Pattern Generation and Estimation for Power Supply Noise Analysis

Mehrdad Nourani[1], Mohammad Tehranipoor[2], Nisar Ahmed[3]
[1]Dept. of EE, Univ. of Texas at Dallas, nourani@utdallas.edu
[2]Dept. of CSEE, Univ. of Maryland Baltimore County, tehrani@umbc.edu
[3]Texas Instruments, n-ahmed2@ti.com

ABSTRACT

This paper presents a new automatic pattern generation methodology to stimulate the maximum power supply noise in deep submicron CMOS circuits. Our ATPG-based approach first generates the required patterns to cover $0 \rightarrow 1$ and $1 \rightarrow 0$ transitions on each node of internal circuitry. Then, we apply a greedy heuristic to find the worst-case (maximum) instantaneous current and stimulate maximum switching activity inside the circuit. The quality of these patterns were verified by SPICE simulation. Experimental results show that the pattern pair generated by this approach produces a tight lower bound on the maximum power supply noise.

I. INTRODUCTION

A. Motivation

Power supply noise (PSN) due to switching current has become an important factor for deep submicron designs. This noise effect is becoming more detrimental as VLSI technology scales. As the number of interconnect layers and gate density increases, the switching activity increases which lead to increase current density and voltage drop along the power supply net. Increasing the frequency and decreasing the rise/fall transition time in today's designs causes more simultaneous switching activity within a small time interval and increases the instantaneous currents. The power supply noise reduces the actual voltage level reaching a device, which increases the signal delay and results in signal integrity loss and performance degradation. It may also cause logic errors, degradation in switching speed and hence timing errors.

PSN includes the inductive ΔI noise ($L * \frac{dI}{dt}$) and IR voltage drop. The former is derived from the distributed RLC model of on-chip power lines and the latter is caused by the switching inside the circuit as well as input and output buffers. Applying input patterns to a CMOS circuit creates the signal switching and causes the switching currents. To activate the switching in a circuit, a pair of patterns is required to be applied to the inputs of the circuit. Assuming there are n number of inputs, $2^n * 2^n = 2^{2n}$ number of pair patterns are required for an exhaustive search to find the pair of patterns that generate the maximum PSN. Therefore, applying all possible patterns to a circuit to find such pairs is possible only for the circuits with very small number of inputs. New techniques are needed to estimate power supply noise efficiently and find the pattern(s) that generate the maximum PSN in reasonable amount of time.

B. Prior Work

Several approaches have been proposed for power supply noise analysis and estimation in recent years. Some closed-form equations are derived in [1] to calculate simultaneous switching noise. Estimation of the ground bounce, caused by the switching in internal circuitry for deep-submicron circuits, using a scaling model is discussed in [2]. Reference [3] proposes a simulated switching circuit model to estimate PSN which includes IR voltage drop and ΔI noise based on an integrated package-level and chip-level power bus mode. Modeling of PSN on distributed on-chip power networks is described in [4].

ATE and neural network are used to find the patterns generating maximum instantaneous current [5]. The neural network is used to learn the behavior of chip power consumption and changes due to different input patterns applied. Several genetic algorithms for finding pattern(s) that stimulate the worst cases are proposed in [6-10]. In [7], the standard cells in the technology library are pre-characterized with SPICE to derive the delay and switching current waveform characteristics and a event-driven simulator along with a delay lookup table is used to perform timing analysis of switching events. A combination of Monte Carlo and genetic algorithm is employed to search for the worst case input vector pair(s) that induce the maximum switching noise.

The current waveform of the entire design is not a direct superposition of the individual block current waveforms when RC power/ground network is considered. The wire/substrate capacitances provide some of the current drawn and help in reducing the instantaneous current surge. The authors in [6][9] tackle this problem. Current/voltage waveform libraries for each cell in a library are derived using SPICE. A current waveform simulator is used to simulate a small set of patterns derived iteratively using a genetic algorithm. Finding the maximum voltage drop in the power bus of digital VLSI circuits using a genetic algorithm is discussed in [10]. In this work, the fitness value for different input vector pairs is the worst-case voltage drop at a specified node in the power bus.

