An Exact Algorithm for Low Power Library-Specific Gate Re-Sizing

De-Sheng Chen
Electronics Research & Service Organization
Industrial Technology Research Institute
HsinChu, Taiwan

and

Majid Sarrafzadeh
Department of EECS, Northwestern University
Evanston, IL 60208

Abstract – In this paper we examine the problem of reducing the power consumption of a technology mapped circuit under timing constraints. Consider a cell library that contains multiple implementations (cells) of the same Boolean function. We first present an exact algorithm for the problem when a *complete library* is given – in a complete library, "all" implementations of each cell are present. We then propose an efficient algorithm for the problem if the provided library is not complete.

1 Introduction

The portable consumer electronics is currently under a period of rapid growth. The constraints of battery operation, device density, and operating frequency have forced designers to consider power consumption as well as speed and area. Even for non-portable devices, the high cost of packaging and cooling equipments has led to increasing efforts aimed at minimizing power.

For CMOS circuits, dynamic power is the dominant source of power dissipation. The average dynamic power consumed by a CMOS gate is given by

$$P_{average} = 0.5 \frac{V_{dd}^2}{T_{cycle}} C_{load} E(switching) \ ,$$

where V_{dd} is the supply voltage, T_{cycle} is the global clock period, C_{load} is the load capacitance, and $E(switching)$ is the expected number of gate output transitions per clock cycle. Power efficiency of a silicon chip has been investigated at various levels of design phases. For instance, architectural and algorithmic transformations can trade off throughput, area, and power dissipation [3] and different logic optimization methods can result in different power dissipation of combinational logics [1, 6, 8, 9]. More recently, a voltage scaling technique that makes use of various supply voltages in a design to reduce power without changing circuit performance was shown to be effective both at the behavioral level [7] and at the circuit level [10].

In this paper we examine the problem of reducing the power consumption of a technology mapped circuit under timing constraints. We consider a cell library that contains multiple implementations (cells) of the same Boolean function. These cells differ in size, delay, and driving capability; smaller cells are also slower. (For the rest of the discussion we will use the terms "cell" and "gate" interchangeably.)

Given that the dynamic power dissipated by a cell is directly proportional to its capacitive load, reducing the capacitive load leads to a reduction of the power dissipated by the circuit. Our objective is to select a set of gates in a circuit that can be slowed down, by replacing the gates with cells in the cell library having smaller area and thus smaller capacitance load, without violating the timing constraints, and in the meantime minimizing the power consumption. Here the process of replacing a gate with a logically equivalent but smaller cell is called *gate re-sizing*. The *slack* of a gate is the timing difference between the latest time that a signal has to arrive at the output of a gate to satisfy the timing constraints and the slowest time that a transition may happen at the output of the gate. Intuitively, slack of a gate indicates the amount by which the gate can be slowed down.

In [2], a heuristic approach is presented to deal with this problem. The gates of a circuit are processed from the primary outputs in a depth-first search manner. If a gate has a positive slack, they will try to replace the gate with a functionally equivalent cell from the target library which has a larger cell delay. Once the gate is re-sized, the new gate delay is updated and the slacks for its fanin gates are computed using this new delay information. As noted, this approach lacks a global view of the whole circuit. It is common that sizing down a gate at the early stage of the process may prevent further power reduction. To overcome this drawback, we exploit the delay independence among gates in a circuit and *find a set of gates that can be re-sized simultaneously*. We first present an exact algorithm for the problem when a complete library is given – in a complete library, "all" implementations of each cell are present. If the provided library is not complete, an efficient algorithm to reduce the power consumption of a technology mapped circuit is proposed. The efficiency of the proposed approach has been shown by experiments.

The rest of this paper is organized as follows. Section 2 gives basic terminologies, notations, and formulation of the problem. In Section 3, some theoretical analysis of the problem is discussed and an exact algorithm for the problem, when a complete library is given, is

33rd Design Automation Conference ®
Permission to make digital/hard copy of all or part of this work for personal or class-room use is granted without fee provided that copies are not made or distributed for profit or commercial advantage, the copyright notice, the title of the publication and its date appear, and notice is given that copying is by permission of ACM, Inc. To copy otherwise, or to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
DAC 96 - 06/96 Las Vegas, NV, USA
©1996 ACM, Inc. 0-89791-833-9/96/0006.. $3.50

presented. In Section 4, an efficient algorithm is proposed, when the library is not complete. In Section 5, we present experimental results based on the MCNC benchmark examples.

