan. En caso afirmativo, una vez se hallan recibido las palabras que conforman un paquete serie completo se procede a analizar la información que contiene mediante el Analizador de Palabras. Si existe un fichero de referencia para la comparación, se ordena al Lector de Paquetes que obtenga la información de referencia y se procede a comparar ambos paquetes en el Analizador de Paquetes.

Tanto en la sección de recepción como en la de transmisión se registran todos los procesos aplicados a los paquetes recibidos y emitidos. Para ello se invoca tanto al Registro de Traza como al Registro de Paquetes. Estos producen el Fichero de Traza y el Fichero de Paquetes de Salida.

La gestión de la interfaz como recurso se realiza mediante las llamadas petición de recurso o CRQ[12] y asignación de recurso o ACK[13]. Estos elementos se incorporan en los campos dispuestos, a tal efecto, dentro de los paquetes transmitidos y recibidos.

Por ejemplo, supóngase un sistema de colas y una matriz de conmutación de paquetes que posee una única interfaz de comunicación. Las diversas colas solicitan la conexión a la matriz de conmutación y esta última concede la interfaz disponible a una de las colas solicitante.

En el ejemplo, el recurso escaso es la interfaz que asignará la matriz de conmutación. Las diversas colas envían su petición en un CRQ, que la matriz de conmutación escucha. La matriz de conmutación hace uso de alguna política de asignación para resolver los posibles conflictos y envía un ACK a la cola que concede la interfaz.

En este escenario, si se desea verificar el sistema de colas, el entorno de verificación ha de ser capaz de recibir CRQ y enviar ACK. Si por el contrario, se desea verificar el funcionamiento de la matriz de conmutación, el entorno de verificación ha de ser capaz de enviar ACK y recibir CRQ.

Sistema de Petición y Asignación de Recursos

La figura 2.6 representa la configuración de la interfaz cuando el entorno de verificación comprueba el sistema de colas. Por ello, en la Capa de Verificación, el Analizador de Paquetes extrae la palabra de CRQ incluida dentro de cada paquete que se recibe, detectada la presencia de una petición tras la llegada de un nuevo paquete.

Recibido un nuevo CRQ, el Analizador de CRQ invoca al Lector de CRQ con el objeto de conocer si existe algún CRQ de referencia a comparar. Si existiese, cuando se comprueban las características, se toma la decisión de enviar o no un reconocimiento en el próximo paquete a enviar a la interfaz. La decisión tomada

[12] Acrónimo del término ingles *connection requests (CRQ)*.

[13] Acrónimo del término anglosajón *acknowlements (ACK)*.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

acata lo especificado en el fichero de CRQ. Si no se especifica el suceso en el fichero de CRQ, se toma un valor aleatorio que determina si se envía o no la aceptación a algún puerto.

Una vez el Analizador de CRQ toma la decisión de reconocer o no alguna solicitud, el Generador de ACK toma la correspondiente palabra de ACK del fichero de ACK, haciendo una llamada al Lector de ACK. Si existe dicha entrada, la palabra de ACK allí especificada se encapsula dentro del próximo paquete a enviar. Si por el contrario, no se especifica palabra de reconocimiento alguna; el puerto al que se le concede el reconocimiento, se selecciona aleatoriamente de entre los que originalmente solicitaron conexión con el CRQ recibido.

Como se puede apreciar en la figura 2.6 durante el normal funcionamiento de la interfaz, se crea en todo momento la traza de todos los eventos acaecidos, tanto a nivel de paquetes enviados y recibidos, como a nivel de operaciones aplicadas. Ello se realiza invocando al Registro de Traza y al Registro de Paquetes.

2.6.3. Interfaz de CPU

Configuración, Control y Gestión del DBV

La interfaz de CPU propuesta se muestra en la figura 2.7. También se ha empleado la metodología de implementación por capas en esta interfaz. A su vez, la interfaz se puede dividir en cinco secciones independientes: Configuración, Gestión, Interrupción, Registro y Direccionamiento.

La sección de Configuración toma el control de la interfaz de CPU para inicializar y configurar el mismo. La sección de Gestión se encuentra operando con la interfaz cuando esta se halla en su normal desarrollo. En caso de surgir una interrupción, la sección de Gestión cede el control a la sección de Interrupción donde se trata este tipo de evento. En todo momento, hay una sección de Direccionamiento que atiende a toda solicitud de acceso a dirección. Esta sección en particular se ocupa de comprobar y gestionar las direcciones de memoria de la CPU. Por último, cada sección genera una traza de ejecución con diversas llamadas a la sección de Registro.

A continuación se introduce el funcionamiento de cada una de las diversas capas de la interfaz.

Capa de Comunicación

Bajo Nivel

La Capa de Comunicación provee a la Capa de Verificación de las funciones de acceso a la CPU a bajo nivel. A diferencia de las interfaces paralela y serie ya presentados, esta capa está compuesta por una Unidad de Comunicación que puede ejecutar sobre la interfaz de CPU un ciclo de lectura, espera o escritura. Esta unidad provee el protocolo específico de cada interfaz de CPU.

Capa de Verificación

Comprobaciones Básicas

Mientras que la Capa de Comunicación posee una única unidad que realiza diversas tareas, la Capa de Verificación posee múltiples unidades. Inmediatamente después de comenzar la ejecución de la verificación toma el control de la interfaz el Gestor de Inicialización. Esta unidad recoge los datos de configuración del fichero dispuesto a tal efecto mediante el Lector de Configuración. Finalizada la inicialización, es decir, configurado el dispositivo a verificar, el Gestor de Inicialización cede el control al Gestor de CPU. Si el verificador ha especificado algún

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

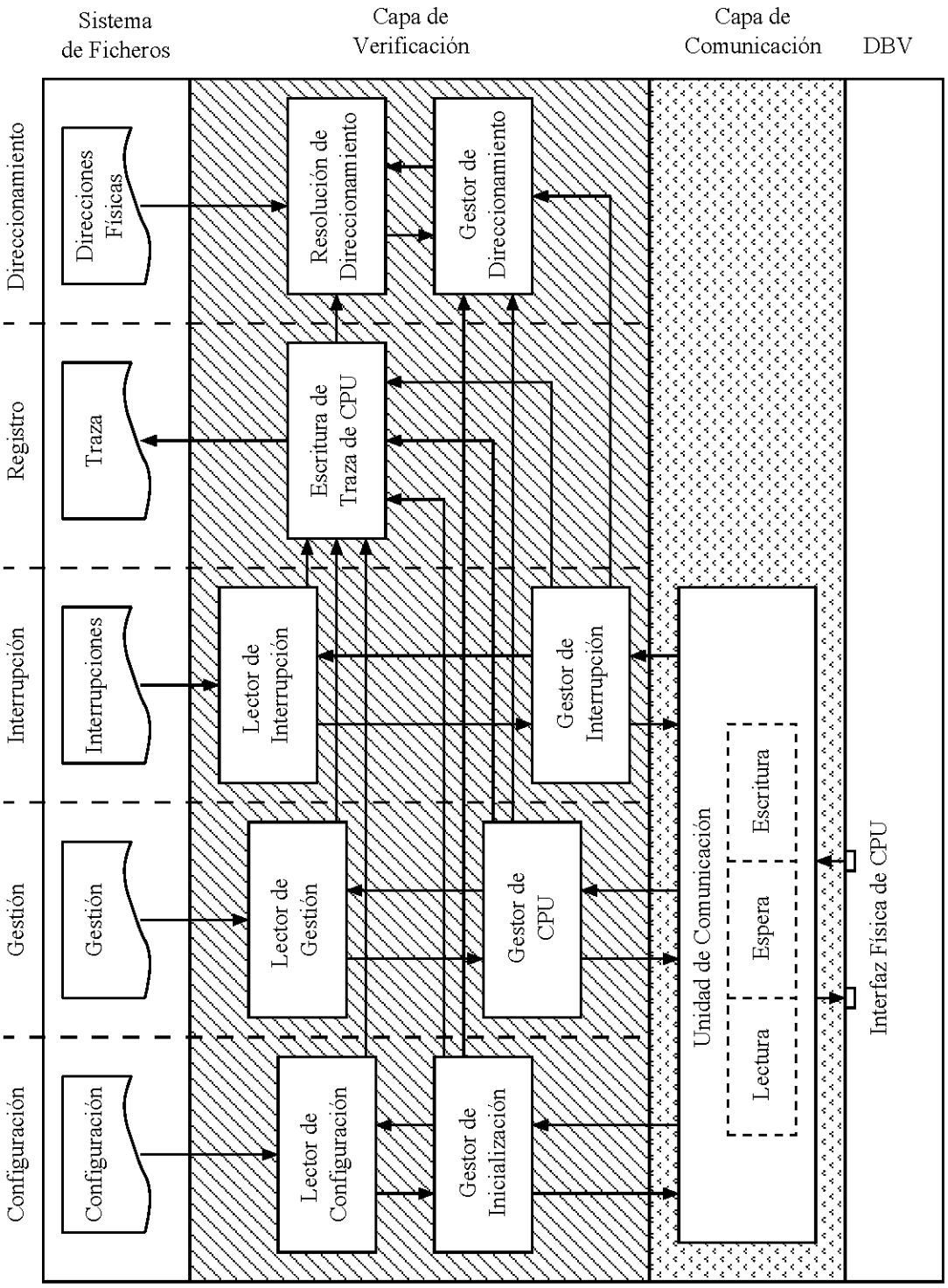


Figura 2.7: Diagrama de bloques de la interfaz de CPU.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

fichero de gestión, este es leído por el Lector de Gestión y ejecutado en el Gestor de CPU.

Si durante el normal funcionamiento de la interfaz surge una interrupción, el Gestor de CPU guarda el contexto de ejecución y procede a dar el control al Gestor de Interrupciones. Si se ha especificado un fichero de interrupción se procede a ejecutar el código del programa que allí se indique tras leerlo con el Lector de Interrupción.

Puesto que algunas interfaces de CPU proveen diferentes niveles de interrupción en su normal ejecución. Se ha dotado a la interfaz de CPU la posibilidad de trabajar en dos modos:

Modo Independiente.– Cada interrupción a atender se le asigna un fichero a ejecutar. Cuando surge una interrupción, el fichero asignado es ejecutado inmediatamente.

Modo No–Independiente.– Sólo existe un único fichero de interrupción que contiene diversas subrutinas. Cuando surge una interrupción es ejecutada la siguiente subrutina del fichero de interrupciones.

Direccionamiento

En la interfaz, cada vez que se hace uso de una dirección de memoria o registro, esta se comprueba con una simple llamada al Gestor de Direccionamiento. El Gestor de Direccionamiento también hace funciones de traductor de direcciones especificadas mediante etiquetas a direcciones físicas. Para ello se hace uso de la unidad de Resolución de Direcciones. Esta última unidad de la Capa de Verificación requiere la presencia del fichero de direccionamiento, el cual posee la relación de direcciones válidas del sistema acompañada cada una de ellas de su correspondiente etiqueta.

Para finalizar, en todo momento se traza la ejecución de la interfaz mediante llamadas a la unidad de Escritura de Traza de CPU.

2.7. El Sistema de Ficheros

Especificación de Estímulos y Respuesta

El sistema de ficheros se define para cada interfaz física como un conjunto de archivos de texto que permite al verificador describir los casos críticos, rápidamente, y mediante un lenguaje sencillo basado en comandos.

Los ficheros de salida contienen los paquetes salientes de cada interfaz. Los ficheros de entrada describen los estímulos de entrada. Y los ficheros de comprobación contienen paquetes de referencia a comprobar. Además, existen ficheros de traza y registro que permiten hacer seguimientos estadísticos y de depuración.

Los comandos que contienen dichos ficheros pueden comportarse de tres formas distintas según lo que dispongan sus argumentos; por defecto, condicional o incondicional:

- *Por Defecto*: define un comportamiento que se seguirá de forma genérica salvo que se especifique lo contrario. Por ejemplo, denegar todas las peticiones de conexión que se reciban hasta que se completen un número total de denegaciones.

- *Condicional*: el comando es ejecutado si se verifica una condición dada. Por ejemplo, la existencia de una petición determinada o CRQ especificado.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

Este modo permite guiar la verificación en aquellos casos que no se tiene la certeza exacta de la ocurrencia de un evento concreto. Por ejemplo, acepta las n primeras peticiones que procedan de un puerto cuya dirección sea par.

- *Incondicional*: es un comando que es ejecutado inmediatamente — sin atender a circunstancia alguna. Una vez éste es leído por la Capa de Verificación, se genera la trama de datos que se corresponde a lo especificado en el mismo y se introduce en el DBV.

2.7.1. Ficheros de la Interfaz de Enlace Serie

Los ficheros de estímulos de la interfaz de enlace serie están compuestos por los reconocimientos de conexión o ACK, las peticiones de conexión o CRQ, los paquetes de referencia o de comprobación y los paquetes de estímulos o de entrada. La tabla 2.1 presenta los comandos soportados en el sistema de ficheros serie. La primera columna de la tabla 2.1 indica el fichero específico donde se albergan los comandos. La segunda columna presenta los comandos soportados. Y para finalizar, la última columna realiza una pequeña descripción del objetivo del fichero.

Tabla 2.1: DESCRIPCIÓN DE LOS COMANDOS PERMITIDOS EN LA INTERFAZ SERIE

Fichero	Comando	Descripción
ACK	genACK	Aceptación de conexión† ‡ §
CRQ	genCRQ	Solicitud de conexión ‡ §
Estímulos y Comprobación	genSFrame ACKParity CRC OverHead Framming	Genera una trama de datos a enviar/comprobar † ‡ §
	PayLoad FlowControl Idle	Generales sobre el contenido de las tramas† §

† Comando que define un funcionamiento por defecto, ‡ condicional e § incondicional

Fichero ACK

En este fichero, para un enlace serie dado, se especifican los reconocimientos de conexión. El comando de Aceptación de una petición de conexión puede definir un estado por defecto para el comportamiento de la interfaz, condicionar la aceptación a una petición CRQ concreta o especificar en todo momento incondicionalmente qué conexión se concede o no.

Asignación de Recurso

Fichero CRQ

En este fichero se especifican los CRQ esperados en un enlace serie. Se permite especificar qué CRQ ha de aparecer en la próximo paquete a recibir. Este fichero no permite la inclusión de comandos generales.

Petición de Recurso

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

Fichero de Entrada y Fichero Comprobación

En estos ficheros se especifican las características de los paquetes que se enviarán o se comprobarán en la interfaz serie.

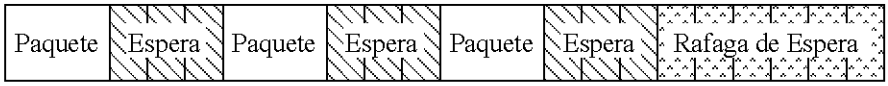


Figura 2.8: Generación de células de datos, relleno y ráfaga en la interfaz serie.

El comando de generación crea un conjunto de paquetes, dado una fuente y un puerto de destino, un tipo y una clase, una longitud de carga y una carga útil. Cuando no se genera paquete alguno se insertan por defecto paquetes de relleno o espera[14], estos son llamadas ráfagas de paquetes de relleno o espera. La figura 2.8 ilustra una ráfaga de paquetes de relleno generadas con el comando. Además tras una secuencia de paquetes de datos y de relleno, suele especificarse también la llamada ráfaga de paquetes de relleno.

También se definen una serie de comandos genéricos que permiten modificar diversas características implementadas en el interfaz como el control de flujo de control y de datos. Además existen comandos específicos que inyectan errores y alteran las características normales de las tramas como lo son los errores de CRC, paridad, *framming*, entre otras — para más detalle véase [35] y [33].

2.7.2. Interfaz Paralela

La interfaz paralela es más simple que la interfaz serie. Las entradas son el fichero de paquetes a enviar y el fichero de comprobación.

Fichero de Paquetes de Entrada y Comprobación

Los paquetes pueden ser especificados de tres formas: como una traza de paquetes, como porcentajes de los paquetes generados aleatoriamente o como porcentaje de trazas que han de ser generados aleatoriamente.

Generación de Estímulos

El comando de generación para la interfaz paralela — sigue la misma filosofía de generación que las tramas de la interfaz serie — crea un conjunto de paquetes, dado un puerto de destino, un tipo, una clase, una longitud de carga y la carga útil. Entre paquete y paquete especificado, se inserta el número de paquetes de relleno indicados. Al final del proceso de generación de los paquetes especificados se añadirán también el número de células de relleno que se especifica, este último conjunto de paquetes de relleno se llama ráfaga de paquetes de relleno al igual que en el puerto serie.

[14]Identificadas en la literatura como paquetes *idle*. Son paquetes que se emplean cuando hay carencia de datos que transmitir y se ha de mantener el sincronismo de la interfaz.

[35] J. Sosa, H. Navarro, M.V. Shahdadpuri, J.A. Montiel-Nelson, and R. Sarmiento. A Chip–Level Verification Methodology for Next–Generation Switch Fabric. In *XVII Conference on Design of Circuits and Integrated Systems*, November 2002.

[33] J. Sosa, J. A. Montiel-Nelson, H. Navarro, M. V. Shahdadpuri, and R. Sarmiento. System–Level Verification Methodology for Advanced Switch Fabrics. In *Proc. of SPIE of VLSI Circuits and Systems*, May 2003.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

Tabla 2.2: DESCRIPCIÓN DE LOS COMANDOS PERMITIDOS EN LA INTERFAZ PARALELA

Fichero	Comando	Descripción
Estímulos y Comprobación	genCFrame class type paylen Vparity Hparity	Genera una trama de datos a enviar/comprobar † ‡ §
	PayLoad FlowControl DataFlow Memory	Generales sobre el contenido de las tramas† §

† Comando que define un funcionamiento por defecto, ‡ condicional e § incondicional

La tabla 2.2 presenta el conjunto de comandos disponibles en el sistema de ficheros de la interfaz paralela. Al igual que los de la interfaz serie, hay comandos que permiten inyectar errores en la interfaz y comandos que permiten modificar las características del normal funcionamiento del mismo entre otros — para más detalle diríjase a la referencias [35] y [33].

2.7.3. Interfaz de CPU

Se provee un traductor de direcciones. Se define una tabla con las etiquetas de la CPU y sus direcciones físicas. El otro fichero de CPU posee tres comandos básicos; *cpuwr* para escribir en memoria, *cpurd* para leer de esta y *wait* para demorar la ejecución varios ciclos.

Ejecución de Rutinas de Servicio e Interrupción

Al comienzo de la verificación, el fichero de configuración de la CPU define la inicialización de los registros y memoria de CPU del DBV. Una vez se han inicializado dichos registros, el fichero de comprobación es ejecutado. Si surge una interrupción, el fichero de interrupción de CPU se ejecuta tal como se ha definido en el apartado 2.6.3. Cuando la rutina de servicio de la interrupción finaliza, la interfaz de CPU retorna a la sentencia previa del fichero de comprobación de CPU.

2.8. Características Avanzadas

La verificación de los sistemas actuales requieren de técnicas avanzadas no implementadas hasta ahora en los entornos de verificación tradicionales. A continuación se describen cuatro técnicas avanzadas en la verificación de sistemas integrados.

2.8.1. Inyección de Errores

En general es necesario no sólo comprobar que el sistema integrado bajo verificación se comporta correctamente en condiciones de ejecución normales. Es usual que los nuevos dispositivos se diseñen para soportar cierto conjunto de errores, lo cual induce la necesidad de insertar dicho conjunto de errores durante su

Inducir Comportamientos Erráticos

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

normal funcionamiento.

Todas las interfaces desarrolladas poseen diversos comandos que permiten inyectar errores en la propia interfaz con el objeto de comprobar si el sistema bajo verificación continua su ejecución adecuadamente o, por el contrario, su ejecución se altera. Por ejemplo, introducir errores en la paridad o el CRC de un paquete, acceder a posiciones de memoria o registros inexistentes, errar el tipo o longitud del paquete, son algunos de los errores utilizados más comúnmente.

2.8.2. Sincronización de Interfaces

Aplicación de Estímulos Simultáneos

Frecuentemente, se requiere que durante el normal procesamiento de los estímulos de entrada en dos o más interfaces exista una determinada sincronización. Por ejemplo, dos interfaces han de ser estimulados simultáneamente con un conjunto de vectores introducidos síncronamente a partir de un punto determinado de la verificación.

La figura 2.9 ilustra un ejemplo de las secuencias de estímulos de entrada en ambas interfaces. Se asume que, para la interfaz paralela 0, la secuencia a procesar antes de la sincronización es mayor que la secuencia a procesar en la interfaz paralela 1, y se requiere sincronizar los estímulos identificados como $\oplus$ y $\triangle$.

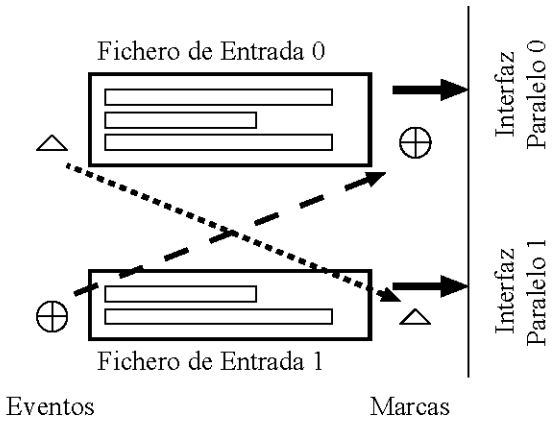


Figura 2.9: Sincronización de la interfaz paralela.

Para la interfaz paralela 0, el símbolo $\triangle$ es su evento de disparo en la sincronización. Y para la interfaz 1 lo es el símbolo $\oplus$. Además, el símbolo $\oplus$ es una marca de sincronización en la interfaz 0 y el símbolo $\triangle$ lo es para la interfaz 1. Obsérvese que para cada interfaz del ejemplo, los símbolos de marca y evento se encuentran en la misma línea de comandos.

Ambas interfaces paralelas procesarán cada uno de sus estímulos de entrada, independientemente. Este comportamiento se modifica cuando una interfaz lee una marca de sincronización. En ese caso, la interfaz se mantiene a la espera hasta que aparezca el evento de disparo correspondiente. Una vez el evento de disparo aparezca, la interfaz continuará su normal ejecución.

Igualmente, este sistema de sincronización ha sido adoptado entre las interfaces serie y CPU. Además, se ha provisto de un sistema idéntico para lograr la sincronización entre interfaces de distinta naturaleza, como por ejemplo serie–paralelo. Con la implementación realizada, se puede hacer aguardar a múltiples interfaces por un único evento que se producirá en otra interfaz.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

2.8.3. Jitter de Recepción Serie

Simulación de Errores de Sincronismo Serie

En general, toda interfaz serie asíncrona está sometida a la aparición de un retraso en el inicio de la comunicación. Con el objeto de dar soporte a este tipo de comportamiento, se ha implementado una unidad de gestión del *jitter*[15] [36]. En particular, aplicada a los puertos serie de alta velocidad empleados en los conmutadores de paquetes, se ha implementado una unidad con un retardo de *jitter* máximo de $\pm k$ golpes de reloj serie [37] y [38].

Gracias a esta unidad el entorno de verificación es capaz de adaptarse al posible *jitter* que aparecerá en su sección de recepción. Por otro lado, se ha implementado un generador de *jitter* en la etapa de transmisión, que permite comprobar si el dispositivo bajo verificación soporta sus especificaciones en lo referente al *jitter* admitido.

Estas unidades se incorporan en la Capa de Comunicación y se insertan entre la línea de recepción y la unidad de transmisión, y la línea de transmisión y la unidad de recepción. La figura 2.10 muestra la conexión para el caso de la línea de recepción del dispositivo.

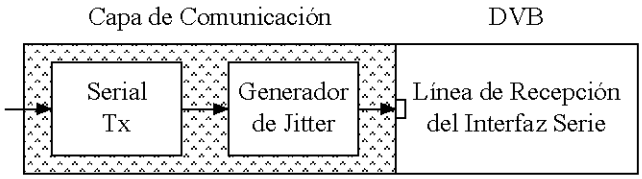


Figura 2.10: Detalle del módulo de generación de *jitter* en el diagrama de bloques de la interfaz serie.

2.8.4. Inserción de Nuevos Casos en Tiempo de Ejecución

Una vez ha sido iniciada la ejecución del entorno, éste no se para por si sólo. El usuario puede finalizarlo si lo interrumpe definitivamente o lo pausa momentáneamente desde el panel de control. El entorno permite cambiar los ficheros de verificación en tiempo de ejecución. Para hacer esto, el usuario ha de pausar momentáneamente la simulación y escribir los nuevos ficheros, los cuales serán ejecutados según se reanude la simulación. Cuando se reanude la simulación, el entorno sustituirá el contenido de los ficheros actuales por el contenido del nuevo.

Reducción del Tiempo de Inicialización

Esta característica permite evitar tener que inicializar los sistemas bajo verificación ya que, por lo general, es muy costoso; hay que configurar el sistema, sincronizar cada uno de las interfaces y, finalmente, situar el dispositivo en las

[15] Se define el *jitter* como: las pequeñas variaciones no acumulativas de los instantes significativos de una señal digital con relación a sus posiciones ideales en el tiempo.

[36] G.701. Vocabulary of Digital Transmission and Multiplexing, and Pulse Code Modulation (PCM) Terms. Technical report, International Telecommunication Union (ITU), March 1993.

[37] Whenwu Zhu, Yiwei Thomas Hou, Yao Wang, and Ya-Qin Zhang. End–to–End Modeling and Simulation of MPEG–2 Transport Streams over ATM Networks with Jitter. *IEEE Transactions on Circuits and Systems for Video Technology*, 8(1):9–12, February 1998.

[38] Yishay Mansour and Boaz Patt-Shamir. Jitter Control in QoS Networks. *IEEE Transactions on Networking*, 9(4):492–502, August 2001.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

condiciones iniciales de la prueba a la que se va a someter. Esta característica es necesaria en la mayoría de los casos, porque son pocos los simuladores que poseen función alguna de almacenamiento y carga del estado de la simulación (*snapshot*[16]).

2.9. Implementación Interna

Implementación Jerárquica

El entorno de verificación descrito en este trabajo se ha implementado mediante diversas librerías programadas en C++ y organizadas en cuatro niveles de jerarquía. Dichas librerías están ordenadas desde el nivel 0 al nivel 3. El nivel inferior ó 0 es el de menor abstracción y, por tanto, más cercano al nivel físico o simulador RTL. El nivel 3 o nivel superior, posee un alto nivel de abstracción que permite desarrollar la verificación en lenguajes de alto nivel. La figura 2.11 muestra el esquema de niveles descrito.

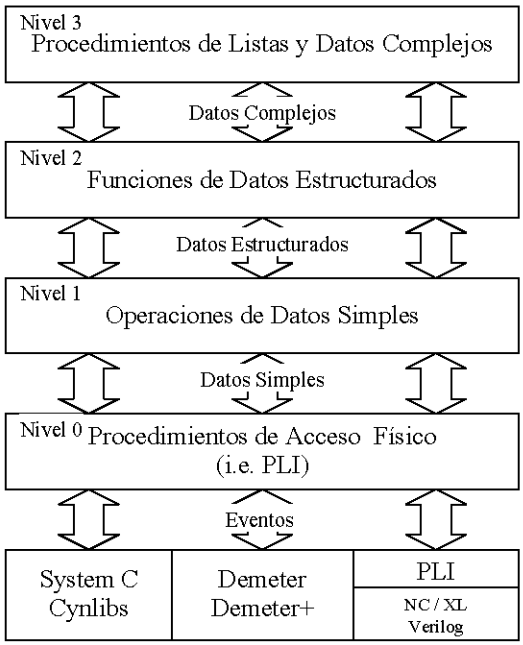


Figura 2.11: Esquema a nivel de bloques de las librerías implementadas.

2.9.1. Nivel 0

El principal propósito de este nivel es el de comunicar las diversas funciones de alto nivel descritas en C++ con el simulador HDL, convirtiendo las estructuras de datos y tipos de C++ a tipos soportados por el simulador empleado. El simulador podría ser SystemC, Demeter [21] o Cynlib que están basados en C++; o

Nivel Bajo

[16] Un *snapshot* se define como una captura de las variables del sistema en ejecución. Un *snapshot* permite retornar la simulación de un sistema al tiempo en el cual este *snapshot* se realizó, sin pasar por los estados anteriores que le dieron origen.

[21] H. Navarro, Juan A. Montiel-Nelson, J. Sosa, and R. Sarmiento. DEMETER: A Novel Framework for Hardware and Software System Specification, Simulation and Verification in C++. *XVI Conference on Design of Circuits and Integrated Systems*, pages 20–23, November 2001.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

podría ser un simulador comercial HDL como, por ejemplo, NC–Verilog, Verilog XL, o NC–VHDL. En caso de ser el simulador Verilog se emplea la Interfaz de Lenguaje de Programación PLI [39], [40] a la hora de implementar la pasarela HDL/C++.

2.9.2. Nivel 1

El propósito principal de esta capa es operar con las señales y manipular sus valores. Este nivel actúa como interfaz entre los tipos de datos físicos, i.e., cableado, registros, Booleanos, enteros y enteros largos[17]; y los tipos de datos de alto nivel del usuario del entorno. Los tipos de datos de usuario son aquellos datos que puede emplear el diseñador dentro del entorno. Los tipos de dato de usuario son: Booleano (bool_t), enteros con signo y sin signo (int_t y uint_t respectivamente), real (float_t) y complejo (complex_t).

Tipos Básicos de Alto Nivel

Las operaciones en este nivel dependen del ancho del bus y del modo de operación o alineamiento del tipo de dato empleado. El modo de operación o alineamiento se define en referencia a la representación bajo nivel del dato (modo más significativo[18], o modo menos significativo[19]). Por ejemplo, si el byte más significativo es el de mayor peso en un bus o, por el contrario, el byte de más peso en el bus es el menos significativo. Nótese que este nivel es transparente al usuario, aunque él puede redefinirlo, si así lo desea.

2.9.3. Nivel 2

En este nivel se definen los datos simples de usuario. En muchas aplicaciones se emplean cientos de bits de precisión; pero pocas de ellas necesitan miles o millones de bits. El objetivo es disponer un entorno con un número ilimitado de bits por bus. Así sería posible definir un bus que no estaría limitado por el sistema operativo donde se ejecuta el entorno. En este sentido, este entorno supera las funcionalidades provistas por otros entornos de desarrollo orientado a objetos, tal como Specman de Verisity Inc [20], que no permite declarar un entero que supere los 64 bits.

Tipos Básicos Extendidos de Alto Nivel

Para soslayar dicho escollo, se propone emplear las librerías aritméticas multiprecisión de GNU[20]. Estas son unas librerías portables escritas en C para aritmética entera arbitraria sobre enteros, números racionales y números en punto flotante. La librería GMP es la implementación más rápida, en tiempo de computación, posible para las aplicaciones que necesitan altas precisiones. En la actualidad la implementación de estas librerías se realiza directamente en el código nativo de cada máquina y soporta los tipos básicos de C.

Las librerías GMP poseen algoritmos que ajustan la precisión de las variables para alcanzar la precisión requerida, pero manteniendo mínimo la cantidad de

Librería Matemática de GNU

[17] En Verilog, este tipo se llama *time*.

[18] Identificado en la literatura anglosajona como *Big Endian Mode*(BEM).

[19] Traducción del término *Little Endian Mode(LEM)*.

[20] GNU Multiple Precision Arithmetic Library (GMP).

[39] IEEE 1364 Verilog Standard. *Verilog PLI 1.0*, November 2000.

[40] Stuart Sutherland. *The Verilog PLI Handbook*. Kluwer Academic Publishers, 1999.

[20] T. Kuhn, T. Oppold, M Witerholer, W. Rosenstiel, Marc Edwards, and Yaron Kashai. A Framework for Object Oriented Hardware Specification, Verification, and Synthesis. *IEEE Design Automation Conference*, pages 18–22, June 2001.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

Tabla 2.3: TIPOS DE DATOS Y OPERANDOS SOPORTADOS POR EL NIVEL 2

Tipo	Operadores Soportados	Descripción
boot_t	igual a, no igual y, o, o exclusiva, negación conversión desde/a entero de C++	Booleano
int_t	igual a, no igual, mayor que, menor que, mayor o igual, menor o igual suma, resta, multiplicación, división pre/posincremento, potencia, raíz enésima y, o, o exclusiva, negación conversión desde/a entero con/sin signo de C++	Entero con signo†
uint_t	Ídem que int_t	Entero sin signo†
float_t	igual a, no igual, mayor que, menor que, mayor o igual, menor o igual suma, resta, multiplicación, división, potencia raíz enésima, logaritmo, logaritmo natural arco/seno, arco/coseno, arco/tangente conversión desde/a entero con/sin signo de C++ conversión desde/a flotante/double de C++	Real‡
complex_t	Ídem que float_t más funciones hiperbólicas	Número complejo†

† Tipo de dato de longitud variable y configurable.

memoria requerida.

La velocidad de ejecución de las funciones en las librerías GMP se ha optimizado mediante el empleo de palabras completas de tipos básicos y su aritmética empleando algoritmos avanzados. Además se han optimizando dichos algoritmos hasta nivel de ensamblador. En este sentido, dichas librerías se han desarrollado con más énfasis en la optimización a nivel de ensamblador que en la claridad y simplicidad del código. Por ello, además de existir un código genérico que se puede compilar en cualquier máquina que posea gcc[21], hay versiones de código optimizada para la mayoría de los procesadores actuales como los AMD K6, Intel Pentium, IA–64 y SuperSPARC, entre otros [41]. Los tipos básicos de GMP son el mpz_t para la ejecución de aritmética entera y mpf_t para punto flotante.

Los tipos de datos estructurados implementados se muestran en la tabla 2.3. En la primera columna se presenta el nombre del tipo. En la segunda columna se introducen los operadores que soporta dicho tipo de datos estructurado. Por último, la tercera columna realiza una pequeña descripción de cada tipo. A continuación se esbozará someramente cada uno de los tipos, para más detalle diríjase

[21] *gcc es el acrónimo de GNU C Compiler, compilador C de GNU.*

[41] Torbjorn Franlund. The GNU Multiple Precision Arithmetic Library. *Free Software Foundation*, 2001.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

a la referencias [35] y [33].

Tipo Booleano: bool_t

Este es el único tipo de datos estructurado que no depende de la librería GMP. Puesto que posee siempre el ancho de un bit, i.e., verdadero (nivel lógico 1) o falso (valor lógico 0). Internamente contiene un tipo de datos de C++ *bool*.

Tipo Entero Con Signo: int_t

Este tipo de datos estructurado es un entero sin signo con la posibilidad de definir su ancho en bits. Cuando se declara un elemento de esta clase, el usuario ha de especificar el ancho y modo de operación de sus bits. Esto es, el ancho de palabra y si trabaja en modo BEM o LEM.

Este tipo de datos estructurado está compuesto por 3 elementos mpz_t de la librería GMP. Uno de ellos se emplea para mantener el valor del dato, y los otros dos contienen los valores máximo y mínimo permitidos para la variable, es decir, su rango.

Entero Con Signo

Tipo Entero Sin Signo: uint_t

Este tipo es un entero sin signo donde el usuario puede definir su ancho de palabra. Es idéntico al tipo int_t salvo la particularidad que este no contiene el bit de signo. El resto de información es idéntica.

Entero Sin Signo o Bus

Tipo Real: float_t

Este tipo de dato estructurado almacena un número real. Se basa en un mpf_t y dos int_t, uno para la mantisa y el otro para el exponente. Los anchos de palabra de la mantisa y del exponente son definibles por el usuario. Como en el caso del tipo int_t, el usuario debe especificar el ancho de palabra y su modo de funcionamiento a nivel bajo (LEM o BEM) si desea modificar los valores por defecto. El ancho de la mantisa y del exponente se pueden definir por separado.

Punto Flotante

Tipo Imaginario: complex_t

Este tipo de datos estructurado está compuesto por dos float_t. Uno de ellos para representar la parte imaginaria, y el otro para representar la parte real de un número complejo. La representación del número admite ser tratado ya no sólo en representación cartesiana[22], sino que soporta notación polar[23]. Al igual que en el resto de tipos, se puede definir tanto el ancho de cada número como el modo de funcionamiento.

[22] Sea X + Yi un número imaginario, se dice que su representación es cartesiana pues se puede expresar como una coordenada (X, Y) donde X se corresponde con la abscisa del vector imaginario e Y la ordenada del mismo.

[23] Número imaginario expresado como módulo y radio.

[35] J. Sosa, H. Navarro, M.V. Shahdadpuri, J.A. Montiel-Nelson, and R. Sarmiento. A Chip–Level Verification Methodology for Next–Generation Switch Fabric. In *XVII Conference on Design of Circuits and Integrated Systems*, November 2002.

[33] J. Sosa, J. A. Montiel-Nelson, H. Navarro, M. V. Shahdadpuri, and R. Sarmiento. System–Level Verification Methodology for Advanced Switch Fabrics. In *Proc. of SPIE of VLSI Circuits and Systems*, May 2003.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

2.9.4. Nivel 3

Acceso y Gestión de Memoria

Este nivel define básicamente un tipo de lista enlazada muy versátil. Esta lista es la clave del entorno de verificación descrito en este documento. La lista se encuentra optimizada tanto en velocidad de acceso como en memoria necesaria. Su implementación se corresponde con una lista enlazada simple. Además, con el objeto de acceder por índice, se ha incorporado un vector de índices a cada elemento. Esta modificación incrementa la memoria requerida por la lista simplemente enlazada a lo requisitos de memoria de una lista doblemente enlazada. Pero, al dar cobertura al acceso por índice, aumenta su versatilidad frente a una lista simple o doblemente enlazada.

La lista definida en este nivel no posee restricción alguna en cuanto al dato almacenado en cada elemento de la misma. Por otro lado esta lista tiene diversos métodos de acceso. Puede ser empleada como una pila de tipo LIFO[24], una FIFO[25], un vector indexado de memoria, una memoria circular.

El modo de funcionamiento de la lista se determina según el conjunto de métodos empleados. No por emplear un modo determinado, la lista queda fijada en dicho modo. Se pueden emplear los diversos modos de acceso y, por tanto, todos los métodos de acceso tanto en cuanto sean necesarias en su empleo. El primer elemento del vector de elementos se identifica con el índice 0 y el último elemento de la lista es en $n-1$. La figura 2.4 presenta los métodos disponibles en la lista implementada en este nivel 3.

Tabla 2.4: DESCRIPCIÓN DE LAS OPERACIONES PERMITIDAS SOBRE LISTAS

Nombre de la Función	Descripción
count()	Número de elementos
size()	Tamaño de la lista
swap(index,index)	Intercambia dos elementos
reverse()	Orden inverso de los elementos
isort()	Algoritmo de ordenación por inserción
qsort()	Algoritmo de ordenación rápida
push(item), add(item), pop(item) last(item), insert(item), append(item) insert(item,index)	Inserción de elementos desde el comienzo, el fin o en una posición concreta
remove(), remove(index) removeAll()	Eliminar un elemento o todos
for_each(function)	Ejecución de una función definida por el usuario en cada uno de los elementos de la lista
head(), tail(), first() last(), top(), bottom() next(), prev(), go(index) back(index)	Mover el puntero actual como una lista, una pila o un buffer
CircularGo(index), CircularBack(index) CircularNext(), CircularPrev()	Mover el puntero actual como si fuera una memoria circular
<list_name>[index]	Acceso a elemento por índice
push(list), add(list), insert(list) append(list), insert(list,index)	Inserción de elementos de una lista en otra

[24] Siglas de del término anglosajón *Last In Last Out* — último en entrar, último en salir.
[25] Acrónimo del término inglés *First in, First out* — primero en entrar, primero en salir.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

2.10. Resultados Experimentales

Con el objetivo de demostrar la utilidad de la metodología de verificación desarrollada tanto a nivel de circuito integrado como *cluster* o módulo, se ha aplicado en el desarrollo de varios circuitos comerciales y no comerciales [35], [42] y [43].

El principal circuito verificado, empleando el entorno de verificación, ha sido el VSC882 y el VSC872. El VSC882 es una matriz de conmutación que posee 16 enlaces de alta velocidad (2'125–2'644 Gb/s). El circuito VSC872 es un sistema de colas con 2 interfaces paralelas y 8 enlaces series de alta velocidad. El sistema de colas soportado por el VSC872 posee 256 FIFOs independientes, con soporte avanzado de células unicast, multicast y broadcast [44].

Conmutador de Paquetes

La verificación a nivel de circuito integrado de esos circuitos ha sido desarrollada para comprobar 19 casos básicos, en 6 modos diferentes de funcionamiento, con diversas variaciones del costo interno de conexión y otras características que soporta el sistema GIGAStream.

El empleo del modelo de alto nivel de ambos circuitos ha permitido al modo automático comprobar las prestaciones del circuito antes de obtener el primer prototipo. En la verificación realizada — incluidos, casos críticos y modo automático — se han conmutado con éxito más de 32 millones de células entre el VSC872 y el VSC882.

Es igualmente reseñable el desarrollo de casos específicos para comprobar las características de control de flujo de estos circuitos. Finalmente, el entorno ha permitido igualmente comprobar no sólo el correcto funcionamiento de dichos circuitos bajo las condiciones normales, sino que se han introducido múltiples paquetes con diversos errores con el objetivo de comprobar la conformidad de las interfaces tanto serie, paralela y de CPU.

Se ha desarrollado el entorno de verificación descrito con dos lenguajes de programación de alto nivel como es C++ y lenguaje–e, siguiendo una técnica a medida y la metodología propuesta en este trabajo. La figura 2.12 muestra una comparativa en términos de tiempo necesario, medido en días por hombre, frente a la cobertura alcanzada, en términos de porcentaje verificados, para realizar la verificación del VSC872. Se describieron 19 casos básicos, más seis modos funcionales distintos, con diversas variaciones del costo interno de conexión y otras características soportadas por el circuito bajo verificación. El número total de casos críticos fueron 456 en los cuales se han procesado, aproximadamente, dos millones de paquetes en el sistema de colas implementado en el VSC872. Las curvas presentadas en la figura 2.12 incluyen los tiempos debidos a la curva de aprendizaje, el desarrollo de la infraestructura de verificación y la codificación de

Implementación y Ejecución de la Verificación

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

[35] J. Sosa, H. Navarro, M.V. Shahdadpuri, J.A. Montiel-Nelson, and R. Sarmiento. A Chip–Level Verification Methodology for Next–Generation Switch Fabric. In *XVII Conference on Design of Circuits and Integrated Systems*, November 2002.

[42] J. Sosa, J. A. Montiel-Nelson, H. Navarro, José C. García, and R. Sarmiento. *Circuit Path Coverage using Genetic Algorithms*, pages 51–52. Evolutionary Methods for Design optimization and Control Applications to Industrial and Societal Problems. International Center for Numerical Methods in Engineering, 2003.

[43] H. Navarro, J. Sosa, J.A. Montiel-Nelson, and R. Sarmiento. A Real Time VLSI Bidimensional Pipelined Architecture for Image Processing. In *XVII Conference on Design of Circuits and Integrated Systems*, November 2002.

[44] Vitesse Semiconductor Corporation. Gigastream Chip Set VSC872 and VSC882. Vitesse Semiconductor Corporation, 2001.

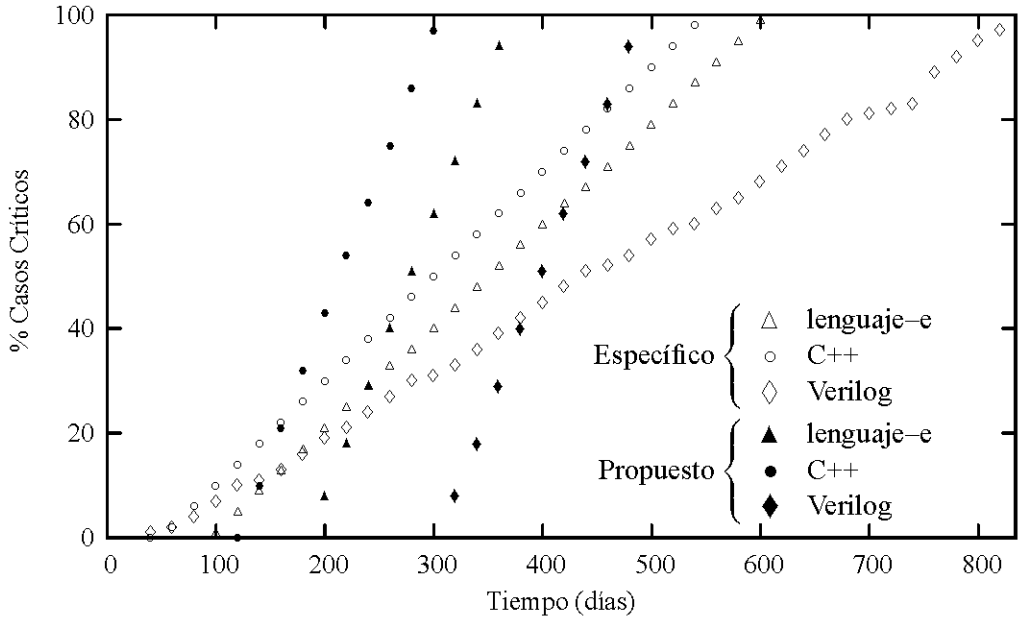


Figura 2.12: Comparativa de la evolución de la cobertura alcanzada en función del lenguaje empleado.

los casos críticos para un entorno basado en verificación de casos críticos.

La tabla 2.5 presenta con más exactitud los tiempos, medidos en días, de las curvas de desarrollo de la verificación planteada utilizando un desarrollo a medida y la metodología propuesta con C++, lenguaje–e y Verilog.

Tabla 2.5: COMPARATIVA DE LA METODOLOGÍA BASADA EN DESARROLLO A MEDIDA FRENTE A LA METODOLOGÍA PROPUESTA EN ESTE TRABAJO

Lenguaje	Metodología Propuesta			Desarrollo a Medida		
	C++	lenguaje–e	Verilog	C++	lenguaje–e	Verilog
Codificación del Entorno†	45	90	45	45	90	45
Infraestructura	70	90	255	0	0	158
Aprendizaje de Casos‡	5	5	5	0	0	0
Codificación Casos Críticos	185	185	185	504	515	631
Total	305	370	490	549	735	834
Mejora (%)	63'43	55'63	41'12	34'17	27'46	–

† Curva de aprendizaje del lenguaje escogido para codificar el entorno.

‡ Curva de aprendizaje del lenguaje de especificación de casos críticos.

Curva de Aprendizaje

Planteada la verificación, sea cual sea la metodología escogida, es necesario contabilizar el tiempo necesario para que los ingenieros que van a desarrollarla, se adapten al lenguaje escogido. O lo que es lo mismo, superen la curva de aprendizaje del lenguaje. La elección del lenguaje puede estar muy condicionada a factores subjetivos como el conocimiento previo del lenguaje y las reticencias a aprender uno nuevo. La mayoría de los verificadores hoy día conocen C++ y/o Verilog, lo que supone una corta curva de aprendizaje frente a lenguajes como lenguaje–e que no están tan extendidos. Una vez superada la fase de aprendizaje comienza la codificación del entorno de verificación para la metodología propuesta en este trabajo. La disponibilidad de diversas facilidades y librerías para desarrollar el entorno de verificación permite concentrar el esfuerzo de verificación en las tareas propias de esta. La carencia de dichas utilidades y/o librerías hace imprescindible su desarrollo antes de comenzar la implementación del entorno de verificación. En

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

ese sentido, el soporte de los lenguajes de alto nivel supera en gran medida a lenguajes HDL como Verilog o VHDL. Sin embargo, las técnicas que emplean desarrollos a medida presentan resultados al poco tiempo de comenzar la codificación de la verificación. La técnica de pruebas a medida codifica directamente los casos críticos, mientras que la metodología propuesta codifica un entorno que permite describir los casos críticos. Esa diferencia motiva que la técnica a medida ofrezca resultados a muy corto plazo. Según progresa la codificación de casos, estos se vuelven más complejos; y poco a poco la verificación basada en casos codificados a medida depende, más y más, de la reutilización de código y de las utilidades y facilidades del lenguaje empleado. Desarrollar un entorno permite maximizar la reutilización de la mayoría del código desarrollado, ya que la metodología fuerza su planificación y, por tanto, su estructuración. Esta última característica de la metodología propuesta permite desarrollar los casos complejos con mucho menos esfuerzo que las técnicas tradicionales a medida.

Codificación de la Verificación

La tabla 2.6 presenta un resumen de las principales características que posee la metodología propuesta y las técnicas de verificación a medida. De ellas reseñar la baja flexibilidad y reusabilidad de las técnicas a medida. A favor de estas, las curvas de aprendizaje son cortas. Pero, la depuración y refinado de la codificación de casos es ardua y tediosa. La dependencia de la codificación de casos a medida es alta, pues no se sigue regla alguna a la hora de implementarlos. En el entorno de verificación, la descripción por capas del mismo fuerza una estrategia organizada, que permite cambiar rápidamente el simulador HDL si se escoge un lenguaje de alto nivel como C++ o lenguaje–e. Por último, siempre es deseable que la persona que verifica el código RTL sea distinta a la persona que lo ha diseñado. Si se emplea una técnica a medida (*ad–hoc*), la relación entre el código HDL a verificar y el código de verificación es muy alta, lo cual dificulta la tarea de verificación. En este caso el verificador ha de conocer exactamente los detalles del código a verificar para implementar el código del caso crítico. El empleo de un lenguaje de alto nivel para describir el caso crítico permite abstraer al verificador de dichos detalles.

Estrategia de Verificación

Tabla 2.6: PRINCIPALES CARACTERÍSTICAS DE LA METODOLOGÍA PROPUESTA FRENTE A LAS TÉCNICAS A MEDIDA

Características	Técnicas a medida	Metodología propuesta
Reusabilidad	Baja	Alta
Flexibilidad	Baja	Alta
Curva de aprendizaje	Ninguna	Baja
Depuración del código de los casos críticos	Baja	Alta
Dependencia con el simulador	Alta	Baja
Dependencia del verificador frente al codificador	Alta	Baja

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

2.11. Conclusiones

Se ha presentado una metodología de verificación de circuitos de conmutación de paquetes [33], [35]. La metodología propuesta puede ser aplicada a cualquier módulo, *cluster* o CI a verificar. Esta metodología se basa en la descomposición en diversos niveles de abstracción, de cada interfaz del módulo, *cluster* o CI a verificar. Dicha descomposición permite la reconfiguración del entorno de verificación cuando es necesaria una actualización del mismo. Cada interfaz se descompone en los niveles de Comunicación, Verificación, Sistema de Ficheros y de Referencia.

La Capa de Comunicación permite enlazar los aspectos físicos de la implementación HDL — señales y niveles lógicos — a las descripciones algorítmicas equivalentes de los lenguajes de programación como C/C++ — eventos y variables. La capa de Verificación describe todas las utilidades y funcionalidades necesarias para gestionar la verificación basada en casos críticos. El Sistema de Ficheros permite especificar los casos críticos, comprobar los resultados y trazar el funcionamiento del entorno de verificación y del DBV. La Capa de Referencia en la verificación es la clave del modo automático, debido a que se puede reutilizar un modelo de alto nivel como referencia para comprobar la conformidad del DBV de forma automática, resolviendo las cuestiones referentes a la naturaleza divergente de los distintos simuladores.

Se han identificado y desarrollado diversas características novedosas como: la generación de vectores de máxima cobertura basada en la ejercitación de rutas o en la potencia de consumo, generación automática basada en un modelo de referencia, especificación de casos críticos en un lenguaje de muy alto nivel, e inserción de errores, que ofrecen un mayor control sobre el proceso de verificación [42]. Esto último redunda en el incremento de la cobertura alcanzada.

Para el desarrollo de los distintos niveles de las interfaces se proponen las funciones básicas: Transmisión, Recepción, Conformador/Analizador/Ensamblador de Paquetes, Analizador de Palabras, Generador/Analizador de ACK y/o CRQ, Registro de Traza, Registro/Lector de Paquetes. Estas funcionalidades se han desarrollado en C++ [31], como librerías del entorno de verificación. Se ha integrado Demeter [21] en dicho entorno [32].

[33] J. Sosa, J. A. Montiel-Nelson, H. Navarro, M. V. Shahdadpuri, and R. Sarmiento. System–Level Verification Methodology for Advanced Switch Fabrics. In *Proc. of SPIE of VLSI Circuits and Systems*, May 2003.

[35] J. Sosa, H. Navarro, M.V. Shahdadpuri, J.A. Montiel-Nelson, and R. Sarmiento. A Chip–Level Verification Methodology for Next–Generation Switch Fabric. In *XVII Conference on Design of Circuits and Integrated Systems*, November 2002.

[42] J. Sosa, J. A. Montiel-Nelson, H. Navarro, José C. García, and R. Sarmiento. *Circuit Path Coverage using Genetic Algorithms*, pages 51–52. Evolutionary Methods for Design optimization and Control Applications to Industrial and Societal Problems. International Center for Numerical Methods in Engineering, 2003.

[31] M. V. Shahdadpuri, J. Sosa, H. Navarro, Juan A. Montiel-Nelson, and R. Sarmiento. A New Framework for Hardware System Verification Based on C++. *XVI Conference on Design of Circuits and Integrated Systems*, pages 20–23, November 2002.

[21] H. Navarro, Juan A. Montiel-Nelson, J. Sosa, and R. Sarmiento. DEMETER: A Novel Framework for Hardware and Software System Specification, Simulation and Verification in C++. *XVI Conference on Design of Circuits and Integrated Systems*, pages 20–23, November 2001.

[32] M. V. Shahdadpuri, J. Sosa, H. Navarro, J.A. Montiel-Nelson, and R. Sarmiento. VESTA: A System Level Verification Environment Based on C++. In *Proc. of SPIE of VLSI Circuits and Systems*, May 2003.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

La estructura interna — descomposición de tipos y funciones — de las librerías de verificación facilita el desarrollo del entorno de verificación, sus métricas de cobertura, el lenguaje de especificación de alto nivel para casos específicos de verificación y la automatización de la verificación empleando un modelo de referencia. La descomposición presentada permite sustituir estas funcionalidades fácilmente cuando, por ejemplo, la librería de verificación se desarrolla en un lenguaje que las posee nativas. Se han desarrollado un conjunto de tipos de datos para la verificación, con sus modelos hardware y software transparentes. El empleo y extensión de dichos tipos, a datos complejos del verificador, también es transparente al verificador[31].

Se ha demostrado la efectividad de la metodología aplicada a tres lenguajes de los más empleados en la actualidad en la verificación de sistemas integrados — C/C++, Verilog y lenguaje–e. De ese estudio se desprende que al margen del tiempo de aprendizaje inicial, de las utilidades y funcionalidades naturales del lenguaje de verificación, aplicar la metodología propuesta reduce en más de un 50 % el tiempo requerido para completar el proceso de verificación cuando se selecciona C++ o lenguaje–e.

Tanto la metodología presentada como el entorno que la implementa han sido utilizados ampliamente en la verificación de circuitos integrados comerciales y no comerciales. La consecución del sistema GIGAStream es una buena garantía de que el entorno de verificación funciona correctamente y que la metodología propuesta es práctica en la verificación de circuitos de conmutación.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

Capítulo 3

Generación de Vectores de Máxima Cobertura

Índice General

3.1. Introducción 71

3.2. Definiciones de Teoría de Grafos 75

3.3. Cobertura Basada en Rutas (CBR) 76

3.3.1. Modelo de Propagación de Puerta 82

Puerta AND 82

Puerta OR 84

Puerta INV 85

3.3.2. Función Inversa del Modelo de Puerta 85

3.3.3. Obtención del Patrón de Entrada para una Ruta 87

3.3.4. Cálculo de la Cobertura para un Conjunto de Rutas . . 90

3.3.5. Búsqueda con Algoritmos Genéticos 91

Función de Coste 91

Codificación del Individuo Genético 91

Exploración de la Cobertura 91

Resultados Experimentales 93

Conclusiones 95

3.3.6. Satisfabilidad de la Propagación de Rutas 97

3.3.7. Modelo de propagación 97

Descripción del Circuito con MILP 98

Optimización mediante MILP 100

Resultados Experimentales 101

Mejora del Tiempo de Computación 105

Conclusiones 106

3.4. Cobertura Basada en Potencia de Consumo 107

3.4.1. Modelo de Potencia de Consumo CMOS 109

3.4.2. Función de Coste y Codificación 109

3.4.3. Resultados Experimentales 116

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

3.4.4. Conclusiones 118

Glosario de Términos

AND	Función lógica Y.
ATPG	Advanced Test Pattern Generation.
BDD	Binary Decision Diagram.
CBR	Cobertura Basada en Rutas.
$cond(v_j)$	Condición de propagación de la puerta v_j.
$fanin(v_j)$	Puertas conectadas como fuentes de señal a la puerta v_j.
$fanout(v_i)$	Puertas conectadas como carga a la puerta v_i.
$\vec{e}_j$	Arco que modela la conexión j entre dos puertas lógicas.
$\vec{G}$	Circuito combinacional expresado como grafo.
GLPK	GNU Linear Programming Kit.
$\vec{E}$	Conjunto de pares ordenados de vértices.
$\vec{e}$	Lado dirigido.
INV	Función lógica NO.
MILP	Mixed Integer Linear Programming.
NAND	Función lógica NO–Y.
NOR	Función lógica NO–O.
$\vec{OE}(v_l)$	Arcos de la puerta v salvo el arco l.
OR	Función lógica O.
$path_i$	Ruta i–ésima.
$patterns(cond(v_j))$	Patrones de entrada que cumplen las condiciones del nodo v_j.
$\mathcal{PC}(path_{v_i,v_j})$	Condiciones de la ruta $path_{v_i,v_j}$.
$patterns(path_{v_i,v_j})$	Patrones que permiten ejercitar la ruta $path_{v_i,v_j}$.
PI	Entradas primarias del circuito.
PO	Salidas primarias del circuito.
PROLOG	Programation et Logique, lenguaje de programación lógica.
RV	Conjunto de rutas a verificar.
RE	Rutas Ejecutadas.
τ_v	Retardo de vértice v.
V	Conjunto de vértices.
$v(path_i)$	Vértices de la ruta i–ésima.

En el capítulo anterior se presentó en detalle la infraestructura del entorno de verificación propuesto en esta Tesis Doctoral desde un punto de vista puramente descriptivo. Este capítulo centra su atención en la generación de vectores de entrada para la verificación. Se explica su importancia y la de las métricas de cobertura dentro del área de la verificación funcional. La generación de patrones de verificación se plantea desde un punto de vista objetivo, mediante el uso de varias métricas de cobertura.

Se propone el empleo de la métrica de Cobertura Basada en Rutas. La metodología propuesta para la generación de vectores de máxima cobertura con esta métrica emplea un modelo de puerta y circuital novel, que es también presentado en este capítulo. La implementación de la metodología de generación de vectores de máxima cobertura se realiza con ayuda de algoritmos de búsqueda tanto heurísticos como deterministas. La solución heurística se solventa mediante una implementación basada en Algoritmos Genéticos y la solución determinista se ha desarrollado empleado Programación Lineal Mixta Entera. También se propone una metodología eficiente para la generación de vectores con una métrica de cobertura basada en consumo de potencia. En este caso, la implementación de la metodología se lleva a cabo mediante una solución desarrollada con Algoritmos Genéticos.

Ambas metodologías de generación de vectores de máxima cobertura se prueban extensamente con el conjunto de circuitos MCNC'91 multinivel y dos niveles. Los resultados de las diversas comparativas demuestran que las soluciones adoptadas obtienen vectores de máxima cobertura de una forma eficiente — en términos

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

de tiempo de CPU — lo cual redunda en el objeto de esta Tesis Doctoral, es decir, reducir el tiempo de verificación. El capítulo está organizado como sigue.

En el apartado 3.1 se introduce la generación de vectores de máxima cobertura y su impacto en la etapa de verificación de cualquier sistema integrado. En segundo lugar, en el apartado 3.2 se presentan aspectos básicos de teoría de grafos como el concepto de vértice, camino y ruta, entre otros, necesarios para abordar adecuadamente la generación de vectores. Se continúa en el apartado 3.3 con la presentación formal de Cobertura Basada en Rutas. En los siguientes apartados se describe el modelo funcional directo e inverso de puerta, la obtención del patrón de entrada y un ejemplo de utilización del modelo para construir un patrón de verificación. A continuación, en el apartado 3.3.5 se describe la implementación heurística con Algoritmos Genéticos; su función de coste, codificación, experimentos y conclusiones. En el apartado 3.3.7 se expone la metodología propuesta e implementada con Programación Lineal Mixta Entera, el modelo lineal de puerta, el problema de optimización, los experimentos y las conclusiones vistos los resultados.

En el apartado 3.4 se presenta la Cobertura Basada en Potencia de Consumo. Y a continuación el modelo de potencia empleado, así como su codificación y optimización con Algoritmos Genéticos en detalle. El capítulo finaliza exponiendo las conclusiones de la metodología propuesta para la obtención de vectores de máxima cobertura.

3.1. Introducción

El principal objetivo de la verificación funcional es comprobar la conformidad de un diseño con respecto a su especificación [18]. Una de las claves importantes en la verificación es la métrica de cobertura [24]. Las métricas de cobertura intentan cuantificar, objetivamente, el grado de confianza[1] alcanzado en un proceso de verificación. En otras palabras, enumerado el conjunto de funcionalidades a verificar y sus correspondientes estímulos de entrada, a medida que se comprueba cada una de esas funcionalidades, o parte de ellas, la métrica de cobertura cuantifica el número de funcionalidades comprobadas frente al número total de funcionalidades a comprobar.

Conformidad con la Especificación

A pesar de que la métrica de cobertura se define para tener una medida objetiva sobre el proceso de verificación, la percepción del grado de confianza, sobre la verificación de un sistema, es totalmente subjetivo y dependiente, en la mayoría de los casos, de factores externos al flujo de diseño. En general estos factores externos son el *time–to–market* y la inclusión de funcionalidades o características avanzadas en diseños actuales para futuras versiones.

time–to–market

Puesto que el *time–to–market* acota el plazo máximo de entrega, también restringe el periodo de verificación de cualquier desarrollo. Esto implica que para

[1] El grado de confianza expresa la probabilidad de que un evento se produzca. Aplicado a la Verificación, indica la probabilidad de que el circuito no posea un fallo.

[18] Dhananjay S. Brahme, Steven Cox, Jim Gallo, Mark Glasser, William Grudmann, C. Norris Ip, William Paulsen, John L. Pierce, John Rose, Dean Shea, and Karl Whiting. Creating a C++ Library for Test Bench Authoring. Technical Report CDNL-TR-2000-0820, Cadence Berkeley Labs, August 2000.

[24] Hong Peng and Sofiene Tahar. Verification and Validation of Complex Digital Designs - A Practical Perspective. *IEEE VTS Tutorial*, June 2001.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

cumplir en tiempo y forma, se ha de ajustar el conjunto de funcionalidades a comprobar al tiempo definido para desarrollar el producto.

Influencia de la Métrica de Cobertura

En ese caso, la métrica de cobertura se definiría como el conjunto de funcionalidades comprobadas sobre el total de funciones verificables durante el *time–to–market*. Es evidentemente, que la cuantificación de las funcionalidades comprobadas es objetiva y permite estimar el grado de ejecución de las tareas de verificación. Sin embargo, un parámetro como el *time–to–market*, condiciona la medida de la cobertura. Dos procesos de verificación realizados sobre un mismo diseño, pero con distinto *time–to–market* poseen distinto grado de confianza. Y por tanto, ambos procesos de verificación no son comparables, a pesar de emplear la misma métrica de cobertura.

Funcionalidad Restringida

Una práctica muy extendida es la de desarrollar productos muy complejos, que son introducidos en el mercado cuando aún se encuentran en una fase temprana de verificación. Ello implica que no toda la funcionalidad del mismo ha sido comprobada. Lo que supone que las primeras revisiones del producto puesto en el mercado poseen una funcionalidad muy restringida, para el usuario final de ese producto inicial, con respecto al objetivo marcado para la versión definitiva del mismo. Esto es, la inclusión de funcionalidades ocultas o avanzadas es otro factor que distorsiona la objetividad de la métrica de cobertura. La inclusión de dichas funcionalidades dentro del proceso de verificación modifica el conjunto de requisitos a comprobar, y por ende distorsiona la métrica de cobertura.

Por otro lado, parece evidente que no existe una métrica genérica que se le pueda aplicar a todos los diseños. Por ello, y con el objetivo de alcanzar un grado de confianza determinado para un conjunto particular de estímulos de entrada se han propuesto múltiples métricas [45], [13], [46], [1] y [24].

La mayoría de las métricas de cobertura propuestas hasta el momento han sido planteadas desde el punto de vista puramente algorítmico. En general, partiendo de las técnicas existentes en la verificación de software, mediante pequeños refinamientos, estas han sido adaptadas al diseño electrónico. A nivel de puerta, la generación de estímulos se centra en obtener un conjunto de vectores de entrada empleando diagramas de decisión binaria (BDDs) [47], modelos de fallos [48], satisfabilidad [49] y generación automática de patrones de test (ATPG) [50].

[45] Tracy Larrabee. Test Pattern Generation using Boolean Satisfiability. *IEEE trans. on Computer–Aided Design of Integrated Circuits and Systems*, 11(1):4–15, January 1992.

[13] John R. Wallack and Ramaswami Dandapani. Coverage Metrics for Functional Test. *Proc. of the 12th IEEE VLSI Test Symposium*, pages 176–181, 1994.

[46] M. Kantrowitz and Lisa M. Noack. I'm Done Simulating; Now What? Verification Coverage Analysis and Correctness Checking of the DECchip 21164 Alpha Microprocessor. *Proc. of Design Automation Conference*, pages 325–330, June 1996.

[1] Sedar Tasiran and Kurt Keutzer. Coverage Metrics for Functional Validation of Hardware Designs. *IEEE Design and Test of Computers*, 18(4):36–45, August 2001.

[24] Hong Peng and Sofiene Tahar. Verification and Validation of Complex Digital Designs - A Practical Perspective. *IEEE VTS Tutorial*, June 2001.

[47] Randal E. Bryant. Graph–Based Algorithms for Boolean Function Manipulation. *IEEE transaction on Computers*, C-35(8):677–691, August 1986.

[48] M. Abramovici, M. Breumer, and A. Friedman. *Digital Systems Testing and Testable Design*. Wiley–IEEE Press, January 1993.

[49] E. Goldberg and Y. Novikov. BerkMin: a Fast and Robust Sat–Solver. *Proc. of Design, Automation and Test in Europe*, pages 149–149, March 2002.

[50] Tom Kirkland and M. Ray Mercer. Algorithms for Automatic Test Pattern Generation. *IEEE Design and Test of Computers*, June 1988.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

Métricas de Cobertura

Los BDDs permiten representar un circuito combinacional de forma óptima [47] — en términos de memoria y tiempo de computación. De hecho, la representación interna de la mayoría del software desarrollado en la actualidad emplea BDDs, para describir la funcionalidades de cada uno de los nodos que componen el grafo que representa a la red de lógica combinacional en estudio [51], [52], [53] y [15].

Los modelos de fallos permiten realizar una cuantificación de la medida en la verificación. Contabilizar el número de fallos que se pueden descubrir con un determinado vector de entrada, permite definir una cualidad numérica del vector, relativa a la capacidad de detectar errores — asignación objetiva de una propiedad a una magnitud o valor. Puesto que dicha cualidad numérica es objetiva por naturaleza, esta puede ser empleada como métrica de cobertura.

La satisfabilidad es un problema matemático, ampliamente estudiado [54]–[57], que plantea la búsqueda del conjunto de valores de entrada a una expresión Booleana que hacen que esta sea verdadera. Matemáticamente, y dentro del campo de los números reales, dicho problema está resuelto mediante diversos métodos como Programación Lineal [56], Programación Cuadrática [58], Algoritmos Genéticos [59], entre otros. La Programación Lineal es el método más empleado, pues garantiza la solución determinista al problema planteado en tiempo lineal con el número de variables que definen el problema a resolver. Como caso particular de la Programación Lineal, se define la Programación Lineal Mixta Entera

[51] S. Yang. *Logic Synthesis and Optimization Benchmarks User Guide version 3.0.* Microelectronics Center of North Carolina, 1991.

