Power Delay Tradeoff Using the Genetic Algorithm

Javier Sosa, and Juan A. Montiel-Nelson
Instituto Universitario de Microelectronica Aplicada
Universidad de Las Palmas de Gran Canaria
E–35017 Las Palmas de Gran Canaria, Spain
Email: {jsosa,montiel}@iuma.ulpgc.es

Saeid Nooshabadi
Department of Information and Communications
Gwangju Institute of Science and Technology
Republic of Korea
Email: saeid@gist.ac.kr

***Abstract*—This paper presents a novel methodology to obtain the entire power versus delay tradeoff curve for the critical paths of a combinational logic circuit in a very efficient way using genetic algorithm (GA). In order to evaluate the proposed GA method a wide set of two-level and multi-level networks from the MCNC'91 benchmark suite was processed. The proposed optimization using the GA methodology is several times better than linear programming (LP) technique in terms of CPU time. On the other hand the minimum power dissipation obtained by GA and LP methods are very close to each other to within** 0.3%

I. INTRODUCTION

CMOS technology scaling has historically allowed for dramatic performance improvements; however, in recent years, undesirable effects have begun to hamper performance, requiring more energy to overcome the performance penalty. As VLSI systems migrate into new areas such as the rapidly growing markets of consumers devices in the form of PDA and mobile communication devices, sensor networks and embedded and implantable systems, power remains the most critical and limiting factor. Ultra low-power is the next design approach as VLSI is moving into bio and health sensory applications, relying on long battery life and energy scavenging.

Therefore, efficient optimization methodologies for high performance circuits where power is minimized is required. In this paper, we propose a novel genetic algorithm (GA) optimization methodology that obtains the entire power versus delay tradeoff curve for the critical path of a large combinational logic circuit with multiple inputs multiple outputs.

This paper is organized as follows. Section II presents the power consumption model. Our GA methodology, and the associated cost function are explained in detail in Section III. Comparisons based on a set of MCNC'91 circuits are presented in SectionIV. Finally in Section we will provide some conclusions and directions for future work.

II. POWER CONSUMPTION MODEL

Both delay and power consumption optimization are gate sizing problems. The main aim is to obtain the size of each gate in a Boolean network that minimizes the delay of the circuit, while reducing the total power consumption. To this end we are also interested in obtaining the complete tradeoff curve in terms of delay and power consumption.

The total power consumption of a CMOS circuit is the summation of the power consumption from all circuit gates, as:

$$\mathcal{P} = \sum_{i \epsilon N} P_i \tag{1}$$

$$P_i = a_p \ \beta_{load} \ f \ + \ b_p \ \beta_{drive} \ \tau_i \ f + c_p V_{dd} \tag{2}$$

where, a_p is the dynamic power coefficient, β_{load} is the load size factor, b_p is the short–circuit power coefficient, β_{drive} is the drive size factor, τ is the rise/fall time of input signals and c_p is the leakage factor.

The first term models the dynamic power consumption of the gate, the second term defines its short–circuit power consumption, and the last term defines the leakage power.

III. CIRCUIT OPTIMIZATION

We use symbols u and v to, respectively, denote the primary input and output nodes of the network. The schedule time of a circuit $\vec{G}$ is the maximum schedule time of the primary outputs in response to a change in the primary input, that is $T = \max(T_v), \forall v \in \vec{G}_{outputs}$.

The directed path $path_{u \to v}$ is a *critical path* of a combinational logic network $\vec{G}$ when the path delay is the schedule time of the circuit, that is $T(path_{u \to v}) = T = \max(T_v) = T$. Because $\vec{G}$ contains at least one *critical path*, we define the *set of critical paths* of network $\vec{G}$ as:

$$path_{crit} \quad = \quad \{path_{u \to v} | T(path_{u \to v}) = T = \max(T_v)\} \tag{3}$$

A. Dependent/independent path on critical path

A $path_{u \to v}$ depends on a critical path $path_{crit}$, when they share at least one node, that is the intersection between $\vec{G}$ vertex sets in $path_{u \to v}$ and $path_{crit}$ is not the empty set. In such a case, $path_{u \to v}$ is dependent on $path_{crit}$. We define the *set of* $path_{crit}$ *dependent paths* as:
$D(path_{crit}) = \{path_{u \to v} | v(path_{u \to v}) \cap v(path_{crit}) \neq \emptyset\}$

