Efficient Peak Power Estimation using Probabilistic Cost-Benefit Analysis*

Hadi Hajimiri, Kamran Rahmani, and Prabhat Mishra
Department of Computer and Information Science and Engineering
University of Florida, USA
{hadi, kamran, prabhat}@cise.ufl.edu

Abstract*— Estimation of peak power consumption is an essential task in order to design reliable systems. Optimistic design choices can make the circuit unreliable and vulnerable to power attacks, whereas pessimistic design can lead to unacceptable design overhead. The power virus problem is defined as finding input patterns that can maximize switching activity (dynamic power dissipation) in digital circuits. In this paper, we present a fast and simple to implement power virus generation technique utilizing a probabilistic cost-benefit analysis. To maximize switching activity, our proposed algorithm iteratively enables transitions in high fanout gates while considering the trade-off between switching of new gates (benefit) and blocking of gate transitions in the future iterations (cost) due to switching of the currently selected one. Extensive experiments using both combinational and sequential benchmarks demonstrate that our approach can achieve up to 64% more toggles (30.7% on average) for zero-delay model and improvements of up to 319% (109% on average) for unit-delay model compared to the state-of-the-art techniques.

I. Introduction

Power dissipation is a major concern for integrated circuits (ICs) due to various reasons such as increasing operating frequencies, design complexity growth, etc. [1] [2] [3]. Power dissipation is directly related to energy consumption (battery life) and the cost of cooling. If the cooling is not adequate or power dissipation is not controlled, it can lead to reliability concerns [5] [7]. Therefore, it is critical to accurately estimate the power consumption early in the design stages.

Power dissipation is one of the key considerations in CMOS-based digital designs. Dynamic and static (leakage) are the two major components in power dissipation of CMOS circuits. The dynamic power dissipation is directly related to the switching activity in the circuit. In this paper, we study the problem of maximizing the switching activity, known as *power virus*. The basic idea is to identify the assignments to the primary inputs of the circuit to maximize switching activity (toggles). This causes the peak instantaneous dynamic power dissipation. The power virus problem is important because the CMOS circuit should be designed to support the peak current supply in such a worst case scenario [2]. In case of combinational circuits, the complexity of power virus generation is NP-complete [8].

* This work was partially supported by National Science Foundation (NSF) grant CNS-1441667 and Semiconductor Research Corporation (SRC) grant 2014-TS-2554.

The power virus problem has been studied in two different ways. One set of approaches perform extensive circuit simulation using a large number of input patterns. The other set of approaches perform power estimation based on static analysis of circuit structure and input patterns without explicit simulation [9]. For example, Automatic Test Pattern Generation (ATPG) based techniques [4], [10], [11] try to maximize circuit switching activity without circuit simulation. These techniques sort the gates based on fanout (output capacitance) and then assign transition to the gate with the highest fanout. Fig. 1 demonstrates that relying on fanout number can greatly reduce the solution quality.

In this paper, we propose a novel probabilistic cost-benefit methodology that utilizes fanout and fanin cones of gates to guide what gate should be assigned in each iteration. The basic idea is to select the gate with highest fanout that is likely to create more transitions (benefit) than selecting other gates. At the same time, we need to consider the negative effect (cost) of selecting a gate that may block transitions for a set of gates in the future iterations. We have developed an algorithm that assigns a transition to a node with a largest fanout and performs backward justification and forward propagation to maximize the toggles. When performing justification, there are always many choices. Consequently, cost functions are used to generate assignments that are more likely to achieve the goal as quickly as possible. A major contrast between our approach and the work in [10] [11] [4] is that we allow non-switching assignments in addition to switching assignments. This enables the ATPG-based greedy algorithm to use our probabilistic cost-benefit analysis in order to prevent gate assignment decisions that cause several gates not to switch, and therefore generate better results. Our experimental results demonstrate that our approach can increase the switching activity by 64% compared to the state-of-the-art power virus generation techniques.

