社団法人 情報処理学会 研究報告
IPSJ SIG Technical Reports

2008－SLDM－135 (10)
2008／5／9

Fine-Grained Power Gating Based on the Controlling Value of Logic Gates

Lei CHEN†, Takashi HORIYAMA‡, Yuichi NAKAMURA*‡ and Shinji KIMURA†

†Graduate School of Information Production and Systems, Waseda University 2-7 Hibikino, Wakamatsu-ku, kitakyushu-shi, Fukuoka 808-0135 Japan

‡Saitama University

*‡NEC Corporation

E-mail: †chenlei@fuji.waseda.jp

Abstract: Leakage power dissipation of logic gates has become an increasingly important problem. A novel fine-grained power gating approach based on the controlling value of logic gates is proposed for leakage power reduction. In the method, sleep signals of the power-gated blocks are extracted based on the probability of the controlling value of logic gates without any extra control logic. A basic algorithm and a probability-based heuristic algorithm have been developed to implement this method. The steady maximum delay constraint has also been introduced to handle the delay overhead. Experiments on the ISCAS'85 benchmarks show the effectiveness of our algorithms and the effect on the extra delay.

Keyword *Power gating, Multi-threshold CMOS (MTCMOS) technology, BDD, Controlling value, Leakage power reduction.*

1. INTRODUCTION

As we know, leakage current increases exponentially as threshold voltage reducing. As a result, leakage power consumption is becoming a key issue in sub-100 nanometer process of LSI fabrication [1]-[4].

Multi-threshold CMOS (MTCMOS) technology, also known as power gating, has been accepted as an effective method to reduce leakage power consumption [1], [5]-[9]. In MTCMOS design, high-threshold voltage transistors are inserted as switches between the ground/power line and low-threshold voltage logic blocks, as shown in Fig.1. In the active mode, the high-Vth transistors are turned on so that low-Vth logic blocks can run at high performance, while in standby mode, high-Vth transistors are turned off to cut off the power of low-Vth logic blocks. The MTCMOS technique has been applied not only to static but also dynamic leakage power reduction.

The essential issues of MTCMOS are (1) how to generate sleep signals without additional control logic; (2) how to insert sleep transistors without too much area and delay overhead. In [10] and [11], a fine-grained power gating method has been proposed, in which local enable signals in a gated clock structure are adopted as control signals. These methods depend on explicit signals such as enable signals in gated clock or FSM, etc., therefore new methods are expected to manage general circuits without explicit enable signals. In this

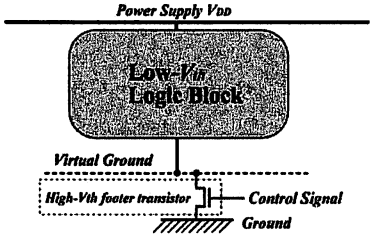


Fig.1. MTCMOS design structure for leakage power reduction.

manuscript, we propose a novel method to control sleep transistors in MTCMOS circuits based on the controlling value of logic gates. In the method, controlling value is used to justify whether some logic gates are necessary or not. Signals taking controlling value are assigned to unnecessary logic blocks as sleep signals, therefore, no additional logic gates are attached to the original circuits. This method is applicable to circuits without explicit enable signals. The granularity extracted by the method is much finer than previously proposed methods. A backward circuit trace algorithm has been developed to search the sleep gates. The probability of controlling value is also introduced for the selection of optimum power gating control signals and the probability is calculated using a Binary Decision Diagram (BDD).

The rest of the manuscript is organized as follows: Section 2 shows several definitions and the basic of

Binary Decision Diagram (BDD). Section 3 describes the basic algorithm and a probability-first heuristic algorithm of the proposed method. Section 4 shows applications of the proposed method to benchmark circuits. Section 5 concludes this manuscript.

2. PRELIMINARIES

2.1. Controlling Value of Logic Gates

A combinational logic circuit can be modeled as a directed acyclic graph (V,E), where V is a set of logic elements, and E is the connection of logic elements ($E \subseteq V \times V$). If a node v has no entering edge like (v', v), then it is called a primary input. If a node w has no emanating edge like (w, w'), it is called a primary output.