$path_{u,v}$ is *independent* of the critical path $path_{crit}$ when they do not share any node, that is the intersection between $\vec{G}$ vertex sets in $path_{u \to v}$ and $path_{crit}$ is the empty set. We define the *set of independent paths* as:
$I(path_{crit}) = \{path_{u \to v} | v(path_{u \to v}) \cap v(path_{crit}) = \emptyset\}$

B. Circuit optimization using the critical path

During the delay optimization procedure the upper limit of delay $\overline{T}$ of the critical path set $path_{crit}$ is decreased to a lower limit $\underline{T}$. The interval $(\underline{T}, \overline{T}]$ is the *optimization domain* of critical path set $path_{crit}$. The critical path seizes to be $path_{crit}$ as soon as either of the following conditions are met.

- the delay time T_v of *a dependent* $path_{u\rightarrow v}$ *of the critical path set* $path_{crit}$ is greater than or equal to $\underline{T}$,
- the maximum delay time T_v of $path_{u\rightarrow v}$ that is *independent* of $path_{crit}$ is greater than or equal to $\underline{T}$,

Beyond the $(\underline{T}, \overline{T}]$ interval, a new set of critical paths determines the maximum delay of the circuit $\vec{G}$. Therefore, during the delay optimization process critical path set changes in a iterative fashion. To account for the iterative nature of the optimization, we introduce an iteration index i to denote the sets T_i, $path_{crit,i}$, $D(path_{crit,i})$ and $I(path_{crit,i})$.

C. Critical path set delay time lower limit

The lower limit critical path set in the *optimization domain* interval $\underline{T_i}$ is obtained using the following abstract algorithm:

Minimize path delay and Power dissipation of $path_{crit,i}$
s.t. $T(path_{crit,i}) > T(path_{u\rightarrow v}) \forall\, path_{u\rightarrow v} \in D(path_{crit,i})$
and $\underline{T_i} \leq max(T(path_{u\rightarrow v})), \forall\, path_{u\rightarrow v} \in I(path_{crit,i})$

Fig. 1. Abstract algorithm for delay and power reduction tradeoff

The first minimization criterion reduces the delay through the critical paths $path_{crit,i}$ such that it lower than the delay of the *independent paths*. The second minimization criterion minimizes the power consumption of $path_{crit,i}$, subject to the condition that the delay of the $path_{crit,i}$ is maintained above the delay of the *dependent paths*.

D. Optimization Algorithm

Figure 2 presents the algorithm to obtain the complete tradeoff curve of power versus delay of a network. This algorithm is the detailed implementation of the algorithm in Figure 1. The first stage of the algorithm is to obtain the critical path and propose a reduction in its delay. The initial reduction step proposed is the delay of a unit inverter; that is the delay of a minimum sized inverter loaded by a similar inverter.

After initializing all the necessary variables, the algorithm starts its optimization loop. The first step is to reduce the delay to $\underline{T}$ for the current $path_{crit}$. It then calls the linear programming (LP) or GA or algorithm to optimized the network to obtain the minimum power dissipation for the complete circuit while meeting the set delay. While LP provides one solution for the power dissipation the GA provides several solutions.

After the GA optimization we next select the best power reduction solution that meets the specified delay of $path_{crit}$. If the selected $\underline{T}$ GA solution is less than delay through $I(path_{crit})$ or $D(path_{crit})$ sets, then the current $path_{crit}$, is updated to the longest path in one of the two sets.

If the GA optimization engine does not obtain a valid solution for the specified initial delay reduction step, algorithm iterates with a reduction step reduced by half. The maximum number of iterations is set to 10 corresponding one thousandth of the delay of a unit inverter.

E. Cost Function and GA Codification

For each *optimization domain* of the critical paths in algorithm of Figure 2, the GENEsYs GA engine [4] is invoked iteratively to a find a minimum power point solution. Using GA solutions we obtain a complete delay power tradeoff curve.

The GENEsYs GA engine is based on the evolutionary generation of multiple populations and selective migration of *individuals*(candidate solutions) [5], [6], [7]. The GA engine requires a *cost function* and codification for *individuals*. The power Equations 1 and 2 are used by the GA *cost function* to evaluate the goodness of gate sizing proposal for the minimal power in current optimization domain. From Equation 2, we note that resizing a gate will affect the power dissipation of the gate and the neighboring fan in and fan out gates.

