A Pattern Generation Technique for Maximizing Power Supply Currents

Kunal Ganeshpure, Alodeep Sanyal, Sandip Kundu
Electrical and Computer Engineering, University of Massachusetts, Amherst
{kganeshp, asanyal, kundu}@ecs.umass.edu

***Abstract*—Max-current analysis is essential in power rail design and power supply switching noise analysis. Traditionally, maximum current from all CMOS gates are added together to compute maximum current level. This approach ignores all Boolean relationships. The problem of finding the input vector pair that will cause worst case current draw from the power rails when Boolean relationships are considered is an NP-hard problem. In this paper, we propose a Current Maximizing Pattern Generation (CMPG) algorithm which greatly reduces the computational complexity by using a parameterized branch-and-bound heuristic that prunes the search space by looking for a lower as well as an upper bound for maximum switching currents. When allowed to proceed indefinitely, the CMPG algorithm converges on an exact solution instead of finding upper and lower bounds. When coupled with a switch level SAT solver, CMPG can generate patterns for cell library characterization.**

***Index Terms*— Peak Current Analysis, Power Supply Current, Pattern Generation, Max-satisfiability**

I. Introduction

A major concern in present day VLSI circuits is the design of power and ground lines in a way that ensures design reliability and performance. Excessive currents can affect performance by causing voltage droop in the power rails, which in turn degrade the switching speed of CMOS circuits [1]. Excessive current-flow through metal conductors accelerates electro-migration and cause long term reliability problems [1]. Excessive current through die-package interface, namely the C4 pads can cause thermal meltdown of solder bumps and also cause reliability problems [2].

Therefore it becomes imperative to estimate current through power supply lines more accurately. The total current through power supply is constituted of two major components: current due to switching of CMOS gates and the CMOS leakage current. CMOS leakage current has been steadily rising due to increased sub-threshold and gate oxide leakages and has gained attention in the literature [3]. However, the switching current still dominates overall current consumption and accurate estimation of worst case switching current is the focus of this paper.

Current is drawn from the power rail when a node output switches from logic 0 state to logic 1, and to the ground rail when it switches from logic 1 to logic 0. An input pattern for a circuit with *n* inputs is defined as a vector of *n* excitations, where each excitation could be any one of four possibilities: *l (low), h (high), hl (high* to *low)* or *lh (low* to *high).* For different input patterns, different *transient current waveforms* are drawn at the contact points. The transient current waveform can be described by a piece-wise linear (PWL) function with a peak current value I_{peak}. Accurate estimation of the maximum current waveform at every contact point implies determining current waveforms corresponding to all possible input patterns. For a circuit with *n* primary inputs, we need to simulate for 4^n input patterns, since each input can be *l*, *h*, *hl* or *lh*. Therefore, estimating maximum current for a large CMOS logic network is computationally intractable because it implies that the number of simulations which must be performed in order to find the maximum current is *exponential* in the number of inputs to the network, which makes it fall into the *NP-hard* class of problems.

In this paper, instead of finding an *exact* solution to this NP-hard problem, we seek to establish a tight upper and lower bound for switching current estimates. This approach whittles down the computational complexity and makes a practical solution attainable. However, we retain the completeness of a NP-hard solver by maintaining a parameter η that can be controlled to close the gap between upper and lower bounds to achieve an exact solution at the obvious cost of extreme computation.

The rest of the paper is organized as follows: in section 2, we review previous work in this domain. In section 3 we describe our new approach of establishing upper and lower bounds for the maximum currents drawn from the supply rails. Section 4 narrates the results obtained through simulation of the proposed algorithm on ISCAS85 benchmark circuits. We conclude in section 5.

