Maximum Current Estimation Considering Power Gating *

Fei Li
University of Wisconsin
ECE Department
Madison, WI 53706

Lei He
University of Wisconsin
ECE Department
Madison, WI 53706

ABSTRACT

As semiconductor technology scales down, the leakage power will soon become comparable to the dynamic power. To reduce both dynamic and leakage power, power gating in addition to clock gating should be used because clock gating saves only dynamic power. The knowledge of maximum current is needed to design high-performance and reliable circuits using power gating. However, all existing techniques for maximum current estimation are not applicable to power gating. In this paper, we study the maximum current estimation problem considering power gating. We develop two algorithms based on automatic test pattern generation (ATPG), and apply them to ISCAS'85 benchmarks. Experiments show that our new estimation algorithms can finish the largest benchmark circuit within ten seconds, and achieve up to 87% larger current when compared to an existing ATPG-based estimation algorithm that is able to obtain maximum current estimation 6% less than the theoretical maximum current without considering power gating. This implies that power gating may lead to a larger maximum current when compared to the normal maximum switching current, and open a new avenue for maximum current estimation as well as circuit reliability research.

Keywords

power estimation, ATPG, power gating, low-power design

1. INTRODUCTION AND MOTIVATION

The maximum current of an integrated circuit plays an important role in physical design. Excessive instantaneous currents may lead to electromigrations, IR voltage drops, or ground bounces on Vdd/Ground wires (in short, P/G wires). With ever-growing integration scales and clock frequencies, excessive currents may also lead to di/dt inductive noise especially for P/G wires. All these may cause permanent circuit failures, timing errors, or logic malfunctions.

The information of maximum current is needed to guide the physical design. Since the maximum current depends on the input-pattern, the maximum current estimation problem can be transformed into the weighted max-satisfiability problem, which is NP-hard [1]. Concepts of signal uncertainty and maximum current envelope were used to find the maximum current [2]. This approach was further improved by a branch and bound approach to consider the signal correlation [3]. Moreover, the genetic algorithm was adopted to search for a pair of input vectors that lead to the maximum current [4]. These methods in [2] [3] [4] are simulation-based. In addition, the Automatic Test Pattern Generation(ATPG) technique was used in [5] [6], and the timed-ATPG approach with consideration of delay impacts was used in [7]. In essence, all the aforementioned work finds a pair of input vectors which lead to the maximum switching current for combinational circuits. In general, a pair of vectors containing both primary inputs and circuit states can be found to obtain the maximum switching current for sequential circuits [8] [9] [10]. We can classify [1]-[10] as two-vector based approaches.

Power dissipation has become one of the primary constraints for high-performance microprocessor designs and mobile/embedded system designs [13]. Clock gating is effective to reduce the dynamic power that is the major component of the power dissipation for current CMOS designs [11] [12] [13]. The aforementioned work [1]-[10] is still applicable to estimate the maximum current considering clock gating. As semiconductor technology continues to scale down, the leakage power gains more significance in the total power dissipation. It is predicted that the leakage power will become comparable to the dynamic power in only a few generations [14]. Therefore, power gating in addition to clock gating should be used to reduce both leakage power and dynamic power [15], as clock gating is only able to reduce the dynamic power.

In Figure 1 (a), we show a circuit structure employing the power-gating technique. The PMOS *sleep transistor* with a high threshold voltage is used to turn on or turn off the Vdd supply to the conventional functional unit, which is a cascade chain of 5 inverters here.[1] The SPICE simulation results for a 0.13μ process technology are shown in Figure 1

*This research was partially supported by NSF CAREER Award CCR-0093273, SRC grant 2000-HJ-782, and a grant from Intel. We used computers donated by SUN Microsystems and Hewlett-Packard. Address comments to lhe@ece.wisc.edu.

ISPD'01, April 1-4, 2001, Sonoma, California, USA.

[1]For power gating, an NMOS sleep transistor may also be used to disconnect the ground from the functional unit.

