Vector Generation for Power Supply Noise Estimation and Verification of Deep Submicron Designs

Yi-Min Jiang and Kwang-Ting (Tim) Cheng, *Fellow, IEEE*

***Abstract*—This paper presents new techniques for generating a small set of patterns for power network simulation to estimate the maximum power supply noise of the chip, as well as to identify cells/blocks for which the power supply noise at their $V_{\rm dd}$ ports exceeds a specified threshold. We first present an efficient, cell-level simulator for estimating power supply noise of any given vectors. Based on this simulator, we then apply the genetic algorithm (GA) to derive a small set of patterns producing high power supply noise. To identify critical nodes with power supply noise exceeding a threshold, the multiobjective GA is adapted for pattern generation. To achieve high coverage of such critical nodes, we model the search criteria as the maximum weighted matching of a bipartite graph, and guide the search direction according to the matching results. The derived patterns will be simulated on a power network simulator to obtain a lower bound of the maximum power supply noise and to identify the critical nodes. Experimental results on public benchmark circuits, as well as some industrial designs, are presented to demonstrate the efficiency and effectiveness of the proposed approaches.**

***Index Terms*—Genetic algorithm (GA), lower bound, pattern generation, power network simulation, power supply noise.**

I. Introduction

POWER supply noise due to switching current is becoming an important factor for deep submicron designs. This noise reduces the actual voltage level reaching a device, and thus leads to an increase in signal delay that may result in performance degradation [6], [13]. Moreover, excessive noise may cause logic and/or timing errors [1], [2]. With the increasing demand for high frequency and short rise/fall signal transition time in today's designs, more devices are switched simultaneously or within a small time interval, which causes large instantaneous currents and considerable change of currents. Therefore, the power supply noise has been increasing dramatically.

Power supply noise includes the inductive ΔI noise and IR voltage drop caused by the switching in internal circuitry as well as input and output buffers. The inductive ΔI noise is induced due to the change of instantaneous current on either package lead inductance or wire/substrate inductance (this noise is proportional to $L * di/dt$), while the IR voltage drop is caused by the instantaneous current through the resistive power and ground lines. For CMOS circuits, the switching currents are mainly due to the signal switching, which depends on the input patterns applied to the circuits. To be able to observe switching at the signals, a two-vector sequence, $V = (v_1, v_2)$, has to be applied at the inputs of the combinational portion of the circuit. One way to find the maximum power supply noise and verify all critical noise nodes would be to simulate all possible patterns at these inputs including primary inputs and pseudo primary inputs (i.e., outputs of flip-flops). For a circuit with n inputs, this would require simulation of 4^n patterns. This is practical only for circuits with a very small number of inputs.

This paper focuses on the estimation of the maximum power supply noise and the verification of the power nodes of cells with power supply noise exceeding a threshold. To estimate the maximum power supply noise, we try to simulate a large number of patterns and use the genetic algorithm (GA) [10] to select/derive a small set of patterns that would cause high power supply noise. Since all power/ground segments' RC's need to be considered in simulation to derive accurate power/ground segments' currents and voltages, circuit-level simulation will be unacceptably slow for this application due to the large number of simulation runs required. We therefore first derive comprehensive current/voltage waveform libraries for each cell (which can be repeatedly used by all designs based on these libraries). We then perform the simulation at the cell level. We use an efficient event-driven waveform/logic simulator extended from [7] for current waveform simulation. Based on the waveform simulation results and the current/voltage waveform library, the current flowing through each cell with respect to a given pattern can be efficiently estimated. Note that for a segment in a power/ground net tree, the current waveform is not a direct superposition of the current waveforms of the cells downstream of the segment. In Section III-B, we will discuss the related issues and discuss how to derive an estimated current waveform of a power/ground net segment based on the current waveforms of the cells. Based on this simulation framework, we use GA to derive a small set of patterns. Finally, we can use a lower-level simulator to validate these patterns and identify the worst one among the selected set. This framework can be used to identify the patterns that would cause high power supply noise at any specified block in the chip, or the IR voltage drop at any given power supply segment. The difference for different blocks or different segments will be in the fitness functions used in the GA.

To verify the power network reliability, we attempt to generate a set of patterns that would cause high power supply noise at the $V_{\rm dd}$ ports of all the cells whose worst-case drop could exceed a given threshold. This problem is referred to as the critical power supply noise problem. We propose a heuristic procedure for this problem and attempt to maximize the coverage of these

Manuscript received February 11, 2000; revised November 21, 2000. This work was supported by National Science Foundation, California State MICRO program, Hewlett-Packard, and Synopsys.

Y.-M. Jiang is with the Synopsys, Mountain View, CA 94043 USA (e-mail: jym@synopsys.com).

K.-T. (Tim) Cheng is with the University of California, Santa Barbara, CA 93106 USA (e-mail: timcheng@bryce.ece.ucsb.edu).

Publisher Item Identifier S 1063-8210(01)03298-X.

cells. Our approach extends the technique of [14] to generate input patterns for identifying all potential problematic blocks. We propose to induce the maximum current drawn from *each* individual block using GA. Without losing the functional correlation between different blocks, we transform the single-objective GA to satisfy correlated multiobjectives simultaneously, where each objective denotes the maximum current associated with a specific block. To achieve this, we model the search criteria in GA as the maximum weighted matching of a bipartite graph, which can be efficiently solved by the Hungarian method [17].

Our experimental results show that, for the maximum power supply noise, our approach produces, on the average, 23% and 17% tighter lower bounds for the benchmark set, than the bounds obtained by the weighted random approach (which uses random patterns with very high primary input transition probabilities) and the GA approach directly based on a transition-level simulator, respectively. Also, the estimation time of our method is significantly faster. We have also implemented the pattern generator, named *VIP*, for identifying critical nodes. The obtained patterns, which will cause high power supply noises for most potential problematic blocks, can be used by any power network simulator to analyze the critical power supply noise. *VIP* has been tested on a set of benchmarks with completed physical designs, which is comprised of twelve large industrial designs with a wide variety of applications such as microprocessors, DSP processors, and large memory banks. Experimental results show that the patterns generated by *VIP* identifies more critical cells and causes higher power supply noise than those generated by other approaches.

The rest of this paper is organized as follows. Section II reviews the previous work in the area of power supply noise and the maximum instantaneous current estimation. The section also gives a brief introduction to the GA [10]. Section III introduces the technique for maximum power supply noise estimation. Section IV describes the vector generation technique for identification of critical nodes with excessive power supply noise. Section V gives the experimental results. Section VI concludes the paper.

II. Preliminaries

A. Related Works

Several approaches have been proposed recently for power supply noise [2], [3], [18] and maximum instantaneous current [15], [14] estimation. Senthinathan and Prince [18] proposed a detailed electrical model of a typical chip-package interface. In this approach, several closed-form equations considering the negative feedback influence are derived to calculate simultaneous switching noise. Chang *et al.* [2] proposed a scaling model to estimate the ground bounce caused by the switching in internal circuitry for deep submicron circuits. Several experiments were conducted to explore the properties of ground bounce. Chen and Ling [3] proposed a switching circuit model to estimate the power supply noise including the IR voltage drop and inductive ΔI noise based on an integrated package-level and chip-level power bus model. They also estimate the amount of on-chip decoupling capacitance needed to reduce the power supply noise to an acceptable level.

For the maximum instantaneous current estimation, Kriplani *et al.* [15] presented a pattern-independent algorithm, named **iMax**, to find an upper bound on the maximum instantaneous current. Jiang *et al.* [14] proposed a timed-ATPG algorithm and a probability-based algorithm to generate patterns causing high instantaneous current. In the timed-ATPG approach, the **iMax** algorithm is first applied. Based on the **iMax** results, a set of internal signals whose simultaneous switching would likely produce the highest current are identified. These signals are then assigned transitions and the corresponding times that the transitions need to take place. ATPG techniques enhanced to handle timing are then used to derive the corresponding test patterns that would cause these transitions at the specified times. In the probability-based approach, the first phase is similar to the timed-ATPG approach. A set of signals are identified and assigned transitions and the corresponding times that the transitions need to take place. These assignments are then converted into signal transition probabilities based on the driving gates' possible contribution in total current at a given time instance. Then, these signal transition probabilities are propagated backward to the primary inputs, and the probabilities derived at the primary inputs are used as weights for generating random vectors. Jiang *et al.* [14] proposed a genetic-algorithm-based approach for finding patterns causing high instantaneous current. It applies the GA to iteratively generating new patterns that potentially could cause higher instantaneous current than those generated in the previous iterations. The new patterns are generated using genetic operations, based on "good" patterns derived in the previous iterations.