The rest of this paper is organized as follows. Section

II provides the problem formulation and overview of the related approaches. Section III describes our proposed power virus generation methodology. Section IV presents our experiments. Finally, Section V concludes the paper.

II. Background

A. Problem Formulation

The dynamic power consumption constitutes major a portion of the overall power consumption in CMOS circuits. Dynamic power can be computed by the following equation:

$$P = \frac{1}{2}.V_{dd}^2.f.\sum_{\text{for all gates}} N(g).C(g) \quad (1)$$

where $C(g)$ is the output capacitance of gate g, and $N(g)$ is the number of times gate g is switched from 1 to 0 (or 0 to 1) during a clock cycle. Clock frequency is denoted by f and V_{dd}^2 represents the supply voltage. The maximum power estimation problem is defined as to find the input patterns (test vectors) that can maximize the power consumption outlined in Equation 1. In this paper, we assume output capacitance is proportional to the gate fanout. Therefore, the power consumption over the course of applying the input vectors can be estimated by:

$$P \propto \sum_{\text{for all gates}} T(g).F(g) \quad (2)$$

where $T(g)$ is the number of times gate g is toggled by applying the successive inputs and $F(g)$ represents number of fanout of gate g.

The goal of power virus generation is to maximize the power consumption by maximizing gate switching (according to Equation 2). To exhaustively search for a power virus in a combinational circuit with n primary inputs, we need to consider 4^n possible two input vector sequences. The time complexity of trying all possible input vectors is $O(4^n)$ which is considered to be infeasible for circuits with large number of primary inputs. The time complexity for sequential circuits is even worse - $O(2^{mn})$ where m is the number of considered cycles.

B. ATPG-Based Estimation

Since the time complexity of exhaustive search exponentially grows with the circuit primary inputs, a practical solution is to find a tight lower bound for the maximum power estimation. Wang and Roy [10] proposed a power virus generation method using Automatic Test Patter Generation (ATPG) techniques. To maximize Equation 2, the gates are sorted by the output capacitance ($C(g)$) in non-increasing order. Next, the algorithm assigns transitions to gates with the largest fanout. To ensure that the assigned transition is feasible (i.e. does not conflict with previous assignments), they used D-Algorithm with modified justification technique [10]. D-Algorithm is a widely used test pattern technique for stuck-at faults. The D-Algorithm is comprised of three basic operations:

- Backtracing: It determines assignments of binary values (0 or 1) to the primary inputs to achieve an objective. For example, in case of stuck-at faults, it tries to find the input assignments such that specific wires (nets) are stuck-at-0 (or stuck-at-1).
- Implication: It is the process of justifying an assignment. For example, backward implication can infer gate inputs in many scenarios based on gate output values. Similarly, when the gate inputs are known, forward implication can help in determining the gate outputs.
- Backtracking: It allows alternate options when there is a conflict. A conflict happens when implications produce values that are not consistent. In such cases, alternate options are explored using backtracking.

The methodology proposed by [10] is a greedy approach that assigns transition to high-fanout gates first. Each of the assigned transitions are justified and during the justification process, values are assigned to gates connected to the gate being justified. When deciding what gate should be assigned next, it does not take into account the effect of other assignments made in the justification process to the gates connected to the chosen gate. We call this approach Largest-Fanout-First (*LFF*). In this paper, we present a novel ATPG-based power virus generation methodology that models and utilizes fanout as well as fanin cone of gates in the circuit and uses a probabilistic cost-benefit decision making process.

III. Probabilistic Cost-Benefit Method for Power Virus Generation

In this section, we first describe our proposed approach for combinational circuits. Next, we extend this approach to maximize switching activity for sequential circuits using unit delay model.

