Power Reduction by Gate Sizing with Path-Oriented Slack Calculation

How-Rern Lin and TingTing Hwang
Department of Computer Science, National Tsing Hua University
Hsinchu, Taiwan 30043, R.O.C.

Abstract

This paper describes methods for reducing power consumption. We propose using gate sizing technique to reduce power for circuits that have already satisfied the timing constraint. Replacement of gates on noncritical paths with smaller templates is used in reducing the dissipated power of a circuit. We find that not only gates on noncritical paths can be down-sized, but also gates on critical paths can be down-sized.A power reduction algorithm by means of single gate resizing as well as multiple gates resizing will be proposed. In addition, to identify gates to be resized, a path-oriented method in calculating slack time with false path taken into consideration will be also proposed. During the slack time computation, in order to prevent long false path from becoming sensitizable and thus increasing the circuit delay, slack constraint will be set for gates. Results on a set of circuits from MCNC benchmark set demonstrate that our power reduction algorithm can reduce about 10% more power, on the average, than a previously proposed gate sizing algorithm.

Keywords : *low power, gate sizing, time slack, sensitizable path.*

1 Introduction

There have been ever-increasing portable applications developed, e.g., notebook, laptop computer, and personal communications services which are demanding the same computation capabilities as found in desktop machines. However, it is unlikely that dramatic solution to the power problem is forthcoming; it was projected that only 30% improvement in battery performance will be obtained in the next five years. Moreover, as the fabrication technology advances, high density, large volume, and high-speed chips have been developed. But, the cooling problem for the heat generated by the dissipated energy remains unsolved. Therefore, low power digital design becomes more and more important.

The power dissipation of CMOS devices can be reduced by using the lowest possible supply voltage coupled with architectural, logic style, circuit, and technology optimization [CSB92]. In addition, accurate selection of technology mapping algorithms [TPD93, TAM93]has also been proposed to reduce the dissipated energy of a circuit. Moreover, for a delay optimized circuit, the power consumption can be reduced by down-sizing the gates on noncritical paths.

Replacement of the gates on noncritical paths with smaller templates has been used in reducing the dissipated power of a circuit in [BHM94]. However, we find that not only the gates on noncritical paths can be down-sized, but the gates on critical paths can be down-sized in two cases. First, after the gates on noncritical paths are down-sized, a critical path may become non-critical since the loading capacitances of some gates on the critical path are decreased. Consequently, the gates on the originally critical path may be down-sized. Secondly, simultaneously down-sizing and up-sizing multiple gates may reduce power consumption. The satisfaction of delay constraint can be retained if up-sizing gates can compensate the loss in delay caused by down-sizing gates on the critical path. Then, the power may be reduced if the gate with high transition density inputs is selected for down-sizing and the gate with low transition density inputs for up-sizing since the dissipated power of a gate is proportional to the product of fanout loading capacitance and the transition density.

Gate sizing for delay optimization relies on the detailed timing information to identify gates to be resized. Traditionally, the circuit is modeled as a directed acyclic graph, such as the PERT digraph [Eve79]. This static timing analysis will underestimate the slack of a gate since the path sensitizability is not taken into account. Nevertheless, it is impractical to apply all possible input vectors to find out the time slack of a gate since the total number of input vectors grows exponentially with the number of primary inputs.

In this paper, we will propose a path-oriented method in calculating slack time with false path taken into consideration. First, we will partition paths into three sets: sensitizable path, changeable false path, and function-false path. Based on the type of a path, we compute the slack time for the gates on the path. During the computation, in order to prevent long false path from becoming sensitizable and thus increasing the circuit delay, slack constraint will be set for gates.

The remaining of this paper is organized as follows. Section 2 will discuss how the static timing analyzer underestimates the time slack. Taking into account the path sensitizability, a path-oriented method for slack calculation will be proposed in Section 3. In Section 4, a power reduction algorithm based on single gate resizing and multiple gates resizing will be presented. In Section 5, we give benchmark results on a set of circuits from MCNC benchmark set.

2 Static Timing Analysis Underestimating Time Slack

To reduce the power consumption of a circuit by iterative gate sizing without violating the timing constraint, selecting gates with time slack for replacement is one of practical ways. Traditionally, time slack is calculated as follows. The circuit is modeled as a directed acyclic graph, such as the PERT digraph [Eve79]. The arrival time at the output of a gate is computed by the length of the longest path from the primary inputs to this gate.