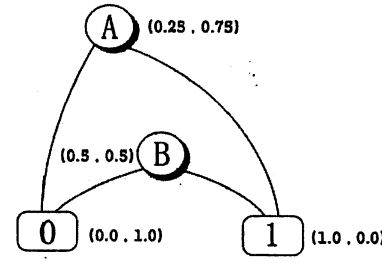
An n-variable logic function is a function from $\{0,1\}^n$ to $\{0,1\}$. In some cases, we call the n-variable function an n-input function. A NOT function is defined as {(0,1), (1,0)}, where each element represents a pair of an input value and the corresponding output value. A 2-input AND gate is defined as {((0,0),0),((0,1),0),((1,0),0),((1,1),1)}. From this definition, if one input takes the value 0, then output is 0 regardless of the other inputs. So 0 is called the controlling value of the AND Boolean function. The controlling value of a 2-input NAND function is also 0. The controlling value of a 2-input OR/NOR function is 1.

For a logic circuit (V,E), we specify the functionality of each element by defining a function Fv from V to a set of logic functions. Each node in a logic circuit is called a logic element or logic gate.

2.2. Binary Decision Diagram and Controlling Value Probability

A binary decision diagram (BDD) is a data structure used to represent a Boolean function [12]. Any Boolean function can be represented as a rooted directed acyclic graph, which consists of decision nodes and two terminal nodes called constant-1 node and constant-0 node. Each decision node is labeled by a variable and has two edges called 0-edge and 1-edge, representing an assignment of the variable to 0 (1). A BDD has 1 root node with no entering edge, and 2 leaf nodes with no emanating edges (constant-0 node and constant-1 node). BDD is a simplified representation of the truth table for the circuit, thus any circuit can be represented by a unique BDD with a specified variable ordering.

Note that each node in a BDD also represents a sub logic function. An example of a BDD with variables A and B is shown in Fig. 2. The top node represents the Boolean function ($A \cdot B$).



(1-probability , 0-probability)

Fig.2.An Binary Decision Diagram example.

For each node in a BDD, we can define 1-probability and 0-probability. The 1-probability of the constant-0 node is defined to be 0, and the 1-probability of the constant 1 node is defined to be 1. For a variable node v, we define the 1-probability using the 1-probability of nodes $v0$ and $v1$ connected to the 0-edge and 1-edge of node v.

$$P_1(v) = \frac{1}{2}(P_1(v_0) + P_1(v_1)) \qquad (1)$$

where $P1(v)$ denotes 1-probability of node v, and the 0-probability of node v is:

$$P_0(v) = 1.0 - P_1(v) \qquad (2)$$

3. POWER GATING BASED ON CONTROLLING VALUE

3.1. Basic Power Gating of Logic Gates

If some input of a logic gate takes the controlling value of the logic gate, the output is decided by this input, and values of other inputs become unnecessary, i.e., the input taking controlling value can be used to control the sleep transistors of logic blocks generating other inputs.

An example is shown in Fig. 3, in which signal a controls the power supply of Part B. Therefore, if signal a becomes 0, the switch is turned off so that the power consumed by Part B is avoided. The point of this method is that the control signals are extracted locally and no extra control logic or explicit enable signals are needed. The control granularity of the proposed method is much finer than the previously proposed MTCMOS methods.

The proposed controlling value based method allows us to dynamically cut the power supply to gates in logic circuits. Note that for an n-input logic gate, we actually have n candidates for the choice of sleep signal. Using Fig. 2 as an example, the logic block Part B can be power gated using signal a, while signal b can be adopted as the

sleep signal of Part A, too. However, we cannot control both of them at the same time without extra logic elements since the two inputs will run into an interlocking situation and cause an error. Therefore, we should select

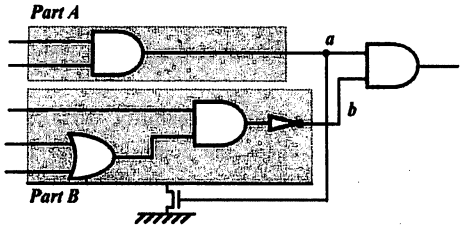


Fig. 3 Power gating with the controlling value of an AND gate.

one input out of others to be the control signal. To maximize the power reduction, we should select the signal which can reduce more power consumption.

