

Multicore Processor Cluster Based Sleep Transistor Sizing Considering Delay Profile

Huang Huang, Jeffrey Fan*

Abstract— This paper proposed a novel method to size the sleep transistor by considering the slack time of the gates in non-critical path in delay profile. The circuit topology was considered to calculate the switching factor, which consequently gave us a more accurate estimation of gates' discharge current. We implemented our method on a 4-bit adder. The proposed switching factor calculation could provide a more accurate estimation of switching current. In consideration of the slack time based on the delay profile, the equivalent worst case discharge current can be reduced, so that the number and the total size of sleep transistors can be dramatically reduced. In theory, the size of sleep transistor can be saved by up to a factor of 3, depending upon the design. In this paper, we also extended the sleep transistor sizing techniques into general multicore architecture. As an example, we modeled nine cores in design. Each core is assumed with different speed degradation allowed to process work loads with various performance requirements. Thus, a balanced design between power dissipation and circuit performance can be maintained.¹

Index Terms – Low Power, Sleep Transistor, Slack Time, Multicore

I. INTRODUCTION

As the technology scaling trends continue and become more and more aggressive, the leakage power will become the dominant factor of the total power consumption, thus leakage power reduction techniques have been intensively studied in the past decade. Several component level leakage power reduction techniques have been proposed in the past a few years such as *Dual Threshold Voltage Partitioning* [1], *Variable Threshold CMOS (VTCMOS)* [2] and *Multi-threshold CMOS (MTCMOS or Sleep transistor)* [3].

The concept of *MTCMOS* proposed in [3] is to insert extra HVT (High Threshold Voltage) transistors as sleep transistors between the functional blocks and the ground to minimize the leakage currents. In the functional blocks, the circuits are implemented with LVT (Low Threshold Voltage) devices that guarantee the circuit speed. The sleep transistors are controlled by a "sleep" signal, which is generated based on the workload of the function blocks. How to properly size the sleep transistors is becoming a challenging task for designers.

Various sleep transistor sizing techniques have been proposed in past years[3][4][5][6][7]. Generally speaking, sizing the sleep transistor is based on an estimation of discharging current of the gates. Thus, by assuming that a 5% speed degradation could be tolerated by the circuit, the size of the sleep transistor can be calculated. However, it is important to notice that the speed of the circuit is determined by its critical path, thus those gates on non-critical paths may have various slack time that can tolerate more performance loss without sacrificing the speed of the overall design. Therefore, if the slack time has been considered, the size of the sleep transistor could be reduced. Another observation is that in multicore processor, different tasks may have different timing requirements. If we can assign different tasks to designated

cores that have a pre-defined speed degradation after inserting sleep transistors, the area of the sleep transistors can be further saved.

In this paper, we have made the following contributions: First, the slack time has been considered to size the sleep transistor to save area. Second, the topology of the gates in the circuit has been considered to find the switching factor. Thus the discharging current can be calculated more accurately. Finally, we extend the proposed sleep transistor sizing techniques into general multicore architecture that each core could have different speed to accommodate different workloads requirements.

The rest of the paper is organized as follows: In Section 2, we briefly introduce the traditional method to calculate the size of the sleep transistor. Section 3 discusses how to find the discharge current for each gate. The algorithm used to find the slack time as well as sizing the sleep transistor considering slack time and gate clustering technique are presented in Section 4. Also in Section 4, we present the concept of sizing the sleep transistor in multicore processor based on workload timing constrains. Section 5 shows the experimental results. Finally, we conclude the paper in Section 6.

II. SIZING THE SLEEP TRANSISTOR

The gate delay time in CMOS digital circuits with the presence of sleep transistor can be approximated by:

$$T_D^{sleep} \propto \frac{C_L V_{DD}}{((V_{DD} - V_{sleep}) - V_{th})^\alpha} \quad (1)$$

where V_{sleep} is the voltage drop on the sleep transistor, which can be modeled as a resistor during active mode.

In previous works [4][5][7], the sizing of the sleep transistor has been done by assuming 5% performance degradation in the whole circuit due to the decrease of gate's driving capability. It also assumes that the velocity saturation index $\alpha \approx 1$. Therefore, the above relationship can be expressed as:

$$\frac{T_D}{T_D^{sleep}} = \frac{((V_{DD} - V_{sleep}) - V_{th})^\alpha}{(V_{DD} - V_{th})^\alpha} = 1 - \frac{V_{sleep}}{V_{DD} - V_{th}} = 95\% \quad (2)$$

The discharge current flowing through the sleep transistor can be approximated by [4][5][7]:

$$I_{sleep} \approx \mu_n C_{ox} \left(\frac{W}{L}\right)_{sleep} (V_{DD} - V_{TH}) V_{sleep} \quad (3)$$

where μ_n is the mobility of electrons, C_{ox} is the oxide capacitance, V_{TH} is high threshold voltage of the sleep transistor and $\left(\frac{W}{L}\right)_{sleep}$ is the size of the sleep transistor.