For a given delay constraint on the primary outputs, the required time is the time at which the output of the gate is required to be stable. The time slack is defined as the difference of the required time and the arrival time of a gate. If the time slack is greater than 0, the gate can be slow-downed. However, only sensitizable path which can be activated by at least one input vector contributes to the delay of a circuit. This method for slack calculation apparently underestimates the slack of a gate since the path sensitizability is not taken into account.

Figure 1 shows an example of static timing analysis underestimating time slack. For simplicity, the rising and falling delays of each gate are assumed to be the same. The delay of the inverter is set to be 1 and all other gates 4. For each pair of values [a,b] of signal s_i, a and b mean the stable times of logic value 0 and 1 on signal s_i, respectively. Figure 1(a) shows the stable time calculated by static timing analyzer and dynamic timing analyzer. For static timing analysis, the circuit is modeled as a digraph. The signal stable time is estimated by the length of the longest path from the primary input. There are three longest paths $P_1 = \overline{i_1} - g_4 - \overline{s_4} - g_5 - \overline{s_5} - g_3 - s_3$, $P_2 = \overline{i_3} - g_4 - \overline{s_4} - g_5 - \overline{s_5} - g_3 - s_3$, and $P_3 = i_2 - g_1 - \overline{s_1} - g_2 - \overline{s_2} - g_3 - s_3$, by which the value of signal s_3 will settle to 1. Since lengths of P_1, P_2 and P_3 are 12, 9 and 12, respectively, the stable time of s_3 (logic value 1) will be 12 by static timing analysis. For dynamic analysis which takes the path sensitizability into account, all 8 input vectors in this circuit are applied to determine the real stable time of each signal. The stable time of s_3 (logic value 1) will be 9 because both paths P_1 and P_3 are false.

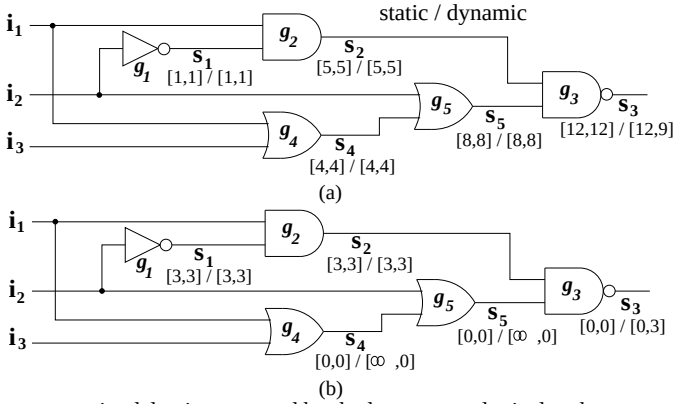

Figure 1: The time slack calculation with and without considering path sensitizability (a) Stable time (b) Time slack

Let the required arrival time of the primary output s_3 be 12 in this circuit. Figure 1(b) shows the time slack calculated by static and dynamic timing analyzers. Static timing analyzer first computes the required time of each signal from output to input. The time slack is then calculated by the difference of required time and stable time. For example, the time slack of s_4 is 4 - 4 = 0. Dynamic timing analyzer uses a similar procedure to calculate time slack but it takes the path sensitizability into consideration. For example, signal s_4 is settled to value 0 ($\overline{s_4}$) only by paths P_1 and P_2. However, both paths P_1 and P_2 are false. Therefore, the slack of $\overline{s_4}$ is ∞. Obviously, the static timing analyzer underestimates the time slack.

3 Slack Calculation Taking Path Sensitizability into Account

The delay of a circuit is the length of the longest sensitizable path in the circuit. During an iterative optimization process, sensitizable paths may become false and false paths may become sensitizable [LiH94]. Hence, path sensitizability should be taken into account in time slack calculation.

Taking the path sensitizability into account, the calculation of slack time is formulated as follows. For an input vector v, let $AT(g_i, v)$ be the arrival time of gate g_i and $RT(g_i, v)$ be the required time of gate g_i under a given delay constraint. The time slack of gate g_i with respect to the input vector v is defined as $slack(g_i, v) = RT(g_i, v) - AT(g_i, v)$. For all input vectors, the slack of gate g_i is defined as:

$$slack(g_i) = \min_{\forall v} \ slack(g_i, v). \qquad (1)$$

However, for a circuit with n primary inputs, there is 2^n possible input vectors. It is impractical to apply all possible input vectors to find out the time slack of all gates. Therefore, we will take a path-oriented approach to calculating the slack of a gate.

