Coarse-Grain MTCMOS Sleep Transistor Sizing Using Delay Budgeting

Ehsan Pakbaznia and Massoud Pedram
University of Southern California
{pakbazni, pedram}@usc.edu

Abstract

Power gating is one of the most effective techniques in reducing the standby leakage current of VLSI circuits. In this paper we introduce a new approach for sleep transistor sizing which minimizes the total sleep transistor width for a coarse-grain multi-threshold CMOS circuit assuming a given standard cell and sleep transistor placement. First, the circuit is decomposed into a set of modules, each containing the set of logic cells that are closest to a sleep transistor cell. Next given an upper bound on the overall circuit speed degradation, the global timing slack is distributed among different clusters using a delay-budgeting. The slack distribution result is then used to size the sleep transistors such that the total sleep transistor width is minimized while accounting for the parasitic resistances of the virtual ground net. Results show that the proposed sizing algorithm produces sleep transistor sizes that are 40% smaller than those produced by previous approaches.

1. Introduction

Multi-threshold CMOS (MTCMOS) technology provides low leakage and high performance operation by utilizing high speed, low Vt transistors for logic cells and low leakage, high Vt devices as sleep transistors. Sleep transistors disconnect logic cells from the power supply and/or ground to reduce the leakage in the sleep mode. There is a performance degradation associated with the sleep transistor insertion. This is due to the IR-drop across the MTCMOS cells in the active mode of operation. For a fixed placement, the amount of the performance degradation depends on the size of the MTCMOS switch cells. The larger the sleep transistors are, the lower the performance degradation is. However, the amount of the power consumption will increase with the size of the sleep transistors. Therefore, there is a trade-off between the amount of the performance degradation and the power consumption of the sleep transistors in an MTCMOS circuit. This makes MTCMOS cell sizing one of the most important issues in the coarse-grain MTCMOS design flows.

In some applications performance is too critical and the designer cannot afford any performance degradation due to MTCMOS. In [1] authors propose to separate timing critical standard cells from the non-critical ones by placing them in different rows and by doing the power gating only for the non-critical standard cell rows. They have shown that a high leakage saving can be achieved while losing a small amount of performance. In this paper, we assume that MTCMOS is applied to all standard cell rows. Furthermore, no rail sharing is assumed for the neighbor rows.

There have been several works addressing sleep transistor sizing for MTCMOS circuits [2]-[8]. In [3] cells inside the circuit are clustered such that their switching current profiles are mutually exclusive. In [4] cells in the circuit are clustered using bin-packing or set partitioning to reduce total sleep transistor width. In [5] virtual rail routing is employed to use a distributed sleep transistor network (DSTN) in order to reduce the total sleep transistor size. In [6] and [7], the authors propose algorithms to calculate the drop voltages in a distributed sleep transistor network and use that in sizing the sleep transistors.

Most of the clustering-based sleep transistor sizing algorithms, propose special type of circuit clustering to reduce the total sleep transistor width. In order to implement these approaches, logic cells inside the same cluster need to be placed close together; however, since most of the state of the art industrial flows use timing-driven placement, MTCMOS circuit clustering will result in performance degradation. On the other hand DSTN-based sleep transistor sizing approaches do not use the total available slack optimally. Therefore, they tend to oversize the sleep transistors [5]-[7]. In this paper, we present a delay-budgeting algorithm to size the sleep transistors in a circuit. We assume that the placement of the logic cells and sleep transistor cells are known and given. We then propose a delay-budgeting algorithm to optimally use the total available slack and size the sleep transistors optimally.

The remainder of this paper is organized as follows. Section 2 talks about the coarse-grain MTCMOS layout style. Section 3 describes the proposed sizing algorithm while section 4 shows the results obtained by applying the sizing algorithm. Section 5 concludes the paper.

2. Coarse-Grain MTCMOS Layout

Figure 1 shows a typical standard cell row in a coarse-grain MTCMOS design which comprises of standard cells and an MTCMOS sleep transistor (which is also included in the cell library as a standard cell). There are two types of coarse-grain MTCMOS switches: *headers* and *footers*. A footer cell basically consists of an NMOS sleep transistor which is used to disconnect the true V_{SS} (TV_{SS}) net from the virtual V_{SS} (VV_{SS}) rail. A header cell, however, consists of a PMOS sleep transistor used to disconnect the true V_{DD} (TV_{DD}) net from the virtual V_{DD} (VV_{DD}) rail. From here on wherever we talk about MTCMOS switch cells, or sleep transistors, footer cells are intended. Discussions about footer cells, with obvious modifications, are also applicable to header cells.