From Eq. 2 we get:

$$V_{sleep} = 0.05(V_{DD} - V_{th}) \quad (4)$$

¹Huang Huang and *Jeffrey Fan are with the Department of Electrical and Computer Engineering, Florida International University, Miami, FL 33174, USA, e-mail: {hhuan001, jeffrey.fan}@fiu.edu

Substitute V_{sleep} into Eq. 3 and solve for $(\frac{W}{L})_{sleep}$, the size of the sleep transistor can be expressed as:

$$\left(\frac{W}{L}\right)_{sleep} = \frac{I_{sleep}}{0.05\mu_n C_{ox}(V_{DD} - V_{TH})(V_{DD} - V_{th})} \quad (5)$$

Therefore, if I_{sleep} is known, the size of the sleep transistor can be calculated immediately.

One important observation is that in these methods, a 5% speed degradation is assumed for the whole circuit. In fact, there are some non-critical paths with various slack time T_δ that can tolerate more speed degradation without sacrificing the overall performance of the circuit.

For example, assume the propagation delay of the gates on the critical path and non-critical path are $T_{Delay}^{critical}$ and $T_{Delay}^{non-critical}$ respectively, where $T_{Delay}^{non-critical} = 90\% T_{Delay}^{critical}$. If again we assume there is a 5% speed degradation for the gates on critical paths, then the degradation of gates on the non-critical paths become:

$$\frac{T_{Delay}^{non-critical}}{T_D^{sleep}} = \frac{90\% T_{Delay}^{critical}}{T_D^{sleep}} = 0.9 \times 95\% = 86\% \quad (6)$$

which means 14% performance loss can be tolerable on non-critical path gates without changing the original delay profile. Therefore, the size of sleep transistor can be saved, if we treat the gates on critical paths and non-critical paths separately.

III. GATE CURRENT ESTIMATION

We improve the accuracy of the methods proposed in [5][7] to estimate the worst case discharging current of the circuit. Based on the previous methods, the expected discharge current can be obtained by:

$$I_{exp} = \alpha_{switching} \times I_{peak} \quad (7)$$

where I_{peak} is the worst case discharging current of each gate.

The switching factor $\alpha_{switching}$ for each gate in the circuit. $\alpha_{switching}$ is the probability of a gate's output switches. It can be expressed as:

$$\alpha_{switching} = P_0 \cdot P_1 = P_0 \cdot (1 - P_0) \quad (8)$$

If we assume that the input value has equal chance to be 0 or 1, then Eq. 8 can be rewritten as:

$$\alpha_{switching} = \frac{N_0 \times (2^N - N_0)}{2^{2N}} \quad (9)$$

where N is the number of input, N_0 is the number of zero entries in the output column of the truth table.

This method is based on the assumption that all the inputs are independent and uniformly distributed. But this is rarely the case in actual circuit, where signal correlation exhibits. Consider an example that the outputs of three OR2 gates are connected to the input of an AND3 gate. If we use the above assumption, the switching factor for AND3 gate is $7/64 \approx 0.11$. However, we noted the inputs of the AND3 gate are the outputs of the OR2 gates. Thus, the assumption that the inputs are uniformly distributed is no longer valid. In this case, we

cannot use $P_A=P_B=P_C=1/2$ to calculate the switching factor. Instead, $P_A=P_B=P_C=3/4$ should be used to find $\alpha_{switching}$. In this case, for a 3-input AND gate, we have

$$\alpha_{switching} = [1 - (P_A \cdot P_B) \cdot P_C](P_A \cdot P_B \cdot P_C) \approx 0.24 \quad (10)$$

In the above example, the traditional method used to calculate $\alpha_{switching}$ apparently underestimated the current flowing through the sleep transistor, which will in consequence underestimate the size of the sleep transistor. In some other cases, $\alpha_{switching}$ could also be overestimated. The proposed technique with the calculation of $\alpha_{switching}$ will take the actual topology location of the gate into consideration.