B. Genetic Algorithm (GA)

GA [10] is a robust algorithm which has been applied to solve many search problems. The key idea is that, if solutions are represented by strings, the string associated with the optimal solution can eventually be found through "evolution-like" string operations. The search engine is an iterative process which employs three operations: selection, crossover, and mutation. The objective of these operations is to remove "poor" strings and produce new strings which is comprised of parts of "healthy" strings.

To use GA, the elements in the solution space need to be coded into finite length strings. Each string has an associated fitness which depends on the application. An initial population needs to be specified as the input of GA. The initial population contains N random strings of length L, where N and L are parameters used in GA. Generation of a new population is found by: 1) evaluating the fitness for each string; 2) selecting two individuals from the current population; 3) crossing the two selected strings to generate two child-strings from two parent-strings; and 4) mutating the elements of the new strings with a given mutation probability. The selection process is biased toward individuals with higher fitness values. The next population is generated based on the current population using the same procedure. During the string generation process, the strings with the highest fitness would be recorded.

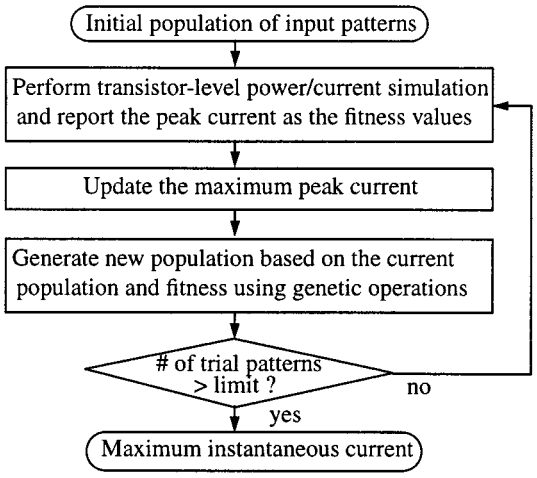

Fig. 1. GA-based approach for maximum instantaneous current.

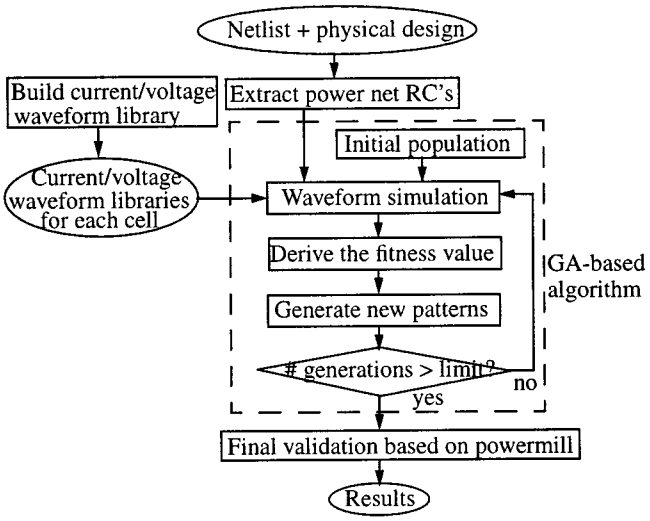

Fig. 2. The flow of our technique for maximum power supply noise estimation.

Under this scenario, the technique in [14] transforms the solution space of two-vector sequences causing maximum instantaneous current into the GA search domain, and drives the search engine to find the solution. In the transformation, each input sequence is coded into a string, and the associated peak current corresponds to the fitness of this string. According to this, GA starts with a population of strings and iteratively generates successive population with likely higher fitness. The procedure is shown in Fig. 1. In this approach, the initial set can be either generated randomly or specified by users. To ensure high accuracy, a transistor-level power/current simulator (e.g., PowerMill [20]) is used to simulate each sequence and report the peak current as the fitness. The maximum instantaneous current is updated based on the fitness for each iteration. The selection and crossover schema used are tournament selection without replacement [16] and one-point crossover [10], respectively. In the one-point crossover schema, a bit position is randomly selected between 2 and $(L-1)$, where L is the number of primary inputs of the circuit, and the two parent-strings are crossed at that point. Thus, the first child is identical to the second parent after the crossing point. For the mutation probability, we use $1/L$. The process continues until no further improvement is achieved or the number of iterations reaches a prespecified limit.

III. Estimation of Maximum Power Supply Noise

We apply the GA-based technique for generating vectors for the purpose of estimating the maximum power supply noise. The overall process of the method is shown in Fig. 2. We assume that the netlist and physical design are given. In the iterative GA process, we basically need an efficient simulator to estimate each new trial vector's fitness (for causing high power supply noise including both IR drop and inductive noise). It is clear that directly using a transistor-level power network simulator to derive the fitness value is too computationally expensive. Therefore, we first develop an efficient cell-level waveform simulator for such fitness value estimation. To estimate power supply noise, this simulator needs a pre-characterized current/voltage waveform library for each cell. The detailed procedure of building such a library will be discussed in Section III-A. The effective power/ground net RC's for each block also need to be extracted first and used as input for the vector generation process. Starting from the initial pattern population, the cell-level waveform simulator simulates each pattern to estimate its corresponding power supply noise. The fitness value of a pattern is simply the highest power supply noise at the target areas, which are the special P/G nodes of interests. The details of the waveform simulation and the computation of the power supply noise based on the simulation results are described in Section III-B. New patterns are then generated by using the GA operations including selection, crossover and mutation. The final population at the end of the iterative GA process, which is a small set of patterns, can then be simulated again using a transistor-level power network simulator to derive a more accurate estimate of power supply noise.

A. Characterization: Building Current/Voltage Waveform Library

1) Circuit Model for Power Supply Noise: We consider the inductive ΔI noise and IR voltage drop caused by the switching in internal circuitry and I/O buffers. In the following discussion, we assume that the topologies of the power and ground nets are single-pad trees. The models can be easily extended to handle multipad tree and general graph topologies but the details of such extension will not be elaborated. For inductive ΔI noise we consider only the part caused by the change of instantaneous current on the package lead inductance and ignore the one from the wire/substrate inductance which is considerably smaller. Fig. 3 shows a circuit model for each cell. The model is used to derive the current waveform flowing through each cell. We use $V_{\rm dd}$ and $V_{\rm ss}$ to denote power and ground, respectively. Each $V_{\rm dd}$ and $V_{\rm ss}$ pin is modeled by an RLC network ($L_{\rm pd}$, $R_{\rm pd}$ and $C_{\rm pd}$ for $V_{\rm dd}$ pin, and $L_{\rm ps}$, $R_{\rm ps}$ and $C_{\rm ps}$ for $V_{\rm ss}$ pin) as shown in Fig. 3. Symbol C_s is used to denote the substrate and on-chip decoupling capacitance. Symbol $R_{\rm nd}$ and $C_{\rm nd}$ ($R_{\rm ns}$ and $C_{\rm ns}$) correspond to the effective resistance and capacitance of $V_{\rm dd}$ ($V_{\rm ss}$) line from the $V_{\rm dd}$ pin to the $V_{\rm dd}$ node of the cell (the $V_{\rm ss}$ node of the cell to the $V_{\rm ss}$ pin), respectively. Note that different cells in a circuit have different $R_{\rm nd}$, $C_{\rm nd}$, $R_{\rm ns}$ and $C_{\rm ns}$. When the

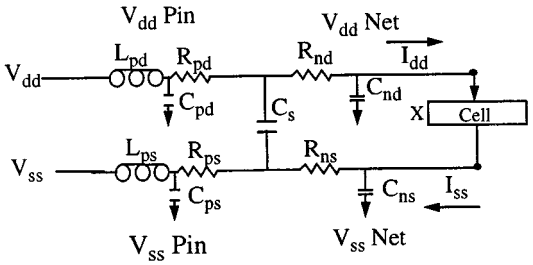

Fig. 3. Circuit model for power supply noise caused by the switching in internal circuitry and I/O buffers/pins.

TABLE I
THE RANGES AND INTERVALS FOR INPUT INDEXES IN LIBRARY

	input voltage waveform			effective power net resistance	effective power net capacitance	loading capacitance
	starting voltage	ending voltage	rising (falling) time			
range	0 V - 3.3 V	0 V - 3.3 V	0.1 ns - 1.0 ns	1 Ω - 151 Ω	50 fF - 650 fF	100 fF - 1600 fF
interval	0.6 V	0.6 V	0.3 ns	30 Ω	200 fF	500 fF

inputs to the cell change from vector X_1 to vector X_2 a charge (discharge) current will be drawn from (flow into) the $V_{\rm dd}$ ($V_{\rm ss}$) pin, which contributes to the $L * di/dt$ voltage swing. Also, the charge (discharge) current flowing through the effective resistance $R_{\rm nd}$ ($R_{\rm ns}$) and power pin resistance $R_{\rm pd}$ ($R_{\rm ps}$) causes the IR voltage drop from the external $V_{\rm dd}$ to the $V_{\rm dd}$ node of the cell (the $V_{\rm ss}$ node of the cell to the external $V_{\rm ss}$). The $I_{\rm dd}$ and $I_{\rm ss}$ correspond to the charge and discharge currents for the cell, respectively.