C. Contribution and Paper Organization

In previous test pattern generation methods, impact of noise on the transient characteristics is not taken into consideration during the initial generation of current/voltage waveform libraries. Hence, the estimated noise level may not be accurate. On average 10% overestimate in noise voltage was reported compared to SPICE for $0.25\mu m$ technology [6]. This estimation error may increase as the technology scales down.

Proceedings of the 23rd IEEE VLSI Test Symposium (VTS'05)
1093-0167/05 $ 20.00 **IEEE**

IEEE COMPUTER SOCIETY

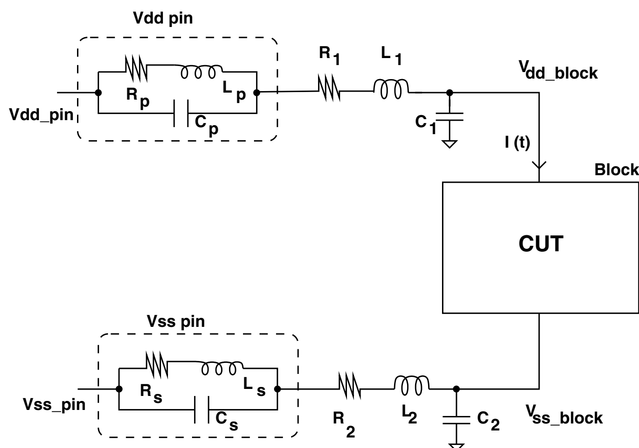


Fig. 1. The circuit model and power supply noise measurement.

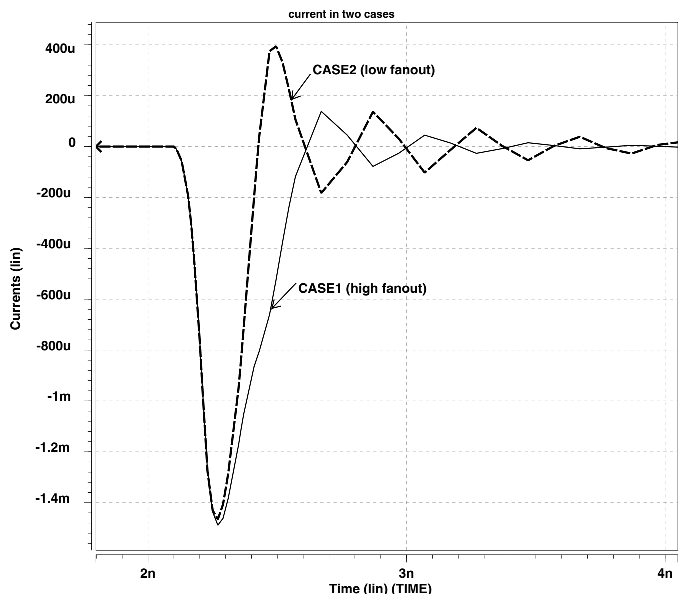


Fig. 2. Current $I(t)$ of the entire block in two different cases.

We propose a pattern generation algorithm that targets power supply noise. Our methodology employs common Automatic Test Pattern Generation (ATPG) technique applied to conventional stuck-at faults. With the aid of an ATPG our technique quickly and accurately identifies the transient characteristics of gates for a given pattern and its relationship with PSN. The pattern generation process is independent of the physical layout information and preprocessing of library cells and guarantees a tight lower bound for maximum PSN.

The rest of the paper is organized as follows. Section II describes power noise model that includes the effect of gate fanout on the maximum PSN induced in the circuit. The pattern generation strategy to obtain maximum PSN is explained in Section III. The algorithm and a small example are shown in Section IV. The experimental results are discussed in Section V. Finally, the concluding remarks are in Section VI.

II. Power Supply Noise (PSN) Model

In general, PSN includes two components: inductive ΔI noise and power net IR voltage drop and is given by $PSN = L * \frac{dI}{dt} + IR$. The inductive ΔI noise ($L * \frac{dI}{dt}$) depends on the rate of change of the instantaneous current, while the IR voltage drop is caused by the instantaneous current through the resistive power and ground network. The inductance is mainly due to package lead and wire/substrate parasitics.