2 Preliminaries

A combinational circuit can be represented as a directed acyclic graph, called a *delay graph* $D = (V, E)$. Each node $v \in V$ is in one-to-one correspondence with a gate v in the circuit. The gates in the circuit are assumed to be single output gates. There is an edge $(u, v) \in E$ if the output of gate u is connected to an input pin of gate v. Input signals to the circuit are called primary inputs (PIs) and output signals are called primary outputs (POs).

Each node $v \in V$ has an associated delay $d(v)$. Let the set of input pins of gate v be $inputs(v)$. We use a simple delay model to compute $d(v)$ as follows:

$$d(v) = \max_{i \in inputs(v)} d_{i,v} \ ,$$

where $d_{i,v}$ is the intrinsic gate delay from input pin i to the output of gate v. Figure 1 shows a circuit and the corresponding delay graph. The integer shown inside each node is the delay of the node. For simplicity of presentation, the delay of a gate is represented as an integral multiple of unit delay.

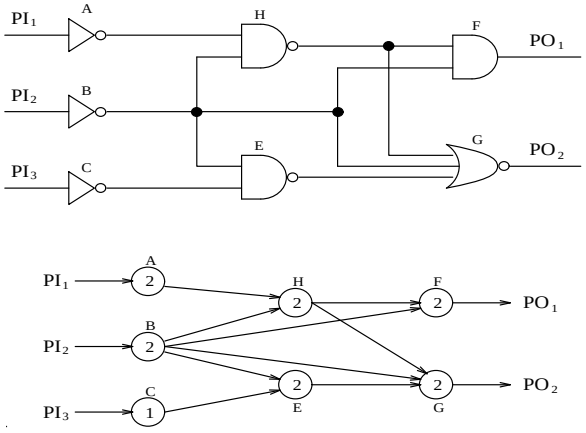


Figure 1: Circuit diagram and the corresponding delay graph D.

Each node v in D is labeled with two values: arrival time $t_a(v)$ and required time $t_r(v)$. The arrival time $t_a(v)$ is the delay of the slowest path from primary inputs to gate v. The required time is the latest time the signal has to arrive at the output of gate v to make it on time to the output pad of the circuit. Given the arrival time at each primary input, the arrival time of gate v is obtained by

$$t_a(v) = \max_{u \in Fanins\ of\ v} (t_a(u) + d(v)) \ .$$

Similarly, given the required time at each primary output, the required time at the output of gate v is obtained by

$$t_r(v) = \min_{w \in Fanouts\ of\ v} (t_r(w) - d(w)) \ .$$

Then, for each node v we define a slack $s(v)$ to be:

$$s(v) = t_r(v) - t_a(v)$$

Figure 2 gives the calculated arrival time, required time, and slack for each node in Figure 1. The triplet shown next to each node v denotes $t_a(v)/s(v)/t_r(v)$.

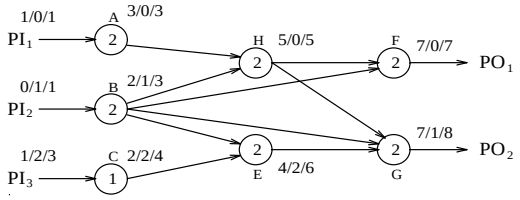


Figure 2: The arrival times, required times, and slacks of all gates in Figure 1.

Definition 2.1 *A circuit is safe, meaning it satisfies the given timing constraints, if $\forall v$ in the corresponding delay graph D, $s(v) \geq 0$.*

Definition 2.2 *Let P be a path $\{v_1, ..., v_k\}$, $v_1 \in PIs$ and $v_k \in POs$. The path slack $s(P)$ is defined as $s(P) = t_r(v_k) - t_a(v_1) - \sum_{1<i\leq k} d(v_i)$.*

Lemma 2.1 *A circuit is safe if and only if for all paths from PIs to POs in D, their path slacks are non-negative.*

Lemma 2.2 *Consider a node u in D with $s(u) > 0$. If the circuit is originally safe, then it is still safe if the delay of u increases by no more than $s(u)$.*