[52] E.M. Sentovich, K.J. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P.R. Stephan, Robert K. Brayton, and Alberto L. Sangiovanni-Vincentelli. SIS: A System for Sequential Circuit Synthesis. Technical Report UCB/ERL M92/41, EECS Department, University of California, Berkeley, 1992.

[53] R. K. Brayton, G. D. Hachtel, A. Sangiovanni-Vincentelli, F. Somenzi, A. Aziz, S. T. Cheng, S. Edwards, S. Khatri, Y. Kukimoto, A. Pardo, S. Qadeer, R. K. Ranjan, S. Sarwary, T. R. Shiple, G. Swamy, and T. Villa. VIS: a System for Verification and Synthesis. In Rajeev Alur and Thomas A. Henzinger, editors, *Proceedings of the Eighth International Conference on Computer Aided Verification CAV*, volume 1102, pages 428–432, New Brunswick, NJ, USA, 1996. Springer Verlag.

[15] Synopsys Inc. *Synopsys User Manual*, v1.0 edition.

[54] Dingzhu Zu, Hun Gu, and Panos M. Pardalos. *Satisfiability Problem: Theory and Applications.* American Mathematical Society, 1997.

[55] F. Fallah, S. Devadas, and K. Keutzer. Functional Vector Generation for HDL Models using Linear Programming and 3–Satisfiability. *Proc. on Design Automation Conference*, June 1998.

[56] Zeng Zhihong, P. Kalla, and M. Ciesielski. LPSAT: A Unified Approach to RTL Satisfiability. *Proc. on Design, Automation and Test in Europe*, pages 398–402, March 2001.

[57] J. A. Montiel-Nelson, H. Navarro, J. Sosa, and José C. García. Theory and Applications of Satisfiability Testing. In *Evolutionary and Deterministic Methods for Design, Optimization and Control with Applications to Industrial and Societal Problems*, November 2005.

[58] Srimat T. Chakradhar, Vishwani D. Agrawal, and Michael L. Bushnell. Automatic Test Generation using Quadratic 0–1 Programming. In *Proceedings of the 27th ACM/IEEE conference on Design automation*, pages 654–659, New York, NY, USA, 1990. ACM Press.

[59] André; Baresel, David Binkley, Mark Harman, and Bogdan Korel. Evolutionary Testing in the Presence of Loop-assigned Flags: A Testability Transformation Approach. In *ISSTA '04: Proceedings of the 2004 ACM SIGSOFT International Symposium on Software Testing and Analysis*, pages 108–118, New York, NY, USA, 2004. ACM Press.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

(MILP[2]), donde las variables de dicho problema se restringen a valores enteros en vez de reales. Este último problema aún continúa siendo un campo de investigación incipiente, pues no se ha encontrado un algoritmo o metodología que garantice una solución en un tiempo razonable.

Problema de Satisfabilidad

En el campo de la verificación de circuitos, se plantea el problema de encontrar el vector de entrada que permite asegurar un conjunto de condiciones, con el objeto de verificar algún requisito específico. Este problema se puede plantear en términos de Programación Lineal Mixta Entera, donde las variables del problema se restringen a valores Booleanos (uno y cero), para el caso de los circuitos combinacionales[3].

El empleo, de una o más de una, de las técnicas descritas en los párrafos anteriores para la generación de una secuencia de vectores requiere de una estrategia que guíe la generación de estímulos, con el fin de alcanzar el grado de confianza especificado. Las técnicas definidas como ATPG[4], a pesar ser ideadas inicialmente para la generación de vectores de test[5], y vista su similitud con la generación de vectores para la verificación, son aplicadas igualmente a las diversas etapas de verificación, con el fin de optimizar la generación de vectores y dotar de cierta inteligencia al sistema de verificación [55], [56] y [59].

Día a día, los diseños aumentan su complejidad debido a que se incluyen más y nuevas funcionalidades. Aumentar el número de funcionalidades implica incrementar el tiempo necesario para comprobar la integridad de las especificaciones en el diseño. Además, y debido a la creciente complejidad de las nuevas funcionalidades requeridas, se torna muy complicado el empleo de una única métrica de cobertura para todo el circuito. Por todo ello, es necesario plantear el dominio de definición de la métrica empleada, atendiendo exclusivamente a su utilidad y ámbito de aplicación. Ya no sólo es necesario aplicar una única métrica, sino que se hace imprescindible emplear más de una métrica con el fin de cubrir todos los aspectos relevantes de la verificación, y así cuantificar de forma adecuada el proceso de verificación. En otras palabras, si se desea cubrir ciertos aspectos singulares, pero fundamentales, durante el proceso de verificación, es necesario plantear métricas específicas que contabilicen esos aspectos particulares simultáneamente a las cuantificaciones globales de la verificación realizada.

Simultaneidad de Métricas

En particular, como métrica de cobertura para la verificación, la métrica de cobertura basada en rutas indica cuál es el grado de ejercitación del conjunto de

[2] Acrónimo del término inglés Mixed Integer Lineal Programming, traducido al castellano como Programación Lineal Mixta Entera.

[3] Todo circuito combinacional puede ser visto como una red Booleana donde el ancho de palabra es el bit y no el ancho de palabra definido en alto nivel.

[4] Acrónimo anglosajón Advanced Test Pattern Generation, traducido como Generación Avanzada de Patrones de Test.

[5] Se define el test como al proceso de comprobación de defectos físicos en la fabricación de Circuitos Integrados.

[55] F. Fallah, S. Devadas, and K. Keutzer. Functional Vector Generation for HDL Models using Linear Programming and 3–Satisfiability. *Proc. on Design Automation Conference*, June 1998.

[56] Zeng Zhihong, P. Kalla, and M. Ciesielski. LPSAT: A Unified Approach to RTL Satisfiability. *Proc. on Design, Automation and Test in Europe*, pages 398–402, March 2001.

[59] André; Baresel, David Binkley, Mark Harman, and Bogdan Korel. Evolutionary Testing in the Presence of Loop-assigned Flags: A Testability Transformation Approach. In *ISSTA '04: Proceedings of the 2004 ACM SIGSOFT International Symposium on Software Testing and Analysis*, pages 108–118, New York, NY, USA, 2004. ACM Press.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

rutas a verificar [13] y [60]. Se entiende como ejercitación, a la propagación de una transición alto–bajo o bajo–alto a través de las puertas que componen la ruta ejercitada. Si se aplica verificación basada en simulación, el grado de confianza alcanzado con esta métrica de cobertura es mayor cuantos más vectores se ejecuten o introduzcan en el sistema bajo verificación. Por ello, siempre es deseable obtener el conjunto mínimo de vectores que permiten maximizar la cobertura en la verificación de una funcionalidad.

Antes de proseguir, en el próximo apartado, se introducirán los conceptos necesarios de teoría de grafos para poder abordar, adecuadamente, el problema de generación de vectores de máxima cobertura empleando la métrica CBR.

3.2. Definiciones de Teoría de Grafos

Un circuito de lógica combinacional es una red Booleana representada por un multigrafo, finito, conectado mediante arcos dirigidos, y con un costo definido en sus arcos $\vec{G} = G(V, \vec{E}, \tau)$, tal y como se describe en [61]. V es un conjunto de vértices y $\vec{E}$ es un conjunto de pares ordenados de vértices distintos, llamados arcos dirigidos[6]. Un arco dirigido $\vec{e} = (u, v)$ incide en u y v, los vértices son la *cabeza* y la *cola* de $\vec{e}$, respectivamente; $\vec{e}$ es un arco de entrada de v y un *arco de salida* de u.

Salida y Entrada Inmediata

La existencia de un *arco dirigido* $\vec{e} = (u, v) \in \vec{G}$ implica que el nodo u es una *entrada inmediata* del nodo v, o el nodo v es una *salida inmediata* del nodo u, tal y como es presentado en [62]. El conjunto de todas las entradas inmediatas de v se denota por $fanin(v)$, y al conjunto compuesto por todas las salidas inmediatas de u se denomina $fanout(u)$.

Retardo de Puerta

En [63] se presenta una definición para el término multigrafo con costo asociado a sus arcos, aplicado al modelado de dispositivos computacionales. Los vértices del grafo o nodos modelan las puertas lógicas. Cada vértice v posee un costo de retardo no negativo, esto es $\tau(v) = \tau_v$. Los arcos dirigidos E del grafo modelan las interconexiones entre puertas lógicas.

Un camino[7] de $\vec{G}$ es una secuencia de vértices v_i y arcos e_j, que se expresa como $v_0 \xrightarrow{e_0} v_1 \xrightarrow{e_1} \ldots \xrightarrow{e_{k-1}} v_k$, tal que, e_i conecta los vértices v_i y v_{i+1}. Un tour es un camino donde todos los arcos son distintos. Una ruta es un camino con distintos

[6] Traducción directa del término anglosajón *directed edges*.

[7] Traducción del término anglosajón *walk*.

[13] John R. Wallack and Ramaswami Dandapani. Coverage Metrics for Functional Test. *Proc. of the 12th IEEE VLSI Test Symposium*, pages 176–181, 1994.

[60] J. Sosa, J. A. Montiel-Nelson, H. Navarro, and José C. García. Functional Vector Generation for Maximum Data Path Coverage using Mixed Integer Linear Programming. In *XVIII Conference on Design of Circuits and Integrated Systems*, November 2005.

[61] K.S. Lowe and P.G. Gulak. A Joint Gate Sizing and Buffer Insertion Method of Optimizing Delay and Power in CMOS and BiCMOS Combinational Logic. *IEEE Trans. Comput.–Aided Des. Integr. Circuits Syst.*, 17(5):419–434, 1998.

[62] C. Chen, A. Srivastava, and M. Sarrafzadeh. On Gate Level Power Optimization using Dual–Supply Voltages. *IEEE Trans. Comput.–Aided Des. Integr. Circuits Syst.*, 9(5):616–629, 2001.

[63] P.Y. Calland, A. Mignotte, O. Peyran, Y. Robert, and F. Vivien. Retiming DAG's. *IEEE Trans. Comput.–Aided Des. Integr. Circuits Syst.*, 17(12):1319–1324, 1998.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

vértices [64].

Ruta

Así, una ruta que conecta los nodos v_0 y v_k se define en [63] y [65] como $path_{v_0,v_k} = v_0 \stackrel{e_0}{\rightarrow} v_1 \stackrel{e_1}{\rightarrow} \ldots \stackrel{e_{k-1}}{\rightarrow} v_k$; donde la cola de la ruta $t(path_{v_0,v_k})$ es v_0, el primer arco de la cola $\vec{e}_0$; La cabeza de la ruta $h(path_{v_0,v_k})$ es v_k, el último arco de la cabeza $\vec{e}_{k-1}$; y $v(path_{v_0,v_k})$ es $\{v_0, v_1, \ldots, v_k\}$, el conjunto de vértices. Por otra parte, este trabajo centra su interés en aquellas rutas donde la cola de la ruta v_0 y su cabeza v_k son entrada y salida primaria al circuito respectivamente. El conjunto vértices entradas y salidas primarias de $\vec{G}$ es $PI(\vec{G})$ y $PO(\vec{G})$, respectivamente.

3.3. Cobertura Basada en Rutas (CBR)

En todo momento, dentro del proceso de verificación, cuantificar la calidad de los patrones de entrada y, por tanto, la cobertura lograda con ellos, es un objetivo primordial a alcanzar. Una medida de la calidad de vectores empleados en la verificación funcional es la Cobertura Basada en Rutas (CBR) [13]. La CBR puede ser definida formalmente como:

$$CBR = \frac{RE}{RV}, \tag{3.1}$$

donde RE representa el número de Rutas Ejercitadas y RV el número de Rutas Verificables.

Esta métrica se basa en el estudio de la ejercitación de las diversas rutas que conforman un circuito de lógica combinacional a verificar. En este contexto, verificar o ejercitar una ruta consiste en estimular el circuito de lógica combinacional, de forma que, permita la propagación de una transición desde una entrada, hasta una salida, a través de las puertas lógicas e interconexionado que componen la ruta ejercitada.

Ruta Ejercitable

Sea una ruta $path_{v_i,v_j}$, a ser ejercitada. Cada una de las puertas lógicas que la componen $v(path_{v_i,v_j})$ tienen que permitir la propagación de una transición desde la entrada primaria v_i hasta la salida primaria v_j. Por lo tanto, el resto de entradas de cada puerta lógica que pertenecen al $path_{v_i,v_j}$ han de poseer los valores lógicos que permita dicha propagación.

Definición 1 (Ruta ejercitable). *Sea $\vec{G}$ el grafo de un circuito combinacional y $path_{u,v}$ una de sus rutas. Esta ruta $path_{u,v}$ es ejercitable si existe un vector de entrada a $\vec{G}$ que permite propagar una transición desde la entrada primaria u a la salida primaria v a través de cada uno de los vértices de la ruta $path_{u,v}$.*

Definición 2 (Condiciones de propagación de una puerta). *Sea v_k un vértice de la ruta ejercitable $path_{v_h,v_l}$. Las condiciones de propagación $cond(v_k, path_{v_h,v_l})$ del vértice v_k para la ruta $path_{v_h,v_l}$ son el conjunto posible de valores lógicos que ha de poseer cada uno de los vértices de entrada $v_i \mid v_i \in fanin(v_k)$ &*

[64] N. Sherwani. *Algorithms for VLSI Physical Design Automation*. Kluwer Academic Publishers, Boston, 1993.

[63] P.Y. Calland, A. Mignotte, O. Peyran, Y. Robert, and F. Vivien. Retiming DAG's. *IEEE Trans. Comput.–Aided Des. Integr. Circuits Syst.*, 17(12):1319–1324, 1998.

[65] C.E. Leiserson and J.B. Saxe. Retiming Synchronous Circuitry. *Algorithmica*, 6:5–35, 1991.

[13] John R. Wallack and Ramaswami Dandapani. Coverage Metrics for Functional Test. *Proc. of the 12th IEEE VLSI Test Symposium*, pages 176–181, 1994.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

$v_i \notin v(path_{v_h,v_l})$, *tal que permiten propagar una transición a* $fanout(v_k)$, *desde el vértice de entrada* $v_j \mid v_j \in fanin(v_k)\ \&\ v_j \in v(path_{v_h,v_l})$.

Condiciones de Contorno

El conjunto de condiciones que imponen cada uno de los vértices de una ruta se agrupa en las llamadas condiciones de contorno, o condiciones de propagación de transiciones de la ruta $\mathcal{PC}(path_{v_i,v_j}) = \{cond(v_{i+1}, path_{v_i,v_j}), ..., cond(v_{j-1}, path_{v_i,v_j})\}$. Si no es posible satisfacer cada una de esas premisas individuales, entonces, esa ruta no puede ser ejercitada.

Sea el circuito de lógica combinacional que se presenta en la figura 3.1(a), la figura 3.1(b) muestra su grafo $\vec{G}$. Para abordar de forma didáctica el ejemplo, que a continuación se va a plantear, es necesario enumerar todas y cada una de las rutas existentes entre las entradas primarias y salidas primarias del circuito de la figura 3.1(a). La tabla 3.1 resume todas las rutas posibles del circuito. La primera columna muestra la etiqueta de cada ruta. La segunda columna presenta la entrada primaria, puertas y salida primaria que componen la ruta a verificar. La tercera columna presenta la ruta a través del grafo. Y, la última columna presenta las condiciones de contorno necesarias para verificar la ruta en estudio.

Según la métrica de cobertura de rutas, para alcanzar la totalidad del nivel de confianza es necesario ejercitar cada una de las rutas enumeradas en la tabla 3.1.

Tabla 3.1: RUTAS Y CONDICIONES PARA LA COBERTURA BASADA EN RUTAS DEL CIRCUITO DEL EJEMPLO

Nombre	Ruta del Circuito	Ruta en el Grafo	Condiciones
$path_1$	A–I_1–I_2–O	$v_1 \xrightarrow{e_1} v_7 \xrightarrow{e_7} v_{11} \xrightarrow{e_{14}} v_{17}$	$e_2 = 0$
$path_2$	A–I_1–I_3–I_4–P	$v_1 \xrightarrow{e_1} v_7 \xrightarrow{e_8} v_{12} \xrightarrow{e_{15}} v_{14} \xrightarrow{e_{18}} v_{18}$	$e_2 = e_9 = 0$
$path_3$	B–I_1–I_2–O	$v_2 \xrightarrow{e_2} v_7 \xrightarrow{e_7} v_{11} \xrightarrow{e_{14}} v_{17}$	$e_1 = 0$
$path_4$	B–I_1–I_3–I_4–P	$v_2 \xrightarrow{e_2} v_7 \xrightarrow{e_8} v_{12} \xrightarrow{e_{15}} v_{14} \xrightarrow{e_{18}} v_{18}$	$e_1 = e_9 = 0$
$path_5$	C–I_5–I_3–I_4–P	$v_3 \xrightarrow{e_3} v_8 \xrightarrow{e_9} v_{12} \xrightarrow{e_{15}} v_{14} \xrightarrow{e_{18}} v_{18}$	$e_4 = e_8 = 0$
$path_6$	C–I_5–I_6–Q	$v_3 \xrightarrow{e_3} v_8 \xrightarrow{e_{10}} v_{13} \xrightarrow{e_{16}} v_{19}$	$e_4 = e_{11} = 0$
$path_7$	C–I_5–I_6–I_8–R	$v_3 \xrightarrow{e_3} v_8 \xrightarrow{e_{10}} v_{13} \xrightarrow{e_{17}} v_{15} \xrightarrow{e_{19}} v_{20}$	$e_4 = e_{11} = e_{12} = 0$
$path_8$	D–I_5–I_3–I_4–P	$v_4 \xrightarrow{e_4} v_8 \xrightarrow{e_9} v_{12} \xrightarrow{e_{15}} v_{14} \xrightarrow{e_{18}} v_{18}$	$e_3 = e_8 = 0$
$path_9$	D–I_5–I_6–Q	$v_4 \xrightarrow{e_4} v_8 \xrightarrow{e_{10}} v_{13} \xrightarrow{e_{16}} v_{19}$	$e_3 = e_{11} = 0$
$path_{10}$	D–I_5–I_6–I_8–R	$v_4 \xrightarrow{e_4} v_8 \xrightarrow{e_{10}} v_{13} \xrightarrow{e_{17}} v_{15} \xrightarrow{e_{19}} v_{20}$	$e_3 = e_{11} = I_{12} = 0$
$path_{11}$	E–I_7–I_6–Q	$v_5 \xrightarrow{e_5} v_9 \xrightarrow{e_{11}} v_{13} \xrightarrow{e_{16}} v_{19}$	$e_{10} = 0$
$path_{12}$	E–I_7–I_6–I_8–R	$v_5 \xrightarrow{e_5} v_9 \xrightarrow{e_{11}} v_{13} \xrightarrow{e_{17}} v_{15} \xrightarrow{e_{19}} v_{20}$	$e_{10} = e_{12} = 0$
$path_{13}$	F–I_9–I_8–R	$v_6 \xrightarrow{e_6} v_{10} \xrightarrow{e_{12}} v_{15} \xrightarrow{e_{19}} v_{20}$	$e_{17} = 0$
$path_{14}$	F–I_9–I_{10}–S	$v_6 \xrightarrow{e_6} v_{10} \xrightarrow{e_{13}} v_{16} \xrightarrow{e_{20}} v_{21}$	—

Del circuito representado en la figura 3.1(a) con ayuda de la tabla 3.1 se puede deducir fácilmente que, por ejemplo, para verificar la ruta etiquetada como $path_4$, se debe asegurar que en los arcos e_1 y e_9 ha de existir un cero lógico. De otra forma, este $path_4$ no puede ser ejercitado. La figura 3.1(c) muestra la ejercitación de la ruta y las condiciones sobre la red Booleana para este caso.

Prosiguiendo con el ejemplo anterior, asegurar que el arco e_1 posea un cero lógico (0) implica que la entrada primaria v_1 también se encuentre a nivel lógico cero. Y por ende, el vector de entrada que permite verificar la ruta $path_4$ ha de poseer el bit que representa al terminal de entrada v_1 con un cero lógico. Sin embargo, para el caso del arco e_9, la necesidad de imponer un cero lógico se puede satisfacer de formas diversas. Ello implica que las condiciones de contorno particulares de cada puerta, que va a permitir la propagación de una transición,

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

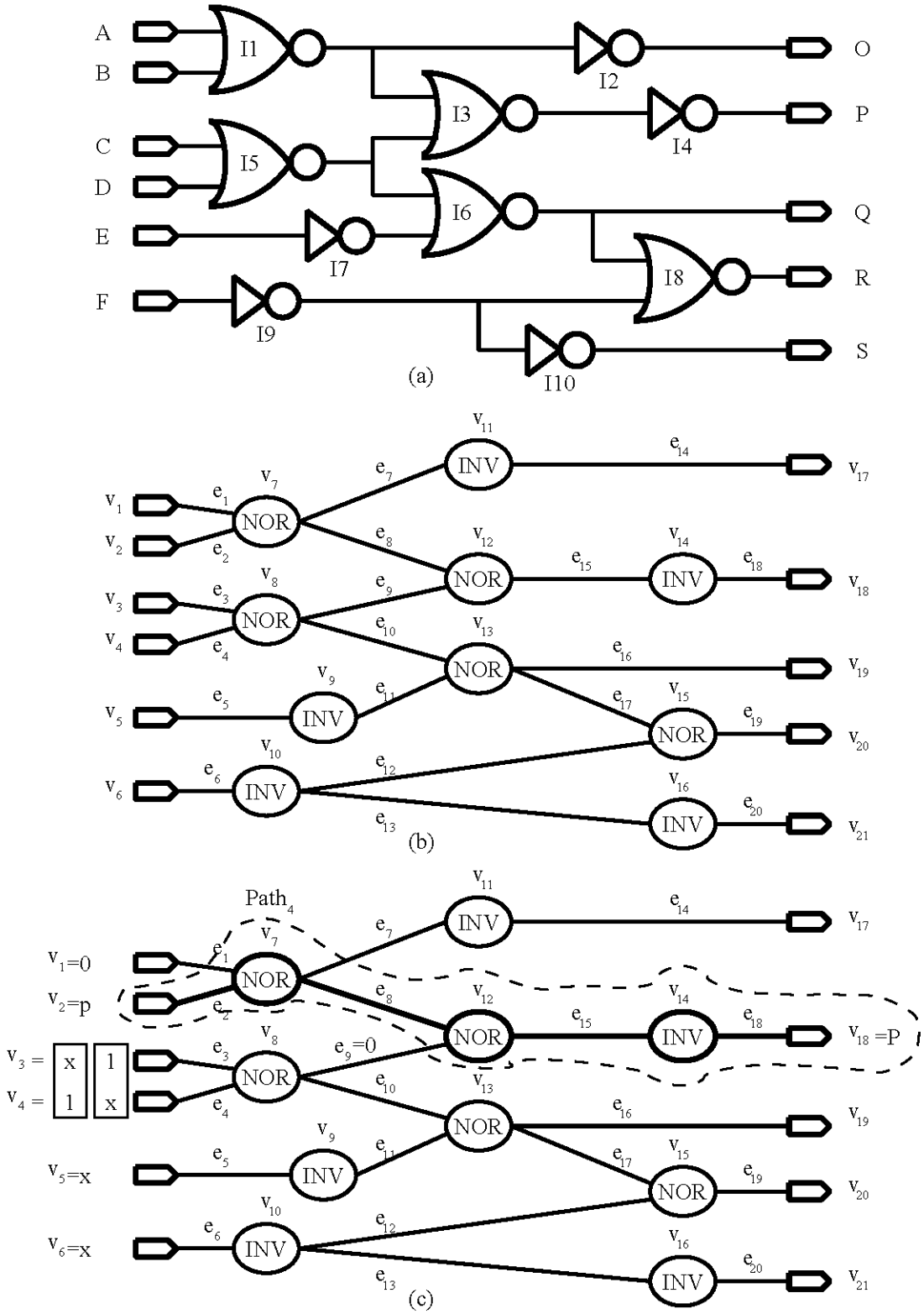


Figura 3.1: Ejemplo de cobertura basada en rutas: (a) circuito combinacional, (b) grafo del circuito, (c) ejercitación del $path_4$.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

se pueden satisfacer con múltiples vectores de entrada o lo que es lo mismo, en función de cada red combinacional se define un grado de libertad que permite seleccionar de entre varios patrones.

A modo de resumen, una condición de contorno impuesta por una puerta que ha de propagar una transición, define un estado lógico sobre las puertas de la red Booleana que la contiene, que denominaremos valor lógico de la red Booleana. Para cumplir dicho valor y ejercitar la ruta seleccionada pueden existir diversos patrones de entrada. Formalmente, se podría decir que para una ruta $path_{v_i,v_j}$, sus vértices son $v(path_{v_i,v_j}) = \{v_i, ..., v_j\}$. Entonces, para cada $v_k \in v(path_{v_i,v_j})$ se define $cond(v_k, path_{v_i,v_j})$ como las condiciones de propagación del vértice v_k y a $patterns(cond(v_k, path_{v_i,v_j}))$ al conjunto de vectores de entrada que cumple la condición impuesta por el vértice v_k para ejercitar la ruta $path_{v_i,v_j}$. La ruta $path_{v_i,v_j}$ no se puede verificar si no se cumplen todas las condiciones $cond(v_k, path_{v_i,v_j})$.

Condiciones de Propagación

$$path_{v_i,v_j} \text{ es no verificable} \Rightarrow \exists\, v_k \in v(path_{v_i,v_j}) \mid patterns(cond(v_k, path_{v_i,v_j})) = \emptyset \quad (3.2)$$

Y además, en base a esta última definición, la condición necesaria para identificar un conjunto de vectores que permitan verificar la ruta $path_{v_i,v_j}$ como:

$$patterns(path_{v_i,v_j}) = \bigcap_{v_k \in v(path_{v_i,v_j})} patterns(cond(v_k, path_{v_i,v_j})) \quad (3.3)$$

Entonces, para la CBR, se define una ruta verificable como:

$$path_{v_i,v_j} \text{es verificable} \iff \exists\, patterns(path_{v_i,v_j}) \neq \emptyset \mid patterns(path_{v_i,v_j}) = \bigcap_{\forall v_k \in v(path_{v_i,v_j})} patterns(cond(v_k, path_{v_i,v_j})) \quad (3.4)$$

como el conjunto de vectores de entrada que permiten verificar la ruta $path_{v_i,v_j}$ con la métrica de cobertura de rutas.

En estas condiciones, se puede definir el conjunto de rutas a verificar RV_{paths} como el conjunto de rutas que poseen al menos un patrón de entrada que permite ejercitarlo. Evidentemente RV_{paths} es un subconjunto de las rutas del circuito $\vec{G}$ a verificar.

Ejercitar Ruta

$$RV_{paths} = \{path_1,\ path_2,\ \ldots,\ path_k,\ \ldots,\ path_{n-1}\} \quad path_k = path_{v_i,v_j} \mid patterns(path_{v_i,v_j}) \neq \emptyset \quad (3.5)$$

donde $path_k$ es una ruta ejercitable. Además, hay que tener en cuenta que dado un patrón de verificación, son necesarios dos vectores de entrada para propagar una transición a través de la ruta a verificar. Atendiendo al ejemplo seguido en la figura 3.1 para verificar la ruta $path_4$ y aplicando la ecuación 3.6 la cobertura alcanzada resulta:

Patrón de Entrada Frente a Vector de Entrada

$$CBR(path_4) = \frac{RE(path_4)}{RV(\vec{G})} = \frac{1}{14} = 7'1473\,\% \quad (3.6)$$

Si para verificar cada ruta es necesario un patrón de entrada, es previsible que para verificar la totalidad del circuito es necesario emplear un patrón — dos

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

vectores — por ruta a verificar mediante la métrica CBR. Es decir, será necesario introducir 28 vectores — una pareja de vectores por patrón — para completar la verificación de las 14 rutas que componen este circuito. Una vez sean ejercitados estas 14 rutas de forma satisfactoria, se puede concluir diciendo que la cobertura de verificación de rutas alcanzada ha sido el 100 %.

El número de vectores de verificación, para alcanzar un 100 % en el nivel de confianza, puede ser tan elevado que el tiempo dedicado a la verificación del producto limite su número. Puede darse el caso, que dicho tiempo sea inferior al necesario para realizar la verificación completa. Es decir, no hay tiempo material para ejercitar todas las rutas del circuito a verificar. Ello introduce la necesidad de acelerar el proceso de generación y chequeo con cualquier técnica que permita aumentar el número de rutas comprobadas por patrón de entrada.

Analizando el conjunto de rutas a ejercitar, las condiciones que estos requieren, conjuntamente con los posibles patrones que permiten su verificación, se deduce que en ciertas ocasiones hay patrones de entrada que permiten verificar más de una ruta al mismo tiempo. Es decir, al ejercitar un conjunto de rutas determinadas, es posible que cierto subconjunto de rutas de entre todas las rutas verificables pueda ser ejercitado simultáneamente.

Sea $path_{set} = \{path_a,\ path_b,\ \ldots,\ path_j\}$ un subconjunto de rutas de entre el conjunto de rutas verificables RV_{paths}, esto es, $path_{set} \subset RV_{paths}$. Se define que ese subconjunto de rutas $path_{set}$ es compatible si:

$$path_{set} \text{ es compatible} \iff \exists\, patterns(path_{set}) \neq \emptyset \mid \qquad (3.7)$$
$$patterns(path_{set}) = \bigcap_{\forall path_k \in path_{set}} patterns(path_k) \neq \emptyset$$

La figura 3.2(b) muestra la verificación de cuatro rutas compatibles del circuito combinacional de ejemplo empleado hasta ahora (véase la figura 3.2(a)). En este caso, las rutas $path_1$, $path_2$, $path_{13}$ y $path_{14}$ pueden ser ejercitadas de forma simultanea mediante los patrones de entrada `p0x10q` ó `p01x0q` donde `p` y `q` son las transiciones de `1` a `0` (ó de `0` a `1`) en las entradas primarias 1 y 13, respectivamente.

Incompatibilidad entre Rutas

Es sencillo comprobar que la ruta $path_1$ impone que el arco e_2 posea un cero lógico. Por otro lado, la ruta $path_2$ impone que este arco e_2 también posea un cero lógico. Es evidente que a pesar de ser dos condiciones independientes, exigidas por dos rutas distintas, se convierten en una única condición pues coinciden. Esas dos condiciones independientes podrían haber ocasionado una incompatibilidad entre las rutas $path_1$ y $path_2$ si hubiesen exigido valores contradictorios sobre el arco e_2. Igualmente, el $path_2$ necesita fijar el valor del arco e_9 a un cero lógico. A diferencia del valor del arco e_2 que impone también un cero lógico a la entrada v_2, el cero lógico del arco e_9 se puede satisfacer de formas diversas. En este caso es posible seleccionar un valor `1` y `x` ó `x` y `1` para las entradas v_3 y v_4, respectivamente.

Propagación de Múltiples Rutas

Simultáneamente, existe la posibilidad de propagar una transición a través de la ruta $path_{14}$ sin necesidad de imponer condición alguna sobre la red de lógica combinacional. Igualmente, se puede propagar una transición por la ruta $path_{13}$, si se asegura un cero lógico en el arco e_{17}. Si se desea incluir a la ruta $path_{13}$ dentro del conjunto de rutas que comprueba el patrón actual, se ha de tener en cuenta que entra en juego la compatibilidad de rutas preseleccionadas anteriormente. La nueva condición, necesaria para ejercitar la ruta $path_{13}$, ha de coexistir con las

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

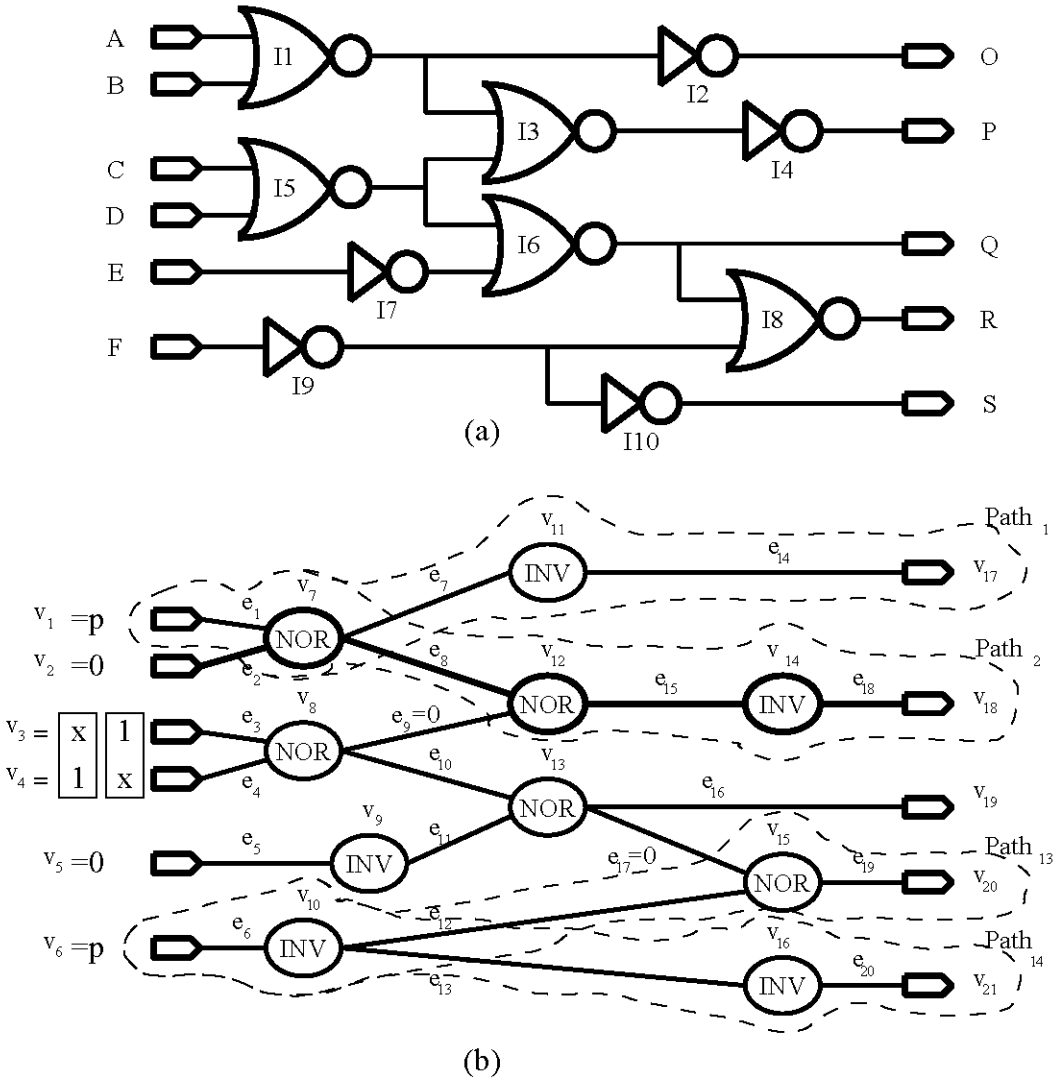


Figura 3.2: Ejemplo de cobertura basada en rutas con ejercitación múltiple: (a) circuito combinacional a verificar, (b) verificación simultánea de múltiples rutas.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

restricciones impuestas por el resto de rutas sin entrar en contradicción con ellas. En el ejemplo, la condición impuesta al arco e_9 en conjunción con la nueva condición impuesta sobre el arco e_{17} inducen que el arco e_{11} posea un uno lógico; lo que ocasiona que el arco e_5 posea un cero lógico, para conseguir que el inversor v_9 posea ese valor deseado en su salida. Como consecuencia de esta cadena de condiciones para, comprobar la ruta $path_{13}$ de forma conjunta al resto de rutas ($path_1$, $path_2$ y $path_{14}$), se ha de fijar la entrada E, el vértice equivalente v_5, a un cero lógico.

Si se emplea alguno de estos patrones que permiten ejercitar cuatro rutas de forma simultánea, la cobertura basada en rutas es:

$$CBR(path_{Set_1}) = \frac{RE(path_{Set_1})}{RV(\vec{G})} = \frac{4}{14} = 28'57\,\% \tag{3.8}$$

donde,

$path_{Set_1} = \{path_1, path_2, path_{13}, path_{14}\}$, y $\vec{G}$ el circuito combinacional.

3.3.1. Modelo de Propagación de Puerta

Con el fin de soportar el concepto de ejercitación de rutas y, por tanto, el de las condiciones de contorno de una ruta, se ha de introducir un modelo funcional de puerta basado en el modelo de lógica combinacional tradicional modificado.

De los dos ejemplos seguidos en los apartados anteriores, se deduce que es necesario seguir sólo dos reglas básicas para modelar el comportamiento de una puerta lógica durante el proceso de verificación basado en cobertura por rutas:

- **Cada puerta puede propagar a su salida, exclusivamente, una transición desde alguna de sus entradas.**

  La presencia de una transición implica un retardo de propagación. La convergencia de más de una transición sobre una puerta conlleva la posibilidad de la aparición de valores espúreos transitorios. La presencia de estos valores espúreos impedirían realizar la medida de la cobertura basada en rutas.

- **Si no hay una transición presente en alguna de las entradas, la puerta determina su respuesta empleando su modelo de comportamiento tradicional.**

  Las puertas que no se emplean para propagar una transición se utilizan para fijar los valores requeridos por las condiciones de propagación. Es decir, las puertas que no propagan una transición se emplean para garantizar un valor Booleano o eliminar una transición particular de la subred que si propaga una transición.

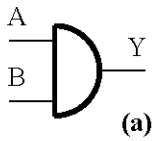
Modelo de Propagación de la puerta AND

Puerta AND

Sea una puerta AND con entradas A, B y salida Y, como la mostrada en la figura 3.3(a). La figura 3.3(b) presenta la nueva descripción funcional, propuesta en esta Tesis Doctoral, expresada como tabla de verdad. En este modelo de la puerta AND, cuando una de sus entradas propaga una transición y la otra entrada posee un uno lógico, la salida propaga la transición presente en alguna de sus entradas. Cuando en una entrada hay presente una transición y la otra entrada esta puesta a

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

cero, la salida no propaga transición alguna y el estado de la salida presenta cero lógico en dicho terminal. La presencia de una propagación en una entrada de una puerta, no determina por sí mismo la presencia de dicha propagación en la salida de la puerta. El resto de terminales de entrada actúan como variables de control, existiendo un valor predominante que permite eliminar la transición propagada y un conjunto de valores de paso. Para la puerta AND el terminal de control es la entrada que no propaga, y esta ha de poseer un uno lógico para permitir el paso de la propagación, o un cero lógico en caso de querer que la puerta AND no permita dicha transición.



(a)

A	B	Y
0	0	0
0	1	0
0	p	0
1	0	0
1	1	1
1	p	p
p	0	0
p	1	p
p	p	-

(b)

Figura 3.3: Modelo básico de la puerta AND para la propagación de transición.

No está permitida la aparición de transiciones en ambas entradas. En otro caso surge un error como se indica a continuación. La concurrencia de más de una propagación sobre una puerta, implica directamente la existencia de valores espúreos debidos a los tiempos de propagación de cada una de las transiciones que convergen en esa puerta. Dichos valores espúreos no permitirían realizar la verificación basada en rutas, pues introducirían transiciones indeseadas en la salida primaria de la ruta bajo verificación.

En el resto de casos, cuando no hay presente una transición propagada hasta las entradas de la puerta AND, el modelo de la puerta AND opera en el modo convencional de una AND .

Es necesario aclarar que dentro del campo de la generación de vectores de test, se define el término de controlabilidad y observabilidad de un nodo. El primero de ellos indica la capacidad de fijar el valor de un arco interno en una red Booleana mediante algún patrón de entrada a determinar. Por otro lado, el término observabilidad es definido como la propiedad de examinar el valor estático de un nodo interno específico, mediante la fijación de un patrón de entradas que permita extraer dicho valor, a alguna salida primaria del circuito. En nuestro caso, se define una transición con el objeto de comprobar todos los valores posibles (uno y cero lógico) en cada uno de los nodos de la ruta bajo verificación. Es decir observar en la salida primaria de la ruta bajo verificación, la propagación de un cero y la propagación de un uno lógico. Por otro lado, cada puerta de la ruta a verificar define una serie de condiciones y estas han de ser garantizadas, es necesario encontrar un patrón de entrada que garantice cada uno de los valores impuestos por las condiciones de la ruta.

Test Frente a Verificación

El objetivo de la generación de vectores test es determinar si la red estudiada

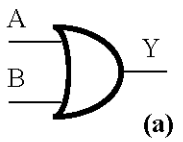
© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

posee algún tipo de fallo físico, por ejemplo del estilo *stuck–at*[8] [48]. Para ello, se determina el conjunto de nodos a observar y el vector estático de entrada que permite realizar la observación.

Cada vector de entrada permite comprobar si existe un error del tipo *stuck–at zero* o no, sobre un conjunto de arcos determinados. Para verificar si existe un *stuck–at one*, localizado en el mismo emplazamiento que el *stuck–at zero* comprobado anteriormente, es necesario introducir otro vector de entrada. Si existe un error, el patrón de salida que presenta la red, difiere del valor estimado si y sólo si el error es observable. Además un único vector de entrada permite descubrir más de un error, demostrándose que una vez descubierto el error, su localización no es sencilla.

Por el contrario, la verificación basada en rutas intenta identificar un fallo funcional. La medida de la cobertura permite cuantificar que partes del circuito se han ejercitado y, por tanto, que funcionalidades no han sido verificadas. La relación entre las rutas ejercitadas y las funcionalidades que implementan es directa, así que una vez se determine que una ruta viola las especificaciones de su funcionamiento, la tarea de localización de la funcionalidad que contiene el fallo es inmediata y directa.

Puerta OR



(a)

A	B	Y
0	0	0
0	1	1
0	p	p
1	0	1
1	1	1
1	p	1
p	0	p
p	1	1
p	p	-

(b)

Figura 3.4: Modelo básico de la puerta OR para la propagación de transición.

Sea una puerta lógica OR de dos entradas A y B como la mostrada en la figura 3.4(a). El modelo de puerta OR desarrollado en esta Tesis Doctoral es a su vez presentado en la figura 3.4(b).

Modelo de Propagación de la Puerta OR

La presencia de una transición en alguna de las entradas ha de ser acompañada de la presencia de un cero lógico en la otra entrada, para que la puerta OR propague la transición a su salida. Por el contrario, si existe una transición en una entrada y la otra entrada posee un uno lógico, la puerta OR en su salida fija un uno lógico.

Como se puede observar, el modelo de la puerta OR se comporta como el modelo de puerta AND, al no permitir la propagación de transiciones desde más

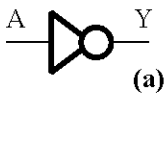
[8]En la bibliografía se define un *stuck–at* como un fallo — del tipo corto circuito o circuito abierto — donde la señal permanece constante a un valor lógico verdadero o falso.

[48] M. Abramovici, M. Breumer, and A. Friedman. *Digital Systems Testing and Testable Design*. Wiley–IEEE Press, January 1993.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

de una de sus entradas al mismo tiempo. Para el resto de condiciones de entrada, el modelo de puerta para la OR funciona como una puerta lógica OR convencional.

Puerta INV



(a)

A	Y
0	1
1	0
p	p

(b)

Figura 3.5: Modelo básico de la puerta INV para la propagación de transición.

El nuevo modelo de puerta de un INV es presentado en la figura 3.5(b). Debido a que la puerta INV sólo posee un terminal de entrada, este siempre propaga la transición. De otra forma, la inversión de la entrada es modelada en su única salida. Finalmente, el modelo de puerta del INV nunca produce un error.

Modelo de Propagación de la Puerta INV

3.3.2. Función Inversa del Modelo de Puerta

Seleccionado una ruta a verificar, se ha de garantizar el conjunto de condiciones que permiten propagar una transición a través de dicha ruta. Realizar la búsqueda de un patrón de entrada que permita ejercitarlo, se traduce sin complicación alguna, a un problema de satisfabilidad. Reducir el problema de satisfabilidad es siempre un objetivo deseado.

La obtención de la función inversa de cada puerta, y en su defecto la función inversa condicionada a ciertos valores de entrada, permite extender las condiciones impuestas por las puertas de la ruta a verificar. Extender dichas condiciones permite reducir el problema de satisfabilidad y, en ciertos casos, permite identificar la incompatibilidad de condiciones y, por tanto, entre rutas mediante la mera expansión de estas condiciones, sin ni si quiera formular el problema de satisfabilidad.

Función Inversa Condicionada

Por ejemplo, sea el circuito de la figura 3.6(a), cuyo grafo representa la figura 3.6(b). Se desea comprobar si se puede ejercitar simultáneamente la pareja de rutas etiquetadas como $path_7$ y $path_{16}$. Al igual que con los ejemplos descritos en los apartados anteriores, ahora se puede recapitular que: "para ejercitar la ruta $path_7$ es necesario asegurar que los arcos e_2 y e_9 posean un cero lógico". A su vez, de entre las condiciones necesarias para ejercitar la ruta $path_{16}$ se plantea la necesidad de asegurar un cero lógico en el arco e_{16}. Del proceso de anotación de las condiciones necesarias para asegurar ambas rutas no ha aparecido ningún tipo de incompatibilidad. Ello es debido a que no ha surgido contradicción alguna en el proceso seguido. Anotado el grafo, es necesario plantear un problema de satisfabilidad con el objeto de obtener un patrón de entrada que ejercite ambas rutas o en su defecto, una incompatibilidad entre las rutas $path_7$ y $path_{17}$.

Sin embargo, mediante unas simples asociaciones derivadas de la estructura del grafo y las funcionalidades involucradas, es posible determinar que ambas rutas son incompatibles. Para el ejemplo de la figura 3.6(c) se puede decir que: "la condición de cero lógico sobre el arco e_9 implica que la puerta v_8 ha de poseer un

Rutas Incompatibles

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

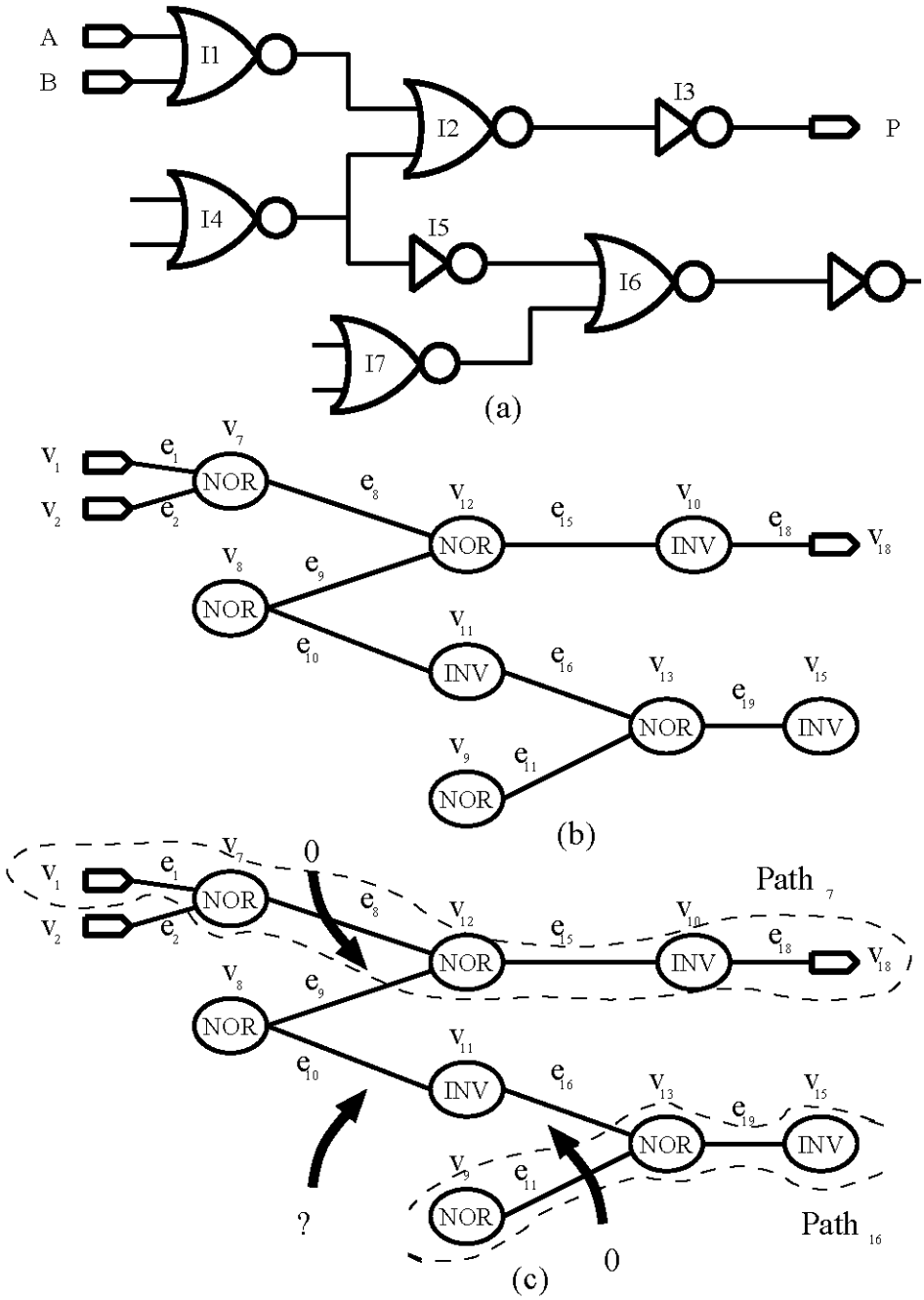


Figura 3.6: Ejemplo de la expansión de las condiciones de propagación: (a) esquema del subcircuito, (b) grafo del subcircuito, y (c) expansión de la condición de propagación de la ruta $path_{16}$ y contradicción sobre la ruta $path_7$.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

cero lógico en su salida". Esto supone que el arco e_{10} también posee un cero lógico. A su vez, del tratamiento de las condiciones de propagación de la ruta $path_{16}$, y en particular, la condición de cero lógico sobre el arco e_{16}, se observa que ésta condición puede ser extendida a través del inversor v_{11}. Ese cero lógico presente en la salida del inversor tiene que ser debido a la presencia de un uno lógico en su entrada. Luego la condición de cero lógico sobre el arco e_{16} se expande sobre el arco e_{10} con una condición de uno lógico.

Ahí surge la contradicción entre las rutas $path_7$ y $path_{16}$. El arco e_9 ha de poseer un cero lógico para asegurar la ruta $path_7$ y, a su vez, ese mismo arco ha de poseer un uno lógico para asegurar la ruta $path_{16}$. Lo que implica que ambas rutas no pueden ser ejercitados simultáneamente.

Realizar este sencillo estudio permite identificar incompatibilidades derivadas de las condiciones de propagación sin la necesidad de plantear el problema de satisfabilidad. Ello redunda en la reducción del tiempo necesario para determinar si dos o más rutas son incompatibles. Y por supuesto, una vez superada esta etapa de análisis, el problema de satisfabilidad queda reducido; pues en la mayoría de los casos, la red sobre la que se plantea el problema de satisfabilidad es menor que la original sin el preprocesado. La contraprestación a este preprocesado, es la necesidad de determinar las funciones inversas de cada una de las puertas lógicas empleadas.

La función inversa del modelo de la puerta INV es sencilla e intuitiva. Esta función inversa se corresponde con la propia función directa del inversor.

Para el resto de puertas lógicas combinacionales no existe función inversa directa. Esto es porque existe al menos una salida que puede ser obtenida mediante más de una combinación de entrada. El modelo de puerta presentado puede ser visto como una función matemática dependiente de varias variables de entrada. El conocimiento de un valor puntual de dicha función, no implica el conocimiento exacto de los valores de las variables que lo produjeron, salvo que estos valores se encuentren inducidos por otra condición o extensión de condición. Por ello, para conocer el valor de entrada de una de las variables, por lo general, se ha de conocer no sólo el valor de salida de la función, sino que se ha de conocer también, el valor del resto de las entradas.

Condiciones de Entrada

Igualmente, para determinar las condiciones de entrada que producen una condición a la salida en una puerta, es necesario conocer el valor que toman todas las entradas de la puerta, menos una. Salvo, que se de en las entradas un valor dominante, esto es, un valor de entrada en alguno de sus terminales que fije la salida de la puerta de forma independiente al resto de entradas. Este valor dominante también suele hacer de cerrojo. El valor cerrojo evita que se propague una transición desde alguna entrada a la salida. Por ejemplo el valor dominante o cerrojo de una puerta AND es el cero lógico. Si hay presente en al menos una entrada, un cero lógico, en la salida habrá presente un cero lógico, independientemente de lo que haya en el resto de entradas. El valor cerrojo de la puerta OR es el uno lógico. Si en una puerta lógica OR se fija alguna entrada a un uno lógico, la salida de la puerta se situará en un uno lógico independientemente de los valores presentes en el resto de entradas a la puerta.

3.3.3. Obtención del Patrón de Entrada para una Ruta

En este punto se presentará, formalmente, el algoritmo que permite determinar si una ruta es o no ejercitable. En caso afirmativo, también, se definirá su patrón

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

de entrada. La figura 3.7 presenta el algoritmo.

$ver_pattern()$
given a $\vec{G} = (V, \vec{E}, \tau)$;
set $\mathcal{PC}(path_{v_0,v_k})$ to $\emptyset$;
for each v vertex $\in$ path $path_{v_0,v_k}$ to be verified
get $cond(v, path_{v_0,v_k})$, propagation constraints at v vertex;
include $cond(v, path(v_0, v_k)$ in $\mathcal{PC}(path_{v_0,v_k})$;
end for;
for each $cond(v, path_{v_0,v_k}) \in \mathcal{PC}(path_{v_0,v_k})$
propagate constraint satisfability to $PI(\vec{G})$;
propagate constraints to $PO(\vec{G})$;
end for;
for each v vertex $\in$ path $path_{v0,vk}$ to be verified
extend propagations to $PO(\vec{G})$;
end for;
fix the unknow input logic values;
end $ver_pattern$

Figura 3.7: Algoritmo para la generación del patrón de entrada en la verificación de una ruta.

Este algoritmo, para un circuito de lógica combinacional $\vec{G} = (V, \vec{E}, \tau)$, consiste en:

- **Determinar las puertas lógicas que conforman la ruta a ejercitar, y fijar sus condiciones de contorno.–** Dada una ruta a ejercitar, este paso localiza cada uno de los vértices y arcos que componen la ruta que ha de propagar la transición. Cada una de esas puertas posee unas condiciones de contorno. Este paso, también, anota dichas condiciones de contorno en el grafo que representa a la red de lógica combinacional. De esta anotación puede surgir alguna incompatibilidad, si ese es el caso, el algoritmo finaliza pues no es posible encontrar un patrón de entrada que asegure las dos condiciones contradictorias.

- **Extender las condiciones de contorno.–** Las condiciones de contorno de la ruta, anotadas en la fase anterior, han de ser extendidas en dirección a las entradas y, posteriormente, hacia las salidas.

  Para extender una condición de contorno hacia las entradas se hace necesaria la existencia de un vector de entrada unívoco que permita satisfacer dicha condición. Como se apuntó en el apartado 3.3.2, una condición en una salida puede satisfacerse mediante múltiples vectores de entrada. En este paso, la condición será expandida hacia las entradas, si y sólo si, hay un conjunto de entradas que tienen definido su estado y existe un vector de entrada que cumple con esos valores y con la salida propuesta, de forma unívoca.

  Problema de Satisfabilidad

  En esta fase, también, se extienden las condiciones en dirección a las salidas. La expansión de las condiciones se realiza de forma similar a la ex-

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

puesta en el párrafo anterior. Es decir, no se condicionará la salida de una puerta, si la condición impuesta en la entrada no determina, unívocamente, dicha salida. Esto se hace así, para no restringir en exceso la red de lógica combinacional.

Si en el transcurso de esta fase se encontrase una contradicción entre dos o más condiciones, significaría que la ruta a verificar no es ejercitable. En ese caso el algoritmo finalizaría, pues, no es posible satisfacer todas las condiciones de contorno de la ruta bajo verificación.

Ruta no Verificable

- **Propagar la transición de la ruta bajo verificación a los vértices no pertenecientes a dicha ruta, pero con arcos comunes a la ruta a verificar.–** Propagar una transición a través de una ruta implica que otras puertas, también, se vean afectadas por esa transición. Esta propagación en dirección a las salidas, sólo se puede realizar en el caso de que ésta sea determinada por la funcionalidad de la puerta de forma unívoca. Si durante este proceso se diera una reconvergencia de la propagación, el algoritmo terminaría igualmente la búsqueda del patrón de entrada, pues la reconvergencia de transiciones no está permitida en la verificación basada en la métrica de cobertura de rutas.

- **Encontrar el patrón de entrada que cumpla con todos los anteriores.–** Alcanzada esta última fase, la red de puertas combinacional contiene anotaciones referentes a la ruta a verificar, las condiciones de contorno a cumplir, junto a sus expansiones y la propagación de las transiciones. Este paso busca el patrón de entrada que cumple con todas las anotaciones de la red Booleana. Además, atendiendo al estado de las entradas del circuito bajo verificación, se puede observar que existen terminales de entrada condicionados o fijos a un valor, hay un terminal que define el comienzo de una propagación y, por último, hay terminales con valor desconocido.

  El objetivo de esta última fase es descubrir el valor de los terminales que no poseen un valor conocido. Para ello se puede emplear un simulador lógico [66], [21], simbólico [67], [9] o establecer las condiciones de satisfabilidad [49], [68]. El empleo de uno u otro es indiferente.

[66] D. Becker, R.K. Singh, and S.G. Tell. An Engineering Environment for Hardware/Software Co–Simulation. *Design Automation Conference*, pages 129–134, June 1992.

[21] H. Navarro, Juan A. Montiel-Nelson, J. Sosa, and R. Sarmiento. DEMETER: A Novel Framework for Hardware and Software System Specification, Simulation and Verification in C++. *XVI Conference on Design of Circuits and Integrated Systems*, pages 20–23, November 2001.

[67] Randal E. Bryant. Symbolic Simulation Techniques and Applications. In *Proceedings of the 27th ACM/IEEE conference on Design automation*, pages 517–521, New York, NY, USA, 1990. ACM Press.

[9] H. Navarro, J. A. Montiel-Nelson, J. Sosa, and José C. García. A Symbolic Solver to Obtain the Entire Solution Space of the Boolean Satisfiability Problem using Genetic Algorithms. In *Evolutionary and Deterministic Methods for Design, Optimization and Control with Applications to Industrial and Societal Problems*, September 2005.

[49] E. Goldberg and Y. Novikov. BerkMin: a Fast and Robust Sat–Solver. *Proc. of Design, Automation and Test in Europe*, pages 149–149, March 2002.

[68] H. Navarro, J. A. Montiel-Nelson, J. Sosa, and José C. García. Solving the State Justification Problem using MILP for RTL Specifications. In *XIX Conference on Design of Circuits and Integrated Systems*, November 2004.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

Finalizada la ejecución del algoritmo, éste puede ofrecer dos resultados posibles. El primero de ellos es el patrón de entrada que permite verificar la ruta deseada. El patrón resuelto poseerá valores definidos, la transición y/o valores indiferentes. El otro resultado posible es la no existencia de un patrón.

3.3.4. Cálculo de la Cobertura para un Conjunto de Rutas

El algoritmo de obtención del patrón de máxima cobertura es mostrado en la figura 3.8.

$path_coverage()$
  given a $\vec{G} = (V, \vec{E}, \tau)$;
  set $path_{set}$ to $\emptyset$;
  add $path_{r,s} \in \vec{G}$ to $path_{set} \mid patterns(path_{r,s}) \neq \emptyset$;
  do
    select $path_{u,v} \in \vec{G} \mid path_{u,v} \notin path_{set}$;
    get the input pattern $ver_pattern(path_{set} \cup path_{u,v})$;
    if $(path_{set} \cup path_{u,v})$ is compatible then
      add $path_{u,v}$ to $path_{set}$;
    end if;
  until *Objective Function Criteria*
end $path_coverage$

Figura 3.8: Algoritmo para la generación de patrones de máxima cobertura.

Dado un circuito de lógica combinacional $\vec{G} = (V, \vec{E}, \tau)$ el algoritmo $path_coverage$ busca un conjunto de rutas compatibles para maximizar la CBR. En primer lugar, el algoritmo inicializa el conjunto de rutas compatibles $path_{set}$ al conjunto vacío. A continuación, selecciona una ruta $path_{r,s}$ verificable — esto es, $\exists\ patterns(path_{r,s})$ — y lo añade al conjunto $path_{set}$.

Entonces, comienza un bucle que busca rutas compatibles entre sí. En cada iteración de este bucle se selecciona una ruta $path_{u,v}$ del circuito $\vec{G}$, que no esté incluida en el subconjunto de rutas ya encontradas $path_{set}$. En segundo lugar averigua si existe un patrón de entrada que verifica la ruta seleccionada y el conjunto compatible que ya se posee de iteraciones anteriores. Si resulta que la ruta seleccionada unión el conjunto $path_{set}$ se define compatible, incluye la ruta $path_{u,v}$ en el conjunto de rutas verificables $path_{set}$.

Tras la primera iteración el conjunto de rutas $path_{set}$ puede contener un par de rutas compatibles o sólo la ruta con la que se inició el bucle dependiendo de si el algoritmo escoge una ruta compatible o no con la previamente seleccionada. En las sucesivas iteraciones, el conjunto $path_{set}$ va albergando a las rutas compatibles encontradas. El algoritmo finaliza cuando se alcanza la función objetivo.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

3.3.5. Búsqueda con Algoritmos Genéticos

Toda solución basada en Algoritmos Genéticos posee dos pilares claves. Uno de ellos es la codificación del problema, y el otro es la definición de una función objetivo. La codificación del problema consiste, de forma somera, en implementar un individuo de la población genética de tal forma que modele aquellas variables del problema que se ha de resolver. En la mayoría de los casos, la dificultad de implementar el individuo radica en identificar adecuadamente las variables del sistema a optimizar. La función de coste permite al núcleo de optimización genética guiar y seleccionar a los individuos y así estimar la bondad de la solución alcanzada. A continuación se procederá a presentar la función de coste, para posteriormente explicar la codificación del individuo.

Función de Coste

Si todas las rutas posibles de una red Booleana $\vec{G}$ están contenidas en el conjunto γ, y el conjunto α contiene un subconjunto de γ ($\alpha \subset \gamma$), donde α representa las rutas a verificar, la función de coste a minimizar se define como:

$$GA_{Cost} = \text{path count}(\gamma) - \text{path count}(\alpha) \tag{3.9}$$

Donde path count es una función que proporciona el número de elementos de un conjunto. Además, el modelo de propagación de transiciones propuesto en el apartado 3.3.1 no permite la verificación de más rutas, con un único patrón, que el número de salidas primarias disponibles ($PO(\vec{G})$). Esta consideración limita el número de rutas incluidas en el subconjunto. Entonces, es más apropiada la función de coste:

Función Objetivo

$$GA_{Cost} = \text{vertex count}(PO(\vec{G})) - \text{path count}(\alpha) \tag{3.10}$$

donde PO count es una función que proporciona el número de salidas primarias de una red Booleana.

Codificación del Individuo Genético

La función objetivo es obtener un patrón de entrada que maximice la cobertura de verificación del circuito. Esto es, encontrar el patrón que contenga el mayor número de rutas que puedan ser verificadas simultáneamente.

El individuo de la población para la solución adoptada con Algoritmos Genéticos consiste en un arreglo de n números enteros. El individuo representa el subconjunto de rutas seleccionadas, y su longitud n coincide con el número de salidas primarias del circuito. El número entero contenido en cada elemento del individuo define el índice de la ruta que es seleccionada de todas las posibles rutas existentes.

Codificación

Exploración de la Cobertura

La exploración de la cobertura del diseño empleando algoritmos genéticos es abordada mediante el algoritmo presentado en la figura 3.9. De forma iterativa, el algoritmo va obteniendo el patrón de máxima cobertura y eliminando las rutas que ya están cubiertas por dicho patrón de la lista de rutas posibles. Este proceso

Búsqueda del Patrón de Máxima Cobertura

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

de búsqueda y eliminación se realiza hasta que la lista de rutas posibles esté vacía o sólo se encuentren las rutas no verificables. Al finalizar el algoritmo, éste devolverá una lista con los patrones de máxima cobertura para la verificación.

$GApath_coverage()$
  given a $\vec{G} = (V, \vec{E}, \tau)$;
  do
    obtain maximum coverage pattern using GA;
    remove path of maximum coverage from $\vec{G}$;
  while $size(\vec{G}) > 0$
end $GApath_coverage$

Figura 3.9: Algoritmo de generación de patrones de máxima cobertura empleando Algoritmos Genéticos.

Dado un individuo, se comprueba la primera ruta seleccionada. Si se determina que existe un vector de entrada que permite verificarlo, se procede a comprobar la segunda ruta especificada. Si la ruta seleccionada no es compatible, la comprobación del subconjunto de rutas especificadas finaliza. El coste del subconjunto evaluado es estimado mediante la ecuación 3.10, sobre las rutas que se han evaluado positivamente en los pasos previos para el individuo en estudio.

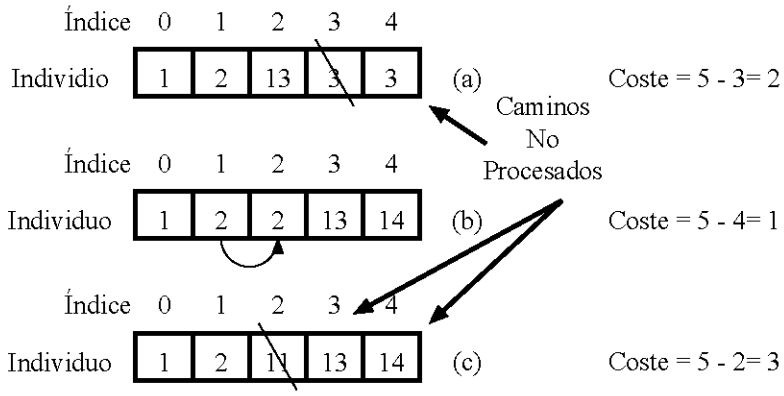


Figura 3.10: Cálculo del coste de un individuo: (a) individuo con 3 rutas válidas, (b) individuo con 4 rutas válidas y una de ellas duplicada, (c) individuo con tan sólo 2 rutas válidas.

La figura 3.10 muestra tres ejemplos del cálculo de la función de coste para tres situaciones distintas sobre el ejemplo seguido de la figura 3.1. En la figura 3.10(a) y (c), cuando son comprobados los elementos 3 y 2 de los respectivos individuos, surge una incompatibilidad entre rutas y, por tanto, el estudio del coste finaliza. Para esos casos, el coste evaluado es de 2 y 3 en el orden dado. Sin embargo, en el caso mostrado en la figura 3.10(b) el individuo a evaluar posee dos de las rutas seleccionadas duplicadas. Esta duplicidad es identificada y tratada por el algoritmo de calculo del coste eliminándola y evaluando solamente una única ocurrencia. Aplicando este proceso al último ejemplo citado, el coste final del individuo es 1.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

Resultados Experimentales

Con el propósito de evaluar la solución basada en técnicas de programación evolutiva, aplicadas a la verificación con la métrica de cobertura de rutas, se ha escogido un conjunto de circuitos multinivel y de dos niveles, del banco de pruebas MCNC'91 [51]. Todos los experimentos presentados han sido procesados en un servidor Sun–Fire 280R con dos CPU UltraSPARC III a 900 MHz y 4 $GBytes$ de RAM, ejecutando SunOS v5.8. Además se ha empleado GENEsYs [69] como núcleo del Algoritmo Genético y MISII [70] como sistema de síntesis lógica.