On the other hand, if we focus on each gate, that might have several power-control signals. By selecting the best one, we can achieve the optimum case. Note that it might not be realized because of the interlock situation, but that is meaningful because it shows the theoretical limitation of this method. In the calculation of the power reduction limit, we choose the best control signal for each logic gate in the given circuit and ignore the interlocking situation, i.e., all the logic elements that are capable of being power gated will be controlled by the corresponding signal with the highest controlling value probability.

3.2. Algorithm Using Backward Trace

In the basic control algorithm, a backward trace is performed from primary outputs to primary inputs. In reality, we sort logic elements by level from primary inputs, and check one by one from higher level to lower level. In the following description, we assume that the number of inputs for each gate is 2, in fact, logic gates with more than 2 fan-ins can be divided into 2-input logic gates, like a 3-input AND gate can be treated as two 2-input AND gates.

For each gate, we compute two logic blocks that can be power gated by the two inputs respectively. The computation of the blocks can be done by a backward traversal procedure searching for controllable logic gates by checking the input gates of the current-checked logic gate recursively. The recursion stops at primary inputs and logic gates with more than 2 fan-outs. Note that if a gate's output signal is connected to 2 or more other gates, it is difficult to control the power of this element without

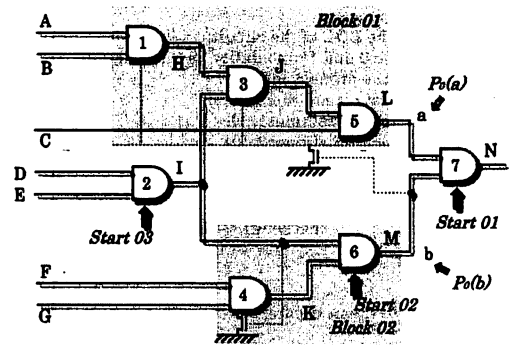


Fig. 4 An application example of the basic algorithm.

additional logic gates. Therefore, logic gates with fan-outs are not considered into sleep blocks in the basic algorithm.

Figure 4 shows an example of applying the basic algorithm. We start from gate 7, which is the primary output gate, and perform the backward trace. To compute Block 01, which should be controlled by signal *b*, gate 5, 3, 2, signal A and B are checked orderly. A and B are primary inputs and gate 2 has fan-outs to both gate 3 and gate 4, i.e., gate 5, 3 and 1 are clustered into Block 01. The same procedure is executed from gate 6 to compute Block 02 and gate 4 and 6 are clustered. A comparison is then performed on the numbers of logic gates in these two blocks. Block 01 is selected as the sleep block for gate 7 since it contains one more logic gate than Block 02. So we insert a footer sleep transistor to Block 01 and assign signal *b* to control this transistor. Next, all the logic gates in Block 01 are marked so that these logic gates will not be checked in the further procedure.

Then, we move to one of the remaining logic element of the highest level from primary inputs, which is gate 6 in the case of Fig. 4. The same procedure is executed and a block including gate 4 is selected as sleep block and is controlled by the output of gate 2. Finally, the only remaining gate 2 is checked and there is no controllable block for the gate, hence the procedure for the whole circuit is finished.

The pseudo-code of the basic algorithm is shown below in Fig. 5, together with the sub-function used to find sleep gates.