3.1 Classification of Path Set

Since only sensitizable paths contribute to the delay of a combinational circuit, it is necessary to check if a path is sensitizable, so as to estimate the delay of a circuit accurately. A sensitization criterion is needed to decide whether a path is sensitizable. The *loose* sensitization criterion defined in [ChD91] will be used in this paper.

Definition 3.1 (the loose sensitization criterion): *Given a path $P = s_0 - g_0 - s1 - g_1 - ... - s_{k-1} - g_{k-1} - s_k$. Path P is a sensitizable path if and only if there exists at least one input vector I such that for each signal s_i along P satisfies the following conditions :*
(1) if s_i is a controlling input to gate g_i, s_i must be the earliest controlling input.
(2) if s_i is a non-controlling input to gate g_i, then all other side-inputs to g_i must be non-controlling inputs.

It is reported in [ChD91] that the loose sensitization criterion does not overestimate the circuit delay. Based on this sensitization criterion, the paths in a circuit can be classified into two subsets: sensitizable path and false path. During an iterative optimization process [LiH94], sensitizable paths may become false and false paths may become sensitizable. Therefore, the path sensitizability should be considered in each iteration of an optimization process.

However, not all false paths may become sensitizable. Certain false paths never become sensitizable during circuit optimization process, e.g., one that results from incompatibility in propagation conditions, from the existence of redundancy in a circuit etc. These paths are defined as *function false paths* in [LiH94]. Since function false paths will never become sensitizable during optimization, they can be put aside during optimization process and computation effort on slack calculation for function false paths can be saved. Based on the above observation, the false path can be further classified into

function false path and changeable false path. Figure 2 depicts the classification of path set.

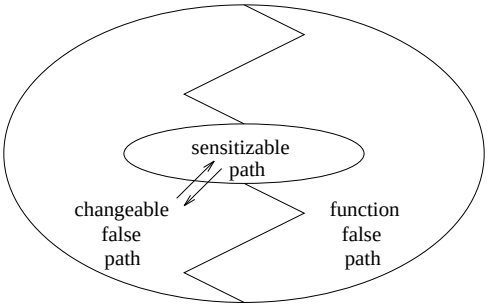

Figure 2: classification of path set.

3.2 Slack Calculation

Now we are showing how to calculate the slack with the false path taken into account. Consider a delay-optimized circuit, where the timing constraint is already satisfied and the delay is the length of the longest sensitizable path. Then, when calculating the time slack of gates on a path P, the following two cases happen, given the path length ($path\_delay(P)$)and the delay constraint ($delay\_constraint$) :

Case 1. *path_delay(P) $\leq$ delay_constraint:*
In this case, the time slack of path P is calculated as *slack(P)= delay_constraint - path_delay(P).* Under the delay constraint on path P, the gates on P can be replaced by smaller templates if the increment in delay is not greater than the $slack(P)$. Therefore, the time slack of gate g_i for all gates $g_i \in P$ is $slack(g_i, P) = slack(P)$.

Case 2. *path_delay(P) > delay_constraint:*
In this case, path P must be a false path since the delay constraint is the length of the longest sensitizable path. Since function false path never become sensitizable, we only have to concern about changeable false path. Let P be a changeable false path. There must exist at least one gate on P, where the on-input to this gate is a *controlling input* and there is at least one earlier side-input which is also a *controlling input* such that the on-input is not the earliest controlling input for all input vectors. If the earliest controlling side-input is restricted to no later than the on-input, P will not become sensitizable during the optimization process. Let Δd be the value that the earliest controlling side-input can be slowed down such that path P remains false during the optimization process. Then, the time slack of all gates, g_i, in the cone which fans into the earliest controlling side-input can not be greater than Δd. Therefore, for all such gates g_i, we set

