Power Reduction by Simultaneous Voltage Scaling and Gate Sizing

Chunhong Chen and Majid Sarrafzadeh
Department of Electrical and Computer Engineering
Northwestern University, Evanston, IL 60208, USA
e-mail: {chen , majid}@ece.nwu.edu

Abstract — This paper proposes to use voltage-scaling (VS) and gate-sizing (GS) simultaneously for reducing power consumption without violating the timing constraints. We present algorithms for simultaneous VS and GS based on the Maximum-Weighted-Independent-Set problem. We describe the slack distribution of circuit, completeness of gate library and discreteness of supply voltage, and discuss their effects on power optimization. Experimental results show that the average power reduction ranges from 23.3% to 56.9% over all tested circuits.

I. INTRODUCTION

Because of the increased circuit density and speed, the power dissipation has emerged as an important consideration in circuit design. A lot of efforts on power reduction have been made at various levels of design abstraction (such as system, architectural, logic and layout levels). Considering the fact that the charging/discharging of capacitance is the most significant source of power dissipation in well-designed CMOS circuits, most research work optimize the power by reducing some or all of three factors: supply voltage, loading capacitance and switching activity. In this work, we are interested in power optimization by reducing both supply voltage and loading capacitance.

Reducing supply voltage, or *voltage scaling* (VS), promises to be an effective low-power technique because the dynamic power consumption is quadratically related to supply voltage [1-7]. The major overhead in using different supply voltages in a circuit is the additional *level converters* required at the interface and layout design. For this reason, it is advisable to restrict oneself to dual-voltage approach where only two different supply voltages are available for power optimization (throughout the paper, VS means dual-voltage approach). Unlike VS, *gate sizing* (GS) is a well-known technique which targets power optimization by reducing load capacitance. Several approaches have been published [8-12].

From a general point of view, reducing either supply voltage or physical size of a gate, at logic level, leads to the gate delay increase which implies the decreased slack time. In this sense, VS and/or GS can be effective for delay-constrained optimization only if the given circuit has significant timing slack available in some or all of its constituent gates. Because of the discrete nature of supply voltage (or gate sizes), VS or GS alone tends to leave more slacks *unutilized*, preventing effective power reduction. This fact motivates us to find the best combination or simultaneous application of VS and GS for low power design. More recently, an approach of using GS to create new timing slack for VS was reported in [13]. However, essentially GS and VS were done separately and locally in the algorithm.

In this paper, we deal with the problem of reducing power dissipation of a technology-mapped circuit under the timing constraints by using simultaneous voltage scaling and gate sizing. Our optimization problem may be described as

$$\begin{aligned} &\textbf{\textit{minimize}} \quad Power(\boldsymbol{W}, \boldsymbol{V}) \\ &\textbf{\textit{subject to}} \quad Delay(\boldsymbol{W}, \boldsymbol{V}) \le T_{spec} \\ &\qquad V_i = V_{high} \text{ or } V_{low}\,, \forall \text{ gate } i \\ &\qquad Maxsize(i) \ge w_i \ge Minsize(i)\,, \forall \text{ gate } i \end{aligned} \tag{1}$$

where both *Power* and *Delay* are functions of gate sizes ($\boldsymbol{W}$) and supply voltages ($\boldsymbol{V}$), T_{spec} is the timing constraints, V_{high} and V_{low} are two supply voltages, V_i and w_i are the supply voltage and size of gate i , respectively, and *Minsize*(i) and *Maxsize*(i) are given by gate library.

The remainder of the paper is organized as follows. Section 2 discusses the delay and power modeling with both VS and GS. Section 3 describes the basic algorithm for VS and GS based on the *Maximum-Weighted-Independent-Set* (MWIS). In Section 4, we investigate the simultaneous VS and GS for power optimization. Finally, experiment and conclusion are given in Section 5 and 6, respectively.

II. TIMING AND POWER MODELS

In this section, we will give the timing and power models and discuss the delay/power change with VS and/or GS.