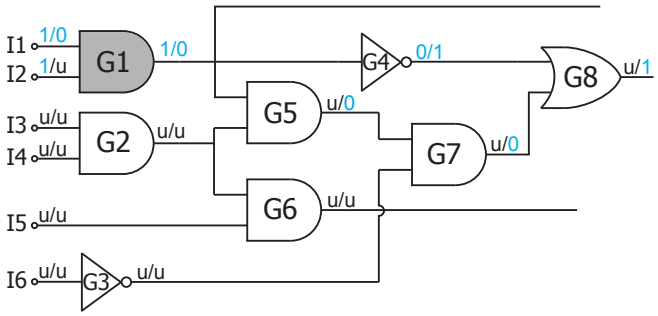
A. Zero-delay Maximum Switching Activity Estimation for Combinational Circuits

In zero-delay model, only steady state value of gates in circuit is considered when input vectors change. Therefore, each gate can at most switch once by applying two consecutive input vectors. For a zero-delay modeled combinational circuit, the computed power in Equation 2 translates to:

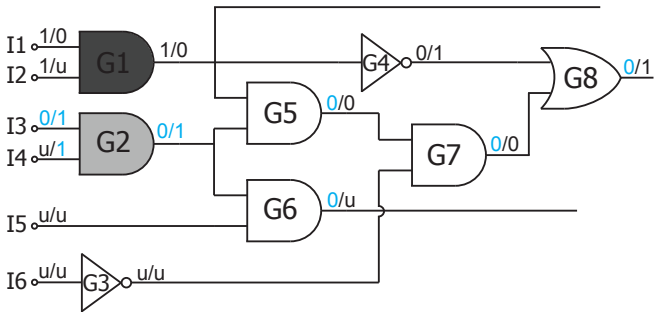
$$P \propto \sum_{\text{for all gates}} (g(v_1) \oplus g(v_2)).F(g) \quad (3)$$

where v_1 and v_2 are two consecutive vectors applied to the primary inputs of the circuit. Here, $g(v_1)$ and $g(v_2)$ represent the logical value of gate g after applying v_1 and v_2 respectively. $F(g)$ denotes fanout of gate g. To maximize circuit switching activity, ATPG-based power virus generation assigns switching values to high-fanout gates and traces back these assignments to primary inputs. After the

last assignment, the algorithm returns the input vectors, resulted from backtracings, that would generate the maximum switching activity (two consecutive input vectors for zero-delay combinational circuit model).



(a) *LFF*, iteration 1: pick *G1* and assign 1/0.



(b) *LFF*, iteration 2: pick *G2* and assign 0/1.

Fig. 1. Illustrative example using Largest-Fanout-First (LFF).

a) Limitation of Existing LFF-based Methods: Consider the circuit in Fig. 1(a) to illustrate the limitation of existing Largest-Fanout-First (LFF) approach as well as to motivate the need for our proposed approach. In zero-delay model, each gate in the combinational circuit can have a logical value pair $g(v_1)/g(v_2)$ where each one can be 0, 1, or u (unknown value). *LFF* approach starts with the gate with the highest fanout, *G1* with the fanout of 3 in this example, and tries to assign either a 0/1 or 1/0 transition. New assignments should not conflict with previous values of this gate. Since gate *G1* has unknown logical values for both $g(v_1)$ and $g(v_2)$, i.e. u/u, assigning a 0 or 1 would not conflict with previous values. In the first iteration of *LFF* approach, logical values 1/0 are assigned to gate *G1*. Each assignment has to be justified so that it does not conflict with values previously assigned to other gates. The justification process includes *backtracing* and *implication*. Assigning a 1 to AND gate backtraces 1 to all of its inputs while a 0 assignment backtraces 0 to only one of the inputs (*LFF* randomly chooses which input). In this example, assignment 1/0 to gate *G1* backtraces 1/0 to *I1* and 1/u to *I2*. It implies new values to other gates if value of them can be determined by the values of the new assignment. Therefore, *G4, G5, G7*, and *G8* are implied 0/1, u/0, u/0, and u/1 respectively. All new values resulted from the assignment in iteration 1 are highlighted in Fig. 1(a). If the assignment cannot be justified, a backtracking mechanism is used to try other available choices. *LFF* continues with other nodes and in each iteration it tries to assign a switching transition to the highest fanout gate in the remaining nodes until all nodes have been assigned. In the second iteration, it picks the next highest-fanout gate *G2* (with fanout of 2) and assigns a switching value 0/1. This assignment implies new values to *G5, G6, G7*, and *G8*. It also backtraces new values to *I3* and *I4*.