II. Related Work

The problems of estimating maximum current and the peak power dissipation for a CMOS circuit are mathematically identical in nature and have been addressed in literature with significant importance over the last decade. Previous studies of supply rail currents includes pattern independent approach by Kriplani *et al.* [4], which offer improvement in execution times compared to SPICE-based methods [5][6], but is overly pessimistic because Boolean filtering is not used. Chowdhury *et al.* [7] addressed the problem of maximum current estimation by partitioning the circuit into macro modules and

then applying the exact search technique or a suitable heuristic separately on each of them to come up with the solution. However, their methodology suffers from an over-estimation trend because of their assumption that all the macros draw their maximum currents simultaneously.

A set of different algorithms and their relative performance was studied by Jiang *et al.* [8]. They reported that for small circuits the ILP-with-partitioning approach provides tightest upper bound, and for large circuits the lower bound obtained by the GA-based approach seems to be most effective and outperforms the other two approaches *viz.* the timed-ATPG approach and the probability-based approach. They also observed that the timed-ATPG, probability-based, and ILP approaches are only applicable to combinational circuits, while the GA approach applies to sequential circuits as well. Estimation of worst case power dissipation in CMOS combinational circuits was studied by Devadas *et al.* [9]. They reduced the problem to weighted max-satisfiability problem on a set of multi-output Boolean functions followed by using either a *disjoint cover enumeration* algorithm or the *branch-and-bound* algorithm to solve the *NP-hard* problem. However, for a multilevel logic circuit, even under a unit gate delay assumption, the functions generated by their algorithm are fairly complex, and suffers from long execution time. Moreover, this approach requires that the problem be solved optimally – no sub-optimal solution can be used as upper bounds of switching activity. Chai *et al.* [10] formulated an integer linear program (ILP) based on the signal correlations within a circuit. They claimed a faster solution compared to [9] only through relaxing the constraints of the integer program. Hsiao *et al.* [11] proposed a peak power estimation tool, K2, that generates a specific vector sequence that produces maximum power dissipation in both combinational and sequential circuits. A Hamming distance-based approach was studied by Gupta *et al.* [12], where they estimate energy and peak current for every input vector pair.

Several ATPG-based approaches have also been reported in literature [13][14], which generate input vectors to either estimate the power supply noise or in the context of delay testing, where delay is seen as an effect of variable IR-drop across the power supply rails. Tirumurti *et al.* [1] proposed a fault model to address the problem of vector generation for delay faults arising out of power delivery problems.

III. The Proposed Approach

From the above survey, it becomes clear that existing methods for calculation of maximum currents in the power supply rails suffer from limitations such as long execution time or weak upper or lower bounds for maximum currents while handling large VLSI circuits.

In this paper, we propose a four step approach to simplify this problem:

a) Given the first input pattern I_1, generate the second input pattern I_2 that causes maximum supply rail currents

b) Provide a tight lower as well as upper bound for the supply rail currents for the input vector pair $<I_1,I_2>$

c) Solve the problem of finding an initial pattern I_1 that creates worst case $<I_1,I_2>$ combination. The mathematical solution of this step turns out to be exact equivalent of step (a) mentioned above. We will revisit this in section 3.2.

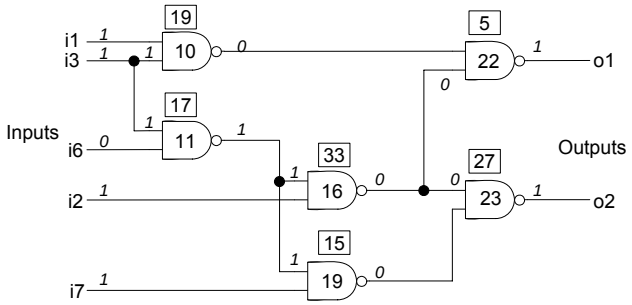

Fig. 1: c17 benchmark circuit with peak current weights associated with individual nodes

d) By observing sub-optimality of the solution obtained in step (c), which will be explained in details through an example later, propose a technique of concurrently generating the input vector pair $<I_1,I_2>$ which leads to a near-optimal solution. This technique is also based on the basic algorithm proposed for solving step (a) and will be discussed in depth in section 3.3.