A. Timing Model

In most standard-cell libraries, the gate delay is defined as

$$d_i = \tau_i + c_i \frac{C^i_{Load}}{w_i} \tag{2}$$

where τ_i is the intrinsic delay, w_i and C^i_{Load} are size and load capacitance of gate i respectively, and c_i is a constant. The load drive capability of gate i increases with w_i . The internal capacitance[1] of gate i, however, varies almost linearly with w_i . These together keep τ_i almost independent of w_i . C^i_{Load} is determined by the size of fanout gates and wiring capacitance, i.e.,

$$C^i_{Load} = C_{wire} + c \sum_{j \in FO(i)} w_j \tag{3}$$

where $\boldsymbol{FO}(i)$ is the set of fanouts of gate i, and c is a constant. When ignoring the wiring capacitance, (2) can be rewritten as

$$d_i = \tau_i + k_i \sum_{j \in FO(i)} w_j / w_i \tag{4}$$

where $k_i = c \cdot c_i$. Basically, (4) indicates that a larger gate is required for the delay reduction if it drives more fanouts. Furthermore, it has been shown in [14] that the gate delay at supply voltage V_{dd} is approximately proportional to $kV_{dd}/(V_{dd} - V_t)^2$, where

[1] The internal capacitance includes the internal cell wiring, parasitic and internal channel capacitance. For good cell layout, we can assume that the internal capacitance is dominated by effects proportional to cell size.

V_t is the threshold voltage, and k is a constant. Assuming d_i in (4) is the delay at V_{high}, the gate delay with size w_i and supply voltage V_i is given by

$$d_i(w_i, V_i) = (\tau_i + k_i \sum_{j \in FO(i)} w_j / w_i) \cdot \alpha_i \qquad (5)$$

$$\text{where} \quad \alpha_i = \frac{V_i}{(V_i - V_t)^2} \cdot \frac{(V_{high} - V_t)^2}{V_{high}}$$

For the purpose of VS, V_i can be either V_{high} or V_{low} . From (5), reducing supply voltage results in increased delay of the gate, while reducing gate size does not always degrade the delay. The reason is that the loading and, hence, the delay of its fanins decreases with the reduced size of this gate.

B. Power Model

The average dynamic power consumption for gate i is given by

$$P_i = 0.5 f \cdot V_i^2 \cdot E_i \cdot (C_{\text{int}}^i + C_{Load}^i) \qquad (6)$$

where f is the clock frequency, V_i is the supply voltage, E_i is the switching activity, $C_{\text{int}}^i = c \cdot w_i$ is the internal capacitance of gate i, and C_{Load}^i is as defined earlier. It can be seen that reducing the size of gate i leads to the saved power consumption of both gate i itself and its fanin gates.

C. Weight Function on a Single Gate

Having established the relations between delay/power consumption and both of gate size and supply voltage, we next discuss the possible delay change and power saving when GS or VS is performed on a single gate. Let us first look at GS by considering a gate i with n fanins as shown in Fig. 1. If gate i is down-sized by $\Delta w_i = w_i - w_{i'} > 0$, where i' is the largest gate (with the same function as i) whose size is smaller than w_i in library, from (5), its delay increases by $\Delta w_i \cdot \alpha_i \cdot k_i \cdot \Sigma w_j / (w_i \cdot (w_i - \Delta w_i))$ and the delay of its m-th fanin decreases by $\Delta w_i \cdot \alpha_{jm} \cdot k_{jm} / w_{jm}$, $1 \le m \le n$. The net effect of GS on delay will be the delay penalty which, in the worst case, is given by

$$\Delta d_i^{GS} = \frac{k_i \sum_{j \in FO(i)} w_j}{w_i (w_i - \Delta w_i)} \cdot \Delta w_i \cdot \alpha_i - \min_{1 \le m \le n} \frac{k_{j_m} \cdot \Delta w_i}{w_{j_m}} \cdot \alpha_{j_m} \qquad (7)$$

The negative delay penalty implies that the delay improves with gate down-sizing. The power reduction due to down-sized gate i is given as:

$$\Delta P_i^{GS} = 0.5 f \cdot c \cdot \Delta w_i \cdot (V_i^2 \cdot E_i + \sum_{m=1}^{n} V_{j_m}^2 \cdot E_{j_m}) \qquad (8)$$

where the first term is the power saving of gate i itself, and the second term accounts for the power reduction of its fanins due to their reduced load capacitance.