2) Building the Current/Voltage Waveform Library: Power supply noise at the $V_{\rm dd}$ and $V_{\rm ss}$ nodes of a cell can be computed by summing up the inductive ΔI noise and IR voltage drop along the series of power lines segments from the $V_{\rm dd}$ pin to the $V_{\rm dd}$ node of the cell ($V_{\rm ss}$ node of the cell to $V_{\rm ss}$ pin). Therefore, we need to derive accurate current waveforms for all segments of $V_{\rm dd}$ and $V_{\rm ss}$ lines and pins, which depend on the charge and discharge current waveforms of all cells. Based on the circuit model, for a given circuit with the netlist and its physical design, we first estimate the current waveform for each cell with respect to a given input pattern. The current waveforms for all the cells are then used to compute the current waveforms flowing through the $V_{\rm dd}$ and $V_{\rm ss}$ pins as well as all segments of $V_{\rm dd}$ and $V_{\rm ss}$ nets. The power supply noise on the $V_{\rm dd}$ and $V_{\rm ss}$ nodes of each cell can then be estimated based on these current waveforms.

The charge/discharge current and output voltage waveforms for a cell depend on various characteristics including the type of the cell, the starting/ending voltage and rising (falling) time of the input voltage waveform, loading capacitance of the cell, $V_{\rm dd}$ ($V_{\rm ss}$) pin RLC values, and effective power/ground net RC's (see Fig. 3). To characterize the current and output voltage waveforms, we build the current/voltage waveform libraries with indices including the above characteristics by using the circuit level simulator HSPICE. The ranges and intervals for all indices used in our libraries for a sample cell library are shown in Table I. The two indices are used for all the characteristics (input voltage waveform, effective power net resistance and capacitance, and loading capacitance) to characterize the library. For example, the range and interval of the rising time in Table I are 0.1–1.0 ns, and 0.3 ns, respectively. That means the rising time of the input voltage waveform used for library characterization are 0.1 ns, 0.4 ns, 0.7 ns, and 1.0 ns. For a given package specification, the pin RLC values are fixed. To reduce the sizes of our libraries, we assume that only one input of a cell changes the value from low to high (or high to low), and the values of the other inputs are kept in their stable values such that the output of the cell switches and thus draws current. The input voltage waveform in our library is a ramp, which is characterized by three characteristics: starting voltage, ending voltage, and the slope (rising time or falling time). The input voltage waveforms with other types can be piecewise linearly approximated. We sample the HSPICE results for output voltage and charge/discharge current waveforms with a fixed time step (say, 20 ps) and store the discrete values of the waveforms in the library. These current/voltage waveform libraries are used to estimate the current waveform of each cell for a given input pattern applied at the primary input. To reduce the sizes of the waveform libraries, we build the comprehensive libraries only for cells with up to 4 inputs. For complex gates with more than four fanins, we apply various heuristics to reduce the number of entries. For example, we group all possible input patterns of the gate into several sets in such a way that each pattern in a set exhibits similar current. Then, we perform HSPICE simulation for these sets to build the current/voltage waveform libraries. Note that all libraries are built only once, and can be repeatedly used for power supply noise estimation for all designs based on the same cell library.

Note that we ignore the $V_{\rm dd}/V_{\rm ss}$ voltage drop impact caused by the switching of other cells during the library characterization. However, the library is used only to derive the fitness value of an input pattern. Even though we realize that this effect might affect the accuracy of the waveforms in this phase, it would be too expensive to consider it. However, our experimental results show that the used fitness function can still successfully guide the GA to derive good results. All the results are validated by transistor-level simulation.

B. Waveform Simulation

1) Deriving Current Waveforms Flowing Through Blocks: Given a netlist, its physical design, the topologies of power and ground nets, and the $V_{\rm dd}$ and $V_{\rm ss}$ pin characteristics, we first group the cells which are physically close to each other into small blocks and compute the effective power/ground RC's for each block by using an RC reduction tool. Then, we apply a waveform simulator extended from [7], which is based on the event-driven logic simulation algorithm, to simulate a given input pattern. The waveform simulator can handle the input voltage waveforms of the cells containing glitches as well as partial voltage swing, and produce the output voltage waveforms for all cells. The resulting voltage waveforms at all internal nodes along with the current/voltage waveform libraries discussed in Section III–B2 are then used to estimate the charge/discharge current for each internal cell and thus each block. For the current/voltage values of the time

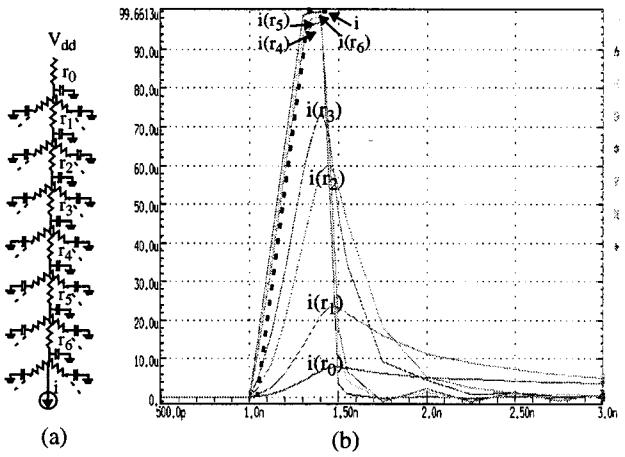

Fig. 4. The current distribution for the charge current of a block with a small time period.

instances within one time step (say, 20 ps), we approximate their values using interpolation on the values in current/voltage waveform libraries.

In the following, we propose a technique to efficiently obtain the currents flowing through the $V_{\rm dd}$ and $V_{\rm ss}$ pins as well as each segment of power and ground lines based on the obtained current waveforms of all blocks.

2) Deriving Current Waveforms Flowing Through Power/Ground Net Segments: The current waveforms flowing through power and ground net segments depend on the charge/discharge currents of all blocks. However, even for a power supply net with a one-pad tree topology, the current waveform of a segment is not a direct superposition of the current waveforms of the blocks downstream of the segment.

For CMOS circuits, when the output value of a cell is changed from low to high (high to low), the charge (discharge) current passes through the cell. If the duration of the current pulse is much smaller than the RC time constant from $V_{\rm dd}$ ($V_{\rm ss}$) pin to the $V_{\rm dd}$ ($V_{\rm ss}$) node of the block, not all charge (discharge) current is instantly coming from (to) $V_{\rm dd}$ ($V_{\rm ss}$) pin. Part of the current is coming from (to) the neighboring capacitances along power and ground lines, and these capacitances will be charged up slightly later by the current from its neighboring capacitances again and eventually by the external $V_{\rm dd}$ source. Therefore, the current waveforms in different segments along the path from $V_{\rm dd}$ pin to $V_{\rm dd}$ node ($V_{\rm ss}$ node to $V_{\rm ss}$ pin) of the block are different. We conducted a simple experiment to illustrate this point. Consider the $V_{\rm dd}$ net of a circuit with a tree topology shown in Fig. 4(a), where each node except the terminal nodes has three branches, and the total number of terminal nodes is 5000. Suppose each terminal node corresponds to the $V_{\rm dd}$ node of a cell. The R and C values in Fig. 4(a) are extracted from the power supply net of a corresponding physical design. We assume that just one block has switching current and all cells in other blocks are in stable values and do not draw any current. In this figure the charge current flowing through the switching block is modeled as a current source i at one of the terminal nodes. Fig. 4(b) shows the waveforms for current i and the current in all segments $(r_0$–$r_6)$ along the path, which are derived by HSPICE. The vertical axis is the current (in μA) of each segment in the path, and the horizontal axis is the time (in ns). Note that the peak current through the root segment r_0 is much smaller than that the current source i, and the current waveforms in different segments are different. However, the segments closer to the switching block have waveforms similar to that of the current source.