Simultaneous switching of a large number of gates often induces a very large current spike on the power/ground lines in a short time interval. With low-k copper (Cu) interconnects being used in deep-submicron designs, the resistance of the wires is drastically reduced. This will generate considerable inductive noise $L * \frac{dI}{dt}$ even though the inductance L can be relatively small. The simulation results in literature, e.g. [8], shows that inductive noise dominates the resistive noise. For the worst case analysis, the idea is to generate the steepest maximum switching current spike. In order to create maximum switching noise, it is important to analyze the characteristics of the switching current waveform.

The switching current waveform of each gate is determined by the propagation delay (t_d) and its drive capacity (I_{max}). Empirical evidence shows that all switching currents last for approximately $3t_d$ and the peak drive current may slightly change for different capacitive load. The propagation delay is directly related to the fanout of the gate. Therefore, a gate with a smaller fanout has less propagation delay and hence a shorter current waveform duration, i.e the rate of change of current $\frac{dI}{dt}$ is higher. Hence, it induces greater inductive noise.

To illustrate the effect of fanout on power supply noise, we performed a simple experimentation. Consider a block consisting of 10 NAND gates which switch simultaneously in two different cases. In $Case1$, each gate has a fanout of 3 minimum-sized inverters while in $Case2$, each gate has a fanout of 2 minimum sized inverters. The circuit model used for power/ground pin and power/ground network is shown in Figure 1. Each V_{dd} and V_{ss} pin is modeled as an RLC circuit. The pin parasitics are R_p, L_p and C_p for V_{dd} pin and R_s, L_s and C_s for V_{ss} pin. The power/ground network is essentially modeled as a lumped RLC network.

The simulations are performed on the circuit implemented in $0.13\mu m$ technology. Figure 2 shows the SPICE simulation results for the block current waveforms in the two different cases. The variation of peak current value in the two cases is insignificant. The duration of the switching current waveform in $Case1$ is greater than in $Case2$ due to large propagation delay which is proportional to fanout. Figure 3 shows the corresponding rate of current change. The rate of change of switching current $\frac{dI}{dt}$ is greater in $Case2$ and hence induces greater inductive noise.

Figure 4 shows the corresponding PSN waveforms. We compute the power supply noise from the transient voltage waveforms on the power/ground lines as:

$$V_{PSN}(t) = \{V_{dd_pin} - V_{ss_pin}\} - \{V_{dd_block}(t) - V_{ss_block}(t)\}$$

where V_{dd_pin} (V_{ss_pin}) is the input supply (ground) voltages to the package lead (1.2 V and 0 V respectively in our case), $V_{dd_block}(t)$ and $V_{ss_block}(t)$ are the transient voltage waveforms on the power and ground network, respectively. It is clear that noise induced in $Case2$ is greater than in $Case1$ even though

IEEE COMPUTER SOCIETY

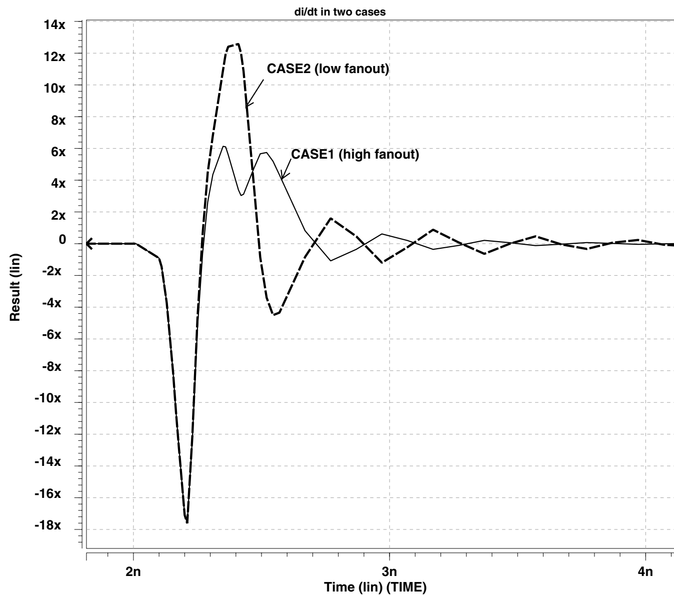


Fig. 3. Rate of change ($\frac{dI}{dt}$) of block current in two different cases.