$$slack\_constraint(g_i, P) = \Delta d.$$

Based on the classification of path set shown in Figure 2, we can calculate the slack time of gates for each path. If a path belongs to sensitizable path, Case 1 can be applied to calculate the time slack of gates residing on P. If a path is a function false path, it can be ignored since it never become sensitizable. For a changeable false path P, if the length of P is less than the circuit delay, Case 1 can be applied and the satisfaction of delay constraint of path P is guaranteed. But, if the length of P exceeds the circuit delay, Case 2 is used in setting slack constraint for gates in the cone which fans into the earliest controlling side input to prevent P from becoming sensitizable.

```
procedure slack_calculation(circuit, delay_constraint, R)
  for all paths P
    if (path P is not a function-false path)
      if (path_delay(P) ≥ R × delay_constraint)
        if (path_delay(P) ≤ delay_constraint)
          slack(g_i)=min(slack(g_i), slack(g_i,P))
            for all gate g_i in P;
        else
          slack(g_i)=min(slack(g_i), slack_constraint(g_i,P))
            for all gates g_i in the cone which fans into
            the earliest controlling side input;
        end if
      end if
    end if
  end for
endprocedure
```

Figure 3: Slack calculation algorithm

Since a gate is passed by many paths, the calculation of slack time of a gate should take all of them into consideration. Therefore, the time slack of a gate g_i, $slack(g_i)$, is determined by the following formula:

$$slack(g_i) = \min_{\forall P} \ slack(g_i, P). \qquad (2)$$

That is, the slack of a gate is set to the minimum of all slacks computed for paths passing through g_i. If Equation 2 is used directly in determining the time slack of a gate, a large number of paths will be checked. It will consume a lot of computation time. Fortunately, we find that the time slack of a gate is usually determined by the long paths passing through this gate. Therefore, we only have to consider paths whose length are greater than a threshold value, says $R \times delay\_constraint$ *for* $0 \leq R < 1$, to calculate the time slack of gates. The closer the R to 0, the more accurate slack of gate can be obtained. However, more computation time is needed. Moreover, during setting slack time of each gate, if there is slack constraint for the gate, the minimum value of slack constraint and the smallest slack time computed so far will be set. Figure 3 shows the slack calculation algorithm. In the algorithm, the initial slack time of each gate is set to be ∞. In each iteration, for all gates on a path, if the currently computed slack value is smaller than the minimum value obtained so far, it replaces the old value.

4 Power Reduction by Iterative Gate Sizing

4.1 Power Estimation

For a given circuit, the average dissipated energy of a gate is computed as

$$P_{avg} = \frac{1}{2} \times C_{load} \times \frac{V_{dd}^2}{T_{cycle}} \times E(Transitions), \qquad (3)$$

where C_{load} is the output capacitive load, V_{dd} is the supply voltage, T_{cycle} is the global clock, and E ($Transitions$) is the transition density of the output. In our power dissipation model, the C_{load} includes not only the output capacitance of the gate but also the capacitance of the internal nodes. For example, for the 4-input AND gate shown in Figure 4, the capacitances of internal nodes node1, node2, node3 and node4 are extracted from the cell layout in the standard cell library and included in the term of C_{load}.

4.2 Single Gate Resizing

To reduce the power consumption of a delay-optimized circuit without increasing the circuit delay, the gate on noncritical path can be slowed down by replacing the gate with a smaller template. The reason

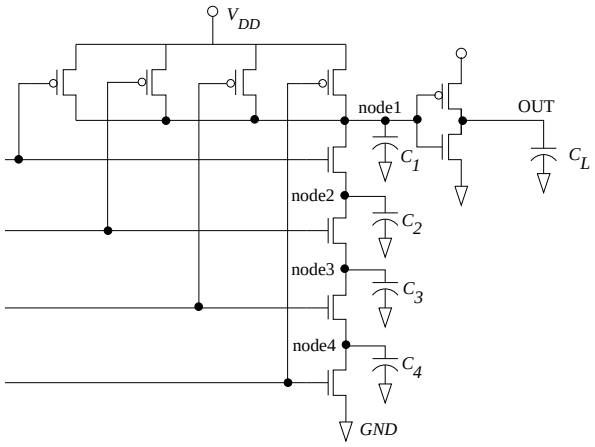

Figure 4: The capacitances of internal nodes in a cell

why the dissipated power of the circuit can be reduced is because the output capacitive load of the gate which fans in to the resized gate is decreased. Therefore, all gates with slack time greater than zero are candidates for down-sizing. To choose a specific one, we have the following two considerations. To achieve the largest reduction of power consumption, the gate with least delay increment has higher priority to be down-sized. To save the total consumed time slack of multiple paths by down-sizing a gate, the gate which passed by smaller number of paths is preferable for resizing. Therefore, the gain function used in selecting a gate to be down-sized is defined as follow:

$$gain(g_i) = \frac{\Delta power(g_i)}{\Delta delay(g_i) \times |P_{g_i}|}, \quad (4)$$

where $\Delta power(g_i)$ is the decrement of dissipated power, $\Delta delay(g_i)$ is the delay increment by down-sizing gate g_i, and $|P_{g_i}|$ is the number of noncritical paths passing through gate g_i. The gate with the maximum gain value is selected to be resized.