On the other hand, if we do VS for gate i, the resulting delay penalty, independent of its fanins, can be obtained, again from (5), as:

$$\Delta d_i^{VS} = (\tau_i + k_i \sum_{j \in FO(i)} w_j / w_i) \cdot (\frac{V_{low}}{(V_{low} - V_t)^2} \cdot \frac{(V_{high} - V_t)^2}{V_{high}} - 1) \qquad (9)$$

From (6), the resulting power reduction is:

$$\Delta P_i^{VS} = 0.5 f \cdot (V_{high}^2 - V_{low}^2) \cdot E_i \cdot (C_{\text{int}}^i + C_{Load}^i) \qquad (10)$$

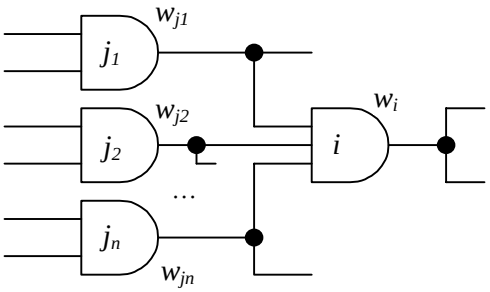

Fig. 1. Delay/power with gate sizing

If we define the *weight function*, g_i , for gate i as the average power saving per unit delay penalty, we have

$$\left.\begin{aligned} g_i^{VS} &= \Delta P_i^{VS} / \Delta d_i^{VS} \qquad \text{and} \\ g_i^{GS} &= \Delta P_i^{GS} / \Delta d_i^{GS} \qquad , \text{if } \Delta d_i^{GS} > 0 \end{aligned}\right\} \qquad (11)$$

for VS and GS, respectively. It should be noticed that Δd_i^{GS} may be non-positive. The negative value of Δd_i^{GS} indicates that down-sizing of gate i will lead to the decrease of both power and delay. In this case, one can obtain the power reduction by GS on this gate without delay penalty. This leads us to define: $g_i^{GS} = \Delta P_i^{GS} \cdot (M - \Delta d_i^{GS})$, if $\Delta d_i^{GS} \le 0$, where M is a positive constant large enough to separate the case with positive Δd_i^{GS} from that with non-positive Δd_i^{GS}. Intuitively, gates with high weight are better candidates for VS or GS.

III. Basic Algorithm

Traditionally, a technology-mapped circuit is modeled as a *directed acyclic graph*, $\boldsymbol{G}$, where each node (edge) corresponds to a gate (signal net) of the circuit. If the circuit initially meets the timing constraints, we have slack time $s(i) \ge 0$ for each node i . The problem is how to assign the slacks to nodes/edges such that the initial slacks can be fully exploited for power optimization [15]. A typical approach for the slack assignment is the *Zero-Slack-Algorithm* [16]. However, the algorithm is not able to take into account the discrete nature of node delay in VS/GS technique. In this section, we will describe a basic algorithm for VS and GS. First, we have the following definitions.

Definition 3.1 A gate (node) i is called *resizable* if (a) $w_i >$ *Minsize*(i) , where w_i is the current size of gate i , and *Minsize*(i) is the smallest size of all gates, in gate library, which have the same function as gate i, and (b) $s(i) \ge \Delta d_i^{GS}$, where $s(i)$ is the slack of gate i and Δd_i^{GS} is the delay penalty given by (7).

Definition 3.2 A gate (node) i is called *scalable* if (a) $V_i = V_{high}$, where V_i is the supply voltage of gate i, and (b) $s(i) \ge \Delta d_i^{VS}$, where Δd_i^{VS} is the delay penalty given by (9).

Definition 3.3 A *transitive closure graph* $\boldsymbol{G_t} = (\boldsymbol{Q_t}, \boldsymbol{E_t})$ of graph $\boldsymbol{G}$ is a directed graph such that there is an edge $(x, y) \in \boldsymbol{E_t}$ if and only if there is a directed path from node x to y in $\boldsymbol{G}$.

Definition 3.4 An *object graph*, $\boldsymbol{G_q} = (\boldsymbol{Q}, \boldsymbol{E_q})$, is an induced subgraph of $\boldsymbol{G_t}$ on a subset $\boldsymbol{Q} \subseteq \boldsymbol{Q_t}$ such that there is an edge $(x, y) \in \boldsymbol{E_q}$ if $(x, y) \in \boldsymbol{E_t}$, $\forall x, y \in \boldsymbol{Q}$.

Let Q_r be the set of *resizable* nodes. Any node $i \in Q_r$ may be singly down-sized without violating the timing constraints. In general, however, not all nodes in Q_r can be down-sized at the same time. The reason is that, once a gate is down-sized, the slack of other nodes in Q_r may be reduced and, hence, they may turn out to be no longer *resizable*. Similarly, if we use Q_s to represent the set of *scalable* nodes, not all nodes in Q_s can be selected to work at V_{low} without violating the timing constraints. Formally, we have the following lemmas (their proofs are omitted for brevity):

Lemma 3.1 The *Maximum-Independent-Set* (MIS) in the object graph on Q_r (or Q_s) is the maximum number of nodes which can be down-sized (or voltage-scaled) simultaneously with a guarantee that the timing constraints are met.