To solve the steps (a) and (b), we propose a *branch-and-bound* heuristic which reduces the exponential complexity of the original problem by using a parameter η. The value of the parameter η is defined as the ratio between the target peak current and the maximum peak current that could be obtained starting from the first input pattern I_1. The following example explains the concept:

Example 1: An input pattern $I_1 = \{1,1,0,1,1\}$ is applied to the primary inputs of the ISCAS85 benchmark circuit c17 (shown in Fig. 1). The peak current weights (random values without any unit) associated with individual nodes are shown in square boxes on the top of the nodes in Fig. 1. The input pattern I_1 produces 0 at the output of nodes 10, 16 and 19. Therefore, maximum peak current weight that can be drawn from the power rail is $W_{max} = 19+33+15 = 67$ when all these three nodes switch to 1 on application of a second input pattern I_2. Now, if we set the parameter $\eta = 0.5$, then the proposed algorithm will search for finding an I_2 such that it switches at least those many nodes whose cumulative weight will exceed $\eta W_{max}=33.5$. A possible solution for $I_2=\{0,1,0,1,0\}$ with total switched weight = 34 for nodes 10 and 19 switching back to 1. ■

With this discussion on the parameter η, we now delve into a detailed study of the proposed algorithm.

Assumption(s): We are given an input netlist consisting of gates (cells) and weights associated with each gate (cell) corresponding to its peak current to V_{DD} or GND.

For solving steps (a) and (b), we proceed as follows:

Step 1: Through logic simulation of the given input pattern I_1, we obtain the set of internal nodes S_0 (S_1) that produce 0 (1) and W_{max} for the sets S_0 (S_1).

Step 2: Then we sort the above lists of internal nodes by the descending order of their peak current consumption measure (weight).

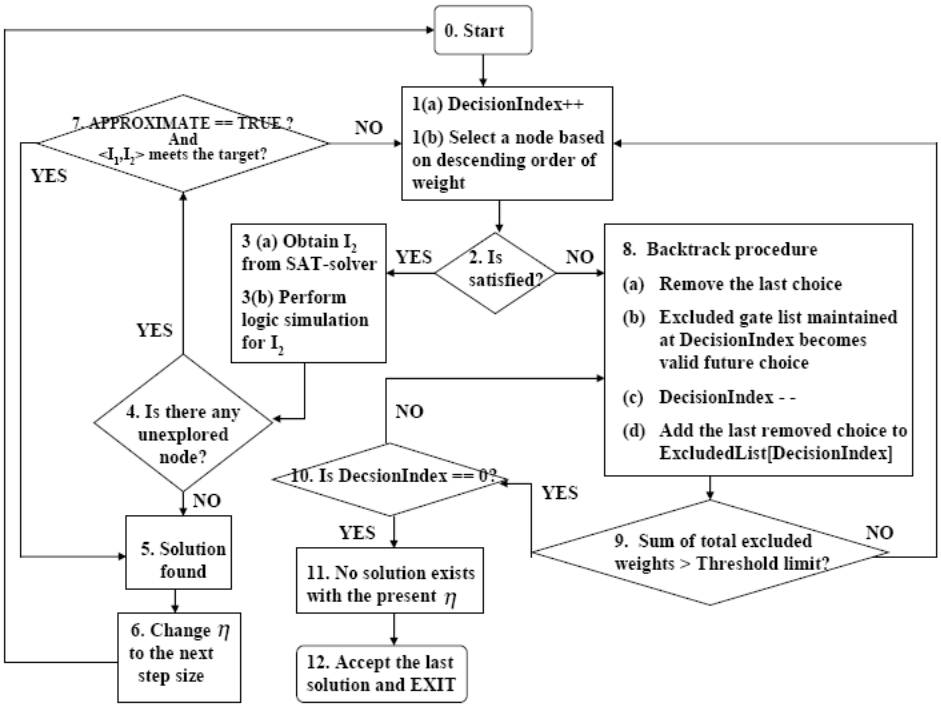

Fig. 2: The CMPG (Current Maximizing Pattern Generation) algorithm

Step 3: For a given I_1, we perform a *preprocessing* step by randomly choosing a finite number of candidates for input pattern I_2 and by logic simulation computing the total peak current drawn by the respective $<I_1,I_2>$ pair. During this process, we keep the maximum peak current noted so far and stop the preprocessing when no change is observed in the noted maximum peak current for 100 consecutive simulations. The ratio between the noted maximum peak current in the preprocessing step and W_{max} obtained in step 1 defines the initial value of η.

