Probabilistic Simulation for Reliability Analysis of CMOS VLSI Circuits

by

Farid N. Najm†, Richard Burch‡, Ping Yang‡, and Ibrahim N. Hajj†

†Coordinated Science Laboratory
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801

‡VLSI Design Laboratory
Texas Instruments Inc.
Dallas, Texas 75265

Abstract

A novel current-estimation approach is developed to support the analysis of electromigration failures in power supply and ground busses of CMOS VLSI circuits. It uses the original concept of *probabilistic simulation* to efficiently generate accurate estimates of the *expected current waveform* required for electromigration analysis. As such, the approach is *pattern-independent* and relieves the designer of the tedious task of specifying logical input waveforms. This approach has been implemented in the program CREST which has shown excellent accuracy and dramatic speedups compared to traditional approaches. We describe the approach and its implementation, and present the results of numerous CREST runs on real circuits.

F. Najm is now with the VLSI Design Laboratory, Texas Instruments Inc., Dallas, Texas 75265

This work was supported by Texas Instruments Incorporated, and the US Air Force Rome Air Development Center.

1 Introduction

The reliability of integrated circuits is a major concern for the electronics industry. As higher levels of integration are used, the minimum line width and line separation will decrease, thereby increasing the chip failure rate. This indicates that the importance of reliability can only increase in the future. It is, therefore, imperative that circuits are designed with reliability in mind.

This work addresses *electromigration* [1, 2] (EM), which is a major reliability problem caused by the transport of atoms in a metal line due to the electron flow. Under persistent current stress, this can cause deformations of the metal leading to either short or open circuits. The failure rate due to EM depends on the current density in the metal lines and is usually expressed as a median time-to-failure (MTF). There is a definite need for CAD tools that predict the susceptibility of a given design to EM failures.

A simulation tool, SPIDER [3], has been developed to estimate the MTF for each section of a metal bus corresponding to any user-selected interconnect signal. It requires the user to specify current sources to load the metal bus at specified contact points. Using these current sources, SPIDER extracts an equivalent resistance network to represent the bus, simulates the network using SPICE [4] to determine the current density in each section, and then estimates the MTF of each section using models developed in [5]. The user is, however, left with the problem of specifying current sources. This can be very hard to do for a big chip, especially for CMOS circuits, because they draw current only during switching transients. Hence there is a need for a CAD tool that derives these currents.

We present a new technique for solving this *current estimation* problem for CMOS circuits. This has been implemented in the program CREST (CuRrent ESTimator) and has proven to be very effective both in terms of accuracy and speed. We focus our attention on the power and ground busses, and derive loading currents for them to be used for MTF estimation. These busses are the usual, although not only, locations of severe EM failures. Preliminary results of this works have been presented in [6, 7].

It is important to understand exactly what information about the current is needed for EM analysis. CMOS circuits, as pointed out above, draw current only during switching, and, therefore, produce a *non-dc* current waveform. It is well known [5] that, in the presence

of such waveforms, the MTF due to electromigration is dependent on the shape of the waveform and not simply on its time-average (ie area). So a simple averaging approach is unacceptable. On the other hand, the MTF is the combined effect of a large number of current waveforms, corresponding to the variety of logical waveforms that can be applied at the circuit inputs during typical operation. It would be insufficient, therefore, to use a standard timing simulator to derive the current corresponding to a single set of logical input transitions. It is also obvious that redoing a such a simulation for every possible input transition is impractical, since for a circuit with n inputs, the number of possible transitions at the inputs is 2^{2n}.

CREST overcomes this problem by deriving an *expected current waveform*; this is a waveform whose value at a given time is the weighted average of all possible current values at that time, as shown in Fig. 1. Such a waveform is a good compromise between an *unacceptable* time-average and an *insufficient* single-transition estimate of the current, and provides an appropriate current estimate for electromigration analysis.

To derive this waveform, CREST considers a user-specified range (or set) of possible input values and/or transitions and calculates the expected current waveform *over* this range. Rather than enumerating the set of inputs and averaging the corresponding current results (which would be impractical), CREST uses *statistical* information about the inputs to directly derive the required expected current waveform. The resulting methodology is what we call a *probabilistic simulation* of the circuit. In general, it can be slightly more time consuming than standard timing simulation, but it needs to be applied only once, resulting in significant speedup.

Several simplifying assumptions and/or approximations will be made in the following sections to make the problem computationally tractable. Whenever possible, we will attempt to justify these assumptions. However, for lack of space, this will not always be possible, and the reader will be referred to appropriate references. Nevertheless, we will offer a verification of the overall approach on a *global* scale, by comparing the end result of the simulation (expected waveform) from CREST with that derived using SPICE.

The rest of this paper is organized as follows. The next section gives a system overview to introduce some basic concepts and the overall simulation strategy. Section 3 explains the basic current estimation algorithm as part of the simulation of standard CMOS gates.