In the above procedure, we compute the maximum depth with the power gating and the expected number of gates whose power can be cut based on the controlling value probability of the control signal for each gate.

```
1: Construct the BDD of the circuit;
2: Compute the 0 and 1 probability for each signal;
3: Start from the primary output;
4: While the gate is not checked  {
5:     find_sleep_gate(input1, 1);
6:     find_sleep_gate(input2, 2);
7:     if(count_1 > count_2)  {
8:         for gate i in block 1  {
9:             ctrl_signal(i) <= input2;
10:            checked(i) <= 1;
11:            Ex_num <= Ex_num + P(i)
12:        }
13:    }
14:    move to the next gate.
15: }
16: Compute maximum depth
17: return;
```

```
Function: find_sleep_gate(signal j, label):
1: while (fanout(j) < 2 && checked(j) != 1
          && signal j is not primary input)  {
2:     label (j) <= label;
3:     count_(label) ++;
4:     find_sleep_gate(input1_j, label);
5:     find_sleep_gate(input2_j, label);
6: }
```

Fig. 5 Pseudo-code of the basic algorithm.

3.3. Maximum Depth Degradation

The insertion of sleep transistors in MTCMOS design may cause delay overhead to the original circuits because the critical path might be increased.

To measure the effect of the delay overhead in the proposed method, we compute the depth including the connection via the sleep transistors. In the circuit shown in Fig. 5, the maximum depth of the original circuit was 4, and the maximum depth paths are 1→3→5→7 and 2→3→5→7. By manipulating the connection via the sleep transistors as a usual input of a sleep logic element, we can define the maximum depth with the power gating. For the MTCMOS circuit in Fig. 5, the maximum depth path with the power gating is the following sequence of gates: 2→4→6→1→3→5→7. In this case, all gates are connected in series, and the maximum depth is 7. We will discuss a method without increasing maximum depth later.

3.4. A Heuristic Algorithm Based on Controlling Value Probability

In our basic algorithm, the controlling value probability is used to calculate the expected number of sleep gates (NE), which is the summation of the controlling value probabilities of control signals.

Since the controlling value probability is directly related to the power saving, it can be used to enlarge the expected number of sleep gates. In the basic algorithm, we check the control signals and the sleep blocks from primary outputs to primary inputs based on the level. In basic algorithm, we do not care about the controlling value probability, so the expected number of sleep gates might not be large.

So we develop a high-probability first heuristic algorithm, where we check the control signals from the highest probability to the lowest probability. At first, we compute the controlling value probability of each signal based on the type of logic gates using BDD-based method. The controlling value probability of each primary input is assumed to be 0.5. Then we search the sleep block with respect to the signal with the highest probability by using the same procedure in the basic algorithm. Note that gates in the sleep block and the gate generating the sleep signal are marked and will not be touched in the future process. Next we repeat the same process for the remaining signals with the highest probability until there is no signal capable of being sleep signal. With this algorithm, we can improve the expected number of sleep gates.

3.5. Steady Maximum Depth Constraint

The proposed power gating method introduces delay overhead. In the worst case, all gates might be connected in series. Since the delay overhead is from the power gating control, it can be managed by inhibiting the control.

Based on the heuristic algorithm, a mechanism is introduced that inhibits the usage of some sleep signals if the delay larger than the original circuit. This mechanism prevents the increase of the maximum delay.

The depth from primary inputs and that from primary outputs are computed at first. Let D_{PI} (v) be the depth of logic element v from the primary inputs, $D_{PO}(v)$ be the depth from primary outputs, and D_{MAX} be the maximum depth of the circuit. Note that if
D_{PI} (v)+$D_{PO}(v)$ = D_{MAX}, then v is on the longest path.

The depth based mechanism is added when we assign a signal w as the control signal for logic element v. D_{PI} (w) + $D_{PO}(v)$ + 1 is computed and compared to the maximum depth D_{MAX}, if larger than D_{MAX}, then signal w will not be assigned to logic element v as a sleep signal.

When connecting a sleep signal w to a node v, we check whether the maximum depth does not exceed D_{MAX} by D_{PI} (w)+$D_{PO}(v)$+1. If we adopt the power gating only when D_{PI} (w)+$D_{PO}(v)$+1 < D_{MAX}, then maximum delay is

Table 1 Application results of the basic algorithm and the probability-first algorithm.