Note that *LFF* is a greedy approach in which once a decision is made and is justified, it will not be changed in future iterations. Therefore, assigning values to nodes may limit the available choices for future decisions. For example, after the second iteration of applying LFF to the circuit in Fig. 1(b) is complete, gates *G5* and *G7* are left with non-switching 0/0 values and it is not possible to change these values in the future iterations (when for example the algorithm pick *G5* in the next iteration). Among the unassigned gates, LFF chooses the gates based on the fanout number. As shown in Fig. 1, this may greatly reduce the solution quality reported by *LFF*.

b) Proposed Approach: To alleviate this problem, we propose to use fanin and fanout cones of gates in the decision making metric. In our approach, not only the fanout number is taken into the account but also the likelihood of switches in the fanin and fanout cones of the gate, that may occur by a switching assignment to the gate, is considered. Increased likelihood of switching and new switches caused by the assignment can be considered as benefit of the new assignment. Similarly, reduced likelihood of switching and non-switching assignments can be considered as costs of the new assignment. We define the likelihood of switching for a gate based on its input values as a *Switching Probability (SP)* function:

$$SP(g) = \frac{\text{number of fanin combinations resulting a switch}}{\text{number of combinations of values taken by } g \text{ fanins}} \quad (4)$$

This function estimates the probability that future assignments will cause gate g to switch. Initially, all gates and primary inputs have logical values u/u which means, in the future assignments, they can take any of 0 or 1 logical values for each of the first and second primary input vectors, $g(v_1)$ and $g(v_2)$. An unassigned fanin can take four different combinations (2×2). We call the switching probability calculated prior to any assignment the *Initial Switching Probability* ($SP_{initial}$). By each value assignment to a gate, we lose the degree of freedom and number of possible future values reduces. Assignments to a gate can increase or decrease the *SP* of the gate. Consider gate *G5*

TABLE I
G5 FANIN COMBINATIONS AND THEIR RESULTING OUTPUT.

G5 fanins		G5 output	G5 fanins		G5 output
G1	G2		G1	G2	
0/0	0/0	0/0	1/0	0/0	0/0
0/0	0/1	0/0	1/0	0/1	0/0
0/0	1/0	0/0	1/0	1/0	**1/0**
0/0	1/1	0/0	1/0	1/1	**1/0**
0/1	0/0	0/0	1/1	0/0	0/0
0/1	0/1	**0/1**	1/1	0/1	**0/1**
0/1	1/0	0/0	1/1	1/0	**1/0**
0/1	1/1	**0/1**	1/1	1/1	1/1

in circuit in Fig. 1. Initially, both *G1* and *G2* (fanins of *G5*) have u/u values. Each of *G1* and *G2* can have four different combinations. Therefore, there are $4 \times 4 = 16$ fanin combinations for gate *G5*. Table I shows the possible combinations of *G5* fanins and the resulting values of gate *G5*. Among all 16 combinations, only 6 fanin combinations can cause a switch in *G5* (highlighted in bold). Therefore, the initial switching probability of *G5* is:

$$SP_{initial}(G5) = \frac{6}{16} = 0.375$$

Assigning 1/0 to *G1* (Fig. 1(a)) fixes the values for *G1* as a fanin of *G5* and reduce *G5*'s future fanin value possibilities to four among which two can cause *G5* to switch. The SP of gate *G5* after the first iteration is:

$$SP_1(G5) = \frac{2}{4} = 0.5$$

We observe that this assignment increases SP(G5) from 0.375 to 0.5 meaning that it is now more probable to have *G5* switch in the future. We define the difference in switching probability of gate g at i_{th} iteration as:

