UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA

ESCUELA TÉCNICA SUPERIOR DE INGENIEROS DE TELECOMUNICACIÓN

PROYECTO FIN DE CARRERA

IMPLEMENTACIÓN DE LA 2-D DCT EN TECNOLOGÍA GaAs PARA LA COMPRESIÓN DE IMÁGENES EN TIEMPO REAL

CARLOS J. PULIDO SANTANA

Las Palmas de Gran Canaria, Octubre de 1995

Título del Proyecto Fin de Carrera:

IMPLEMENTACIÓN DE LA 2-D DCT EN TECNOLOGÍA GaAs PARA LA COMPRESIÓN DE IMÁGENES EN TIEMPO REAL

UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA
ESCUELA TÉCNICA SUPERIOR DE INGENIEROS DE TELECOMUNICACIÓN

Proyecto Fin de Carrera

Implementación de la 2–D DCT en Tecnología GaAs para la Compresión de Imágenes en Tiempo Real

Autor: D. CARLOS J. PULIDO SANTANA
Tutor: DR. D. ROBERTO SARMIENTO RODRÍGUEZ
Co–tutor: DR. D. KAMRAN ESHRAGHIAN

Las Palmas de Gran Canaria, Octubre de 1995.

UNIVERSIDAD DE LAS PALMAS DE GRAN CANARIA
ESCUELA TÉCNICA SUPERIOR DE INGENIEROS DE TELECOMUNICACIÓN

Proyecto Fin de Carrera

Implementación de la 2–D DCT en Tecnología GaAs para la Compresión de Imágenes en Tiempo Real

Autor: D. CARLOS J. PULIDO SANTANA
Tutor: DR. D. ROBERTO SARMIENTO RODRÍGUEZ
Co–tutor: DR. D. KAMRAN ESHRAGHIAN

TRIBUNAL

Presidente: D. ANTONIO NÚÑEZ ORDÓÑEZ
Vocal: D. MIGUEL A. FERRER BALLESTER
Secretario: D. VALENTÍN DE ARMAS SOSA

Calificación: MATRÍCULA DE HONOR (10)

Las Palmas de Gran Canaria, a Octubre de 1995.

Universidad de Las Palmas de Gran Canaria
Escuela Técnica Superior de Ingenieros de Telecomunicación

Proyecto Fin de Carrera

Implementación de la 2–D DCT en Tecnología GaAs para la Compresión de Imágenes en Tiempo Real

Autor: D. Carlos J. Pulido Santana
Tutor: Dr. D. Roberto Sarmiento Rodríguez
Co–tutor: Dr. D. Kamran Eshraghian

El Director

El Proyectando

Las Palmas de Gran Canaria, Octubre de 1995.

A mis padres y hermanos.
La familia lógica perfecta.

Agradecimientos

El documento que tienen ahora en sus manos es fruto de días, tardes y noches interminables de trabajo que no habría podido soportar sin el apoyo de un grupo de personas a las que siempre les estaré agradecido.

Primero quiero agradecer a mi familia su apoyo, comprensión y ayuda brindada durante todos estos años. A ellos les debo y dedico este proyecto.

Quiero agradecer al Dr. Kamran Eshraghian la oportunidad que me ha dado de trabajar con un científico de su nivel. Los conocimientos, que de una forma tan sencilla me ha transmitido, han sido básicos para la realización de este proyecto.

Agradecer, como no, al Dr. Roberto Sarmiento, que me brindó la oportunidad de entrar en su grupo de trabajo y de conocer al Dr. Kamran Eshraghian. Gracias a su inestimable ayuda este proyecto ha visto finalmente la luz.

Quiero agradecer, de una forma muy especial a Valentín de Armas, José Fco. López, y Juan A. Montiel, que tanto me han ayudado durante estos meses de trabajo. A ellos sólo les puedo dar las gracias, y esperar que el tiempo me de la oportunidad de devolverles el favor.

A Pedro Peréz Carballo, que tanto ha sufrido, en el concepto más extenso de la palabra, en la realización de este proyecto. Sus frases de ánimo me han ayudado mucho más de lo que él imagina.

Quiero agradecer también la desinteresada colaboración de los Drs. Weiping Li de la Universidad de Lehigh y Javier Bruguera de la Universidad de Santiago de Compostela, que me han resuelto todas las dudas que me han surgido a lo largo del desarrollo de este proyecto a través de ese maravilloso invento llamado e–mail.

A mis compañeros Félix B. Tobajas y Silvia E. Martel. No hay palabras para expresar los momentos que hemos pasado juntos. Ellos me han visto reir y llorar, han compartido (y sentido) mis momentos difíciles, me han ayudado y soportado durante mucho tiempo en lo que se puede definir como una unión casi simbiótica. Gracias, amigos.

No puedo despedirme sin hacer, al menos, una breve mención, no por eso menos importante, a grandes compañeros y amigos que, de una u otra manera, han deja-

do una huella importante en mi vida: Leticia Pérez, Fernando de la Puente, Miguel A. Cabrera–Pinto, Ermina Herrera, Tomás Bautista, Enrique Montesdeoca, Roberto Esper–Chaín, Marta García, Antonio y Angeles Jurado, Noelia Quintana, Rosario Pérez, Gustavo Marrero, Francisco González, Luis Hernández, Francisco Cabrera, Jose Rabadán, David Ortega, Israel Villar, Alicia Hernández, Luis Benitez, Pedro D. González, Miguel Barrera, Jose Fco. Cáceres y Pedro Cortés. Todos y cada uno de ellos han puesto su granito de arena en lo que han sido seis largos años de carrera.

Índice General

Índice de Figuras

Capítulo 1

Introducción

Las aplicaciones multimedia requieren un elevado ancho de banda, al combinar la transmisión de textos, sonido e imágenes. Estas últimas son, comparativamente, las que demandan un mayor ancho de banda de transmisión, además de una gran cantidad de memoria para su almacenamiento. La compresión de imágenes significa, por tanto, una importante reducción tanto del ancho de banda necesario para transmitirlas, como de la cantidad de memoria necesaria para almacenarlas, lo que conlleva un ahorro tanto de costos como de tiempo.

Es posible hablar de compresión de imágenes estáticas y de compresión de imágenes en movimiento, caracterizando a esta última la estimación de la siguiente trama a partir de una o varias tramas previas y/o futuras, evitando así la transmisión o el almacenamiento de tramas muy similares. Así se puede continuar hablando de métodos de compresión con pérdidas, que permiten recuperar una aproximación de la imagen original, y de métodos de compresión sin pérdidas, que permiten recuperar una copia exacta de la imagen original.

Existen muchos algoritmos de compresión de imágenes. La mayor parte de ellos tienen en común la *Transformada Discreta del Coseno* (*DCT*), desarrollada por Ahmed, Natarajan y Rao en 1974 y aplicada por primera vez en este campo por Chen y Pratt en 1984.

La DCT ha demostrado una gran eficiencia en reducción de imágenes, y la complejidad de su realización está al alcance de tecnologías actualmente consolidadas. La aplicación de la DCT se efectúa sobre una porción de imagen previamente digitalizada y constituida por un rectángulo de 8×8 pixels, bien de luminancia, bien de crominancia, reduciendo la redundancia espacial, basándose en la propiedad de que una gran proporción de la energía en una imagen se concentra en un pequeño número de coeficientes.

Entre los estándares que han adoptado esta transformada como parte de su proceso se pueden nombrar el CCITT H.261 (también conocido como Px64), para videotelefonía y teleconferencia, ISO JPEG (Joint Photographic Experts Group), para transmisión y almacenamiento de imágenes estáticas, e ISO MPEG (Moving Pictures Experts Group), para transmisión y almacenamiento de imágenes en movimiento.

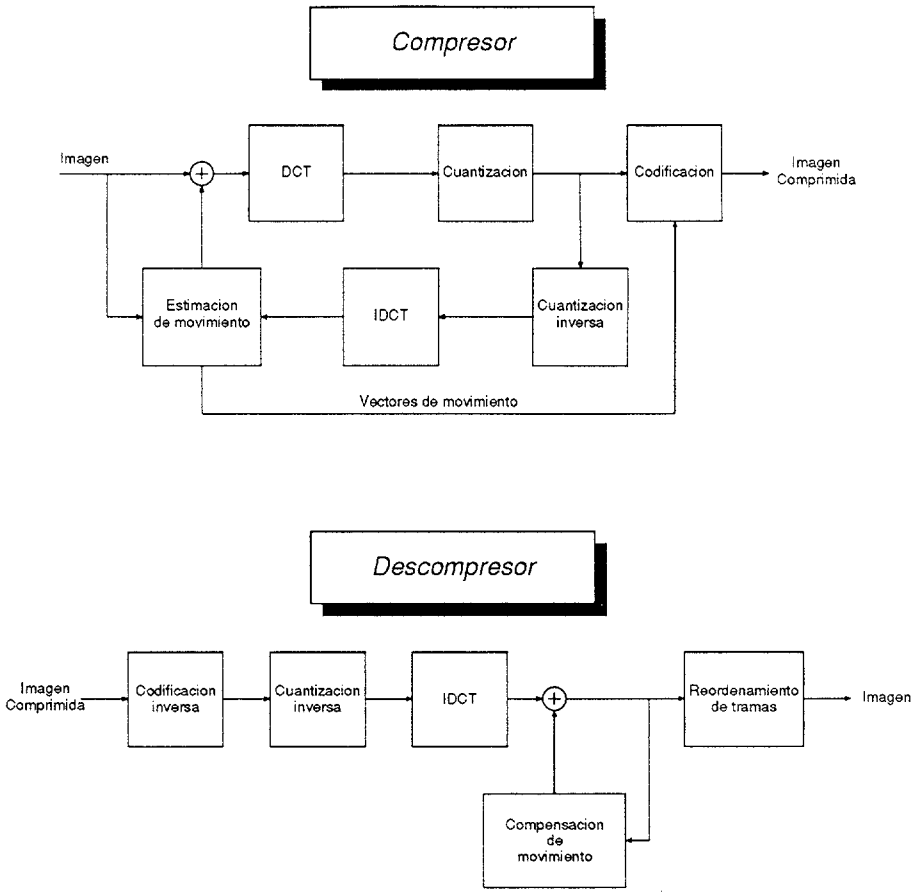


Figura 1.1: Diagrama conceptual del compresor–descompresor MPEG

La compresión de imágenes mediante MPEG se realiza siguiendo el esquema presentado en la figura 1.1, que se detalla brevemente a continuación:

- Transformada Discreta del Coseno: aprovechando el hecho de que el ojo humano es relativamente insensible a la información visual a altas frecuencias, y que la transformada discreta del coseno tiende a concentrar la energía de una imagen, al descomponerla en sus componentes frecuenciales, esta transformada permite obtener una imagen que se puede cuantizar de forma selectiva, despreciando cierta cantidad de información de las frecuencias más altas, sin que ello afecte significativamente a la calidad de la imagen.
- Cuantización: los coeficientes resultantes de la transformación se cuantizan para reducir su magnitud y para incrementar el número de coeficientes nulos, que se darán, debido a la propia naturaleza de las imágenes, en las altas frecuencias. En la figura 1.2 se puede observar los efectos de la cuantización sobre la matriz resultante de la transformación.
- Modulación DPCM y Codificación RLE: la codificación reagrupa los coeficientes de la DCT ya cuantizados en una trama formada por la lectura en zig–zag de dichos coeficientes, comenzando por las frecuencias más bajas y terminando por las más altas, como se puede observar en la figura 1.3. El primer coeficiente de la matriz resultante es el llamado *coeficiente DC*, y los restantes, los *coeficientes AC*. Los coeficientes DC varían ligeramen-

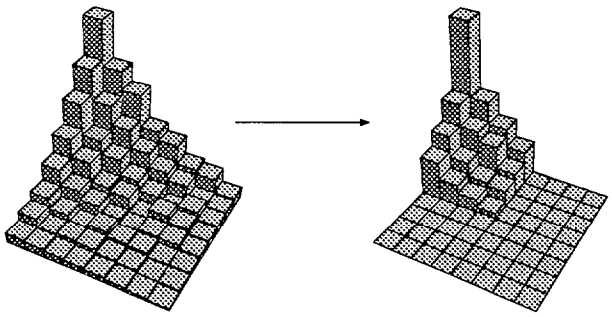


Figura 1.2: Cuantización

te entre bloques sucesivos. La codificación de estos coeficientes aprovecha esta propiedad a través de la modulación DPCM (Differential Pulse Code Modulation). Esta técnica codifica la diferencia entre el coeficiente DC del bloque actual y el del bloque previo. Los coeficientes AC suelen contener varios términos nulos, por lo que se usa una codificación RLE (Run–Length Encoding) para evitar la transmisión continua de dichos términos.

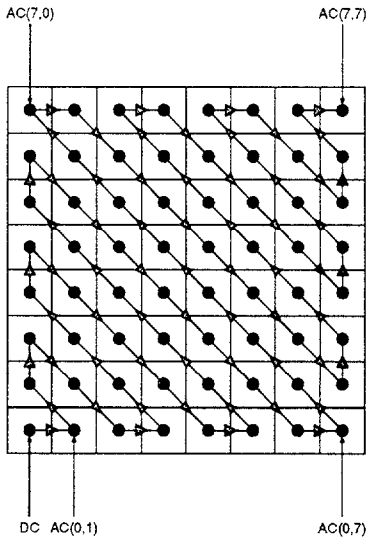


Figura 1.3: Secuencia de lectura de la matriz

- Codificación de Huffman: para incrementar el ratio de compresión se usa la codificación de Huffman sobre los resultados de la codificación DPCM y RLE. Esta codificación se basa en asignar códigos más cortos a aquellos símbolos que más se repitan, y asignar mayores códigos para los símbolos que se producen ocasionalmente.
- Estimación y compensación de movimiento: el concepto que subyace en la estimación y la compensación de movimiento es la similitud entre tramas sucesivas. Como muchas tramas en una secuencia son muy similares, debiéndose la mayoría de las variaciones a desplazamientos de una misma imagen, es posible evitar la transmisión de una segunda imagen enviando un vector de desplazamiento basado, bien en imágenenes previas, o bien en imágenes previas y posteriores. El algoritmo MPEG, tomando como base el

estándar H.261, distingue tres tipos de tramas:

- Tramas I: Tramas originales (no son el resultado de una predicción). En ellas se basa la predicción temporal.
- Tramas P: Tramas que se predicen en base a tramas *previas* de los tipo I o P. Es un tipo de predicción "hacia adelante".
- Tramas B: Tramas que se predicen en base a la interpolación entre tramas *previas* y *futuras*, de los tipos I y P. Es un tipo de predicción bidireccional.

En la figura 1.4 se puede ver la relación entre estas tramas.

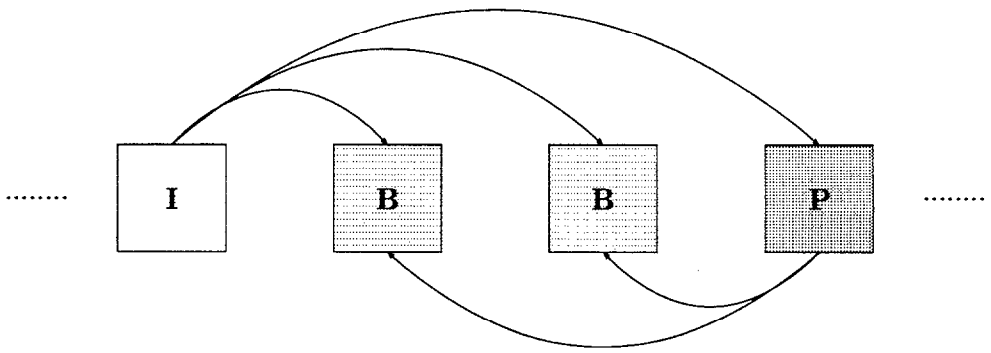


Figura 1.4: Relación entre las tramas definidas en el estándar MPEG

Los modelos matemáticos para realizar la estimación de los vectores de movimiento se pueden clasificar, básicamente, en dos grupos:

- Estimación de movimiento por coincidencia de bloques: la idea fundamental es que los vectores de movimiento se puede obtener mediante la búsqueda del bloque más semejante de la trama previa o de la trama posterior. El algoritmo de búsqueda exhaustiva está dirigido a encontrar la semejanza óptima en la región de búsqueda, a costa de una enorme carga computacional. Por este motivo, la mayoría de los esfuerzos en el método de coincidencia de bloques van dirigidos a encontrar un método de búsqueda que reduzca la carga computacional a costa de una pérdida en la precisión.
- Algoritmos recursivos basados en el método del gradiente (diferenciales): supóngase que las variaciones de la luminancia en tramas sucesivas se deben al desplazamiento de un objeto. El vector de movimiento realizará iteraciones sucesivas en la dirección del gradiente de la luminancia de forma que, tras varias aproximaciones convergerá hacia el desplazamiento actual. Este tipo de algoritmos puede

realizar la iteración tanto bloque por bloque, como pixel por pixel, dando lugar a los métodos recursivos por bloque o recursivos por pixel. El rendimiento de este tipo de algoritmos es bastante bueno para desplazamientos lentos, pero en imágenes con cambios bruscos y movimientos complicados, se produce una degradación importante de los resultados.

Dos estructuras son las que mayor capacidad de cálculo requieren en el compresor de imágenes: la *transformada discreta del coseno* y el *estimador de movimiento*. Una mayor velocidad en el cómputo de la transformada discreta del coseno permitirá una mayor presición en el algoritmo estimador de movimiento, al aumentar el tiempo que éste dispone para obtener los vectores de movimiento, lo que se traducirá, finalmente, en una mejor calidad de la imagen.

1.1 Objetivos del proyecto

La *Transformada Discreta del Coseno* es una técnica para la reducción de la redundancia espacial. Es ampliamente utilizada, por tanto, en la compresión de imágenes, ya que una característica de éstas es la pequeña diferencia entre los valores de los pixels adyacentes.

Para ilustrar este efecto se parte de un polígono, representado por una matriz de 8×8 elementos, donde el símbolo O representa el exterior del polígono y el símbolo X representa el interior del mismo.

O	O	O	X	X	O	O	O
O	O	X	X	X	X	O	O
O	X	X	X	X	X	X	O
X	X	X	X	X	X	X	X
X	X	X	X	X	X	X	X
O	X	X	X	X	X	X	O
O	O	X	X	X	X	O	O
O	O	O	X	X	O	O	O

Asignando un valor de -10 al símbolo O, y un valor de $+10$ al símbolo X, la representación del polígono quedará:

-10	-10	-10	$+10$	$+10$	-10	-10	-10
-10	-10	$+10$	$+10$	$+10$	$+10$	-10	-10
-10	$+10$	$+10$	$+10$	$+10$	$+10$	$+10$	-10
$+10$	$+10$	$+10$	$+10$	$+10$	$+10$	$+10$	$+10$
$+10$	$+10$	$+10$	$+10$	$+10$	$+10$	$+10$	$+10$
-10	$+10$	$+10$	$+10$	$+10$	$+10$	$+10$	-10
-10	-10	$+10$	$+10$	$+10$	$+10$	-10	-10
-10	-10	-10	$+10$	$+10$	-10	-10	-10

Al aplicar la DCT, la matriz resultante concentra la energía en unos pocos coeficientes, con lo cual la cantidad de datos se ve reducida, no así la cantidad de información, ya que aplicando el proceso inverso (la *transformada discreta inversa del coseno*), se recupera la matriz original. El resultado de la aplicación de la DCT se puede observar en la siguiente matriz:

$$\begin{matrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -3 & 0 & 0 & 0 & -8 & 0 & -20 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 18 & 0 & 20 & 0 & -8 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -45 & 0 & -20 & 0 & 18 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 20 & 0 & -45 & 0 & 0 & 0 & -3 & 0 \end{matrix}$$

Para ver el efecto en un caso real, se parte de la imagen mostrada en la figura 1.5.

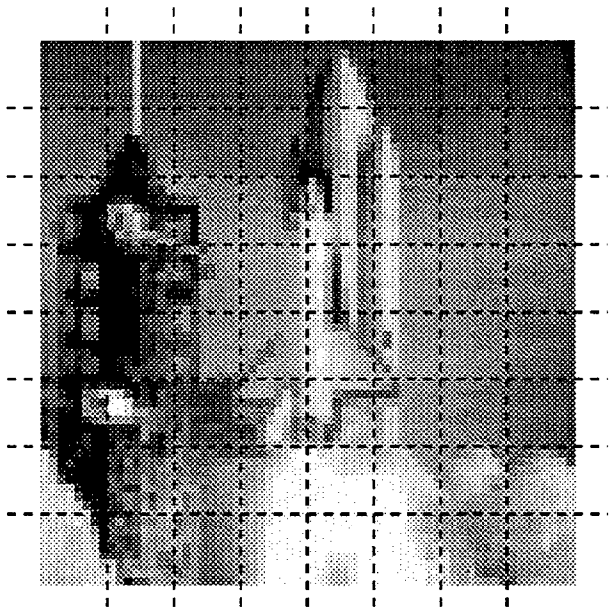


Figura 1.5: Imagen original

Dividiendo dicha imagen en bloques de 8×8 pixels, y aplicando la DCT se obtiene la imagen transformada de la figura 1.6, donde se puede observar que la energía tiende a concentrarse en ciertos coeficientes. En dicha figura se puede ver una mayor dispersión en la zona inferior izquierda comparada con la superior derecha, donde la imagen es más uniforme, y por tanto existe una mayor redundancia espacial.

Una mirada más precisa a uno de los bloques de la imagen transformada (figura 1.7) confirma la propiedad de que la mayor cantidad de energía tiende a concentrarse en el coeficiente (0,0) (*coeficiente DC*).

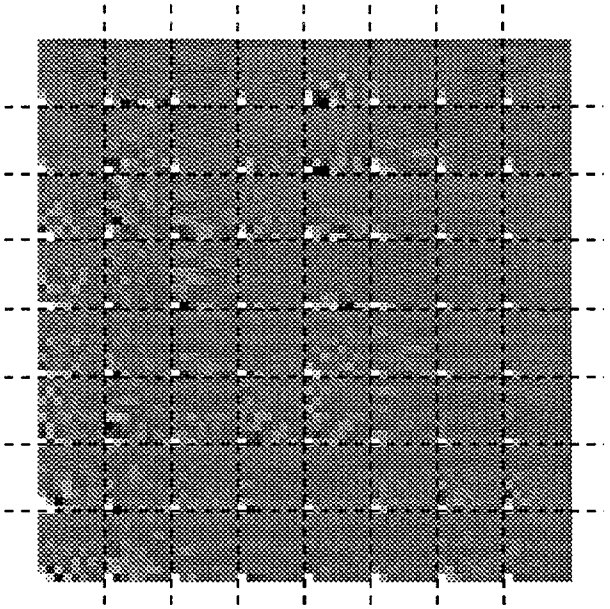


Figura 1.6: Imagen transformada

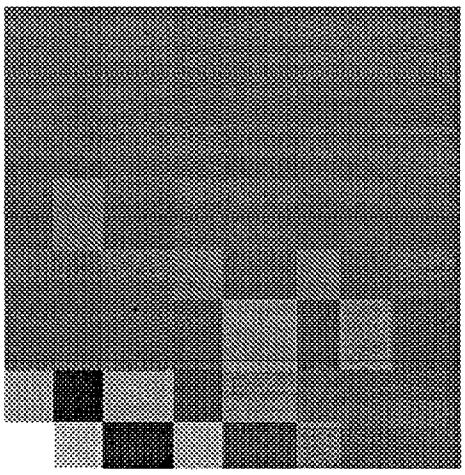


Figura 1.7: Bloque transformado

En este proyecto, y como primera aproximación a la compresión de imágenes, se realizará un estudio arquitectural de la *Transformada Discreta del Coseno* (fácilmente extrapolable a la transformada inversa), buscando una estructura viable que permita realizar la transformación en el menor tiempo posible. Para ello la implementación se realizará en *Arseniuro de Galio*, que, además de ser un material que permite alcanzar altas frecuencias de trabajo, actualmente posee una relativamente alta densidad de integración. La elección del Arseniuro de Galio (una tecnología no tan consolidada como la del Silicio) conllevará la necesidad de estudiar muchas de las estructuras a utilizar, ya que su implementación supondrá un aspecto novedoso en esta tecnología.

1.2 Estructura de la memoria del proyecto

Este proyecto ha sido estructurado en cuatro grandes bloques.

El primero de ellos (*Capítulo 2*) consiste en una descripción de la tecnología utilizada, el Arseniuro de Galio, sus parámetros más importantes, así como las opciones y limitaciones que tiene el diseñador a la hora de realizar un circuito integrado.

En el segundo bloque (*Capítulo 3*), se realiza una descripción del algoritmo de la *Transformada Discreta del Coseno*, aportando una solución viable para su implementación VLSI. Además, se estudia el tipo de aritmética a utilizar así como un compendio de conceptos arquitecturales básicos para la implementación en base a dicha aritmética. Por último, se estudian en profundidad las dos arquitecturas que serán las finalmente implementadas.

En el tercer bloque (*Capítulo 4*) se realiza un estudio de las primitivas necesarias para la implementación de los circuitos, aportando soluciones novedosas en aspectos críticos del diseño en Arseniuro de Galio.

En el cuarto y último bloque (*Capítulo 5*) se realiza el estudio de la unidad de control de los circuitos y se dan las prestaciones de la implementación final de los mismos.

Capítulo 2

Introducción a la tecnología de Arseniuro de Galio

El *Arseniuro de Galio* es un material que, en sus comienzos, se destinó a la implementación de amplificadores de microondas y de dispositivos LED. Actualmente, también es usado para el diseño de circuitos integrados digitales dirigido a aquellas aplicaciones que requieren altas velocidades. Esta utilización comenzó en 1974 con circuitos SSI destinados a divisores con una alta frecuencia de trabajo, y, con el paso de los años, se ha consagrado en aplicaciones LSI y VLSI.

2.1 Arseniuro de Galio *vs* Silicio

El Arseniuro de Galio presenta una serie de ventajas frente al Silicio, fácilmente deducibles de la figura 2.1, que se pueden resumir en:

- El Arseniuro de Galio es un semiconductor de "gap–directo"; lo que significa que la mínima separación de energía entre la banda de conducción y la de valencia se produce para el mismo *momento cristalino del electrón.* Por otro lado, el Silicio es un material de "gap–indirecto", ya que el mínimo de la banda de conducción y el máximo de la de valencia se encuentran separadas en dicho *momento cristalino.* La característica de semiconductor de "gap–directo" le brinda al Arseniuro de Galio unas características especiales para la emisión de luz, ya que el paso de un electrón de la banda de valencia a la de conducción se produce sin emisión de *fonones* (sin cambio de *momento cristalino*).
- La mayor separación entre las bandas de valencia y conducción para el Arseniuro de Galio produce un sustrato semi–aislante, con alta resistencia, lo cual influye en la minimización de la capacidades parásitas que permiten aislar eléctricamente distintos dispositivos en el mismo sustrato.
- Debido a esta mayor separación de la bandas de valencia y conducción es posible un incremento en el rango de temperaturas en el que se puede emplear esta tecnología.
- La movilidad de los electrones en Arseniuro de Galio es muy superior a la de los electrones en el Silicio, debido a que existe una relación inversa entre

ésta y la masa efectiva de los electrones, que es, a su vez, directamente proporcional a la inversa de la curvatura $(d^2E/dK^2)^{-1}$.

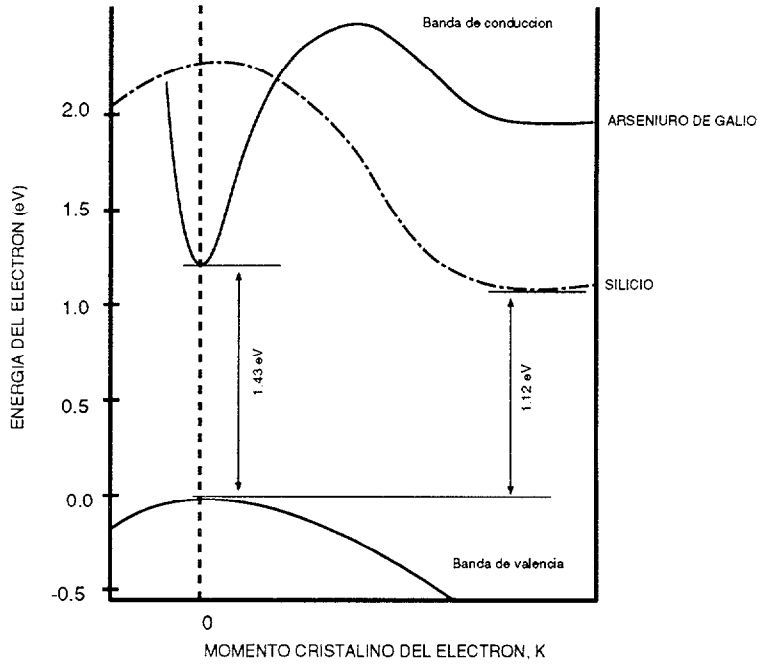


Figura 2.1: Estructura de Bandas Energéticas del GaAs y el Si

En la figura 2.2 se representa la velocidad de los electrones en función del campo eléctrico aplicado. El incremento en el campo eléctrico aplicado produce un aumento de la velocidad de los electrones. A su vez, y debido a las colisiones con la red cristalina, dichos electrones van perdiendo parte de su energía. El balance entre los dos efectos condiciona la velocidad final de los electrones. Si continúa aumentando el campo eléctrico, se alcanza un determinado valor en el que el balance anteriormente mencionado se compensa, y, a partir del cual, la velocidad de los electrones comienza a disminuir hasta un valor límite, denominado *velocidad de saturación.*

En dicha figura, además, se puede observar cómo en las regiones de bajo campo eléctrico, el Silicio tiene mucha menor movilidad que el Arseniuro de Galio. Sin embargo, la velocidad de saturación del GaAs es prácticamente la misma que la del Silicio, con la salvedad que la del primero se consigue para campos eléctricos sustancialmente menores.

2.2 Dispositivos GaAs

En los últimos años, se han desarrollado distintos tipos de dispositivos en GaAs. Hasta el momento puede hablarse de, al menos, dos generaciones. La primera incluye:

- D–MESFET: Transistor de efecto campo metal semiconductor en modo de deplexión.

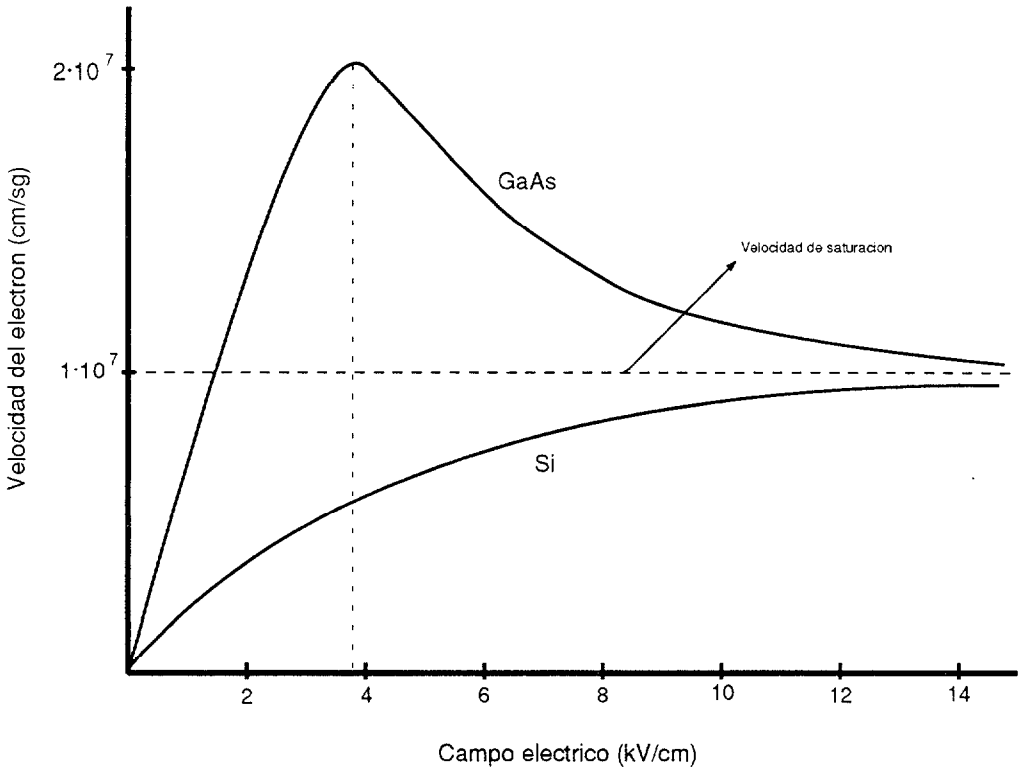


Figura 2.2: Característica campo–velocidad de los e^- en GaAs y Si

- E–MESFET: Transistor de efecto campo metal semiconductor en modo de enriquecimiento.
- E–JFET: Transistor de efecto campo de unión en modo de enriquecimiento.
- CE–JFET: Transistor de efecto campo de unión complementario.

Con esta primera generación de dispositivos se alcanzan velocidades de conmutación del orden de 80ps, con un consumo de potencia entre los 0.1 y 1.5 mW.

La segunda generación está constituida por dispositivos más sofisticados:

- HEMT: Transistor de alta movilidad de electrones.
- HBT: Transistor bipolar de heterounión.

En esta segunda generación, los transistores pueden llegar a tener una movilidad de electrones de hasta 5 veces mayor que los de la primera.

En esta sección se estudiarán los transistores MESFET, tanto de deplexión como de enriquecimiento, comenzando por una descripción de su estructura, que ayudará a comprender el comportamiento de las familias lógicas estudiadas en secciones posteriores.

2.2.1 MESFETs de deplexión

La estructura básica de un D–MESFET, figura 2.3, es muy simple, estando formada por una fina región tipo N^+, con una alta concentración de impurezas, que permite la

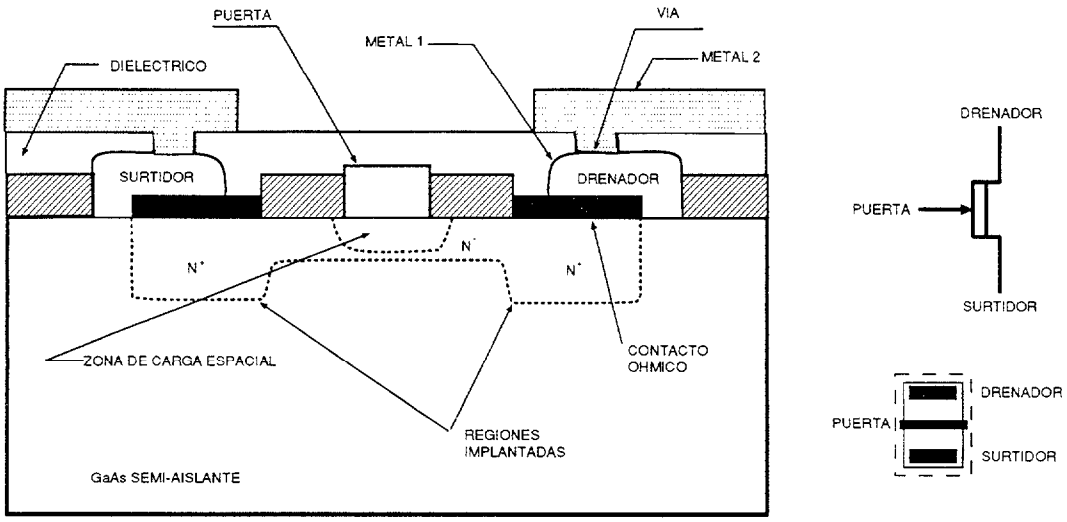


Figura 2.3: Dispositivo MESFET de deplexión, símbolo y layout simbólico

circulación de corriente entre drenador y surtidor sin la aplicación de un potencial en la puerta. El paso a estado de corte se realiza por medio de una aplicación de tensión negativa a la puerta con respecto a la terminal de fuente. Mientras la tensión puerta–surtidor va disminuyendo, el ancho de la zona de carga espacial aumenta, y el ancho de canal a través del cual fluye la corriente disminuye. Con una tensión de puerta igual a la tensión de estrangulamiento, desaparece el canal, con lo cual la corriente cesa y el dispositivo pasa a estado de corte. Esta tensión de estrangulamiento se puede variar en el proceso de fabricación modificando la profundidad y el nivel de dopaje de la región activa, hasta colocarla en el valor negativo deseado.

2.2.2 MESFETs de enriquecimiento

El transistor E–MESFET, figura 2.4, es similar al D–MESFET anteriormente explicado, con la salvedad de que ahora la profundidad de canal es menor así como su nivel de dopaje. Esto se traduce en un estado de conducción nulo para una tensión entre puerta y surtidor igual a cero.

Con la aplicación de una tensión positiva en la puerta, el diodo Schottky comenzará a estar directamente polarizado y la zona de carga espacial disminuirá, aumentando la conductancia del canal y permitiendo que fluya corriente entre drenador y surtidor.

2.2.3 Diodo Schottky

El efecto de la barrera Schottky se produce por el contacto de un metal (el de puerta) con un semiconductor, que crea una barrera de potencial electroestático. Para ilustrar este efecto, se partirá de un metal de puerta que se va acercando gradualmente a la superficie del semiconductor, hasta que la distancia que los separa sea nula.

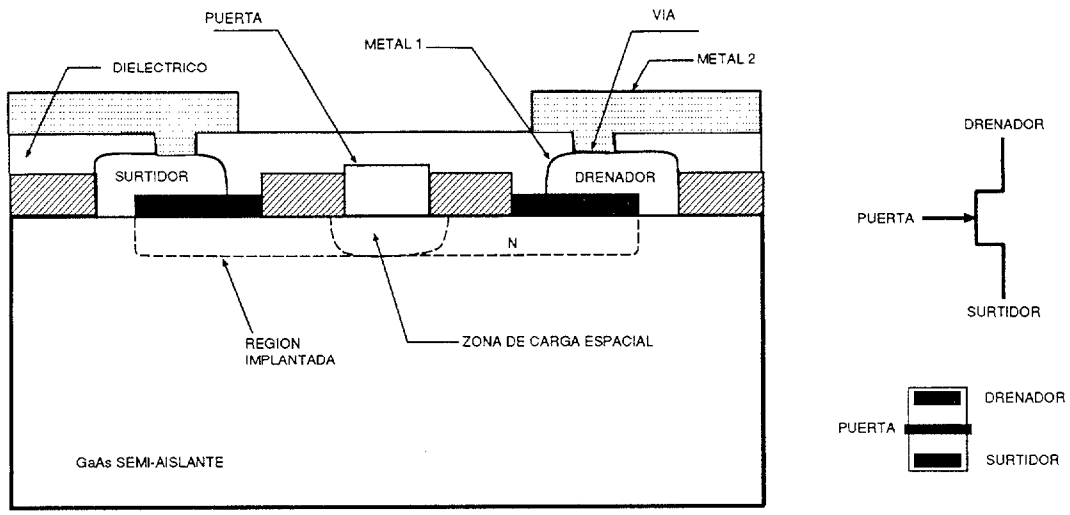


Figura 2.4: Dispositivo MESFET de enriquecimiento, símbolo y layout simbólico

Según se reduce la distancia metal–canal, la carga inducida en el canal se incrementa, a la vez que aumenta la zona de carga espacial. Una gran parte de la diferencia de potencial aparece en la zona de carga espacial, dentro del canal. Como la concentración de portadores en el metal es varios órdenes de magnitud mayor que la que existe en el canal, cuando la separación se hace cero, la caída de potencial total se produce dentro del propio canal.

La puerta de un transistor MESFET es, por tanto, un diodo de barrera Schottky que conduce cuando está polarizado en directa. La magnitud de la corriente que es capaz de conducir depende de la altura de la barrera Schottky, la tensión aplicada y el área de la puerta.

2.3 Familias lógicas: DCFL, SDCFL y SBFL

Este apartado se realizará una descripción del funcionamiento básico de las puertas implementadas usando lógica **DCFL**, **SDCFL** y **SBFL**, que han sido las utilizadas en la implementación de la DCT, y su influencia en términos de potencia, margen de ruido, *fan–in*, *fan–out*, retardo y área en el circuito.

2.3.1 DCFL (Direct Coupled FET Logic)

La familia lógica más utilizada en circuitos de alta velocidad es la DCFL, debido principalmente a la simplicidad de su estructura y a los bajos consumos de potencia que presenta.

En la familia lógica DCFL se emplean dispositivos de enriquecimiento como elementos de conmutación, y un dispositivo de deplexión como carga activa, cortocircuitando la puerta y el surtidor de forma que la tensión V_{gs} sea cero, y trabajando, consecuentemente, en la región de saturación, actuando como una fuente de corriente constante. En esta configuración, el transistor de deplexión se denomina transistor

de *pull-up* y los transistores de enriquecimiento se denominan transistores de *pull-down.*

En la figura 2.5 se representa tanto el esquema a nivel de transistores como el layout simbólico del inversor DCFL.

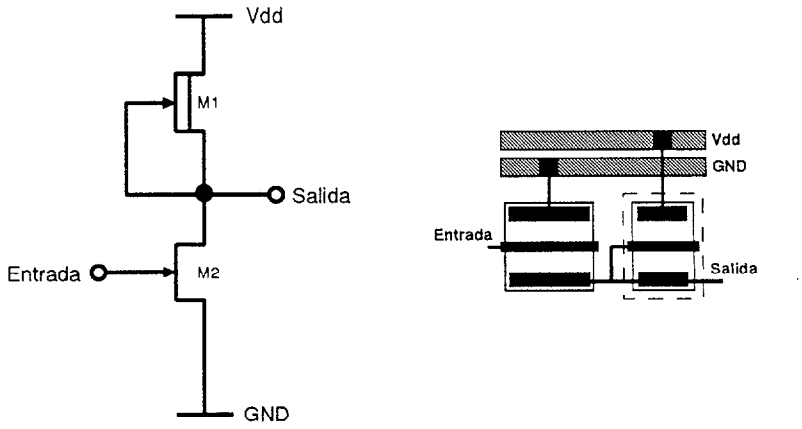


Figura 2.5: Inversor DCFL

Cuando la entrada está a nivel bajo, el transistor E–MESFET de conmutación M2 se halla en estado de corte, por lo que la conducción se produce a través del transistor D–MESFET de carga M1, que conecta el bus de alimentación positivo *Vdd* a la salida, como se muestra en la figura 2.6(a). Por lo tanto, la salida se situará a un nivel lógico alto. Cuando la entrada está a nivel alto, el transistor E–MESFET de conmutación M2 se halla en región lineal y la salida del inversor está conectada a *GND* a través de la *resistencia de canal* del transistor tipo E (R_{M2}), comportándose el transistor de enriquecimiento como una resistencia controlada por tensión, como se muestra en la figura 2.6(b). Consecuentemente, la salida se situará a un nivel lógico bajo.

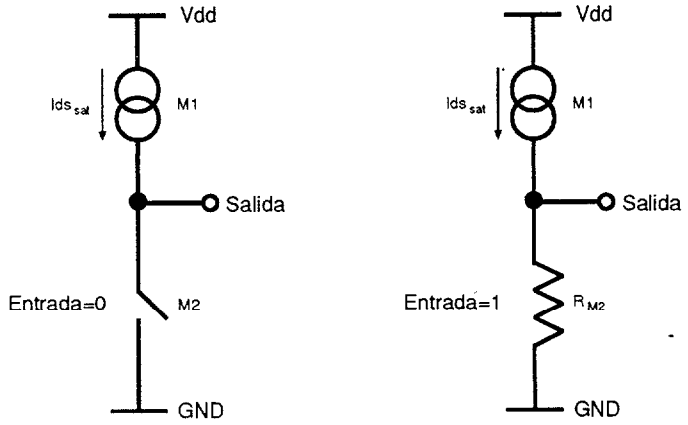


Figura 2.6: Circuito equivalente (a) con la entrada a nivel bajo y (b) con la entrada a nivel alto

Se pueden conectar en paralelo varios elementos de conmutación (transistores E–MESFET) para formar puertas NOR de múltiples entradas, tal como se muestra

en la figura 2.7.

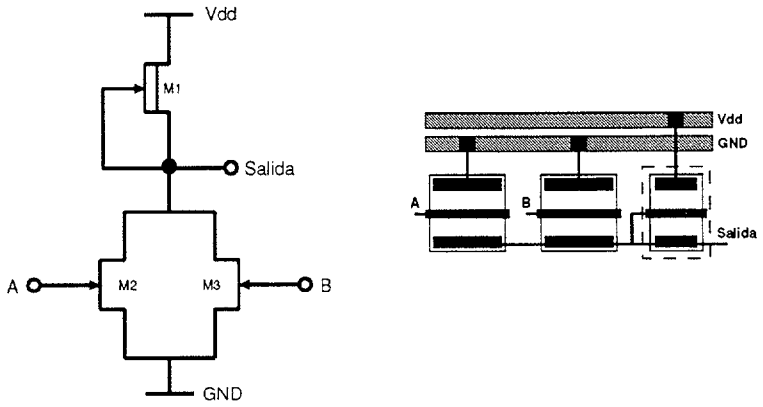


Figura 2.7: NOR de 2 entradas DCFL

Esta familia lógica es la que menos área requiere, así como la que menos potencia disipa. Como desventaja, hay que señalar una alta sensibilidad, tanto con el *fan–in*, como con el *fan–out*, así como una capacidad de carga muy pobre.

2.3.1.1 Concepto de *fan–in*

El *fan–in* de una puerta NOR DCFL se puede definir como el número de transistores de enriquecimiento que pueden situarse en paralelo. El transistor de conmutación E–MESFET se considera que está en estado lineal o en corte dependiendo de la tensión de entrada aplicado a la puerta. Sin embargo, en realidad, cuando el transistor E–MESFET está en estado de corte, fluye todavía una corriente drenador–surtidor, denominada *corriente de fugas.*

Cuando se conectan varios transistores tipo E en paralelo, como en el caso de las puertas NOR DCFL, se obtiene un flujo significativo de corriente de fugas, como se muestra en la figura 2.8, que tendrá influencia en el nivel de tensión correspondiente al 1 lógico. Es decir, con todos los transistores E–MESFET en estado de corte, la tensión de salida no alcanzará el nivel que se alcanzaba en el caso del inversor. A medida que se introducen transistores de enriquecimiento en paralelo, el nivel de tensión de salida correspondiente al 1 lógico se ve degradado.

2.3.1.2 Concepto de *fan–out*

El *fan–out* de una puerta DCFL se refiere básicamente al número de puertas que la puerta DCFL puede atacar. En la figura 2.9 se presenta un modelo simplificado equivalente que ilustra la influencia del *fan–out* en la puerta DCFL.

Como se puede observar en la figura 2.9(b), el transistor de *pull–up* se representa por una fuente de corriente. A medida que el número de puertas atacadas aumenta, la

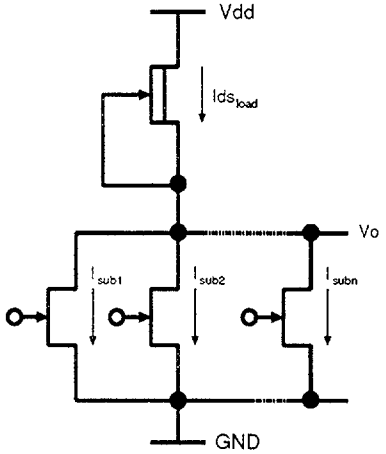


Figura 2.8: Corrientes de fugas en una puerta NOR DCFL.

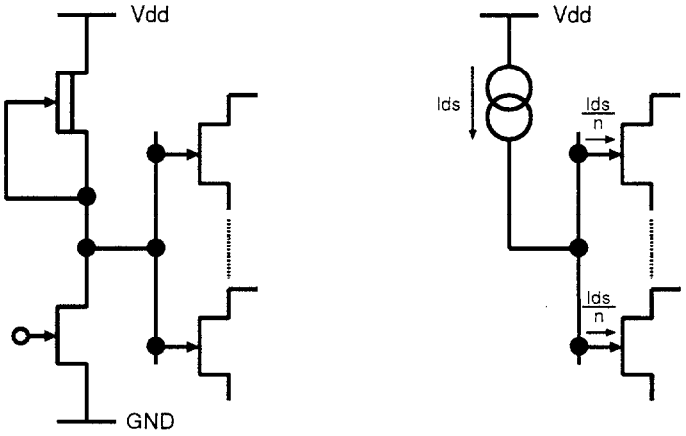


Figura 2.9: Influencia del *fan–out* sobre las puertas DCFL (a) Puerta lógica (b) Circuito equivalente

corriente que va hacia cada puerta disminuye, lo que implica una disminución del nivel de tensión correspondiente al nivel alto (1 lógico), y por lo tanto del *margen de ruido*, motivo por el cual se debe limitar el *fan–out* de una puerta DCFL de dimensiones mínimas, a 2 o 3 como máximo.

2.3.1.3 Determinación del *margen de ruido*

Intuitivamente, el *margen de ruido* se puede definir como el nivel de ruido máximo que puede presentarse en la entrada antes de que se produzca un valor lógico erróneo en la salida. Los *márgenes de ruido* son, por tanto, una medida de la tolerancia del circuito frente al ruido.

Para la determinación del margen de ruido, se utiliza una cadena de inversores (figura 2.10), de forma que la salida de uno de ellos actúe como entrada del *elemento a estudiar*, y la salida de éste, como entrada del siguiente.

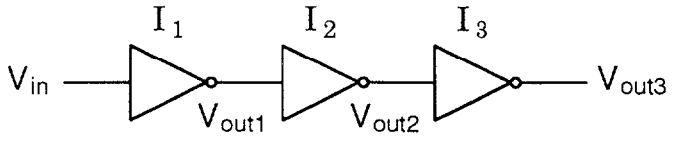


Figura 2.10: Circuito usado para hallar el margen de ruido del inversor I_2

En la figura 2.10, el inversor I_1 transforma la señal de entrada V_{in} en un nivel lógico compatible con la familia lógica DCFL. El inversor I_3 actúa como carga del inversor I_2, cuyo *margen de ruido* se desea hallar.

En la figura 2.11 se representa la característica de transferencia (V_{out2} *vs* V_{out1}) del inversor I_2 presentado en la figura 2.10.

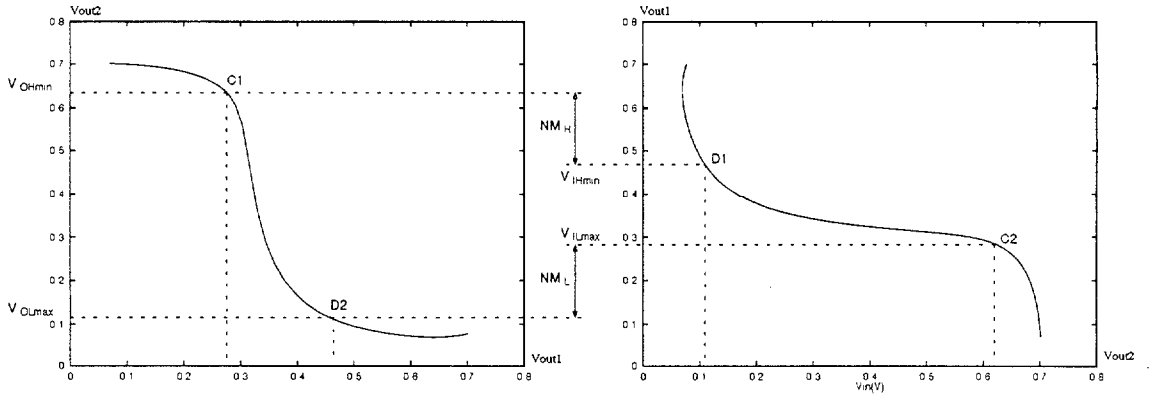


Figura 2.11: Característica de transferencia del inversor DCFL

Considerando el punto C1, en el que la tensión de salida V_{out2} entra en la región de transición (denominando a la tensión en este punto V_{OHmin}), así como el punto D1, en el que la tensión de la señal de entrada V_{out1} produce la transición de la salida del inversor I_2 (denominando a la tensión en este punto V_{IHmin}), se puede definir el *margen de ruido del nivel alto* del inversor I_2 como NM_H, donde:

$$NM_H = V_{OHmin} - V_{IHmin}$$

De la misma forma, se puede definir el *margen de ruido del nivel bajo* del inversor I_2 como NM_L, donde

$$NM_L = V_{ILmax} - V_{OLmax}$$

siendo el *margen de ruido* global el menor de los valores NM_H y NM_L. Este método de cálculo del *margen de ruido* se denomina *método de la máxima pendiente.*

Una técnica más exacta para el cálculo del *margen de ruido*, consiste en representar la tensión de entrada al inversor frente a la tensión de salida del inversor (V_{out1} *vs* V_{out2}) así como la curva V_{out2} *vs* V_{out1} sobre el mismo conjunto de ejes, tal y como se representa en la figura 2.12.

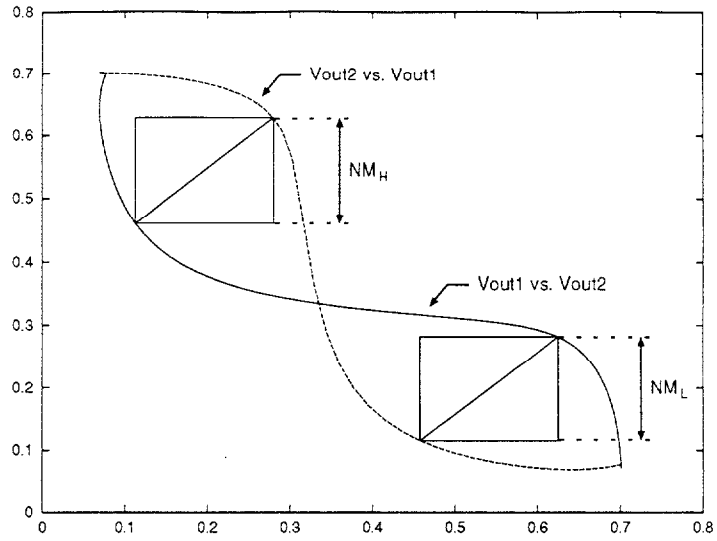


Figura 2.12: Márgenes de ruido del inversor I_2

La intersección de las dos curvas representadas, da lugar a dos regiones en las que se construye el *máximo cuadrado inscrito posible*. El margen de ruido calculado usando el método del máximo cuadrado inscrito se define como la dimensión del máximo cuadrado que puede ser dibujado, de lados paralelos a los ejes, dentro de las regiones determinadas por las curvas representadas. Los márgenes de ruido pueden hallarse para las regiones superior e inferior inscritas, tomándose la menor de las dos dimensiones para definir el *margen de ruido* global.

El método del máximo cuadrado inscrito proporciona una representación visual del margen de ruido. Cuanto mayores son las regiones inscritas en las curvas, mejor definidos están los niveles de tensión, y por lo tanto, mejores son los márgenes de ruido.

2.3.1.4 Notación en anillo

Para la realización del trazado geométrico de circuitos DCFL, puede utilizarse la metodología usada en tecnología nMOS. En ella el transistor de conmutación (*pull-down*) se sitúa por debajo del de carga (*pull-up*). El bus de alimentación (V_{dd}) se localiza en la parte superior y el bus de tierra (GND) en la parte inferior del trazado, tal y como se muestra en la figura 2.13.

El principal problema que se presenta en esta metodología, cuando se aplica al diseño en GaAs y se utiliza la lógica DCFL, es que el transistor tipo E es de un orden de magnitud mayor que el tipo D, por lo que se desaprovecha un espacio importante. Para facilitar el paso del esquema a nivel de transistores a la implementación a nivel de *layout*, se usará la metodología denominada **Notación en Anillo**. En esta metodología, el bus GND se sitúa entre la lógica y el bus V_{dd}, tal y como se ilustra en la figura 2.14.

La **Notación en Anillo** permite una formulación intermedia que se muestra en el inversor de la figura 2.15. En dicha figura, la línea de puntos representa un transis-

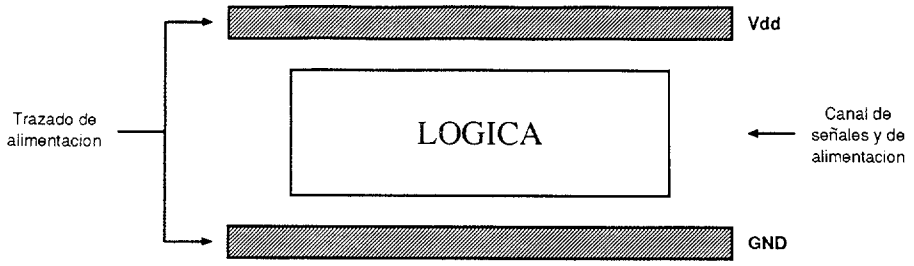


Figura 2.13: Trazado típico de nMOS

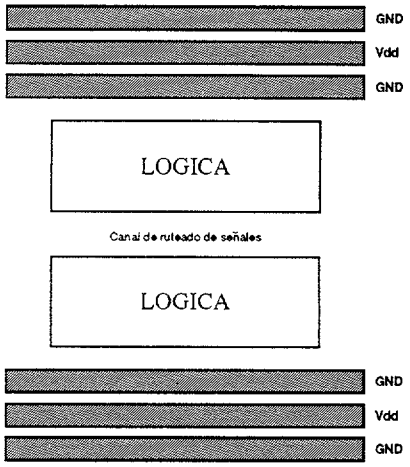


Figura 2.14: Notación en anillo

tor E–MESFET de conmutación, mientras que la línea sólida representa al transistor D–MESFET de carga. Puesto que en esta tecnología sólo se admite el uso de puertas NOR, la representación de las mismas se realiza añadiendo líneas de entrada (sin necesidad de añadir líneas que representen transistores E–MESFET), y asumiendo que eso conlleva la adición de un transistor de conmutación por cada línea de entrada añadida. En la figura 2.16 se representa, como ejemplo, una NOR DCFL de 2 entradas.

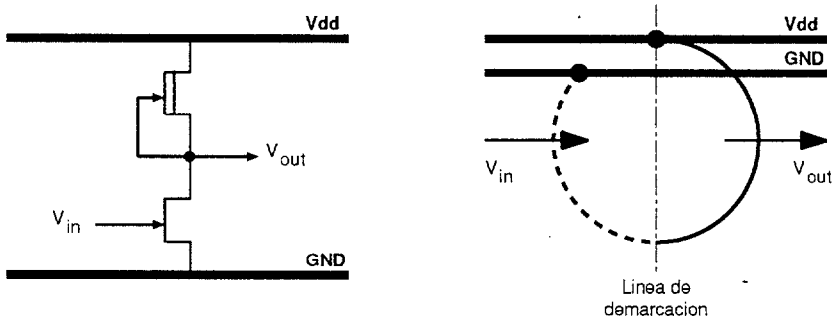


Figura 2.15: Notación en anillo del inversor DCFL

En la figura 2.17 se puede observar la representación de un inversor DCFL y su implementación a nivel de layout simbólico. La traducción desde la **Notación en Anillo** a un trazado simbólico convencional y posteriormente a un trazado geométrico es directa. Lo que viene a ser importante en la **Notación en Anillo**, como método

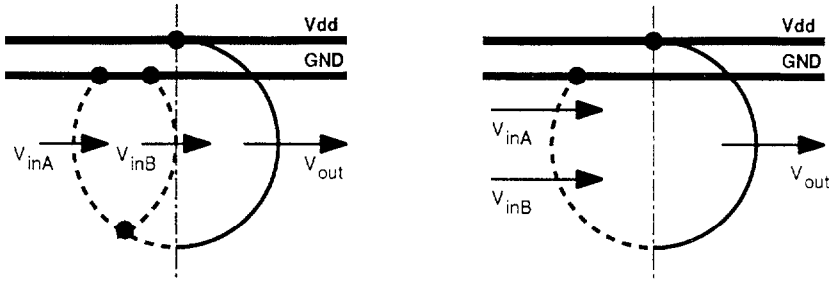


Figura 2.16: Notación en anillo de la puerta NOR DCFL

intermedio de describir un trazado físico, es que se pueden destacar las rutas de la señales y a continuación formular una estrategia de interconexión, antes de traducirla en un trazado definitivo.

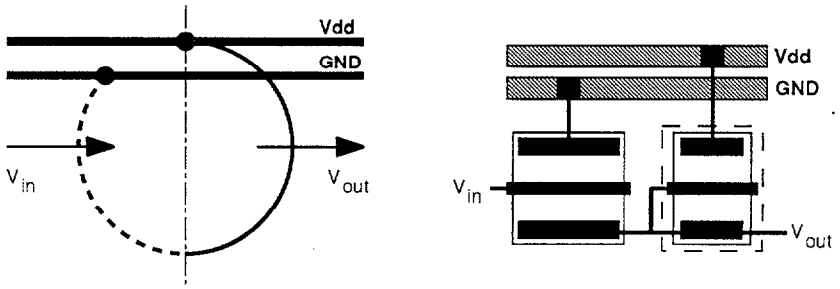


Figura 2.17: Traducción de Notación en anillo a layout simbólico

2.3.2 SDCFL (Source Direct Coupled FET Logic)

Esta familia lógica nace por la necesidad de aumentar tanto la capacidad de carga como de disminuir la sensibilidad frente al *fan-out* de la lógica DCFL.

Usa una primera etapa en lógica DCFL (M1 y M2) y a continuación una etapa *seguidora de fuente* (M3 y M4), como se muestra en la figura 2.18.

Cuando la entrada está a nivel bajo (0 lógico) el transistor D–MESFET M1 opera en la región lineal, mientras que el transistor E–MESFET de la etapa DCFL (M2) se encuentra en estado de corte, por lo que la salida de la etapa DCFL está a nivel alto. Consecuentemente, el transistor E–MESFET M3 opera en la región de saturación, mientras que el dispositivo tipo D M4 opera en la región lineal, por lo que a la salida del inversor SDCFL se obtiene un nivel alto (1 lógico). Como el transistor E–MESFET M3 está en saturación, el nivel de tensión de salida disminuye una cantidad equivalente a la tensión umbral del transistor tipo E. Esta situación se muestra en la figura 2.19(a). Cuando la entrada está a nivel alto (1 lógico), situación que se muestra en la figura 2.19(b), el transistor D–MESFET M1 está en saturación, mientras que el transistor tipo E M2 opera en la región lineal, comportándose como una resistencia controlada por tensión, por lo que la salida de la etapa DCFL está a nivel bajo. Sin embargo, la tensión *puerta-surtidor* del transistor E–MESFET M3 (V_{gsM3}) es menor

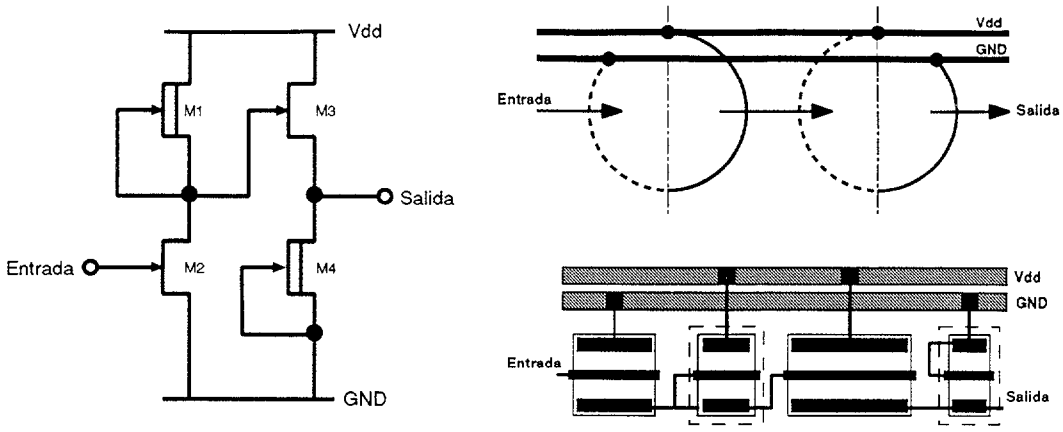


Figura 2.18: Inversor SDCFL

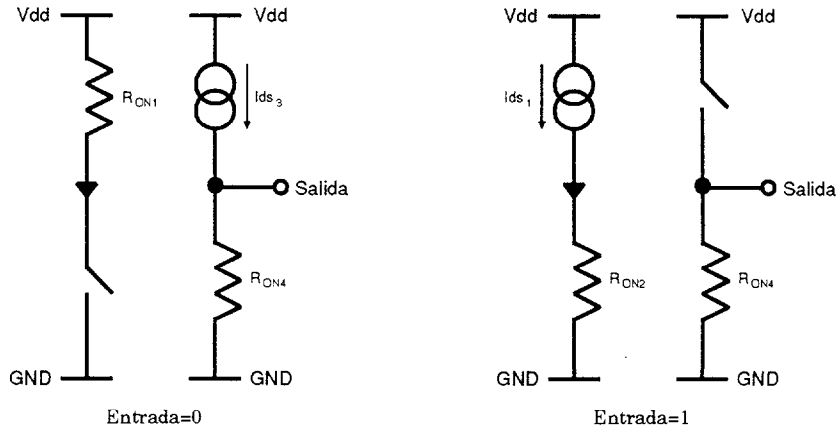


Figura 2.19: Circuito equivalente (a) con la entrada a nivel bajo y (b) con la entrada a nivel alto

a la *tensión umbral* del dispositivo tipo E, por lo que, como resultado, el transistor M3 está en estado de corte, mientras que el transistor D–MESFET M4 opera en región lineal, por lo que a la salida del inversor SDCFL se obtiene un nivel bajo (0 lógico).

El inversor básico SDCFL puede extenderse para formar funciones lógicas. Por ejemplo, en la figura 2.20 se presenta una NOR de 3 entradas SDCFL, formada por una NOR de 3 entradas DCFL y un *seguidor de fuente*.

En comparación con la familia lógica DCFL, la familia SDCFL presenta un mayor *margen de ruido*, aumentando, por otro lado, el área requerida y la potencia disipada por las mismas.

Una característica importante de esta lógica es que permite la implementación de *OR cableadas*. Eso significa un considerable ahorro en la implementación de puertas XOR, XNOR y multiplexores, frente a la implementación con lógica DCFL. La estructura típica de una puerta AOI (And–Or–Invert) se puede observar en la figura 2.21.

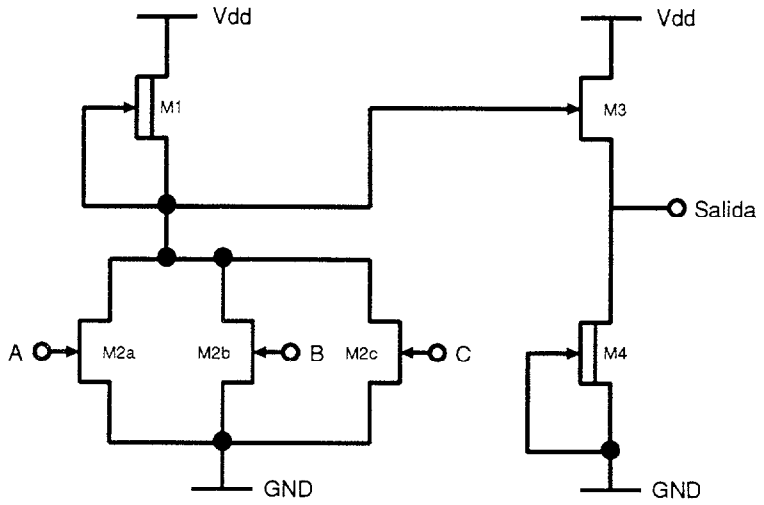


Figura 2.20: NOR de 3 entradas SDCFL

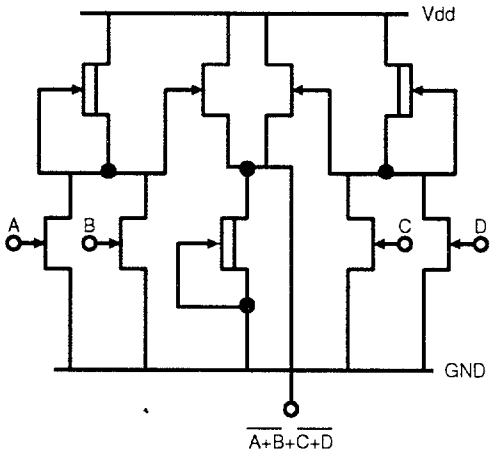


Figura 2.21: Puerta AOI

2.3.3 SBFL (Super Buffer FET Logic)

Tiene su aplicación cuando la capacidad de carga o el *fan–out* es alto. Consiste en una **etapa DCFL**, seguida por un **super buffer push/pull** formado por dos transistores tipo E de forma que uno actúa como seguidor de fuente (M3) y el segundo (M4) como *pull–down* (figura 2.22).

Cuando la entrada está a nivel bajo (0 lógico) el transistor D–MESFET M1 opera en la región lineal, mientras que el transistor E–MESFET de la etapa DCFL (M2) se encuentra en estado de corte, por lo que la salida de la etapa DCFL está a nivel alto. Por otro lado, el transistor E–MESFET de *pull-down* (M4) pasa al estado de corte, permitiendo a la carga capacitiva de salida recibir la corriente de carga proporcionada por el transistor tipo E que forma el *seguidor de fuente* (M3), que opera en saturación. Por lo tanto, en la salida del inversor SBFL se obtiene un nivel alto (1 lógico). Esta situación se muestra en la figura 2.23(a). Cuando la entrada está a nivel alto (1 lógico), situación que se muestra en la figura 2.23(b), el transistor D–MESFET M1 está en saturación, mientras que el transistor tipo E M2 opera en región lineal,

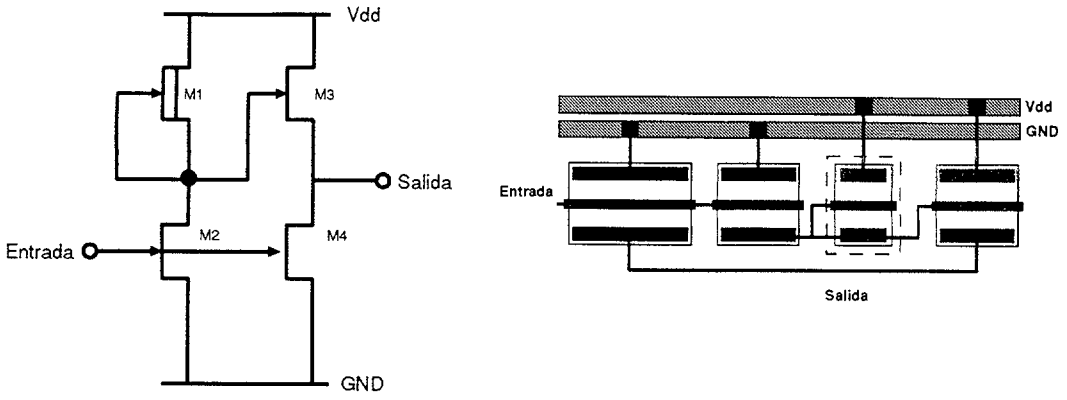


Figura 2.22: Inversor SBFL

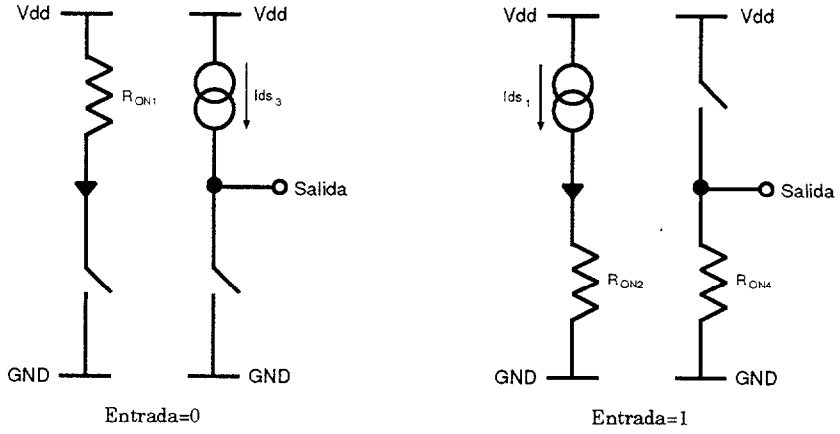


Figura 2.23: Circuito equivalente (a) con la entrada a nivel bajo y (b) con la entrada a nivel alto

por lo que la salida de la etapa DCFL está a nivel bajo. Por otro lado, el transistor E–MESFET de *pull–down* comienza a descargar la capacidad de carga a la salida antes de que el inversor DCFL corte al dispositivo tipo E que constituye el *seguidor de fuente.* Por lo tanto, a la salida del inversor SBFL se obtiene un nivel bajo (0 lógico).