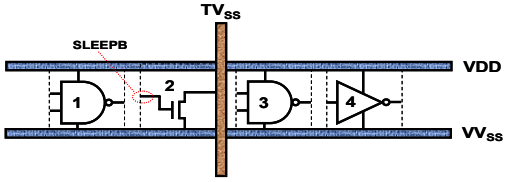

Figure 1: Portion of a typical cell row in a coarse-grain MTCMOS circuit, also showing a sleep transistor.

To make the coarse-grain MTCMOS flow better adapted to the ASIC design flow, MTCMOS switch cells have to be treated as regular standard cells by the CAD tools. This requires these cells to be designed similar to the regular standard cells. More precisely, all the MTCMOS switch cells have to include power and ground rails that are aligned with the corresponding rails of other standard cells. In addition, the switch cells must also have the same height as any other library cell. Figure 2 shows typical layout of coarse-grain header and footer cells. It can be seen from the figure that both header and footer cells have separate V_{SS} and V_{DD} rails similar to all the other standard cells.

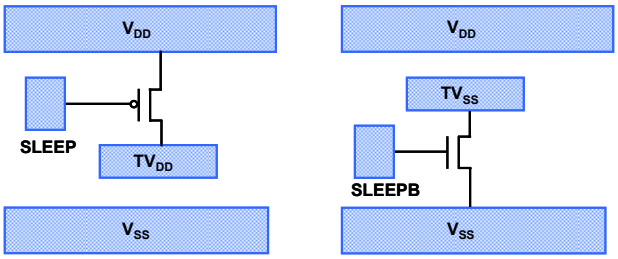

Figure 2: Typical layout styles of coarse-grain MTCMOS switches: (left) header, (right) footer.

The V_{DD} rail in the footer cell is not connected to anything inside the cell; in contrast, the V_{SS} rail is connected to the TV_{SS} pin through an NMOS transistor. The V_{SS} rail of each footer cell will be connected to the V_{SS} rail of the row that this footer cell belongs to. The TV_{SS} pin, on the other hand, will be connected to the true ground mesh which will be routed in a separate metal layer, e.g., M4. Therefore, the V_{SS} rail (V_{SS} net) of the footer cell becomes part of the VV_{SS} net of the cell row after the footer cell is inserted into the row. Each MTCMOS switch cell contains an input pin and an output pin which are used for cell characterization. The input pins for the header and footer cells are SLEEP and SLEEPB ($\overline{\text{SLEEP}}$), respectively. These pins are the control pins to turn the switch ON and OFF. The output of the footer cell is V_{SS}, while the input is the TV_{SS}.

MTCMOS switch cells can be placed in many different fashions among the cells in a circuit. Figure 3 shows the *column-aligned* sleep transistor placement style.

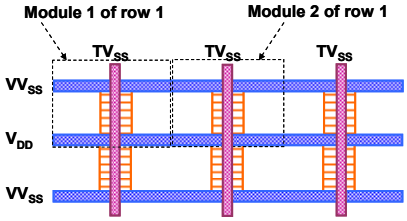

Figure 3: Column-aligned sleep-transistor placement.

The dashed boxes represent MTCMOS switch cells. All the other standard cells are assumed to be placed in the blank area between switch cells. The TV_{SS} mesh lines are also shown in the figure. They will be used for routing the TV_{SS} pins in various switch cells. Because of its simple power/ground network routing strategy, it is desirable to uniformly distribute the switch cells on each standard cell row and to have them aligned vertically one under the other as we traverse different cell rows.

The switch cell placement problem may be formulated and solved as an optimization problem by itself; however, we assume here that the placement of the logic and MTCMOS switch cells is fixed and given. We present an algorithm to optimally size the sleep transistors for the given placement.

