Simultaneous Control of Power/Ground Current, Wakeup Time and Transistor Overhead in Power Gated Circuits

Yongho Lee, Deog-Kyoon Jeong, and Taewhan Kim
School of Electrical Engineering and Computer Science
Seoul National University, Korea

***Abstract*— Power gating in circuits is one of the effective technologies to allow low leakage and high performance operations. This work aims to *analyze and establish the relations between the three important design parameters in power gated circuits: (i) the maximum current flowing from/to power/ground (ii) the wakeup (sleep to active mode transition) time delay and (iii) the number of sleep transistors.* With the understanding of relations between the parameters, we propose *solutions to the two problems: (1) finding logic clusters and their wakeup schedule to minimize the sleep transistor overhead* under the constraints of wakeup time and peak current and *(2) finding logic clusters and their wakeup schedule to minimize the wakeup time* under the constraints of peak current and the number of sleep transistors. From an experimentation using ISCAS benchmarks, it is shown that our proposed technique is able to explore the search space, finding solutions with 65% $\sim$ 77% reduced number of sleep transistors and 30% $\sim$ 36% reduced wakeup time delay, compared to the results by the previous work.**

I. Introduction

With the advance of technology, power supply voltage scaling technique has been applied to reduce the dynamic power because the dynamic power consumption is proportional to the supply voltage quadratically. To maintain the performance, threshold voltage (Vt) also should be scaled down. However, this increases the subthreshold leakage exponentially. In nanometer technology, the increased leakage power dominates the dynamic power. There are many approaches for reducing leakage power. Among them, the power gating technology using MTCMOS (Multi-Threshold CMOS) provides a powerful and effective leakage power reduction.

In this power gating technology, the three important design parameters are the wakeup (from sleep to active mode transition) time, the amount of current flowing to ground when the sleep transistors are turned on, and the sleep transistor overhead. Note that reducing the wakeup time affects the overall performance of the circuit, and reducing the peak current flowing to ground when the sleep transistors are turned on mitigates the noise on the power distributed network. Furthermore, reducing the number of sleep transistors saves the design area and also reduces the design complexity. Clearly, there are trade offs between the number of sleep transistors used, the peak current flowing to ground, and the transition time from sleep to active mode.

For designing power gated circuits, most of the previous work focused on controlling/optimizing one or two of the parameters: the sleep transistor size [1], [2], [3], [4], the ground bounce [5], [6] (i.e., maximum current flow), and the wakeup time delay. Recently, the work in [7] proposed a logic cell clustering method to reduce wakeup time delay. The method partitioned the circuit into several clusters according to the peak current and wakeup dependency constraints, followed by scheduling the wakeup times of clusters. The limitation of the method is that the wakeup delay cannot be fully minimized due to the unawareness of the effects of the wakeup dependency constraint and the sleep transistor overhead by logic clustering on the wakeup delay.

This work addresses a new approach to the problem of logic cell clustering for controlling peak current, wakeup time delay, and sleep transistor overhead in power gated circuits. Specifically, this work *analyzes and establishes the relations of the wakeup time delay, peak current, and the number of sleep transistors with various sizes of clusters, and wakeup time schedule.*

II. Observations and Analyses

This section covers key observations and analyses.

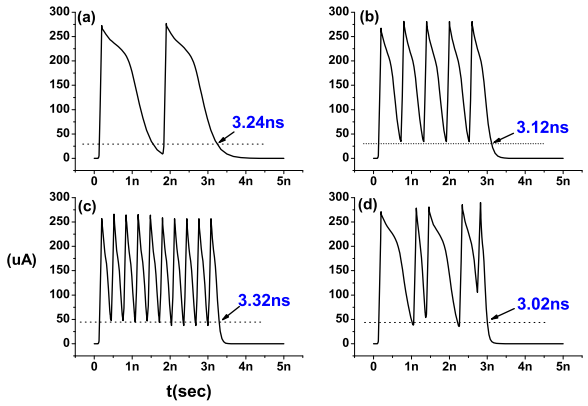

Fig. 1. Wakeup sink current profiles for various logic clusters of a chain of 20 inverters: (a) using two clusters of same sizes; (b) using five clusters of same sizes; (c) using ten clusters of same sizes; (d) using five clusters of different (non-uniform) sizes. (We used HSPICE simulation using 130nm technology.)