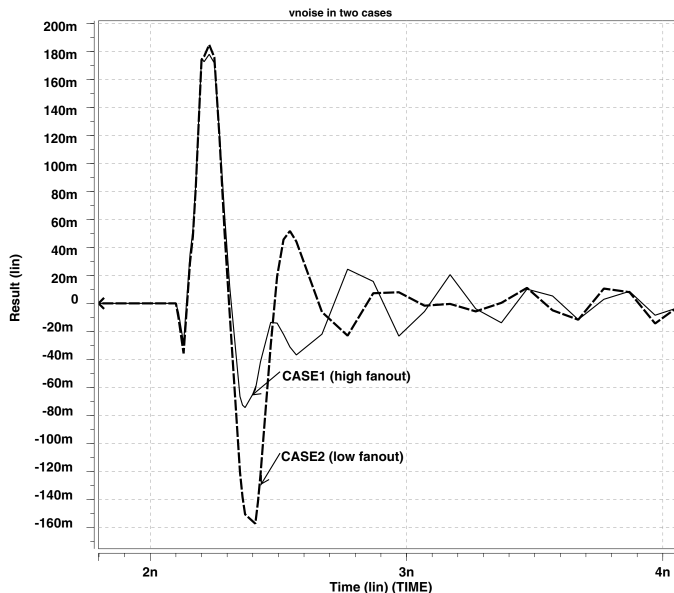


Fig. 4. PSN induced in two different cases.

the peak current occurs in $Case1$. This confirms that maximum switching current does not necessarily generate maximum switching noise.

Based on these analytical and empirical observations, as a main guideline to generate maximum PSN, we give preference to patterns that cause more switching in gates with smaller fanouts. More formally, suppose a circuit G has n gates $g_1, g_2, \cdots, g_n$ with fanout values of $f_1, f_2, \cdots, f_n$ corresponding to those gates. Let $g_i(V_1)$ and $g_i(V_2)$ be the output of gate g_i for two input patterns V_1 and V_2, respectively. $s_i^{V_1 \to V_2} = |g_i(V_1) - g_i(V_2)|$ will be a binary variable indicating if gate g_i has a transition in its output when pattern pair (V_1, V_2) is applied. According to our guideline, to maximize

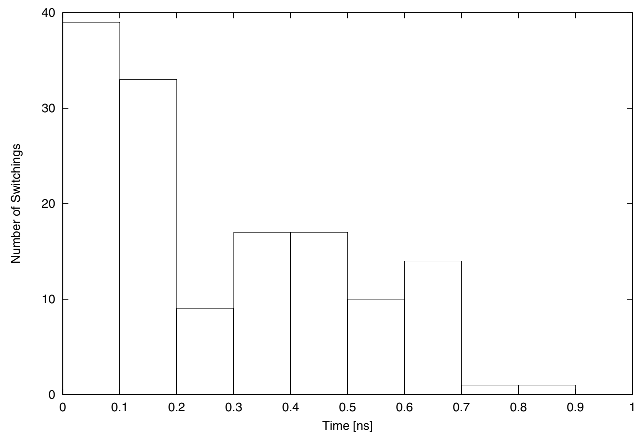


Fig. 5. Switching activity vs time for circuit c432.

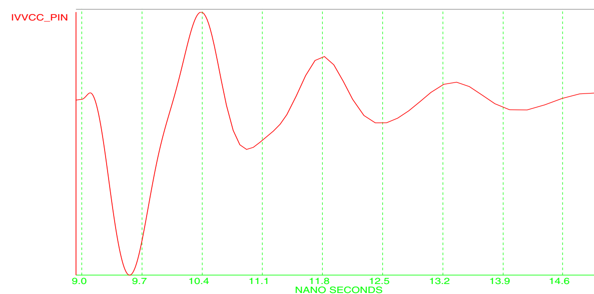


Fig. 6. Current drawn from the V_{dd} supply voltage.

PSN, we must be looking for those (V_1, V_2) pairs that maximize $\sum_{i=1}^{n} s_i^{V_1 \to V_2}$ while simultaneously minimizing $\sum_{i=1}^{n} s_i^{V_1 \to V_2} \cdot f_i$.