Now consider a technology mapped circuit with given timing constraints and a cell library that contains multiple cells of the same Boolean function. Given that the dynamic power dissipated by a cell is directly proportional to its capacitive load, if we replace a gate v in the circuit with a new functionally equivalent cell having smaller area, the power reduction in this re-sizing step would be:

$$0.5 \frac{V_{dd}^2}{T_{cycle}} \sum_{u \in Fanin\ of\ v} E(u) \times C_{diff}(v) \qquad (1)$$

where $E(u)$ is the switching activity at the output of gate u and $C_{diff}(v)$ is the capacitance difference after re-sizing gate v. However, in order to satisfy the given timing constraints, not all gates in the circuit can be re-sized; only the gates with positive slacks will be considered. Thus our *low power library-specific gate re-sizing* problem can be defined as:

Given a technology mapped circuit, timing constraints, and a cell library, replace some (or all) gates with positive slacks in the circuit with logically equivalent, smaller, but slower cells, such that the timing constrains are satisfied and the total power consumption is minimized.

This is a discrete optimization problem, and no exact polynomial time algorithm has been reported yet. As mentioned, a heuristic to deal with this problem was presented in [2]. Figure 3 shows the result of applying this approach to the graph shown in Figure 2. In this case, the delay of node E and node G increases by one unit each; the corresponding decrease in gate capacitance leads to a reduction in power dissipation. However, we could increase the delay of nodes B, C, and G by one unit each, as shown in Figure 4, without violating the timing constraints and obtain a better result if the power reduction according to the gate re-sizing in nodes B, C, and G is larger than the power reduction obtained from the above case. This observation motivates our interest in exploiting the potential delay independence among gates such that we can re-size a set of gates simultaneously and obtain a better solution than that of [2].

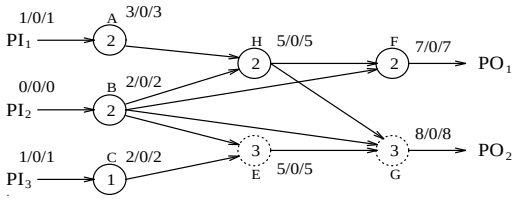


Figure 3: Result obtained by using the approach in [2].

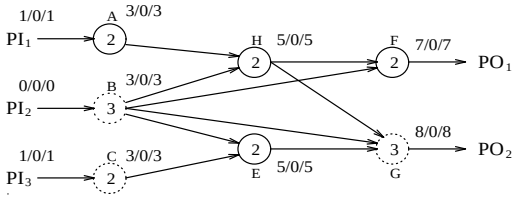


Figure 4: Result after increasing the delay of nodes B, C, and G by one unit.

3 Complete Library: An Exact Algorithm

In this section, we present an exact algorithm to minimize the power consumption of a technology mapped circuit, when a *complete library* is given. A cell library $\mathcal{L}$ is a *complete library* if it satisfies the following properties:

1. For any gate v in a circuit, if its gate delay has to be increased by τ, where τ is an integral multiple of unit delay, then there exists a functionally equivalent cell $w \in \mathcal{L}$ with delay $d(w) = d(v) + \tau$.
2. For any two cells of the same Boolean function f in $\mathcal{L}$, if their delay difference is τ, then their capacitance difference is $\tau \times C_f$, where C_f is a constant.

We now introduce the concept of *slack insensitive nodes*. Consider a delay graph $D = (V, E)$. An edge $(u, v) \in E$ is called *sensitive* if either $t_a(v) - t_a(u) = d(v)$ or $t_r(v) - t_r(u) = d(v)$. A sensitive edge (u, v) implies that the slack of u and v are sensitive to each other's timing change. The sensitive transitive closure graph $D_s = (V, E_s)$ of D is a directed graph such that there is an edge (v, w) in D_s if and only if there is a directed path from v to w in D and the path consists of only sensitive edges. Two nodes $u, v \in V$ are called *slack insensitive* if there does not exist a path from u to v or from v to u in D such that the path consists of only sensitive edges. A set $SI = \{v_1, v_2, ..., v_k\}$ of nodes in D is called *slack insensitive* if all nodes in the set are pair-wise slack insensitive.