• *Relations between logic clusters and wakeup time*: Fig. 1 shows what the current profiles and the wakeup time delays look like for various numbers and sizes of logic clusters. Fig. 1(a) shows the wakeup current profile when the logic circuit of 20 chained inverters is partitioned into two logic clusters of almost equal sizes and they are waked up sequentially by using their own sleep transistors. The wakeup time is about 3.24ns and the peak current is around 270uA. On the other hand, Fig. 1(b) shows the wakeup current profile when the circuit is partitioned into five clusters of equal sizes and they are waked up sequentially, using five sleep transistors of the same size as that used in Fig. 1(a). Note that the wakeup time is reduced from 3.24ns to 3.12ns using the same peak current as that in Fig. 1(a). The reduction of wakeup time indicates that increasing the number of logic clusters provides a higher chance of shortening the wakeup time delay due to the short time interval of current flow through each sleep

978-1-4244-2820-5/08/$25.00 ©2008 IEEE

transistor. However as shown in Fig. 1(c), using too many logic clusters increases the wakeup time delay (from 3.12ns for five clusters to 3.32ns for ten clusters). This is mainly because of the high increase of the additional current by the sleep transistors. This implies an optimal logic clustering strategy is required to minimize the wakeup time delay. In addition, Fig. 1(d) shows another current profile when five logic clusters are used as that in Fig. 1(b), but their clustering sizes are not the same. The wakeup time is now reduced to 3.02ns. The data shown in Fig. 1(d) clearly indicate that *an optimal logic clustering should consider not only the number of clusters but also the sizes of clusters.*

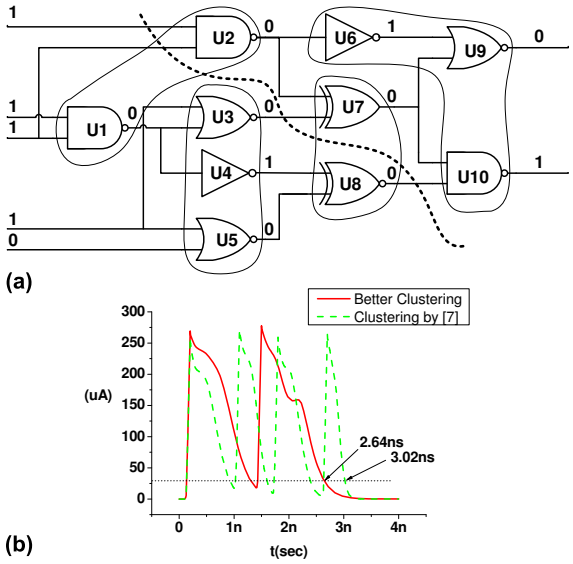

Fig. 2. Example showing the effect of the wakeup dependency on wakeup time: (a) logic circuit for clustering; (b) wakeup current profiles for the logic clusters by [7] and a better partitioning.

• *Wakeup dependency constraint of logic clustering*:
Recently, the work in [7] proposed a logic clustering methodology whose objective is to reduce the wakeup time. The clustering procedure was controlled and guided by the peak current and the wakeup dependency constraints. *The wakeup dependency constraint* refers to satisfy the logic clustering so that two gates connected by a net with 0-state in sleep mode should be in the different clusters. Otherwise, the logic clusters generate short circuit current when waked up. It used a greedy clustering approach. The limitation is that it unnecessarily generates too many clusters even the clustering completely eliminates the possible short circuit current during the wakeup time period. For example, Fig. 2(b) shows two current profiles of two logic clusterings for the circuit in Fig. 2(a) where with given states of primary inputs and peak current constraint, the work in [7] generates four clusters $\{U1,U2\}$, $\{U3,U4,U5\}$, $\{U7,U8\}$, $\{U6,U9,U10\}$ (shown by four solid circles in Fig. 2(a)) to satisfy the wakeup dependency constraint (for avoiding short circuit current), resulting in 3.02ns of wakeup time while the clustering result of two clusters $\{U1,U3,U4,U5,U8\}$, $\{U2,U6,U7,U9,U10\}$ (partitioned by the dotted line in Fig. 2(a)) reduces the wakeup time to 2.64ns. Note that for the latter clustering, there are violations of wakeup dependency constraint at $U1 \rightarrow U3$, $U1 \rightarrow U4$, $U5 \rightarrow U8$, $U2 \rightarrow U6$, $U2 \rightarrow U7$, $U7 \rightarrow U9$ and $U7 \rightarrow U10$. This comparison indicates that strictly satisfying the wakeup dependency in clustering is not a primary condition. Instead, it is beneficial to explore the sizes and numbers of logic clusters as a primary objective and to minimize the violations of wakeup dependency constraint as many as possible as a secondary objective.