The delay increment $\Delta delay(g_i)$ is the difference between the stable time after and before down-sizing g_i and is computed as in the following equation:

$$\Delta\text{delay}(g_i) = \text{new\_stable\_time}(g_i) - \text{old\_stable\_time}(g_i), \quad (5)$$

where $new\_stable\_time(g_i)$ and $old\_stable\_time(g_i)$ are the stable times of gate g_i after and before down-sizing g_i. The computation of stable time can be performed by static timing analysis or dynamic timing analysis. Two analysis methods will result in different accuracy in the amount of delay that can be increased. In static timing analysis, only local delay change is considered. When gate g_i is down-sized, the delay of the gate g_j which fans in to g_i will be decreased because the fanout loading is decreased. Let the delay of fanin gate g_j is decreased by $\Delta delay(g_j)$ after g_i is down-sized. The new stable time of fanin gate g_j is determined by:

$$\text{new\_stable\_time}(g_j) = \text{old\_stable\_time}(g_j) - \Delta\text{delay}(g_j), \quad (6)$$

where $new\_stable\_time(g_j)$ and $old\_stable\_time(g_j)$ are the stable time of fanin gate g_j after and before g_i is down-sized. The stable time of g_i after down-sizing g_i is then computed according to the maximum new stable time among the fanin gates and the new gate delay of g_i as in the following equation:

$$\begin{aligned} new\_stable\_time(g_i) &= \max_j\{new\_stable\_time(g_j)\} \\ &+ new\_gate\_delay(g_i), \end{aligned}$$

where $new\_gate\_delay(g_i)$ is the gate delay of g_i after down-sizing g_i. We now take the circuit shown in Figure 5 as an example. Let the gate delay (stable time) of g_1, g_2 and g_3 are 4 (4), 5 (5), and 5 (10), respectively. Suppose gate g_3 is down-sized and the gate delay is increased to 7 and the delay of gate g_1 and g_2 are decreased by 0.5 and 1 due to the decrease in output capacitive load, respectively. Then the new stable time of g_1 and g_2 will be 3.5 and 4, respectively, and the new stable time of g_3 will be calculated as:

$$\begin{aligned} new\_stable\_time(g_3) &= \max\{3.5, 4\} + 7 \\ &= 4 + 7 \\ &= 11. \end{aligned}$$

Finally, the delay increment by down-sizing g_3 will be 11-10=1.

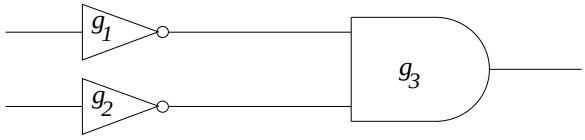

Figure 5: Delay increment calculation using static timing analysis

The above method does not take the path sensitizability into account. Hence, the delay increment may be overestimated. In the dynamic timing analysis, the new stable time of down-sized gate is computed by analyzing all the paths passing through this gate. That is, the longest sensitizable path which passes through the resized gate is extracted to calculate the new stable time. Therefore, the real delay increment can be obtained by dynamic timing analysis. However, compared to the static delay increment calculation, the dynamic one may consume more computation time.

It is worth noticing that not only the gates on the noncritical paths in the initial circuit can be down-sized, but also the gates on critical paths can be down-sized after some noncritical gates are down-sized. Consider the example circuit shown in Figure 6. In this figure,

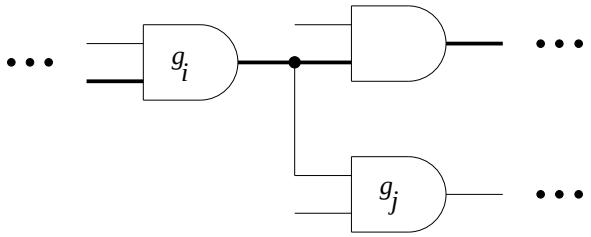

Figure 6: Single gate resizing

gate g_i is passed by a critical path P and g_j is not passed by any critical path. Simply changing gate g_i with a smaller template will cause the violation of delay constraint. However, if gate g_j, on the noncritical path, is down-sized first, then the output capacitive load of gate g_i is reduced and thus the gate delay of g_i as well as the path delay of P are shortened. Consequently, the gates on path P which is critical originally may be down-sized. In such case, the dissipated power of the circuit is reduced and the satisfaction of delay constraint can be retained.