Lemma 3.2 When each node in the object graph is associated with its *weight function*, the maximum power reduction can be achieved if we select all nodes in the *Maximal-Weighted-Independent-Set* (MWIS) of the object graph on Q_r (or Q_s) for gate sizing (or voltage scaling), while maintaining the timing performance. The power reduction is *maximum* in the sense that no other subsets can generate more power saving.

Each time all nodes in MWIS are down-sized (or voltage-scaled), the node slack, weight and object graph are updated and the new MWIS is found again. This process repeats until Q_r (or Q_s) is empty. It should be pointed out that the MWIS problem is NP-complete on general graphs. It is, however, polynomial-time solvable for transitive graphs [17]. A formal description of the GS algorithm is given below (the VS-algorithm is similar and hence omitted).

GS-Algorithm (*circuit, timing-constraints, gate library*)
begin
  calculate the node delay, slack for all nodes in the circuit (**G**);
  identify the set of resizable nodes, Q_r ;
  do {
    assign the weight function, g_i^{GS} *, to each node* $i \in Q_r$;
    construct the object graph, G_q *, on* Q_r ;
    find MWIS of G_q ;
    down-size each node in MWIS ;
    update the node slack and Q_r ;
  } **while** ($Q_r \neq \varnothing$)
end

Experimental results and further discussions on the above algorithm (together with the VS-algorithm) will be given later on.

IV. Combination of Voltage Scaling and Gate Sizing

While the VS is basically related to supply voltages, the GS is based on gate library. One can't conclude that one is more effective than the other without looking at given supply voltage, gate library and the circuit to be optimized. In this section, we deal with the simultaneous VS and GS problem, and discuss how both of VS and GS are affected by the specific circuit, supply voltage and gate library.

A. Simultaneous VS and GS

The VS and GS can be combined in one of three ways: VS followed by GS, GS followed by VS and simultaneous VS and GS. From discussions in Section 3, it is straightforward to carry out first two combinations. In order to perform simultaneous VS and GS, we need to construct the object graph on $Q = Q_r \cup Q_s$. Particularly, if any node i is both *resizable* and *scalable* (i.e., $i \in Q_r \cap Q_s$), it is assigned the weight of $max\{g_i^{VS}, g_i^{GS}\}$ and down-sized or voltage-scaled accordingly so long as it is in MWIS of the object graph. By modifying the GS algorithm, we give the algorithm for simultaneous VS and GS as follows.

SIMULtaneous-VS-and-GS-Algorithm (*circuit, timing-constraints, gate library, two supply voltages*)
begin
  calculate the delay, slack for all nodes in circuit (**G**);
  identify Q_r *and* Q_s ;
  do {
    for *each node* $i \in Q = Q_r \cup Q_s$
      if ($i \in Q_s - (Q_r \cap Q_s)$)
        then *weight*(i) = g_i^{VS} ; *flag*(i) = *To-Be-Scaled* ;
      end if
      if ($i \in Q_r - (Q_r \cap Q_s)$)
        then *weight*(i) = g_i^{GS}; *flag*(i) = *To-Be-Resized* ;
      end if
      if ($i \in Q_r \cap Q_s$)
        then *weight*(i) = $\max\{ g_i^{VS} , g_i^{GS} \}$;
        if ($g_i^{VS} > g_i^{GS}$)
          then *flag*(i) = *To-Be-Scaled* ;
          else *flag*(i) = *To-Be-Resized* ;
        end if
      end if
    end for
    construct the object graph, G_q *, on* $Q = Q_r \cup Q_s$;
    find MWIS *of* G_q ;
    for *each node j in* MWIS
      if (*flag*(j) = *To-Be-Scaled*)
        then *reduce the supply voltage of node j* ;
        else *down-size node j* ;
      end if
    end for
    update the node delay, slack, Q_r *and* Q_s ;
  } **while** (($Q_r \cup Q_s$)$\neq \varnothing$)
end

B. Slack Distribution of Circuit

As we mention earlier, the effectiveness of VS or GS depends upon how much slack is available for all gates of the given circuit. This availability can be approximated by the *slack distribution* which means the number of gates with specific slack (or within specific ranges of slack). To normalize the slack distribution of a circuit quantitatively, we introduce the *slack expectation* which is the average slack over all gates in the circuit, i.e.,

$$slack - \exp ectation = \frac{\sum_i s_i^N \cdot n_i}{\sum_i n_i} \qquad (12)$$

where s_i^N is the *i-th* slack value (normalized to the longest path delay) and n_i is the number of gates with slack s_i^N. Given a timing model, the *slack distribution* and, hence, *slack expectation* can be determined by both the topologic structure and timing constraints of the specific circuit.