Lemma 3.1 *A slack insensitive set SI in D is an independent set in D_s.*

Lemma 3.2 *Consider a slack insensitive set SI in D such that for each node $v \in SI$, $s(v) > 0$. If the circuit is originally safe and we increase the delay of each node $v \in SI$ by one unit, then the circuit is still safe.*

Corollary 3.1 *Consider an independent set I in D_s such that for each node $v \in I$, $s(v) > 0$. If the circuit is originally safe and we increase the delay of each node $v \in I$ by one unit, then the circuit is still safe.*

From the above lemmas, an algorithm to maximize the power reduction of a technology mapped circuit, under timing constraints, is proposed. In each pass of the algorithm, we identify the set of nodes, Q, in D with positive slack. Then construct D_q, an induced subgraph of D_s on Q, and optimize the reduction in power consumption by reducing the slack of each node in Q by exactly one unit delay. The whole process stops when there is no positive slack node in D.

Algorithm *Exact-Gate-ReSizing*

Input: A technology mapped circuit, represented as a delay graph D, the timing constraints, the switching activity of each node in D, and a complete cell library.

Output: A circuit with minimum power consumption, while satisfying the timing constraints.

begin

1. Calculate the slack for each node in D.
2. If there is no positive slack node in D, halt.
3. Construct the sensitive transitive closure graph D_s of D.
4. Identify the set of nodes, Q, in D with positive slack, and construct $D_q = (Q, E_q)$, an induced subgraph of D_s on Q such that there is an edge $(u, v) \in E_q$ if (u, v) is in D_s.
5. For each node $v \in Q$, assign a weight $w(v)$, the amount of power reduction if the delay of v were increased by one.
6. Find the maximum weighted independent set of D_q [5], and re-size the gates corresponding to the nodes in the set.
7. reduce the slack of each nodes in Q by one unit delay.
8. Goto Step 2.

end

A maximum weighted independent set of a graph is defined as a set of pair-wise independent nodes with maximum weight. By Corollary 3.1, the circuit is safe if at each pass of the algorithm, the delay of nodes in MWIS increases by one unit. Applying this argument inductively, when the algorithm halts, the circuit is still safe.

Lemma 3.3 *Let MI be the maximum weighted independent set of D_q. If we reduce the slack of each node in D_q by exactly one, the maximum power reduction is equal to the weight of MI.*

Now we would claim that the maximum power reduction at each step of the algorithm does indeed optimize the power reduction globally.

Theorem 3.1 *The algorithm Exact-Gate-ReSizing minimizes the power consumption of a given circuit.*

An example to illustrate a pass of the algorithm is given in Figure 5. In Figure 5(a), timing information of a circuit is shown and sensitive edges are indicated by dashed lines. Figure 5(b) shows the graph D_q and the weights assigned to the nodes in Q. In this case, the MWIS of D_q is {A,B,F} and the power reduction is 3.5. The updated graph D is shown in Figure 5(c).