3. Sleep Transistor Sizing with Delay Budgeting

The notion of a *module* associated with each sleep transistor is explained with the help of Figure 3. A module is defined based on the existing cell placement and the location of the TV_{SS} lines (or alternatively, the sleep transistor cell that lies underneath this line) over the standard cell layout. In particular, module (r,i) denotes the module that is formed around the i^{th} sleep transistor in the r^{th} row of the standard cell layout. The cells belonging to this module are those that are in the r^{th} row and are closest in distance to the i^{th} sleep transistor in that row. We ignore the VV_{SS} rail resistance between the cells inside each such module. The VV_{SS} nodes of different modules are connected through the VV_{SS} rail, whose resistance is taken into account by considering a resistor between the VV_{SS} nodes of two adjacent modules as shown in Figure 4 by $r_{V_{SS(r,i-1)}}$ and $r_{V_{SS(r,i)}}$. In this figure, $I_{M_{(r,i)}}(t)$ and $I_{st_{(r,i)}}(t)$ denote, respectively, the discharging current of module $M_{(r,i)}$ and the current flowing through the sleep transistor corresponding to this module as a function of time. Note for the rest of this paper, each row is considered separately from the other rows in the circuit; therefore, the index r can be eliminated for simplicity. For example, M_i represents the i^{th} module of a typical row in the circuit.

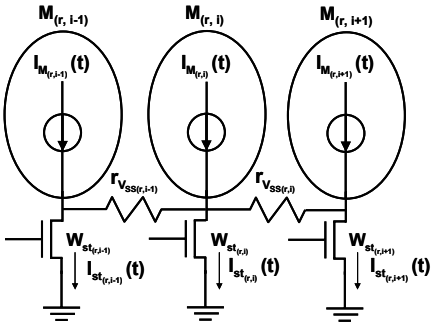

Figure 4. Sleep transistors, the corresponding modules, and parasitic model of the VV_{SS} rail.

During the active operation, sleep transistors work in the linear mode, and each sleep transistor may be replaced by its equivalent linear region resistor. For the i^{th} sleep transistor, of a typical row, the value of this resistor is calculated as:

$$R_{st_i} = \left(\mu_i \left(\frac{W_{st_i}}{L} \right) C_{ox} \left(V_{DD} - V_{tH} \right) \right)^{-1}$$

Current state-of-the-art sleep transistor sizing algorithms [6]-[7] minimize the total sleep transistor width subject to a maximum IR voltage drop on the virtual node of each MTCMOS switch cell. In these approaches, the DC noise constraint for the virtual node of a MTCMOS switch is somehow related to the tolerable delay increase in the circuit.

In fact, none of these approaches talk about selecting the drop constraints optimally. The most trivial way that is used is to uniformly slow down all the modules which results in a single drop constraint for all modules. In reality, using a single maximum IR voltage drop value on all virtual nodes is over constraining the problem and indeed avoidable. Instead, one would like to set the DC noise constraint for the virtual node of each MTCMOS switch based on the minimum tolerable delay increase (i.e., the positive timing slack) for any logic cell in the corresponding module. The voltage drop allocation on the virtual nodes of the MTCMOS switches should thus be closely related to the timing slack allocation to individual cells in the circuit. In the next section, we provide an example to show that for a specified maximum delay penalty for the whole circuit, the manner in which the positive timing slack is distributed among different modules in the circuit greatly affects the sleep transistor sizing solution. Solving this delay budgeting problem and combining it with sleep transistor sizing is precisely the contribution of the present paper.

3.1. Background and Motivational Example

Consider a logic cell located in the i^{th} module, M_i, of a typical row of a CMOS circuit. Let d denote the 50% propagation delay of this cell. To a first order, we have:

$$d \propto \frac{C_L V_{DD}}{(V_{DD} - V_{tL})^{\alpha}} \quad (1)$$

where C_L denotes the load capacitance of this cell, V_{tL} is the threshold voltage of low-Vt devices in the cell, and α is the velocity saturation index, which models the short channel effect [9]. Suppose this cell is placed in module M_i in the MTCMOS circuit. Let d' denote the propagation delay of this cell in the MTCMOS circuit. Again we have:

$$d' \propto \frac{C_L V_{DD}}{(V_{DD} - v_i - V_{tL})^{\alpha}} \quad (2)$$

where v_i is the voltage of the VV_{SS} node associated with module M_i, the module that this cell belongs to. Using Taylor series expansion, the delay increase is calculated as [8]:

$$\Delta d = d' - d \propto \frac{v_i}{V_{DD} - V_{tL}} d \quad (3)$$

It can be seen that the degree of the *delay degradation ratio* (DDR), i.e., *Δd/d*, for each cell is directly proportional to the voltage drop at the VV_{SS} node of the module that this cell belongs to. In order to achieve a fixed given DDR value for a circuit, it is enough to have a set of constraints guaranteeing that none of the v_i voltages exceed a fixed voltage value, $V_{i\text{-}max}$. This is the approach that most of the conventional methods use to obtain the voltage drop constraints for different modules. We show that the voltage drop constraints obtained using this approach are not the optimal values. The best way to explain this observation is with the aid of an example.