4.3 Multiple Gates Resizing

After performing the power reduction process by single gate resizing, replacing any single gate with a smaller template will cause violation of delay constraint. But,

by up-sizing and down-sizing multiple gates simultaneously, the dissipated power of the circuit can still be reduced. Consider the example circuit shown in Figure 7. Let paths $P_1 = a - g_1 - s_1 - g_2 - s_2 - g_3 - s_3$ and $P_2 = a - g_1 - s_1 - g_2 - s_2 - g_4 - s_4$. Assume the transition density of node s_2 is much higher than that of primary input a. If gate g_3 is considered solely to be down-sized to reduce the capacitive load of gate g_2, the power can be reduced but the delay constraint on path P will become unsatisfied. If up-sizing gate g_1 can compensate the loss of delay caused by down-sizing g_3, then down-sizing g_3 followed by up-sizing g_1 can retain the satisfaction of delay constraint. Moreover, resizing these two gates simultaneously may reduce the dissipated power of the circuit if the transition density in node s_2 is much higher than a because the dissipated power of a gate is proportional to the product of fanout loading capacitance and the transition density.

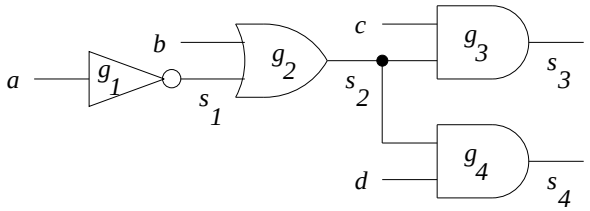

Figure 7: Multiple gates resizing

Multiple gates resizing is to replace a group of gates at the same time. Its goal is to reduce the dissipated power of a circuit while retaining the satisfaction of timing constraint. In the following, we will describe how to select a group of gates for resizing. First, we select the gate, say g_i, to which the transition density of the fanin gate is highest as candidate for down-sizing. However, solely down-sizing g_i will violate the delay constraint. We need to up-size some gates to compensate the loss of delay. The compensating gates are selected as follows. Presumably, we down-size the gate g_i and lock it as non-replaceable. Then, a delay optimization procedure is invoked which uses gate sizing as optimization technique (e.g., the system proposed in [LiH94]). After the delay optimization is completed, several gates are selected for replacement. These gates are referred to as $partner(g_i)$. The gates g_i and $partner(g_i)$ form a group for resizing. Let $\Delta power(G)$ be the improvement of dissipated power when the gates in the group of gates G are resized. If $\Delta power(G)$ is greater than 0, then the gates in G are candidates to be resized. The same procedure is repeated several times to form group candidate for the gate with input having the next higher transition density. The number of groups to form, N_G, is a user-provided parameter. After N_G number of candidate groups are formed, the group of gates G with the largest $\Delta power(G)$ value is selected for replacement. The above group-replacement procedure is repeated until no more improvement can be made.

4.4 Power Reduction Algorithm

Based on the above observations, a power reduction algorithm is proposed in Figure 8. The power reduction algorithm is a two-phase procedure. Phase one performs single gate resizing. At the beginning of each iteration, the time slack of each gate is calculated. Then, the gate with the maximum gain value is resized with the constraint that the delay increment by resizing this gate

```
Algorithm power_reduction(circuit, delay_constraint, R, N_G)
  Phase 1: single gate resizing
  do {
    slack_calculation(circuit, delay_constraint, R);
    selected_gate = single_gate_selection(circuit);
    if (selected_gate ≠ ∅)
      replace the selected_gate by a smaller template;
    end if
  } while (selected_gate ≠ ∅)
  Phase 2: multiple gates resizing
  unmark all gates;
  do {
    high_density_gates = sort_gate(unmarked_gates, N_G);
    for all gates g_i in high_density_gates
      if (gate g_i can be replaced by smaller template)
        then partner(g_i)= delay_optimization(g_i, circuit);
             form {g_i, partner(g_i)} as a candidate group G_i;
        else partner(g_i)= ∅;
             mark gate g_i;
      end if
    end for
    G_k = {g_k, partner(g_k)}, where G_k has the largest
          power reduction among all candidate groups;
    if (Δpower(G_k) > 0)
      then resize all gates in G_k simultaneously;
           mark gate g_k;
    end if
  } while (Δpower(G_k) > 0)
endalgorithm
```