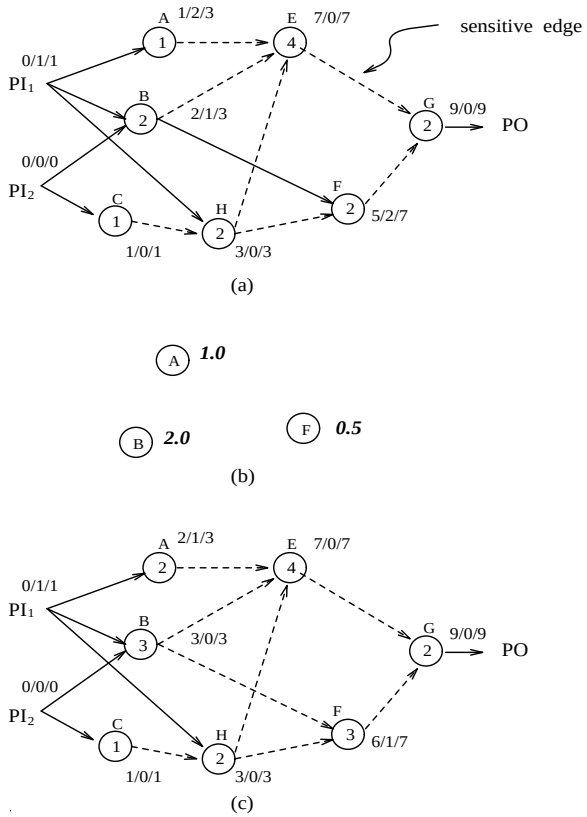


Figure 5: An example to illustrate a pass of the algorithm

4 Incomplete Library: An Efficient Algorithm

The exact algorithm described in the previous section is based on the assumption that a complete library is provided. However, this assumption is not always true in real designs, when the number of logically equivalent cells in a library is limited, and thus the algorithm does not gurantee to generate the optimal solution. For example, consider a delay graph D and the corresponding library cells shown in Figure 6(a). There is no cell mapping for node A if the gate delay is required to be two unit delay, and no cell mapping for node B if the gate delay is required to be four unit delay. If the exact algorithm is applied, in the first pass of the algorithm, the weight for node A is 0.0 because there is no cell mapping for it and the weight for node B is 0.3. Node B will be selected to increase one unit delay, as shown in Figure 6(b). However, in the second pass of the algorithm, no node will be selected to increase gate delay because there is no proper cell mapping for nodes A and B, although their slacks are still positive. In this example, we notice that the optimal solution would be to increase the delay of node A by two units directly. In the following, we will propose an efficient heuristic algorithm for the problem when an *incomplete library* is given.

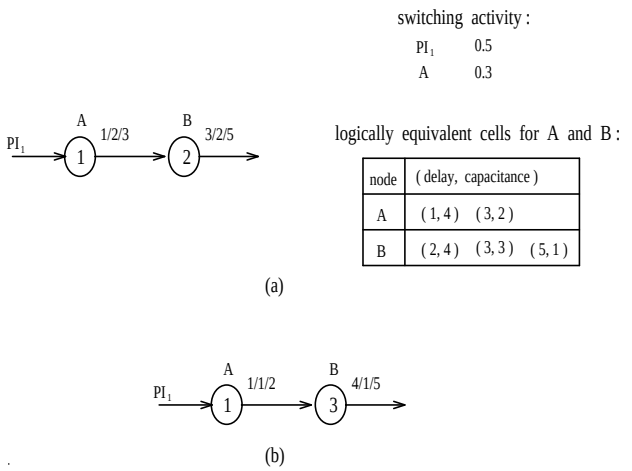


Figure 6: An example to show the *Exact-Gate-ReSizing* does not generate optimal solution when an incomplete library is used.

Again, the slacks for all nodes in the delay graph $D = (V, E)$ are first calculated. Then a transitive closure graph $D_c = (V, E_c)$ of D is constructed. (D_c is a directed graph such that there is an edge (v, w) in D_c if and only if there is a directed path from v to w in D.) Note that we use a transitive closure graph D_c here, instead of the sensitive transitive closure graph D_s. The reason is that if we use the sensitive transitive closure graph D_s and find the MWIS of D_s, the timing constraints might be violated after gate re-sizing. Each node in D_c is assigned a weight, the amount of maximal power reduction if the corresponding gate were re-sized

without violating the timing constraints. The maximum weighted independent set MI of D_c is obtained, for each $v \in MI$, $s(v) > 0$. The gates corresponding to the nodes in MI will then be re-sized.

After the process we just described, the slack of a selected node v might not become zero because of the lack of proper cell mapping in the library. The power dissipated by the circuit could be further reduced if the slack of other nodes that are in the paths passing through v are still positive and the corresponding gates can be re-sized. Therefore, in order to get a better result, the described process would need to be applied more than once such that the number of nodes with positive slack is as small as possible. Here we use a variable K, a design parameter, to control the number of passes of the process. A formal description of the algorithm is given below.

Algorithm *Heuristic-Gate-ReSizing*

Input: A technology mapped circuit, represented as a delay graph D, the timing constraints, the switching activity of each node in D, an incomplete library, and a design parameter K.

Output: A circuit that satisfies the timing constraints and has less power consumption.

begin

while $K > 0$ {

- Calculate the slack for each node in D, if there is no positive slack node in D, halt.
- Construct a transitive closure graph D_c of D.
- For each node g_i with positive slack, select the best functionally equivalent cell g_{best} from the library, g_{best} is the smallest cell such that if we replace the gate corresponding to node g_i with it, the slack of g_i would still be non-negative. Then calculate and assign a weight, according to Equation (1), to node g_i.
- Find the maximum weighted independent set of D_c, and re-size gates corresponding to the nodes in the set, accordingly.
- $K - -$.