TABLE II
THE RANGES AND INTERVALS FOR THE INDEXES IN POWER SEGMENT WAVEFORM LIBRARY

	triangular current waveform		
	rising time	falling time	peak value
range	0.1 ns - 0.7 ns	0.1 ns - 0.7 ns	10 μA- 510 μA
interval	0.2 ns	0.2 ns	100 μA

The current waveform in each segment along the power net $(r_0$–$r_6)$ is a function of the current waveforms flowing through the leaf cells and the RC's of all segments in the power net tree. It will be unacceptably slow to explicitly perform circuit-level simulation on the circuit consisting of the power lines RC tree and current source of the block [like Fig. 4(a)] for each derived block current waveform and for each pattern. We therefore develop a "library" to speed up this process. We attempt to build a power segment waveform library for each block using the block current waveforms as parameter. In this phase, we first reduce the RC's in the power net tree using a RC reduction tool (e.g., [22]), for each block except the ones in the target path (In other words, we construct the effective power net tree like Fig. 4(a) for each block). We then assume a triangular current source, which is characterized by rising time, falling time and the peak value as parameters to approximate the charge/discharge current waveform of the block. Sample ranges and intervals for these indices used in this library constructing phase are shown in Table II. Then, we perform HSPICE simulation for each instance of the parameterized current sources and derive the current waveforms flowing through all segments in the path. We then sample these obtained current waveforms of all segments with a fixed time step (e.g., say, 20 ps), and store the sampled discrete waveforms in a power segment waveform library. The libraries for ground net are built in the same way. Note that the power segment waveform library are built only once for a given design, and can be repeatedly used to estimate the power supply noise for simulation of a large number of input patterns.

For given input patterns, the estimation process for power supply noise is summarized as follows. First, we build the power segment waveform libraries, compute the effective power and ground net RC's for each small block (which consists of a set of adjacent cells), and apply the waveform simulator to simulate the input pattern to obtain the charge and discharge current waveforms for all blocks. Then, for each block, based on the block current waveform and the power segment waveform libraries, we derive the *effective* current waveforms flowing through all power net segments along the target path (charging/discharging path from $V_{\rm dd}$ pins to the $V_{\rm dd}$ nodes of the blocks), and the effective current waveform flowing through power pins. For the current values of the time instances within one fixed-time step (say, 20 ps), we approximate their values using interpolation on the values in the power segment waveform library. The overall current waveform flowing through each power net segment can then

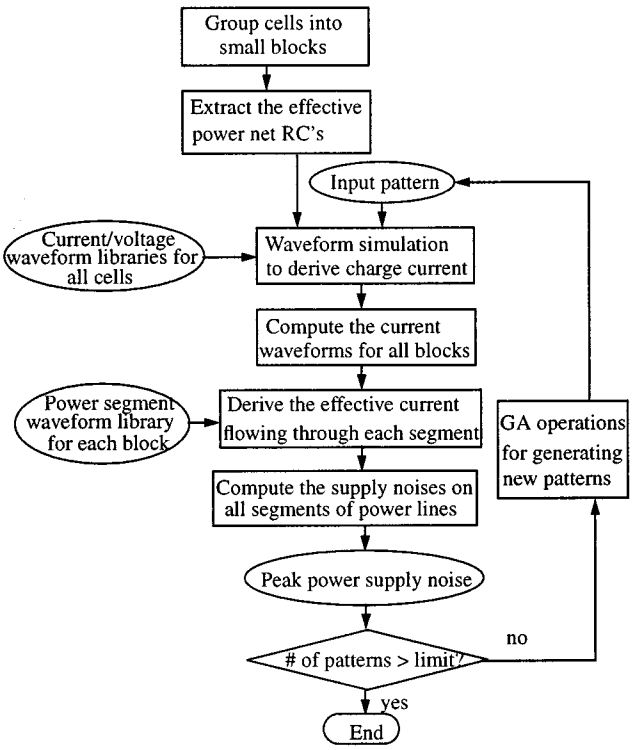

Fig. 5. The overall GA-based process to estimate the power supply noise.

be computed by summing up all the corresponding *effective* current waveforms for all blocks downstream of the target segment. The IR voltage drop for each block can be computed by summing up the IR voltage drops of all power net segments along the path. The inductive ΔI noise on $V_{\rm dd}$ ($V_{\rm ss}$) pin can be obtained by evaluating the equation $L * di/dt$ where i is the total current waveform passing through the $V_{\rm dd}$ ($V_{\rm ss}$) pin.

Based on this efficient framework for deriving the power supply noise for any given 2-vector sequence, we apply the GA to generate a small set of patterns that would cause high power supply noise at a specified area. We use the tool described in [14] to generate such patterns. The vector generation process is based on the GA [10] and is an iterative process. The iterative process stops after the number of simulated patterns reaches a limit and the tool reports a small number of patterns causing the highest power supply noise at the specified block(s). The overall process is shown in Fig. 5.

Note that the RC reduction tool [22] is used in two ways. First, in the power segment waveform library characterization phase, for each block (which consists of a set of adjacent cells), the entire RC network is reduced using this tool to a path from $V_{\rm dd}$ pin to the $V_{\rm dd}$ node of the block. Second, in the phase when the charging/discharging current waveforms computed for a block, for each block, the reduction tool collapses the RC network into a single equivalent lumped RC pair. The resultant RC pair are referred to as the effective power and ground net RC for each block.

IV. Identification of All Nodes with Critical Power Supply Noise

We further extend the method to generate vectors used for simulation to identify as many nodes as possible with power

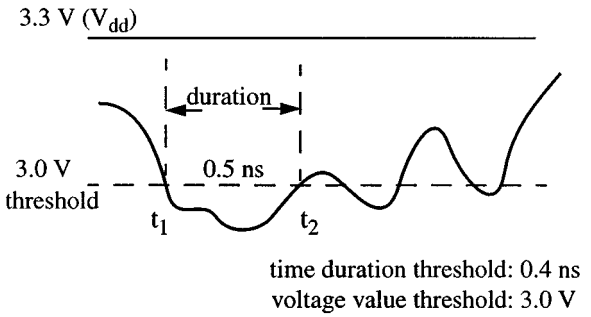

Fig. 6. An exemplar voltage waveform for a cell's $V_{\rm dd}$ port w.r.t. an input pattern.

supply noise exceeding a threshold. In this section, we first define the problem and then describe the vector generation method for solving this problem. In the following, the term "an input pattern" is referred to as a two-vector sequence $V = (v_1, v_2)$, where the first vector v_1 is used to initialize the circuit, and v_2 causes switching at internal nodes.

Definition 1: A cell is called a *critical power supply noise cell* if there exists at least one input pattern that causes power supply noise at the cell's $V_{\rm dd}$ or $V_{\rm ss}$ ports to exceed a specified threshold and the duration is longer than a specified period.

Fig. 6 shows an exemplar voltage waveform of the $V_{\rm dd}$ port of a cell after applying an input pattern. During the period from t_1 to t_2, the voltage level is lower than the threshold and this time period is longer than the specified one. This cell is thus a critical power supply noise cell. Note that there may exist multiple input patterns which could identify a critical power supply noise cell. On the other hand, an input pattern may identify multiple critical power supply noise cells.

Definition 2: The *critical power supply noise set* of an input pattern is the set of the critical voltage cells identified by this pattern.

There is one corresponding critical power supply noise set for each input pattern. The union of the critical power supply noise sets of all input patterns covers all critical power supply noise cells in a design. We propose to find a small set of input patterns such that the union of their critical power supply noise sets would be identical to the union of the sets of all input patterns. We apply the GA-based technique proposed in [14] (introduced in Section II-B) to generate such input patterns. The key issue here is the selection of a suitable fitness function for GA as the quality of the input patterns generated by GA is strongly dependent on the fitness function used. The power supply noise computation requires accurate simulation of the power supply network together with the transistor netlist that drives them. If we use power supply noise as fitness, this comprehensive simulation needs to be performed once for each pattern. In such a way, the overall GA-based procedure will be prohibitively slow for most of today's large designs with millions of transistors and power net RC's. Therefore, we need to find an easy-to-compute metric as the fitness in which high fitness corresponds to the high coverage of critical power supply noise cells.

Peak current of a design is the maximum current which the design draws in response to an input pattern. This current distributes through the power network to transistors and capacitances. Higher peak current tends to cause higher power supply

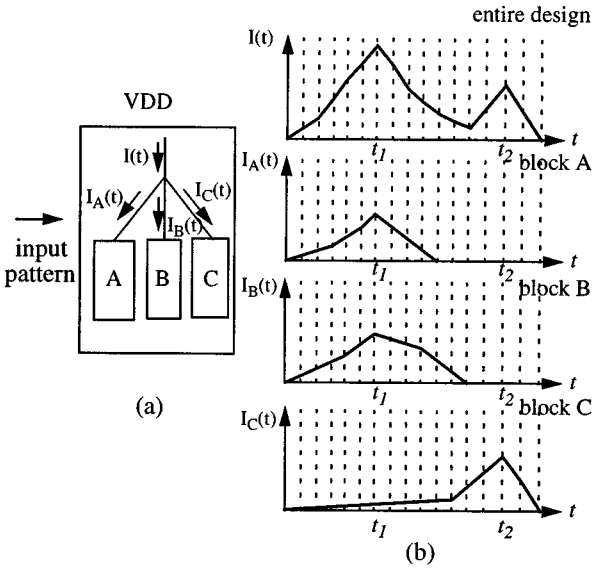

Fig. 7. The exemplar current waveforms of the entire design and each individual block.