III. Pattern Generation

PSN depends on the switching activity inside the circuit and thus is highly input pattern dependent. The spatio-temporal correlation among signals determines the level of switching activity inside a circuit. To stimulate the power supply noise, we need a pattern pair, i.e. (V_1, V_2), to be applied to the circuit under test. In order to stimulate the worst case (maximum PSN), it is necessary to maximize both the rate of change of current $\frac{dI}{dt}$ and also the peak current drawn. As explained in Section II, the inductive noise dominates the resistive noise. More specifically, the rate of current change can be increased by stimulating simultaneous switching in large number of gates with low fanout in a circuit.

A. Timing of Switching Events

The propagation delay of a gate depends on many factors such as fanout load, input rise/fall time and drive strength. Due to difference in propagation delay of the gates, a change in primary input (PI) will trigger a sequence of switching events in the gates that are directly or indirectly connected to it. Since the switching activity inside the circuit determines the switching noise, it is important to find the time intervals where maximum simultaneous switchings occur. To determine the simultaneous switching activity within a clock cycle T, we break down the

IEEE COMPUTER SOCIETY

clock cycle into N small time frames. Each time frame has a duration of $\frac{T}{N}$ and N is chosen based on the required resolution.

We simulated an ISCAS'85 benchmark circuit ($c432$) for a random pattern pair that generated high PSN, using PowerMill tool in Synopsys which is an event driven transistor-level simulator [11]. Figure 5 shows the number of switchings over a period of time when the pattern pair (V_1, V_2) is applied. The horizontal axis plots the time intervals ($\frac{T}{N} = 10ps$) from the time the second vector (V_2) is applied . Maximum simultaneous switching activity occurs at the beginning of the simulation time frame and small peaks occur later in the simulation period. Simulation results for all ISCAS'85 benchmarks confirm that the maximum simultaneous switching activity occurs in the early period of the simulation cyle.

Figure 6 shows the corresponding SPICE simulation current waveform for the pattern applied to circuit c432. It shows that the current drawn from the power supply is maximum at the early stage of the clock cycle and decreases later on. The first peak in the current waveform is due to the initial maximum simultaneous switching activity. Therefore, to generate the steepest maximum current (maximum noise) we need to increase the number of switchings in the low-fanout gates of the circuit during the early period of the cycle.

B. Preprocessing

For each time frame T, a subset of active gates in a circuit G will be chosen and the pattern generation works according to the following three guidelines:

1) *Gate Fanout:* Sort gates in increasing order of their fanouts.
2) *Gate Level:* Within each group, formed in the previous step, sort them according to the *level* that they are positioned in. A level of a gate is the distance of the gate from the primary inputs (PI). When back-traced, a gate close to the PI's has more number of don't-cares (X's) in the input pattern than a gate far away from the PI's. Hence, choosing a node with more number of X's, i.e. a gate in lower level, leaves us with more choices of assigning transitions on the other nodes and increases the chance of generating maximum switching activity in the time frame.
3) *Gate Transition:* Both types of transitions ($0 \to 1$ and $1 \to 0$) are tested in each iteration. Depending on the topology of the circuit, the location of the gate and the way that it affects others, one of these transitions may have a better chance in maximizing PSN.

In the next subsection, we show how our algorithm uses these guidelines and the conventional model of stuck-at-fault (*saf*) and ATPG process to justify a transition at each gate and find vector pairs that maximize PSN.

IV. Algorithm

The pattern generation algorithm is shown in Figure 7. Given a design, in the preprocessing phase the target time frames with the likelihood of having switching activity are obtained. We then use an ATPG algorithm, independent of the simulation method, to find target time frames.

```
01: For each target time frame T
02: {
03:    G ← gates switch in time frame T
04:    Sort gates in G in increasing order of their fanouts
05:    Sort equal-fanout gates in G in increasing order of their levels
06:    PSN_max = 0
07:    V_max = {}
08:    for (i = 1, ···, |G_i|)
09:    {
10:       Perform ATPG using TetraMax to get sa0/sa1 patterns for g_i ∈ G
11:       if (both patterns V_1 and V_2 exists)
12:       {
13:          Try V_1 → V_2; if successful run PowerMill to compute PSN_1
14:          Try V_2 → V_1; if successful run PowerMill to compute PSN_2
15:       }
16:       if (successful and PSN_1 and/or PSN_2 are computed)
17:       {
18:          PSN_max ← MAX{PSN_1, PSN_2, PSN_max}
19:          V_max ← Update input vector pair(s) accordingly
20:       }
21:    }
22: }
23: Return PSN_max and vector pair set V_max creating it.
```

Fig. 7. Deterministic test pattern generation procedure.