Consecuentemente, durante una transición nivel alto–nivel bajo, la salida de los dos transistores E–MESFET están activos durante un corto periodo de tiempo. Por lo tanto, durante cada transición positiva de la entrada se puede producir una caída de tensión momentánea en el bus de alimentación *Vdd.*

El inversor básico SBFL puede extenderse para formar funciones lógicas. Por ejemplo, en la figura 2.24 se presenta una NOR de 2 entradas SBFL, formada por una NOR de 2 entradas DCFL y un buffer.

Esta familia lógica tiene una mayor capacidad de carga, menor sensibilidad frente al *fan–out* y un mayor margen de ruido que las familias vistas anteriormente. Su mayor problema radica en su alto consumo de potencia y a los picos de corriente que produce en los buses de alimentación, lo que obliga a un especial cuidado en el diseño de los buses de alimentación.

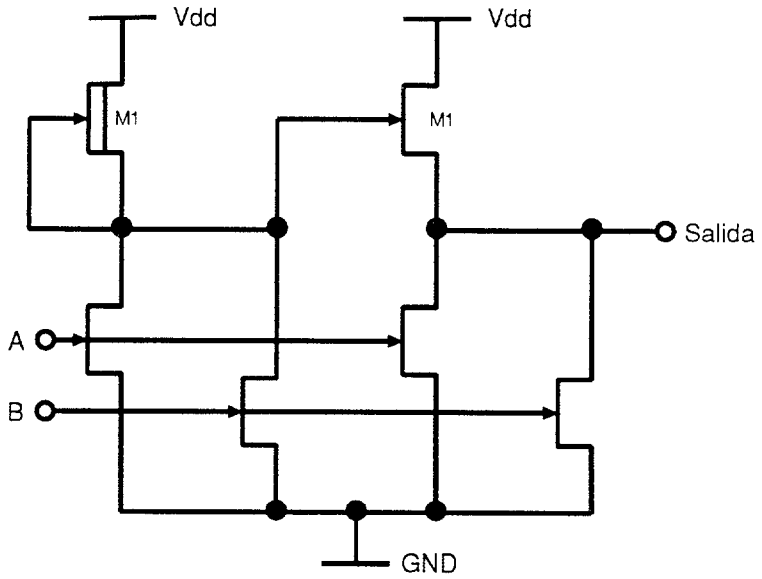


Figura 2.24: NOR de 2 entradas SBFL

2.4 Tecnología H–GaAs III

Esta tecnología, desarrollada por Vitesse como una tercera generación de procesos, caracterizada por una longitud efectiva de puerta de $0.6\mu m$, es una versión escalada de la inicialmente desarrollada en 1986. Fue diseñada para obtener circuitos con baja disipación de potencia, alta velocidad, y elevados niveles de integración (actualmente permite la integración de mas de un millón de dispositivos activos). Para tales efectos, se permiten 5 niveles de metalización, estando el cuarto y el quinto destinados a simplificar la distribución de señales de alimentación a lo largo de grandes circuitos.

2.4.1 Tensión umbral

Dos tipos de transistores están disponibles para la implementación de funciones lógicas: los "normally–off" o tipo E y los "normally–on" o tipo D. Como se comentó en secciones anteriores, estos dispositivos son idénticos, difiriendo tan sólo en la concentración de impurezas en la región del canal. El nivel de dopado determina el potencial electrostático en el canal del MESFET. Cuando la concentración de impurezas es baja, la zona de carga espacial se extiende a través del canal evitando que la corriente fluya de drenador a surtidor. La tensión que es necesario aplicar a la puerta para que que la corriente fluya es la tensión umbral (V_{te}). En un transistor de deplexión el canal es grande, y es necesario aplicar una tensión inversa para cortar el flujo de corriente. La tensión inversa necesaria que debe ser aplicada es la tensión umbral (V_{td}). En la figura 2.25 se pueden observar las curvas características de los dos transistores, haciendo especial referencia a las tensiones umbrales de los mismos. Es fácil observar cómo, para una tensión de puerta–surtidor menor que cero, el transistor tipo D permite ya un flujo de corriente entre drenador y surtidor, mientras que es necesario que dicha ten-

sión alcance los 0.2 voltios para que el transistor tipo E permita dicho flujo de corriente.

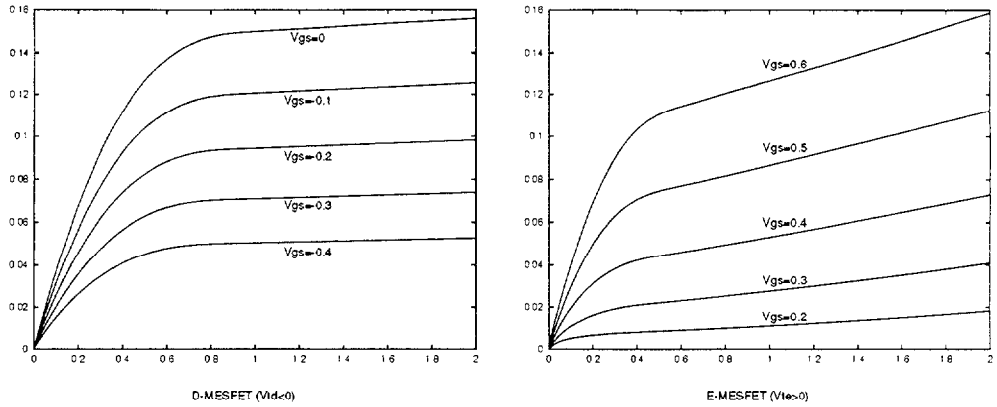


Figura 2.25: Curvas características de los transistores MESFET

2.4.2 Dispersión del proceso de fabricación

La tensión umbral de los transistores MESFET no es constante a lo largo de una oblea, ni entre distintas obleas. La variación en la tensión umbral se debe a cambios de la concentración de impurezas en el canal, variaciones del voltaje "built–in" y cambios en la concentración efectiva de aceptores dentro del sustrato. Estas variaciones producen una distribución de los valores de V_t que pueden caracterizarse por un valor típico y un rango global. A partir de esta información se pueden construir modelos de dispositivos en el peor caso para su uso en la simulación de circuitos.

El diseño de circuitos complejos requiere tolerancia a variaciones paramétricas para obtener *yields* aceptables. Los valores de dispersión se obtienen y se van completando con datos estadísticos obtenidos de circuitos fabricados. Las variaciones de V_t, tanto para el transistor de enriquecimiento como el de deplexión, usando el proceso de H–GaAs III, se pueden observar en la figura 2.26.

Los valores extremos de la tensión umbral, para los transistores E–MESFET y D–MESFET, se denominan *punto de proceso "fast"*, ya que los transistores conducen más corriente para una tensión puerta–surtidor determinada, y *punto de proceso "slow"*, donde se produce la situación contraria. El proceso tecnológico usado consta tanto de transistores E–MESFET como de transistores D–MESFET, con la característica de que los segundos son obtenidos a partir de los primeros mediante una implantación que aumenta el ancho del canal. Es por ello por lo que los procesos se denominarán "slow–slow" o "fast–fast". Nunca se darán procesos "fast–slow" o "slow–fast" debido a la analogía existente entre los dos tipos de transistores. El punto nominal (*punto de proceso "típico"*) se obtiene cuando todos los parámetros adquieren sus valores medios.

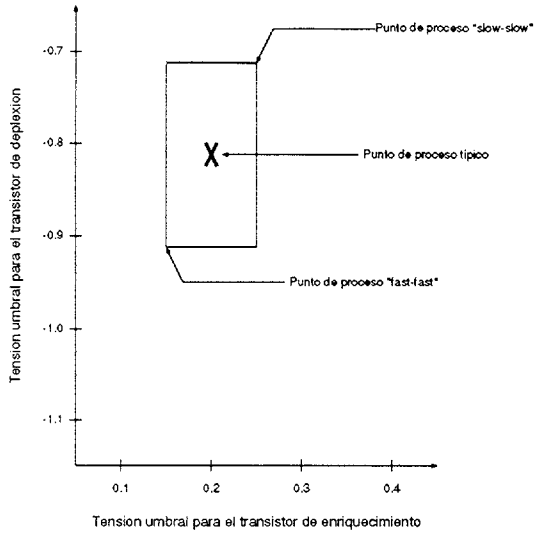


Figura 2.26: Variación de la tensión umbral

2.4.3 Niveles de metalización

Como se ha dicho anteriormente, Vitesse proporciona cinco niveles de interconexión: metal de puerta, metal 1, metal 2, metal 3 y metal 4. Los tres primeros son utilizados principalmente para ruteado de señales, mientras que los metales 3 y 4 se emplean para buses de tensión y tierra. Las interconexiones mediante el metal de puerta quedan restringidas a distancias cortas, debido a la alta resistencia de este material.

Los distintos niveles de metalización llevan asociados unos valores de capacidades y resistencias. En la tabla 2.1 se presentan estos valores, si bien en el caso de las capacidades sólo se exponen aquellas existentes entre la metalización y el sustrato, aunque éstas no son las únicas que hay que tener en cuenta a la hora de realizar la extracción de un *layout*, puesto que también influyen, y de forma más pronunciada, las capacidades entre metales, así como aquellas debidas a los bordes de las metalizaciones, encontrando que la capacidad total viene dada por la expresión

$$C_{TOT} = C_{pp} + 2 \cdot C_f$$

donde C_{pp} es la capacidad entre los niveles de metalización (o entre metalización y sustrato) y C_f es la capacidad debida a los bordes.

	$R_s\,(\Omega/\square)$	$I_{max}\,(mA/\mu m)$	$C_{pp}\,(fF/\mu m^2)$	$C_f\,(fF/\mu m^2)$
Metal de puerta	1	5.0	0.1024	0.061
Metal 1	< 0.07	1.0	0.0404	0.036
Metal 2	< 0.035	2.0	0.0194	0.023
Metal 3	< 0.025	2.8	0.0109	0.014
Metal 4	< 0.025	2.8	0.0076	0

Tabla 2.1: Características de los niveles de metalización

Capítulo 3

Transformada discreta del coseno y aritmética distribuida: Un estudio arquitectural

3.1 Transformada Discreta del Coseno

La *Transformada Discreta del Coseno en dos Dimensiones* (2–D DCT) de una matriz de datos reales {x(i,j): i,j=0,1,...,N-1} es una matriz, también real, de dimensión $N \times N$, cuyos elementos vienen dados por la siguiente ecuación:

$$y(m,n) = \frac{2}{N} \cdot u(m) \cdot u(n) \cdot \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} x(i,j) \cdot \cos\left(\frac{(2i+1)m\pi}{2N}\right) \cdot \cos\left(\frac{(2j+1)n\pi}{2N}\right)$$

donde los factores de escalado, $u(m)$ y $u(n)$, se definen como:

$$u(m), u(n) = \begin{cases} \frac{1}{\sqrt{2}}; & m,n = 0 \\ 1; & resto \end{cases}$$

La ecuación anterior equivale al producto de tres matrices de dimensión $N \times N$:

$$[Y] = [C] \cdot [X] \cdot [C]^t \cdot$$

donde $[X]$ es la matriz de entrada, $[C]$ es la matriz de coeficientes cosenos, e $[Y]$ es la matriz transformada resultante.

Una importante propiedad de la 2–D DCT es su separabilidad. Es posible calcular la 2–D DCT de una matriz de dimensión $N \times N$ llevando a cabo N 1–D DCTs (*Transformada Discreta del Coseno en una Dimensión*) sobre las columnas de la matriz de entrada y N 1–D DCTs sobre las filas de la matriz resultante. Este método se representa en la figura 3.1.

La 1–D DCT de una secuencia de datos reales {x(i): i=0,1,...,N-1} es un vector, también real, de longitud N, cuyos elementos vienen dados por la siguiente ecuación:

$$z(m) = \sqrt{\frac{2}{N}} \cdot u(m) \cdot \sum_{i=0}^{N-1} x(i) \cdot \cos\left(\frac{(2i+1)m\pi}{2N}\right)$$

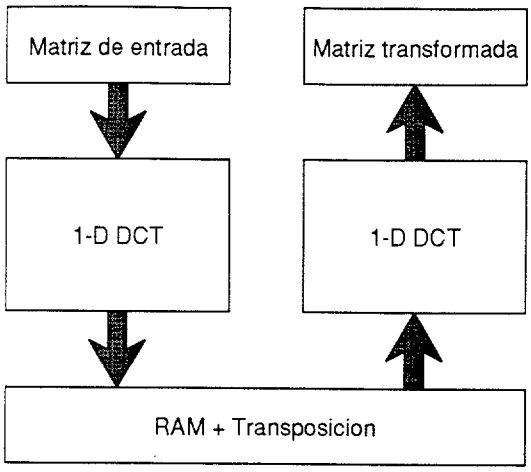


Figura 3.1: Separación de la 2–D DCT

donde el factor de escalado, $u(m)$, se define como:

$$u(m) = \begin{cases} \frac{1}{\sqrt{2}}; & m = 0 \\ 1; & resto \end{cases}$$

Cada elemento $z(m)$ se obtiene mediante el producto del vector de entrada y un *vector de base*, que corresponde a una función senoidal de una determinada frecuencia, mayor cuanto mayor sea el índice del elemento a calcular. En la figura 3.2 se puede observar la representación de esos vectores para una DCT de longitud 8.

La ecuación de la 1–D DCT equivale, por tanto, a un producto matricial:

$$[Z] = [C] \cdot [X]$$

donde $[X]$ es la matriz de entrada, $[C]$ es la matriz de coeficientes cosenos, y $[Z]$ es la matriz transformada resultante.

Sustituyendo esta ecuación en la de la 2–D DCT:

$$[Y] = [Z] \cdot [C]^t$$

y usando las propiedades de las matrices:

$$[Y]^t = [C] \cdot [Z]^t$$

se obtiene la matriz resultante de la 2–D DCT traspuesta.

La división de la 2–D DCT en dos 1–D DCT reduce el coste computacional, pasando del producto de tres matrices de dimensión $N \times N$ a dos productos de 2 matrices de la misma dimensión.

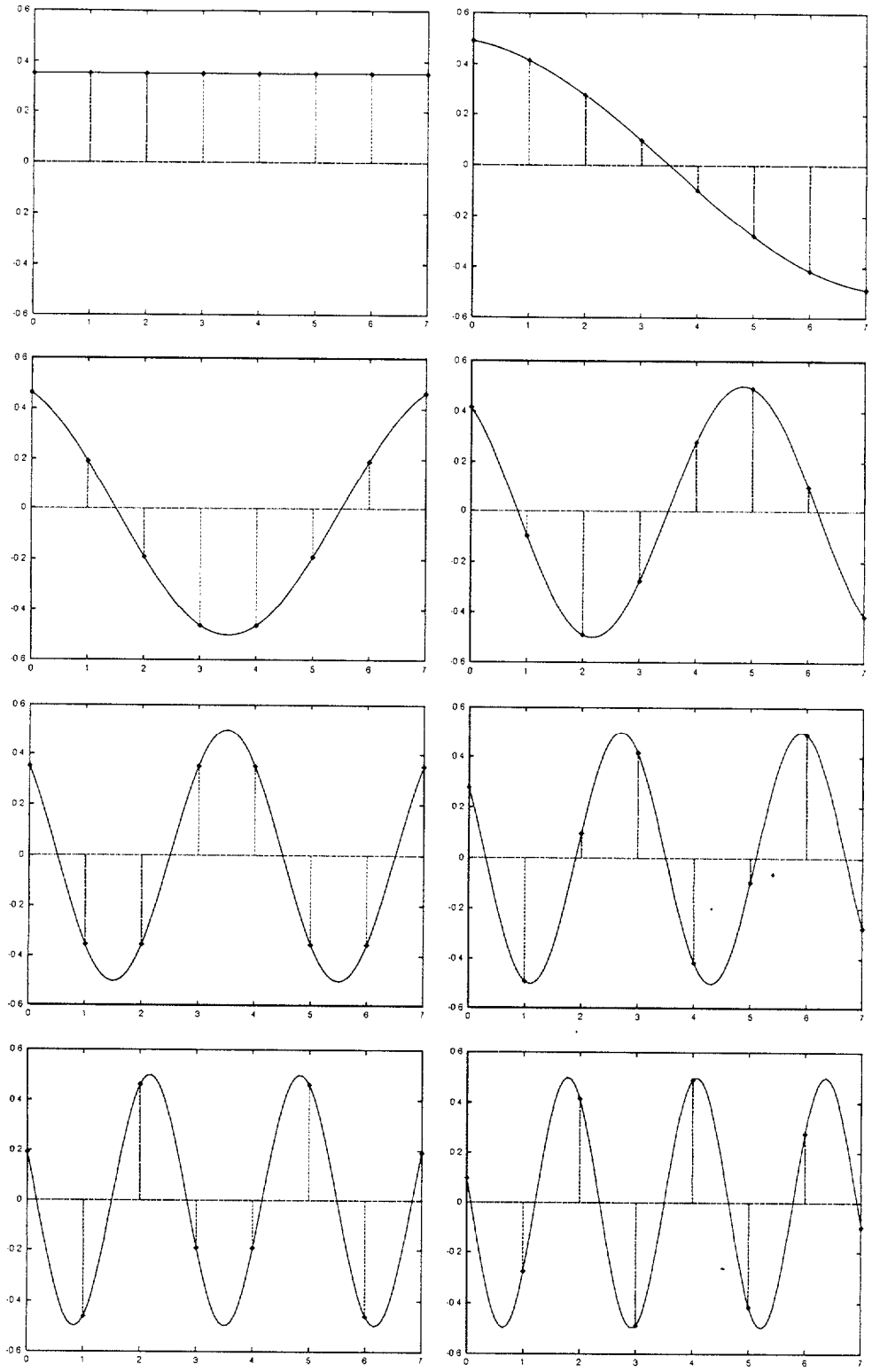


Figura 3.2: Vectores base para el cómputo de la 1–D DCT

3.1.1 Clasificación de los algoritmos

Los algoritmos para el cálculo de la transformada bidimensional basados en la descomposición en transformadas unidimensionales se pueden clasificar en:

- MMM (Matrix–Matrix Multiplication)
- FCT (Fast Cosine Transform)
- FCT–MMM

3.1.1.1 MMM

La interpretación matricial de la ecuación que define la 1–D DCT, con una longitud 8, se muestra en la siguiente ecuación.

$$\begin{bmatrix} z_0 \\ z_1 \\ z_2 \\ z_3 \\ z_4 \\ z_5 \\ z_6 \\ z_7 \end{bmatrix} = \frac{1}{2} \cdot \begin{bmatrix} c_4 & c_4 & c_4 & c_4 & c_4 & c_4 & c_4 & c_4 \\ c_1 & c_3 & c_5 & c_7 & -c_7 & -c_5 & -c_3 & -c_1 \\ c_2 & c_6 & -c_6 & -c_2 & -c_2 & -c_6 & c_6 & c_2 \\ c_3 & -c_7 & -c_1 & -c_5 & c_5 & c_1 & c_7 & -c_3 \\ c_4 & -c_4 & -c_4 & c_4 & c_4 & -c_4 & -c_4 & c_4 \\ c_5 & -c_1 & c_7 & c_3 & -c_3 & -c_7 & c_1 & -c_5 \\ c_6 & -c_2 & c_2 & -c_6 & -c_6 & c_2 & -c_2 & c_6 \\ c_7 & -c_5 & c_3 & -c_1 & c_1 & -c_3 & c_5 & -c_7 \end{bmatrix} \cdot \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix}$$

donde

$$c_i = \cos\left(\frac{i \cdot \pi}{16}\right)$$

Una operación similar se lleva a cabo sobre las filas de la matriz intermedia $[Z]$, obteniéndose así la matriz transformada $[Y]$. Cada elemento de la matriz $[Z]$ se puede obtener por medio de una operación de multiplicación y acumulación (MAC).

El mapeado directo de esta expresión matricial conlleva un flujo de datos muy regular, pero por contra, implica un gran número de operaciones.

3.1.1.2 FCT

Los algoritmos FCT se basan en la implementación de la DCT mediante la estructura en celosía que se muestra en la figura 3.3. La estructura más típica y utilizada en las implementaciones basadas en este algoritmo se muestra en la figura 3.4, donde también se puede observar la matriz que resulta al desarrollar dicha estructura.

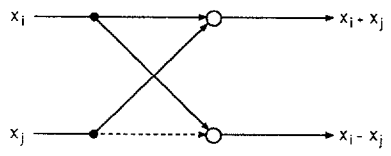


Figura 3.3: Estructura básica para la implementación FCT

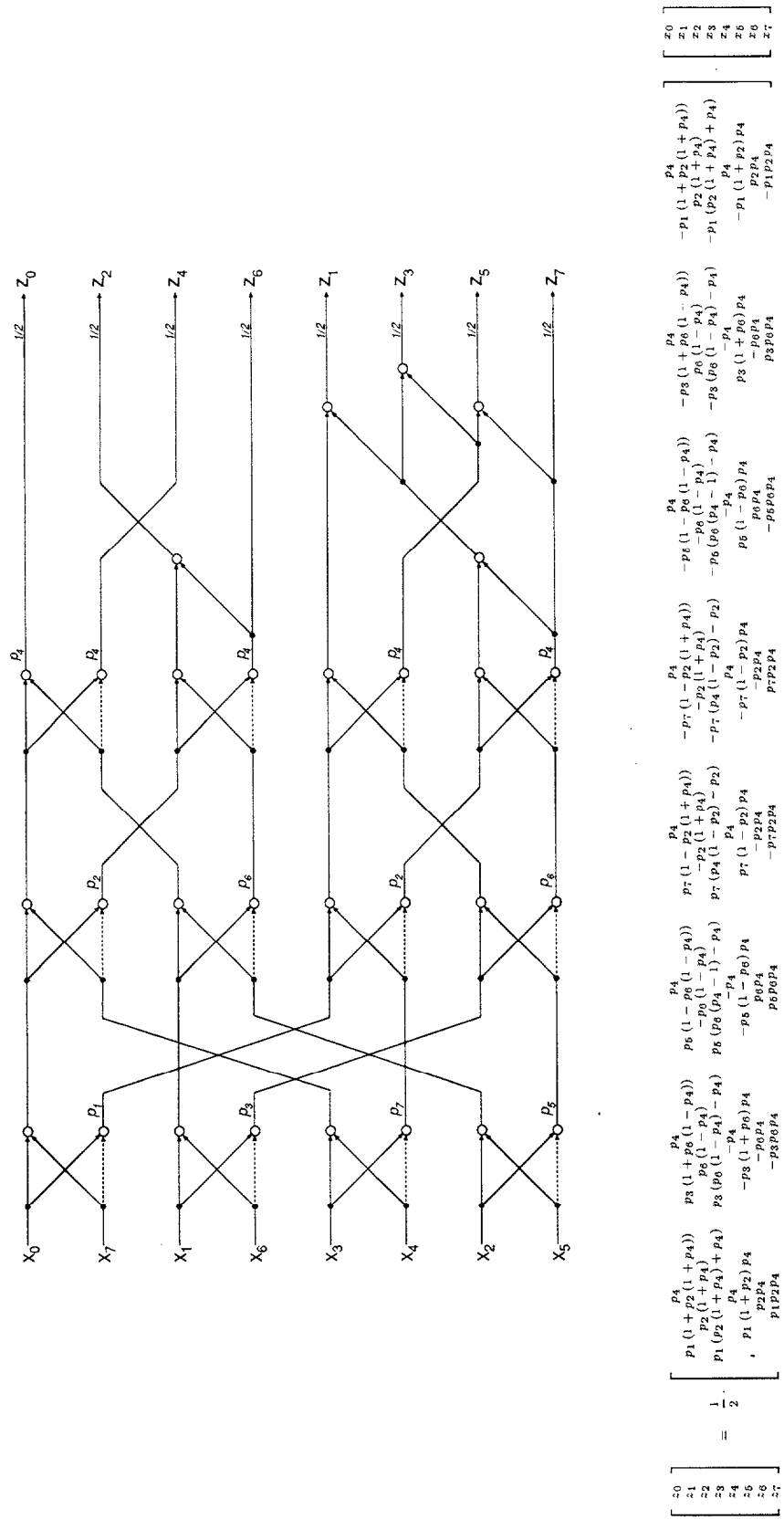


Figura 3.4: Implementación de la DCT mediante el algoritmo FCT

El cálculo de la DCT mediante el algoritmo FCT reduce el número de operaciones a realizar, aumentando drásticamente la complejidad del ruteado, por lo que no es apropiado para implementaciones VLSI.

3.1.1.3 FCT–MMM

Este algoritmo combina los dos anteriores, reduciendo por un lado el número de multiplicaciones necesarias, incluyendo un primer nivel de FCT, y manteniendo la regularidad en el flujo de datos que proporciona el algoritmo MMM.

La ecuación que define la DCT, en base a las propiedades trigonométricas de los cosenos, se puede dividir fácilmente según el índice del elemento a calcular, tal y como ocurría en la primera descomposición de la implementación FCT.

La siguiente ecuación muestra la notación matricial que define una DCT de 8 puntos mediante el algoritmo FCT–MMM.

$$\begin{bmatrix} z_0 \\ z_2 \\ z_4 \\ z_6 \end{bmatrix} = \frac{1}{2} \cdot \begin{bmatrix} c_4 & c_4 & c_4 & c_4 \\ c_2 & c_6 & -c_6 & -c_2 \\ c_4 & -c_4 & -c_4 & c_4 \\ c_6 & -c_2 & c_2 & -c_6 \end{bmatrix} \cdot \begin{bmatrix} x_0 + x_7 \\ x_1 + x_6 \\ x_2 + x_5 \\ x_3 + x_4 \end{bmatrix}$$

$$\begin{bmatrix} z_1 \\ z_3 \\ z_5 \\ z_7 \end{bmatrix} = \frac{1}{2} \cdot \begin{bmatrix} c_1 & c_3 & c_5 & c_7 \\ c_3 & -c_7 & -c_1 & -c_5 \\ c_5 & -c_1 & c_7 & c_3 \\ c_7 & -c_5 & c_3 & -c_1 \end{bmatrix} \cdot \begin{bmatrix} x_0 - x_7 \\ x_1 - x_6 \\ x_2 - x_5 \\ x_3 - x_4 \end{bmatrix}$$

La 1–D DCT de 8 puntos se descompone en dos productos de dos matrices cada uno, que se realizan independientemente. Uno de los productos calcula los elementos de índice par, para lo cual los datos de entrada a la estructura MAC se calculan sumando elementos del vector de entrada, mientras el otro producto matricial calcula los elementos de índice impar, para lo cual los datos de entrada a la estructura MAC se calculan restando elementos del vector de entrada. Es fácil observar que la matriz de coeficientes será distinta, según sea par o impar el índice del elemento a calcular.

La estructura característica de este algoritmo viene reflejada en la figura 3.5. Con este algoritmo se reduce el numero de operaciones MAC con respecto al algoritmo MMM, manteniendo un flujo de datos muy regular, motivo por el que ha sido elegido como base para la realización de este proyecto.

3.2 Estudio arquitectural

Como se pudo comprobar en el apartado anterior, la división de la 2–D DCT en dos transformadas unidimensionales, actuando la salida traspuesta de la primera como entrada de la segunda, implica un importante ahorro en carga computacional.

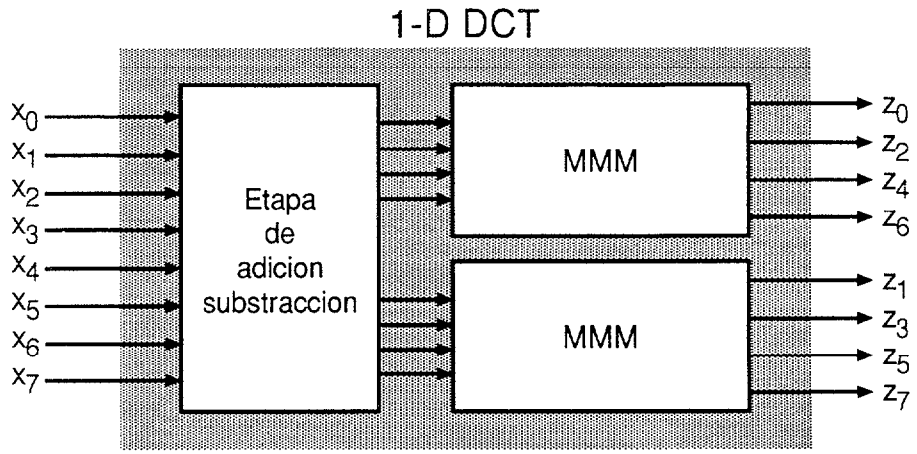


Figura 3.5: Arquitectura FCT-MMM

Por otro lado, la implementación de cada transformada unidimensional según el algoritmo FCT-MMM conlleva un diseño regular y un ruteado simple. Si la implementación de los multiplicadores–acumuladores se basa en aritmética distribuida, se produce, además, una importante disminución en el consumo de potencia con respecto a la implementación con multiplicadores convencionales.

En este apartado se realizará un estudio sobre la *aritmética distribuida* y las posibles *arquitecturas* de la 1–D DCT en base a dicha aritmética, siempre dentro del algoritmo FCT–MMM, dando así un abanico de posibilidades a la hora realizar el diseño final.

3.2.1 Diagrama de bloques

El diagrama de bloques del circuito, donde se reflejan los elementos necesarios para el cómputo de la 2–D DCT se muestra en la figura 3.6.

La primera operación a realizar sobre los datos de llegada es una conversión paralelo/serie, ya que si bien los datos de entrada al circuito son las palabras que forman las columnas de una matriz, los de entrada a la etapa de pre–proceso son los bits que forman dichas palabras.

La conversión paralelo/serie de los datos de entrada se realizará con 2 bancos de registros, de forma que en el primero de ellos se vayan almacenando las palabras de la columna de entrada, según van llegando al circuito. Cuando toda la columna esté almacenada en el primer banco de registros, y comience la entrada de la siguiente columna, los datos almacenados en el primer banco de registros pasan al segundo, desde el que comienza la entrega serie de los datos.

Los datos, según se van serializando, son sumados y restados entre sí, *comenzando por los menos significativos* (esta afirmación quedará justificada en el estudio de la aritmética distribuida, apartado 3.2.2). El número de bits de los sumadores vendrá determinado por el número de bits que vayan a ser procesados en cada ciclo, o lo que es lo

mismo, por la implementación de la estructura MAC (Multiplicator and ACcumulator). En el caso de que se vaya a procesar un bit cada ciclo, los sumadores serán de un bit. Si el número de bits a procesar en un ciclo es 2, éste será también el número de bits de los sumadores. En definitiva, para procesar M bits, serán necesarios sumadores de M bits.

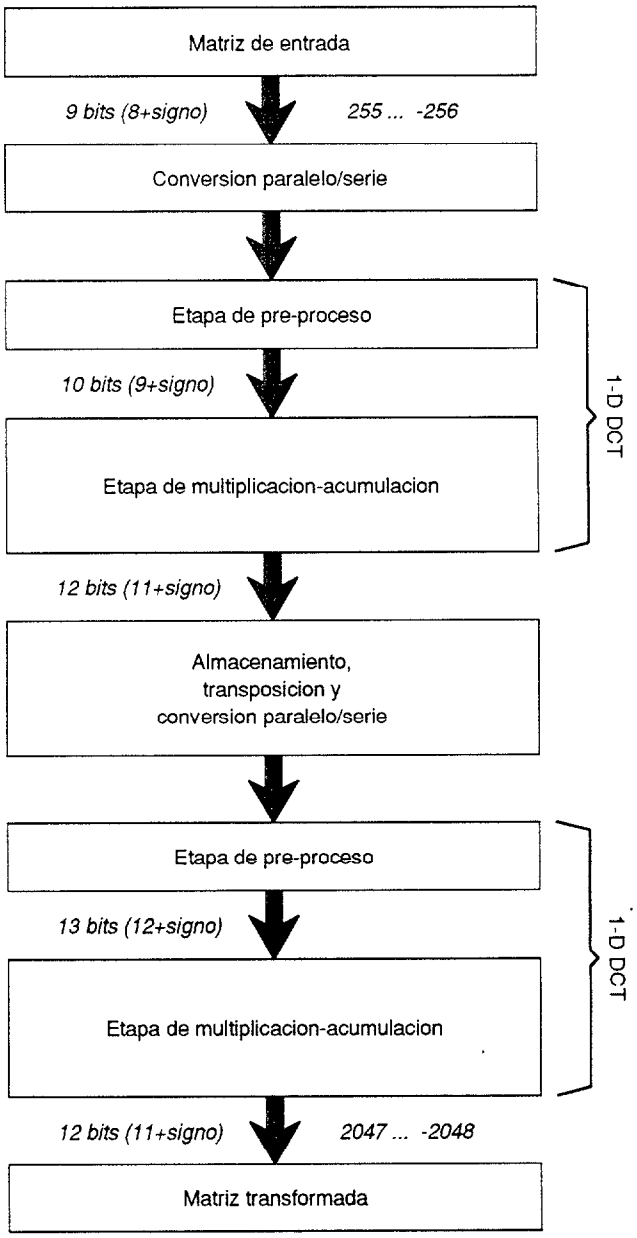


Figura 3.6: Estructura del circuito

La longitud de las palabras de entrada a la primera y a la segunda etapa de pre–procesamiento (antes y después del almacenamiento y la transposición) es de 9 y 12 bits, respectivamente. Para prevenir el caso de desbordamiento a la salida de los sumadores y restadores se añade un bit, con lo que la longitud de las palabras de entrada a las etapas de multiplicación–acumulación pasará a ser de 10 bits para la primera y 13 bits para la segunda.

A medida que se van sumando y restando los datos, los bits resultantes van a estructuras MAC donde, basándose en aritmética distribuida, se realizan las operaciones de multiplicación y acumulación.

El esquema de bloques de la transformada unidimensional (etapas de pre–proceso y de multiplicación–acumulación) se puede ver en la figura 3.7, en la que se observa como los datos, tras ser sumados y restados entre sí (etapa FCT), van a una estructura de elementos multiplicadores–acumuladores (etapa MMM). La tabla de coeficientes almacenada en la estructura MAC dependerá del índice del elemento a calcular.

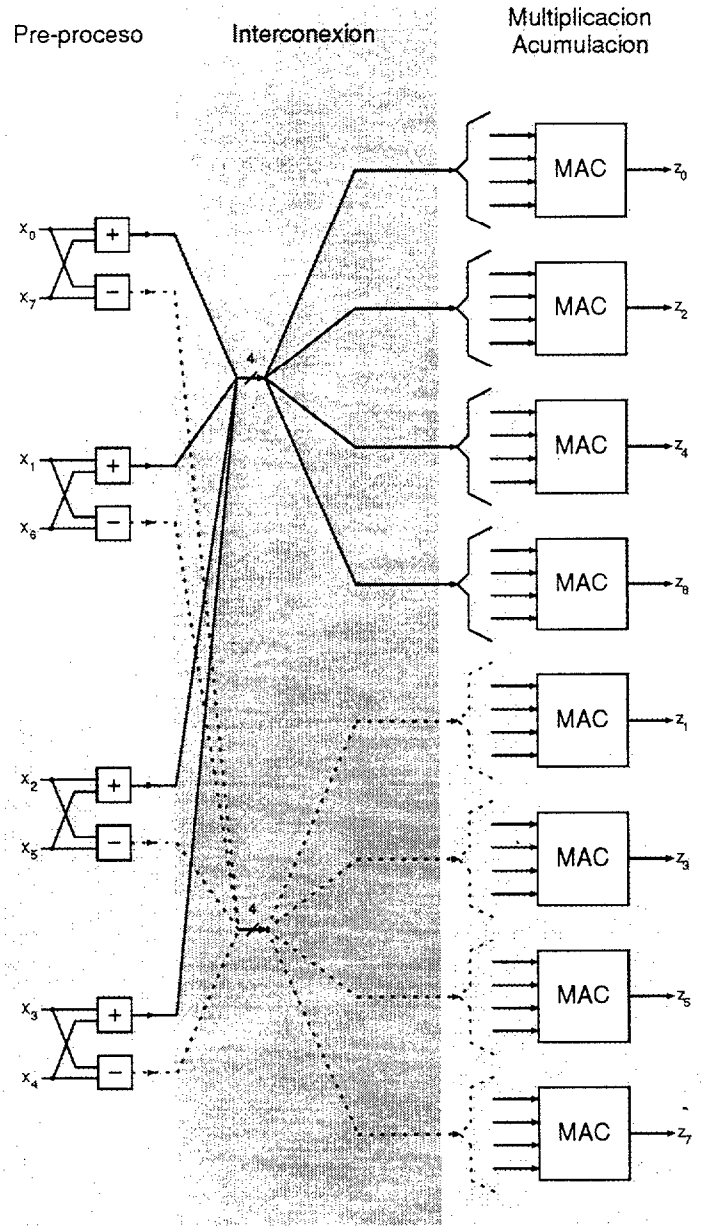


Figura 3.7: Esquema de la 1–D DCT basada en el algoritmo FCT–MMM

Una vez terminado el procesado de toda la matriz, los datos almacenados y transpuestos en la primera RAM, se van entregando a la segunda parte del circuito, donde

se aplica la segunda 1–D DCT, obteniendo de esta forma la matriz transformada.

Según se van leyendo y procesando datos de la primera RAM en la segunda etapa de transformación, se van escribiendo los que se van obteniendo a la salida de la primera 1–D DCT en una segunda RAM, de forma que cuando se haya completado el proceso de una primera transformación unidimensional de la segunda matriz, es esta segunda RAM la que es leída, pasándose de esta forma a escribir en la primera RAM, manteniendo así la segmentación del circuito.

Con todo esto, el diagrama final de bloques de la estructura quedará como se muestra en la figura 3.8.

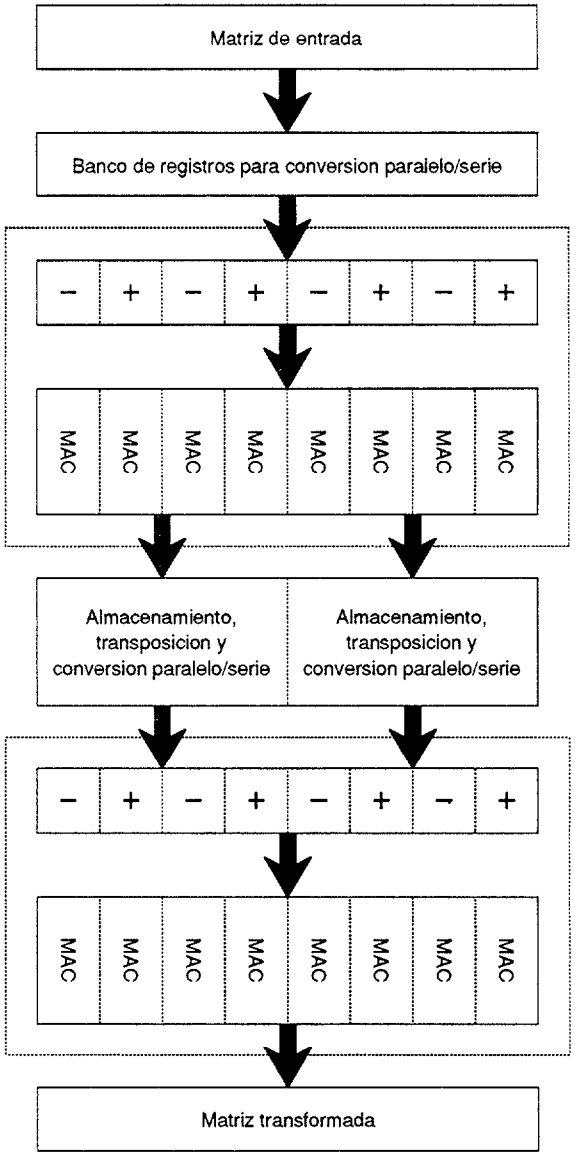


Figura 3.8: Diagrama de bloques de la 2–D DCT

3.2.2 Aritmética distribuida

Debido al número de transistores requerido para la realización de los multiplicadores–acumuladores (y por tanto al área que ocupan y a la potencia que disipan éstos), se ha optado por la implementación de las estructuras MAC usando aritmética distribuida.

Para ilustrar su funcionamiento, se parte de la siguiente operación, que implementa el producto de dos vectores de longitud M:

$$y = \sum_{k=0}^{M-1} A_k \cdot x_k$$

donde A_k son coeficientes conocidos (en el caso de la 1–D DCT son los coeficientes cosenos) y x_k son los elementos del vector de entrada.

Asumiendo que los datos de entrada vienen en complemento a 2, y que, por tanto, se pueden expresar como:

$$x_k = -b_{k0} + \sum_{n=1}^{N-1} b_{kn} \cdot 2^{-n}$$

siendo N el número de bits de cada elemento de x, b_{kn} el bit en la posición n del elemento x_k y b_{k0} el bit más significativo, y por tanto el bit de signo de x_k.

Combinando estas dos últimas ecuaciones, se obtiene:

$$y = \sum_{k=0}^{M-1} A_k \cdot \left(-b_{k0} + \sum_{n=1}^{N-1} b_{kn} \cdot 2^{-n} \right)$$

y si se intercambian las sumas:

$$y = \sum_{n=1}^{N-1} \left(\sum_{k=0}^{M-1} A_k \cdot b_{kn} \right) \cdot 2^{-n} + \left(\sum_{k=0}^{M-1} A_k \cdot (-b_{k0}) \right)$$

Considérese ahora el término:

$$\sum_{k=0}^{M-1} A_k \cdot b_{kn}$$

Como b_{kn} puede tomar valores 0 o 1, esta expresión solo puede tomar 2^M valores. En lugar de calcular estos valores, se pueden almacenar en una ROM. Los datos de entrada, ahora tomados de bit en bit (comenzando por el menos significativo) de cada elemento x_k, y que tiene M bits, direccionan la ROM. Estos valores leídos se pueden acumular usando un sumador. Tras N ciclos, a la salida del acumulador estará disponible el valor de y.

La memoria contiene todas las combinaciones de sumas entre elementos A_k y sus opuestos para incluir los términos del bit de signo. Consecuentemente, el tamaño final de la ROM es 2^{M+1}. En la figura 3.9(a) se puede ver el esquema de esta estructura,

donde se ha tomado $M = 4$ (ya que esa es la longitud de los vectores en la implementación de la DCT) y donde *bs* es una señal que toma el valor lógico 1 cuando se está procesando el bit de signo.

La relación entre la salida de la memoria y el código de la entrada, se puede observar en la siguiente tabla:

Código de entrada					*Salida de la ROM*
bs	b_{0n}	b_{1n}	b_{2n}	b_{3n}	
0	0	0	0	0	0
0	0	0	0	1	A_3
0	0	0	1	0	A_2
0	0	0	1	1	$A_2 + A_3$
0	0	1	0	0	A_1
0	0	1	0	1	$A_1 + A_3$
0	0	1	1	0	$A_1 + A_2$
0	0	1	1	1	$A_1 + A_2 + A_3$
0	1	0	0	0	A_0
0	1	0	0	1	$A_0 + A_3$
0	1	0	1	0	$A_0 + A_2$
0	1	0	1	1	$A_0 + A_2 + A_3$
0	1	1	0	0	$A_0 + A_1$
0	1	1	0	1	$A_0 + A_1 + A_3$
0	1	1	1	0	$A_0 + A_1 + A_2$
0	1	1	1	1	$A_0 + A_1 + A_2 + A_3$
1	0	0	0	0	0
1	0	0	0	1	$-A_3$
1	0	0	1	0	$-A_2$
1	0	0	1	1	$-(A_2 + A_3)$
1	0	1	0	0	$-A_1$
1	0	1	0	1	$-(A_1 + A_3)$
1	0	1	1	0	$-(A_1 + A_2)$
1	0	1	1	1	$-(A_1 + A_2 + A_3)$
1	1	0	0	0	$-A_0$
1	1	0	0	1	$-(A_0 + A_3)$
1	1	0	1	0	$-(A_0 + A_2)$
1	1	0	1	1	$-(A_0 + A_2 + A_3)$
1	1	1	0	0	$-(A_0 + A_1)$
1	1	1	0	1	$-(A_0 + A_1 + A_3)$
1	1	1	1	0	$-(A_0 + A_1 + A_2)$
1	1	1	1	1	$-(A_0 + A_1 + A_2 + A_3)$

El tamaño de la memoria se puede reducir a 2^M palabras cambiando el sumador que acumula los valores de la ROM por un sumador/restador y usando la señal *bs* como control de suma o resta. Esta estructura se puede ver en la figura 3.9(b). En ese caso, sólo es necesario almacenar la mitad de la tabla anterior.

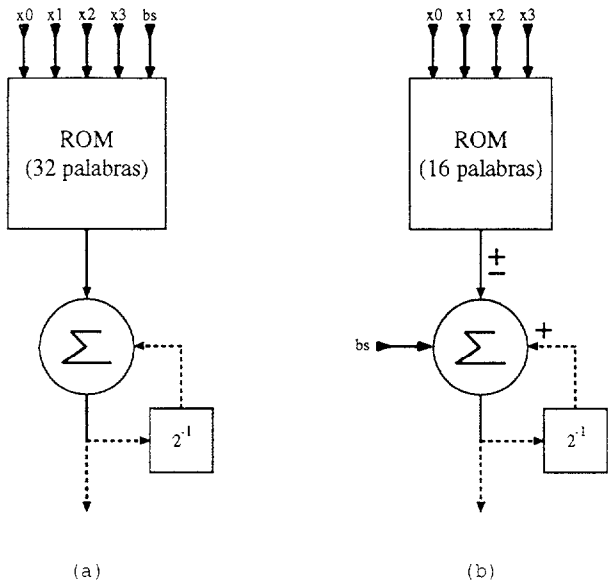


Figura 3.9: Implementación con una memoria de (a) 32 palabras (b) 16 palabras

Es posible reducir aún más el tamaño de la memoria, a 2^{M-1} palabras, si se usa una técnica conocida como *offset binary coding.*

Partiendo de la igualdad:

$$x_k = \frac{1}{2} \cdot (x_k - (-x_k))$$

y considerando de nuevo una representación en complemento a 2, se obtiene:

$$y = \sum_{k=0}^{M-1} \frac{A_k}{2} \cdot \left[\left(-b_{k0} + \sum_{n=1}^{N-1} b_{kn} \cdot 2^{-n} \right) - \left(-\overline{b_{k0}} + \sum_{n=1}^{N-1} \overline{b_{kn}} \cdot 2^{-n} + 2^{-(N-1)} \right) \right]$$

que se puede expresar como:

$$y = \sum_{k=0}^{M-1} \frac{A_k}{2} \cdot \left(-b_{k0} + \overline{b_{k0}} \right) + \sum_{n=1}^{N-1} \sum_{k=0}^{M-1} \frac{A_k}{2} \left(b_{kn} - \overline{b_{kn}} \right) \cdot 2^{-n} - \sum_{k=0}^{M-1} \frac{A_k}{2} \cdot 2^{-(N-1)}$$

Para simplificar la notación, la expresión de y tomará la forma:

$$y = \sum_{n=0}^{N-1} Q(b_n) \cdot 2^{-n} - Q(0) \cdot 2^{-(N-1)}$$

siendo

$$Q(b_n) = \sum_{k=0}^{M-1} \frac{A_k}{2} \cdot c_{kn}$$

$$Q(0) = \sum_{k=0}^{M-1} \frac{A_k}{2}$$

donde

$$c_{k0} = -\left(b_{k0} - \overline{b_{k0}}\right)$$

si n es 0 y

$$c_{kn} = b_{kn} - \overline{b_{kn}}$$

en el resto.

Teniendo en cuenta que $Q(b_n)$ solo puede tomar 2^{M-1} valores, este será, por tanto, el tamaño de la ROM. Es necesario la inclusión de un registro de condiciones iniciales, para almacenar el valor de $Q(0)$, así como la lógica de control adecuada para el sumador/restador.

La relación entre el código de entrada y el valor de $Q(b_n)$ viene reflejada en la tabla 3.1.

Para que en la implementación de la estructura MAC, el tamaño de la ROM sea de 8 palabras, el direccionamiento de la misma ha de obtenerse a partir de puertas XOR, según la relación:

$$\begin{aligned} D0 &= b_{0n} \oplus b_{1n} \\ D1 &= b_{0n} \oplus b_{2n} \\ D2 &= b_{0n} \oplus b_{3n} \\ Suma/resta &= b_{0n} \oplus bs \end{aligned}$$

siendo así posible obtener a la salida todas las combinaciones reflejadas en la tabla anterior.

En base a esa relación, los valores almacenados en la ROM serán:

D0	*D1*	*D2*	*Dato almacenado*
0	0	0	$-1/2 \cdot (A_0 + A_1 + A_2 + A_3)$
0	0	1	$-1/2 \cdot (A_0 + A_1 + A_2 - A_3)$
0	1	0	$-1/2 \cdot (A_0 + A_1 - A_2 + A_3)$
0	1	1	$-1/2 \cdot (A_0 + A_1 - A_2 - A_3)$
1	0	0	$-1/2 \cdot (A_0 - A_1 + A_2 + A_3)$
1	0	1	$-1/2 \cdot (A_0 - A_1 + A_2 - A_3)$
1	1	0	$-1/2 \cdot (A_0 - A_1 - A_2 + A_3)$
1	1	1	$-1/2 \cdot (A_0 - A_1 - A_2 - A_3)$

En la tabla 3.2 se muestra la relación entre el código de entrada, la dirección de la ROM y la operación a realizar por el sumador.

Código de entrada					$Q(b_n)$
bs	b_{0n}	b_{1n}	b_{2n}	b_{3n}	
0	0	0	0	0	$-1/2 \cdot (A_0 + A_1 + A_2 + A_3)$
0	0	0	0	1	$-1/2 \cdot (A_0 + A_1 + A_2 - A_3)$
0	0	0	1	0	$-1/2 \cdot (A_0 + A_1 - A_2 + A_3)$
0	0	0	1	1	$-1/2 \cdot (A_0 + A_1 - A_2 - A_3)$
0	0	1	0	0	$-1/2 \cdot (A_0 - A_1 + A_2 + A_3)$
0	0	1	0	1	$-1/2 \cdot (A_0 - A_1 + A_2 - A_3)$
0	0	1	1	0	$-1/2 \cdot (A_0 - A_1 - A_2 + A_3)$
0	0	1	1	1	$-1/2 \cdot (A_0 - A_1 - A_2 - A_3)$
0	1	0	0	0	$1/2 \cdot (A_0 - A_1 - A_2 - A_3)$
0	1	0	0	1	$1/2 \cdot (A_0 - A_1 - A_2 + A_3)$
0	1	0	1	0	$1/2 \cdot (A_0 - A_1 + A_2 - A_3)$
0	1	0	1	1	$1/2 \cdot (A_0 - A_1 + A_2 + A_3)$
0	1	1	0	0	$1/2 \cdot (A_0 + A_1 - A_2 - A_3)$
0	1	1	0	1	$1/2 \cdot (A_0 + A_1 - A_2 + A_3)$
0	1	1	1	0	$1/2 \cdot (A_0 + A_1 + A_2 - A_3)$
0	1	1	1	1	$1/2 \cdot (A_0 + A_1 + A_2 + A_3)$
1	0	0	0	0	$1/2 \cdot (A_0 + A_1 + A_2 + A_3)$
1	0	0	0	1	$1/2 \cdot (A_0 + A_1 + A_2 - A_3)$
1	0	0	1	0	$1/2 \cdot (A_0 + A_1 - A_2 + A_3)$
1	0	0	1	1	$1/2 \cdot (A_0 + A_1 - A_2 - A_3)$
1	0	1	0	0	$1/2 \cdot (A_0 - A_1 + A_2 + A_3)$
1	0	1	0	1	$1/2 \cdot (A_0 - A_1 + A_2 - A_3)$
1	0	1	1	0	$1/2 \cdot (A_0 - A_1 - A_2 + A_3)$
1	0	1	1	1	$1/2 \cdot (A_0 - A_1 - A_2 - A_3)$
1	1	0	0	0	$-1/2 \cdot (A_0 - A_1 - A_2 - A_3)$
1	1	0	0	1	$-1/2 \cdot (A_0 - A_1 - A_2 + A_3)$
1	1	0	1	0	$-1/2 \cdot (A_0 - A_1 + A_2 - A_3)$
1	1	0	1	1	$-1/2 \cdot (A_0 - A_1 + A_2 + A_3)$
1	1	1	0	0	$-1/2 \cdot (A_0 + A_1 - A_2 - A_3)$
1	1	1	0	1	$-1/2 \cdot (A_0 + A_1 - A_2 + A_3)$
1	1	1	1	0	$-1/2 \cdot (A_0 + A_1 + A_2 - A_3)$
1	1	1	1	1	$-1/2 \cdot (A_0 + A_1 + A_2 + A_3)$

Tabla 3.1: Relación entre el código de entrada y $Q(b_n)$

Código de entrada					*Operación*	*Dirección de la ROM*		
bs	b_{0n}	b_{1n}	b_{2n}	b_{3n}	*del sumador*	*D0*	*D1*	*D2*
0	0	0	0	0	+	0	0	0
0	0	0	0	1	+	0	0	1
0	0	0	1	0	+	0	1	0
0	0	0	1	1	+	0	1	1
0	0	1	0	0	+	1	0	0
0	0	1	0	1	+	1	0	1
0	0	1	1	0	+	1	1	0
0	0	1	1	1	+	1	1	1
0	1	0	0	0	−	1	1	1
0	1	0	0	1	−	1	1	0
0	1	0	1	0	−	1	0	1
0	1	0	1	1	−	1	0	0
0	1	1	0	0	−	0	1	1
0	1	1	0	1	−	0	1	0
0	1	1	1	0	−	0	0	1
0	1	1	1	1	−	0	0	0
1	0	0	0	0	−	0	0	0
1	0	0	0	1	−	0	0	1
1	0	0	1	0	−	0	1	0
1	0	0	1	1	−	0	1	1
1	0	1	0	0	−	1	0	0
1	0	1	0	1	−	1	0	1
1	0	1	1	0	−	1	1	0
1	0	1	1	1	−	1	1	1
1	1	0	0	0	+	1	1	1
1	1	0	0	1	+	1	1	0
1	1	0	1	0	+	1	0	1
1	1	0	1	1	+	1	0	0
1	1	1	0	0	+	0	1	1
1	1	1	0	1	+	0	1	0
1	1	1	1	0	+	0	0	1
1	1	1	1	1	+	0	0	0

Tabla 3.2: Relación entre el código de entrada, la dirección de la ROM y la operación a realizar por el sumador

La implementación final de la estructura MAC en aritmética distribuida se representa en la figura 3.10(a). Sin embargo, tras estudios realizados se ha comprobado que la inclusión de la lógica necesaria para utilizar un sumador/restador en esta estructura (es necesario realizar el complemento a 1 de los datos de la ROM si éstos van a ser restados) resulta menos eficiente que la duplicación del tamaño de la ROM, pasando de 8 a 16 palabras.

Para que en la implementación de la estructura MAC, el tamaño de la ROM sea de 16 palabras, el direccionamiento de la misma ha de obtenerse, al igual que en el caso anterior, a partir de puertas XOR, según la relación:

$$\begin{aligned} D0 &= b_{0n} \oplus b_{1n} \\ D1 &= b_{0n} \oplus b_{2n} \\ D2 &= b_{0n} \oplus b_{3n} \\ D3 &= b_{0n} \oplus bs \end{aligned}$$

siendo así posible obtener a la salida todas las combinaciones reflejadas en la tabla anterior.

En base a esa relación, los valores almacenados en la ROM serán:

D0	*D1*	*D2*	*D3*	*Dato almacenado*
0	0	0	0	$-1/2 \cdot (A_0 + A_1 + A_2 + A_3)$
0	0	0	1	$1/2 \cdot (A_0 + A_1 + A_2 + A_3)$
0	0	1	0	$-1/2 \cdot (A_0 + A_1 + A_2 - A_3)$
0	0	1	1	$1/2 \cdot (A_0 + A_1 + A_2 - A_3)$
0	1	0	0	$-1/2 \cdot (A_0 + A_1 - A_2 + A_3)$
0	1	0	1	$1/2 \cdot (A_0 + A_1 - A_2 + A_3)$
0	1	1	0	$-1/2 \cdot (A_0 + A_1 - A_2 - A_3)$
0	1	1	1	$1/2 \cdot (A_0 + A_1 - A_2 - A_3)$
1	0	0	0	$-1/2 \cdot (A_0 - A_1 + A_2 + A_3)$
1	0	0	1	$1/2 \cdot (A_0 - A_1 + A_2 + A_3)$
1	0	1	0	$-1/2 \cdot (A_0 - A_1 + A_2 - A_3)$
1	0	1	1	$1/2 \cdot (A_0 - A_1 + A_2 - A_3)$
1	1	0	0	$-1/2 \cdot (A_0 - A_1 - A_2 + A_3)$
1	1	0	1	$1/2 \cdot (A_0 - A_1 - A_2 + A_3)$
1	1	1	0	$-1/2 \cdot (A_0 - A_1 - A_2 - A_3)$
1	1	1	1	$1/2 \cdot (A_0 - A_1 - A_2 - A_3)$

En la tabla 3.3 se muestra la salida de la ROM, en relación con el código de entrada.

La implementación de la estructura MAC elegida se basa, por tanto, en una ROM de 16 palabras y un sumador, y se puede ver en la figura 3.10(b).

Código de entrada					*Dirección de la ROM*			
bs	b_{0n}	b_{1n}	b_{2n}	b_{3n}	$D0$	$D1$	$D2$	$D3$
0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	1	0
0	0	0	1	0	0	1	0	0
0	0	0	1	1	0	1	1	0
0	0	1	0	0	1	0	0	0
0	0	1	0	1	1	0	1	0
0	0	1	1	0	1	1	0	0
0	0	1	1	1	1	1	1	0
0	1	0	0	0	1	1	1	1
0	1	0	0	1	1	1	0	1
0	1	0	1	0	1	0	1	1
0	1	0	1	1	1	0	0	1
0	1	1	0	0	0	1	1	1
0	1	1	0	1	0	1	0	1
0	1	1	1	0	0	0	1	1
0	1	1	1	1	0	0	0	1
1	0	0	0	0	0	0	0	1
1	0	0	0	1	0	0	1	1
1	0	0	1	0	0	1	0	1
1	0	0	1	1	0	1	1	1
1	0	1	0	0	1	0	0	1
1	0	1	0	1	1	0	1	1
1	0	1	1	0	1	1	0	1
1	0	1	1	1	1	1	1	1
1	1	0	0	0	1	1	1	0
1	1	0	0	1	1	1	0	0
1	1	0	1	0	1	0	1	0
1	1	0	1	1	1	0	0	0
1	1	1	0	0	0	1	1	0
1	1	1	0	1	0	1	0	0
1	1	1	1	0	0	0	1	0
1	1	1	1	1	0	0	0	0

Tabla 3.3: Salida de la ROM en relación con el código de entrada

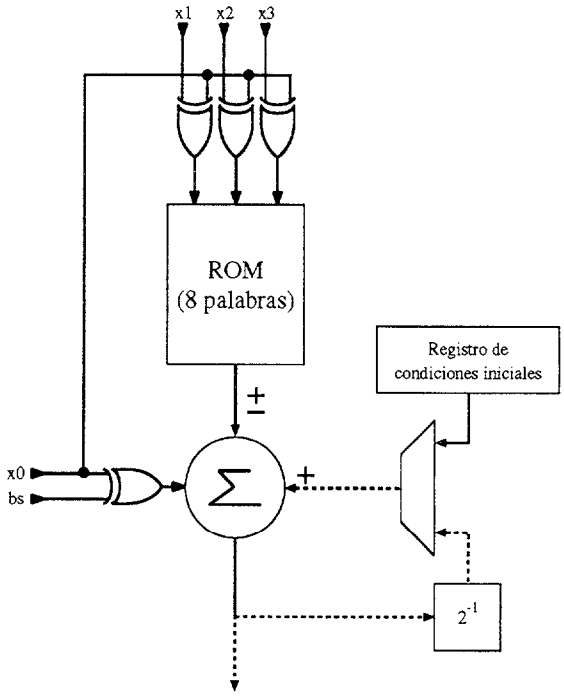


Figura 3.10: Implementación con una memoria de (a) 8 palabras y (b) 16 palabras

3.2.3 Estructura del multiplicador–acumulador basado en aritmética distribuida

La estructura MAC condicionará la frecuencia de trabajo del circuito, así como el *throughput* del mismo. Es, por tanto, una de las partes más importantes del diseño.

La estructura del multiplicador–acumulador se estudiará para la arquitectura de la DCT presentada en el apartado 3.2. Esta arquitectura, segmentada pero no paralela, puede ser paralelizada, con lo que la unidad MAC puede sufrir variaciones mas o menos importantes. En este apartado se estudiará, también, el efecto del aumento del grado de paralelismo sobre la estructura MAC. De esta forma se dispondrá de una variedad de posibilidades que se resolverán en función de las prestaciones finales que se quieran obtener.

En la implementación más sencilla de una estructura MAC se incluye una sola ROM, lo que implica que se procesará un único bit de las palabras de entrada en cada ciclo. Obviamente, multiplicando el número de ROMs por n, es decir, realizando una partición n, se obtendrá una arquitectura capaz de trabajar con n bits en cada ciclo.

Para direccionar la ROM, los bits que forman las palabras que han sido pre–procesadas, $x0$, $x1$, $x2$ y $x3$, pasan por una puerta OR exclusiva, conectados como se indica en la figura 3.11, donde bs es una señal que se activa cuando se está procesando el bit de signo.

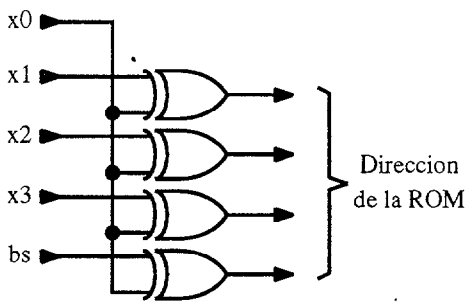


Figura 3.11: Generación de la dirección de la ROM

La implementación básica de una estructura MAC usando aritmética distribuida se muestra en la figura 3.12. El contenido del registro de condiciones iniciales coincide con el almacenado en la posición 0000 de la ROM.

En la tabla 3.13(a) se puede observar el flujo de datos que se produce en esta estructura (CI indica el valor del registro de condiciones iniciales, $[ROM]_i$ indica la salida de la ROM para el dato de entrada i, $RAM = Ri$ indica el ciclo en el que se almacena el dato en la RAM y $SAL = Ri$ indica el ciclo en el que la salida está disponible), para una palabra de entrada de 10 bits, mientras que en la tabla 3.13(b) se muestra el flujo de datos cuando dicha palabra de entrada es de 13 bits. Estas longitudes de palabra corresponden a las procesadas por la primera y segunda etapa de transformación unidimensional (previa y posterior al elemento de almacenamiento)

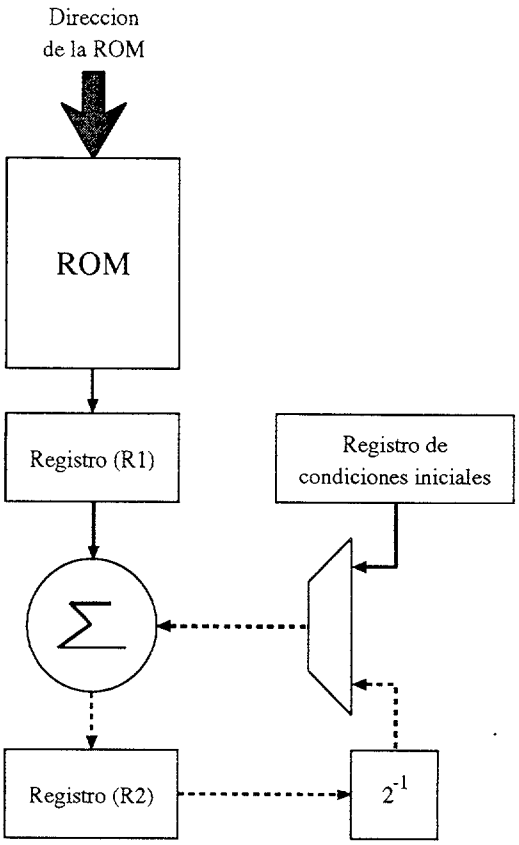


Figura 3.12: Implementación básica de la estructura MAC

respectivamente.

Para el caso de una palabra de 10 bits, la secuencia de operaciones del multiplicador–acumulador se puede expresar como:

$$Salida_{MAC} = \frac{CI + [ROM]_1}{2^9} + \frac{[ROM]_2}{2^8} + \frac{[ROM]_3}{2^7} + \frac{[ROM]_4}{2^6} + \frac{[ROM]_5}{2^5} + \\ + \frac{[ROM]_6}{2^4} + \frac{[ROM]_7}{2^3} + \frac{[ROM]_8}{2^2} + \frac{[ROM]_9}{2} + [ROM]_{10}$$

mientras que para una palabra de 13 bits, dicha secuencia es:

$$Salida_{MAC} = \frac{CI + [ROM]_1}{2^{12}} + \frac{[ROM]_2}{2^{11}} + \frac{[ROM]_3}{2^{10}} + \frac{[ROM]_4}{2^9} + \frac{[ROM]_5}{2^8} + \\ + \frac{[ROM]_6}{2^7} + \frac{[ROM]_7}{2^6} + \frac{[ROM]_8}{2^5} + \frac{[ROM]_9}{2^4} + \frac{[ROM]_{10}}{2^3} + \\ + \frac{[ROM]_{11}}{2^2} + \frac{[ROM]_{12}}{2} + [ROM]_{13}$$

3.2.3.1 Aumento del grado de paralelismo

Realizando una primera partición, se obtendrá una estructura capaz de procesar dos bits de las palabras de entrada en cada ciclo de reloj, duplicando así el *throughput* de la estructura original. En la figura 3.14 se muestra el diagrama de bloques de dicha

T_1	$R1 = [ROM]_1$
T_2	$R2 = R1 + CI\ ;\ R1 = [ROM]_2$
T_3	$R2 = R1 + \frac{R2}{2}\ ;\ R1 = [ROM]_3$
T_4	$R2 = R1 + \frac{R2}{2}\ ;\ R1 = [ROM]_4$
$\vdots$	$\vdots$
T_{10}	$R2 = R1 + \frac{R2}{2}\ ;\ R1 = [ROM]_{10}$
T_{11}	$R2 = R1 + \frac{R2}{2}\ ;\ R1 = [ROM]_1$
T_{12}	$RAM = R2$
	$R2 = R1 + CI\ ;\ R1 = [ROM]_2$
T_{13}	$R2 = R1 + \frac{R2}{2}\ ;\ R1 = [ROM]_3$
$\vdots$	$\vdots$
T_{21}	$R2 = R1 + \frac{R2}{2}\ ;\ R1 = [ROM]_1$
T_{22}	$RAM = R2$
	$R2 = R1 + CI\ ;\ R1 = [ROM]_2$
T_{23}	$R2 = R1 + \frac{R2}{2}\ ;\ R1 = [ROM]_3$
$\vdots$	$\vdots$

(a)

T_1	$R1 = [ROM]_1$
T_2	$R2 = R1 + CI\ ;\ R1 = [ROM]_2$
T_3	$R2 = R1 + \frac{R2}{2}\ ;\ R1 = [ROM]_3$
T_4	$R2 = R1 + \frac{R2}{2}\ ;\ R1 = [ROM]_4$
$\vdots$	$\vdots$
T_{13}	$R2 = R1 + \frac{R2}{2}\ ;\ R1 = [ROM]_{13}$
T_{14}	$R2 = R1 + \frac{R2}{2}\ ;\ R1 = [ROM]_1$
T_{15}	$SAL = R2$
	$R2 = R1 + CI\ ;\ R1 = [ROM]_2$
T_{16}	$R2 = R1 + \frac{R2}{2}\ ;\ R1 = [ROM]_3$
$\vdots$	$\vdots$
T_{27}	$R2 = R1 + \frac{R2}{2}\ ;\ R1 = [ROM]_1$
T_{28}	$SAL = R2$
	$R2 = R1 + CI\ ;\ R1 = [ROM]_2$
T_{29}	$R2 = R1 + \frac{R2}{2}\ ;\ R1 = [ROM]_3$
$\vdots$	$\vdots$

(b)

Figura 3.13: Evolución temporal del *pipeline* con (a) 10 bits (b) 13 bits

estructura.

La secuencia de operaciones llevadas a cabo por la estructura, para el caso de una palabra de 10 bits, se puede expresar como:

$$Salida_{MAC} = \frac{CI + [ROM]_1}{2^9} + \frac{[ROM]_2}{2^8} + \frac{[ROM]_3}{2^7} + \frac{[ROM]_4}{2^6} + \frac{[ROM]_5}{2^5} + \\ + \frac{[ROM]_6}{2^4} + \frac{[ROM]_7}{2^3} + \frac{[ROM]_8}{2^2} + \frac{[ROM]_9}{2} + [ROM]_{10}$$

que coincide con la secuencia de operaciones llevada a cabo por la estructura original.

Para el caso de una palabra de 13 bits, como la estructura solo es capaz de procesar palabras con un número de bits múltiplo de 2, habrá que añadir un bit a esa palabra. Para ello, se estudia primero la secuencia de operaciones de la estructura cuando procesa palabras de 14 bits.

$$Salida_{MAC} = \frac{CI + [ROM]_1}{2^{13}} + \frac{[ROM]_2}{2^{12}} + \frac{[ROM]_3}{2^{11}} + \frac{[ROM]_4}{2^{10}} + \frac{[ROM]_5}{2^9} + \\ + \frac{[ROM]_6}{2^8} + \frac{[ROM]_7}{2^7} + \frac{[ROM]_8}{2^6} + \frac{[ROM]_9}{2^5} + \frac{[ROM]_{10}}{2^4} + \\ + \frac{[ROM]_{11}}{2^3} + \frac{[ROM]_{12}}{2^2} + \frac{[ROM]_{13}}{2} + [ROM]_{14}$$

Si el valor de $[ROM]_1$ coincide con el de la condición inicial, la secuencia anterior equivaldrá a:

$$\begin{aligned} Salida_{MAC} = & \frac{2 \cdot CI}{2^{13}} + \frac{[ROM]_2}{2^{12}} + \frac{[ROM]_3}{2^{11}} + \frac{[ROM]_4}{2^{10}} + \frac{[ROM]_5}{2^9} + \\ & + \frac{[ROM]_6}{2^8} + \frac{[ROM]_7}{2^7} + \frac{[ROM]_8}{2^6} + \frac{[ROM]_9}{2^5} + \frac{[ROM]_{10}}{2^4} + \\ & + \frac{[ROM]_{11}}{2^3} + \frac{[ROM]_{12}}{2^2} + \frac{[ROM]_{13}}{2} + [ROM]_{14} \end{aligned}$$

que se puede expresar como:

$$\begin{aligned} Salida_{MAC} = & \frac{CI + [ROM]_2}{2^{12}} + \frac{[ROM]_3}{2^{11}} + \frac{[ROM]_4}{2^{10}} + \frac{[ROM]_5}{2^9} + \\ & + \frac{[ROM]_6}{2^8} + \frac{[ROM]_7}{2^7} + \frac{[ROM]_8}{2^6} + \frac{[ROM]_9}{2^5} + \frac{[ROM]_{10}}{2^4} + \\ & + \frac{[ROM]_{11}}{2^3} + \frac{[ROM]_{12}}{2^2} + \frac{[ROM]_{13}}{2} + [ROM]_{14} \end{aligned}$$

que coincide con la secuencia que realiza la estructura original. Por tanto, para usar esta estructura cuando las palabras de entrada tienen una longitud de 13 bits, ha de realizarse un desplazamiento a la izquierda de un bit, asignando a ese bit de desplazamiento un valor nulo, consiguiendo así que la salida inicial de la ROM ($[ROM]_1$) coincida con la condición inicial.