After VS or GS optimization which generally results in the delay penalty, the number of gates with small slack increases while the number of gates with large slack decreases. This leads to the reduced *slack expectation*. Ideally, as a result of power optimization, the *slack expectation* tends to be zero. However, the discrete nature of gate library and/or supply voltage may leaves some gates with positive (or even large) slack. The reason that those with large slack can not further contribute to power reduction is because they have been scaled to V_{low} and/or down-sized to their minimum sizes. A good optimization tool should take full advantage of slacks in specific circuits.

C. Completeness of Gate Library

The GS is strongly related to the underlying gate library. In order to find the optimal solution using GS, *Chen* and *Sarrafzadeh* first proposed the notion of *complete library* in [10]. A *complete library* implies that, for each type of gate, there is sufficiently large number of cells available with different size and delay. In real designs, it is impossible and unnecessary to create a complete library. To characterize a library, we introduce the *completeness* of library which is used to measure both the number of cells and the maximum size difference between cells in the library. Quantitatively, we use the *global-completeness* of library to measure the maximum size difference for each *type* of gate:

$$global-completeness(type) = \exp(-\frac{Minsize_{type}}{Maxsize_{type} - Minsize_{type}}) \quad (13)$$

In extreme case where $Maxsize_{type} = \infty$, the global-completeness will be 1, meaning the gate library is *globally* complete. In contrast, we use the *local-completeness* to measure the number of cells in library:

$$local-completeness(type) = \exp(-\frac{1}{Num_{type}}) \quad (14)$$

where Num_{type} is the number of *type* cells in the library. In extreme case where $Num_{type} = \infty$, any size will be available ranging from $Minsize_{type}$ to $Maxsize_{type}$, making the library *locally* complete. Intuitively, high (global or local) completeness of library facilitates GS for power optimization with increased computation time.

D. Discreteness of Supply Voltage

Instead of changing gate sizes, the VS optimizes power by using two supply voltages: V_{high} and V_{low}. For each gate, the reduced V_{low} promises to achieve high power reduction at the cost of increased gate delay. However, it does not necessarily mean that the total power consumption can be improved since the number of *scalable* gates may be reduced. Therefore, the supply voltage should be selected carefully. To represent the discrete nature of supply voltage, we define the *discreteness* of supply voltage as:

$$discreteness(V_{high}, V_{low}) = \exp(-\frac{1}{V_{high} - V_{low}}) \quad (15)$$

For example, when V_{high} = 5V and V_{low} = 3.5V, *discreteness*(V_{high}, V_{low}) = 0.51, and when V_{high} = 5V and V_{low} = 2.0V, *discreteness*(V_{high} , V_{low}) = 0.72. This indicates that the latter is more "*discrete*" than the former. When most gates have large slack and, hence, are allowed to work at lower voltage for more power saving, it is preferable to use supply voltages of high *discreteness.*

V. Experiment and Discussion

We implemented our algorithms for VS, GS, and simultaneous VS and GS under SIS environment [18]. Experiments were carried out on a set of MCNC benchmark circuits using some combinations of VS and GS: single VS, single GS, VS plus GS, GS plus VS, and simultaneous VS and GS. Before running our algorithms, we performed technology mapping on the given circuit under minimum delay mode with SIS and then used this delay as the timing constraints. The power consumption was estimated based on the clock frequency of 20MHz, threshold voltage of 0.6V and supply voltage of V_{high} = 5.0V and V_{low} = 3.5V (unless otherwise stated).

First, we run our VS, GS, and simultaneous VS & GS algorithms using a standard cell library (with *global-completeness* = 0.08, and *local-completeness* = 0.78)[2]. The average power reduction[3] over all tested circuits is 6.6% for GS, 19.5% for VS and 23.3% for simultaneous VS & GS (specific data will be shown later). As an example, Fig. 2 shows the power reduction and slack distribution for circuit *9symml* before/after optimization. Before optimization, its *slack expectation* can be estimated as: $\Sigma s_i^N \cdot n_i$ / Σ n_i = (16×0.005 + 1×0.04 + 10×0.075 + 65×0.15 + 99×0.3 + 9×0.7) / 200 = 0.233. By similar calculation, the *slack expectations* after GS, VS and simultaneous VS & GS are 0.154, 0.112 and 0.103, respectively. It can be seen that the slack expectation gets smaller as more power reduction is obtained. The maximum power reduction of 16.1% is achieved by the simultaneous VS and GS, and the final slack expectation reaches the minimum value of 0.103.

