COSMOS: A Continuous Optimization Approach for Maximum Power Estimation of CMOS Circuits [*]

Chuan-Yu Wang and *Kaushik Roy*, ECE, Purdue University, West Lafayette, IN 47907-1285

Abstract

Maximum instantaneous power in VLSI circuits has a great impact on circuit's reliability and the design of power and ground lines. To synthesize highly reliable systems, accurate estimates of maximum power must be obtained in various design phases. Unfortunately, determining the input patterns to induce the maximum current (power) is essentially a combinatorial optimization problem. Even for circuits with small number of primary inputs (PI's), it is CPU time intensive to conduct exhaustive search in the input vector space. The only feasible way is to find good upper and lower bounds of the maximum power, and to make the gap between these two bounds as narrow as possible. In this paper, we present a continuous optimization approach to efficiently generate tight lower bounds of the maximum instantaneous power for CMOS circuits. In our approach, each primary input (PI) of the circuit is allowed to assume any real number between 0 and 1. Maximum power estimation for CMOS circuits is then transformed into a continuous optimization problem, in which a smooth function is maximized over a unit hypercube in the Euclidean space. The continuous problem can be solved efficiently to generate good lower bounds of the maximum power. Our experiments with ISCAS and MCNC benchmark circuits demonstrate the superiority of this approach. For all the circuits tested, the mean value of the ratio "CPU time of the continuous optimization approach divided by CPU time of the simulation-based technique" is equal to 0.41. For 60% of the circuits tested, our approach gives a better estimate (1.16 times larger, on an average) than the simulation-based technique does. Compared to the ATPG-based technique [3], the continuous optimization approach generates a tighter lower bound (1.19 times larger, on an average) of maximum power for 60% of all the circuits tested.

1 Introduction

Maximum power estimation in CMOS circuits is essential to determine the IR drop on supply lines and to optimize the power and ground routing. The estimation involves searching for two consecutive binary input vectors, which maximize the switching activity in the circuits. In the worst case, this problem has complexity exponential to the number of PI's (Primary Inputs) of the circuit. Hence, for large-scaled circuits, the only feasible approach is to generate tight upper and lower bounds of the maximum instantaneous power within reasonable amount of CPU time so that the gap between the bounds is as narrow as possible.

In the past, several approaches have been proposed to cope with maximum power estimation for CMOS circuits. In [5, 6], Kriplani et. al. propagated signal uncertainty through circuits to obtain a loose upper bound of the maximum power. The bound is then successively made tighter by partially enumerating the primary inputs (PI's). The progress of improving the upper bound is expected to be slow when the circuit has a large number of PI's. Moreover, without a good lower bound, the estimate of maximum power cannot be obtained with the upper bound alone. In [4], Devadas et. al. formulated power consumption of CMOS circuits as a *Boolean* function in terms of two consecutive primary input vectors. The function is then exactly maximized by a branch-and-bound technique. The process to obtain the expression of the *Boolean* function and to maximize it are both CPU time intensive. Hence, the application of this technique is limited to small circuits. In [3], Wang and Roy proposed an Automatic Test Pattern Generation (ATPG) based technique to efficiently generate good lower bounds of maximum power for large-scaled circuits. The application of this technique currently has its limitations. Some characteristics of logic gates (e.g. inertial delay) cannot be easily captured at the logic level via this technique.

In this paper, we present an approach which can efficiently generate tight lower bounds of maximum power for large-scaled circuits under arbitrary delay model. Traditional techniques estimate the maximum power of a circuit based on the operation of binary numbers (0 and 1). In our approach, the output of a logic gate is allowed to assume any real number between 0 and 1. A smooth function denoting power can be derived over a unit hypercube in the Euclidean space. Maximum power estimation is then transformed into a continuous optimization problem, which can be solved efficiently. Compared to the traditional estimation techniques based on random simulation, our approach can generate tighter lower bounds within much shorter CPU time. On the other hand, compared to the ATPG based technique, this approach can easily capture more characteristics of logic gates.

The paper is organized as follows. Section 2 describes and formulates the problem of maximum power estimation. Section 3 transforms the problem into a continuous optimization problem. In Section 4, we discuss some issues in solving the continuous problem. Experimental results are presented in Section 5. Finally, the conclusions are given in Section 6.

2 Formulation of the Problem

In this section, we first discuss how to calculate the transition count at the output of logic gates within a clock cycle. Based on such discussion, we then formulate the problem of maximum power estimation.