Step 4:: We then employ a greedy branch-and-bound heuristic which selects a subset *S'* of S_0 (S_1) and form a Boolean function. Then we invoke a SAT solver to determine whether *S'* is satisfiable.

Step 5: Based on the satisfiability result obtained from the SAT-solver, we grow the list of selected nodes by appropriate inclusions and exclusions while maintaining the lower bound established by η.

In stead of building our own SAT-solver, we used. *ZChaff* developed by Malik *et al.* [15][16] to illustrate the idea of bounding introduced by our algorithm. *Zchaff* accepts the Boolean expression, to be checked for satisfiability, in Conjunctive Normal Form (CNF). To extract the CNF form, the circuit topology is represented in the form of a *directed acyclic graph* (DAG) [17]. The nodes of the graph represent the circuit inputs, outputs and gates. This requires an extra step for converting the Boolean functions to CNF format and introduces inefficiency. However, our thrust in this paper is to show the practicality of the bounding approach in solving an NP-hard problem.

A. The Current Maximizing Pattern Generation (CMPG) Algorithm

One approach to solving a max-satisfiability problem is to use a *branch-and-bound* heuristic. We use a threshold parameter η as discussed above which sets the lower bound of the maximum supply rail currents. The CMPG algorithm is then invoked with the parameter η as obtained from the preprocessing step. The algorithm starts by selecting the node with the highest weight (say node 1) and invoking the SAT-solver to check its satisfiability, followed by exhaustively exploring all the combinations of this node with the other nodes. If none of the combinations satisfy the threshold condition set by η, the algorithm permanently excludes the node 1 from the list of valid future choices and starts exploring the combinations of the node with second highest weight and so on. This strategy of starting from the node with the highest weight becomes truly advantageous in pruning an appreciable number of useless searches and was studied by Devadas *et al.* [18][19] in the context of identifying and removing false paths in the domain of delay testing. The proposed algorithm allows user to supply termination condition with upper and lower bound.

Theorem: The algorithm CMPG is complete, i..e., given enough time and space it will be able to produce the list of gates that switch from 0 to 1 (or vice versa) to cause maximum supply rail current.

Proof: We prove the above theorem using induction method in the following way:

Base case: When there exists only one node, the algorithm chooses the node and invokes the SAT-solver to check its

satisfiability.

In case of two nodes $\{n_1, n_2\}$ arranged in descending order of weight, the algorithm chooses the node n_1 and checks whether it is satisfiable. If the SAT-solver returns TRUE, next it tries the combination $<n_1,n_2>$. Otherwise, the algorithm checks for the satisfiability of the node n_2. Therefore, for a set of two nodes, the algorithm explores all 2^2=4 combinations either explicitly or implicitly.

Inductive hypothesis: For *k* choice of nodes, the algorithm will visit all the 2^k combinations, either explicitly or implicitly *i.e.* the algorithm is complete with *k* choice of nodes.

Inductive step: Now for *k*+1 choice of nodes, the algorithm will check first whether the (*k*+1)th node is satisfiable, and irrespective of the satisfiability result, will recurse on *k* choice of nodes, which is the inductive hypothesis. Therefore, it will explore all the $2.2^k=2^{k+1}$ combinations.