To see how the underlying library affects power optimization, we used different libraries with different *completeness* for the testing purpose. Table I summarizes the results with four libraries (library A, B, C and D). Library A is the least complete library with *global completeness* of 0.08 and *local completeness* of 0.78. Library D is the most complete one of four libraries. It can be seen that, for most circuits (except *i3*), the GS is less effective than VS when library A is used. The effectiveness of GS, however, improves as a more complete library is used. For example, in the case of library D, the average power reduction of as high as 51.2% is achieved using GS

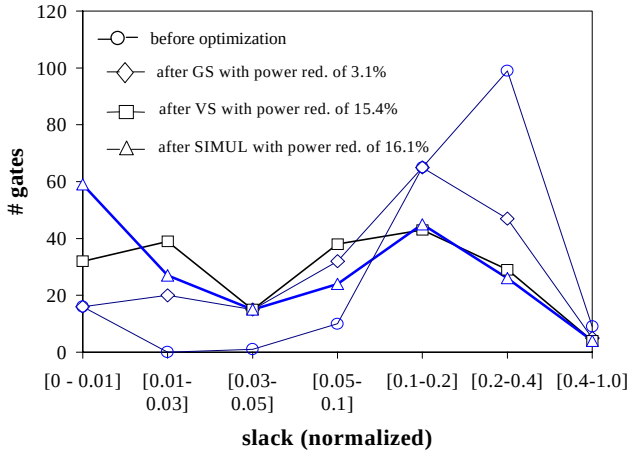

Fig. 2. Slack distribution and power reduction for circuit *9symml*

[2] These are average values over all types of gates.
[3] In all experimental results, the power consumption includes the power penalty due to level converters [5].

TABLE I

POWER REDUCTION (%) WITH DIFFERENT GATE LIBRARIES

(USING V_{high} = 5.0 V AND V_{low} = 3.5 V)

Circuit	Slack Expec-tation (Initial)	VS	Library A Global-completeness: 0.08 Local-completeness: 0.78			Library B Global-completeness: 0.37 Local-completeness: 0.85			Library C Global-completeness: 0.08 Local-completeness: 0.94			Library D Global-completeness: 0.74 Local-completeness: 0.94		
			GS	SIMUL	CPU*	GS	SIMUL	CPU	GS	SIMUL	CPU	GS	SIMUL	CPU
9symml	0.233	15.4	3.1	16.1	1.68	17.0	27.2	1.80	4.7	18.1	2.60	50.7	55.8	8.70
C1908	0.217	18.0	5.9	18.9	117.7	24.5	30.0	136.6	8.8	21.8	256.6	56.4	57.3	1485.7
C880	0.513	26.3	6.6	31.0	17.23	23.8	41.9	24.08	9.1	32.7	32.38	54.1	62.0	122.82
apex6	0.584	30.5	6.6	35.1	40.47	25.0	47.2	42.90	9.4	37.0	45.83	56.6	68.6	75.97
apex7	0.357	22.1	7.4	26.0	3.03	21.5	35.9	3.47	9.7	27.9	4.63	53.7	60.1	10.68
b9	0.430	22.3	7.9	26.4	1.20	22.4	36.4	1.40	10.4	28.5	1.92	53.4	59.7	4.40
c8	0.166	10.0	7.8	16.3	0.68	19.7	28.2	0.95	9.1	17.9	1.25	55.7	59.7	3.52
frq1	0.386	20.3	5.8	24.5	1.53	20.1	34.8	1.93	7.4	23.4	2.18	52.2	57.6	6.97
frq2	0.352	20.5	6.4	24.4	52.72	20.3	35.4	49.15	9.0	23.6	55.73	54.8	61.8	100.35
i1	0.430	18.9	11.6	26.7	0.20	25.7	38.1	0.27	13.4	28.5	0.33	48.7	55.8	1.07
i3	0.022	0.1	1.4	1.5	1.98	10.1	10.2	2.42	3.2	3.3	2.47	20.6	20.7	6.02
i5	0.442	21.2	7.3	25.4	2.27	22.0	35.7	2.95	11.7	28.6	3.33	50.5	57.5	10.50
i6	0.322	14.4	7.3	21.2	3.37	11.1	23.0	3.80	7.7	20.9	4.48	49.8	53.0	10.23
i7	0.386	20.1	6.4	24.2	10.42	18.4	33.5	12.25	7.1	24.9	12.18	52.9	58.5	25.25
rot	0.529	30.2	7.3	33.4	56.38	23.2	44.6	60.87	9.7	36.0	70.7	55.2	66.4	126.93
term1	0.296	21.5	7.4	21.5	2.62	21.4	33.6	3.00	9.4	25.9	4.33	54.2	56.1	11.33
Average	0.354	19.5	6.6	23.3	19.59	20.4	33.5	21.74	8.7	24.9	31.31	51.2	56.9	125.65