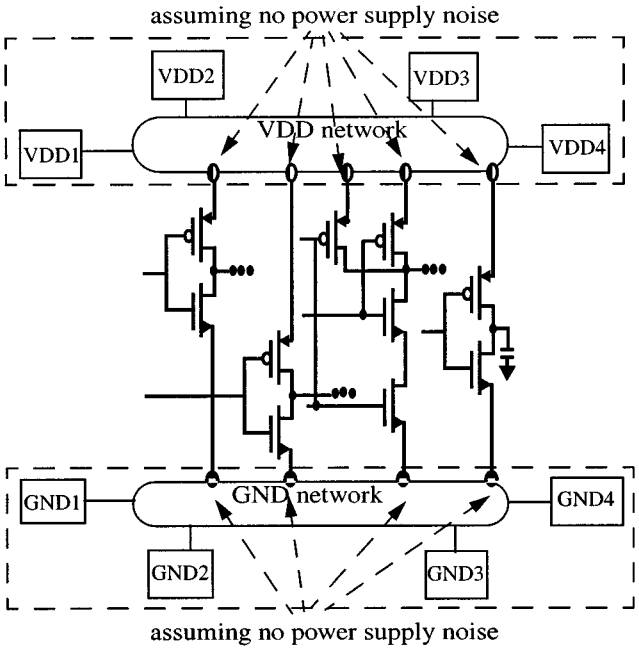

Fig. 8. The model for transistor-level simulation speed-up approach.

noise because more current flows in the resistive network. Similarly, we can define the peak current of a functional block as the maximum current the block draws in response to an input pattern. If we use the peak current of an entire design as fitness, the generated patterns may not activate all the critical power supply noise cells as the times at which the peak current of different functional blocks occurs may not coincide with each other. Maximizing peak current of the whole design usually results in exhibiting the worse cases only for cells in some critical blocks and may miss some critical cells. Consider a 3-block design in Fig. 7(a). The current waveform with respect to an input pattern for the entire design and each block is shown in Fig. 7(b). The peak current of the entire design occurs at time t_1 at which the current drawn by block C is much lower than its peak current. For this case, even though the voltage drop at block C exceeds the threshold at time t_2, the vectors generated by the GA are unlikely to activate it if we use the entire design' peak current as the fitness.

Motivated by this, we propose to use the peak current of each individual block as a major factor of fitness. In the meantime, we also include the current of the entire design into fitness. This is because current from other blocks also contributes to the power supply noise of the target block. We use an approximate but efficient approach proposed in [5] and [25] to transistor-level simulation for extracting the peak current. Also, during the transistor-level simulation, power supply network and thus the power supply noise is ignored to speed up the simulation. In other words, we simulate only the transistor netlist and assume constant voltages at power buses during the simulation. The model for simulation is illustrated in Fig. 8. Note that the V_{DD} current obtained by this model would be higher than the one obtained by simulating both transistor and power network netlists as the power supply noise on power network will result in lower V_{DD} current. It is important to note that this estimated peak current is used only to *guide* the generation of the input patterns. After good patterns are generated, we simulate both transistor and power network netlists using these patterns to identify critical power supply noise cells.

We define the power supply noise factor of an input pattern v to block m as follows:

$$\text{power\_supply\_noise\_factor}\ (v,m) = \text{peak\_current}\ (v,m) + \alpha * \text{current}\ (v,t) \quad (1)$$

where *peak_current* (v,m) denotes the peak current of block m with respect to input pattern v. The argument t in the term *current* (v,t) represents the time when *peak_current* (v,m) occurs. *Current* (v,t) is the current of the entire design at time t to v. The correlation factor α is defined as the reciprocal of the number of blocks.

We perform a new experiment to demonstrate the high correlation between the number of critical cells found and the power supply noise factor in (1). We select the largest design used (D6 in Table VIII), randomly generate 100 input patterns, and then perform transistor-level power/current simulation to the design by applying these input patterns. The power supply noise factor with respect to each input pattern and block (*power_supply_noise_factor* (v,m) in(1)) is then computed based on the simulation results. To identify the critical cells, we also perform power network simulation for the design based on the same set of input patterns, and then obtain the number of critical cells found in each block in the design caused by each pattern. Fig. 9 shows the relationship between the power supply noise factor and the number of critical cells of the largest block in the design. The horizontal axis represents the power supply noise factor and the vertical axis corresponds to the number of the cells. This figure shows that input patterns with high power supply noise factor can identify large number of critical cells. The correlation between these two quantities can further be illustrated by calculating the correlation coefficient (ρ). Table III shows the values of ρ for all the blocks in the design.

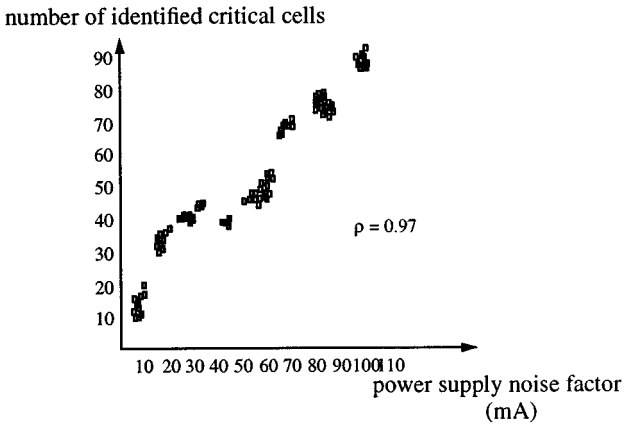

Fig. 9. The relationship between the power supply noise factor and the number of critical cells.

TABLE III
CORRELATION COEFFICIENT BETWEEN POWER SUPPLY NOISE FACTOR AND NUMBER OF CRITICAL CELLS FOUND

block index	1	2	3	4	5	6	7	8	9	10	11	12
ρ	0.94	0.96	0.95	0.95	0.97	0.97	0.97	0.98	0.96	0.97	0.96	0.97

For most of the blocks, ρ is close to 1. This indicated that these two factors are strongly correlated.

We use this power supply noise factor as a measure of power supply noise. The objective is to find the input pattern with the maximum factor value for each block. Instead of performing GA for one block at a time, we propose a new approach, which utilizes the "group search" feature of GA, to maximize multi-objectives within a single GA run. The method proceeds as follows. To make GA maximization process covering each block, we set the population size as the number of blocks, and perform one-to-one mapping between input patterns and blocks. For each pattern, the fitness is referred to as the corresponding power supply noise factor of the mapped block. We propose to find the mapping with the maximum summation of the fitness, and then use the fitness to generate the new population of input patterns.

At each generation of GA, selection process is biased toward input patterns with higher fitness values so that the average fitness values, and hence average power supply noise factor of the input patterns in the next generation tends to increase. Also, the mapping with the maximum summation of the power supply noise factor allows guiding the GA toward selecting the patterns with high power supply noise factor for each block. This achieves the objective of maximizing the power supply noise factor values of all the blocks.

Consider the example shown in Fig. 10 with four input patterns: 1, 2, 3, and 4, and four blocks: a, b, c, and d. The weight of edge V_{ij} between pattern i and block j is the power supply noise factor. Suppose the mapping (denoted by the thick lines) have the maximum summation. Then, the fitness for patterns 1, 2, 3, and 4 are V_{1c}, V_{2a}, V_{3d}, and V_{4b}, respectively. The mapping is referred to as the maximum weighted matching of a bipartite graph, which can be efficiently solved by the Hungarian method [17].

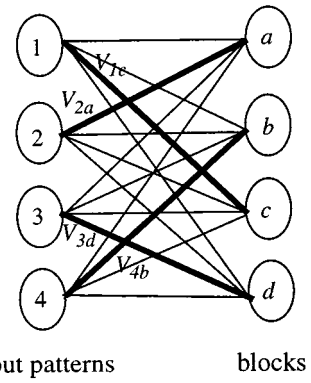

Fig. 10. The fitness determination for input patterns.