Hence, the proof. ■

B. Pre-condition pattern generator

We find the first input pattern I_1 through the Pre-condition pattern generator which essentially invokes the CMPG algorithm with the list of all nodes and their associated weights as the input. Here our objective is to maximize the sum of weights of the nodes producing 0 (1) in the process of estimating the V_{DD} (GND) rail current. In order to achieve that goal, all the nodes of the circuit are initialized at 1(0) and then CMPG is invoked to find out the first input pattern I_1 which sets a subset (say, S_1) of all the nodes to 0 fulfilling the threshold set by η. Once I_1 is known, CMPG will be invoked again to work on the subset S_1 to find the second input pattern I_2 which will switch a further subset of S_1 (say, S_2) to 1 satisfying the threshold condition. The input vector pair $<I_1,I_2>$ together establishes the bound on the worst case supply rail switching currents for a given circuit. De-coupling the steps for generating pattern 1 and 2 makes them sub-optimal as explained next.

Example 2: We consider two separate input vector pairs applied to the benchmark c17 (shown in Fig. 1).

The first pair $<I_1,I_2>$ = {(1,1,0,1,1);(0,1,0,1,0)}. The first pattern I_1, obtained from CMPG, sets the nodes 10, 16 and 19 to 0 with an initial switching weight W_{init}= 19+33+15 = 67. Then CMPG is invoked again and it finds the second pattern I_2 which switches node 10 and 19 back to 1 thereby causing the final switched weight W_{final} = 19+15 = 34. Now let's consider the second vector pair $<I_1,I_2>$ = {(1,1,0,1,0);(0,1,0,0,0)}. The first pattern in this case sets only the nodes 10 and 16 to 0 with W_{init} = 19+33 = 52. We see that the initial switching weight here is lower compared to the previous case. However, the second pattern I_2 switches both the nodes back to 1 thereby causing the final switched weight W_{final} = 19 +33 = 52, which is higher compared to the previous case. ■

C. The CONCURRENT-INPUT-PAIR Algorithm

One way to solve the above sub-optimality is to concurrently generate the input vector pair $<I_1,I_2>$ to account for the correlation between the two patterns to manifest their combined effect in the final switched weight. The following five steps briefly describe our technique:

Step 1: Two copies of the circuit are created and merged to form a combined circuit.

Step 2: Logic simulation for the combined circuit is performed repeatedly on randomly generated input patterns until the maximum switched weight does not change for 100 consecutive patterns.

Step 3: The threshold parameter η is obtained from the maximum switched weight (obtained through logic simulation in step 2) in the way mentioned before.

Step 4: CMPG is invoked for the combined circuit with the threshold set appropriately by the parameter η obtained above. The CNF equations are formed in such a way that it tries to maximize 0's at the node outputs on one copy of the circuit and 1's on the same node outputs of the other. These CNF equations are then combined followed by invoking the SAT-solver.

Step 5: The input vector obtained through CMPG in step 4 is split into two halves which form the input vector pair $<I_1,I_2>$.

IV. RESULTS

Experiments were conducted on all 3 approaches described in the previous section. Results from these experiments are shown in Tables I-III respectively.

The lower bound is determined by starting with the base η value obtained from preprocessing step, incrementing it by a small step size of 0.05. We continue to increment this η value if SAT portion of CMPG returns with success. If SAT portion of CMPG results in a failure, lower bound on η is established. For upper bound, we start with η = 1.0, and decrement η with the same step size with each failure. A timer is used to terminate the process when the search takes too long for either the lower or the upper bound of η. The time-out limit was set to 5 hours.

In Table I, results from CMPG on ISCAS-85 benchmark circuits are presented for the case where the second input pattern I_2 is determined for a given first input pattern I_1. In this case the first input pattern I_1 was chosen randomly. For determining I_2, first random patterns were simulated to establish a lower bound for the maximum weight (that corresponds to current) that can be switched by simulation. Simulation of random patterns continues, until 100 consecutive patterns do not show any improvement of the lower bound. At that point, simulation is terminated and deterministic algorithm is invoked. The simulations already establish a lower bound on how much current can be drawn from the supply rail. We try to improve on that lower bound through deterministic branch and bound technique as described in section III A. The adaptive random simulation runs longest for c3540, where most current is drawn in pattern 249. In Fig. 3, we show how the relative η value changes during random logic simulation for the circuit c3540. In Table I, the η values are given in both relative and absolute sense, where by relative we mean relative to input pattern I_1, and absolute means relative to maximum supply rail current that

could be drawn if all nodes could be switched for the circuit. Same definition of η also applies for Table II and Table III.