* This is the CPU time in seconds using SIMULtaneous algorithm on a SUN SPARCstation 5 with 32MB RAM.

TABLE II

POWER REDUCTION (%) WITH DIFFERENT SUPPLY VOLTAGES**

(USING LIBRARY A)

Circuit	Slack Expectation (Initial)	V_{high} = 5.0 V V_{low} = 3.5 V Discreteness: 0.51		V_{high} = 5.0 V V_{low} = 2.8 V Discreteness: 0.63		V_{high} = 5.0 V V_{low} = 2.0 V Discreteness: 0.72		V_{high} = 5.0 V V_{low} = 1.0 V Discreteness: 0.78	
		VS	SIMUL	VS	SIMUL	VS	SIMUL	VS	SIMUL
9symml	0.233	15.4	16.1	12.8	13.9	9.9	10.2	0.0	3.1
C1908	0.217	18.0	18.9	18.8	22.1	16.1	17.7	3.0	6.5
C880	0.513	26.3	31.0	39.1	43.3	38.5	40.1	8.3	14.6
apex6	0.584	30.5	35.1	40.4	44.0	38.7	41.1	4.4	9.6
apex7	0.357	22.1	26.0	25.7	32.4	26.2	28.7	0.0	7.4
b9	0.430	22.3	26.4	25.8	30.7	22.6	26.6	0.0	7.9
c8	0.166	10.0	16.3	10.4	15.7	4.8	8.3	0.0	7.8
frg1	0.386	20.3	24.5	28.5	31.5	27.2	29.0	0.0	5.8
frg2	0.352	20.5	24.4	23.3	27.7	22.3	26.9	5.1	9.0
i1	0.430	18.9	26.7	24.8	32.2	27.0	33.6	0.0	11.6
i3	0.022	0.1	1.5	0.2	1.6	0.0	1.5	0.0	1.4
i5	0.442	21.2	25.4	22.8	26.9	16.4	21.5	0.0	7.3
i6	0.322	14.4	21.2	15.5	20.2	22.1	26.2	0.0	7.3
i7	0.386	20.1	24.2	25.0	28.3	28.2	31.2	0.0	6.4
rot	0.529	30.2	33.4	37.7	40.7	37.3	40.2	8.4	13.3
term1	0.296	21.5	21.5	20.8	26.4	11.3	15.0	0.0	7.4
Average	0.354	19.5	23.3	23.2	27.4	21.8	24.9	1.8	7.9

** The CPU time in this experiment is almost the same as that in Column "*Library A*" of Table I.

algorithm. No matter what library is used, the simultaneous algorithm always leads to best results, as shown in this table. The average power reduction by the simultaneous algorithm ranges from 23.3% to 56.9% over all tested circuits, and it takes more CPU time when a more complete library is used.

Also, we tested our algorithms using different supply voltages. Table II shows the comparison of results (using library A) with four groups of supply voltages. For some circuits (such as *C880*, *apex6* and *rot*) with high slack expectation, more power reduction is achieved using more *discrete* supply voltage (such as V_{low} = 2.8V or 2.0V). In contrast, if the slack expectation of circuits (such as *9symml*, *C1908*, *c8*, *i3* and *term1*) is small, the low discreteness of supply voltage is preferred. Thus, the best supply voltage should be chosen carefully, depending on the slack distribution of specific circuits. In general, however, using too *discrete* supply voltage (e.g., V_{high} = 5.0V and V_{low} = 1.0V) is not advisable for most circuits, since it prevents most or all gates in the circuit operating at V_{low} under the given timing constraints, resulting in little or no power saving. That is what happens in the last column of Table II.