Consider the circuit shown in Figure 5. The circuit consists of four inverters and two sleep transistors modeled as resistors in the figure. Each inverter drives a FO4 load. We divide this circuit to two modules, M_1 and M_2. Module M_1 comprises of the first two inverters, i.e., inverters with size 1 and 4 while module M_2 consists of the last two inverters, i.e., inverters with size 16 and 64.

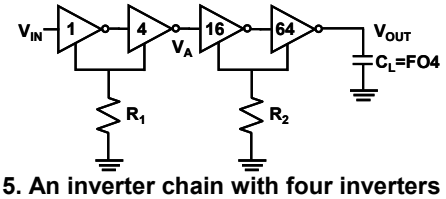

Figure 5. An inverter chain with four inverters and two modules. Sleep transistors are replaced by resistors.

One sleep transistor is used per module in the MTCMOS circuit. When $R_1=R_2=0$, using a 65nm CMOS process technology deck, the total V_{IN}-V_{OUT} low-to-low propagation delay is 103ps. Table 1 shows the propagation delay share and the peak discharge current value for each module in the normal operation mode (as opposed to the sleep mode).

Table 1. Propagation delay and peak current values for the two modules of Figure 5.

Module	Module Delay (pico sec)	Module Peak Current (mA)
M_1	46	0.3
M_2	57	4.65

We assume that after inserting sleep transistors a maximum DDR of 10% is acceptable, which gives us a total positive timing slack of 10.3ps. This slack can be distributed between the two modules in many different ways. Depending on how this slack is distributed between the two modules, different maximum voltage drop constraints and different total sleep transistor widths are obtained. Table 2 shows some of these choices. It can be seen that how precisely the total slack is distributed between the two modules will have a large impact on the total sleep transistor size (which is proportional to summation of inverse resistance values).

Table 2. Total sleep transistor conductance as a function of the timing slack distribution for two modules.

Circuit	Module Delay (ps)	Total Delay (ps)	Sleep Tx Resistance (Ω)	$\sum R_i^{-1}$ (Ω^{-1})
CMOS	T_{M1}=46 T_{M2}=57	103	R_1=0 R_2=0	—
MTCMOS	T_{M1}=50.6 T_{M2}=62.7	113.3	R_1=250 R_2=9	0.1151
	T_{M1}=52 T_{M2}=61.3	113.3	R_1=330 R_2=2	0.5030
	T_{M1}=48 T_{M2}=65.3	133.3	R_1=110 R_2=25	0.0491

The first row in the MTCMOS section in the table corresponds to the uniformly distributed slack between both modules, i.e., both modules have the same DDR of 10%. In this case $\sum R_i^{-1}$ value is $0.1151\Omega^{-1}$. The second row in the MTCMOS section corresponds to the case when most of the total available slack (approximately 80%) is given to M_1 and the rest (20%) is given to M_2. In this case $\sum R_i^{-1}$ value is $0.5030\Omega^{-1}$ which is much more than the first case. Finally the third row in the MTCMOS section corresponds to the case when only 20% of the total available slack is given to M_1, and most of the total

available slack is reserved for M_2. This case results in the minimum $\sum R_i^{-1}$ value, which is $0.0491\Omega^{-1}$.

This example clearly shows that slowing down all the modules in a circuit uniformly, i.e., with the same DDR, will not result in the minimum total sleep transistor width solution. The problem statement has to be formulated in such a way that the total available slack due to the maximum allowed DDR is distributed among different modules optimally while being aware of the discharge current of different modules. Intuitively, we should slow down modules with large amount of discharging current more than the ones with smaller amount of discharging current, current-aware optimization.

In this paper we first formulate the sleep transistor sizing problem as a delay-budgeting problem. Then we present a current-aware sizing algorithm to find the optimum solution.