Algoritmo Genético Frente a Solución de Fuerza Bruta

La solución basada en Algoritmos Genéticos es comparada con un algoritmo de búsqueda de rutas (fuerza bruta) [71], [72]. El algoritmo de búsqueda de rutas comprueba todas las posibles combinaciones de rutas hasta encontrar el subconjunto de cobertura máxima. En todos los experimentos realizados con la solución evolutiva, se adoptó una estrategia de selección de los mejores individuos[9] en el momento de determinar qué población permanecía en la siguiente generación como población descendiente[10]. A su vez, el esquema de selección fijado al núcleo del Algoritmo Genético ha sido el proporcional. Todo ello sobre una población de 100 individuos. Finalmente, se ha permitido el empleo de Mutación Multipunto con cruzamiento, además de la Mutación Estándar.

La tabla 3.2 muestra los resultados de la comparativa entre el Algoritmo Genético y el algoritmo de búsqueda exhaustiva de rutas. La primera columna presenta el nombre del circuito MCNC'91 [51] empleado. La columna segunda presenta la complejidad del circuito medida términos de número de puertas. La tercera y cuarta columna muestra el número total de rutas y el número de salidas del circuito, respectivamente. En la quinta columna se ofrece el número de intentos a realizar por parte del núcleo evolutivo. La sexta columna muestra el número de rutas que incluye el mejor vector encontrado por el Algoritmo Genético. En la columna séptima se presenta el tiempo de CPU tomado por el Algoritmo Genético para encontrar el mejor conjunto de rutas. En la columna octava se muestra el tiempo de CPU que toma el Algoritmo de búsqueda de rutas(fuerza bruta), para encontrar un subconjunto equivalente al encontrado con el Algoritmo Genético. Para finalizar, la última columna estima los ratios entre los tiempos de CPU de ambas soluciones.

La figura 3.11 presenta gráficamente los ratios de tiempo de CPU y el número de intentos realizados para cada uno de los circuitos de la tabla 3.2.

[9] Identificado en la bibliografía como *Steady State*.

[10] nombrado en la literatura anglosajona como *offspring*.

[51] S. Yang. *Logic Synthesis and Optimization Benchmarks User Guide version 3.0*. Microelectronics Center of North Carolina, 1991.

[69] Thomas Bäck. *A User's Guide to GENEsYs*. University of Dortmund, Department of Computer Science, Systems Analisys Research Group, Dortmund, v1.0 edition, 1992.

[70] R.K. Brayton, R. Rudell, A.L. Sangiovanni-Vincentelli, and A. Wang. MIS: A Multiple Level Logic Optimization System. *IEEE Trans. Comput.–Aided Des. Integr. Circuits Syst.*, 6(11):1062–1081, 1987.

[71] J. Sosa, J. A. Montiel-Nelson, H. Navarro, and José C. García. Functional Vector Generation for Maximum Path Coverage using Evolutionary Programming. In *XVIII Conference on Design of Circuits and Integrated Systems*, November 2003.

[72] J. Sosa, J. A. Montiel-Nelson, H. Navarro, and José C. García. Stimuli Sequence Generation for Verification and Validation of Maximum Power Consumption using Evolutionary Programming. In *Evolutionary and Deterministic Methods for Design, Optimization and Control with Applications to Industrial and Societal Problems 2005*, September 2005.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

Tabla 3.2: COMPARATIVA DE LOS EXPERIMENTOS REALIZADOS ENTRE LA METODOLOGÍA BASADA EN ALGORITMOS GENÉTICOS Y EL ALGORITMO DE BÚSQUEDA EXHAUSTIVA DE RUTAS PARA LA VERIFICACIÓN EMPLEANDO LA MÉTRICA DE COBERTURA DE RUTAS

Circuito				Algoritmo Genético			A. Exhaustivo	Ratio
Nombre	Puertas	Rutas	Salidas	Intentos	Rutas	Tiempo CPU	Tiempo CPU	Tiempo CPU
5xp1	175	172	10	1000	4	8'700E-01	4'484E+04	5'15E+04
				5000	4	6'250E+00	4'484E+04	7'17E+03
				10000	5	1'346E+01	5'605E+06	4'16E+05
				50000	6	5'227E+01	7'961E+08	1'52E+07
				100000	9	1'664E+02	1'160E+13	6'97E+10
apex4	3947	14435	19	1000	2	1'216E+02	1'602E+05	1'32E+03
				5000	2	1'515E+02	3'906E+05	2'58E+03
				10000	3	3'690E+02	2'333E+09	6'32E+06
				50000	3	5'014E+02	2'333E+09	4'65E+06
				100000	3	8'757E+02	9'877E+08	1'13E+06
b12	165	126	9	1000	5	4'922E+01	4'334E+02	8'80E+00
				5000	5	8'621E+01	4'334E+02	5'03E+00
				10000	6	1'751E+03	1'507E+04	8'61E+00
				50000	7	3'138E+03	1'063E+08	3'39E+04
				100000	8	1'004E+04	2'126E+08	2'12E+04
bw	314	388	28	1000	2	2'100E-01	2'633E+01	1'25E+02
				5000	3	1'440E+00	3'401E+02	2'36E+02
				10000	3	3'140E+00	2'690E+06	8'57E+05
				50000	6	1'719E+01	1'121E+11	6'52E+09
				100000	9	6'584E+01	2'891E+18	4'39E+16
clip	177	234	5	1000	4	3'500E-01	3'731E+01	1'07E+02
				5000	4	2'070E+00	3'511E+01	1'70E+01
				10000	4	4'440E+00	4'170E+01	9'39E+00
				50000	5	5'469E+01	2'205E+06	4'03E+04
				100000	5	1'099E+02	2'205E+06	2'01E+04
con1	31	19	2	1000	2	9'325E-02	1'456E-02	1'56E-01
				5000	2	2'953E-01	1'456E-02	4'93E-02
				10000	2	5'284E-01	2'800E-03	5'30E-03
				50000	2	4'181E+00	1'456E-02	3'48E-03
				100000	2	1'041E+01	1'456E-02	1'40E-03
misex1	108	119	7	1000	3	7'848E-01	1'129E+02	1'44E+02
				5000	5	2'339E+00	1'715E+04	7'33E+03
				10000	5	5'408E+00	1'178E+03	2'18E+02
				50000	6	3'128E+01	6'028E+07	1'93E+06
				100000	6	6'179E+01	7'524E+06	1'22E+05
misex3c	1065	1163	14	1000	2	1'128E+02	6'900E+01	6'12E-01
				5000	2	4'050E+02	6'900E+01	1'70E-01
				10000	2	7'537E+02	6'900E+01	9'15E-02
				50000	3	4'002E+03	6'494E+03	1'62E+00
				100000	4	7'697E+03	5'072E+06	6'59E+02
inc	318	520	9	1000	2	5'595E-01	5'796E+01	1'04E+02
				5000	3	2'751E+00	2'649E+04	9'63E+03
				10000	3	6'139E+00	2'649E+04	4'32E+03
				50000	5	6'066E+01	1'085E+09	1'79E+07
				100000	6	8'332E+01	1'874E+09	2'25E+07
rd53	88	71	3	1000	2	9'325E-02	2'720E-02	2'92E-01
				5000	3	5'828E-01	3'840E-01	6'59E-01
				10000	3	1'267E+00	2'283E+01	1'80E+01
				50000	3	1'128E+01	2'283E+01	2'03E+00
				100000	2	2'274E+01	2'283E+01	1'00E+00
rd73	212	235	3	1000	2	1'258E+00	6'242E+00	4'96E+00
				5000	2	4'094E+00	6'242E+00	1'52E+00
				10000	2	5'408E+00	6'242E+00	1'15E+00
				50000	3	1'819E+01	9'624E+02	5'29E+01
				100000	3	3'028E+01	9'624E+02	3'18E+01
rd84	365	949	4	1000	2	2'226E+00	1'415E+02	6'36E+01
				5000	3	3'721E+01	2'508E+05	6'74E+03
				10000	3	1'414E+01	7'395E+04	5'23E+03
				50000	4	1'122E+01	4'685E+05	4'18E+04
				100000	4	3'059E+02	4'685E+05	1'53E+03
sao2	295	255	4	1000	2	2'036E+00	8'482E+00	4'17E+00
				5000	3	4'477E+00	3'024E+03	6'75E+02
				10000	3	8'780E+00	7'164E+02	8'16E+01
				50000	4	4'760E+01	2'024E+05	4'25E+03
				100000	4	9'044E+01	1'074E+05	1'19E+03
Z5xp1	815	2866	10	1000	2	2'310E+00	3'075E+03	1'33E+03
				5000	2	1'433E+01	1'185E+03	8'27E+01
				10000	2	2'017E+01	7'127E+03	3'53E+02
				50000	3	1'482E+02	5'215E+06	3'52E+04
				100000	5	4'164E+02	9'473E+10	2'28E+08

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

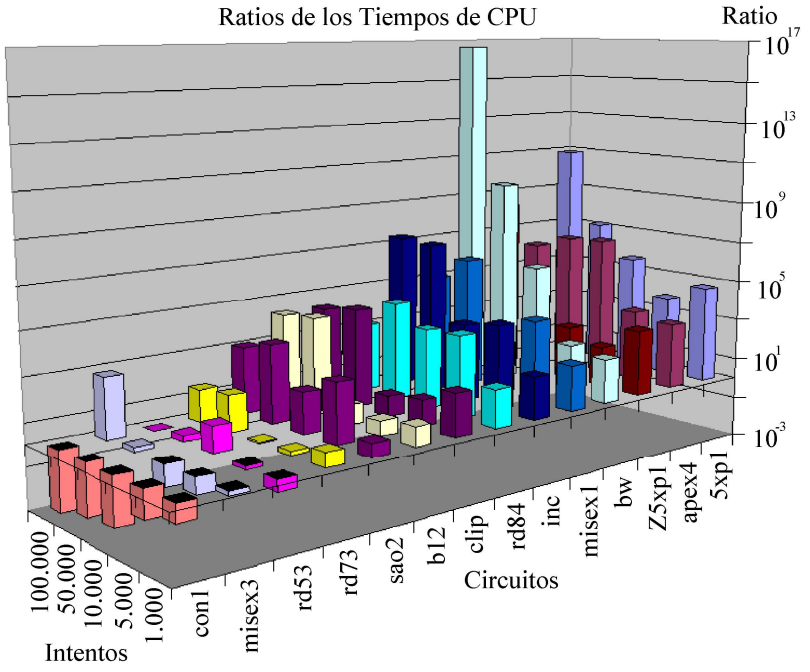


Figura 3.11: Ratios del tiempo de CPU para el conjunto de circuitos MCNC'91.

Convergencia del Algoritmo Genético

Por otro lado, la figura 3.12 y 3.13 muestra la convergencia de la función de coste del Algoritmo Genético en la optimización del circuito 5xp1 para los mejores individuos y para los individuos de coste medio, respectivamente. Se presenta la convergencia para un subconjunto de 9 rutas, dos subconjuntos de 7 rutas y por último un subconjunto de 4 rutas. El número de intentos en este experimento ha sido de 100.000.

La figura 3.14 presenta la función de convergencia seguida por los peores individuos del experimento, para un subconjunto con 9 y 4 rutas verificables.

Conclusiones

La cobertura basada en rutas es una métrica objetiva para la verificación y el test. Pero esta no ha sido explotada aún debido a que la complejidad de la misma crece exponencialmente con el número de puertas. La solución presentada en este trabajo de investigación introduce un nuevo concepto llamado propagación de puerta. El nuevo modelo de puerta permite emplear dicho concepto en la generación de patrones de verificación de máxima cobertura. La novedosa metodología basada en Algoritmos Genéticos desarrollada en esta Tesis Doctoral, permite obtener patrones de verificación óptimos en un tiempo muy reducido [71].

La solución basada en Algoritmos Genéticos es siempre mejor que el algoritmo de búsqueda de rutas cuando el número de salidas es mayor a 4. Sin embargo, el número de intentos debe ser ajustado para cada circuito en función del número total de puertas, entradas y salidas que posee el circuito de lógica combinacional

[71] J. Sosa, J. A. Montiel-Nelson, H. Navarro, and José C. García. Functional Vector Generation for Maximum Path Coverage using Evolutionary Programming. In *XVIII Conference on Design of Circuits and Integrated Systems*, November 2003.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

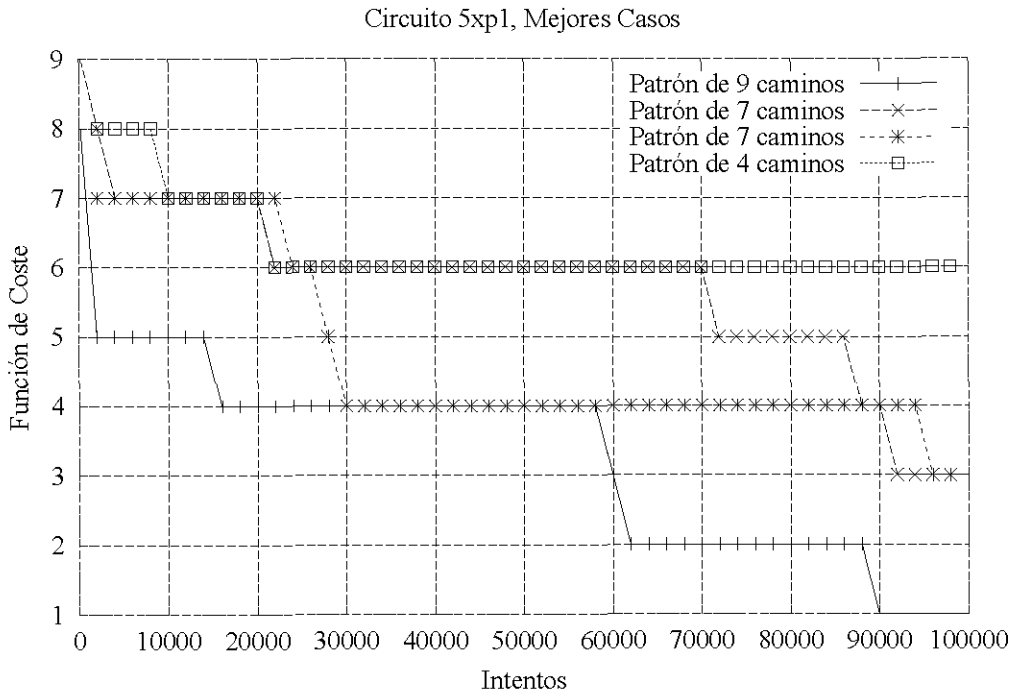


Figura 3.12: Convergencia de la función de coste de los mejores individuos en la generación de patrones de verificación para el circuito 5xp1.

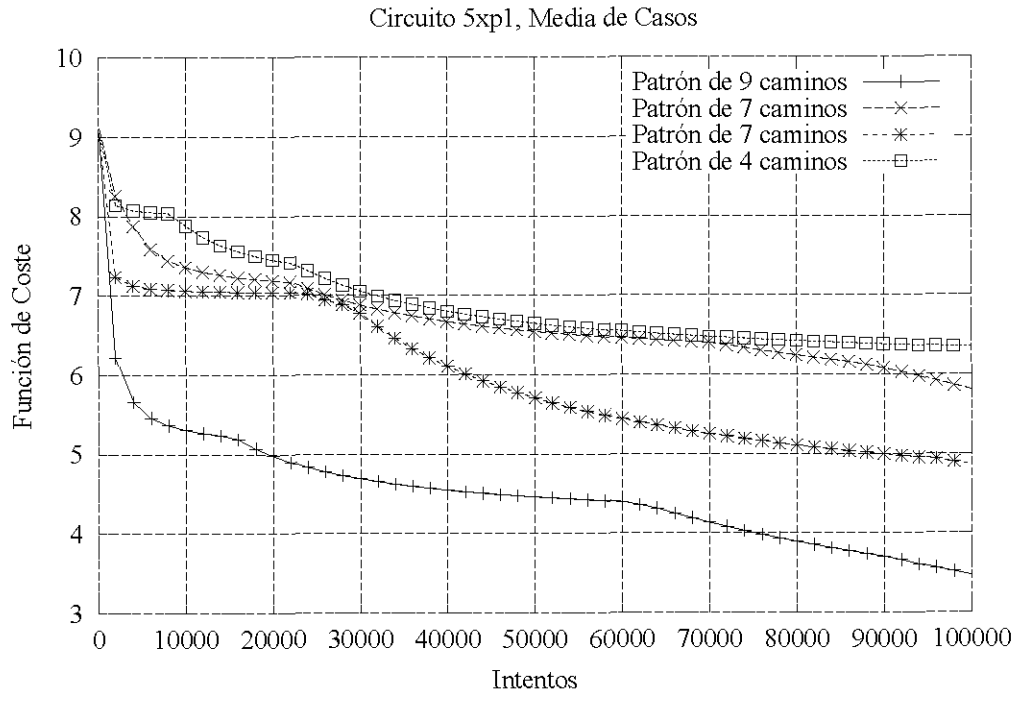


Figura 3.13: Convergencia de la función de coste medio de los individuos en la generación de patrones de verificación para el circuito 5xp1.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

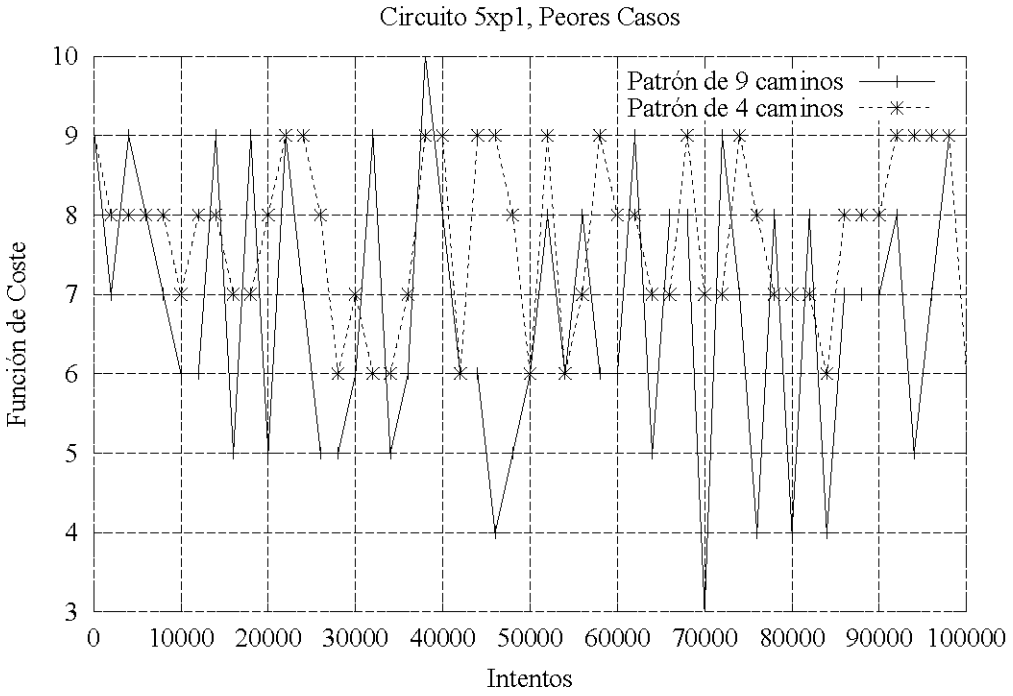


Figura 3.14: Convergencia de la función de coste de los peores individuos en la generación de patrones de verificación para el circuito 5xp1.

a verificar [72].

A modo de resumen y apoyado en el amplio conjunto de experimentos realizados, y como no, a lo ya expuesto en este capítulo, se puede concluir que: "la metodología propuesta para la generación de patrones de máxima cobertura con la métrica de cobertura de rutas ha demostrado ser eficaz, no sólo en términos de la calidad de los patrones de verificación generados, sino también en términos del tiempo de CPU requerida para producirlos".

3.3.6. Satisfabilidad de la Propagación de Rutas

El principal problema de la solución basada en algoritmos evolutivos es la de encontrar el vector de entrada que satisface todas las condiciones de las rutas a ejercitar. Este problema se encuentra muy estudiado dentro del área de los llamados SAT solvers. Con una adecuada modificación de la representación del modelo de propagación, el problema de encontrar el vector de entrada que propague el máximo número de rutas puede ser expresado como un problema de Programación Lineal.

3.3.7. Modelo de propagación

La solución inmediata sólo requiere que, una vez planteadas las condiciones de propagación por parte del algoritmo de la figura 3.7, se resuelva el problema

Formulación del Modelo de Propagación

[72] J. Sosa, J. A. Montiel-Nelson, H. Navarro, and José C. García. Stimuli Sequence Generation for Verification and Validation of Maximum Power Consumption using Evolutionary Programming. In *Evolutionary and Deterministic Methods for Design, Optimization and Control with Applications to Industrial and Societal Problems 2005*, September 2005.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

de satisfabilidad. El algoritmo genético propondría subconjuntos de prueba, los cuales serían evaluados mediante un problema de Programación Lineal Entera.

Analizando el problema en profundidad, se puede deducir que si se realiza una codificación adecuada de los estados válidos del modelo de propagación propuesto en el apartado 3.3.1, se puede formular la búsqueda del mejor patrón de entrada que maximice el número de rutas ejercitadas.

Para caracterizar adecuadamente la propagación de rutas en una red combinacional, es necesario codificar el modelo presentado en la figura 3.2. La codificación de cada una de las señales presentes en el circuito se realiza mediante 2 bits. Un bit representa los datos y el otro la propagación de una transición (por ejemplo a_d y a_p para la señal A). Si existe una propagación a través de la señal, su bit de propagación está activo y el bit de datos no tiene validez. Si el bit de propagación está inactivo, la señal está transmitiendo el dato presente en el bit de datos.

Del modelo presentado en la figura 3.3 se extrae que, para una puerta AND con entradas A, B y salida Y: la salida propaga una transición que llega a una de sus entradas si la otra entrada posee un uno lógico. Si la transición llega a una entrada y la otra entrada posee un cero lógico, la transición no es transmitida a la salida Y. En este último caso, la salida Y presentará un cero lógico. Si llega más de una transición simultáneamente, es decir, la puerta se convierte en un punto de convergencia de diversas transiciones, no será posible conocer el valor de la salida. Por este motivo, no se permite que se produzca esta posibilidad. Cuando no hay transición presente, la puerta AND opera con su modelo de puerta tradicional. Formalmente, este comportamiento puede ser descrito por:

$$y_p = ((a_p \cap b_d \neq \phi) \vee (a_d \cap b_p \neq \phi)) \wedge (a_p \cap b_p = \phi) \tag{3.11}$$

$$y_d = ((a_d \cap b_d \neq \phi) \wedge (a_p \cup b_p = \phi)) \tag{3.12}$$

De forma similar, la puerta OR propaga una transición, si esta llega a una de sus entradas y la otra entrada posee un cero lógico. Si por el contrario, llega una transición a una entrada y la otra entrada posee un uno lógico, la salida no propaga la transición y además la salida se sitúa en un uno lógico. Cuando no hay transición, el comportamiento de la puerta OR es modelado con su modelo de puerta tradicional. Este modelo se puede describir mediante las ecuaciones:

$$y_p = ((a_p \cap \bar{b}_d \neq \phi) \vee (\bar{a}_d \cap b_p \neq \phi)) \wedge (a_p \cap b_p = \phi) \tag{3.13}$$

$$y_d = ((a_d \cup b_d \neq \phi) \wedge (a_p \cup b_p = \phi)) \tag{3.14}$$

En el caso del INV, puesto que siempre que hay una transición en su única entrada, este la propaga a su salida, por lo que su modelado es simple.

$$y_p = a_p \tag{3.15}$$

$$y_d = \bar{a}_d \tag{3.16}$$

Descripción del Circuito con MILP

Programación Lineal Mixta Entera

El comportamiento de una puerta lógica combinacional puede ser modelado mediante una descripción MILP como se describe en la referencia [56]. El único requisito que existe sobre dicha formulación es que los valores de entrada se

[56] Zeng Zhihong, P. Kalla, and M. Ciesielski. LPSAT: A Unified Approach to RTL Satisfiability. *Proc. on Design, Automation and Test in Europe*, pages 398–402, March 2001.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

mantengan dentro del margen $[0, 1]$. Ello implica que las variables que representan las entradas primarias se definan como variables Booleanas dentro del problema MILP.

El comportamiento de una puerta AND de dos entradas como la presentada en las ecuaciones 3.11 y 3.12 se define mediante cláusulas MILP con:

$$
\begin{aligned}
y_d &< a_d && (3.17)\\
y_d &< b_d && (3.18)\\
y_d &> a_d + b_d - 1 && (3.19)
\end{aligned}
$$

$$
\begin{aligned}
y'_p &< b_p && (3.20)\\
y'_p &< a_d && (3.21)\\
y'_p &> a_d + b_p - 1 && (3.22)
\end{aligned}
$$

$$
\begin{aligned}
y''_p &< a_p && (3.23)\\
y''_p &< b_d && (3.24)\\
y''_p &> b_d + a_p - 1 && (3.25)
\end{aligned}
$$

$$
\begin{aligned}
y_p &= y'_p + y''_p && (3.26)\\
1 &> a_p + b_p && (3.27)
\end{aligned}
$$

El bit y_p modela el bit de propagación y el bit y_d modela la presencia de un dato Booleano. Las ecuaciones 3.17–3.19 modelan el comportamiento tradicional de la puerta AND, es decir, el bit de datos de la ecuación 3.12.

Codificación de Dato y de Propagación

A su vez, la variable y'_p modela la llegada de una transición al terminal B de la puerta AND de dos entradas en presencia de un uno lógico en el terminal A de dicha puerta (véase las ecuaciones 3.20–3.22). De manera similar, con las ecuaciones lineales 3.23–3.25, la variable y''_p identifica cuando se ha producido la propagación de una transición llegada al terminal A de la puerta AND mientras existiendo un uno lógico en el terminal B.

Mediante la ecuación 3.26 se determina cuando hay propagación. Finalmente, la ecuación 3.27 evita que se propague más de una transición de forma simultánea en los terminales de entrada A y B de la puerta AND.

La formulación en MILP del modelo de la puerta OR de dos entradas descrito

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

en las ecuaciones 3.13 y 3.14 es:

$$y_d > a_d \quad (3.28)$$
$$y_d > b_d \quad (3.29)$$
$$y_d < a_d + b_d \quad (3.30)$$

$$y'_p < b_p \quad (3.31)$$
$$y'_p < a_d \quad (3.32)$$
$$y'_p > a_d + b_p - 1 \quad (3.33)$$

$$y''_p < a_p \quad (3.34)$$
$$y''_p < b_d \quad (3.35)$$
$$y''_p > b_d + a_p - 1 \quad (3.36)$$

$$y_p = y'_p + y''_p \quad (3.37)$$
$$1 > a_p + b_p \quad (3.38)$$

Las ecuaciones 3.28–3.30 implementan el comportamiento tradicional de la puerta OR sobre la variable y_d. La variable temporal y'_p, mediante las ecuaciones 3.31–3.33, modela la presencia de una transición propagada hasta el terminal B, mientras hay un cero lógico en el terminal A. Igualmente, y''_p determina si hay una transición propagada en el terminal A y un cero lógico en el terminal B con las ecuaciones 3.34–3.36.

Modelado del bit de Datos y Propagación

En la ecuación 3.37 se conforma la propagación de una transición presente en alguna de las entradas de la puerta OR que representa con una combinación lineal de las variables y'_p y y''_p. Para finalizar, mediante la ecuación 3.38 se inhibe la posibilidad de la presencia de más de una transición simultánea en las entradas de la puerta OR.

El comportamiento descrito en las ecuaciones 3.15 y 3.16 de una puerta INV se describe en MILP como:

$$y_d = 1 - a_d \quad (3.39)$$
$$y_d = a_p \quad (3.40)$$

Optimización mediante MILP

Descripción del Problema en MILP

Dada una red de Booleana $\overrightarrow{G}$ que representa un circuito de lógica combinacional, obtenido mediante un procedimiento de síntesis. Donde, $\overrightarrow{G}$ es una descripción estructural, empleando AND de dos entradas, OR de dos entradas e INV. Este circuito puede ser expresado como un problema MILP empleando los modelos descritos mediante las ecuaciones 3.17–3.40. La función objetivo a optimizar es maximizar el número de rutas ejercitadas.

Es de esperar que cada puerta que pertenece a alguna ruta ejercitada posea su bit de propagación activo. Y motivado porque el modelo de puerta desarrollado en esta Tesis Doctoral elimina la convergencia de múltiples transiciones propagadas a través de una misma puerta, cuando se determina que hay una ruta ejercitada es porque hay asegurado una ruta entre una entrada y una salida. Esa ruta entre entradas y salidas no posee puerta alguna en común con otra ruta. Ello significa que,

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

de forma unívoca, el número de rutas ejercitadas se corresponde con el número de salidas primarias que tengan los bits activos. De las entradas no se puede decir nada pues una única transición presente en una entrada y el patrón de verificación adecuado puede producir una transición en múltiples salidas. Es reseñable indicar que el modelo prohíbe la existencia de una convergencia de propagaciones de transición, pero sí permite la existencia de bifurcaciones, con lo cual de una única fuente, se puede obtener a la salida la ejercitación de múltiples rutas.

Como ya se puede intuir, la función objetivo del problema MILP es obtener el máximo número de salidas primarias propagando o, su equivalente; maximizar el número de bits de propagación que pertenecen a las salidas primarias del circuito en estudio.

Función Objetivo

Resultados Experimentales

Al igual que se realizó con el procedimiento basado en programación evolutiva, se han procesado un conjunto grande de circuitos del banco de pruebas MCNC'91; que incluyen tanto circuitos combinacionales de dos niveles como los circuitos combinacionales multinivel. En este case se ha de determinar los vectores que maximizan la verificación funcional de los circuitos basada en la cobertura de rutas [73],[60].

Los resultados han sido evaluados en un servidor Sun–Fire 280R dotado de dos CPUs a 900 MHz con 4 $GBytes$ de memoria RAM, ejecutando SunOS v5.8. El software que implementa esta metodología basada en Programación Lineal Mixta Entera, y en particular el algoritmo presentado en la figura 3.9, ha sido desarrollado en lenguaje C, empleando el software libre GLPK v3.2.3 como núcleo para la resolución MILP.

GLPK con núcleo de Resolución MILP

Cada circuito del banco de pruebas MCNC'91 ha sido preprocesado por la herramienta de síntesis lógica MISII y mapeado a una librería de puertas lógicas — NAND, NOR, INV — con mínima área. Se ha determinado el vector de máxima cobertura empleando el mismo algoritmo de búsqueda exhaustiva (fuerza bruta) del apartado 3.3.5. De igual forma, se ha obtenido mediante la metodología basada en MILP, el vector de máxima cobertura para la verificación de los circuitos con la métrica de cobertura de rutas.

Las tablas 3.3, 3.4 y 3.5 muestran una comparativa, en términos de tiempo de computo de CPU, para obtener el primer vector funcional que maximiza la cobertura de rutas de datos de los circuitos combinacionales en estudio. Las cinco primeras columnas muestran el nombre, el número de puertas, entradas primarias, salidas primarias y rutas del circuito MCNC'91 en estudio. En la sexta columna se indica el número de rutas ejercitadas con el vector funcional de máxima cobertura, para la métrica de cobertura de ruta de datos. A continuación, en las columnas séptima y octava se presenta el número de variables necesarias en la formulación del problema MILP y el número de ellas que son de tipo binario. El tiempo de CPU requerido para obtener el resultado de la optimización mediante la metodo-

[73] J. Sosa, J. A. Montiel-Nelson, H. Navarro, V. de Armas, and R. Sarmiento. Functional Vector Generation for Combinational Circuits Based on Data Path Coverage Metric and Mixed Integer Linear Programming. In *5th International Society for Quality Electronic Design*, November 2004.

[60] J. Sosa, J. A. Montiel-Nelson, H. Navarro, and José C. García. Functional Vector Generation for Maximum Data Path Coverage using Mixed Integer Linear Programming. In *XVIII Conference on Design of Circuits and Integrated Systems*, November 2005.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

Tabla 3.3: RESULTADOS EXPERIMENTALES Y COMPARATIVAS DE LA GENERACIÓN DE VECTORES FUNCIONALES PARA LA VERIFICACIÓN MEDIANTE MILP Y COBERTURA DE RUTAS DE DATOS

Circuito MCNC'91					Máxima Cobertura Alcanzada	Metodología MILP			Fuerza Bruta			Relación Tiempo CPU
						Variables		Tiempo	Tiempo	Combinaciones	Total	
Nombre	Puertas	Entradas	Salidas	Rutas		Máximas	Binarias	CPU	CPU	Necesarias	CPU	
5xp1	122	7	10	172	9	234	14	1'44	6'00E-05	1'37E+19	8'19E+14	5'69E+14
9sym	239	9	1	297	1	434	18	3'31	1'40E-04	9'70E+01	1'36E-02	4'10E-03
9symml	235	9	1	286	1	416	18	3'03	1'20E-04	1'34E+02	1'61E-02	5'31E-03
C17	776	5	2	9	2	28	10	0'02	1'30E-04	2'00E+01	2'60E-03	1'30E-01
C2670	1.053	233	140	47.295	11	2.328	466	790'20	6'70E-04	7'13E+47	4'78E+44	6'04E+41
C7552	2.697	207	108	565.784	9	4.270	414	1.105'75	1'06E-03	2'32E+22	2'46E+19	2'23E+16
C880	497	60	26	22.151	25	868	120	504'91	2'70E-04	1'15E+51	3'12E+47	6'17E+44
Z5xp1	635	7	10	2.866	9	846	14	121'79	1'90E-04	1'81E+30	3'44E+26	2'83E+24
Z9sym	232	9	1	292	1	414	18	3'06	9'00E-05	5'40E+01	4'86E-03	1'59E-03
alu2	559	10	6	39.874	6	808	20	6'10	2'00E-04	2'23E+24	4'46E+20	7'30E+19
alu4	1.110	14	8	402.207	8	1.514	28	245'63	3'10E-04	4'92E+42	1'53E+39	6'21E+36
apex1	1.312	45	45	8.377	16	1.822	90	844'62	4'70E-04	9'06E+57	4'26E+54	5'04E+51
apex2	482	39	3	914	3	818	78	13'76	2'00E-04	1'01E+08	2'01E+04	1'46E+03
apex3	1.762	54	50	7.893	10	2.450	108	1.494'39	5'00E-04	1'96E+38	9'79E+34	6'55E+31
apex4	2.718	9	19	15.171	15	3.416	18	2.004'89	7'40E-04	1'22E+51	9'06E+47	4'52E+44
apex5	1.067	117	88	3.362	84	1.900	234	797'05	5'10E-04	> 1'00E+99	> 1'00E+99	> 1'00E+99
apex6	974	135	99	1.783	96	1.824	270	220'64	5'70E-04	> 1'00E+99	> 1'00E+99	> 1'00E+99
apex7	299	49	37	198	35	604	98	11'44	2'10E-04	9'43E+88	1'98E+85	1'73E+84
b12	105	15	9	126	7	220	30	1'69	1'20E-04	1'18E+10	1'42E+06	8'40E+05
b1	11	3	4	12	3	30	6	0'02	1'30E-04	7'20E+02	9'36E-02	4'68E+00
b9	142	41	21	214	18	324	82	1'62	1'10E-04	7'94E+33	8'74E+29	5'39E+29
bw	211	5	28	388	15	366	10	3'51	1'40E-04	1'52E+31	2'13E+27	6'06E+26
c8	149	28	18	189	18	300	56	0'43	1'10E-04	2'33E+33	2'56E+29	5'95E+29
cc	82	21	20	109	17	188	42	0'31	1'70E-04	2'64E+31	4'49E+27	1'45E+28
cht	192	47	36	235	36	408	94	0'93	2'20E-04	1'36E+57	2'99E+53	3'21E+53
clip	130	9	5	234	5	228	18	0'75	1'20E-04	5'80E+09	6'96E+05	9'28E+05
cm138a	32	6	8	48	2	68	12	0'07	8'00E-05	5'00E+00	4'00E-04	5'71E-03
cm150a	47	21	1	55	1	128	42	0'05	1'00E-04	5'40E+01	5'40E-03	1'08E-01
cm151a	27	12	2	46	2	72	24	0'03	1'30E-04	1'98E+03	2'57E-01	8'58E+00
cm152a	24	11	1	22	1	66	22	0'03	8'00E-05	8'00E+00	6'40E-04	2'13E-02
cm162a	56	14	5	83	5	110	28	0'06	7'00E-05	4'53E+08	3'17E+04	5'29E+05
cm163a	47	16	5	57	5	112	32	0'05	1'00E-04	3'68E+07	3'68E+03	7'37E+04
cm42a	34	4	10	44	2	72	8	0'09	1'00E-04	4'00E+00	4'00E-04	4'44E-03
cm82a	29	5	3	45	3	56	10	0'03	1'20E-04	3'15E+04	3'78E+00	1'26E+02
cm85a	56	11	3	91	3	108	22	0'08	1'00E-04	1'22E+03	1'22E-01	1'52E+00
cmb	56	16	4	112	2	122	32	0'23	1'20E-04	2'50E+03	3'00E-01	1'30E+00

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

Tabla 3.4: RESULTADOS EXPERIMENTALES Y COMPARATIVAS DE LA GENERACIÓN DE VECTORES FUNCIONALES PARA LA VERIFICACIÓN MEDIANTE MILP Y COBERTURA DE RUTAS DE DATOS (CONTINUACIÓN I)

Circuito MCNC'91					Máxima Cobertura	Metodología MILP			Fuerza Bruta			Relación
						Variables		Tiempo	Tiempo	Combinaciones	Total	Tiempo
Nombre	Puertas	Entradas	Salidas	Rutas	Alcanzada	Máximas	Binarias	CPU	CPU	Necesarias	CPU	CPU
comp	212	32	3	1.536	2	356	64	7'11	1'40E-04	1'01E+06	1'42E+02	1'99E+01
con1	23	7	2	19	2	52	14	0'04	9'00E-05	1'26E+02	1'13E-02	2'84E-01
cordic	23	23	2	140	2	164	46	0'08	1'00E-04	7'02E+02	7'02E-02	8'78E-01
count	77	35	16	368	16	354	70	0'56	1'60E-04	1'15E+31	1'85E+27	3'30E+27
cu	59	14	11	96	3	136	28	0'38	9'00E-04	2'90E+10	2'61E+06	6'86E+06
dalu	1462	75	16	1.157.487	16	2.112	150	311'41	4'60E-04	1'34E+90	6'18E+86	1'98E+84
decod	32	5	16	80	2	98	10	0'22	1'10E-04	1'76E+03	1'94E-01	8'80E-01
duke2	4704	22	29	2.051	13	776	44	394'80	2'20E-04	6'85E+38	1'51E+35	3'82E+32
ex4	459	128	28	577	14	1.126	168	12'04	3'20E-04	1'05E+32	3'35E+28	2'78E+27
example2	373	85	66	1.327	51	836	170	370'66	2'80E-04	8'89E+92	2'49E+89	6'72E+86
f51m	137	8	8	188	8	262	16	1'19	1'20E-04	4'19E+15	5'03E+11	4'23E+11
frg1	142	28	3	148	3	286	56	0'25	2'10E-04	1'32E+02	2'77E-02	1'11E-01
frg2	1144	143	139	6.075	13	2.050	286	42'76	6'00E-04	4'83E+39	2'90E+36	6'77E+34
i1	55	25	16	72	13	156	50	0'13	1'80E-04	8'07E+21	1'45E+18	1'12E+19
i2	217	201	1	228	1	822	402	1'55	4'50E-04	2'27E+02	1'02E-01	6'59E-02
i3	152	132	6	132	6	528	264	0'67	3'60E-04	4'56E+06	1'64E+03	2'45E+03
i5	198	133	66	672	66	662	266	0'80	4'00E-04	$>$ 1'00E+99	$>$ 1'00E+99	$>$ 1'00E+99
i6	559	138	67	778	67	1.182	276	9'50	3'60E-04	$>$ 1'00E+99	$>$ 1'00E+99	$>$ 1'00E+99
i7	755	199	67	1.003	67	1.558	398	11'79	5'90E-04	$>$ 1'00E+99	$>$ 1'00E+99	$>$ 1'00E+99
i8	1331	133	81	9.811	81	2.262	266	51'15	5'40E-04	$>$ 1'00E+99	$>$ 1'00E+99	$>$ 1'00E+99
i9	792	88	63	19.919	63	1.382	176	23'32	3'80E-04	$>$ 1'00E+99	$>$ 1'00E+99	$>$ 1'00E+99
inc	208	7	9	520	6	310	14	8'08	1'00E-04	1'00E+15	1'00E+11	1'24E+10
k2	1313	45	45	9.127	16	1.796	90	330'13	3'90E-04	3'44E+58	1'34E+55	4'07E+52
lal	122	26	19	197	17	240	52	0'25	1'10E-04	2'32E+33	2'55E+29	1'02E+30
majority	13	5	1	10	1	30	10	0'01	9'00E-05	3'00E+00	2'70E-04	2'70E-02
misex1	70	8	7	119	5	128	16	0'45	1'00E-04	3'27E+09	3'27E+05	7'28E+05
misex2	122	25	18	174	5	254	50	21'94	1'30E-04	1'70E+09	2'21E+05	1'01E+04
misex3	864	14	14	2.792	11	1.380	28	451'35	2'80E-04	2'68E+31	7'51E+27	1'66E+25
misex3c	622	14	14	1.163	10	1.056	28	227'94	2'20E-04	1'40E+30	3'08E+26	1'35E+24
mux	55	21	1	55	1	130	42	0'06	1'40E-04	5'40E+01	7'56E-03	1'26E-01
my_adder	268	33	17	1.310.505	17	420	66	1'72	2'20E-04	2'10E+75	4'62E+71	2'69E+71
o64	131	130	1	130	1	520	260	0'61	2'70E-04	6'90E+01	1'86E-02	3'05E-02
pair	1991	173	137	15.513	45	3.412	346	794'10	7'80E-04	$>$ 1'00E+99	$>$ 1'00E+99	$>$ 1'00E+99
parity	69	16	1	352	1	124	32	0'09	1'20E-04	2'04E+02	2'45E-02	2'72E-01
pcle	84	19	9	122	9	166	38	0'13	1'70E-04	2'37E+11	4'03E+07	3'10E+08
pcler8	126	27	17	234	16	230	54	0'16	1'80E-04	4'72E+25	8'49E+21	5'31E+22

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

Tabla 3.5: RESULTADOS EXPERIMENTALES Y COMPARATIVAS DE LA GENERACIÓN DE VECTORES FUNCIONALES PARA LA VERIFICACIÓN MEDIANTE MILP Y COBERTURA DE RUTAS DE DATOS (CONTINUACIÓN II)

Circuito MCNC'91					Máxima Cobertura Alcanzada	Metodología MILP			Fuerza Bruta		Relación Tiempo CPU
						Variables		Tiempo CPU	Tiempo CPU	Combinaciones Necesarias	Total CPU	
Nombre	Puertas	Entradas	Salidas	Rutas		Máximas	Binarias					
pm1	55	16	13	70	9	134	32	0'06	9'00E-05	8'00E+00	7'20E-04	1'20E-02
rd53	58	5	3	71	3	116	10	0'14	1'00E-04	2'10E+01	2'10E-03	1'50E-02
rd73	147	7	3	235	3	250	14	0'63	1'10E-04	2'01E+06	2'21E+02	3'51E+02
rd84	285	8	4	949	4	424	16	1'56	1'90E-04	1'78E+11	3'39E+07	2'17E+07
rot	808	135	107	8.695	29	1.634	270	0'01	5'80E-04	2'83E+95	1'64E+92	3'13E+90
sao2	164	10	4	255	4	322	20	1'99	1'50E-04	2'98E+08	4'47E+04	2'24E+04
sct	90	19	15	150	13	184	38	0'20	8'00E-05	2'63E+19	2'10E+15	1'05E+16
seq	1952	41	35	4.947	17	2.882	82	1120'72	5'30E-04	6'81E+45	3'61E+42	3'22E+39
squar5	180	5	8	409	6	284	10	3'38	1'10E-04	3'59E+13	3'95E+09	1'17E+09
t481	563	16	1	1.077	1	850	32	17'12	1'90E-04	7'49E+02	1'42E-01	8'31E-03
table3	1019	14	14	6.747	11	1.540	28	1504'04	3'60E-04	1'95E+41	7'00E+37	4'66E+34
table5	1007	17	14	6.180	12	1.570	34	278'25	3'00E-04	1'46E+37	4'38E+33	1'58E+31
tcon	40	17	16	40	16	114	34	0'04	1'10E-04	7'89E+23	8'68E+19	2'17E+21
term1	299	34	10	601	10	516	68	2'57	2'90E-04	5'98E+26	1'73E+23	6'75E+22
too_large	482	38	3	941	3	82	76	0'02	2'10E-04	8'00E+07	1'68E+04	8'40E+05
ttt2	252	24	21	348	19	446	48	6'59	1'80E-04	2'50E+41	4'49E+37	6'82E+36
unreg	119	36	16	160	16	272	72	0'40	1'60E-04	4'68E+33	7'49E+29	1'87E+30
vda	817	17	39	4.768	19	1.344	34	854'95	2'80E-04	2'04E+61	5'72E+57	6'69E+54
vg2	113	25	8	246	7	232	50	1'32	1'00E-04	2'02E+11	2'02E+07	1'53E+07
x1	412	51	35	438	26	748	102	158'78	2'20E-04	2'95E+02	6'49E-02	4'09E-04
x2	78	10	7	81	4	142	20	0'42	1'00E-04	3'98E+05	3'98E+01	9'49E+01
xor5	18	5	1	46	1	36	10	0'02	8'00E-05	4'10E+01	3'28E-03	1'64E-01

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

logía MILP es mostrado en la columna novena. Por otro lado, el tiempo necesario para realizar la comprobación de la validez de un conjunto de rutas (satisfabilidad) con el método de búsqueda exhaustiva es indicado en la columna décima. A continuación, en la décimo primera columna se muestra el número de combinaciones sin repetición que hay que realizar para encontrar el vector de máxima cobertura, según la métrica de cobertura de rutas de datos. En la columna décimo segunda se presenta el tiempo estimado para que el algoritmo de búsqueda exhaustiva encuentre el vector de máxima cobertura. Y la última columna indica la relación en tiempo de CPU entre la solución de búsqueda exhaustiva y la metodología basada en MILP.

Mejora del Tiempo de Computación

Recodificación del Problema MILP

Hasta ahora se ha planteado una metodología basada en Programación Lineal Mixta Entera (MILP), que permite abordar el problema de la generación de vectores de máxima cobertura. En ningún momento se ha abordado la optimización del problema MILP planteado, o incluso la utilización de un lenguaje de programación como PROLOG para solventar este problema. En este sentido, la figura 3.15 presenta cuatro versiones distintas de la formulación de una puerta NOR, donde tres de ellas están basadas en Programación Lineal Mixta Entera y la cuarta solución descrita en PROLOG.

La figura 3.15(a) presenta el modelo introducido con anterioridad, donde hacen falta 13 ecuaciones y un total de 40 literales. La figura 3.15(b) muestra en cambio si se consideran valores enteros a las variables y' y y''. En ese caso hacen falta 11 ecuaciones con 37 literales. En la figura 3.15(c) se presenta el modelo comprimido de la NOR. La compresión surge de la codificación de los estados de la puerta en dos bits y, no en asignar un bit a datos y otro a la condición de propagación. El número de ecuaciones en este caso es de 10 y 34 literales. Finalmente, la figura 3.15(d) presenta el modelo de puerta NOR descrito en PROLOG. En este caso sólo son necesarias 3 ecuaciones que poseen 11 literales en total.

Tabla 3.6: COMPARATIVAS ENTRE DIVERSAS FORMAS DE CODIFICAR EL PROBLEMA CON MILP Y PROLOG

Circuito MCNC'91				Modelo MILP (CPU)			PROLOG (CPU)	
Nombre	Puertas	Entradas	Salidas	Normal	Medio	Comprimido	Compilar	Ejecución
b12	105	15	9	101'41	63'09	5'96	410	0'03
5xp1	122	7	10	19'23	12'86	6'91	420	0'03
alu2	559	10	6	386'80	303'79	110'42	710	0'05
t481	563	16	1	171'34	69'77	139'58	770	0'03

Con el objeto de evaluar las prestaciones de dichas codificaciones, se han sometido las mismas a la obtención del vector de mayor cobertura para cuatro circuitos combinacionales de distinta complejidad, medido en número de puertas. Los circuitos seleccionados del banco de pruebas MCNC son apex5, b12, t481 y alu2. Estos circuitos fueron mapeados a mínima área con una librería de puertas — NOR, NAND e INV — con MISII y ejecutados en un Pentium IV a 1'8 GHz, 128 $MBytes$ de RAM, y ejecutando Linux Debian v2.1. La tabla 3.6 presenta los resultados de dicha comparativa. Los tiempos de la tabla están expresados en milisegundos. La primera columna indica el circuito MCNC empleado. Las segunda, tercera y cuarta columna presentan la complejidad del circuito medida en el número de puertas, entradas y salidas del circuito, respectivamente. La columna

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

$$
\begin{aligned}
y_d + a_d &< 1\\
y_d + b_d &< 1\\
y_d + a_d + b_d &> 1\\
y' - a_p &< 0\\
y' + b_d &< 1\\
y' + d_d - a_p &> 0\\
y'' - b_p &< 0\\
y'' + a_d &< 1\\
y'' + a_d - b_p &> 0\\
y_p - y' &> 0\\
y_p - y'' &> 0\\
y_p - y' - y'' &< 0\\
a_p + b_p &< 1
\end{aligned}
$$

(a)

$$
\begin{aligned}
y_d + a_d &< 1\\
y_d + b_d &< 1\\
y_d + a_d + b_d &> 1\\
y' - a_p &< 0\\
y' + b_d &< 1\\
y' + d_d - a_p &> 0\\
y'' - b_p &< 0\\
y'' + a_d &< 1\\
y'' + a_d - b_p &> 0\\
y_p - y' - y'' &= 0\\
a_p + b_p &< 1
\end{aligned}
$$

(b)

$$
\begin{aligned}
y_0 - a_1 &< 0\\
y_0 - b_1 &< 0\\
y_0 - a_1 - b_1 &> -1\\
y_1 - a_0 &> 0\\
y_1 - b_0 &> 0\\
y_1 - a_0 - b_0 &< 0\\
y_1 + y_1 &> 1\\
a_0 + a_1 + b_0 + b_1 &< 3\\
a_0 + a_1 &> 1\\
b_0 + b_1 &> 0
\end{aligned}
$$

(c)

$$
\begin{aligned}
a_p\# \wedge b_d\# \vee a_d\# \wedge b_p &<=> y_p\\
a_d \vee b_d &<=> y_d\\
a_p + b_p &\ \#=< 1
\end{aligned}
$$

(d)

Figura 3.15: Diversos modelos de descripción para la puerta NOR: a) completo, b) suponiendo y' y y'' variables enteras, c) con las entradas codificadas, y d) PROLOG.

quinta presenta el tiempo necesario para obtener el vector de máxima cobertura empleando el modelo inicial presentado en este trabajo. La sexta columna muestra, en cambio, el tiempo de CPU necesario para obtener dicho vector con la ligera modificación presentada en la figura 3.15(b). La evaluación del modelo que emplea codificación de estados (véase la figura 3.15(c)) se presenta en la columna antepenúltima. Las columnas penúltima y última muestran el tiempo necesario para compilar y obtener el vector de máxima cobertura con PROLOG.

Conclusiones

La complejidad computacional del problema de optimización de la generación de vectores funcionales basados en la cobertura de rutas en circuitos combinacionales crece exponencialmente con el número de puertas [42]. En este trabajo se propone una metodología eficiente para determinar los vectores funcionales que ejercitan rutas y maximizan la cobertura basada en rutas. Se ha empleado Programación Lineal Mixta Entera (MILP) para implementar la metodología propues-

[42] J. Sosa, J. A. Montiel-Nelson, H. Navarro, José C. García, and R. Sarmiento. *Circuit Path Coverage using Genetic Algorithms*, pages 51–52. Evolutionary Methods for Design optimization and Control Applications to Industrial and Societal Problems. International Center for Numerical Methods in Engineering, 2003.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

ta [73]. Mediante una comparativa con un método de fuerza bruta se ha demostrado que la metodología propuesta obtiene vectores funcionales para optimizar la cobertura basada en rutas de forma eficiente (en términos de tiempo de CPU). La implementación de la metodología propuesta es siempre mejor que la implementación basada en fuerza bruta cuando el número de entradas primarias y salidas primarias del circuito es superior a ocho y tres, respectivamente. La mejora en términos de tiempo de CPU de la solución basada en MILP es varios órdenes de magnitud superior que la solución basada en fuerza bruta. Esta reducción permite comprobar circuitos de lógica combinacional en un tiempo de CPU práctico, frente a la aproximación basada en una simulación extensa [60].

Por otro lado, la optimización de las ecuaciones planteadas en MILP mediante la codificación de las entradas permite reducir el tiempo requerido para resolver el problema. A medida que el circuito crece en complejidad, dicha ventaja se incrementa. Además, el empleo de PROLOG par resolver los problemas propuestos es inoperativo, puesto que la etapa de compilación del problema supera en gran medida el tiempo requerido por el MILP.

La metodología presentada posee obviamente una limitación en cuanto al crecimiento del circuito en términos de su número de puertas, a medida que éste crece, la generación de vectores toma más tiempo de CPU.

3.4. Cobertura Basada en Potencia de Consumo

Debido al auge las aplicaciones donde la potencia de consumo es un factor crítico — multimedia y o móviles — la estimación de dicho factor ha tomado la relevancia que posee el ahorro de área.

Métrica de Cobertura

Existe un amplio espectro de aportaciones en este campo, las cuales se centran en diversos aspectos tanto en el campo del análisis como de la estimación[11] [74]. Las aportaciones en alto nivel se centran en la estimación de la potencia de consumo mediante macromodelos definiendo conceptos como granularidad y parámetros de actividad circuital [75]–[77]. También se han propuesto diversas herra-

[11] El análisis consiste en aplicar un conjunto de modelos a un diseño existente implementado en cualquier nivel de descripción. La estimación por el contrario evalúa el circuito desconociendo a priori algún dato del mismo, por ejemplo parte de la estructura o los patrones de entrada.

[73] J. Sosa, J. A. Montiel-Nelson, H. Navarro, V. de Armas, and R. Sarmiento. Functional Vector Generation for Combinational Circuits Based on Data Path Coverage Metric and Mixed Integer Linear Programming. In *5th International Society for Quality Electronic Design*, November 2004.

[60] J. Sosa, J. A. Montiel-Nelson, H. Navarro, and José C. García. Functional Vector Generation for Maximum Data Path Coverage using Mixed Integer Linear Programming. In *XVIII Conference on Design of Circuits and Integrated Systems*, November 2005.

[74] Christian Piguet. *Low–Power Electronics Design*. CRC Press, July 2004.

[75] M. Nemani and F. Najm. Towards a High–Level Power Estimation Capability. *IEEE Trans. on Computer–Aided Design*, 15(6):588–598, 1996.

[76] D. Marculescu, R. Marculescu, and M. Pedram. Information Theoretic Measures for Power Analysis. *IEEE Trans. on Computer–Aided Design*, 15(6):599–609, June 1996.

[77] M. Nermani and F. Najm. High–Level Area and Power Estimation for VLSI Circuits. *IEEE Trans. on Computer–Aided Design*, 18(6):697–713, June 1999.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

mientas. De entre las comerciales destacar Synopsys [78][79] o Magma [80] y de entre las desarrolladas en el ámbito universitario MIS [70] o VIS [53].

La mayoría de las técnicas presentadas emplean aproximaciones basadas en métodos probabilísticos y estadísticos [81]–[83]. Además, las soluciones propuestas emplean aproximaciones medias. En otras palabras, la estimación de la potencia de consumo total se realiza en base a un modelo de puerta para estimar la actividad media y a la distribución estadística de señales presentes a la entrada de la red Booleana. El principal objetivo de esas técnicas es obtener la potencia de consumo media de un circuito VLSI en condiciones típicas. Por otro lado, la estimación de la potencia de consumo media es un problema distinto a la estimación del peor caso de potencia de consumo [84]. El peor caso de potencia de consumo ha de ser considerado cuando sea un factor crítico del diseño, y haya una alta probabilidad de que exista. No obstante, es importante someter el sistema a esta condición a fin de conocer cómo se degradan sus prestaciones, entre otros objetivos.

El problema del peor caso de consumo de potencia es equivalente a obtener la secuencia de estímulos de entrada que maximicen la potencia de consumo total. En términos de complejidad computacional, la naturaleza de la búsqueda de los estímulos de entrada que maximicen la potencia de consumo es definido como un problema NP–completo.

Complejidad del Problema

En general, existen tres tendencias claras en los aportaciones previas; procesos estocásticos, simulaciones Monte Carlo y los métodos basados en fractales. El grupo de aportaciones basadas en procesos estocásticos reutiliza el aparato estadístico desarrollado para estimar valores medios en el cálculo de las potencias máximas de disipación [84]. Por el contrario las técnicas basadas en simulaciones Monte Carlo obtiene la potencia máxima del circuito aplicando diversas pertur-

[78] A. J. de Geus. Logic Synthesis Speeds ASIC Design. *IEEE Spectrum*, 26(8):27–31, August 1989.

[79] B. Chen and I. Nedelchev. Power Compiler: A Gate–Level Power Optimization and Synthesis System. *Computer Design: VLSI in Computers and Processors*, pages 74–79, October 1997.

[80] Xiaoyong Tang, Hai Zhou, and Prith Banerjee. Leakage Power Optimization With Dual–Vth Library in High–Level Synthesis. *42nd IEEE Design Automation Conference*, pages 202–207, June 2005.

[70] R.K. Brayton, R. Rudell, A.L. Sangiovanni-Vincentelli, and A. Wang. MIS: A Multiple Level Logic Optimization System. *IEEE Trans. Comput.–Aided Des. Integr. Circuits Syst.*, 6(11):1062–1081, 1987.

[53] R. K. Brayton, G. D. Hachtel, A. Sangiovanni-Vincentelli, F. Somenzi, A. Aziz, S. T. Cheng, S. Edwards, S. Khatri, Y. Kukimoto, A. Pardo, S. Qadeer, R. K. Ranjan, S. Sarwary, T. R. Shiple, G. Swamy, and T. Villa. VIS: a System for Verification and Synthesis. In Rajeev Alur and Thomas A. Henzinger, editors, *Proceedings of the Eighth International Conference on Computer Aided Verification CAV*, volume 1102, pages 428–432, New Brunswick, NJ, USA, 1996. Springer Verlag.

[81] F. Najm. Transition Density: A New Measure of Activity in Digital Circuits. *IEEE Trans. Computer–Aided Design*, 12(2):310–323, February 1993.

[82] Z. Chen, K. Roy, and T. Chuo. Efficient Statistical Approach to Estimate Power Considering Uncertain Properties of Primary Inputs. *IEEE Tran. on Very Large Scale Integration (VLSI) Systems*, 6(3):484–492, February 1993.

[83] S. Gupta and F. Najm. Power Modeling for High–Level Power Estimation. *IEEE Tran. on Very Large Scale Integration (VLSI) Systems*, 8(1):18–29, February 2000.

[84] C. Wang and K. Roy. Maximum Power Estimation for CMOS Circuits using Deterministic and Statistical Approaches. *IEEE Tran. on Very Large Scale Integration (VLSI) Systems*, 6(1):130–140, February 1999.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

baciones a los vectores de entrada en cada iteración de la simulación Monte Carlo [85]. También, hay publicaciones recientes que estiman la potencia máxima de consumo basándose estrictamente en simulación o en fractales [86]. No obstante, estas últimas aportaciones se centran en la compactación de vectores más que en la estimación de potencia de consumo.

3.4.1. Modelo de Potencia de Consumo CMOS

La potencia de consumo en los circuitos CMOS se compone de dos términos; la potencia estática y la potencia dinámica [87].

La potencia estática es debida a las corrientes de fuga de los diodos polarizados en inversa y es una fuente constante de disipación de potencia. El consumo dinámico a su vez se divide en dos componentes. La potencia de cortocircuito y la potencia de carga y descarga. La primera se produce por la llamada corriente de cortocircuito que se genera cuando conducen las redes NMOS y PMOS al mismo tiempo. Esto es, hay un camino directo desde V_{DD} a tierra (GND). La segunda componente de la potencia dinámica es debida a la carga y descarga de las capacidades.

Potencia Estática y Dinámica

Se ha demostrado que aumentar la actividad de conmutación del circuito conlleva a un aumento de la potencia dinámica [81]. La actividad de conmutación mide el número de transiciones realizados en un circuito durante un ciclo de reloj. Obtener el máximo consumo de potencia es equivalente al problema de buscar la secuencia de estímulos de entrada que maximicen el número total de transiciones (actividad de conmutación) en un circuito.

La aportación presentada en este capítulo es independiente del modelo de potencia de consumo. En el capítulo 5 se discutirá en detalle diversos modelos de potencia, todos son perfectamente válidos para aplicarlos a la metodología propuesta.

3.4.2. Función de Coste y Codificación

Como ya se ha introducido en el apartado previo, el principal objetivo es obtener la secuencia de estímulos de entrada que maximicen la actividad de conmutación de un circuito dado. En este sentido, y con el objetivo de medir la actividad de transición, se ha desarrollado un estimador de actividad. Dado un circuito, el estimador de actividad mide el número de transiciones que aparecen cuando dos vectores de entrada se introducen secuencialmente. Es evidente que la función de coste de la solución basada en Algoritmos Genéticos es el propio estimador de transiciones.

Estimador de Actividad

El vector de entrada de un circuito de lógica combinacional especifica los valores de entrada a la red Booleana de un circuito. Además, aplicado el vector de

[85] N. Evmorfopoulos, G. Stamoulis, and J. Avaritsiotis. A Monte Carlo Approach for Maximum Power Estimation Based on Extreme Value Theory. *IEEE Tran. on Computer-Aided Design of Integrated Circuits and Systems*, 21(4):415–432, April 2002.

[86] R. Radjassamy and J. Carothers. Faster Power Estimation of CMOS Designs using Vector Compaction — a Fractal Approach. *IEEE Tran. on Systems, Man and Cybernetics, Part B*, 33(3), June 2003.

[87] K. Roy and S. Prasad. *Low–Power CMOS VLSI Circuit Design*. John Willey and Sons, Inc., New York, 1st edition, February 2000.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

entrada al circuito, se define al vector de salida y a los estados internos de cada puerta del circuito de lógica combinacional.

Una vez es aplicado un vector de entrada a un circuito de lógica combinacional, el estado de cada puerta en el circuito y su vector de salida permanecerá estable a partir de un cierto tiempo. Entonces, si se cambia el vector de entrada, se ha de esperar el tiempo de propagación de la red Booleana[12][88] para obtener, otra vez, un vector de salida estable y unos estados internos estables.

Durante el tiempo de propagación, la red Booleana propaga los valores del vector de entrada a través de las funciones Booleanas internas del circuito de lógica combinacional. El estimador de actividad de transición mide el número de transiciones producidas por el nuevo vector introducido basándose en el estado estable proporcionado por el vector de entrada anterior.

Origen de la Transición

Es también bien conocido que para que una transición presente en una entrada de un circuito de lógica combinacional aparezca en alguna de sus salidas es necesaria la coordinación de tres factores; la conectividad de la red Booleana, su funcionalidad y el retardo de cada puerta en el circuito de lógica combinacional. Este trabajo estudia en detalle los dos primeros factores. El retardo de puerta sin embargo es muy dependiente de la tecnología seleccionada.

En esta Tesis Doctoral se ha fijado un valor de retardo unitario en todas las puertas del circuito de lógica combinacional. No se considerará el error introducido por el empleo de este modelo simple de retardo dentro del estimador de actividad y ha sido ampliamente estudiado en la Tesis como recoge los capítulos 4 y 5.

Dada una puerta INV, con entrada A y salida Y, una transición $T_{y,t}$ surge el en periodo $(t-1,t]$ cuando su estimulo de entrada cambia el estado de la puerta. Formalmente[13]:

$$Y_{t-1} = \bar{A}_{t-1} \tag{3.41}$$

$$Y_t = \bar{A}_t \tag{3.42}$$

$$T_{y,t} = Y_t \oplus Y_{t-1} = \bar{A}_t \oplus \bar{A}_{t-1}, \tag{3.43}$$

Siendo $t-1$ es el instante cuando el nuevo vector A_t es aplicado a la entrada del inversor y, t es el tiempo cuando el vector de salida Y_t es estable.

Para el caso de una puerta AND, con entradas A y B y salida Y, una transición $T_{y,t}$ se define tal y como sigue:

$$Y_t = A_t \& B_t \tag{3.44}$$

$$Y_{t-1} = A_{t-1} \& B_{t-1} \tag{3.45}$$

$$T_{y,t} = Y_t \oplus Y_{t-1} = (A_t \& B_t) \oplus (A_{t-1} \& B_{t-1}) \tag{3.46}$$

De un modo similar, dada una puerta OR con entradas A y B y salida Y, una

[12] El tiempo de propagación de una red Booleana lo determina el retardo de propagación de su ruta crítica.

[13] $\bar{A}$ denota el valor negado de A, & la función Booleana and, $+$ la función lógica or, $\oplus$ la función Booleana xor.

[88] J.A. Montiel-Nelson, J. Sosa, H. Navarro, R. Sarmiento, and A. Nuñez. Efficient Method to Obtain the Entire Active Area against Circuit Delay Time Trade–Off Curve in Gate Sizing. In *IEE Proc. Circuits, Devices & Systems*, volume 152 (2), April 2005.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

transición $T_{y,t}$ se define como:

$$Y_t = A_t + B_t \quad (3.47)$$
$$Y_{t-1} = A_{t-1} + B_{t-1} \quad (3.48)$$
$$T_{y,t} = Y_t \oplus Y_{t-1} = (A_t + B_t) \oplus (A_{t-1} + B_{t-1}) \quad (3.49)$$

El algoritmo presentado en la figura 3.16 mide la actividad de conmutación de un circuito de lógica combinacional dados dos vectores de entrada.

```
Evaluate(prev_input_vector, next_input_vector)
    set t_transitions to zero;
    set prev_input_vector in the circuit inputs;
    simulate Boolean network;
    for each gate of the Boolean network
        save previous gate state;
    end for;
    set next_input_vector in the circuit inputs;
    simulate the Boolean network;
    for each gate of the Boolean network
        if current state is not equal to previous state then
            t_transitions++;
        end if;
    end for;
    return t_transitions;
end Evaluate
```

Figura 3.16: Algoritmo para evaluar la actividad de conmutación producida por dos vectores de entrada consecutivos.

Evaluación del Número de Transiciones

Primero, el algoritmo inicializa el número total de transiciones identificadas. Después, fija el vector de entrada que define el estado previo de la red Booleana. A continuación guardar el estado de cada una de las puertas de la red Booleana. Entonces, el algoritmo fija el vector de entrada siguiente. Finalmente, evalúa para cada puerta si su estado actual es igual o distinto al estado anterior. Cuando una puerta posee distinto estado previo y actual, el número de transiciones es incrementado en una unidad.

La figura 3.17 presenta dos ejemplos de la estimación de la actividad de conmutación. La figura 3.17(a) muestra la representación del circuito empleado en forma de grafo. El circuito está compuesto de tres puertas NOR y tres INV.