$$\Delta SP_i(g) = SP_i(g) - SP_{i-1}(g) \tag{5}$$

For example, $\Delta SP_1(G5) = 0.5 - 0.375 = +0.125$ for gate *G5* for the first assignment made in Fig. 1. As we observe in Fig. 1, assigning values to a chosen gate may also assign values to other gates by the *justification* process. Therefore, we need to consider the fanin and fanout cones of the gate. Fig. 2 shows the conceptual illustration of fanin and fanout cones of gate *g*. The part of the circuit through which the logical information propagates from primary inputs to a gate is called fanin cone. The fanin cone is affected by the *backtrace* process. Similarly, the part of the circuit through which the logical information propagates from a gate to primary outputs is called fanout cone. *Implication* process affects gates in the fanout cone. *Backtrace* and *implication* are part of the justification process that is done after each assignment.

In the *LFF* approach, the output capacitance (fanout number) of only the chosen gate is the decision factor. It

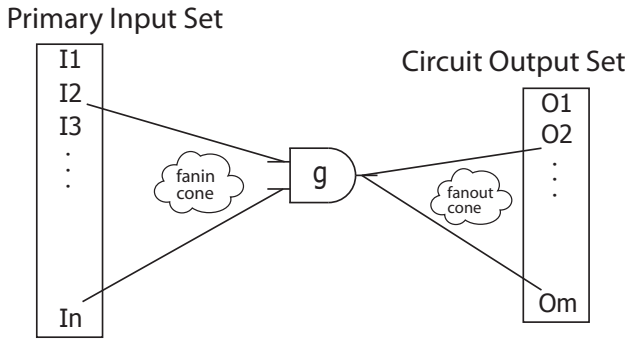


Fig. 2. Logic fanin and fanout cones (n primary inputs, m outputs).

essentially assumes by making the chosen gate switch, we benefit in the amount of the fanout factor of the gate. In this paper, we propose to use the changes made to all gates after each assignment. Each gate can have a benefit proportional to the gate fanout number and the amount of increase in its switching probability (ΔSP). We define the Cost-Benefit function for an assignment as:

$$CB(a_i) = \sum_{\text{for all gates}} \Delta SP_i(g).F(g) \tag{6}$$

Where a_i is the assignment made at the i_{th} iteration and $F(g)$ is the fanout number of gate g. This function sums the benefit, switching probability difference times fanout number, over all gates in the circuit. If an assignment increases the switching probability of a gate, ΔSP would be a positive number and $\Delta SP_i(g).F(g)$ is considered as the benefit. If it reduces the switching probability, ΔSP would be a negative number and thus is considered as the cost introduced by the assignment.

Algorithm 1 presents major steps of our power virus generation approach. It starts with the circuit with no assignments and picks gates based on their maximum cost-benefit (among different assignment values) in non decreasing order. In each iteration, to find the gate and assignment that provides the maximum cost-benefit, it tries all the gates that do not have final values and assigns various assignment values (0/0, 0/1, 1/0, or 1/1). The cost-benefit is computed based on the circuit after the trial assignment. After cost-benefit is calculated for each trial assignment, it uses the function *revert_assignment* to revert the assignment (including the assignments made in the justification process) so that the circuit is ready for another trial assignment. After trying all gates, it picks the gate and the assignment giving the maximum cost-benefit and finalizes it (the values will not change in future iterations) and start the next iteration. A major contrast between our approach and the traditional *LFF* is that we allow non-switching assignments in addition to switching assignments. It is possible to have gates structured in circuit in such a way that a switching assignment (0/1 or 1/0) will assign values to

gates in fanin/fanout cone (during justification) that prevent them from switching, resulting a large cost. Allowing non-switching assignments enables us to handle this problem. In cost-benefit computation, we only consider gates that are in the fanin and fanout of the gate under justification since the gates outside of these cones are not affected by the assignment.