3.2. Problem Formulation

Consider a combinational circuit. The timing constraints for the circuit are given as an input arrival time A_n for each primary input PI_n, and as a required arrival time R_k at each primary output PO_k. We let a_n and r_n denote the output arrival and required times of cell C_n and d_n denote the propagation delay of this cell. Knowing the primary input arrival times, we can calculate arrival time at the output of each cell as the summation of the maximum input arrival times of the cell and the cell propagation delay. Similarly, required time can be calculated knowing the required time for the primary outputs and the propagation delays of different cells in the circuit. The slack at each node is:

$$s_n = r_n - a_n \tag{4}$$

After inserting the MTCMOS cells and imposing the new required times for the primary outputs, all the arrival and required times for different nodes in the circuit will change. We show arrival time, required time and slack for the output of C_n in the MTCMOS circuit by $a_n^{'}, r_n^{'}, s_n^{'}$, respectively. Thus:

$$s_n^{'} = r_n^{'} - a_n^{'} \tag{5}$$

The propagation delay of each cell will also change. We show delay of C_n in the MTCMOS circuit by $d_n^{'}$ where:

$$d_n^{'} = d_n + \Delta d_n \tag{6}$$

Now suppose C_n is placed in module M_i in the MTCMOS circuit. From (3) we have: $\Delta d_n = \delta \times d_n$ where:

$$\delta = \frac{v_i}{V_{DD} - V_{tL}} \tag{7}$$

The arrival time for C_n in the MTCMOS circuit, $a_n^{'}$, is:

$$a_n^{'} = \max\left\{a_{fanins\ of\ C_n}^{'}\right\} + d_n^{'} \tag{8}$$

From (6) and (7), the arrival times in the MTCMOS circuit can be calculated in terms of the VV_{SS} node voltages of different modules in the MTCMOS circuit. Thus, arrival times in the MTCMOS circuit, $a_n^{'}$'s, can be written in terms of v_i's.

Required time of the output of C_n in MTCMOS circuit is:

$$r_n^{'} = \min\left\{r_{fanouts\ of\ C_n}^{'}\right\} - d_n^{'} \tag{9}$$

The delay-budgeting constraints can be written as follows:

$$s_n^{'} = r_n^{'} - a_n^{'} \geq 0; \quad 1 \leq n \leq \text{CELL\_NUM} \tag{10}$$

Where $a_n^{'}$ and $r_n^{'}$ are calculated from (8) and (9) while CELL_NUM denotes the total number of the cells (nodes) in the circuit. Since the propagation delay values for each cell in the MTCMOS case are not known and they depend on the v_i values of different modules, and since (8) and (9) include $\max\{.\}$ and $\min\{.\}$ operations, the complexity of optimizing an objective function on the domain defined by these constraints is high. To simplify the problem, we may consider only the critical timing paths when formulating the problem constraints, i.e., we get rid of the min and max operators in (8) and (9). However, the potential weakness of this approach is that the critical paths in the CMOS circuit are not necessarily the critical paths in the MTCMOS circuit [2]. Fortunately, this difficulty can be addressed by taking into account the K most critical paths in the CMOS circuit to build the set of constraints for the optimization problem.

The delay degradation of a given path π_k in the circuit due to applying power gating can be written as the summation of the delay degradations of all the cells in that path. The delay degradation for any cell C_n in the circuit can be calculated from (3) assuming that C_n belongs to M_i. Note as far as delay degradation of C_n is concerned, v_i in (3), or (7), can be calculated in terms of R_{st_i} as follows:

$$v_i = R_{st_i} I_{st_i}^{\max\left[t_{\min}^{C_n}, t_{\max}^{C_n}\right]} \tag{11}$$

Where $I_{st_i}^{\max\left[t_{\min}^{C_n}, t_{\max}^{C_n}\right]}$ is the maximum value of the current flowing through R_{st_i} during the time window that cell C_n is switching, i.e., $[t^{Cn}_{min},\ t^{Cn}_{max}]$. The delay degradation of a given path π_k can be calculated using (6), (7) and (11) [8]:

$$\Delta d_{\pi_k} = \sum_{C_n \in \pi_k} \Delta d_n = \sum_{C_n \in \pi_k} \frac{R_{st_{\theta(C_n)}} I_{st_{\theta(C_n)}}^{\max\left[t_{\min}^{C_n}, t_{\max}^{C_n}\right]}}{V_{DD} - V_{tL}} d_n \tag{12}$$

where the summation is taken over all cells in path π_k. C_n represents a cell in π_k, and $\theta(C_n)$ is the index of the module that cell C_n belongs to; e.g., if C_n is in M_i, then $\theta(C_n)=i$. Based on what we have discussed so far, the delay-budgeting based sizing problem can be formulated as follows:

Minimize $\sum_{i=1}^{M} R_{st_i}^{-1}$

s.t. :
1. $\Delta d_{\pi_k} \leq \text{DDR\_MAX}; \quad 1 \leq k \leq K$
2. $R_{st_i} I_{st_i}(t_j) \leq \text{VVSS\_MAX}; \ 1 \leq i \leq M, 1 \leq j \leq N$

where:

$$\Delta d_{\pi_k} = \sum_{C_n \in \pi_k} \frac{R_{st_i} I_{st_i}^{\max\left[t_{\min}^{C_n}, t_{\max}^{C_n}\right]}}{V_{DD} - V_{tL}} d_n; \qquad 1 \leq k \leq K, 1 \leq j \leq N$$

$$\forall i, j : I_{st_0}(t_j) = I_{st_{N+1}}(t_j) = 0 \text{ and}$$

$$I_{st_i}(t_j) = I_{M_i}(t_j) + \frac{R_{st_{i-1}} I_{st_{i-1}}(t_j)}{r_{VSS_{i-1}}} + \frac{R_{st_{i+1}} I_{st_{i+1}}(t_j)}{r_{VSS_i}} - \frac{R_{st_i} I_{st_i}(t_j)}{r_{VSS_{i-1}}} - \frac{R_{st_i} I_{st_i}(t_j)}{r_{VSS_i}}$$

Figure 6. Delay-budgeting based sleep transistor sizing problem.

In Figure 6, the clock cycle is divided into N equal time intervals and t_j denotes the beginning time of the j^{th} interval. $I_{M_i}(t_j)$ is the switching current of module M_i at time t_j. These equations implicitly construct the maximum current waveform through each sleep transistor at N timing instances in a clock cycle while considering the timing windows during which a logic cell can change its output value. The equation corresponding to $I_{st_i}(t_j)$ calculation in Figure 6 is obtained by writing the KCL equations for different nodes of the VV_{SS} rail, i.e., this equation accounts for different current flow paths in the virtual ground net through adjacent sleep transistors. The first set of the constraints are the critical path constraints, while the second set of constraints capture the maximum allowed voltage drop on the VV_{SS} rail.

3.3. Algorithm

In this section we describe a current-aware sizing algorithm (c.f. Section 3.1) which solves the sleep transistor sizing problem presented in 3.2. We can show that the first set of the constraints in Figure 6 can be written as a set of linear equations in terms of variables, R_{st_i}, as follows:

$$\sum_{i=1}^{M} a_{ki} R_{st_i} \le \text{DDR\_MAX}; \quad 1 \le k \le K \tag{13}$$

where a_{ki} is 0 if module i does not lie on the k^{th} critical path; otherwise, it is calculated by collecting all the coefficients corresponding to R_{st_i} in Δd_{π_k}.

Definition 1: At any given step of the sizing algorithm, the most critical module (MCM) is the module with the maximum delay contribution to the K most critical paths, i.e.,

$$MCM = \arg\max_{M_i} \sum_{k=1}^{K} a_{ki} R_{st_i} \tag{14}$$

Definition 2: At any step of the algorithm the best candidate module (BCM) is defined as the module whose sleep transistor upsizing by a certain percentage will result in the largest delay improvement for unsatisfied paths.

Lemma 1: BCM is the MCM over the paths that do not meet the delay constraint, i.e.:

$$BCM = MCM\left(\left\{\pi_k \mid 1 \le k \le K,\ \Delta d_{\pi_k} > \text{DDR\_MAX}\right\}\right) \tag{15}$$

Proofs are straight-forward and omitted for brevity. Note that BCM is not unique and there can exist more than one BCM at any step of the algorithm.

Definition 3: Least-cost BCM (LBCM) is defined as the BCM whose sleep transistor upsizing will result in the minimum increase in the objective function in Figure 6.

If there is only one BCM, then we have LBCM=BCM.