By taking a further look at how the slack distribution of a circuit contributes to power reduction, we see that the circuits with high slack expectation generally promise the significant power reduction. In Column "library A" of Table I, for example, the maximum power reduction of 35.1% occurs at circuit *apex6* whose slack expectation is 0.584 (maximum value of all circuits). On the other end of spectrum is circuit *i3* whose slack expectation is 0.022 (minimum value of all circuit) with power improvement of only 1.5%. This trend of power-slack relationship holds true even if we change gate library, supply voltage and/or algorithm, as can be seen for details in Table I and Table II.

VI. Conclusion

We have presented the first work on power reduction using simultaneous voltage scaling and gate sizing. The algorithms optimize power consumption under the given timing constraints by dealing with the Maximum-Weighted-Independent-Set problem on transitive graphs. It is demonstrated that voltage- scaling technique is related to the supply voltage, while gate- sizing technique depends on the underlying gate library. For specific circuits, their slack distribution guides what supply voltage (or gate library) should be used to promise more power reduction. It has also been shown that the proposed simultaneous voltage-scaling and gate-sizing provides globally good solutions with inexpensive computation cost.

References

[1] S. Raje and M. Sarrafzadeh, "Variable voltage scheduling", *International Symposium on Low Power Design*, pp. 9-14, April 1995.

[2] J. M. Chang and M. Pedram, "Energy minimization using multiple supply voltages ", *IEEE Transactions on VLSI Systems*, vol.5, no.4, pp.1-8, December 1997.

[3] K. Usami and M. Horowitz, "Cluster voltage scaling technique for low power design", *International Symposium on Low Power Design*, pp.3-8, April 1995.

[4] K. Usami *et al*., "Automated low power technique exploiting multiple supply voltages applied to a media processor", *Custom Integrated Circuit Conference*, pp.131-134, 1997.

[5] C. Chen and M. Sarrafzadeh, "An effective algorithm for gate-level power-delay tradeoff using two voltages", *International Conference on Computer Design*, pp.222-227, October 1999.

[6] S. Raje and M. Sarrafzadeh, "Scheduling with multiple voltages", *Integration, VLSI Journal 23*, pp.37-59, 1997.

[7] K. Usami *et al*., "Design methodology of ultra low-power MPEG4 codec core exploiting voltage scaling techniques", *ACM/IEEE Design Automation Conference*, pp.483-488, June 1998.

[8] P. Girard, C. Landrault, S. Pravossoudovitch, and D. Severac, "A gate resizing technique for high reduction in power consumption", *International Symposium on Low Power Electronics and Design*, pp.281-286, 1997.

[9] H.R. Lin and T. Hwang, "Power reduction by gate sizing with path-oriented slack calculation", *IEEE ASP-DAC'95/CHDL'95/VLSI'95*, pp.7-12, June 1995.

[10] D.S. Chen and M. Sarrafzadeh, " An exact algorithm for low power library-specific gate re-sizing", *ACM/IEEE Design Automation Conference*, pp.783-788, June 1996.

[11] O. Coudert, "Gate sizing: a general purpose optimization approach", *IEEE European Design & Test Conference*, pp.214-218, March 1996.

[12] O. Coudert, R. Haddad, and S. Manne, "New algorithms for gate sizing: a comparative study", *ACM/IEEE Design Automation Conference*, pp.734-739, June 1996.

[13] C. Yeh, M. Chang, S. Chang and W. Jone, "Gate-level design exploiting dual supply voltages for power-driven applications", *ACM/IEEE Design Automation Conference*, pp.68-71, June 1999.

[14] A.P. Chandrakasan, S. Sheng, and R.W. Brodersen, "Low-power CMOS digital design", *Journal of Solid-State Circuits*, vol.27, no.4, pp.473-484, April 1992.

[15] C. Chen and M. Sarrafzadeh, "Slack equalization algorithm: precise slack distribution for low-level synthesis and optimization", *International Workshop on Logic Synthesis*, pp.190-192, June 1999.

[16] R. Nair, C.L. Berman, P.S. Hauge, and E.J.Yoffa, "Generation of performance constraints for layout", *IEEE Transactions on Computer-Aided Design*, CAD-8(8), pp.860-874, 1989.

[17] R.H. Moehring, "Graphs and orders: the role of graphs in the theory of ordered sets and its application", *Published by D. Reidal Publishing Company, edited by I. Rival, New York and London*, pp.41-101, May 1984.

[18] E.M. Sentovish *et al*., "SIS: a system for sequential circuit synthesis", *Technical Report UCB/ERL M92/41, Univ. of California, Berkeley*, May 1992.