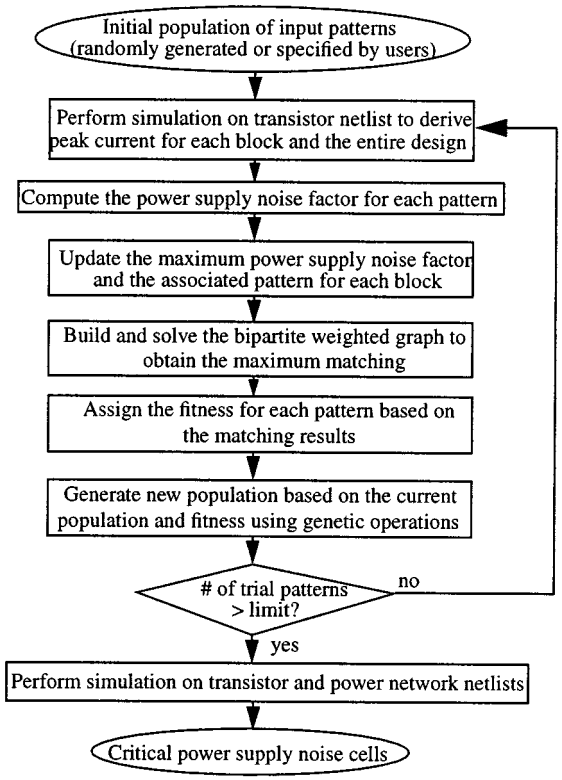

Fig. 11. The overall flow of our technique for critical power supply noise problem.

For each iteration, we update the maximum power supply noise factor and the associated input pattern with respect to each block. The genetic operation schema and the termination condition of the process are the same as described in Section II-B. Finally, we perform the power network simulation to these recorded input patterns. The overall flow of our technique is shown in Fig. 11.

V. EXPERIMENTAL RESULTS

A. Experiments for Calibrating Maximum Power Supply Noise Estimation

We conducted an experiment to validate the vector generation method for estimating maximum power supply noise. In

TABLE IV
POWER SUPPLY NOISE FOR 100 RANDOM PATTERNS FOR BENCHMARK CIRCUITS

Circuits	Power supply noise estimation								Average CPU time (sec.)		
	Waveform simulator					PowrMill		HSPICE	Wave form simulator	Power Mill	HSPICE
	Min (V)	Ave (V)	Max (V)	Ave |ΔV|	Ave error (%)	Ave |ΔV|	Ave error (%)	Ave (V)			
C1355	0.31	0.44	0.50	0.03	6.4	0	0	0.47	0.1	12.4	116
C1908	0.11	0.28	0.43	0.04	12.5	0	0	0.32	0.1	42.4	709
C2670	0.31	0.47	0.64	0.04	7.8	0.02	3.9	0.51	0.1	18.5	1042
C3540	0.20	0.31	0.44	0.02	6.9	0.02	6.8	0.29	1.6	31.9	1665
C5315	0.48	0.62	0.78	0.04	6.9	0.04	6.8	0.58	2.0	63.5	3559
s1196	0.28	0.34	0.42	0.04	13.3	0	0	0.30	0.1	3.8	2162
s1238	0.31	0.34	0.44	0.04	10.5	0.02	5.3	0.38	0.1	5.1	3065
s1423	0.54	0.62	0.71	0.05	8.7	0.02	3.5	0.57	0.1	6.5	461
s1488	0.34	0.39	0.47	0.04	11.4	0.02	5.7	0.35	0.1	8.8	2155
s1494	0.32	0.38	0.46	0.04	9.5	0.01	2.4	0.42	0.1	9.3	1666
s5378	0.41	0.63	0.77	0.05	8.6	0	0	0.58	0.2	23.6	5230
average	-	-	-	-	9.3	-	3.1	-	0.4	20.5	1985

this experiment, we used a 0.55 μm and 3.3 V CMOS library and a physical design system GARDS [11] to layout each experimented benchmark circuit. After the physical design, we further used GARDS to extract the power/ground net RC-trees and compute the RC's of all segments. For the $V_{\rm dd}$ and $V_{\rm ss}$ pins characteristics, we apply the values used in [2], which are from the I/O buffer information specification of Intel Pentium chip, where $L = 11.33$ nH, $C = 3.48$ pF, $R \cong 0\ \Omega$ (1 Ω was used in our experiment) [12]. We used the same RLC characteristics of $V_{\rm dd}/V_{\rm ss}$ pins for all the benchmark circuits and 12 industrial circuits during PowerMill and RailMill simulations. For generating all current/voltage waveform libraries and power segment waveform libraries, we perform the HSPICE simulation with level-3 model parameters for 0.55 μm feature size used in GARDS.

1) Estimation Errors of Power Supply Noise for a Given Input Pattern: In order to evaluate the accuracy of the waveform simulator given in Section III-B, we generate 100 random patterns (2-vector sequence) and perform simulation for each circuit by applying three different simulators: (1) the waveform simulator, (2) PowerMill and (3) HSPICE. Table IV show the results for the peak power supply noise for the 11 ISCAS89 benchmark circuits based on the three different simulators. Columns 2–4 show: 1) the minimum; 2) the average; and 3) the maximum power supply noise for the 100 patterns based on the waveform simulator. Columns 5–6, 7–8 show the average absolute errors ($|\Delta V|$) and the corresponding error percentage of our results and PowerMill results as compared with the noise derived by HSPICE simulation. The average power supply noise based on HSPICE is shown in Column 9. The average CPU times per simulation run for the three simulators are also reported.

On the average, the average estimation error of our method compared to HSPICE is 9.3%. The average estimation error for PowerMill is 3.1%. On the other hand, HSPICE simulation cannot estimate the power supply noise for large benchmark circuits in a reasonable time. For circuit s5378, PowerMill needs on average 23.6 s to simulate one pattern, and the waveform simulator needs only 0.2 s. The reasonable accuracy and the high efficiency of the waveform simulator make it possible to serve as the core of the GA-based test generation to explore the huge solution space for this application.

TABLE V
THE COMPARISON OF POWER SUPPLY NOISE BY OUR PATTERNS AND ALL PATTERNS (BASED ON 0.25 μ, 2.5 V TECHNOLOGY)

ckt	maximum power supply noise			
	our patterns		all possible patterns	
	value	norm.	value	norm.
cm42a	5.4	0.96	5.6	1.00
cm82a	4.1	1.00	4.1	1.00
cm85a	9.8	0.99	9.9	1.00
cm138a	4.5	0.96	4.7	1.00
cmb	5.7	1.00	5.7	1.00
cu	9.4	0.97	9.7	1.00
vda	50.3	0.95	53.1	1.00
Ave.	-	0.98	-	1.00

2) Comparison of Maximum Power Supply Noise by the Obtained Value and True Value: We perform the following experiment to show the effectiveness of the GA-based characterization pattern generation procedure in Section III. We try to simulate all possible input patterns using HSPICE and then extract the maximum power supply noise which are used to compare with the ones derived by simulating only the patterns generated by our technique. Due to the large number of HSPICE simulation runs needed for circuits with a large number of primary inputs, this validation experiment is only applied to circuits with a small number of inputs. We run HSPICE for two sets of input patterns: 1) our patterns (the number is 20 in this experiment) and 2) all possible input patterns. Table V shows the worst-case power supply noise caused by the two sets of patterns for the 7 small MCNC91 benchmark circuits. The worst-case power supply noise (ΔI noise, and IR drop) and normalized values by: 1) our patterns and 2) all patterns are shown in Columns 2–3, and 4–5, respectively. All normalized values are with respect to the values derived by all patterns, and the experimental results are based on a 0.25 μm with supply voltage 2.5 V library. The experimental results show that, on average, our patterns give only 2% lower worst-case power supply noise.

3) Estimation for Maximum Power Supply Noise for Benchmark Circuits: For each benchmark circuit, we apply the GA-based test generation process based on the waveform simulator to generate the input patterns and then select ten patterns producing the highest power supply noise. The size of population of GA in the experiment is 30, and the number of generations is 50. That means the number of patterns simulated by the waveform simulator is 1500. These settings of these two control parameters have been used for all the benchmarks in our experiments. Different sets of values were tried out, and the values listed were selected as they were consistently yielding good results. Then we simulate the obtained ten patterns using PowerMill [20]. The reported maximum power supply noise