Lemma 2: LBCM can be found as:

$$LBCM = \arg\min_{M_i = BCM} \sum_{\substack{k=1 \\ \Delta d_{\pi_k} > \text{DDR\_MAX}}}^{K} a_{ki} \tag{16}$$

Lemma 2 makes the sizing algorithm be aware of the discharging current of the module (current-aware algorithm). From the discussion presented above, we propose the following sleep transistor sizing algorithm. At the beginning we use an algorithm similar to the one presented in [7], *Slp_Initialize*, in order to satisfy the second set of the constraints in Figure 6 (i.e., the virtual ground voltage upper bound). The resulted R_{st_i} values will typically be too large to meet the first set of constraints in Figure 6 (i.e., the timing constraints). They are thus fed into the main sleep transistor sizing algorithm, *Slp_Sizing*, which will iteratively size up the sleep transistors until all the timing constraints are met.

Algorithm: Slp_Initialize(I_{Mi}(t), VVSS_MAX)

1: /*Initializing variables*/
2: **for** *i=1* to M **do**
3: $R_{st_i} = R_{MAX}$;
4: **end for**
5: calculate $I_{st_i}(t_j)$ and $v_i(t_j) = R_{st_i} I_{st_i}(t_j)$ for all *i, j* ;
6: **while** ($v_i(t_j)$ > VVSS_MAX for some *i* or *j*) **do**
7: M_m=FindMinModule{VVSS_MAX - $v_i(t_j)$};
8: $R_{st_m} = VVSS\_MAX / I_{st_m}(t_j)$ for all *j*;
9: update $I_{st_i}(t_j)$ and $v_i(t_j) = R_{st_i} I_{st_i}(t_j)$ for all *i, j;*
10: **end while**
11: **return** R_{st_i} for all *i;*

Figure 7. Initializing optimization variables and satisfying the second set of constraints.

At each iteration step, the *Slp_Sizing* algorithm checks if all the constraints are satisfied. If there is any unsatisfied constraint, the algorithm searches for the *LBCM* and reduces the corresponding resistance value by *α%,* and updates $I_{st_i}(t_j)$ and $v_i(t_j)$ values, and passes them to the next iteration. *Slp_Sizing* algorithm stops when all the constraints are satisfied. This algorithm is described in detail in Figure 8.

Algorithm: Slp_Sizing($R_{sti\text{-}initial}$, $I_{Mi}(t)$, VVSS_MAX)

1: calculate $I_{st_i}(t_j)$ and $v_i(t_j) = R_{st_i - initial} I_{st_i}(t_j)$ for all *i, j* ;
2: **while** (min_slack < 0)
3: find *LBCM* and *m=LBCM;*
4: $R_{st_m} = R_{st_m} - \alpha R_{st_m}$;
5: update $I_{st_i}(t_j)$ and $v_i(t_j) = R_{st_i} I_{st_i}(t_j)$ for all *i, j;*
6: min_slack = ∞;
7: **for** *k=1* to *K*, *j=1* to *N*
8: **if** (Δd_{π_k} – DDR_MAX< min_slack)
9: min_slack = Δd_{π_k} – DDR_MAX;
10: **end if**
11: **end for**
12: **end while**
13: **return**(R_{st_i}) for all *i*;

Figure 8. Current-aware sleep transistor sizing algorithm by delay budgeting.

4. Results

ISCAS-85 benchmark circuits have been used in this paper. We use SIS to generate optimized gate level netlists. All the benchmark circuits are first optimized using "script.rugged" in SIS. We use a 65nm technology library to perform timing-

driven technology mapping. Output information of SIS is passed to our sizing algorithm written in MATLAB. Placement of the sleep transistors is fixed, and we use column-based placement described in section 2. A maximum DDR of 10% has been used in the simulations (DDR_MAX=10%). The rail resistance between each pair of module is assumed to be $r_{VSS_i} = 0.1\Omega$ for all i values. The maximum number of the critical paths considered in this paper, K in Figure 8, is 100, and $\alpha = 0.1$ in Figure 8.

In order to estimate discharging current for each module, we use rectangular current model used in [8]. Table 3 shows the total sleep transistor width in units of λ for the ISCAS-85 benchmark circuits where λ is the minimum feature size, 32.5nm in this paper. We have also compared the results of our delay-budgeting algorithm with the proposed algorithm in [6] and TP algorithm in [7]. Table 3 also shows results for these two algorithms, and the saving that is achieved by the delay-budgeting algorithm compared to these two approaches.