El flujo de los datos a través del *pipeline* se puede ver en las tablas 3.15(a) y (b) para los casos de 10 y 14 bits respectivamente.

Partición	*Bits entrada*	*Ciclos Salida*	*Ciclos segmentación*
1	10	11	10
1	13	14	13
2	10	7	5
2	13	9	7
3	10	7	4
3	13	8	5
4	10	6	3
4	13	7	4
5	10	6	2
5	13	7	3
...	...	...	...

Tabla 3.4: Aumento del paralelismo en las estructuras MAC

Para incrementar aún más la partición ha de realizarse un proceso similar al estudiado, teniendo en cuenta cuantos bits ha de procesar la estructura, y si es necesario realizar un desplazamiento de las palabras de entrada. En el Anexo A se incluye un

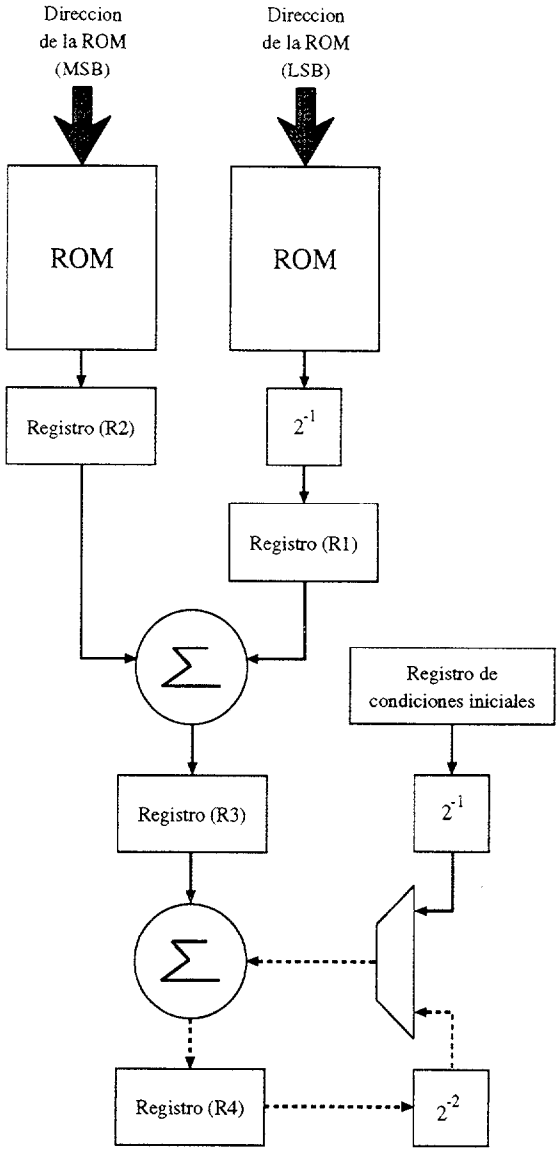


Figura 3.14: Estructura MAC con una partición de 2

estudio de la estructura MAC con particiones 3, 4 y 5.

En la tabla 3.4 se muestra el número de ciclos necesarios para obtener el primer resultado a la salida de la estructura MAC, así como los necesarios para que vaya dando los siguientes, teniendo en cuenta que el número de bits de las palabras de entrada puede ser 10 (previo a la RAM) o 13 (después de la RAM).

Existe un claro compromiso entre el área de la estructura multiplicador–acumulador, y número de ciclos necesarios para completar la operación, es decir, el *throughput* del circuito. La elección de la partición será, por tanto, un compromiso entre ambos factores.

T_1	$R2 = [ROM]_2 \; ; \; R1 = \frac{[ROM]_1}{2}$
T_2	$R3 = R2 + R1$
	$R2 = [ROM]_4 \; ; \; R1 = \frac{[ROM]_3}{2}$
T_3	$R4 = R3 + \frac{CI}{2} \; ; \; R3 = R2 + R1$
	$R2 = [ROM]_6 \; ; \; R1 = \frac{[ROM]_5}{2}$
T_4	$R4 = R3 + \frac{R4}{2^2} \; ; \; R3 = R2 + R1$
	$R2 = [ROM]_8 \; ; \; R1 = \frac{[ROM]_7}{2}$
T_5	$R4 = R3 + \frac{R4}{2^2} \; ; \; R3 = R2 + R1$
	$R2 = [ROM]_{10} \; ; \; R1 = \frac{[ROM]_9}{2}$
T_6	$R4 = R3 + \frac{R4}{2^2} \; ; \; R3 = R2 + R1$
	$R2 = [ROM]_2 \; ; \; R1 = \frac{[ROM]_1}{2}$
T_7	$R4 = R3 + \frac{R4}{2^2} \; ; \; R3 = R2 + R1$
	$R2 = [ROM]_4 \; ; \; R1 = \frac{[ROM]_3}{2}$
T_8	$RAM = R4$
	$R4 = R3 + \frac{CI}{2} \; ; \; R3 = R2 + R1$
	$R2 = [ROM]_6 \; ; \; R1 = \frac{[ROM]_5}{2}$
$\vdots$	$\vdots$
T_{12}	$R4 = R3 + \frac{R4}{2^2} \; ; \; R3 = R2 + R1$
	$R2 = [ROM]_4 \; ; \; R1 = \frac{[ROM]_3}{2}$
T_{13}	$RAM = R4$
	$R4 = R3 + \frac{CI}{2} \; ; \; R3 = R2 + R1$
	$R2 = [ROM]_6 \; ; \; R1 = \frac{[ROM]_5}{2}$
T_{14}	$R4 = R3 + \frac{R4}{2^2} \; ; \; R3 = R2 + R1$
	$R2 = [ROM]_8 \; ; \; R1 = \frac{[ROM]_7}{2}$
$\vdots$	$\vdots$
$\vdots$	$\vdots$

(a)

T_1	$R2 = [ROM]_2 \; ; \; R1 = \frac{[ROM]_1}{2}$
T_2	$R3 = R2 + R1$
	$R2 = [ROM]_4 \; ; \; R1 = \frac{[ROM]_3}{2}$
T_3	$R4 = R3 + \frac{CI}{2} \; ; \; R3 = R2 + R1$
	$R2 = [ROM]_6 \; ; \; R1 = \frac{[ROM]_5}{2}$
T_4	$R4 = R3 + \frac{R4}{2^2} \; ; \; R3 = R2 + R1$
	$R2 = [ROM]_8 \; ; \; R1 = \frac{[ROM]_7}{2}$
$\vdots$	$\vdots$
T_7	$R4 = R3 + \frac{R4}{2^2} \; ; \; R3 = R2 + R1$
	$R2 = [ROM]_{14} \; ; \; R1 = \frac{[ROM]_{13}}{2}$
T_8	$R4 = R3 + \frac{R4}{2^2} \; ; \; R3 = R2 + R1$
	$R2 = [ROM]_2 \; ; \; R1 = \frac{[ROM]_1}{2}$
T_9	$R4 = R3 + \frac{R4}{2^2} \; ; \; R3 = R2 + R1$
	$R2 = [ROM]_4 \; ; \; R1 = \frac{[ROM]_3}{2}$
T_{10}	$SAL = R4$
	$R4 = R3 + \frac{CI}{2} \; ; \; R3 = R2 + R1$
	$R2 = [ROM]_6 \; ; \; R1 = \frac{[ROM]_5}{2}$
$\vdots$	$\vdots$
T_{16}	$R4 = R3 + \frac{R4}{2^2} \; ; \; R3 = R2 + R1$
	$R2 = [ROM]_4 \; ; \; R1 = \frac{[ROM]_3}{2}$
T_{17}	$SAL = R4$
	$R4 = R3 + \frac{CI}{2} \; ; \; R3 = R2 + R1$
	$R2 = [ROM]_6 \; ; \; R1 = \frac{[ROM]_5}{2}$
T_{18}	$R4 = R3 + \frac{R4}{2^2} \; ; \; R3 = R2 + R1$
	$R2 = [ROM]_8 \; ; \; R1 = \frac{[ROM]_7}{2}$
$\vdots$	$\vdots$

(b)

Figura 3.15: Evolución temporal del *pipeline* con (a) 10 bits y (b) 14 bits

3.2.4 Estudio de la segmentación del circuito

Mantener la segmentación entre las estructuras MAC de las transformadas unidimensionales no es un problema de solución inmediata. El hecho de tener estructuras MAC con diferentes requisitos de ciclos de cálculo y que una misma estructura precise un número diferente de ciclos según el número de bits que tenga la palabra de entrada complica la segmentación del circuito completo.

Para resolver el problema de la segmentación, se estudian los bloques de multiplicación–acumulación, comenzando por el de salida, que, al tener más bits de entrada, es el que marca el *throughput* del circuito.

3.2.4.1 Mínimo paralelismo

El primer caso estudiado es la opción más simple, en la que ambas transformadas unidimensionales se implementan con la estructura MAC básica. En este caso, cada estructura MAC de la segunda transformada unidimensional procesa un dato cada 13 ciclos (exactamente los que tarda en leer los datos de entrada). Como las palabras de entrada de cada estructura MAC de la primera transformada unidimensional constan de 10 bits, cada 10 ciclos producen un dato a su salida, que son, al igual que en el caso anterior, los ciclos que tardan en leer los datos de entrada. Esto conlleva la necesidad de aumentar el número de ciclos para que las estructuras MAC que forman la primera transformada unidimensional produzcan un dato, concretamente hasta 13, para mantener la sincronización entre ambas estructuras.

Esto tiene consecuencias directas en el flujo de entrada de los datos al circuito. Se necesitan 13 ciclos para realizar la transformada unidimensional de las columnas de entrada, por lo que en lugar de estar continuamente suministrando un dato por ciclo al circuito, entran los 8 datos que forman una columna, siendo necesaria la posterior inclusión de 5 ciclos de espera, para completar los 13 ciclos requeridos para la lectura de los datos por parte de la estructura MAC.

Esto se muestra en la figura 3.16, donde se puede observar la forma en la que entran los datos a las estructuras multiplicadoras–acumuladoras que componen la primera y la segunda 1–D DCT. Cada una de las 8 estructuras MAC procesa palabras de 13 bits, que entrarán en éstas una vez obtenido el resultado del proceso de las anteriores palabras. Para el caso de la primera 1–D DCT, los datos que entran a las estructuras multiplicadoras–acumuladoras tienen una longitud de 10 bits. Eso significa que, antes de comenzar a procesar la columna "útil" (8 palabras de 10 bits), proceso que se realiza en 10 ciclos, es necesaria la adición de 3 ciclos de relleno (*dummy cycles*) para mantener la sincronización.

Un parámetro importante en la descripción arquitectural es el *factor de utilización* de las estructuras MAC. Las palabras de entrada a las estructuras MAC de la segunda 1–D DCT tienen una longitud "útil" de 13 bits, que coinciden con los bits que entran a dichas estructuras. Por lo tanto, el *factor de utilización* de las estructuras MAC que conforman la segunda 1–D DCT es de un 100%. Por otro lado, las palabras de entrada a las estructuras MAC de la primera 1–D DCT tienen una longitud "útil" de 10 bits, pero han de añadirse 3 bits, que no influyen en el resultado, para mantener la sincronización. El *factor de utilización* será, aproximadamente, de un 77%. El factor de utilización total del circuito, calculado como la media aritmética de los factores de utilización de las estructuras MAC, es de un 88.5%.

3.2.4.2 Paralelismo para flujo óptimo

En el apartado anterior se ha podido comprobar que existe un problema en la sincronización de los datos entre la primera y segunda 1–D DCT. En ese sentido, una segmentación muy eficiente se puede conseguir con estructuras MAC con una partición 3 para la primera etapa transformadora, y usando estructuras MAC con una partición de 4 en la segunda. De esta forma el flujo de datos (que es variable en cuanto al número

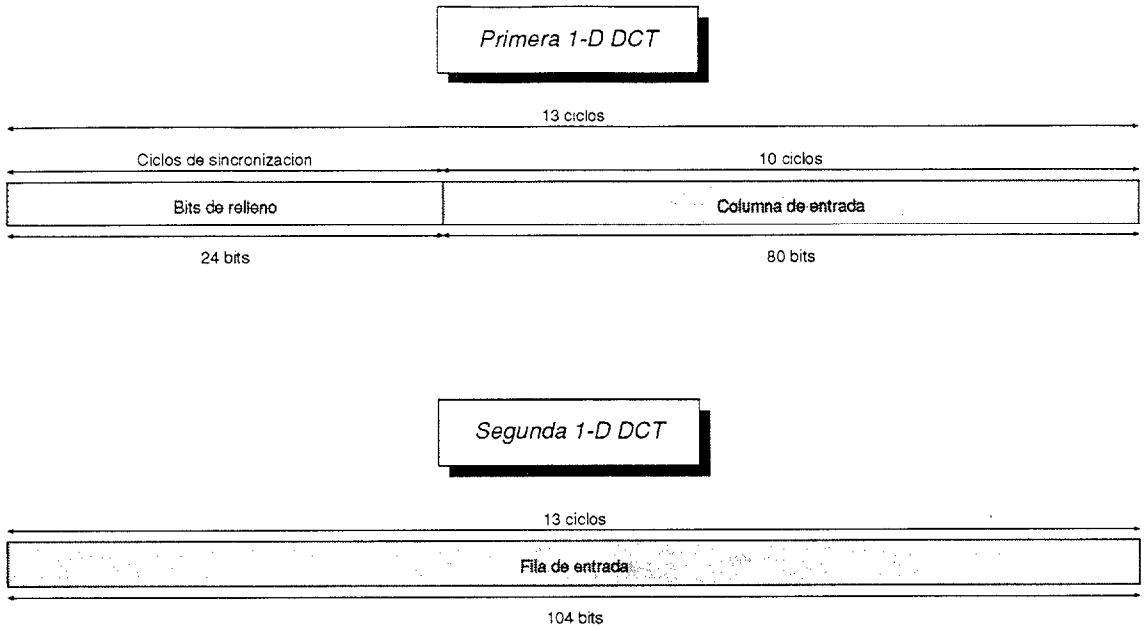


Figura 3.16: Entrada por columnas o por filas

de bits) queda unificado, ya que ambas estructuras producen un dato a la salida cada 4 ciclos.

La lectura de los datos se puede hacer a pares en la columna de entrada. Como cada dato tiene una longitud de 9 bits, se necesitan un total de 18 patillas de entrada para este fin. Cuando se ha leído la columna completa, se comienzan a procesar los bits, y se entregan al multiplicador–acumulador de 3 en 3 (como la estructura MAC con una partición de 3 sólo es capaz de procesar palabras con un número de bits múltiplo de 3, hay que adaptar dichas palabras de forma que alcancen los 12 bits de longitud), mientras siguen entrando los datos al circuito, de forma que en 4 ciclos se ha entregado la columna completa a las estructuras MAC de la primera etapa transformadora, y la siguiente ya está disponible para ser entregada. Cada 4 ciclos, se entrega la columna transformada a la RAM, que, además de actuar como elemento de almacenamiento, lo hace como circuito de transposición. Este elemento de almacenamiento entrega 4 bits cada ciclo a la segunda etapa de transformación (como la estructura MAC con una partición de 4 sólo es capaz de procesar palabras con un número de bits múltiplo de 4, hay que adaptar dichas palabras de forma que alcancen los 16 bits de longitud), de forma que en 4 ciclos se ha entregado toda la columna, y se comienza a entregar la siguiente. Usando esta implementación se consigue mantener el flujo de datos a la entrada del circuito.

El proceso de lectura de la matriz para las dos etapas transformadoras se muestra en la figura 3.17, donde se observa que se completa en 32 ciclos.

Esta estructura, produce una fila de la matriz resultante cada 4 ciclos. En cada ciclo se podrán obtener dos datos de la fila de salida correspondiente, para lo que sería necesario disponer de 24 patillas a la salida. De esta forma, el procesador de la DCT matendría un flujo de datos uniforme.

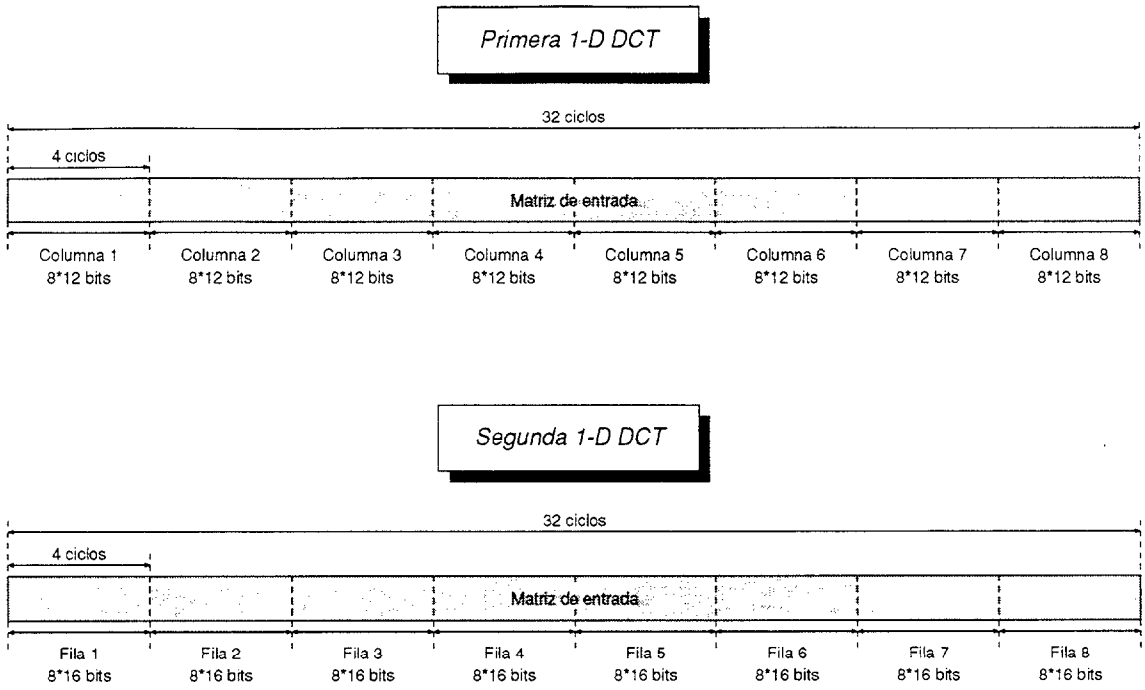


Figura 3.17: Entrada de la matriz sin la inclusión de *dummy cycles*

El *factor de utilización* de las estructuras MAC que forman la segunda 1–D DCT, al tener tres bits añadidos que no influyen en el resultado final, es de un 81%. La longitud "útil" de las palabras de entrada a las estructuras MAC de la primera 1–D DCT es de 10 bits, pero es necesaria la adición de 2 bits para adaptar la longitud de la palabra de entrada. El *factor de utilización* será, por tanto, de un 83%. Con todo esto, es posible calcular el factor de utilización total del circuito, que, para esta arquitectura, es de un 82%.

Mediante un estudio similar se pueden utilizar conjuntamente diferentes particiones para alcanzar así el rendimiento deseado, en base a las necesidades y limitaciones de cada diseño.

3.2.5 Latencia, throughput y recursos necesarios

En este apartado se realizará un estudio de la latencia, el *throughput* y los recursos necesarios para la implementación de la *Transformada Discreta del Coseno*, para las arquitecturas de **mínimo paralelismo** y **paralelismo para flujo óptimo**, en base al estudio realizado sobre estos parámetros en las estructuras MAC.

3.2.5.1 Mínimo paralelismo

Tal y como se demostró en el apartado anterior, en la implementación más simple, con estructuras MAC sin particionar para ambas transformadas, los datos entran a razón de palabra por ciclo, lo que implicaría 8 ciclos para leer toda la columna. Debido a que la segunda 1–D DCT se realiza sobre 13 bits, a estos ciclos hay que añadir 5 más, necesarios para conservar la sincronización entre etapas transformadoras. Por tanto, la lectura de la matriz se realiza en 104 ciclos de reloj.

Con la ayuda de la figura 3.18, donde se representa el diagrama de bloques de la realización de la DCT, y la figura 3.19, se expondrá el flujo de los datos a través del circuito. La entrada de datos se produce con una cadencia de 1 por ciclo. Los datos de la columna comienzan a ser entregados en serie a los sumadores y restadores que conforman la etapa de pre–proceso. Durante 13 ciclos, los registros de salida de dicha etapa van a estar ocupados por los datos de una misma columna. Los datos que han sido pre–procesados pasan a las estructuras MAC y los datos resultantes, con un desplazamiento de dos ciclos respecto a la entrada del último bit, se almacenan temporalmente en los registros de entrada de la RAM. Debido a que la estructura MAC proporciona un resultado cada 13 ciclos, éste será el tiempo disponible para almacenarlos en la memoria RAM. Una vez se ha procesado la última columna, y los resultados correspondientes a su primera transformación se han almacenado en la RAM, la memoria comienza a entregar datos en serie a la segunda etapa de pre-procesamiento, lo que hace que, con un desface de un ciclo, los bits se encuentren en el registro previo a las segundas estructuras MAC. A partir de aquí, el procedimiento se repite en la segunda 1–D DCT.

El proceso global de transformación se puede expresar como:

- Proceso de lectura: Se realiza la lectura y el almacenamiento de la columna de entrada. Esta columna consta de 8 palabras de 9 bits de entrada. La entrada se realiza en 8 ciclos (una palabra por ciclo) y se añaden 5 ciclos para mantener la sincronización.
- Conversión paralelo/serie y pre–proceso: Una vez que se ha leído toda la columna de entrada, se almacena en un segundo banco de registros, y comienza la entrega de los datos en serie a la etapa de suma y resta. Este proceso se realiza en 13 ciclos, de forma que cuando haya entrado la siguiente columna, es posible comenzar a serializar sus datos.
- MAC: Proceso de multiplicación–acumulación que cada trece ciclos produce el resultado válido que se escribe en la RAM.

- Repetición de este proceso hasta que haya sido procesada toda la matriz de entrada.
- RAM: Una vez que la matriz esté almacenada en la RAM, ésta comienza a entregar los datos serializados a la estapa de suma y resta.
- MAC: Segundo elemento multiplicador–acumulador, que produce el dato de salida cada 13 ciclos.
- Salida: Los ocho resultados se mandan a la salida, uno por ciclo, teniendo que esperar 5 ciclos para obtener un nuevo resultado.

Los recursos requeridos por esta estructura son:

- Dieciséis estructuras MAC sin particionar (8 por cada etapa de transformación unidimensional).
- Dos RAM para mantener la segmentación del circuito.
- Un conversor paralelo/serie.
- Dos etapas pre–procesadoras consistentes en 4 sumadores y cuatro restadores de 1 bit cada uno.

Para calcular la latencia (tiempo que tarda el circuito en dar el primer resultado, medido desde el punto en el que se le ha entregado el primer dato), basta con realizar una simple suma:

$$\begin{aligned} \textit{Latencia} \;=\; & 104 \text{ ciclos para la entrada de la matriz completa} + \\ & + 1 \text{ ciclo para la conversión a serie} + \\ & + 13 \text{ ciclos para que el último bit sea sumado y restado} + \\ & + 2 \text{ ciclos para que la estructura MAC de un resultado} + \\ & + 13 \text{ ciclos para escribir la columna en la RAM} + \\ & + 13 \text{ ciclos para leer el último bit de la fila de la RAM} + \\ & + 1 \text{ ciclo para que el último bit sea sumado y restado} + \\ & + 2 \text{ ciclos para que la estructura produzca el resultado} \end{aligned}$$

Realizando la operación se obtiene:

$$\textit{Latencia} \;=\; 149 \text{ ciclos}$$

A partir de ese ciclo, el circuito produce una fila de la matriz transformada, cada 13 ciclos.

Un dato importante es el tiempo máximo que están los datos almacenados en la RAM en esta estructura, ya que es la más lenta de todas las posibles, y por tanto marca la frontera máxima de almacenamiento. Los ciclos necesarios para almacenar los datos en la RAM son 104, que coinciden con los necesarios para leerlos, pero debido a que en el octavo ciclo se almacena la última palabra de la primera columna (que es la que más tiempo va a estar almacenada), se puede considerar que, en el peor caso, los datos

están almacenados 200 ciclos de reloj.

Este dato será básico en el desarrollo de este proyecto, ya que permitirá el uso de memorias dinámicas en la implementación.

3.2.5.2 Paralelismo para flujo óptimo

Centrando el estudio ahora en el caso de la arquitectura con estructuras MAC con una partición de 3 para la primera 1–D DCT y una partición de 4 para la segunda 1–D DCT, a las etapas multiplicadoras–acumuladoras de la segunda etapa transformadora llegan palabras de 13 bits (a las que hay que añadir 3 bits), con lo que dichas estructuras necesitan 4 ciclos para procesarlas. A las etapas multiplicadoras–acumuladoras de la primera etapa llegan palabras de 10 bits (a las que hay que añadir 2 bits), con lo que también son necesarios 4 ciclos para que sean procesadas.

Volviendo a la figura 3.18, y en referencia a la 3.20 se expondrá el flujo de los datos a través de éste circuito. La entrada de datos se produce con una cadencia de 2 por ciclo. Los datos de la columna comienzan a ser entregados en serie a los sumadores y restadores que conforman la etapa de pre–proceso. Durante 4 ciclos, los registros de salida de dicha etapa van a estar ocupados por los datos de una misma columna. Los datos que han sido pre–procesados pasan a las estructuras MAC y los datos resultantes, con un desplazamiento de cuatro ciclos respecto a la entrada del último bit, se almacenan temporalmente en los registros de entrada de la RAM. Debido a que la estructura MAC proporciona un resultado cada 4 ciclos, este será el tiempo disponible para almacenarlos en la memoria RAM. Una vez se ha procesado la última columna, y los resultados correspondientes a su primera transformación se han almacenado en la RAM, la memoria comienza a entregar datos en serie a la segunda etapa de pre-procesamiento, lo que hace que, con un desface de un ciclo, los bits se encuentren en el registro previo a las segundas estructuras MAC. A partir de aquí, el procedimiento se repite en la segunda 1–D DCT.

El proceso global de transformación para esta implementación se puede expresar como:

- Proceso de lectura: Se realiza la lectura y el almacenamiento de la columna de entrada. Esta columna consta de 8 palabras de 9 bits de entrada, y la entrada se realiza en 4 ciclos, dos palabras por ciclo.
- Conversión paralelo/serie y pre–proceso: Una vez que se ha leído toda la columna de entrada, se almacena en un segundo banco de registros, y comienza la entrega de los datos, de tres en tres, a los sumadores y restadores. En 4 ciclos se completa el proceso, y el recurso está disponible para procesar los datos de la siguiente columna.
- MAC: Proceso de multiplicación–acumulación que cada cuatro ciclos produce el resultado válido que se escribe en la RAM.
- Repetición de este proceso hasta que haya sido procesada toda la matriz de entrada.

- RAM: Una vez que la matriz esté almacenada en la RAM, ésta comienza a entregar los datos serializados, de 4 en 4 a la estapa de suma y resta.
- MAC: Segundo elemento multiplicador–acumulador, que produce un dato de salida cada 4 ciclos.
- Salida: Los ocho resultados se mandan a la salida, dos por ciclo, de forma que al terminar de entregar los resultados de una fila, ya se habrán calculado los siguientes, y estarán disponibles para ser entregados, manteniendo así un flujo ininterrumpido de datos a la salida del circuito.

Los recursos requeridos por esta estructura son:

- Ocho estructuras MAC con una partición de 3 y 8 con una partición de 4.
- Dos RAM para mantener la segmentación del circuito.
- Un conversor paralelo/serie.
- Dos etapas pre–procesadoras, la primera consistente en 4 sumadores y cuatro restadores de 3 bits, y la segunda consistente en 4 sumadores y cuatro restadores de 4 bits.

En este caso, la latencia vendrá dada por:

$$\begin{aligned} \textit{Latencia} \;=\; & 32 \text{ ciclos para la entrada de la matriz completa} + \\ & + 1 \text{ ciclo para la conversión a serie} + \\ & + 4 \text{ ciclos para que los últimos 3 bits sean sumados y restados} + \\ & + 4 \text{ ciclos para que la estructura MAC de un resultado} + \\ & + 4 \text{ ciclos para escribir la columna en la RAM} + \\ & + 4 \text{ ciclos para leer los últimos 4 bits de la fila de la RAM} + \\ & + 1 \text{ ciclo para que los cuatro bits sean sumados y restados} + \\ & + 4 \text{ ciclos para que la estructura produzca el resultado} \end{aligned}$$

que, expresado numéricamente:

$$\textit{Latencia} \;=\; 54 \text{ ciclos}$$

A partir de ese ciclo, el circuito produce una fila cada 4 ciclos.

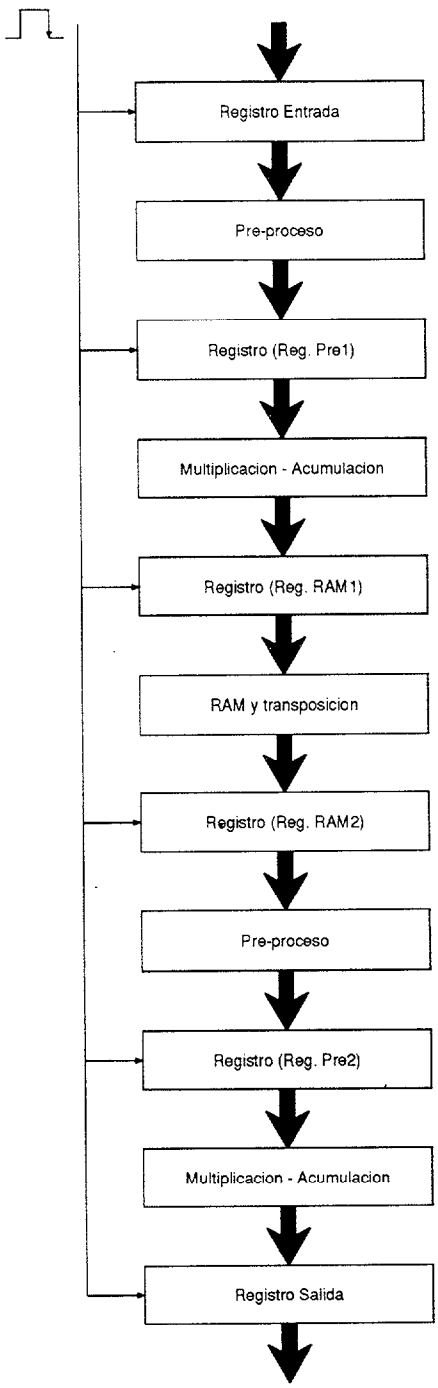


Figura 3.18: Implementación de la 2–D DCT

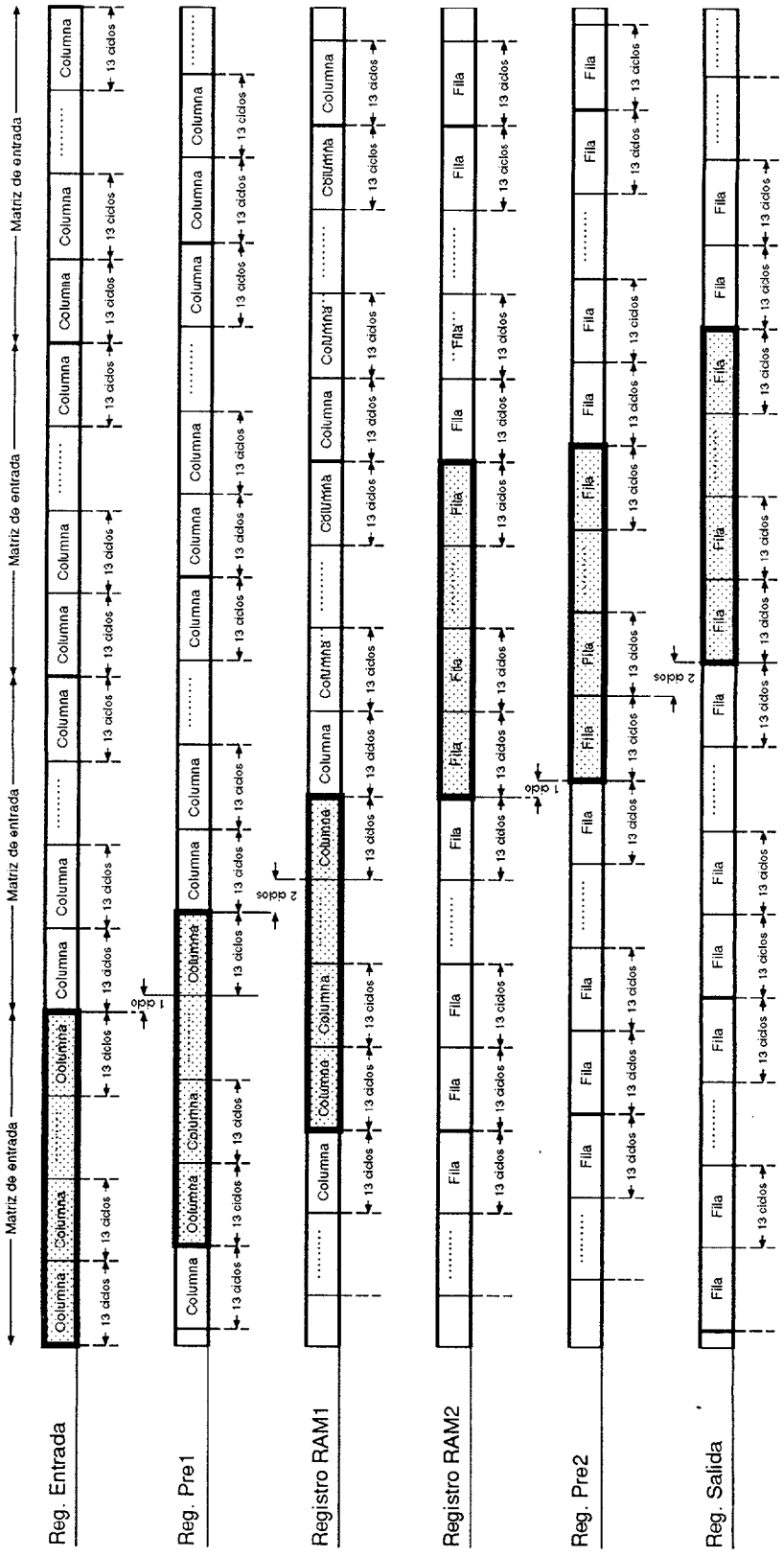


Figura 3.19: Flujo a través del circuito de mínimo paralelismo

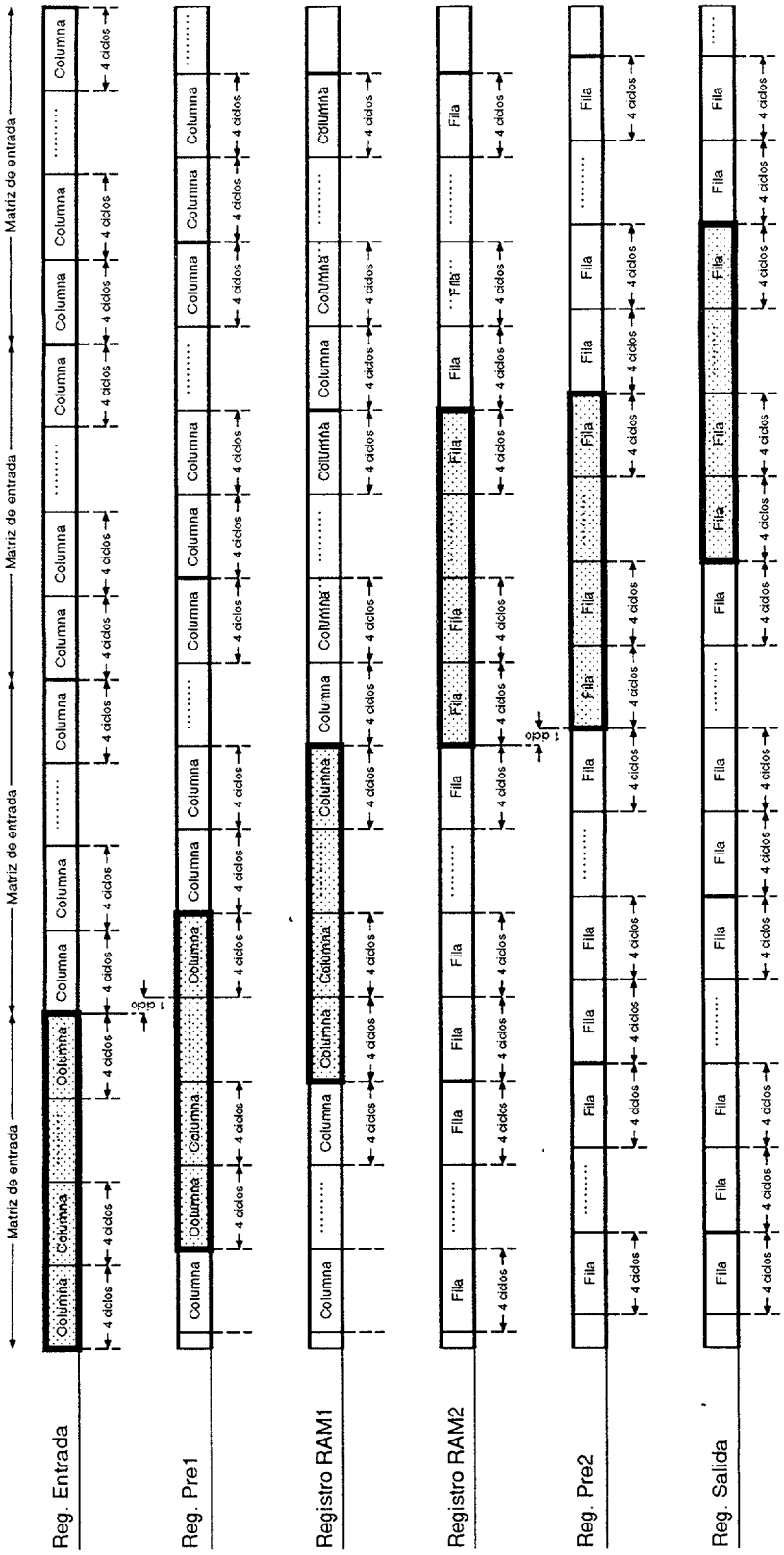


Figura 3.20: Flujo a través del circuito de paralelismo para flujo óptimo

3.3 Conclusiones

En este capítulo se ha realizado una descripción de la *Transformada Discreta del Coseno*, estudiando los diversos algoritmos eligiendo, finalmente, el algoritmo FCT–MMM para su implementación. Se ha realizado, también, un estudio de la *aritmética distribuida* como base para las estructuras multiplicadoras–acumuladoras, principal componente del circuito, evitando así el uso de los multiplicadores convencionales, que tienen unos altos requisitos de área y potencia. Asimismo, se ha realizado un *estudio arquitectural*, en el que se ha hecho especial hincapié en dos arquitecturas:

- Mínimo paralelismo: Requiere poca área, pero rompe con el flujo de los datos al circuito.
- Paralelismo para flujo óptimo: Pensada para obtener una segmentación óptima de la arquitectura a expensas de un mayor requerimiento de área.

Ambas estructuras tienen un factor de utilización muy alto y son implementaciones muy eficientes.

Capítulo 4

Implementación de las primitivas de diseño

En el capítulo anterior se han estudiado las arquitecturas multiplicadoras–acumuladoras, así como la segmentación del circuito global usando esas estructuras, haciendo especial hincapié en la implementación con estructuras MAC sin particionar y en una implementación que permite una segmentación perfecta sin necesidad de romper el flujo de datos de entrada ni de salida.

La implementación en *Arseniuro de Galio* del circuito comprende una serie de estructuras sobre las que es necesario realizar un estudio previo. Tal es el caso de:

- Registro: Es el elemento básico en la segmentación del circuito. Debido al alto número de registros necesarios, tiene un importante impacto en el área y en la potencia disipada por el circuito.
- Sumador: Es el elemento que determina la frecuencia de funcionamiento de la estructura MAC, y por tanto, del circuito.
- ROM: Su implementación presenta problemas debido a las altas corrientes de fuga que se producen en Arseniuro de Galio, muy superiores a las que se producen en CMOS.
- RAM: La elección del tipo de RAM (estática o dinámica) y su implementación con transistores MESFET no especialmente dedicados a este propósito es un punto crítico y novedoso.

En este capítulo se realizará el estudio de estas estructuras, definiendo así los bloques básicos para la implementación de la *Transformada Discreta del Coseno Bidimensional.*

4.1 Registro

Un elemento indispensable en la segmentación de cualquier circuito es el registro. En la implementación de la 2–D DCT los registros utilizados son *flip–flops* tipo D activos por flancos de bajada con señal de *clear* asíncrona, cuya representación a nivel de esquemático se puede ver en la figura 4.1, y *flip–flops* tipo D activos por flancos de

bajada con señal de *preset* asíncrona, representado en la figura 4.2. Como se puede observar, estos *flip–flops* constan de seis puertas NOR, tres de ellas tienen un *fan–in* de tres, y las tres restantes un *fan–in* de dos.

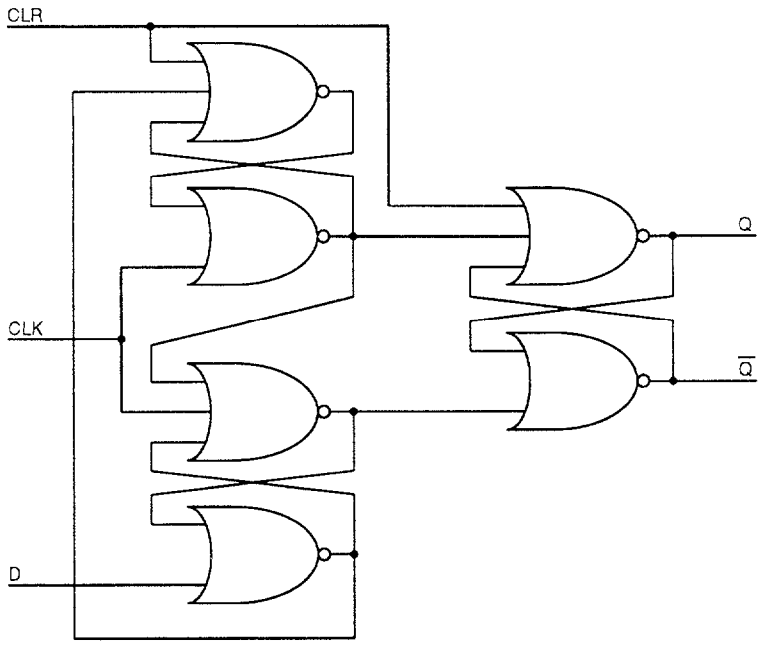


Figura 4.1: Esquemático del *flip–flop* tipo D con señal de *clear*

Debido al alto número de registros utilizados en la implementación de la *Transformada Discreta del Coseno*, se ha optado por el diseño de éstos usando la lógica DCFL, que es la que menos área requiere y menos potencia disipa, y porque en la implementación con esta lógica se obtienen unos márgenes de ruido más que aceptables para el *fan–in* y el *fan–out* característico de las puertas NOR de los *flip–flops*. Los layout de ambos *flip–flops* se pueden ver en las figuras 4.3 y 4.4.

Las características principales de estos *flip–flops*, incluyendo los efectos debidos a dispersión del proceso y a los cambios de temperatura, se pueden observar en la siguiente tabla:

Dispersión	*Tiempo de bajada (ns)*	*Tiempo de subida (ns)*	*Retardo (ns)*	*Potencia disipada (mW)*	*Area (μm^2)*
SS2 ($0°C$)	0.15	0.24	0.26	1.23	3868.8
TT ($75°C$)	0.13	0.20	0.22	1.83	
FF1H ($100°C$)	0.12	0.20	0.22	2.15	

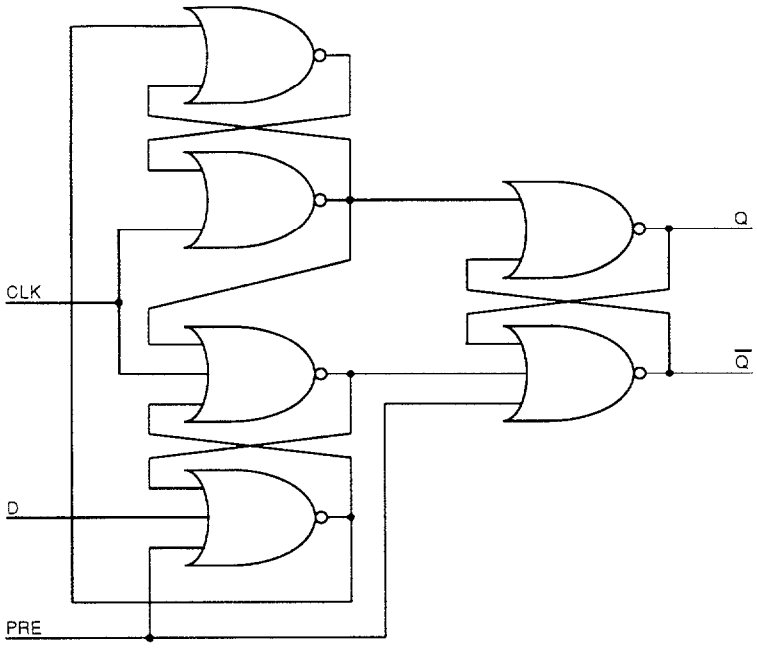


Figura 4.2: Esquemático del *flip–flop* tipo D con señal de *preset*

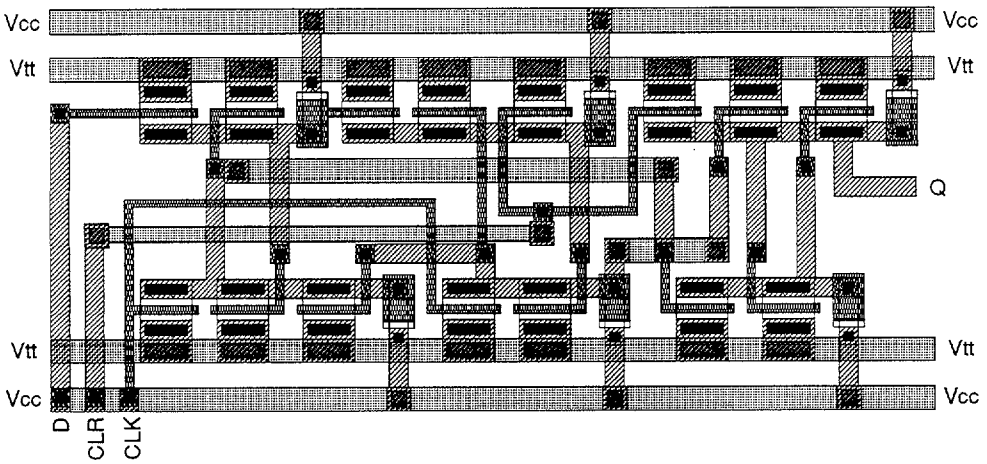


Figura 4.3: Layout del *flip–flop* tipo D con señal de *clear*

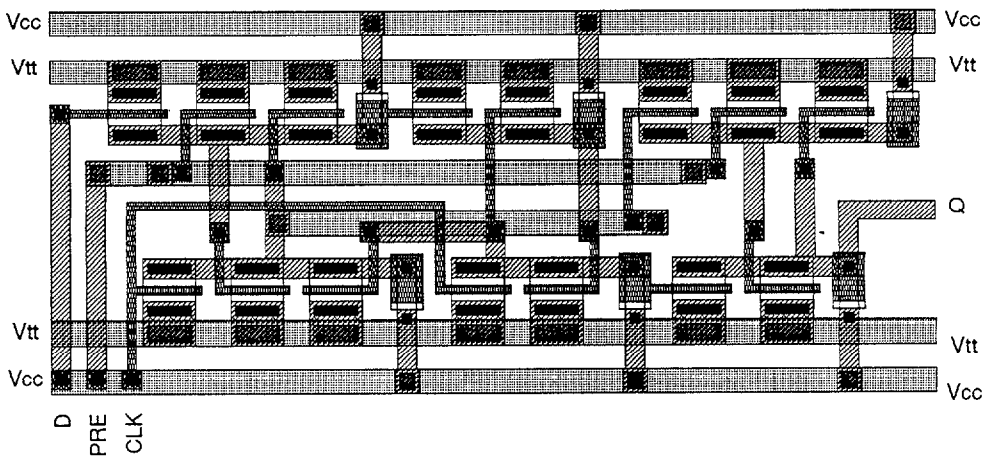


Figura 4.4: Layout del *flip–flop* tipo D con señal de *preset*

4.2 Sumadores

El sumador es un elemento crítico en la implementación de la DCT, ya que lo es en la implementación de la estructura MAC. Tiene un gran impacto sobre el área global del circuito, y condiciona la frecuencia de trabajo del mismo. Es, también, el elemento encargado de realizar el pre–procesamiento de los datos, previo a las estructuras MAC, aunque en este caso, y debido a la pequeña longitud de las palabras a sumar, no tiene una gran influencia en el área del circuito. Por ejemplo, en la implementación de la 1–D DCT con estructuras MAC sin particionar, la longitud de la palabra de entrada a la etapa de pre–proceso es de 1 bit, y si se elige una estructura MAC con una partición 4, la longitud de las palabras de entrada pasa a ser de 4 bits, con lo que, para la implementación de la etapa de pre–proceso, serían necesarios sumadores de 1 y 4 bits respectivamente.

En este capítulo se realiza un estudio de los sumadores más comunes:

- Sumador de acarreo serie (*Ripple–Carry Adder*).
- Sumador de acarreo anticipado (*Carry–Lookahead Adder*).
- Sumador selector de acarreo (*Carry–Select Adder*).

permitiendo así al diseñador una elección correcta en base a sus necesidades de frecuencia, y a sus limitaciones en área y potencia disipada.

4.2.1 Sumador completo

El sumador completo (*full–adder*) viene caracterizado por las siguientes ecuaciones:

$$\begin{aligned} s &= a \oplus b \oplus c_{in} \\ c_{out} &= (a+b) \cdot c_{in} + a \cdot b \end{aligned}$$

donde a y b son los bits a sumar, c_{in} es el acarreo de entrada, y c_{out} es el acarreo de salida.

Partiendo de dichas ecuaciones, se obtiene una implementación básica, a nivel de puertas lógicas, como la mostrada en la figura 4.5. La implementación de este sumador a nivel de layout se muestra en la figura 4.6.

4.2.2 Sumador de acarreo serie

Mediante la conexión en cascada de sumadores completos, es posible implementar sumadores de cualquier número de bits. Esto se muestra en la figura 4.7, donde se han conectado cuatro sumadores completos de 1 bit, obteniendo así un sumador de acarreo serie de cuatro bits.

El diseño a nivel de layout del sumador completo ha sido realizado de manera que la entrada y la salida del acarreo quedan a la misma altura, de forma que el diseño

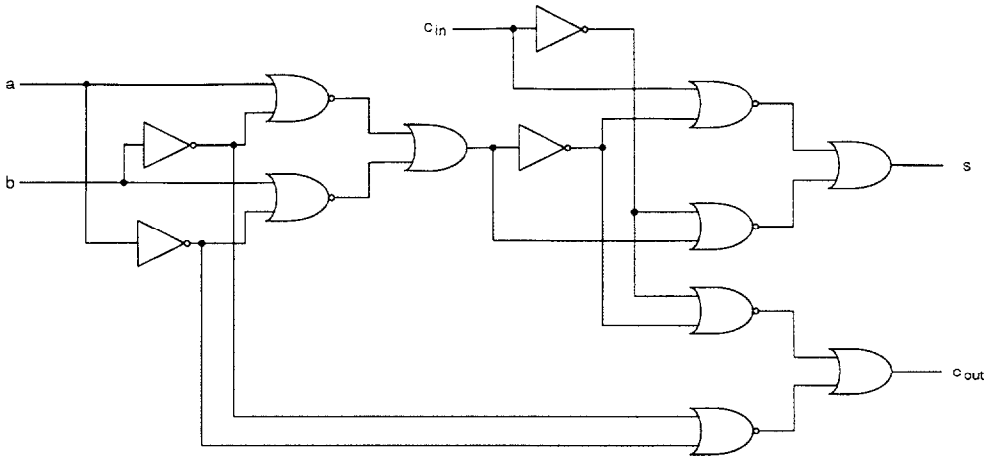


Figura 4.5: Diagrama lógico del sumador completo

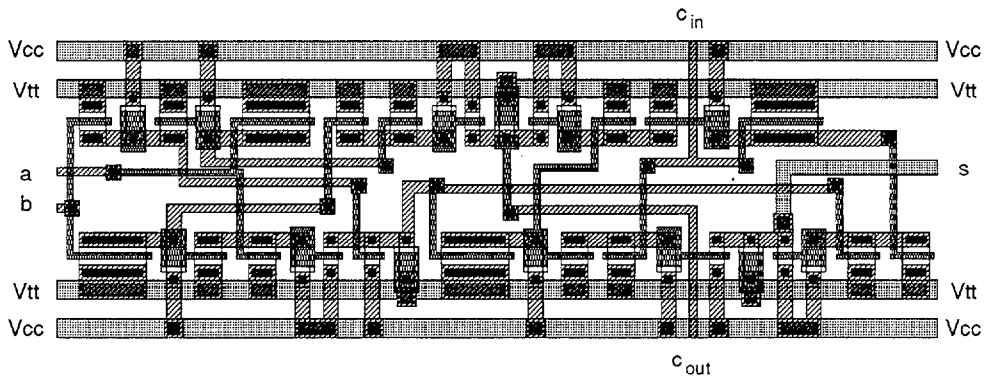


Figura 4.6: Layout del sumador completo

de un sumador de acarreo serie de cualquier número de bits sea muy sencilla. Esto se puede ver en la figura 4.8, donde se muestra la implementación a nivel de layout de un sumador de cuatro bits, realizado mediante la conexión directa de cuatro sumadores completos.

El sumador de acarreo serie es, conceptualmente, el más sencillo de los sumadores. Tiene a su favor un área y un consumo de potencia reducido, siendo su mayor desventaja el tiempo de propagación del acarreo, que en el peor caso ha de pasar a través de todos los sumadores. Por este motivo, este sumador suele ser desechado para un número de bits muy alto.

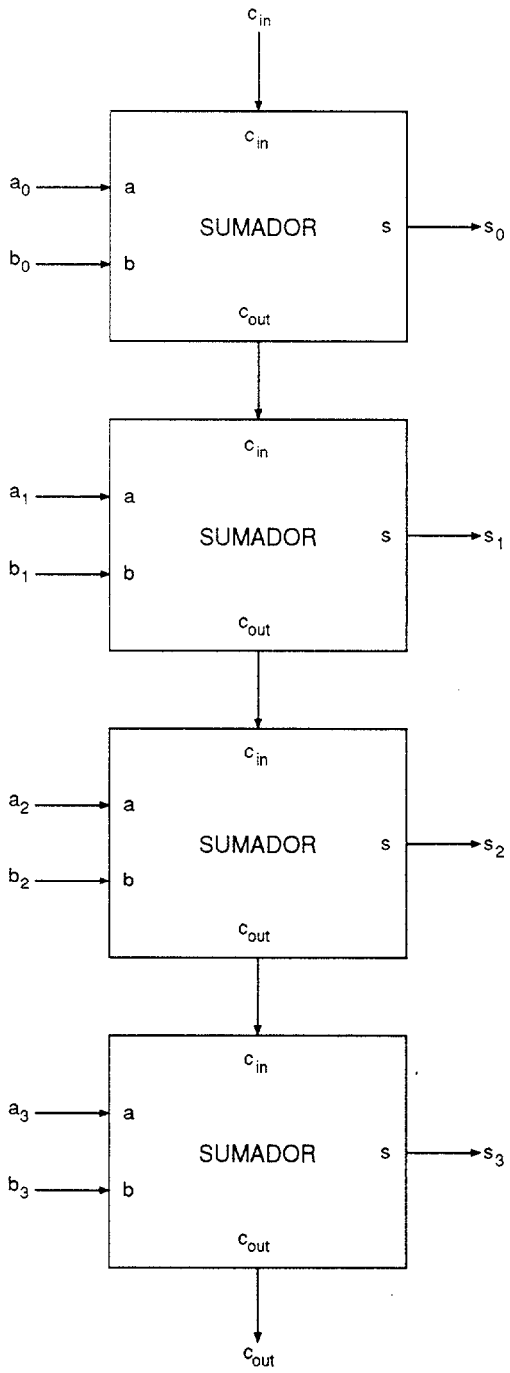


Figura 4.7: Conexión de sumadores completos para formar un sumador de 4 bits

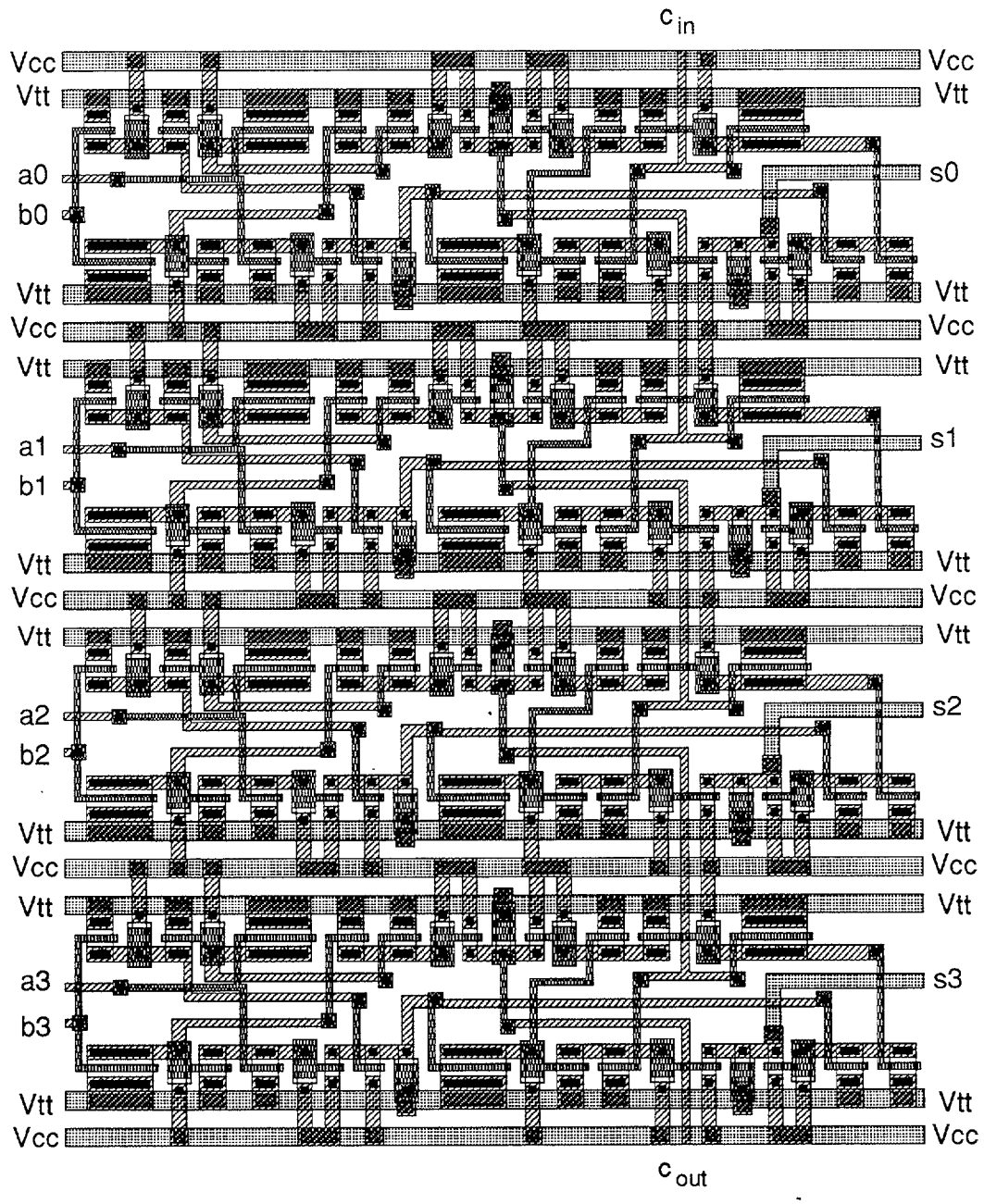


Figura 4.8: Layout del sumador de acarreo serie de 4 bits

4.2.3 Sumador de de acarreo anticipado

Para solventar el problema de la espera por el acarreo en los sumadores de acarreo serie, se diseña una estructura que permite calcular dicho acarreo de forma paralela. Esta estructura recibe el nombre de sumador de acarreo anticipado.

Dos definiciones son la clave del sumador de acarreo anticipado:

- Para una combinación de entradas a_i y b_i, la etapa de suma i **genera** un acarreo si en esa etapa se produciría un acarreo de valor lógico 1 ($c_{i+1} = 1$) independientemente de las entradas $a_0,\ldots, a_{i-1}$, $b_0,\ldots, b_{i-1}$ y c_0.

- Para una combinación de entradas a_i y b_i, la etapa de suma i **propaga** un acarreo si en esa etapa se produciría un acarreo de valor lógico 1 ($c_{i+1} = 1$) si la combinación de entradas $a_0,\ldots, a_{i-1}$, $b_0,\ldots, b_{i-1}$ y c_0, produciría un acarreo de entrada de valor lógico 1 ($c_i = 1$).

De acuerdo con estas definiciones, se pueden escribir las ecuaciones lógicas para los términos **generador** (g_i) y **propagador** (p_i), para cada etapa del sumador de acarreo anticipado:

$$\begin{aligned} g_i &= a_i \cdot b_i \\ p_i &= a_i + b_i \end{aligned}$$

Es decir, cada etapa *generará* un acarreo si los bits a ser sumados tienen un valor lógico 1, y *propagará* un acarreo si al menos uno de los bits tiene ese valor. Por tanto, el acarreo de salida de una etapa se puede expresar como:

$$c_{i+1} = g_i + p_i \cdot c_i$$

Para eliminar la propagación del acarreo, se expanden los términos c_i para cada etapa. De esta forma, la expresión del acarreo para un sumador de cuatro etapas sería:

$$\begin{aligned} c_1 &= g_0 + p_0 \cdot c_0 \\ c_2 &= g_1 + p_1 \cdot c_1 \\ &= g_1 + p_1 \cdot (g_0 + p_0 \cdot c_0) \\ &= g_1 + p_1 \cdot g_0 + p_1 \cdot p_0 \cdot c_0 \\ c_3 &= g_2 + p_2 \cdot c_2 \\ &= g_2 + p_2 \cdot (g_1 + p_1 \cdot g_0 + p_1 \cdot p_0 \cdot c_0) \\ &= g_2 + p_2 \cdot g_1 + p_2 \cdot p_1 \cdot g_0 + p_2 \cdot p_1 \cdot p_0 \cdot c_0 \\ c_4 &= g_3 + p_3 \cdot c_3 \\ &= g_3 + p_3 \cdot (g_2 + p_2 \cdot g_1 + p_2 \cdot p_1 \cdot g_0 + p_2 \cdot p_1 \cdot p_0 \cdot c_0) \\ &= g_3 + p_3 \cdot g_2 + p_3 \cdot p_2 \cdot g_1 + p_3 \cdot p_2 \cdot p_1 \cdot g_0 + p_3 \cdot p_2 \cdot p_1 \cdot p_0 \cdot c_0 \end{aligned}$$

Sin embargo la implementación directa de estas ecuaciones (figura 4.9) no es viable, debido al alto *fan-in* y *fan-out* que tendrían las puertas. Por este motivo se realiza

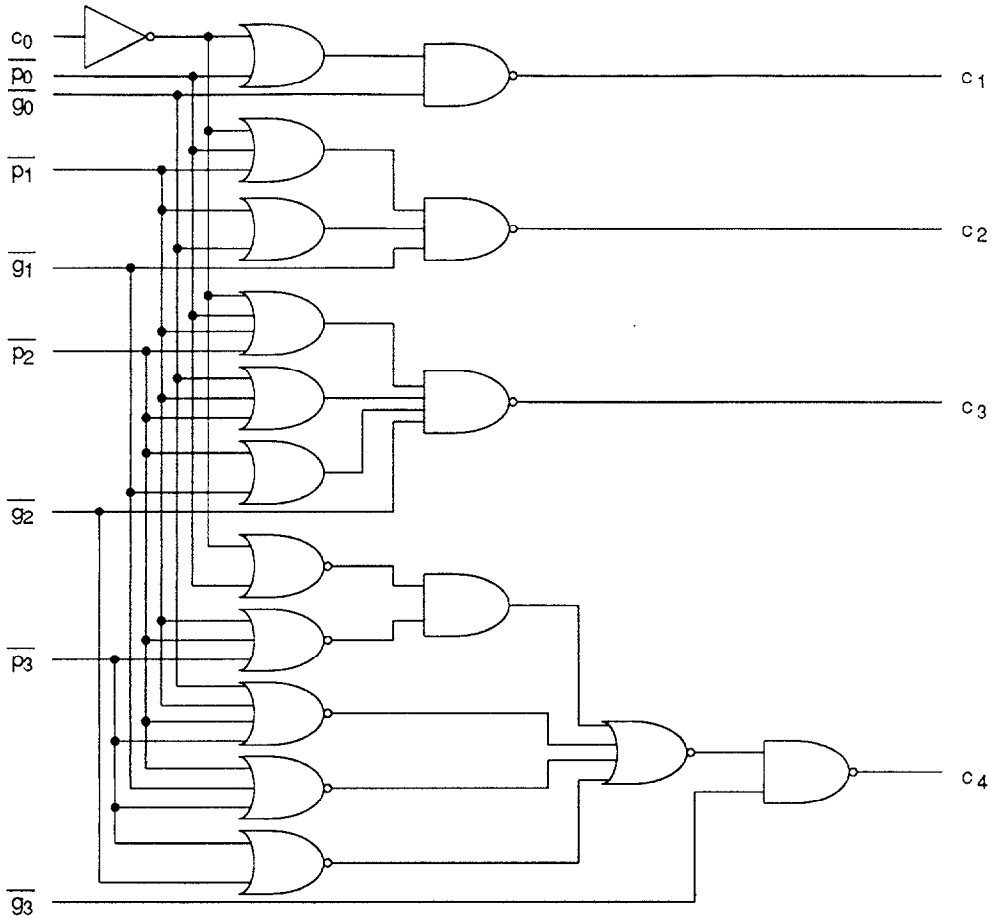


Figura 4.9: Implementación directa

una modificación de la ecuación de generación del acarreo, para que éste sea generado por cuatro células *A, B, C* y *D*.

La modificación de la ecuación de generación del acarreo consiste en dividir éste en dos términos v_i y t_{i-1}. El término v_i se genera en la célula de entradas a_i y b_i, mientras que el término t_{i-1} se genera en la célula anterior, de entradas a_{i-1} y b_{i-1}.

Para repartir la carga, y así disminuir el *fan-out*, de las señales a_i y b_i, cada célula propagará dichos términos en su forma directa o complementada, alternativamente, lo que no supone ningún incremento de área, ya que la inversión es necesaria para la realización de la XOR encargada de obtener el término de la suma.

La expresión del acarreo que se obtiene para cada célula es diferente, siendo para la *célula B*:

$$\overline{c_i} = \overline{t_{i-1} + v_i} \tag{4.1}$$

y para la *célula C*:

$$c_i = \overline{t_{i-1}} \cdot \overline{v_i} \tag{4.2}$$

La expresión del acarreo para la *célula D*, dependerá de la célula previa. En el caso de que dicha célula sea la *B* (sumador de un número par de bits), la expresión

del acarreo es la definida por la ecuación 4.2, mientras que si la célula previa es la C (sumador de un número impar de bits), la expresión queda definida por la ecuación 4.1.

De las ecuaciones anteriores se deduce que la *célula B* propaga el acarreo en su forma negada, mientras que la *célula C* lo propaga en su forma directa. Por otro lado, los términos v_i y t_{i-1} son distintos según la célula que se trate. Para la *célula B* el término v_i tiene la expresión:

$$v_i = a_{i-1} \cdot b_{i-1} = g_{i-1} \tag{4.3}$$

mientras que para la *célula C*, la expresión de dicho término es:

$$v_i = \overline{a_{i-1}} \cdot \overline{b_{i-1}} = \overline{p_{i-1}} \tag{4.4}$$

La expresión del término v_i para la *célula D* dependerá de la célula previa. Cuando dicha célula previa es la B, el término v_i tomará la expresión definida por la ecuación 4.4, mientras que si la célula previa es la C, v_i vendrá definido por la ecuación 4.3.

Los términos t_{i-1} generados en los bloques anteriores tienen diferentes expresiones según la célula que los genere. La expresión del término t_{i-1} propagado por las *células* A y C es:

$$t_{i-1} = p_{i-1} \cdot g_{i-2} + p_{i-1} \cdot p_{i-2} \cdot g_{i-3} + \ldots + p_{i-1} \cdot p_{i-2} \cdot p_{i-3} \cdot \ldots \cdot c_0$$

donde c_0 es el acarreo de entrada a la *célula A*. La expresión de t_{i-1} para la *célula B* es:

$$t_{i-1} = \overline{g_{i-1} + g_{i-2} + p_{i-2} \cdot g_{i-3} + p_{i-2} \cdot p_{i-3} \cdot g_{i-4} + \ldots + p_{i-2} \cdot p_{i-3} \cdot p_{i-4} \cdot \ldots \cdot c_0}$$

La suma en todas las células se obtiene de forma tradicional a partir de dos puertas XOR. La primera de ellas, correspondiente a las entradas a_i y b_i es la misma para todas las células. Sin embargo, la segunda puerta XOR varía de una célula a otra, dependiendo de si el acarreo generado en dicha célula es directo o negado.

Para el caso de la *célula A*, el acarreo de entrada es negado, por lo que la segunda etapa corresponde a una puerta XNOR. El acarreo generado en la *célula B* también es negado, por lo que su segunda etapa es, igualmente, una puerta XNOR. La *célula C*, por el contrario, genera el acarreo en su forma directa, por lo que su segunda etapa es una puerta XOR. En ambas puertas la suma tiene la misma expresión, saliendo siempre en forma directa.

Cada célula dependerá, por tanto, de sus entradas directas (a_i y b_i), de las entradas de la etapa anterior (a_{i-1} y b_{i-1}) bien en su forma directa o negada, y del término de propagación de la etapa anterior (t_{i-1}), limitando de esta forma tanto el *fan-in* como el *fan-out* de las puertas.

La descripción de cada una de las células que componen el sumador de selección de acarreo se detalla a continuación.

4.2.3.1 Célula A

Esta célula aparece en primer lugar en todos los sumadores, independientemente del número de bits, y sólo se utiliza una vez. Tiene como entradas los dos bits a sumar (a_i y b_i), y el acarreo inicial, que entra negado. La *célula A*, no sólo propaga el termino t_0, sino que también propaga sus entradas negadas.

Las ecuaciones que definen el comportamiento de la célula son:

$$\begin{aligned} s_0 &= (a_0 \oplus b_0) \odot \overline{c_0} \\ \overline{c_0} &= \overline{c_{in}} \\ t_0 &= (a_0 + b_0) \cdot c_{in} \quad = \quad p_0 \cdot c_0 \end{aligned}$$

En la figura 4.10 se puede ver el diagrama a nivel de puertas lógicas de esta célula.

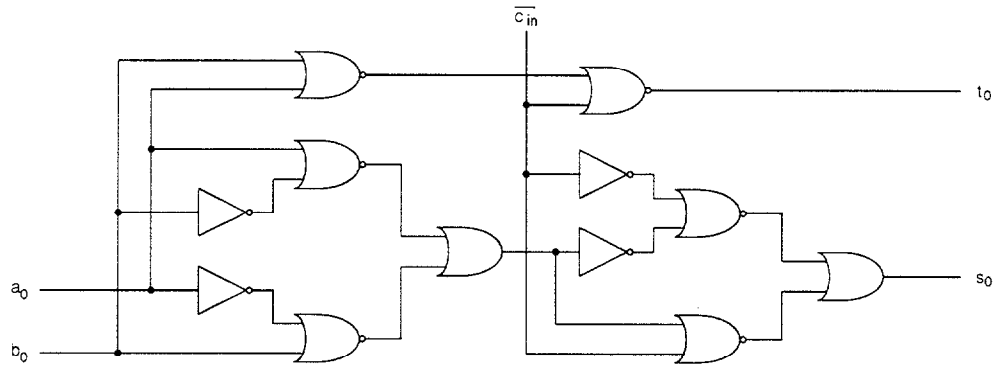


Figura 4.10: Esquemático de la célula A

En la figura 4.11 se muestra el plano de base de la *célula A*.