Algorithm 1 DISCHARGING CURRENT ESTIMATION

```

Assign a level number to all gates
Find  $I_{peak}$  for all gates in the circuit using SPICE
for all gates in  $i_{th}$  level do
  if the input is primary input then
     $P_i^{input} \leftarrow 1/2$ 
  else
     $P_i^{input} \leftarrow P_{i-1}^{output}$  {  $P_{i-1}^{output}$  is the probability that the
    output of the previous gate is one }
  end if
  Calculate  $\alpha_{switching}$  based on  $P_i^{input}$  {As Eq. 10 for a
  AND3 gate}
  Calculate  $P_i^{output}$  {Prepare for calculating  $\alpha_{switching}$  of
  gate in next level}
   $I_{exp} \leftarrow \alpha_{switching} \times I_{peak}$ 
end for

```

Instead of calculating the accumulated delay for gates as in [5], we use static timing analysis tool to get a timing window that predicts the possible time when the high-to-low transition occurs. For each gate, the previous calculated I_{exp} is bounded by its timing window. Finally, these current profiles are transferred to vector to facilitate the succeeding computation.

IV. PROPOSED SLEEP TRANSISTOR SIZING TECHNIQUE

A. Identify Critical Path and Calculate Slack Time

In our proposed sleep transistor sizing algorithm, the slack time of the gates on the non-critical paths is considered, so that the sleep transistor can be sized "just enough" to make sure that when the sleep transistor has been inserted, the delay of the non-critical path will not exceed the propagation delay of the critical path.

In this section, we adopt the method used in [1] to find the critical path(s) as well as the slack time for each gate on non-critical path. During the preprocessing stage, a level number will be assigned to each gate. The high-to-low transition propagation delay (T_p^{HL}) of the gate will be collected, because only during the high-to-low transition, there will be current flowing through the sleep transistor. It is important to notice that the delay of the gate also related to the input vector combination. Therefore, we choose the worst case as the T_p^{HL} of the gate. Proceeding level by level from the primary input to primary output, three parameters associated with each gate can be calculated, which include T_p^{HL} , T_{arrive}^{max} (the maximum accumulate delay for each fan-in path) and T_{left} (which

defined as $T_{arrive}^{max} + T_p^{HL}$). The algorithm for identify the critical path is shown in Algorithm 2[1].

Algorithm 2 IDENTIFY CRITICAL PATH

```

Assign a level number to all gates
Find  $T_p^{HL}$  for all gates in the circuit
Calculate  $T_{arrive}^{max}$  and  $T_{left}$  of each gate
 $T_{critical} \leftarrow Max\{T_{left}\}$  {The largest  $T_{left}$  of the primary
output will be critical delay}
Mark the primary output with the largest  $T_{left}$  as the gate
in critical path  $G_{critical}^n$ 
for each  $G^{n-1}$ , the fan-in gate of  $G_{critical}^n$  do
  if ( $T_{left}(n-1) = T_{arrive}^{max}(n)$ ) && ( $G^{n-1}$  is not a primary
input) then
    Mark the gate as  $G_{critical}^{n-1}$ 
  end if
end for

```

After the gates on critical path have been identified, we can calculate the slack time T_δ for gates on non-critical paths by going backward from output to input level by level.

B. Size Sleep Transistor Considering Slack Time

When the slack time has been considered, the new performance loss for gates on non-critical path can be defined as:

$$\frac{T_D}{T_{D-slack}^{sleep}} = \frac{T_D}{T_D^{sleep} + T_\delta} \quad (11)$$

where $T_{D-slack}^{sleep}$ is the delay that can be tolerated for a gate on non-critical path without changing the delay profile of the circuit. In other words, after inserting the sleep transistors, the gates in critical path will still suffer 5% speed degradation, while the gates on non-critical path can tolerate more performance loss depending upon how many slack time they may have.

By solving T_D^{sleep} from Eq. 2, we can rewrite Eq. 11, so that the new performance loss can be expressed as:

$$\frac{T_D}{T_D^{sleep} + T_\delta} = \frac{T_D}{\frac{T_D}{0.95} + T_\delta} = X\% \quad (12)$$

Therefore, the size of the sleep transistor will be:

$$\left(\frac{W}{L}\right)_{sleep} = \frac{I_{sleep}}{(1-X)\mu_n C_{ox}(V_{DD} - V_{TH})(V_{DD} - V_{th})} \quad (13)$$

If we incorporate the effect of slack time into the numerator of Eq. 13, then Eq.13 can be rewritten as:

$$\left(\frac{W}{L}\right)_{sleep} = \frac{\frac{0.05}{1-X} I_{sleep}}{0.05\mu_n C_{ox}(V_{DD} - V_{TH})(V_{DD} - V_{th})} \quad (14)$$

For gates on critical path, $\frac{0.05}{1-X}$ is equal to 1. In this case, Eq.14 and Eq.5 should be the same. For gates in non-critical paths, the various slack time will cause different X. Since $\frac{0.05}{1-X}$ is always equal or less than 1, the smaller $\left(\frac{W}{L}\right)_{sleep}$ can be achieved.