Si se supone que el vector previo aplicado es el vector `0000` y es aplicado a las entradas A, B, C y D, respectivamente. Después de fijar el vector de entrada, el algoritmo presentado en la figura 3.16 simula la red Booleana, tal y como se muestra en la figura 3.17(b). El estado de cada puerta es almacenado en cada nodo del grafo. Entonces, el siguiente vector de entrada es fijado en las entradas del circuito y la red Booleana es simulada. Tomando este nuevo vector de entrada como el `1111`, el resultado de su simulación es presentado en la figura 3.17(c). A partir de aquí, para cada puerta del circuito, se compara el estado previo y el nuevo estado. Cuando en una puerta no coincide el estado previo y nuevo, el

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

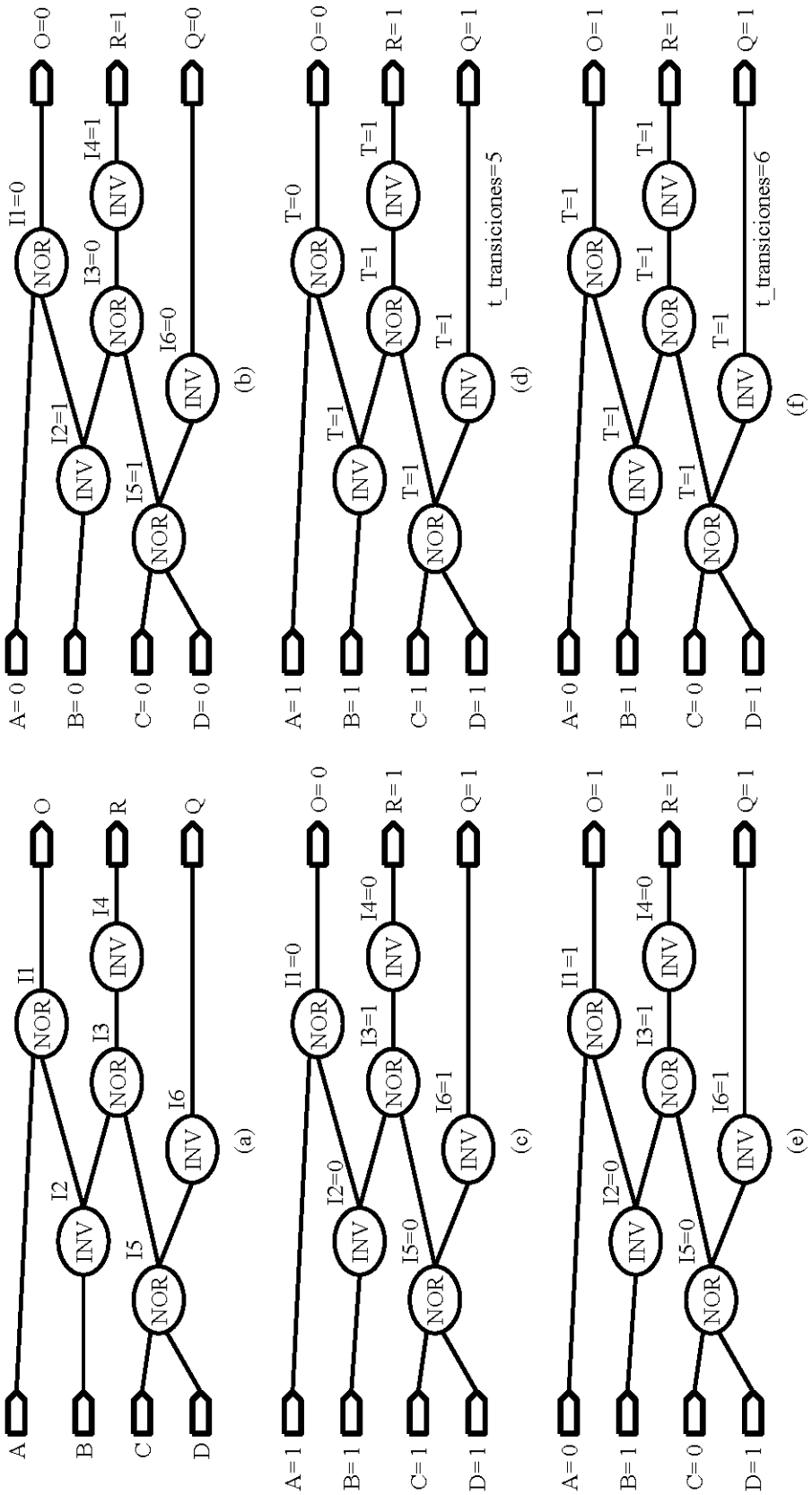


Figura 3.17: Ejemplo de actividad de conmutación. a) Circuito de lógica combinacional bajo estudio representado como grafo, b) estado interno estable de la red Booleana cuando el vector de entrada es el 0000, c) estado interno estable de la red Booleana cuando el vector de entrada es el 1111, d) detección de las transiciones cuando el vector de entrada cambia de 0000 a 1111, e) estado interno estable cuando el vector de entrada es el 0101, y f) detección de las transiciones cuando el vector de entrada cambia de 0000 a 0101.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

número de transiciones contabilizado es incrementado en una unidad. El resultado del proceso de cuenta del número de transiciones se presenta en la figura 3.17(d), donde el número de transiciones es 5 cuando el vector de entrada va de `0000` a `1111`.

Influencia de los Vectores Escogidos

La figura 3.17(e) muestra el ejemplo de simulación del circuito cuando el vector de entrada es el `0101`. En este caso, el número total de transiciones es igual a 6, tal como se muestra en la figura 3.17(f). Esto es, todas las puertas del circuito conmutan si el vector de entrada cambia de `0000` a `0101`.

La codificación adecuada del individuo de la población genética no es una tarea trivial. Con el objetivo de emplear la función de coste propuesta en la figura 3.16, el Algoritmo Genético ha de proponer un par de vectores de entrada. Pero, el interés de este estudio se centra en obtener la secuencia de estímulos de entrada que maximizan la potencia de consumo. Dicho de otra forma, se ha de buscar una secuencia de entrada y no un conjunto de pares de vectores sin relación alguna que maximice la potencia de consumo.

Entonces, se ha de redefinir el algoritmo propuesto en la figura 3.16 con el objetivo de obtener la secuencia de entrada.

Iteraciones del Algoritmo Genético

El refinamiento consiste en optimizar la función de costo propuesta en la figura 3.16 reorganizando las tareas que en ella se realizan de forma que se compare siempre el último vector aceptado con uno nuevo propuesto. La reorganización de la función de coste para acometer esta nueva forma de operar se puede solventar con tres funciones independientes. Estas tres nuevas funciones se llaman *setsim_vector, update_previous, check_proposal.*

setsim_vector : Inicializa el entorno de generación de vectores, calculado el estado interno de cada puerta lógica del circuito en estudio y fija el primer vector de la secuencia de estímulos que maximizan la potencia de consumo.

update_previous : Define el estado actual de la red Booleana como estado anterior para las futuras iteraciones. Ello se debe a que el vector de entrada que genera el estado actual ha sido aceptado como válido y, por tanto, se ha introducido dentro de la secuencia de vectores de entrada de máxima potencia.

check_proposal : Evalúa el vector propuesto y el estado de la red Booleana debido a ese vector con respecto al último estado aceptado. Esta función determina el número de transiciones que se alcanzan al emplear el vector de entrada propuesto con respecto al último vector aceptado.

Las funciones anteriores se utilizan como sigue. En primer lugar, tan pronto como la aplicación de Algoritmos Genéticos comienza, la rutina de inicialización llama a la función setsim_vector con el vector de entrada definido por defecto. Esto es, el algoritmo fija el primer vector de la secuencia a encontrar. El valor por defecto para el vector de entrada se ha definido en este trabajo como aquel que posee todos sus bits puestos al valor lógico 0.

Función de Coste

La nueva Función de Coste es presentada en la figura 3.19. Esta función de coste se basa en una codificación individual que posee un arreglo de N valores reales definidos entre 0 y 1. N corresponde al número de entradas. Cada entrada del circuito se corresponde con un único elemento del arreglo de valores reales. El índice del elemento real identifica unívocamente la entrada que representa. Cada

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

```
setsim_vector(input_vector)
    set input_vector in the circuit inputs;
    simulate Boolean network;
end setsim_vector

update_previous()
    for each gate of the Boolean network
        save previous gate state;
    end for;
end update_previous

check_proposal(next_input_vector)
    set t_transitions to zero;
    set next_input_vector in the circuit inputs;
    simulate the Boolean network;
    for each gate of the Boolean network;
        if current state is not equal to previous state then
            t_transitions++;
        end if;
    end for;
    return t_transitions;
end check_proposal
```

Figura 3.18: Algoritmo para obtener el estado interno de un circuito; dado un vector de entrada, actualiza y evalúa la actividad de conmutación basado en el estado anterior de la red Booleana.

```
evaluate_proposal(Individual)
    set t_transitions to zero;
    Convert Individual to next_input_vector;
    t_transitions = check_proposal(next_input_vector);
    if the current t_transitions is better or
            equal than the previous computed then
        update_previous();
        accept current proposal;
        store t_transitions as the previous computed;
    end if;
    return t_transitions;
end evaluate_proposal
```

Figura 3.19: Algoritmo de cálculo del costo para evaluar la actividad de conmutación en un circuito de lógica combinacional basado en el último vector aceptado.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

valor real del arreglo de N elementos representa la probabilidad de existir un uno lógico en la entrada del circuito que representa.

Codificación del Individuo

En la Función de Coste comienza inicializando a cero su contador de transiciones. Entonces, el arreglo de N valores reales se convierte a un vector de valores Booleana. Con el objetivo de que la asignación al uno y al cero lógico tengan la misma probabilidad, se ha definido el nivel de decisión al valor real de 0'5. Una vez el vector de entrada es traducido desde el individuo codificado, el algoritmo calcula el número de transiciones con el vector previo aceptado y el vector propuesto.

Si el número de transiciones alcanzado es mayor o igual al número de transiciones del vector aceptado con anterioridad, la propuesta actual es aceptada. El estado actual de cada puerta y el número de transiciones obtenidas pasan a ser valores previos para la siguiente iteración. Si la propuesta actual no es aceptada se desecha para el cálculo de la siguiente iteración, y el vector de entrada previo se mantiene.

Búsqueda de una Secuencia

De esta forma, es posible buscar la secuencia de estímulos de entrada que maximizan la potencia de consumo. Con el objeto de seleccionar la mejor secuencia de estímulos de entrada, se asigna a cada individuo de cada generación una secuencia de M secuencias distintas. Ello implica que se calculan M secuencias de máxima potencia de consumo, simultáneamente. Por lo tanto, la Función de Coste ha de ser modificada para realizar la búsqueda de las M secuencias. La figura 3.20 muestra los cambios.

Las modificaciones en los algoritmos se corresponden, exclusivamente, a la vectorización del estado de la red Booleana y del número de transiciones alcanzado en la iteración anterior, puesto que ahora es necesario almacenar el estado de la red y el número de transiciones alcanzado por cada secuencia independiente. El número de secuencia asignado a cada individuo es calculado como el resto de la división entera entre el número de Funciones de Coste evaluadas y el número total de secuencias permitidas.

```
evaluate_proposal(Individual)
    set t_transitions to zero;
    total_evaluations++;
    Current_seq = Total_evaluations % Number of sequences;
    Convert Individual to next_input_vector;
    t_transitions = check_proposal(next_input_vector, Current_seq);
    if the current t_transitions is better or
            equal than the previous[Current_seq] computed then
        update_previous(Current_seq);
        accept current[Current_seq] proposal;
        store t_transitions as the previous[Current_seq] computed;
    end if;
    return t_transitions;
end evaluate_proposal
```

Figura 3.20: Algoritmo final de coste para evaluar la actividad de conmutación en un circuito de lógica combinacional.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

3.4.3. Resultados Experimentales

Con el objetivo de evaluar la metodología propuesta empleando Algoritmos Genéticos para obtener la secuencia de estímulos que maximizan la potencia de consumo de un circuito combinacional, el conjunto de circuitos de dos niveles y multinivel con más de 1000 puertas del banco de pruebas MCNC'91 ha sido procesado [72].

Núcleo del Algoritmo Genético

Los resultados han sido calculados en un Pentium IV a 2'8 GHz, con 1 $GByte$ de RAM, ejecutando el Sistema Operativo Linux Debian 3.3.4 con el kernel 2.6.7. Se ha empleado GCC v3.3.5, y el software GENEsYs v1.0 como núcleo del algoritmo genético, y MISII como sistema de síntesis lógica. Se ha desarrollado un estimador de actividad de conmutación en lenguaje C. Tanto el estimador de actividad de conmutación como el núcleo de GENEsYs han sido completamente integrados en la suite de herramientas OLYMPO [89].

Cada circuito del banco de pruebas MCNC'91 ha sido preprocesado con el sistema de síntesis lógica y mapeado a una librería de puertas lógicas — NAND, NOR e INV — con mínima área. Cada circuito ha sido procesado con la metodología basada en Algoritmos Genéticos para obtener la secuencia de estímulos de entrada que maximizan la potencia de consumo. Además, los circuitos han sido procesados con el estimador de actividad de conmutación propuesto en [90]. Ambos algoritmos han sido ejecutados para obtener la potencia de consumo máxima.

En todos los experimentos basados en Algoritmos Genéticos, la estrategia para el reemplazo de las generaciones antiguas, después de generar la nueva población, ha sido el de seleccionar los mejores individuos. El esquema de selección ha sido el proporcional y la población de 100 individuos. Finalmente, se ha empleado Cruzamiento Multipunto y Mutación Estándar.

Técnica Probabilística Frente a Solución Heurística

La tabla 3.7 muestra una comparativa en términos de la actividad de conmutación estimada para obtener la potencia de consumo máxima con una técnica probabilística y la solución aquí presentada basada en Algoritmos Genéticos. La primera columna presenta el nombre del circuito de acuerdo con el banco de pruebas MCNC'91. La columna segunda presenta la complejidad del circuito medida en número de puertas. La tercera columna muestra la actividad de conmutación máxima alcanzada cuando se aplica una técnica probabilística. La quinta columna presenta el número de puertas que conmutan con los vectores generados por la aportación basada en Algoritmos Genéticos. La sexta columna muestra la actividad de conmutación como el porcentaje de puertas que conmutan del total del circuito. Finalmente, la última columna presenta la diferencia entre estimar la potencia de consumo mediante métodos probabilísticos y la técnica basada en Algoritmos Genéticos.

La complejidad de los circuitos tratados, medida en términos de puertas, ha sido desde circuitos poco complejos como los circuitos ex5 y table5 con algo más

[72] J. Sosa, J. A. Montiel-Nelson, H. Navarro, and José C. García. Stimuli Sequence Generation for Verification and Validation of Maximum Power Consumption using Evolutionary Programming. In *Evolutionary and Deterministic Methods for Design, Optimization and Control with Applications to Industrial and Societal Problems 2005*, September 2005.

[89] J.A. Montiel-Nelson, V. De Armas, and A. Sarmiento, R. and Nuñez. A Cell and Macrocell Compiler for GaAs VLSI Full–Custom Design. *Proc. Design Automation and Test in Europe, Conf. and Exhibition*, pages 947–948, feb 1998.

[90] A. Ghosh, S. Devadas, K. Keutzer, and J. White. Estimation on Average Switching Activity in Combinational and Sequential Circuits. *Proc. 29th ACM/IEEE Design Automation Conference*, pages 243–259, 1992.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

Tabla 3.7: COMPARATIVA DE LOS RESULTADOS EXPERIMENTALES EMPLEANDO ALGORITMOS GENÉTICOS Y UNA TÉCNICA PROBABILÍSTICA PARA OBTENER EL CONSUMO DE POTENCIA MÁXIMO, MEDIDO COMO ACTIVIDAD DE CONMUTACIÓN

Circuito MCNC'91		Actividad Circuital				Diferencia (%)
		Solución Probabilística		Algoritmo Genético		
Nombre	Puertas	Absoluta	%	Absoluta	(%)	
alu4	1.360	759	55'84	908	66'76	10'92
apex1	1.527	838	54'91	766	50'16	-4'74
apex3	1.985	1.068	53'78	1.103	55'57	1'78
apex4	3.010	1.597	53'07	1.811	60'17	7'10
apex5	1.281	673	52'57	745	58'16	5'58
apex6	1.196	626	52'32	724	60'54	8'22
C2670	1.232	704	57'11	734	59'58	2'47
C3540	2.198	1.210	55'06	1.263	57'46	2'40
C5315	2.988	1.633	54'66	1.756	58'77	4'11
C6288	4.139	2.473	59'76	2.396	57'89	-1'87
C7552	3.095	1.723	55'68	2.124	68'63	12'94
cps	1.666	894	53'68	986	59'18	5'50
dalu	2.294	1.293	56'35	1.568	68'35	12'01
ex1010	3.558	1.823	51'24	2.187	61'47	10'23
ex5	1.034	543	52'50	638	61'70	9'20
frg2	1.307	706	54'04	829	63'43	9'39
i10	3.643	2.042	56'05	1.870	51'33	-4'72
i8	1.603	846	52'79	878	54'77	1'98
k2	1.523	829	54'44	773	50'76	-3'68
misex3	1.197	648	54'17	756	63'16	8'99
pair	2.562	1.393	54'36	1.413	55'15	0'80
rot	1.058	579	54'68	651	61'53	6'85
table3	1.094	565	51'68	614	56'12	4'45
table5	1.034	564	54'53	535	51'74	-2'79
x3	1.323	718	54'28	879	66'44	12'16

de mil puertas. Por otro lado, el más complejo ha sido el C6288 con más de cuatro mil puertas. La actividad de conmutación media ha superado en todos los casos el 50 % del circuito, independientemente a la técnica empleada.

Hay que tener en cuenta que la comparativa se realiza con una técnica probabilística, y debido exclusivamente a su naturaleza estadística, no tiene como objetivo obtener la secuencia de entrada que maximiza la potencia de consumo. Esta técnica se ha planteado como referencia ya que en la literatura se emplea para establecer la potencia máxima de la red Booleana en estudio. Sin embargo, la metodología basada en Algoritmos Genéticos difiere hasta un 13 % en valor absoluto del valor precedido por la técnica estadística. Y en particular, para cinco de los casos estudiados la potencia máxima determinada por la solución basada en Algoritmos Genéticos posee valor inferior que la potencia estimada por el método probabilístico.

Precisión

El esfuerzo computacional en términos de tiempo de CPU necesario no es significativo. El tiempo de CPU necesario para todas las ejecuciones tanto de la solución basada en Algoritmos Genéticos como la probabilística están por debajo del segundo.

La figura 3.21 muestra la convergencia de una búsqueda con cinco secuencias. La mayoría de las secuencias convergen en un número corto de intentos. La convergencia comienza sobre los mil intentos. Una vez se alcanzan los mil intentos, el esfuerzo computacional requerido para incrementar la actividad de conmutación es al menos diez veces mayor que la requerida para alcanzar el punto de convergencia, medido en términos de intentos.

Convergencia

Además, no todas las secuencias poseen un grado parecido de actividad de

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

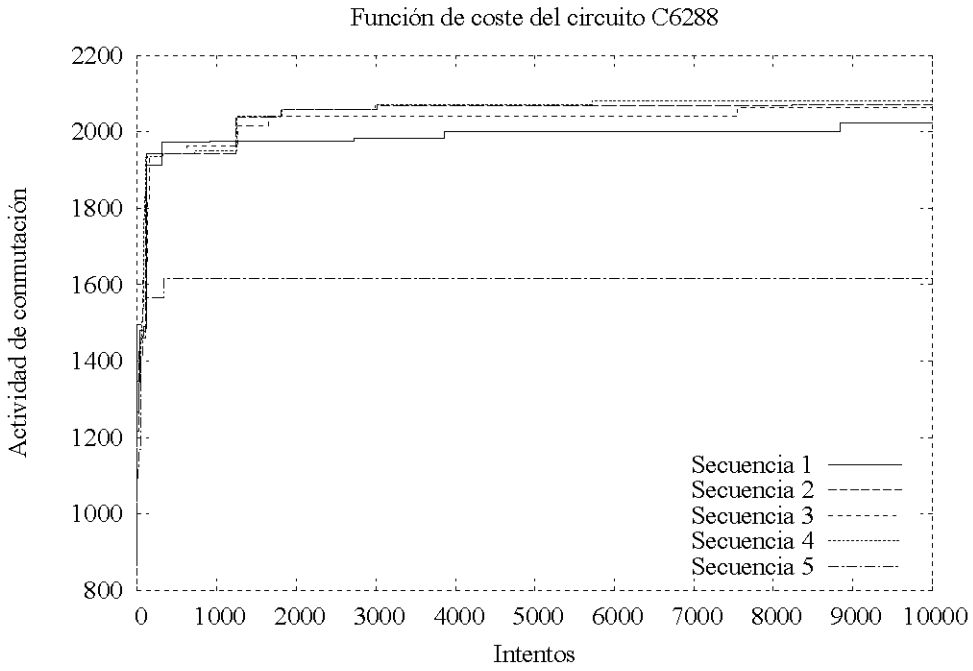


Figura 3.21: Convergencia de la función de coste del circuito C6288 para la búsqueda de la secuencia de estímulos de entrada con un conjunto de 5 búsquedas simultáneas.

conmutación. Por ejemplo, se puede observar que la quinta secuencia del ejemplo de la figura 3.21 posee un grado muy bajo de convergencia con respecto a las otras secuencias encontradas. En este ejemplo, la actividad de conmutación de la quinta secuencia está lejos del valor final de la actividad de conmutación del resto de secuencias encontradas.

3.4.4. Conclusiones

Se ha propuesto una metodología eficiente para obtener la secuencia de estímulos de entrada que maximizan la potencia de consumo de un circuito de lógica combinacional [42]. Se emplean Algoritmos Genéticos como procedimiento heurístico para implementar la metodología propuesta. Las comparativas con una estrategia probabilística han demostrado que la solución adoptada obtiene secuencias de entrada que maximizan la potencia de consumo de una forma eficiente (en términos de tiempo de CPU y número de intentos necesarios) [72].

Se ha probado de forma unívoca que las técnicas probabilísticas dependen de la distribución de probabilidad de los estímulos de entrada, y que estas no son conocidas a priori si se desea obtener los valores máximos de potencia de consumo. La calidad de las secuencias generadas — medida en actividad de conmutación — con la solución aportada con la técnica evolutiva son excelentes en todos

[42] J. Sosa, J. A. Montiel-Nelson, H. Navarro, José C. García, and R. Sarmiento. *Circuit Path Coverage using Genetic Algorithms*, pages 51–52. Evolutionary Methods for Design optimization and Control Applications to Industrial and Societal Problems. International Center for Numerical Methods in Engineering, 2003.

[72] J. Sosa, J. A. Montiel-Nelson, H. Navarro, and José C. García. Stimuli Sequence Generation for Verification and Validation of Maximum Power Consumption using Evolutionary Programming. In *Evolutionary and Deterministic Methods for Design, Optimization and Control with Applications to Industrial and Societal Problems 2005*, September 2005.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

los casos. Además, el número de intentos pueden ser ajustados en función del tamaño total del circuito — medido en número de puertas — para reducir aún más el tiempo computación.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

Capítulo 4

Optimización del Retardo

Índice General

4.1. Introducción 124

4.2. Trabajos Previos 126

4.3. Sinopsis 128

4.4. Formulación del Problema de Optimización de la Ruta Crítica 129

4.4.1. Conjunto Ruta Crítica 129

4.5. Optimización de un Circuito Empleando el Conjunto Ruta Crítica 131

4.6. Curva de Prestaciones de un Circuito 132

4.6.1. Mapeado a Mínima Área Activa 132

4.6.2. Curva de Prestaciones 134

4.6.3. Algoritmo de Cálculo de la Curva de Prestaciones . . 135

4.6.4. Modelo de Retardo y Área Activa de Puerta 137

4.6.5. Curva de Prestaciones empleando Programación Lineal 139

4.6.6. Complejidad de la Optimización 140

4.6.7. Comparativa 144

4.6.8. Curvas de Prestaciones 146

4.7. RTL versus Circuito de Lógica Combinacional 146

4.8. Conclusiones 151

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

Glosario de Términos

$\mathcal{A}$	Área activa del circuito.
(A, P)	Punto del espacio de diseño que posee el área A y la potencia P.
β_{gate}	Factor de velocidad de la puerta $gate$.
$\mathcal{C}$	Espacio de diseño completo de un circuito.
C_{load}	Capacidad de cableado y de puerta soportado.
$\mathcal{C}_{tradeoff}$	Curva de prestaciones óptimas de un circuito.
$C_{int,i}$	Capacidad de la puerta i.
C_{wire}	Capacidad de cableado.
$D(path_{cp})$	Conjunto de rutas dependientes de la ruta crítica.
$\vec{e}_j$	Arco que modela la conexión j entre dos puertas lógicas.
$fanin(v_j)$	Puertas conectadas como fuentes de señal a la puerta v_j.
$fanin(n, g)$	Entradas del cono lógico de entrada que define la ocurrencia de g en n.
$fanout(v_i)$	Puertas conectadas como carga a la puerta v_i.
$\vec{G}$	Circuito combinacional expresado como grafo.
$h(path_{v_0,v_k})$	Última puerta desde las entradas de la ruta $path_{v_0,v_k}$.
$I(path_{cp})$	Conjunto de rutas independientes de la ruta crítica.
LL	Librería de puertas.
$\mathcal{L}$	Librería de puertas.
$\breve{\mathcal{L}}$	Librería de puertas convexa.
$\mathcal{L}_{\underline{A}}$	Librería de puertas con versiones de mínima área.
$\mathcal{L}_{\underline{\parallel}}$	Conjunto de descripciones físicas de la puerta k.
$M_{\underline{A}}$	mapeado a mínima área.
$merge(g, n)$	Ocurrencia de la puerta de librería g en el vértice n.
$path$	Ruta.
$path_{cp}$	Ruta crítica o camino crítico.
PO	Conjunto de salidas primarias del circuito.
RTL	Register Transfer Language, lenguaje de transferencia de registros.
$\overline{T}$	Tiempo máximo del dominio de definición de la ruta crítica.
$\breve{T}$	Retardo de un punto no inferior.
$\underline{T}$	Tiempo mínimo del dominio de definición de la ruta crítica.
T_{gate}	Tiempo máximo en la salida de la puerta $gate$.
T_{max}	Tiempo máximo del circuito.
τ_{gate}, τ_v	Retardo de la puerta $gate$.
$\tau_{int,i}$	Retardo interno de puerta i.
$t(path_{v_0,v_k})$	Primera puerta desde las entradas de la ruta $path_{v_0,v_k}$.
v_i	Vértice, nodo que representa a la puerta combinacional i.
VLSI	Very Large Scale Integration.

En el capítulo anterior se presentó la forma de generar vectores de máxima cobertura con el objeto de reducir el tiempo de verificación. Otro aspecto muy importante en la verificación de sistemas integrados es el modelo temporal adoptado durante la verificación. Si el objetivo es comprobar estrictamente la funcionalidad implementada sin atender a las prestaciones de las mismas, un modelo temporal a nivel de ciclo de reloj puede ser válido. Pero, el problema surge cuando no sólo se requiere comprobar la funcionalidad con respecto a un conjunto de reglas y es necesario contrastar un conjunto de prestaciones y además la complejidad del diseño apura al límite las características de una tecnología — velocidad, número de puertas entre registros, entre otras. En ese caso, no es válido cualquier modelo de retado, es necesario analizar el circuito para obtener los retardos significativos de las diversas funcionalidades y anotar dichos retardos en los modelos funcionales diseñados, para verificarlos temporalmente.

Por otro lado, dado un diseño, éste puede poseer diversas implementaciones circuitales. Cada una de esas implementaciones circuitales da origen a retardos diversos. Un handicap existente hoy día es que en vez de explorar el espacio de diseño de los sistemas integrados, con el objeto de obtener la curva de prestaciones de estos, se opta por encontrar una solución que puede cumplir o no la verificación del diseño. Consecuentemente, cada vez que se viola alguna regla de verificación

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

debido a la temporización de la implementación obtenida, hay que buscar otra nueva solución circuital que supere toda la verificación. Este capítulo propone una metodología de exploración del espacio de diseño de un circuito con el objeto de encontrar el rango de soluciones válidas que cumplen la verificación.

En este capítulo se presenta una metodología para la exploración de las prestaciones de circuitos combinacionales en términos de los parámetros área activa y retardo de la ruta crítica. La técnica propuesta realiza una optimización de las dimensiones de las puertas utilizando factores que compiten entre sí, el área activa y el retardo. El área activa total del circuito se supone que es una función lineal del tamaño de las puertas. Existe una ruta crítica cuyo retardo — el retardo temporal más largo del circuito — depende del tamaño de las puertas individuales. Durante el proceso de optimización este retardo cambiará muchas veces. El proceso de optimización se utilizará para generar la curva de prestaciones, que es el objetivo de este capítulo. La metodología propuesta se basa en la optimización de la ruta crítica del circuito. La optimización de la ruta crítica del circuito es un problema de dimensionado de puertas en aquellas rutas que definen el retardo del circuito. Dado un tamaño de *driver*[1] para todas las puertas del circuito, la curva de prestaciones contendrá aquellos puntos del espacio de diseño de mínimo retardo y área activa. En este capítulo se presenta un método que optimiza un subconjunto variable de la red Booleana que representa al circuito bajo optimización. Sólo aquellas puertas de la ruta crítica serán optimizadas. Aquellas puertas en la red Booleana que no forman parte de la ruta crítica no necesitan ser optimizadas. Se propone un método eficiente para actualizar la ruta crítica a medida que la curva de prestaciones se obtiene. La comparación de prestaciones y los resultados expuestos se obtienen sobre el banco de pruebas o *benchmark* MCNC'91, circuitos combinacionales de dos niveles y multinivel.

Curva de Prestaciones

La metodología propuesta genera curvas de prestaciones, en circuitos relativamente grandes, decrementando notoriamente el número de variables y el tiempo de ejecución en comparación con el método de obtención de la curva de prestaciones para la red Booleana completa — el método tradicional. Con esta propuesta, el número de variables se reduce en un factor de hasta 7'3 veces en comparación con el método tradicional — la computación de la red Booleana completa utilizando Programación Lineal. La característica de tiempo de ejecución es excelente, con una mejora entre 1'5 y 32'5 veces el tiempo de CPU empleado en el cálculo con la red Booleana completa. Cuanto mayor sea el número de puertas del circuito, mayor será la mejora en tiempo de CPU y en su complejidad — número de variables.

Este capítulo comenzará en su apartado 4.1 con una introducción al análisis de prestaciones, para a continuación en el apartado 4.2 presentar los estudios previos desarrollados en este ámbito. En el apartado 4.3 se plantea una sinopsis del trabajo aquí presentado. El apartado 4.4 perfila el conjunto de definiciones necesario para abordar la formulación del problema de optimización de la ruta crítica. En este apartado se definirán conceptos básicos de teoría de grafos, la ruta crítica y el conjunto ruta crítica y sus dependencias con el resto de circuitos. El apartado 4.5 propone una metodología para optimizar un circuito empleando su ruta crítica. La curva de prestaciones de un circuito y un algoritmo para obtenerla se abordan en profundidad en el apartado 4.6. Se discutirá el particular punto de arranque

[1] Término anglosajón que identifica a un circuito amplificador que adapta la impedancia entre los circuitos conectados a su entrada y a su salida.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

del algoritmo de análisis de prestaciones. Además, se muestra una versión del algoritmo de análisis de prestaciones empleando Programación Lineal. También se analiza la complejidad de la optimización y se plantea la comparativa de la metodología presentada con la metodología tradicional. Se presenta la curva de prestaciones para el circuito bw. En el apartado 4.7 se extiende la metodología propuesta al nivel RTL lo cual permite emplearla tanto a nivel estructural como funcional. Para finalizar este capítulo, en el apartado 4.8 presenta las conclusiones del mismo.

4.1. Introducción

Aumentar la frecuencia de funcionamiento de un circuito implica reducir el retardo de su ruta crítica. Dada una red Booleana $\vec{G}$ que representa el circuito a optimizar y que se encuentra mapeado[2] en un punto del espacio de diseño óptimo en área y retardo. La subred que define el retardo máximo del circuito es la ruta crítica . El retardo de la red Booleana $\vec{G}$ viene fijado por esa subred. Además, dicha subred es susceptible de cambiar durante el proceso de optimización.

Ruta Crítica Dinámica

El objetivo de la optimización en retardo basado en la ruta crítica es el de obtener menores retardos del circuito, manteniendo el resto de las puertas del circuito mínimos en área. La ruta crítica es dinámica, es decir, a medida que el retardo del circuito cambia, también cambia el tamaño de la subred ruta crítica. Por tanto, dado un punto del espacio de diseño existe una ruta crítica cuyo retardo puede ser optimizado hasta un límite. El límite de esta optimización del retardo se alcanza cuando la ruta deje de ser crítica o no pueda ser mejorada en retardo.

Optimización de la Ruta Crítica

El problema de optimización de la ruta crítica de un circuito puede ser abordado como un problema de dimensionado de puertas sobre aquellas rutas que definen el retardo más largo del circuito. El dimensionado de puertas es un proceso de optimización del retardo. En una red Booleana que representa a un circuito, la capacidad de carga de una puerta es ajustada adecuadamente tal que el área del circuito o la potencia de consumo es minimizado bajo unas restricciones de temporización. Durante la optimización de un circuito, una ruta crítica nueva podría surgir. Por ejemplo, dado un tamaño a cada puerta del circuito, una ruta crítica — un subconjunto de la red — define el máximo retardo del circuito T_{max}; cuando el máximo retardo es reducido, mediante el dimensionado de cada una de las puertas, pueden aparecer otros subconjuntos con mayor retardo.

La curva de prestaciones área–retardo de un circuito es empleada por los diseñadores VLSI para obtener las prestaciones de los dimensionados de las puertas de la red, siempre bajo alguna restricción de temporización. Cada conjunto de dimensiones de puerta define un retardo en el circuito. Este retardo lo define la ruta crítica del circuito. El proceso de optimización del tamaño de puerta se emplea para calcular un único punto de la curva de prestaciones área–retardo.

Las aportaciones publicadas en este área obtienen la curva de prestaciones empleando la red completa y no el subconjunto de esta que supone su ruta crítica . En estas soluciones, tanto el número de restricciones como el de variables, necesarias para resolver el problema planteado, es lineal con el número de puertas del circuito. A mayor complejidad — medida en numero de puertas — mayor esfuerzo

[2]Una de las etapas de la síntesis lógica es el mapeado tecnológico. Esta fase asigna células de una librería física a las puertas lógicas de una red Booleana $\vec{G}$ que ha sido optimizada a nivel lógico.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

computacional [91].

Procedimiento Normal

El procedimiento normal para obtener la curva de prestaciones requiere el cálculo de muchos puntos de la curva, mediante la repetición del proceso de dimensionando las puertas con la red completa. Para calcular un punto de la curva de prestaciones área–retardo, se fija una restricción de retardo y se optimiza el área, eligiendo los tamaños de puerta correctos — empleando algún algoritmo de dimensionado. De un punto a otro de la curva, la red que define la ruta con más retardo, el conjunto ruta crítica, podría cambiar. A pesar que el cálculo de un punto de la curva de prestaciones se realizada mediante un método iterativo de dimensionado de puerta [92], el conjunto de puertas ha de ser actualizado en cada iteración. En este capítulo, se propone un método para obtener la curva de prestaciones área–retardo, dimensionando exclusivamente las puertas de la ruta crítica del circuito y no todas las puertas del mismo. Dimensionado solamente las puertas que pertenecen a la ruta crítica del circuito, se eliminan de los cálculos todas aquellas puertas que no contribuyen al mayor retardo del circuito. Ello redunda en la reducción del número de variables y por tanto el esfuerzo computacional necesario para resolver el problema. En el método presentado, el cálculo de la curva de prestaciones comienza con un circuito mapeado a mínima área y, mediante optimizaciones sucesivas de dimensionado de puerta, éste finaliza en el punto de mínimo retardo de la curva de prestaciones — óptimo en área. Cuando no es posible reducir el retardo con mínima área activa, el método finaliza. Durante el cálculo de la curva de prestaciones, de un punto a otro, el retardo del circuito se reduce. Para calcular un nuevo punto de la curva de prestaciones con menor retardo, se fija una restricción de temporización tal que el conjunto de puertas de la ruta crítica no cambie. Se restringe la reducción de retardo de la ruta crítica actual empleando un límite inferior. El límite inferior del conjunto ruta crítica actual es el retado máximo del siguiente conjunto ruta crítica. Por ello, el rango de variación del retardo del conjunto ruta crítica — el dominio temporal de definición del conjunto ruta crítica — está limitado. Además se propone un procedimiento de actualización de las puertas del conjunto ruta crítica, cuando su retardo no puede ser reducido más.

La propuesta desarrollada en este trabajo de investigación para mapear, tecnológicamente, una red Booleana $\vec{G}$ en una librería de células *full–custom* consiste en: comenzar la optimización en un punto del espacio de diseño conocido (mínimo en área) e ir disminuyendo el retardo de la ruta crítica dinámica utilizando la mínima de área requerida.

Mínimo Minimorum en Área

En el espacio de diseño hay un punto que es mínimo en área, que es el idóneo para iniciar la exploración del espacio de diseño. Este punto de arranque de la exploración del espacio de diseño es el de mínima área global, que se ha obtenido mapeando la descripción lógica al conjunto de puertas que tienen mínima área (véase figura 4.1). A partir del punto del espacio de diseño mínimo minimorum en área, el objetivo es ir mejorando el retardo definido por la ruta crítica dinámica del circuito, optimizando siempre el área. Para una ruta crítica del circuito, el interés

[91] M.R.C.M. Berkelaar, P.H.W. Buurman, and J.A.G. Jess. Computing the Entire Active Area/Power Consumption versus Delay Tradeoff Curve for Gate Sizing with a Piecewise Linear Simulator. *IEEE Trans. Comput.–Aided Des. Integr. Circuits Syst.*, 15(11):1424–1434, 1996.

[92] G. Chen, H. Onodera, and K. Tamaru. An Iterative Gate Sizing Approach with Accurate Delay Evaluation. *Proc. ACM/IEEE Design Automation Conf.*, pages 422–427, November 1995.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

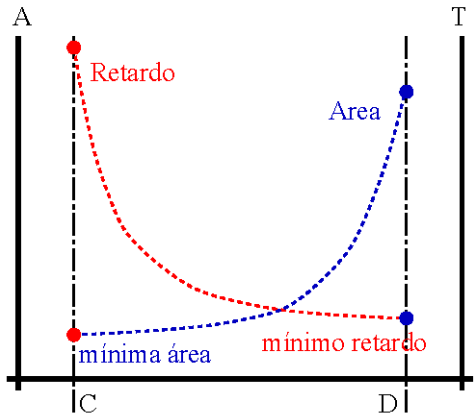


Figura 4.1: Curva de prestaciones área–retardo. La recta C representa el punto de partida de la exploración, mínima área, máximo retardo del circuito. La recta D representa el punto de mínimo retardo del circuito y, por tanto, el de máxima área.

Análisis de Prestaciones

estriba en conocer un nuevo punto del espacio de diseño — el de mínima área que satisfaga una restricción de retardo — sin modificar la subred que representa la ruta crítica. La herramienta que realiza este tipo de exploración se denomina Analizador de Prestaciones.

4.2. Trabajos Previos

Optimizadores de Punto Único

Un problema de dimensionado de puertas es una particularización del problema general de dimensionado de transistores. Hay una extensa cantidad de trabajo realizado con el objetivo de resolver este problema, y desde los inicios de los 80, han sido propuestos varios optimizadores heurísticos y combinados heurísticos–algorítmicos [93]–[94]. En [95], se presenta una solución imprecisa que fue ideada para guiar al diseñador. TILOS [96] optimiza la ruta crítica mediante el dimensionado de transistores de forma iterativa. Éste emplea una estrategia de fuerza bruta[3] para resolver una formulación posinomial[4].

La optimización del dimensionado de puerta ha sido resuelto en [97] empleando Programación Lineal y aproximaciones lineales a tramos de la formulación no

[3] Traducción directa del término anglosajón *greedy*.

[4] Un posinomio es similar a un polinomio, excepto que los coeficientes polinomiales son positivos y los exponentes podrían ser números reales, no necesariamente un entero positivo. Formalmente, un posinomio es una función f de una variable positiva $\omega \in \mathbf{R}^2$, tal que $f(\omega_1, \omega_2, ...\omega_n) = \sum_{i=1}^{m} \alpha_i \prod_{j=1}^{n} \omega_j^{\gamma_{ij}}$, donde los exponentes $\gamma_{ij} \in \mathbf{R}$ y los coeficientes $\alpha_i \geq 0$.

[93] L.A. Glasser and L.P.J. Hoyte. Delay and Power Optimization in VLSI Circuits. *Proc. 21st ACM/IEEE Design Automation Conf.*, pages 529–535, jun 1984.

[94] J. Shyu, A. Sangiovanni-Vincentelli, J.P. Fishburn, and A.E. Dunlop. Optimization–based Transistor Sizing. *IEEE J. Solid–State Circuits*, 23(2):400–409, 1988.

[95] C.M. Lee and H. Soukup. An Algorithm for CMOS Timing and Area Optimization. *IEEE J. Solid–State Circuits*, 19(5):781–787, 1984.

[96] J.P. Fishburn and A.E. Dunlop. TILOS: A Posynomial Programming Approach to Transistor Sizing. *Proc. ACM/IEEE Int. Conf. Computer–Aided Design*, pages 326–328, November 1985.

[97] M.R.C.M. Berkelaar and J.A.G. Jess. Gate Sizing in MOS Digital Circuits with Linear Programming. *Proc. European Design Automation Conf.*, pages 217–221, March 1990.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

lineal del retardo. Mediante técnicas de Programación No Lineal en [98], [99], y [100] se demuestra que las soluciones heurísticas están lejos del óptimo global. En [101], el problema de dimensionado del transistor ha sido formulado como un problema de Programación Convexa, y resuelto empleando el método del punto interior. A pesar que los métodos basados en Programación No Lineal adolecen de rapidez en sus tiempos de ejecución y convergencia, la aportación presentada en [101] provee una técnica que ofrece una solución exacta, en tiempo polinomial.

Así como la tecnología continúa reduciendo la escala, el retardo de interconexión empieza a tomar un papel determinante en las prestaciones del circuito. Debido a que el cableado afecta a las soluciones del problema de dimensionado de puerta, y viceversa, se han realizado estudios que abarcan ambos efectos en conjunto en las referencias [102]–[103]. En [103] se presenta un algoritmo rápido y exacto basado en técnicas de relajación de Lagrange, que minimiza el área total para un límite superior de retardo.

De otra forma y expresando los retardos de puerta con el modelo de Elmore, en [104] se emplea un analizador de tiempos dinámico muy preciso para calcular una solución exacta, a costa de un tiempo de cálculo poco razonable. Recientemente, en [105] se propone una técnica de relajación iterativa con una estrategia de optimización de dos pasos. Esta aproximación provee una solución exacta y un tiempo de ejecución aproximadamente igual a TILOS [96].

Modelo de Puerta

Tanto las aproximaciones para el dimensionado de transistores como el de puerta sólo calculan un único punto del espacio de diseño, a pesar que el diseñador está interesado en la curva de prestaciones completa. Para obtener la curva de prestaciones, es necesario obtener un gran número de puntos mediante el empleo repetitivo de alguno de los algoritmos de punto único presentados en los párrafos anteriores.

Una reducción del conjunto de puntos calculados, y por tanto de su tiempo de computación, se logra combinando Programación Lineal y un simulador lineal a

[98] M.A. Cirit. Transistor Sizing in CMOS Circuits. *Proc. 24th ACM/IEEE Design Automation Conf.*, pages 121–124, June 1987.

[99] K.S. Hedlund. AESOP: A Tool for Automated Transistor Sizing. *Proc. 24th ACM/IEEE Design Automation Conf.*, pages 114–120, May 1987.

[100] D. Marple. Transistor Size Optimization in the TAILOR Layout System. *Proc. 26th ACM/IEEE Design Automation Conf.*, pages 43–48, June 1989.

[101] S.S. Sapatnekar, V.B. Rao, P.M. Vaidya, and S.M. Kang. An Exact Solution to the Transistor Sizing Problem for CMOS Circuits using Convex Optimization. *IEEE Trans. Comput.–Aided Des. Integr. Circuits Syst.*, 12(11):1621–1634, 1993.

[102] J. Cong and C.K. Koh. Simultaneous Driver and Wire Sizing for Performance and Power Optimization. *Proc. ACM/IEEE Int. Conf. Computer–Aided Design*, pages 206–212, November 1994.

[103] C.P. Chen, C.C.N. Chu, and D.F. Wong. Fast and Exact Simultaneous Gate and Wire Sizing by Lagrangian Relaxation. *IEEE Trans. Comput.–Aided Des. Integr. Circuits Syst.*, 18(7):1014–1025, 1999.

[104] A.R. Conn, P.K. Coulman, R.A. Haring, G.L. Morrill, C. Visweswariah, and Chai Wah Wu. JiffyTune: Circuit Optimization using Time–Domain Sensitivities. *IEEE Trans. Comput.–Aided Des. Integr. Circuits Syst.*, 17(12):1292–1309, 1998.

[105] V. Sundararajan, S.S. Sapatnekar, and K.K. Parhi. Fast and Exact Transistor Sizing Based on Iterative Relaxation. *IEEE Trans. Comput.–Aided Des. Integr. Circuits Syst.*, 21(5):568–581, 2002.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

tramos según lo presentado en [91]. En esta aportación, un simulador con paso de integración variable determina el número de puntos necesarios para realizar la interpolación lineal y así construye una estimación precisa de la curva de prestaciones.

Optimización de la Red Completa

En los métodos anteriores, tanto en el cálculo de un único punto como la curva completa para el problema de dimensionado de puerta incluyen la red completa — cada puerta del circuito. Obtener la curva de prestaciones para el dimensionado de puertas requiere el cálculo de muchos puntos mediante el empleo de un método de optimización de punto único sobre la red completa. Esto es computacionalmente caro. Además, puesto que la ruta crítica de una red cambia durante el proceso de dimensionado de un circuito, existe una gran complejidad para aplicar las aproximaciones de punto único sobre la ruta crítica de tamaño variable.

4.3. Sinopsis

En este capítulo, se presenta un algoritmo eficiente para obtener la curva de prestaciones completa de un circuito. El procedimiento de optimización propuesto sólo afecta a un subconjunto de la red — al conjunto ruta crítica. En primer lugar, cada puerta de la red es dimensionada a mínima área. Este dimensionado del circuito presenta un área que no puede ser reducida más para obtener otro punto de la curva de prestaciones con un retardo mayor, o un retardo que no puede ser aumentado más para obtener otro punto de la curva de prestaciones con menor área.

Optimización de la Ruta Crítica

Para aquellas rutas de la red que no comparten puerta alguna con el conjunto ruta crítica, sus retardos no cambian cuando se reduce el retardo de la ruta crítica. Esos retardos son independientes del conjunto ruta crítica, y su retardo máximo es un límite inferior del retardo de la ruta crítica del circuito.

Dependencias con la Ruta Crítica

Sin embargo, una reducción del retardo de la ruta crítica afecta a los retardos de esas rutas que comparten al menos una puerta con la ruta crítica o poseen puertas que hacen de entrada al conjunto ruta crítica. Esas rutas son las denominadas dependientes de la ruta crítica.

En el segundo paso del algoritmo, el retardo del conjunto ruta crítica actual se reduce con mínima área activa. Puesto que es conocido el límite inferior del retardo del conjunto ruta crítica por adelantado, el proceso de optimización debe detenerse si el retardo de las rutas dependientes exceden el retardo del conjunto ruta crítica actual, en caso contrario, ese límite inferior es alcanzado.

De todas formas, se obtiene un nuevo punto de la curva de prestaciones y la subred que representa al conjunto ruta crítica es actualizado con rutas dependientes e independientes. Esta segunda fase del algoritmo es ejecutada iterativamente hasta que el retardo que se obtiene con mínima área no pueda ser reducido más.

El algoritmo propuesto se emplea para obtener el límite inferior — menor retardo — de un conjunto ruta crítica. En las curvas de prestaciones área–retardo, la variable independiente es el retardo. El conjunto de valores de la variable independiente para el que las prestaciones están definidas es denominado dominio de definición temporal de la curva de prestaciones. Los límites inferior y superior del

[91] M.R.C.M. Berkelaar, P.H.W. Buurman, and J.A.G. Jess. Computing the Entire Active Area/Power Consumption versus Delay Tradeoff Curve for Gate Sizing with a Piecewise Linear Simulator. *IEEE Trans. Comput.–Aided Des. Integr. Circuits Syst.*, 15(11):1424–1434, 1996.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

retardo de cualquier conjunto ruta crítica es denominado dominio de definición del conjunto ruta crítica.

Extensión de la Metodología

Aunque la aportación se ha centrado en el dimensionado de puerta, la solución adoptada es también aplicable a la obtención de la curva de prestaciones completa en el dimensionado de transistores. Otra característica de la aproximación aportada es que ésta puede ser adaptada a cualquier sistema de Programación Lineal, No Lineal o cualquier técnica basada en algoritmos–combinados heurísticos para el modelado del retardo y el área activa.

4.4. Formulación del Problema de Optimización de la Ruta Crítica

En este apartado se comenzará con una definición informal de la ruta crítica como la ruta con el mayor retardo. Nótese, que puede existir más de una ruta que cumpla esta condición, simultáneamente, así que se empleará el concepto conjunto ruta crítica. Se asume, que para un circuito combinacional dado, se ha asignado un tamaño a cada una de sus puertas. Entonces, en ese circuito, se busca al menos una ruta que defina el mayor retardo del circuito. Si el dimensionado de las puertas cambia, entonces una nueva ruta crítica podría surgir. Sin embargo, como primer paso, interesa restringir las variaciones de retardo del conjunto ruta crítica.

Con el propósito de solventar esta cuestión, en este trabajo, se formula un proceso de optimización del área activa y del retardo del conjunto ruta crítica. Se propone un proceso de optimización que provee el límite inferior del retardo. Cualquier reducción de este límite inferior requerirá una actualización del conjunto ruta crítica. Primero, es necesario presentar formalmente al conjunto ruta crítica, las rutas dependientes e independientes del conjunto ruta crítica. Después, se describirá el proceso para optimizar el retardo de un conjunto ruta crítica con mínima área. Esta optimización se realiza de tal forma que el conjunto ruta crítica continua definiendo el retardo máximo del circuito.

4.4.1. Conjunto Ruta Crítica

Definición 3 (Tiempo de Propagación de una Puerta). *El tiempo de propagación en un vértice v_k es el retardo de la ruta $path_{v_0,v_k}$, que es:*

$$T(v_k) \;=\; T_{v_k} = T(path_{v_0,v_k}) \tag{4.1}$$

Definición 4 (Tiempo de Propagación de un Circuito). *El tiempo de propagación de un circuito $\vec{G}$ es el tiempo de propagación máximo de sus salidas primarias, esto es:*

$$T \;=\; \text{máx}(T_{v_j}),\ \forall\, v_j \in PO(\vec{G}) \tag{4.2}$$

En [106] se define a aquellas rutas que pueden ser activadas por al menos un

Ruta Sensible

[106] H.C. Chen and D.H.C. Du. Path Sensitization in Critical Path Problem. *IEEE Trans. Comput.–Aided Des. Integr. Circuits Syst.*, 12(2):196–207, 1993.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

vector de entrada como rutas sensibles[5]. El retardo total de la red $\vec{G}$ se calcula independientemente de que las rutas empleadas sean sensibles, es decir las rutas pueden ser sensibles o incluir rutas falsas[6].

Definición 5 (Conjunto Ruta Crítica de un Circuito). *La ruta $path_{v_0,v_k}$ es la ruta crítica de un circuito de lógica combinacional $\vec{G}$ cuando el retardo de esta ruta es el tiempo de propagación del circuito (véase la expresión 4.2), esto es $T(path_{v_0,v_k}) = T_{v_k} = T$.*

Debido a que $\vec{G}$ puede contener más de una ruta crítica, se define el conjunto ruta crítica como:

$$path_{cp} \quad = \quad \{path_{u,v} \mid T(path_{u,v}) = T(v) = T\} \tag{4.3}$$

donde T fue definido en la Fórmula 4.2.

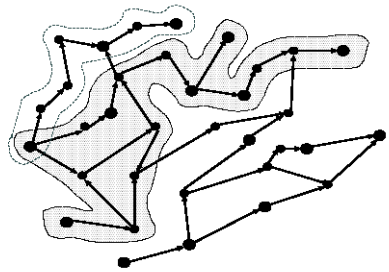


Figura 4.2: Conjunto ruta crítica como subred del circuito: a) conjunto ruta crítica (zona sombreada), b) ruta dependiente del conjunto ruta crítica (área punteada), y c) rutas dependientes del conjunto ruta crítica.

Definición 6 (Ruta Dependiente del Conjunto Ruta Crítica). *Un $path_{u,v}$ depende del conjunto ruta crítica $path_{cp}$ cuando*

- *comparte al menos un nodo, o*
- *al menos un nodo de $path_{cp}$ es una salida inmediata de $path_{u,v}$.*

Esto es, la intersección entre los vértices de $v(path_{cp})$ y del conjunto $v(path_{u,v}) \cup fanout(v(path_{u,v}))$, no es el conjunto vacío.

Por ello, dada una red Booleana $\vec{G}$, se define conjunto de las rutas dependientes del conjunto ruta crítica $path_{cp}$ como:

$$D(path_{cp}) = \{path_{u,v} \mid (v(path_{u,v}) \cup fanout(v(path_{u,v}))) \cap v(path_{cp}) \neq \emptyset\} \tag{4.4}$$

Mientras $D(path_{cp})$ no pertenezca al conjunto ruta crítica $path_{cp}$ entonces, su retardo de ruta es menor que T, esto es T_v es menor que T, para cada $path_{u,v}$ perteneciente a $D(path_{cp})$.

Definición 7 (Ruta Independiente del Conjunto Ruta Crítica). *Un $path_{l,m}$ no depende del conjunto ruta crítica $path_{cp}$ cuando*

[5]Una ruta ejercitable difiere de una ruta sensibilizada, en que esta última puede poseer reconvergencia de transiciones.

[6]Traducción directa del término anglosajón *false path*.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

- *no comparte nodo alguno, y*
- *ningún nodo del* $path_{cp}$ *es una salida inmediata del* $path_{l,m}$.

El conjunto de rutas independientes es el complemento del conjunto de rutas dependientes. Esto es, la intersección entre el conjunto de vértices $v(path_{l,m}) \cup fanout(v(path_{l,m}))$ y $v(path_{cp})$ es el conjunto vacío.

Por lo tanto, dada una red Booleana $\vec{G}$, se define el conjunto independiente de rutas del conjunto ruta crítica $path_{cp}$ como:

$$I(path_{cp}) = \{path_{l,m} \mid (v(path_{l,m}) \cup fanout(v(path_{l,m}))) \cap v(path_{cp}) = \emptyset\} \quad (4.5)$$

Puesto que $I(path_{cp})$ no pertenece al conjunto ruta crítica $path_{cp}$ entonces, el retardo de la ruta es menor que T, esto es, T_m es menor que T, para cada $path_{l,m}$ perteneciente a $I(path_{cp})$.

4.5. Optimización de un Circuito Empleando el Conjunto Ruta Crítica

Asumiendo que el retardo máximo $\overline{T}$ de un conjunto ruta crítica $path_{cp}$ es reducido mediante algún proceso de optimización. Este conjunto pierde la propiedad de definir el retardo máximo del circuito si:

- El retardo T_n de una ruta dependiente $path_{l,n}$ del conjunto ruta crítica $path_{cp}$ es mayor o igual al retardo de $path_{cp}$,
- es alcanzado el máximo retardo T_r de las rutas independientes del conjunto ruta crítica $path_{cp}$,
- o se dan las ambas condiciones previas simultáneamente.

En consecuencia, hay un intervalo de optimización $(\underline{T}, \overline{T}]$ para el retardo de un conjunto ruta crítica $path_{cp}$. $\underline{T}$ y $\overline{T}$ son los límites inferior y superior de T, respectivamente. El intervalo $(\underline{T}, \overline{T}]$ es el dominio de definición temporal del conjunto ruta crítica $path_{cp}$.

Más allá de este intervalo, un nuevo conjunto de rutas críticas determinan el retardo máximo del circuito $\vec{G}$. En este sentido, durante el proceso de optimización para reducir el retardo de un circuito, el conjunto ruta crítica es dinámico. Por ello, es necesario introducir un índice i en las fórmulas 4.2–4.5 con el objeto de diferenciar los distintos conjuntos: T, $path_{cp}$, $D(path_{cp})$ y $I(path_{cp})$ ahora se nombran T_i, $path_{cp,i}$, $D(path_{cp,i})$ y $I(path_{cp,i})$, respectivamente. Para cada nuevo conjunto $path_{cp,j}$, un nuevo T_j será encontrado.

Definición 8 (Límite Inferior del Retardo del Conjunto Ruta Crítica). *El límite inferior del retardo de un conjunto ruta crítica se expresa en el dominio de definición como el mínimo retardo, tal que, desde este retardo en adelante, el conjunto ruta crítica* $path_{cp,i}$ *pierde la condición de ser la ruta crítica del circuito* $\vec{G}$.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

De aquí que se obtiene el límite inferior $\underline{T_i}$ como una solución de la siguiente optimización de punto simple:

$$\begin{aligned} Minimise \quad & \text{the delay time of } path_{cp,i},\ T(path_{cp,i}) \\ Minimise \quad & \text{the active area, } \mathcal{A}, \text{ of } path_{cp,i} \\ s.t. \quad & \text{(i) a lower bound delay time defined as:} \\ & max(T(path_{l,m})),\ \forall\, path_{l,m} \in I(path_{cp,i}) \\ & \text{(ii) } T(path_{cp,i}) > T(path_{u,v}),\ \forall\, path_{u,v} \in D(path_{cp,i}) \end{aligned} \quad (4.6)$$

La optimización del retardo del conjunto ruta crítica $path_{cp,i}$ posee un límite inferior que es el máximo retardo de las rutas independientes del conjunto ruta crítica. El retardo y el área activa son minimizados. El retardo es restringido para no ser más rápido que el retardo de las rutas dependientes del conjunto ruta crítica.

4.6. Curva de Prestaciones de un Circuito

Dominio de Definición de una Ruta Crítica

En el apartado previo, se ha introducido el dominio de definición temporal de un conjunto ruta crítica. Dado un $path_{cp,i}$, se ha minimizado el retardo T_i aplicando la ecuación 4.6, y ello ha derivado en el retardo inferior de la ruta. El par $(\underline{T_i}, \underline{\mathcal{A}_i})$ es un punto de la curva de prestaciones — área activa $\mathcal{A}_i$ frente a retardo T_i. El retardo menor de la ruta $\underline{T_i}$ no está incluido en el dominio de definición temporal de $path_{cp,i}$.

Dado un tamaño de driver para todas las puertas del circuito, en los puntos no inferiores del espacio de diseño, el retardo y el área activa son mínimos. En particular, se representa el componente de retardo T de un punto no inferior como $\check{T}$. En el menor retardo $\underline{T_i}$, de la ruta crítica de un circuito $\vec{G}$, se define otro nuevo conjunto $path_{cp,j}$ pues hay que unir a la ruta crítica $path_{cp,i}$ todas aquellas rutas dependientes e independientes que alcanzan dicho valor de retardo. El dominio de definición de $path_{cp,j}$ es ahora $(\underline{T_j}, \underline{\check{T}_i}]$, donde el límite superior es $\underline{\check{T}_i}$, y el límite inferior es la solución de aplicar nuevamente la ecuación 4.6 a $path_{cp,j}$. Además, el retardo menor $\underline{T_j}$ no es incluido en el dominio de definición de $path_{cp,j}$.

Aplicando de forma sucesiva la ecuación 4.6 a la secuencia apropiada de rutas críticas se obtiene la curva de prestaciones de un circuito. La búsqueda de esta secuencia de conjuntos de rutas críticas ($path_{cp,0}$, $path_{cp,1}$, ..., $path_{cp,i}$, $path_{cp,j}$, ...) requiere un proceso de actualización del conjunto ruta crítica, para que la curva de prestaciones sea calculada, y además es necesario un conjunto ruta crítica $path_{cp,0}$ de inicio.

En este apartado se presenta al mapeado de mínima área activa y retardo. A partir de este punto se define el conjunto ruta crítica de comienzo $path_{cp,0}$. También se presenta un método — y su implementación algorítmica — para actualizar el conjunto ruta crítica.

4.6.1. Mapeado a Mínima $\mathcal{A}$rea Activa

Librería de Puertas Lógicas

Sea $LL = \{1, \ldots, N\}$ un conjunto de puertas lógicas — NAND, NOR, INV, ... — esto es, una librería de puertas lógicas. N es el número de puertas lógicas en

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

la librería LL. Para cada puerta lógica, hay varias versiones de su implementación[7] física. Sea $L_k = \{1, \ldots, M_k\}$ un conjunto de implementaciones físicas de una puerta lógica k, $k \in LL$. Sea $L = \{L_k \mid \forall\, k \in LL\}$ una librería de mapeado.

Dada una ruta crítica $\vec{G}$ correspondiente a un circuito de lógica combinacional obtenido mediante un procedimiento de síntesis tecnológicamente independiente, e.g. $\vec{G}$ es una descripción estructural NAND–INV, un conjunto de restricciones de diseño y una librería L; un mapeado M es una transformación de $\vec{G}$ en una *net–list*[8] de puertas lógicas N, que es formalmente equivalente a $\vec{G}$ y satisface las restricciones de diseño.

La medida de las prestaciones de una puerta lógica k, $k \in LL$, es el conjunto de pares (retardo, área activa). Ello se expresa formalmente tal como sigue:

$$\mathcal{L}_k = \{(\tau_{k,l}, \mathcal{A}_{k,l}) \mid k \in LL,\ \forall\, l \in L_k\} \tag{4.7}$$

donde L_k es el conjunto de versiones de descripciones físicas de la puerta lógica k. Por ejemplo, las puertas BUF1 y BUF4 en una librería de células estándar son las versiones $1\times$ y $2\times$ de un buffer no inversor BUF. Una librería de prestaciones $\mathcal{L}$ se emplea para mapear $\vec{G}$ en N mediante la transformación M, y se define como:

Versiones de Puerta

$$\mathcal{L} = \{\mathcal{L}_k \mid \forall\, k \in LL\} \tag{4.8}$$

En una librería de células estándar, las prestaciones retardo y área activa son funciones no continuas del tamaño de driver de puerta. Sin embargo, para las células *full–custom*, sus prestaciones son funciones continuas del driver de puerta.

Definición 9 (Librería Convexa Objetivo). *Una librería de puertas de prestaciones $\mathcal{L}$ es una librería convexa objetivo $\breve{\mathcal{L}}$ cuando para cada una puerta lógica k, $k \in LL$, la función área activa es una función convexa del retardo:*

$$\breve{\mathcal{L}} = \{(\tau_k, \breve{\mathcal{A}}_k) \mid \breve{\mathcal{A}}_k = \breve{\mathcal{A}}(\tau_k),\ \forall\, k \in LL\} \tag{4.9}$$

donde $\breve{\mathcal{A}}(\tau_k)$ es una función convexa de τ_k.

Definición 10 (Espacio de Diseño). *Dada una ruta crítica de un circuito $\vec{G}$, su espacio de diseño es definido por una función C, donde una transformación M determina la 3–tupla (retardo, área activa y potencia de consumo), $(\mathcal{T}, \mathcal{A}, \mathcal{P})$, esto es $C : M \longrightarrow \mathcal{T} \times \mathcal{A} \times \mathcal{P}$ para una condición carga salida de $\vec{G}$.*

Definición 11 (Puertas de Mínima Área Activa). *Se define las puertas de mínima área activa $\mathcal{L}_{\underline{\mathcal{A}}}$ como sigue:*

$$\mathcal{L}_{\underline{\mathcal{A}}} = \{j \mid j \in \mathcal{L}_{k,\underline{\mathcal{A}}},\ \forall\, k \in LL\} \tag{4.10}$$

$$\mathcal{L}_{k,\underline{\mathcal{A}}} = \{(\tau_k, \underline{\mathcal{A}_k}) \mid \underline{\mathcal{A}_k} = min\ (\mathcal{A}_{k,l}),\ k \in LL,\ \forall\, l \in L_k\} \tag{4.11}$$

donde $min\ (\mathcal{A}_{k,l})$ es el área activa mínima.

[7] Traducción directa del término anglosajón *layout*. Con este término se identifica a la vista que representa la disposición de los elementos que conforman los dispositivos e interconexiones de un diseño desde un punto de vista físico.

[8] Una *net–list* es una descripción estructural donde se identifican las implementaciones físicas de los elementos que la componen y sus interconexiones.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

Definición 12 (Curva de Prestaciones). *En particular, todos los puntos no inferiores del espacio de diseño $C_{tradeoff}$ definen el mapeado óptimo $\breve{M}$ en retardo y área activa $(\breve{T}, \breve{A})$, es decir:*

$$C_{tradeoff} \subseteq C,\ C_{tradeoff} = \{\breve{M} \mid C(\breve{M}) \in (\breve{T}, \breve{A})\}, \tag{4.12}$$

donde $\breve{T}$ y $\breve{A}$ son los puntos no inferiores (retardo y área activa) del espacio de diseño C.

Teorema 1 (Mapeado de Mínima Área Activa). *Dada una red Booleana multinivel $\vec{G}$ y una librería de puertas convexa $\breve{\mathcal{L}}$, el mapeado $M_{\underline{A}}$ de $\vec{G}$ en N con esas puertas de mínima área activa es un punto no inferior del espacio de diseño C:*

Punto No Inferior

$$M_{\underline{A}} \in C_{tradeoff} \Leftrightarrow C(M_{\underline{A}}) \in (\breve{T}, \breve{A}) \tag{4.13}$$

Demostración. El área activa total de una *net–list* N son puntos no inferiores de C, es decir $\mathcal{A} = \sum_{n \in N} \mathcal{A}(n)$, donde $\mathcal{A}(n)$ es el área activa de la puerta n. Debido a que $\mathcal{A}(n)$ es la mínima área activa de cada puerta n, $\mathcal{A}$ pertenece a $\breve{A}$.

Para demostrar que T pertenece a $\breve{T}$, por contradicción, se asume que existe un mapeado M con igual área activa pero menor retardo. En tal caso, la librería $\breve{\mathcal{L}}$ no sería una librería convexa. □

Desde el mapeado inicial $M_{\underline{A}}$, se puede solamente reducir el retardo, puesto que $M_{\underline{A}}$ emplea las puertas de mínima área de la librería $\breve{\mathcal{L}}$.

Por lo tanto, $M_{\underline{A}}$ es un punto final de mapeado del espacio de diseño C y la curva de prestaciones $C_{tradeoff}$, por ello el cálculo de este punto no inferior del espacio de diseño no requiere proceso de optimización alguno. Además, la importancia de este punto es trascendental, pues sin conocer el conjunto ruta crítica se obtiene un punto de la curva de prestaciones.

4.6.2. Curva de Prestaciones

Desde el punto de la curva de prestaciones $M_{\underline{A}}$ de un circuito de lógica combinacional $\vec{G}$, un conjunto ruta crítica se define mediante $path_{cp,0}$, donde el retardo de ruta $\overline{T_0}$ es $T(path_{cp,0})$, y mediante la resolución de la ecuación 4.6, se obtiene un segundo punto de mapeado de la curva de prestaciones $\breve{M}_0$ para el conjunto ruta crítica $path_{cp,0}$ y con un retardo inferior $\underline{T_0}$. Además, $(\underline{T_0}, \overline{T_0}]$ es el dominio de definición del conjunto ruta crítica $path_{cp,0}$, el intervalo $(M_{\underline{A}}, \breve{M}_0]$ es el dominio de definición del mapeado (o dominio de definición) del conjunto ruta crítica $path_{cp,0}$.

Ruta Crítica Dinámica

La ruta crítica de $\vec{G}$ cambian a medida que se obtienen los puntos de la curva de prestaciones. Por lo tanto, el conjunto inicial definido como conjunto ruta crítica $path_{cp,0}$ de $\vec{G}$ debe ser redefinido — como una ruta $path_{cp,i}$ dinámica — siempre que la optimización progrese.