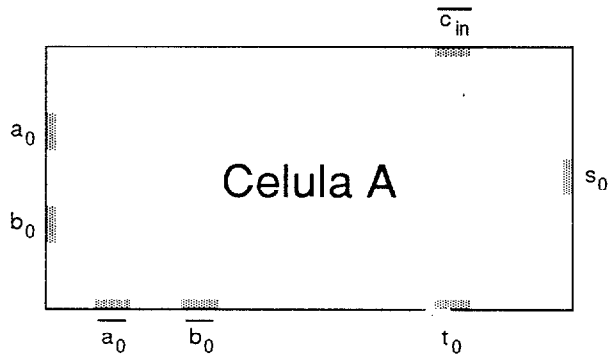


Figura 4.11: Célula A

La implementación a nivel de layout se muestra en la figura 4.12.

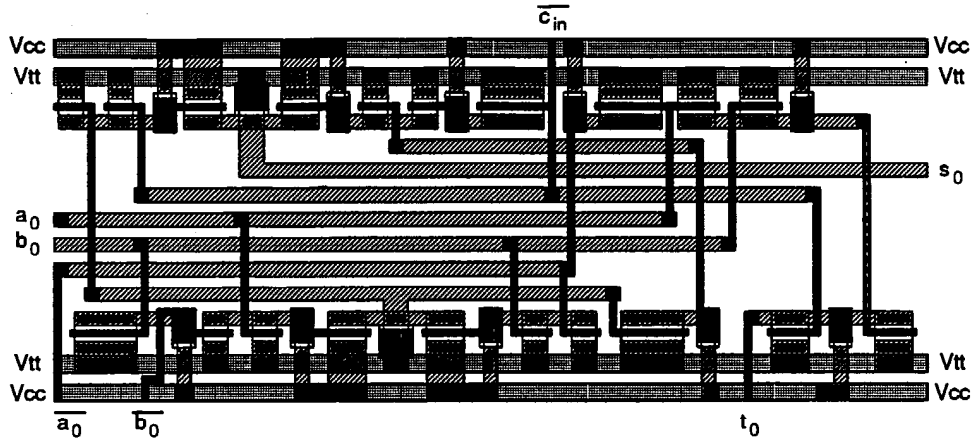


Figura 4.12: Layout de la célula A

4.2.3.2 Célula B

Esta es una de las células intermedias (junto con la *célula C*), y que se repite según el número de bits del sumador. Tiene como entradas los dos bits a sumar (a_i y b_i), así como los términos t_{i-1}, $\overline{a_{i-1}}$ y $\overline{b_{i-1}}$, generadas en la etapa anterior. Propaga a la siguiente etapa el termino t_i, y sus entradas a_i y b_i.

Las ecuaciones que definen el comportamiento de ésta célula son:

$$\begin{aligned} s_i &= (a_i \oplus b_i) \odot \overline{c_i} \\ \overline{c_i} &= \overline{t_{i-1} + (a_{i-1} \cdot b_{i-1})} \\ &= \overline{t_{i-1} + g_{i-1}} \\ t_i &= \overline{t_{i-1} + (a_{i-1} \cdot b_{i-1}) + (a_i \cdot b_i)} \\ &= \overline{t_{i-1} + g_{i-1} + g_i} \end{aligned}$$

En la figura 4.13 se puede ver el diagrama a nivel de puertas lógicas de la célula.

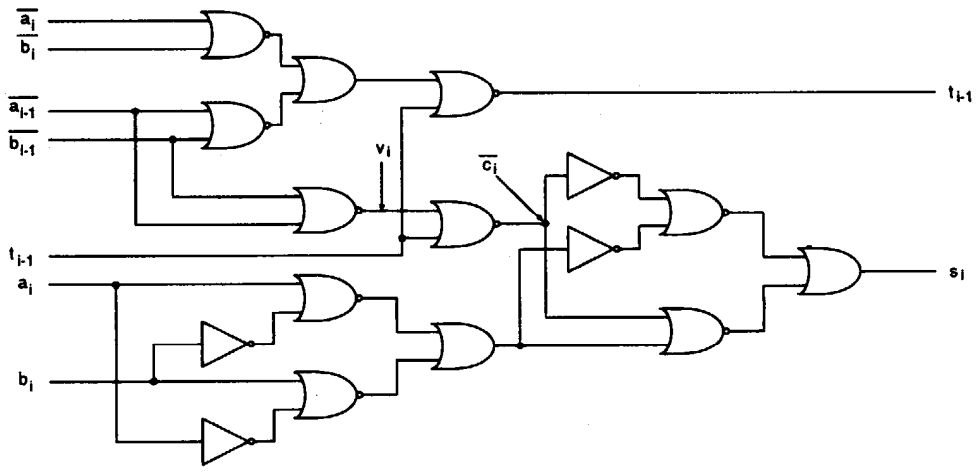


Figura 4.13: Esquemático de la célula B

En la figura 4.14 se muestra el plano de base de la *célula B*.

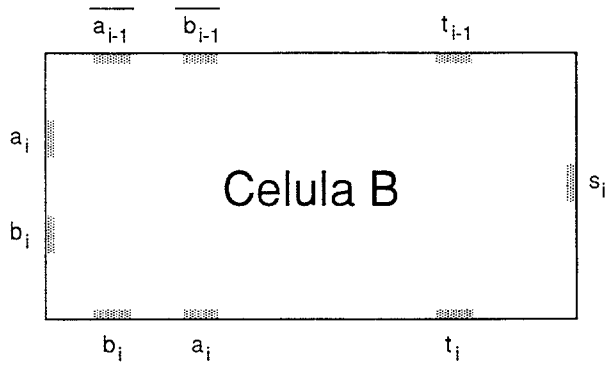


Figura 4.14: Célula B

La implementación a nivel de layout se muestra en la figura 4.15.

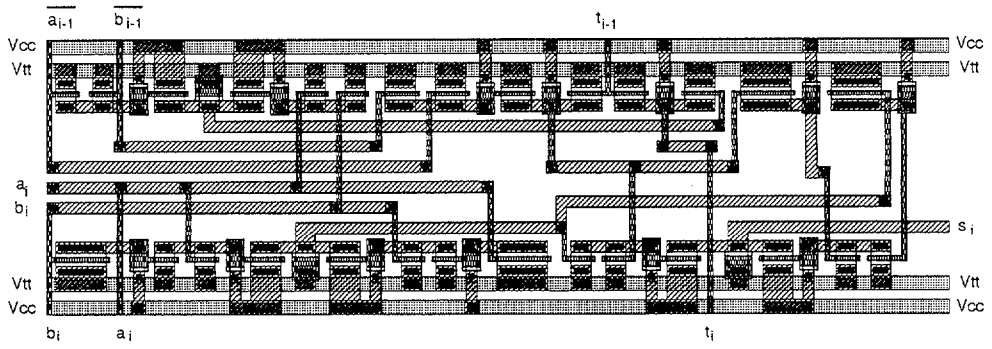


Figura 4.15: Layout de la célula B

4.2.3.3 Célula C

Junto con la anterior, es la célula que se repite en el sumador de acarreo anticipado, según el número de bits del mismo. Sucede siempre a la *célula B* y se repite tantas veces como ella cuando el número de bits es impar, haciéndolo una vez menos cuando el número de bits es par.

Tiene como entradas los dos bits a sumar (a_i y b_i), así como los términos t_{i-1}, a_{i-1} y b_{i-1}, generados en la etapa anterior. Propaga a la siguiente etapa el término t_i, y sus entradas $\overline{a_i}$ y $\overline{b_i}$.

Las ecuaciones que definen el comportamiento de esta célula son:

$$\begin{aligned} s_i &= a_i \oplus b_i \oplus c_i \\ c_i &= \overline{t_{i-1} + (\overline{a_{i-1}} \cdot \overline{b_{i-1}})} \\ &= \overline{t_{i-1}} \cdot p_{i-1} \end{aligned}$$

$$\begin{aligned} t_i &= \overline{t_{i-1} + \overline{a_{i-1}} \cdot \overline{b_{i-1}} + \overline{a_i} \cdot \overline{b_i}} \\ &= \overline{t_{i-1}} \cdot p_{i-1} \cdot p_i \end{aligned}$$

El diagrama a nivel de puertas lógicas de esta célula se puede observar en la figura 4.16.

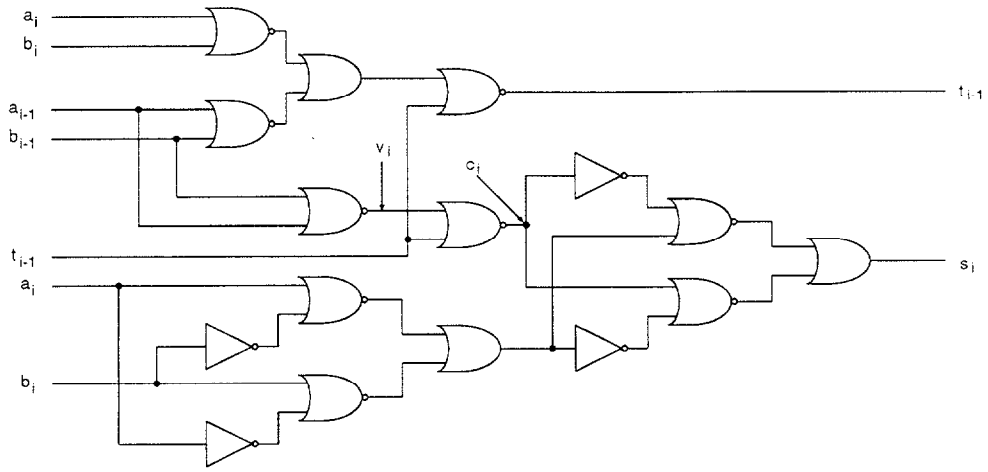


Figura 4.16: Esquemático de la célula C

En la figura 4.17 se muestra el plano de base de la *célula C*.

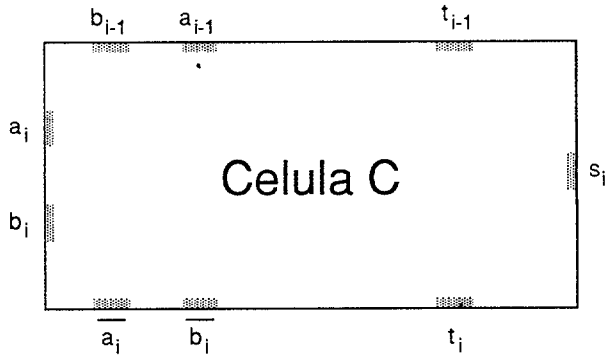


Figura 4.17: Célula C

La implementación a nivel de layout se muestra en la figura 4.18.

4.2.3.4 Célula D

Esta célula aparece una sola vez en el sumador, y siempre al final del mismo, ya que es la encargada de generar el acarreo de salida. Si a esta célula le antecede la *célula B* (sumadores con un número de bits par), el acarreo de salida del sumador es directo, mientras que si le antecede la *célula C* (sumadores con un número de bits impar), el acarreo de salida es negado.

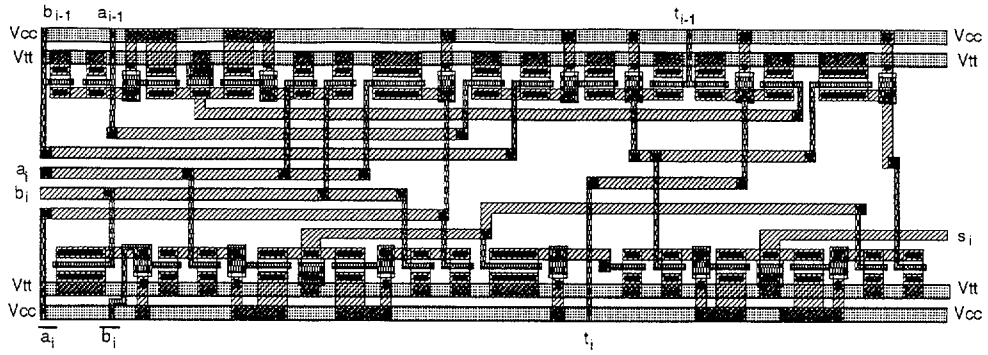


Figura 4.18: Layout de la célula C

La ecuación que define el comportamiento de ésta célula es, en el caso de que la célula previa sea la *célula B*:

$$\begin{aligned} c_{out} &= \overline{\overline{a_{i-1} + b_{i-1}} + t_{i-1}} \\ &= \overline{t_{i-1}} \cdot p_{i-1} \end{aligned}$$

mientras que, si la célula previa es la C:

$$\begin{aligned} \overline{c_{out}} &= \overline{(a_{i-1} \cdot b_{i-1}) + t_{i-1}} \\ &= \overline{t_{i-1} + g_{i-1}} \end{aligned} \tag{4.5}$$

El diagrama a nivel de puertas lógicas de esta célula, cuando el número de bits del sumador es par, se muestra en la figura 4.19.

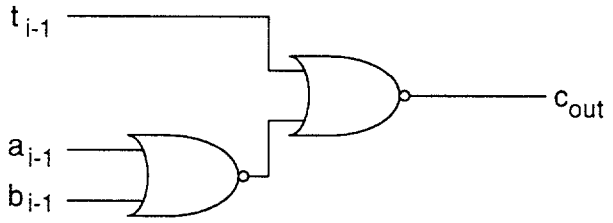


Figura 4.19: Esquemático de la célula D

En la figura 4.20 se muestra el plano de base de la *célula D.*

La implementación a nivel de layout se muestra en la figura 4.21.

Para un sumador de un número de bits impar, lo único a tener en cuenta es que las entradas a_{i-1} y b_{i-1} vienen negadas, por lo que a su salida se obtiene el acarreo negado, como se demuestra en la ecuación 4.5.

4.2.3.5 Ejemplo de realización de un sumador de 4 bits

En la implementación de sumadores de acarreo anticipado, las células A y D sólo aparecen una vez en el sumador. La primera de ellas al principio, y la segunda al final de

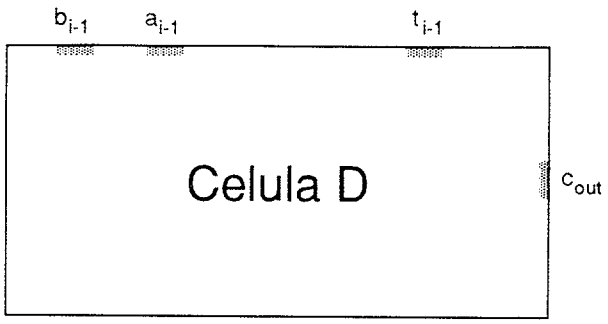


Figura 4.20: Célula D

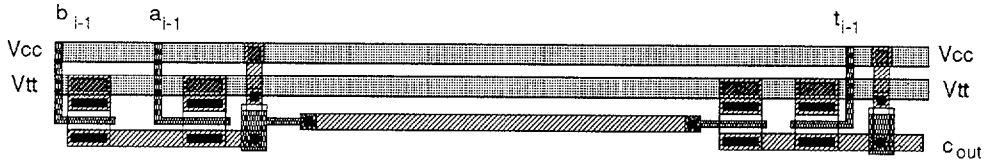


Figura 4.21: Layout de la célula D

todas, ya que su única función es la generación del acarreo de salida. Sin embargo, las células B y C se repiten de forma alternativa, según el número de bits del sumador.

La estructura de bloques del sumador de cuatro bits se puede ver en la figura 4.22.

En la figura 4.23 se muestra la implementación de un sumador de acarreo anticipado de 4 bits.

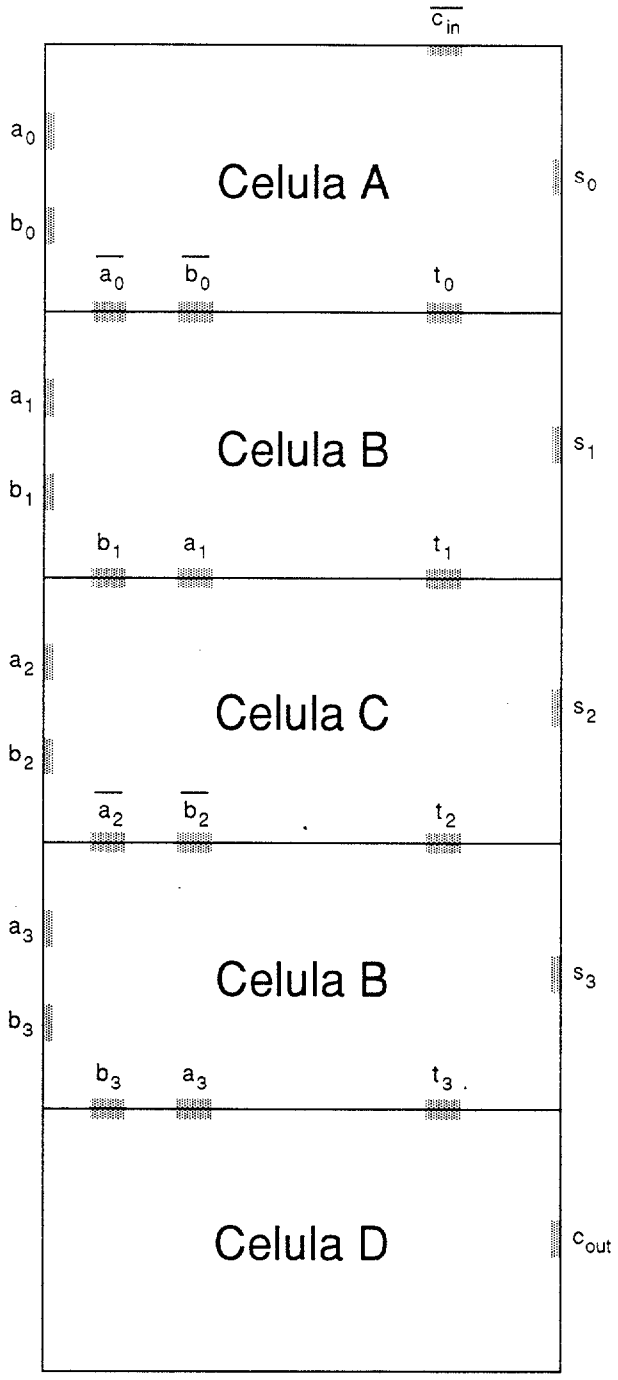


Figura 4.22: Esquema de bloques del sumador de acarreo anticipado de 4 bits

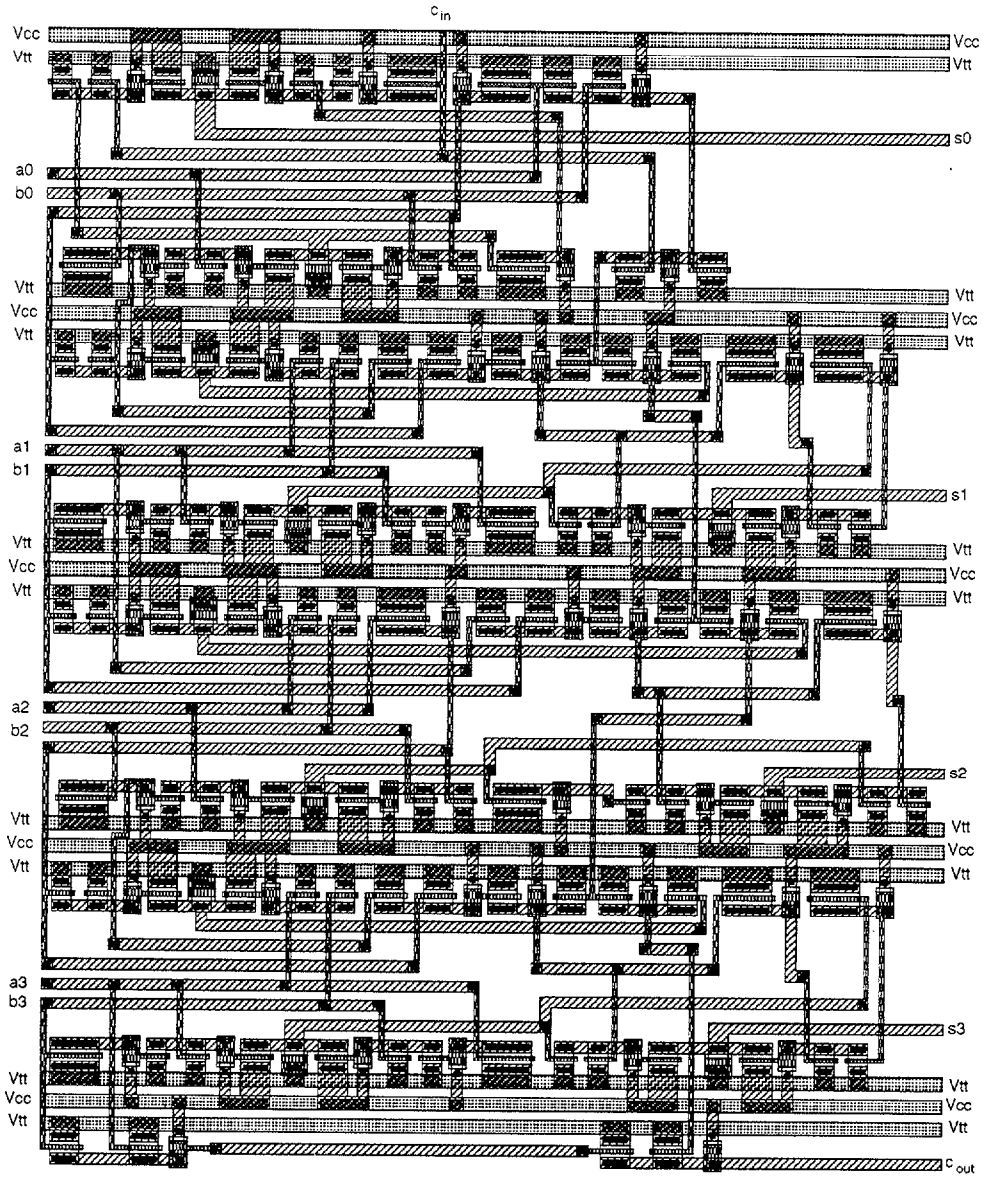


Figura 4.23: Layout del sumador de acarreo anticipado de 4 bits

4.2.4 Sumador de selección de acarreo

En la estructura del sumador de acarreo serie de dos bits, el acarreo de salida del primer sumador completo actúa como acarreo de entrada del segundo. Como el acarreo de entrada del segundo sumador completo solo puede tomar dos valores lógicos, es posible calcular la suma de los bits de mayor peso con dos sumadores completos, el primero con la entrada de acarreo conectada al valor lógico 0, y el segundo con la entrada de acarreo conectada al valor lógico 1, de forma que sea el acarreo de salida del primer sumador completo el que seleccione la salida adecuada. Esta es la idea básica del sumador de selección de acarreo: usar éste como control de un multiplexor que selecciona la salida adecuada en base a su valor lógico.

Esto se muestra en la figura 4.24, donde se representa un sumador de selección de acarreo de 4 bits.

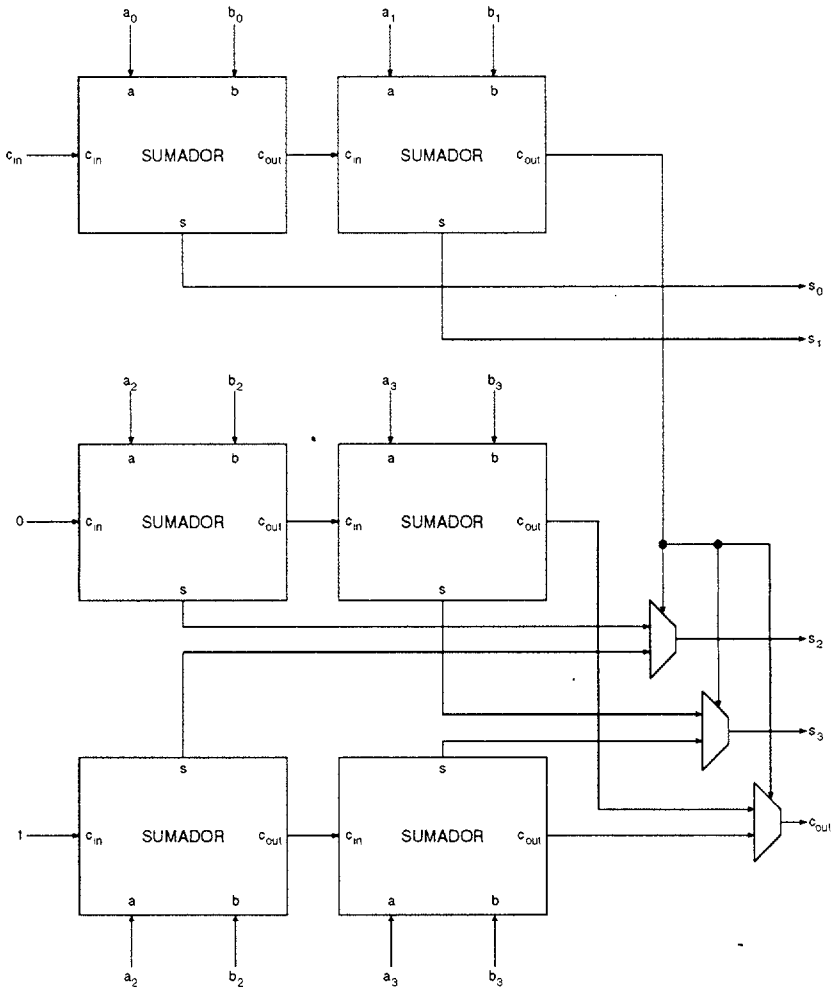


Figura 4.24: Esquema de bloques del sumador de selección de acarreo

Las ecuaciones que definen el sumador completo se pueden simplificar cuando el valor del acarreo de entrada es fijo. De esta forma, cuando el acarreo de entrada tiene un valor 0:

$$\begin{aligned} s &= a \oplus b \\ c_{out} &= a \cdot b \end{aligned}$$

y cuando el acarreo de entrada tiene el valor 1:

$$\begin{aligned} s &= \overline{a \oplus b} \\ \overline{c_{out}} &= \overline{a + b} \end{aligned}$$

En las figuras 4.25 y 4.26 se muestran los layouts implementados para los casos de un acarreo de entrada igual a 0 e igual a 1 respectivamente.

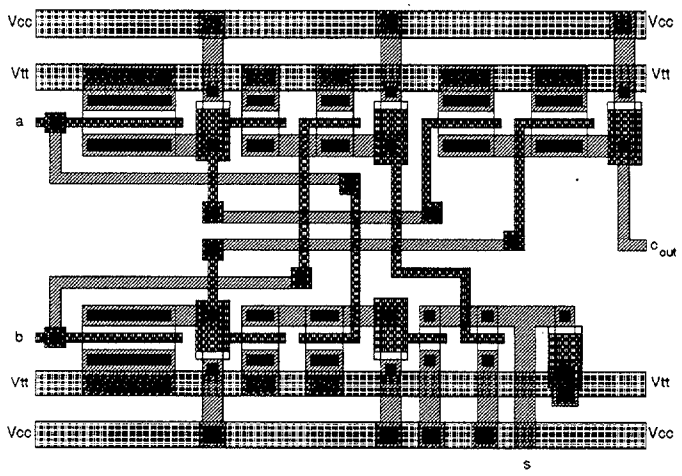


Figura 4.25: Sumador completo para acarreo de entrada 0

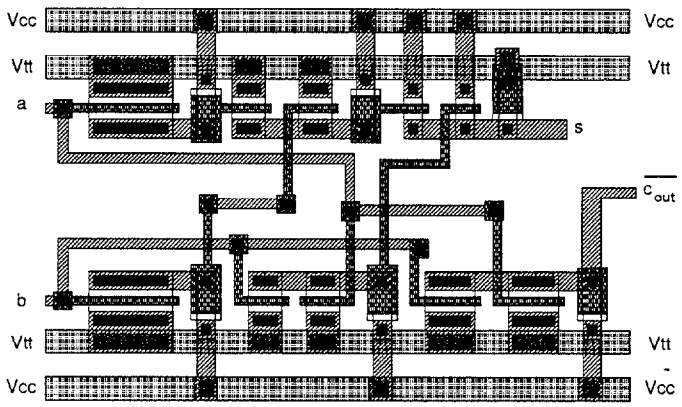


Figura 4.26: Sumador completo para acarreo de entrada 1

El layout del sumador selector de acarreo de 4 bits, donde ya se incluyen las células modificadas para el caso de un acarreo de entrada de un valor lógico constante, se muestra en la figura 4.27.

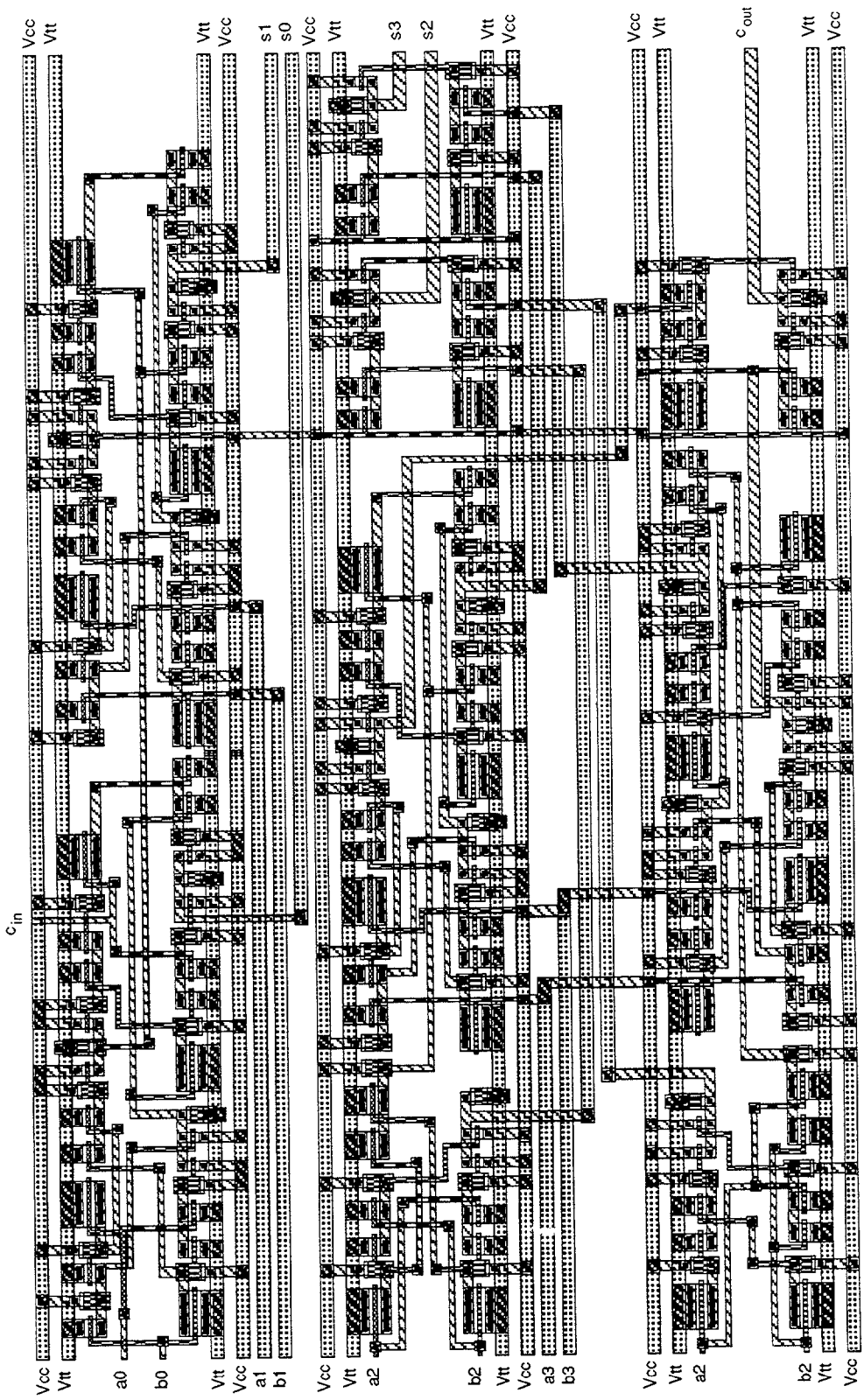


Figura 4.27: Layout del sumador selector de acarreo de 4 bits

4.2.5 Estudio comparativo

En esta sección se realizará una estudio comparativo de los sumadores anteriormente expuestos, basando dicho estudio en tres parámetros principales: retardo, potencia disipada y área ocupada.

En la figura 4.28 se muestra la comparación de los tres sumadores estudiados. En dicha figura se puede observar como, para sumadores de un número pequeño de bits, el sumador de acarreo serie tiene muy buenas prestaciones, pero a medida que aumenta la longitud de las palabras a sumar, el retardo crece de forma notable. Por otro lado, el sumador de selección de acarreo tiene un retardo muy pequeño, pero sus requerimientos tanto de área como de potencia son muy elevados. El término medio viene dado por el sumador de acarreo anticipado. Si bien el retardo se ve sustancialmente reducido con respecto al sumador de acarreo serie, el incremento en área y potencia disipada no es tan dramático como en el caso del sumador de selección de acarreo.

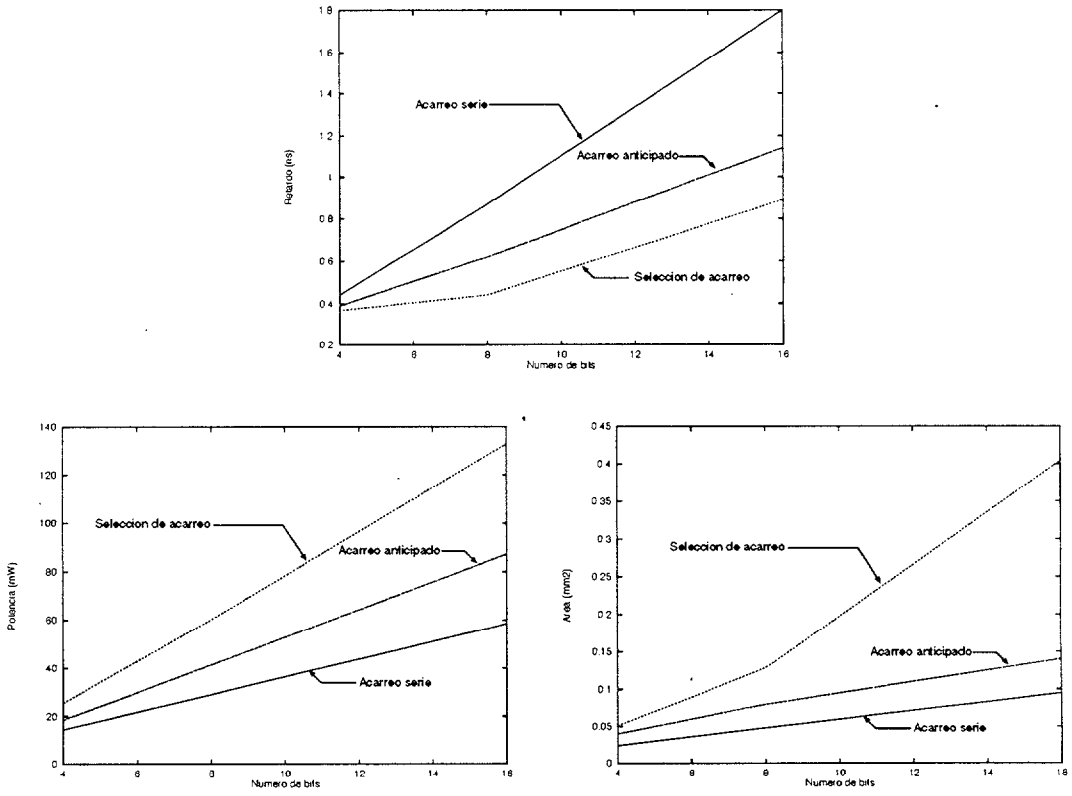


Figura 4.28: Comparación entre los tres sumadores

4.2.6 Tamaño de los sumadores en las estructuras MAC

El tamaño de los sumadores a utilizar en las estructuras MAC viene condicionado por el número de bits con los que se codifican los *coeficientes cosenos* en la ROM. Codificando dichos coeficientes con 16 bits, y añadiendo el bit necesario para realizar una *extensión de signo* se obtienen los 17 bits que forman cada una de las palabras de entrada a los sumadores.

La necesidad de realizar una *extensión de signo* viene reflejada en la figura 4.29, donde se puede observar como, cuando se realiza una suma binaria de dos números con signo, no se puede asegurar cuál es el dígito que determina el signo, ya que si hay *overflow*, el *bit de signo* coincide con el acarreo de salida, pero si no lo hay, el *bit de signo* es el resultado de la suma de los dos bits más significativos. Realizando una *extensión de signo*, se asegura que el *bit de signo* corresponde siempre a la suma de los dos bits más significativos.

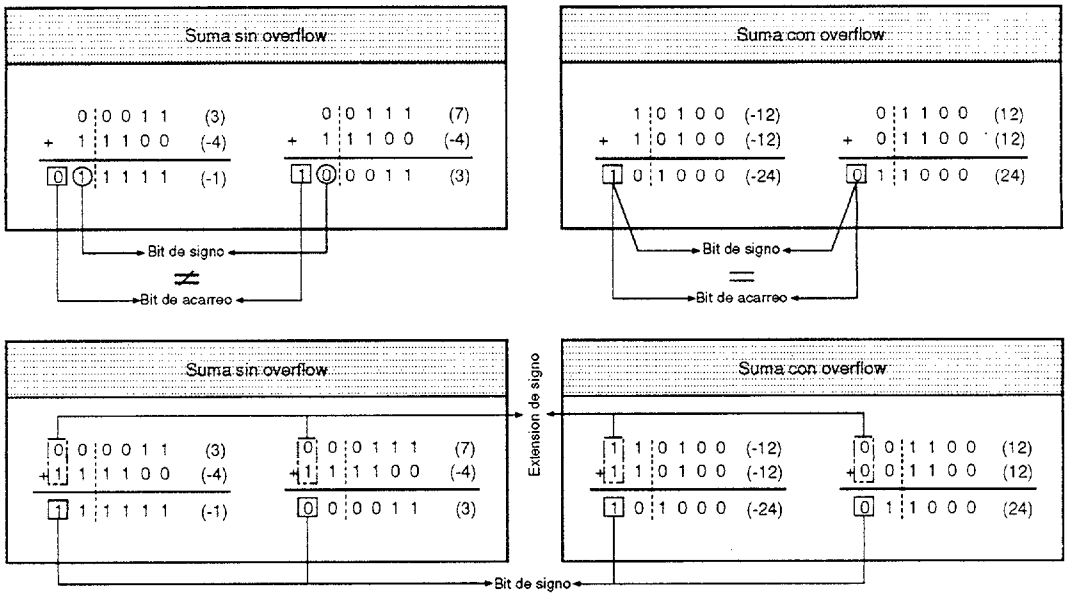


Figura 4.29: Justificación de la extensión de signo

En la tabla 4.1 se pueden observar las caracteríticas de retardo, potencia disipada y área del sumador de acarreo anticipado de 17 bits, tanto para los procesos de dispersión extremos como para el caso típico de funcionamiento.

Dispersión	*Retardo (ns)*	*Potencia (mW)*	*Area (mm^2)*
SS2 ($0°C$)	1.70	61.4	0.167
TT ($75°C$)	1.16	90.9	
FF1H ($100°C$)	1.07	100.7	

Tabla 4.1: Características del sumador de acarreo anticipado de 17 bits

4.3 Estudio e implementación de una ROM

La implementación de la estructura MAC basada en aritmética distribuida conlleva el uso de memorias de solo lectura como elemento de almacenamiento de los coficientes multiplicativos. En el caso de la implementación de la DCT, los coeficientes almacenados son los coeficientes cosenos, para cuya codificación se ha optado por longitudes de palabra de 16 bits.

4.3.1 Estructura básica

La estructura básica de una memoria ROM de N líneas de palabra y M líneas de bit puede verse en la figura 4.30.

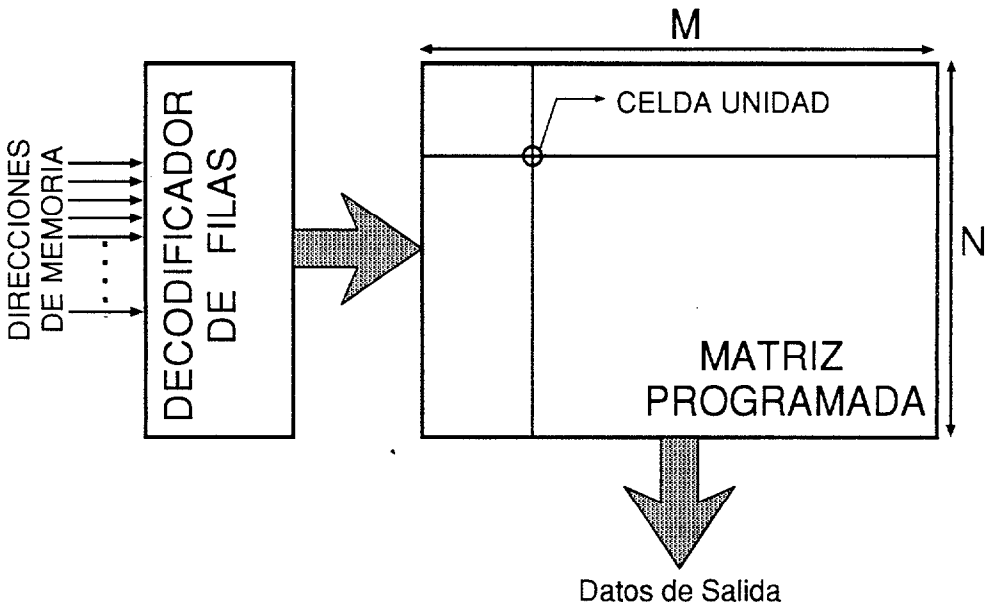


Figura 4.30: Estructura de bloques de una memoria ROM.

En la ilustración se observan dos bloques principales:

- Decodificador de filas.
- Matriz programada.

La matriz programada está formada por una agrupación de células de forma que puedan compartir conexiones entre filas horizontales y columnas verticales. Las líneas horizontales son gobernadas desde fuera de la matriz mediante el decodificador de filas, y se denominan *líneas de palabra.* Las líneas verticales se denominan *líneas de bit.* En las células de la matriz es donde se hallan almacenados los datos, uno por célula.

Aunque esquemáticamente el diagrama de bloques de una memoria ROM es muy sencillo, existen una serie de factores que dificultan la implementación del circuito. Entre esos factores, se puede destacar las corrientes de fuga, que son las que un mayor impacto tienen en la implementación de memorias de solo lectura.

4.3.2 Decodificador de filas

Es el circuito encargado de seleccionar la palabra a leer. Su implementación corresponde al mapeado directo de la ecuación:

$$Selección \quad = \quad \overline{A_0 + A_1 + A_2 + A_3}$$

que equivale a una puerta NOR de 4 entradas.

La línea de *Selección* solo se puede activar con una combinación determinada de entradas. Conectando una combinación de los bits que determinan la posición a leer y sus complementarios, se puede acceder a cada una de las 16 palabras de la matriz.

A la salida de la NOR se coloca un inversor debidamente dimensionado para atacar a una estructura SBFL, ya que la salida del decodificador se enfrenta a la alta capacidad y al alto *fan–out* de la línea de palabra.

La implementación a nivel de layout del decodificador de lectura se puede ver en la figura 4.31.

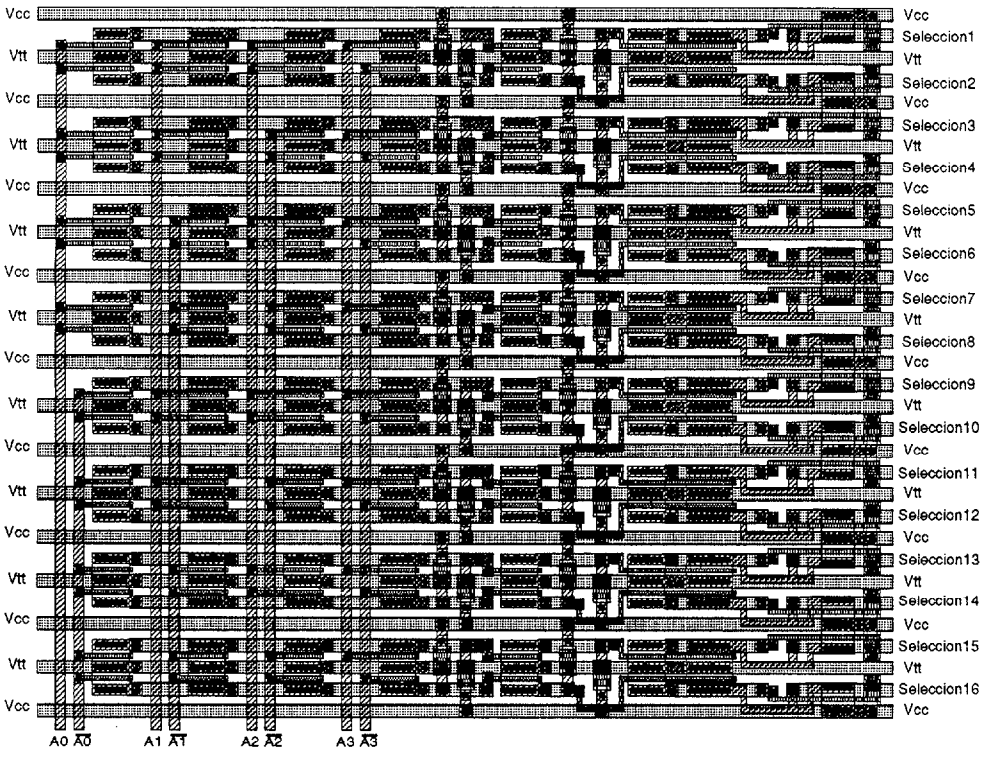


Figura 4.31: Decodificador de filas

4.3.3 Matriz de la ROM

La matriz de la ROM está formada por una agrupación de transistores que forman puertas NOR, una por cada bit de la palabra de salida, con tantas entradas como palabras haya almacenadas, lo que se traduce en 16 puertas NOR de 16 entradas cada

una para el caso de esta implementación. El mayor problema en su implementación son, como se ha dicho anteriormente, las altas corrientes de fuga, que limitan de forma sustancial el número de palabras que pueden ser almacenadas en una ROM.

4.3.3.1 Corrientes de fuga

Las corrientes de fuga son aquellas corrientes residuales que continúan produciendose entre drenador y surtidor cuando el transistor está en corte, o lo que es lo mismo, cuando $V_{gs} < V_t$.

En los MESFETs, estas corrientes de fuga son varios órdenes de magnitud superiores que en los MOSFETs, lo que indica que tendrán una gran influencia en la memoria, en la que cada columna de la matriz será equivalente a una NOR con 16 entradas, de las que solo una estará activa en un instante determinado.

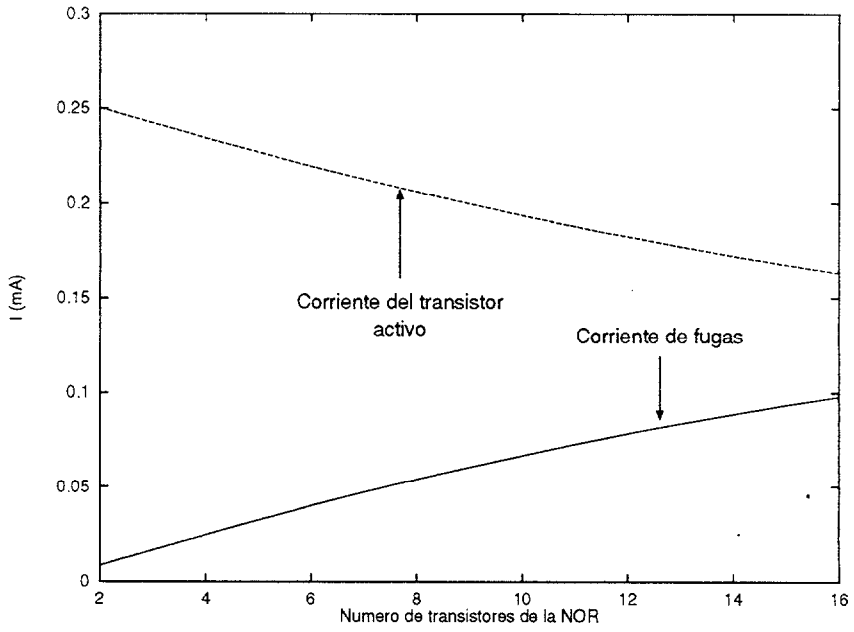


Figura 4.32: Corriente de fugas vs Corriente transistor activo

En la figura 4.32 se representa una comparación entre la corriente que pasa por el transistor cuya puerta está a nivel alto, es decir, que está activo, y la suma de las corrientes de fuga que circulan por el resto de los transistores de la NOR. Como se puede observar en dicha figura, para una ROM de 16 palabras (equivalente a una NOR de 16 entradas) la corriente que circula a través del transistor que está activo tiene un valor cercano a la que circula por los transistores que están en estado de corte.

4.3.3.2 Degradación de los niveles lógicos

Si se toma un inversor con $\beta = 10$, que es el valor recomendado por *Vitesse*, a su salida, tanto los niveles lógicos como los tiempos de subida y bajada son buenos, pero según aumenta el *fan–in*, y sólo se mantiene una entrada a nivel alto, los niveles

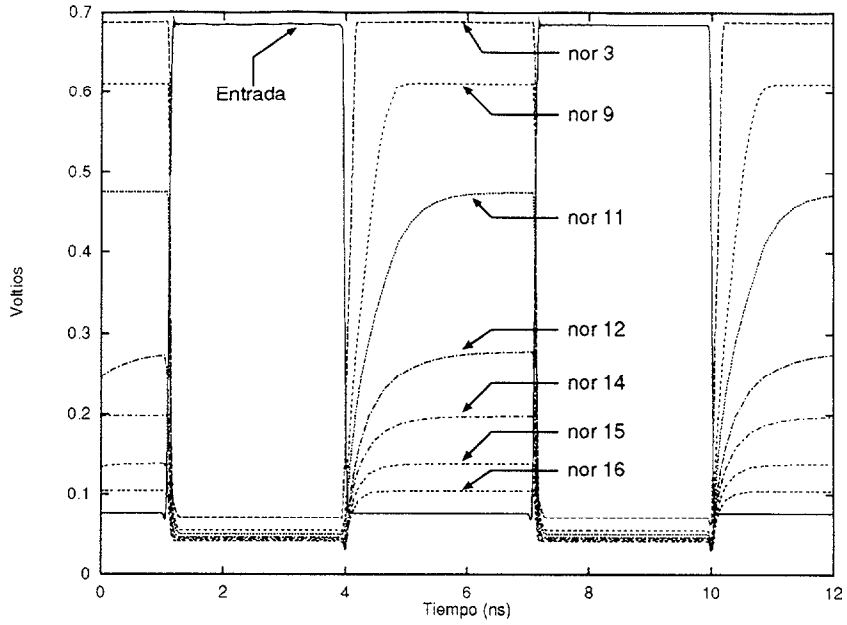


Figura 4.33: Degradación de los niveles lógicos

lógicos se degradan, debido a las corrientes de fuga, como se puede ver en la figura 4.33.

Lo que se representa es la respuesta transitoria de una puerta NOR a 75 $°C$ en proceso típico. Se puede observar como, para una NOR de 9 entradas, el tiempo de subida comienza a ser un problema, ya que de éste depende en gran medida el margen de ruido. Por otro lado, a partir de una NOR de 11 entradas, es imposible utilizar este esquema, porque los niveles lógicos ya no son aceptables. Debe tenerse en cuenta que en esta simulación no se ha incluido la capacidad de línea a la salida de la NOR. Eso indica que los tiempos de subida serán aún peores que los representados.

Para reducir el efecto de las corrientes de fuga, se puede disminuir el ancho del canal de los transistores de enriquecimiento, reduciendo así las corrientes de fuga, o bien aumentar el ancho del canal del transistor de deplexión, aumentando así la corriente que fluye a través de dicho transistor. La disminución del ancho del canal de los transistores de enriquecimiento y el incremento en el ancho del canal del transistor de deplexión equivalen a una disminución del parámetro β, y será *directamente proporcional* al número de transistores que contribuyan con su corriente de fuga a la corriente de fuga total de la puerta.

4.3.3.3 Solución de la beta modificada

Para obtener un valor de β que se adapte a las necesidades propias de esta adaptación, se aplica una fórmula que relaciona el nuevo valor de dicho parámetro (β^*) con las corrientes que circulan por los transistores de la NOR. Esta relación viene dada por:

$$\beta^* = \frac{1}{(1+p)\cdot\gamma+1}\cdot\beta \tag{4.6}$$

donde β es el factor de dimensionamiento dado por Vitesse, p es un parámetro empírico relacionado con la razón entre las pendientes de subida y bajada de la señal de salida

y γ es un parámetro denominado FCC (Factor Cociente de Corrientes) cuya expresión viene dada por

$$\gamma = I_L/I_x$$

Este parámetro relaciona las corrientes de fuga, I_L, con la corriente de drenador del transistor activo, I_x.

Para calcular γ, se divide la corriente que circula por los transistores que están en corte y la que circula por el transistor activo:

$$\gamma = I_L/I_x = 0.6$$

Tomando $p = 1$, es decir, asumiendo que el valor de la pendiente de la señal de salida es próxima en las transiciones de carga y en las de descarga, y aplicando la ecuacion 4.6, se obtiene:

$$\beta^* = 4.5$$

que, debido a las simplificaciones que se realizan en la deducción de dicho parámetro, no es el valor óptimo de la relación de aspecto, pero si es un buen punto de partida para el dimensionamiento de los transistores.

4.3.3.4 Programación

La matriz programada consiste en una serie de filas y columnas. Las primeras forman las denominadas *líneas de palabra*, y las segundas las *líneas de bit*. Un bit se almacena en una ROM mediante la ausencia o presencia de un camino entre una fila y una columna. La ausencia de camino o conexión se consigue simplemente omitiendo la conexión que une la línea de palabra con la línea de bit.

En la matriz implementada con puertas NOR, una columna queda a nivel bajo cuando alguna fila unida a ella mediante un transistor está a nivel alto. Cuando una fila seleccionada alcanza un nivel alto, todos los transistores con puertas conectadas a esa línea se ponen en conducción. Las columnas a las cuales están conectados pasan a un nivel bajo. Las restantes columnas continúan a nivel alto gracias a los dispositivos de carga. Por tanto, un "uno" almacenado se define como la ausencia de un transistor.

El esquema general, a nivel de transistores de la ROM se puede ver en la figura 4.34.

4.3.3.5 Layout

Para la realización del layout se ha optado por la disposición en columnas de los transistores de la NOR, colocando en la parte inferior el transistor de carga. Un detalle del layout se puede ver en la figura 4.35.

La programación de la matriz se muestra en la figura 4.36. Cuando la *línea de palabra* está a nivel alto, la salida de la *línea de bit 1* pasa a nivel alto, debido a que el transitor de enriquecimiento no está conectado (debido a la ausencia de vías en puerta y drenador), mientras que la salida de la *línea de bit 2* pasa a nivel bajo, debido a la existencia de un transistor de enriquecimiento que actúa como *pull-down*.

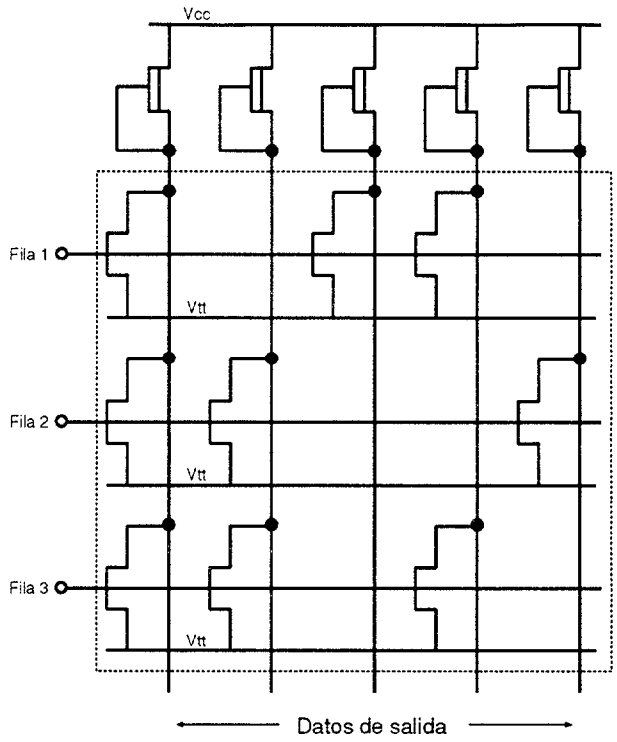


Figura 4.34: Esquema general a nivel de transistores de una memoria ROM

4.3.4 Prestaciones

Partiendo del valor inicial obtenido del factor de dimensionamiento equivalente a una $\beta = 4.5$ y mediante simulaciones se ha optado por la implementación de la ROM usando una $\beta = 4$, con la que se han alcanzado las mejores prestaciones, obteniendo los siguientes resultados:

Dispersión	*Potencia (mW)*	*Tiempo subida (ps)*	*Tiempo bajada (ps)*	*Retardo (ns)*
SS2 ($0°C$)	31.9	250	205	0.57
TT ($75°C$)	48.5	218	182	0.38
FF1H ($100°C$)	57.7	327	175	0.34

En la figura 4.37 se muestra la salida de la ROM de 16 palabras para la β elegida.

El área ocupada así como el número de transistores necesarios para esta implementación vienen reflejados en la siguiente tabla:

Area (mm^2)	*Nº Transistores*
0.083	464

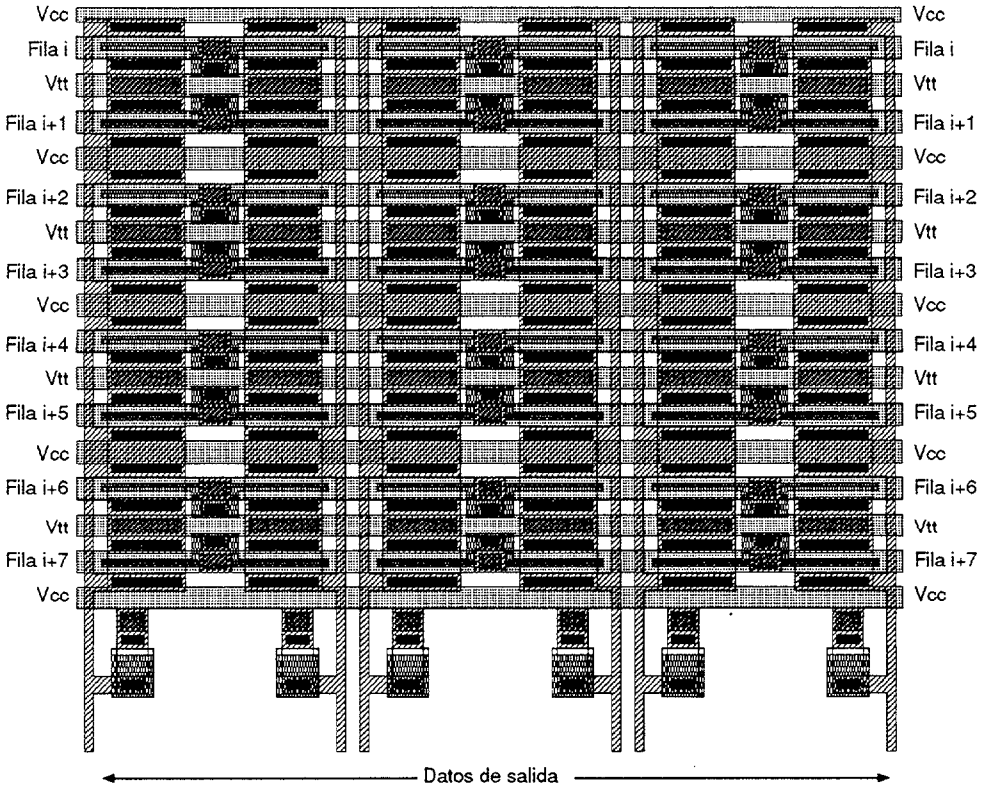


Figura 4.35: Detalle del layout de la memoria ROM

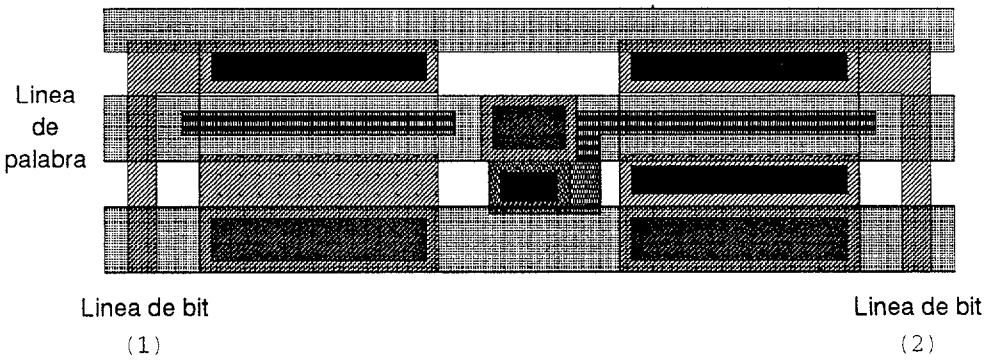


Figura 4.36: Programación de la memoria ROM

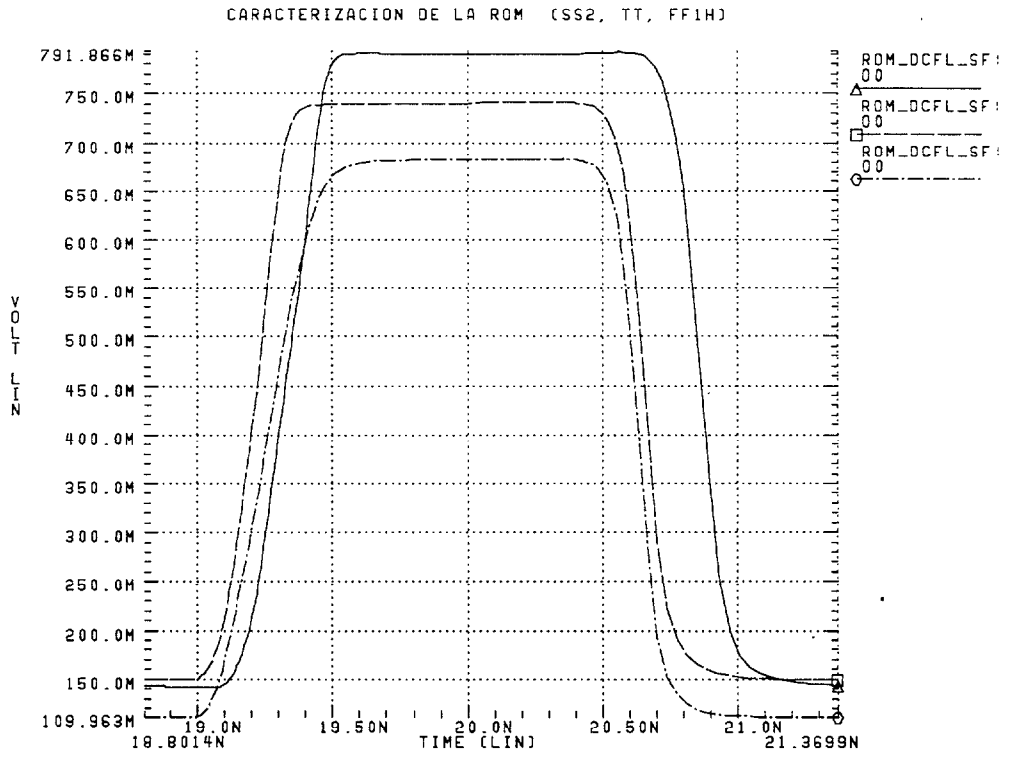


Figura 4.37: Caracterización de la ROM de 16 palabras de 16 bits

4.4 Estudio e implementación de una RAM dinámica

Típicamente, el diseño de estructuras de memoria en Arseniuro de Galio se ha basado en células *estáticas*, usando la estructura clásica de dos inversores DCFL realimentados. El resultado son memorias con alta disipación de potencia y que ocupan mucha área. Las memorias *dinámicas* evitan el problema de la alta disipación, a la vez que disminuyen el área requerida a costa de limitar el tiempo de almacenamiento del dato. Las DRAM tienen su aplicación en aquellos casos en los que el tiempo necesario de almacenamiento de los datos sea relativamente corto. En otros casos se requieren sistemas de refresco que pueden originar estructuras más o menos complejas.

La aplicación de una memoria de almacenamiento dinámico en el cómputo de la DCT bidimensional, implementada como dos 1–D DCT independientes es inmediata. Tras aplicar la primera transformación unidimensional los datos han de ser almacenados el tiempo suficiente como para que puedan ser leídos por el bloque que realiza la segunda 1–D DCT.

En el caso estudiado, la escritura de los datos se realiza de forma paralela, es decir, en un sólo ciclo se almacenan todos los bits de un número determinado de palabras, mientras que la lectura se realiza de forma serie, leyendo en un ciclo un número determinado de los bits de las todas la palabras almacenadas en una determinada columna de la matriz.

4.4.1 Diseño estructural

Tal y como se ha visto en el capítulo dedicado al estudio arquitectural, para evitar el uso de un segundo banco de registros que actuarían a modo de conversor paralelo/serie a la salida de la DRAM, la lectura de los datos ha de hacerse de modo serie, leyendo el número de bits que corresponde según la partición de la estructura MAC que se haya elegido para implementar la segunda 1–D DCT.

En el caso más simple de implementación de la 2–DCT en el que ambas estructuras MAC no están particionadas, en el que los datos han de estar más tiempo almacenados y, por tanto, el *peor caso* respecto al factor crítico en el diseño de la memoria dinámica, la DRAM ha de permitir una *escritura paralela* de 12 bits y una *lectura serie* de 8 bits, uno de cada fila.

La estructura diseñada a tal efecto se puede ver en la figura 4.38. Los números incluidos dentro de cada casilla de la DRAM indican la posición original que ocupaba el dato en una columna determinada, que es el que va a ocupar en la fila correspondiente cuando se haya realizado la transposición, teniendo en cuenta que en la primera fila se almacena la primera columna, en la segunda fila, la segunda columna, y así sucesivamente.

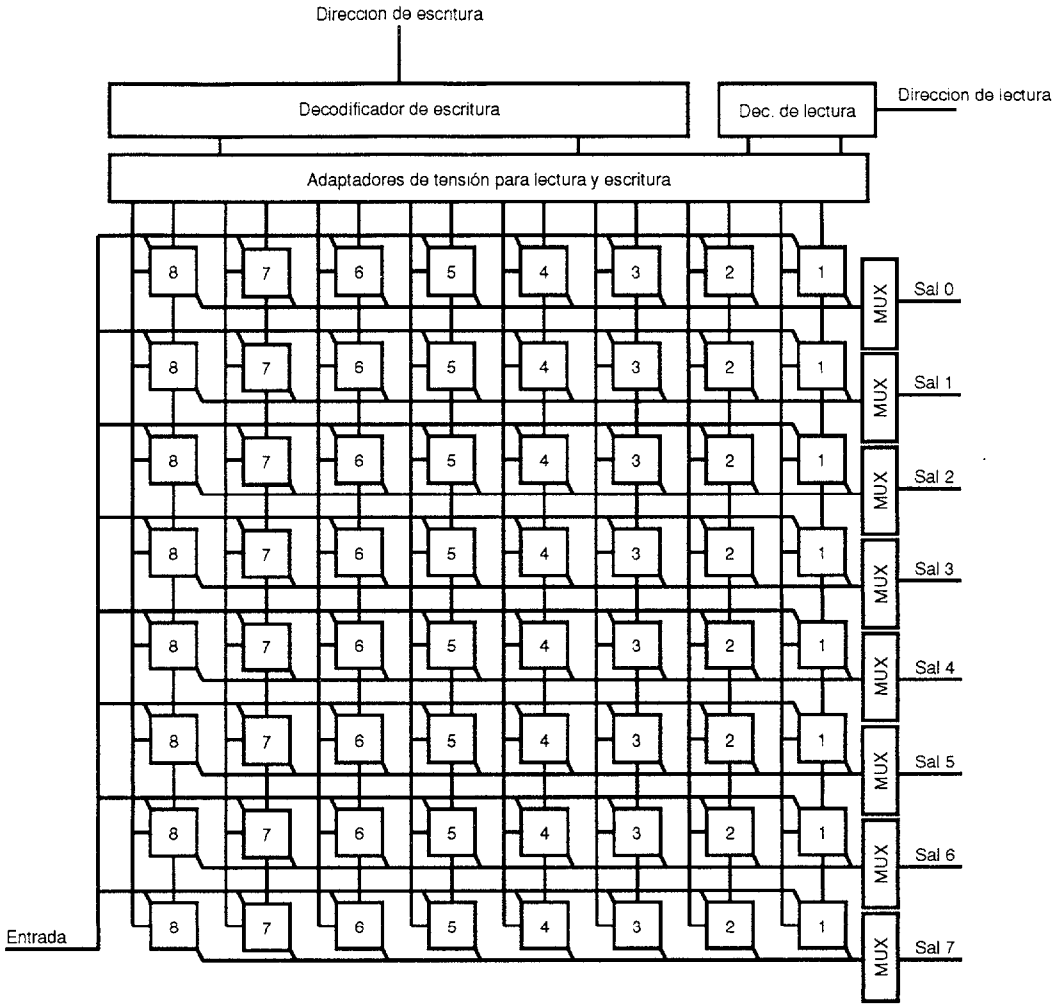


Figura 4.38: Estructura de la memoria

Para su realización, será necesario un estudio previo de la *célula básica de memoria*, que consiste en un elemento de almacenamiento situado entre dos transistores de paso, el de entrada, controlado por la *señal de escritura*, y el de salida, controlado por la *señal de lectura* (figura 4.39).

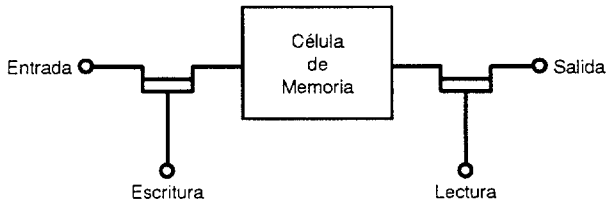


Figura 4.39: Estructura básica de una célula de memoria

4.4.1.1 Transistores de paso

Debido a que, para unas dimensiones y para una tensión de puerta–surtidor determinadas, la corriente que permite pasar un transistor MESFET de deplexión (DFET)

cuando está operando en la región de saturación es mayor que la que permite pasar un transistor MESFET de enriquecimiento (EFET) que opera en la misma región, se usarán los primeros como transistores de paso. Esto se demuestra en la figura 4.40, donde se representa la variación de la *corriente drenador–surtidor* frente a variaciones de la *tensión de puerta* para los transistores DFET y EFET, manteniendo las mismas dimensiones en ambos.