Algorithm 1: Power Virus Generation

```
1  Input: Circuit C with list of PIs
2  Output: Maximum Power
3  while There exists an unassigned or partially assigned
   gate do
4      max_benefit = smallest integer
5      for each g in unassigned/partially assigned gates do
6          for assignment a in {0/0, 0/1, 1/0, 1/1} do
7              if a does not conflict with current values of
               g then
8                  Assign a to g
9                  if justify(g) then
10                     cb = Cost-Benefit //according to Eq.
                       6
11                     if cb > max_benefit then
12                         max_benefit = cb
13                         best_gate = g
14                         best_assignment = a
15                     end
16                 end
17                 revert_assignment(a)
18             end
19         end
20     end
21     Assign best_assignment to best_gate
22     justify(best_gate)
23 end
24 return ∑_{for all gates}(g(v1) ⊕ g(v2)).F(g)
```

B. Unit-delay Maximum Switching Activity Estimation

In order to model unit-delay switching activity, we unroll the circuit *level* times. Assume the *level* of a circuit is defined as the length of the longest path to an output, in terms of number of gates, starting from a primary input in the circuit. Switching assignments can be assigned to gates in two consecutive time stamps in the new circuit. Unit delay estimation always yields a better maximum power estimation (larger value) than zero-delay estimation. However, since we unroll the circuit, the time complexity of unit-delay is larger than zero-delay. Therefore, unit-delay modeling gives a trade off - better quality solution with longer execution time.

IV. Experiments

A. Experimental Setup

To evaluate the effectiveness of our technique, we have developed a cycle-accurate circuit simulator that models circuit nodes with a connectivity graph using C++. Back-tracing is done by traversing backwards on the directed graph. Similarly, the implication process is performed by both forward and backward traversal on the graph. In order to achieve backtracking we implemented a recovery mechanism to return to previously traversed graph states. To verify the correctness of our approach we measured the circuit switching activity using the final max power solution as the input vector. We used circuits from ISCAS'85 benchmark suites.

B. Results

Table II shows the maximum weighted circuit switching for ISCAS85 combinational benchmarks under zero-delay model. When comparing two different solutions, the solution that gives a higher maximum power estimate is considered to be a better quality solution. We compare our approach with the following four power virus generation methodologies.

- **Random** approach provides the maximum switching activity by random input vector simulations in a fixed timeout (1000 seconds).
- **LFF** [10] is an ATPG based approach that assigns values based on the fanout number in non-decreasing order.
- **CDPV** [4] improves LFF by using 0- and 1-controllability values to guide the justification mechanism.
- **PBS** [6] proposes pseudo-boolean satisfiability (PB-SAT) based circuit activity estimation.

Results in Table II show that our approach outperforms other methods by a large margin (in most cases). The column *Improvement* shows the percentage of improvement of our approach compared to the best solution (shown in bold) among the existing approaches. For example, our approach produced 64% more toggles compared to the best state of the art solution (1001 vs 610) for circuit *C1355*. Our approach did not perform well in case of *C499* benchmark for two reasons: high fanout and too many XOR gates. The average fanout in case of *C499* is 4.34. Such a high fanout number means an assignment affects more gates (both in the implication process and the backtrace process). Moreover, there are a lot of XOR gates in this benchmark. Backtracing an XOR gate requires all of its inputs to be assigned. Similarly, the implication process also gets affected. These two factors leave little room for the other gates to be explored. In fact, most of the gates will be assigned after a few gates are explored. As a result, simple heuristics such as as fanout based assignments perform better if a benchmark has high fanout or too many XOR gates.