El conjunto ruta crítica $path_{cp,i}$ se actualiza como sigue. Si el retardo inferior $\underline{\breve{T}_i}$ iguala al máximo retardo de alguna ruta independiente del conjunto ruta crítica, entonces el nuevo subconjunto $path_{cp,j}$ de la red $\vec{G}$ es el resultado de unirlos con el conjunto ruta crítica $path_{cp,i}$. Además, los retardos de esas rutas incluidas han

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

de ser contemplados en el cómputo del retardo de la nueva ruta crítica $path_{cp,j}$. Esto se expresa formalmente como:

$$\begin{aligned} path_{cp,j} = \; & \{path_{cp,i} \cup path_{o,p} \cup path_{l,m} \mid \\ & T(path_{o,p}) \geq \underline{\check{T}_i},\ \forall path_{o,p} \in D(path_{cp,i}) \\ & T(path_{l,m}) = \underline{\check{T}_i},\ \forall path_{l,m} \in I(path_{cp,i})\} \end{aligned} \tag{4.14}$$

$UpdatePath_{cp,i}()$
  for each independent path $path_{l,m}$ on critical path set $path_{cp,i}$
    if $T(path_{l,m}) \geq T(path_{cp,i})$ then
      $path_{cp,i} = path_{cp,i} \cup path_{l,m}$;
    end if;
  end for;
  for each dependent path $path_{o,p}$ on critical path set $path_{cp,i}$
    if $T(path_{o,p}) \geq T(path_{cp,i})$ then
      $path_{cp,i} = path_{cp,i} \cup path_{o,p}$;
    end if;
  end for;
end $UpdatePath_{cp,i}()$

Figura 4.3: Algoritmo de actualización del conjunto ruta crítica.

La figura 4.3 muestra el algoritmo para actualizar el conjunto ruta crítica $path_{cp,i}$. Más allá del punto de la curva de prestaciones $\check{M}_0$, el circuito presenta un nuevo conjunto ruta crítica $path_{cp,1}$. Una posterior optimización mediante la aplicación de 4.6 provee un nuevo punto de mapeado de la curva de prestaciones $\check{M}_1$. El intervalo $(\check{M}_0, \check{M}_1]$ es el dominio de definición del mapeado del conjunto ruta crítica $path_{cp,1}$. Sobre el límite superior de $\check{M}_1$, el circuito presenta un nuevo conjunto ruta crítica $path_{cp,2}$, y así sucesivamente. El proceso de cálculo finaliza cuando se alcance un punto para el cual no hay reducción del retardo con mínima área activa.

Dominio de Definición del Mapeado

4.6.3. Algoritmo de Cálculo de la Curva de Prestaciones

$C_{tradeoff}()$ es el algoritmo propuesto para calcular la curva entera de retardo frente a área activa de un conjunto ruta crítica (véase la figura 4.4), cuando una red $\vec{G}$ es mapeada a una librería de prestaciones convexa $\breve{\mathcal{L}}$.

El algoritmo encuentra el mapeado y el dominio de definición temporal para cada conjunto ruta crítica $path_{cp,i}$ de un circuito de lógica combinacional, donde se ha ignorado el problema de la ruta falsa [106]. La salida del procedimiento $C_{tradeoff}()$ es un conjunto de pares $(\check{T}, \check{A})$, y para cada pareja de pares se define un mapeado $\check{M}$, $C_{tradeoff} = \{\check{M} \mid C(\check{M}) \in (\check{T}, \check{A})\}$.

Algoritmo de Exploración de la Curva de Prestaciones

[106] H.C. Chen and D.H.C. Du. Path Sensitization in Critical Path Problem. *IEEE Trans. Comput.–Aided Des. Integr. Circuits Syst.*, 12(2):196–207, 1993.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

$C_{tradeoff}()$
  given a $\vec{G} = (V, \vec{E}, \tau)$;
  map $\vec{G}$ to a $\mathcal{L}_{\underline{A}}$, $\breve{M}_0 = M_{\underline{A}}$;
  obtain the critical path set $path_{cp,i}$;
  do
    update $\breve{T}_i = T(path_{cp,i})$;
    optimise $path_{cp,i}$ by Equation 4.6;
    $UpdatePath_{cp,i}(path_{cp,i})$;
    update $\breve{T}_j = T(path_{cp,i})$;
  while $\breve{T}_i \neq \breve{T}_j$
end $C_{tradeoff}()$

Figura 4.4: Algoritmo para la generación de la curva completa de prestaciones.

El dominio de definición de un conjunto ruta crítica $path_{cp,i}$ está contenido dentro de dos puntos sucesivos de mapeado $\breve{M}_i$ y $\breve{M}_{i+1}$. Dado el dominio temporal de definición $(\breve{T}_i, \breve{T}_{i+1}]$ de un conjunto ruta crítica $path_{cp,i}$, dos puntos de la curva de prestaciones son calculados como $C(\breve{M}_i) = (\breve{T}_i, \breve{A}_i)$ y $C(\breve{M}_{i+1}) = (\breve{T}_{i+1}, \breve{A}_{i+1})$.

En un dominio de definición temporal $(\breve{T}_i, \breve{T}_{i+1}]$ de un conjunto ruta crítica $path_{cp,i}$ se calcula la curva de prestaciones, con una resolución, como un problema de dimensionado sobre una red invariante. La red representa en el conjunto ruta crítica $path_{cp,i}$ no necesita ser actualizada en el intervalo $(\breve{T}_i, \breve{T}_{i+1}]$. Por lo tanto, cualquier mapeado $\breve{M}_j$ dentro del dominio de definición del conjunto ruta crítica $path_{cp,i}$ depende de un subconjunto invariante de la red $\vec{G}$.

Cálculo de la Curva de Prestaciones

La curva completa de prestaciones de un circuito de lógica combinacional $\vec{G}$ se calcula mediante sucesivas optimizaciones de dimensionado sobre diferentes rutas críticas. Inicialmente, en el mapeado $\breve{M}_{\underline{A}}$, el conjunto ruta crítica $path_{cp,0}$ representa un subconjunto de la red $\vec{G}$. Mientras que los sucesivos puntos de mapeado $\breve{M}_i$ se obtienen, la subred crece debido a que el conjunto ruta crítica $path_{cp,i}$ es actualizado al unir otras rutas.

Cualquier método de optimización para resolver 4.6 es válido, tales como optimizadores heurísticos o combinados heurísticos y algorítmicos, Programación Lineal y No Lineal, entre otros.

El problema de dimensionado de puerta es una transformación particular de una red $\vec{G}$ en una *net–list* N de puertas. Dada una condición de carga a la salida, las restricciones del diseño son formuladas en términos de área activa frente a prestaciones de retardo.

Para mapear $\vec{G}$ en N, es necesaria una librería convexa objetivo. La librería convexa objetivo $\breve{\mathcal{L}}$ contiene células *full–custom* para puertas lógicas.

Factor de Driver

Puesto que el problema de dimensionado de puerta se establece sobre puertas de tamaño variable, es conveniente definir la capacidad de tamaño de *driver* (factor β) de la puerta.

El tamaño de driver o factor β de una puerta crece linealmente con el ancho de sus transistores internos. Las capacidades internas de la puerta se incrementan

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

linealmente también con el ancho de los transistores internos, y consecuentemente con el factor de tamaño de driver.

En un mapeado M_i de $\vec{G}$ en N, cada puerta del circuito posee un factor de tamaño de driver particular. Para una condición de carga de salida, el cálculo de la 2–tupla $(\mathcal{T}, \mathcal{A})$ depende del retardo de propagación y del área activa de las puertas.

En este apartado, se detallará el algoritmo propuesto (véase el algoritmo de la figura 4.4) para calcular la curva de prestaciones entera para el dimensionado de puerta con una técnica de Programación Lineal. La Programación Lineal es empleada como procedimiento de optimización para un punto simple para la red completa $\vec{G}$ y el conjunto ruta crítica $path_{cp,i}$ — un subconjunto de tamaño variable de la red. Además, se presenta una primera descripción de la complejidad computacional del problema equivalente descrito en Programación Lineal en términos de número de variables y restricciones.

4.6.4. Modelo de Retardo y Área Activa de Puerta

Función de Retardo Convexa

En esta aportación se emplea una función de retardo de propagación convexa, tal como el presentado en [91],[97]. La función emplea el peor caso de retardo de propagación de una puerta lógica, tanto respecto a sus entradas y con respecto a los tiempos de subida y bajada. El retardo total del circuito es determinado indiferente si las rutas críticas son sensibilizables [106]–[107]; pueden llegar a ser rutas falsas durante el proceso de optimización [107]–[108]; o son rutas falsas que pueden llegar a ser sensibles [107],[109],[110]. Se modela el retardo del circuito mediante la identificación de la ruta más larga en la red Booleana [111]. A continuación se muestra la notación empleada en la descripción del modelado del retardo de puerta:

τ: retardo de puerta.

C: capacidad.

[91] M.R.C.M. Berkelaar, P.H.W. Buurman, and J.A.G. Jess. Computing the Entire Active Area/Power Consumption versus Delay Tradeoff Curve for Gate Sizing with a Piecewise Linear Simulator. *IEEE Trans. Comput.–Aided Des. Integr. Circuits Syst.*, 15(11):1424–1434, 1996.

[97] M.R.C.M. Berkelaar and J.A.G. Jess. Gate Sizing in MOS Digital Circuits with Linear Programming. *Proc. European Design Automation Conf.*, pages 217–221, March 1990.

[106] H.C. Chen and D.H.C. Du. Path Sensitization in Critical Path Problem. *IEEE Trans. Comput.–Aided Des. Integr. Circuits Syst.*, 12(2):196–207, 1993.

[107] H.R. Lin and T.T. Hwang. On Determining Sensitization Criterion in an Iterative Gate Sizing Process. *IEEE Trans. Comput.–Aided Des. Integr. Circuits Syst.*, 18(2):231–238, 1999.

[108] H.R. Lin and T.T. Hwang. Dynamical Identification of Critical Paths for Iterative Gate Sizing. *Proc. ACM/IEEE Int. Conf. Computer–Aided Design*, pages 481–484, November 1994.

[109] H.C. Chen, D.H.C. Du, and L.R. Liu. Critical Path Selection for Performance Optimization. *IEEE Trans. Comput.–Aided Des. Integr. Circuits Syst.*, 12(2):185–195, 1993.

[110] S.T. Huang, T.M. Parng, and J.M. Shyu. A New Method of Identifying Critical Paths for Performance Optimization. *Proc. European Design Automation Conf.*, pages 455–459, February 1993.

[111] F. Chen-Lian, , and J. Wen-Ben. Timing Optimization by Gate Resizing and Critical Path Identification. *IEEE Trans. Comput.–Aided Des. Integr. Circuits Syst.*, 14(2):201–217, 1995.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

k: constante.

β: factor de velocidad[9].

Un modelo de retardo, básico y ampliamente utilizado es:

$$\begin{aligned} \tau_{gate} &= \tau_{gate,int} + \tau(C_{load}, \beta_{gate}) \\ C_{load} &= C_{wire} + \sum_{i \in fanout(gate)} C_i \\ C_i &= \beta_i \times C_{i,int} \end{aligned} \tag{4.15}$$

Donde C_{wire} es la capacidad de cableado y C_i es la capacidad conectada a la puerta i. La capacidad conectada a la puerta C_i crece linealmente con el tamaño del factor de velocidad β_i. $C_{i,int}$ es la capacidad interna de una puerta i con tamaño unitario en su factor de velocidad β_i.

El factor de velocidad β_{gate} de una puerta está relacionado con el retardo de puerta τ_{gate} tal como sigue:

$$\tau(C_{load}, \beta_{gate}) = k \times C_{load}/\beta_{gate} \tag{4.16}$$

A mayor factor de velocidad β_{gate}, menor es el retardo de puerta τ_{gate}. De hecho, el retardo interno de puerta $\tau_{gate,int}$ es casi constante porque las capacidades internas de la puerta crecen linealmente con el factor de velocidad β_{gate}. Sin embargo, el término $k \times C_{load}/\beta_{gate}$ decrece con ese mismo factor. El modelo de retardo de puerta se obtiene mediante la combinación de las ecuaciones 4.15 y 4.16.

Modelo No lineal

Como se muestra en la ecuación 4.16, el modelo de retardo de puerta es no lineal. Con el objeto de adaptar este modelo a una técnica de Programación Lineal, la función ha de ser linealizada acotando el error en torno al punto de evaluación. En la referencia [91] se muestra una forma de realizar la linealización. En la figura 4.5 se muestra la linealización para una puerta NAND de dos entradas sobre tecnología CMOS de $0'18\mu m$[10] bajo varias condiciones de carga y con su linealización en 2 tramos.

Área Activa

El área activa de una puerta crece linealmente con su tamaño de driver. El área activa total de un circuito es la sumatoria de área activa de todas las puertas en del circuito, como sigue:

$$\mathcal{A} = \sum_{i \in N} a_i \beta_i \tag{4.17}$$

Donde a_i es una constante de proporcionalidad entre el área activa y el factor de velocidad β_i para la puerta i.

[9] Traducción directa del término anglosajón *speed factor*.

[10] TSMC Logic 0'18um Process 1'8V Supply Voltage Generic II Spice.

[91] M.R.C.M. Berkelaar, P.H.W. Buurman, and J.A.G. Jess. Computing the Entire Active Area/Power Consumption versus Delay Tradeoff Curve for Gate Sizing with a Piecewise Linear Simulator. *IEEE Trans. Comput.–Aided Des. Integr. Circuits Syst.*, 15(11):1424–1434, 1996.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

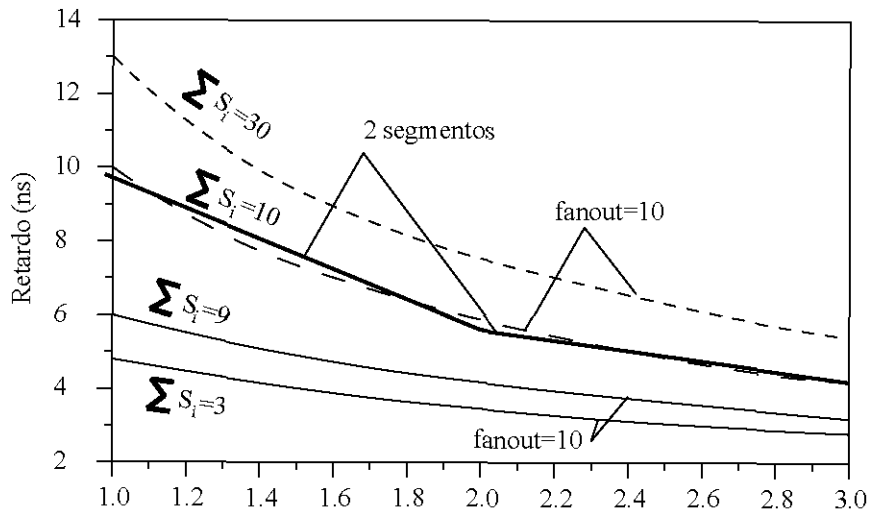


Figura 4.5: Curva de retardo de una NAND bajo diferentes cargas frente al factor de velocidad.

4.6.5. Curva de Prestaciones empleando Programación Lineal

Se emplea una técnica de Programación Lineal como optimización de un punto único de la función multiobjetivo presentada en la ecuación 4.6. La función objetivo de área activa es un modelo lineal del factor de velocidad β, como fue presentado en la ecuación 4.17. Sin embargo, el modelo completo de retardo es una función no lineal de β_{gate}

Modelo de Retardo No lineal

$$\tau_{gate} = \tau_{gate,int} + k \times \frac{C_{wire} + \sum_{i \in fanout(gate)} \beta_i \times C_{i,int}}{\beta_{gate}} \tag{4.18}$$

Este modelo de retardo no lineal es linealizado con la precisión deseada mediante un ajuste lineal a tramos para convertirse en:

$$\tau_{gate} \geq k_1 - k_2 \beta_{gate} + k_3 \sum_{i \in fanout(gate)} \beta_i C_{int,i} \tag{4.19}$$

Consecuentemente, cuando la Programación Lineal minimice la función multiobjetivo $c_A \mathcal{A} + c_T \mathcal{T}$, la siguiente restricción a añadir es a la ecuación 4.6, para cada puerta del circuito:

Función Objetivo

- Los n modelos de retardo linealizados

$$\begin{aligned} \tau_{gate} &\geq k_{1,1} - k_{1,2}\beta_{gate} + k_{1,3} \sum_{i \in fanout(gate)} \beta_i C_{int,i} \\ &\ldots \\ \tau_{gate} &\geq k_{n,1} - k_{n,2}\beta_{gate} + k_{n,3} \sum_{i \in fanout(gate)} \beta_i C_{int,i} \end{aligned}$$

- Límite del factor de velocidad

$$\beta_{gate} \in [\underline{\beta_{gate}}, \overline{\beta_{gate}}]$$

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

Donde $\underline{\beta_{gate}}$ y $\overline{\beta_{gate}}$ son el límite inferior y superior del factor de velocidad, respectivamente.

- Definición del tiempo de propagación

$$T_{gate} \geq T_j + \tau_{gate}, \forall\, j \in fanin(gate)$$

Algoritmo de Cálculo de la Curva de Prestaciones

El algoritmo propuesto $C_{tradeoff}()$ es reescrito con el objeto de embeber la técnica de Programación Lineal (LP) (véase el algoritmo $LPC_{tradeoff}()$ en la figura 4.6). Además y con el propósito de realizar comparativas, se presenta en la figura 4.7 el algoritmo $LPSizingA()$ que realiza la formulación en Programación Lineal del problema de dimensionado de puerta. En ambos algoritmos, se precalcula una aproximación lineal a tramos de la formulación no lineal del retardo de propagación.

4.6.6. Complejidad de la Optimización

El esfuerzo computacional de la optimización se obtiene cuantificando el número de variables necesarias para calcular cada punto de la curva de prestaciones. Se supone que la complejidad de la optimización está directamente relacionada con el tamaño del problema de optimización resuelto mediante Programación Lineal, esto es, el número de variables y restricciones.

Esfuerzo Computacional

Para una red $\vec{G}$ con $|V|$ vértices (uno por cada puerta) y $|E|$ lados (uno por cada conexión) el tamaño del problema resuelto mediante técnicas de Programación Lineal es conocido. El número de restricciones es $|E|+(3+n)|V|+3$, como sigue:

- 1 para expresar el retardo total $\mathcal{T}$,
- 1 para limitar el retardo total $\mathcal{T}$,
- 1 para expresar el área activa total $\mathcal{A}$,
- n del modelo lineal de propagación del retardo por vértice ($n|V|$),
- 2 de la limitación del factor de velocidad por vértice ($2|V|$),
- 1 para limitar la definición de tiempo de propagación por vértice ($|V|$), y
- 1 por cada relación con un predecesor ($|E|$).

Síntesis de Circuitos

En todos los circuitos sintetizados el número de entradas es uno o dos y la carga se limita a tres, así que $|E| < k|V|$, donde $k > 1$, y por tanto el orden es lineal con el número de puertas ($(3 + n + k)|V| + 3$).

Para el conjunto ruta crítica $path_{cp,i}$, en vez de limitar el retardo total $\mathcal{T}$ al máximo retardo de las salidas primarias, éste es restringido al retardo de la rutas dependientes del conjunto ruta crítica.

Además, el límite más bajo del retardo total es el retardo máximo de las rutas independientes. Entonces, el número de restricciones distintas es como sigue:

- n por modelo lineal de propagación del retardo por vértice de $path_{cp,i}$ ($< n|V|$),
- 2 de la limitación del factor de velocidad a cada vértice de $path_{cp,i}$ ($< 2|V|$),

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

$LPC_{tradeoff}()$
given a $\vec{G} = (V, \vec{E}, \tau)$;
map $\vec{G}$ to $\mathcal{L}_{\underline{A}}$, $\breve{M}_i = M_{\underline{A}}$;
obtain the critical path set $path_{cp,i}$;
do
update $\mathcal{T}_i = T(path_{cp,i})$;
minimise $c_A\mathcal{A} + c_T\mathcal{T}$ of $path_{cp,i}$;
s.t.
$\forall\ gate\ \in\ path_{cp,i}$:
The n modelos de propagación lineales

$$\begin{aligned} \tau_{gate} &\geq k_{1,1} - k_{1,2}\beta_{gate} + k_{1,3} \sum_{i \in fanout(gate)} \beta_i C_{int,i} \\ &\dots \\ \tau_{gate} &\geq k_{n,1} - k_{n,2}\beta_{gate} + k_{n,3} \sum_{i \in fanout(gate)} \beta_i C_{int,i} \end{aligned}$$

Limitación del factor de velocidad

$$\beta_{gate}\ \in\ [\underline{\beta_{gate}}, \overline{\beta_{gate}}]$$

Definiciones de los tiempos de propagación

$$T_{gate}\ \geq\ T_j\ +\ \tau_{gate},\ \forall\ j\ \in\ fanin(gate)$$

s.t.
Máximo tiempo de propagación

$$\begin{aligned} \mathcal{T}\ &\geq\ T(path_{o,p}), \forall\ path_{o,p}\ \in\ D(path_{cp,i}) \\ \mathcal{T}\ &\geq\ max(T(path_{l,m})), \forall\ path_{l,m}\ \in\ I(path_{cp,i}) \end{aligned}$$

$\breve{\mathcal{T}}_j = T(path_{cp,i})$;
$UpdatePath_{cp,i}()$;
while $\breve{\mathcal{T}}_i \neq \breve{\mathcal{T}}_j$
end $LPC_{tradeoff}()$

Figura 4.6: Algoritmo de generación de la curva completa de prestaciones empleando la técnica de Programación Lineal.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

$LPSizingA()$

minimise $c_A\mathcal{A} + c_T\mathcal{T}$ of $\vec{G}$;

s.t.

$\forall\ gate\ \in\ \vec{G}$:

The n linear propagation delay models

$$\begin{aligned} \tau_{gate} &\geq k_{1,1} - k_{1,2}\beta_{gate} + k_{1,3} \sum_{i \in fanout(gate)} \beta_i C_{int,i} \\ &\dots \\ \tau_{gate} &\geq k_{n,1} - k_{n,2}\beta_{gate} + k_{n,3} \sum_{i \in fanout(gate)} \beta_i C_{int,i} \end{aligned}$$

The drive size factor limitation

$$\beta_{gate}\ \in\ [\underline{\beta_{gate}}, \overline{\beta_{gate}}]$$

Definiciones de los tiempos de propagación

$$T_{gate}\ \geq\ T_j\ +\ \tau_{gate},\ \forall\ j\ \in\ fanin(gate)$$

s.t.

Máximo tiempo de propagación

$$\mathcal{T}\ \geq\ T(v_i), \forall\ v_i\ \in\ PO(\vec{G})$$

end $LPSizingA()$

Figura 4.7: Algoritmo de dimensionado de puerta empleando técnicas de Programación Lineal.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

- 1 para limitar la definición del tiempo de propagación de cada vértice de $path_{cp,i}$ ($< |V|$), y
- 1 por cada relación de predecesión de $path_{cp,i}$ ($< |E|$).

Número de Variables del Problema

Se emplean $3|V|+2$ variables cuando la red $\vec{G}$ es resuelta mediante técnicas de Programación Lineal, como sigue:

- 3 variables por vértice para soportar β_{gate}, τ_{gate} y T_{gate} ($3|V|$), y
- 2 variables para el retardo total $\mathcal{T}$ y el área activa total $\mathcal{A}$.

En resumen, puesto que el conjunto ruta crítica $path_{cp,i}$ es un subconjunto de la red $\vec{G}$, el tamaño del problema a resolver mediante Programación Lineal se reduce sustancialmente.

Con el objetivo de demostrar la validez de la metodología propuesta para calcular la curva entera de prestaciones área activa frente a retardo, se ha aplicado al conjunto de circuitos de dos niveles y múltiples niveles del banco de pruebas MCNC'91 [51].

A continuación, se discutirán los aspectos numéricos de los resultados que se han alcanzado en términos de número de variables y del tiempo de ejecución. La curva de prestaciones del área activa frente a retardo pare el circuito bw del banco de pruebas MCNC'91 *benchmark* se muestra en la figura 4.11. Además, se ilustra el retardo de la ruta crítica del circuito bw frente a la suma de factores de velocidad bajo distintas condiciones de carga.

Cada circuito del banco de pruebas MCNC fue preprocesado con la herramienta de síntesis y mapeado lógico de sistemas MISII [70] a una librería de puertas lógicas — NAND, NOR, INV — con mínima área activa. El proceso tecnológico empleado es el CMOS 0'18 μm de TSMC con una tensión de alimentación de 1'8 V. Se han calculado las curvas completas de prestaciones de área activa frente a retardo para la red completa $\vec{G}$ y para el conjunto ruta crítica $path_{cp,i}$. Las ecuaciones de retardo de propagación para las células *full–custom* de las puertas lógicas NAND, NOR, INV, están expresadas en unidades de picosegundos (ps), son como siguen:

Ecuaciones del Modelo de Retardo

$$\begin{aligned} \tau_{NAND2} &= 64'148 + (179'598 + \sum_i \beta_i \times 17'854)/\beta_{NAND2} \\ \tau_{NOR2} &= 69'275 + (222'847 + \sum_i \beta_i \times 21'344)/\beta_{NOR2} \\ \tau_{INV} &= 36'398 + (150'922 + \sum_i \beta_i \times 15'902)/\beta_{INV} \end{aligned} \tag{4.20}$$

[51] S. Yang. *Logic Synthesis and Optimization Benchmarks User Guide version 3.0.* Microelectronics Center of North Carolina, 1991.

[70] R.K. Brayton, R. Rudell, A.L. Sangiovanni-Vincentelli, and A. Wang. MIS: A Multiple Level Logic Optimization System. *IEEE Trans. Comput.–Aided Des. Integr. Circuits Syst.*, 6(11):1062–1081, 1987.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

4.6.7. Comparativa

Las tablas 4.1, 4.2 y 4.3 muestran las comparativas en términos de número de variables y tiempo de ejecución para calcular la curva de prestaciones completa. La primera columna da el nombre del circuito bajo prueba de acuerdo con el banco de pruebas MCNC'91. La columna segunda presenta la complejidad del circuito medida en número de puertas. El número total de variables, cuando la curva de prestaciones es calculada para la red $\vec{G}$ empleando Programación Lineal y aproximaciones lineales a tramos en la formulación del retardo, es presentada en la columna tercera. La cuarta columna ofrece el tiempo de CPU, en segundos, necesario para resolver el problema anterior. La quinta columna provee el número total de dominios de definición para el conjunto ruta crítica de cada circuito.

Debido a que la ruta crítica es dinámica, en las columnas sexta, séptima y octava se muestran el número mínimo, medio y máximo de variables empleadas respectivamente, cuando es generada la curva entera de prestaciones empleando la metodología propuesta [112], [88]. La novena columna exhibe el tiempo de CPU total de esta última aproximación. Finalmente, las columnas décima y undécima ilustran el ratio del número de variables y tiempo de CPU, respectivamente. El tiempo de CPU fue medido en un procesador Pentium III a 800 MHz, con 384 $MBytes$ de RAM.

El cálculo de la curva de prestaciones del problema completo — la red $\vec{G}$ — es comparado con el cálculo de la curva de prestaciones para el conjunto ruta crítica $path_{cp,i}$ en términos de tiempo de ejecución y esfuerzo computacional. Las propiedades de convergencia en el cálculo y las soluciones — puntos de la curva de prestaciones — son idénticas. Esto es así porque se emplean métodos numéricos idénticos. Lo último es debido a que el dimensionado solo afecta a esas puertas que pertenecen a la ruta crítica.

Número de Variables

El número de variables para determinar la curva completa de prestaciones de una red $\vec{G}$ es lineal con el número de puertas de un circuito. Esto es absolutamente natural porque, en las redes prácticas, el número de variables es $3|V|+2$. La figura 4.8 ilustra este comportamiento. Téngase en cuenta que ambos ejes de coordenadas están expresados en escala logarítmica. Los triángulos negros representan el número de variables empleadas para el método basado en el empleo de la red $\vec{G}$ completa y los diamantes para el método del conjunto ruta crítica $path_{cp,i}$. Se observa claramente qué método es mejor tanto para circuitos pequeños como para circuitos grandes en términos del número de variables. Del cálculo de la curva entera de prestaciones del conjunto ruta crítica $path_{cp,i}$ se deduce una reducción en el número de variables, entre el 1'1 y 7'3 veces menos sobre la media para el conjunto de datos del MCNC'91 y, predeciblemente, mayor será el ahorro cuanto más compleja sea la red.

Con un mayor número de puertas en el circuito, hay una diferencia grande del número de variables. Esto es, en los circuitos pequeños cada ruta en la red $\vec{G}$ contiene solamente unas pocas puertas, y la subred del conjunto ruta crítica $path_{cp,i}$

[112] J. Sosa, H. Navarro, V. de Armas, J. A. Montiel-Nelson, and R. Sarmiento. *A Novel Methodology for Efficient Computation of the Area/Power Consumption versus Delay Tradeoff Curve in Circuit Optimization Problems*, pages 190–195. Evolutionary Methods for Design Optimization and Control. International Center for Numerical Methods in Engineering, 2002.

[88] J.A. Montiel-Nelson, J. Sosa, H. Navarro, R. Sarmiento, and A. Nuñez. Efficient Method to Obtain the Entire Active Area against Circuit Delay Time Trade–Off Curve in Gate Sizing. In *IEE Proc. Circuits, Devices & Systems*, volume 152 (2), April 2005.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

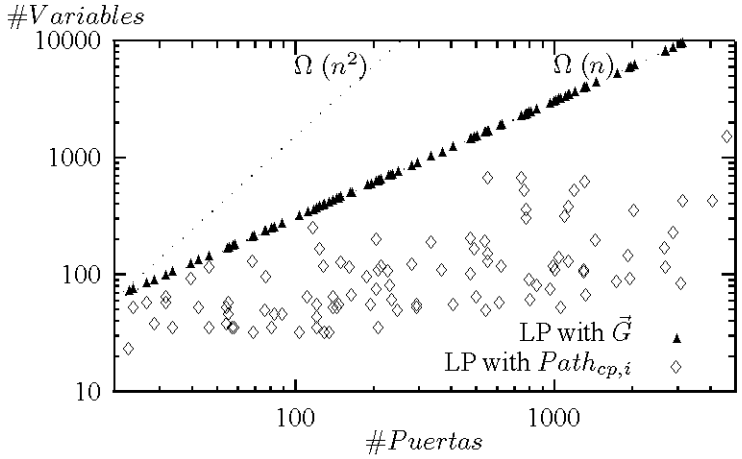


Figura 4.8: Número de variables frente al número de puertas.

es casi la red $\vec{G}$. Sin embargo, la optimización de circuitos grandes es menos restringida, y contienen, normalmente, varias rutas críticas con un alto número de puertas. En estos casos, la mayor parte del esfuerzo computacional es dedicado exclusivamente a optimizar las grandes rutas críticas del circuito.

Variación del Número de Variables

La figura 4.9 además ilustra que, a lo largo de la optimización de un conjunto ruta crítica $path_{cp,i}$, con un gran número de puertas del circuito, hay una variación grande del número de variables (vea las columnas sexta, séptima y octava de la tablas 4.1, 4.2 y 4.3). Ambos ejes de la figura 4.9 se presentan en escala lineal. Este resultado se explica puesto que la optimización en circuitos pequeños está muy restringida, debido a que el número de conjuntos ruta crítica $path_{cp,i}$ es reducido. En otras palabras, en los circuitos pequeños, el tamaño del conjunto ruta crítica es aproximadamente el tamaño de $\vec{G}$.

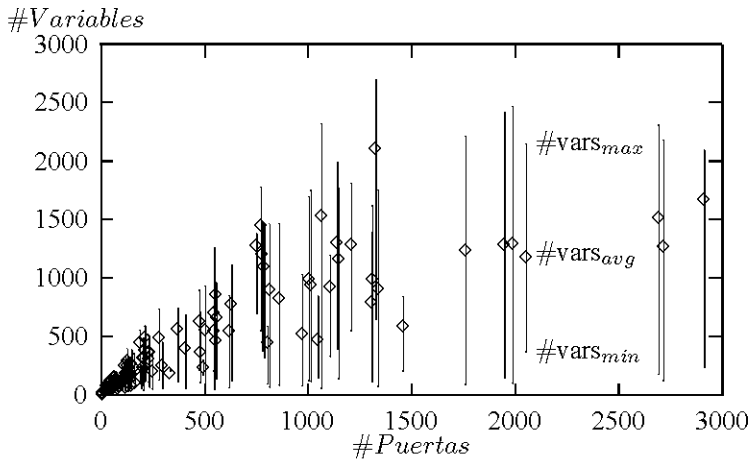


Figura 4.9: Variaciones en el número de variables frente al número de puertas.

Esfuerzo Computacional

La figura 4.10 ilustra el tiempo de CPU (en segundos) frente al tamaño del problema (en número de puertas) para el cálculo de la curva completa de prestaciones de la red $\vec{G}$ y el conjunto ruta crítica $path_{cp,i}$. Ambos ejes están expresados en escala logarítmica.

De la figura 4.10 se estima que el orden de tiempo de ejecución es $\Omega\ (n^2)$. Además, la curva completa de prestaciones $path_{cp,i}$ se obtiene con menos compu-

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

tación para circuitos grande que para pequeños. El ratio de tiempo de CPU se reduce entre 1'5 y 32'5. Cuanto mayor sea el tamaño del circuito, mayor es el ratio de tiempo de CPU.

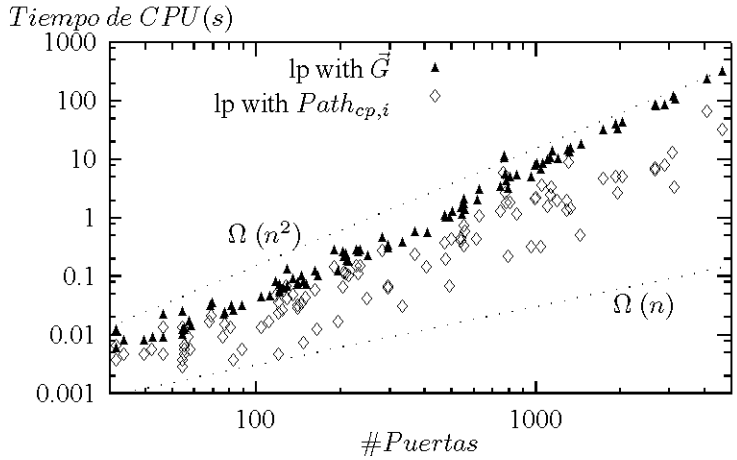


Figura 4.10: Tiempo de CPU frente al número de puertas.

4.6.8. Curvas de Prestaciones

Exploración de la Curva de Prestaciones

Dado un circuito del banco de pruebas MCNC'91, la curva entera de prestaciones del conjunto ruta critica y la curva entera de prestaciones de la red completa son idénticas. La figura 4.11 ilustra la curva completa de prestaciones para el circuito bw del banco de pruebas MCNC'91. El eje x de la figura 4.11 representa cada punto de mapeado, el eje y derecho muestra el área activa total (en micrómetros cuadrados), y el eje y izquierdo el retardo del circuito (en nanosegundos). Todos los ejes se encuentran expresados en escala lineal. Los círculos negros representan el retardo y los blancos el área activa, para cada punto de mapeado. Por ejemplo, el primer punto de mapeado de la curva de prestaciones es el punto de prestaciones para el dimensionado mínimo del circuito. El circuito de prueba bw es un circuito de dos niveles con 5 entradas, 28 salidas y un total de 211 puertas. Con este circuito se obtuvieron un total de 30 puntos de mapeado. Entre dos puntos de mapeado sucesivos, el conjunto ruta crítica no varía. Los transistores de dimensiones mínimas de una puerta lógica son de 0'48 $\mu m \times$ 0'18 μm para el *pull–up*, y 0'24 $\mu m \times$ 0'18 μm para el *pull–down*. Por ejemplo, el área activa de un inversor con factor de velocidad unitaria es de 0'1296 μm^2.

Esfuerzo Computacional

Curva de Prestaciones

La figura 4.12 muestra la relación entre el retardo y la sumatoria de factores de velocidad ($\sum \beta$) para diferentes condiciones de carga del circuito bw. Se eligió el inversor como carga con un transistor de *pull–up* de 0'48 $\mu m \times$ 0'18 μm, y un transistor de *pull–down* de 0'24 $\mu m \times$ 0'18 μm. Esta configuración provee una capacidad de cableado parásita de 15 fF. Cada salida del circuito bw fue cargada con uno, dos o tres inversores de carga.

4.7. RTL versus Circuito de Lógica Combinacional

Es evidente que varias de las métricas de cobertura, como la exploración del espacio de diseño aportadas en esta Tesis Doctoral, parten de una descripción

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

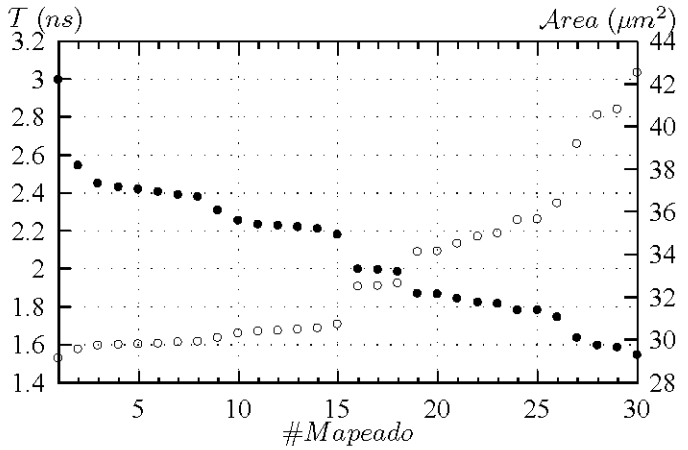


Figura 4.11: Curva de prestaciones completa de área activa frente a retardo para el circuito bw.

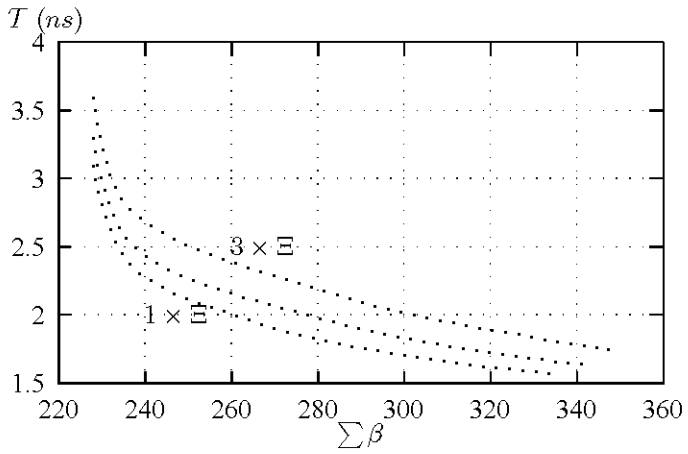


Figura 4.12: Retardo del circuito bw bajo diferentes cargas frente a la suma de factores de velocidad. El símbolo Ξ representa la carga de un inversor con 15 fF de capacidad de cableado parásita.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

estructural que no se corresponde directamente con la codificación RTL de los módulos, *clusters* y sistema a verificar.

Síntesis Lógica

Una vez la codificación RTL pasa la etapa de síntesis lógica, se posee dicha descripción estructural y, por tanto, todas las técnicas y metodologías descritas hasta ahora son válidas, mientras las estructura del circuito no varíe. De aquí que la técnica de exploración del espacio de diseño, atendiendo al criterio de la ruta crítica, puede ser aplicada fácilmente a la etapa de mapeado tecnológico.

Sea una librería $\mathcal{L}$ convexa con múltiples versiones de diversas puertas y $\breve{\mathcal{L}}$ una librería que contiene la versión de mínima área de cada una de las puertas disponibles en la librería $\mathcal{L}$. Cada una de las puertas de la librería $\breve{\mathcal{L}}$ se encuentran descritas en función de puertas NAND e INV.

Sea un circuito de lógica combinacional $\vec{G}$ — resultado del proceso de Síntesis Lógica mencionado en los párrafos anteriores — que además, se encuentra descrito como una red de puertas NAND e INV.

En estas condiciones se pueden buscar todas las puertas lógicas definidas en la librería $\breve{\mathcal{L}}$ que pueden sustituir a una o más puertas lógicas del circuito combinacional $\vec{G}$. Se dice que existe una ocurrencia de la puerta g de la librería $\breve{\mathcal{L}}$ sobre una puerta n del circuito combinacional $\vec{G}$, si la descripción NAND e INV de la puerta de librería g coincide con el cono lógico de entrada de la puerta del circuito donde aparece la ocurrencia. Si se da la ocurrencia, tanto la puerta donde se identifica la ocurrencia como el cono lógico que coincide, puede ser sustituido por la puerta de librería g.

Se define $merge(n, g)$ al cono lógico de entrada de la puerta n del circuito $\vec{G}$ en la que existe una ocurrencia de la puerta de librería g. En caso de sustituirse g por $merge(n, g)$, las entradas al cono lógico de entrada coincidente de la puerta n serán las entradas de la nueva puerta g. Esas entradas se definen como $fanin(n, g)$.

$explore_circuit()$
  $\forall n_i \in \vec{G}$, search each g_i, if $\exists\, merge_{(n_i,g_i)} \cap g_i \in \breve{\mathcal{L}}$
  obtain the critical path set $path_{cp,i}$;
  do
    update $\breve{T}_i = T(path_{cp,i})$;
    Minimise: the delay time of $path_{cp,i}$, $T(path_{cp,i})$
    Minimise: the active area, $\mathcal{A}$, of $path_{cp,i}$, s.t.
      (i) a lower bound delay time defined as:
        $max(T(path_{l,m}))$, $\forall\, path_{l,m} \in I(path_{cp,i})$
      (ii) $T(path_{cp,i}) > T(path_{u,v})$, $\forall\, path_{u,v} \in D(path_{cp,i})$
    update the critical path set $path_{cp,i}$;
    update $\breve{T}_j = T(path_{cp,i})$;
  while $\breve{T}_i \neq \breve{T}_j$
end $explore_circuit()$

Figura 4.13: Algoritmo para la generación de la curva completa de prestaciones.

En este contexto, se puede explorar el espacio de diseño del circuito a mapear

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

tecnológicamente con el algoritmo presentado en la figura 4.13. El algoritmo permite optimizar un circuito expresado en forma de árbol binario atendiendo a su ruta crítica. El algoritmo incorpora automáticamente las rutas dependientes que surgen durante el proceso de optimización y finaliza cuando dicho proceso no puede mejorar el retardo con mínima área.

Optimización Circuital

La metodología propuesta plantea el problema de la elección del mejor mapeado tecnológico para las puertas de la ruta crítica. Para ello, el optimizador determinista (Programación Lineal) o heurístico (Algoritmo Genético) resuelve qué coincidencia g ha de ser sustituida en la ruta crítica del circuito. El punto obtenido tras la sustitución es un punto de mínimo retardo y área.

Este proceso se repite mientras la ruta crítica sea susceptible de ser mejorada y no pierda su condición de ruta crítica. En este último caso, sólo bastará añadir las rutas dependientes que forman parte de la nueva definición de ruta crítica.

Debido a los diversos procesos de minimización y optimización lógica — con el objeto de reducir el área final del circuito — las diversas salidas de un circuito de lógica combinacional actual comparten en mayor o menor medida las subredes que definen su funcionalidad. Esto es, las redes que implementan las funcionalidades, de las salidas de cualquier circuito, comparten varios conos lógicos entre sí. Es decir, no se implementa un cono lógico de entrada por cada salida, en la que cada cono lógico de entrada no comparte puerta alguna con el resto de conos lógicos del circuito.

Ya que el algoritmo propuesto, al igual que todos los presentados hasta la actualidad en la literatura, supone que una salida se implementa con un cono lógico independiente del resto de conos lógicos del resto de salidas, se ha de someter el circuito a un proceso de particionado con el objeto de cumplir dicho requisito.

Exploración del Espacio de Diseño

El resultado de explorar el espacio de diseño con la librería de puertas de mínimas dimensiones es el conjunto de puntos de mínimo retardo y área activa para ese circuito particionado con esa librería tecnológica. Entre esos puntos se encuentra también el llamado punto de mínima área activa.

La figura 4.14(a) muestra el resultado de aplicar este nuevo algoritmo de exploración del espacio de diseño en la etapa de mapeado. En dicha figura se puede observar los puntos óptimos encontrados. A partir de cada uno de esos puntos se puede explorar el espacio de diseño como se ha explicado a lo largo de este capítulo y así obtener el espacio de diseño completo de circuito en estudio.

Como se puede observar en la figura 4.14(b) una vez se ha explorado el espacio de diseño de cada una de las implementaciones se observa que:

- Pueden aparecer escalones en lo que hasta ahora era una curva continua dentro del dominio del dimensionado de puerta. Ello se debe a que existe una disposición circuital que, a partir de cierto punto del espacio de diseño, es más óptima que la solución circuital anterior optimizada con el dimensionamiento de su ruta crítica.

- Se evidencia que existen soluciones circuitales con mejor área activa y retardo cuando se realiza un proceso de dimensionado. Sean un par de puntos de mapeado óptimos. Existen circuitos en los cuales, el mero dimensionado de las puertas del punto de mapeado de menores prestaciones mejora en gran medida las prestaciones del punto de mapeado de rango superior.

Política de Particionado

La metodología expuesta en este apartado es independiente del método de particionado aplicado antes de la búsqueda del espacio de diseño del circuito. Pero

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

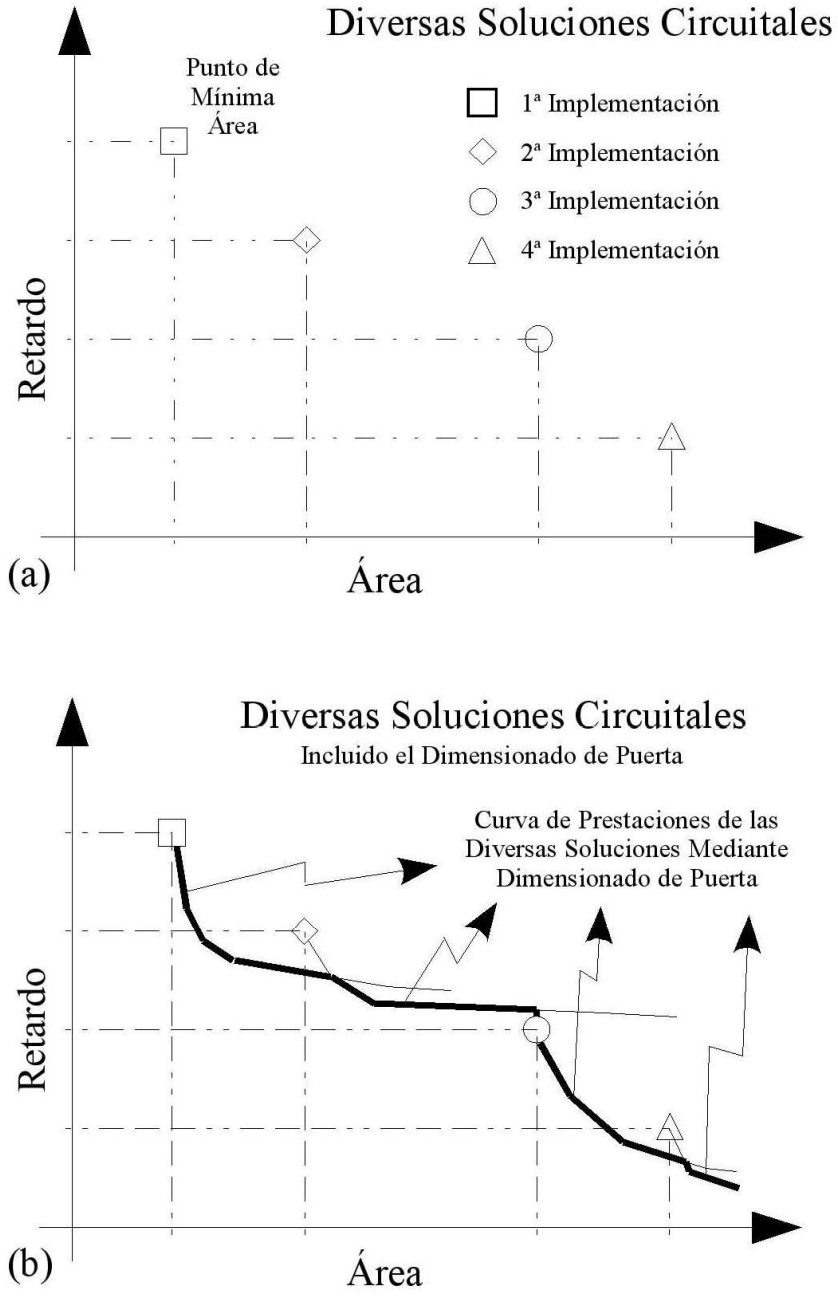


Figura 4.14: Exploración del espacio de diseño completo en mapeado tecnológico atendiendo a la ruta crítica del circuito: a) metodología tradicional de mapeado tecnológico, b) exploración del espacio de diseño atendiendo a la ruta crítica.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

es evidente que la elección de uno u otro método de particionado influye directamente en el resultado final. No obstante, la metodología expuesta permite acelerar el proceso de mapeado tecnológico frente a los métodos y algoritmos propuestos en la actualidad. En concreto, la aportación que realiza esta Tesis Doctoral en este sentido es la de ofertar un amplio abanico de posibles soluciones y no una única solución del espacio de diseño de un circuito.

Esta solución integra la exploración del espacio de diseño completo y el análisis de prestaciones. Permite, tanto al diseñador como al verificador, determinar si el módulo, *cluster* o sistema cumple las especificaciones funcionales y temporales en un rango de puntos del espacio de diseño. Esta metodología acelera la verificación temporal, adelantándola a una etapa temprana del diseño. De aquí, que el diseñador a nivel RTL, puede prescindir de los informes de *slack–time*[11], que sólo indican si la implementación del módulo hay que recodificarla y, por tanto verificarla nuevamente o no.

Solución Completa

4.8. Conclusiones

En este trabajo, se propone una metodología para calcular la curva entera de área activa frente a retardo — curva de prestaciones — de un circuito de lógica combinacional mediante la optimización de la ruta crítica que ha demostrado ser muy efectiva. Debido a que la curva de prestaciones, para un circuito de lógica combinacional, es generada empleando un subconjunto de la red Booleana que representa el circuito — conjunto ruta crítica — se consigue una reducción sustancial del tamaño y tiempo de ejecución del problema de optimización [112], [88].

Cuando se aplica una técnica de Programación Lineal y aproximaciones lineales a tramos de las fórmulas no lineales del retardo para resolver el problema de dimensionado, la metodología aquí presentada el número de variables de entre 1'1 y 7'3 veces.

Con circuitos de alta complejidad (medida en número de puertas), se alcanza una reducción considerable tanto en el número de variables, como el tiempo de ejecución. El tiempo de ejecución se mejora entre 1'5 y 32'5 veces menos tiempo de CPU en comparación con el cómputo de la red completa. Por lo tanto, se ha reducido la complejidad de la optimización mediante la eliminación de esas puertas que no contribuyen al mayor retardo del circuito.

El método provee el rango de optimización o dominio de definición de cada conjunto ruta crítica en cualquier punto de la curva de prestaciones. Una de las características principales de este método es que este puede ser adaptado a cualquier

[11] Término que identifica en la literatura a la diferencia temporal entre la velocidad de una ruta y el tiempo requerido para ésta. Cuando el *slack–time* es positivo, la ruta es más veloz que el tiempo que se le requiere. Por el contrario, si el *slack time* es negativo, el tiempo requerido es inferior al alcanzado por la ruta en estudio.

[112] J. Sosa, H. Navarro, V. de Armas, J. A. Montiel-Nelson, and R. Sarmiento. *A Novel Methodology for Efficient Computation of the Area/Power Consumption versus Delay Tradeoff Curve in Circuit Optimization Problems*, pages 190–195. Evolutionary Methods for Design Optimization and Control. International Center for Numerical Methods in Engineering, 2002.

[88] J.A. Montiel-Nelson, J. Sosa, H. Navarro, R. Sarmiento, and A. Nuñez. Efficient Method to Obtain the Entire Active Area against Circuit Delay Time Trade–Off Curve in Gate Sizing. In *IEE Proc. Circuits, Devices & Systems*, volume 152 (2), April 2005.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

técnica de Programación Lineal, No Lineal, Heurística o combinada algorítmica–heurística y cualquier modelo de retardo y área.

Tabla 4.1: RESULTADOS, COMPLEJIDAD, NÚMERO DE VARIABLES, TIEMPOS DE EJECUCIÓN Y GANANCIAS DE LA METODOLOGÍA PROPUESTA. EJEMPLOS DE LÓGICA COMBINACIONAL MULTINIVEL

Circuito MCNC'91		Red Booleana $\vec{G}$ (Red Completa)		Conjunto Ruta Crítica (Sólo $path_{cp,i}$)					Relación	
		Variables	Tiempo	Rutas	Variables Necesarias			Tiempo	Variables	Tiempo
Nombre	Puertas	Necesarias	CPU	Encontradas	Mínimo	Medio	Máximo	CPU	Necesarias	CPU
9symml	235	708	0'266	16	84	323'16	486	0'117	2'19	2'3
C1355	776	2.331	11'311	22	546	1.468'99	1.776	6'284	1'59	1'8
C17	8	27	0'002	3	18	22'44	24	0'001	1'20	2'0
C1908	784	2.355	5'578	60	372	1.223'26	1.473	2'845	1'93	2'0
C2670	1.053	3.162	6'877	29	144	485'32	843	0'343	6'52	20'0
C3540	2.058	6.177	42'502	73	369	1.196'08	2.145	5'392	5'16	7'9
C432	547	1.644	1'513	32	201	556'47	741	0'487	2'95	3'1
C499	776	2.331	10'758	23	546	1.468'96	1.776	6'329	1'59	1'7
C5315	2.918	8.757	85'519	61	237	1.684'75	2.091	8'448	5'20	10'1
C6288	4.139	12.420	235'887	24	441	3.889'43	5.844	70'367	3'19	3'4
C7552	2.697	8.094	84'129	79	174	1.531'29	2.307	7'000	5'29	12'0
C880	497	1.494	1'018	12	171	248'34	297	0'072	6'02	14'1
alu2	559	1.680	1'759	34	135	482'12	654	0'352	3'48	5'0
alu4	1.110	3.333	9'961	52	330	934'50	1.191	1'686	3'57	5'9
apex6	974	2.925	4'847	31	78	538'29	1.029	0'350	5'43	13'8
apex7	299	900	0'334	23	54	267'10	411	0'068	3'37	4'9
b1	11	36	0'002	4	15	27'60	33	0'001	1'30	2'0
b9	142	429	0'078	18	54	208'54	321	0'034	2'06	2'3
c8	149	450	0'080	5	57	90'66	105	0'008	4'96	10'0
cc	82	249	0'031	9	36	123'08	168	0'014	2'02	2'2
cht	192	579	0'281	9	99	463'65	555	0'156	1'25	1'8
cm138a	32	99	0'006	2	66	72'31	78	0'004	1'37	1'5
cm150a	47	144	0'022	2	120	132'82	135	0'014	1'08	1'6
cm151a	27	84	0'006	2	60	67'70	72	0'004	1'24	1'5
cm152a	55	168	0'025	7	54	131'92	159	0'014	1'27	1'8
cm162a	56	171	0'013	5	48	70'10	93	0'005	2'44	2'6
cm163a	47	144	0'009	7	36	73'16	105	0'005	1'97	1'8
cm42a	34	105	0'008	9	36	71'09	93	0'005	1'48	1'6
cm82a	29	90	0'005	6	39	55'73	72	0'003	1'61	1'7
cm85a	56	171	0'013	4	60	76'86	87	0'006	2'22	2'2
cmb	56	171	0'012	7	48	86'51	114	0'007	1'98	1'7
comp	212	639	0'187	30	114	400'73	585	0'118	1'59	1'6
cordic	77	234	0'022	8	51	114'27	174	0'010	2'05	2'2
count	167	504	0'102	9	69	120'72	186	0'013	4'17	7'8
cu	59	180	0'014	8	36	85'23	117	0'006	2'11	2'3

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

Tabla 4.2: RESULTADOS, COMPLEJIDAD, NÚMERO DE VARIABLES, TIEMPOS DE EJECUCIÓN Y GANANCIAS DE LA METODOLOGÍA PROPUESTA. EJEMPLOS DE LÓGICA COMBINACIONAL MULTINIVEL, CONTINUACIÓN

Circuito MCNC'91		Red Booleana $\vec{G}$ (Red Completa)		Conjunto Ruta Crítica (Sólo $path_{cp,i}$)					Relación	
		Variables	Tiempo	Rutas	Variables Necesarias			Tiempo	Variables	Tiempo
Nombre	Puertas	Necesarias	CPU	Encontradas	Mínimo	Medio	Máximo	CPU	Necesarias	CPU
dalu	1.462	4.389	17'869	41	204	604'82	840	0'550	7'26	32'5
decod	32	99	0'012	3	60	90'64	96	0'007	1'09	1'7
des	4.704	14.115	317'859	71	1.563	3.329'16	6.000	33'890	4'24	9'4
example2	373	1.122	0'571	33	114	579'26	741	0'247	1'94	2'3
frg1	142	429	0'072	9	66	187'28	264	0'031	2'29	2'3
frg2	1.144	3.435	10'336	29	393	1.312'87	1.986	2'609	2'62	4'0
i1	55	168	0'010	4	54	69'30	81	0'004	2'42	2'5
i2	217	654	0'179	5	123	503'73	591	0'112	1'30	1'6
i3	152	459	0'070	5	132	285'94	384	0'046	1'61	1'5
i5	198	597	0'122	9	57	162'15	258	0'018	3'68	6'8
i6	559	1.680	2'082	9	696	876'56	966	0'791	1'92	2'6
i7	755	2.268	3'339	7	696	1.292'76	1.380	1'385	1'75	2'4
i8	1.331	3.996	14'005	30	648	2.116'02	2.694	9'337	1'89	1'5
i9	792	2.379	4'298	12	315	1.112'94	1.461	1'985	2'14	2'2
i10	3.178	9.537	106'884	92	444	1.412'93	1.890	3'529	6'75	30'3
k2	1.313	3.942	14'305	56	108	1.007'77	1.614	2'088	3'91	6'9
lal	122	369	0'054	5	57	77'65	123	0'005	4'75	10'8
majority	13	42	0'002	3	27	31'66	39	0'001	1'33	2'0
mux	24	75	0'005	2	54	61'35	66	0'003	1'22	1'7
my_adder	207	807	0'259	11	207	252'86	321	0'070	3'19	3'7
pair	1.991	5.976	32'678	68	96	1.308'75	2.466	2'822	4'57	11'6
parity	69	210	0'031	10	135	167'89	186	0'018	1'25	1'7
pcle	84	255	0'026	5	48	64'19	84	0'004	3'97	6'5
pcler8	126	381	0'062	5	171	193'18	216	0'028	1'97	2'2
pm1	55	168	0'012	8	39	60'68	114	0'003	2'77	4'0
rot	808	2.427	3'137	25	93	466'31	588	0'240	5'20	13'1
sct	90	273	0'031	6	48	89'75	114	0'006	3'04	5'2
t481	563	1.692	1'400	33	156	675'69	930	0'612	2'50	2'3
tcon	40	123	0'008	2	96	100'80	120	0'005	1'22	1'6
term1	299	900	0'299	17	57	260'16	447	0'073	3'46	4'1
too_large	482	1.449	0'988	24	105	375'69	714	0'208	3'86	4'8
ttt2	252	759	0'221	18	51	213'96	378	0'045	3'55	4'9
unreg	119	360	0'081	3	258	262'51	264	0'050	1'37	1'6
vda	817	2.454	5'082	80	63	916'62	1.464	1'925	2'68	2'6
vg2	113	342	0'045	6	66	141'07	204	0'018	2'42	2'5
x1	412	1.239	0'560	41	57	416'51	684	0'153	2'97	3'7
x2	78	237	0'024	13	99	154'02	195	0'016	1'54	1'5
x3	1.211	3.636	10'208	40	546	1.295'99	1.809	2'064	2'81	4'9
x4	479	1.440	1'110	20	210	647'83	891	0'400	2'22	2'8
z4ml	43	132	0'009	7	54	81'98	102	0'006	1'61	1'5

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

Tabla 4.3: RESULTADOS, COMPLEJIDAD, NÚMERO DE VARIABLES, TIEMPOS DE EJECUCIÓN Y GANANCIAS DE LA METODOLOGÍA PROPUESTA. EJEMPLOS DE LÓGICA COMBINACIONAL DE DOS NIVELES

Circuito MCNC'91		Red Booleana $\vec{G}$ (Red Completa)		Conjunto Ruta Crítica (Sólo $path_{cp,i}$)					Relación	
		Variables	Tiempo	Rutas	Variables Necesarias			Tiempo	Variables	Tiempo
Nombre	Puertas	Necesarias	CPU	Encontradas	Mínimo	Medio	Máximo	CPU	Necesarias	CPU
5xp1	122	369	0'073	19	36	197'45	324	0'038	1'87	1'9
9sym	239	720	0'281	22	63	378'80	498	0'160	1'90	1'8
apex1	1.312	3.939	14'031	48	114	806'83	1.386	1'416	4'88	9'9
apex2	482	1.449	0'989	24	105	375'69	714	0'207	3'86	4'8
apex3	1.762	5.289	31'342	84	90	1.250'49	2.214	4'975	4'23	6'3
apex4	2.718	8.157	83'235	70	120	1.279'91	2.178	7'725	6'37	10'8
apex5	1.067	3.204	8'485	62	54	1.550'00	2.316	3'838	2'07	2'2
b12	105	318	0'044	14	33	122'60	213	0'014	2'59	3'1
bw	211	636	0'243	30	36	331'30	480	0'120	1'92	2'0
clip	130	393	0'131	20	123	270'17	345	0'073	1'45	1'8
con1	23	72	0'003	5	24	49'00	66	0'002	1'47	1'5
cordic	77	234	0'022	8	51	114'27	174	0'010	2'05	2'2
cps	1.343	4.032	15'276	78	69	922'09	1.752	1'562	4'37	9'8
duke2	507	1.524	1'253	39	66	566'64	933	0'464	2'69	2'7
e64	338	1.017	0'370	5	195	202'30	222	0'033	5'03	11'2
ex1010	3.125	9.378	119'036	95	87	1.700'27	2.718	14'072	5'52	8'5
ex4	549	1.650	1'126	25	51	719'74	1.260	0'420	2'29	2'7
ex5	1.151	3.456	13'607	52	135	1.175'82	1.776	3'526	2'94	3'9
inc	208	627	0'213	30	78	333'12	459	0'123	1'88	1'7
misex1	70	213	0'035	13	33	162'36	195	0'022	1'31	1'6
misex2	122	369	0'056	17	45	175'53	264	0'025	2'10	2'2
misex3	864	2.595	5'259	69	84	838'43	1.461	1'246	3'10	4'2
misex3c	622	1.869	2'003	38	60	566'35	846	0'457	3'30	4'4
o64	131	396	0'066	7	33	304'78	393	0'044	1'30	1'5
rd53	58	177	0'017	11	36	97'84	153	0'010	1'81	1'7
rd73	147	444	0'102	15	54	189'93	306	0'037	2'34	2'8
rd84	285	858	0'463	40	126	502'75	735	0'294	1'71	1'6
sao2	164	495	0'120	17	120	252'35	351	0'063	1'96	1'9
seq	1.952	5.859	39'583	81	150	1.304'03	2.412	5'383	4'49	7'4
t481	563	1.692	1'400	33	156	675'69	930	0'612	2'50	2'3
table3	1.019	3.060	8'526	89	114	955'00	1.749	2'329	3'20	3'7
table5	*1.007*	3.024	8'002	75	*123*	*1.009'26*	*1.692*	*2'266*	*3'00*	3'5
vg2	113	342	0'045	6	66	141'07	204	0'018	2'42	2'5
xor5	18	57	0'003	4	42	48'50	51	0'002	1'18	1'5
Z5xp1	635	1.908	3'002	36	123	788'32	1.110	1'160	2'42	2'6
Z9sym	231	696	0'284	21	111	379'84	513	0'162	1'83	1'8

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

Capítulo 5

Exploración del Espacio de Diseño Potencia de Consumo–Retardo

Índice General

5.1. Introducción 157

5.2. Trabajos Previos 158

5.2.1. Consumo de Potencia en Circuitos CMOS 160

Potencia de Consumo Estático 161

Potencia de Consumo Dinámico 162

Potencia de Consumo de una Puerta 164

Tiempo de Transición de una Señal a la Salida de una Puerta 165

Inversores en Cadena 165

Densidad de Transición 166

Potencia de Consumo por Interconexionado 168

5.3. Modelo de Puerta 168

5.4. Formulación del Problema de Optimización de la Ruta Crítica 169

5.4.1. Definiciones de Teoría de Grafos 170

5.5. Optimización de un Circuito Empleando el Conjunto Ruta Crítica 170

5.6. Curva de Prestaciones de un Circuito 171

5.6.1. Mapeado a Mínima $\mathcal{P}$otencia de Consumo 172

5.6.2. Curva de Prestaciones 174

5.6.3. Algoritmo de Exploración 174

5.7. El Problema del Punto de Arranque 177

5.7.1. Solución Basada en Algoritmos Genéticos 178

Función de Coste 178

Codificación del Problema 179

Resultados Experimentales 181

5.7.2. Solución Mediante Programación Lineal 182

Modelo de Potencia de Consumo Lineal 183

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

Planteamiento del Problema 184
Complejidad del Problema 185
Resultados Experimentales 185
5.7.3. Comparativa 186
5.8. Formulación del Dimensionado de Puerta Empleando Programación Lineal 188
5.8.1. Modelo Lineal de Puerta 188
5.8.2. Curva de Prestaciones Empleando Programación Lineal 188
5.8.3. Complejidad de la Optimización 190
Puntualizaciones sobre la Potencia de Corto Circuito . 193
5.9. Resultados Experimentales 195
5.9.1. Comparativa 196
5.9.2. Curvas de Prestaciones 199
5.10. Conclusiones 203

Glosario de Términos

β_{gate}	Factor de velocidad de la puerta $gate$.
$\mathfrak{C}_{fanin}(gate)$	Cono lógico de entrada a la puerta $gate$.
$\mathfrak{C}_{fanout}(gate)$	Cono lógico de salida de la puerta $gate$.
C_{load}	Capacidad de cableado y de puerta soportado.
C_{tr}	Capacidad de puerta.
C_l	Capacidad de interconexionado.
CAD	Computer Aided Design.
$D(path_{cp})$	Conjunto de rutas dependientes de la ruta crítica.
$D(y_i)$	Densidad de transición de la señal y_i.
$fanout(v_i)$	Puertas conectadas como carga a la puerta v_i.
$\overline{f_{sw}}$	Frecuencia de conmutación media.
$\vec{G}$	Circuito combinacional expresado como grafo.
GA_{Cost}	Función de coste del algoritmo genético.
GCC	GNU C Compiler.
$head(path)$	Última puerta desde las entradas de la ruta $path$.
$I(path_{cp})$	Conjunto de rutas independientes de la ruta crítica.
ITRS	International Technology Roadmap for Semiconductors.
M_A	Circuito mapeado al punto de mínima área.
$n_x(T)$	Número de transiciones de una señal.
$\mathcal{P}$	Potencia total de un circuito.
$P_{gate,d}$	Potencia dinámica consumida por la puerta $gate$.
$P_{\vec{G}}$	Potencia total del circuito representado por el grafo $\vec{G}$.
$P_{gate,s}$	Potencia estática consumida por la puerta $gate$.
$P_{gate,sc}$	Potencia de cortocircuito consumida por la puerta $gate$.
$path$	Ruta.
$path_{cp}$	Ruta crítica o camino crítico.
$path_{i,j}$	Ruta que comienza en el nodo i y finaliza en el j.
t	Tiempo medio de subida y bajada.
T_{gate}	Tiempo máximo en la salida de la puerta $gate$.
τ_{gate}	Retardo de la puerta $gate$.
$\tau_{rf,gate}$	Tiempo medio de subida y bajada a la salida de la puerta $gate$.
$\tau_{rf,gate_{in}}$	Tiempo medio de subida y bajada en las entradas de la puerta $gate$.
T_{max}	Tiempo máximo del circuito.
$tail(path)$	Primera puerta desde las entradas de la ruta $path$.
PI	Entradas primarias del circuito.
PO	Salidas primarias del circuito.
VLSI	Very Large Scale Integration.