			Basic Algorithm				Probability-First Algorithm				
Name	Number Of Total Gates	Original Max Depth	Number Of Controlled Gates	Number Of Sleep Signals	Max Depth	Expected Sleep Gates E(n)	Number Of Controlled Gates	Number Of Control Signals	Max Depth	Expected Sleep Gates E(n)	run time
C432	252	31	166	54	42	49.9(19.8%)	132	83	53	77.3(30.6%)	0.4
C499	454	19	190	120	37	74.4(16.4%)	184	126	35	97.2(21.4%)	0.8
C880	435	30	213	80	45	72.2(16.6%)	168	129	43	104.0(23.9%)	0.6
C1355	590	26	190	120	44	74.4(12.6%)	184	126	42	97.2(16.5%)	1.0
C2670	1400	40	758	281	68	282.2(20.2%)	687	440	66	470.4(33.6%)	2.4
C3540	1983	56	1273	307	85	672.1(33.9%)	1129	624	73	837.7(42.2%)	4.1
C5315	2973	52	1811	630	80	655.5(22.0%)	1612	1045	71	1058.2(35.6%)	10.2
C6288	2416	124	480	480	155	51.6(2.10%)	465	465	154	417.5(17.3%)	361.3
C7552	4042	45	2163	995	70	691(17.1%)	2047	1459	61	1270.5(31.4%)	10.4

the same as the original circuit, otherwise we should re-compute the depth of the related gates.

With the method, we can keep the maximum depth when applying the power gating, so the method is called the power gating under the steady state constraints. We will show experimental results on the expected number of sleep gates under the steady maximum delay constraints.

4. EXPERIMENTAL RESULTS

The proposed algorithms were implemented in C language and applied to ISCAS'85 standard benchmarks. The experiments were executed on a computer with an Intel Core 2 Duo 6600 CPU (2.40 GHz) and 2 GB of memory except for the benchmark C6288. All the experimental results, except for C6288, were obtained less than 10 seconds. For the benchmark C6288, a computer with 2.8 GHz×4 Operon CPU with 32 GB of memory was used. The runtime of probability-first algorithm for C6288 is 361 seconds in which about 350seconds are used to construct the BDD for the C6288circuit, i.e., only a few seconds are actually needed for the proposed algorithms.

The experimental results for the basic algorithm and the probability-first algorithm are shown in Table 1. This table includes the total number of elements (gates) that of controlled gates with its ratio to the total gate number, the number of control signals used, the expected number of sleep gates with its ratio, the maximum depth of the original circuit and the maximum depth of the power gating circuit.

The ratio of controlled gates ranges from 19.9% to 65.9%. The value of "Expected sleep gates" shows the expected number of sleep gates; in general, it is less than half the controlled gates. The expected sleep gates range from 12.3% to 33.5%. The number can be obtained by summing the controlling value probabilities of the controlling signals of all the gates if such a signal exists.

The maximum depth ranges from 1.3 to 1.9 times compare to the original depth of the benchmark circuits. For the probability-first algorithm, the expected numbers of sleep gates are significantly improved, ranging from 16% to 42%, however, the maximum depths also become even larger than the basic algorithm, showing a trade-off between power saving and delay overhead.

As is shown in these applications, the maximum depth increasing caused by the insertion of sleep transistors becomes very large. We also tested the expected number of sleep gates under the steady maximum delay constraint. The results are shown in Table 3. The "reduction ratio" of the controlled gates varies from 14% to 88%. However, there are still 2% to 30% power saving even we restrict that the maximum depths of resultant circuits are the same with the original circuits.

5. CONCLUSION

In this manuscript, a novel fine-grained power gating method based on the controlling value of logic elements is proposed. For a logic element, if one input takes the controlling value, the output of the logic element will be decided, i.e., the other inputs become unnecessary, so we can introduce the power gating technique to cut the power supply for logic blocks generating other inputs. A basic algorithm using backward trace is developed to search sleep block from primary outputs to primary inputs.

A highest probability-first algorithm is then introduced, in which the control signal is ordered with the controlling value probability and sleep block is decided based on the order. By this method, the expected number of logic elements can be significantly improved compared to the

Table 3 Effect of the steady maximum depth constraint.