For synchronous circuits, let us assume that the switchings of PI's are synchronized at the leading edge of each clock cycle. However, spurious transitions may occur at internal gates due to different propagation delays through different paths. The output of a gate may have transient pulses (spurious transitions) before it is finally stabilized. Such spurious transitions consume power and hence, such phenomenon should be considered in the maximum power estimation.

In a CMOS circuit, the total energy dissipated due to two consecutive input vectors can be described as: $E = \frac{1}{2}V_{dd}^2 \sum_{\forall gate} C_{load}(g) * T(g)$, where $T(g)$ denotes the transition count at the output of gate g during the clock cycle, and $C_{load}(g)$ represents the capacitive load of g. Hence, the average value of the instantaneous power over the clock cycle is: $p_{ave} = \frac{E}{|T|}$, where $|T|$ denotes the length of a clock cycle. Under the assumptions that (i) instantaneous power is a continuous function of time, and (ii) $|T|$ is sufficiently small, it is reasonable to view p_{ave} as the instantaneous power during the clock cycle. $C_{load}(*)$ can be approximated by the fanout ($F(*)$) of logic gates. Therefore, instantaneous power is proportional to:

$$P(V_1, V_2) = \sum_{\forall gate} F(g) * T(g) \tag{1}$$

where V_1 and V_2 denote the two consecutive input binary vectors applied to the circuit.

*This research was supported in part by DARPA (under contract F33615-95-C-1625), by NSF CAREER award (9501869-MIP), IBM Corporation, AT&T/Lucent, and Rockwell Corporation

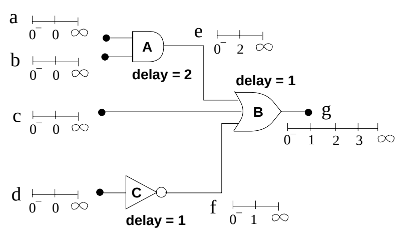

Figure 1: building time-sequences for a circuit

To calculate the transition count ($T(*)$) of a gate in a clock cycle, we associate with each gate a certain data structure to record the gate's output pattern. We call each such structure a *time-sequence* (a sequence of time instants at which the gate might switch). We implement the time-sequence of a gate as a linked list, in which each node represents a time instant at which the gate might switch. To avoid confusion, we refer to the nodes in time-sequences as *element*s. Each element in a time-sequence is associated with:

1. A start time, which denotes the corresponding switching time of the gate.
2. A logic value (0 or 1) which denotes the value at the output of the gate during a time interval beginning at the starting time of the element, and ending at the starting time of the next element.

The use of these structures will be described below.

Based on *static timing analysis* [10, 9], we have developed a procedure with linear complexity (in terms of the number of gates in the circuit) to create the time-sequences for a given circuit. By propagating the possible switching events level by level from PI's to PO's, the procedure can decide all the possible switching events at each node in the circuit, and equip the node with appropriate structure of time-sequence. Let us use the circuit in Figure 1 as an example. In Figure 1, each primary input (a, b, c, or d) is associated with two elements (with starting time 0^- and 0). Since the propagation delay of gate A is assumed to be 2, the time-sequence of node e is comprised of two elements with starting time 0^- and 2. Similarly, the time-sequence of node f can be built from that of primary input d. Finally, the time-sequence at the fanout of gate B is built by merging the time-sequences of node e, c, and f.

Based on the time-sequences, $T(g)$ in Equation 1 can be formulated as: $T(g) = \sum_{i=1}^{n(g)-1}(f_g(i) \oplus f_g(i+1))$. Here we use $n(g)$ to denote the number of elements in the time-sequence of gate g, and $f_g(i)$ to denote the logic value associated with the ith element of the time-sequence of gate g. Hence, we can define our problem as follows:

$$\text{Maximize} \quad P(V_1, V_2) = \sum_{\forall gate} F(g) * \sum_{i=1}^{n(g)-1}(f_g(i) \oplus f_g(i+1))$$

$$\text{Subject to } \mathcal{C} \equiv \{\textbf{spatial correlation between logic gates}\}$$

Note that $f_g(j)$ (denoting the value of the jth element of gate g) is decided by the value of the corresponding elements of the fanins of gate g. If we expand $f_g(j)$ from gate g toward primary inputs gate by gate, $f_g(j)$ can be represented as a Boolean function in terms of the entries of input vectors V_1 and V_2. $\mathcal{C}$ represents the structure of the logic circuit, and is the constraint set in this combinatorial optimization problem.