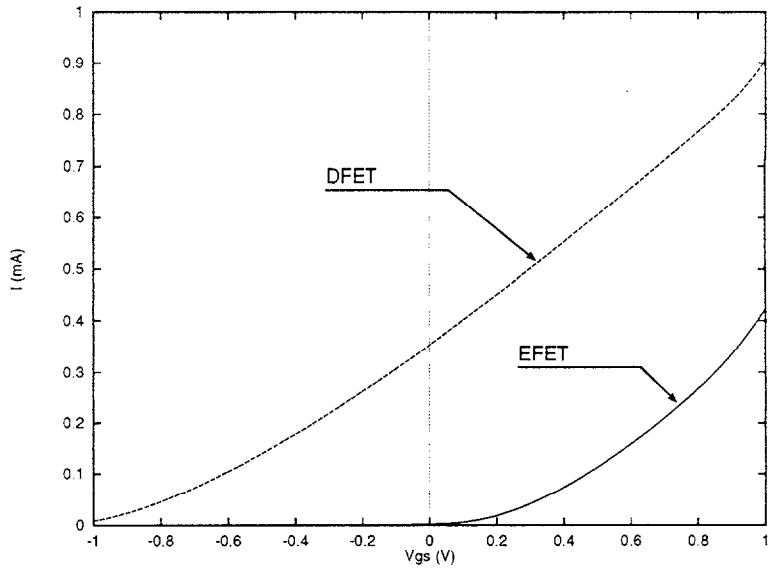


Figura 4.40: Corriente del DFET vs Corriente del EFET

Un segundo motivo, quizás más importante que el anterior, para elegir los transistores tipo D como transistores de paso se puede deducir de las ecuaciones que definen el punto de operación del transistor. Para que el transistor esté saturado:

$$V_{gs} = V_g - V_s \geq V_t$$

Por tanto, la tensión de surtidor para que el transistor esté saturado:

$$V_s \leq V_g - V_t$$

Pero, por otro lado, la tensión puerta–surtidor está limitada por el diodo Schottky:

$$V_{gs} = V_g - V_s \leq V_{sch}$$

Eso implica que la tensión máxima, en la puerta del transistor, para que el transistor esté en saturación es:

$$V_g \leq V_s + V_{sch}$$

Llamando V_0 a la tensión almacenada cuyo valor lógico es 0, y partiendo de la ecuación anterior, se obtiene la máxima tensión que se puede aplicar a la puerta del transistor de paso:

$$V_{g_{max}} = V_{s_{min}} + V_{sch} = V_0 + V_{sch}$$

y por tanto:

$$V_{s_{max}} = V_{g_{max}} - V_t = V_0 + V_{sch} - V_t$$

Consecuentemente, la máxima excursión de tensión que se puede producir en el surtidor del transistor de paso es:

$$V_{s_{max}} - V_{s_{min}} = V_{sch} - V_t$$

A partir de esta ecuación es fácil deducir que la excursión de tensión en el surtidor del transistor MESFET aumenta según disminuye la tensión umbral del transistor. Para incrementar el margen de ruido y conservar la integridad de la señal es preferible tener una excursión lógica lo mayor posible en el surtidor del transistor de paso. Como la tensión del diodo Schottky es fija (aproximadamente 0.6 voltios), y la tensión umbral del D–MESFET es menor que la del E–MESFET, es conveniente usar los primeros como transistores de paso.

El uso de transistores de deplexión (*Normally–ON*) como transistores de paso conlleva el uso de dos lógicas en el mismo circuito: lógica positiva para los datos y lógica negativa para el control, necesaria para la conmutación de los transistores de paso. Eso implica el uso de 3 buses de alimentación en el circuito: +2, 0 y -2 voltios.

El uso de lógica negativa para el control se deduce de las ecuaciones que definen el comportamiento del transistor.

$$\begin{array}{lcll} V_{gs} - V_t & < & 0 & \textit{Corte} \\ V_{gs} - V_t & \geq & 0 & \textit{Saturación} \end{array}$$

Teniendo en cuenta las tensiones umbrales dadas por Vitesse para la temperatura nominal $25°C$:

Dispositivo	*SS2*	*TT*	*FF1H*
Deplexión	-0.735	-0.825	-0.915

y la fórmula que indica la variación de la tensión umbral con respecto a la temperatura:

$$V_{t0} = V_{t0} - TCV * \Delta t$$

Donde:

TCV=Coeficiente de variación de la tensión umbral con la temperatura.
Δt = Variaciones de la temperatura con respecto a la nominal.

es posible calcular las tensiones umbrales para el resto de las temperaturas:

Dispersión	*Temperatura (°C)*	*Tensión umbral (V)*
SS2	0	-0.71
	25	-0.74
	50	-.76
	75	-.79
	100	-.82
TT	0	-0.80
	25	-0.83
	50	-0.85
	75	-0.88
	100	-0.91
FF1H	0	-0.89
	25	-0.92
	50	-0.94
	75	-0.97
	100	-0.99

Si se asume lógica positiva de entrada, que corresponde a unos valores de tensión entre 0.1 y 0.7 voltios aproximadamente, sustituyendo en las ecuaciones anteriores, y tomando los casos extremos en lo que a la tensión umbral se refiere para calcular el valor de la tensión de puerta se llega a:

$$\begin{array}{lcll} V_g & < & -0.89\text{v} & \textit{Corte} \\ V_g & \geq & 0\text{v} & \textit{Saturación} \end{array}$$

Estos valores de tensión no se pueden conseguir con las estructuras típicas, que en lógica negativa producen salidas entre -1.9 y -1.3 voltios. Será necesario el uso de estructuras especiales –estructuras adaptadoras– con alimentaciones entre 2 y -2 voltios para poder controlar los transistores de paso.

4.4.1.2 Elemento de almacenamiento

La célula de memoria dinámica más utilizada en el diseño de DRAM, en tecnología de Arseniuro de Galio, aparece en la figura 4.41(a). La principal ventaja de esta solución es el bajo consumo de potencia y la reducida área que requiere la célula. Sin embargo, existe el inconveniente de tener que generar y distribuir una señal de *precarga* previa a la lectura del dato, que hace que el tiempo de acceso sea elevado, alcanzando frecuencias del orden de *300MHz*. Para paliar este problema, se han desarrollado y comparado las caraterísticas de las estructuras que se observan en la figura 4.41, que se denominarán *estructura inversora* –figura 4.41(b)– y *estructura no–inversora* –figura 4.41(c)–. La célula inversora equivale a un inversor DCFL y la no–inversora consiste básicamente en un seguidor de tensión. El cambio del transistor de enriquecimiento (cuya puerta actúa como nodo de almacenamiento), que pasa de actuar como *pull–down*, a actuar como *pull–up*, se traduce en una disminución de la polarización directa del diodo Schottky que forma la puerta del transistor. Como la corriente a través del diodo Schottky débilmente polarizado es menor que la que pasa a través del diodo cuando está fuertemente polarizado, se puede esperar un incremento en el

tiempo de almacenamiento.

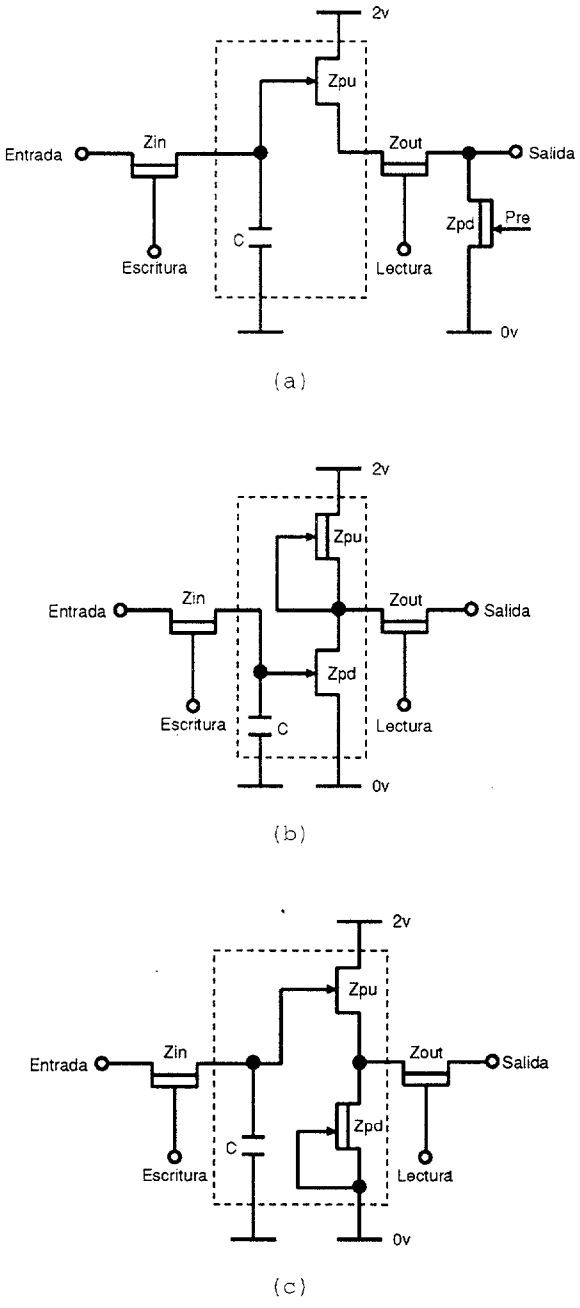
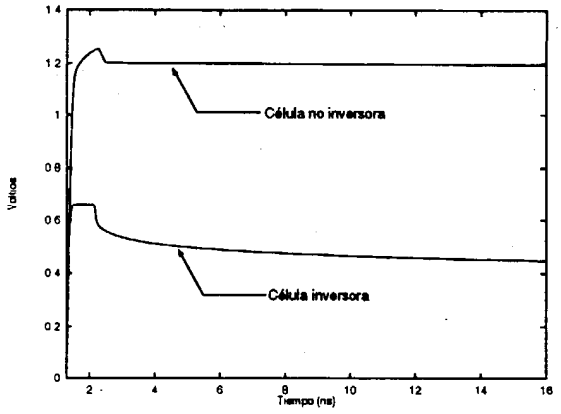


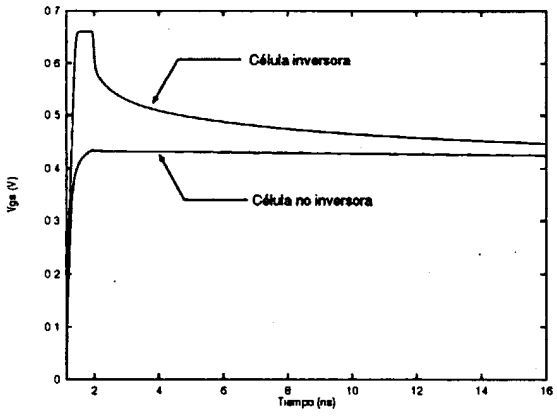
Figura 4.41: (a) Célula típica (b) Célula inversora (c) Célula no–inversora

En la figura 4.42(a) se puede observar cómo se comporta el nodo de almacenamiento de la estructura inversora y de la no–inversora. En la figura 4.42(b) se muestra la tensión puerta–surtidor de los transistores que actúan como nodo de almacenamiento en las células inversora y no–inversora. En el caso de la estructura inversora, se produce una alta tensión puerta–surtidor, lo que origina una alta corriente de fugas. Esta corriente de fugas produce la caída de la tensión en el nodo de almacenamiento, hasta llegar al punto en que dicha tensión se estabiliza debido a la disminución de la polarización del diodo asociado a la puerta. Por este motivo se opta por la implemen-

tación de la memoria usando la estructura *no-inversora.*



(a)



(b)

Figura 4.42: (a) Tensión en el nodo de almacenamiento (b) Tensión puerta-surtidor

En la figura 4.43 se muestra el layout de la célula de memoria implementada, en la cual se incluye una capacidad de 20fF. Como se puede apreciar, esta capacidad asociada a la puerta del transistor de enriquecimiento, se consigue mediante la superposición de *Metal 2* "por encima" de la célula, sin necesidad de aumentar el tamaño de la misma.

En la figura 4.44(a), se puede ver como varía la tensión del nodo de almacenamiento con el tiempo, y en la figura 4.44(b) se muestra la salida del elemento de almacenamiento. En esas figuras es fácil observar que la pendiente de caída de la tensión del nodo de almacenamiento no es "brusca", y que la salida del elemento de almacenamiento es lo suficientemente alta como para que no pierda su significado lógico al pasar por el transistor de paso de lectura, de tal forma que la salida se puede considerar válida hasta aproximadamente $1\mu s$.

La corriente que consume la célula viene reflejada en la siguiente tabla, para distintos procesos de dispersión y diversas temperaturas, además de la corriente para el caso de dos inversores DCFL, que formarían una memoria estática, dimensionados

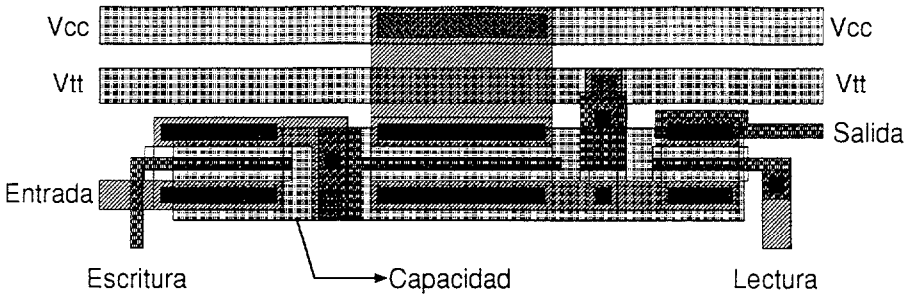


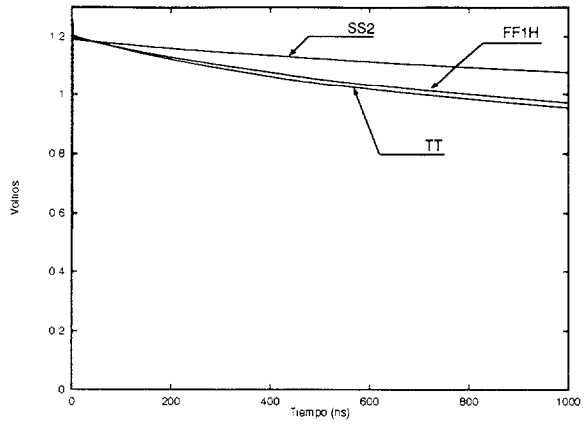
Figura 4.43: Layout de la célula de memoria

con $\beta = 10$, valor recomendado por el fabricante para obtener un buen margen de ruido.

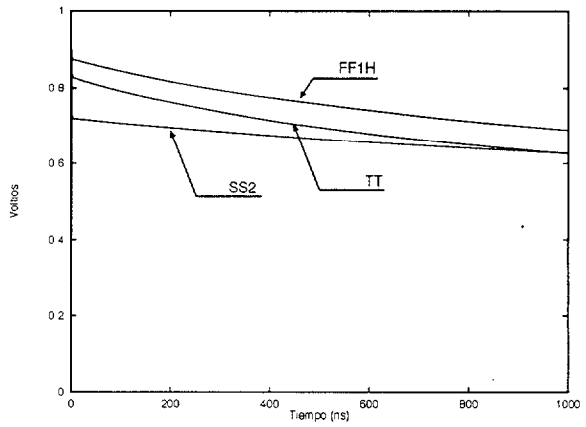
Dispersión	*Temperatura (°C)*	*Corriente Cél. Dinámica (mA)*	*Corriente Cél. Estática (mA)*
SS2	0	0.1	0.18
	25	0.11	0.19
	50	0.12	0.20
	75	0.13	0.21
	100	0.14	0.22
SS1H	0	0.12	0.20
	25	0.12	0.21
	50	0.13	0.22
	75	0.14	0.23
	100	0.15	0.24
TT	0	0.14	0.24
	25	0.14	0.25
	50	0.15	0.26
	75	0.15	0.27
	100	0.16	0.28
FF1	0	0.14	0.26
	25	0.15	0.27
	50	0.16	0.28
	75	0.16	0.29
	100	0.16	0.29
FF1H	0	0.15	0.28
	25	0.16	0.29
	50	0.17	0.3
	75	0.17	0.3
	100	0.17	0.31

Como se puede observar, el consumo de corriente, y por tanto, la potencia disipada, es mucho menor en la célula de memoria dinámica que en la célula estática, llegando a ser, en un caso típico, del orden de un 45% inferior, lo que indica que la potencia disipada en una memoria implementada con este tipo de células será sensiblemente inferior a la implementada con células estáticas.

El dimensionamiento de los transistores de la célula básica ha seguido la siguiente secuencia:



(a)



(b)

Figura 4.44: (a) Nodo de almacenamiento (b) Salida del seguidor de tensión

- Dimensionamiento del transistor de paso de entrada: se dimensiona para que permita pasar una corriente tal que el nivel de tensión en el nodo de almacenamiento –puerta del transistor de enriquecimiento del seguidor de tensión– tenga el nivel de tensión necesario para que a la salida del transistor de enriquecimiento se obtenga un buen nivel de tensión, ya que éste dependerá de la tensión V_{gs} de dicho transistor. En la figura 4.45(a) se puede observar como varía la tensión del nodo de almacenamiento frente a variaciones de la relación de aspecto del transistor de paso.
- Dimensionamiento de los transistores del seguidor de tensión: se dimensionan para tener un valor lógico lo suficientemente alto a la salida del seguidor de tensión como para que la degradación que sufre la señal al pasar por el transistor de paso de lectura no tenga influencia en su significado lógico, así como para que produzca la suficiente corriente como para alimentar una estructura SBFL. En la figura 4.45(b) se puede observar como varía la tensión del nodo de de salida del seguidor de tensión frente a las variaciones de la relación de aspecto del transistor de *pull-up*. El transistor de *pull-down* se dimensiona de forma que consuma la menor corriente posible, ya que su única función es fijar el nivel lógico 0 cuando el transistor de *pull-up* esté en

estado de corte.

- Dimensionamiento del transistor de paso de salida: se dimensiona para que deje pasar una corriente que permita "atacar" a una estructura SBFL, considerando la corriente que produce el seguidor de tensión que actúa como elemento de memoria. En la figura 4.45(c) se puede observar como varía la tensión de salida frente a las variaciones de la relación de aspecto del transistor de paso.

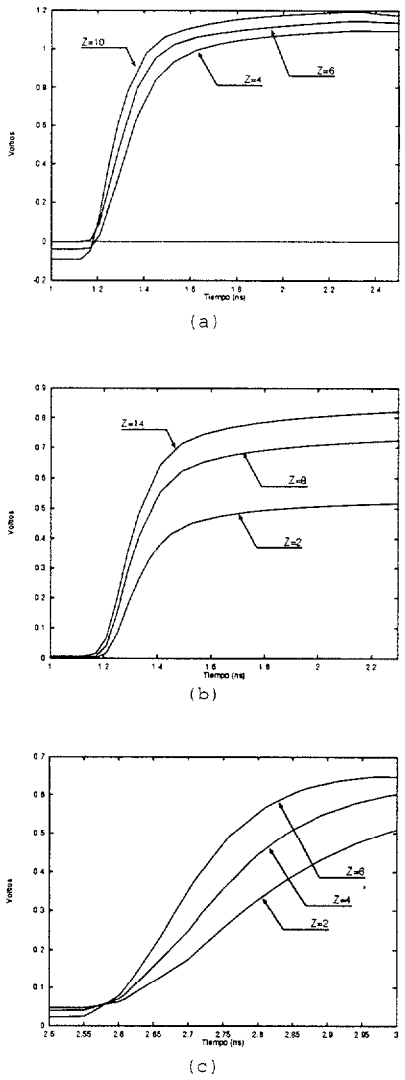
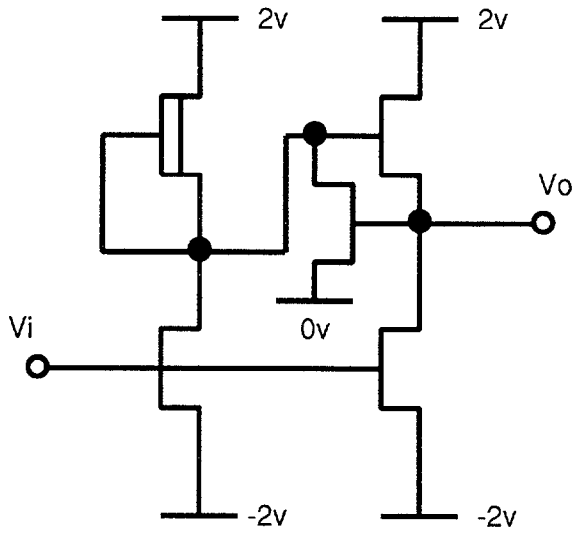


Figura 4.45: (a) Tensión del nodo de almacenamiento (b) Tensión del nodo previo al transistor de paso de salida (c) Salida

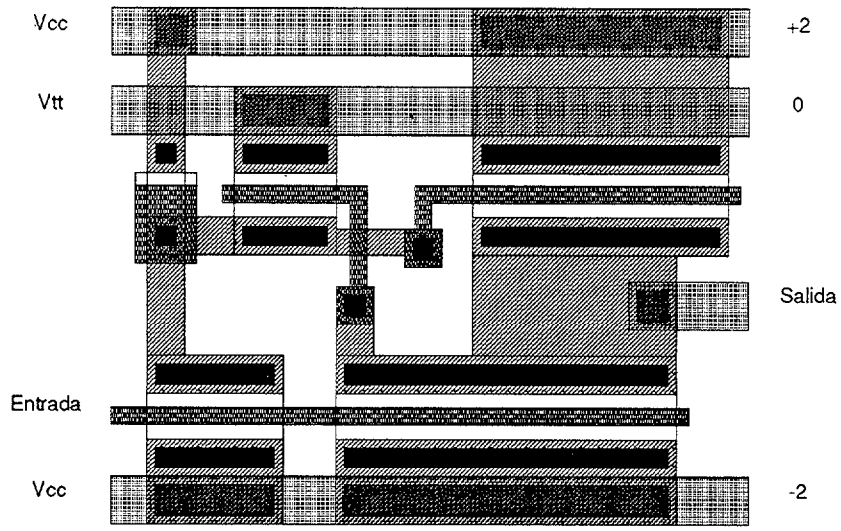
4.4.1.3 Adaptador

El adaptador es el circuito encargado de realizar el paso de lógica negativa (entre -1.9v y -1.3v) a los valores necesarios para el control de los transistores de paso.

El esquema a nivel de transistores del adaptador usado se muestra en la figura 4.46(a). Un ejemplo del layout para esta estructura se puede observar en la figura 4.46(b).



(a)



(b)

Figura 4.46: (a) Esquema de transistores del adaptador de tension (b) Layout

El funcionamiento de esta estructura ha de estudiarse en los dos casos posibles:

- Entrada de un valor lógico 1: la entrada de un valor lógico 1, que corresponde a un nivel de tensión de -1.35 voltios aproximadamente, hace que los transistores de *pull-down* estén en saturación, condicionando de esta mane-

ra la salida de la etapa de adaptación, y fijándola a un nivel bajo, cercano a los -1.9 voltios.

- Entrada de un valor lógico 0: la entrada de un valor lógico 0, que corresponde a un nivel de tensión de -2 voltios aproximadamente, hace que los transistores de *pull-down* estén cortados, por lo que la salida viene condicionada únicamente por los transistores de *pull-up*. La salida de la primera etapa y la salida global pasan a nivel alto. Cuando la salida supera la tensión umbral de los EFETs, el transistor conectado a masa actúa como *pull-down*, limitando la tensión del nodo de salida.

Los niveles de salida del adaptador se pueden ver en la figura 4.47, donde se pueden observar las variaciones de los niveles alto y bajo para varios parámetros tecnológicos y a varias temperaturas. Es fácil observar que con estos valores de tensión se alcanzan sobradamente los necesarios para controlar los transistores de paso.

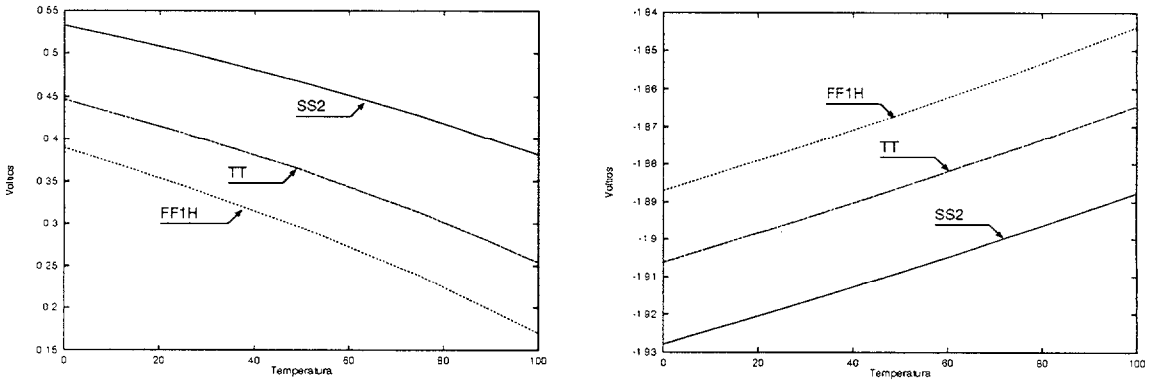


Figura 4.47: Variaciones de los niveles alto y bajo a la salida del adaptador

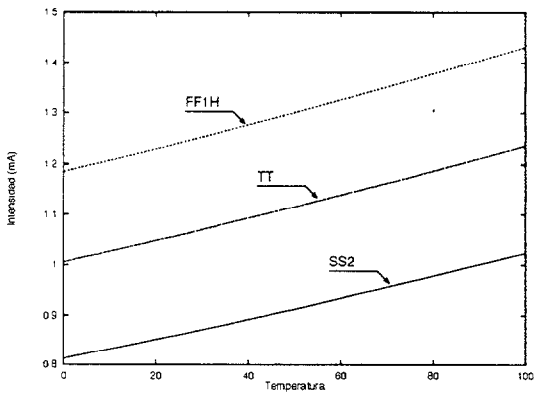


Figura 4.48: Corriente consumida por el adaptador de tensión

El consumo de corriente del adaptador es alto, como se puede ver en la figura 4.48. Además, el adaptador tiene tres tensiones de alimentación, llegando a una diferencia máxima de potencial de 4 voltios, lo que hace que la potencia disipada sea bastante alta, debido, por un lado a la corriente consumida, y por otro, a la alta diferencia de potencial.

4.4.1.4 Decodificadores

La *decodificación de direcciones* para escribir o leer una posición de memoria viene definida por la siguiente ecuación:

$$Selección = \overline{A_0 + A_1 + A_2 + \ldots + A_n + habilitación}$$

que equivale a una puerta NOR de $n+2$ entradas, siendo $n+1$ el número de señales de selección de direccion.

Cuando la línea de *Selección* esté activa (todas sus entradas a 0), se accederá a la posición de memoria definida por la palabra de dirección (A_0, A_1, A_2, ..., A_n). La línea de *Selección* sólo se puede activar con una combinación determinada de entradas. Conectando una combinación de los bits que determinan la posición a leer o escribir y sus complementarios, se puede acceder a todas las posiciones de memoria. La señal de *Selección* sólo se puede activar cuando se lo permita la señal de *habilitación*.

▷ Decodificador de escritura

Es el decodificador encargado de seleccionar la célula de memoria en la que se va a almacenar el dato de entrada.

Es un decodificador de 7 entradas (6 de selección de dirección y 1 de control de escritura) y 64 salidas.

Su implementación directa llevaría a una NOR de 7 entradas, lo cual es un alto *fan-in* cuando se diseña en Arseniuro de Galio, por lo que se divide el decodificador en dos *predecodificadores* (figura 4.49), con una NOR de 3 entradas y otra de 4 entradas, cuyas salidas van a una AND, implementada como 2 inversores y una NOR de 2 entradas convenientemente dimensionada, ya que ataca a una estructura SBFL, necesaria para el posterior ataque al *adaptador*.

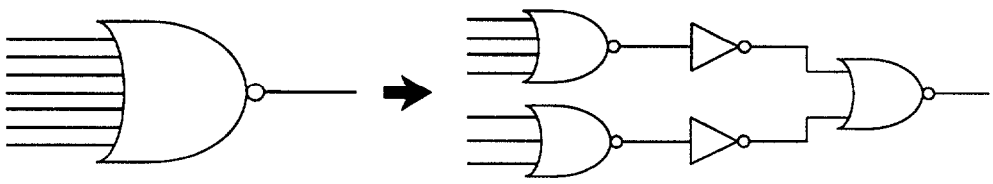


Figura 4.49: Conversión de la NOR del decodificador

En la figura 4.50 podemos ver la célula básica implementada.

▷ Decodificador de lectura

Es el decodificador encargado de seleccionar la célula de memoria de la que se va a leer el dato previamente almacenado.

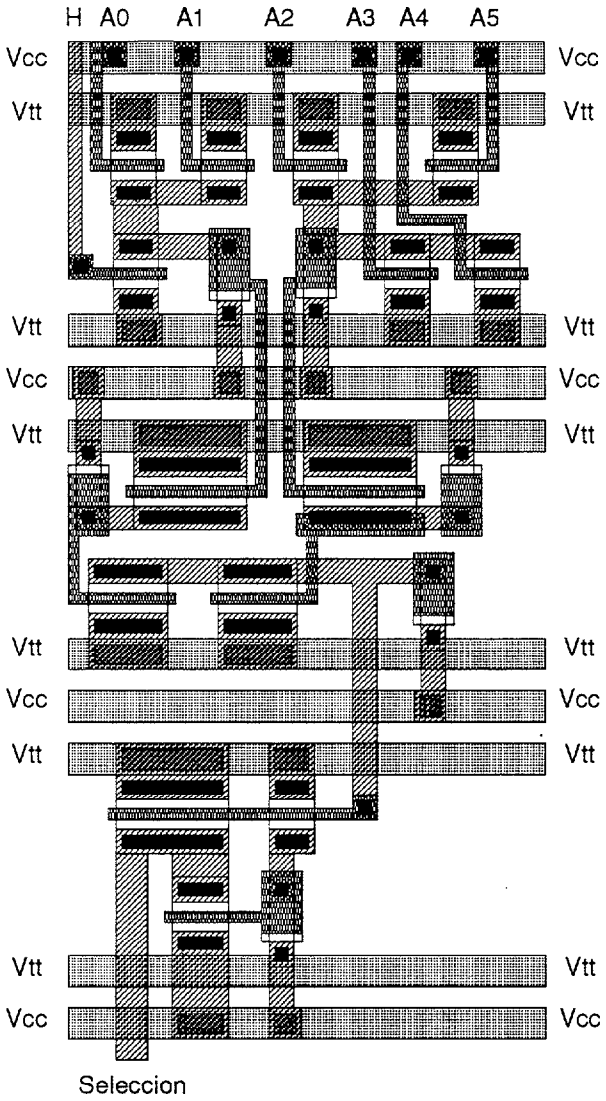


Figura 4.50: Layout de la célula básica del decodificador de escritura

Es un decodificador de 5 entradas (4 de selección de dirección y 1 de control de lectura) y 12 salidas.

Ha sido implementado con una NOR de 3 entradas y una de 2 entradas, que van a una AND de 2 entradas, implementada como una NOR de 2 entradas con 2 inversores (figura 4.51) que, al igual que en el caso anterior, esta dimensionada de forma que sea capaz de atacar a una estructura SBFL, encargada de atacar, a su vez, a 8 estructuras adaptadoras y una alta capacidad.

En la implementación de este decodificador hay que tener un especial cuidado al dimensionar las líneas de salida de las estructuras SBFL del mismo, ya que, debido a la resistencia que presentan por su longitud (que puede llegar a 2.6 mm), y a la corriente que circula por ellas, se va a producir una caída de tensión que puede ser importante.

Para minimizar la resistencia, en principio, se ha optado por usar *METAL 2* para

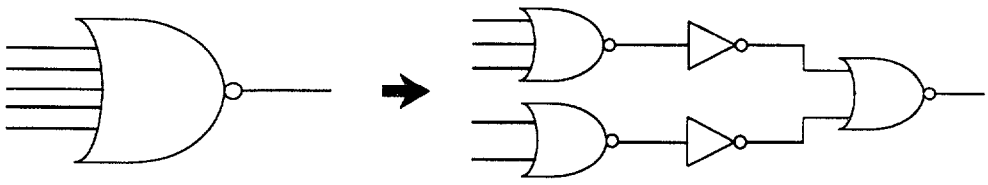


Figura 4.51: Conversión de la NOR del decodificador

esas líneas, ya que presenta una resistencia menor que 0.035 $\Omega/\square$. Al tener un ancho de 2.8 μm, el número de cuadros es aproximadamente 928. Por tanto, la resistencia que ofrece la línea es de 32.5 Ω.

La corriente de salida del decodificador de lectura se puede ver en la siguiente tabla:

Dispersión	*Temperatura (°C)*	*Corriente (mA)*
SS2	0	2.7
	25	2.67
	50	2.6
	75	2.6
	100	2.5
TT	0	3.3
	25	3.2
	50	3.1
	75	3
	100	3
FF1H	0	3.6
	25	3.5
	50	3.4
	75	3.2
	100	3

La figura 4.52 muestra la salida del decodificador de lectura para el proceso de dispersión y la temperatura en los que se produce la mayor caída de tensión (unos 110 mv). Como se puede observar, la salida tiene un nivel lo suficientemente alto como para que dicha caída de tensión no influya en su valor lógico.

En la figura 4.53 podemos ver la célula básica implementada.

4.4.1.5 Multiplexor

El multiplexor es el circuito encargado de la selección de la señal de salida. Es una típica estructura AOI, cuyo layout se puede observar en la figura 4.54.

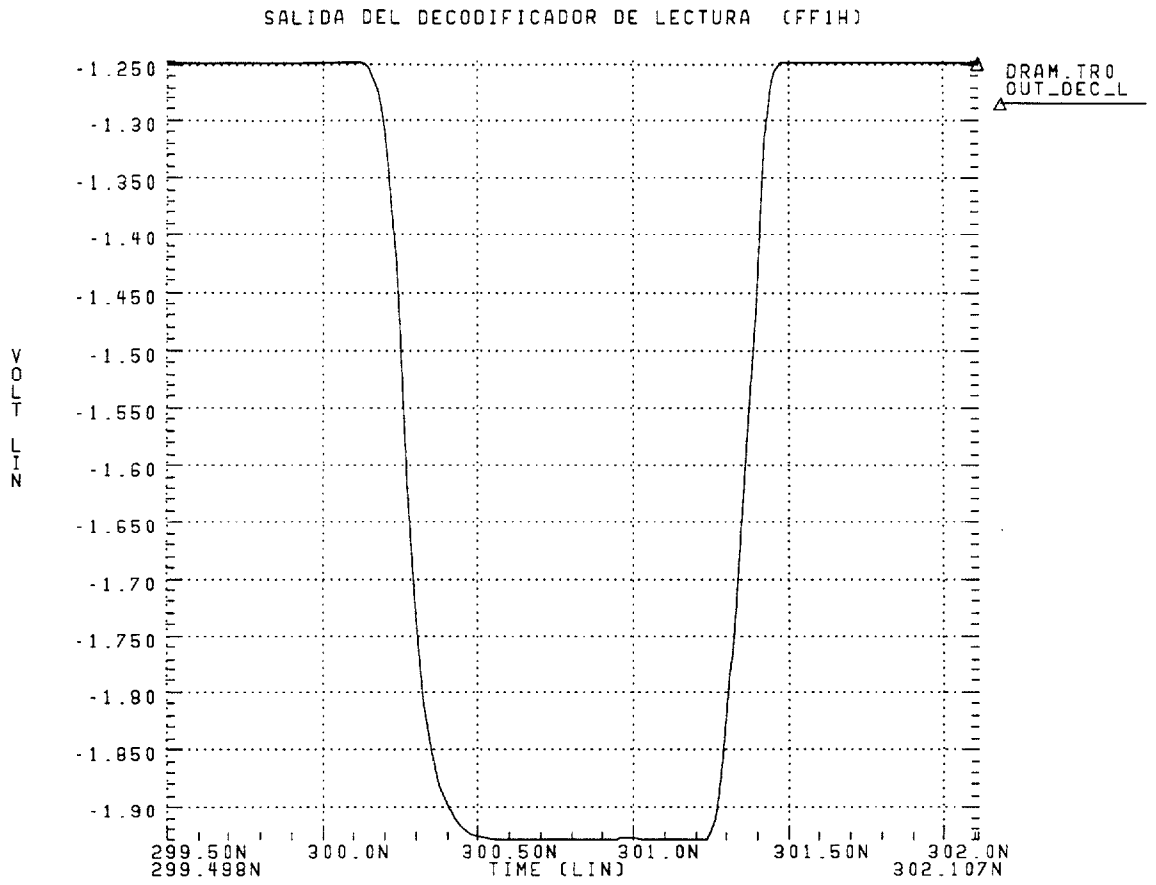


Figura 4.52: Simulación del decodificador de lectura en el caso extremo

4.4.1.6 Implementación

Para la implementación de la DRAM, es necesario, primero, el diseño de registros de 12 bits dispuestos de tal forma que permita leer los bits de forma independiente. La distribución por la que se ha optado se muestra en la figura 4.55(a). El layout de esta célula se puede ver en la figura 4.55(b).

A la salida de cada uno de los registros de 12 bits, hay que poner una estructura que sea capaz de atacar la capacidad debida a la línea que une la salida de dichos registros con el multiplexor. Cada columna tiene una estructura SBFL adaptada a la capacidad que ha de atacar. En la figura 4.56 se puede ver un ejemplo de como queda

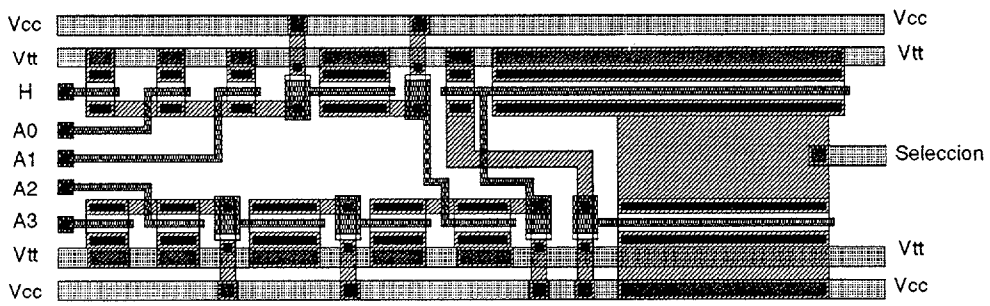


Figura 4.53: Layout de la célula básica del decodificador de lectura

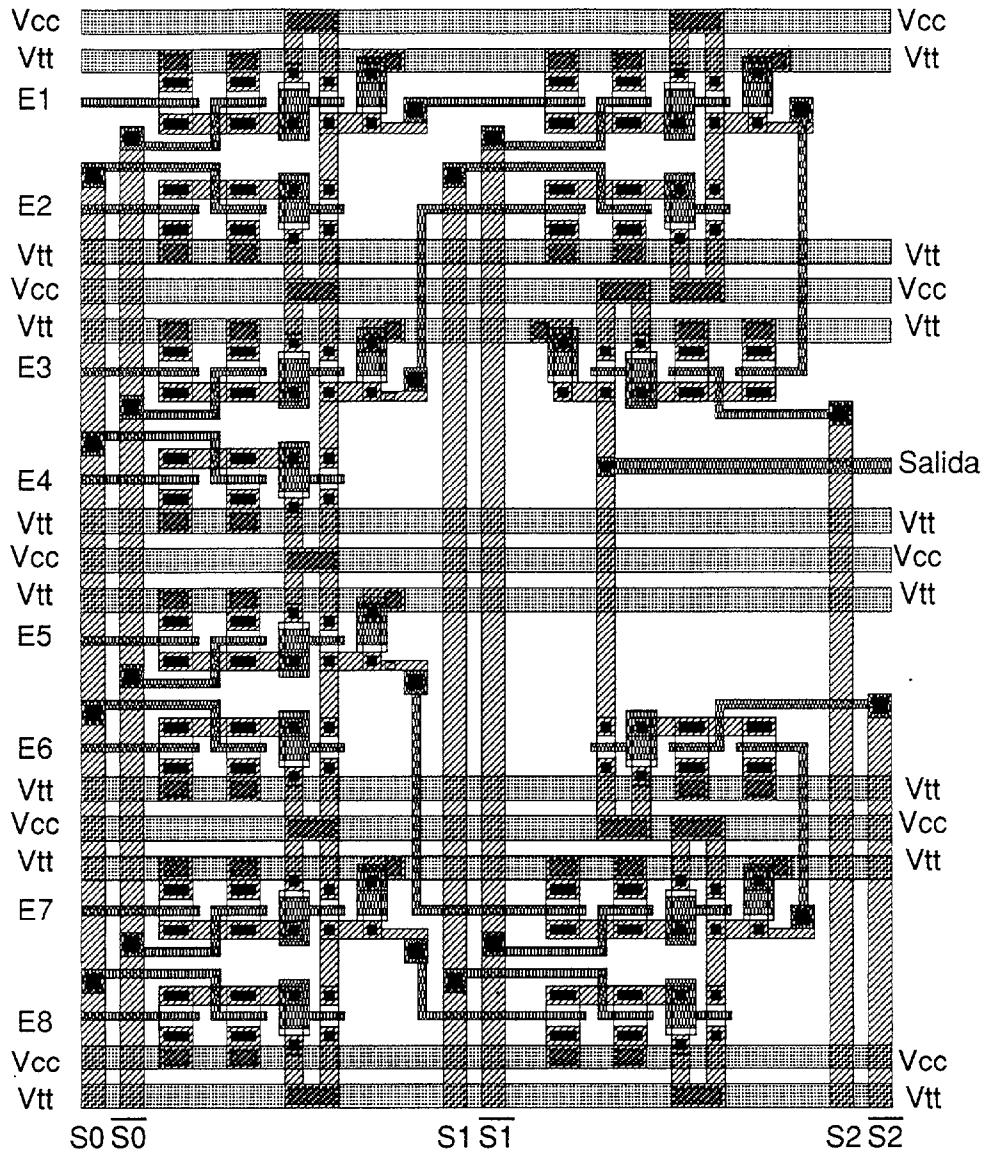
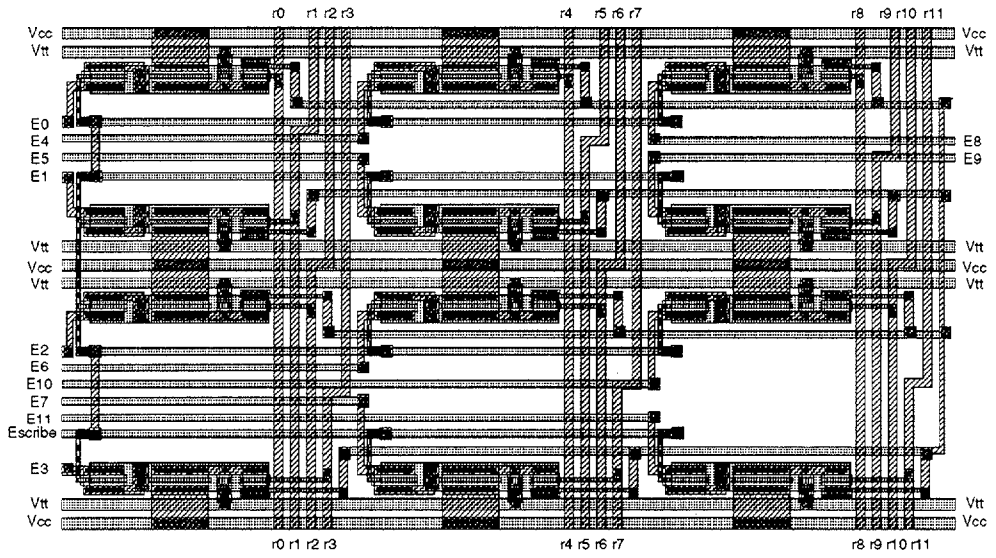


Figura 4.54: Layout del multiplexor de 8 a 1

el registro de 12 bits.

0	4	8
1	5	9
2	6	10
3	7	11

(a)



(b)

Figura 4.55: (a) Distribución del registro de 12 bits (b) Layout

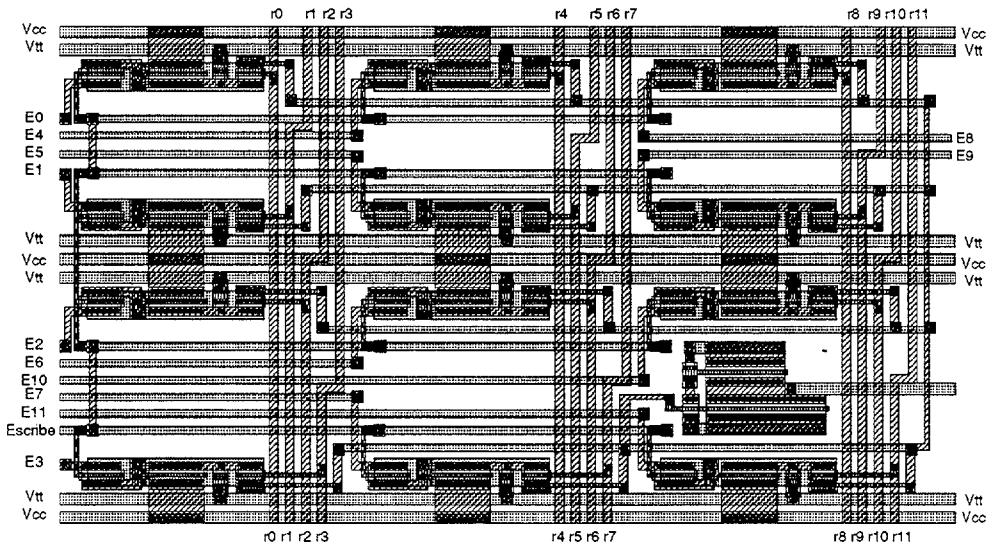


Figura 4.56: Layout del registro de 12 bits con SBFL de salida

4.4.2 Funcionamiento

El funcionamiento de la DRAM se puede dividir en las operaciones de escritura y de lectura:

- Escritura del dato: Una vez se tenga el dato a la entrada de la memoria, se pone la dirección de la célula a escribir en el decodificador de escritura, lo que hace que se active la línea de escritura de la célula, y el dato pasa al nodo de almacenamiento.
- Lectura del dato: Cuando se vaya a leer el dato, se pone la dirección de la célula a leer en el decodificador de lectura, lo que hace que se active la línea de lectura de la célula, obteniendo a la salida de la célula el dato almacenado. Como el transistor de paso deja pasar el dato sólo mientras la línea de lectura está activa, ese es el único momento en el que el dato de lectura es válido. Cuando la señal de lectura pasa a nivel bajo, la salida de la célula de memoria no es más que un nodo flotante, y no es posible asegurar su estado.

En la figura 4.57 se puede observar el funcionamiento de la DRAM.

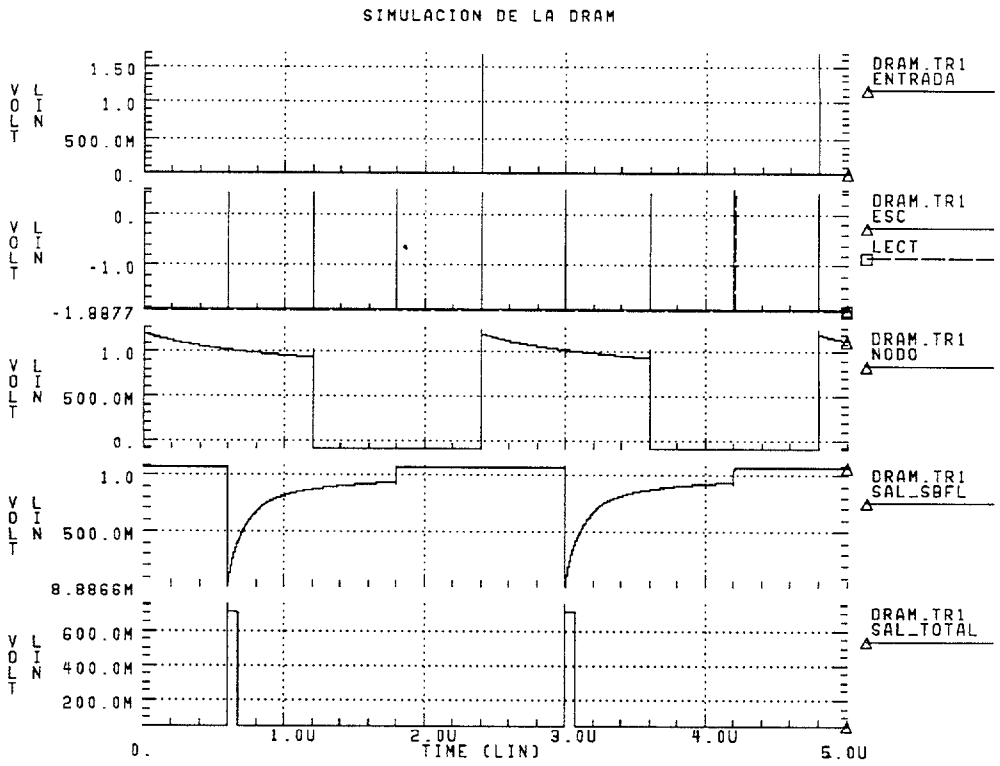


Figura 4.57: Funcionamiento de la RAM dinámica

En las figuras 4.58 y 4.59 se muestra una mirada más precisa, destacando la escritura y lectura de un nivel alto y de un nivel bajo.

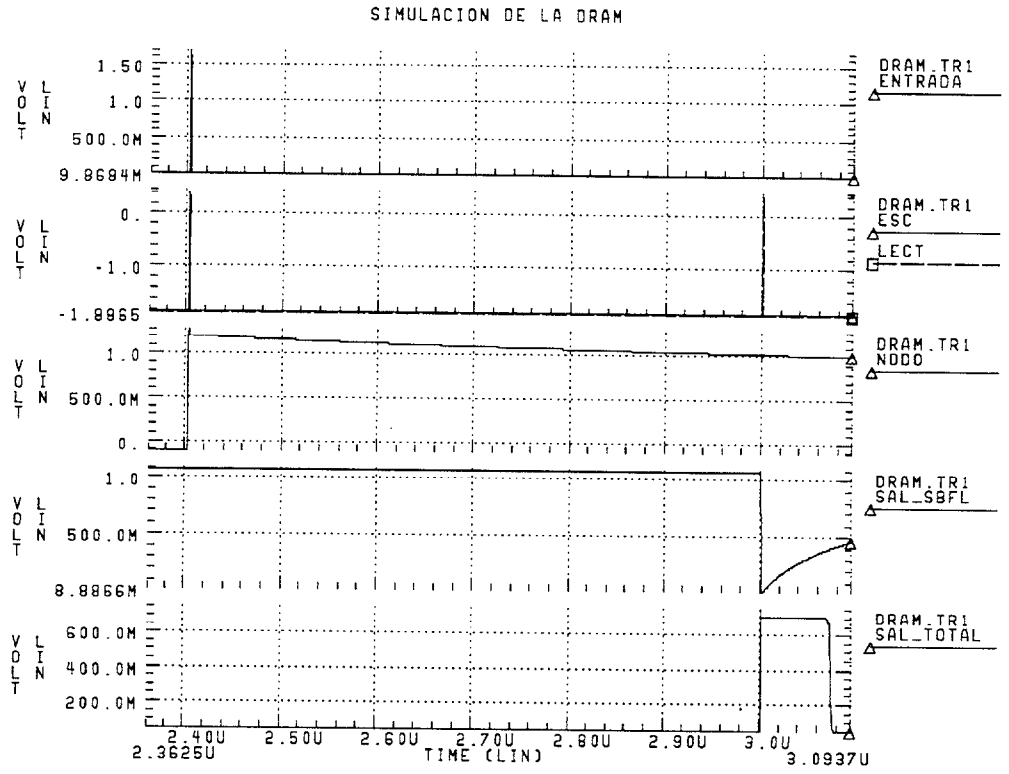


Figura 4.58: Escritura y lectura de un nivel alto

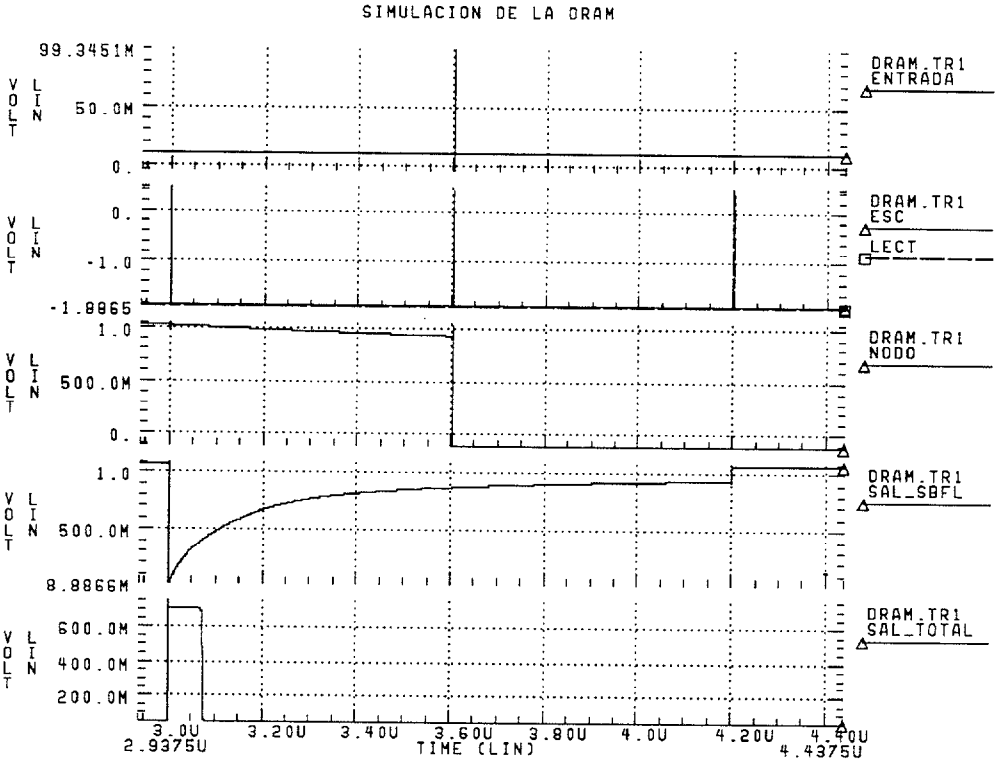


Figura 4.59: Escritura y lectura de un nivel bajo

Un punto muy delicado en el diseño de la DRAM radica en el *nodo de almacenamiento*. En la figura 4.60 se muestra el comportamiento de dicho nodo, observándose la suave pendiente de caída de la tensión.

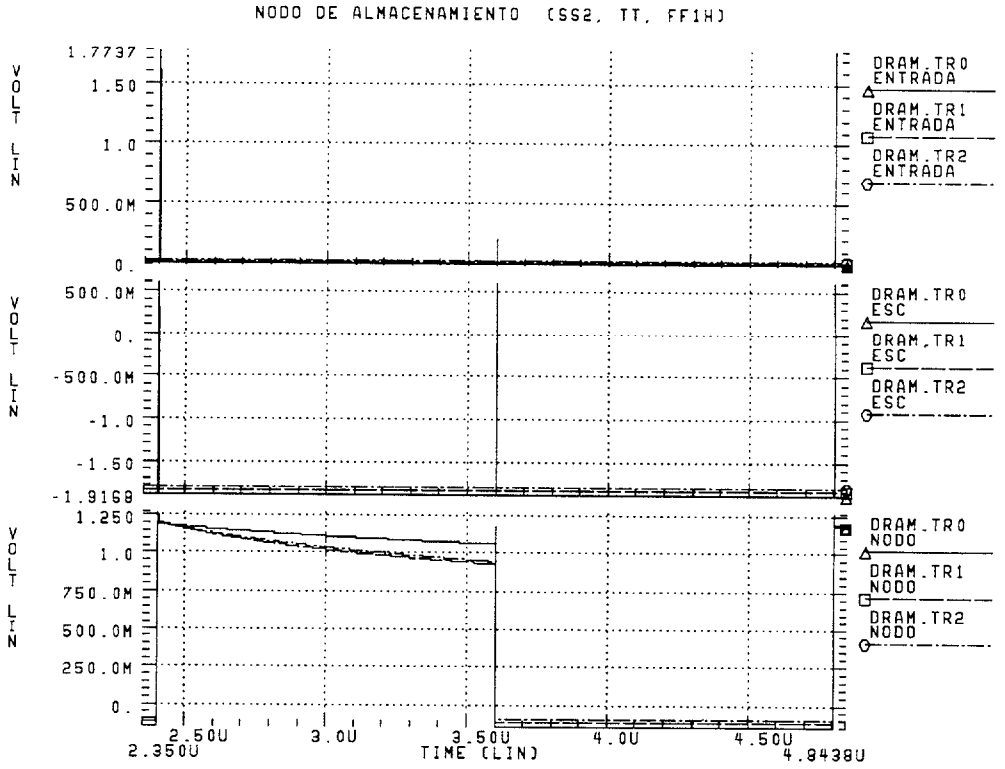


Figura 4.60: Comportamiento del nodo de almacenamiento para varios procesos de dispersión y varias temperaturas

Las figuras 4.61 y 4.62 representan una mirada más detallada del nodo de almacenamiento, prestando especial atención al almacenamiento de un nivel alto (figura 4.61) y de un nivel bajo (figura 4.62).

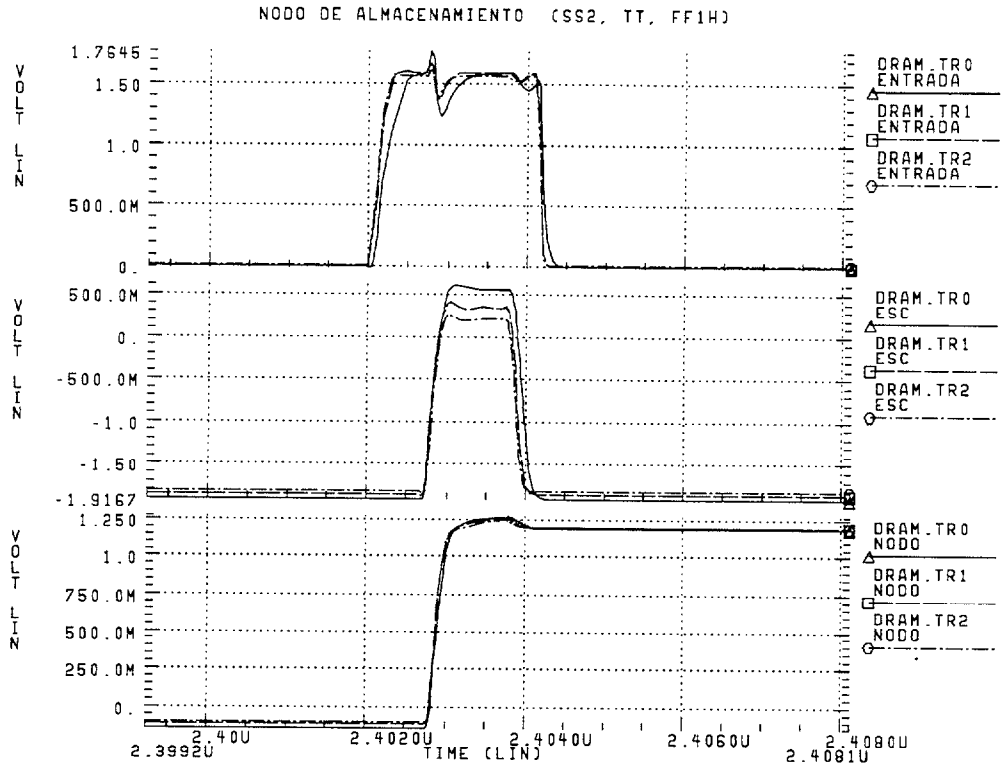


Figura 4.61: Almacenamiento de un 1

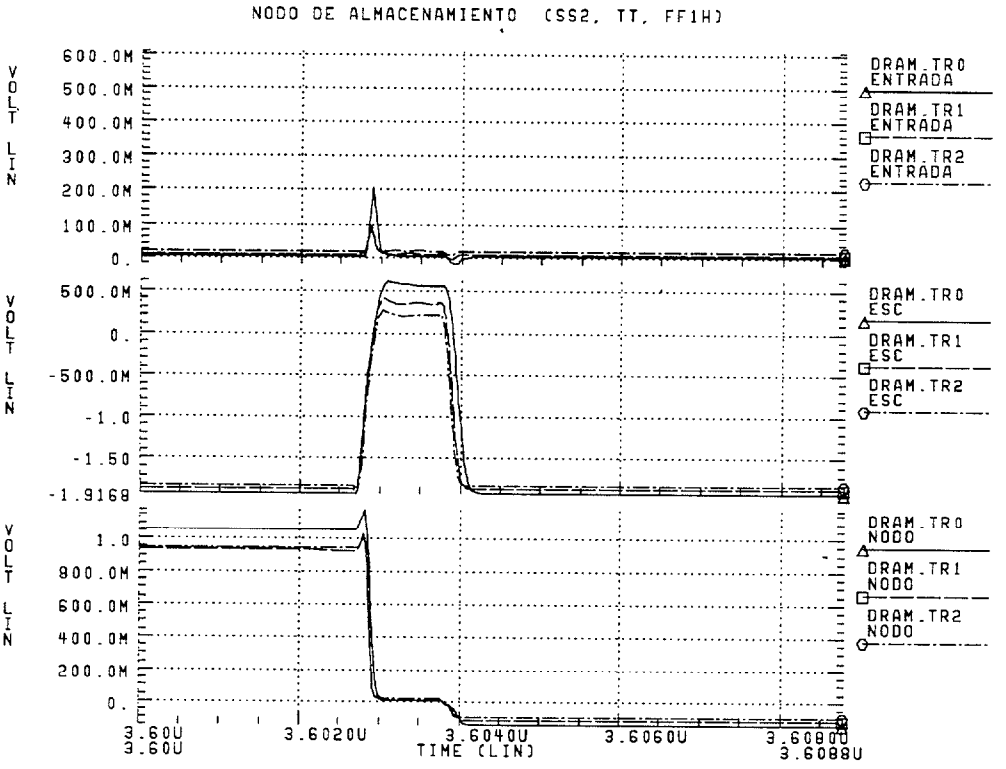


Figura 4.62: Almacenamiento de un 0

En la figura 4.63 se representa la lectura del valor almacenado. En este caso se simula la lectura del dato 600ns después de que haya sido almacenado, tiempo más que suficiente para justificar el correcto funcionamiento de esta memoria en la arquitectura crítica en lo que a tiempo de almacenamiento del dato se refiere (implementación con estructuras MAC sin particionar).

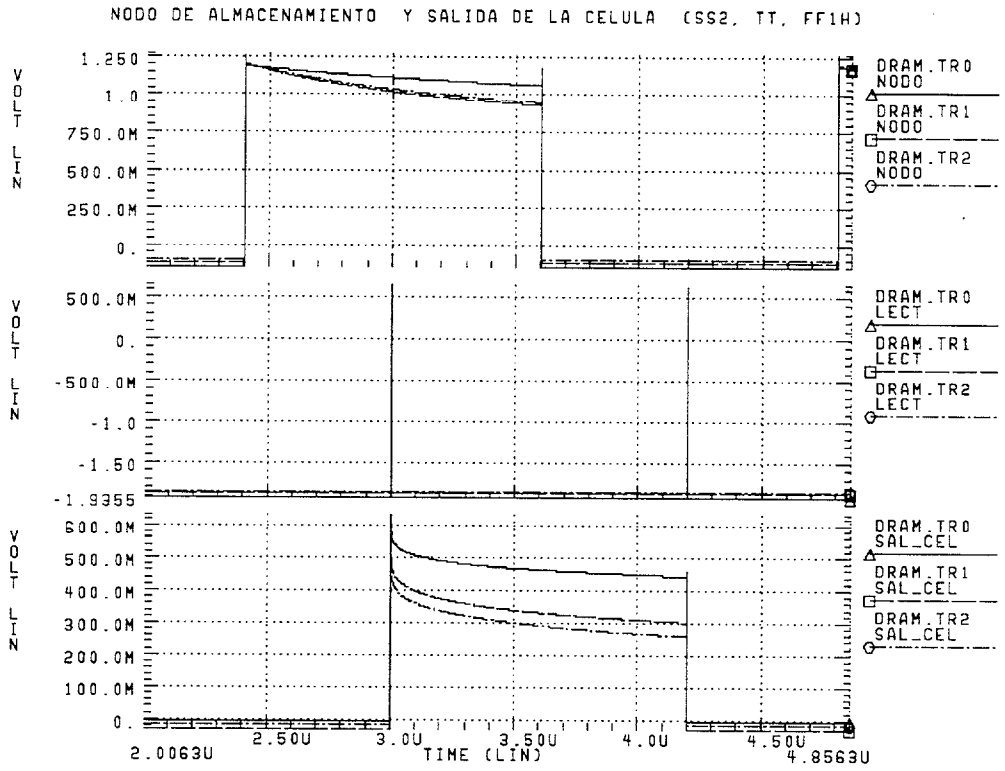


Figura 4.63: Lectura de un nivel alto

En la figura 4.64 y en la figura 4.65 se representa una mirada más precisa al proceso de lectura. Como se puede observar, en los tres procesos de dispersión, y a temperaturas extremas, el valor lógico del dato es más que aceptable.

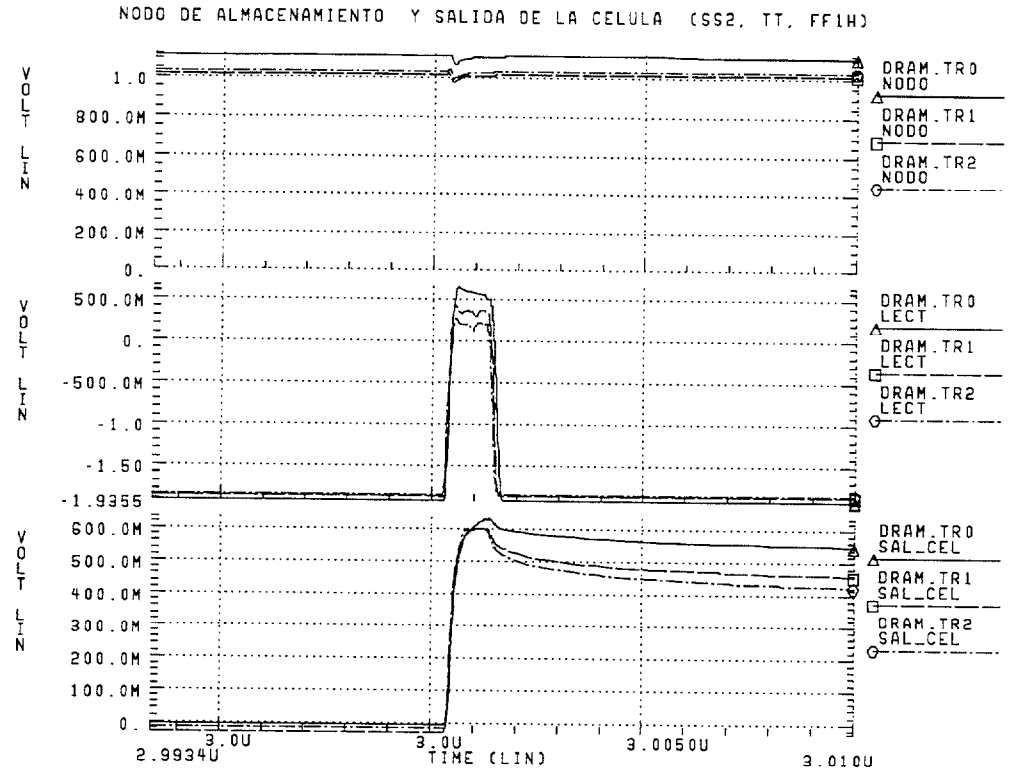


Figura 4.64: Lectura de un nivel bajo

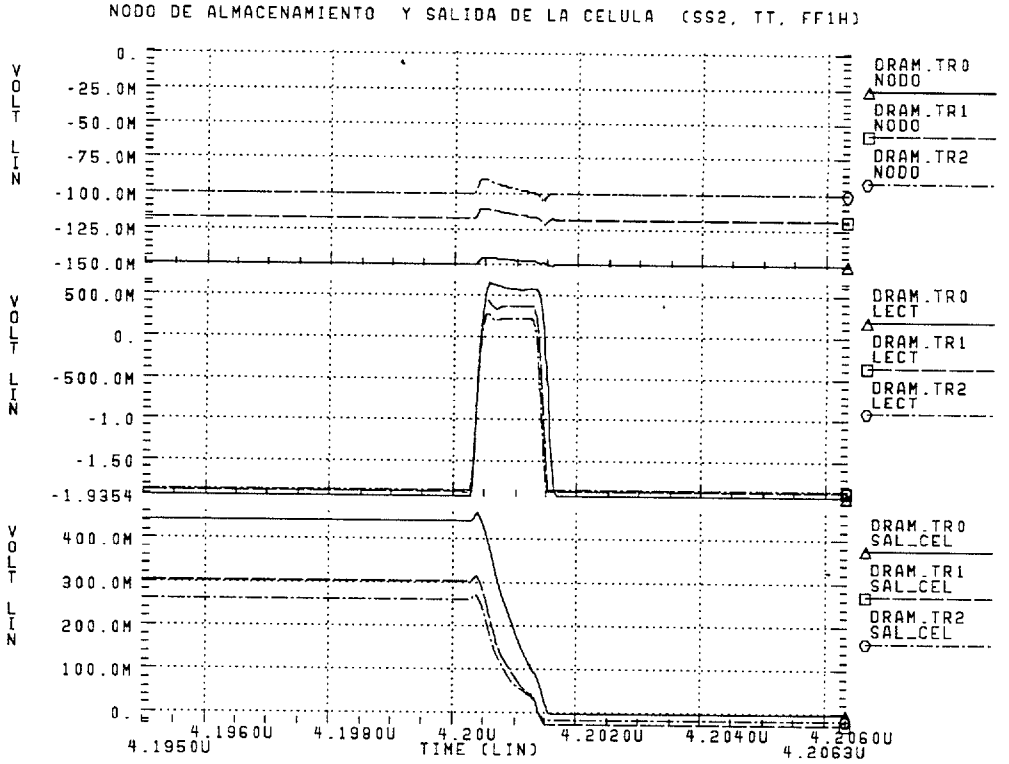


Figura 4.65: Nodo de almacenamiento y salida de la célula de memoria

4.4.3 Prestaciones

En esta sección se realizará un estudio a nivel de tiempos, potencia disipada y área de la implementación realizada.

El estudio de tiempos se dividirá en dos bloques. La escritura y la lectura, indicando los tiempos necesarios para cada operación.

- Escritura.
  El tiempo que pasa desde que la dirección a la entrada del decodificador de escritura es válida, hasta que se obtiene la señal de habilitación de escritura de esa célula, viene reflejado en la siguiente tabla, donde también se pueden observar los tiempos de subida y bajada de esa señal de habilitación.

Dispersión	*Tiempo subida (ns)*	*Tiempo bajada (ns)*	*Retardo (ns)*
SS2 ($0°C$)	0.25	0.22	0.21
TT ($75°C$)	0.23	0.18	0.17
FF1H ($100°C$)	0.22	0.16	0.15

  El tiempo que pasa desde que la dirección a la entrada del decodificador de escritura es válida, hasta que el dato de entrada está almacenado en la célula correspondiente de la DRAM viene reflejado en la siguiente tabla.

Dispersión	*Retardo (ns)*
SS2 ($0°C$)	0.33
TT ($75°C$)	0.3
FF1H ($100°C$)	0.3

- Lectura.
  El tiempo que pasa desde que la dirección a la entrada del decodificador de lectura es válida, hasta que se obtiene la señal de habilitación de lectura de esa célula, viene reflejado en la siguiente tabla, donde también se pueden observar los tiempos de subida y bajada de esa señal de habilitación.

Dispersión	*Tiempo subida (ns)*	*Tiempo bajada (ns)*	*Retardo (ns)*
SS2 ($0°C$)	0.3	0.17	0.18
TT ($75°C$)	0.28	0.15	0.15
FF1H ($100°C$)	0.28	0.15	0.14

  El tiempo que pasa desde que la dirección a la entrada del decodificador de lectura es válida, hasta que el dato, previamente almacenado, está disponible a la salida de la DRAM, viene reflejado en la siguiente tabla.

Dispersión	*Retardo (ns)*
SS2 ($0°C$)	1.0
TT ($75°C$)	0.77
FF1H ($100°C$)	0.73

En la siguiente tabla se realiza un balance de la potencia disipada por el circuito y el área ocupada.