TABLE I
EXPERIMENTAL RESULTS FOR GENERATING INPUT PATTERN I_1, GIVEN INPUT PATTERN I_2 FOR ISCAS-85 BENCHMARK CIRCUITS

Circuit Name	η from logic simulation			η obtained from CMPG			
	# of patterns	Rel. η	Abs. η	Lower Bound		Upper Bound	
				Rel. η	Abs . η	Rel. η	Abs. η
C17	103	1.00	0.529	1.00	0.529	1.0	0.529
C432	288	0.72	0.358	0.85	0.423	0.85	0.423
C499	240	0.46	0.314	0.50	0.341	0.70	0.478
C880	132	0.53	0.273	0.75	0.387	0.95	0.490
C1355	253	0.60	0.226	0.67	0.252	0.95	0.358
C1908	130	0.61	0.259	0.65	0.276	1.0	0.424
C2670	164	0.44	0.238	0.60	0.324	1.0	0.540
C3540	349	0.36	0.213	0.36	0.213	1.0	0.592
C5315	231	0.43	0.248	0.50	0.288	1.0	0.577
C6288	144	0.34	0.229	0.37	0.250	1.0	0.608
C7552	102	0.48	0.244	0.50	0.254	1.0	0.509

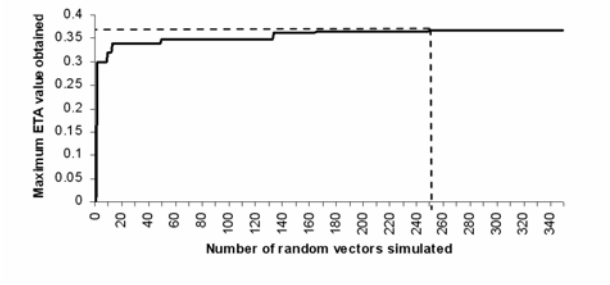

Fig. 3: Plot showing the gradual saturation of η with number of random pattern logic simulations for c3540

Table II shows the experimental results for the case when CMPG algorithm is used to generate both the input pattern I_1 and I_2 for ISCAS-85 benchmark circuits. Here random pattern simulation is done by generating random pattern pairs and stopping when the fraction of the total maximum current drawn from power supply rail does not change for 100 consecutive simulations.

TABLE II
EXPERIMENTAL RESULTS FOR INDEPENDENT GENERATION OF PATTERNS I_1 AND I_2 FOR ISCAS-85 BENCHMARK CIRCUITS

Circuit Name	η from logic sim	Abs. η for Pattern I_1			Abs. η for Pattern I_2		
		Logic sim. 1	CMPG Alg.		Logic sim. 2	CMPG Alg.	
			LB	UB		LB	UB
C17	0.543	0.543	0.543	0.543	0.543	0.543	0.543
C432	0.344	0.520	0.520	0.520	0.344	0.448	0.448
C499	0.312	0.730	0.852	1.000	0.414	0.473	0.639
C880	0.240	0.559	0.559	1.000	0.257	0.403	0.531
C1355	0.240	0.425	0.425	1.000	0.260	0.298	0.340
C1908	0.270	0.459	0.459	1.000	0.300	0.300	0.466
C2670	0.259	0.582	0.601	1.000	0.270	0.332	0.601
C3540	0.229	0.619	0.619	1.000	0.229	0.229	0.619
C5315	0.249	0.605	0.605	1.000	0.274	0.274	0.605
C6288	0.198	0.690	0.700	1.000	0.271	0.304	0.700
C7552	0.258	0.545	0.545	1.000	0.474	0.501	0.545