The above model describes the problem of maximum power estimation in circuits implemented with static CMOS gates. In the following section, we will discuss how to transform $P(V_1, V_2)$ into a function which is defined over the Euclidean space.

3 Transformation to Continuous Domain

We describe below a transformation which turns Boolean operators into arithmetic ones. After such transformation, $P(V_1, V_2)$ is well defined even if the entries in V_1 and V_2 are real numbers.

Any Boolean function can be transformed into an arithmetic function by repeatedly applying the following fundamental rules:

1. $a \vee b = a + b - ab$
2. $a \wedge b = a.b$ (a multiplied by b)
3. $\sim a = 1 - a$

where $\vee$, $\wedge$ and $\sim$ on the left-hand side are Boolean operators, while "+","-" and "." on the right-hand side are arithmetic operators. Any function transformed from some Boolean function according to these three rules preserves its original values at all vertices of $[0,1]^M$, where M is the number of dimensions of the function.

In the following discussion, we relax the entries in input vectors V_1 and V_2 from the discrete set $\{0,1\}$ to a continuous range [0,1]. To avoid ambiguity, let us use $\tilde{P}(V_1, V_2)$ to denote the function $P(V_1, V_2)$ after the relaxation. It can be noted that $\tilde{P}(*)$ is *continuous* and *differentiable* everywhere in $[0,1]^{2n}$, where n is the number of primary inputs of the circuit. Hence, the *gradient* of $\tilde{P}(*)$ (i.e. $\nabla\tilde{P}$) is defined everywhere in $[0,1]^{2n}$. This justifies our application of continuous optimization technique to $\tilde{P}(*)$.

Due to the smoothness and continuity of $\tilde{P}(*)$, we reach the following conclusion:

- Within the hypercube $[0,1]^{2n}$, if point a is sufficiently close to point b, the value of $\tilde{P}(a)$ and $\tilde{P}(b)$ tend to be the same. In other words, $||\tilde{P}(a) - \tilde{P}(b)|| \rightarrow 0$ if $||a - b|| \rightarrow 0$.

Suppose a relative maximum point v of $\tilde{P}(*)$ is sufficiently close to some vertex $\dot{v}$ of the hypercube $[0,1]^{2n}$. The vertex $\dot{v}$ tends to be a good choice to maximize the original combinatorial function $P(*)$ if v is sufficiently good for maximizing $\tilde{P}(*)$. This argument directly follows the above conclusion. Note that the maximum value of $\tilde{P}(*)$ over $[0,1]^{2n}$ is an upper bound of the maximum value of $P(*)$ over $\{0,1\}^{2n}$.

Since we are only interested in the discrete solutions, it is desired that the distance between the maxima (of $\tilde{P}(*)$) and vertices of the hypercube is as small as possible. Obviously there is no guarantee this is always the case. Hence, $\tilde{P}(*)$ is modified to favor this requirement. Some non-biasing term $T(*)$ is subtracted from the function $\tilde{P}(*)$ to make the maxima of $\tilde{P}(*)$ move toward the vertices of the hypercube $[0,1]^{2n}$. We call the term "non-biasing" because the term should vanish at the vertices. In all our experiments, we determine the non-biasing term as a summation of $2n$ concave functions: $T(X) = \sum_{i=1}^{2n} x_i(1-x_i)$. Here n is the number of primary inputs and x_i is the ith entry of vector X. It can be noted that $T(*)$ reaches its maximum value at $X = [0.5, 0.5, 0.5, ..., 0.5]$, and is also symmetric to this point. The resultant function $\widehat{P}(X) = \tilde{P}(X) - wT(X) = \tilde{P}(X) - w\sum_{i=1}^{2n} x_i(1-x_i)$ will be our objective function to be maximized. Here w is some positive real number used to adjust the shape of $\widehat{P}(*)$. For convenience, the notation $\widehat{P}(X) = \tilde{P}(X)$ - $w \sum_{i=1}^{2n} x_i(1-x_i)$ will be used in the rest of this paper.