Figure 8: Power reduction algorithm

should not be greater than the time slack of this gate. The slack calculation and single gate resizing are performed iteratively until down-sizing any gate causes the violation of delay constraint. The algorithm then proceeds to the second phase. Phase two is also an iterative procedure. In each iteration, a number of groups for replacement are formed. The group of gates with the highest reduction in power are chosen to be resized simultaneously. The iteration repeats itself until no more improvement can be made.

5 Experimental Results

The algorithm *power_reduction* has been implemented in C language on a SUN SPARC-10 workstation. Benchmarking process is performed on some combinational circuits from the MCNC benchmark set [Yan91]. First, the circuits are technology mapped to a standard cell library using SIS mapper [SSM92] with delay optimization option. The delay constraint is set to be the length of the longest sensitizable path of the mapped circuit which will not be increased during power reduction process.

The benchmarking process is first conducted to investigate the effectiveness of single gate resizing by different delay increment computation methods. First, the *slack_calculation* procedure in Figure 3 is performed to compute the slack of each gate in the circuit. Then, with static and dynamic delay increment computation methods, the single gate resizing phase is performed. Table 1 shows the power reduction comparisons of two delay increment computation methods in calculating Equation 4. Columns **circuit** and **gates** are the benchmark circuit and the number of gates in the circuit, respectively. Columns **static** and **dynamic** show the power reduction by using static and dynamic delay increment computation methods, respectively. Table 1 shows that dynamic delay increment computation method outperforms static one for all the circuits. About 16% reduction in power consumption can be obtained on the average. The reason is that the delay increment is overestimated in static timing analysis

such that some gates which can be down-sized are not selected for down-sizing.

Table 1: Comparisons of two delay increment computation methods

circuit	gates	static (A) %	dynamic (B) %	(B) - (A) %
alu2	508	55.14	57.48	2.34
alu4	951	51.82	56.64	4.82
b9	144	30.55	55.34	24.79
c432	247	39.05	53.91	14.86
c8	215	36.56	54.42	17.86
c880	493	45.83	57.34	11.51
cht	278	28.29	55.42	27.13
cmb	50	25.75	53.25	27.50
comp	182	42.14	55.88	13.74
cordic	108	36.26	57.75	21.49
count	173	47.00	57.66	10.66
f51m	177	50.33	57.89	7.56
lal	163	32.83	53.68	20.85
parity	75	41.32	58.76	17.44
rot	818	32.28	55.28	23.00
term1	493	45.05	55.98	10.93
average		40.01	56.04	16.03

The gate selection strategy proposed in [BHM94] is also implemented. A backward searching of gates for resizing from the primary outputs to the primary inputs is performed in this gate selection strategy. The gate is selected for down-sizing if the delay increment is not greater than the time slack of this gate. After a gate is resized, the time slack of all its fanin gates are recomputed. The comparison of this method with our single gate resizing phase is performed. Table 2 shows the results of these two gate selection and resizing procedures. The

Table 2: Power reduction by single gate resizing

circuit	BHM		phase1		(B)-(A)
	ΔArea %	**ΔPower** (A) %	**ΔArea** %	**ΔPower** (B) %	
alu2	47.50	43.99	63.07	57.48	13.49
alu4	57.94	53.31	62.86	56.64	3.33
b9	56.33	50.57	63.51	55.34	4.77
c432	37.88	36.67	64.04	53.91	17.24
c8	57.63	51.32	63.06	54.42	3.10
c880	53.79	50.76	63.16	57.34	6.58
cht	60.93	55.15	62.60	55.42	0.27
cmb	49.11	44.43	63.23	53.25	8.82
comp	34.83	32.31	62.84	55.88	23.57
cordic	58.06	54.30	63.09	57.75	3.45
count	54.78	53.23	64.27	57.66	4.43
f51m	58.76	54.76	62.31	57.89	3.13
lal	55.74	49.03	63.40	53.68	4.65
parity	0.00	0.00	62.40	58.76	58.76
rot	62.05	55.78	62.02	55.28	-0.50
term1	59.69	54.02	62.70	55.98	1.96
average	50.31	46.23	63.04	56.04	9.82

column *circuit* is the name of the benchmark circuit. The column **ΔArea** and **ΔPower** are the percentages of area and power improvement of the optimized circuit to the initial circuit, respectively. As shown in Table 2, our single gate resizing procedure outperforms BHM for all selected circuits except rot. It is worth noticing that no power reduction can be obtained in circuit parity by BHM, but substantial power reduction is achieved by our method. On the average, our method gains about 9.8% more power reduction than BHM.