Dispersión	*Potencia (W)*	*Area (mm^2)*	*N° Transistores*
SS2 ($0°C$)	.949	4.75	6112
TT ($75°C$)	1.409		
FF1H ($100°C$)	1.659		

De la potencia disipada, sólo el 25% es debido a la matriz, mientras que el 51% se debe a las estructuras adaptadoras. El resto se debe a los decodificadores de lectura y escritura y a los multiplexores

4.5 Conclusiones

En la implementación de la estructura MAC (y por tanto de la DCT) se ha optado por el uso de un sumador de acarreo anticipado, en el que el incremento en la relación *área–potencia* disipada es de un 62%, mientras que la disminución en el retardo es de un 65%. En el sumador implementado, y debido a la propia estructura del multiplicador–acumulador, no es necesaria la entrada del acarreo inicial, por lo que la *célula A* queda simplificada, asimismo, tampoco es necesario el acarreo de salida, por lo que no es necesaria la inclusión de la *célula D* en el circuito. Para la implementación de la etapa de pre–procesamiento se ha optado por sumadores de acarreo serie, debido a que son los que menos área requieren y menos potencia disipan, y porque para las longitudes de las palabras de entrada, el retardo es muy inferior a 1 *ns*, no influyendo así en la frecuencia del circuito.

Por otro lado, se ha resuelto el problema de las altas corrientes de fuga, que imposibilitaban la implementación de la ROM necesaria en la estructura MAC, obteniendo una memoria de muy altas prestaciones, reducida área y potencia.

Por último, se ha diseñado una memoria dinámica destinada a la implementación más exigente, en lo que a tiempo de almacenamiento se refiere, partiendo de la simple interconexión de células dinámicas básicas, que permite resolver el elevado tiempo de acceso de implementaciones previas, que marcaban el cuello de botella de la implementación de la 2–D DCT en Arseniuro de Galio. Partiendo de la célula básica de memoria, es relativamente fácil la realización de memorias para cualquier arquitectura a implementar.

Capítulo 5

Implementación de los circuitos

En el Capítulo anterior se han definido las unidades básicas que permiten la realización de la Transformada Discreta del Coseno, siguiendo los esquemas presentados en el Capítulo 3. En este Capítulo es necesario estudiar como se integran cada una de estas unidades para formar el circuito integrado final deseado.

En este Capítulo se realizará, además, el estudio de las unidades de control para las dos arquitecturas definidas en el estudio arquitectural: la de *mínimo paralelismo* y la de *paralelismo para flujo óptimo*. Asimismo se presentarán las prestaciones finales de ambas realizaciones, en términos de frecuencia de trabajo, área del circuito y potencia disipada por el mismo, lo que permitirá determinar cuál de ellas es la más adecuada dependiendo de la aplicación final.

5.1 Estructura básica para la unidad de control

La implementación de la *unidad de control* se llevará a cabo mediante la utilización de *contadores binarios*, que consisten, básicamente, en *flip-flops* tipo D, dispuestos de forma que la secuencia de sus estados equivalga a una cuenta binaria. En la figura 5.1 se representa la sucesión de estados para un contador de 2 bits.

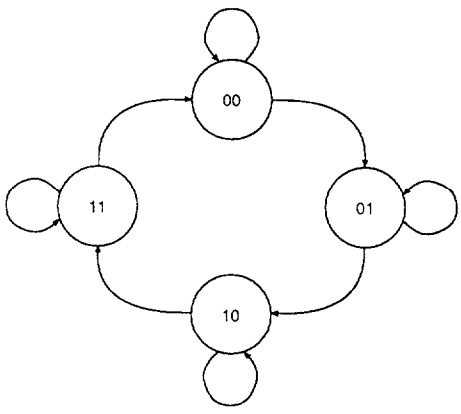


Figura 5.1: Diagrama de estados de un contador binario de 2 bits

La deducción de las ecuaciones que rigen los estados del contador se basará en el contador binario de 4 bits. En la siguiente tabla se puede observar como varía la salida de dicho contador dependiendo de su estado anterior, así como de la señal de habilitación de cuenta (*Enable*).

	Estado i				*Estado i + 1*			
E	$A3_i$	$A2_i$	$A1_i$	$A0_i$	$A3_{i+1}$	$A2_{i+1}$	$A1_{i+1}$	$A0_{i+1}$
0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	1
0	0	0	1	0	0	0	1	0
0	0	0	1	1	0	0	1	1
0	0	1	0	0	0	1	0	0
0	0	1	0	1	0	1	0	1
0	0	1	1	0	0	1	1	0
0	0	1	1	1	0	1	1	1
0	1	0	0	0	1	0	0	0
0	1	0	0	1	1	0	0	1
0	1	0	1	0	1	0	1	0
0	1	0	1	1	1	0	1	1
0	1	1	0	0	1	1	0	0
0	1	1	0	1	1	1	0	1
0	1	1	1	0	1	1	1	0
0	1	1	1	1	1	1	1	1
1	0	0	0	0	0	0	0	1
1	0	0	0	1	0	0	1	0
1	0	0	1	0	0	0	1	1
1	0	0	1	1	0	1	0	0
1	0	1	0	0	0	1	0	1
1	0	1	0	1	0	1	1	0
1	0	1	1	0	0	1	1	1
1	0	1	1	1	1	0	0	0
1	1	0	0	0	1	0	0	1
1	1	0	0	1	1	0	1	0
1	1	0	1	0	1	0	1	1
1	1	0	1	1	1	1	0	0
1	1	1	0	0	1	1	0	1
1	1	1	0	1	1	1	1	0
1	1	1	1	0	1	1	1	1
1	1	1	1	1	0	0	0	0

Simplificando las cuatro funciones que determinan el valor de la salida de los *flip-flops*, se obtienen las siguientes expresiones lógicas:

$$\begin{aligned} A0_{i+1} &= A0_i \oplus E \\ A1_{i+1} &= A1_i \oplus (A0_i \cdot E) \\ A2_{i+1} &= A2_i \oplus (A0_i \cdot A1_i \cdot E) \\ A3_{i+1} &= A3_i \oplus (A0_i \cdot A1_i \cdot A2_i \cdot E) \end{aligned}$$

que definen las entradas de los *flip-flops* en el estado i, de forma que en el estado $i+1$ se obtengan las salidas deseadas.

De las ecuaciones anteriores se deduce que, la expresión lógica de la entrada de cualquiera de los n *flip-flops* tipo D que forman el *contador binario síncrono de n bits*, en el estado i (An_{i+1}), es:

$$An_{i+1} = An_i \oplus (A0_i \cdot A1_i \cdot A2_i \cdots A(n-1)_i \cdot E)$$

El diseño de la célula básica del contador, realizada en base a las ecuaciones previas, se puede ver en la figura 5.2. La entrada E_i equivale a la señal de habilitación del contador para el término A_i, mientras que E_{i+1} equivale a la habilitación para el término A_{i+1}. En la figura 5.3 se puede observar la implementación a nivel de layout de dicha célula. El esquemático de un contador de 4 bits, así como su implementación a nivel de layout usando dicha célula básica, se representa en las figuras 5.4 y 5.5.

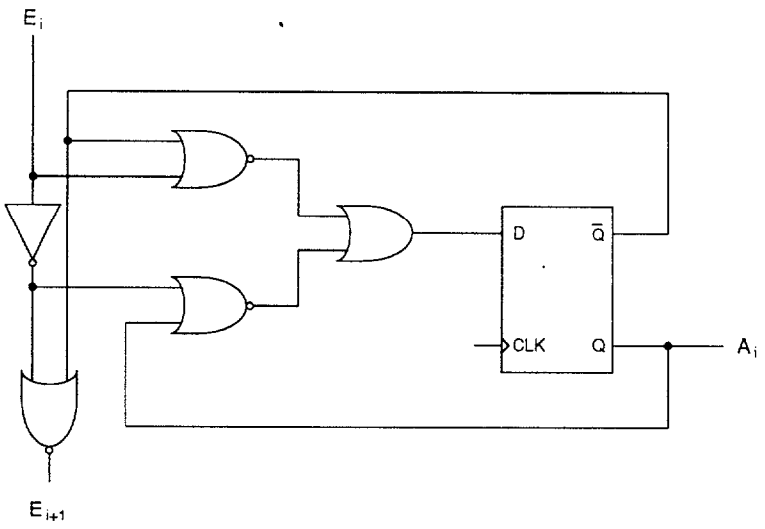


Figura 5.2: Célula básica del contador binario

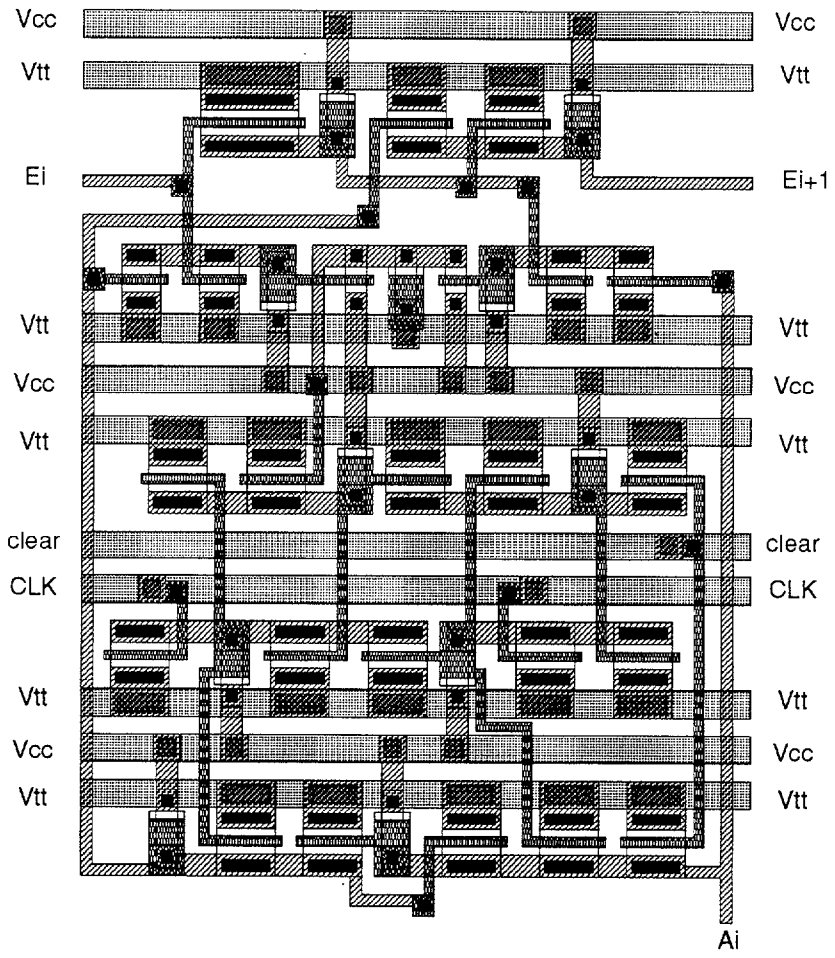


Figura 5.3: Layout de la célula básica del contador binario

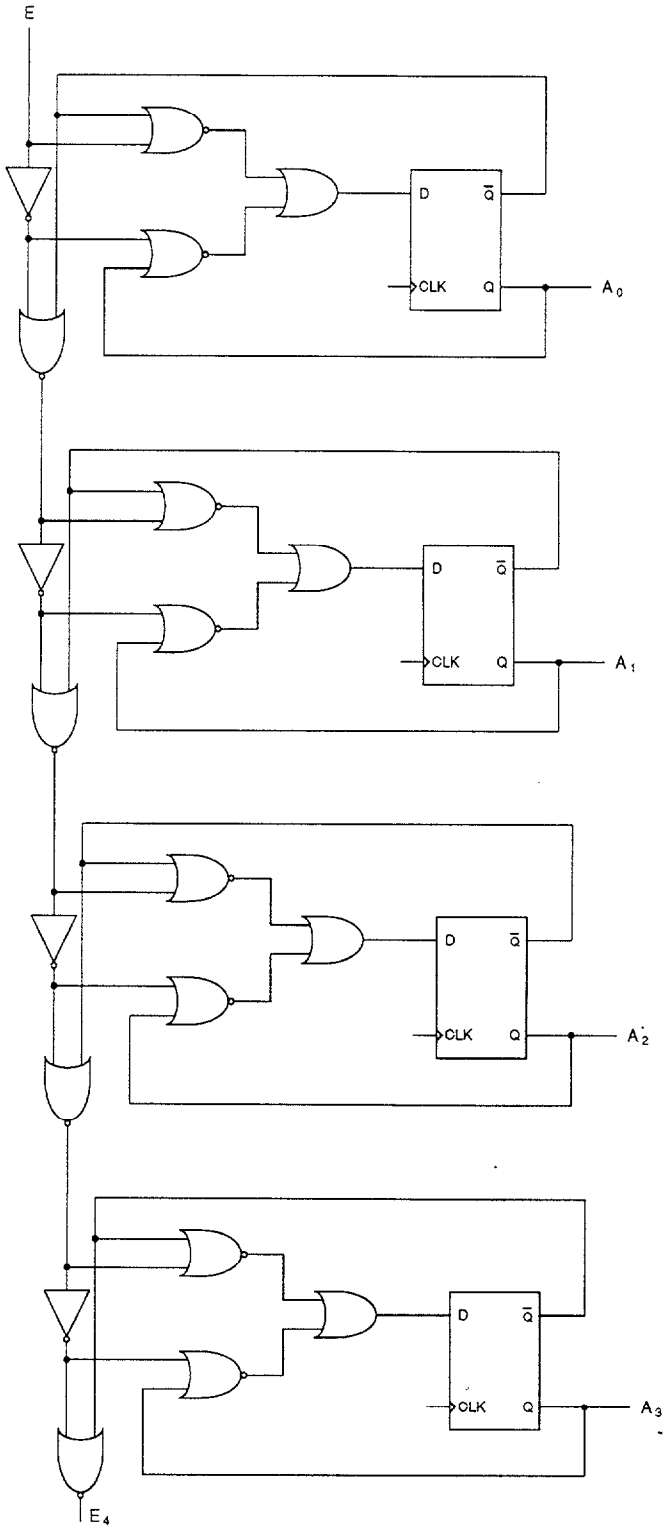


Figura 5.4: Esquema de bloques del contador de 4 bits

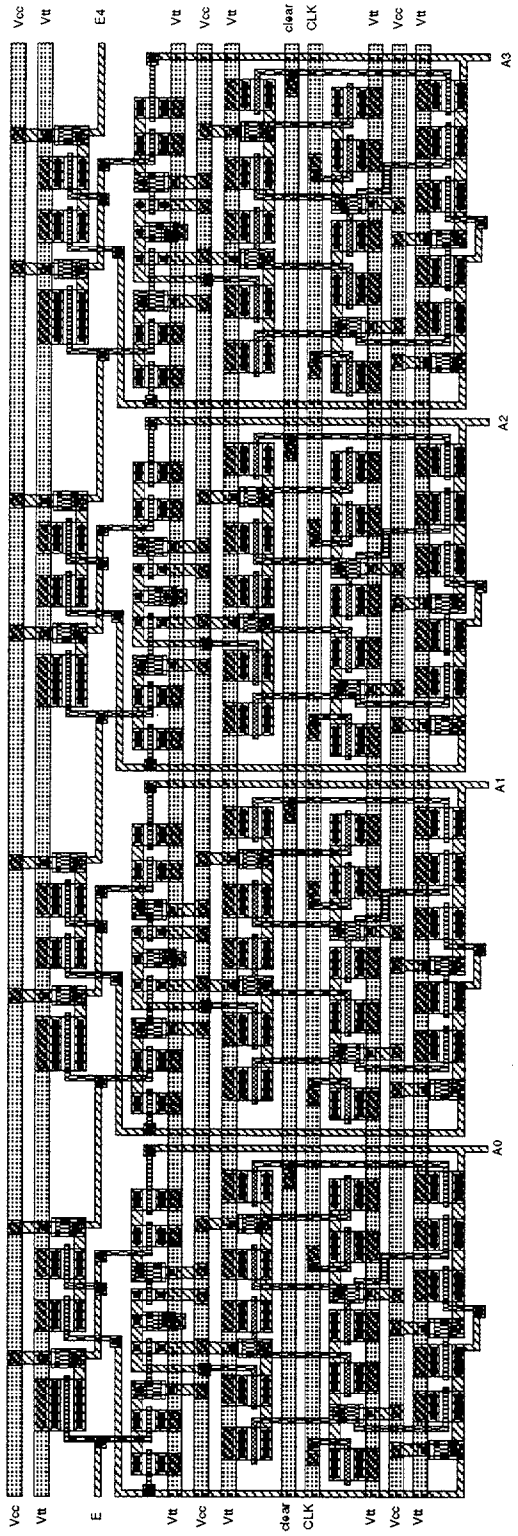


Figura 5.5: Layout del contador de 4 bits

5.2 GaAsDCT11: solución de mínimo paralelismo

La primera solución propuesta, denominada GaAsDCT11 debido a que en este caso las estructuras MAC usan una partición 1, se observa en la figura 5.6. Tal y como se especificó en el Capítulo 3, la DCT se realiza mediante dos unidades FCT–MMM conectadas entre sí a través de una memoria RAM donde, además, se realiza la transposición. Asimismo, todas las unidades básicas, su descripción, optimización e implementación, están recogidas en el Capítulo 4. En este apartado se expondrá la forma en que estas unidades básicas son conectadas para conseguir la implementación final. Al mismo tiempo y como parte verdaderamente relevante, se desarrolla la unidad de control del circuito.

5.2.1 Conversión paralelo/serie

Para la serialización de los datos de entrada, se dispone de dos bancos de registros. El primer banco está formado por 8 registros de 9 bits cada uno, que permite el almacenamiento de las 8 palabras que forman una columna de entrada a razón de una palabra (9 bits) por ciclo. El contenido de este banco de registros es transferido a un segundo banco, del mismo tamaño y organización que el anterior, después de que hayan pasado los 5 ciclos de sincronización (ver apartado 3.2.5.1). A partir de aquí se realiza la serialización de las palabras que constituyen la columna de entrada. Para ello se dispone de los multiplexores representados en la figura 5.7. La lectura de los datos en serie comienza con la introducción de tres bits nulos (cero lógico) y finaliza con la lectura del bit de mayor peso (b0(k)) dos veces consecutivas, tal y como es requerido para la extensión de signo.

Las señales de control a generar para esta etapa se pueden ver en la figura 5.8, donde se puede observar que en cada ciclo se habilita la escritura de un grupo de 9 registros del primer banco de registros. Reiterando el proceso en ciclos sucesivos, se consigue almacenar todas las palabras de la columna correspondiente. Cuando se vuelve a habilitar el almacenamiento del primer banco de registros, se habilita también el almacenamiento del segundo banco de registros. Una vez almacenados los datos en ese segundo banco de registros, comienza la lectura serie de los mismos con la introducción previa de tres bits de valor lógico 0.

Para controlar el conversor paralelo/serie, es necesario el uso de dos contadores, ambos de 4 bits, de forma que el primero sea el que delimite los 13 *ciclos que constituyen el periodo* de las señales, y el segundo sea el que genere *las señales de selección* de los multiplexores. También es necesario un banco de 8 registros, que actúen a modo de registro de desplazamiento, encargado de generar las *señales de habilitación* de los diversos grupos de registros que constituyen el conversor paralelo/serie. Estas estructuras están representadas en la figura 5.9.

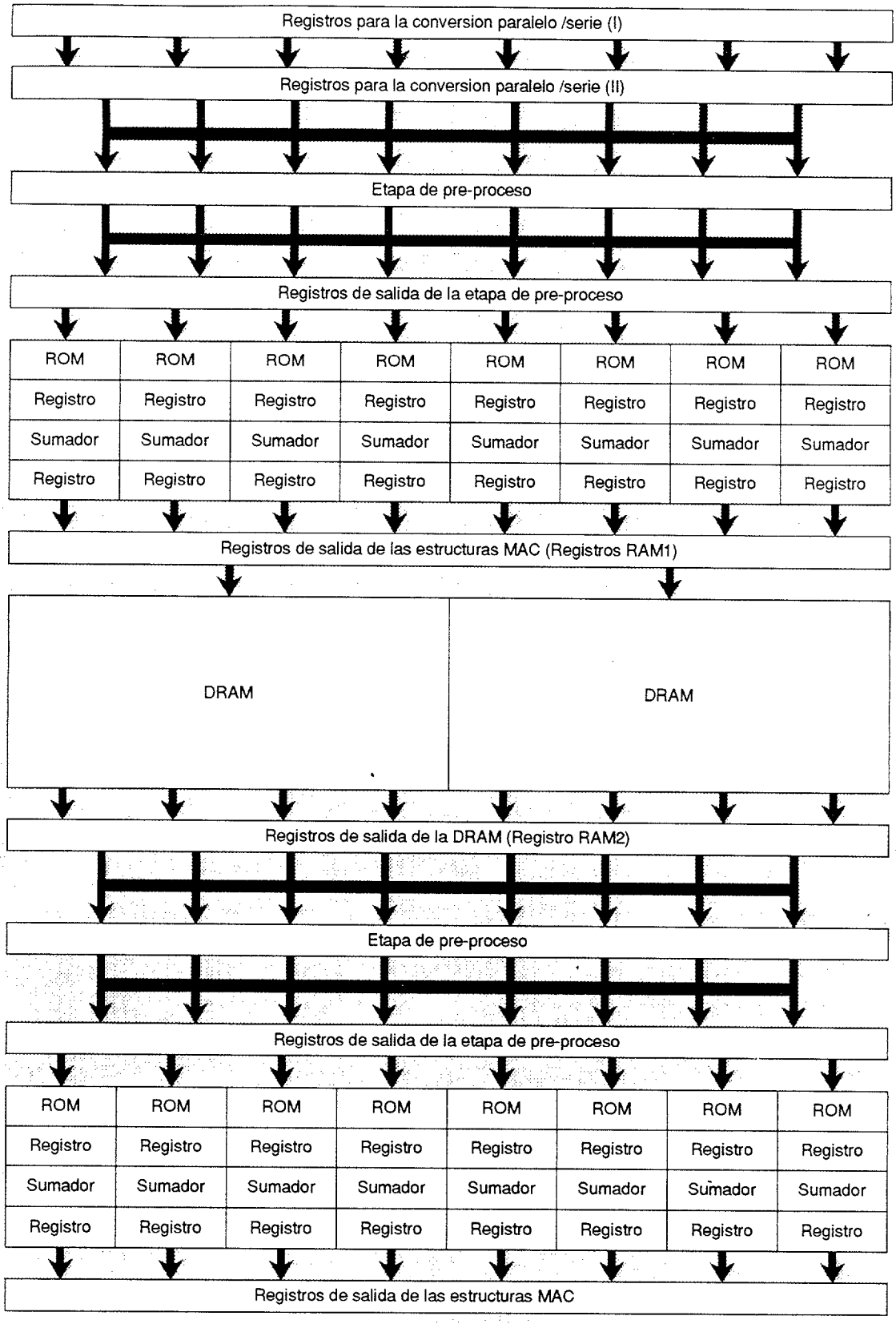


Figura 5.6: Datapath del circuito GaAsDCT11

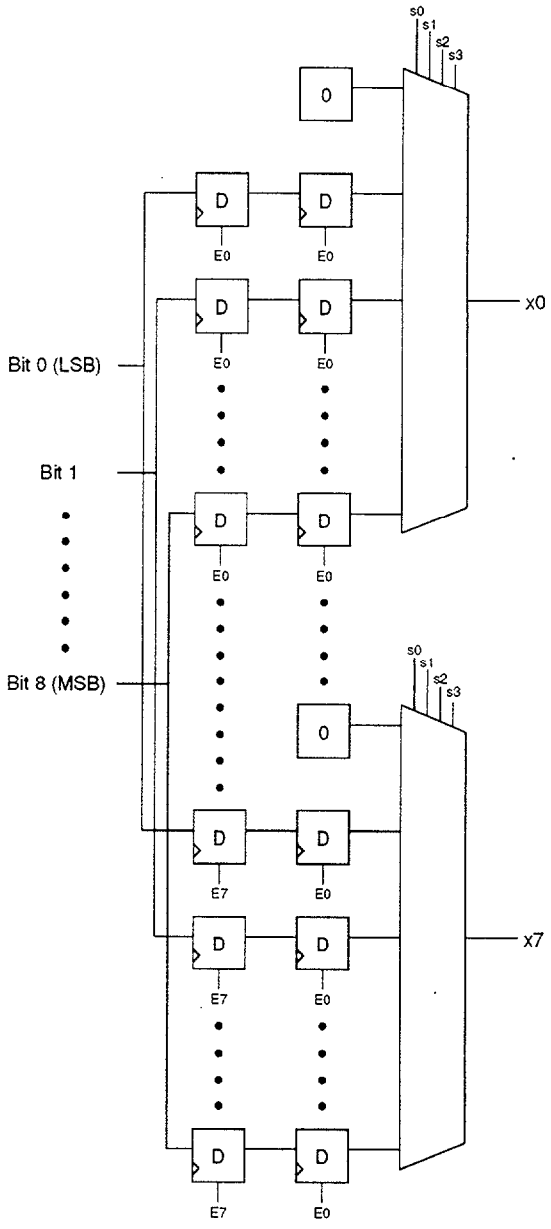


Figura 5.7: Esquema de bloques del conversor paralelo/serie

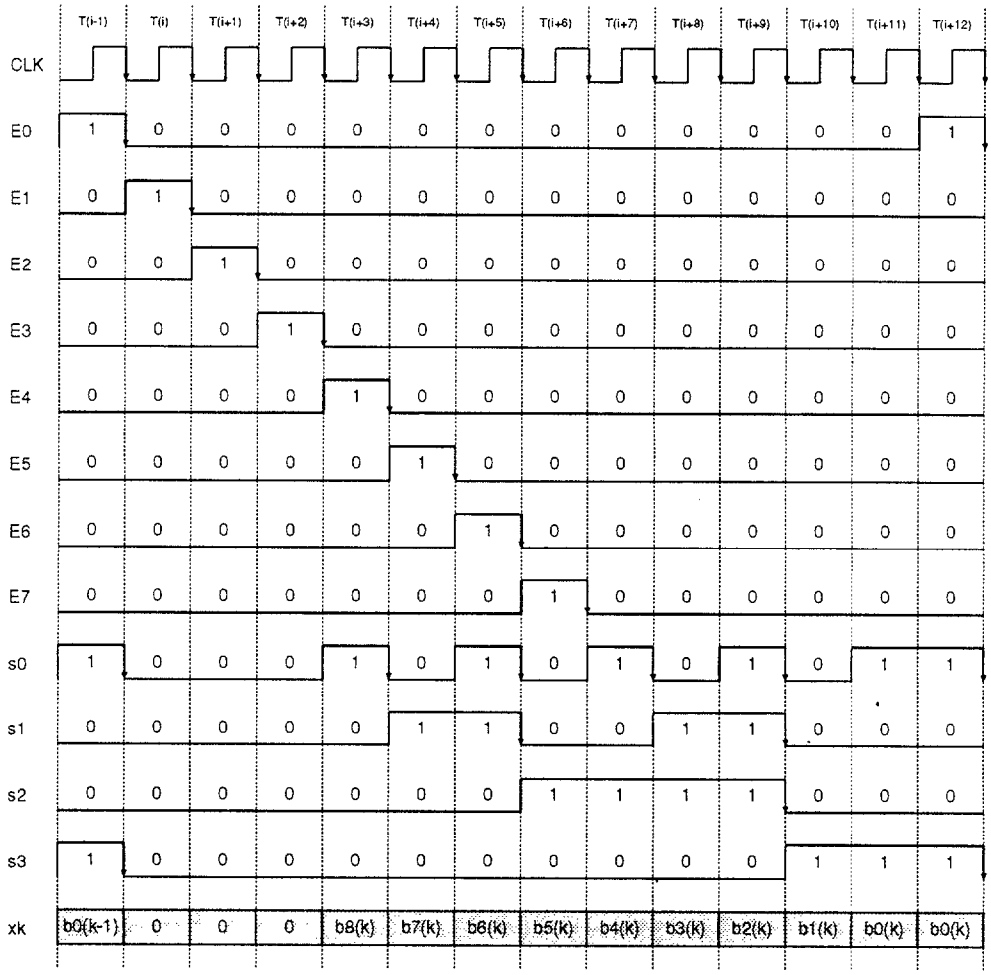


Figura 5.8: Diagrama de tiempos del conversor paralelo/serie

Para la implementación del contador encargado de delimitar los 13 ciclos que constituyen el periodo de las señales, es necesaria la inclusión de una lógica adicional al contador de 4 bits, que permita la cuenta hasta el valor 1100 y en el siguiente ciclo reinicialice asíncronamente el contador. Para realizar dicha operación, se hará uso de la señal de *clear* del contador, que tomará la expresión:

$$clear \;=\; c0 \cdot c2 \cdot c3$$

de modo que, cuando el contador alcance el estado 1101, se active y haga que se reinicie la cuenta.

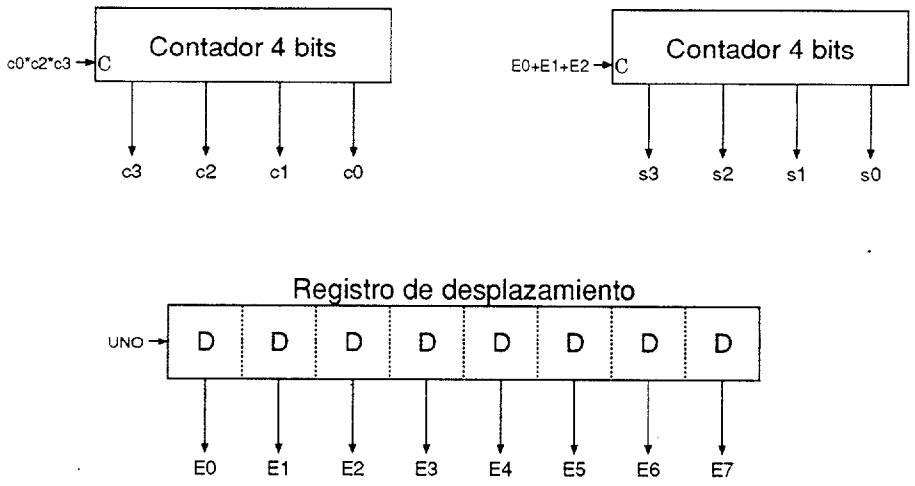


Figura 5.9: Unidad de control del conversor paralelo/serie

El registro de desplazamiento está constituido por 1 *flip–flop* tipo D con señal de *preset* y 7 *flip–flops* tipo D con señal de *clear*. De esta forma, pasado el estado de inicialización, en el que el primer registro está cargado con un nivel lógico 1, el nivel lógico almacenado en dicho registro se va desplazando cada ciclo de reloj a través del registro de desplazamiento. Cuando pasan 13 ciclos, se activa la señal *uno*, de forma que en el siguiente ciclo se vuelve a cargar un nivel lógico 1 en el primer *flip–flop*. La expresión de dicha señal viene dada por:

$$uno \;=\; c2 \cdot c3$$

de forma que, cuando el contador llega al estado 1100 se activa dicha señal, y cuando pasa al estado 0000, se almacena en el primer *flip–flop* del registro de desplazamiento, produciendo el cambio a nivel alto de la señal E0.

El contador encargado de generar las señales de selección del multiplexor tiene una entrada de *clear*, cuya expresión es:

$$clear \;=\; E0 + E1 + E2$$

de forma que comienze la cuenta a partir de la llegada de E3. Eso implicaría que siempre que alguna de las señales E0, E1 o E2, tenga un valor lógico uno, las señales de

selección del multiplexor estarán a nivel bajo. Como cuando E0 tiene un valor lógico uno, las señales $s0$ y $s3$ han de estar también a nivel alto, las señales de selección del multiplexor vendrán finalmente dadas por:

$$\begin{aligned} s0^* &= s0 + E0 \\ s1^* &= s1 \\ s2^* &= s2 \\ s3^* &= s3 + E0 \end{aligned}$$

5.2.2 Estructura FCT–MMM de la primera 1–D DCT

Los bits serializados pasan a la etapa de pre–proceso, donde se realiza la suma y la resta de los mismos. Los bits resultantes van a las estructuras MAC, que funcionan internamente a ciclo de reloj, y producen un resultado cada 13 ciclos, dos después de la entrada del último bit, coincidiendo con la señal E3. Esta señal también se utiliza como control del multiplexor interno a dichas estructuras, de forma que cuando toma el valor lógico 1, permite el paso de las condiciones iniciales.

5.2.3 RAM dinámica con transposición

La DRAM ha sido extensivamente estudiada en el apartado 4.4. Para su utilización en el diseño final se toman dos memorias que se sitúan entre dos registros denominados Registro–RAM1 y Registro–RAM2. En este apartado se estudia la generación de las señales de control de la RAM y las de selección del dato para la segunda 1-D DCT.

Las operaciones de escritura y lectura se realizan de forma simultánea, pero sobre memorias distintas: mientras se realiza la escritura sobre una de ellas, se realiza la lectura sobre la otra, y viceversa. Cuando finaliza la operación sobre una de ellas, se comienza a realizar la operación contraria sobre la siguiente, y así sucesivamente.

5.2.3.1 Escritura de la DRAM

Para escribir los datos en la DRAM, operación que se comienza a realizar coincidiendo con la tercera vez que se activa la señal E4, es necesario escribir los 8 registros, y añadir 5 ciclos de espera, en los que la señal de habilitación de escritura permanece a nivel bajo. Este proceso se realiza 8 veces para cada DRAM de forma alternativa. Las señales de control se pueden ver en las figuras 5.10 y 5.11, donde se puede observar que equivalen a la salida de un contador de 6 bits que tiene la cuenta habilitada durante 8 ciclos, pasando a deshabilitarse durante los 5 ciclos siguientes.

Para generar estas señales se utilizan 3 contadores (figura 5.12), uno de 2 bits que indica el comienzo de la cuenta (comienzo de la operación), uno de 4 bits para delimitar los 13 ciclos de periodo de las señales, y uno de 6 bits para generar las direcciones de escritura.

La señal de habilitación del contador de 2 bits viene dada por:

$$Enable = \overline{b0 \cdot b1} \cdot E3$$

de tal forma que, cuando se active por tercera vez la señal E4, sea el producto lógico de $b0$ y $b1$ el que dicte el comienzo de la operación de escritura. El diagrama de estados de la señal *Enable* del contador de 2 bits se puede ver en la siguiente tabla, donde se puede observar como, tras haberse activado tres veces la señal E3, la señal de comienzo de escritura pasa a nivel alto, y no vuelve a cambiar de estado.

$b1$	$b0$	$E3$	*Enable*	$b0 \cdot b1$
0	0	0	0	0
0	0	1	1	0
0	1	0	0	0
0	1	1	1	0
1	0	0	0	0
1	0	1	1	0
1	1	0	0	1
1	1	1	0	1

El contador de 4 bits, encargado de delimitar los 13 ciclos de periodo de las señales, tiene al producto lógico de $b0$ y $b1$ como señal de habilitación de cuenta, mientras que, el contador de 6 bits, encargado de generar la dirección de la DRAM, tiene al producto lógico de $b0$, $b1$ y el *bit de mayor peso de salida del contador de 4 bits complementado* como habilitación, de modo que, los primeros 8 ciclos permite la cuenta, inhabilitándola los cinco ciclos siguientes.

$$Enable = b0 \cdot b1 \cdot \overline{c3}$$

La señal de habilitación del contador de 6 bits coincide con la habilitación de escritura de la DRAM. En la siguiente tabla, donde se ha asumido un valor 1 para el producto lógico entre $b0$ y $b1$, se puede observar como varía la señal de habilitación, en función del bit de mayor peso del contador de 4 bits.

$c3$	$c2$	$c1$	$c0$	*Enable*
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	1
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0

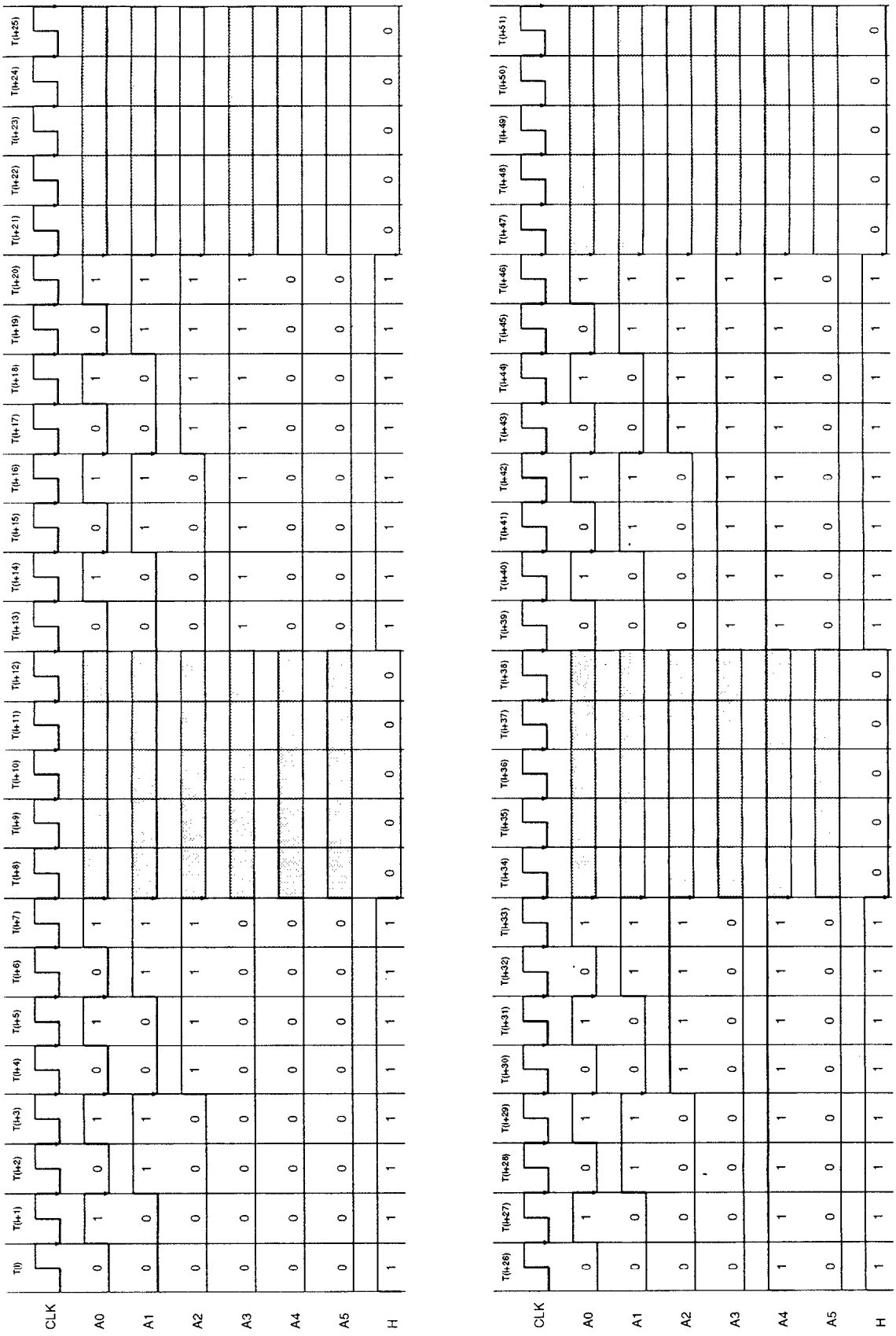


Figura 5.10: Diagrama de tiempos de la RAM (Escritura 1)

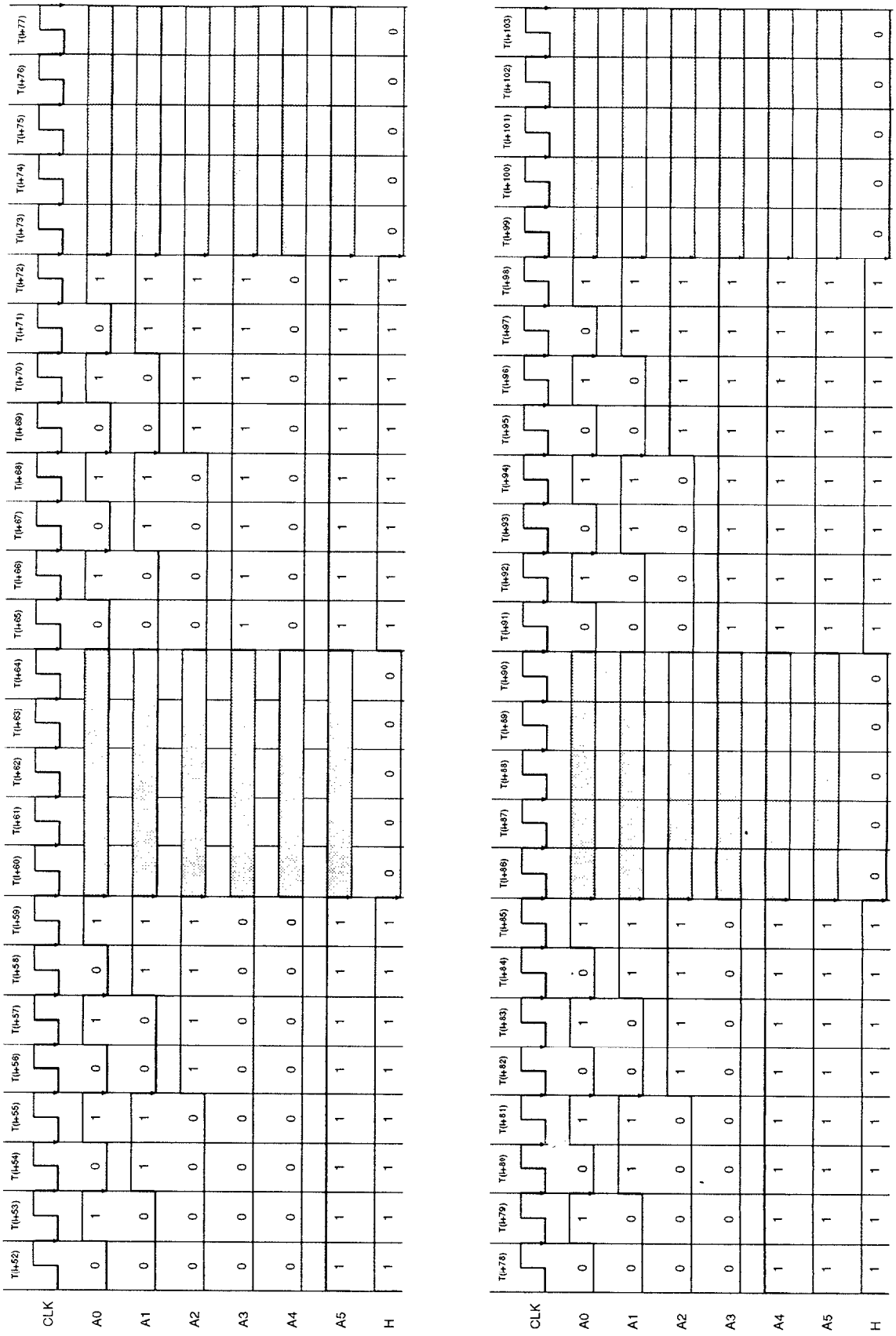


Figura 5.11: Diagrama de tiempos de la RAM (Escritura 2)

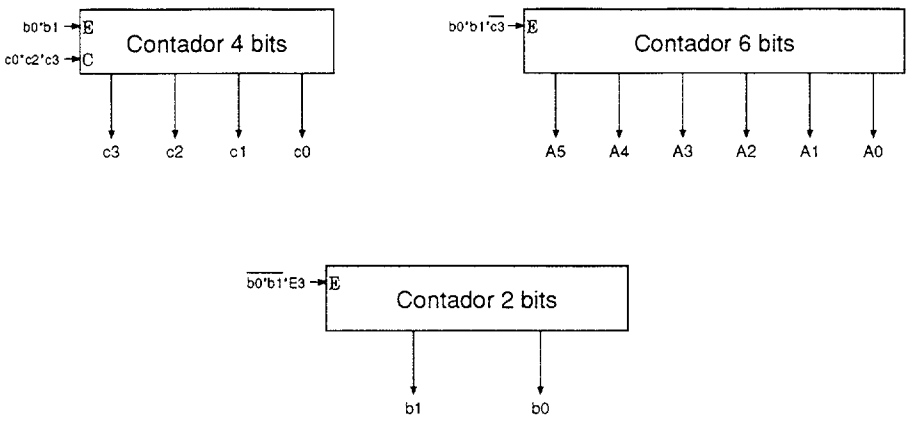


Figura 5.12: Unidad de control para escritura en la DRAM

5.2.3.2 Lectura de la DRAM

Simultáneamente a la escritura se va realizando la operación de lectura en la memoria adyacente. Para leer los datos de la DRAM, se va generando la dirección que indica la posición del bit a leer, seleccionando con el multiplexor la columna que se está leyendo. Es necesario leer el último bit dos veces consecutivas para realizar la extensión de signo. El proceso de lectura de la memoria también se realiza de forma alternativa, de forma que cuando se termina de leer una de ellas se comienza a leer la siguiente y viceversa. Las señales de control se pueden ver en la figuras 5.13 y 5.14, donde se puede observar que equivalen a una cuenta de 12 donde se repite del último estado.

Para generar las señales se utilizan dos contadores y tres multiplexores de dos a uno (figura 5.15). El primero de los contadores (el mismo que se usa en la escritura para generar las señales que delimitan el periodo de 13 ciclos) genera la dirección a leer, mientras que el segundo genera las señales de selección de los multiplexores de salida de la DRAM.

Es necesario el uso de tres multiplexores para repetir la última dirección de lectura, de forma que, realizando la selección de los mismos mediante el producto lógico:

$$sel = c2 \cdot c3$$

cuando se genere el último estado, se niegan los tres bits de menor peso, tal y como se muestra en la siguiente tabla.

	Salida del contador				*Dirección de lectura*			
sel	*c3*	*c2*	*c1*	*c0*	*A3*	*A2*	*A1*	*A0*
0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	1
0	0	1	0	0	0	0	1	0
0	0	1	1	0	0	0	1	1
0	1	0	0	0	0	1	0	0
0	1	0	1	0	0	1	0	1
0	1	1	0	0	0	1	1	0
0	1	1	1	0	0	1	1	1
0	1	0	0	0	1	0	0	0
0	1	0	0	1	1	0	0	1
0	1	0	1	0	1	0	1	0
0	1	0	1	1	1	0	1	1
1	1	1	0	0	1	0	1	1

La habilitación del contador de 3 bits que selecciona la columna en la DRAM viene dada por el producto lógico de *b0*, *b1* y la señal *sel*, de tal forma que cambia de estado cada 13 ciclos, contados desde el comienzo de la operación de lectura.

5.2.3.3 Selección del elemento de almacenamiento

Tomando las señales de dirección de lectura de la DRAM, y las señales de selección de los multiplexores de salida de la misma, de forma que se obtenga una señal que valga uno sólo cuando termina dicha operación, es posible controlar un contador de 1 bit para que se realize una demultiplexación de las señales de habilitación de lectura y de escritura, y las direcciones generadas por los contadores. La expresión que toma dicha señal es:

$$Select = c2 \cdot c3 \cdot s0 \cdot s1 \cdot s2$$

Esta señal se utiliza como *Enable* de un contador de 1 bit, cuya salida actúa como selección de demultiplexores (figura 5.16) de modo que, al finalizar las operaciones de lectura y escritura, se produzca un cambio en el elemento en el que se realizan dichas operaciones.

5.2.4 Estructura FCT–MMM de la segunda 1–D DCT

Los bits en serie pasan a la segunda etapa de pre–proceso, donde se realiza la suma y la resta de los mismos. Los bits resultantes van a las estructuras MAC, y de aquí a los registros de salida, habilitados por el producto lógico:

$$Enable = \overline{c3} \cdot \overline{c2} \cdot c1 \cdot c0$$

donde *ci* se refiere a la salida del contador encargado de delimitar el periodo de 13 ciclos en la unidad de control de lectura y de escritura. Esta señal también se utiliza para controlar el multiplexor de las estructuras MAC, de forma que cuando toma el valor lógico 1, permite el paso de las condiciones iniciales.

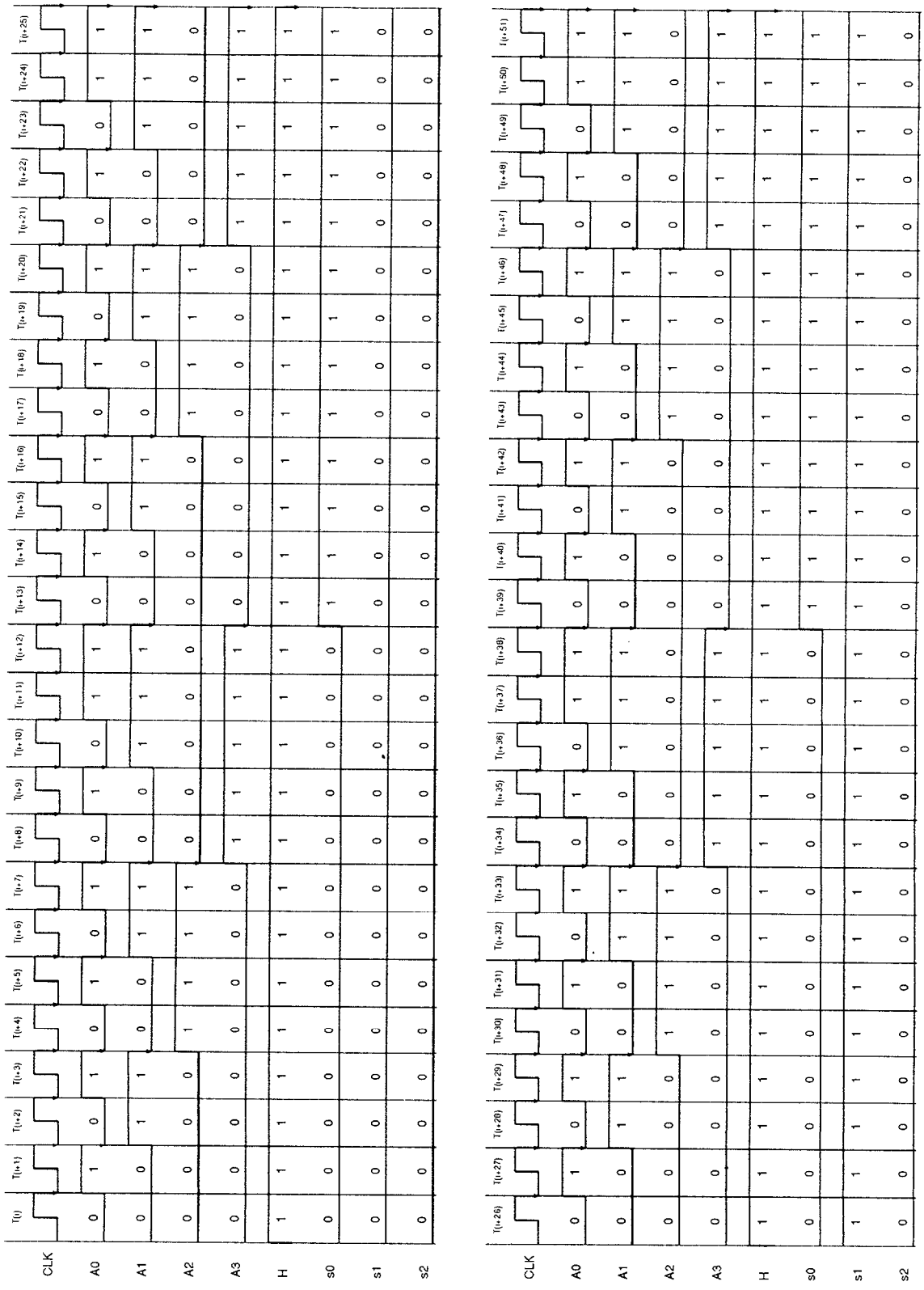


Figura 5.13: Diagrama de tiempos de la RAM (Lectura 1)

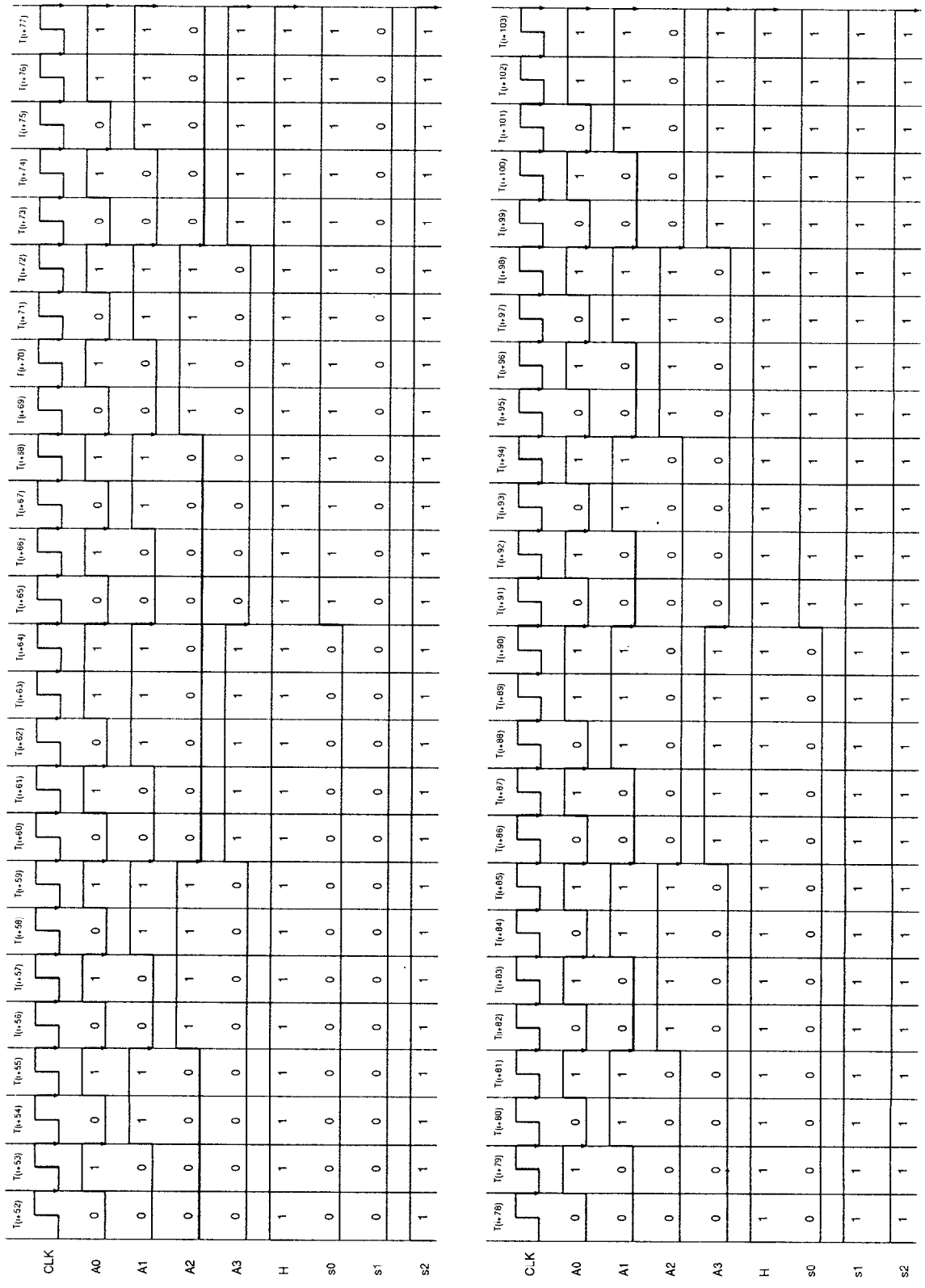


Figura 5.14: Diagrama de tiempos de la RAM (Lectura 2)

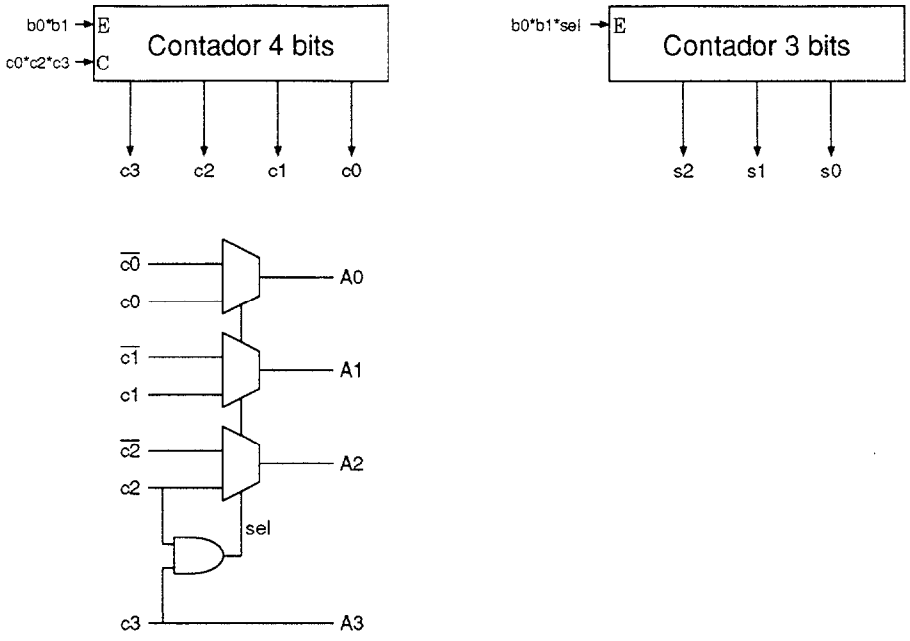


Figura 5.15: Unidad de control para lectura en la DRAM

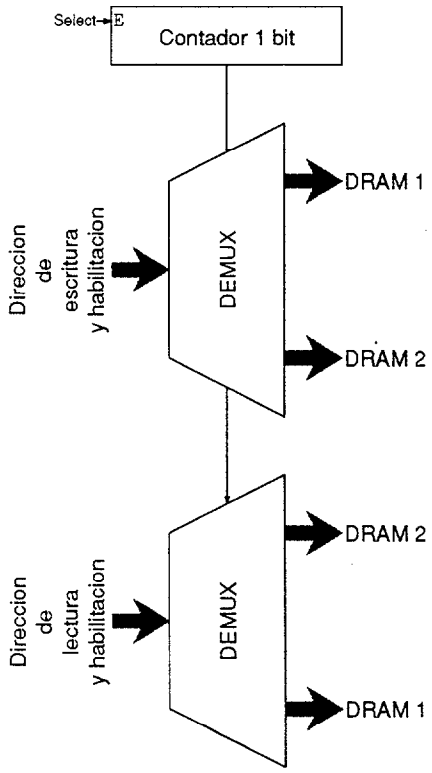


Figura 5.16: Selección del elemento de almacenamiento

5.3 GaAsDCT34: solución de paralelismo para flujo óptimo

La segunda solución propuesta, denominada **GaAsDCT34** (ya que la primera 1–D DCT se implementa con estructuras MAC con una partición 3 y la segunda con estructuras MAC con una partición 4), tiene un esquema muy similar a la solución anterior. La mayor diferencia radica en las estructuras MAC, en las que el paralelismo es mayor.

5.3.1 Conversor paralelo/serie

El conversor paralelo/serie, implementado también con dos bancos de 72 registros (representado en la figura 5.17) tiene a su entrada, en este caso, dos palabras de 9 bits cada ciclo. Para la conversión se dispone, al igual que en el caso anterior, un primer banco de 72 registros (dividido, en este caso, en cuatro grupos de registros de forma que cada grupo almacena **dos palabras** de la columna de entrada), que almacena las 8 palabras que forman la columna de entrada. Un segundo banco, también de 72 registros, se coloca a la salida del primero, de forma que, cuando haya entrado toda la columna, entran las dos primeras palabras de la siguiente columna, que se almacenan en el primer grupo de registros y los 72 bits que estaban almacenados en el primer banco de registros pasan al segundo banco de registros. Cuando las palabras están almacenadas en el segundo banco de registros, comienza la lectura de los datos de 3 en 3 bits, siendo necesaria la introducción previa de 2 bits de valor lógico 0, y la lectura del bit de mayor peso (b0(k)) dos veces consecutivas, para así aplicar la extensión de signo.

Las señales de control a generar para esta etapa se pueden ver en la figura 5.18, donde se puede observar que, en cada ciclo, se habilita la escritura de dos grupos de 9 registros del primer banco de registros, almacenando así en 4 ciclos todas las palabras de la columna correspondiente. Cuando se vuelve a habilitar el almacenamiento de los dos primeros grupos de registros, se habilita también el almacenamiento del segundo banco de registros. Una vez almacenados los datos en ese segundo banco de registros, comienza la lectura de 3 en 3 bits de los datos con la introducción previa de dos bits de valor lógico 0.

Para controlar el conversor paralelo/serie, es necesario el uso de un contador, encargado de delimitar los 4 *ciclos que constituyen el periodo* de las señales. También es necesario un banco de 4 registros, que actúen a modo de registro de desplazamiento, encargado de generar las *señales de habilitación* de los diversos grupos de registros que constituyen el conversor paralelo/serie. Esta estructura se representa en la figura 5.19.

El registro de desplazamiento está constituido por 1 *flip–flop* tipo D con señal de *preset* y 3 *flip–flops* tipo D con señal de *clear*. De esta forma, pasado el estado de inicialización, en el que el primer registro está cargado con un nivel lógico 1, éste nivel lógico se va desplazando cada ciclo de reloj a través del registro de desplazamiento. Cuando pasan 4 ciclos, la salida del último *flip–flop* pasa al primero, repitiéndose la secuencia.

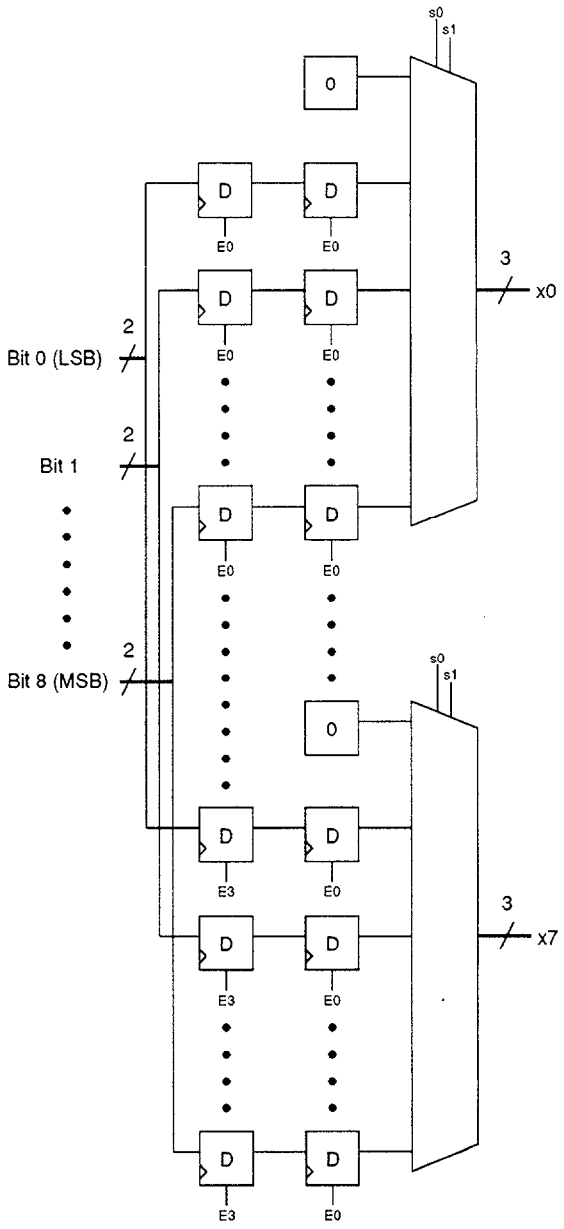


Figura 5.17: Esquema de bloques del conversor paralelo/serie

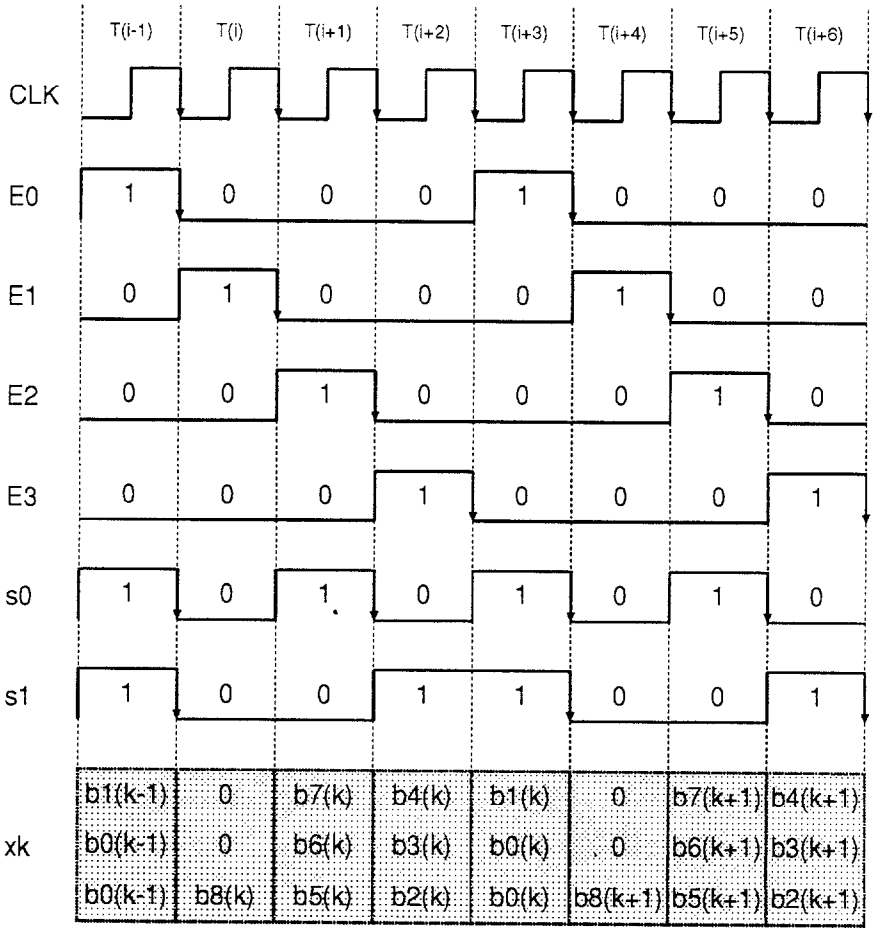


Figura 5.18: Diagrama de tiempos del conversor paralelo/serie

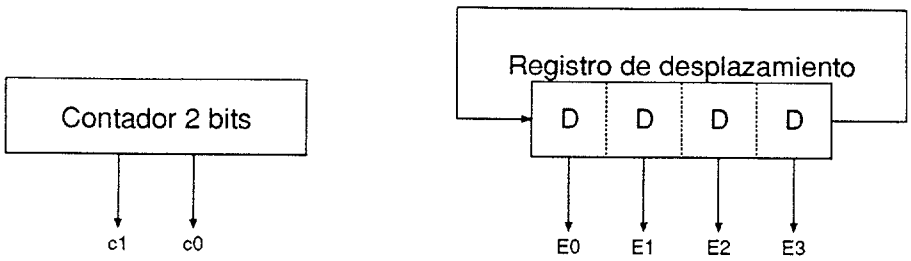


Figura 5.19: Unidad de control del conversor paralelo/serie

Para generar las señales de selección del multiplexor es necesaria la adición de una puerta XNOR, que, en base al estado del contador, produzca las señales de control deseadas. La expresión de dichas señales es:

$$\begin{aligned} s0 &= \overline{c0} \\ s1 &= \overline{c0 \oplus c1} \end{aligned}$$

La relación entre la salida del contador y las de selección de los multiplexores se puede ver en la siguiente tabla, donde se puede observar que las señales de selección del multiplexor equivalen a las del contador con un desfase de un ciclo de reloj:

c1	*c0*	*s1*	*s0*
0	0	1	1
0	1	0	0
1	0	0	1
1	1	1	0

5.3.2 Estructura FCT–MMM de la primera 1–D DCT

Los bits pasan a la etapa de pre–proceso, donde se realiza la suma y resta de los mismos. Los bits resultantes van a las estructuras MAC, que funcionan internamente a ciclo de reloj, y producen un resultado cada cuatro ciclos, coincidiendo con la señal E1, señal que también actúa como selección del multiplexor, permitiendo el paso de las condiciones iniciales cuando dicha señal toma el valor lógico 1.

5.3.3 RAM dinámica con transposición

Al igual que en el circuito GaAsDCT11, en el diseño final se toman dos memorias que se sitúan entre dos bancos de registros. En este apartado se estudia la generación de las señales de control de la RAM y las de selección del dato para la segunda 1–D DCT.

5.3.3.1 Escritura de la DRAM

Para escribir los datos en la DRAM, operación que se comienza a realizar coincidiendo con la cuarta vez que se activa la señal E2, es necesario escribir 2 registros cada ciclo de forma que en cuatro ciclos se almacena toda la columna. Este proceso se realiza 8 veces para cada DRAM de forma alternativa.

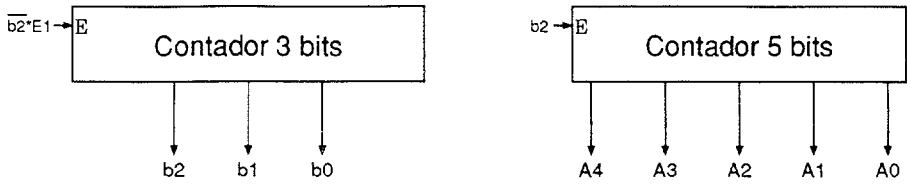


Figura 5.20: Unidad de control para escritura en la DRAM

Para generar estas señales se utilizan 2 contadores (figura 5.20), uno de 3 bits que indica el comienzo de la cuenta (comienzo de la operación) y uno de 5 bits para generar las direcciones de escritura.

La señal de habilitación del contador de 2 bits viene dada por:

$$Enable = \overline{b2} \cdot E1$$

de tal forma que, cuando se active por cuarta vez la señal E2, sea $b2$ la señal que dicte el comienzo de la operación de escritura. El diagrama de estados de la señal *Enable* del contador de 3 bits se puede ver en la siguiente tabla, donde se puede observar como, tras haberse activado cuatro veces la señal E1, la señal de comienzo de escritura pasa a nivel alto, y no vuelve a cambiar de estado.

$b2$	$b1$	$b0$	$E1$	*Enable*	$b2$
0	0	0	0	0	0
0	0	0	1	1	0
0	0	1	0	0	0
0	0	1	1	1	0
0	1	0	0	0	0
0	1	0	1	1	0
0	1	1	0	0	0
0	1	1	1	1	0
1	0	0	0	0	1
1	0	0	1	0	1

La señal de habilitación del contador de 5 bits coincide con la habilitación de escritura de la DRAM, que en este caso no necesita la inclusión de ciclos de espera, ya que almacena 2 palabras cada ciclo de reloj, y necesita escribir 8 palabras cada cuatro ciclos, lo que implica una escritura ininterrumpida en la memoria.

5.3.3.2 Lectura de la DRAM

Simultáneamente a la escritura, se va realizando la operación de lectura en la memoria adyacente. Para leer los datos de la DRAM, se va generando la dirección que indica la posición de los bits a leer (de 3 en 3), seleccionando con el multiplexor la columna de la que se está leyendo. Es necesaria la adición de tres bits de valor lógico cero para adaptar la longitud de la palabra así como la lectura del último bit dos veces consecutivas para realizar la extensión de signo. El proceso de lectura de la memoria también

se realiza de forma alternativa, de forma que cuando se termina de leer una de ellas se comienza a leer la siguiente y viceversa.