En las aplicaciones actuales — multimedia y/o móviles, entre otras — se requiere un ajuste óptimo de la potencia de consumo. Ello es debido a que este

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

parámetro es tan crítico como el propio retardo o área. Es usual que parte la verificación contenga referencias específicas de consumo máximo o medio, lo que implica generar patrones de verificación para cubrir esos casos. El estudio de las soluciones circuitales, en términos de potencia de consumo, para anotar las diversas funcionalidades con sus potencias de consumo y retardos es entonces una tarea primordial. La idea de no anotar una solución circuital única, sino de explorar el espacio de diseño potencia de consumo–retardo es aplicable aquí también.

En el capítulo anterior se expuso en detalle la metodología propuesta para exploración del espacio de diseño retardo–área. De acuerdo con la premisa: "un circuito óptimo en área para un retardo de propagación no es necesariamente óptimo para la prestación potencia de consumo" (véase la referencia [113]) en este capítulo se aborda la exploración del espacio de diseño utilizando el *trade–off* potencia de consumo–retardo de propagación, de forma abreviada potencia–retardo.

En este capítulo se propone una metodología que obtiene los puntos óptimos del espacio de diseño potencia de consumo–retardo, eficientemente, desde el punto de vista del esfuerzo computacional — memoria y velocidad.

Validez de la Metodología

Se demuestra que la metodología basada en la exploración del espacio de diseño de la ruta crítica sigue siendo válida para obtener la curva de prestaciones potencia–retardo. Es decir, no es necesario incorporar la red Booleana que representa el circuito, sino una subred — un subconjunto de puertas. Esta reducción de la complejidad del problema, una vez más, conlleva a una reducción drástica en el tiempo de computación.

Este capítulo está organizado como sigue. En primer lugar se realiza una breve introducción al problema de exploración del espacio de diseño atendiendo a criterios de potencia de consumo y retardo. En el apartado 5.2 se exponen los trabajos previos realizados en este campo y las diversas aportaciones en los modelos de potencia de consumo. En el apartado 5.3 se presenta el modelo de puerta escogido para realizar la exploración de la curva completa de potencia–retardo. En el apartado 5.4 se introduce la formulación matemática de optimización potencia–retardo. La optimización del conjunto ruta crítica y la obtención de la curva de prestaciones se aborda en los apartados 5.5 y 5.6, respectivamente. En el apartado 5.7 se discute en profundidad cómo determinar el punto de arranque de la optimización, tanto con técnicas de Programación Lineal como con técnicas heurísticas basadas en Algoritmos Genéticos. En el apartado 5.8 se presenta la formulación del dimensionado y exploración de la curva de prestaciones empleando técnicas de Programación Lineal. Por último, se presentan los resultados experimentales y las conclusiones.

5.1. Introducción

Predicciones Tecnológicas

A medida que avanza la escala de integración, se empaquetan en circuitos integrados de menor tamaño más transistores, cada vez más rápidos y pequeños. El constante aumento de la velocidad de reloj y la capacidad de procesamiento — mídase en ancho de palabra — por circuito integrado, ha incrementado la potencia disipada hasta unos límites sin precedentes. Como predice el ITRS [114], por

[113] S. Sapatnekar and Weitong Chuang. Power vs. Delay in Gate Sizing: Conflicting Objectives? *IEEE/ACM International Conference on Computer-Aided Design*, pages 463–466, November 1995.

[114] ITRS. International Technology Roadmap for Semiconductors. http://public.itrs.net, 2001.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

el 2010 se podrán integrar más de mil millones de transistores en un único circuito monolítico. En el año 2013 el número de transistores dentro de un circuito integrado será aproximadamente igual a la población total humana, y en el 2015 el número de transistores integrados en un único circuito será superior al número de neuronas que posee un humano en su cerebro.

Potencia de Consumo, Punto Clave

El incremento de la potencia disipada por unidad de área es crítico, limitando las prestaciones de los circuitos actuales, y en particular, los numerosos diseños para aplicaciones de alta velocidad y/o móviles. La exploración de la curva de prestaciones potencia–retardo permite determinar el punto de trabajo de su circuito — si existe — de forma que dicho punto cumpla con el conjunto de especificaciones originales del diseño.

Dada una red Booleana y un retardo máximo especificado que ha de satisfacerse, se entiende la minimización en potencia consumida como el proceso de optimización que permite obtener las dimensiones de cada una de las puertas, de forma que la potencia total consumida tras este proceso, sea la mínima requerida para satisfacer la restricción de tiempo especificada.

Es evidente que la optimización en potencia no deja de ser un problema de dimensionado, donde la función objetivo representa la potencia consumida en el circuito lógico combinacional a optimizar. El fin último de la optimización en potencia es obtener la mínima potencia necesaria para un retardo dado.

En este capítulo se presenta una metodología novedosa que permite explorar la curva completa de prestaciones potencia–retardo de un circuito empleando un subconjunto de puertas del circuito, y no la red Booleana completa que representa el circuito. La técnica aplicada se basa en el empleo del conjunto ruta crítica para optimizar el retardo, tomando en consideración de aquellas puertas que ven afectada su potencia de consumo por la restricción temporal especificada.

5.2. Trabajos Previos

Exploración del Espacio de Diseño

El problema de exploración de la curva completa de prestaciones potencia–retardo no deja de ser un problema de dimensionado de transistores; al igual que ya se demostró en la exploración de la curva completa de prestaciones área–retardo. Ahora el objetivo es obtener aquellos puntos óptimos en potencia y retardo del espacio de diseño del circuito bajo estudio.

Es posible encontrar numerosas aportaciones en este sentido, aplicando diversas técnicas de optimización. En general, al margen de los parámetros empleados para realizar la optimización[1], las aportaciones se pueden clasificar en técnicas basadas en el análisis estadístico y en simulación.

Las aproximaciones no basadas en simulación utilizan la información determinista o estocástica del circuito y de las entradas que se aplican, con el objetivo de realizar una estimación de la potencia de consumo. Estas aportaciones realizan las estimaciones aplicando un esfuerzo computacional extremadamente bajo — medido en tiempo de CPU. Si embargo las simplificaciones adoptadas conllevan cierta pérdida de precisión de sus soluciones.

Pros y Contras

Por otro lado, las técnicas basadas en simulación emplean la ejecución de múltiples simulaciones, con los vectores críticos del sistema bajo estudio, para estimar la potencia de consumo. La precisión de estas aportaciones es uno de sus

[1]Los parámetros más recurridos son las dimensiones de los transistores W_i, la tensión de alimentación V_{dd} y la tensión umbral de los dispositivos V_{th}.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

puntos fuertes. Con este tipo de aproximación, se pueden tener en cuenta hasta efectos de segundo orden como, por ejemplo, las transiciones espúreas o la reconvergencia de señales. En contra, se demuestra que la estimación de potencia es extremadamente dependiente del conjunto de vectores introducidos en el circuito. De hecho, las simulaciones con distintos conjuntos de vectores presentan grandes diferencias en su estimación de potencia. La única forma de evitar este efecto indeseado es incrementar, sustancialmente, el tiempo de simulación, con el objeto de introducir el mayor número posible de vectores.

En [115] se presenta un modelo del tiempo de transición de las puertas CMOS de submicra profunda. La inclusión de este parámetro dentro de la optimización de puerta permite modelar la potencia de cortocircuito en función de, no sólo la capacidad de *driver* propia, sino en función de las pendientes de las señales a la entrada.

La actividad del circuito, desde un planteamiento estadístico, se presenta en [116], [82] y [117]. Permite estimar la frecuencia de conmutación aproximada de cada una de las puertas del circuito. Incluso en [118] se introduce una metodología para evitar computar esos espúreos que alcanzan la entrada de puerta y que no logran alcanzar su salida, debido a su alta frecuencia.

Actividad Circuital

También, se han publicado diversas herramientas para la optimización de circuitos basadas en procedimientos heurísticos o combinados algorítmicos–heurísticos. Todas las aportaciones realizadas para la optimización del área activa son válidas para la optimización de potencia de consumo, siempre que haya sido considerado en la función objetivo de dichos optimizadores.

Hay aportaciones específicas, como la expuesta en [119], donde se presenta una herramienta capaz de estudiar la correlación de las señales y la conmutación simultánea de las mismas basándose en numerosas simulaciones de diversos patrones de entrada. Por otro lado, en [120] no sólo se pone énfasis en la influencia del modelo de potencia de consumo, sino que se tiene en cuenta el modelo de interconexionado.

Modelo de Potencia

De toda la literatura se desprende que existen múltiples formas de plantear el problema de optimización en potencia de consumo y, en general, las técnicas

[115] P. Maurine, M. Rezzoug and N. Azemard, and D. Auvergne. Transition Time Modeling in Deep Submicron CMOS. *IEEE Trans. on Computer–Aided Design of Integrated Circuits and Systems*, 21(11):1352–1363, November 2002.

[116] Kaushik Roy and Sharat. C. Prasat. Circuit Activity Based Logic Synthesis for Low Power Reliable Operations. *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, 1(4):503–513, December 1993.

[82] Z. Chen, K. Roy, and T. Chuo. Efficient Statistical Approach to Estimate Power Considering Uncertain Properties of Primary Inputs. *IEEE Tran. on Very Large Scale Integration (VLSI) Systems*, 6(3):484–492, February 1993.

[117] P. Maurine, M. Rezzoug, and D. Auvergne. Output Transition Time Modeling of CMOS Structures. *IEEE International Symposium on Circuits and Systems*, 5:363–366, May 2001.

[118] F. Najm. Low–Pass Filter for Computing the Transition Density in Digital Circuits. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 13(9):1123–1131, September 1994.

[119] Tan-Li Chou and Kaushik Roy. Estimation of Activity for Static and Domino CMOS Circuits Considering Signal Correlations and Simultaneous Switching. *IEEE Trans. on Computer–Aided Design of Integrated Circuits and Systems*, 15(10):1257–1265, October 1996.

[120] Kuy-Won Choi and Abhijit Chatterjee. UDSM (Ultra–Deep Sub–Micron)–Aware Post–Layout Power Optimization for Ultra Low–Power CMOS VLSI. *International Symposium on Low Power Electronics and design*, pages 72–77, 2003.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

basadas en simulación requieren un tiempo de computación dependiente a la complejidad del circuito. Por otro lado, la precisión del resultado de las técnicas no basadas en simulación se fundamentan en la elección del modelo de potencia de consumo para la puerta lógica y su interconexión. A continuación se va a estudiar someramente las diversas aportaciones que existen en este campo, hasta la actualidad.

5.2.1. Consumo de Potencia en Circuitos CMOS

La tecnología MOS de lógica complementaria (CMOS) es la tecnología más empleada en la fabricación de circuitos integrados digitales. Una de las razones principales de la elección de esta tecnología es que permite reducir en gran medida el consumo de potencia de los circuitos. En un circuito CMOS la mayor parte del consumo[2] tiene lugar durante las transiciones y cuando no hay conmutación el consumo es prácticamente nulo. Desgraciadamente, esto ya no es cierto para las tecnologías submicras, las corrientes de fuga se incrementan, sobre todo cuando el circuito pasa mucho tiempo sin conmutar [121].

Potencia Total de Puerta

En general, el consumo de los circuitos CMOS se divide en dos partes distintas: el consumo estático y el consumo dinámico [74].

$$P = P_{\text{estática}} + P_{\text{dinámica}} \tag{5.1}$$

La disipación estática se debe fundamentalmente a la corriente de fuga de los transistores en el estado no activo[3], aunque los diodos polarizados en inversa de las regiones de pozo y difusión también contribuyen a esta corriente. La corriente de *off–state* — referida como corriente de fuga subumbral — es cada vez más importante con el escalado tecnológico, dado que la tensión umbral del dispositivo V_{th} se reduce para mantener las prestaciones del circuito [122].

Corriente de fugas

En el International Conference on VLSI on CAD de 1999, el Profesor Sakurai predijo tres crisis importantes en el área del diseño VLSI [123]. Una de estas crisis es debida al consumo de potencia.

[2]En la literatura se distingue claramente la potencia de consumo de la potencia disipada. En particular, en tecnología CMOS no coinciden. Ello es debido al trasvaso de carga entre las diversas puertas, originado por su naturaleza capacitiva y los cambios de estado. Sin embargo, cuando se analiza una puerta CMOS aislada ambas potencias sí coinciden. En un circuito cualquiera, la potencia consumida se corresponde con el producto de la tensión de alimentación por la corriente que se suministra al circuito. Por el contrario la potencia disipada no es obvia de obtener.

[3]Término identificado en la literatura como *off–state*.

[121] K. Roy, S. Mukhopadhyay, and H. Mahmoodi-Meimand. Leakage Current Mechanisms and Leakage Reduction Techniques in Deep-submicrometer CMOS Circuits. *Proceedings of the IEEE*, 91(2):305–327, feb 2003.

[74] Christian Piguet. *Low–Power Electronics Design*. CRC Press, July 2004.

[122] K.S. Khouri and N.K. Jha. Leakage Power Analysis and Reduction During Behavioral Synthesis. *IEEE Transactions on Very Large Scale Integration (VLSI)*, 10(6):876–885, December 2002.

[123] T. Sakurai. LSI Design Toward 2010 and Low–Power Technology. *Proceedings of the 1999 International Conference on VLSI and CAD*, pages 325–334, 1999.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

Potencia de Consumo Estático

El consumo de potencia por *leakage current* — corriente de fuga — puede llegar a ser el 20 % del consumo total de potencia del diseño; de ahí la importancia en disponer de un modelo preciso. Debido a que el consumo en potencia por la corriente de fuga crece exponencialmente con una V_t reducida, la corriente de fuga se convertirá cada vez más importante, contribuyendo cada vez más en el consumo total de potencia. Este efecto es inevitable a medida que la V_t se va escalando con cada nueva generación tecnológica debido a un número de razones, pero principalmente, por requerimientos de retardo [124].

Debido a que la potencia de conmutación, tradicionalmente, es la que contribuye en mayor medida al total del consumo de potencia de un circuito, ha requerido mayor atención en la investigación. En la actualidad hay varios modelos precisos.

La corriente de fuga de un dispositivo NMOS se determina mediante la expresión:

$$I_{ds} = \mu_{eff} C_{ox} \frac{W}{L}(m-1)\left(\frac{KT}{q}\right)^2 \underbrace{e^{\frac{q(V_{gs}-V_t)}{mKT}}}_{I} \underbrace{(1-e^{-\frac{qV_{ds}}{KT}})}_{II} \qquad (5.2)$$

donde m es una relación de capacidades en el dispositivo MOS. En esta expresión, se llama V_{th} al factor kT/q.

Durante la operación del dispositivo sólo los factores I y II, en la ecuación 5.2, cambian. Para el factor I, los cambios en V_{gs} causan un cambio exponencial en la corriente subumbral. Para el factor II, los cambios en V_{ds} causan un cambio en la corriente subumbral, que son distintas a las generadas por el factor I. Cuando V_{ds} esté por encima de V_{th} la corriente subumbral es insensible a los cambios en V_{ds}.

Normal Funcionamiento de la Puerta

Definir la corriente de fuga de una puerta no es tan simple como para un dispositivo MOS. En este caso la corriente de fuga no sólo depende de los parámetros de los dos conjuntos de transistores (NMOS y PMOS), sino que también depende de la topología de los transistores de la puerta. En la figura 5.1 se muestra una puerta CMOS estática con cuatro corrientes.

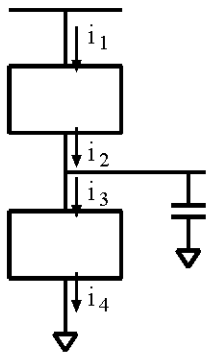


Figura 5.1: Puerta CMOS con las cuatro corrientes empleadas para definir su corriente de fuga.

A partir de esta figura es intuitivo definir la corriente de fuga de una puerta

Corrientes de Fuga

[124] D. Eckerbert and P. Larsson-Edefors. Cycle–True Leakage Current Modeling for CMOS Gates. *IEEE International Symposium on Circuits and Systems*, 5:507–510, May 2001.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

como la corriente que fluye desde su alimentación a tierra — V_{dd} a GND — cuando las tensiones de las entradas son menores que V_t o mayor que $V_{dd} - |V_t|$, es decir, cuando las corrientes i_1, i_2, i_3 e i_4 poseen su valor menor, no negativo y la entrada posee un valor menor a V_t con respecto a alguno de sus buses de alimentación.

Luego la potencia estática o de corriente de fugas se puede describir como:

$$P_{gate,l} = \{ V_{dd} I_{ds} \mid (V_{gs} > V_{dd} - |V_t|) \cap (V_{gs} < V_t) \} \tag{5.3}$$

Potencia de Consumo Dinámico

Consumo Dinámico

El consumo dinámico de un circuito CMOS se divide en dos componentes: el consumo debido a las corrientes de corto circuito y, el consumo debido a la carga o descarga de la carga capacitiva. Desde este punto de vista, la potencia consumida/disipada por una puerta CMOS $P_{gate,d}$, si se desprecia la potencia estática, viene dada por $P_{gate,C_L} + P_{gate,sc}$, donde P_{gate,C_L} es la potencia consumida durante la transferencia de carga entre niveles lógicos, y $P_{gate,sc}$ es la disipación por corto circuito durante el periodo en el que las redes *pull–up* y *pull–down* están activas. La potencia debida a la capacidad de carga viene dada por

$$P_{gate,C_L} = C_L V^2 f \tag{5.4}$$

donde C_L es la capacidad total de la puerta, V es la tensión de alimentación, y f es la frecuencia de conmutación del nodo de salida.

Consumo de Corto Circuito

Existen diversos trabajos que modelan la potencia de consumo de corto circuito. En particular, Veendrick [125] obtiene una expresión para *buffers*, a pesar de que la componente de corto circuito posee una fuerte dependencia con la capacidad en la salida. Hedenstierna *et al.* [126] calcula la corriente de corto circuito empleando modelos Shockley del MOSFET para *buffers* cargados. Este modelo no puede ser aplicado a dispositivos de canal corto[4] ya que presenta una dependencia lineal de la corriente de saturación con la tensión de puerta, frente al modelo tradicional de ley cuadrática para los dispositivos de canal largo [127]. Sakurai *et al.* [127] presenta un modelo para dispositivos de canal largo y canal corto empleando su modelo MOSFET que sigue la ley de la potencia de α[5] para *buffers* descargados. Este modelo, al igual que el modelo de [125], es adecuado para estimar el límite superior de la componente de corto circuito, pero no puede ser empleado en el cálculo preciso de potencia puesto que el efecto de la carga de salida ha de ser incluido.

[4] Traducción directa del término anglosajón *short–channel*.

[5] En la literatura referenciado como α–*power law*.

[125] H. Veendrick. Short–Circuit Dissipation of Static CMOS Circuitry and its Impact on the Design of Buffer Circuits. *IEEE J. Solid–State Circuits*, SC-19:468–473, August 1984.

[126] N. Hendestierna and K. O. Jeppson. CMOS Circuit Speed and Buffer Optimization. *IEEE Trans. Computer–Aided Design*, CAD–6:270–281, mar 1987.

[127] T. Sakurai and R. Newton. Alpha–Power Law MOSFET Model and its Applications to CMOS Inverter Delay and Other Formulas. *IEEE J. Solid–State Circuits*, 25:584–594, April 1990.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

Turgis et al. [128] presenta una expresión para la disipación de corto circuito asumiendo que los portadores están siempre en movimiento a la velocidad de saturación y considerando el efecto de *overshooting*[6]. Este modelo es válido sólo para tecnología de submicra profunda e introduce el concepto de capacidad equivalente, permitiendo realizar una comparativa directa y dependiente de la frecuencia de la potencia de varios componentes. Hamoui et al. [129] obtiene una expresión para la potencia de disipación de corto circuito, empleando una versión modificada del modelo MOSFET, siguiendo la ley de la potencia enésima[7] en [130]. Esta aportación requiere un proceso de cálculo numérico de tres fases para determinar el tiempo cuando los transistores en corto circuito cambian su modo de operación.

Overshooting

Nikolaidis et al. [131] obtienen una expresión de la disipación de corto circuito para *buffers* atacando líneas de interconexión grandes empleando un modelo π de una carga RC y, siguiendo la ley de la potencia α para los dispositivos submicra. Tienen en cuenta los efectos de capacidad de entrada–salida, mientras la contribución de la corriente de corto circuito es descuidada cuando se calcula la forma de onda a la salida. Recientemente, Nose et al.[132] derivan una expresión cerrada para la potencia disipada de corto circuito de puertas CMOS, teniendo en cuenta los efectos de canal corto, pero no los efectos de *overshooting*, de tanta importancia en los circuitos CMOS actuales. Este trabajo concluye que la relación entre la potencia de corto circuito y la potencia total $(\frac{P_{sc}}{P_{total}})$ no cambia con el escalado si se mantiene constante la relación $\frac{V_{th}}{V_{dd}}$.

Modelo de Interconexión

La potencia de corto circuito viene dada por la expresión (véase la referencia bibliográfica [125]):

$$P_{gate,sc} = \frac{\beta\, \tau_{rf,gate_{in}}}{12}(V - 2V_T)^3 f \tag{5.5}$$

Donde, t es el tiempo de la transición de entrada, β es el factor de ganancia del transistor. V es la tensión de alimentación, V_T es el voltaje umbral y f la frecuencia de conmutación[8]. La ecuación 5.5 ha sido obtenida para la máxima disipación de un inversor bajo una condición de ausencia de carga en su salida. En la referencia [125] se demuestra empíricamente que si se mantiene la formulación expresada en la ecuación 5.5, la potencia disipada por corto circuito pasa a ser la mitad, para un inversor con igual tiempo de transición en sus entradas.

Transición de Entrada

[6] Se denomina *overshooting* a las variaciones de la tensión de alimentación por encima de su valor nominal.

[7] Traducción directa del término *nth–power law*.

[8] En el caso de puertas complejas, el término apropiado es el de densidad de disparo — *trigger density* — i.e. la actividad de conmutación debida a una entrada como la entrada de disparo.

[128] S. Turgis and D. Auvergne. A Novel Macromodel for Power Estimation in CMOS Structures. *IEEE Trans. Computer–Aided Design*, 17:1090–1098, November 1998.

[129] A. Hamoui and N. Rumin. An Analitical Model for Current, Delay and Power Analysis of Submicron CMOS Logic Circuits. *IEEE Trans. Circuits Syst.*, 47:999–1007, October 2000.

[130] T. Sakurai and R. Newton. A Simple MOSFET Model for Circuit Analysis. *IEEE Trans. Electron. Devices*, 38:887–894, apr 1991.

[131] S. Nikolaidis, A. Chantzigeorgiou, and Kyriakis-Bitzaros. Delay and Power Estimation for a CMOS Inverter Driving RC Interconnect Loads. *Proc. IEEE Int. Symp. Circuits Sustems*, VI:368–371, May 1998.

[132] K. Nose and T. Sakurai. Analisys and Future Trend of Short–Circuit Power. *IEEE Trans. Computer–Aided Design*, 19:1023–1030, September 2000.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

El factor de ganancia β se determina mediante el ancho de los transistor W y la movilidad de los portadores responsables de la transición — μ_p para una transición de bajo a alto y μ_n para una transición de alto a bajo.

Por lo tanto, la ecuación 5.5 se reduce a

$$P_{gate,sc} = k\,\mu\,W\,\tau_{rf,gate_{in}} f_{gate} \tag{5.6}$$

donde k es una constante de proporcionalidad dependiente del proceso tecnológico y de la tensión. Con la equación 5.6 queda claro que la potencia de consumo de corto circuito es directamente proporcional tanto con el ancho del transistor como con el tiempo de la transición de entrada.

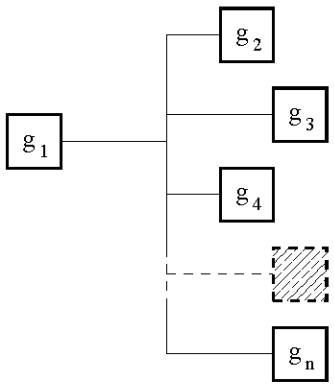


Figura 5.2: Puerta sometida a una alta carga de *fanout*.

Potencia de Consumo de una Puerta

Considérese el caso donde una puerta CMOS cargada con varias puertas (véase la figura 5.2). La puerta g_1 ataca a las puertas $g_2, g_3, \ldots, g_n$. Sea $W_1, W_2, \ldots, W_n$ los anchos de los transistores de las puertas $g_2, g_3, \ldots, g_n$, respectivamente. La señal de entrada aplicada a g_1 posee un tiempo de transición $\tau_{rf,1_{in}}$. El tiempo de transición de la señal para la entrada de las puertas $g_2, g_3, \ldots, g_n$ es $\tau_{rf,2_{in}}$, que es a su vez es el tiempo de transición de la salida de la puerta g_1, o lo que es lo mismo $\tau_{rf,1}$. La potencia total consumida por el circuito es:

Potencia de Consumo de la Puerta

$$\begin{aligned} P &= V^2 \sum_{i=1}^{n} C_{L_i} f_i + cV^2 W_1 f_1 \\ &\quad + k\left(W\mu\tau_{rf,1_{in}} f_1 + \sum_{i=2}^{n} W_i \mu' \tau_{rf,2_{in}} f_i \right) \end{aligned} \tag{5.7}$$

El primer término de la ecuación 5.7 es la potencia de consumo capacitiva del *driver* y las puertas de carga. La capacidad de carga C_L posee tres componentes: la capacidad de puerta de los transistores de carga (C_g), la capacidad de interconexionado (C_l) y la capacidad fuente/drenador del *driver* (C_d). Sólo C_d cambia con el dimensionado de los transistores del *driver*. Puesto que C_d es pequeña en comparación con C_g y C_l, la contribución de la carga en C_d al consumo capacitivo de potencia puede ser despreciado.

Desglose de Términos

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

El segundo término es el consumo de potencia capacitivo debido a la capacidad del *driver* (c es una constante). El tercer término es la potencia de consumo de corto circuito de todas las puertas. Aquí, μ es la movilidad de los portadores responsables de la transición en g_1 y μ' es la movilidad de los portadores responsables de la transición opuesta en las puertas $g_2, g_3, \ldots, g_n$.

Tiempo de Transición de una Señal a la Salida de una Puerta

Transición de la Señal

El tiempo de transición de una señal a la salida de la puerta g_1 ($\tau_{rf,1}$) se ve afectada por el dimensionado de los transistores de g_1. Dicho tiempo de transición de la señal a la salida de un inversor se presenta en [126].

$$\tau_{rf,1} = \frac{\tau_{rf,1_{in}}}{6}\left(1+\frac{2V_T}{V}\right)+\phi\frac{C_{L_1}}{\mu W_1} \tag{5.8}$$

El primer término es el retardo debido al tiempo de transición de la señal presente en la entrada. Aquí ϕ es una constante dependiente del proceso tecnológico. El segundo término es el retardo de la respuesta a una función escalón, basado en la geometría de la puerta. Sustituyendo la ecuación 5.8 en 5.7 se obtiene:

$$\begin{aligned} P &= V^2\sum_{i=1}^{n} C_{L_i}f_i + (k_1 + k\mu\tau_{rf,1_{in}})W_1 f_1 \\ &\quad + k\mu'\sum_{i=2}^{n} W_i f_i\left(\phi\frac{C_{L_1}}{\mu W_1} + k_2\tau_{rf,2_{in}}\right) \end{aligned} \tag{5.9}$$

simplificando,

$$\begin{aligned} P &= V^2\sum_{i=1}^{n} C_{L_i}f_i + (k_1 + k\mu\tau_{rf,1_{in}})W_1 f_1 + \frac{k\phi\mu' C_{L_1}}{\mu W_1}\sum_{i=2}^{n} W_i f_i \\ &\quad + kk_2\mu'\tau_{rf,2_{in}}\sum_{i=2}^{n} W_i f_i \end{aligned} \tag{5.10}$$

aquí, k_1 y k_2 son constantes. La función expresada a la derecha de la ecuación 5.10 es una función convexa con la forma

$$\alpha \times W_1 + \frac{\sigma}{W_1} + \delta \tag{5.11}$$

Inversores en Cadena

Puertas Conectadas en Serie

Hasta ahora se ha considerado el análisis de inversores CMOS. A continuación se abordará el problema de dimensionado de series de transistores conexionados. Se asume que los transistores conectados en cadenas son dimensionados uniformemente. El ajuste de los anchos de los transistores en cadenas en serie largas es tal que el transistor cercano al nodo de alimentación o tierra posea el mayor ancho, por tanto, redundando en el aumento de velocidad y el ahorro de potencia disipada [133]. Este ajuste es satisfactorio sólo cuando la capacidad total interna

[126] N. Hendestierna and K. O. Jeppson. CMOS Circuit Speed and Buffer Optimization. *IEEE Trans. Computer–Aided Design*, CAD–6:270–281, mar 1987.

[133] M. Shoji. FET Scaling in Domino CMOS Gates. *IEEE J. Solid–State Circuits*, SC–20:1067–1071, October 1985.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

surtidor/drenador de la puerta es mayor o comparable a la capacidad de carga, lo cual no es verdadero en el caso de la alta carga de salida de puerta.

En varios trabajos se han propuesto una extensión del modelo de retardo para transistores conectados en serie [134], [135]. Este problema se identifica, comúnmente, como la búsqueda del ancho equivalente de un inversor que tuviese el mismo retardo que la cadena de transistores. Sakurai y Newton en [135] indican que debido al efecto de la velocidad de los portadores en saturación en los dispositivos de canal corto, el retardo de una cadena de N inversores de ancho W es menor a N veces el retardo de un inversor simple de ancho W, es decir

$$\frac{\text{Retardo}_{\text{N inversores en serie}}}{\text{Retardo}_{\text{transistor equivalente}}} = 1 + \zeta(N-1) \tag{5.12}$$

donde ζ es una constante, menor que la unidad para dispositivos de canal corto e igual a uno para transistores de canal ancho.

Densidad de Transición

En general, ya que las señales lógicas pueden no ser periódicas, el término frecuencia no puede ser empleado. En su lugar, se ha de calcular la potencia tal como sigue. Si $x(t)$ es una señal lógica a la salida de una puerta y $n_x(t)$ es el número de transiciones de $x(t)$ en el intervalo de tiempo (-$T/2$,+$T/2$], entonces la potencia media es:

Actividad Circuital

$$\begin{aligned} P_{med} &= \lim_{T\to\infty} V_{dd}\frac{CV_{dd}n_x(T)/2}{T} \\ &= \frac{1}{2}CV_{dd}^2\left\{\lim_{T\to\infty}\frac{n_x(T)}{T}\right\} \end{aligned} \tag{5.13}$$

Naturalmente, se puede aproximar $\lim_{T\to\infty}\frac{n_x(T)}{T}$ mediante la simulación del circuito para una cantidad grande de transiciones de entrada mientras se contabiliza la función $n_x(T)$ en cada nodo. La ambigüedad de la expresión cantidad grande es precisamente el problema que surge al aplicar esta solución tradicional. Es imposible determinar a priori cuan larga ha de ser la simulación. Además, las simulaciones largas de circuitos grandes poseen un costo computacional muy elevado.

Sin embargo, el algoritmo de cálculo de la densidad de transición propuesto en [81] es muy rápido y simple. Pero adolece de no tener en cuenta el efecto inercial de los retardos de las puertas lógicas [118]. Por ello, la densidad de transición se sobrestima para los circuitos de alta velocidad, tal como se demuestra a continuación.

[134] A. Nabavi-Lishi and N. C. Rumin. Inverter Models of CMOS Gates for Supply Current and Delay Evaluation. *IEEE Trans. Computer–Aided Design*, 13:1271–1279, October 1994.

[135] T. Sakurai and A. R. Newton. Delay Analysis of Series–Connected MOSFET Circuits. *IEEE J. Solid–State Circuits*, 26:122–131, feb 1991.

[81] F. Najm. Transition Density: A New Measure of Activity in Digital Circuits. *IEEE Trans. Computer–Aided Design*, 12(2):310–323, February 1993.

[118] F. Najm. Low–Pass Filter for Computing the Transition Density in Digital Circuits. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 13(9):1123–1131, September 1994.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

Considérese un módulo lógico multientrada y multisalida M cuyas salidas son funciones Booleanas de sus entradas. M podría ser tan simple como una puerta lógica o un módulo de alto nivel. En [118] se emplea un modelo simplificado de temporización para representar los retardos de propagación a través de M, consistente en un valor simple de retardo para par de nodos entrada–salida. El principal resultado de [81] es una expresión sencilla para la densidad de transición en las salidas de M, en términos de la densidad de transición de sus entradas y sus probabilidades de diferencia Booleana, tal y como sigue

Circuitos Reales

$$D(y_j) = \sum_{i=1}^{n} P\left(\frac{\partial y_j}{\partial x_j}\right) D(x_i) \tag{5.14}$$

donde se supone que M posee n entradas independientes x_i, y m salidas y_i. El algoritmo resultante requiere solamente dos datos mínimos de cada nodo de entrada primaria. Uno es la llamada probabilidad de equilibrio $P(x)$[9] y el segundo es su densidad de transición $D(x)$. Así, con la ecuación 5.14 se dispone de un método muy eficiente para propagar esos valores a través del circuito y así calcular la $P(x)$ y la $D(x)$ en cada uno de los nodos internos.

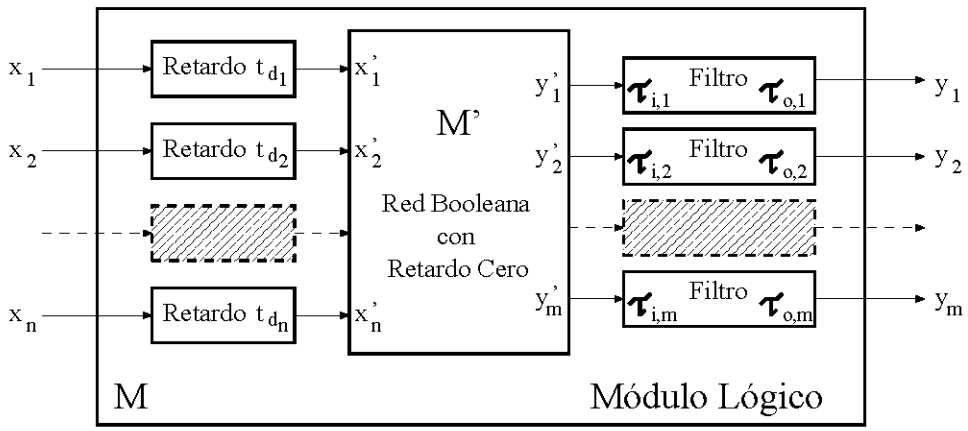


Figura 5.3: Diagrama de bloques del modelo de retardo empleado en el cálculo de la densidad de transiciones aplicando el filtrado.

El problema de esta aproximación es que se realiza sin restringir la máxima densidad de transición — o, equivalentemente, el ancho de pulso mínimo en los nodos de salidas de los módulos. Ello es resultado del modelo simplista de temporización empleado. Para ilustrarlo, considere una OR de n entradas que poseen igual probabilidad $P = 0'5$ e igual densidad de transición $D = d$. La diferencia Booleana $\frac{\partial y_i}{\partial x_i}$ es la OR de las $(n-1)$ otras entradas, su probabilidad es al menos 0'5, lo cual conduce a que $D(y) \geq (nd/2)$. Así, para un valor grande de n, la salida de la puerta presentará densidades altas y por tanto pulsos cortos poco realistas. En la práctica, estos pulsos cortos no son generados; son eliminados por filtrado, debido a que el módulo no es tan rápido como para responder a ese corto pulso. Con el objetivo de modelar ese efecto de filtrado de la inercia del retardo, se introduce un bloque de retardo llamado bloque de filtrado en cada salida del módulo, tal y como muestra la figura 5.3.

Sobreestimación

Filtrado

[9] Fracción del tiempo que este nodo esta a nivel alto.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

Potencia de Consumo por Interconexionado

Capacidades Parásitas

La potencia de consumo debida a la carga y descarga de la capacidad parásita, es caracterizada mediante:

$$P_{med} = \frac{1}{2} \sum_{n \in interconexiones} C_n \times f_n \times V^2 \quad (5.15)$$

donde, P_{med} representa la media del consumo de potencia, C_n es la capacidad de la interconexión n. V es la tensión de alimentación, y f_n es la velocidad de conmutación media — actividad de conmutación — de la interconexión n. Para estimar la potencia mediante la ecuación 5.15 es necesario obtener dos parámetros en cada interconexión: la actividad de conmutación de la interconexión y su capacidad parásita.

Se puede concebir diversos puntos de vista para la actividad de conmutación, dependiendo de como se contabilizan los retardos del circuito. En primer lugar, los valores de actividad pueden calcularse asumiendo que los retardos de la lógica y el interconexionado son cero — actividad para retardo cero. En segundo lugar, los valores de actividad pueden ser calculados considerando los retardos de la lógica, pero no los retardos de interconexión — actividad para retardos de lógica. Y por último, los valores de actividad pueden ser calculados considerando tanto los retardos de lógica e interconexión. De la literatura existente, se deduce que las diversas aportaciones referentes al cálculo de la actividad de conmutación, pueden ser clasificadas generalmente como métodos basados en simulación o aportaciones probabilísticas [136]–[137].

Cuando se tienen en cuenta los retardos, la actividad de conmutación normalmente se incrementa debido a la introducción de transiciones indeseadas[10], que son transiciones lógicas indeseadas en las interconexiones causadas por rutas con retardos distintos que atacan a una puerta. Como las transiciones en las entradas de una puerta ocurren en diferentes tiempos, la red experimenta múltiples transiciones antes de alcanzar su valor final. La actividad extra debida a las transiciones indeseadas consumen potencia dinámica, sugiriéndose en [138] que este consumo alcanza en muchos casos hasta el 20 % del consumo total, e incluso hay demostraciones especificas basadas en sumadores donde esta potencia se eleva hasta el 70 % de la potencia disipada.

5.3. Modelo de Puerta

Retardo de Propagación

La metodología propuesta en este trabajo para el análisis de prestaciones es independiente del modelo de puerta empleado. La elección de uno u otro no afecta al mismo. El modelo básico de retardo empleado en la exploración del espacio

[10] En la literatura anglosajona este término se denomina *glitches*.

[136] F. Najm. A Survey of Power Estimation Techniques in VLSI Circuits. *IEEE Trans. VLSI Syst.*, 2:446–455, dec 1994.

[137] H. Soeleman, K. Roy, and T.-L. Chou. Estimating Circuit Activity in Combiational CMOS Digital Circuits. *IEEE Des. Test Comput.*, pages 112–119, April 2000.

[138] A. Shen, A. Ghosh, and S. Devadas. On Average Power Dissipation and Random Pattern Testability of CMOS Combinational Logic Networks. *Proc. IEEE Int. Conf. Computer-Aided Design*, pages 402–407, 1992.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

de diseño retardo–área, continua siendo válido (véase el apartado 4.6.4 del capítulo 4). Éste es:

$$\tau_{gate} = \tau_{gate,int} + k \times \frac{C_{wire} + \sum_{i \in fanout(gate)} \beta_i \times C_{i,int}}{\beta_{gate}} \quad (5.16)$$

donde, τ es el retardo de puerta, C representa la capacidad. k es una constante y β es el factor de velocidad.

Potencia de Consumo

El modelo de potencia de consumo de puerta empleado se basa en las ecuaciones 5.7 y 5.15, que combinándolas, se obtiene:

$$\begin{aligned} \mathcal{P}_i = \; & p_{1,i}\beta_i + \\ & \left(p_{2,i}\, C_{l,i} + p_{3,i} \sum_{j=fanout(i)} C_{int,j}\beta_j\right) f_i + \\ & p_{4,i}\beta_i \tau_{rf,i_{in}} f_i \end{aligned} \quad (5.17)$$

El primer término se corresponde con la potencia de consumo estático (véase las ecuaciones 5.2 y 5.3). El segundo término modela el consumo de potencia debido a las capacidades de las puertas dispuestas como carga en su salida, la propia capacidad de la puerta como *driver* y la capacidad de interconexión. El último término de esta ecuación modela la potencia de corto circuito.

Tiempo de Transición

Nótese que este último término depende del tiempo de transición de las señales presentes en las entradas de la puerta. Atendiendo a lo expuesto en la introducción y en particular a la ecuación 5.8, dicho tiempo se puede expresar como:

$$\tau_{rf,gate} = a_{trf} \times \tau_{rf,gate_{in}} + k \times \frac{C_l + \sum_{j \in fanout(gate)} \beta_j \times C_{in,j}}{\beta_{gate}} \quad (5.18)$$

Al igual que en la ecuación 5.8, el primer término representa el retardo debido al tiempo de transición de la señal presente en la entrada, y el segundo término modela el retardo de la respuesta a una entrada del tipo escalón.

Como se observa en la ecuación 5.18, el tiempo medio de subida y bajada de una señal a la salida de una puerta t_{gate}, es directamente proporcional al tiempo de subida y bajada de las señales presentes a la entrada $t_{gate,in}$, a la capacidad de interconexionado C_l y al factor de velocidad de la carga β_{load}. Además, es inversamente proporcional al factor de velocidad de la puerta en estudio β_{gate}.

5.4. Formulación del Problema de Optimización de la Ruta Crítica

A pesar de parecer redundante, se hace imprescindible recordar ciertos términos ya presentados en el capítulo anterior, además de introducir nuevos conceptos para abordar y comprender adecuadamente la metodología de exploración de la curva de prestaciones potencia de consumo–retardo con ciertas garantías de éxito. A continuación se introducirán conceptos básicos de teoría de grafos necesarios en este capítulo.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

5.4.1. Definiciones de Teoría de Grafos

Definición 13 (Cono Lógico de Salida de una Puerta). *Se define el cono lógico de salida de una puerta $\mathfrak{C}_{fanout}(v_o)$, al conjunto de rutas que tienen como cola al nodo v_o y como cabeza alguna salida primaria. Formalmente:*

$$\mathfrak{C}_{fanout}(v_o) = \{path_{v_o,v_k} \mid h(path_{v_o,v_k}) \in PO(\vec{G})\}$$

Definición 14 (Cono Lógico de Entrada de una Puerta). *De igual forma, se define el cono lógico de entrada a una puerta $\mathfrak{C}_{fanin}(v_k)$, al conjunto de rutas que tienen como cabeza una salida primaria del circuito y como cola al nodo que representa la puerta v_k. Formalmente:*

$$\mathfrak{C}_{fanin}(v_k) = \{path_{v_o,v_k} \mid t(path_{v_o,v_k}) \in PI(\vec{G})\} \quad (5.19)$$

Nótese que el cono lógico de entrada del conjunto ruta crítica es igual al cono lógico de los nodos de salida del conjunto ruta crítica, es decir:

$$\mathfrak{C}_{fanin}(path_{cp,i}) = \{\mathfrak{C}_{fanin}(v_k) \mid v_k \in V_o \text{ and } h(path_{cp,i}) = v_k\} \quad (5.20)$$

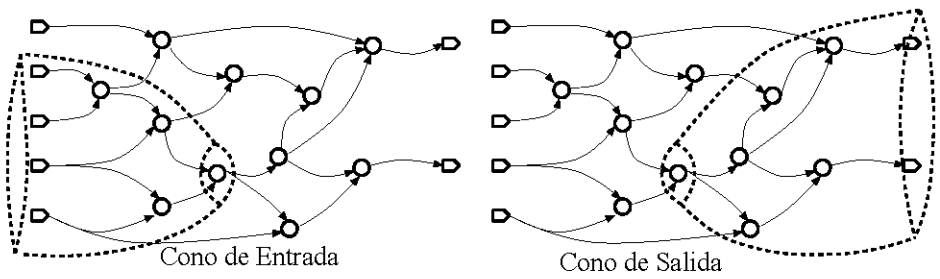


Figura 5.4: Ejemplo de cono lógico de entrada y cono lógico de salida de una puerta lógica.

5.5. Optimización de un Circuito Empleando el Conjunto Ruta Crítica

Asumiendo que el retardo máximo $\overline{T}$ de un conjunto ruta crítica $path_{cp}$ es reducido mediante algún proceso de optimización. Este conjunto pierde la propiedad de definir el retardo máximo del circuito si:

- el retardo T_n de una ruta dependiente $path_{l,n}$ del conjunto ruta crítica $path_{cp}$ es mayor o igual al retardo de $path_{cp}$,
- es alcanzado el máximo retardo T_r de las rutas independientes del conjunto ruta crítica $path_{cp}$,
- o se dan las ambas condiciones previas simultáneamente.

Dominio de Definición

En consecuencia, hay un intervalo de optimización $(\underline{T}, \overline{T}]$ para el retardo de un conjunto ruta crítica $path_{cp}$. $\underline{T}$ y $\overline{T}$ son los límites inferior y superior de T, respectivamente. El intervalo $(\underline{T}, \overline{T}]$ es el dominio de definición temporal del conjunto ruta crítica $path_{cp}$.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

Más allá de este intervalo, un nuevo conjunto de rutas críticas determinan el retardo máximo del circuito $\vec{G}$. En este sentido, durante el proceso de optimización para reducir el retardo de un circuito, el conjunto ruta crítica es dinámico. Por ello, es necesario introducir un índice i en las formulas con el objeto de diferenciar los distintos conjuntos: T, $path_{cp}$, $D(path_{cp})$ y $I(path_{cp})$ ahora se nombran T_i, $path_{cp,i}$, $D(path_{cp,i})$ y $I(path_{cp,i})$, respectivamente. Para cada nuevo conjunto $path_{cp,j}$, un nuevo T_j será encontrado.

Definición 15 (Límite Inferior del Retardo del Conjunto Ruta Crítica). *El límite inferior del retardo de un conjunto ruta crítica se expresa en el dominio de definición como el mínimo retardo, tal que, desde este retardo en adelante, el conjunto ruta crítica $path_{cp,i}$ pierde la condición de ser la ruta crítica del circuito $\vec{G}$.*

En esta aportación se obtiene el límite inferior $\underline{T_i}$ como una solución de la siguiente optimización de punto simple:

$$\begin{aligned} &Minimise && \text{the delay time of } path_{cp,i},\ T(path_{cp,i}) \\ &Minimise && \text{the power consumption, } \mathcal{P}, \text{ of } \mathfrak{C}(path_{cp,i}) \\ &s.t. && \text{(i) a lower bound delay time defined as:} \\ & && \quad max(T(path_{l,m})),\ \forall\, path_{l,m} \in I(path_{cp,i}) \\ & && \text{(ii) } T(path_{cp,i}) > T(path_{u,v}),\ \forall\, path_{u,v} \in D(path_{cp,i}) \end{aligned} \quad (5.21)$$

Límite de la Optimización

La optimización del retardo del conjunto ruta crítica $path_{cp,i}$ posee un límite inferior que es el máximo retardo de las rutas independientes del conjunto ruta crítica. El retardo de la ruta crítica $T(path_{cp,i})$ y la potencia de consumo del cono lógico de entrada del conjunto ruta crítica $\mathfrak{C}(path_{cp,i})$ es minimizado. El retardo es restringido para no ser menor que el retardo de las rutas dependientes del conjunto ruta crítica.

Tener en cuenta que a pesar de ser prácticamente el mismo algoritmo que se emplea en la búsqueda del límite inferior de tiempo para el conjunto ruta crítica en la optimización de retardo–área, ahora con la optimización de retardo–potencia de consumo, se ha de incluir el cono lógico de entrada a la ruta crítica $\mathfrak{C}(path_{cp,i})$. La modificación de la pendiente de las señales de entrada $\tau_{gate,in}$ es un factor que en ocasiones posibilita mejorar la potencia de consumo, tal y como se avanzó en el apartado 5.1.

5.6. Curva de Prestaciones de un Circuito

Exploración del Espacio de Diseño

En el apartado previo, se ha introducido el dominio de definición temporal de un conjunto ruta crítica. Dado un $path_{cp,i}$, se ha minimizado el retardo T_i aplicando la ecuación 5.21, y ello ha derivado en el retardo inferior de la ruta. El par $(\underline{T_i}, \underline{\mathcal{P}_i})$ es un punto de la curva de prestaciones — potencia de consumo $\mathcal{P}_i$ frente a retardo T_i. El retardo menor de la ruta $\underline{T_i}$ no está incluido en el dominio de definición temporal de $path_{cp,i}$.

Dado un tamaño de *driver* para todas las puertas del circuito, en los puntos no inferiores del espacio de diseño, el retardo y la potencia de consumo son mínimos. En particular, se representa la componente de retardo T de un punto no inferior como $\bar{T}$. Para el menor retardo $\underline{T_i}$ de la ruta crítica $path_{cp,i}$ de un circuito $\vec{G}$ se define otro nuevo conjunto ruta crítica $path_{cp,j}$. El dominio de definición de $path_{cp,j}$

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

es ahora $(\underline{T_j}, \check{T}_i]$. El límite superior es $\check{T}_i$, y el límite inferior es la solución de aplicar nuevamente la ecuación 5.21 a $path_{cp,j}$. Además, el retardo menor $\underline{T_j}$ no es incluido en el dominio de definición de $path_{cp,j}$.

Se obtiene la curva de prestaciones completa de un circuito con las sucesivas soluciones de aplicar la ecuación 5.21 a la secuencia apropiada de rutas críticas. La búsqueda de esta secuencia de conjuntos de rutas críticas ($path_{cp,0}$, $path_{cp,1}$, ..., $path_{cp,i}$, $path_{cp,j}$, ...) requiere un proceso de actualización del conjunto ruta crítica, para que la curva de prestaciones sea calculada, y además es necesario un conjunto ruta crítica $path_{cp,0}$ solución inicial.

En este apartado se presenta el mapeado de mínima potencia de consumo y retardo, para de esta forma obtener el conjunto ruta crítica de comienzo $path_{cp,0}$. Además, se presenta un método — y su implementación algorítmica — para actualizar el conjunto ruta crítica.

5.6.1. Mapeado a Mínima $\mathcal{P}$otencia de Consumo

Librería de Puertas

Sea $LL = \{1, \ldots, N\}$ un conjunto de puertas lógicas — NAND, NOR, INV, XOR, MUX, ... — que constituyan una librería de puertas lógicas. N es el número de puertas lógicas en la librería LL. Para cada puerta lógica, hay varias versiones de su realización[11] física. Sea $L_k = \{1, \ldots, M_k\}$ el conjunto de versiones de realización física de una puerta lógica k, $k \in LL$. Sea $L = \{L_k \mid \forall\, k \in LL\}$ una librería de células.

Mapeado Tecnológico

Dada una red Booleana $\vec{G}$ correspondiente a un circuito de lógica combinacional obtenido mediante un procedimiento de síntesis tecnológicamente independiente — $\vec{G}$ es una descripción estructural NAND–INV — un conjunto de restricciones de diseño y una librería L; se define como el mapeado M a una transformación de $\vec{G}$ en una *net–list*[12] de puertas lógicas N, que es formalmente equivalente a $\vec{G}$ y satisface las restricciones de diseño.

La medida de las prestaciones de una puerta lógica k, $k \in LL$, es el conjunto de pares (potencia, retardo). Ello se expresa formalmente tal como sigue:

$$\mathcal{L}_k = \{(\tau_{k,l}, \mathcal{P}_{k,l}) \mid k \in LL,\ \forall\, l \in L_k\} \tag{5.22}$$

donde L_k es el conjunto de versiones de realizaciones físicas de la puerta lógica k. Por ejemplo, las puertas BUF1 y BUF4 en una librería de células estándar son las versiones $1\times$ y $2\times$ de un *buffer* no inversor BUF. Una librería de prestaciones $\mathcal{L}$ se emplea para mapear $\vec{G}$ en N mediante la transformación M, y se define como:

$$\mathcal{L} = \{\mathcal{L}_k \mid \forall\, k \in LL\} \tag{5.23}$$

En una librería de células estándar, las prestaciones retardo y potencia son funciones no continuas del tamaño de *driver* de puerta. Sin embargo, para las células full–custom, sus prestaciones son funciones continuas del *driver* de puerta.

Definición 16 (Librería Convexa Objetivo). *Una librería de puertas de prestaciones $\mathcal{L}$ es una librería convexa $\breve{\mathcal{L}}$ cuando para cada una puerta lógica k, $k \in LL$,*

[11] Traducción directa del término *layout*. Con este término se identifica a la vista que representa la disposición de los elementos que conforman los dispositivos e interconexiones de un diseño desde un punto de vista físico.

[12] Una *net–list* es una descripción estructural donde se identifican las implementaciones físicas de los elementos que la componen y sus interconexiones.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

la función potencia de consumo es una función convexa del retardo:

$$\breve{\mathcal{L}} = \{(\tau_k, \breve{\mathcal{P}}_k) \mid \breve{\mathcal{P}}_k = \breve{\mathcal{P}}(\tau_k), \forall\, k \in LL\} \tag{5.24}$$

donde $\breve{\mathcal{P}}(\tau_k)$ *es una función convexa de* τ_k.

Definición 17 (Espacio de Diseño). *Dada una ruta crítica de un circuito* $\vec{G}$*, su espacio de diseño es definido por una función* C*, donde una transformación* M *determina la 3–tupla (retardo, área activa y potencia de consumo),*$(\mathcal{T}, \mathcal{A}, \mathcal{P})$*, esto es* $C : M \longrightarrow \mathcal{T} \times \mathcal{A} \times \mathcal{P}$ *para una condición de carga a la salida de* $\vec{G}$.

Definición 18 (Puertas de Mínima Potencia de Consumo). *Se define las puertas de mínima potencia de consumo* $\mathcal{L}_{\underline{\mathcal{P}}}$ *como sigue:*

$$\mathcal{L}_{\underline{\mathcal{P}}} = \{j \mid j \in \mathcal{L}_{k,\underline{\mathcal{P}}}, \forall\, k \in LL\} \tag{5.25}$$

$$\mathcal{L}_{k,\underline{\mathcal{P}}} = \{(\tau_k, \underline{\mathcal{P}_k}) \mid \underline{\mathcal{P}_k} = min\,(\mathcal{P}_{k,l}), k \in LL, \forall\, l \in L_k\} \tag{5.26}$$

donde $min\,(\mathcal{P}_{k,l})$ *es la potencia de consumo mínima.*

Definición 19 (Curva de Prestaciones). *En particular, todos los puntos no inferiores del espacio de diseño* $C_{tradeoff}$ *definen el mapeado óptimo* $\breve{M}$ *en retardo y potencia de consumo* $(\breve{\mathcal{T}}, \breve{\mathcal{P}})$*, es decir:*

Puntos No Inferiores

$$C_{tradeoff} \subseteq C,\ C_{tradeoff} = \{\breve{M} \mid C(\breve{M}) \in (\breve{\mathcal{T}}, \breve{\mathcal{P}})\}, \tag{5.27}$$

donde $\breve{\mathcal{T}}$ *y* $\breve{\mathcal{P}}$ *son los puntos no inferiores (retardo y potencia de consumo) del espacio de diseño* C.

Teorema 2 (Mapeado de Mínima Potencia de Consumo). *Dada una red Booleana multinivel* $\vec{G}$ *y una librería de puertas convexa* $\breve{\mathcal{L}}$*, el mapeado* $M_{\underline{\mathcal{P}}}$ *de* $\vec{G}$ *en* N *con esas puertas de mínima potencia de consumo es un punto no inferior del espacio de diseño* C*:*

$$M_{\underline{\mathcal{P}}} \in C_{tradeoff} \Leftrightarrow C(M_{\underline{\mathcal{P}}}) \in (\breve{\mathcal{T}}, \breve{\mathcal{P}}) \tag{5.28}$$

Demostración. La potencia de consumo total de una *net–list* N son puntos no inferiores de C, es decir $\mathcal{P} = \sum_{n \in N} \mathcal{P}(n)$, donde $\mathcal{P}(n)$ es la potencia de consumo la puerta n. Debido a que $\mathcal{P}(n)$ es la mínima potencia de consumo de cada puerta n, $\mathcal{P}$ pertenece a $\breve{\mathcal{P}}$.

Para demostrar que T pertenece a $\breve{T}$, por contradicción, se supone que existe un mapeado M con igual potencia de consumo pero menor retardo. En tal caso, La librería $\breve{\mathcal{L}}$ no sería una librería convexa. □

Por otro lado, si se disminuye de forma óptima sólo el retardo de la ruta crítica, se incrementará el tamaño de alguna de las puertas de la ruta crítica. Aumentando, por tanto, la carga de alguno de los nodos de la subred cono lógico de entrada de

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

las puertas $gate$ de la ruta crítica ($\mathfrak{C}_{fanin}(gate)$) produciéndose un incremento de la potencia de consumo de la subred $\mathfrak{C}_{fanin}(gate)$.

Aunque se haya partido de un punto del espacio de diseño mapeado a mínima potencia $M_{\underline{\mathcal{P}}}$, el nuevo punto no pertenece a la curva *trade–off* $M'_{\underline{\mathcal{P}}} \notin C_{tradeoff}$, pues existe la posibilidad de optimizar las señales de entrada a las puertas de la subred $\mathfrak{C}_{fanin}(gate)$, sin afectar al retardo de la ruta crítica. Ello implica un aumento del factor de velocidad de algunas de las puertas que pertenecen al cono lógico de entrada del conjunto ruta crítica.

5.6.2. Curva de Prestaciones

Exploración del Espacio de Diseño

Desde el punto de la curva de prestaciones $M_{\underline{\mathcal{P}}}$ de un circuito de lógica combinacional $\vec{G}$, se define el conjunto ruta crítica se define $path_{cp,0}$, donde el retardo de ruta $\overline{T_0}$ es $T(path_{cp,0})$. Mediante la resolución de 5.21, se obtiene un segundo punto de mapeado de la curva de prestaciones $\breve{M}_0$ para el conjunto ruta crítica $path_{cp,0}$, y con un retardo inferior $\underline{T_0}$. Además, $(\underline{T_0}, \overline{T_0}]$ es el dominio de definición del conjunto ruta crítica $path_{cp,0}$, el intervalo $(M_{\underline{\mathcal{P}}}, \breve{M}_0]$ es el dominio de definición del mapeado (o dominio de definición) del conjunto ruta crítica $path_{cp,0}$.

Ruta Dinámica

La ruta crítica de $\vec{G}$ cambia a medida que se obtienen los puntos de la curva de prestaciones. Por lo tanto, el conjunto inicial definido como conjunto ruta crítica $path_{cp,0}$ de $\vec{G}$ debe ser redefinido — como una ruta $path_{cp,i}$ dinámica — siempre que la optimización progrese.

El conjunto ruta crítica $path_{cp,i}$ se actualiza como sigue. Si el retardo inferior $\underline{\breve{T}_i}$ iguala al máximo retardo de alguna ruta independiente del conjunto ruta crítica, entonces el nuevo subconjunto $path_{cp,j}$ de la red $\vec{G}$ es el resultado de unir la ruta independiente con el conjunto ruta crítica $path_{cp,i}$. Además, los retardos de esas rutas incorporadas a la ruta crítica han de ser contemplados en el cálculo del retardo. Esto se expresa formalmente como:

$$\begin{aligned} path_{cp,j} = \; & \{path_{cp,i} \cup path_{o,p} \cup path_{l,m} \mid \\ & T(path_{o,p}) \geq \underline{\breve{T}_i},\ \forall path_{o,p} \in D(path_{cp,i}) \\ & T(path_{l,m}) = \underline{\breve{T}_i},\ \forall path_{l,m} \in I(path_{cp,i})\} \end{aligned} \tag{5.29}$$

El algoritmo para actualizar el conjunto ruta crítica $path_{cp,i}$ se representa en la figura 5.5. Más allá del punto de la curva de prestaciones $\breve{M}_0$, el circuito presenta un nuevo conjunto ruta crítica $path_{cp,1}$. Una posterior optimización mediante aplicación de la ecuación 5.21 provee un nuevo punto de mapeado de la curva de prestaciones $\breve{M}_1$. El intervalo $(\breve{M}_0, \breve{M}_1]$ es el dominio de definición del mapeado del conjunto ruta crítica $path_{cp,1}$. Sobre el límite superior de $\breve{M}_1$, el circuito presenta un nuevo conjunto ruta crítica $path_{cp,2}$, y así sucesivamente. El proceso de cálculo finaliza cuando se alcance un punto para el cual no hay reducción del retardo con mínima potencia.

5.6.3. Algoritmo de Exploración

$C_{P,tradeoff}()$ es el algoritmo propuesto para calcular la curva entera de retardo frente a potencia de un conjunto ruta crítica (véase la figura 5.6), cuando una red $\vec{G}$ es mapeada a una librería de prestaciones convexa $\breve{\mathcal{L}}$.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

$Updatepath_{cp,i}()$
for each independent path $path_{l,m}$ on critical path set $path_{cp,i}$
if $T(path_{l,m}) \geq T(path_{cp,i})$ then
$path_{cp,i} = path_{cp,i} \cup path_{l,m}$;
end if;
end for;
for each dependent path $path_{o,p}$ on critical path set $path_{cp,i}$
if $T(path_{o,p}) \geq T(path_{cp,i})$ then
$path_{cp,i} = path_{cp,i} \cup path_{o,p}$;
end if;
end for;
end $Updatepath_{cp,i}()$

Figura 5.5: Algoritmo de actualización del conjunto ruta crítica.

$C_{P,tradeoff}()$
given a $\vec{G} = (V, \vec{E}, \tau)$;
map $\vec{G}$ to a $\mathcal{L}_{\underline{A}}$, $\check{M}_0 = M_{\underline{\mathcal{P}}}$;
obtain the critical path set $path_{cp,i}$;
do
update $\check{T}_i = T(path_{cp,i})$;
optimise $path_{cp,i}$ and $\mathfrak{C}_{fanin}(path_{cp,i})$ by Equation 5.21;
$Updatepath_{cp,i}()$;
$\check{T}_j = T(path_{cp,i})$;
while $\check{T}_i \neq \check{T}_j$
end $C_{P,tradeoff}()$

Figura 5.6: Algoritmo para la generación de la curva completa de prestaciones.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

En los algoritmos de mapeado y dominio de definición temporal, para cada conjunto ruta crítica $path_{cp,i}$ de un circuito de lógica combinacional, se ha ignorado el problema de la ruta falsa [106]. La salida del procedimiento $C_{P,tradeoff}()$ es un conjunto de pares $(\breve{T}, \breve{P})$, y para cada 2–tupla se define un mapeado $\breve{M}$, $C_{P,tradeoff} = \{\breve{M} \mid C(\breve{M}) \in (\breve{T}, \breve{P})\}$.

Dominio de Definición

El dominio de definición de un conjunto ruta crítica $path_{cp,i}$ contiene dos puntos sucesivos de mapeado $\breve{M}_i$ y $\breve{M}_{i+1}$. Dado el dominio temporal de definición $(\breve{T}_i, \breve{T}_{i+1}]$ de un conjunto ruta crítica $path_{cp,i}$, dos puntos de la curva de prestaciones se calculan como $C(\breve{M}_i) = (\breve{T}_i, \breve{P}_i)$ y $C(\breve{M}_{i+1}) = (\breve{T}_{i+1}, \breve{P}_{i+1})$.

En un dominio de definición temporal $(\breve{T}_i, \breve{T}_{i+1}]$ de un conjunto ruta crítica $path_{cp,i}$ se calcula la curva de prestaciones, como un problema de dimensionado sobre una red invariante. La red conjunto ruta crítica $path_{cp,i}$ no necesita ser actualizada en el intervalo $(\breve{T}_i, \breve{T}_{i+1}]$. Por lo tanto, cualquier mapeado $\breve{M}_j$ dentro del dominio de definición del conjunto ruta crítica $path_{cp,i}$ depende de un subconjunto invariante de la red $\vec{G}$.

Curva de Prestaciones

La curva completa de prestaciones de un circuito de lógica combinacional $\vec{G}$ se calcula mediante sucesivas optimizaciones de dimensionado sobre diferentes rutas críticas. Inicialmente, en el mapeado $\breve{M}_{\underline{A}}$, el conjunto ruta crítica $path_{cp,0}$ representa un subconjunto de la red $\vec{G}$. A medida que los sucesivos puntos de mapeado $\breve{M}_i$ se obtienen, la subred crece debido a que el conjunto ruta crítica $path_{cp,i}$ es actualizado por la unión con otras rutas.

Cualquier método de optimización para resolver 5.21 es válido, por ejemplo optimizadores heurísticos o combinados heurísticos y algorítmicos, Programación Lineal o No Lineal, entre otros.

El problema de dimensionado de puerta es una transformación particular de una red $\vec{G}$ en una *net–list*[13] de N de puertas. Dada una condición de carga a la salida, las restricciones del diseño son formuladas en términos de potencia de consumo frente a prestaciones de retardo.

Para mapear[14] $\vec{G}$ en N, es necesaria una librería convexa. La librería convexa $\breve{\mathcal{L}}$ contiene células full–custom de puertas lógicas.

Tamaño de driver

Puesto que el problema de dimensionado de puerta se establece sobre puertas de tamaño variable, es conveniente definir la capacidad de tamaño de *driver* (factor β) de la puerta.

El tamaño de *driver* o factor β de una puerta crece linealmente con el ancho de sus transistores internos. Las capacidades internas de la puerta se incrementan linealmente también con el ancho de los transistores internos y, consecuentemente, con el factor de tamaño de *driver*.

En un mapeado M_i de $\vec{G}$ en N, cada puerta del circuito posee un factor de tamaño de *driver* particular. Para una condición de carga de salida, el cálculo de la 2–tupla $(\mathcal{T}, \mathcal{P})$ depende del retardo de propagación y de la potencia de consumo de las puertas.

[13] Este término anglosajón define una red mediante sus elementos computacionales, p.e. puertas, y las interconexiones que la constituye.

[14] Traducción directa del término inglés *mapping*: este consiste en asignar elementos físicos de librería a los términos lógicos que componen la funcionalidad a implementar.

[106] H.C. Chen and D.H.C. Du. Path Sensitization in Critical Path Problem. *IEEE Trans. Comput.–Aided Des. Integr. Circuits Syst.*, 12(2):196–207, 1993.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

5.7. El Problema del Punto de Arranque

Con el fin de obtener la curva de prestaciones retardo–potencia, de similar manera a como se obtuvo la curva de prestaciones retardo–área, es necesario obtener un punto de arranque de la curva de prestaciones retardo–potencia. A partir de ese punto óptimo se obtendrán el resto de puntos de la curva retardo–potencia de consumo. Al igual que en el caso de la obtención de la curva retardo–área, es deseable que dicho punto óptimo de arranque sea conocido y sencillo de obtener [139].

Punto de Arranque

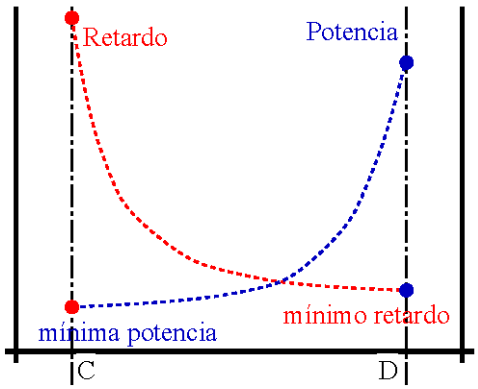


Figura 5.7: Curva de prestaciones retardo–potencia de consumo. La recta C representa el punto de partida de la exploración, mínima potencia de consumo, máximo retardo del circuito. La recta D representa el punto de mínimo retardo del circuito y, por tanto, de máxima potencia de consumo.