III. Simultaneous Control of Peak Current, Wakeup Time, and Sleep Transistors

Based on the analyses, we propose a graph based approach, called PowerG-ctr, to the two problems:

(**Problem 1**) *Finding logic clusters and their wakeup schedule that minimize the number of sleep transistors while satisfying the wakeup time and peak current constraints*;

(**Problem 2**) *Finding logic clusters and their wakeup schedule that minimize the wakeup time delay while satisfying the constraints of peak current and the number of sleep transistors.*

A. The Proposed Approach

Let I_{max} be the limit of peak current flowing through a sleep transistor when the sleep transistor is turned on. Let $\mathcal{C}$ and $\mathcal{S}(\mathcal{C})$ be the set of partitioned clusters of a given circuit C and the wakeup schedule of the clusters in $\mathcal{C}$, respectively, and $T_{wakeup}(\mathcal{S}(\mathcal{C}))$ be the delay required for waking up all the logic clusters in $\mathcal{C}$ while satisfying I_{max}.

Algorithm for solving problem 1: The value of I_{max} and the wakeup delay limit, denoted as T_{max}, are given to be satisfied. The proposed algorithm is an iterative one. Initially, the entire circuit C is considered as one logic cluster. That is, $\mathcal{C} = \{C\}$. We then measure the current profile of $\mathcal{S}(\mathcal{C})$ under the I_{max} constraint. If $T_{wakeup}(\cdot) \leq T_{max}$, then we found a solution and stop the process. If not, we partition C into two logic clusters. Suppose C_1 and C_2 are two partitioned clusters of C so that $\mathcal{C} = \{C_1,C_2\}$ and $\mathcal{S}(\mathcal{C}) = C_1 \rightarrow C_2$. Then, we measure the current profile of $\mathcal{S}(\mathcal{C})$ under the I_{max} constraint to check the wakeup time delay. If $T_{wakeup}(\cdot) \leq T_{max}$, then we stop and found a solution. Otherwise, the partitioning process repeats until the T_{max} constraint is satisfied or the value of $T_{wakeup}(\cdot)$ does not decrease any more. Here, the core part of the procedure that greatly affects the quality of results is the partitioning of logic circuit.

• *Circuit partitioning*: Suppose $\mathcal{C}$ is $\{\cdots,C_i,C_j,C_k,\cdots\}$ and $\mathcal{S}(\mathcal{C})$ is $\cdots \rightarrow C_i \rightarrow C_j \rightarrow C_k \rightarrow \cdots$ in the previous iteration of our algorithm. The cluster to be partitioned in the current iteration is the largest sized cluster in $\mathcal{C}$. Suppose $|C_j| = max\{\cdots,|C_i|,|C_j|,|C_k|,\cdots\}$, and C_{j1} and C_{j2} are the two partitioned clusters with data dependency from C_{j1} to C_{j2}. Then, the updated cluster set and schedule become $\mathcal{C}$ = $\{\cdots,C_i,C_{j1},C_{j2},C_k,\cdots\}$ and $\mathcal{S}$ = $\cdots \rightarrow C_i \rightarrow C_{j1} \rightarrow C_{j2} \rightarrow C_k \rightarrow \cdots$. In the following, we describe our partitioning procedure.