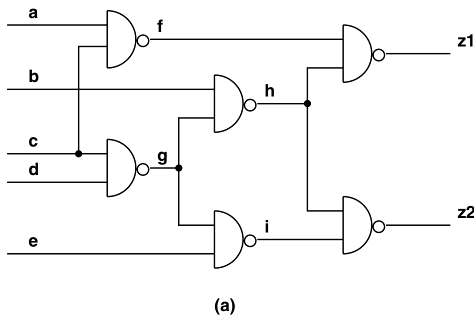
A. Pseudocode

For each target time frames, the corresponding set of gates (G) that switch in this time frame are extracted (line 03). The gates are sorted (lines 04-05) using the criteria explained in the Section III. In the pattern generation process (lines 08-21), transitions are assigned and justified to gates from the sorted list of gates. To justify a transition at a node, we use an ATPG mechanism (i.e. TetraMax [12] in our case) originally used for stuck-at fault testing. The algorithm obtains patterns to justify a value '0' at a node, (viewed as a stuck-at-1 ($sa1$) fault) and a '1' at a node, (viewed as a stuck-at-0 ($sa0$) fault). When both patterns exist (line 11-15), a $(0 \to 1)$ or $(1 \to 0)$ transition can be generated at the output of a selected gate.

The ATPG process generates the pattern pair (V_1, V_2) based on zero-delay model. Now we use a power simulator (i.e. PowerMill [11] in our case) to accurately measure PSN (lines 13-14). PowerMill is a variable delay event-driven simulator and it takes into account the hazards and glitches caused due to difference in the gate propagation delays during the PSN measurement. The result of PSN measurement for this pattern pair is compared to the maximum found so far (PSN_{max}) and the worst case scenario of power supply noise is saved. The vector pair set (V_{max}) are also updated accordingly (line 19). Note that the ATPG based pattern generation is technology independent and does not require any pre-processing of library cells. On the other hand, the power estimator is used to evaluate the patterns based on the library/technology. Instead of finding one pair of patterns, the procedure can be slightly modified to find and report all pattern pairs that create noise in a given range, e.g. $[PSN_{max}, PSN_{max} + \Delta]$.

B. Example

For purpose of illustration consider Figure 8 showing generic test pattern generation process applied to a small example circuit. The stuck-at fault patterns are generated by back-tracing the node towards the primary inputs and are listed in Figure



(a)

Primary Input	f		i		z1		z2		g		h	
	sa0	sa1	sa0	sa1	sa0	sa1	sa0	sa1	sa0	sa1	sa0	sa1
a	0	1	X	X	1	0	X	X	X	X	X	X
b	X	X	X	X	X	0	X	0	X	X	0	1
c	X	1	X	0	1	X	0	X	0	1	X	0
d	X	X	X	X	X	X	X	X	X	1	X	X
e	X	X	0	1	X	X	1	0	X	X	X	X

(b)

Sorted Gates	Transition Selected	Backtracing Result	(sa0, sa1) patterns	(V_1, V_2)	
				(XXXXX,XXXXX)	Initial pair
f	0->1	✓	(0XXXX,1X1XX)	(1X1XX,0XXXX)	
i	1->0	✓	(XXXX0,XX0X1)	(1X1X0,0X0X1)	
z1	0->1 1->0	conflict ✓	(1X1XX,00XXX)	(1X1X0,000X1)	
z2	1->0 0->1	conflict ✓	(XX0X1,X0XX0)	(101X0,000X1)	
g	0->1	✓	(XX0XX,XX11X)	(10110,000X1)	
h	1->0 0->1	conflict conflict	(X0XXX,X10XX)		
				(10110,00001)	Final pair

(c)

Fig. 8. ATPG process for a small example circuit.

8(b). In conventional stuck-at fault test generation, the observation points are the primary outputs. In our method, however, as we are interested in only back-tracing, the node itself is considered to be the observation point.