	Circuit Information		No Depth Constraint		Max Depth Constraint			
Name	Total Gates	Depth	Expected Sleep Gates (S1)	Depth	Expected Sleep Gates (S3)	Decrease (S1 vs. S3)	Ratio (S3/T)	Depth
C432	252	31	77.3	53	19.4	74.9%	7.7%	31
C499	454	19	97.2	35	16.0	83.5%	3.5%	19
C880	435	30	104.0	43	60.8	41.5%	14.0%	30
C1355	590	26	97.2	42	16.0	83.5%	2.7%	26
C1908	1057	44	253.9	65	218.1	14.1%	20.6%	44
C2670	1400	39	470.4	66	261.9	44.3%	18.7%	39
C3540	1983	56	837.7	73	615.9	26.5%	31.1%	56
C5315	2973	52	1058.2	71	632.0	40.3%	21.3%	52
C6288	2416	124	417.5	154	46.2	88.9%	1.9%	124
C7552	4042	45	1270.5	61	732.9	42.3%	18.1%	45

basic algorithm.

According to the results of these algorithms, we found that the maximum depth becomes about 1.6 times as large as that of the original circuits on average. The maximum delay can be the same as original if the power gating control is introduced only when the maximum delay does not change. We show the effect of this restriction by experiments.

There is a trade-off between the delay overhead and the expected number of sleep gates, so we need to improve the algorithm under the delay constraints. BDD based probability computation might have the problem on the scalability, so we should consider the application of scalable probability computation method.

Acknowledgements

The work is supported in part by CREST Ultra Low Power Project of JSPS and by Grant in Aid for Scientific Research from JSPS.

This research was supported in part by "Ambient SoC Global COE Program of Waseda University" of the Ministry of Education, Culture, Sports, Science and Technology, Japan.

References

[1] B.H. Calboun, F.A. Honore, and A.P. Chandrakasan, "A leakage reduction methodology for distributed MTCMOS," IEEE Journal of Solid-State Circuits, vol.39, pp.818-826, May 2004.

[2] C. Kim and K. Roy, "Dynamic Vth scaling scheme for active leakage power reduction," Proceeding of Design, Automation and Test in Europe, 2002, pp.163-167, 2002.

[3] N. Kim, T. Austin, D. Blaauw, T. Mudge, K. Flautner, J. Hu, M. Irwin, M. Kandemir, and V. Narayanan, "Leakage current: Moore's law meets static power," Computer,vol.36, no.12, pp.68-75, Dec. 2003.

[4] L.Wei, "Design and optimization of low voltage high performance dual threshold CMOS circuits," Proceeding of 35th Design Automation Conference, pp.489-494, Jun. 1998.

[5] T.W. Chang, T.T. Hwang, and S.Y. Hsu, "Functionality directed clustering for low power MTCMOS design," Proceeding of the 10th Asia and South Pacific Design Automation Conference, pp.862-867, Jan. 2005.

[6] M. Anis, S. Areibi, M. Mahmoud, and M. Elmasry, "Dynamic and leakage power reduction in MTCMOS circuits using an automated efficient gate clustering technique," Proceeding of the 39th Design Automation Conference, pp.480-485, Jun. 2002.

[7] S. Shigematsu, S. Mutoh, Y. Matsuya, and J. Yamada, "A 1-V high-speed MTCMOS circuit scheme for power-down applications," Symposium on VLSI Circuits Digest of Technical, pp.125-126, 1995.

[8] 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, vol.30, no.8, pp.847-854, Aug. 1995.

[9] R. Vilangudipitchai and P. Balsara, "Power switch network design for MTCMOS," Proceeding of the 18th International Conference on VLSI Design, pp.836-839, Jan. 2005.

[10] K. Usami and N. Ohkubo, "A design approach for fine-grained run-time power gating using locally extracted sleep signals," IEEE International Conference on Computer Design, Oct. 2006.

[11] R. Bryant, "Symbolic boolean manipulation with ordered binary-decision diagrams," ACM Computing Surveys, vol.24, no.3, pp.293-318, Sep. 1992.