The objective of the partitioning is *to eliminate the possible short circuit currents that occur when the sleep transistors are turned on to wake up the circuits of clusters*. For example, we want to partition the circuit in Fig. 3. Suppose that the logic

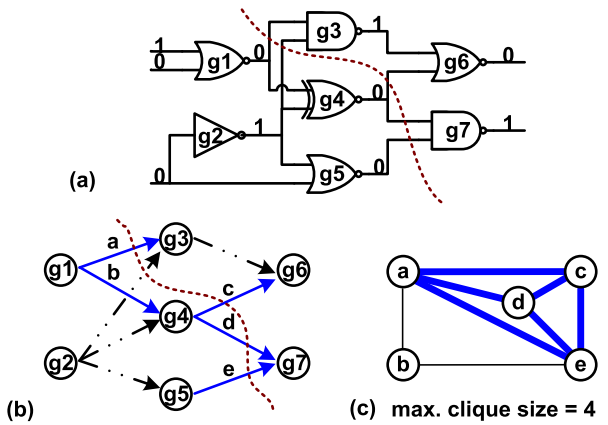

Fig. 3. Example showing the transformation of circuit into a graph to formulate the circuit partitioning problem into an equivalent max. clique finding problem in a graph: (a) circuit C to be partitioned; (b) network $N(V,A)$ transformed from C; (c) graph $G(V',E)$ transformed from $N(V,A)$.

states of the gates specified in Fig. 3(a) are the states that are to be retained when the circuit is waked up. Then, there will be current flows from the nets that are to be in 0-state. Those nets are $(g1,g3,g4)$ $(g4,g6,g7)$ and $(g5,g7)$ in Fig. 3(a). Consequently, to minimize the short circuit current that occurs on the gates connected from/to those nets, partitioning should cut those nets that are to be 0-state as many as possible while satisfying the data dependency constraint in the partition (i.e., creating no data dependency cycle among the partitioned clusters).

We *formulate the partitioning problem into the problem of finding a maximum clique in a graph.* We first transform circuit C to be partitioned into a network $N(V,A)$. Each node in V represents a distinct gate in C. There is an arc (v,w) in A if and only if there is a net that is to be in 0-state after when C is waked up, and that v and w correspond to the gates that are connected from and connected to the net, respectively. Fig. 3(b) shows $N(V,A)$ of C in Fig. 3(a) where the arcs are shown with solid lines. Then, $N(V,A)$ is used to derive a graph representation $G(V',E)$. Each node in V' represents a distinct arc in $N(V,A)$, and there is an edge (x,y) in E if and only if the two arcs in $N(V,A)$ corresponding to x and y have no flow dependency from one to the other. For example, Fig. 3(c) shows $G(V',E)$ for the network $N(V,A)$ in Fig. 3(b). Note that there are edges between a and b and between a and c because there is no flow dependency in $N(V,A)$, but there is no edge between b and c and between b and d because there are paths from b to c and from b to d in $N(V,A)$. Then, from the transformed graph $G(V'E)$, we find a clique of maximum size. The nodes of the clique will be the net to be cut in partitioning C. For example, the maximum clique in Fig. 3(c) is (a,c,d,e) and the corresponding cut of partitioning C and $N(V,A)$ are shown as dotted line in Figs. 3(a) and (b). Since the problem of finding a maximal clique is NP-complete, our algorithm uses the efficient clique partitioning heuristic proposed in [8].

Algorithm for solving problem 2: The value of I_{max} and the maximum number of clusters (or maximum number of sleep transistors), denoted as N_{max}, are given to be satisfied. The proposed algorithm is essentially the same as that for solving problem 1. The only difference is the condition of stopping iteration. For this case, the iteration of partitioning stops when the number of clusters equals to N_{max} or there is no further reduction on the value of $T_{wakeup}(S(C))$.

IV. Experimental Results

We implemented the proposed algorithm PowerG-ctr in C++ and tested it on a set of ISCAS benchmark circuits [9] to assess its effectiveness. All experiments were run on a 2.8GHz Intel processor equipped with 1GB of RAM. Each of the circuits is decomposed using the logic gates of INV, NAND2, NOR2, XOR2 and XNOR2. To simulate the characteristics of sleep transistor, 130nm standard cell HSPICE library is used.