The benchmarking process is continued to investigate the effectiveness of the multiple gates resizing which is applied when down-sizing any single gate will cause the violation of delay constraint. Table 3 shows the results of phase1 and phase1+phase2. As shown in Table 3, on the average, phase 2 can further reduce about 0.6% more power.

Table 3: Power reduction by single gate resizing and multiple gates resizing

circuit	phase1 (A) %	phase1+phase2 (B) %	(B) - (A) %
alu2	57.48	57.48	0.00
alu4	56.64	57.22	0.58
b9	55.34	56.42	1.08
c432	53.91	55.03	1.12
c8	54.42	55.50	1.08
c880	57.34	57.34	0.00
cht	55.42	56.54	1.12
cmb	53.25	54.45	1.20
comp	55.88	55.88	0.00
cordic	57.75	57.75	0.00
count	57.66	58.48	0.82
f51m	57.89	57.89	0.00
lal	53.68	54.87	1.19
parity	58.76	58.76	0.00
rot	55.28	56.38	1.10
term1	55.98	55.98	0.00
average	56.04	56.62	0.58

6 Conclusions

In this paper, a power reduction algorithm by single gate resizing and multiple gate resizing is presented. A path-oriented time slack calculation algorithm with sensitizability taken into account is proposed. For single gate resizing, a gain function which takes into account the fanin transition density and the number of noncritical paths passing through a gate is defined to guide the gate selection for resizing. For multiple gates resizing, simultaneously down-sizing the gate with high fanin transition density coupled with up-sizing the gate with low fanin transition density is used in reducing the total dissipated power without increasing the circuit delay.

References

[BHM94] R.I. Bahar, G.D. Hachtel, E. Macii, and F. Somenzi, "A Symbolic Method to Reduce Power Consumption of Circuits Containing False Pat hs," *Proc. ICCAD'94,* pp. 368-371, Nov. 1994

[CSB92] A.P. Chandrakasan, S. Sheng, and R.W. Brodersen, "Low-Power CMOS Digital Design," *IEEE Journal of Solid-State Circuits,* Vol. 27, No.4, pp. 473-484, April 1992.

[ChD91] H.C. Chen and H.C. Du, "Path Sensitization in Critical Path Problem," *Proc. of ICCAD'91,* pp. 208-211, Nov. 1991.

[Eve79] S. Even, *"Graph Algorithms,"* Potomac, Md.: Computer Science Press, 1979.

[LiH94] H.R. Lin and T.T. Hwang, "Dynamical Identification of Critical Paths for Iterative Gate Sizing," *Proc. of ICCAD'94,* pp.481-484, Nov. 1994.

[SGD92] A. Shen, A. Ghosh, S. Devadas, and K. Keutzer, "On Average Power Dissipation and Random Pattern Testability of CMOS Combinational Logic Networks," *Proc. of ICCAD'92,* pp. 402-407, Nov. 1992.

[SSM92] E. Sentovich, K. Singh, C. Moon, H. Savoj, R. Brayton, and A. Sangiovanni-Vincentelli, "Sequential Circuit Design Using Synthesis and Optimization," *Proc. of ICCD'92,* pp. 328-333, Oct. 1992.

[TAM93] V. Tiwari, P. Ashar, and S. Malik, "Technology Mapping for Low Power," *Proc. of 30th Design Automation Conf.,* pp.74-79, June 1993.

[TPD93] C.Y. Tsui, M. Pedram, and A.M. Despain, "Technology Decomposition and Mapping Targeting Low Power Dissipation," *Proc. of 30th Design Automation Conf.,* pp.68-73, June 1993.

[Yan91] S. Yang, "Logic Synthesis and Optimization Benchmarks User Guide Version 3.0," *Technical Report, MCNC,* Research Triangle Park, NC, January 1991.