For a network of "n" gates in $path_{crit}$, the *individuals* of the GA population will be arrays of "n" values. Each element of an *individual* array is a small increment over the current gate sizing rather than the gate size, as was introduced in [3]. The GA engine progressively uses the *cost function* to updates the *individuals* in order to increase the amount of successful evaluations. The proposed cost function codification for the GA engine [1] is shown in Figure 3. It should be noted that the GA optimization engine as many solutions as the number of *individuals* that are specified in the formulation GA problem.

The key point of this algorithm is that each gate sizing is performed in an incremental fashion. That is, each *individual* suggests an increment over the current gate sizes. The algorithm stores the current gate sizes, the delay of critical path and the total power. Once the current gate sizes are updated, the power consumption, and the gate delays arerecomputed only on those gates affected by the modification (modified gates and their fanins and fanouts gates). If the local modification reduces the total power within the set delay bound, then it is accepted and included in the current sizing.

In order to obtain the right sizing values, we propose an additional improvement. In general, the increment proposed by the GA *individual* can be accepted or rejected. When the proposal is accepted, the new gate size reduces the total power consumption. However, when an increment is rejected, it is due to the fact that either current gate size is optimum or the proposed *individual* increment is too big. So, when a gate size increment is rejected the maximum allowed *individual* increment is set to this rejected value.

Figure 4 plots the intra domain tradeoff curve for power versus delay of the `C6288` circuit network from MCNC'91 benchmark suite. `C6288` is a 16×16 multiplier with 4139 gates, and 32 inputs and output, a typical medium size circuit. If the optimization changes the domain from the current $path_{crit}$ the best obvious solution is the lowest power point. On the other hand, if the domain does not change during the optimization any point on the plot can be chosen as a tradeoff.

[1]This cost function is compiled with GENEsYs GA engine source code

Evaluate Delay Power Consumption

```
path_crit = critical path set
unit_delay = unit inverter delay (one unit drive with unit load)
Tdomain = Delay(crit)
iteration = 0

while (iterations < 10) do // path_crit delay cannot be reduced below unit delay/1024
  {Tnext = Tdomain - unit_delay
  if(Tnext < T(D(path_crit)) Tnext = T(D(path_crit)) // limit the reduction to path_crit delay lower bound
  if(Tnext < T(I(path_crit)) Tnext = T(I(path_crit)

  optimize power_consumption of circuit // Call GA/LP Optimization Engine
    s.t. Tdomain <= Tnext //Tdomain is recomputed inside the GA/LP engines

  if GA/LP solutions do not meet Tdomain <= Tnext
    delay_reduction = delay_reduction / 2
    iteration = iteration + 1

  else
    select Best Power consumption of GA/LP Solution
     s.t. Tdomain < = Tnext and update Tdomain

    if Tdomain < T(D(path_crit)) or Tdomain < T(I(path_crit)),
    update path_crit to D(path_crit) or I(path_crit) // New path_crit domain time

  endif
enddo
```

Fig. 2. Detailed algorithm for delay and power reduction tradeoff

Evaluate Power Consumption

```
PCEvaluate(GA individual)  // an individual is an array of real numbers
begin
  foreach gate g of  path_crit network
    begin
      save current size of gate g;
      set prevPower to current power consumption;
      increment g size with individual[g]; //individual[g] is a real number
      recompute the power consumption in gate g, fanout(g) & fanin(g) gates;
      recompute the delay of path_crit (Tdomain)
      if (prevPower < new power consumption) or (Tdomain > Tnext) then
        begin
          restore the size of g;
          limit the maximum increment size of g to current individual[g] value;
        end
    end
end
```

Fig. 3. Power consumption cost function.

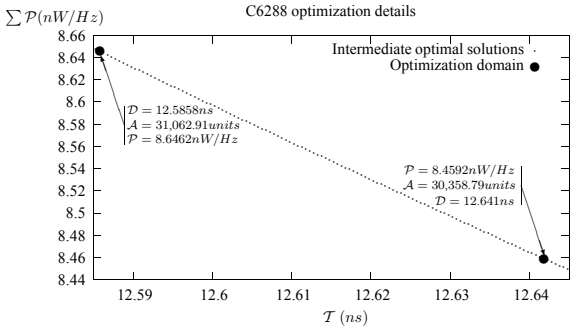

Fig. 4. C6288 power dissipation versus the delay in an optimization domain obtained by the invocation of GA. Each unit of area corresponds to the area of an minimum sized inverter.