TABLE VI
ESTIMATION FOR MAXIMUM POWER SUPPLY NOISE FOR BENCHMARK CIRCUITS

	Maximum Power Supply Noise						CPU time (min.)		
	Weighted Random		GA with PowerMill Only		Ours		Weighted Random	GA with PowerMill Only	Ours
	(V)	normal.	(V)	normal.	(V)	normal.			
C1355	0.42	1	0.53	1.26	0.68	1.62	62	63	23
C1908	0.64	1	0.67	1.05	0.68	1.06	212	213	34
C2670	0.70	1	0.70	1.00	0.79	1.13	93	95	31
C3540	0.69	1	0.72	1.04	0.79	1.15	160	164	65
C5315	0.72	1	0.81	1.13	0.92	1.28	318	321	71
C6288	0.91	1	1.02	1.12	1.15	1.26	760	767	155
C7552	0.91	1	0.88	0.97	1.13	1.24	785	789	132
s1196	0.66	1	0.66	1.00	0.66	1.00	10	10	9
s1238	0.63	1	0.63	1.00	0.75	1.19	13	13	13
s1423	0.51	1	0.53	1.04	0.62	1.22	16	16	16
s1488	0.57	1	0.55	0.96	0.61	1.07	22	22	21
s1494	0.58	1	0.61	1.05	0.62	1.69	23	23	21
s5378	0.69	1	0.78	1.13	0.84	1.22	59	60	31
s9234	0.62	1	0.67	1.08	0.82	1.32	48	50	23
s13207	0.81	1	0.77	0.95	0.86	1.06	203	207	60
s15850	0.72	1	0.66	0.92	0.83	1.15	737	742	131
s38417	0.72	1	0.75	1.04	0.86	1.19	1317	1321	225
s38584	0.85	1	0.95	1.12	1.10	1.29	1530	1532	247
s35932	1.06	1	1.16	1.09	1.25	1.18	1715	1721	276
average	-	1	-	1.05	-	1.23	425	427	83

is referred to as a tight lower bound of the maximum power supply noise. To evaluate our technique, we compare the results with those produced by two different test generation techniques. 1) Apply an input pattern generator embedded in PowerMill, named *GAP* [20], which employs the same GA-based procedure but uses PowerMill as the underlying simulator instead of the waveform simulator. Because of the higher simulation time per pattern, we reduce the number of total simulation runs to 300 for combinational circuits (the size of population is 10 and the number of generations is 30), and 150 simulation runs for sequential circuits (with the same size population and 15 generations). 2) Simply apply the same number of the weighted random patterns with primary input switching probability of 0.9, and simulate these patterns using PowerMill to identify the one producing the highest power supply noise. The reason for using this switching probability is because simulating patterns in which each input has a transition does not have to necessarily produce the maximum current. This is especially true for circuits with XOR gates [23].

The estimated maximum power supply noise for the benchmark circuits are shown in Table VI. The maximum supply noise and normalized values estimated by: 1) weighted random approach; 2) *GAP*; and 3) our approach are shown in Columns 2–3, 4–5 and 6–7, respectively. All the normalized values are with respect to the values derived by the weighted random approach. Note that all the values of power supply noise are reported by the same simulator—PowerMill. The CPU times for the three approaches are reported in Columns 8, 9, and 10, respectively. The CPU times shown in Column 10 include the CPU time for computing the effective power/ground nets RC's for each block, the waveform simulation time for 1500 patterns, the runtimes for RC-reduction and HSPICE simulation for building the power segment waveform libraries, and the PowerMill simulation time for the final ten patterns.

The experimental results show that, on the average, our approach gives 23% and 17% tighter lower bounds for the benchmark set, than the bounds obtained with the weighted random approach and *GAP*, respectively. For the CPU time of the largest benchmark circuits s35932, the weighted random and *GAP*-only approaches need 28.5 hours to estimate the maximum power supply noise, and our approach needs only 4.6 hours.

B. Critical Power Supply Noise Identification

The second experiment is to characterize the effectiveness and efficiency of the pattern generator *VIP* for identifying critical nodes. In this experiment, we use a 0.25 μm library with a supply voltage of 2.5 V. For small benchmark circuits, we simulated the power network and the transistor netlist using HSPICE for all possible input patterns and then report all the critical power supply noise cells. These cells are used to compare with the ones derived by simulating only the patterns which are generated by *VIP* and *GAP*, which uses to the peak current of the entire designs as the fitness of each pattern. Due to the large number of simulation runs needed for circuits with a large number of primary inputs, this validation experiment is only applied to circuits with a small number of inputs. Before performing the simulation, we partition each circuit into blocks based on the sizes and topology of circuits. The simulation are performed for three sets of input patterns: 1) patterns generated by *VIP* (the number is the same as the number of blocks for each circuit); 2) patterns generated by *GAP* (the number is the same as the *VIP* uses); and 3) all possible input patterns. Table VII shows the number of critical power supply noise cells identified by the three sets of patterns for the 7 small MCNC91 benchmark circuits.

The number of critical power supply noise cells and normalized values by: 1) *VIP*; 2) *GAP*; and 3) all patterns are shown in Columns 2–3, 4–5, and 6–7, respectively. All normalized values are with respect to the values derived by all patterns. Column 8 gives the number of blocks for each circuit where this number is the same as the number of simulated patterns used by *VIP* and *GAP*. Columns 9 and 10 show the specified threshold power supply noise and threshold time period. The experimental results show that, on average, four patterns generated by *VIP* identify 96% of the critical power supply noise cells.

VIP is also tested to a set of industrial designs with a wide range of applications such as CPUs, DSP processors, and large memory banks. The number of transistors ranges from 61 K to 1.14 M, and the technology from 0.25 μ to 0.8 μ. An industrial RC extraction tool Arcadia [19] is used to extract the power/ground net RCs from its layout database and to generate the power/ground netlists. PowerMill [20] is used as the embedded simulator to report the current for fitness computation. Table VIII shows the design statistics. Columns 2, 3, and 4 show the number of primary inputs, process technology, and power supply voltage. The numbers of transistors and power net RCs are shown in Columns 5 and 6, respectively. Columns 7 and 8 show the specified threshold power supply noise and time period.

TABLE VII
THE COMPARISON OF CRITICAL POWER SUPPLY NOISE CELLS FOR THREE SETS OF INPUT PATTERNS

Ckt.	# of the identified critical supply noise cells						#. of blocks	threshold power supply noise (mV)	threshold time period (ns)
	VIP		GAP		all possible patterns				
	#.	norm.	#.	norm.	#.	norm.			
cm42a	4	1.00	4	1.00	4	1.00	3	5.6	0.02
cm82a	5	1.00	4	0.80	5	1.00	3	4.1	0.02
cm85a	6	0.86	2	0.29	7	1.00	3	9.9	0.02
cm138a	5	1.00	4	0.80	5	1.00	4	4.7	0.02
cmb	0	1.00	0	1.00	0	1.00	4	5.7	0.02
cu	1	1.00	1	1.00	1	1.00	3	9.7	0.02
vda	18	0.86	11	0.52	21	1.00	8	53.1	0.04
Ave.	-	0.96	-	0.77	-	1.00	4	-	-

TABLE VIII
STATISTICS OF 12 INDUSTRIAL DESIGNS USED FOR EXPERIMENTS

Designs	# of PIs	technology (μ)	supply voltage (V)	#. of transistors	#. of power net' RCs	threshold supply noise (V)	threshold time period (ns)
1	35	0.25	2.5	1.0M	898K	0.09	0.03
2	194	0.28	2.2	27.9K	83K	0.10	0.04
3	112	0.5	3.3	1.01M	1.81M	0.60	0.06
4	165	0.5	3.3	472K	937K	0.45	0.05
5	108	0.5	3.3	508K	602K	0.55	0.06
6	109	0.6	3.3	1.14M	1.19M	0.50	0.03
7	135	0.8	4.0	628K	590K	1.05	0.1
8	97	0.8	4.0	329K	863K	1.50	0.15
9	112	0.8	5.0	208K	375K	1.25	0.07
10	175	0.8	5.0	62K	310K	0.80	0.08
11	14	1.0	5.0	61K	52K	0.75	0.1
12	98	1.0	5.0	73K	88.5K	0.10	0.1

In this experiment, a power network simulator RailMill [21] is used to validate the quality of the obtained input patterns. We compare the simulation results reported by RailMill for three different pattern sets: 1) patterns generated by *VIP*; 2) patterns generated by *GAP*; and 3) functional verification vectors provided by the designers. Table IX shows the comparison for the 12 tested industrial designs. The numbers of the critical power supply noise cells found by: 1) *VIP*; 2) *GAP*; and 3) functional vectors are shown in Columns 2, 3, and 4, respectively. Columns 5–7 show the level of the obtained maximum power supply noise. The total number of patterns used for running both PowerMill and RailMill is shown in Columns 8–10. Table X gives the CPU time for each technique. The CPU time consumed by *VIP* for: 1) solving the bipartite graph for the maximum weight matching; 2) PowerMill simulation; 3) RailMill simulation is shown in Columns 2, 3, and 4, respectively. Column 5 gives the overall CPU time *VIP* consumes. The CPU time for *GAP* is shown in Columns 6–8. Column 9 gives the time for functional vectors used by RailMill simulation.

For designs D3, D6, and D8, *VIP* identifies a large number of critical power supply noise cells and none of them can be found by the other two approaches. For designs D2 and D12, no critical power supply noise appears and these two designs qualify the power supply noise test. For all the other designs, the number of critical cells found by *VIP* is much higher than those by the others. *VIP* also obtains higher or equal maximum power supply noise compared to those by *GAP* for eleven out of the 12 tested designs. For design D4, *VIP* obtains lower maximum power supply noise than *GAP*. However, the number of critical nodes found is 2.6 times higher. For the largest design D6, simulating the functional vectors needs 68.3 h and *VIP* needs only 7.0 h.