Table 3. Total sleep transistor size in units of λ. DDR_MAX=10%, K=100, α=0.1.

Circuit	# of cells	# of Footers	Total sleep TX width (λ)			
			[6]	TP in [7]	Proposed	Proposed vs. [6] (%)	Proposed vs. [7] (%)
C17	7	2	53	44	16	70	64
9sym	276	30	786	715	312	60	56
C432	214	30	811	665	343	57	48
C880	467	55	1290	1173	579	55	51
C1355	546	60	1437	1597	727	49	54
C3540	1307	280	3920	3469	1679	57	52
C5315	1783	320	5799	5631	3372	42	40
Avg.			2.0	1.89	1	55	52

In order to compare the results of the proposed delay-budgeting algorithm with the TP algorithm in [7], we implemented the TP algorithm. In our implementation of this algorithm, we picked the fixed drop constraints for all the modules such that all the modules would slow down by 10%. However, the proposed delay-budgeting algorithm distributes the given 10% slack optimally among the modules and achieves smaller total sleep transistor width. In order to approximate the results of the algorithm proposed in [6], we used the total sleep transistor width obtained from our implementation of [7] and estimate the total sleep transistor width in [6] using the data given in Table 1 of [7]. As it is seen from the table, the proposed approach saves more than 40% of the total sleep transistor area compared to [6] and [7].

5. Conclusions

We introduced a new approach for minimizing the total sleep transistor width for a coarse-grain MTCMOS circuit assuming a given standard cell and sleep transistor placement. Our algorithm takes a maximum allowed circuit slowdown factor and produces the sizes of various sleep transistors in the standard cell layout while considering the DC parasitics of the virtual ground net. We showed that the problem can be formulated as a sizing with delay budgeting problem and solved efficiently using a heuristic sizing algorithm which implicitly performs maximum current calculation through sleep transistors while accounting for different current flow paths in the virtual ground net through adjacent sleep transistors. This technique uses at least 40% less total sleep transistor width compared to other approaches.

Acknowledgement - This research was supported in part by the National Science Foundation under grant no. 0509564.The authors would like to thank the strategic technology group in LSI Corporation for their valuable discussions and comments.

References

[1] A.Sathanur, A.Pullini, L.Benini, A.Macii, E.Macii, M.Poncino, "Timing-driven row-based power gating," *Proc. Int'l Symp. on Low Power Electronics and Design*, pp. 104-109, 2007.

[2] J. Kao, A. Chandrakasan, and D. Antoniadis, "Transistor sizing issues and tool for multi threshold CMOS technology," *Proc. Design Automation Conference*, pp. 409-414, 1997.

[3] J. Kao, S. Narenda and A. Chandrakasan, "MTCMOS hierarchical sizing based on mutual exclusive discharge patterns," in *Proc. Design Automation Conference*, pp. 495-500, 1998.

[4] Mohab Anis, S. Areibi, and M. Elmasry, "Design and optimization of multithreshold CMOS (MTCMOS) circuits," *IEEE Trans. on CAD of Integrated Circuits and Systems,* pp. 1324-1342, October 2003.

[5] C. Long, L. He, "Distributed sleep transistor network for power reduction," *IEEE Trans. on VLSI Systems*, Volume: 12, No. 9, pp. 937-946, September 2004.

[6] D. S. Chiou, S. H. Chen, S. C. Chang, and C. Yeh, "Timing driven power gating," *Proc. of the Design Automation Conference*, pp. 121-124, 2006.

[7] D. S. Chiou, D. Juan, Y. Chen, and S. Chang, "Fine-grained sleep transistor sizing algorithm for leakage power minimization," *Proc. Design Automation Conference,* pp. 81-86, 2007.

[8] A. Ramalingam, B. Zhang, A. Devgan and D. Pan, "Sleep transistor sizing using timing criticality and temporal currents," *Proc. Asia South Pacific Design Automation Conference,* pp. 1094-1097, 2005.

[9] T. Sakurai and A. Newton, "Alpha-power law MOSFET model and its applications to CMOS inverter delay and other formulas," *IEEE J. Solid-State Circuits*, vol. 25, pp. 584–594, Apr. 1990.