In the second phase, I_1 and I_2 are generated using algorithm presented in 3.2. For example for the circuit c880 the random logic simulation for the pattern pair (col. 2 in Table II) draws a fraction of 0.24 of the worst case supply rail current. The input pattern I_1 obtained from the Pre-condition pattern generator draws a fraction of 0.559 of the worst case current (columns 3 and 4 in Table II). Finally, by using the CMPG algorithm for the second time, we find an input vector pair $<I_1,I_2>$ which sets the lower as well as the upper bound of the worst case power supply rail current to be a fraction of 0.403 and 0.531 of worst case current. We also observe that the lower and the upper bounds established by this method is consistently higher than the bounds established by method 1 where input vector 1 was randomly chosen and the deterministic algorithm was used to find the second pattern.

In Table III, the CMPG algorithm is used to determine the pattern pair concurrently. For the circuit c880 the logic simulation gives the base value of η to be 0.263 (col. 3). Starting from this base η value, the CMPG algorithm is able to establish a lower bound of 0.309 of the worst case current.

Looking at the result for c432 in Table II and III, we observe that the upper bound obtained in Table II (the case for finding I_1, followed by I_2) is less than even the lower bound obtained in Table III (the case for finding $<I_1,I_2>$ concurrently), which establishes the notion of sub-optimality inherent in method 2 as discussed in Example 2 above. Using the same time out period results in looser bounds for this approach.

TABLE III
EXPERIMENTAL RESULTS FOR CONCURRENT GENERATION OF INPUT PATTERNS I_1 AND I_2 FOR ISCAS-85 BENCHMARKS

Circuit	Maximum weight that could be switched	Logic-simulation	Algorithm	
			Absolute LB	Absolute UB
C17	3.0705	0.543	0.543	0.543
C432	73.50	0.312	0.452	0.800
C499	96.38	0.316	0.512	0.849
C880	186.82	0.263	0.309	1.000
C1355	278.35	0.255	0.255	1.000
C1908	431.80	0.281	0.300	1.000
C2670	591.24	0.269	0.350	1.000
C3540	843.54	0.215	0.215	1.000
C5315	1154.62	0.251	0.299	1.000
C6288	1207.79	0.244	0.250	1.000
C7552	1724.74	0.266	0.300	1.000

V. CONCLUSIONS

In the context of peak supply rail current estimation, we have proposed an approximate solution for a computationally intractable problem. If allowed to run indefinitely, the approximate solution asymptotically converges to an exact solution. Given a circuit description, the algorithm also generates the input vector pair that produces near optimal worst case supply current. We studied three methods for finding out the input vector pair and compared their relative efficiency over each other. The solution can also be used to generate input patterns for cell library characterization. This will require a switch level SAT solver such as [20].

VI. REFERENCES

[1] C. Tirumurti, S. Kundu, S. Sur-Kolay, and Y.-S. Chang, "A Modeling Approach for Addressing Power Supply Switching Noise Related

Failures of Integrated Circuits," in Proc. Design, Automation and Test in Europe (DATE), 2004, pp. 1078-1083.

[2] D. Suryanarayana, R. Hsiao, T.P. Gall, and J.M. McCreary, "Enhancement of flip-chip fatigue life by encapsulation", IEEE Trans. on Components, Packaging, and Manufacturing Technology, Vol: 14, No. 1, Mar. 1991, pp. 218-223

[3] K. Roy. et al. "Leakage Current Mechanisms and Leakage Reduction Techniques in Deep-Submicron CMOS Circuits," Proc. of IEEE, Feb. 2003.

[4] H. Kriplani, F. N. Najm, and I. N. Hajj, "Pattern Independent Maximum current Estimation in Power and Ground Buses of CMOS VLSI Circuits: Algorithms, Signal Correlations, and Their Resolution," IEEE Trans. Computer-Aided Design, Vol. 14, No. 8, Aug. 1995, pp. 998-1012.