However, Eq.13 cannot be used for our cluster based method directly, because we use one sleep transistor to support several gates. The I_{sleep} in the above equation is for the current limit

of the sleep transistor, not the discharge current of the gate. But from Eq. 14, we know that by considering the slack time, the equivalent discharge current could be reduced. We will demonstrate how to relate the slack time to the reduction of the discharge current that could be used in the clustering technique as follows.

The gate propagation delay without sleep transistor can be expressed as:

$$T_D = \frac{C_L V_{DD}}{I_0} \quad (15)$$

where I_0 is the discharge current. From Eq.11 and Eq.12 we get that:

$$T_{D-slack}^{sleep} = \frac{T_D}{X\%} = \frac{C_L V_{DD}}{I_0 \times X\%} \quad (16)$$

Therefore, the equivalent discharge current after considering slack time is $I_0 \times X\%$. We define $X\%$ as the slack factor. Since it is always less than 95%, the smaller equivalent gate discharge current can be achieved, which in consequence save the current capacity of sleep transistor.

C. Gate Clustering Technique

We adopt the similar concept proposed in [5] to model the gate clustering problem as "packing" problem. The idea is to group as many gates as possible without exceeding the current limit of the sleep transistor, while keep the number of sleep transistor minimum. It is a simple binary integer programming problem that can be efficiently solved in ILOG OPL environment.

However, we cannot use I_{exp} , *i.e.* the results from Section 3 as the "weight" used in "packing" problem directly, because the partially overlapped discharge timing windows have not been taken into consideration yet. Therefore, a preprocessing stage that groups gates based on their timing windows is necessary in order to find I_{EQ} . And then we use I_{EQ} of each sub-cluster to further group them into final clusters that supported by sleep transistors [5].

After we obtain I_{EQ} for each sub-cluster, the final clustering process can be simplified to a "packing" problem.

D. Sleep Transistor Sizing In Multicore Scenario

The proposed sleep transistor sizing techniques could be extended to multicore applications. Note that the size of the sleep transistors were calculated by Eq. 14, where uniform 5% performance loss is assumed for all critical paths. However, in real world applications, some tasks require strict timing constraints, whereas some others may tolerate more delay without affecting the quality of service. Therefore, if we can assort the tasks into different categories based on their timing requirements and assign them into different cores, the size of the sleep transistors in each core could be calculated independently. Hence, additional sleep transistor sizing can be saved for those cores, which are designated to process tasks with looser timing constraints. Fig. 1 demonstrates the idea described above. Nine cores are modeled for different performance degradation. Note that core 1 and 2 are designed for applications that are more sensitive to delay timing, thus additional sleep transistor sizing has been traded in for better performance. However, the overall sleep transistor sizing can be reduced compared with uniform 5% minimum speed degradation sizing due to the compensation of the cores design for larger delay.

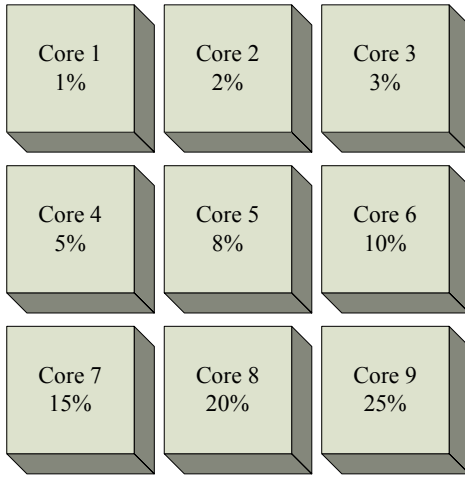


Fig. 1. Example of sleep transistor sizing in multicore processor

V. EXPERIMENTAL RESULTS

The proposed method has been implemented on a 4-bit adder. We use $45nm$ CMOS technology. The channel length of the sleep transistor is set to $45nm$ and the maximum current limit for sleep transistor is set to $200\mu A$.

In the experiment, all the gates in the circuit were matched with $45nm$ standard cell library which contains all the gate information including the propagation delay of each gate. Only one core (with 5% critical path speed degradation) was simulated to demonstrated the efficiency of the algorithm.