Un punto ideal para comenzar el estudio puede ser el punto de mínima potencia de consumo del circuito. En dicho punto óptimo de mínima potencia, se define un retardo que también se considera óptimo. Es decir, no es posible encontrar un dimensionado que mejore el retardo, con igual o inferior potencia de consumo, respecto al que presenta el punto de mínima potencia.

Obtener el punto de mínima potencia de un circuito combinacional no es una tarea trivial. Además, como se apunta en la referencia [113], en general la optimización de potencia de consumo plantea un objetivo que entra en conflicto con el objetivo planteado en la optimización de área. Luego, cabe esperar que el punto de arranque de mínima área, no coincida con el punto de mínima potencia de consumo para una red combinacional dada.

Objetivos Conflictivos

En este trabajo, se han definido dos metodologías para encontrar el punto de mínima potencia de consumo. Una de ellas es heurística y la otra determinista. Mediante comparativas entre los resultados de ambas metodologías se pretende validar el punto obtenido, el error cometido y coste computacional en función de la metodología adoptada. La primera metodología aportada por este trabajo se basa en una solución mediante Algoritmos Genéticos y la segunda metodología es planteada con Programación Lineal.

[139] J. Sosa, J. A. Montiel-Nelson, H. Navarro, and José C. García. Minimum Power Consumption Optimization using Genetic Algorithms. In *Evolutionary and Deterministic Methods for Design, Optimization and Control with Applications to Industrial and Societal Problems 2005*, September 2005.

[113] S. Sapatnekar and Weitong Chuang. Power vs. Delay in Gate Sizing: Conflicting Objectives? *IEEE/ACM International Conference on Computer-Aided Design*, pages 463–466, November 1995.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

5.7.1. Solución Basada en Algoritmos Genéticos

En toda aplicación desarrollada mediante Algoritmos Genéticos es necesario definir dos elementos clave. El primero de ellos es codificar el espacio de soluciones del problema. Ello implica identificar claramente las variables de optimización y definir una codificación para las mismas. En segundo lugar, las diversas tentativas de solución propuestas por el algoritmo genético, han de ser evaluadas con el fin de determinar su bondad. Por ello, es imprescindible especificar una función de coste.

Función de Coste

La solución basada en Algoritmos Genéticos adoptada en este trabajo plantea una función de coste que es la potencia total consumida por el circuito tal que:

$$GA_{Cost} = \sum P_i, \tag{5.30}$$

donde, P_i representa la potencia consumida por la puerta i. La potencia individual de cada puerta sigue el modelo expuesto en la ecuación 5.17.

Función de Coste

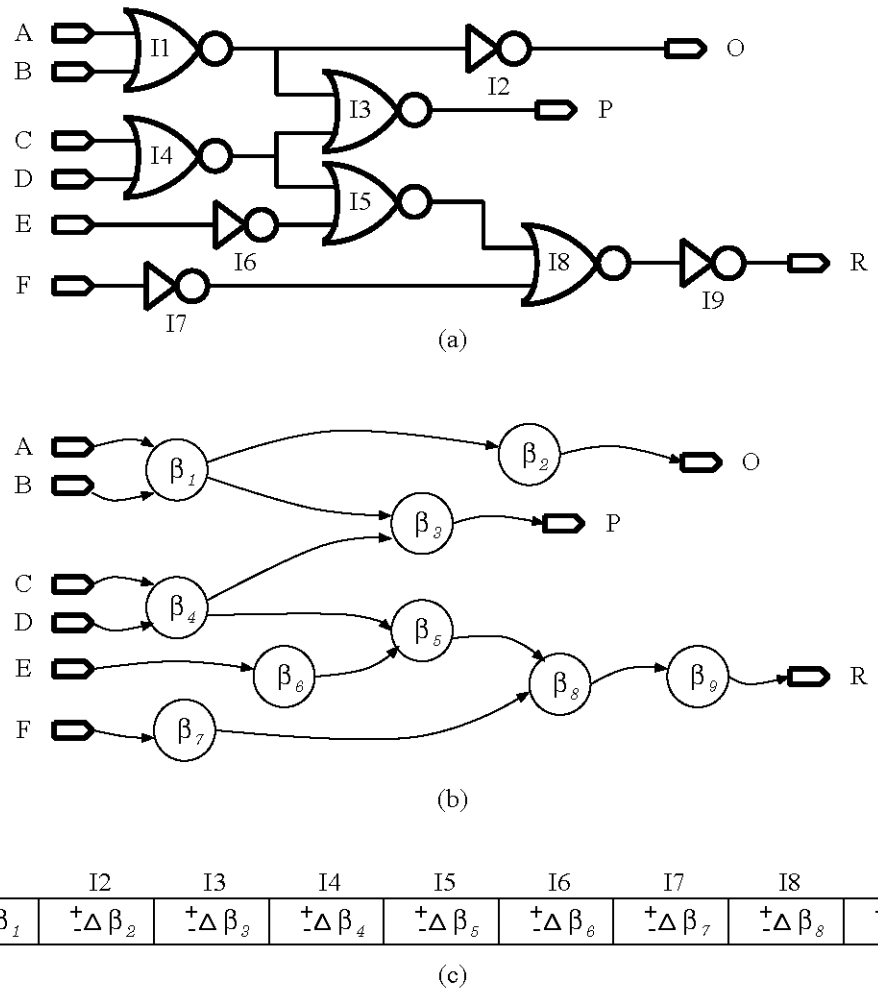


Figura 5.8: Representación y codificación de un circuito de lógica combinacional mediante técnicas basadas en algoritmos genéticos: (a) circuito a optimizar, (b) representación mediante un grafo, y (c) codificación del individuo de la población del algoritmo genético.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

Codificación del Problema

Es evidente que las variables a optimizar son los factores de velocidad de cada una de las puertas. Cada individuo del Algoritmo Genético está formado por un arreglo de tantos cromosomas como puertas contenga el circuito. Cada cromosoma está relacionado biunívocamente con una puerta. El cromosoma contiene una propuesta de cambio relativo al factor de velocidad actual de la puerta que representa. La figura 5.8(c) muestra gráficamente esta codificación.

Función de Coste Incremental

Además, con el fin de reducir la probabilidad de rechazo de un individuo, la evaluación de coste se efectúa de forma incremental. Esto es, la función de coste, se evalúa cromosoma a cromosoma para cada individuo propuesto, frente a la política tradicional de evaluar al individuo como un conjunto de cromosomas indivisibles. Dado un individuo, aplicar la función de coste incrementalmente contribuye a eliminar los cromosomas defectuosos.

Sea el ejemplo de la figura 5.9, el coste total de partida de un individuo es:

$$\begin{aligned} P_{\vec{G}} \;=\; & P_1(max(\tau_{rf,A},\, \tau_{rf,B}),\, \beta_1,\, \beta_2+\beta_3) + \\ & P_2(\tau_{rf,1},\, \beta_2,\, C_O) + \\ & P_3(max(\tau_{rf,1},\, \tau_{rf,4}),\, \beta_3,\, C_P) + \\ & P_4(max(\tau_{rf,C},\, \tau_{rf,D}),\, \beta_4,\, \beta_3+\beta_5) + \\ & P_5(max(\tau_{rf,4},\, \tau_{rf,6}),\, \beta_5,\, \beta_8) + \\ & P_6(\tau_{rf,E},\, \beta_6,\, \beta_5) + \\ & P_7(\tau_{rf,F},\, \beta_7,\, \beta_8) + \\ & P_8(max(\tau_{rf,5},\, \tau_{rf,7}),\, \beta_8,\, \beta_9) + \\ & P_9(\tau_{rf,8},\, \beta_9,\, C_R) \end{aligned} \qquad (5.31)$$

$\Delta\beta_1$	$\Delta\beta_2$	$\Delta\beta_3$	$\Delta\beta_4$	$\Delta\beta_5$	$\Delta\beta_6$	$\Delta\beta_7$	$\Delta\beta_8$	$\Delta\beta_9$
0,0	+2,14e-3	-1,89e-4	0,0	-5,92e-6	0,0	0,0	+7,35e-7	+2,71e-2

Figura 5.9: Ejemplo de individuo propuesto por el algoritmo genético.

Evaluación del Individuo

Supóngase que el individuo propuesto como solución de dimensionado posee unos incrementos/decrementos de factor de velocidad para las puertas tal, que: las propuestas de cambio de las puertas con enumeración par ($\Delta\beta_2$, $\Delta\beta_4$, $\Delta\beta_6$ y $\Delta\beta_8$) aumentan la potencia de consumo total y, las puertas enumeradas con índice impar ($\Delta\beta_1$, $\Delta\beta_3$, $\Delta\beta_5$, $\Delta\beta_7$ y $\Delta\beta_9$) disminuyen la potencia total.

Si se calcula, el coste del individuo de forma tradicional, es evidente que el resultado de dicha operación dependerá de la suma de todas las contribuciones, tanto las que aumentan el coste, como las que lo disminuyen.

En tal caso, el individuo propuesto se evalúa, y la nueva potencia obtenida para

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

la propuesta de cambio del individuo de la figura 5.9 es:

$$\begin{aligned} P'_{\vec{G}} = {} & P_1(\tau_{rf,1_{in}},\ \beta_1+\Delta\beta_1,\ \beta_2+\Delta\beta_2+\beta_3+\Delta\beta_3)+ \\ & P_2(\tau_{rf,2_{in}}+\Delta\beta_{rf,2_{in}},\ \beta_2+\Delta\beta_2,\ C_O)+ \\ & P_3(\tau_{rf,3_{in}}+\Delta\beta_{rf,3_{in}},\ \beta_3+\Delta\beta_3,\ C_P)+ \\ & P_4(\tau_{rf,4_{in}},\ \beta_4+\Delta\beta_4,\ \beta_3+\Delta\beta_3+\beta_5+\Delta\beta_5)+ \\ & P_5(\tau_{rf,5_{in}}+\Delta\beta_{rf,5_{in}},\ \beta_5+\Delta\beta_5,\ \beta_8+\Delta\beta_8)+ \\ & P_6(\tau_{rf,6_{in}},\ \beta_6+\Delta\beta_6,\ \beta_5+\Delta\beta_5)+ \\ & P_7(\tau_{rf,7_{in}},\ \beta_7+\Delta\beta_7,\ \beta_8+\Delta\beta_8)+ \\ & P_8(\tau_{rf,8_{in}}+\Delta\beta_{rf,8_{in}},\ \beta_8+\Delta\beta_8,\ \beta_9+\Delta\beta_9)+ \\ & P_9(\tau_{rf,9_{in}}+\Delta\beta_{rf,9_{in}},\ \beta_9+\Delta\beta_9,\ C_R) \end{aligned} \tag{5.32}$$

Si por el contrario, cada propuesta de cambio es evaluada y aceptada o rechazada parcialmente, el coste final del individuo, poseerá un valor medio superior al coste del computo global; puesto que durante la evaluación parcial del individuo, se han rechazado los cambios de coste negativo. En el ejemplo seguido, los cambios propuestos por las modificaciones planteadas a las puertas con índice par, son rechazadas; asumiendo exclusivamente a las puertas con incremento de potencia negativo, o lo que es lo mismo, los cambios propuestos por las puertas de índice impar. Ello implica que las iteraciones pares no aceptarán el cambio propuesto, frente a las iteraciones impares que si tomarán los cambios propuestos. La traza de la evaluación del coste se describe entonces como:

Iteración I

$$\begin{aligned} P^I_{previous} &= P_1(\tau_{rf,1_{in}},\ \beta_1,\ \beta_2+\beta_3) \\ P^I_{proposal} &= P_1(\tau_{rf,1_{in}},\ \beta_1+\Delta\beta_1,\ \beta_2+\beta_3) \\ P^I_{\vec{G}} &= P_{\vec{G}} - P^I_{previous} + P^I_{proposal} \end{aligned}$$

$$\left.\begin{array}{c} \beta_1 = \beta_1 + \Delta\beta_1 \\ P_{\vec{G}} = P^I_{\vec{G}} \end{array}\right\} \iff P_{\vec{G}} > P^I_{\vec{G}} \cdots (Aceptada)$$

..

Iteración II

$$\begin{aligned} P^{II}_{previous} &= P_1(\tau_{rf,1_{in}},\ \beta_1,\ \beta_2+\beta_3)+P_2(\tau_{rf,2_{in}},\ \beta_2,\ C_O) \\ P^{II}_{proposal} &= P_1(\tau_{rf,1_{in}},\beta_1,\beta_2+\Delta\beta_2+\beta_3)+P_2(\tau_{rf,2_{in}}+\Delta\beta_{rf,2_{in}},\beta_2+\Delta\beta_2,C_O) \\ P^{II}_{\vec{G}} &= P_{\vec{G}} - P^{II}_{previous} + P^{II}_{proposal} \end{aligned}$$

$$\left.\begin{array}{c} \beta_2 = \beta_2 + \Delta\beta_2 \\ P_{\vec{G}} = P^{II}_{\vec{G}} \end{array}\right\} \iff P_{\vec{G}} > P^{II}_{\vec{G}} \cdots (Rechazada)$$

..

Iteración IX

$$\begin{aligned} P^{IX}_{previous} &= P_8(\tau_{rf,8_{in}},\ \beta_8,\ \beta_9)+P_9(\tau_{rf,9_{in}},\ \beta_9,\ C_O) \\ P^{IX}_{proposal} &= P_8(\tau_{rf,8_{in}},\ \beta_8,\ \beta_9+\Delta\beta_9)+P_9(\tau_{rf,9_{in}}+\Delta\beta_{rf,9_{in}},\ \beta_9+\Delta\beta_9,\ C_O) \\ P^{IX}_{\vec{G}} &= P_{\vec{G}} - P^{IX}_{previous} + P^{IX}_{proposal} \end{aligned}$$

$$\left.\begin{array}{c} \beta_9 = \beta_9 + \Delta\beta_9 \\ P_{\vec{G}} = P^{IX}_{\vec{G}} \end{array}\right\} \iff P_{\vec{G}} > P^{IX}_{\vec{G}} \cdots (Aceptada)$$

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

De la descripción realizada en los párrafos anteriores, la aplicación desarrollada con Algoritmos Genéticos se basa en el refinado de una solución previa no óptima en potencia. Dado un punto del espacio de diseño del binomio retardo–potencia, no óptimo en potencia o retardo, el punto de mínima potencia se obtiene mediante el refinado sucesivo de la solución previa. Este refinado finaliza si se alcanza un mínimo no susceptible de ser mejorado o, porque se alcanza el número máximo de intentos permitidos al Algoritmo Genético.

Mínimo Minimorum

Mediante esta técnica heurística es imposible determinar con certeza si el algoritmo ha encontrado un mínimo local o es el mínimo global. Cierto es que el empleo de funciones convexas permite indicar que si se encuentra un mínimo éste es el mínimo global del problema pues, en caso contrario, se llegaría a la conclusión que si existe más de un mínimo, las funciones de potencia empleadas no son funciones convexas.

Punto de Arranque

La elección del punto de partida para alcanzar el punto de mínima potencia no está condicionado a priori en este proceso de refinamiento. Lo deseable es partir de un punto situado entorno al punto de mínima potencia, con el único propósito de reducir el tiempo de búsqueda.

Puesto que no existe manera alguna de identificar un punto cercano al óptimo en potencia, sólo queda determinar el punto de arranque a través de una búsqueda entorno al punto de mínima área.

Refinado Sucesivo

Entonces la obtención del punto de mínima potencia parte de la solución de mínima área de un circuito lógico combinacional. En sucesivas iteraciones del Algoritmo Genético, se refina la solución encontrada hasta que no se alcancen mejoras en potencia o, se alcance el tiempo máximo de ejecución permitido.

Resultados Experimentales

Con el fin de demostrar la efectividad de la metodología propuesta para obtener el punto de mínima potencia de un circuito de lógica combinacional, se ha aplicado al conjunto de circuitos de dos niveles y multinivel, con más de 1000 puertas lógicas, del banco de pruebas denominado MCNC'91.

Núcleo del Algoritmo Genético

Se ha tomado GENEsYs como núcleo que implementa el Algoritmo Genético, adoptando una mutación estándar, cruzamiento multipunto, y la selección de los individuos proporcional. La población total utilizada es de 100 individuos. Y la estrategia de reemplazo de la población antigua por la subpoblación descendiente se basa en la selección de los mejores individuos.

Esta metodología ha sido desarrollada y comprobada en un Pentium IV a 2'8 GHz con 1 $GByte$ de RAM, ejecutando Linux Debian 3.3.4 con el kernel 2.6.7. El compilador empleado ha sido el GCC v3.3.5. El software desarrollado ha sido completamente integrado en el conjunto de herramientas de diseño OLYMPO.

Cada circuito del banco de pruebas MCNC'91 ha sido preprocesado, mediante el sistema de síntesis lógica MISII, a una librería de puertas NAND, NOR e INV para obtener su punto de mínima área. Mediante el refinado de este punto, y con la técnica heurística desarrollada, se ha obtenido el punto de mínima potencia de la red combinacional [139].

Precisión y Esfuerzo Computacional

La tabla 5.1 muestra los resultados de aplicar esta metodología basada en al-

[139] J. Sosa, J. A. Montiel-Nelson, H. Navarro, and José C. García. Minimum Power Consumption Optimization using Genetic Algorithms. In *Evolutionary and Deterministic Methods for Design, Optimization and Control with Applications to Industrial and Societal Problems 2005*, September 2005.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

goritmos genéticos, al punto de mínima área, con el fin de obtener el punto de mínima potencia. La primera columna muestra el nombre del circuito MCNC'91. La segunda y tercera columna presenta el área inicial y potencia de consumo del punto de mínima área. En las dos siguientes columnas se muestra el punto de mínima potencia obtenido mediante la metodología propuesta y, la basada en programación evolutiva. Las columnas sexta y séptima reflejan la diferencia de área entre el punto de mínima área y el punto de mínima potencia obtenido, ambos en términos de área absoluta y área relativa, respectivamente. En las columnas octava y novena se muestra la diferencia entre el punto de mínima área y el punto de mínima potencia alcanzado, en términos de potencia absoluta y relativa, respectivamente. El tiempo necesario para obtener el punto de mínima potencia mediante la solución con algoritmos genéticos se presenta en la última columna. La potencia se encuentra expresada en nW/Hz, el área activa está normalizada al tamaño de un inversor de dimensiones mínimas, que para la tecnología empleada es de 0'1296 μm^2. El tiempo está expresado en segundos.

En la tabla sólo se muestran los resultados para aquellos circuitos con más de 1.000 puertas.

Tabla 5.1: RESULTADOS EXPERIMENTALES EMPLEANDO ALGORITMOS GENÉTICOS PARA LA OBTENCIÓN DEL PUNTO DE MÍNIMA POTENCIA

Circuito MCNC'91	Punto del Espacio de Diseño				Diferencia				Tiempo
	Mínima Área		Mínima Potencia GA		Área		Potencia		
Nombre	Área	Potencia	Área	Potencia	Absoluta	(%)	Absoluta	(%)	CPU
alu4	2.086	704.830	2.091	704.354	5	0'24	476	0'07	8'62
apex1	2.354	790.205	2.358	789.836	4	0'17	369	0'05	9'84
apex3	3.097	1.037.769	3.106	1.036.863	9	0'29	906	0'09	13'69
apex4	4.701	1.593.770	4.717	1.592.099	16	0'34	1.671	0'10	23'19
apex5	2.027	652.248	2.037	651.360	10	0'50	888	0'14	7'11
apex6	1.859	602.470	1.869	601.563	10	0'54	907	0'15	6'62
C2670	1.871	606.149	1.876	605.664	5	0'28	485	0'08	5'89
C3540	3.431	1.148.035	3.441	1.147.110	10	0'29	925	0'08	14'49
C5315	4.716	1.555.337	4.734	1.553.847	18	0'39	1.490	0'10	19'30
C6288	6.587	2.249.785	6.649	2.246.713	62	0'93	3.072	0'14	26'90
C7552	4.809	1.586.014	4.832	1.584.044	23	0'47	1.970	0'12	18'57
cps	2.570	886.551	2.578	885.774	8	0'31	777	0'09	11'89
dalu	2.716	897.651	2.731	896.254	15	0'54	1.398	0'16	10'51
ex1010	5.606	1.887.912	5.621	1.886.386	15	0'27	1.526	0'08	28'57
ex5	1.567	548.178	1.572	547.676	5	0'30	502	0'09	6'51
frg2	2.044	662.695	2.060	660.133	16	0'76	2.563	0'39	6'90
i10	5.600	1.859.808	5.618	1.857.982	18	0'32	1.826	0'10	22'33
i8	2.558	826.523	2.583	818.017	25	0'98	8.506	1'04	9'66
k2	2.340	791.224	2.346	790.576	6	0'24	648	0'08	9'98
pair	3.935	1.294.829	3.943	1.294.097	8	0'19	732	0'06	15'47
rot	1.617	523.469	1.619	523.342	2	0'10	127	0'02	5'43
table3	1.702	576.719	1.707	576.249	5	0'28	470	0'08	7'01
table5	1.608	544.575	1.613	544.157	5	0'28	418	0'08	6'61
x3	2.063	664.064	2.075	662.792	12	0'59	1.272	0'19	7'20

La figura 5.10 presenta los valores de los mejores individuos de cada población, el valor medio de cada población y los resultados de los peores individuos de cada población.

5.7.2. Solución Mediante Programación Lineal

Pros y Contras

A pesar de demostrarse la eficiencia del empleo de Algoritmos Genéticos, y debido a la naturaleza heurística del método, siempre existe la posibilidad de que algún circuito que no converja en el tiempo especificado. De igual forma, una vez

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

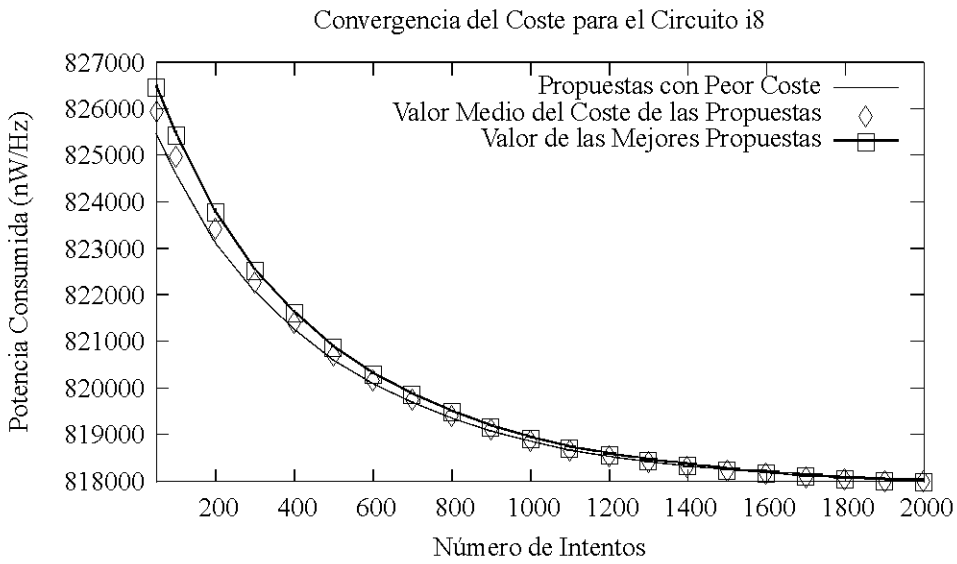


Figura 5.10: Convergencia de la función de coste en la optimización del circuito i8 para la obtención del punto de mínima potencia.

finalizada la ejecución del núcleo, no existe certeza alguna que indique que el punto del espacio de diseño obtenido es el de mínima potencia.

Por el contrario, el empleo de Programación Lineal, debido a su naturaleza determinista, permite afirmar que cuando el algoritmo lineal finaliza, el punto del espacio de diseño obtenido es el punto de mínima potencia. La dificultad de este método radica en la propia linealización del problema.

Modelo de Potencia de Consumo Lineal

Modelo No Lineal

El empleo de un modelo de puerta lineal se hace inevitable cuando se desarrolla una metodología basada en Programación Lineal. Observando en detenimiento el modelo de puerta presentado en el apartado 5.2.1 se desprende que existen varios términos dentro de las ecuaciones 5.7 y 5.8 que son no lineales.

Linealización del Modelo de Potencia

En particular, para la ecuación 5.7, la no linealidad reside en su último término. Esto es cierto mientras se considere que el circuito se estimula con los mismos patrones de entrada durante todo el proceso de exploración de la curva de prestaciones potencia–retardo. Es decir, que la densidad de conmutación de las puertas es constante.

La ecuación 5.7 se describe linealmente mediante las n ecuaciones lineales siguientes:

$$\begin{aligned} P_{gate} &\geq p_{1,1} + p_{1,3}\tau_{rf,gate_{in}} + p_{1,2}\beta_{gate} + p_{1,4}\sum_i \beta_i C_{in,i} \\ &\ldots \\ P_{gate} &\geq p_{n,1} + p_{n,3}\tau_{rf,gate_{in}} + p_{n,2}\beta_{gate} + p_{n,4}\sum_i \beta_i C_{in,i} \end{aligned} \tag{5.33}$$

Linealización del Modelo de Retardo

Del modelo del tiempo medio de transición de las señales presentes a la entrada de la puerta (véase la ecuación 5.18) se puede concluir que es no lineal debido también a su último término. Su representación lineal mediante m tramos lineales

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

es:

$$\begin{aligned} \tau_{rf,gate} &\geq rf_{1,1}\tau_{rf,gate_{in}} - rf_{1,2}\beta_{gate} + rf_{1,3}\sum_i \beta_i C_{in,i} \\ &\ldots \\ \tau_{rf,gate} &\geq rf_{m,1}\tau_{rf,gate_{in}} - rf_{n,2}\beta_{gate} + rf_{n,3}\sum_i \beta_i C_{in,i} \end{aligned} \tag{5.34}$$

Planteamiento del Problema

En general, la optimización en potencia de una red dimensionada a mínima área involucra a todas las puertas del circuito, pues todas ellas son en mayor o menor medida susceptibles de ser optimizadas en potencia. Consecuentemente, la obtención del punto de mínima potencia de un circuito mediante Programación Lineal define un problema que ha de incluir las ecuaciones de potencia de cada una de las puertas que componen el circuito lógico combinacional a optimizar y, por ende, cada una de las ecuaciones de tiempo de transición de la señal presente a la entrada en cada una de las puertas del circuito.

Sea $\vec{G}$ un circuito a optimizar en potencia. Sea β_{gate} el factor de velocidad de la puerta $gate \in \vec{G}$, y $\tau_{rf,gate}$ el tiempo medio de subida y bajada de las señales presentes a la salida de la puerta $gate$. Igualmente, sea P_{gate} la potencia de consumo para de la puerta $gate$.

La descripción del problema en Programación Lineal se define entonces como:

1. Los n modelos de potencia de consumo linealizados para cada puerta del circuito

$$\begin{aligned} &\forall\, gate \in \vec{G} \\ P_{gate} &\geq p_{1,1} + p_{1,3}\tau_{rf,gate_{in}} + p_{1,2}\beta_{gate} + p_{1,4}\sum_i \beta_i C_{in,i} \\ &\ldots \\ P_{gate} &\geq p_{n,1} + p_{n,3}\tau_{rf,gate_{in}} + p_{n,2}\beta_{gate} + p_{n,4}\sum_i \beta_i C_{in,i} \end{aligned} \tag{5.35}$$

2. Los m modelos del tiempo medio de transición presentes a la salida de cada puerta.

$$\begin{aligned} &\forall\, gate \in \vec{G} \\ \tau_{rf,gate} &\geq rf_{1,1}\tau_{rf,gate_{in}} - rf_{1,2}\beta_{gate} + rf_{1,3}\sum_i \beta_i C_{in,i} \\ &\ldots \\ \tau_{rf,gate} &\geq rf_{m,1}\tau_{rf,gate_{in}} - rf_{n,2}\beta_{gate} + rf_{n,3}\sum_i \beta_i C_{in,i} \end{aligned} \tag{5.36}$$

3. La limitación del factor de velocidad β de cada puerta.

$$\begin{aligned} &\forall\, gate \in \vec{G} \\ &\beta_{min} \leq \beta_{gate} \leq \beta_{max} \end{aligned} \tag{5.37}$$

Función Objetivo

La función objetivo se define como la suma de las potencias de cada una de las puertas del circuito.

$$\text{minimize:} \sum_{\forall gate \in \vec{G}} P_{gate} \tag{5.38}$$

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

Complejidad del Problema

Si el circuito se representa como una red Booleana $\vec{G}$ con V nodos (uno por puerta) y E lados (uno por conexión), se puede determinar el tamaño del problema de Programación Lineal equivalente. El número de restricciones es $|E|+(2+n+m)|V|$:

- 1 por cada relación de precedencia,
- 2 por cada puerta con limitación de la β,
- n por cada puerta con su linealización en potencia,
- m por cada puerta con su linealización del tiempo medio de la rampa de salida.

El número de variables necesarias es $3|V|+1$, que corresponden a las variables β, τ_{rf} y P por cada nodo; y para la red completa P. Además. en la práctica las redes Booleanas tienen el *fanin* y el *fanout* limitado, implicando que $|E| < c|V|$ para una constante $c > 1$, siendo el tamaño del problema lineal con el número de puertas del circuito.

Resultados Experimentales

Al igual que con la metodología basada en Algoritmos Genéticos y con el fin de demostrar la efectividad de la metodología, planteada en el apartado anterior, para obtener el punto de mínima potencia de un circuito de lógica combinacional, se ha aplicado la metodología basada en Programación Lineal al conjunto de circuitos de dos niveles y multinivel, con más de 1000 puertas lógicas, del banco de pruebas denominado MCNC'91 [51].

Núcleo de Optimización

Se ha desarrollado el software de formulación del problema lineal para la optimización de potencia sobre la plataforma de síntesis lógica MISII. Y se ha empleado como núcleo de resolución de problemas lineales el programa comercial CPLEX v8.1 [140]. Las pruebas se han desarrollado sobre una plataforma Intel Pentium IV a 2'8 GHz con 1 $GByte$ de RAM, ejecutando Linux Debian 3.3.4 con el kernel 2.6.7. El compilador empleado ha sido el GCC v3.3.5. El software desarrollado ha sido completamente integrado en el conjunto de herramientas de diseño OLYMPO.

Al igual que con las pruebas desarrolladas para comprobar la metodología basada en algoritmos genéticos, cada circuito del banco de pruebas MCNC'91 ha sido preprocesado, mediante el sistema de síntesis lógica MISII, a una librería de puertas NAND, NOR e INV para obtener su punto de mínima área. Mediante la técnica determinista aquí presentada, se ha obtenido el punto de mínima potencia de la red optimizada.

Precisión y Esfuerzo Computacional

La tabla 5.2 muestra los resultados de aplicar esta metodología basada en Programación Lineal, al punto de mínima área, con el fin de obtener el punto de mínima potencia. La tabla identifica en su primera columna el nombre del circuito MCNC'91 optimizado. En su segunda y tercera columna se presenta el área y

[51] S. Yang. *Logic Synthesis and Optimization Benchmarks User Guide version 3.0.* Microelectronics Center of North Carolina, 1991.

[140] Inc. CPLEX Optimization. *CPLEX, ILOG CPLEX 7.0: User's Manual.* Incline Village, NV, June 2000.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

potencia del punto de mínima área, respectivamente. En las dos siguientes columnas se muestra el área y potencia del punto de mínima potencia, respectivamente; obtenido con la optimización basada en Programación Lineal. Desde la columna sexta a la novena se presenta la diferencia en términos de área y potencia, tanto absoluta como relativa, entre el punto de partida (mínima área) y el punto de mínima potencia obtenido, respectivamente. Por último, en la décima columna se refleja el tiempo de CPU necesario para alcanzar el punto de mínima potencia con técnicas de Programación Lineal. Las áreas están normalizadas al tamaño del inversor de mínima área activa, esto es 0'1296 μm^2, así como la potencia de consumo se define en nW/Hz y el tiempo de CPU en segundos. En la tabla sólo se muestran los resultados para aquellos circuitos con más de 1000 puertas.

Tabla 5.2: RESULTADOS EXPERIMENTALES EMPLEANDO PROGRAMACIÓN LINEAL PARA LA OBTENCIÓN DEL PUNTO DE MÍNIMA POTENCIA

Circuito MCNC'91	Punto del Espacio de Diseño				Diferencia				Tiempo
	Mínima Área		Mínima Potencia LP		Área		Potencia		
Nombre	Área	Potencia	Área	Potencia	Absoluta	(%)	Absoluta	(%)	CPU
alu4	2.086	704.830	2.094	704.385	8	0'39	445	0'06	50'60
apex1	2.354	790.205	2.360	789.824	6	0'26	381	0'05	38'34
apex3	3.097	1.037.769	3.110	1.037.026	13	0'42	743	0'07	65'07
apex4	4.701	1.593.770	4.731	1.591.820	30	0'64	1.950	0'12	127'68
apex5	2.027	652.248	2.037	651.435	10	0'50	813	0'12	27'60
apex6	1.859	602.470	1.868	601.544	9	0'48	926	0'15	26'37
C2670	1.871	606.149	1.875	605.826	4	0'24	323	0'05	31'22
C3540	3.431	1.148.035	3.447	1.147.056	16	0'46	979	0'09	143'50
C5315	4.716	1.555.337	4.736	1.553.981	20	0'43	1.356	0'09	256'50
C6288	6.587	2.249.785	6.663	2.246.569	76	1'14	3.216	0'14	301'29
C7552	4.809	1.586.014	4.828	1.584.642	19	0'40	1.372	0'09	234'27
cps	2.570	886.551	2.583	885.760	13	0'51	792	0'09	49'77
dalu	2.716	897.651	2.728	896.732	12	0'45	919	0'10	196'56
ex1010	5.606	1.887.912	5.638	1.885.999	32	0'57	1.913	0'10	916'73
ex5	1.567	548.178	1.575	547.613	8	0'51	565	0'10	140'51
frg2	2.044	662.695	2.052	661.788	8	0'40	907	0'14	113'77
i10	5.600	1.859.808	5.621	1.858.327	21	0'37	1.481	0'08	646'48
i8	2.558	826.523	2.574	820.148	16	0'62	6.375	0'78	358'75
k2	2.340	791.224	2.349	790.573	9	0'38	651	0'08	194'87
pair	3.935	1.294.829	3.944	1.294.172	9	0'22	657	0'05	297'93
rot	1.617	523.469	1.620	523.365	3	0'17	104	0'02	43'38
table3	1.702	576.719	1.709	576.226	7	0'43	492	0'09	114'50
table5	1.608	544.575	1.616	544.140	8	0'50	435	0'08	100'73
x3	2.063	664.064	2.073	663.049	10	0'48	1.016	0'15	111'93

5.7.3. Comparativa

Esfuerzo Computacional

La tabla 5.3 presenta una comparativa entre las dos metodologías desarrolladas en este trabajo en términos de esfuerzo computacional. La primera y segunda columna presentan, respectivamente, el nombre y el número de puertas del circuito optimizado a mínima potencia. La columna tercera y cuarta a su vez muestran el tiempo de cálculo necesario para obtener el punto de mínima potencia para ambas metodologías, partiendo en ambos casos del punto de mínima área. La quinta columna presenta el ratio de tiempos de cálculo entre ambas metodologías. Los tiempos están expresados en segundos.

Precisión Entre Metodologías

Por otro lado, la tabla 5.4 muestra la comparativa entre los resultados experimentales empleando Algoritmos Genéticos y Programación Lineal en términos de potencia y área obtenidos. La primera columna presenta el nombre del circuito MCNC'91 optimizado. Las tres siguientes tuplas presentan el área y potencia del

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

Tabla 5.3: COMPARATIVA EN TÉRMINOS DE TIEMPO DE CPU PARA LA OBTENCIÓN DEL PUNTO DE MÍNIMA POTENCIA MEDIANTE ALGORITMOS GENÉTICOS FRENTE A PROGRAMACIÓN LINEAL

Circuito MCNC'91		Tiempo CPU		Ratio
Nombre	Puertas	LP	GA	
alu4	2.086	50'60	8'62	5'87
apex1	2.354	38'34	9'84	3'90
apex3	3.097	65'07	13'69	4'75
apex4	4.701	127'68	23'19	5'51
apex5	2.027	27'60	7'11	3'88
apex6	1.859	26'37	6'62	3'98
C2670	1.871	31'22	5'89	5'30
C3540	3.431	143'50	14'49	9'90
C5315	4.716	256'50	19'30	13'29
C6288	6.587	301'29	26'90	11'20
C7552	4.809	234'27	18'57	12'62
cps	2.570	49'77	11'89	4'19
dalu	2.716	196'56	10'51	18'70
ex1010	5.606	916'73	28'57	32'09
ex5	1.567	140'51	6'51	21'58
frg2	2.044	113'77	6'90	16'49
i10	5.600	646'48	22'33	28'95
i8	2.558	358'75	9'66	37'14
k2	2.340	194'87	9'98	19'53
pair	3.935	297'93	15'47	19'26
rot	1.617	43'38	5'43	7'99
table3	1.702	114'50	7'01	16'33
table5	1.608	100'73	6'61	15'24
x3	2.063	111'93	7'20	15'55

punto de mínima área, mínima potencia obtenido mediante programación evolutiva y mínima potencia obtenido mediante Programación Lineal, respectivamente. En la octava, novena y décima columna se presenta la diferencia relativa en términos de área entre el punto de mínima área y mínima potencia, así como la diferencia absoluta entre los dos puntos de mínima potencia. La undécima y duodécima columna muestran el área incrementada en términos absolutos. De idéntica forma, desde la columna decimotercera a la decimoquinta se presenta la diferencia en potencia del punto de mínima área con los puntos de mínima potencia, obtenidos mediante Algoritmos Genéticos y Programación Lineal. Las dos últimas columnas presentan la diferencia porcentual de potencia entre el punto de partida y los puntos optimizados en potencia.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

5.8. Formulación del Dimensionado de Puerta Empleando Programación Lineal

Conocida la forma de obtener una solución inicial del espacio de diseño potencia de consumo–retardo, ya sea con una metodología heurística (véase el apartado 5.7.1) o mediante Programación Lineal (véase el apartado 5.7.2), se puede realizar la exploración de la curva completa de prestaciones potencia–retardo. A continuación se detallará, en primer lugar, el modelo de puerta empleado en dicha exploración. En segundo lugar, se presentará el algoritmo de exploración de la curva completa aplicando técnicas de Programación Lineal. Acto seguido se estudiará la complejidad de la optimización planteada en términos de variables y restricciones necesarias para resolver el problema de Programación Lineal planteado.

5.8.1. Modelo Lineal de Puerta

Para realizar la exploración de la curva de prestaciones potencia–retardo, es necesario plantear no sólo un modelo lineal de potencia de consumo (véase el apartado 5.7.2) sino que es imprescindible incluir un modelo lineal de retardo de propagación de puerta. El modelo de propagación empleado en este trabajo fue presentado en el apartado 4.6.4, capítulo 4. Dicho modelo continúa siendo válido en el estudio de la curva de prestaciones potencia de consumo–retardo.

Retardo de Puerta

5.8.2. Curva de Prestaciones Empleando Programación Lineal

El problema de optimización, mediante Programación Lineal, se plantea tal que, definido T_i como el retardo de la salida primaria i, τ_{gate} como el retardo del tiempo de propagación de la puerta $gate$, β_{gate} como el factor de velocidad de la puerta $gate$, y T_{gate} como el retardo a la salida de la puerta $gate$. El tiempo de retardo total del circuito es:

Retardo Total de Circuito

$$T_{max} = \max_{i \in \ salida\ primaria} T_i \tag{5.39}$$

Restricciones

En Programación Lineal, para cada puerta $gate$ que pertenece a la subred ruta crítica $path_{cp}$, se definen las siguientes restricciones.

1. Los n modelos de retardo linealizados de puerta.

$$\begin{aligned} & \forall\ gate\ \in\ path_{cp} \\ \tau_{gate} \ &\geq\ c_{1,1} - c_{1,2}\beta_{gate} + c_{1,3}\sum_i \beta_i C_{in,i} \\ & \dots \\ \tau_{gate} \ &\geq\ c_{n,1} - c_{n,2}\beta_{gate} + c_{n,3}\sum_i \beta_i C_{in,i} \end{aligned} \tag{5.40}$$

2. Los m modelos linealizados de potencia de consumo de puerta.

$$\begin{aligned} & \forall\ gate\ \in\ path_{cp} \\ P_{gate} \ &\geq\ p_{1,1} + p_{1,2}\tau_{rf,gate_{in}} + p_{1,3}\beta_{gate} + p_{1,4}\sum_i \beta_i C_{in,i} \\ & \dots \\ P_{gate} \ &\geq\ p_{m,1} + p_{m,2}\tau_{rf,gate_{in}} + p_{m,3}\beta_{gate} + p_{m,4}\sum_i \beta_i C_{in,i} \end{aligned} \tag{5.41}$$

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

Tabla 5.4: COMPARATIVA, EN TÉRMINOS DE ÁREA, RETARDO Y POTENCIA OPTIMIZADA, PARA OBTENER EL PUNTO DE MÍNIMA POTENCIA DE CONSUMO DE LOS CIRCUITOS DE DOS NIVELES Y MULTINIVEL CON MAS DE 1.000 PUERTAS DEL MCNC'91

Circuito MCNC'91	Punto del Espacio de Diseño						Diferencia entre Puntos									
	Mapeado Mínima Área		Mínima Potencia				Incremento de Área						Reducción de Potencia			
			Algoritmo Genético		Programación Lineal		Absoluta			(%)			Absoluta			(%)
Nombre	Área	Potencia	Área	Potencia	Área	Potencia	GA	LP	|GA-LP|	GA	LP	GA	LP	|GA-LP|	GA	LP
alu4	2.086	704.830	2.091	704.354	2.094	704.385	5'07	8'19	3'12	0'24	0'39	475	444	30	0'07	0'06
apex1	2.354	790.204	2.357	789.835	2.360	789.824	3'96	6'24	2'27	0'17	0'26	368	380	11	0'05	0'05
apex3	3.097	1.037.769	3.105	1.036.863	3.110	1.037.026	8'93	13'20	4'27	0'29	0'42	906	742	163	0'09	0'07
apex4	4.701	1.593.770	4.716	1.592.099	4.731	1.591.819	15'99	30'07	14'09	0'34	0'64	1.671	1.950	279	0'10	0'12
apex5	2.027	652.248	2.037	651.359	2.037	651.435	10'16	10'27	0'12	0'50	0'50	888	813	75	0'14	0'12
apex6	1.859	602.470	1.869	601.562	1.867	601.544	10'00	8'90	1'10	0'54	0'48	907	926	18	0'15	0'15
C2670	1.871	606.149	1.876	605.663	1.875	605.825	5'25	4'46	0'79	0'28	0'24	485	323	162	0'08	0'05
C3540	3.431	1.148.035	3.440	1.147.110	3.447	1.147.055	9'82	16'02	6'20	0'29	0'46	925	979	54	0'08	0'09
C5315	4.716	1.555.337	4.734	1.553.847	4.736	1.553.980	18'37	20'31	1'94	0'39	0'43	1.490	1.356	133	0'10	0'09
C6288	6.587	2.249.785	6.649	2.246.713	6.662	2.246.568	62'07	75'86	13'79	0'93	1'14	3.072	3.216	144	0'14	0'14
C7552	4.809	1.586.014	4.831	1.584.044	4.828	1.584.642	22'93	19'28	3'65	0'47	0'40	1.970	1.371	598	0'12	0'09
cps	2.570	886.551	2.578	885.774	2.583	885.759	8'09	13'27	5'18	0'31	0'51	777	791	14	0'09	0'09
dalu	2.716	897.651	2.730	896.253	2.728	896.731	14'65	12'37	2'28	0'54	0'45	1.397	919	478	0'16	0'10
ex1010	5.606	1.887.912	5.621	1.886.386	5.638	1.885.998	15'34	32'30	16'95	0'27	0'57	1.526	1.913	387	0'08	0'10
ex5	1.567	548.177	1.571	547.675	1.575	547.613	4'69	8'07	3'38	0'30	0'51	502	564	62	0'09	0'10
frg2	2.044	662.695	2.059	660.132	2.052	661.788	15'73	8'22	7'51	0'76	0'40	2.562	906	1.655	0'39	0'14
i10	5.600	1.859.808	5.617	1.857.982	5.620	1.858.327	17'95	20'89	2'94	0'32	0'37	1.826	1.480	345	0'10	0'08
i8	2.558	826.523	2.583	818.017	2.573	820.148	25'42	15'83	9'59	0'98	0'62	8.505	6.374	2.131	1'04	0'78
k2	2.340	791.224	2.345	790.576	2.348	790.573	5'64	8'84	3'20	0'24	0'38	647	650	3	0'08	0'08
pair	3.935	1.294.829	3.942	1.294.097	3.943	1.294.172	7'65	8'82	1'17	0'19	0'22	732	656	75	0'06	0'05
rot	1.617	523.468	1.618	523.342	1.619	523.364	1'59	2'71	1'12	0'10	0'17	126	103	22	0'02	0'02
table3	1.702	576.718	1.706	576.249	1.709	576.226	4'86	7'36	2'50	0'28	0'43	469	492	22	0'08	0'09
table5	1.608	544.574	1.612	544.156	1.616	544.139	4'57	8'02	3'44	0'28	0'50	417	434	16	0'08	0'08
x3	2.063	664.064	2.075	662.792	2.072	663.048	12'21	9'97	2'24	0'59	0'48	1.272	1.015	256	0'19	0'15

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

3. Los k modelos del tiempo de las pendientes presentes en la salida de cada puerta.

$$\begin{array}{rcl} & & \forall\, gate \in path_{cp} \\ \tau_{rf,gate} & \geq & rf_{1,1}\tau_{rf,gate_{in}} - rf_{1,2}\beta_{gate} + rf_{1,3}\sum_i \beta_i C_{in,i} \\ & & \ldots \\ \tau_{rf,gate} & \geq & rf_{k,1}\tau_{rf,gate_{in}} - rf_{k,2}\beta_{gate} + rf_{k,3}\sum_i \beta_i C_{in,i} \end{array}$$

4. La selección del mayor tiempo de pendiente de las entradas.

$$\begin{array}{rcl} & & \forall\, gate \in path_{cp} \\ \tau_{rf,gate_{in}} & \geq \tau_{rf,i} & \forall\, i \in fanin(gate) \end{array} \tag{5.42}$$

5. El factor de velocidad β de cada puerta.

$$\begin{array}{c} \forall\, gate \in path_{cp} \\ \beta_{min} \leq \beta_{gate} \leq \beta_{max} \end{array} \tag{5.43}$$

6. Definición de los tiempos de llegada de cada puerta.

$$\begin{array}{c} \forall\, gate \in path_{cp} \\ T_{gate} \geq T_j + \tau_{gate},\ \forall\, j \in fanin_{gate} \end{array} \tag{5.44}$$

El algoritmo desarrollado $C_{tradeoff,P}()$ (véase la figura 5.11) se puede modificar para incorporar la técnica de Programación Lineal estudiada. El resultado es el algoritmo llamado $LPC_{tradeoff,P}$ expuesto en la figura 5.8.2. Al igual que en el estudio de la curva de prestaciones área–retardo, y con el objeto de comparar la efectividad de la metodología propuesta, se presenta en la figura 5.8.2 el algoritmo $LPSizingP()$. Este último algoritmo incorpora todas las puertas del circuito dentro del problema de optimización.

5.8.3. Complejidad de la Optimización

Dificultad Computacional del Circuito Completo

Si el circuito se representa como una red Booleana $\vec{G}$ con V nodos (uno por puerta) y E lados (uno por conexión), se puede determinar el tamaño del problema de Programación Lineal equivalente. El número de restricciones es $2|E|+(2+n+m+k)|V|+2$:

- 1 por cada relación de precedencia para el retardo de puerta,
- 1 por cada relación de precedencia para el tiempo de transición de la señal de entrada,
- 2 por cada puerta por la limitación de β,
- n por cada modelo de retardo de propagación de puerta,
- m por cada puerta con su modelo su potencia de consumo,
- k por cada puerta con la linealización de su tiempo de transición,

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

$LPC_{tradeoff,P}()$
given a $\vec{G} = (V, \vec{E}, \tau)$;
map $\vec{G}$ to $\mathcal{L}_{\underline{\mathcal{P}}}$, $\breve{M}_i = M_{\underline{\mathcal{P}}}$;
obtain the critical path set $path_{cp,i}$;
do
$\mathcal{T}_i = T(path_{cp,i})$;
minimise $c_P\mathcal{P} + c_T\mathcal{T}$ of $path_{cp,i}$;
s.t.
for each gate $\in path_{cp,i}$
/* n modelos de propagación */
$\tau_{gate} \geq c_{1,1} - c_{1,2}\beta_{gate} + c_{1,3}\sum_i \beta_i C_{in,i}$;
...
$\tau_{gate} \geq c_{n,1} - c_{n,2}\beta_{gate} + c_{n,3}\sum_i \beta_i C_{in,i}$;

/* m modelos de potencia de consumo*/
$P_{gate} \geq p_{1,1} + p_{1,2}\tau_{rf,gate_{in}} + p_{1,3}\beta_{gate} + p_{1,4}\sum_i \beta_i C_{in,i}$;
...
$P_{gate} \geq p_{m,1} + p_{m,2}\tau_{rf,gate_{in}} + p_{m,3}\beta_{gate} + p_{m,4}\sum_i \beta_i C_{in,i}$;

/* k modelos del tiempo de transición a la salida */
$\tau_{rf,gate} \geq rf_{1,1}\tau_{rf,gate_{in}} - rf_{1,2}\beta_{gate} + rf_{1,3}\sum_i \beta_i C_{in,i}$;
...
$\tau_{rf,gate} \geq rf_{k,1}\tau_{rf,gate_{in}} - rf_{k,2}\beta_{gate} + rf_{k,3}\sum_i \beta_i C_{in,i}$;

/* Tiempos de transición */
$\tau_{rf,gate_{in}} \geq \tau_{rf,v}, \forall\, v \in fanin(gate)$;

/* Limitación del factor de velocidad */
$\beta_{gate} \in [\underline{\beta_{gate}}, \overline{\beta_{gate}}]$;

/* Tiempos de propagación */
$T_{gate} \geq T_j + \tau_{gate}, \forall\, j \in fanin(gate)$;
end for;
s.t.
/* Máximo tiempo de propagación */
$\mathcal{T} \geq T(path_{o,p}), \forall\, path_{o,p} \in D(path_{cp,i})$;
$\mathcal{T} \geq max(T(path_{l,m})), \forall\, path_{l,m} \in I(path_{cp,i})$;
update $\breve{\mathcal{T}}_j = T(path_{cp,i})$;
$Updatepath_{cp,i}(path_{cp,i})$;
while $\breve{\mathcal{T}}_i \neq \breve{\mathcal{T}}_j$
end $LPC_{tradeoff,P}()$

Figura 5.11: Algoritmo de generación de la curva completa de prestaciones empleando la técnica de Programación Lineal.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

$LPSizingP()$

minimise $c_P\mathcal{P} + c_T\mathcal{T}$ of $\vec{G}$;

s.t.

$\forall\ gate\ \in\ \vec{G}$:

/* n modelos de propagación */

$\tau_{gate}\ \geq\ c_{1,1} - c_{1,2}\beta_{gate} + c_{1,3}\sum_i \beta_i C_{in,i}$;

...

$\tau_{gate}\ \geq\ c_{n,1} - c_{n,2}\beta_{gate} + c_{n,3}\sum_i \beta_i C_{in,i}$;

/* m modelos de potencia de consumo*/

$P_{gate} \geq p_{1,1} + p_{1,2}\tau_{rf,gate_{in}} + p_{1,3}\beta_{gate} + p_{1,4}\sum_i \beta_i C_{in,i}$;

...

$P_{gate} \geq p_{m,1} + p_{m,2}\tau_{rf,gate_{in}} + p_{m,3}\beta_{gate} + p_{m,4}\sum_i \beta_i C_{in,i}$;

/* k modelos del tiempo de transición a la salida */

$\tau_{rf,gate} \geq rf_{1,1}\tau_{rf,gate_{in}} - rf_{1,2}\beta_{gate} + rf_{1,3}\sum_i \beta_i C_{in,i}$;

...

$\tau_{rf,gate} \geq rf_{k,1}\tau_{rf,gate_{in}} - rf_{k,2}\beta_{gate} + rf_{k,3}\sum_i \beta_i C_{in,i}$;

/* Tiempos de transición */

$\tau_{rf,gate_{in}}\ \geq\ \tau_{rf,v},\ \forall\ v\ \in\ fanin(gate)$;

/* Limitación del factor de velocidad */

$\beta_{gate}\ \in\ [\underline{\beta_{gate}}, \overline{\beta_{gate}}]$;

/* Tiempos de propagación */

$T_{gate}\ \geq\ T_j\ +\ \tau_{gate},\ \forall\ j\ \in\ fanin(gate)$;

s.t.

/* Máximo tiempo de propagación */

$\mathcal{T}\ =\ max(T(v_i)), \forall\ v_i\ \in\ PO$

end $LPSizingP()$

Figura 5.12: Algoritmo de dimensionado de puerta empleando técnicas de Programación Lineal.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

- 1 para expresar el retardo total,
- 1 para limitar el tiempo máximo.

El número de variables necesarias es $5|V|+2$, que corresponden a las variables β, t, τ, T y P por cada nodo; y para la red completa T_{max}, P. Además. en la práctica, las redes Booleanas tienen el *fanin* y el *fanout* limitado, implicando que $|E| < c|V|$ para una constante $c > 1$, siendo el tamaño del problema lineal con el número de puertas del circuito $(2 + n + m + k + c)|V| + 2$.

Dificultad Computacional de la Ruta Crítica

En el caso de optimizar retardo del conjunto ruta crítica $path_{cp,i}$ manteniendo óptima la potencia, en vez de limitar el retardo de las salidas primarias, este tiempo es restringido al retardo de las rutas dependientes del conjunto ruta crítica y a las rutas dependientes de su cono lógico de entrada. Además, el límite inferior del retardo total es el retardo máximo de rutas independientes. El número de restricciones, cuando se optimiza el retardo de la ruta crítica, es menor.

Por cada puerta del cono lógico de entrada del conjunto ruta crítica $path_{cp,i}$ se define el siguiente número de restricciones:

- n por cada modelo de retardo de propagación,
- m por cada modelo de potencia de consumo,
- k para cada tiempo de transición,
- 2 para limitar β y
- 1 por cada relación de precedencia.

Al restringir la formulación a las puertas del cono lógico de entrada del conjunto ruta crítica $path_{cp,i}$ el número de puertas involucradas son un subconjunto de la red Booleana completa y, por tanto, el número de restricciones necesarias es menor a $(n + m + k + 2)|V|$.

Esta metodología, trasladada desde la exploración del espacio de diseño área–retardo, se muestra también eficaz para reducir la complejidad del problema de exploración del espacio de diseño potencia–retardo.

Puntualizaciones sobre la Potencia de Corto Circuito

Atendiendo a la ecuación 5.17, la contribución de la potencia de corto circuito a la potencia total del circuito es $a_{trf}\beta_{gate}\tau_{gate,in}$ en cada puerta.

Nótese que la frecuencia f_i se considera constante en el cálculo, independientemente del método empleado para su obtención. En realidad, f_i no sólo depende del reloj del sistema, sino también está subordinada al conjunto de patrones a los que se somete el circuito. Una variación de dichos patrones supone entonces la modificación de dicha frecuencia. Por ello, elegidos dichos patrones, no se considera la modificación de los mismos y, por tanto, la frecuencia se puede considerar constante.

Igualmente, en referencia a las señales aplicadas, indicar que todas las ecuaciones expresadas hasta ahora son válidas mientras el conjunto de señales de entrada a la puerta, se encuentren definidas dentro del llamado margen de señales

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

con tiempo de subida y bajada de valor medio, tal y como se define en la referencia [141]. Esto es:

$$\tau_i \in [\tau_{i,min}, \tau_{i,max}] \tag{5.45}$$

Tiempo de Subida y Bajada Medio

También, es conocido que el cociente entre los factores de velocidad de una puerta y su *fanout* se mantiene constante a lo largo del circuito (véase la referencia [126]).

Sea n puertas inversoras interconectadas en serie, sus factores de velocidad $\beta_0, \beta_1, \ldots, \beta_{n-1}, \beta_n$ y el cociente entre estos factores $r_0, r_1, \ldots, r_{n-1}$, tal y como se define a continuación:

$$r_1 = \beta_2/\beta_1 \tag{5.46}$$
$$r_2 = \beta_3/\beta_2 \tag{5.47}$$
$$\ldots$$
$$r_{n-1} = \beta_n/\beta_{n-1} \tag{5.48}$$

Entonces

$$\lim_{i\to\infty} \beta_i/\beta_{i-1} = r_{cte} > 1 \tag{5.49}$$

Además, tomando el primer término de la ecuación 5.18 se puede estimar la influencia relativa al tiempo de transición de la señal presente en la entrada.

$$a_{trf} = \frac{1}{6}\left(1 + \frac{2V_T}{V}\right) \tag{5.50}$$

Con $V > 2V_T$:

$$a_{trf} < \frac{1}{3} \tag{5.51}$$

La tabla 5.5 presenta un resumen de las tensiones de alimentación (V) y la tensión umbral de diversas tecnologías CMOS actuales y futuras, según el informe de [114] y otros documentos de diversos fabricantes de tecnologías [142]– [144].

Para todas ellas, el peor caso se da en la tecnología CMOS 0'60 μm donde el coeficiente a_{trf} es aproximadamente un número entre 0'21 y 0'28. Luego se confirma la suposición de la ecuación 5.51.

Ello indica que para cualquier señal que se introduzca en la red Booleana y se propague a través de ella, el tiempo de transición de dicha señal se irá modificando en relación a los parámetros de dimensionado de la red a medida que dicha señal se

Tiempo de Transición

[141] Adnan Kabbani, Dhamin Al-Khalili, and Asim J. Al-Khalili. Technology–Portable Analytical Model for DSM CMOS Inverter Transition–Time Estimation. *IEEE Trans. on Computer–aided Design of Integrated Circuits and Systems*, 22(9):1177–1187, September 2003.

[126] N. Hendestierna and K. O. Jeppson. CMOS Circuit Speed and Buffer Optimization. *IEEE Trans. Computer–Aided Design*, CAD–6:270–281, mar 1987.

[114] ITRS. International Technology Roadmap for Semiconductors. http://public.itrs.net, 2001.

[142] AMS. AustriaMicroSystems AG. Empresa Fabricante de Tecnologias Basadas en Silicio.

[143] TSMC. Taiwan Semiconductor Manufacturing Company Ltd. Empresa Fabricante de Tecnologías, Entre Otras Basadas en Silicio.

[144] UMC. United Microelectronics Corporation. Empresa Fabricante de Diversas Tecnologías Basadas en Silicio.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

Tabla 5.5: COMPARATIVA DEL PARÁMETRO DE INFLUENCIA DEL TIEMPO DE TRANSICIÓN DE LAS SEÑALES EN LA POTENCIA DE CORTO CIRCUITO PARA DIVERSAS TECNOLOGÍAS ACTUALES Y FUTURAS EN TÉRMINOS DE TAMAÑO MÍNIMO DE PUERTA, TENSIÓN DE ALIMENTACIÓN, TENSIÓN UMBRAL Y FABRICANTE

Fabricante	W_{min} (μm)	Alimentación (V)	Umbral (V)	Máximo a_{trf}
IBM [145]	0'13	1'2	0'42	0'28
TSMC [146]	0'18	1'8	0'54	0'27
TSMC [147]	0'25	2'5	0'53	0'24
TSMC [148]	0'35	3'3	0'77	0'24
AMS [149]	0'35	3'3	0'66	0'23
AMS [150]	0'60	5'0	0'70	0'21
AMS [151]	0'80	5'0	0'75	0'22

aleje de las entradas primarias del circuito. Esto es, la influencia del valor inicial del tiempo de transición es despreciable desde que la señal supere un número determinado de puertas. El mencionado margen de señales viene determinado por las posibles variaciones de dimensionado en la red Booleana.

De aquí que en caso de introducir señales fuera de ese margen medio de valores, es decir, más rápidas o lentas, implica que dichas señales serán filtradas y acondicionadas por las puertas de las etapas iniciales. A partir de cierta etapa lógica desde la entrada, el comportamiento de las señales se encuadra dentro de las señales con pendiente de tiempo medio.

La aportación se ha centrado en el estudio del consumo de potencia suponiendo que, las señales presentes a las entradas del circuito tienen tiempos de subida y bajada en el rango definido como valores medios, y nunca señales con tiempos de subida o bajada lentas o muy rápidas. Por otro lado, suponer señales fuera de este margen de valores medios, implica asumir un error acotado y conocido, si se aplican las técnicas descritas. El error cometido en ese supuesto es lineal con el factor de potencia de las puertas las primeras etapas lógicas — más cerca de las entradas primarias — del circuito en estudio.

Por otro lado, siempre que se desea optimizar una puerta con al menos dos entradas, es necesario determinar el tiempo de subida y bajada medio de las señales a la entrada, con el fin de realizar la estimación de su potencia de consumo. En este trabajo se toma el tiempo superior de todos los tiempos medios de subida y bajada presentes en las entradas de la puerta en estudio. Ello implica que las estimaciones tienden a ser pesimistas pues no se emplean los valores estrictamente medios.

Error Acotado

No obstante, escoger el valor máximo o el valor medio no influye sobre el resultado final de manera significativa, pues como se ha indicado, a medida que la señal avanza por la red combinacional su tiempo de transición medio se acerca a un valor medio. Es decir, según una señal supere las primeras etapas lógicas del circuito, el tiempo medio de subida y bajada es, aproximadamente, igual para todas las señales que viajan por la red.

5.9. Resultados Experimentales

A continuación se presentarán los aspectos numéricos de la ejecución del algoritmo de exploración del espacio de diseño, para obtener la curva de prestaciones potencia–retardo, teniendo en cuenta solamente las puertas de la ruta crítica y sus puertas *fanin*, el cono lógico del conjunto ruta crítica y, por último, el circuito

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

completo.

Cada circuito de pruebas MCNC'91 ha sido procesado con la herramienta de síntesis y mapeado lógico de sistemas MISII a una librería de puertas lógicas — NAND, NOR, INV — con mínima área activa. Tras su mapeado a mínima área, se han procesado con el algoritmo de obtención del punto de mínima potencia expuesto en el apartado 5.7. El proceso tecnológico empleado es el CMOS 0'18 μm de UMC, con una tensión de alimentación de 1'8 V.

Se han calculado las curvas de prestaciones de potencia–retardo para la red completa $\vec{G}$ y para el conjunto ruta crítica $path_{cp,i}$. Las ecuaciones de retardo de propagación, la potencia d consumo dinámica y el tiempo de transición de las señales presentes en la salida de las puertas lógicas NAND, NOR, INV son:

$$\begin{aligned}
\tau_{NAND2} &= 64'148 + (179'598 + \sum_i \beta_i \times 17'854)/\beta_{NAND2} \\
\tau_{NOR2} &= 69'275 + (222'847 + \sum_i \beta_i \times 21'344)/\beta_{NOR2} \\
\tau_{INV} &= 36'398 + (150'922 + \sum_i \beta_i \times 15'902)/\beta_{INV}
\end{aligned}$$

$$\begin{aligned}
t_{NAND2} &= 0'164 \times t_{NAND2,in} + 35'174 + (168'737 + \sum_i \beta_i \times 24'652)/\beta_{NAND2} \\
t_{NOR2} &= 0'155 \times t_{NOR2,in} + 37'678 + (223'209 + \sum_i \beta_i \times 30'8212)/\beta_{NOR2} \\
t_{INV} &= 0'199 \times t_{INV,in} + 12'203 + (130'165 + \sum_i \beta_i \times 19'710)/\beta_{INV}
\end{aligned}$$

$$\begin{aligned}
\mathcal{P}_{NAND2} &= 9'889\ C_{l,NAND2} + 148'330 \sum_i \beta_i + 1'164 \beta_i t_{NAND2,in} \\
\mathcal{P}_{NOR2} &= 8'366\ C_{l,NOR2} + 125'490 \sum_i \beta_i + 1'073 \beta_i t_{NOR2,in} \\
\mathcal{P}_{INV} &= 12'091\ C_{l,INV} + 181'370 \sum_i \beta_i + 1'073 \beta_i t_{INV,in}
\end{aligned} \tag{5.52}$$

Los tiempos están expresados en picosegundos (ps) y la potencia en nanovatios por hercio (nW/Hz). Por otro lado, las tecnologías estudiadas presentan la característica *low leakage currents* que garantiza unas corrientes de fuga muy bajas.

5.9.1. Comparativa

Circuitos de Prueba

El algoritmo de exploración de la curva de prestaciones potencia de consumo–retardo se ha aplicado al conjunto de circuitos combinacionales MCNC'91, con dos niveles de lógica. Todos los circuitos son procesados por el sistema de síntesis lógica MISII, siendo mapeados a mínima área con la librería compuesta por puertas NAND, NOR e INV. El modelo de puerta fue ajustado para el proceso CMOS de 0'18 μm de UMC.

La tabla 5.6 muestra los valores referentes al número de variables y tiempo de CPU necesarios para obtener la curva de prestaciones completa en circuitos

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

Tabla 5.6: COMPARACIONES DE RESULTADOS EXPERIMENTALES EMPLEANDO PROGRAMACIÓN LINEAL PARA OBTENER LA CURVA DE MÍNIMA POTENCIA DE CONSUMO DE LOS CIRCUITOS DE DOS NIVELES Y MULTINIVEL CON MAS DE 1.000 PUERTAS DEL MCNC'91

Circuito MCNC'91		Red Booleana $\vec{G}$ (red Completa)		Conjunto Ruta Crítica (sólo $path_{cp,i}$)				Relación	
		Variables	Tiempo	Variables Necesarias			Tiempo	Variables	Tiempo
Circuito	Puertas	Necesarias	CPU	Mínimas	Medias	Máximas	CPU	Necesarias	CPU
alu4	1.360	6.802	2'03	702	3.916'80	4.872	0'66	1'74	3'08
apex1	1.527	7.637	2'24	772	4.531'30	6.102	0'79	1'69	2'84
apex3	1.985	9.927	6'25	1.367	6.616'90	8.347	1'93	1'50	3'24
apex4	3.010	15.052	30'51	2.482	11.883'35	14.427	9'31	1'27	3'28
apex5	1.281	6.407	1'00	617	3.780'25	4.977	0'43	1'70	2'33
apex6	1.196	5.982	0'34	177	1.615'70	2.627	0'08	3'71	4'10
C2670	1.232	6.162	0'58	397	2.504'85	3.332	0'22	2'46	2'64
C3540	2.198	10.992	5'24	1.292	7.295'85	9.072	1'92	1'51	2'73
C5315	2.988	14.942	4'56	1.042	6.700'80	9.132	1'16	2'23	3'93
C6288	4.139	20.697	28'05	1.842	15.259'60	17.217	7'17	1'36	3'91
C7552	3.095	15.477	4'54	1.247	6.841'10	9.637	1'13	2'26	4'02
cps	1.666	8.332	2'94	1.067	5.238'70	7.187	1'02	1'59	2'88
dalu	1.754	8.772	1'48	582	3.692'45	5.092	0'47	2'38	3'15
ex1010	3.558	17.792	48'73	2.982	14.336'50	17.177	15'30	1'24	3'18
ex5	1.034	5.172	2'10	752	4.007'40	4.617	0'89	1'29	2'36
frg2	1.307	6.537	0'75	307	2.774'50	3.837	0'25	2'36	3'00
i10	3.643	18.217	4'10	737	4.565'65	7.057	0'79	3'99	5'19
i8	1.603	8.017	1'30	397	3.012'35	5.522	0'34	2'66	3'82
k2	1.523	7.617	2'98	912	4.846'60	6.352	1'01	1'57	2'95
pair	2.562	12.812	1'08	322	2.778'55	4.887	0'18	4'61	6'00
rot	1.058	5.292	0'29	177	1.302'05	1.637	0'08	4'07	3'62
table3	1.094	5.472	1'80	867	4.229'25	5.142	0'79	1'29	2'28
table5	1.034	5.172	1'76	817	4.113'85	4.767	0'75	1'26	2'35
x3	1.323	6.617	0'67	392	2.349'15	3.312	0'21	2'82	3'19

multinivel y de dos niveles de lógica con al menos 1000 puertas combinacionales. La primera columna da el nombre del circuito de acuerdo con MCNC'91 Benchmark Suite. La segunda columna representa la complejidad del circuito medida en número de puertas mapeadas. El número de variables y el tiempo de CPU empleado cuando se optimiza el total de la red Booleana $\vec{G}$ — empleando Programación Lineal y la aproximación a tramos de las curvas no lineales del modelo de puerta — se presentan en la tercera y cuarta columna. Las tres siguientes columnas presentan el número mínimo, medio y máximo de variables, respectivamente, empleadas durante la optimización del circuito con Programación Lineal mediante el método de la ruta crítica. La octava columna muestra el tiempo medio necesario para resolver el problema de Programación Lineal empleando la ruta crítica. La penúltima y última columna presenta la ganancia en términos de número de variables y tiempo de CPU, respectivamente, entre la metodología basada en la ruta crítica y la metodología tradicional.