A la salida de la DRAM, cada ciclo se obtienen 3 bits de la palabra almacenada, comenzando por los bits menos significativos, de forma que, la secuencia de salida será:

	Bits de salida		
ciclo i	b11	b10	b9
ciclo i+1	b8	b7	b6
ciclo i+2	b5	b4	b3
ciclo i+3	b2	b1	b0

Esta secuencia, antes de pasar a la etapa de pre–proceso, ha de convertirse en:

	Bits de entrada al pre–proceso			
ciclo i	0	0	0	b11
ciclo i+1	b10	b9	b8	b7
ciclo i+2	b6	b5	b4	b3
ciclo i+3	b2	b1	b0	b0

Para ello, los registros posteriores a la DRAM, han de almacenar tanto los bits leidos en un ciclo determinado como en el siguiente, y mediante multiplexores de 4 a 1, se realiza la selección de los datos, como se muestra en la figura 5.21.

La unidad de control (figura 5.22) consiste en un contador de dos bits que vaya generando las direcciones de lectura de los bits de la DRAM, un contador de 3 bits que vaya seleccionando la columna de la que se está realizando la lectura y la lógica necesaria para los multiplexores de 4 a 1, que permitan la lectura de los datos a través de los multiplexores. Esta lógica consiste en una puerta XNOR, de forma que, la salida de la lógica coincida con el estado anterior del contador:

$$\begin{aligned} p0 &= \overline{A0} \\ p1 &= \overline{A0 \oplus A1} \end{aligned}$$

La habilitación del contador de 3 bits que selecciona la columna de la DRAM viene dada por el producto lógico de $b2$ y los bits de salida del contador encargado de generar las direcciones de lectura ($A0$ y $A1$), de forma que cuando se hayan leído todos los bits de una columna, se produzca un cambio en el contador de 3 bits.

5.3.3.3 Selección del elemento de almacenamiento

Al igual que en el caso de la anterior arquitectura, las operaciones de escritura y lectura se realizan de forma simultánea, pero sobre memorias distintas.

Tomando las señales de dirección de lectura de la DRAM, y las señales de selección de los multiplexores de salida de la misma, de forma que se obtenga una señal que

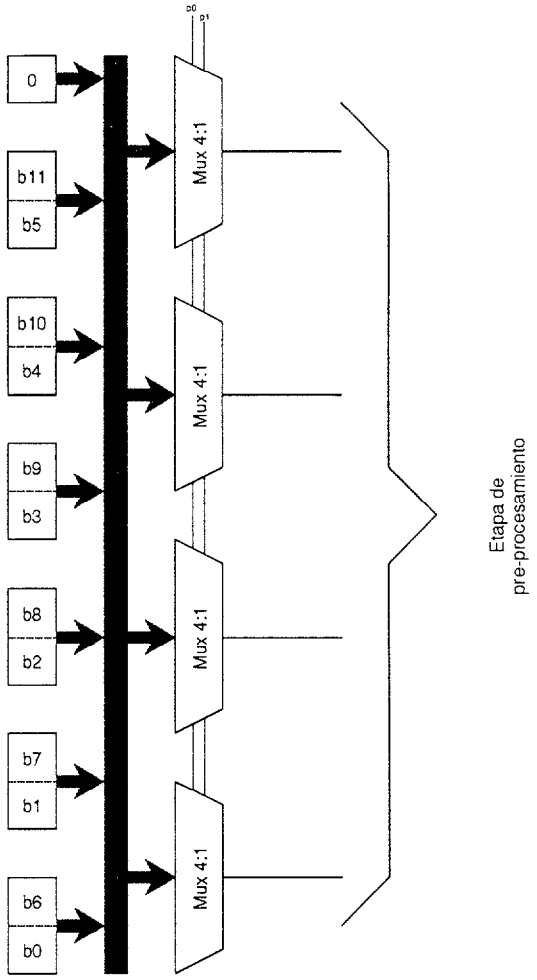


Figura 5.21: Estructura de los registros posteriores a la DRAM

valga uno sólo cuando termina dicha operación, es posible controlar un contador de 1 bit para que se realize una demultiplexación de las señales de habilitación de lectura y de escritura, y las direcciones generadas por los contadores. La expresión que toma dicha señal es:

$$Select \;=\; A0 \cdot A1 \cdot s0 \cdot s1 \cdot s2$$

Esta señal se utiliza como *Enable* de un contador de 1 bit, cuya salida actúa como selección de demultiplexores, al igual que en el caso de la arquitectura anterior.

Figura 5.22: Unidad de control para lectura de la DRAM

5.3.4 Estructura FCT–MMM de la segunda 1–D DCT

Los bits en serie pasan a la segunda etapa de pre–proceso, donde se realiza la suma y la resta de los mismos. Los bits resultantes van a las estructuras MAC, y de aquí a los registros de salida, habilitados por el producto lógico:

$$Enable = A0 \cdot \overline{A1}$$

donde Ai se refiere a la salida del contador encargado de generar la dirección de lectura de la DRAM. Esta señal también se utiliza para controlar el multiplexor de las estructuras MAC, de forma que permite el paso de las condiciones iniciales cuando toma el valor lógico 1.

5.4 Integración de los circuitos

En este apartado se presenta la realización física de los circuitos. Uno de los principales problemas a abordar, tanto en el GaAsDCT11 como en el GaAsDCT34, es que ambos circuitos constituyen lo que se denomina chips limitados por el núcleo. Estos chips son aquellos en los que el número de patillas de entrada/salida es muy bajo y, sin embargo, el área ocupada por el circuito es elevada. En este caso, la única solución para realizar una integración monolítica consiste en tomar aquel encapsulado que permita alojar el diseño, aunque se desaprovechen gran parte de los pads. Ultimamente, una solución adicional es la realización de un módulo multichip (MCM, Multi Chip Module). En este proyecto se ha realizado un estudio prospectivo utilizando ambas soluciones en razón a la complejidad y al coste final de los circuitos.

5.4.1 Circuito GaAsDCT11

Para el circuito GaAsDCT11 se ha optado por el uso del encapsulado **PG149**, con un tamaño de dado de 7.35×4.33 mm^2 y una frecuencia máxima de trabajo de 700 MHz. El encapsulado dispone de 120 pines destinados a la entrada y a la salida de datos, de los que solo se usarán 9 pines de entrada y 12 de salida, a los que hay que sumar el de la señal de *reset* del circuito.

El floorplan de la disposición final de los distintos bloques que conforman el circuito se puede observar en la figura 5.23.

5.4.2 Circuito GaAsDCT34

Para el circuito GaAsDCT34, se pueden optar por dos soluciones:

- Integración en el encapsulado **LD344**, con un tamaño de dado de 13.78×7.73 mm^2 y con una frecuencia máxima de trabajo de 600 MHz. Este encapsulado dispone de 256 pines destinados a la entrada y salida de datos, de los que solo se usarán 18 pines de entrada y 24 de salida, a los que hay que

sumar el de *reset* del circuito. La disposición de los distintos bloques se puede observar en la figura 5.24.

- Integración en un **MCM**, realizando tres particiones, de forma que la primera contenga la primera 1–D DCT (figura 5.25), que tiene 18 líneas de entrada y 25 de salida, la segunda, la etapa de almacenamiento (figura 5.26), con 25 líneas de entrada y 28 de salida, y la tercera, la segunda 1–D DCT (figura 5.27), que tiene 28 líneas de entrada y 24 de salida.

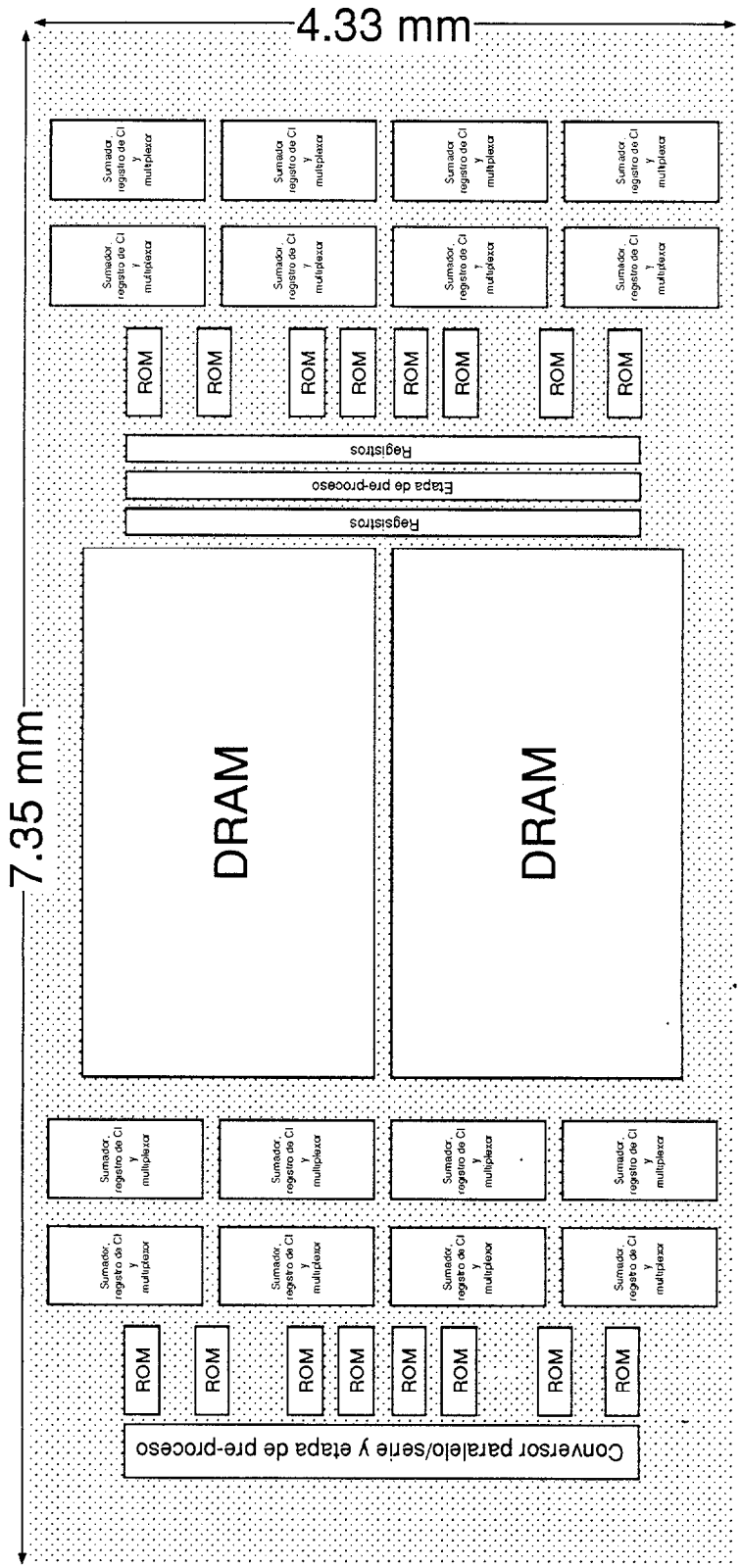


Figura 5.23: Disposición de los distintos bloques del circuito GaAsDCT11

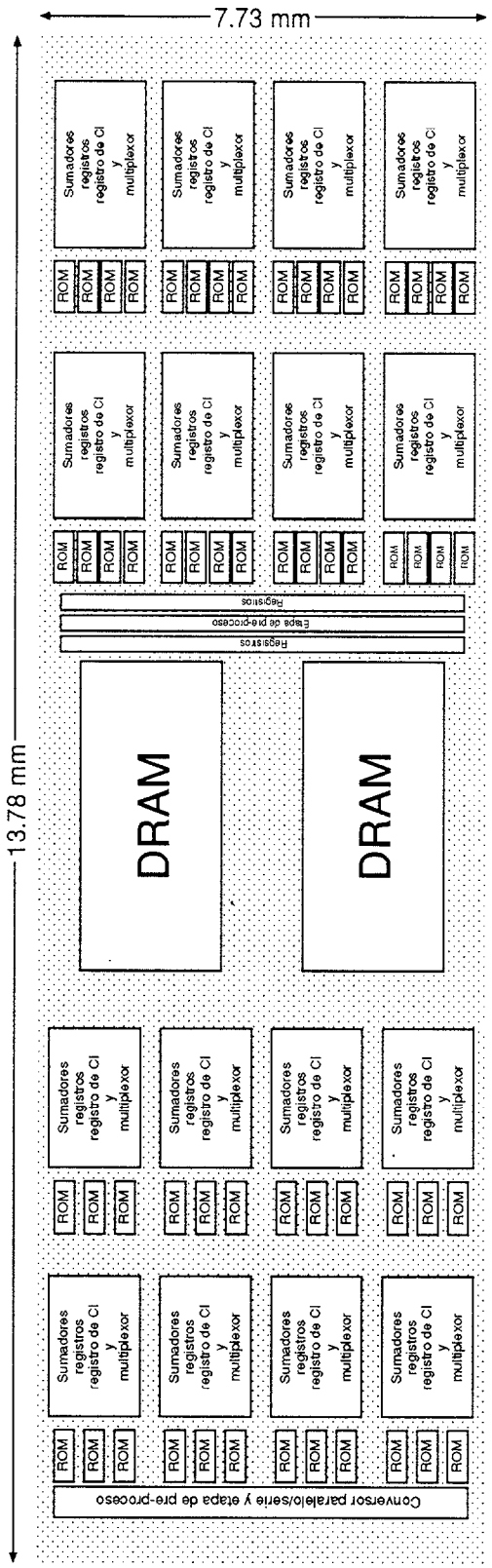


Figura 5.24: Disposición de los distintos bloques del circuito GaAsDCT34

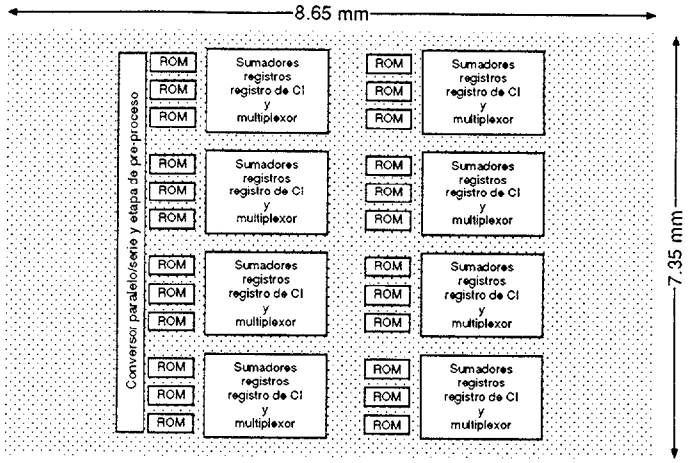


Figura 5.25: Primera partición del circuito GaAsDCT34

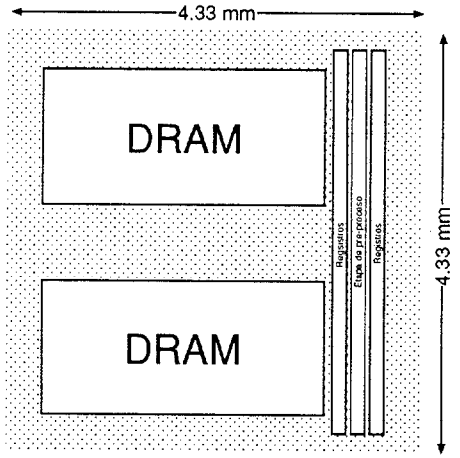


Figura 5.26: Segunda partición del circuito GaAsDCT34

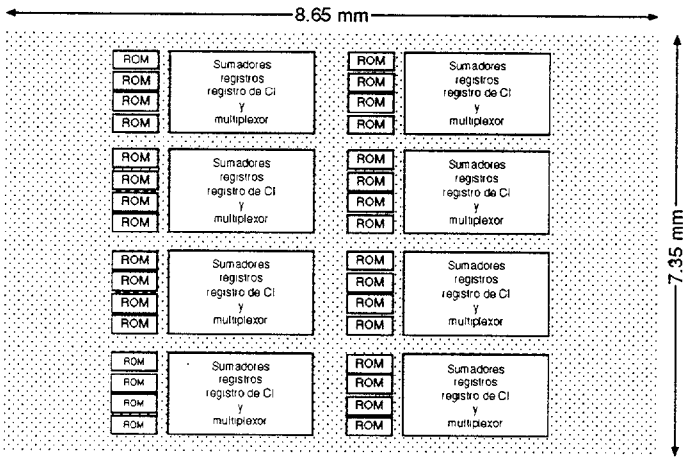


Figura 5.27: Tercera partición del circuito GaAsDCT34

5.5 Prestaciones de los circuitos

La frecuencia de trabajo del circuito viene marcada por el retardo del registro, del sumador y del multiplexor de la unidad de multiplicación-acumulación. Esta frecuencia es independiente de la partición elegida, ya que la paralelización no implica ningún cambio en el bucle de realimentación que determina el ciclo de trabajo. En base a los estudios realizados y a los resultados de retardo obtenidos, la frecuencia de trabajo de las estructuras MAC, y por tanto del circuito, para distintas dispersiones en los parámetros tecnológicos es:

Dispersión	*Frecuencia de trabajo*
SS2 ($0°C$)	475 MHz
TT ($75°C$)	650 MHz
FF1H ($100°C$)	700 MHz

Una comparación entre distintos circuitos y los dos diseñados en este proyecto se puede ver en la siguiente tabla:

Diseño	Ecuación	Tipo de implementación	Proceso (μm)	N° Transistores	Throughput (MPel/s)	Area (mm^2)
CMOS	2 * 1-D	FCT	1.3	80000	27	20
CMOS	2-D	Mult	1.5	220000	20	196
CMOS	2 * 1-D	DA	1.2	50000	27	26
CMOS	2 * 1-D	DA	2	54750	15	50
BiCMOS	2 * 1-D	DA	0.8	78666	100	26
CMOS	2 * 1-D	Mult	0.8	138000	72	48
CMOS	2 * 1-D	Mult	1.5	280000	45	91
CMOS	2 * 1-D	DA	2	67929	55	73
CMOS†	2 * 1-D	DA	0.5	120000	200	17
GaAsDCT11	2 * 1-D	DA	0.6	50822	400	31.8
GaAsDCT34	2 * 1-D	DA	0.6	129794	1300	106.5

† *Lógica diferencial de bajo swing.*

De la tabla de prestaciones de deduce que la versión GaAsDCT34, con paralelismo para flujo óptimo, es más de tres veces más rapida que la version GaAsDCT11. Sin embargo el consumo en área (3 veces más en la primera) y el de potencia (6.8 W para GaAs DCT11 y 15.5 W para GaAsDCT34) parece indicar la necesidad de incluir una comparación de tipo mas general. Para ello se introduce una **figura de mérito** que sirva para evaluar las prestaciones globales de los circuitos presentados en la tabla mostrada anteriormente. Esta figura de mérito debe ser independiente del proceso tecnológico usado (proporcional por tanto a la dimensión característica del mismo) y debe evaluar aspectos como la capacidad computacional, la complejidad de la arquitectura y la bondad final del diseño físico conseguido.

La figura de mérito propuesta es la que aparece en la siguiente expresión:

$$FM = \frac{throughput \cdot proceso}{transistores \cdot área}$$

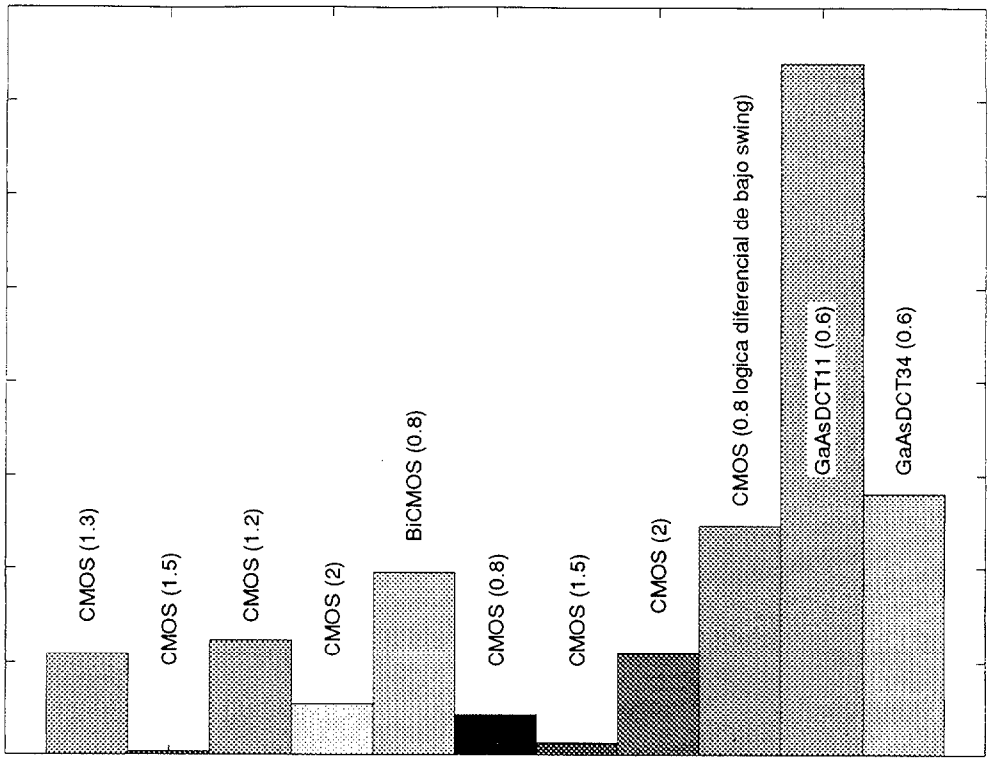


Figura 5.28: Comparación entre los circuitos

En la figura 5.28 se representan los valores obtenidos para la figura de mérito, según el orden reflejado en la tabla. Como se puede observar, las prestaciones son superiores en los diseños realizados en tecnología GaAs, destacando el circuito **GaAsDCT11**, debido al elevado *throghput* que posee y a los pequeños requisitos de área que su implementación requiere.

Bibliografía

[1] K. Joseph et al, "MPEG ++: A robust compresion and transport system for digital HDTV", Signal Processing: Image Communication, Vol. 5, Nos. 1–2, pp. 307-323, February 1993.

[2] Cheng–Tie Chen et al, "Hybryd extended MPEG video coding algorithm for general video applications", Signal Processing: Image Communication, Vol. 5, Nos. 1–2, pp. 21-37, February 1993.

[3] Wei–Wei Lu and M. P. Gough, "A Fast–Adaptative Huffman Coding Algorithm", IEEE Transactions on Communications, Vol. 41, No. 4, pp. 535-539, April 1993.

[4] Weiss and Schremp, "Putting data on a diet", Communications/Tutorial, IEEE Spectrum, pp. 36-39 August 1993.

[5] OII IMSTAND, "Image Compression Techniques", Report prepared by PIRA International for the Commission of the European Communities, URL:http://www.echo.lu/impact/oii/compress.html

[6] Brian Smith and Ramin Zabith, "Computer Science 610", Image Compression Course, URL:http://simon.cs.cornell.edu/Info/Courses/Fall-94/CS610/

[7] "CCITT Rec. H.261 Standard Text", URL:http://icib.igd.fhg.de/icib/telecom/ccitt/rec_h.261-1990/read.html

[8] "CCITT H.261. Understanding H.261 Image Compression", URL:http://icib.igd.fhg.de/icib/telecom/ccitt/rec_h.261-1990/pvrg-descript/chapter2.5.html

[9] "ISO IEC IS 10918 1 JPEG Standard Text", URL:http://icib.igd.fhg.de/icib/it/iso/is_10918-1/read.html

[10] "Understanding JPEG Image Compression", URL:http://icib.igd.fhg.de/icib/it/iso/is_10918-1/pvrg-descript/chapter2.5.html

[11] "A Knoledge–Based Approach to JPEG Acceleration", URL:http://www-vs.informatik.uni-ulm.de/Papers/DVC95/DVC95paper.html#RTFToC6

[12] "MPEG II Video", URL:http://icib.igd.fhg.de/icib/it/iso/cd_13818-2/gen.html

[13] Bryan Ackland, "The Role of VLSI in Multimedia", IEEE Journal of Solid–State Circuits, Vol. 29, No. 4, pp. 381-388 , April 1994.

[14] K.R. Rao, P. Yip, "Discrete Cosine Transform: Algorithms, Advantages and Applications", Academic Press, Inc, 1990, ISBN 0-12-580203-X

[15] Y.H. Chan and W.C. Siu, "On the Realization of the Discrete Cosine Transform using Distributed Arithmetic", IEEE Transactions on Circuits and Systems, Vol. 39, Num. 9, pp. 705-712, September 1992.

[16] Shing–Chow Chan and Ka–Leung Ho, "Fast Algorithms for Computing the Discrete Cosine Transform", IEEE Transactions on Circuits and Systems, Vol. 39, Num. 3, pp. 185-190, Marzo 1992.

[17] P. Barr, "Optimal Architectures for implementation of the 8×8 DCT", `ftp://ftp.crs4.it/mpeg/misc-docs/optimal_architectures_for_dct.ps.Z`

[18] V. Srinivasan and K.J.R. Liu, "VLSI Design of High–Speed Time–Recursive 2–D DCT/IDCT Processor for Video Application", ISR TR 94–60, University of Maryland, `URL:http://dspserv.eng.umd.edu/list/report/report.html`

[19] J.D. Bruguera and T. Lang, "Parallel 2–D DCT Using On Line Arithmetic"

[20] Byeong Gi Lee, "A New Algorithm to Compute the Discrete Cosine Transform", IEEE Transactions on Acoustics, Speech and Signal Processing, Vol. ASSP-32, Num. 6, pp. 1243-1245, December 1984.

[21] Nam Ik Cho and Sang Uk Lee, "Fast Algorithm and Implementation of 2–D Discrete Cosine Transform", IEEE Transactions on Circuits and Systems, Vol. 38, Num. 3, pp. 297-305, March 1991.

[22] Shin-ichi Uramoto et al, "A 100-MHz 2-D Discrete Transform Cosine Core Processor", IEEE Journal of Solid State Circuits, Vol. 27, Num. 4, pp. 492-498, April 1992.

[23] J.D. Bruguera, "VLSI Architectures for the 2–D Discrete Cosine Transform with Low Power Consumption", September 1993.

[24] Anrew B. Watson, "Image Compression Using the Discrete Cosine Transform", Mathematica Journal, 4(1), pp. 81-88, 1994.

[25] Peter Pisch, Nicolas Demassieux and Winfried Gehrke, "VLSI Architectures for Video Compression–A Survey", Proceedings of IEEE, Vol. 83, Num. 2, pp. 220-246, February 1995.

[26] Mario Kovac and N. Ranganathan, "JAGUAR: A Fully Pipelined VLSI Architecture fo JPEG Image Compression Standard", Proceedings of IEEE, Vol. 83, Num. 2, pp. 247-258, February 1995.

[27] P.C. Kerr, "A 180 Mflops Matrix Processor for Multimedia Data Compression", 13th Australian Microelectronic Conference, pp. 103-108, July 1995

[28] Valentín de Armas, "Desarrollo de subsistemas aceleradores hardware de muy alta velocidad en tecnología de Arseniuro de Galio de 0.8μm", Proyecto final de Carrera, Universidad de Las Palmas de Gran Canaria, Septiembre 1991.

[29] R. Sarmiento, P.P. Carballo and Antonio Nuñez, "High Speed primitives of hardware accelerators for DSP in GaAs technology", IEE Proceedings-G, Vol. 139, Num. 2, pp. 205-216, April 1992.

[30] K. Eshraghian "Fundamentals of Gallium Arsenide VLSI Systems", *Silicon Systems Engineering Series*, Australia, 1994.

[31] Vitesse Semiconductor Corporation, "Foundry Design Manual, Version 6.0", Mayo 1993.

[32] Stephen I. Long and Steven E. Butner, "Gallium Arsenide Digital Integrated Circuit Design", *McGraw-Hill*, 1990.

[33] J. Fco. López, "Aportaciones al Diseño de Memorias ROM en Tecnología GaAs", Tesis Doctoral, Universidad de Las Palmas de Gran Canaria, Noviembre 1994.

[34] J. Fco. López, K. Eshraghian, M.K. McGeever, A. Núñez y R. Sarmiento, "Gallium Arsenide MESFET Memory Architectures", IEEE International Workshop on Memory Technology, Design and Testing, pp. 103-108, San José, California, Agosto 1995.

[35] M. K. McGeever, "A 14kbit GaAs DRAM for an ATM packet switch", Centre for Gallium Arsenide VLSI Technology, Department of Electrical and Electronic Engineering, Universidad de Adelaida, Informe Interno.

[36] J. Jacobsen, "Low Power GaAs Multiplexer and Demultiplexer", Proceedings of the European Gallium Arsenide and related III-V Compounds, Applications Symposium, pp. 199-202, Torino 1994.

Layout de las Primitivas de Diseño

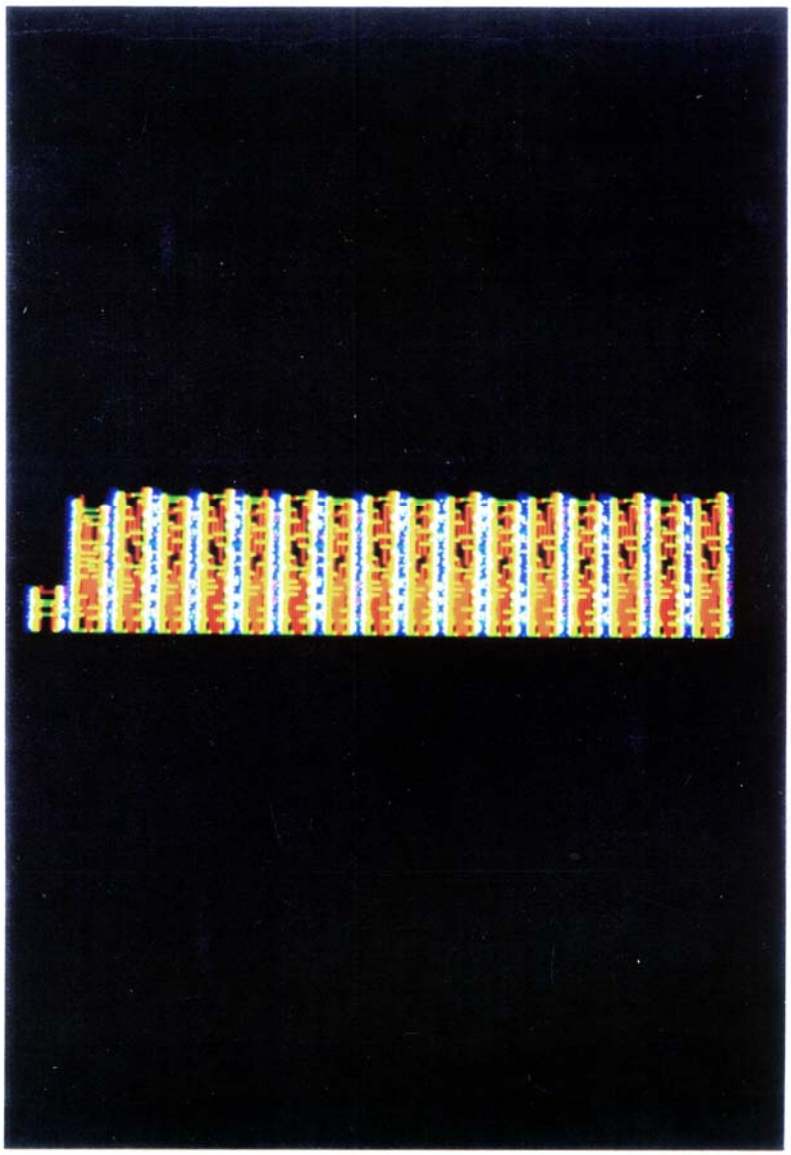


Figura 1: Sumador de acarreo anticipado de 17 bits

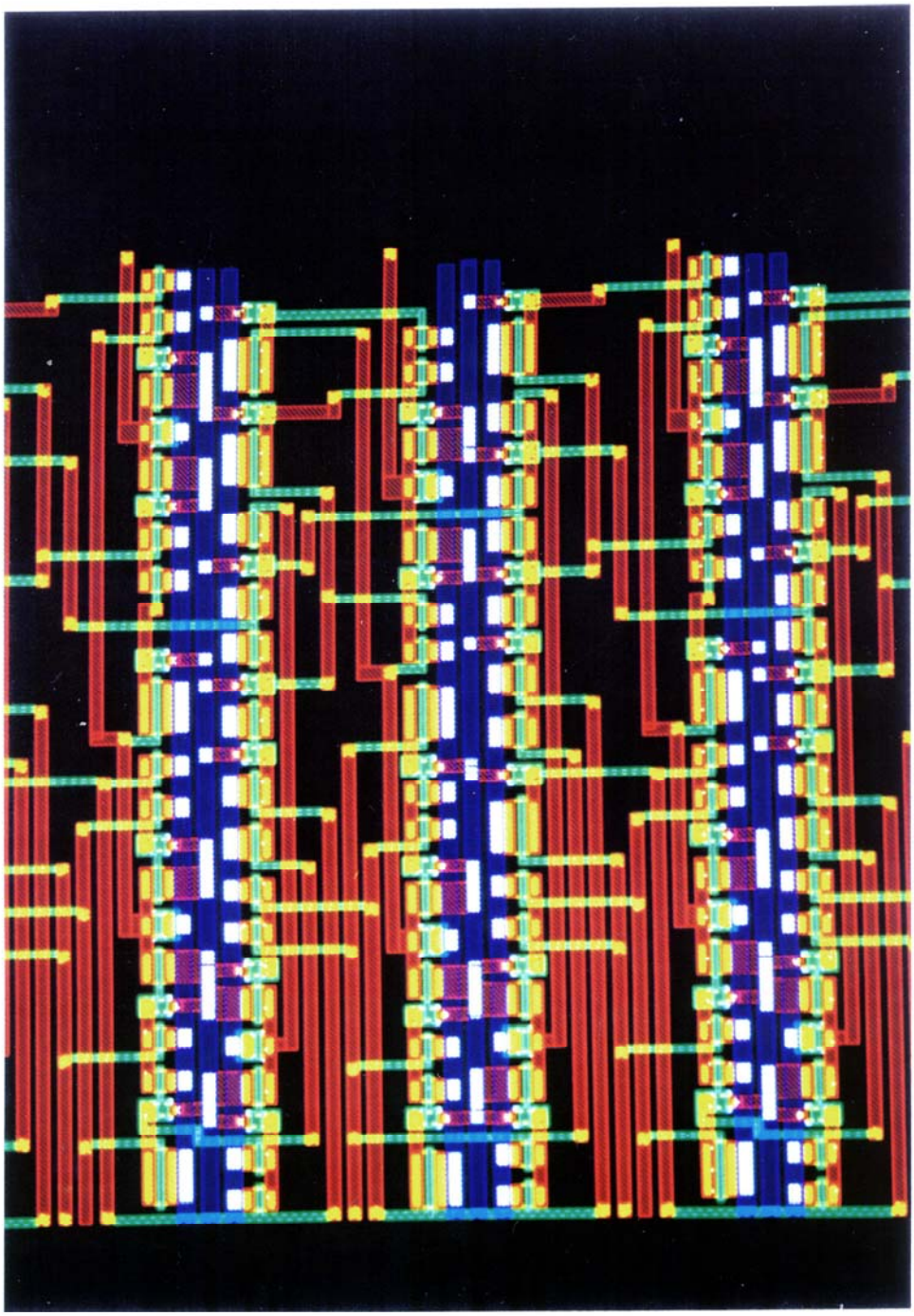


Figura 2: Detalle del sumador

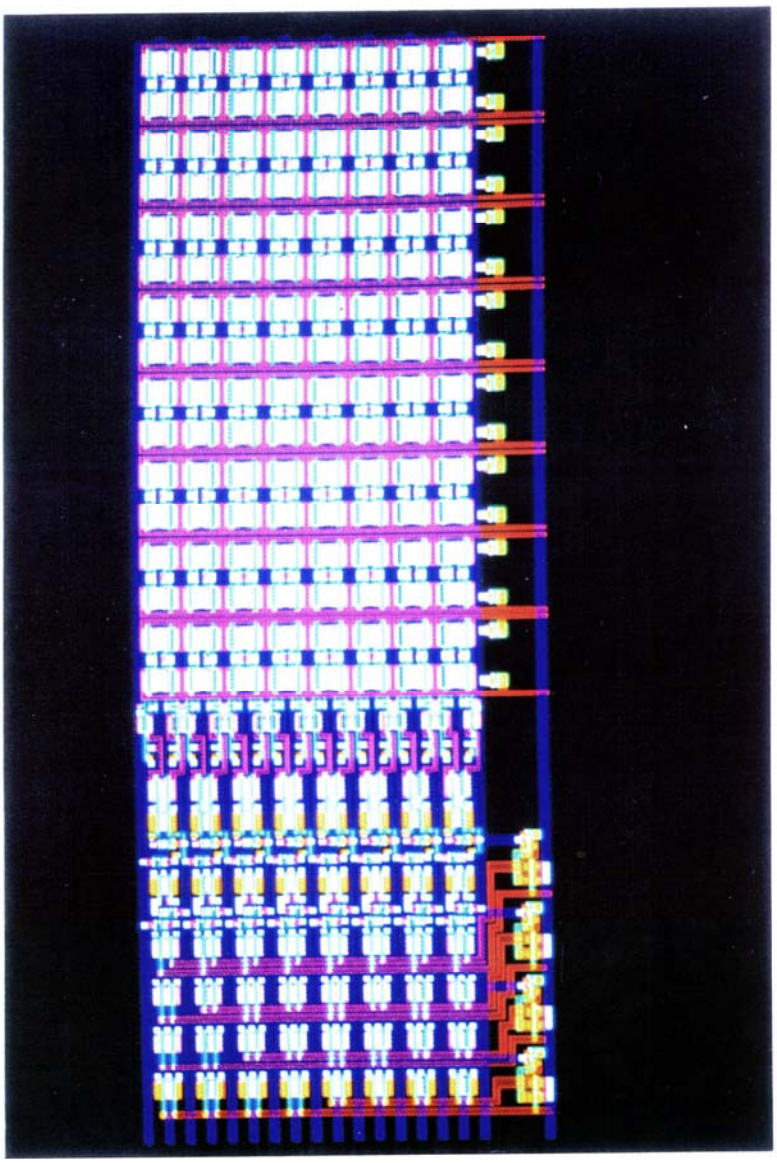


Figura 3: Memoria ROM

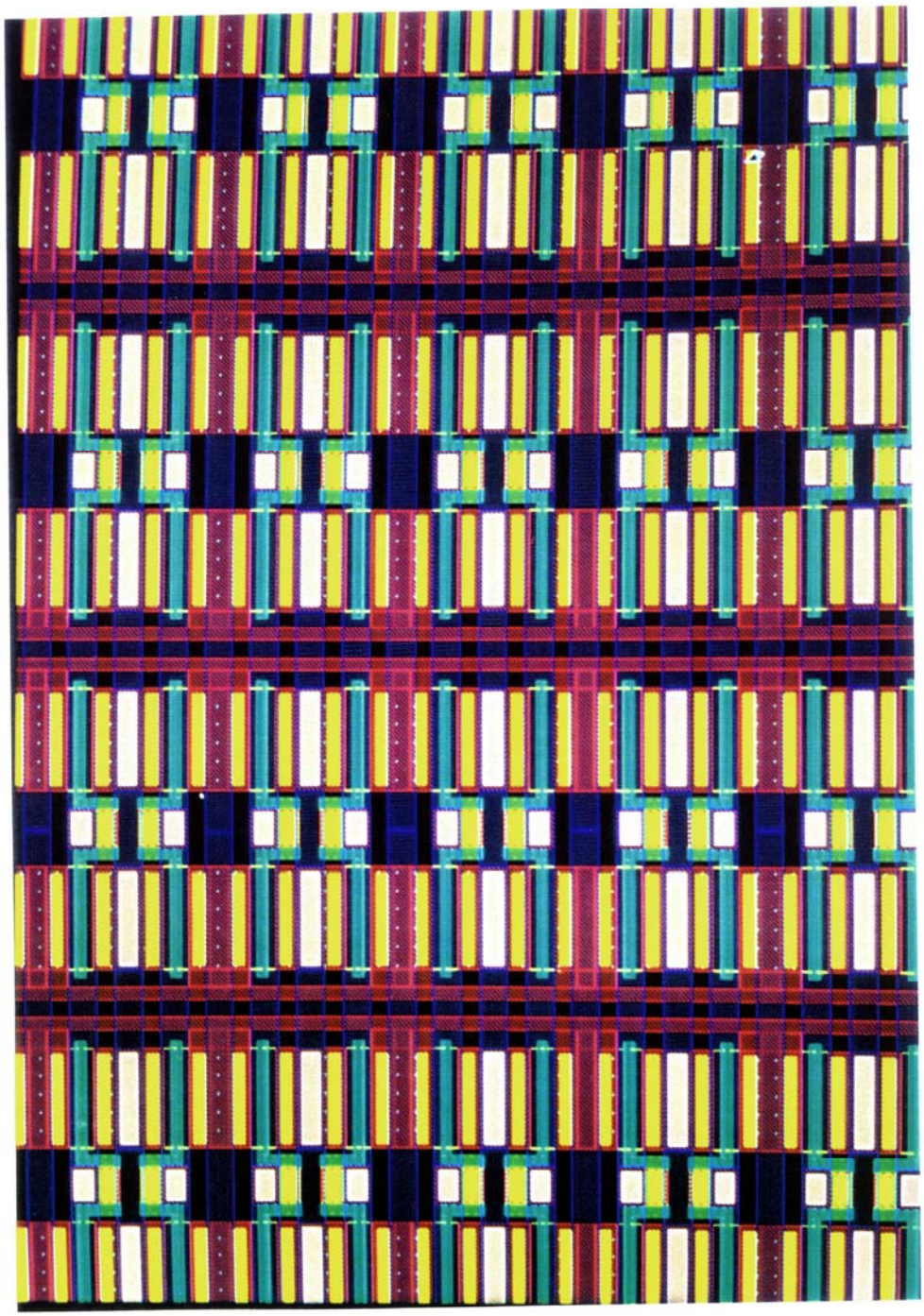


Figura 4: Detalle de la matriz de la ROM

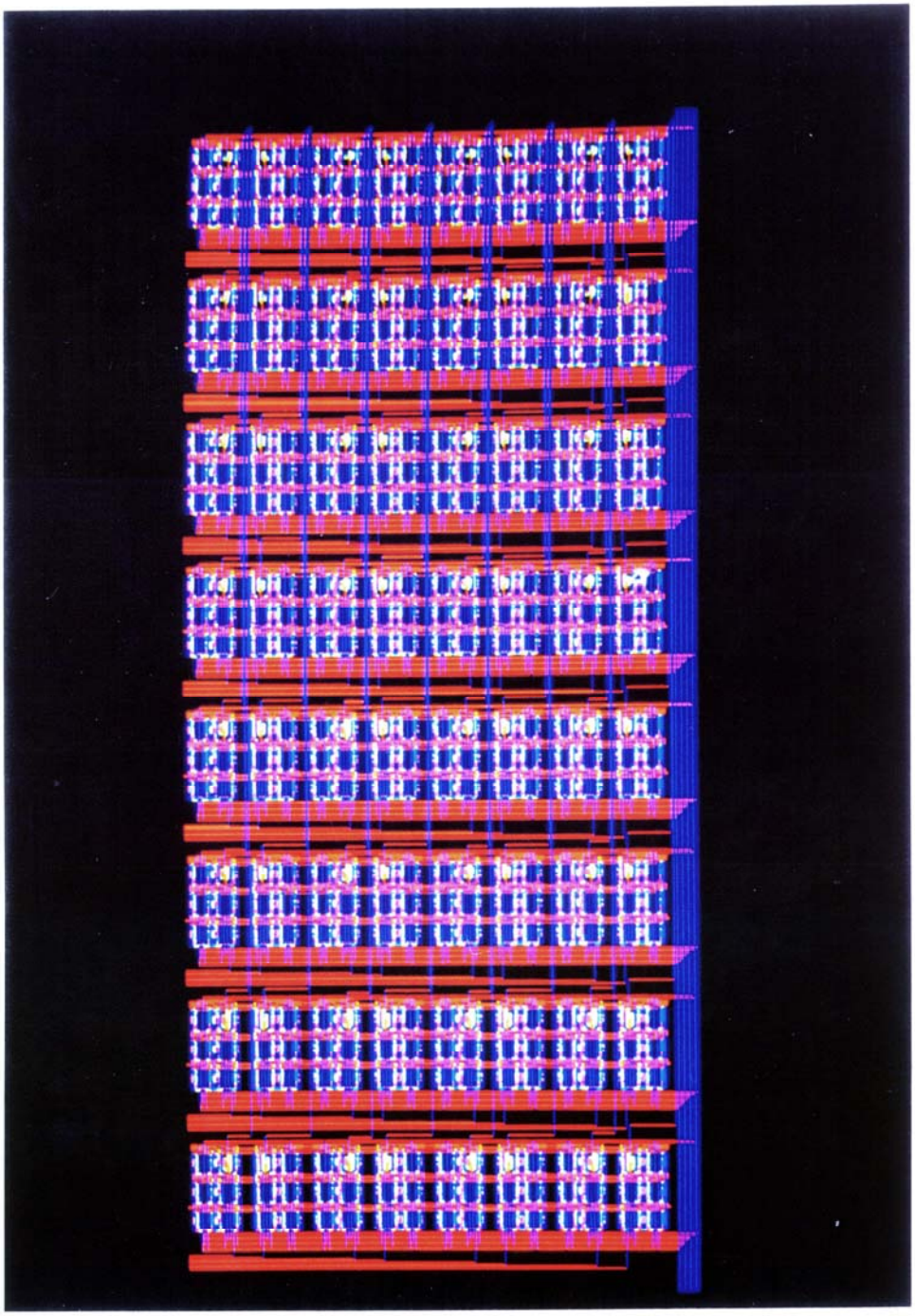


Figura 5: Matriz de la DRAM

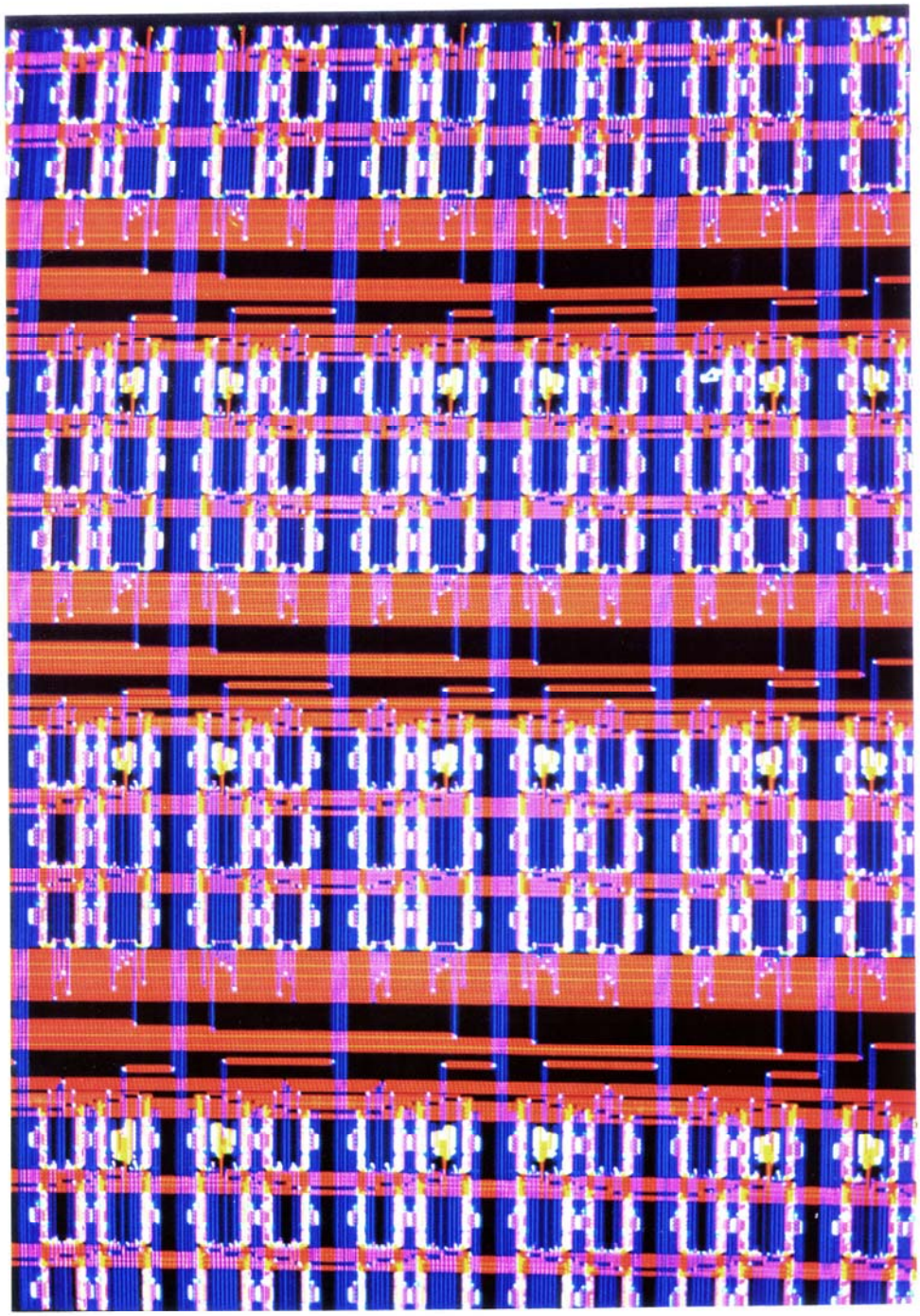


Figura 6: Detalle de la matriz de la DRAM

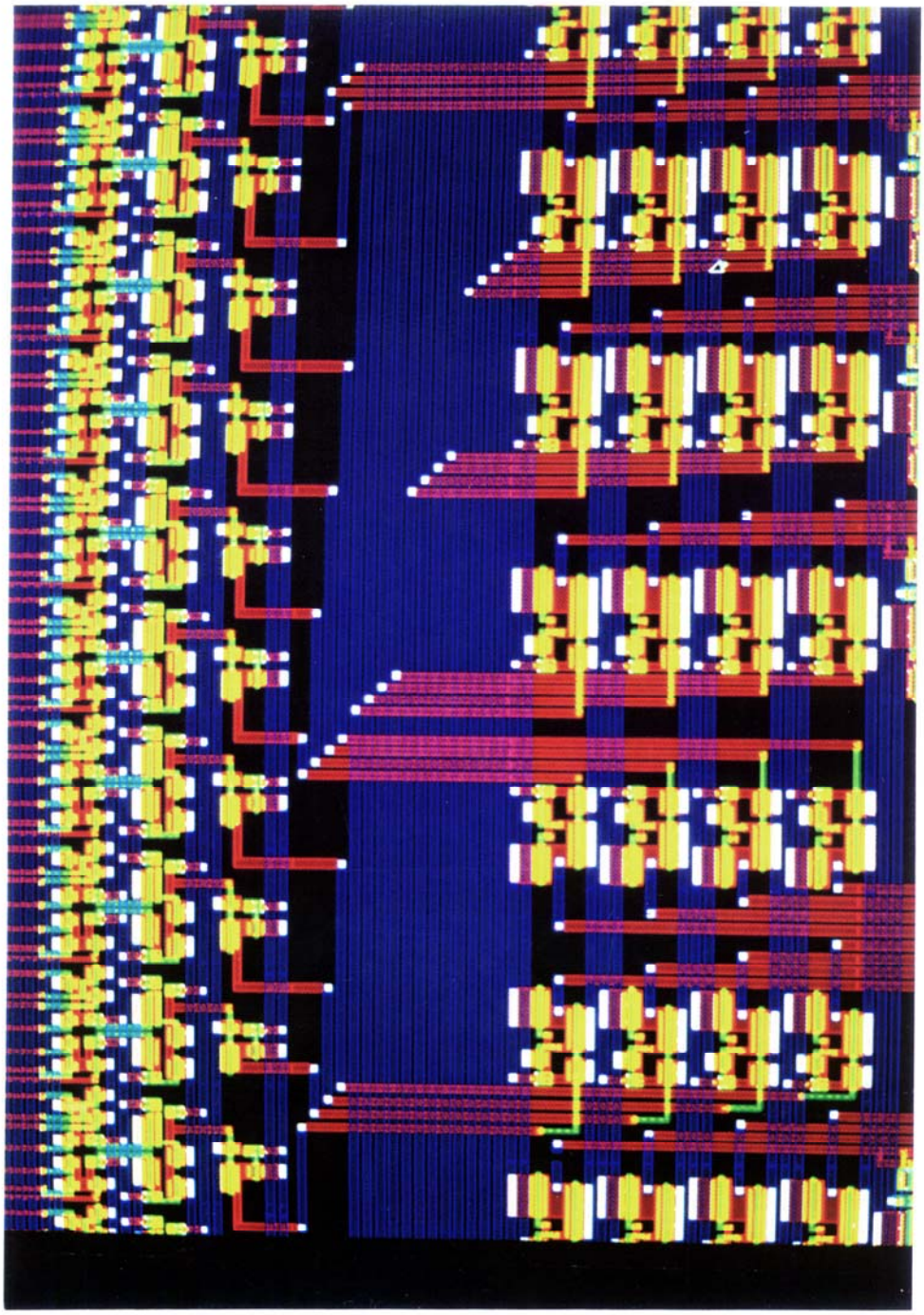


Figura 7: Detalle del decodificador de escritura y de los adaptadores de tensión

Anexo A

Estudio de diferentes particiones en las estructuras de multiplicación-acumulación

Estructuras MAC con partición 3

Esta estructura (figura 1) sólo es capaz de procesar palabras con un número de bits múltiplo de 3, por lo que es necesario adaptar la palabra de entrada tanto cuando se va a utilizar en la primera 1–D DCT, como cuando se va a utilizar en la segunda. En el caso de las palabras de entrada a la primera 1–D DCT hay que adaptar la longitud hasta 12 bits y en el caso de que las palabras de entrada a la segunda 1–D hay que adaptar dicha longitud hasta 15 bits.

En el caso de que las palabras de entrada sean de 10 bits, en el que hay que procesar 12 bits, la secuencia de operaciones equivale a:

$$\begin{aligned} Salida_{MAC} &= \frac{CI+[ROM]_1}{2^{11}} + \frac{[ROM]_2}{2^{10}} + \frac{[ROM]_3}{2^9} + \frac{[ROM]_4}{2^8} + \frac{[ROM]_5}{2^7} + \\ &+ \frac{[ROM]_6}{2^6} + \frac{[ROM]_7}{2^5} + \frac{[ROM]_8}{2^4} + \frac{[ROM]_9}{2^3} + \frac{[ROM]_{10}}{2^2} + \\ &+ \frac{[ROM]_{11}}{2} + [ROM]_{12} \end{aligned}$$

haciendo $[ROM]_1 = CI$ y $[ROM]_2 = CI$, queda:

$$\begin{aligned} Salida_{MAC} &= \frac{2 \cdot CI}{2^{11}} + \frac{CI}{2^{10}} + \frac{[ROM]_3}{2^9} + \frac{[ROM]_4}{2^8} + \frac{[ROM]_5}{2^7} + \\ &+ \frac{[ROM]_6}{2^6} + \frac{[ROM]_7}{2^5} + \frac{[ROM]_8}{2^4} + \frac{[ROM]_9}{2^3} + \frac{[ROM]_{10}}{2^2} + \\ &+ \frac{[ROM]_{11}}{2} + [ROM]_{12} \end{aligned}$$

y simplificando términos:

$$\begin{aligned} Salida_{MAC} &= \frac{CI+[ROM]_3}{2^9} + \frac{[ROM]_4}{2^8} + \frac{[ROM]_5}{2^7} + \\ &+ \frac{[ROM]_6}{2^6} + \frac{[ROM]_7}{2^5} + \frac{[ROM]_8}{2^4} + \frac{[ROM]_9}{2^3} + \frac{[ROM]_{10}}{2^2} + \\ &+ \frac{[ROM]_{11}}{2} + [ROM]_{12} \end{aligned}$$

que es el secuencia esperada.

Para el caso de la palabra de 13 bits, en el que hay que procesar 15 bits:

$$\begin{aligned} Salida_{MAC} &= \frac{CI+[ROM]_1}{2^{14}} + \frac{[ROM]_2}{2^{13}} + \frac{[ROM]_3}{2^{12}} + \frac{[ROM]_4}{2^{11}} + \frac{[ROM]_5}{2^{10}} + \\ &+ \frac{[ROM]_6}{2^9} + \frac{[ROM]_7}{2^8} + \frac{[ROM]_8}{2^7} + \frac{[ROM]_9}{2^6} + \frac{[ROM]_{10}}{2^5} + \\ &+ \frac{[ROM]_{11}}{2^4} + \frac{[ROM]_{12}}{2^3} + \frac{[ROM]_{13}}{2^2} + \\ &+ \frac{[ROM]_{14}}{2} + [ROM]_{15} \end{aligned}$$

haciendo $[ROM]_1 = CI$ y $[ROM]_2 = CI$, queda:

$$\begin{aligned} Salida_{MAC} = & \frac{2 \cdot CI}{2^{14}} + \frac{CI}{2^{13}} + \frac{[ROM]_3}{2^{12}} + \frac{[ROM]_4}{2^{11}} + \frac{[ROM]_5}{2^{10}} + \\ & + \frac{[ROM]_6}{2^9} + \frac{[ROM]_7}{2^8} + \frac{[ROM]_8}{2^7} + \frac{[ROM]_9}{2^6} + \frac{[ROM]_{10}}{2^5} + \\ & + \frac{[ROM]_{11}}{2^4} + \frac{[ROM]_{12}}{2^3} + \frac{[ROM]_{13}}{2^2} + \\ & + \frac{[ROM]_{14}}{2} + [ROM]_{15} \end{aligned}$$

y simplificando términos:

$$\begin{aligned} Salida_{MAC} = & \frac{CI + [ROM]_3}{2^{12}} + \frac{[ROM]_4}{2^{11}} + \frac{[ROM]_5}{2^{10}} + \frac{[ROM]_6}{2^9} + \\ & + \frac{[ROM]_7}{2^8} + \frac{[ROM]_8}{2^7} + \frac{[ROM]_9}{2^6} + \frac{[ROM]_{10}}{2^5} + \\ & + \frac{[ROM]_{11}}{2^4} + \frac{[ROM]_{12}}{2^3} + \frac{[ROM]_{13}}{2^2} + \\ & + \frac{[ROM]_{14}}{2} + [ROM]_{15} \end{aligned}$$

En la figura 2 se muestra el flujo de datos a través del *pipeline*.

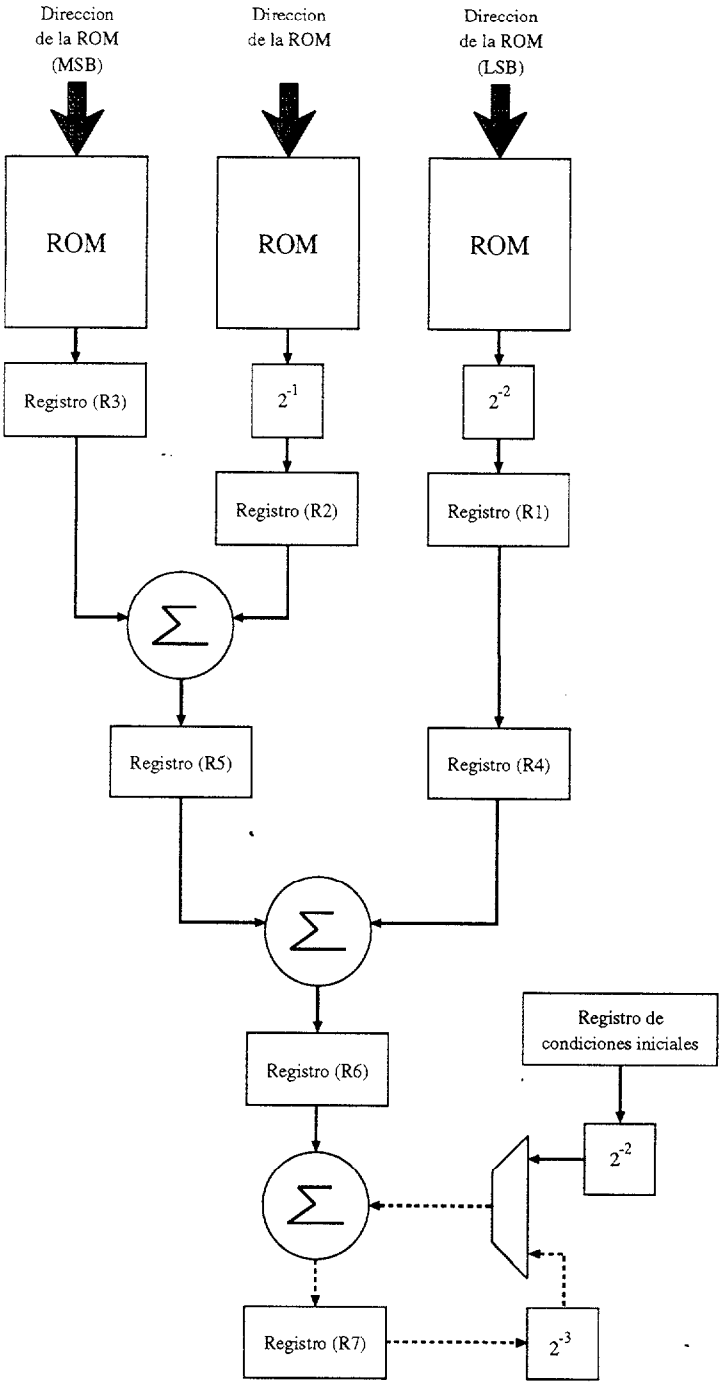


Figura 1: Implementación de la estructura MAC con una partición de 3

T_1	$R3 = [ROM]_3 \; ; \; R2 = \frac{[ROM]_2}{2}$
	$R1 = \frac{[ROM]_1}{2^2}$
T_2	$R5 = R3 + R2 \; ; \; R4 = R1$
	$R3 = [ROM]_6 \; ; \; R2 = \frac{[ROM]_5}{2}$
	$R1 = \frac{[ROM]_4}{2^2}$
T_3	$R6 = R5 + R4$
	$R5 = R3 + R2 \; ; \; R4 = R1$
	$R3 = [ROM]_9 \; ; \; R2 = \frac{[ROM]_8}{2}$
	$R1 = \frac{[ROM]_7}{2^2}$
T_4	$R7 = R6 + \frac{CI}{4} \; ; \; R6 = R5 + R4$
	$R5 = R3 + R2 \; ; \; R4 = R1$
	$R3 = [ROM]_{12} \; ; \; R2 = \frac{[ROM]_{11}}{2}$
	$R1 = \frac{[ROM]_{10}}{2^2}$
$\vdots$	$\vdots$
T_7	$R7 = R6 + \frac{R7}{2^3} \; ; \; R6 = R5 + R4$
	$R5 = R3 + R2 \; ; \; R4 = R1$
	$R3 = [ROM]_9 \; ; \; R2 = \frac{[ROM]_8}{2}$
	$R1 = \frac{[ROM]_7}{2^2}$
T_8	$RAM = R7$
	$R7 = R6 + \frac{CI}{4} \; ; \; R6 = R5 + R4$
	$R5 = R3 + R2 \; ; \; R4 = R1$
	$R3 = [ROM]_{12} \; ; \; R2 = \frac{[ROM]_{11}}{2}$
	$R1 = \frac{[ROM]_{10}}{2^2}$
$\vdots$	$\vdots$
T_{11}	$R7 = R6 + \frac{R7}{2^3} \; ; \; R6 = R5 + R4$
	$R5 = R3 + R2 \; ; \; R4 = R1$
	$R3 = [ROM]_9 \; ; \; R2 = \frac{[ROM]_8}{2}$
	$R1 = \frac{[ROM]_7}{2^2}$
T_{12}	$RAM = R7$
	$R7 = R6 + \frac{CI}{4} \; ; \; R6 = R5 + R4$
	$R5 = R3 + R2 \; ; \; R4 = R1$
	$R3 = [ROM]_{12} \; ; \; R2 = \frac{[ROM]_{11}}{2}$
	$R1 = \frac{[ROM]_{10}}{2^2}$
$\vdots$	$\vdots$

(a)

T_1	$R3 = [ROM]_3 \; ; \; R2 = \frac{[ROM]_2}{2}$
	$R1 = \frac{[ROM]_1}{2^2}$
T_2	$R5 = R3 + R2 \; ; \; R4 = R1$
	$R3 = [ROM]_6 \; ; \; R2 = \frac{[ROM]_5}{2}$
	$R1 = \frac{[ROM]_4}{2^2}$
T_3	$R6 = R5 + R4$
	$R5 = R3 + R2 \; ; \; R4 = R1$
	$R3 = [ROM]_9 \; ; \; R2 = \frac{[ROM]_8}{2}$
	$R1 = \frac{[ROM]_7}{2^2}$
T_4	$R7 = R6 + \frac{CI}{2^2} \; ; \; R6 = R5 + R4$
	$R5 = R3 + R2 \; ; \; R4 = R1$
	$R3 = [ROM]_{12} \; ; \; R2 = \frac{[ROM]_{11}}{2}$
	$R1 = \frac{[ROM]_{10}}{2^2}$
$\vdots$	$\vdots$
T_8	$R7 = R6 + \frac{R7}{2^3} \; ; \; R6 = R5 + R4$
	$R5 = R3 + R2 \; ; \; R4 = R1$
	$R3 = [ROM]_9 \; ; \; R2 = \frac{[ROM]_8}{2}$
	$R1 = \frac{[ROM]_7}{2^2}$
T_9	$SAL = R7$
	$R7 = R6 + \frac{CI}{2^2} \; ; \; R6 = R5 + R4$
	$R5 = R3 + R2 \; ; \; R4 = R1$
	$R3 = [ROM]_{12} \; ; \; R2 = \frac{[ROM]_{11}}{2}$
	$R1 = \frac{[ROM]_{10}}{2^2}$
$\vdots$	$\vdots$
T_{13}	$R7 = R6 + \frac{R7}{2^3} \; ; \; R6 = R5 + R4$
	$R5 = R3 + R2 \; ; \; R4 = R1$
	$R3 = [ROM]_9 \; ; \; R2 = \frac{[ROM]_8}{2}$
	$R1 = \frac{[ROM]_7}{2^2}$
T_{14}	$SAL = R7$
	$R7 = R6 + \frac{CI}{2^2} \; ; \; R6 = R5 + R4$
	$R5 = R3 + R2 \; ; \; R4 = R1$
	$R3 = [ROM]_{12} \; ; \; R2 = \frac{[ROM]_{11}}{2}$
	$R1 = \frac{[ROM]_{10}}{2^2}$
$\vdots$	$\vdots$

(b)

Figura 2: Evolución temporal del *pipeline* con (a) 12 bits y (b) 15 bits

Estructuras MAC con partición 4

En esta estructura (figura 3) para procesar la palabra de 10 bits, es necesario desplazar dicha palabra dos bits a la izquierda, asignando valores nulos a los dos bits de relleno. Es la misma situación que se producía en el caso de una estructura MAC con una partición de 3, cuando la palabra de entrada tenía una longitud de 10 bits.

Para el caso de una palabra de entrada de 13 bits, será necesario desplazar dicha palabra de 3 bits a la izquierda, asignando valores nulos a esos bits, siguiendo el proceso anteriormente visto, como se demuestra a continuación.

$$\begin{aligned} Salida_{MAC} = & \frac{CI + [ROM]_1}{2^{15}} + \frac{[ROM]_2}{2^{14}} + \frac{[ROM]_3}{2^{13}} + \frac{[ROM]_4}{2^{12}} + \frac{[ROM]_5}{2^{11}} + \\ & + \frac{[ROM]_6}{2^{10}} + \frac{[ROM]_7}{2^9} + \frac{[ROM]_8}{2^8} + \frac{[ROM]_9}{2^7} + \frac{[ROM]_{10}}{2^6} + \\ & + \frac{[ROM]_{11}}{2^5} + \frac{[ROM]_{12}}{2^4} + \frac{[ROM]_{13}}{2^3} + \frac{[ROM]_{14}}{2^2} + \\ & + \frac{[ROM]_{15}}{2} + [ROM]_{16} \end{aligned}$$

haciendo $[ROM]_1 = CI$ y $[ROM]_2 = CI$ y $[ROM]_3 = CI$ queda:

$$\begin{aligned} Salida_{MAC} = & \frac{2 \cdot CI}{2^{15}} + \frac{CI}{2^{14}} + \frac{CI}{2^{13}} + \frac{[ROM]_4}{2^{12}} + \frac{[ROM]_5}{2^{11}} + \\ & + \frac{[ROM]_6}{2^{10}} + \frac{[ROM]_7}{2^9} + \frac{[ROM]_8}{2^8} + \frac{[ROM]_9}{2^7} + \frac{[ROM]_{10}}{2^6} + \\ & + \frac{[ROM]_{11}}{2^5} + \frac{[ROM]_{12}}{2^4} + \frac{[ROM]_{13}}{2^3} + \frac{[ROM]_{14}}{2^2} + \\ & + \frac{[ROM]_{15}}{2} + [ROM]_{16} \end{aligned}$$

y simplificando términos:

$$\begin{aligned} Salida_{MAC} = & \frac{CI + [ROM]_4}{2^{12}} + \frac{[ROM]_5}{2^{11}} + \frac{[ROM]_6}{2^{10}} + \frac{[ROM]_7}{2^9} + \\ & + \frac{[ROM]_8}{2^8} + \frac{[ROM]_9}{2^7} + \frac{[ROM]_{10}}{2^6} + \frac{[ROM]_{11}}{2^5} + \\ & + \frac{[ROM]_{12}}{2^4} + \frac{[ROM]_{13}}{2^3} + \frac{[ROM]_{14}}{2^2} \\ & + \frac{[ROM]_{15}}{2} + [ROM]_{16} \end{aligned}$$

que es el resultado esperado.

En la figura 4, se puede observar la evolución del *pipeline*, para la estructura con una partición de 4.