In Fig.2, the switching factors of all non-primary input gates are plotted. The "circles" represent the value calculated by traditional method that assumes all inputs are independent and uniformly distributed, whereas the "squares" denote the gate switching factors found by the proposed method that take the position of each gate into consideration. In Fig. 2, we can see that the switching factors are either underestimated or overestimated by certain degrees. The worst case scenario is that several underestimated gates are clustered together, consequently the total discharge currents might over the limit of the sleep transistor. Therefore, by using the proposed switching factor calculation method, we may lead to more accurate estimation of discharge current.

We used the method introduced in Section 4 to find the critical path and the slack time for each gate. Fig.3 shows the results for all gates in the circuit. The gates with slack factor equal to 0.95 are either on critical path or have zero slack time. The result proved that when the slack time has been considered, the equivalent worst case discharge currents were dramatically reduced, while the overall circuit performance constraints still maintained.

VI. CONCLUSION

Sleep transistor sizing techniques have been proposed by considering the slack time of the gate in non-critical paths. The circuit topology has also been considered when calculating the switching factors, so that a more accurate estimation of gate's discharging current can be found. Experimental result showed that the equivalent worst case discharge current can be reduced. Therefore, the number of sleep transistors can be saved. Furthermore, the proposed algorithm can be used to

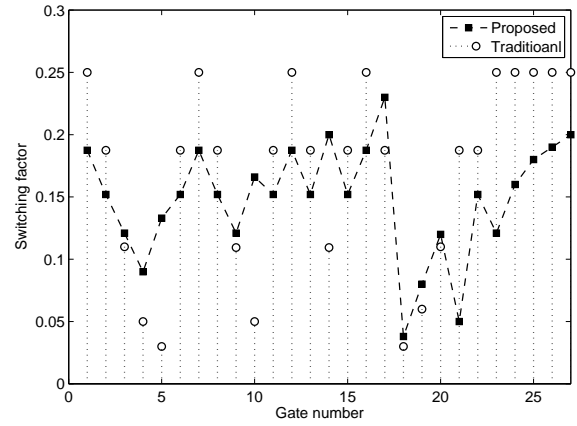


Fig. 2. Calculation of switching factors

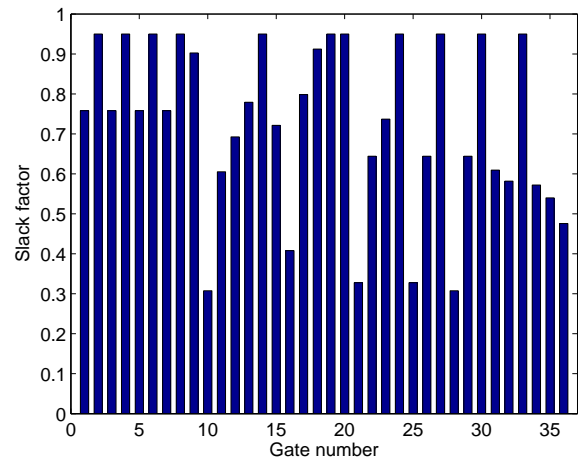


Fig. 3. Calculation of slack factors

size multicore processor that each core has a specific speed to accommodate different tasks with different performance requirements, such that it provides additional reduction of sleep transistor sizing in circuit design.

REFERENCES

- [1] L.We, Z.Chen, K.Roy, M.Johnson, and V.De, Design and Optimization of Dual-Threshold Circuits for Low-Voltage Low-Power Applications, *IEEE Transactions on VLSI Systems*, vol. 7, no. 1, pp. 16C24, March 1999.
- [2] T.Kuroda, "A 0.9v 150mhz 10mw $4mm^2$ 2-d discrete cosine transform core processor with variable threshold voltage scheme," *IEEE Journal of Solid-State Circuits*, vol. 31, pp. 1770-1779, Nov. 1996.
- [3] S. Mutoh, T. Douseki, Y. Matsuya, T. Aoki, S. Shigematsu, and J. Yamada, 1-V power supply high-speed digital circuit technology with multithreshold-voltage CMOS, *IEEE Journal of Solid State Circuits*, 1995.
- [4] J. Kao, S. Narendra, and A. Chandrakasan, MTCMOS hierarchical sizing based on mutual exclusive discharge patterns, *Design Automation Conference*, 1998, pp. 495C500.
- [5] M. Anis, S. Areibi, and M. Elmasry, Design and optimization of multi-threshold CMOS (MTCMOS) circuits, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2003.
- [6] C. Long and L. He, Distributed sleep transistor network for power reduction, in *Design Automation Conference*, 2003, pp. 181C186.
- [7] A. Ramalingam, B. Zhang, A. Devgan, D. Pan "Sleep transistor sizing using timing criticality and temporal currents" *Conference on Asia South Pacific design automation*, 2005