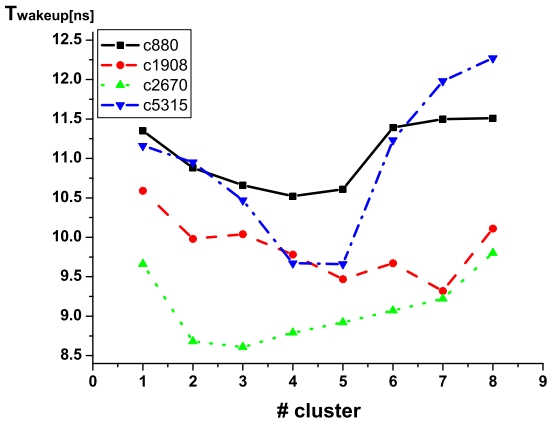

Fig. 4. Curves showing the design trade-offs between T_{wakeup} and the number of clusters in Table I.

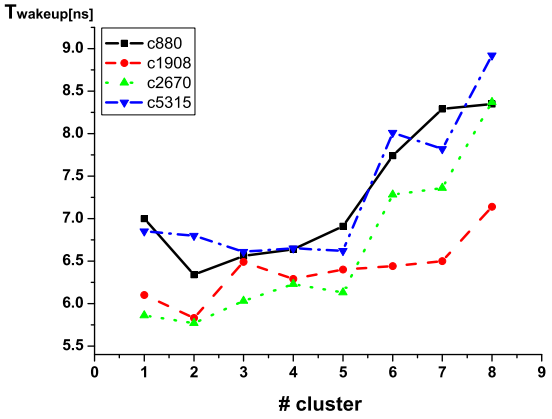

Fig. 5. Curves showing the design trade-offs between T_{wakeup} and the number of clusters in Table II.

Tables I and II show the comparisons of the numbers of logic clusters and the wakeup times used by the conventional method [7] (denoted as CONV) and by our PowerG-ctr by varying the sleep transistor size. The sleep transistor size used in Table I is roughly 1.5% of the total sum of the widths of gates in the circuit. The sleep transistor size used in Table II was then set by multiplying 2 times to that used in Table I. Under the peak current constraint, PowerG-ctr is applied to each circuit to explore the sleep transistor overhead and the wakeup time during the logic partitioning process. I_{max} and *width* in the second column of tables represent the peak

TABLE I

COMPARISONS OF THE SLEEP TRANSISTOR OVERHEAD (BY THE NUMBER OF CLUSTERS) AND THE WAKEUP TIME BETWEEN THE RESULTS PRODUCED BY CONV [7] AND PowerG-ctr UNDER *width* CONSTRAINT.

						difference	
		CONV [7]		PowerG-ctr		$\frac{\text{PowerG-ctr}}{\text{CONV}}$	
circuit	$I_{max}[mA]$ $(width[um])$	#clusters	$T_{wakeup}(ns)$	#clusters	$T_{wakeup}(ns)$	#clusters	T_{wakeup}
C432	0.95 (1.5)	4	9.79	3	7.44	0.75	0.76
C880	1.56 (2.5)	9	11.96	4	10.52	0.44	0.88
C1355	1.55 (2.5)	15	12.50	3	7.86	0.20	0.63
C1908	1.55 (2.5)	14	12.18	7	9.32	0.50	0.77
C2670	3.15 (5)	16	12.65	3	8.61	0.19	0.68
C3540	6.36 (10)	19	15.20	4	9.67	0.21	0.64
C5315	6.37 (10)	17	14.80	5	9.66	0.29	0.65
C7552	6.37 (10)	18	18.69	4	11.70	0.22	0.63
Avg.						**0.35**	**0.70**

TABLE II

COMPARISONS OF THE SLEEP TRANSISTOR OVERHEAD (BY THE NUMBER OF CLUSTERS) AND THE WAKEUP TIME BETWEEN THE RESULTS PRODUCED BY CONV [7] AND PowerG-ctr UNDER *width* = TWO TIMES OF *width* IN TABLE I.