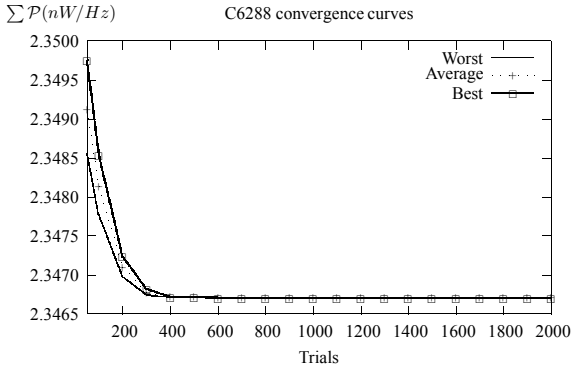

Fig. 5. Best, average and worst fitness results of C6288 network

Figure 5 presents the convergence of the GA algorithm in terms of *best*, *average* and *worst* solutions for the whole *population* for C6288 circuit. As seen the solutions converge to the minimum power very fast to within 500 trials.

IV. EXPERIMENTS

In order to evaluate the proposed method using GA engine in obtaining the complete delay versus power tradeoff curve, the set of two–level and multi–level circuits from

MCNC'91 [1] benchmark suite with more than 1000 gates of the were processed. We compare the results obtained with our GA methodology with those using a LP solver [8].

The results were computed on a Pentium M at 1.7 GHz, with 1 GByte of RAM memory, running Linux Debian 4.0 with kernel 2.6.18. We use GCC v4.1.1-21, GENEsYs [4] software as the GA engine, and a modified version of logic synthesis system MISII [2] that synthesizes for minimum power [3]. The LP solver was CPLEX v8.1 [9].

Each circuit of the MCNC'91 benchmark suite was pre-processed by MISII logic synthesis system and mapped to a library of logic gates — NAND2, NOR2, INV — with minimum power dissipation. Subsequently, each circuit was processed with the proposed algorithm in Figure 2. We minimized the power in the *optimization domains* using two techniques; the GA using the GENEsYs tool and the LP using the CPLEX v8.1 tool. Using both algorithms we obtained the complete tradeoff curve of delay versus power. The LP methodology used is similar to works in [10] and [11].

In all of GA experiments, we used the *steady state strategy* for old generation replacement after generating the *subpopulation* (offspring). The GA selection scheme was the *proportional* and the number of the *population* was 100 *individuals*. Finally, we use *multipoint crossover and standard Mutation* [6]. The number of trials to obtain the minimum power point is 2000. The number of trials must be increased from the 2000 for the larger networks, having more than 5000 gates. The minimum power dissipation obtained by two methods are very close to each other to within 0.3%.

Figure 6 presents the tradeoff curve of Power consumption versus delay of the C6288.

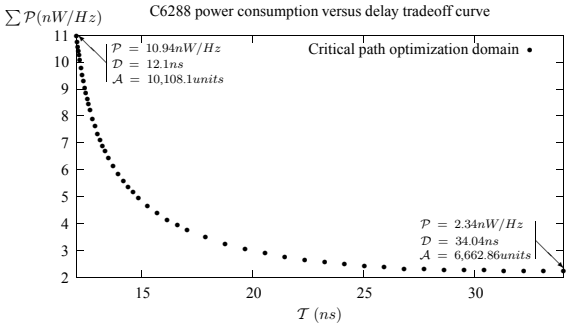

Fig. 6. C6288 circuit network power consumption versus the delay tradeoff curve. Only the domain times for all critical paths are shown. Each unit of area corresponds to the area of an minimum sized inverter.

Table I presents the comparative results in terms of CPU time to obtain complete tradeoff curve for delay versus the power consumption. The first column gives the name of the circuit according the MCNC'91 benchmark suite. The second Column gives the complexity of the circuit measured in number of gates. The third and forth Columns show the CPU time for the LP and GA solutions, respectively. CPU time is expressed in seconds. Finally, last column shows the CPU time ratios between the LP and GA techniques. The ratios are as high as 40, a clear superior performance by the GA algorithm.