}

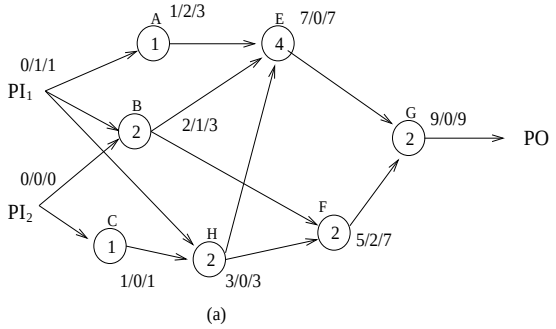
end

An example to illustrate one pass of the algorithm is given in Figure 7. Figure 7(a) gives the timing information of the circuit. Figure 7(b) shows the set of logically equivalent cells for the nodes with positive slack, and the switching activity of the nodes that will be used to calculate the weight. The transitive closure graph and the weights assigned to nodes are shown in Figure 7(c). The MWIS of the graph shown in Figure 7(c) is {A, F}.

Theorem 4.1 *If a circuit is originally safe, then the circuit is still safe after applying Heuristic-Gate-ReSizing to it.*

5 Experimental Results

In this section we present experimental results over a number of combinational circuits taken from MCNC-91 benchmark set. The library we used for the experiment contains NAND, NOR, and invertor gates. Each type of gate has five different implementations, each of which has different area-delay ratio. The calculation of the switching activity factors can be done using the symbolic simulation technique described in [4]. The power dissipation was measured in micro-Watts; it was obtained by summing up the dynamic power consumption of all gates in a circuit, assuming 5V supply voltage and 20MHz clock frequency. The timing constraints for the experiments are specified as the critical path delay of the mapping results from SIS.



(a)

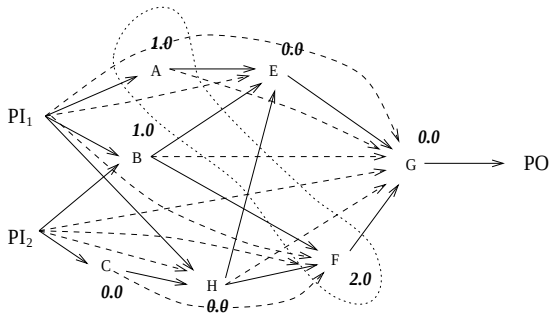
Logically Equivalent Cells:

node	(delay, capacitance)			
A	(1, 4)	(2, 3)	(3, 2)	(4, 1)
B	(1, 5)	(2, 4)	(3, 3)	(4, 2)
F	(2, 4)	(3, 3)	(4, 2)	(5, 1)

switching activity:

PI_1	0.5
PI_2	0.5
B	0.75
H	0.25

(b)



(c)

Figure 7: An example to illustrate one pass of the heuristic algorithm

For comparsion, the algorithm reported in [2] was implemented by us and denoted as **Greedy** in the following. In Table 1 we compare the results from our gate re-sizing heuristic (**K-MWIS**) with the results from **Greedy**, when the initial mappings of circuits were obtained using the SIS command **map**, which targets toward minimizing circuit area. The power dissipation of the initially mapped circuits are reported in the third column. The results obtained from (**K-MWIS**), by setting K=4, are given in the sixth column. The percentage of power reduction obtained by **Greedy** and **K-MWIS** are given in the fifth and seventh columns, respectively. On the average, **K-MWIS** shows 42.6%

power reduction over the initial mapping circuits. It is better than **GREEDY** in all cases, except a little difference for "squar5".

Ckt	Gates	Init	Greedy	red.(%) over Init	K-MWIS	red.(%) over Init
t481	949	89.7	53.9	39.8	46.3	48.4
b12	743	98.6	57.0	42.1	52.0	47.2
rd73	257	35.6	20.3	42.8	20.0	43.6
clip	330	59.0	42.9	27.1	34.1	42.1
squar5	90	7.6	4.3	43.8	4.5	40.9
sct	140	21.1	13.7	35.1	13.0	38.2
ttt2	359	66.6	42.5	36.2	39.6	40.5
sao2	154	27.3	19.7	27.9	17.1	37.4
5xp1	137	26.5	13.7	42.0	12.9	45.3

* using SIS command "map".