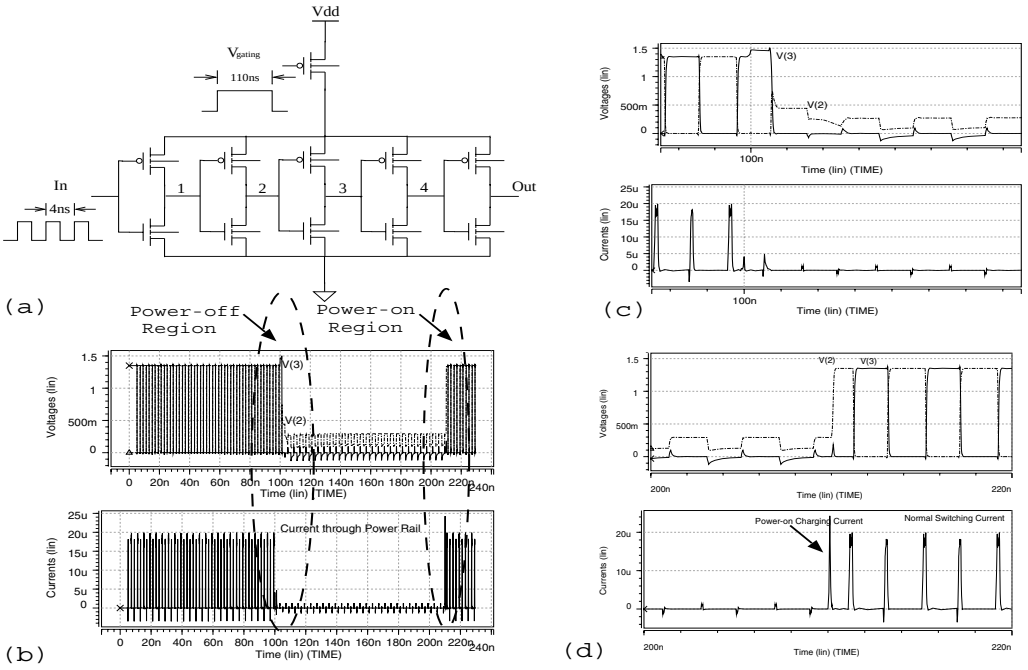

Figure 1: (a)A circuit employing power-gating; (b)Simulation results for the voltage of nodes 2 and 3, and the current in the power rail; (c)Zoom-in of power-off region and (d)Zoom-in of power-on region.

(b)-(d). The input to the inverter chain is a periodic signal as shown in Figure 1 (a). Control signal V_{gating} is used to turn on or turn off the sleep transistor, and therefore power on or power off the circuit. Figure 1 (b) shows the voltage $V(2)$ and $V(3)$ of two nodes 2 and 3, and the current in the power rail. In Figure 1 (c) and (d), the power-off and the power-on regions are zoomed in respectively. The first three pulses with significant amplitudes in Figure 1 (c) are the normal switching currents. After the circuit is powered off, V(2) and V(3) go down in only a few clock cycles.[2] Other nodes whose waveform are not shown in Figure 1 are also discharged quickly. When the circuit is powered on, there is a significant charging current, which is the first pulse with a significant amplitude in Figure 1 (d). The power-on charging current may affect the circuit reliability, as its amplitude is comparable to that of a normal switching current.

In general, the amount of charging current during power-on depends only on one input vector and is different from the switching current, depending on two vectors in the previous maximum current estimation work. Therefore, it cannot be solved by existing approaches such as [1]-[10]. In this paper, we study the maximum current estimation problem considering power gating, i.e. the problem to estimate the maximum turn-on charging current that depends only on one vector. Section 2 gives our one-vector problem formulation and describes the two ATPG-based algorithms in detail. Section 3 presents the experimental results and compares our result to the conventional maximum switching current. Section 4 concludes this paper. We consider combinational circuits in this paper. In this case, the one-vector is equivalent to one primary input vector.

[2]If the primary input signal is connected to a constant voltage source, the discharging time for nodes 2 and 3 is around $1\mu s$ according to our SPICE simulation for the temperature of 120^oC. Even in this case, the circuit can be discharged during a long enough sleep mode.