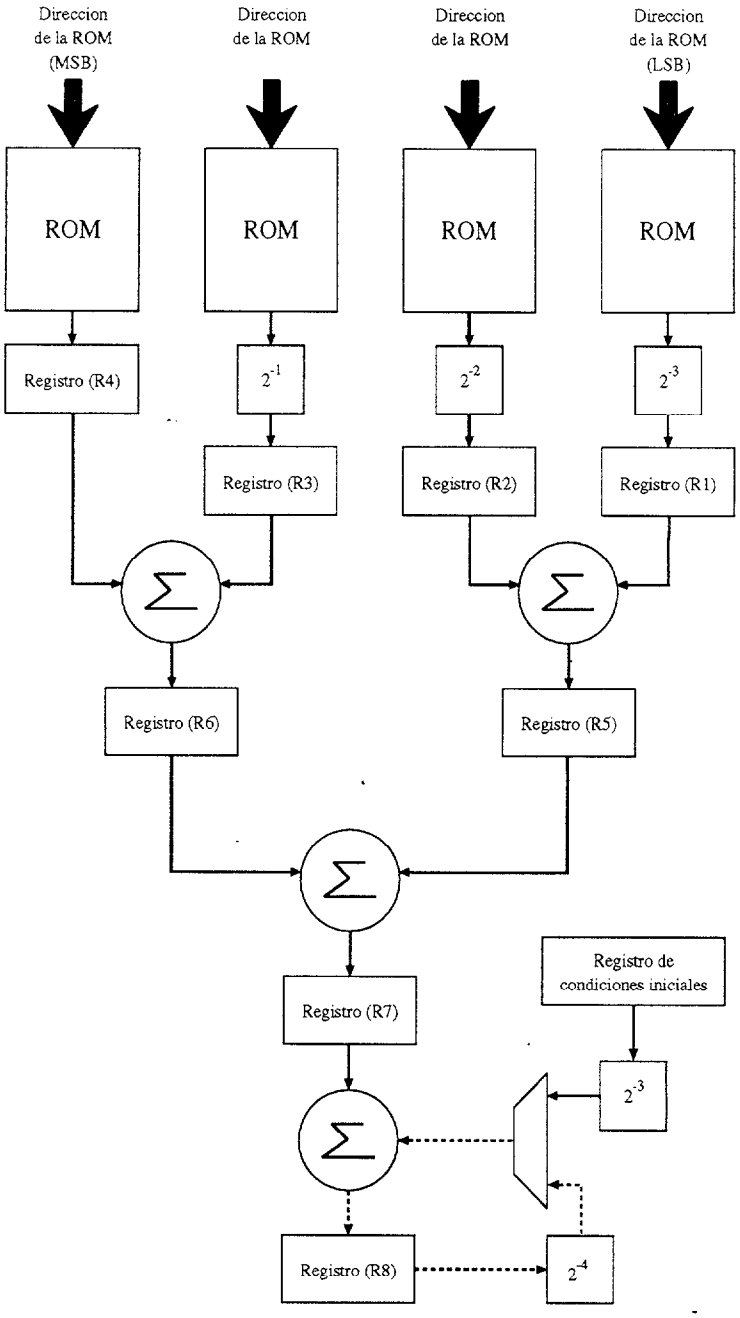


Figura 3: Implementación de la estructura MAC con una partición de 4

T_1	$R4 = [ROM]_4$; $R3 = \frac{[ROM]_3}{2}$
	$R2 = \frac{[ROM]_2}{2^2}$; $R1 = \frac{[ROM]_1}{2^3}$
T_2	$R6 = R4 + R3$; $R5 = R2 + R1$
	$R4 = [ROM]_8$; $R3 = \frac{[ROM]_7}{2}$
	$R2 = \frac{[ROM]_6}{2^2}$; $R1 = \frac{[ROM]_5}{2^3}$
T_3	$R7 = R5 + R6$
	$R6 = R4 + R3$; $R5 = R2 + R1$
	$R4 = [ROM]_{12}$; $R3 = \frac{[ROM]_{11}}{2}$
	$R2 = \frac{[ROM]_{10}}{4}$; $R1 = \frac{[ROM]_9}{2^3}$
T_4	$R8 = R7 + \frac{CI}{2^3}$; $R7 = R5 + R6$
	$R6 = R4 + R3$; $R5 = R2 + R1$
	$R4 = [ROM]_4$; $R3 = \frac{[ROM]_3}{2}$
	$R2 = \frac{[ROM]_2}{2^2}$; $R1 = \frac{[ROM]_1}{2^3}$
T_5	$R8 = R7 + \frac{R8}{2^4}$; $R7 = R5 + R6$
	$R6 = R4 + R3$; $R5 = R2 + R1$
	$R4 = [ROM]_8$; $R3 = \frac{[ROM]_7}{2}$
	$R2 = \frac{[ROM]_6}{2^2}$; $R1 = \frac{[ROM]_5}{2^3}$
T_6	$R8 = R7 + \frac{R8}{2^4}$; $R7 = R5 + R6$
	$R6 = R4 + R3$; $R5 = R2 + R1$
	$R4 = [ROM]_{12}$; $R3 = \frac{[ROM]_{11}}{2}$
	$R2 = \frac{[ROM]_{10}}{2^2}$; $R1 = \frac{[ROM]_9}{2^3}$
T_7	$RAM = R8$
	$R8 = R7 + \frac{CI}{2^3}$; $R7 = R5 + R6$
	$R6 = R4 + R3$; $R5 = R2 + R1$
	$R4 = [ROM]_4$; $R3 = \frac{[ROM]_3}{2}$
	$R2 = \frac{[ROM]_2}{2^2}$; $R1 = \frac{[ROM]_1}{2^3}$
$\vdots$	$\vdots$
T_9	$R8 = R7 + \frac{R8}{2^4}$; $R7 = R5 + R6$
	$R6 = R4 + R3$; $R5 = R2 + R1$
	$R4 = [ROM]_{12}$; $R3 = \frac{[ROM]_{11}}{2}$
	$R2 = \frac{[ROM]_{10}}{2^2}$; $R1 = \frac{[ROM]_9}{2^3}$
T_{10}	$RAM = R8$
	$R8 = R7 + \frac{CI}{2^3}$; $R7 = R5 + R6$
	$R6 = R4 + R3$; $R5 = R2 + R1$
	$R4 = [ROM]_4$; $R3 = \frac{[ROM]_3}{2}$
	$R2 = \frac{[ROM]_2}{2^2}$; $R1 = \frac{[ROM]_1}{2^3}$
$\vdots$	$\vdots$
$\vdots$	$\vdots$

(a)

T_1	$R4 = [ROM]_4$; $R3 = \frac{[ROM]_3}{2}$
	$R2 = \frac{[ROM]_2}{2^2}$; $R1 = \frac{[ROM]_1}{2^3}$
T_2	$R6 = R4 + R3$; $R5 = R2 + R1$
	$R4 = [ROM]_8$; $R3 = \frac{[ROM]_7}{2}$
	$R2 = \frac{[ROM]_6}{2^2}$; $R1 = \frac{[ROM]_5}{2^3}$
T_3	$R7 = R5 + R6$
	$R6 = R4 + R3$; $R5 = R2 + R1$
	$R4 = [ROM]_{12}$; $R3 = \frac{[ROM]_{11}}{2}$
	$R2 = \frac{[ROM]_{10}}{2^2}$; $R1 = \frac{[ROM]_9}{2^3}$
T_4	$R8 = R7 + \frac{CI}{2^3}$; $R7 = R5 + R6$
	$R6 = R4 + R3$; $R5 = R2 + R1$
	$R4 = [ROM]_{16}$; $R3 = \frac{[ROM]_{15}}{2}$
	$R2 = \frac{[ROM]_{14}}{2^2}$; $R1 = \frac{[ROM]_{13}}{2^3}$
T_5	$R8 = R7 + \frac{R8}{2^4}$; $R7 = R5 + R6$
	$R6 = R4 + R3$; $R5 = R2 + R1$
	$R4 = [ROM]_4$; $R3 = \frac{[ROM]_3}{2}$
	$R2 = \frac{[ROM]_2}{2^2}$; $R1 = \frac{[ROM]_1}{2^3}$
$\vdots$	$\vdots$
T_7	$R8 = R7 + \frac{R8}{2^4}$; $R7 = R5 + R6$
	$R6 = R4 + R3$; $R5 = R2 + R1$
	$R4 = [ROM]_{12}$; $R3 = \frac{[ROM]_{11}}{2}$
	$R2 = \frac{[ROM]_{10}}{2^2}$; $R1 = \frac{[ROM]_9}{2^3}$
T_8	$SAL = R8$
	$R8 = R7 + \frac{CI}{2^3}$; $R7 = R5 + R6$
	$R6 = R4 + R3$; $R5 = R2 + R1$
	$R4 = [ROM]_{16}$; $R3 = \frac{[ROM]_{15}}{2}$
	$R2 = \frac{[ROM]_{14}}{2^2}$; $R1 = \frac{[ROM]_{13}}{2^3}$
$\vdots$	$\vdots$
T_{11}	$R8 = R7 + \frac{R8}{2^4}$; $R7 = R5 + R6$
	$R6 = R4 + R3$; $R5 = R2 + R1$
	$R4 = [ROM]_{12}$; $R3 = \frac{[ROM]_{11}}{2}$
	$R2 = \frac{[ROM]_{10}}{2^2}$; $R1 = \frac{[ROM]_9}{2^3}$
T_{12}	$SAL = R8$
	$R8 = R7 + \frac{CI}{2^3}$; $R7 = R5 + R6$
	$R6 = R4 + R3$; $R5 = R2 + R1$
	$R4 = [ROM]_{16}$; $R3 = \frac{[ROM]_{15}}{2}$
	$R2 = \frac{[ROM]_{14}}{2^2}$; $R1 = \frac{[ROM]_{13}}{2^3}$
$\vdots$	$\vdots$

(b)

Figura 4: Evolución temporal del *pipeline* con (a) 12 bits y (b) 16 bits

Estructuras MAC con partición 5

El diagrama de bloques de la estructura MAC con una partición de 5 se muestra en la figura 5. En esta figura es necesario desplazar dos bits a la izquierda la palabra de 13 bits, ya que esta estructura sólo puede procesar palabras con un número de bits múltiplo de 5.

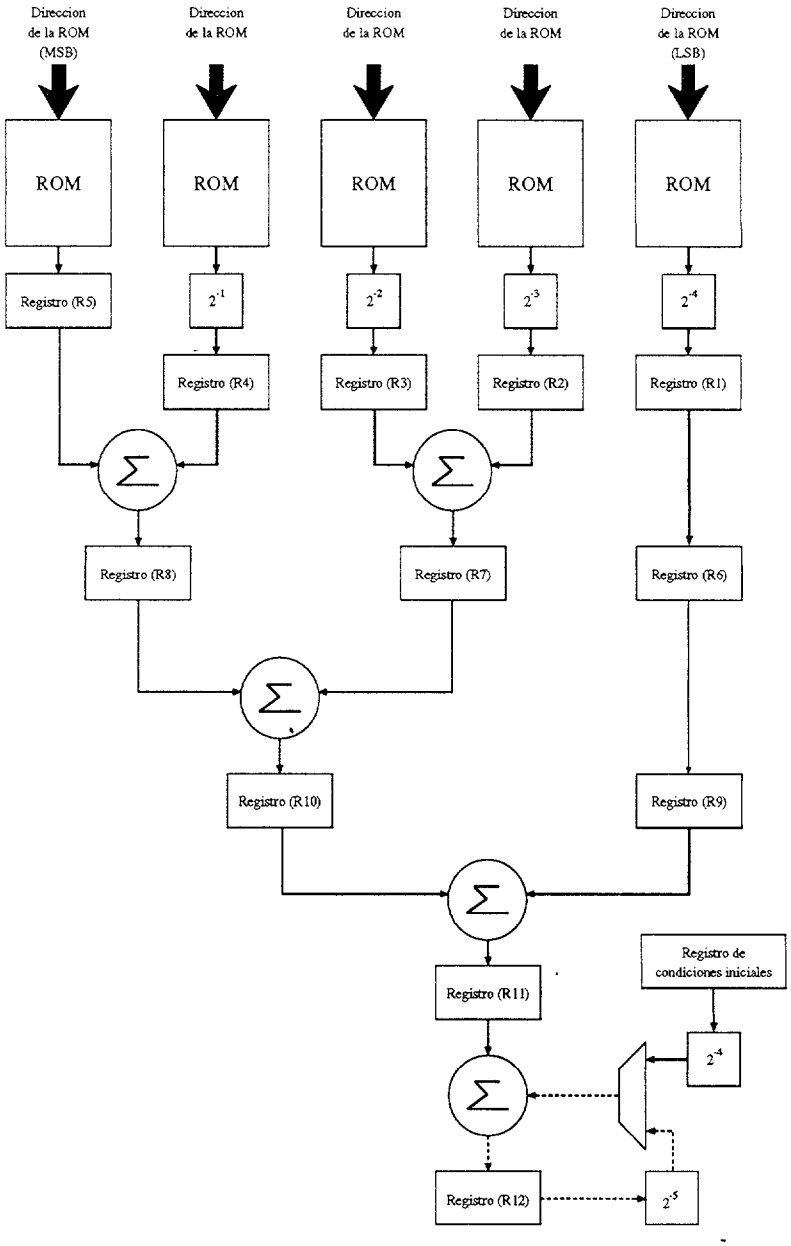


Figura 5: Implementación de la estructura MAC con una partición de 5

En la figura 6 se muestra el flujo de datos a través del *pipeline*.

T_1	$R5 = [ROM]_5$; $R4 = \frac{[ROM]_4}{2}$; $R3 = \frac{[ROM]_3}{2^2}$
	$R2 = \frac{[ROM]_2}{2^3}$; $R1 = \frac{[ROM]_1}{2^4}$
T_2	$R8 = R5 + R4$; $R7 = R2 + R3$; $R6 = R1$
	$R5 = [ROM]_{10}$; $R4 = \frac{[ROM]_9}{2}$; $R3 = \frac{[ROM]_8}{2^2}$
	$R2 = \frac{[ROM]_7}{2^3}$; $R1 = \frac{[ROM]_6}{2^4}$
T_3	$R10 = R8 + R7$; $R9 = R6$
	$R8 = R5 + R4$; $R7 = R2 + R3$; $R6 = R1$
	$R5 = [ROM]_5$; $R4 = \frac{[ROM]_4}{2}$; $R3 = \frac{[ROM]_3}{2^2}$
	$R2 = \frac{[ROM]_2}{2^3}$; $R1 = \frac{[ROM]_1}{2^4}$
$\vdots$	$\vdots$
T_5	$R12 = R11 + \frac{CI}{2^4}$; $R11 = R10 + R9$
	$R10 = R8 + R7$; $R9 = R6$
	$R8 = R5 + R4$; $R7 = R2 + R3$; $R6 = R1$
	$R5 = [ROM]_5$; $R4 = \frac{[ROM]_4}{2}$; $R3 = \frac{[ROM]_3}{2^2}$
	$R2 = \frac{[ROM]_2}{2^3}$; $R1 = \frac{[ROM]_1}{2^4}$
T_6	$R12 = R11 + \frac{R12}{2^3}$; $R11 = R10 + R9$
	$R10 = R8 + R7$; $R9 = R6$
	$R8 = R5 + R4$; $R7 = R2 + R3$; $R6 = R1$
	$R5 = [ROM]_{10}$; $R4 = \frac{[ROM]_9}{2}$; $R3 = \frac{[ROM]_8}{2^2}$
	$R2 = \frac{[ROM]_7}{2^3}$; $R1 = \frac{[ROM]_6}{2^4}$
T_7	$RAM = R12$
	$R12 = R11 + \frac{CI}{2^4}$; $R11 = R10 + R9$
	$R10 = R8 + R7$; $R9 = R6$
	$R8 = R5 + R4$; $R7 = R2 + R3$; $R6 = R1$
	$R5 = [ROM]_5$; $R4 = \frac{[ROM]_4}{2}$; $R3 = \frac{[ROM]_3}{2^2}$
	$R2 = \frac{[ROM]_2}{2^3}$; $R1 = \frac{[ROM]_1}{2^4}$
$\vdots$	$\vdots$
T_9	$RAM = R12$
	$R12 = R11 + \frac{CI}{2^4}$; $R11 = R10 + R9$
	$R10 = R8 + R7$; $R9 = R6$
	$R8 = R5 + R4$; $R7 = R2 + R3$; $R6 = R1$
	$R5 = [ROM]_5$; $R4 = \frac{[ROM]_4}{2}$; $R3 = \frac{[ROM]_3}{2^2}$
	$R2 = \frac{[ROM]_2}{2^3}$; $R1 = \frac{[ROM]_1}{2^4}$
$\vdots$	$\vdots$
$\vdots$	$\vdots$

(a)

T_1	$R5 = [ROM]_5$; $R4 = \frac{[ROM]_4}{2}$; $R3 = \frac{[ROM]_3}{2^2}$
	$R2 = \frac{[ROM]_2}{2^3}$; $R1 = \frac{[ROM]_1}{2^4}$
T_2	$R8 = R5 + R4$; $R7 = R2 + R3$; $R6 = R1$
	$R5 = [ROM]_{10}$; $R4 = \frac{[ROM]_9}{2}$; $R3 = \frac{[ROM]_8}{2^2}$
	$R2 = \frac{[ROM]_7}{2^3}$; $R1 = \frac{[ROM]_6}{2^4}$
T_3	$R10 = R8 + R7$; $R9 = R6$
	$R8 = R5 + R4$; $R7 = R2 + R3$; $R6 = R1$
	$R5 = [ROM]_{15}$; $R4 = \frac{[ROM]_{14}}{2}$; $R3 = \frac{[ROM]_{13}}{2^2}$
	$R2 = \frac{[ROM]_{12}}{2^3}$; $R1 = \frac{[ROM]_{11}}{2^4}$
$\vdots$	$\vdots$
T_5	$R12 = R11 + \frac{CI}{2^4}$; $R11 = R10 + R9$
	$R10 = R8 + R7$; $R9 = R6$
	$R8 = R5 + R4$; $R7 = R2 + R3$; $R6 = R1$
	$R5 = [ROM]_{10}$; $R4 = \frac{[ROM]_9}{2}$; $R3 = \frac{[ROM]_8}{2^2}$
	$R2 = \frac{[ROM]_7}{2^3}$; $R1 = \frac{[ROM]_6}{2^4}$
T_6	$R12 = R11 + \frac{R12}{2^3}$; $R11 = R10 + R9$
	$R10 = R8 + R7$; $R9 = R6$
	$R8 = R5 + R4$; $R7 = R2 + R3$; $R6 = R1$
	$R5 = [ROM]_{15}$; $R4 = \frac{[ROM]_{14}}{2}$; $R3 = \frac{[ROM]_{13}}{2^2}$
	$R2 = \frac{[ROM]_{12}}{2^3}$; $R1 = \frac{[ROM]_{11}}{2^4}$
$\vdots$	$\vdots$
T_8	$SAL = R12$
	$R12 = R11 + \frac{CI}{2^4}$; $R11 = R10 + R9$
	$R10 = R8 + R7$; $R9 = R6$
	$R8 = R5 + R4$; $R7 = R2 + R3$; $R6 = R1$
	$R5 = [ROM]_{10}$; $R4 = \frac{[ROM]_9}{2}$; $R3 = \frac{[ROM]_8}{2^2}$
	$R2 = \frac{[ROM]_7}{2^3}$; $R1 = \frac{[ROM]_6}{2^4}$
$\vdots$	$\vdots$
T_{11}	$SAL = R12$
	$R12 = R11 + \frac{CI}{2^4}$; $R11 = R10 + R9$
	$R10 = R8 + R7$; $R9 = R6$
	$R8 = R5 + R4$; $R7 = R2 + R3$; $R6 = R1$
	$R5 = [ROM]_{10}$; $R4 = \frac{[ROM]_9}{2}$; $R3 = \frac{[ROM]_8}{2^2}$
	$R2 = \frac{[ROM]_7}{2^3}$; $R1 = \frac{[ROM]_6}{2^4}$
$\vdots$	$\vdots$

(b)

Figura 6: Evolución temporal del *pipeline* con (a) 10 bits y (b) 15 bits

Anexo B

Simulación de las unidades básicas:

- Sumadores
- ROM
- DRAM

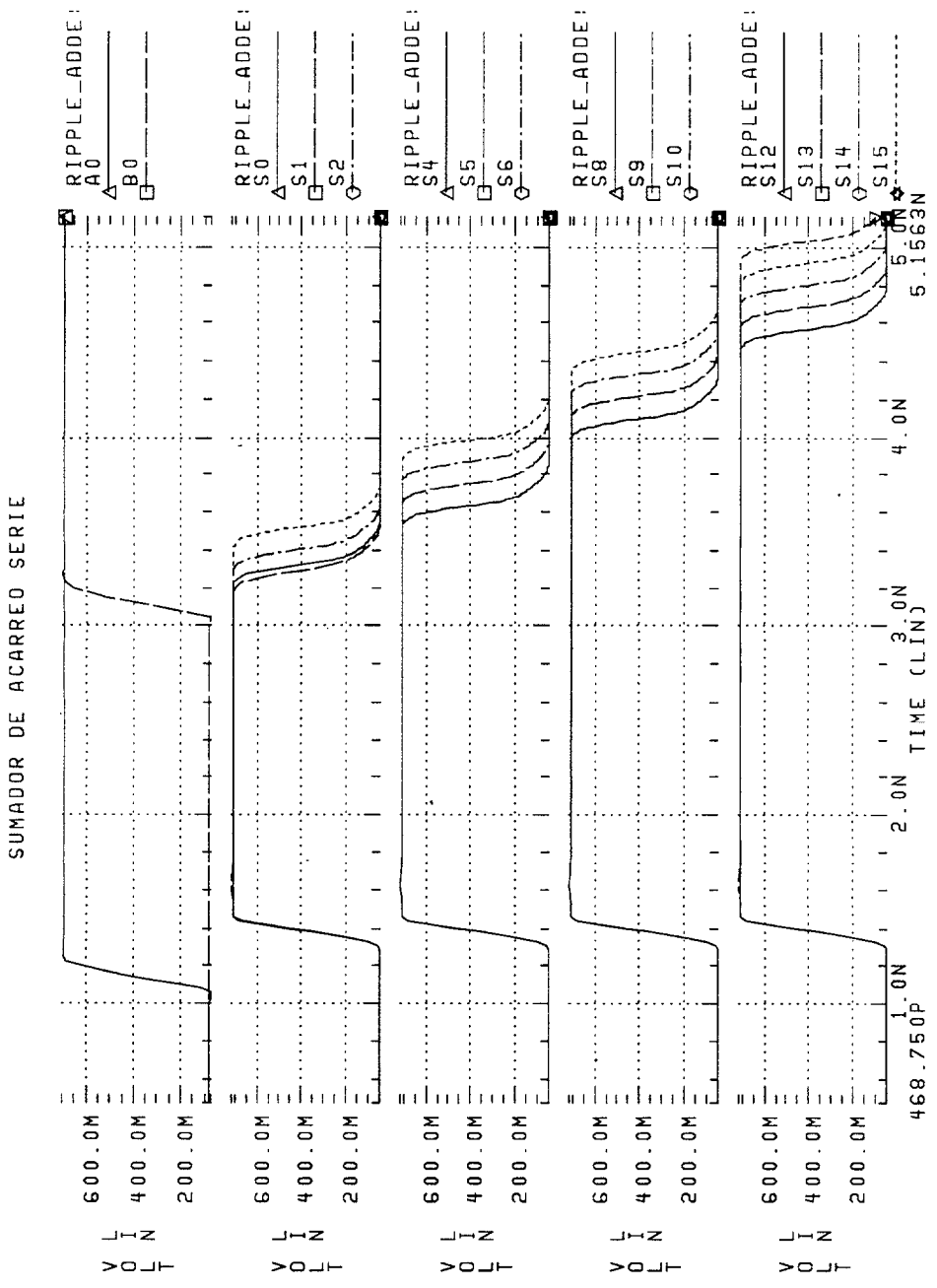


Figura 7: Sumador de acarreo serie de 17 bits

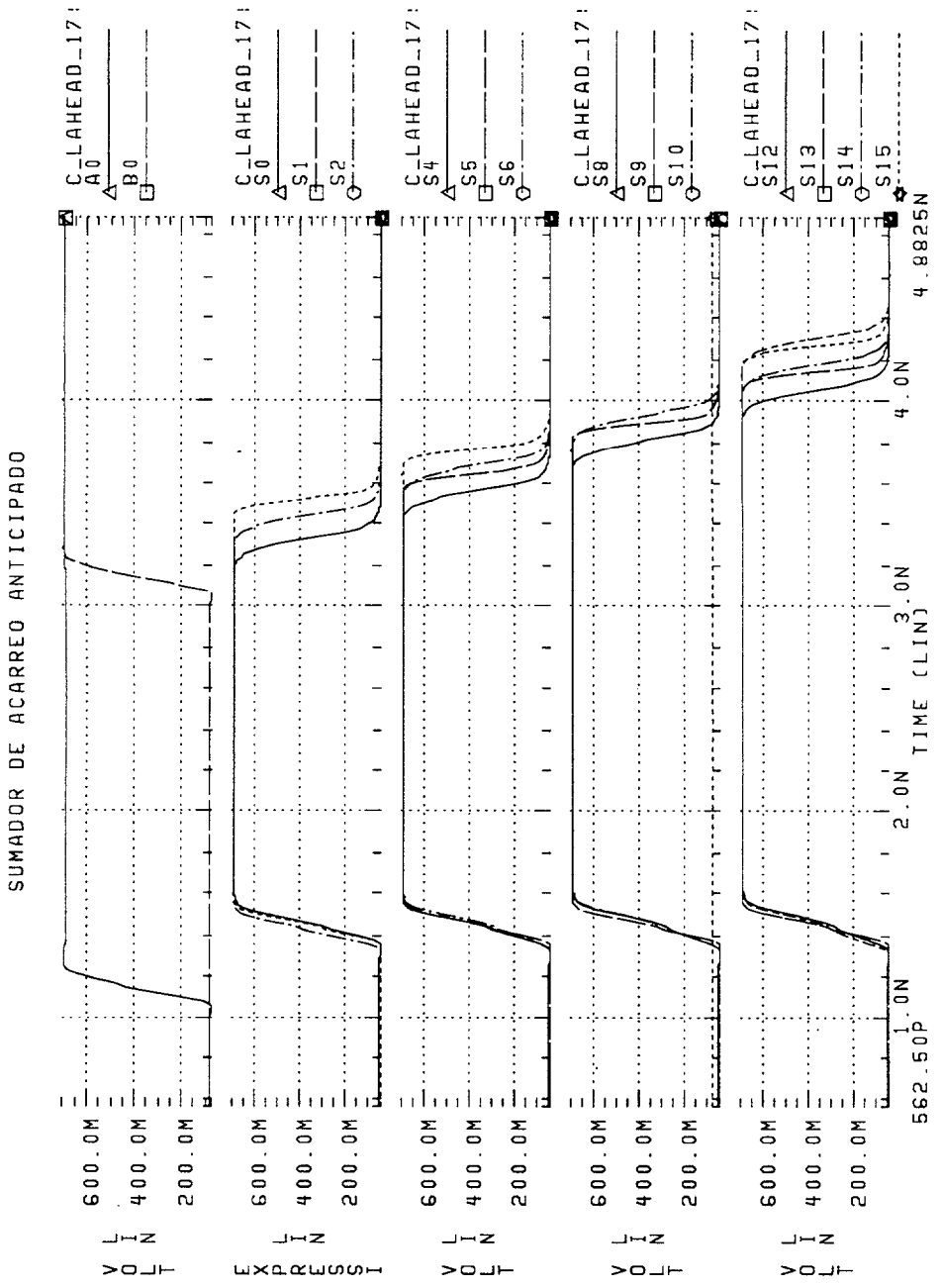


Figura 8: Sumador de acarreo anticipado de 17 bits

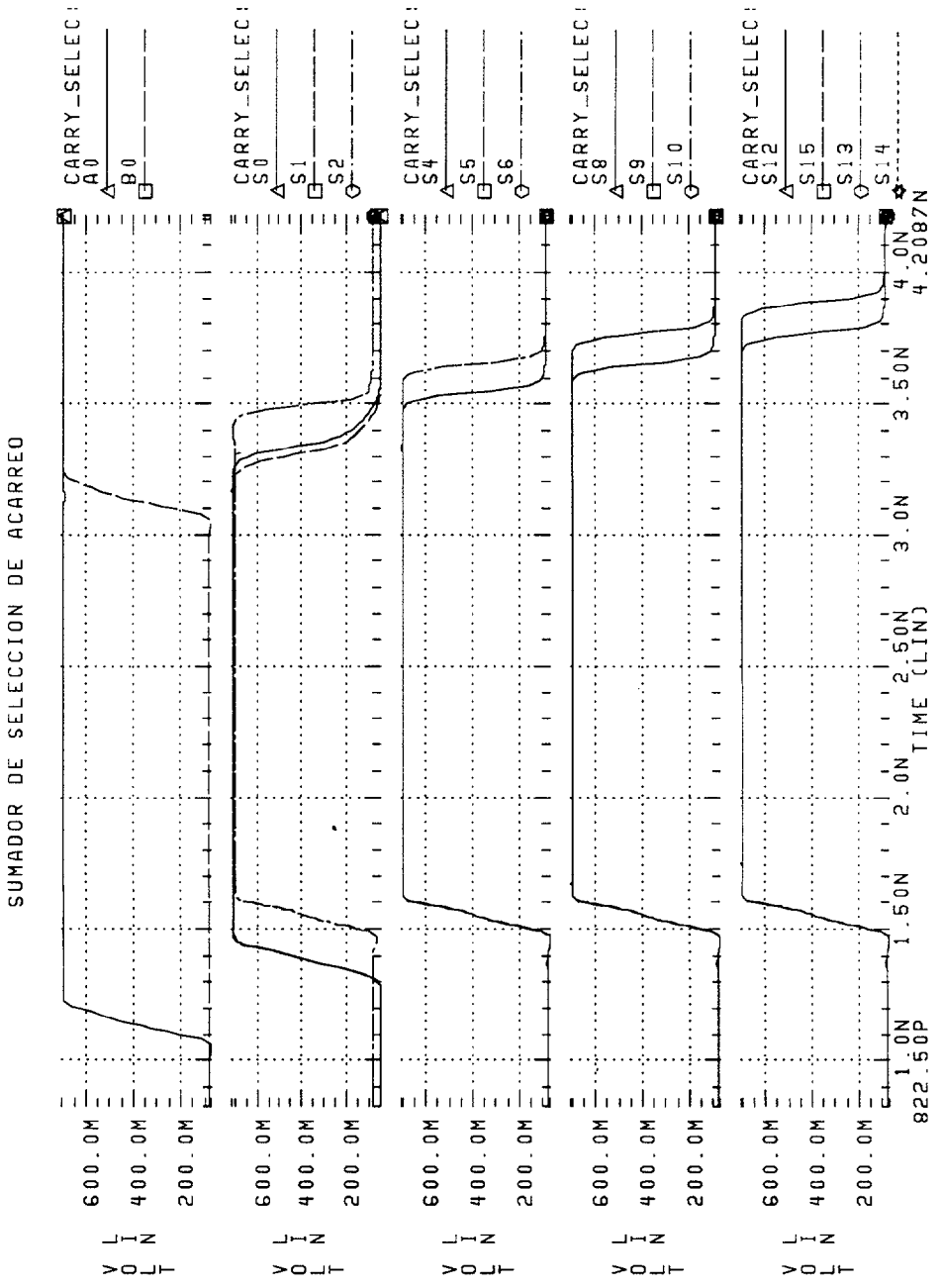


Figura 9: Sumador de selección de acarreo de 17 bits

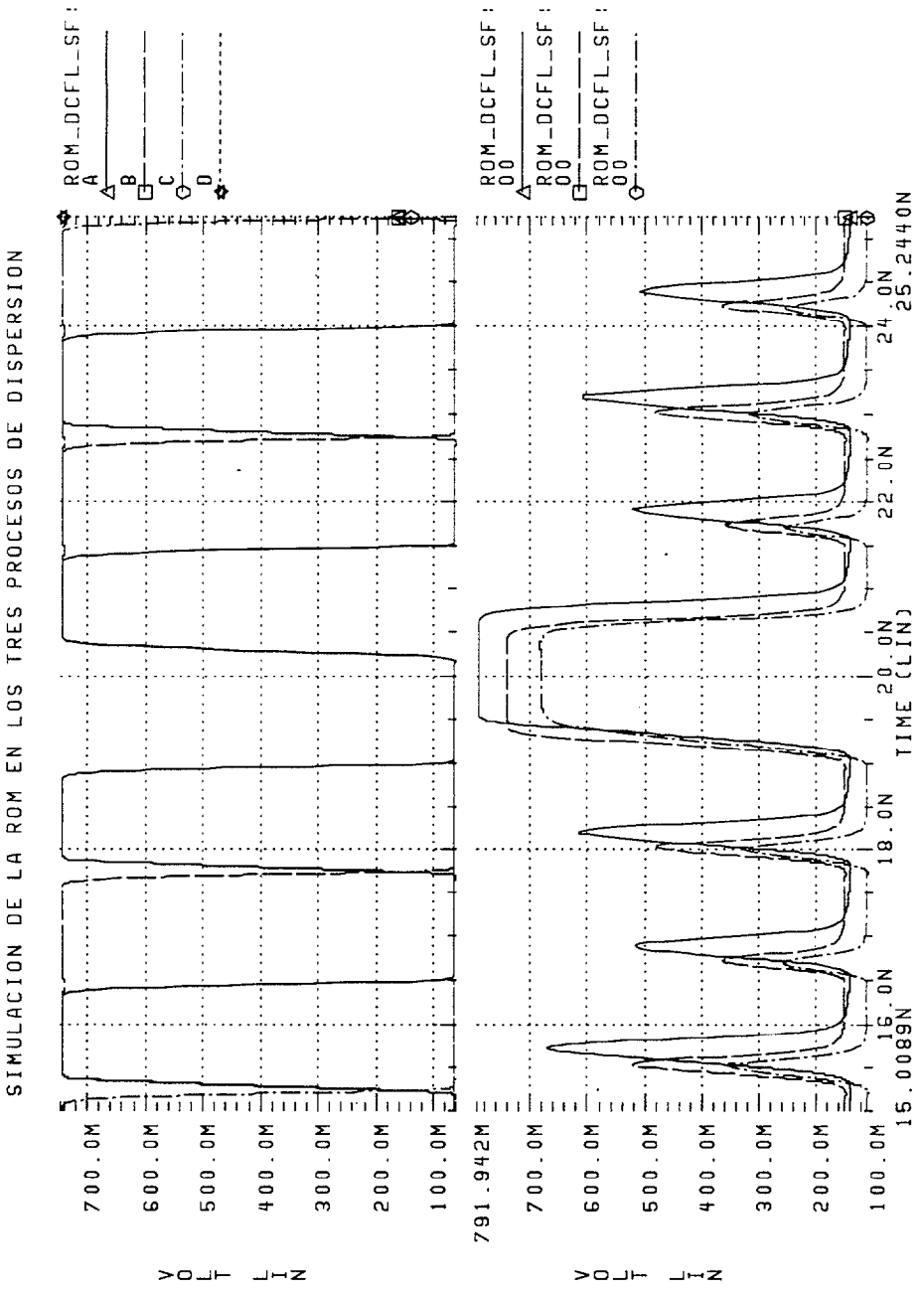


Figura 10: Simulación de la ROM en los tres procesos de dispersión y a temperaturas extremas

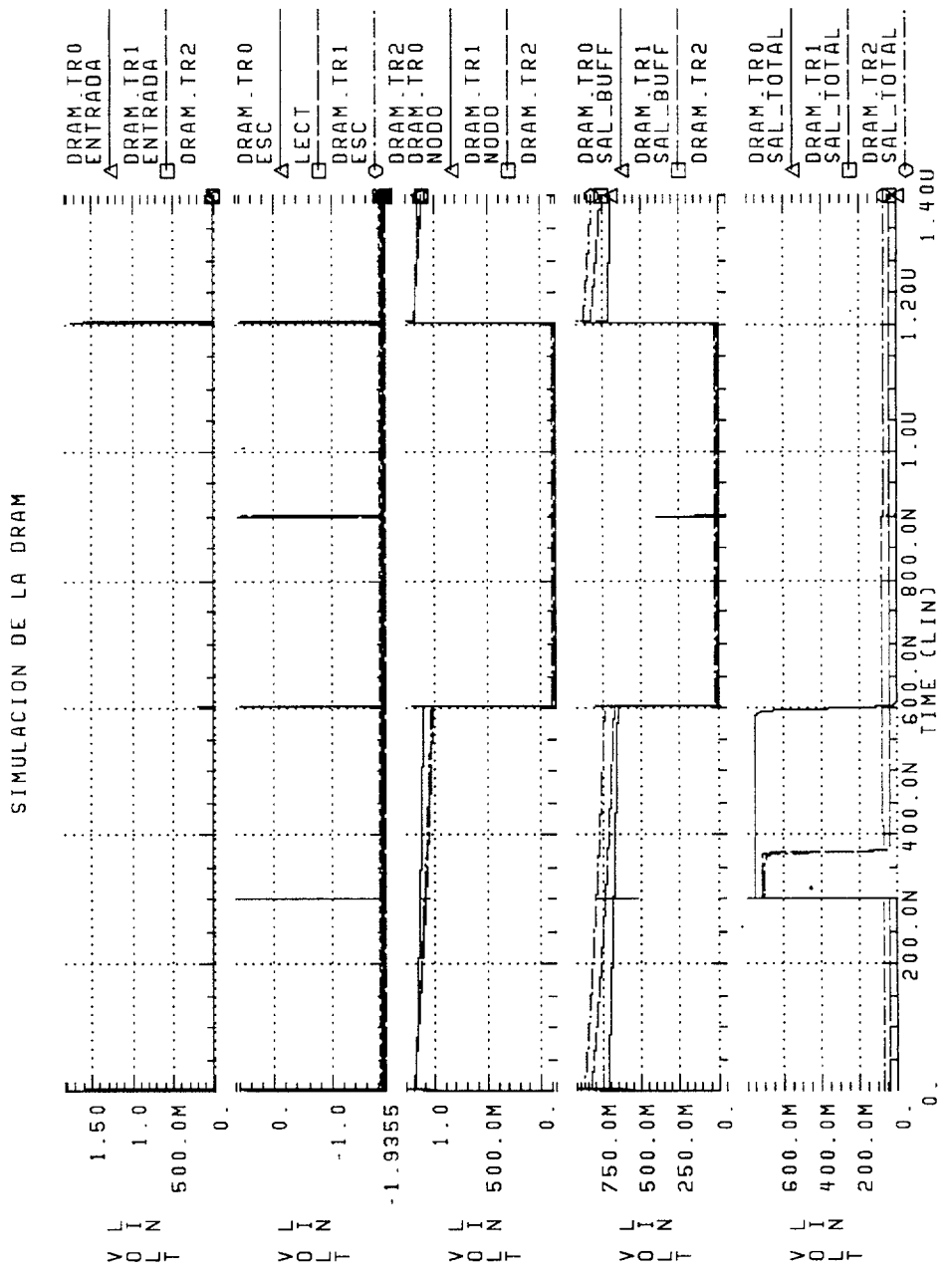


Figura 11: Simulación de la DRAM en los tres procesos de dispersión y a temperaturas extremas

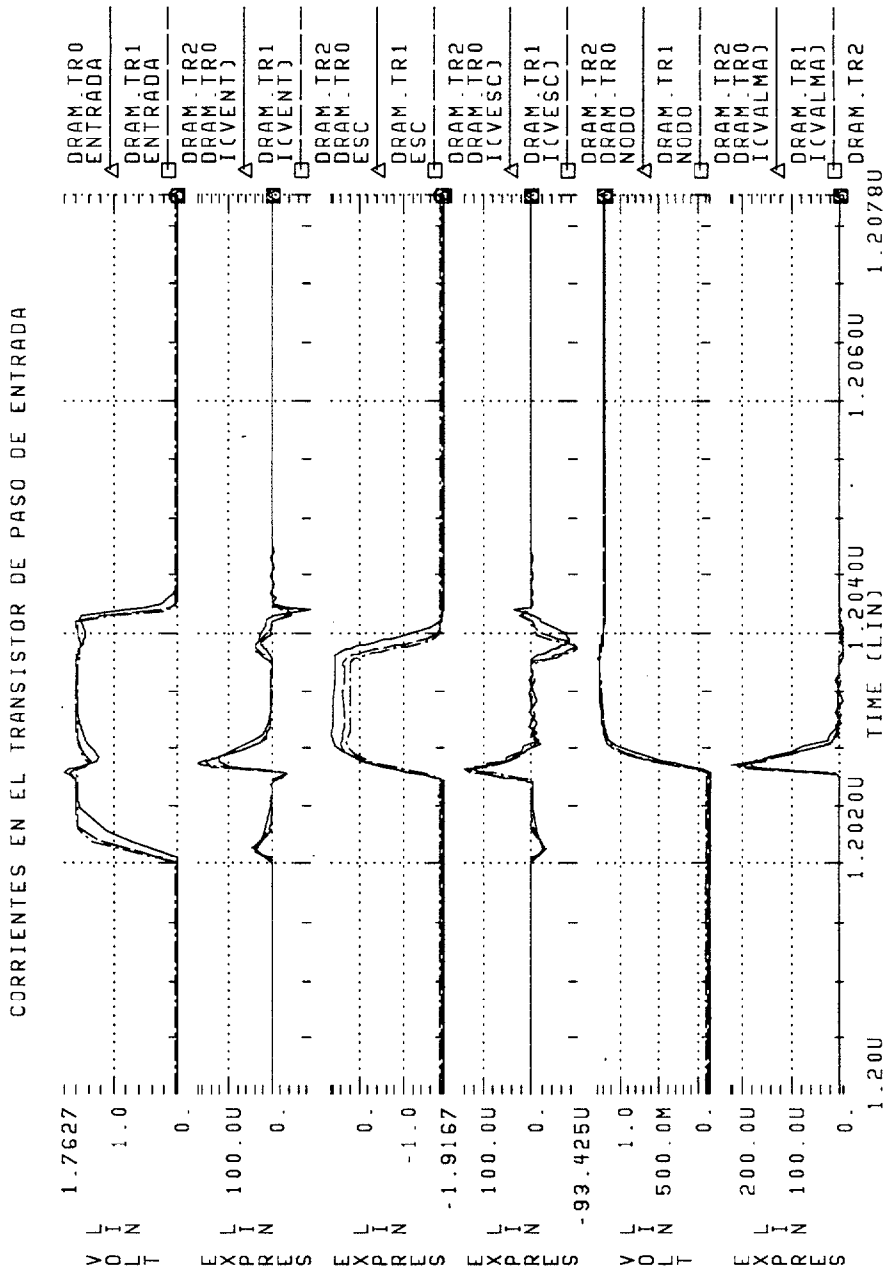


Figura 12: Corrientes en el transistor de paso de entrada de la célula de la DRAM

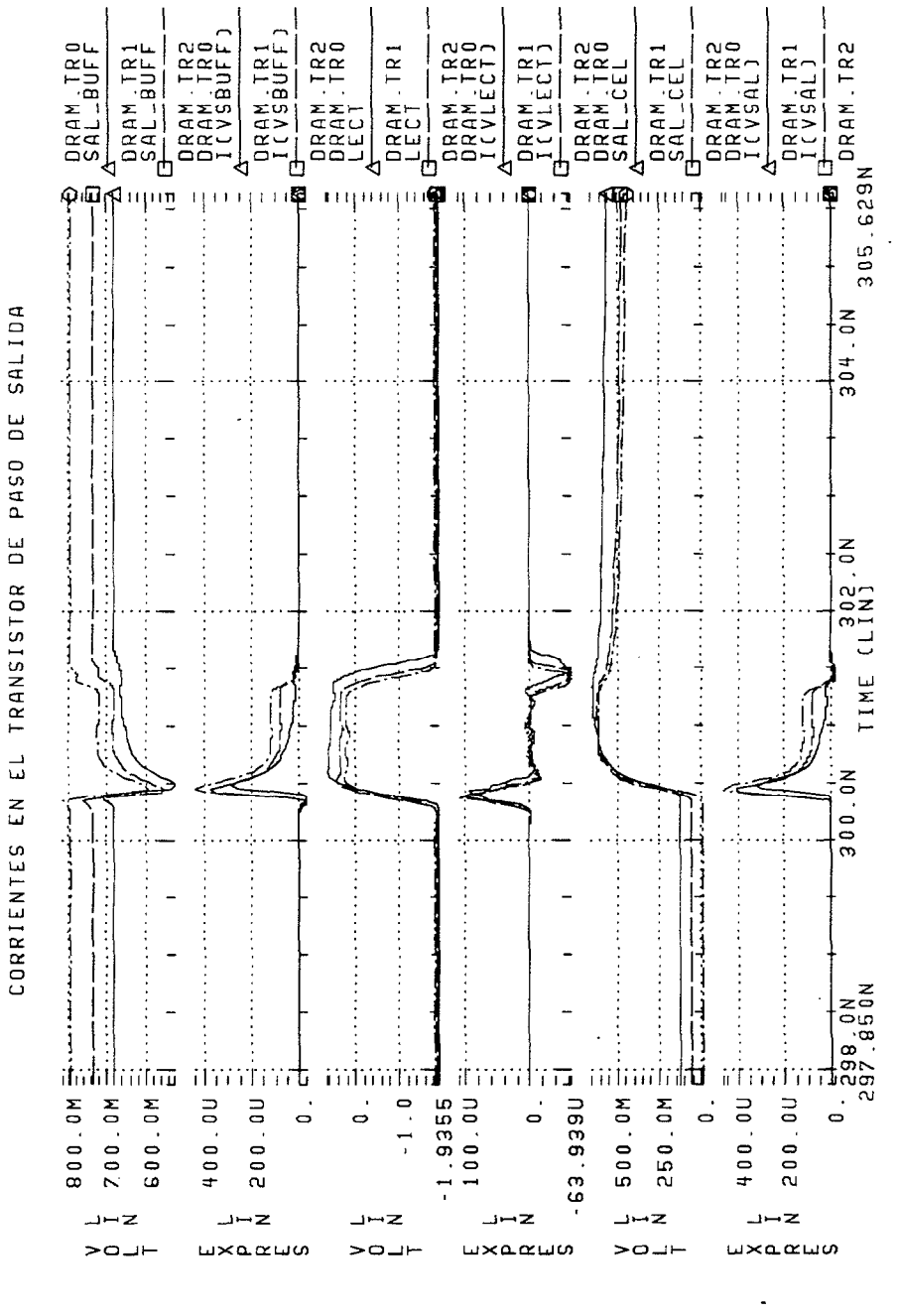


Figura 13: Corrientes en el transistor de paso de salida de la célula de la DRAM

Anexo C

Simulación de las unidades de control:

- Circuito GaAsDCT11
- Circuito GaAsDCT34

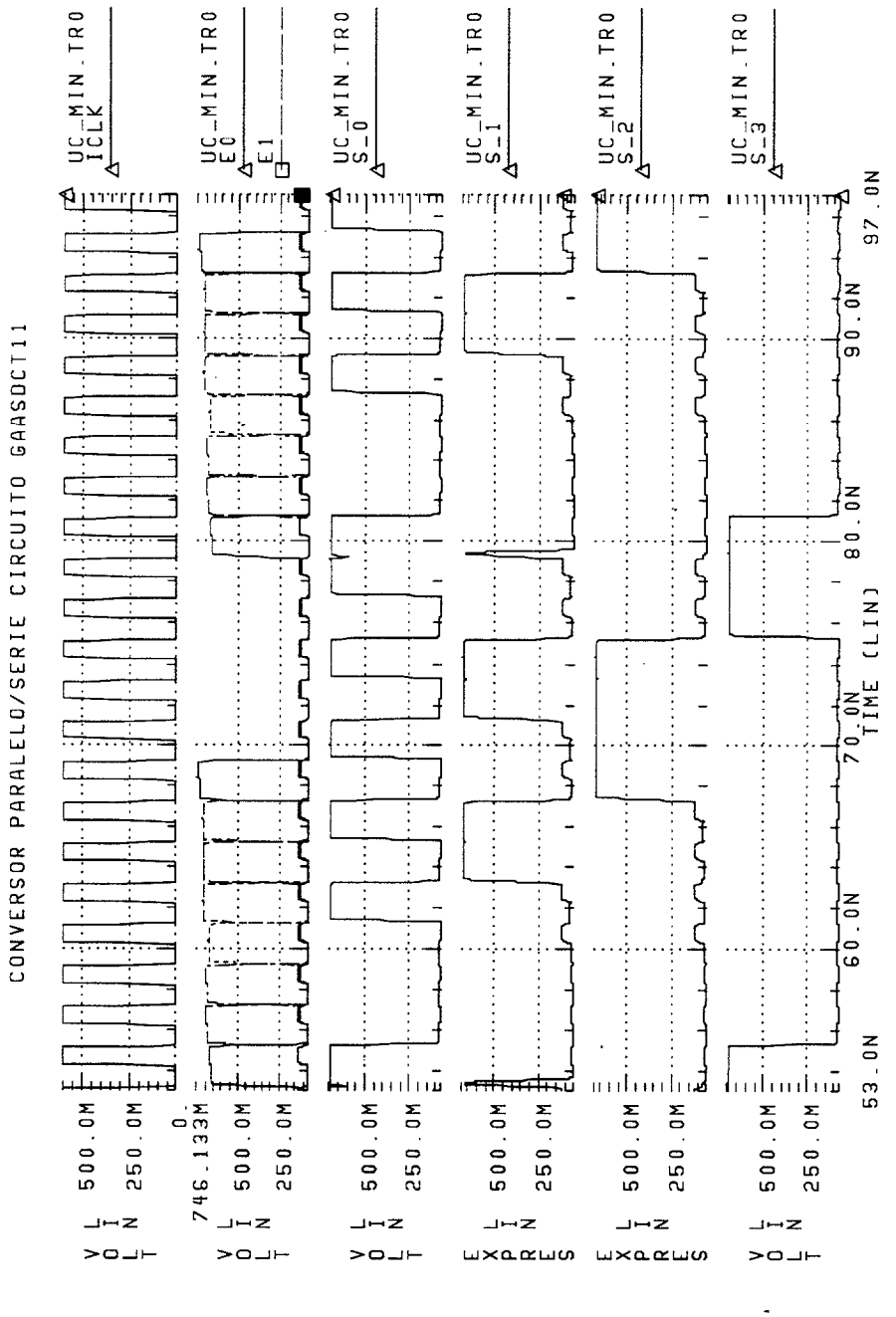


Figura 14: Señales de control del conversor paralelo/serie (GaAsDCT11)

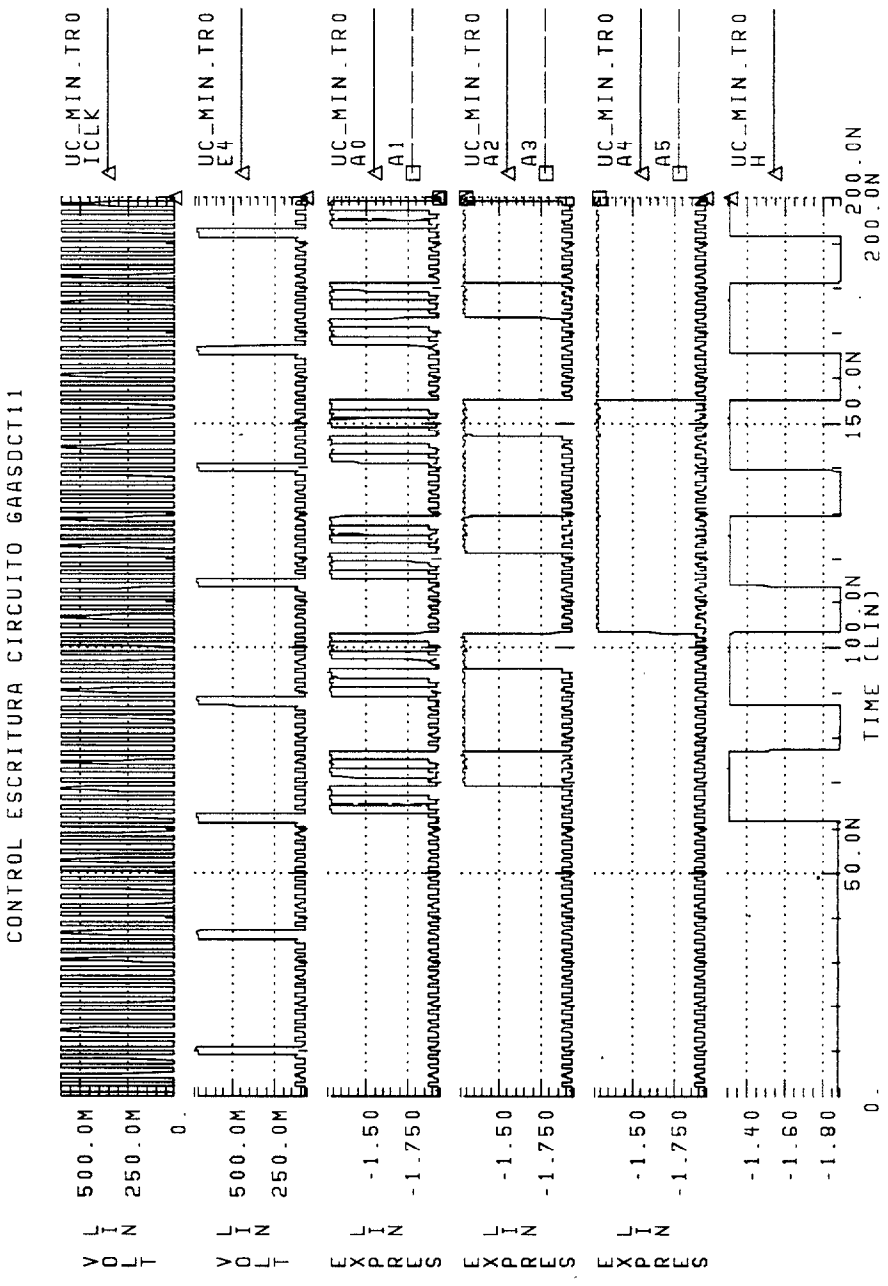


Figura 15: Señales de control para la escritura (GaAsDCT11)

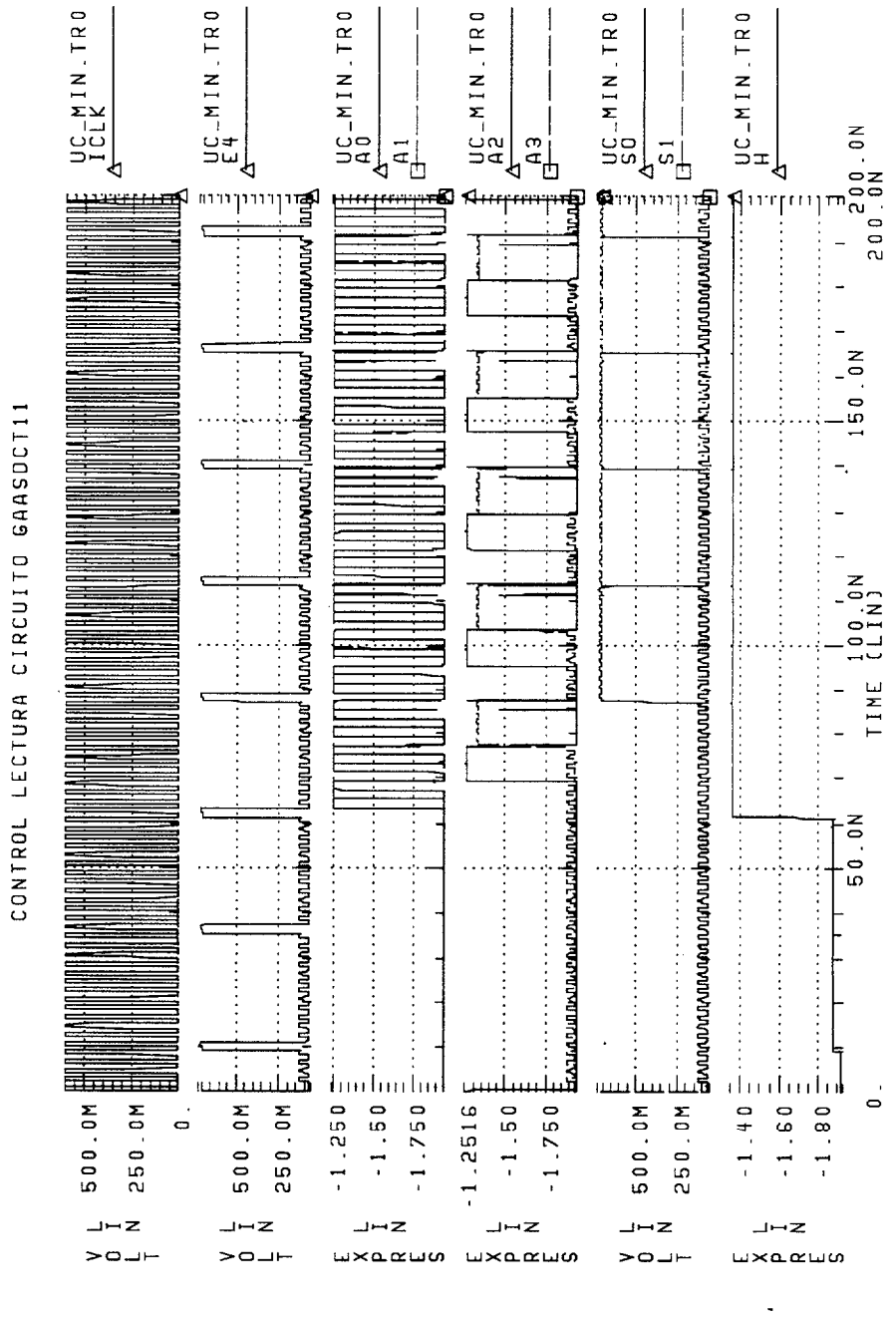


Figura 16: Señales de control para la lectura (GaAsDCT11)

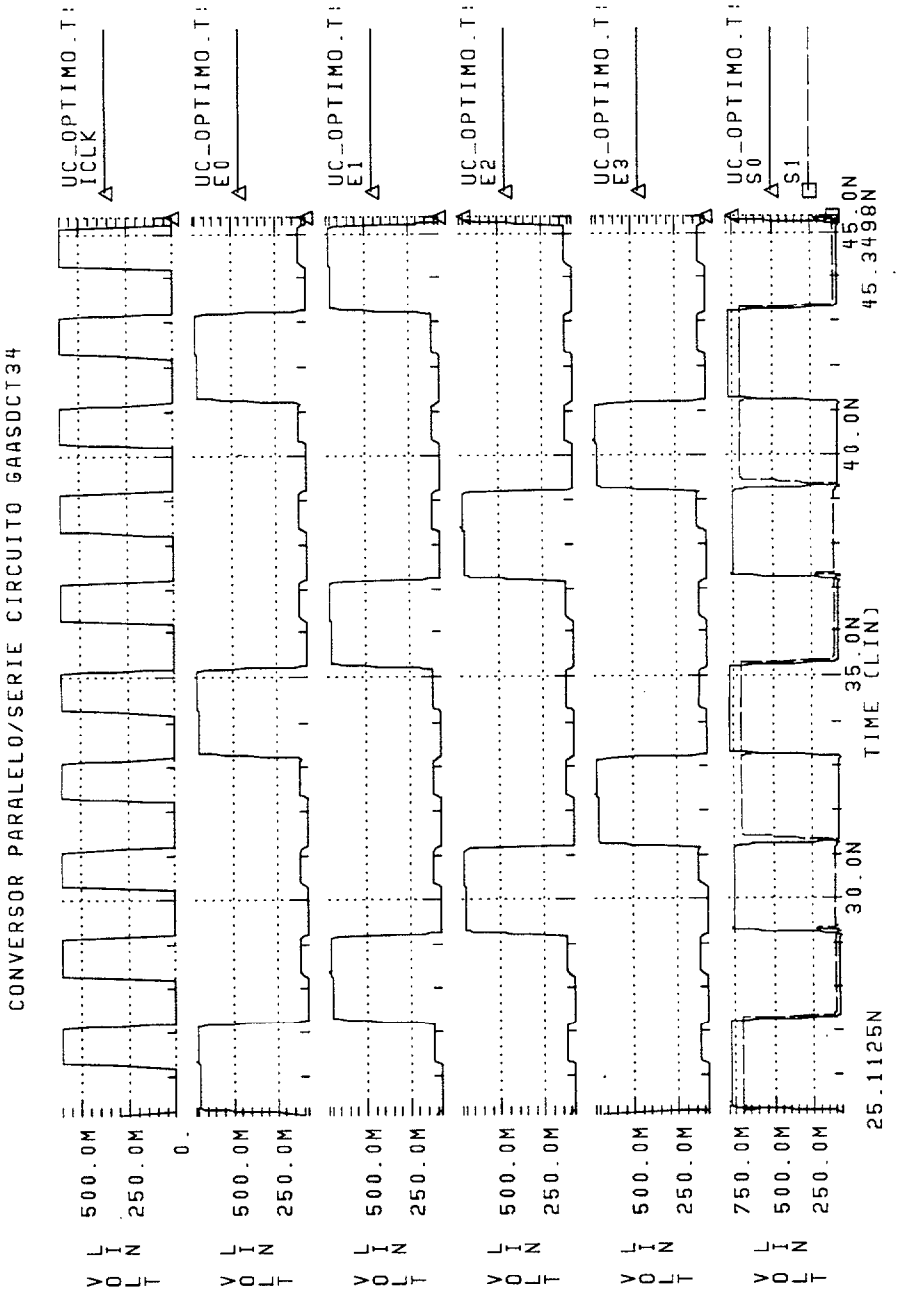


Figura 17: Señales de control del conversor paralelo/serie (GaAsDCT34)

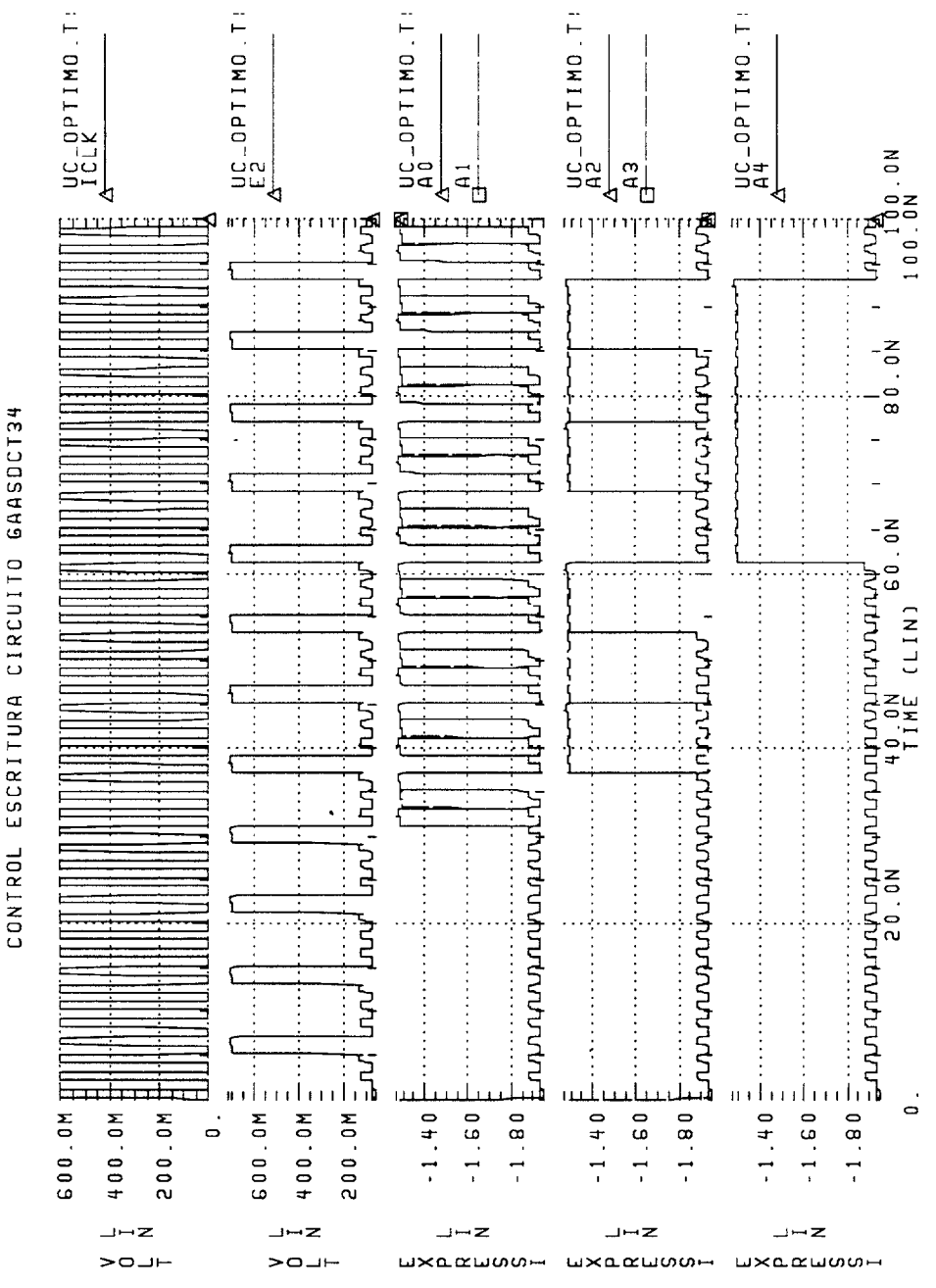


Figura 18: Señales de control para la escritura (GaAsDCT34)

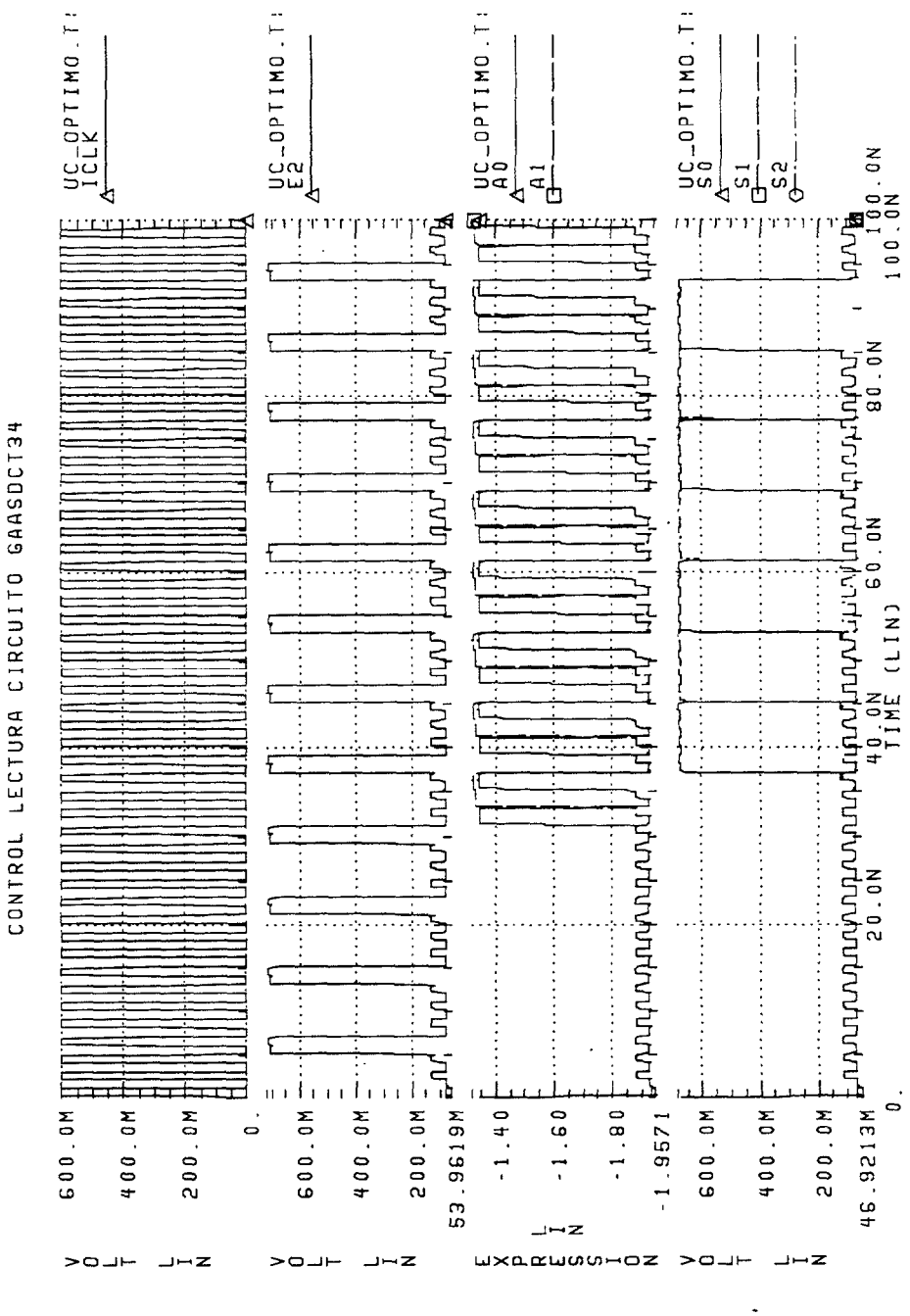


Figura 19: Señales de control para la lectura (GaAsDCT34)

Presupuesto

Capítulo 1

Costes de amortización de equipos y herramientas

Los equipos y paquetes *software* utilizados para la elaboración del proyecto conllevan unos costes de amortización y mantenimiento, que deben ser incluidos en el presupuesto en función del periodo de uso.

Descripción	*Periodo de uso*	*Valor/Año*	*Total*
Estación de trabajo Sun SPARC modelo SPARCstation 10. Con periodo de amortización de 3 años			
Amortización	10 meses	870.000 pts.	725.000 pts.
Mantenimiento	10 meses	262.000 pts.	218.333 pts.
Servidor para simulación Sun SPARC modelo SPARCstation 20. Con periodo de amortización 3 años			
Amortización	10 meses	843.333 pts.	702.777 pts.
Mantenimiento	10 meses	262.000 pts.	218.333 pts.
HARDWARE		TOTAL	1.864.443 pts.

Descripción	*Periodo de uso*	*Valor/Año*	*Total*
Sistema operativo SunOS Release 4.1.3, sistema de ventanas OpenWindows y Librerías X11.	10 meses	43.333 pts.	36.110 pts.
Entorno de diseño y simulación CADENCE Designs Framework II			
Amortización	10 meses	240.000 pts.	200.000 pts.
Mantenimiento	10 meses	88.000 pts.	73.333 pts.
Simulador HSPICE			
Amortización	10 meses	160.000 pts.	133.333 pts.
Mantenimiento	10 meses	42.240 pts.	35.200 pts.
SOFTWARE		TOTAL	477.976 pts.

Capítulo 2

Costes de personal

El proyecto se ha realizado en 13 meses con la participación de un Ingeniero senior y otro Ingeniero junior. El presupuesto final, en lo que respecta a personal, queda reflejado en la tabla siguiente:

Descripción de tarea	*Duración*	*hombres/mes*	*Total*
Especificación			
Ingeniero senior	4 meses	486.000 pts.	1.944.000 pts.
Ingeniero junior	4 meses	250.000 pts.	1.000.000 pts.
Diseño de CIs	7 meses	250.000 pts.	1.750.000 pts.
Preparación de documento	2 meses	250.000 pts.	500.000 pts.
MANO DE OBRA		TOTAL	5.194.000 pts.

Capítulo 3

Costes de fabricación de prototipos

Los cálculos de costes de fabricación de prototipos se han realizado de acuerdo a los precios aplicados para el diseño en *runs* del MPC Francés con tecnología H–GaAs III de Vitesse. El coste de fabricación es de 67.500 pts./mm^2, con un suministro de 10 prototipos sin encapsular por circuito.

- **OPCION A**

  Fabricación del circuito GaAsDCT11, que se integrará en el encapsulado PG149.

Descripción	*Nº de CIs*	*Area (mm^2)*	*Modelo*	*Total*
GaAsDCT11	1	31.83	PG149	2.198.525 pts.
PROTOTIPADO			TOTAL	2.198.525 pts.

• OPCION B

Fabricación del circuito GaAsDCT34 para el que se puede optar, bien por la integración en un encapsulado LD344, o bien por la integración en un MCM, mediante la división del mismo en 3 módulos diferentes. Atendiendo al área y número de entradas/salidas de cada módulo usaremos distintos modelos de *pad rings* suministrados por Vitesse.

▷ *Opción B-1*

En caso de optar por la implementación en el encapsulado LD344, el coste sería:

Descripción	*N° de CIs*	*Area (mm^2)*	*Modelo*	*Total*
GaAsDCT11	1	106.52	LD344	7.240.100 pts.
PROTOTIPADO			TOTAL	7.240.100 pts.

▷ *Opción B-2*

En caso de optar por la implementación en un MCM, el coste sería:

Descripción	*N° de CIs*	*Area (mm^2)*	*Modelo*	*Total*
Primera 1-D DCT	1	63.58	LD256	4.291.650 pts.
DRAMs	1	18.75	LD52	1.265.625 pts.
Segunda 1-D DCT	1	63.58	LD256	4.291.650 pts.
MCM	1	625		516.000 pts.
PROTOTIPADO			TOTAL	10.364.925 pts.

Capítulo 4

Otros

Gastos de material fungible durante los 13 meses de duración del proyecto.

Descripcion	*Unidades*	*Precio/Unidad*	*Total*
Papel	10	500 pts.	5.000 pts.
Accesorios impresora láser	10	1.000 pts.	10.000 pts.
Cintas de back-up y diskettes	3	5.000 pts.	15.000 pts.
MATERIAL FUNGIBLE		TOTAL	30.000 pts.

Capítulo 5

Gastos totales

Resumen de los gastos totales en función de la opción elegida:

	Opción A	*Opción B-1*	*Opción B-2*
Capítulo 1:			
Coste de equipos y herramientas	2.342.419 pts.	2.342.419 pts.	2.342.419 pts.
Capítulo 2:			
Costes de personal	5.194.000 pts.	5.194.000 pts.	5.194.000 pts.
Capítulo 3:			
Costes de fabricación de prototipos	2.198.525 pts.	7.240.100 pts.	10.364.925 pts.
Capítulo 4:			
Otros	30.000 pts.	30.000 pts.	30.000 pts.
TOTAL	9.764.944 pts.	14.806.519 pts.	17.931.344 pts.