[5] A. Nabavi-Lishi and N. Rumin, "Delay and bus current evaluation in CMOS logic circuits," in Proc. IEEE/ACM Int. Conf. Computer-Aided Design, 1992, pp. 198-203.

[6] U. Jagau, "SIMCURRENT-An efficient program for the estimation of the current flow of complex CMOS circuits," in Proc. IEEE/ACM Int. Conf. Computer-Aided Design, 1988, pp. 208-211.

[7] S. Chowdhury, and J. S. Barkatullah, "Estimation of maximum currents in MOS IC logic circuits," IEEE Trans. Computer-Aided Design, Vol. 9, No. 6, Jun. 1990, pp. 642-654.

[8] Y.-M. Jiang, A. Krstic, and K.-T. Cheng, "Estimation for Maximum Instantaneous Current Through supply Lines for CMOS Circuits," IEEE Trans. VLSI Systems, Vol. 8, No. 1, Feb. 2000, pp. 61-73.

[9] S. Devadas, K. Keutzer, and J. White, "Estimation of Power Dissipation in CMOS Combinational Circuits Using Boolean Function Manipulation," IEEE Trans. Computer-Aided Design, Vol. 11, No. 3, Mar. 1992, pp. 373-383.

[10] D. Chai, and A. Kuehlmann, "Circuit-based Preprocessing of ILP and Its applications in Leakage Minimization and Power Estimation," in Proc. IEEE Intl. Conf. Computer Design (ICCD), 2004, pp. 387-392.

[11] M. S. Hsiao, E. M. Rudnick, and J. H. Patel, "Peak Power Estimation of VLSI Circuits: New Peak Power Measures," IEEE Trans. VLSI Systems, Vol. 8, No. 4, Aug. 2000, pp. 435-439.

[12] S. Gupta, and F. N. Najm, "Energy and Peak-Current Per-Cycle Estimation at RTL," IEEE Trans. VLSI Systems, Vol. 11, No. 4, Aug. 2003, pp. 525-537.

[13] Y.-M. Jiang, and K.-T. Cheng, "Vector Generation for Power Supply Noise Estimation and Verification of Deep Submicron Designs," IEEE Trans. VLSI Systems, Vol. 9, No. 2, Apr. 2001, pp. 329-340.

[14] A. Krstic, Y.-M. Jiang, and K.-T. Cheng, "Pattern Generation for Delay Testing and Dynamic Timing Analysis Considering Power Supply Noise Effects," IEEE Trans. Computer-Aided Design, Vol. 20, No. 3, Mar. 2001, pp. 416-425.

[15] M. Moskewicz, C. Madigan, Y. Zhao, L. Zhang, and S. Malik, "Chaff: Engineering an Efficient SAT Solver", in Proc. Design Automation Conference (DAC), 2001, pp. 530-535.

[16] S. Malik, Y.Mahajan, and Z. Fu, "ZChaff2004: An Efficient SAT Solver", Theory and Applications of Satisfiability Testing, Selected Revised Papers Series: Lecture Notes in Computer Science, 2004.

[17] T. Larrabee, "Test Pattern Generation Using Boolean Satisfiability", IEEE Trans. Computer-Aided Design, Vol. 11, No. 1, Jan. 1992, pp. 4-15.

[18] S. Devadas, K. Keutzer, and S. Malik, "Computation of Floating Mode Delay in Combinational Circuits: Theory and Algorithms," IEEE Trans. Computer-Aided Design, Vol. 12, No. 12, Dec. 1993, pp. 1913-1923.

[19] S. Devadas, K. Keutzer, S. Malik, and A. Wang, "Computation of Floating Mode Delay in Combinational Circuits: Practice and Implementation," IEEE Trans. Computer-Aided Design, Vol. 12, No. 12, Dec. 1993, pp. 1924-193.

[20] A. Kuehlmann, A. Srinivasan, and D. P. LaPotin, "Verity — a formal verification program for custom CMOS circuits," IBM J. Res. Dev. 39, 1-2 (Feb. 1995), pp. 149-165.