2. PROBLEM FORMULATION AND ALGORITHMS

2.1 Problem Formulation

When the circuit is powered on, the charge stored at the load capacitance needs to be charged quickly. In general, we may assume that the charging current is proportional to this total charge for all the load capacitance. Therefore, we use the total charge as the figure of merit to measure the maximum current in this paper. The total charge is given by

$$P_i = \sum_{\text{for all the gates}} VAL(g) \cdot C(g) \cdot V_{dd} \quad (1)$$

where $C(g)$ is the load capacitance of gate g, $VAL(g)$ is the logic value of gate output and V_{dd} is the supply voltage. If the gate output is logic "1", there is a charge of $C(g) \cdot V_{dd}$ stored in the load capacitance. For the simplicity of presentation, we assume that all gates have the same input capacitance and ignore the wire capacitance. Then the load capacitance is proportional to the fanout number of the gate. So we may simplify (1) as

$$P_i = \sum_{\text{for all the gates}} VAL(g) \cdot F_{out}(g) \quad (2)$$

where $F_{out}(g)$ is the number of fanouts for gate g. Similar analysis also applies to the charging current. In summary, we find one input vector which maximizes P_i, the figure of merit for the charging current.

2.2 ATPG Techniques in Estimation

For a circuit with n primary inputs (PI), the number of all possible two input vectors is $O(4^n)$ and the number of all possible one input vector is $O(2^n)$. Hence, it is not feasible to enumerate the input vector. In [5], a greedy-based scheme and the 9-Value D algorithm were used to generate two input vectors which maximize the switching current. In this paper, we use the similar greedy scheme to iteratively assign logic value "1" to the gate output which contributes most to the charging current. We use the PODEM algorithm [16] [17] to generate our one input vector because PODEM is much more efficient and fits our problem formulation very well. Before we describe the detailed algorithm, we first introduce several concepts and mechanisms in PODEM.

1. *Backtracing:* In a stuck-at fault Test Generation algorithm, a fault need to be *excited* and then its effects need to be *propagated* to the primary outputs. These two fundamental steps can be converted into a series of line-justification problems[17]. To justify the assignment to a line, instead of assigning the line directly, PODEM treats a value v_k to be justified for line k as an $objective(k, v_k)$ to be achieved via PI assignments. The *backtracing* process maps a desired objective into an assignment to one PI that is likely to contribute to achieving the objective. Hence PODEM assigns no values during *backtracing* process, which avoids logic conflicts and a consistency check. That's why PODEM is more efficient than other ATPG algorithms and why we choose it for our one-vector problem.

2. *Implication:* After the mapped primary input in backtracing is assigned with a value, the logic values of certain gates can be uniquely determined considering all the current assignments for the circuit. The process of computing logic values for these gates is referred to as *implication*. In implication, PODEM uses 3-valued logic, "0", "1", and "x"(unknown value). All lines including the output of primary inputs have initial value "x".

3. *Backtracking:* Whenever there are several alternatives to justify a line or to propagate a fault effect, it involves a *decision process*. If the decision made leads to the situation that the expected assignment can never be justified or the fault effect can never be propagated, a *backtracking strategy* is used to allow a systematic exploration of the completed space of possible solutions and *recovery* from previous incorrect decisions. PODEM uses the *reversing incorrect decision* technique to try an alternative decision and uses implication with value "x" to restore to the state existing before the incorrect decision[16]. This is much simpler than the backtracking mechanism used in the 9-Value D algorithm.

2.3 Fanout-based Algorithm Imax/Fanout

2.3.1 Algorithm Overview

In order to maximize P_i in (2), we iteratively assign logic value "1" to an unassigned gate with the largest fanout number in a greedy fashion. We first sort all the gates in non-increasing order according to the fanout number. Then each time, we pick up an unassigned gate g with the largest fanout number and assign value "1" to its output. To justify the assignment, we use the *backtracing* and *implication* procedures similar to those in the original PODEM algorithm (we will discuss the modification in Section 2.3.2). If justification fails for the current decision, we use *reversing incorrect decision* to backtrack until either the justification succeeds or the assignment is reported as impossible under the current circuit state after exploring the whole *decision tree*.