TABLE I
EXPERIMENTAL RESULTS COMPARISONS USING GENETIC ALGORITHMS AND LINEAR PROGRAMMING.

Circuit		CPU Time (s)		CPU
name	# gates	LP	GA	Time Ratio
Circuit	Gates	LP	GA	Ratio
alu4	1360	64.37	14.59	4.41
apex1	1527	76.05	9.20	8.27
apex3	1985	169.29	7.34	23.06
apex4	3010	320.38	8.00	40.05
apex5	1281	87.38	4.98	17.55
C2670	1232	62.68	12.93	4.85
C3540	2198	165.10	14.58	11.32
C5315	2988	311.21	13.68	22.75
C6288	4139	316.82	34.04	9.31
C7552	3095	331.34	16.09	20.59
cps	1666	125.83	14.58	8.63
dalu	2294	139.71	17.25	8.10
ex1010	3558	422.02	7.90	53.42
ex5	1034	60.75	8.72	6.97
frg2	1307	94.71	8.92	10.62
i10	3643	286.82	20.31	14.12
i8	1603	225.69	13.23	17.06
k2	1523	85.06	8.65	9.83
pair	2562	164.64	9.22	17.86
rot	1058	34.82	8.73	3.99
table3	1094	66.61	7.12	9.36
table5	1034	61.13	6.76	9.04
x3	1323	98.46	7.60	12.96

V. CONCLUSIONS

We propose an efficient methodology to obtain the complete power consumption versus delay tradeoff curve of a logical network. We used GA as a heuristic procedure to implement the proposed methodology. Comparisons with a LP techniques demonstrate that GA obtains the complete tradeoff curve in a very efficient way (in terms of CPU time). The heuristic implementation of our methodology always performs better than the Linear Programming solution.

REFERENCES

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

[2] R. K. Brayton, R. Rudell, A. L. Sangiovanni Vicentetelli and A. Wang, *MIS: A Multiple Level Logic Optimization System*, IEEE trans. on Computer-Aided Design of Integrated Circuits and Systems, vol.6, no. 6, pp. 1062-1081, Nov. 1987.

[3] J. Sosa, J. A. Montiel-Nelson, H. Navarro and J. C. Garcia, *Minimum Power Consumption Optimization Using Genetic Algorithms*, Evolutionary and Deterministic Methods for Design, Optimisation and Control with Applications to Industrial and Societal Problems EUROGEN'05, Munich, Sep. 2005

[4] Thomas Bäck, "A User's Guide to GENEsYs 1.0," in *University of Dortmund, Department of Computer Science, Systems Analisys Research Group*, Dortmund, Germany, 1992.

[5] M. Srinivas and M. Patnaik, *Genetic algorithms: a survey*, IEEE Computer, vol. 27, No. 6, pp. 1424–1434, Nov. 1996.

[6] M. Srinivas and M. Patnaik, *Adaptive probabilities of crossover and mutation in genetic algorithms*, IEEE Trans. on Computers, vol. 24, No. 4, pp. 656-667, Apr. 1994.

[7] K. F. Man, K.S. Tang, and S. Kwong, *Genetic algorithms: concepts and applications [in engineering design]*, IEEE Trans. on Computers, vol. 43, no. 5, pp. 519-534, Oct. 1994.

[8] R. Dorfman, *The Discovery of Linear Programming*, IEEE Annals of the History of Computing, vol. 6 no. 3, pp. pp.283-295, Jul-Sept. 1984.

[9] CPLEX Optimization Inc., "Using the CPLEX linear optimizer and mixed integer optimizer," Incline Village, Nevada, 1992.

[10] M. R. C. M. Berkelaar, P. H. W. Buurman, and J. A. G. Jess, *Computing the entire active area/power consumption versus delay trade–off curve for gate sizing with a piecewise linear simulator*, IEEE trans. on Computer–Aided Design of Integrated Circuits and Systems, vol.15, no.11, pp. 1424–1434, Nov. 1996.

[11] J. A. Montiel-Nelson, J. Sosa, H. Navarro, R. Sarmiento, A. Nunez, *Efficient method to obtain the entire active area against circuit delay time trade-off curve in gate sizing*, IEE Proceed. Circuits, Devices and Systems, vol.152, no.2, pp. 133-145, Apr. 2005.