Let X denote the concatenation of the two input vectors V_1 and V_2. From above discussions, the corresponding continuous problem for maximum power estimation can be defined as follows:

$$\text{Maximize} \quad \widehat{P}(X) = \sum_{\forall gate} F(g) * \sum_{i=1}^{n(g)-1}(\tilde{f}_g(i) + \tilde{f}_g(i+1) - 2\tilde{f}_g(i)\tilde{f}_g(i+1)) - w\sum_{i=1}^{2n} x_i(1-x_i)$$

$$\text{Subject to } \mathcal{C} \equiv \{\textbf{spatial correlation between signals}\} \cup \{x_i \in [0,1], i = 1(1)2n\}$$

where n is the number of primary inputs of the circuit. We solve this constrained optimization problem based on the gradient method, which will be discussed next.

4 Optimization

We use the *steepest descent* strategy to find the maxima of the above constrained optimization problem. Starting from the neu-

Table 1: Results of ISCAS and MCNC benchmarks under unit-delay model

Benchmark Circuit	Number of gates	Number of of levels	Maximum Power			CPU Time (sec)		
			COSMOS	ATG	SIM	COSMOS	ATG	SIM
i1	33	25	99	86	92	1.62	0.12	18.3
i2	36	201	257	280	260	32.5	2.22	62.7
i3	70	132	182	202	274	6.32	0.37	52.7
i4	94	192	352	346	299	21.1	0.54	67.2
i5	199	133	1063	666	723	39.2	1.04	134.5
i6	344	138	1408	1268	1257	56.9	3.06	217.1
i7	406	199	1615	1457	1445	110.6	3.91	248.8
i8	1183	133	5281	5796	5689	372.2	27.0	758.1
i9	353	88	2711	2589	2605	151.3	8.16	361.7
i10	2497	257	12340	11082	10033	2230.5	118.7	2147.9
C432	160	17	717	945	761	38.4	1.34	119.7
C499	202	11	312	674	476	13.3	2.03	81.4
C880	357	24	1706	1403	1027	137.5	5.13	195.2
C1355	514	24	1922	1543	1939	288.2	27.1	322.7
C1908	880	40	4365	3337	4356	459.9	56.4	783.2
C2670	1161	32	5310	4170	4123	423.2	14.7	903.2
C3540	1667	47	7232	5451	6865	1130.8	38.7	1404.6
C5315	2290	49	9685	10736	10160	1317.5	38.6	2488.8
C6288	2416	124	106392	102975	95406	4362.1	178.7	22578.2
C7552	3466	43	15563	14971	14344	1605.2	74.1	4627.4

tral point (i.e. [0.5,0.5,0.5,...,0.5]) of $[0,1]^{2n}$, the following procedure will be performed iteratively until one of the stopping criteria is met:

1. Calculate $\nabla\widehat{P}$ (gradient of $\widehat{P}(*)$) at the point.
2. From the point, move along the direction of $\nabla\widehat{P}$ as far as possible until $\widehat{P}(*)$ starts to decrease. Goto 1.

The above iteration stops under any of the following conditions:

1. The search reaches some relative maximum point at which $\nabla\widehat{P} \rightarrow 0$.
2. The search is stuck at some point on the boundary of the hypercube $[0,1]^{2n}$, where the gradient of $\widehat{P}(*)$ is normal to the boundary.

During the search, the curve gradually moves toward the boundary of $[0,1]^{2n}$. We keep on measuring the distance between the search curve and the vertices of $[0,1]^{2n}$. The following expression is used in our experiments to represent the distance: $d(P) = \sum_{i=1}^{2n} 0.5 - ||0.5 - p_i||$. Here P is a point on the curve, p_i denotes the ith entry of P, and n is the number of primary inputs. $d(P) \rightarrow 0^+$ denotes that P is sufficiently close to some vertex of $[0,1]^{2n}$. Under such a condition, the vertex will be reported as a satisfactory solution for maximizing the original combinatorial function $P(*)$ (i.e. the instantaneous power).

At each iteration of the gradient method, the gradient at a point X is calculated as: $\nabla\widehat{P}(X) = (\partial\widehat{P}/\partial x_1,\ \partial\widehat{P}/\partial x_2,\ \partial\widehat{P}/\partial x_3, ..., \partial\widehat{P}/\partial x_{2n})$, where $\partial\widehat{P}/\partial x_i = (\widehat{P}(x_1, x_2, .., x_{i-1}, x_i + \triangle x, x_{i+1}, ..., x_{2n}) - \widehat{P}(x_1, x_2, .., x_{i-1}, x_i, x_{i+1}, ..., x_{2n}))/\ \triangle x$, $\triangle x \rightarrow 0^+$. Determining the exact expression for $\widehat{P}(*)$ (in terms of primary inputs) is CPU time intensive especially for large-scaled circuits. However, it is not necessary to obtain the expression for $\widehat{P}(*)$ to calculate the gradient. Instead, the value of $\widehat{P}(X)$ at a point X is obtained by simulating the circuit with the input vector $V_1 = (x_1, x_2, ..., x_n)$ followed by $V_2 = (x_{n+1}, x_{n+2}, ..., x_{2n})$. Here x_i is the ith entry of X, and is a real number between 0 and 1.