TABLE IX
THE COMPARISON FOR POWER SUPPLY NOISE ANALYSIS

Designs	# of critical cells			max. supply noise (V)			#. of patterns		
	VIP	GAP	func. vec.	VIP	GAP	func. vec.	VIP	GAP	func. vec.
D1	1976	1300	525	0.09	0.09	0.09	168	152	323
D2	0	0	0	0.05	0.05	0.02	1800	1200	720
D3	1179	0	0	0.70	0.55	0.57	108	180	551
D4	3953	1520	119	0.47	0.56	0.45	360	250	1400
D5	344	285	0	0.60	0.58	0.53	696	400	2000
D6	1322	0	0	0.62	0.48	0.35	120	140	500
D7	816	700	18	1.10	1.10	1.05	240	360	1431
D8	2059	0	0	1.74	1.40	1.05	130	110	488
D9	1191	760	58	1.30	1.30	1.26	252	200	160
D10	2280	1170	0	0.86	0.84	0.34	840	680	823
D11	64	52	35	0.90	0.90	0.90	100	120	983
D12	0	0	0	0.04	0.04	0.04	228	160	650
Ave.	1265	482	63	-	-	-	420	329	836

TABLE X
THE COMPARISON OF CPU TIME FOR POWER SUPPLY NOISE ANALYSIS

Designs	CPU time (min.)							func. vec.
	VIP				GAP			
	solving graph	PowerMill	RailMill	total	PowerMill	RailMill	total	
D1	5	174	121	300	157	129	286	554
D2	3	308	5	316	204	6	210	17
D3	6	391	153	550	414	136	550	830
D4	8	380	187	575	410	174	584	1612
D5	6	278	54	338	163	51	214	1784
D6	5	313	100	418	365	90	455	4096
D7	5	210	88	303	205	90	295	443
D8	4	254	106	364	200	110	310	389
D9	4	106	10	120	89	11	100	924
D10	5	153	178	336	102	170	272	840
D11	3	72	4	79	91	4	95	17
D12	3	94	9	106	68	10	78	207
Ave.	5	228	85	317	206	82	288	976

VI. CONCLUSION

We propose two efficient techniques for generating patterns that would produce high power supply noise and could effectively verify the power network reliability. The obtained patterns can be used to estimate maximum power supply noise. They can

also be used for power network reliability analysis to identify cells experiencing excessive power supply noise. The experimental results show that the patterns generated using the proposed approach result in much tighter lower bound on the maximum power supply noise, in comparison with the results obtained by other test generation schemes. Also, the experimental results indicate that the generated input patterns successfully identify cells that encounter critical voltage drops but missed by other sources of vectors. These two techniques can be applied to generate patterns that would cause high power supply noise at any interested block and included in the design cycle for accurate power network reliability analysis.

REFERENCES

[1] M. A. Breuer and S. K. Gupta, "Process aggravated noise (PAN): New validation and test problems," in *Proc. Int. Test Conf.*, Nov. 1996, pp. 914–923.

[2] Y.-S. Chang, S. K. Gupta, and M. A. Breuer, "Analysis of ground bounce in deep sub-micron circuits," *Proc. 15th IEEE VLSI Test Symp.*, pp. 110–116, Apr. 1997.

[3] H. H. Chen and D. D. Ling, "Power supply noise analysis methodology for deep-submicron VLSI chip design," in *Proc. Design Automation Conf.*, June 1997, pp. 638–643.

[4] S. Chowdhury and J. S. Barkatullah, "Estimation of maximum current in MOS IC logic circuits," *IEEE Trans. Computer-Aided Design*, pp. 642–654, June 1990.

[5] A.-C. Deng, "Power analysis for CMOS/BiCMOS circuits," in *Int. Symp. Low-Power Electronics and Design (ISLPED)*, 1994, pp. 3–8.

[6] ——, "Power estimation and power noise analysis for CMOS circuits," *J. Circuits Syst. Comp.*, pp. 17–30, Feb. 1997.

[7] A.-C. Deng, Y.-C. Shiau, and K.-H. Loh, "Time domain current waveform simulation of CMOS circuits," in *Proc. Int. Conf. Computer-Aided Design*, Nov. 1988, pp. 208–211.

[8] S. Devadas, K. Keutzer, and J. White, "Estimation of power dissipation in CMOS combinational circuits using Boolean function manipulation," *IEEE Trans. Computer-Aided Design*, pp. 373–383, Mar. 1992.

[9] M. S. Hsiao, E. M. Rudnick, and J. H. Patel, "$K2$: An estimator for peak sustainable power of VLSI circuits," in *Int. Symp. Low-Power Electronics Design (ISLPED)*, Aug. 1997, pp. 178–183.

[10] D. E. Goldberg and R. Burch, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading, MA: Addison-Wesley, 1989.

[11] *Command Reference Manual*, vol. 1–4, Silicon Valley Research, Sept. 1996. GARDS.

[12] I/O Buffer Information Specification (IBIS). [Online]. Available: ftp://ftp.vhdl.org/pub/ibis/models/intel/pentium/pp100sem.ibs

[13] Y.-M. Jiang and K.-T. Cheng, "Analysis of performance impact caused by power supply noise in deep submicron devices," in *Proc. Design Automation Conf.*, June 1999, pp. 760–765.

[14] Y.-M. Jiang, A. Krstic, and K.-T. Cheng, "Estimation for maximum instantaneous current through supply lines for CMOS circuits," *IEEE Transactions on VLSI Syst.*, pp. 61–73, Feb. 2000.

[15] 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. Comput.-Aided Design*, pp. 998–1012, Aug. 1995.

[16] B. L. Miller and D. E. Goldberg, "Genetic algorithms, tournament selection, and the effects of noise," *Complex Syst.*, pp. 193–212, June 1995.

[17] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization, Algorithms and Complexity*. Englewood Cliffs, NJ: Prentice-Hall, 1982.

[18] R. Senthinathan and J. L. Prince, *Simultaneous Switching Noise of CMOS Devices and Systems*. Boston, MA: Kluwer, 1994.

[19] *Arcadia User Guide*, SYNOPSYS, June 1999.

[20] *PowerMill Reference Manual*, SYNOPSYS, June 1999.

[21] *RailMill User Guide*, SYNOPSYS, June 1999.

[22] *Ultima-PR User's Guide 2.2.6*, Ultima Interconnect Technology, Inc., 1997.

[23] C.-Y. Wang and K. Roy, "Maximum power estimation for CMOS circuits using deterministic and statistic approaches," *IEEE Trans. VLSI Syst.*, pp. 134–140, Mar. 1998.

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

[25] K. Yoshida, "Speed up of an evaluation speed 5–15 times faster by using hierarchical power supply network," NTT LSI Lab. Rep.

Yi-Min Jiang received the B.S. and M.S. degrees in electrical engineering from National Tsing Hua University, Taiwan, R.O.C., in 1989 and 1991, respectively, and the Ph.D. degree in electrical and computer engineering from the University of California, Santa Barbara, in 1999.

Since 1999, he has been with Synopsys, Mountain View, CA, working in the power net reliability analysis group. His research interests include the maximum power/noise estimation and modeling of noise effects in deep-submicron designs.

Kwang-Ting (Tim) Cheng (S'88–M'88–SM'98–F'00) received the B.S. degree in electrical engineering from National Taiwan University and the Ph.D. degree in electrical engineering and computer science from the University of California, Berkeley in 1983 and 1988, respectively.

From 1988 to 1993, he was with AT&T Bell Laboratories, Murray Hill, NJ. In 1993, he joined the faculty of the University of California, Santa Barbara, where he is currently Professor of Electrical and Computer Engineering and Director of the Computer Engineering Program. His current research interests include VLSI testing, design synthesis, and design verification, and he holds nine U.S. Patents in these areas. He has published over 150 technical papers and coauthored three books. He received Best Paper awards at the 1994 Design Automation Conference, 1999 Design Automation Conference and 1987 AT&T Conference on Electronic Testing.

Dr. Cheng currently serves as Associate Editor-in-Chief for IEEE DESIGN and TEST of COMPUTERS. He also serves on the Editorial Boards of the IEEE TRANSACTIONS on COMPUTER-AIDED DESIGN, and *Journal of Electronic Testing: Theory and Applications*. He served as General Chair and Program Chair of the IEEE International Test Synthesis Workshop. He has also served on the technical program committees for several international conferences on CAD and testing including DAC, ICCAD, ITC, and VTS.