We summarize the *Imax/Fanout* algorithm in Figure 2. Some notations used there are explained as follows:

1. $obj(g, v_g)$: an objective to assign value v_k to the output of gate g;

2. $Justify(obj)$: a procedure which justifies the assignment;

3. $Imply(obj)$: a procedure which completes both the assignment specified by the objective and the assignment of other values that can be uniquely determined.

```
Imax/Fanout Algorithm:
Order the gates by the fanout number;
While( ∃ unassigned gates )
begin
        Select an unassigned gate g with largest fanout;
        Generate an objective obj(g, 1);
        if(Justify(obj(g, 1)) == SUCCESS)
                Mark gate g as an assigned gate;
        else
                Imply(obj(g, 0));
                Mark gate g as an assigned gate;
end
```

Figure 2: Fanout-based Algorithm *Imax/Fanout*

2.3.2 Justification of Assignment

The justification of assignment is performed by the procedure $Justify(obj)$. It is a recursive procedure and its overview is shown in Figure 3. The objective assignment to be justified is mapped to the assignment of one PI by the subroutine $Backtrace(obj)$. $Backtrace(obj)$ is also a recursive subroutine, and it is different from that in the original PODEM algorithm [16] [17] in terms of how to select the path to backtrace. If the justification fails, we will reverse the incorrect decision and try to justify it by backtracing through another path. If all the possible paths have been tried and justification still fails, the objective assignment cannot be achieved under the current state of the circuit.

We illustrate our process of *Backtracing* by the example in Figure 4. If we have an objective $(g7, 1)$, there are several alternative ways to justify the assignment. In Figure 4, we give two possible paths along which we may backtrace and map the objective to certain PI assignments. We have to decide which path should be tried first. Because our goal is to maximize the total charge when the circuit is powered up, we will try the path which may help us achieve this goal. We do this by computing the *weight* of objectives generated during the recursive subroutine $Backtrace(obj)$. If the input objective is (g, g_v) and gate g has N unassigned fanin gates$(f_i, i = 1, 2, \cdots, N)$, $Backtrace()$ will generate all the objectives $(f_i, g_v \oplus i)$, where i is the *inversion parity* of gate g. The weight of objective $(f_i, g_v \oplus i)$ is the fanout

```
Justify(obj)
begin
        if(the objective obj has been achieved already)
                return SUCCESS;
        if(the objective obj cannot be achieved)
                return FAILURE;
        (j, v_j) = Backtrace(obj); /* j is a PI */
        Imply(j, v_j);
        if(Justify(obj) == SUCCESS)
                return SUCCESS;
        /* reversing incorrect decision */
        Imply(j, v̄_j);
        if(Justify(obj) == SUCCESS)
                return SUCCESS;
        Imply(j, x);
        return FAILURE;
end
```

Figure 3: The procedure for justification : $Justify(obj)$

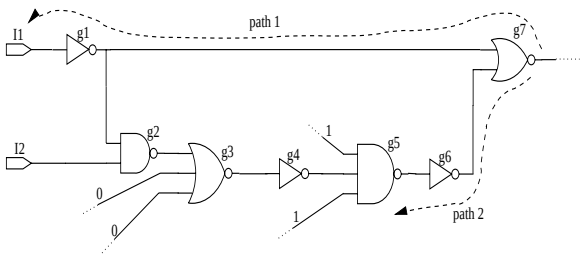

Figure 4: Backtracing for assignment justification.

number of gate f_i. If $g_v \oplus i = 1$, $Backtrace()$ tries the objective with the largest weight and then continues backtracing. If $g_v \oplus i = 0$, $Backtrace()$ tries the objective with the smallest weight and then continues backtracing. For example, when backtracing the objective $(g7, 1)$, $Backtrace()$ will generate two objectives: $(g1, 0)$ and $(g6, 0)$. Because $weight((g1, 0)) = 2$, $weight((g6, 0)) = 1$ and both have assignments of "0", $Backtrace()$ will try $(g6, 0)$ first.

The justification used in our algorithm is an implicitly exhaustive process. It explores the whole decision tree to justify assignment of value "1" so that the gate can contribute to the total charges. If the justification fails, it means the gate of interest now cannot have value "1" under the current circuit state. Therefore, the gate will be assigned with value "0". For example, in Figure 4, if we are going to assign value "1" to gate $g7$, no matter which path is tried and which PI assignment is mapped, the result is that $g7$ will be assigned with "0" instead. Thus $g7$ can only have value "0" under the current state of the circuit.