Initially, the input vector pair (V_1, V_2) is assumed to be all unknown (X) values. The gates are sorted in increasing order of fanout as shown in Figure 8(c). An untried node with the lowest fanout is selected from the sorted list and a transition is assigned to it. For example, in the first iteration, node *f* is selected and a $0 \rightarrow 1$ transition is assigned to it. The $0 \rightarrow 1$ transition assignment can be viewed as a $(sa1, sa0)$ fault pair at the node. Since *f* is the first node in the list and a $0 \rightarrow 1$ transition is selected, therefore V_1 and V_2 patterns are equal to $sa1$ and $sa0$ patterns for node *f*, respectively. Note carefully, there is no conflict for the first chosen node.

A conflict occurs when there is a mismatch in the comparison of the respective stuck-at fault patterns with the input vector pair. If the patterns match then the input pattern pair is updated by replacing the corresponding 'X' values with known justified values in the stuck-at fault patterns. For example, after justifying a $0 \rightarrow 1$ transition at node *f*, the input vector pair becomes $(V_1, V_2) = (1X1XX, 0XXXX)$. The updated input pattern after each gate transition justification is shown in the last column of Figure 8(c). The same procedure is repeated for the next node *i* and a $1 \rightarrow 0$ transition is justified. The input pattern pair becomes $(V_1, V_2) = (1X1X0, 0X0X1)$ after a $1 \rightarrow 0$ transition is justified on gate *i*. When there is a mismatch, then the assigned transition cannot be justified and thus the opposite transition is tried. For node *z1*, when a $0 \rightarrow 1$ transition is tried to be justified, a conflict occurs. In case of a conflict, an opposite transition, i.e. $1 \rightarrow 0$ transition is tried. If both transition assignments fail, the node is skipped and the next node in the list is tried. The process is repeated for all the nodes in the list. After processing the entire list, any leftover X's in the generated pattern input pair (V_1, V_2) will be changed to create transitions because they might still induce more glitches and cause more power supply noise in the circuit. Based on this guideline, 'X' in $V_2 = 000X1$ will be replaced by '0' and the final pattern generated for the example shown in Figure 8(a) is $(V_1, V_2) = (10110, 00001)$.

TABLE I

COMPARING EXHAUSTIVE SIMULATION AND OUR METHOD.

Circuit	*# PI's*	*# Gates*	*Peak Noise [V]*	*CPU Time [sec]*	
				SPICE	*Our Method*
c17	5	6	0.42	181	0.5
cm42	4	18	0.58	354	2.0
cm138	6	15	0.61	2682	3.5

TABLE II

EXPERIMENTAL RESULTS FOR VARIOUS ISCAS85 BENCHMARK CIRCUITS.

Circuit	*# PI's*	*# Gates*	*Peak Noise (near end) [V]*	*Peak Noise (far end) [V]*	*CPU Time [sec]*
c432	36	160	0.76	0.86	179
c499	41	202	0.41	0.52	170
c880	60	357	0.81	0.99	246
c1355	41	514	0.52	0.64	332
c1908	33	880	0.73	0.87	386
c3540	50	1667	0.62	0.75	444
c5315	178	2290	0.82	0.99	568
c6288	32	2416	0.89	1.06	636

V. EXPERIMENTAL RESULTS

Experiments are performed on ISCAS'85 benchmark circuits implemented in $0.25\mu m$ technology. The V_{dd} pin characteristic values used in our simulations are $R_p = R_s = 0.3\Omega$, $L_p = L_s = 8nH$ and $C_p = C_s = 4pF$. These typical values are chosen from the TSMC $0.25\mu m$ library application notes. The effective resistance and capacitance values in the power/ground network are estimated based on the parasitic values per unit length. The resistance and capacitance per unit length used for the power/ground lines are $r = 0.04\Omega/\mu m$ and $c = 10aF/\mu m$, respectively. The primary input's rise time is set to 100ps.

To show the quality of patterns generated by the proposed technique we performed exhaustive simulation for three small benchmark circuits. Our algorithm generated the pattern pairs that cause maximum power supply noise compared to exhaustive pattern simulation results in much shorter CPU time. For

benchmark circuit $c17$, it took 181 *sec* to perform the exhaustive simulation while it takes less than a second for the same vector pair to be generated by our method. Table I shows the results of exhaustive simulation and compares the run times with our method. In all three cases, our method generates the worst case power supply noise test patterns, identical to those found by SPICE, in very short time.