						difference	
		CONV [7]		PowerG-ctr		$\frac{\text{PowerG-ctr}}{\text{CONV}}$	
circuit	$I_{max}[mA]$ $(width[um])$	#clusters	$T_{wakeup}(ns)$	#clusters	$T_{wakeup}(ns)$	#clusters	T_{wakeup}
C432	1.8 (3)	4	6.26	3	4.44	0.75	0.71
C880	3 (5)	9	7.93	2	6.34	0.22	0.80
C1355	3 (5)	15	7.66	2	4.86	0.13	0.63
C1908	3 (5)	14	9.41	2	5.83	0.14	0.62
C2670	6.4 (10)	16	9.40	2	5.77	0.13	0.61
C3540	12 (20)	19	12.60	3	7.06	0.16	0.56
C5315	12 (20)	17	11.03	3	6.61	0.18	0.60
C7552	12 (20)	18	14.06	3	7.75	0.17	0.55
Avg.						**0.23**	**0.64**

(maximum) discharged current that is allowed through sleep transistor and the width of sleep transistor corresponding to the peak current, respectively. As shown in the tables, the numbers of clusters (#*clusters* in tables) used by [7] are 3 to 4 times more than that by PowerG-ctr while its wakeup times (T_{wakeup} in tables) are 40% to 60% longer than that by PowerG-ctr. This clearly indicates that PowerG-ctr which performs a systematic exploration of the effects of logic clustering on the wakeup time and sleep transistor overhead is very effective.

The curves in Figs. 4 and 5 show the design trade-offs between the values of T_{wakeup} and the numbers of clusters for benchmarks C880, C1908, C2670, C5315 using the I_{max} constraints in Tables I and II, respectively. The comparisons of the curves indicate that whatever the I_{max} values are, the shape of curves are similar, but the curves are shifted to the left as the I_{max} value increases. This means that as the I_{max} value increases, the wakeup time is shorten and the number of sleep transistor used decreases.

V. CONCLUSION

This work presented a logic clustering based solution to the problem of controlling/optimizing the power gating parameters. This work solved two problems: (1) finding logic clusters and their wakeup schedule to minimize the sleep transistor overhead under the constraints of wakeup time and peak current, and (2) finding logic clusters and their wakeup schedule to minimize the wakeup time under the constraints of peak current and the number of sleep transistors. It was shown that our proposed techniques were very effective.

VI. ACKNOWLEDGMENT

The work was supported by Nano IP/SoC Promotion Group of Seoul R&BD Program, IT-SoC Program, System IC2010 project of Korea Ministry of Knowledge Economy, the Korea Science and Engineering Foundation (KOSEF) grant funded by the Korea government (No. R01-2007-000-20891-0), and the Korea Ministry of Knowledge Economy (MKE) under the Information Technology Research Center (ITRC) support program supervised by the Institute for Information Technology Advancement (IITA) (IITA-2008-C1090-0804-0009).

REFERENCES

[1] J. Kao, A. Chandrakasan, and D. Antoniadis, "Transistor sizing issues and tool for multi-threshold CMOS technology," *DAC,* 1997.

[2] D. Chiou, S. Chen, S.Chang, and C. Yeh, "Timing driven power gating," *DAC,* 2006.

[3] F. Li, and L. He, "Maximum current estimation considering power gating," *ISLPED,* 2001.

[4] A. Sagahyroon, and F. Aloul, "Maximum power-up current estimation in combinational CMOS circuits," *Proc. of IEEE/ACM Mediterranean Electrotechnical Conference,* 2006.

[5] S. Kim, S. Kosonocky, and D. Knebel, "Understanding and minimizing ground bounce during mode transition of power gating structures," *ISLPED,* 2003.

[6] C. Long, and L. He, "Distributed sleep transistor network for power reduction," *DAC,* 2003.

[7] A. Abdollahi, F. Fallah, and M. Pedram, "An effective power mode transition technique in MTCMOS circuits," *DAC,* 2005.

[8] C-J. Tseng, D. Siewiorek, "Automated Synthesis of Data Paths in Digital Systems," *TCAD,* Vol. 5, 1986.

[9] F. Brglez and H. Fujiwara, "A neutral netlist of 10 combinational benchmark circuits and a target translator in Fortran," *ISCAS,* 1985.