2.4 Gain-based Algorithm Imax/Gain

In the algorithm *Imax/Fanout*, we use the gate fanout number as the metric to determine the order of processing gates. But this heuristic rule is somewhat "local". We show a sub-circuit with fanout numbers in Figure 5. If we use the fanout-based algorithm, we assign "1" to the output of gate g first. But this assignment will imply that both $I1$ and $I2$ should be assigned with "0". So for the sub-circuit in the figure, we get $P_i = 5$. But if we assign "1" to both $I1$

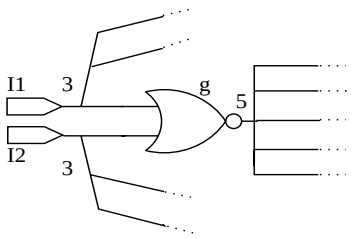

Figure 5: A simple example to illustrate the limitation of the fanout-based algorithm.

and $I2$, we get $P_i = 6$ even though the gate with the largest fanout number is assigned with "0". This occurs because we only use fanout number to determine the priority of the gate to be processed and this is a truly "local" observation. To observe more "globally", we define a new metric *gain* for every gate g which is to be assigned with value v as follows,

$$gain(g, v) = (-1)^{(v+1)} \cdot F_{out}(g) + \sum_{h \in IMP} ((-1)^{V(h)+1} \cdot F_{out}(h)) \quad (3)$$

IMP is the set of all the gates whose output values can be uniquely determined by the implication process of the assignment of gate g. $V(h)$ is the output value of gate h and F_{out} is the fanout number of gate h. (3) can be used to compute the *gain* to the total charge for a gate by assigning "0" or "1" to the output of the gate, respectively. In other words, each gate has two *gains*. For example, in Figure 5 we have $gain(g, 1) = -1$, $gain(g, 0) = -5$. Also we have $gain(I1, 1) = 3$ and $gain(I1, 0) = -3$. The gain for $I2$ is same as that for $I1$. So in our gain-based estimation algorithm, we will try to assign "1" to $I1$ or $I2$ first and avoid being trapped at the local optimum for the example in Figure 5.

```
Imax/Gain Algorithm:
Compute the initial gain for all the gates;
Order gates by gain;
While( ∃ unassigned gates )
begin
        Select the gate g and its assignment v
        with the largest gain;
        Generate an objective obj(g, v);
        if(Justify(obj(g, v)) == SUCCESS)
                Mark gate g as an assigned gate;
        else
                Imply(obj(g, v̄));
                Mark gate g as an assigned gate;
        Update the gain for all the unassigned gates;
end
```

Figure 6: Gain-based Algorithm *Imax/Gain*

We summarize the gain-based algorithm *Imax/Gain* in Figure 6. The *Imax/Gain* algorithm and the *Imax/Fanout* algorithm share a similar greedy-based scheme. The differences between these two algorithms include: First, in the *Imax/Gain* algorithm each gate g is put into the ordered list according to the larger value of its two *gains*, $gain(g, 1)$ and $gain(g, 0)$. Second, because *gain* depends on the result of implication that in turn depends on the current state of

the circuit, we need to update the *gain* for all the unassigned gates each time we finish the assignment of one gate. Note that we need to compute both $gain(g, 1)$ and $gain(g, 0)$ during the updating. This difference is highlighted in Figure 6. To make this dynamic updating efficient, we use a data structure similar to the BUCKET data structure in the FM partition algorithm [18].

3. EXPERIMENTAL RESULTS

Both the *Imax/Fanout* algorithm and *Imax/Gain* algorithm have been implemented in the C language. We use ISCAS'85 benchmarks and a Linux workstation with a 450MHz Pentium III CPU to test the two algorithms.

We compare the results of our two algorithms for the one-vector problem formulation in Table 1. For each circuit, the larger estimation result is in bold-face. Compared to the *Imax/Fanout* algorithm, the *Imax/Gain* algorithm achieves up to 5.8% improvement (circuit C1355) for seven out of the ten ISCAS'85 benchmark circuits. This is mainly because the gain metric enables us to observe more "globally" while assigning logic values. For three circuits the *Imax/Fanout* algorithm gets the better result. Because both algorithms are heuristic, it is possible for either algorithm to achieve the better result.