Table 1: Results for circuits mapped for area.

In Table 2 we show the results for the circuits whose initial mappings were obtained using the SIS command **map -n 1 -AFG**, which targets toward minimizing the circuit delay. Again the results from **K-MWIS** were obtained by setting K=4. Results show that the average power reduction over the initial mapping circuits is 38.2% for **K-MWIS** and 27.7% for **Greedy**. In all cases, **K-MWIS** performs better than **Greedy**.

Ckt	Gates	Init	Greedy	red.(%) over Init	K-MWIS	red.(%) over Init
t481	1748	144.1	100.1	30.4	79.7	44.6
b12	1337	136.4	93.9	31.1	79.1	41.9
rd73	534	63.3	45.7	27.5	38.6	38.6
clip	692	84.4	60.9	27.6	49.8	40.9
squar5	174	15.0	12.3	17.7	9.4	37.5
sct	239	33.8	22.3	34.2	22.2	34.3
ttt2	771	85.4	57.4	32.7	51.0	40.2
sao2	346	44.3	35.0	20.4	31.0	30.5
5xp1	280	37.9	27.6	27.4	24.4	35.6

* using SIS command "map -n 1 -AFG".

Table 2: Results for circuits mapped for speed.

6 Conclusion

The design of low power CMOS circuits is of crucial importance for today's digital devices. For applications such as notebook computers and cellular phones, low-power dissipation is one of the major concerns. In this paper we present algorithms to reduce the power consumption of a technology mapped circuit under timing constraints. When a complete library is given, an exact algorithm is presented to maximize the power reduction of a circuit by gate re-sizing. If the given cell library is not complete, an efficient algorithm is proposed. The experimental results show that the proposed heuristic approach improves the power consumption on the average by 42.6% for circuits mapped for minimizing area, and on the average by 38.2% for circuits mapped for minimizing delay.

REFERENCES

[1] M. Alidina, J. Monteiro, S. Devadas, A. Ghosh, and M. Papaefthymiou, "Precomputation-Based Sequential Logic Optimization for Low Power", *IEEE Trans. on VLSI Systems*, Vol. 2, No. 4, pp. 426–436, December 1994.

[2] R.I. Bahar, G.D. Hachel, E. Macii, and F. Somenzi, "A Symbolic Method to Reduce Power Consumption of Circuits Containing False Paths", *Proceedings of IEEE International Conference of Computer Aided Designed*, pp. 368–371, November 1994.

[3] A.P. Chandrakashan, S. Sheng, and R.W. Brodersen, "Low Power CMOS Digital Design", *IEEE Trans. on Solid State Circuits*, Vol. 27, No. 4, pp. 473–483, April 1992.

[4] A. Ghost, S. Devadas, K. Keutzer, and J. White, "Estimation of Average Switching Activity in Combinational and Sequential Circuits", *Proceedings of the IEEE Design Automation Conference*, pp. 253–259, June 1992.

[5] R.H. Mohring, "Graphs and Orders: the Role of Graphs in the Theory of Ordered Sets and Its Application", *Published by D. Reidel Publishing Company*, edited by I. Rival, New York and London, pp. 41–101, May 1984.

[6] K. Roy and S. Prasad, "SYCLOP: Synthesis of CMOS Logic for Low Power Application", *Proceedings of the International Conference on Computer Design*, pp. 464–467, October 1992.

[7] S. Raje and M. Sarrafzadeh, "Variable Voltage Scheduling", *Internationl Symposium on Low Power Design*, pp. 9–14, April 1995.

[8] C. Tsui, M. Pedram, C. Chen, and A. Despain, "Low Power State Assignment Targeting Two- and Multi-level Logic Implementations", *Proceedings of the IEEE International Conference on Computer Aided Design*, pp. 82–87, Nov. 1994.

[9] C. Tsui, M. Pedram, and A. Despain, "Technology Decomposition and Mapping for Low Power", *Proceedings of the IEEE Design Automation Conference*, pp. 68–73, June 1993.

[10] K. Usami and M. Horowitz, "Clustered Voltage Scaling Technique for Low-Power Design", *Internationl Symposium on Low Power Design*, pp. 3–8, April 1995.