During the steepest descent process, it is possible that the search is stuck at some local maximum which is far away from the vertices of $[0,1]^{2n}$. In this case, no conclusion can be drawn about which vertex is most likely to maximize the instantaneous power. Two approaches can be used to cope with this situation:

- Choose a different starting point (instead of using the neutral point) to start the steepest descent process. A different starting point may result in a local maximum which is close to some vertex of $[0,1]^{2n}$.
- Increase w (weight of the non-biasing term) to adjust the shape of the objective function. Using a larger w will reduce the distance between the local maxima of $\widehat{P}(*)$ and the vertices of $[0,1]^{2n}$.

The second approach is adopted in our experiments. In the 40 circuits tested, we need to use a different setting (larger) of w for 2 circuits to make the search converge. In the future work, we will investigate the possibility of making w adaptive to the structure of the circuit, as well as the delay model.

5 Experimental Results

The proposed continuous optimization approach (*COSMOS*), together with the ATPG-based estimation (*ATG*) [3] and the random simulation approach (*SIM*), has been implemented in *C* under the Berkeley *SIS* environment.

We assume all the benchmark circuits are implemented with static logic gates. In *SIM*, PI's (primary inputs) to a circuit are modeled as stochastic processes each associated with a *signal probability* (the probability of taking the logic value of ONE) and *activity* (average number of transitions per unit time). To stress the circuits, PI's are specified to have signal probability and activity of 0.5 and 0.9 respectively, in order to ensure high switching activity at the inputs. We assume that the high switching activity at the inputs also produces high switching activity at the internal nodes of the circuit [1]. Without loss of generality, *unit* and *fanout* delay models are adopted in our experiments. In the *unit* delay model, each gate is associated with unit delay. In the *fanout* delay model, propagation delay of a gate is assumed to be the gate's fanout. Delay of a gate is roughly proportional to the gate's capacitive load, which is in turn proportional to the fanout of the gate. It should be mentioned that our technique can handle any delay model.

Table 1 shows the experimental results on ISCAS85 and MCNC benchmark circuits under the *unit* delay model. The results for the same set of benchmark circuits under the *fanout* delay model are shown in Table 2. For benchmark circuits $C6288$

Table 2: Results of ISCAS and MCNC benchmarks under fanout-delay model

Benchmark Circuit	Number of gates	Number of of levels	Maximum Power			CPU Time (sec)		
			COSMOS	ATG	SIM	COSMOS	ATG	SIM
i1	33	25	101	87	91	1.84	0.14	23.5
i2	36	201	260	270	250	35.5	2.06	78.0
i3	70	132	182	202	170	6.47	0.37	67.69
i4	94	192	352	346	288	12.8	0.56	88.2
i5	199	133	1063	666	642	20.9	1.02	168.6
i6	344	138	1408	1346	1248	25.6	3.34	240.2
i7	406	199	1615	1549	1481	60.0	4.10	284.7
i8	1183	133	7165	7516	6864	315.9	56.1	866.5
i9	353	88	4306	3106	5061	218.5	32.0	451.5
i10	2497	257	8847	15096	14692	5835.2	1616.4	2531.13
C432	160	36	828	944	831	91.6	32.2	140.1
C499	202	41	355	740	511	18.8	3.41	82.5
C880	357	60	1447	1238	1211	252.8	8.67	203.1
C1355	514	41	2424	1404	2251	129.3	47.0	336.7
C1908	880	33	4945	4235	5577	309.8	56.0	1091.5
C2670	1161	233	5129	5382	4768	597.5	34.2	1232.6
C3540	1667	50	7922	9002	10537	662.4	107.83	2102.0
C5315	2290	178	10945	13739	12290	689.7	61.8	3416.7
C6288	2416	32	105172	97023	118555	3658.4	386.7	32859.1
C7552	3466	207	15947	18279	17178	2381.9	191.4	6129.9

and $i10$ under the fanout delay model, we use a different setting of w to make the steepest descent process converge. This will take extra CPU time in the actual estimation process. The CPU time for these two circuits listed in Table 2 does not include this extra time.