Esfuerzo Computacional

La figura 5.13 presenta el número de variables necesarias para obtener la curva completa de prestaciones en función del número total de puertas del circuito $\vec{G}$ explorado. La figura presenta solamente los resultados para los circuitos MCNC'91 con más de mil puertas lógicas. Los ejes de coordenadas están expresados en unidades logarítmicas. Los triángulos negros representan el número de variables empleadas cuando se optimiza la red Booleana completa, mientras que los diamantes indican el número medio de variables necesarias cuando sólo se optimiza el subconjunto ruta crítica $path_{cp,i}$. Si la optimización se realiza empleando todas las puertas del circuito, el número de variables necesarias es lineal con el número de puertas de la red Booleana completa. Como era de esperar y ya se avanzó en el apartado 5.8.3 dicho comportamiento se modela mediante la ecuación lineal

Número de Variables

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

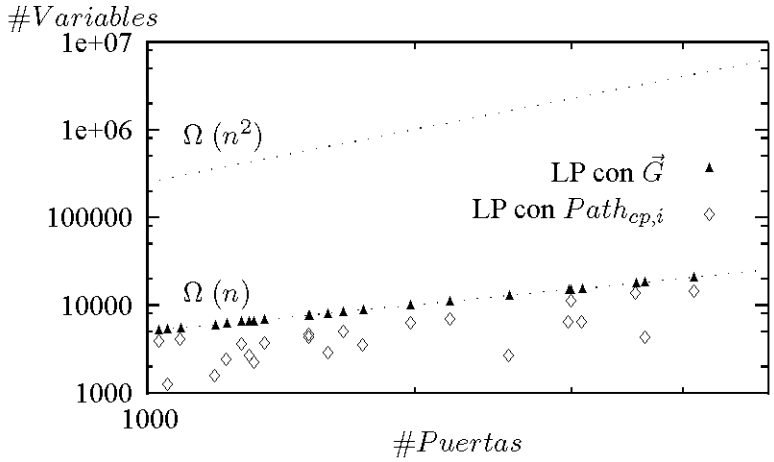


Figura 5.13: Número de variables frente a número de puertas.

$5|V|+2$. Igualmente, se aprecia claramente que emplear el subconjunto ruta crítica reduce el número de variables en todos los casos. La mejora obtenida debido al empleo del subconjunto ruta crítica $path_{cp,i}$ frente a la red completa $\vec{G}$ es de al menos 1'24 veces y hasta 4'61 veces menos variables.

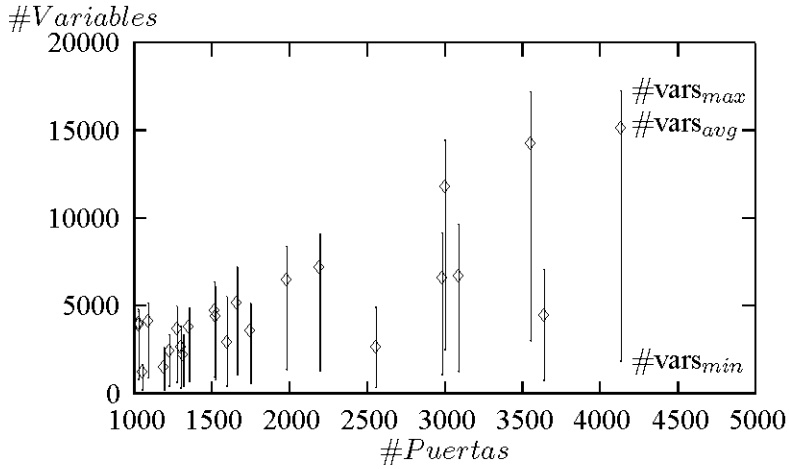


Figura 5.14: Variaciones en el número de variables frente al número de puertas.

Ruta Crítica Dinámica

El carácter dinámico del subconjunto ruta crítica hace necesario el estudio del número de variables necesarias en términos estadísticos. La figura 5.14 presenta el número medio, y la desviación máxima y mínima de variables necesarias para obtener la curva completa de prestaciones. Se observa que el número de variables necesarias es superior a medida que el circuito aumenta su tamaño y por tanto su complejidad. Igualmente, a medida que el circuito contiene mayor número de puertas, la variación del número de puertas necesarias es mayor. Ello es debido a que, en el comienzo de la exploración, sólo unas pocas puertas definen el subconjunto ruta crítica. A medida que se reduce el retardo del circuito, el subconjunto ruta crítica está compuesto por más y más puertas. Al finalizar el estudio de la curva de prestaciones, la mayoría de las puertas del circuito pertenecen al subconjunto ruta crítica. También hacer notar que el valor medio de variables necesarias está más próximo al extremo máximo de variables que al extremo mínimo. Ello indica que, el subconjunto ruta crítica aumenta en número de puertas muy rápida-

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

mente al comienzo de la optimización y, permanece cercano al número máximo de puertas el resto del estudio.

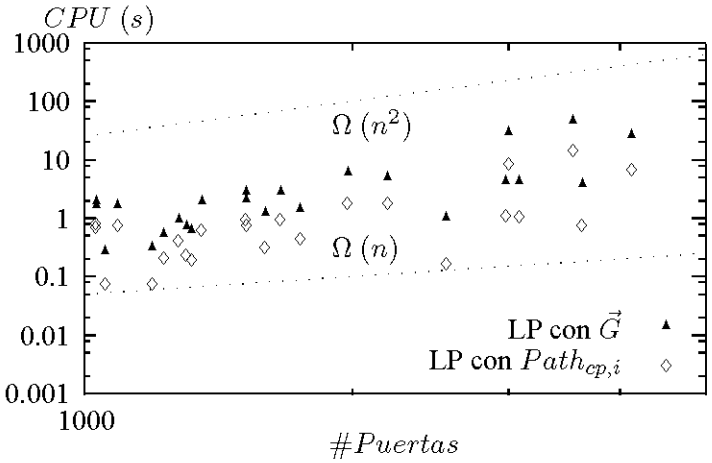


Figura 5.15: Tiempo de CPU frente a número de puertas.

Dimensión del Problema

La figura 5.15 muestra el tiempo de CPU, expresado en segundos, frente al tamaño del circuito completo, medido en número de puertas, para el cálculo de la curva completa de prestaciones de la red $\vec{G}$ y el conjunto ruta crítica $path_{cp,i}$. Reseñar que los ejes de esta figura se encuentran expresados en escala logarítmica. Los triángulos negros representan el tiempo de ejecución cuando se involucra la red Booleana completa $\vec{G}$. Los diamantes presentan el tiempo cuando se optimiza el conjunto ruta crítica $path_{cp,i}$. Se puede observar que el tiempo necesario siempre es superior al orden de magnitud $\Omega(n)$ pero inferior a $\Omega(n^2)$. Igualmente, emplear el conjunto ruta crítica permite reducir el tiempo de cómputo siempre. La mejora se sitúa entre 2'33 y 6 veces.

5.9.2. Curvas de Prestaciones

Cono Lógico Frente a Ruta Crítica

Calculadas todas las curvas completas de prestaciones potencia de consumo–retardo de los circuitos de dos niveles y multinivel del banco de pruebas MCNC'91, tanto empleando la red Booleana completa $\vec{G}$, como utilizando el cono lógico de entrada de la ruta crítica $\mathfrak{C}_{fanin}(path_{cp,i})$, se concluye que ambas metodologías proveen los mismos resultados numéricos.

Error Cometido

Se ha cuantificado la dependencia del computo de las prestaciones potencia de consumo–retardo cuando se tienen en cuenta sólo las puertas de la ruta crítica $path_{cp,i}$ y sus puertas *fanin* $fanin(path_{cp,i})$. Las puertas de la ruta crítica determinan el retardo de la red completa y sus puertas *fanin* modifican la pendiente media de las señales presentes en las entradas de la ruta crítica.

Aplicada la metodología propuesta, a los circuitos MCNC'91 y exclusivamente a las puertas de la ruta crítica y sus puertas *fanin*, se obtienen curvas de prestaciones idénticas. La tabla 5.7 presenta la diferencia numérica existente en términos de error porcentual, entre la exploración del espacio de diseño empleando la ruta crítica junto a sus puertas *fanin* y la red completa. La primera columna de la tabla 5.7 indica el nombre del circuito. La segunda y tercera columnas presentan los errores cometidos en la estimación de la potencia de consumo, como porcentaje medio y máximo, respectivamente, del valor alcanzado con la metodología basada en la red completa frente al alcanzado con la metodología basada en la ruta

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

Tabla 5.7: COMPARACIONES DE POTENCIA Y ÁREA ACTIVA NECESARIA, EN TÉRMINOS DE ERROR PORCENTUAL COMETIDO, DE LAS CURVAS DE PRESTACIONES POTENCIA DE CONSUMO–RETARDO Y ÁREA–RETARDO DE LA METODOLOGÍA DE EXPLORACIÓN DEL ESPACIO DE DISEÑO POTENCIA DE CONSUMO–RETARDO, EMPLEANDO EXCLUSIVAMENTE LA RUTA CRÍTICA O LA RED BOOLEANA COMPLETA DE LOS CIRCUITOS DE DOS NIVELES Y MULTINIVEL CON MÁS DE 1000 PUERTAS DEL BANCO DE PRUEBAS MCNC'91

Circuito MCNC'91 Nombre	Error entre curvas			
	Potencia		Área	
	Medio	Máximo	Medio	Máximo
alu4	0'17	0'44	0'22	0'46
apex1	0'03	0'09	0'14	0'26
apex3	0'01	0'02	0'05	0'08
apex4	0'07	0'14	0'04	0'09
apex5	0'05	0'08	0'04	0'10
apex6	0'05	0'12	0'03	0'08
C2670	0'09	0'20	0'13	0'40
C3540	0'03	0'08	0'14	0'20
C5315	0'07	0'12	0'05	0'12
C6288	0'21	0'59	0'41	1'00
C7552	0'08	0'22	0'14	0'40
cps	0'09	0'15	0'04	0'11
dalu	0'05	0'11	0'11	0'42
ex1010	0'05	0'09	0'04	0'14
ex5	0'08	0'14	0'06	0'18
frg2	0'01	0'03	0'01	0'05
i10	0'07	0'20	0'08	0'31
i8	0'04	0'09	0'02	0'09
k2	0'04	0'08	0'06	0'18
pair	0'02	0'04	0'02	0'04
rot	0'02	0'05	0'02	0'11
table3	0'05	0'08	0'04	0'08
table5	0'04	0'09	0'04	0'14
x3	0'02	0'05	0'01	0'03

crítica y sus puertas *fanin*. Las dos últimas columnas muestran el error porcentual cometido con el área activa empleada en ambos métodos de optimización.

El circuito C6288 tiene las diferencias máximas, tanto en términos de potencia de consumo como de área activa. El error cometido en la estimación de potencia de consumo para este circuito MCNC'91 tiene un valor medio de 0'21 % y un valor máximo de 0'6 %. En cuanto al área activa, esa diferencia se eleva en el caso medio a un 0'41 % y un 1 % en valor máximo.

Curva de Prestaciones

La figura 5.16 presenta las curvas de prestaciones potencia de consumo, área activa y retardo del circuito C6288 tanto optimizando la red Booleana completa, como el subconjunto ruta crítica. En esta figura 5.16 se puede observar como las curvas son idénticas salvo el pequeño error ya indicado. El error cometido se incrementa a medida que la potencia es mayor, o lo que es lo mismo, el retardo se reduce. La mayor divergencia se alcanza en el punto de mínimo retardo–máxima potencia de consumo (lado derecho de la curva).

La figura 5.17 muestra la curva completa de prestaciones potencia de consumo–retardo en función del área activa empleada del circuito apex6. La potencia consumida se representa mediante círculos negros y el retardo de la red mediante triángulos negros cuando se realiza el estudio involucrando la red Booleana completa $\vec{G}$. Igualmente, es esta misma figura 5.17 los triángulos blancos representan el retardo y los círculos blancos la potencia consumida cuando se optimiza el conjunto ruta crítica $path_{cp,i}$ y sus puertas *fanin* $fanin(path_{cp,i})$. El eje de abscisas presenta el área activa empleada en unidades de miles de micrómetros cuadrados. El eje de ordenadas izquierdo indica el retardo en nanosegundos y el eje de

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

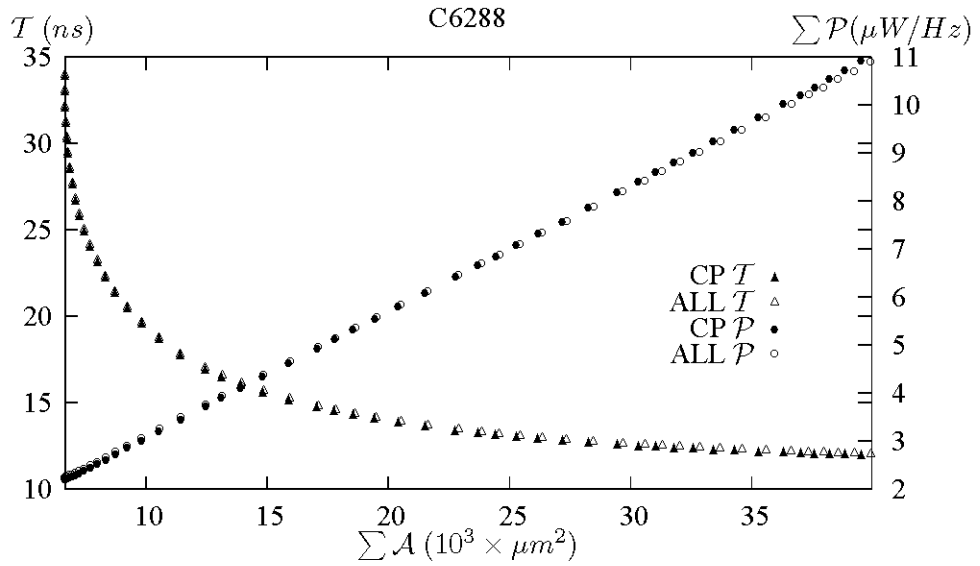


Figura 5.16: Curva completa de prestaciones (potencia de consumo, retardo de propagación y área activa) empleada el circuito C6288 optimizando la red Booleana $\vec{G}$ y el conjunto ruta crítica $path_{cp,i}$.

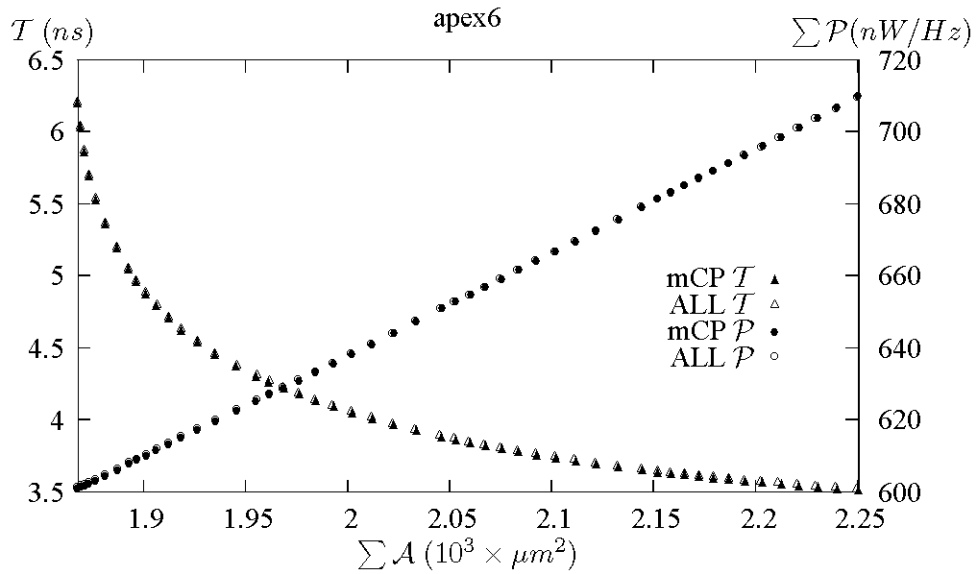


Figura 5.17: Curva completa de prestaciones (consumo de potencia, retardo de propagación y área activa) empleada el circuito apex6 optimizando la red Booleana $\vec{G}$ y el conjunto ruta crítica $path_{cp,i}$.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

ordenadas derecho la potencia consumida en unidades de microvatios por hercio (nW/Hz), es decir, nanojulios (nJ).

Comportamiento Lineal

También se puede observar que la tendencia de la potencia de consumo es lineal con el área activa empleada. Ello se debe a que para cualquier señal que atraviesa la red Booleana, su tiempo de transición alcanza los denominados valores medios de transición. Aumentar dicho valor medio, con el objeto de reducir la potencia de consumo total del circuito, implica aumentar la pendiente de las señales que transitan por la red Booleana. Se logra aumentar la pendiente de estas señales, aumentando la potencia de consumo de ciertas puertas lógicas de la red Booleana escogidas estratégicamente. El incremento de potencia de esas puertas lógicas, que producen el aumento de pendiente de las señales que transitan por la red, es muy superior a la reducción de potencia de cortocircuito lograda aumentando dichas pendientes.

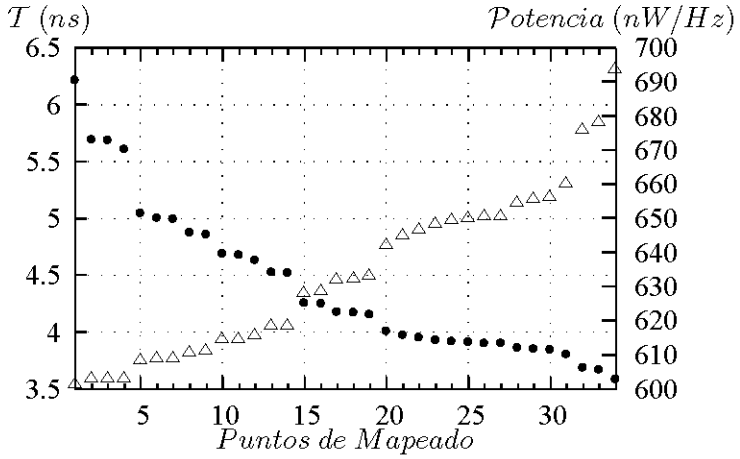


Figura 5.18: Curva completa de la prestación consumo de potencia frente a retardo para el circuito apex6.

Curva de Prestaciones

La figura 5.18 presenta la curva de prestaciones potencia–retardo frente a los diversos puntos de mapeado que la definen, para el circuito apex6. Cada uno de los puntos representados identifica un cambio de dominio del conjunto ruta crítica. Los ejes de ordenadas derecho e izquierdo tienen las mismas unidades que la figura anterior. El eje de abscisas en cambio muestra el punto de mapeado que determina el cambio de dominio del conjunto ruta crítica. Los triángulos representan la potencia de consumo empleada y los círculos, el retardo del circuito.

El circuito apex6 se compone de una red Booleana de 1196 puertas lógicas. Posee 135 entradas primarias y 99 salidas primarias. Para el circuito se obtuvieron 34 dominios temporales del conjunto ruta crítica distintos, es decir, 35 puntos de mapeado distinto. Recordar que entre dos puntos de mapeado sucesivos, el conjunto ruta crítica no varía. Las dimensiones mínimas de los transistores empleados en las puertas lógicas son idénticos a los empleados en la exploración de la curva completa de área–retardo. Es decir, 0'48 $\mu m\times$ 0'18 μm para el transistor de *pull–up* y 0'24 $\mu m\times$ 0'18 μm para el de *pull–down*. La capacidad de cableado parásita contemplada es de 15 fF. Y cada salida del circuito ha sido cargada con un inversor.

Error Cometido

La tabla 5.8 presenta los errores mínimos, medios y máximos cuando se comparan las curvas de prestaciones potencia–retardo empleando, exclusivamente, la ruta $path_{cp,i}$ y, las curvas de prestaciones de potencia–retardo empleando la red Booleana completa $\vec{G}$. La primera columna de la tabla 5.8 presenta el nombre del

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

Tabla 5.8: COMPARACIONES, EN TÉRMINOS DE ERROR PORCENTUAL COMETIDO, DE LAS CURVAS DE PRESTACIONES POTENCIA DE CONSUMO–RETARDO DE LA METODOLOGÍA DE EXPLORACIÓN DEL ESPACIO DE DISEÑO ÁREA–RETARDO BASADO EN LA RUTA CRÍTICA Y LA EXPLORACIÓN DEL ESPACIO DE DISEÑO POTENCIA DE CONSUMO–RETARDO EMPLEANDO LA RED BOOLEANA COMPLETA SOBRE LOS CIRCUITOS DE DOS NIVELES Y MULTINIVEL CON MÁS DE 1000 PUERTAS DEL BANCO DE PRUEBAS MCNC'91

Circuito MCNC'91	Error Porcentual		
Nombre	Mínimo	Medio	Máximo
alu4	0'08	3'32	7'87
apex1	0'05	1'14	2'61
apex3	0'07	0'92	1'40
apex4	0'12	1'61	2'51
apex5	0'13	0'45	0'95
apex6	0'16	0'41	0'96
C2670	0'07	2'08	5'41
C3540	0'10	1'64	3'34
C5315	0'09	2'14	4'83
C6288	0'16	3'77	8'57
C7552	0'09	2'68	7'97
cps	0'09	1'12	2'44
dalu	0'12	3'05	7'14
ex1010	0'10	1'57	2'64
ex5	0'11	2'66	5'82
frg2	0'00	0'11	0'29
i10	0'09	1'80	5'14
i8	0'76	0'96	1'62
k2	0'09	1'20	2'80
pair	0'00	0'09	0'25
rot	0'03	0'82	2'77
table3	0'09	1'90	3'67
table5	0'11	2'05	3'89
x3	0'00	0'09	0'29

circuito MCNC'91. La segunda, tercera y cuarta columna presentan la diferencia porcentual mínima, media y máxima alcanzada, respectivamente.

Como se puede observar en la tabla 5.8 la diferencia media va desde un 0'09 % a un 3'77 %. La diferencia máxima se mantiene por debajo del 8'57 %. El circuito que presenta más divergencia es el C6288, que corresponden los límites superiores de los rangos mencionados. El circuito que presenta menos divergencia es el *pair* el cual posee una divergencia media de sólo el 0'09 % y una máxima del 0'25 %.

5.10. Conclusiones

La exploración del espacio de diseño se puede abordar con una metodología de cómputo de soluciones no inferiores utilizando la ruta crítica dinámica $path_{cp}$. Se ha aportado un procedimiento para la definición del dominio temporal de una ruta crítica. Este método es eficiente, pues permite la actualización de la ruta crítica a medida que se va explorando el conjunto de soluciones no inferiores.

Resultados experimentales con tecnología CMOS de 0'18 μm de UMC indican una reducción en los tiempos de CPU entre 2'33 a 6 veces. Esta disminución del tiempo de computación se debe a la reducción en el número de variables entre 1'50 y 4'61 veces. Cuanto mayor sea la complejidad del circuito mayor es la ganancia obtenida.

Se ha confirmado que el punto de mínima potencia de consumo no coincide con el punto de mínima área en la tecnología seleccionada. Y se ha aportado

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

una metodología para obtener el punto de mínima potencia de consumo a partir del punto de mínima área. La metodología propuesta obtiene el punto de mínima potencia de consumo de forma óptima en términos de tiempo requerido de CPU y número de variables necesarias.

La diferencia entre ambos puntos del espacio de diseño en términos de área activa no supera el 1'14 % lo cual supone un 0'78 % en potencia de consumo. Esta pequeña divergencia es suficiente para provocar una gran divergencia entre el valor óptimo y el calculado a medida que la curva de prestaciones es calculada.

Además, de los resultados se desprende que la potencia de consumo por unidad de retardo es lineal con el área activa empleada en esta tecnología. Ello permite estimar la potencia de consumo con un mínimo esfuerzo computacional sin tener que recurrir al estudio completo de la curva de prestaciones. Las diferencias entre las curvas de retardo–potencia de consumo cuando sólo se emplea la ruta crítica y cuando se emplea el cono lógico de entrada de la ruta crítica poseen un error que en la mayoría de los casos es despreciable y no supera el 8'57 % en el caso más desfavorable de todas las pruebas realizadas.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

Bibliografía

[1] Sedar Tasiran and Kurt Keutzer. Coverage Metrics for Functional Validation of Hardware Designs. *IEEE Design and Test of Computers*, 18(4):36–45, August 2001.

[2] EDA today. Electronic Design Automation User Study, 1999.

[3] Edmund M. Clarke and Jeannette M. Wing. Formal Methods: State of the Art and Future Directions. *ACM Computing Surveys*, 28(4):626–643, 1996.

[4] Israel Cidon, Amit Gupta, Tony Hsiao, Asad Khamisy, Abhay Parekh, Raphael Rom, and Moshe Sidi. OPENET: An Open and Efficient Control Platform for ATM Networks. *IEEE INFOCOM*, pages 824–831, April 1998.

[5] Carl-Johan Seger. *An Introduction to Formal Hardware Verification*. PhD thesis, Deparment of Computer Science, Univ. British Columbia, 1992.

[6] Elliot Mendelson. *Introduction to Mathematical Logic*. Chapman & Hall/CRC, 4th edition, June 1997.

[7] G.V. Bochmann. Hardware Specification with Temporal Logic: An Example. *IEEE Transctions on Computers*, C-31(3):223–231, March 1982.

[8] M. Browne, E. Clarke, D. Dill, and B. Mishra. Automatic Verification of Synchronous Circuits using Symbolic Logic Simulation an Temporal Logic. In *Proc. of IMEC IFIP International Workshop on Applied Formal Methos for Correct VLSI Design*, 1989.

[9] H. Navarro, J. A. Montiel-Nelson, J. Sosa, and José C. García. A Symbolic Solver to Obtain the Entire Solution Space of the Boolean Satisfiability Problem using Genetic Algorithms. In *Evolutionary and Deterministic Methods for Design, Optimization and Control with Applications to Industrial and Societal Problems*, September 2005.

[10] Dinos Moundanos, Jacob A. Abraham, and Yatin-Vasant Hoskote. Abstraction Techniques for Validation Coverage Analysis and Test Generation. *IEEE Transactions on Computers*, 47(1):2–14, 1998.

[11] Gale Morrison. Cadence Open–Sources T–Builder. *Electronics News*, August 2000.

[12] Qualis Design Corporation. Empresa Dedicada a la Verificación de Sistemas (Adquirida Recientemente por Synopsys Inc.).

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

[13] John R. Wallack and Ramaswami Dandapani. Coverage Metrics for Functional Test. *Proc. of the 12th IEEE VLSI Test Symposium*, pages 176–181, 1994.

[14] Verisity Inc. (Recientemente Adquirida por Cadence Inc.). Empresa Desarrolladora del e–lenguaje y del Entorno de Verificación Specman.

[15] Synopsys Inc. *Synopsys User Manual*, v1.0 edition.

[16] Open SystemC Comunity. SystemC: Entorno de Desarrollo y Verificación de Sistemas, 2000.

[17] IEEE 1647 Working Group. e Functional Verification Language Working Group (eWG), 2005.

[18] Dhananjay S. Brahme, Steven Cox, Jim Gallo, Mark Glasser, William Grudmann, C. Norris Ip, William Paulsen, John L. Pierce, John Rose, Dean Shea, and Karl Whiting. Creating a C++ Library for Test Bench Authoring. Technical Report CDNL-TR-2000-0820, Cadence Berkeley Labs, August 2000.

[19] Steve Forde, Steve Bishop, and Rammath S. Velu. Streamlining HDL Code Coverage Analysis. *Integrated System Design*, December 1998.

[20] T. Kuhn, T. Oppold, M Witerholer, W. Rosenstiel, Marc Edwards, and Yaron Kashai. A Framework for Object Oriented Hardware Specification, Verification, and Synthesis. *IEEE Design Automation Conference*, pages 18–22, June 2001.

[21] H. Navarro, Juan A. Montiel-Nelson, J. Sosa, and R. Sarmiento. DEMETER: A Novel Framework for Hardware and Software System Specification, Simulation and Verification in C++. *XVI Conference on Design of Circuits and Integrated Systems*, pages 20–23, November 2001.

[22] Matthias Bauer and Wolfgang Ecker. Hw. Sw. Co–Simulation in a VHDL–Based Test Bench Approach. *Proc. of the 34th Conference on Design Automation*, pages 70–75, June 1987.

[23] Peter L. Flake and Simon J. Davidmann. Superlog, a Unified Design Language for System–on–Chip. *IEEE Design Automation Conference, Asia and South Pacific*, pages 25–28, January 2000.

[24] Hong Peng and Sofiene Tahar. Verification and Validation of Complex Digital Designs - A Practical Perspective. *IEEE VTS Tutorial*, June 2001.

[25] Hong Peng and Sofiene Tahar. Compositional Verification of an ATM Bit Error Rate Monitor Circuit. Technical report, Dept. of Electrical & Computer Eng., Concordia University, March 1999.

[26] Hong Peng and Sofiene Tahar. Hardware Modeling and Verification of an ATM Ring MAC Protocol. *The 12th International Conference on Microelectronics*, November 2000.

[27] Bjarne Stroustrup. *The C++ Programming Language*. Addisson-Wesley, 3rd edition, 1998.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

[28] Sowmitri Swami, Arthur Molin, and Burt Counot. Object Oriented Extensions to VHDL. *IEEE Computers*, 28(10), October 1995.

[29] M. Radetzki, W. Putzke-Roming, and W. Nebel. Objective VHDL: The Object–Oriented Approach to Hardware Reuse. *Advanced ins Information Technologies*, The Business Chalenge, 1997.

[30] CynApps. *System Modelling with Cynlib*, v1.1 edition, December 1999.

[31] M. V. Shahdadpuri, J. Sosa, H. Navarro, Juan A. Montiel-Nelson, and R. Sarmiento. A New Framework for Hardware System Verification Based on C++. *XVI Conference on Design of Circuits and Integrated Systems*, pages 20–23, November 2002.

[32] M. V. Shahdadpuri, J. Sosa, H. Navarro, J.A. Montiel-Nelson, and R. Sarmiento. VESTA: A System Level Verification Environment Based on C++. In *Proc. of SPIE of VLSI Circuits and Systems*, May 2003.

[33] J. Sosa, J. A. Montiel-Nelson, H. Navarro, M. V. Shahdadpuri, and R. Sarmiento. System–Level Verification Methodology for Advanced Switch Fabrics. In *Proc. of SPIE of VLSI Circuits and Systems*, May 2003.

[34] Common Switch Interface Consortium. *CSIX–L1: Common Switch Interface Specification–L1*, v1.0 edition, May 2000.

[35] J. Sosa, H. Navarro, M.V. Shahdadpuri, J.A. Montiel-Nelson, and R. Sarmiento. A Chip–Level Verification Methodology for Next–Generation Switch Fabric. In *XVII Conference on Design of Circuits and Integrated Systems*, November 2002.

[36] G.701. Vocabulary of Digital Transmission and Multiplexing, and Pulse Code Modulation (PCM) Terms. Technical report, International Telecommunication Union (ITU), March 1993.

[37] Whenwu Zhu, Yiwei Thomas Hou, Yao Wang, and Ya-Qin Zhang. End–to–End Modeling and Simulation of MPEG–2 Transport Streams over ATM Networks with Jitter. *IEEE Transactions on Circuits and Systems for Video Technology*, 8(1):9–12, February 1998.

[38] Yishay Mansour and Boaz Patt-Shamir. Jitter Control in QoS Networks. *IEEE Transactions on Networking*, 9(4):492–502, August 2001.

[39] IEEE 1364 Verilog Standard. *Verilog PLI 1.0*, November 2000.

[40] Stuart Sutherland. *The Verilog PLI Handbook*. Kluwer Academic Publishers, 1999.

[41] Torbjorn Franlund. The GNU Multiple Precision Arithmetic Library. *Free Software Foundation*, 2001.

[42] J. Sosa, J. A. Montiel-Nelson, H. Navarro, José C. García, and R. Sarmiento. *Circuit Path Coverage using Genetic Algorithms*, pages 51–52. Evolutionary Methods for Design optimization and Control Applications to Industrial and Societal Problems. International Center for Numerical Methods in Engineering, 2003.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

[43] H. Navarro, J. Sosa, J.A. Montiel-Nelson, and R. Sarmiento. A Real Time VLSI Bidimensional Pipelined Architecture for Image Processing. In *XVII Conference on Design of Circuits and Integrated Systems*, November 2002.

[44] Vitesse Semiconductor Corporation. Gigastream Chip Set VSC872 and VSC882. Vitesse Semiconductor Corporation, 2001.

[45] Tracy Larrabee. Test Pattern Generation using Boolean Satisfiability. *IEEE trans. on Computer–Aided Design of Integrated Circuits and Systems*, 11(1):4–15, January 1992.

[46] M. Kantrowitz and Lisa M. Noack. I'm Done Simulating; Now What? Verification Coverage Analysis and Correctness Checking of the DECchip 21164 Alpha Microprocessor. *Proc. of Design Automation Conference*, pages 325–330, June 1996.

[47] Randal E. Bryant. Graph–Based Algorithms for Boolean Function Manipulation. *IEEE transaction on Computers*, C-35(8):677–691, August 1986.

[48] M. Abramovici, M. Breumer, and A. Friedman. *Digital Systems Testing and Testable Design.* Wiley–IEEE Press, January 1993.

[49] E. Goldberg and Y. Novikov. BerkMin: a Fast and Robust Sat–Solver. *Proc. of Design, Automation and Test in Europe*, pages 149–149, March 2002.

[50] Tom Kirkland and M. Ray Mercer. Algorithms for Automatic Test Pattern Generation. *IEEE Design and Test of Computers*, June 1988.

[51] S. Yang. *Logic Synthesis and Optimization Benchmarks User Guide version 3.0.* Microelectronics Center of North Carolina, 1991.

[52] E.M. Sentovich, K.J. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P.R. Stephan, Robert K. Brayton, and Alberto L. Sangiovanni-Vincentelli. SIS: A System for Sequential Circuit Synthesis. Technical Report UCB/ERL M92/41, EECS Department, University of California, Berkeley, 1992.

[53] R. K. Brayton, G. D. Hachtel, A. Sangiovanni-Vincentelli, F. Somenzi, A. Aziz, S. T. Cheng, S. Edwards, S. Khatri, Y. Kukimoto, A. Pardo, S. Qadeer, R. K. Ranjan, S. Sarwary, T. R. Shiple, G. Swamy, and T. Villa. VIS: a System for Verification and Synthesis. In Rajeev Alur and Thomas A. Henzinger, editors, *Proceedings of the Eighth International Conference on Computer Aided Verification CAV*, volume 1102, pages 428–432, New Brunswick, NJ, USA, 1996. Springer Verlag.

[54] Dingzhu Zu, Hun Gu, and Panos M. Pardalos. *Satisfiability Problem: Theory and Applications.* American Mathematical Society, 1997.

[55] F. Fallah, S. Devadas, and K. Keutzer. Functional Vector Generation for HDL Models using Linear Programming and 3–Satisfiability. *Proc. on Design Automation Conference*, June 1998.

[56] Zeng Zhihong, P. Kalla, and M. Ciesielski. LPSAT: A Unified Approach to RTL Satisfiability. *Proc. on Design, Automation and Test in Europe*, pages 398–402, March 2001.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

[57] J. A. Montiel-Nelson, H. Navarro, J. Sosa, and José C. García. Theory and Applications of Satisfiability Testing. In *Evolutionary and Deterministic Methods for Design, Optimization and Control with Applications to Industrial and Societal Problems*, November 2005.

[58] Srimat T. Chakradhar, Vishwani D. Agrawal, and Michael L. Bushnell. Automatic Test Generation using Quadratic 0–1 Programming. In *Proceedings of the 27th ACM/IEEE conference on Design automation*, pages 654–659, New York, NY, USA, 1990. ACM Press.

[59] André; Baresel, David Binkley, Mark Harman, and Bogdan Korel. Evolutionary Testing in the Presence of Loop-assigned Flags: A Testability Transformation Approach. In *ISSTA '04: Proceedings of the 2004 ACM SIGSOFT International Symposium on Software Testing and Analysis*, pages 108–118, New York, NY, USA, 2004. ACM Press.

[60] J. Sosa, J. A. Montiel-Nelson, H. Navarro, and José C. García. Functional Vector Generation for Maximum Data Path Coverage using Mixed Integer Linear Programming. In *XVIII Conference on Design of Circuits and Integrated Systems*, November 2005.

[61] K.S. Lowe and P.G. Gulak. A Joint Gate Sizing and Buffer Insertion Method of Optimizing Delay and Power in CMOS and BiCMOS Combinational Logic. *IEEE Trans. Comput.–Aided Des. Integr. Circuits Syst.*, 17(5):419–434, 1998.

[62] C. Chen, A. Srivastava, and M. Sarrafzadeh. On Gate Level Power Optimization using Dual–Supply Voltages. *IEEE Trans. Comput.–Aided Des. Integr. Circuits Syst.*, 9(5):616–629, 2001.

[63] P.Y. Calland, A. Mignotte, O. Peyran, Y. Robert, and F. Vivien. Retiming DAG's. *IEEE Trans. Comput.–Aided Des. Integr. Circuits Syst.*, 17(12):1319–1324, 1998.

[64] N. Sherwani. *Algorithms for VLSI Physical Design Automation*. Kluwer Academic Publishers, Boston, 1993.

[65] C.E. Leiserson and J.B. Saxe. Retiming Synchronous Circuitry. *Algorithmica*, 6:5–35, 1991.

[66] D. Becker, R.K. Singh, and S.G. Tell. An Engineering Environment for Hardware/Software Co–Simulation. *Design Automation Conference*, pages 129–134, June 1992.

[67] Randal E. Bryant. Symbolic Simulation Techniques and Applications. In *Proceedings of the 27th ACM/IEEE conference on Design automation*, pages 517–521, New York, NY, USA, 1990. ACM Press.

[68] H. Navarro, J. A. Montiel-Nelson, J. Sosa, and José C. García. Solving the State Justification Problem using MILP for RTL Specifications. In *XIX Conference on Design of Circuits and Integrated Systems*, November 2004.

[69] Thomas Bäck. *A User's Guide to GENEsYs*. University of Dortmund, Department of Computer Science, Systems Analisys Research Group, Dortmund, v1.0 edition, 1992.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

[70] R.K. Brayton, R. Rudell, A.L. Sangiovanni-Vincentelli, and A. Wang. MIS: A Multiple Level Logic Optimization System. *IEEE Trans. Comput.–Aided Des. Integr. Circuits Syst.*, 6(11):1062–1081, 1987.

[71] J. Sosa, J. A. Montiel-Nelson, H. Navarro, and José C. García. Functional Vector Generation for Maximum Path Coverage using Evolutionary Programming. In *XVIII Conference on Design of Circuits and Integrated Systems*, November 2003.

[72] J. Sosa, J. A. Montiel-Nelson, H. Navarro, and José C. García. Stimuli Sequence Generation for Verification and Validation of Maximum Power Consumption using Evolutionary Programming. In *Evolutionary and Deterministic Methods for Design, Optimization and Control with Applications to Industrial and Societal Problems 2005*, September 2005.

[73] J. Sosa, J. A. Montiel-Nelson, H. Navarro, V. de Armas, and R. Sarmiento. Functional Vector Generation for Combinational Circuits Based on Data Path Coverage Metric and Mixed Integer Linear Programming. In *5th International Society for Quality Electronic Design*, November 2004.

[74] Christian Piguet. *Low–Power Electronics Design*. CRC Press, July 2004.

[75] M. Nemani and F. Najm. Towards a High–Level Power Estimation Capability. *IEEE Trans. on Computer–Aided Design*, 15(6):588–598, 1996.

[76] D. Marculescu, R. Marculescu, and M. Pedram. Information Theoretic Measures for Power Analysis. *IEEE Trans. on Computer–Aided Design*, 15(6):599–609, June 1996.

[77] M. Nermani and F. Najm. High–Level Area and Power Estimation for VLSI Circuits. *IEEE Trans. on Computer–Aided Design*, 18(6):697–713, June 1999.

[78] A. J. de Geus. Logic Synthesis Speeds ASIC Design. *IEEE Spectrum*, 26(8):27–31, August 1989.

[79] B. Chen and I. Nedelchev. Power Compiler: A Gate–Level Power Optimization and Synthesis System. *Computer Design: VLSI in Computers and Processors*, pages 74–79, October 1997.

[80] Xiaoyong Tang, Hai Zhou, and Prith Banerjee. Leakage Power Optimization With Dual–Vth Library in High–Level Synthesis. *42nd IEEE Design Automation Conference*, pages 202–207, June 2005.

[81] F. Najm. Transition Density: A New Measure of Activity in Digital Circuits. *IEEE Trans. Computer–Aided Design*, 12(2):310–323, February 1993.

[82] Z. Chen, K. Roy, and T. Chuo. Efficient Statistical Approach to Estimate Power Considering Uncertain Properties of Primary Inputs. *IEEE Tran. on Very Large Scale Integration (VLSI) Systems*, 6(3):484–492, February 1993.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

[83] S. Gupta and F. Najm. Power Modeling for High–Level Power Estimation. *IEEE Tran. on Very Large Scale Integration (VLSI) Systems*, 8(1):18–29, February 2000.

[84] C. Wang and K. Roy. Maximum Power Estimation for CMOS Circuits using Deterministic and Statistical Approaches. *IEEE Tran. on Very Large Scale Integration (VLSI) Systems*, 6(1):130–140, February 1999.

[85] N. Evmorfopoulos, G. Stamoulis, and J. Avaritsiotis. A Monte Carlo Approach for Maximum Power Estimation Based on Extreme Value Theory. *IEEE Tran. on Computer-Aided Design of Integrated Circuits and Systems*, 21(4):415–432, April 2002.

[86] R. Radjassamy and J. Carothers. Faster Power Estimation of CMOS Designs using Vector Compaction — a Fractal Approach. *IEEE Tran. on Systems, Man and Cybernetics, Part B*, 33(3), June 2003.

[87] K. Roy and S. Prasad. *Low–Power CMOS VLSI Circuit Design.* John Willey and Sons, Inc., New York, 1st edition, February 2000.

[88] J.A. Montiel-Nelson, J. Sosa, H. Navarro, R. Sarmiento, and A. Nuñez. Efficient Method to Obtain the Entire Active Area against Circuit Delay Time Trade–Off Curve in Gate Sizing. In *IEE Proc. Circuits, Devices & Systems*, volume 152 (2), April 2005.

[89] J.A. Montiel-Nelson, V. De Armas, and A. Sarmiento, R. and Nuñez. A Cell and Macrocell Compiler for GaAs VLSI Full–Custom Design. *Proc. Design Automation and Test in Europe, Conf. and Exhibition*, pages 947–948, feb 1998.

[90] A. Ghosh, S. Devadas, K. Keutzer, and J. White. Estimation on Average Switching Activity in Combinational and Sequential Circuits. *Proc. 29th ACM/IEEE Design Automation Conference*, pages 243–259, 1992.

[91] M.R.C.M. Berkelaar, P.H.W. Buurman, and J.A.G. Jess. Computing the Entire Active Area/Power Consumption versus Delay Tradeoff Curve for Gate Sizing with a Piecewise Linear Simulator. *IEEE Trans. Comput.–Aided Des. Integr. Circuits Syst.*, 15(11):1424–1434, 1996.

[92] G. Chen, H. Onodera, and K. Tamaru. An Iterative Gate Sizing Approach with Accurate Delay Evaluation. *Proc. ACM/IEEE Design Automation Conf.*, pages 422–427, November 1995.

[93] L.A. Glasser and L.P.J. Hoyte. Delay and Power Optimization in VLSI Circuits. *Proc. 21st ACM/IEEE Design Automation Conf.*, pages 529–535, jun 1984.

[94] J. Shyu, A. Sangiovanni-Vincentelli, J.P. Fishburn, and A.E. Dunlop. Optimization–based Transistor Sizing. *IEEE J. Solid–State Circuits*, 23(2):400–409, 1988.

[95] C.M. Lee and H. Soukup. An Algorithm for CMOS Timing and Area Optimization. *IEEE J. Solid–State Circuits*, 19(5):781–787, 1984.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

[96] J.P. Fishburn and A.E. Dunlop. TILOS: A Posynomial Programming Approach to Transistor Sizing. *Proc. ACM/IEEE Int. Conf. Computer–Aided Design*, pages 326–328, November 1985.

[97] M.R.C.M. Berkelaar and J.A.G. Jess. Gate Sizing in MOS Digital Circuits with Linear Programming. *Proc. European Design Automation Conf.*, pages 217–221, March 1990.

[98] M.A. Cirit. Transistor Sizing in CMOS Circuits. *Proc. 24th ACM/IEEE Design Automation Conf.*, pages 121–124, June 1987.

[99] K.S. Hedlund. AESOP: A Tool for Automated Transistor Sizing. *Proc. 24th ACM/IEEE Design Automation Conf.*, pages 114–120, May 1987.

[100] D. Marple. Transistor Size Optimization in the TAILOR Layout System. *Proc. 26th ACM/IEEE Design Automation Conf.*, pages 43–48, June 1989.

[101] S.S. Sapatnekar, V.B. Rao, P.M. Vaidya, and S.M. Kang. An Exact Solution to the Transistor Sizing Problem for CMOS Circuits using Convex Optimization. *IEEE Trans. Comput.–Aided Des. Integr. Circuits Syst.*, 12(11):1621–1634, 1993.

[102] J. Cong and C.K. Koh. Simultaneous Driver and Wire Sizing for Performance and Power Optimization. *Proc. ACM/IEEE Int. Conf. Computer–Aided Design*, pages 206–212, November 1994.

[103] C.P. Chen, C.C.N. Chu, and D.F. Wong. Fast and Exact Simultaneous Gate and Wire Sizing by Lagrangian Relaxation. *IEEE Trans. Comput.–Aided Des. Integr. Circuits Syst.*, 18(7):1014–1025, 1999.

[104] A.R. Conn, P.K. Coulman, R.A. Haring, G.L. Morrill, C. Visweswariah, and Chai Wah Wu. JiffyTune: Circuit Optimization using Time–Domain Sensitivities. *IEEE Trans. Comput.–Aided Des. Integr. Circuits Syst.*, 17(12):1292–1309, 1998.

[105] V. Sundararajan, S.S. Sapatnekar, and K.K. Parhi. Fast and Exact Transistor Sizing Based on Iterative Relaxation. *IEEE Trans. Comput.–Aided Des. Integr. Circuits Syst.*, 21(5):568–581, 2002.

[106] H.C. Chen and D.H.C. Du. Path Sensitization in Critical Path Problem. *IEEE Trans. Comput.–Aided Des. Integr. Circuits Syst.*, 12(2):196–207, 1993.

[107] H.R. Lin and T.T. Hwang. On Determining Sensitization Criterion in an Iterative Gate Sizing Process. *IEEE Trans. Comput.–Aided Des. Integr. Circuits Syst.*, 18(2):231–238, 1999.

[108] H.R. Lin and T.T. Hwang. Dynamical Identification of Critical Paths for Iterative Gate Sizing. *Proc. ACM/IEEE Int. Conf. Computer–Aided Design*, pages 481–484, November 1994.

[109] H.C. Chen, D.H.C. Du, and L.R. Liu. Critical Path Selection for Performance Optimization. *IEEE Trans. Comput.–Aided Des. Integr. Circuits Syst.*, 12(2):185–195, 1993.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

[110] S.T. Huang, T.M. Parng, and J.M. Shyu. A New Method of Identifying Critical Paths for Performance Optimization. *Proc. European Design Automation Conf.*, pages 455–459, February 1993.

[111] F. Chen-Lian, , and J. Wen-Ben. Timing Optimization by Gate Resizing and Critical Path Identification. *IEEE Trans. Comput.–Aided Des. Integr. Circuits Syst.*, 14(2):201–217, 1995.

[112] J. Sosa, H. Navarro, V. de Armas, J. A. Montiel-Nelson, and R. Sarmiento. *A Novel Methodology for Efficient Computation of the Area/Power Consumption versus Delay Tradeoff Curve in Circuit Optimization Problems*, pages 190–195. Evolutionary Methods for Design Optimization and Control. International Center for Numerical Methods in Engineering, 2002.

[113] S. Sapatnekar and Weitong Chuang. Power vs. Delay in Gate Sizing: Conflicting Objectives? *IEEE/ACM International Conference on Computer-Aided Design*, pages 463–466, November 1995.

[114] ITRS. International Technology Roadmap for Semiconductors. http://public.itrs.net, 2001.

[115] P. Maurine, M. Rezzoug and N. Azemard, and D. Auvergne. Transition Time Modeling in Deep Submicron CMOS. *IEEE Trans. on Computer–Aided Design of Integrated Circuits and Systems*, 21(11):1352–1363, November 2002.

[116] Kaushik Roy and Sharat. C. Prasat. Circuit Activity Based Logic Synthesis for Low Power Reliable Operations. *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, 1(4):503–513, December 1993.

[117] P. Maurine, M. Rezzoug, and D. Auvergne. Output Transition Time Modeling of CMOS Structures. *IEEE International Symposium on Circuits and Systems*, 5:363–366, May 2001.

[118] F. Najm. Low–Pass Filter for Computing the Transition Density in Digital Circuits. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 13(9):1123–1131, September 1994.

[119] Tan-Li Chou and Kaushik Roy. Estimation of Activity for Static and Domino CMOS Circuits Considering Signal Correlations and Simultaneous Switching. *IEEE Trans. on Computer–Aided Design of Integrated Circuits and Systems*, 15(10):1257–1265, October 1996.

[120] Kuy-Won Choi and Abhijit Chatterjee. UDSM (Ultra–Deep Sub–Micron)–Aware Post–Layout Power Optimization for Ultra Low–Power CMOS VLSI. *International Symposium on Low Power Electronics and design*, pages 72–77, 2003.

[121] K. Roy, S. Mukhopadhyay, and H. Mahmoodi-Meimand. Leakage Current Mechanisms and Leakage Reduction Techniques in Deep-submicrometer CMOS Circuits. *Proceedings of the IEEE*, 91(2):305–327, feb 2003.

[122] K.S. Khouri and N.K. Jha. Leakage Power Analysis and Reduction During Behavioral Synthesis. *IEEE Transactions on Very Large Scale Integration (VLSI)*, 10(6):876–885, December 2002.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

Índice alfabético

$(\check{T}_i, \check{T}_{i+1}]$, 136
$C_{tradeoff}$, 134
$I(path_{cp})$, 122, 131
PO, 122
$P_{gate,d}$, 156
RV_{paths}, 79
T_{max}, 156
T_n, 131
V_{dd}, 158
V_{th}, 158
$\Omega\ (n^2)$, 145
$\vec{G}$, 136
$\tau(v) = \tau_v$, 75
$merge(n, g)$, 148
$path_i$, 70
$path_{cp}$, 122
$path_{v_i,v_j}$, 76
$patterns(cond(v_j))$, 70
$v(path_i)$, 70
y_d, 99
y_p, 99

ACK, 49
actividad de conmutación, 109, 116
ajuste lineal a tramos, 139
Algoritmos
    combinados, 129
    genéticos, 91, 178
alineamiento del tipo de dato, 59
AMD K6, 60
Análisis de Prestaciones, 9
Analizador de Prestaciones, 126
ancho
    del bus, 59
    equivalente, 166
Aproximaciones
    lineales, 126
    medias, 108
arco dirigido, 75
Arquitectura, 6, 7
assertion rules, 26
ATM, 6
ATPG, 17, 24, 74

búsqueda exhaustiva, 93
back–annotation, 11
bancos de prueba, 35, 43
basic sanity tests, 26
BEM, 59
bool_t, 61
Bottleneck, 3

C++, 34
C/C++, 18
Cálculo de la Cobertura, 90
CAD, 3
Cadence, 13
calidad de servicio, 5
camino, 75
Capa de
    Comunicación, 39, 45–47, 50
    Referencia, 33, 41
    Verificación, 40, 46, 50
Capa de Verificación, 49
capacidad
    total, 165
capacidad de
    cableado, 138
    interconexionado, 164
carga capacitiva, 162
casos
    avanzados, 38
    básicos, 38
    generales, 40
    intencionales, 38
check_proposal, 113
CI, 3
Ciclo de Diseño, 2, 6
Circuito con MILP, 98
circuito integrado, 3
circuitos multinivel, 93
Clusters, 9
clusters, 20
CMOS 0'18 μm, 143, 196
Co–Design Automation, 37

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

[123] T. Sakurai. LSI Design Toward 2010 and Low–Power Technology. *Proceedings of the 1999 International Conference on VLSI and CAD*, pages 325–334, 1999.

[124] D. Eckerbert and P. Larsson-Edefors. Cycle–True Leakage Current Modeling for CMOS Gates. *IEEE International Symposium on Circuits and Systems*, 5:507–510, May 2001.

[125] H. Veendrick. Short–Circuit Dissipation of Static CMOS Circuitry and its Impact on the Design of Buffer Circuits. *IEEE J. Solid–State Circuits*, SC-19:468–473, August 1984.

[126] N. Hendestierna and K. O. Jeppson. CMOS Circuit Speed and Buffer Optimization. *IEEE Trans. Computer–Aided Design*, CAD–6:270–281, mar 1987.

[127] T. Sakurai and R. Newton. Alpha–Power Law MOSFET Model and its Applications to CMOS Inverter Delay and Other Formulas. *IEEE J. Solid–State Circuits*, 25:584–594, April 1990.

[128] S. Turgis and D. Auvergne. A Novel Macromodel for Power Estimation in CMOS Structures. *IEEE Trans. Computer–Aided Design*, 17:1090–1098, November 1998.

[129] A. Hamoui and N. Rumin. An Analitical Model for Current, Delay and Power Analysis of Submicron CMOS Logic Circuits. *IEEE Trans. Circuits Syst.*, 47:999–1007, October 2000.

[130] T. Sakurai and R. Newton. A Simple MOSFET Model for Circuit Analysis. *IEEE Trans. Electron. Devices*, 38:887–894, apr 1991.

[131] S. Nikolaidis, A. Chantzigeorgiou, and Kyriakis-Bitzaros. Delay and Power Estimation for a CMOS Inverter Driving RC Interconnect Loads. *Proc. IEEE Int. Symp. Circuits Sustems*, VI:368–371, May 1998.

[132] K. Nose and T. Sakurai. Analisys and Future Trend of Short–Circuit Power. *IEEE Trans. Computer–Aided Design*, 19:1023–1030, September 2000.

[133] M. Shoji. FET Scaling in Domino CMOS Gates. *IEEE J. Solid–State Circuits*, SC–20:1067–1071, October 1985.

[134] A. Nabavi-Lishi and N. C. Rumin. Inverter Models of CMOS Gates for Supply Current and Delay Evaluation. *IEEE Trans. Computer–Aided Design*, 13:1271–1279, October 1994.

[135] T. Sakurai and A. R. Newton. Delay Analysis of Series–Connected MOSFET Circuits. *IEEE J. Solid–State Circuits*, 26:122–131, feb 1991.

[136] F. Najm. A Survey of Power Estimation Techniques in VLSI Circuits. *IEEE Trans. VLSI Syst.*, 2:446–455, dec 1994.

[137] H. Soeleman, K. Roy, and T.-L. Chou. Estimating Circuit Activity in Combiational CMOS Digital Circuits. *IEEE Des. Test Comput.*, pages 112–119, April 2000.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

[138] A. Shen, A. Ghosh, and S. Devadas. On Average Power Dissipation and Random Pattern Testability of CMOS Combinational Logic Networks. *Proc. IEEE Int. Conf. Computer-Aided Design*, pages 402–407, 1992.

[139] J. Sosa, J. A. Montiel-Nelson, H. Navarro, and José C. García. Minimum Power Consumption Optimization using Genetic Algorithms. In *Evolutionary and Deterministic Methods for Design, Optimization and Control with Applications to Industrial and Societal Problems 2005*, September 2005.

[140] Inc. CPLEX Optimization. *CPLEX, ILOG CPLEX 7.0: User's Manual.* Incline Village, NV, June 2000.

[141] Adnan Kabbani, Dhamin Al-Khalili, and Asim J. Al-Khalili. Technology–Portable Analytical Model for DSM CMOS Inverter Transition–Time Estimation. *IEEE Trans. on Computer–aided Design of Integrated Circuits and Systems*, 22(9):1177–1187, September 2003.

[142] AMS. AustriaMicroSystems AG. Empresa Fabricante de Tecnologias Basadas en Silicio.

[143] TSMC. Taiwan Semiconductor Manufacturing Company Ltd. Empresa Fabricante de Tecnologías, Entre Otras Basadas en Silicio.

[144] UMC. United Microelectronics Corporation. Empresa Fabricante de Diversas Tecnologías Basadas en Silicio.

[145] IBM-Burlington. *Mosis IBM 0.13 Wafer Acceptance Tests.* Mosis, t59m edition, December 2005.

[146] TSMC. *Mosis TSMC 0.18 Wafer Acceptance Test.* Mosis, snc018 edition, October 2005.

[147] TSMC. *Mosis TSMC 0.25 Wafer Acceptance Test.* Mosis, smc025 edition, October 2005.

[148] TSMC. *Mosis TSMC 0.35 Wafer Acceptance Test.* Mosis, smc035 edition, November 2005.

[149] AustriaMicroSystems. *AMS* 0,35 μm *CMOS C35 Process Parameters, v3.0*, September 2004.

[150] AustriaMicroSystems. *AMS* 0,60 μm *CMOS CUP Process Parameters, revision B*, October 1998.

[151] AustriaMicroSystems. *AMS* 0,8 μm *CMOS CXQ Process Parameters, revision C*, August 1999.

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

cobertura
Basada en Rutas, 76
de rutas, 79, 101
Codificación
del Individuo, 91
RTL, 9
Complejidad de la Optimización, 140
complex_t, 61
Comprobación Funcional, 36
condiciones
contradictorias, 88
de contorno, 88
de entrada, 87
Conjunto Ruta Crítica, 130, 145
Cono Lógico de
Entrada, 170
Salida, 170
Consumo
de Potencia, 160
de una Puerta, 164
Dinámico, 109, 162
Estático, 161, 169
convergencia, 95
de propagaciones, 101
corner cases, 26
Corriente
de fugas, 160
subumbral, 161
cortocircuito, 109
Coste Incremental, 179
CoWare, 37
CRC, 46
CRQ, 49
Cruzamiento Multipunto, 116
CSIX, 6
curva de aprendizaje, 17, 64
Curva de Prestaciones, 132, 134, 139, 171, 173, 174, 188
curva de prestaciones, 124, 128, 136, 151, 177, 183, 188, 202
Curvas de Prestaciones, 146
CynApps, 37

datos de usuario, 59
datos simples, 59
DBV, 20, 23, 39
Densidad de Transición, 166, 167
desarrollo de Internet, 5
descripción estructural, 100
descripciones mixtas, 19
Dimensionado
de puerta, 126, 176, 188
de transistores, 126
directed edges, 75
Diseño Físico, 11
disipación estática, 160
Distribución, 36
dominio de definición, 129, 135, 136, 151, 176
DUT, 20

efecto de filtrado, 167
ejercitación, 75
de rutas, 82
Encapsulación, 36
entorno de verificación, 35
entrada
inmediata, 75
primaria, 77
Equivalence Checking, 15
escala de integración, 157
espúreos, 159
Espacio de Diseño, 133, 149, 173
Especificación de las Interfaces, 9
especificaciones mixtas, 22
esquinas críticas, 38
estímulos de entrada, 115
estado no activo, 160
estimador de transiciones, 109
estructuras de datos, 58
Exploración
de la cobertura, 91
del Espacio de diseño, 27, 125, 169
extracción de las prestaciones, 13

factor de
ganancia, 164
velocidad, 138, 169, 179
FastEthernet, 6
Ficheros de
ACK, 53
Comprobación, 40, 54
CRQ, 53
Entrada, 54
Estímulos, 40
Estímulos y Comprobación, 41
FIFO, 32, 47
flexibilidad, 65
float_t, 61
formulación

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

no lineal, 127
posinomial, 126
frecuencia de conmutación, 159
fuerza bruta, 93, 126
full–custom, 136, 172, 176
Función
de coste, 91, 92, 109, 115, 178
Inversa, 85
multiobjetivo, 139
no continua, 172

generación de vectores, 70
generador de estímulos, 24
GENEsYs, 93
GENEsYs v1.0, 116
GigabitEthernet, 6
glitches, 168
GLPK, 70, 101
GMP, 32, 59
GNU, 32
Golden Reference, 7

HDL, 18
Herramientas, 34
Síntesis física, 13
Síntesis lógica, 14
tradicionales, 22
Verificación, 3
Heurística, 152

IA–64, 60
individuo, 91, 113
influencia del modelo, 159
int_t, 61
Intel Pentium, 60
Interconexionado, 11, 168
Interfaz, 43
Control, 43
de CPU, 43, 50, 55
Paralela, 45
Paralelo, 44, 54
Serie, 43, 47
intervalo de optimización, 131
Inversores en Cadena, 165

Jitter, 57

Límite Inferior del Retardo, 131
layout, 12, 133, 172
leakage current, 161
LEM, 59
lenguaje HDL, 34
lenguaje–e, 18, 34
Librería
Convexa, 133, 172
Tecnológica, 11
LIFO, 32, 62
Linux Debian, 105, 116
lista enlazada, 62

máxima cobertura, 92, 95, 101
métodos probabilísticos, 108, 116
Métrica
de cobertura, 17, 71
objetiva, 95
Módulos, 9, 20
múltiples vectores, 88
Mínima
Área Activa, 128, 132–134
Potencia de Consumo, 173, 177
Mínimo Minimorum, 125, 181
Mapeado Tecnológico, 11, 124
matriz de conmutación, 6
MCNC'91, 93, 101, 116, 143
MILP, 70, 74, 99, 101
MISII, 93, 101, 116, 143
Model–Checking, 10, 15, 23
Modelo
Automático, 40, 45
de Elmore, 127
de Propagación de Puerta, 82, 91
de Puerta, 168
de Puerta Tradicional, 98
de Retardo y Área Activa, 137
modo
de operación, 59
independiente, 52
no–independiente, 52
mpf_t, 60
mpz_t, 60
multihilo, 43
Mutación
Estándar, 93, 116
Multipunto, 93
Muti–thread, 43

número de variables, 144
NC–VHDL, 59
net–list, 11, 133, 136, 172, 176
nivel de
confianza, 77
decisión, 115

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

No Lineal, 152
no lineal, 183

OC–192, 2
OC–48, 2
OC–768, 2
offspring, 93
OLYMPO, 116
OpenVera, 18

parásitos, 12
pares ordenados, 75
Particionado, 7, 149
Patrón
    de Entrada, 87, 89
    de Referencia, 13
Pentium III, 144
Pentium IV, 105, 116
PLI, 18
Población, 93
    Genética, 91, 113
Potencia
    de Carga y Descarga, 109
    de Consumo, 107, 115, 183
    de Consumo–Retardo, 157, 169, 188, 202
    de Corto Circuito, 169
    Estática, 109
    Total, 164
Potencial Tecnológico, 14
prestaciones área–retardo, 125
probabilidad de equilibrio, 167
problema de dimensionado, 158
Proceso
    de Actualización, 132
    de Fabricación, 3
    Tecnológico, 165
Programación
    Evolutiva, 93
    Lineal, 126, 127, 139, 152
    No Lineal, 127
PROLOG, 105
Propagación de Rutas, 97
Prototipo, 12
pseudo–aleatorio, 24
Puerta
    AND, 82, 110
    INV, 85, 110
    OR, 84, 110
pull–down, 146
pull–up, 146
Punto de Arranque, 177, 181
puntos no inferiores, 132, 134, 171, 173

QoS, 2, 5
Quality of Service, 5

red invariante, 136
redes de comunicaciones, 5
Refinado Sucesivo, 181
registro de verificación, 24, 26
relajación iterativa, 127
representación cartesiana, 61
Retardo
    de Propagación, 168
    de Puerta, 137
    No Lineal, 139
    Total, 140
reusabilidad, 65
Reutilización, 36
RTL, 2
RTPG, 17
Ruta, 75
    Crítica, 169
    Critica, 130
    Dependiente, 130, 170
    Ejercitable, 87
    Ejercitada, 100
    Falsa, 176
    Independiente, 130, 134
ruta ejercitada, 100

Síntesis, 10
    Lógica, 11
Salida
    Inmediata, 75
    Primaria, 77
SAT solvers, 97
Satisfabilidad, 97
secuencia de estímulos, 108
setsim_vector, 113
short–channel, 162
Simulador
    Lineal, 127
simulador
    basados en slot–time, 42
    guiado por eventos, 42
    HDL, 19, 34
Sincronización, 56
Sistema, 9
    de conmutación, 20

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007

de Ficheros, 46, 52
sistema, 20
slack–time, 151
slot–time, 42
Specman, 13, 18
speed factor, 138
subconjunto ruta crítica, 198
submicra profunda, 159
Sun–Fire 280R, 93, 101
SunOS v5.8, 93, 101
Superlog, 35
SuperSPARC, 60
Symbolic Simulation, 16
SystemC, 18

t, 10
T–Builder, 18
técnicas evolutivas, 24
tamaño del problema, 140
Technology Roadmap, 3, 14
tensión umbral, 158, 160
Teoría de Grafos, 170
Test, 12
testbench, 13
Theorem Proving, 15
threads, 36
throughput, 7
Tiempo de Transición, 159, 165, 169
time–to–market, 3, 12, 20, 33, 38, 71
Tipo
Booleano, 61
de C++, 58
Entero Con Signo, 61
Entero Sin Signo, 61
Imaginario, 61
Real, 61
tour, 75
transceptor, 6
Transiciones, 75, 82
Identificadas, 111
Indeseadas, 168
transistores de carga, 164
TSMC, 138

uint_t, 61
UltraSPARC III, 93
update_previous, 113

vértices, 75
variables *trade–off*, 7
VCD, 25
vector, 80
de entrada, 92, 111
de test, 74
previo, 111
Verificación
Automática, 10
Basada en Reglas, 25
Basada en Simulación, 14
código de, 22
casos críticos, 12
Clusters, 10
de alto nivel, 35
herramientas, 3
lenguaje, 19
Métodos Formales, 15
Módulos, 9
Modelos de, 38
Prestaciones, 10
síntesis, 12
Sistemas integrados, 2
tareas de, 33
Verification Report, 26
Verilog, 35
NC, 37, 59
XL, 59
verilog, 9
RTL, 9
Verisity, 13
VHDL, 19, 35
VOQ, 6
VSC
872, 63
882, 63

© Del documento, los autores. Digitalización realizada por ULPGC. Biblioteca Universitaria, 2007