TABLE II
ZERO-DELAY MAXIMUM POWER ESTIMATION

Circuit	Maximum Weighed Switching					
	Random	**LFF**[10]	**CDPV**[4]	**PBS**[6]	**Cost-Benefit**	**Improvement**
C432	187	183	**270**	193	**337**	25%
C499	215	196	303	**493**	373	-24%
C880	395	388	**582**	482	**728**	25%
C1355	402	368	**610**	480	**1001**	64%
C1908	**1084**	898	973	459	**1463**	35%
C2670	1427	1161	**1516**	775	**2086**	38%
C3540	1633	1347	**1727**	1058	**2810**	63%
C5315	2949	2556	**3007**	1689	**4202**	14%
C6288	3165	2911	2684	**3678**	**4517**	23%
C7552	**3855**	3556	3670	2620	**5568**	44%
Average Improvement over State-of-the-Art						30.7%

Table III presents our results for unit-delay circuit model. Among the compared approaches, only PBS supported unit-delay model. It can be observed that although PBS provides better results for small circuits it has the same drawback of most of SAT based approaches; it is computationally expensive. Since the required time increases exponentially with the circuit size it performs poorly on large circuits. Our approach provides considerably better results for large circuits compared to PBS approach.

TABLE III
UNIT-DELAY MAXIMUM POWER ESTIMATION

Circuit	Maximum Weighed Switching			
	Random	**PBS**[6]	**Cost-Benefit**	**Improvement**
C432	995	**1041**	**1424**	37%
C499	2126	**2900**	1155	-60%
C880	2556	**3196**	3032	-5%
C1355	2407	**2987**	**7691**	140%
C1908	2932	**3329**	**13949**	319%
C2670	2743	**2804**	**8672**	209%
C3540	6579	**6813**	**20345**	198%
C5315	7614	**8706**	**22555**	159%
C6288	**132209**	101921	**136517**	3%
C7552	**13925**	12744	**26437**	90%
Average Improvement Over State of the Art				109%

V. CONCLUSION

We presented a novel power virus generation method using a probabilistic cost-benefit function. We considered zero-delay as well as unit-delay delay models. Our framework considers both benefits of switching profitable gates and cost associated with negative impact on future assignments. To evaluate effectiveness of our approach, we have experimented with circuits from ISCAS85 and ISCAS89 benchmark suites. Our experimental results show significant improvements, up to 64% (30.7% on average) for zero-delay model and up to 319% (109% on average) for unit-delay model, compared to state of the art approaches.

REFERENCES

[1] International Technology Roadmap for Semiconductors (ITRS), *http://www.itrs.net*

[2] P. Morgado et al. "Generating realistic stimuli for accurate power grid analysis", ACM TODAES, 14(3), 2009.

[3] M. Xakellis and F. Najm "Statistical estimation of the switching activity in digital circuits", DAC, pp.728 -733 1994.

[4] K. Najeeb, K. Gururaj, V. Kamakoti, V. Vedula, "Controllability-driven Power Virus Generation for Digital Circuits", *VLSI Design*, 2007.

[5] F. Najm "A survey of power estimation techniques in VLSI circuits", IEEE Trans. on VLSI 2(4), pp.446 -455 1994.

[6] H. Mangassarian, A. Veneris, F. Najm, "Maximum Circuit Activity Estimation Using Pseudo-Boolean Satisfiability", *IEEE Trans. on CAD*, 2012.

[7] H. Kriplani , F. Najm and I. Hajj "Maximum current estimation in CMOS circuits", DAC, pp.2 -7 1992.

[8] A. Freitas , H. Neto and A. Oliveira "On the complexity of power estimation problems", ILWS, pp.176 -181 2004.

[9] N. Evmorfopoulos , G. Stamoulis and J. N. Avaritsiotis "A Monte Carlo approach for maximum power estimation based on extreme value theory", IEEE Trans. on CAD, 21(4), pp.415 -432 2002

[10] C.-Y. Wang and K. Roy "Maximum power estimation for CMOS circuits using deterministic and statistical approaches", IEEE Trans. on VLSI, 6(1), pp.134 -140 1998

[11] C. Wang and K. Roy "Estimation of maximum power for sequential circuits considering spurious transitions", ICCD, pp.746 -751 1997