In Tables 1 and 2, the instantaneous power of a circuit is calculated based on Equation 1: $P(V_1,V_2) = \sum_{\forall gate} F(g) * T(g)$. Here (V_1,V_2) is the input vector pair applied to the circuit; $F(g)$ is the fanout (denoting the capacitive load) of gate g, and $T(g)$ is the transition count at the output of gate g caused by V_1 and V_2. In $COSMOS$, the optimal choice(s) of (V_1, V_2) is obtained by searching the maxima of the function $\widehat{P}(V_1,V_2)$ over a unit hypercube in the Euclidean space. In the ATPG-based estimation, ATG maximizes the instantaneous power in the circuit by greedily assigning switchings to the outputs of the gates with large capacitive load. The switchings assigned are then justified by the ATPG technique [3]. In the estimation technique based on random simulation, SIM generates the optimal pair of (V_1,V_2) for stressing a circuit based on $10,000$ input patterns conforming to the signal probability of 0.5 and activity of 0.9 at PI's [3]. The CPU time is measured on a Sun *Sparc* 5 workstation.

From Tables 1 and 2, we can conclude the following significances of this continuous optimization approach ($COSMOS$):

1. Compared to the traditional simulation-based technique (SIM), this approach is superior in both speed and performance. In 24 out of the 40 circuits tested, this approach generates a tighter lower bound (1.16 times larger, on an average). For the 40 circuits, the mean value of the ratio "CPU time of the continuous optimization approach divided by CPU time of the simulation-based technique" is equal to 0.41.

2. Although $COSMOS$ converges not as fast as the ATPG-based technique does, our approach is favored under the following conditions:

   - To estimate the power of a circuit under certain complicated delay model that the ATPG-based technique cannot be easily applied.

   - In 24 out of the 40 circuits tested, this approach generates a better estimate of maximum power (1.19 times larger, average speaking) than the ATPG-based technique does. A lower bound of high quality can be decided by these two approaches together.

6 Conclusions

In this paper, we have developed a continuous optimization approach for the maximum power estimation of CMOS circuits. Experimental results show that this approach is superior to the traditional simulation-based technique in both speed and performance. For 24 out of the 40 circuits tested, our approach gives a better estimate than the simulation-based technique does. For the 40 circuits, the mean value of the ratio "CPU time of the continuous optimization approach divided by CPU time of the simulation-based technique" is equal to 0.41. In maximum power estimation, this approach is an alternative to the ATPG-based technique.

References

[1] C.Su, C.Tsui, and Alvin M. Despain, "Saving Power in the Control Path of Embedded Processors," *IEEE Design and Test of Computers*, Vol. 11, No. 4, pp.32-41, winter,1994.

[2] S.Manne, A.Pardo, R.Bahar, G.Hachtel, F.Somenzi, E.Macii and M.Poncino "Computing the Maximum Power Cycles of a Sequential Circuit," in Proc. of *DAC*, 1995.

[3] C.-Y.Wang, K.Roy, and T.-L.Chou, "Maximum Power Estimation for Sequential Circuits Using a Test Generation Approach," *Custom Integrated Circuits Conference*, 1996.

[4] S.Devadas, K.Keutzer and J.White, "Estimation of Power Dissipation in CMOS Combinational Circuits," *IEEE Custom Integrated Circuits Conf.*, 1990.

[5] H.Kriplani, F.Najm, and I.Hajj, "Maximum Current Estimation in CMOS Circuits," in Proc. of *DAC*, 1992.

[6] H.Kriplani, F.Najm, P.Yang and I.Hajj, "Resolving Signal Correlations for Estimating Maximum Currents in CMOS Combinational Circuits," in Proc. of *DAC*, 1993.

[7] R.Burch, F.Najm, P.Yang and T.Trick, "A Monte Carlo Approach for Power Estimation," *IEEE Trans. VLSI Systems*, Vol.1, No. 1, pp.63-71, March 1993.

[8] M.Xakellis, F.Najm, "Statistical Estimation of the Switching Activity in Digital Circuits," in Proc. of *DAC*, 1994.

[9] J.K.Ousterhout, "A Switch-Level Timing Verifier for Digital MOS VLSI ," *IEEE Trans. Computer-Aided Design*, Vol. CAD-4, pp.336-348, June 1985.

[10] R.B.Hitchcock, "Timing Verification and Timing Analysis Program," in Proc. of *DAC*, 1982.