According to the runtime comparison in Table 2, both algorithms can be finished within only a few seconds. Therefore, in practice we can run both algorithms and use the better estimation result. In general, the *Imax/Gain* algorithm takes a longer time than *Imax/Fanout* because the *Imax/Gain* algorithm needs to dynamically update *gain* after each assignment. But we also have two benchmarks, C2670 and C5315, where the *Imax/Gain* algorithm runs faster, because the *Imax/Gain* algorithm may reduce the occurrences of backtracking.

Circuit	runtime	
	Imax/Fanout	*Imax/Gain*
C432	0.08	0.12
C499	0.15	0.24
C880	0.22	0.3
C1355	0.37	0.93
C1908	0.53	1.48
C2670	2.02	1.87
C3540	1.63	6.23
C5315	4.72	4.48
C6288	3.48	6.13
C7552	9.25	9.63

Table 2: Comparison of runtime of the two ATPG-based Algorithm.

In Table 1, we also compare the result of our one-vector problem formulation and the result[3] for the two-vector problem formulation in [6], where the same estimation measurement P_i is defined. The result of the one-vector problem formulation is given by the larger estimation result from our two algorithms. The percentage of difference between the results of the two problem formulations is also shown in Table 1. For eight out of the ten benchmark circuits, our algorithms achieve up to 87% larger current compared to the algorithm for the two-vector problem. The estimated current in [6] is 6% less than the theoretical two-vector maximum current. This implies that power gating may lead to a larger maximum current and may cause more severe problems such as electromigration, IR voltage drops, ground bounces, and di/dt inductive noise for the P/G structure. Therefore, power gating brings more challenges in low-power and reliable circuit design, and our new one-vector problem formulation is needed to tackle the reliability constraint in modern power-efficient VLSI design using power gating.

[3]Our quick implementation of the MAX_P algorithm in [6] leads to estimated current values slightly smaller than those in [6], thus we use results form [6] for comparison.

4. CONCLUSION AND FUTURE WORK

In this paper, we have studied the maximum current estimation problem considering power gating. We have proposed two algorithms based on automatic test pattern generation (ATPG) to estimate the maximum charging current when we turn on the power supply. Experimental results using the ISCAS'85 benchmarks show that our algorithms are able to complete the largest example within ten seconds, and the maximum turn-on charging current considering power gating is up to 87% larger than the conventional maximum switching current without considering power gating. This convincingly shows that power gating may lead to even worse reliability problems for the P/G structure, such as electromigrations, IR voltage drops, ground bounces, and di/dt inductive noise. Our new problem formulation and algorithms for maximum current estimation can be used to guide both circuit design considering power gating [15] [19] and P/G structure planning and optimization.

Our current algorithms consider combinational circuits using power gating. We are extending this work to sequential circuits. We also intend to apply our estimation approach to architecture level simulation and optimization for di/dt inductive noise reduction [20].

5. ACKNOWLEDGMENTS

The authors wish to thank Professor Kewal Saluja at University of Wisconsin-Madison, and Dr. George Cai and Mr. Rakesh Patel at Intel for their helpful discussions.

6. REFERENCES

[1] J. W. Srinivas Devadas, Kurt Keutzer. Estimation of power dissipation in CMOS combinational circuits using boolean function manipulation. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 11(3):373–383, March 1992.

[2] H. Kriplani, F. Najm, and I. Hajj. Maximum current estimation in CMOS circuits. In *29th ACM IEEE Design Automation Conference*, pages 2–7, 1992.

[3] H. Kriplani, F.Najm, P. Yang, and I. Hajj. Resolving signal correlations for estimating maximum currents in CMOS combinational circuits. In *30th ACM IEEE Design Automation Conference*, pages 384–388, 1993.

[4] Y.-M. Jiang, K.-T. Cheng, and A. Krstic. Estimation of maximum power and instantaneous current using a genetic algorithm. In *Proc. IEEE Custom Integrated Circuits Conf.*, pages 135–138, May 1997.