The power supply noise is calculated from the transient voltage waveforms on the power/ground lines as [7]:

$$V_{PSN}(t) = V_{dd_pin} - V_{dd_block}(t) - V_{ss_block}(t)$$

where V_{dd_pin} is the input supply voltage to the package lead (2.5 Volt in our case) $V_{dd_block}(t)$ and $V_{ss_block}(t)$ are the transient voltage waveforms on the power and ground network, respectively. When $V_{noise}(t)$ is positive, the effective supply voltage is less than the nominal supply voltage V_{dd_pin}. Table II shows the peak noise voltages at near end (node closest to the power/ground pins) and far end (node farthest from power/ground pins). As expected (see Section II), the far end noise is more severe due to larger effective resistive parasitics experienced by the blocks close to the far end. The main advantage of our method is its short runtime. For example, SPICE takes 12 minutes to simulate one input vector pair for circuit *c432*, while it takes 179 sec to generate and simulate 500 patterns for maximum power supply noise by our method.

VI. Conclusion

An automatic pattern generation mechanism to stimulate the maximum power supply noise has been presented in this paper. The basic strategy is to maximize the switching activities of those gates in the first few levels of the circuit that have lower fanouts. Our methodology uses conventional ATPG and power simulators to evaluate a gate-level circuit and finds patterns that cause maximum switching activity and thus maximum instantaneous current. We have verified the quality of these patterns using SPICE simulation. In all cases, our method finds the same (or comparable) patterns while its running time is 2 to 3 order of magnitude faster than that of SPICE.

Acknowledgements

This work was supported in part by the National Science Foundation CAREER Award #CCR-0130513.

References

[1] R. Senthinatharr and J. L. Prince, *Simultaneous Switching Noise of CMOS Devices and Systems*, Kluwer Academic Publishers, 1994.

[2] Y. Chang, S. Gupta and M. Breuer, "Analysis of Ground Bounce in Deep Sub-Micron Circuits," in Proc. *VLSI Test Symp. (VTS'97)*, pp. 110-116, 1997.

[3] H. Chen and D. Ling, "Power Supply Noise Analysis Methodology for Deep-Submicron VLSI Design," in Proc. *Design Automation Conf. (DAC'97)*, pp. 638-643, 1997.

[4] L. Zheng, B. Li, and H. Tenhunen, "Efficient and Accurate Modeling of Power Supply Noise on Distributed On-Chip Power Networks," in Proc. *Int. Symposium on Circuits and Systems (ISCAS'00)*, pp. 513-516, 2000.

[5] E. Liau and D. Landsiedel, "Automatic Worst Case Pattern Generation Using Neural Networks & Genetic Algorithm for Estimation of Switching Noise on Power Supply Lines in CMOS Circuits," in Proc. *European Test Workshop (ETW'03)*, pp. 105 -110, 2003.

[6] Y. Jiang, K. Cheng and A. Deng, "Estimation of Maximum Power Supply Noise for Deep Sub-Micron Designs," in Proc. *Int. Symp. on Low Power Electronics and Design (ISLPED'98))*, pp. 233-238, 1998.

[7] S. Zhao, K. Roy and C. Koh, "Estimation of Inductive and Resistive Switching Noise on Power Supply Network in Deep Sub-micron CMOS Circuits," in Proc. *Int. Conf. on Computer Design (ICCD'00)*, pp. 65-72, 2000.

[8] S. Zhao and K. Roy, "Estimation of Switching Noise on Power Supply Lines in Deep Sub-micron CMOS Circuits," in Proc. *Thirteenth Int. Conf. on VLSI Design,*, pp. 168-173, 2000.

[9] Y. Jiang, K. Cheng and A. Krstic, "Estimation of Maximum Power and Instantaneous Current Using a Genetic Algorithm," in Proc. *Custom Integrated Circuits Conf. (CICC'97)*, pp. 135-138, 1997.

[10] G. Bai, S. Bobba and I. Haji, "Maximum Power Supply Noise Estimation in VLSI Circuits Using Multimodal Genetic Algorithms," in Proc. *Int. Conf. on Electronics, Circuits and Systems (ICECS'01)*, vol. 3, pp.1437 -1440, 2001.

[11] *Synopsys Inc., Power Mill Reference Manual*, 2003.

[12] *Synopsys Inc., TetraMAX Reference Manual*, 2003.

IEEE COMPUTER SOCIETY