[5] C.-Y. Wang and K. Roy. Maximum current estimation in CMOS circuits using deterministic and statistical techniques. In *Proc. of 9th Int. Conf. on VLSI Design*, pages 364–369, Jan. 1996.

Circuits	Circuit Function	Estimation (P_i)				
		one-vector				two-vector
		Imax/Fanout		*Imax/Gain*		MAX_P[6]
C432	Priority Decoder	214		**223**	(+21.86%)	183 (0%)
C499	ECAT	**228**	(+16.33%)	210		196 (0%)
C880	ALU and Control	**524**	(+35.05%)	517		388 (0%)
C1355	ECAT	650		**688**	(+86.96%)	368 (0%)
C1908	ECAT	872		**882**	(-1.76%)	898 (0%)
C2670	ALU and Control	1292		**1366**	(+17.66%)	1161 (0%)
C3540	ALU and Control	1527		**1545**	(+14.70%)	1347 (0%)
C5315	ALU and Selector	2644		**2668**	(+4.38%)	2556 (0%)
C6288	16-bit Multiplier	**2207**	(-24.18%)	1799		2911 (0%)
C7552	ALU and Control	3406		**3524**	(-0.90%)	3556 (0%)

Table 1: Comparison of Estimation Results.

[6] C.-Y. Wang and K. Roy. Maximum power estimation for CMOS circuits using deterministic and statistical approaches. *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, 6(1), March 1998.

[7] A. Krstic and K.-T. Cheng. Vector generation for maximum instantaneous current through supply lines for CMOS circuits. In *Proc. Design Automation Conf*, pages 383–388, June 1997.

[8] C.-Y. Wang, K. Roy, and T.-L. Chou. Maximum power estimation for sequential circuits using a test generation based technique. In *Proc. IEEE Custom Integrated Circuits Conf.*, pages 229–232, Apr. 1996.

[9] C.-Y. Wang and K. Roy. Estimation of maximum power for sequential circuits considering spurious transitions. In *Proc. IEEE Int. Conf. on Computer Design*, pages 746–751, Oct. 1997.

[10] F. Li, W. Zhao, and P. Tang. Improved ATPG-based maximum power estimation. *Chinese Journal of Computer-Aided Design and Computer Graphics*, 12(7):538–543, July 2000.

[11] M. Horowitz, T. Indermaur, and R. Gonzalez. Low-power digital design. In *Proc. Int. Symp. on Low Power Electronics and Design*, pages 8–11, 1994.

[12] V. Tiwari, R. Donnelly, S. Malik, and R. Gonzalez. Dynamic power management for microprocessors: A case study. In *Proceedings of the 10th International Conference on VLSI Design*, pages 185–192, 1997.

[13] V. Tiwari, D. Singh, S. Rajgopal, G. Mehta, R. Patel, and F. Baez. Reducing power in high-performance microprocessors. In *Proceedings of the Design Automation Conference*, pages 732–737, 1998.

[14] S. Thompson, P. Packan, and M. Bohr. MOS scaling: Transistor challenges for the 21st century. *Intel Technology Journal,Q3*, 1998.

[15] J. T. Kao and A. P. Chandrakasan. Dual-threshold voltage techniques for low-power digital circuits. *IEEE Journal of Solid-state circuits*, 35(7):1009–1018, July 2000.

[16] P. Goel. An implicit enumeration algorithm to generate tests for combinational logic circuits. *IEEE Transaction on Computers*, C-30(3):215–222, Mar. 1981.

[17] M. Abramovici, M. A. Breuer, and A. D. Friedman. oy wblock *Digital Systems Testing and Testable Design*. IEEE PRESS, 1990.

[18] C. M. Fiduccia and R. M. Mattheyses. A linear-time heuristics for improving network partitions. In *Proceedings of the 19th Design Automation Conference*, pages 175–181, 1982.

[19] J. Kao, S. Narendra, and A. Chandrakasan. MTCMOS hierarchical sizing based on mutual exclusive discharge patterns. In *Proc. Design Automation Conf*, pages 495–500, June 1998.

[20] Z. Tang, N. Chang, S. Lin, W. Xie, S. Nakagawa, and L. He. Ramp up/down floating point unit to reduce inductive noise. In *Workshop on Power-Aware Computer Systems*, Nov. 2000.