Pass-transistor circuits are handled in section 4. Section 5 discusses supergates, required to handle signal dependence. Section 6 presents implementation issues and results, and the last section draws some conclusions and indicates the direction of future research. The appendix discusses a basic graph reduction operation that will be frequently used.

2 System Overview

We have developed a new current estimation approach that derives the expected current waveform directly from probabilistic information about the inputs. A single event-driven simulation, similar in some aspects to timing simulation, is the basis of the technique. Before going into the details, we introduce some basic concepts and provide the theoretical groundwork.

Consider a set Ω, each element of which represents a combination of *logical waveforms* to be applied at the circuit inputs[1]; this is the range of inputs over which the expected current waveform is to be derived. If certain probabilities are assigned to the elements of Ω, then we can think of it as a *probability space* [8]. Associated with each element of Ω is an actual current waveform that the circuit would draw if subjected to that combination of inputs. This association (or mapping) defines a *stochastic process* $i(t)$ whose *mean* $E[i(t)]$ is the expected current waveform to be derived. Likewise, every input node N_i has associated with it a stochastic process $x_{N_i}(t)$ that embodies the different possible logic waveforms allowed at N_i. These in turn define other processes at the internal nodes of the circuit.

The technique to be presented takes the user's input specifications (defining the processes $x_{N_i}(t)$) and uses the circuit topology (which defines the mapping from logical inputs to current waveforms) to derive the corresponding processes at internal nodes, decipher the statistics of $i(t)$, and derive its mean. This, somewhat abstract, description will be made more concrete below.

2.1 Probability waveforms

We build on the concept of *signal probabilities* [9], which has recently become popular in the testing field [10]. This can be summarized as follows : a probability value is assigned

[1] Strictly speaking, if the circuit contains memory elements, then its initial state also affects the structure of Ω.

to each input node to indicate the probability that it is high. The circuit topology is then used to propagate these values so that the probability at any internal node is derived. The probability space in this case is a subset of the set Ω defined above, because it contains steady state values only, and not waveforms.

We extend the signal probabilities concept by defining *transition probabilities*. The transition probability of a signal N at time t is the probability of a low-to-high transition from t^- (just before t) to t^+ (just after t), denoted $P_{N,lh}(t)$. Given these probabilities at the inputs then internal node transition probabilities can be derived from them.

Using transition probabilities we can compactly describe a large set of logic waveforms. For example : input N is high with probability .76 at time 0 ($P_{N,h}(0) = .76$), switches low-to-high with probability .5 at time 2ns ($P_{N,lh}(2ns) = .5$), is then high with probability .35 at time 3ns ($P_{N,h}(3ns) = .35$), etc... . Such an alternating sequence of signal probabilities and transition probabilities will be referred to as a *probability waveform*. An example of a probability waveform and the few logical waveforms it represents is shown in Fig. 2. CREST uses such a waveform as a representation[2] of a stochastic process at a node. The site of a transition probability in this waveform will also be referred to as a *transition edge*, shown as an arrow in Fig. 2. A transition edge is an important part of a probability waveform because it may cause current to be drawn. Transition edges will also be called *probabilistic events*, or simply *events*.

An event at a node N at time t is described by the three probabilities : $P_{N,h}(t^-)$, $P_{N,lh}(t)$, and $P_{N,h}(t^+)$. It is easy to see that these are enough to describe the statistics of the event since other probabilities can be derived from them. For instance, the probability of a high-to-low transition can be easily derived using the identity $P_h(t^+) - P_h(t^-) = P_{lh}(t) - P_{hl}(t)$, which is obtained from simple probability theory.

2.2 Simulation algorithm

CREST is a *probabilistic* simulator since it operates on probabilistic, rather than logical, signals. The simulation algorithm itself, however, is deterministic. The user specifies proba-

[2] This, in fact, is an *incomplete* representation, but is sufficiently accurate for our purposes. It would be prohibitively expensive to maintain a complete representation.

bility waveforms at the primary inputs of the circuit, which are propagated into the circuit to derive the expected current waveform.

The waveforms specified at the circuit inputs constitute the primary means for the user to influence the simulation. They allow one to study the reliability of the circuit under "typical" operating conditions. As described above, every probability waveform consists of a sequence of transition edges, or events, tagged with signal and transition probabilities. If these probabilities are set to 0s and 1s, then the waveform becomes a simple logic waveform. Thus, if enough is known about the activity at the circuit inputs, the user may fine tune the simulation to the point where logical waveforms are specified at some inputs (say, clock inputs). Otherwise, if little is known about the inputs, then one can allow a large variety of possible signals by specifying appropriate probability waveforms.

To prepare a set of probability waveform inputs, there are at least two options. If one is comfortable with the probability waveform concept, then these inputs may be simply entered as sequences of real numbers between 0 and 1. Otherwise, we assume that the user has available a representative set of logical input waveforms from previous logic, timing, or fault simulation runs on his design. The corresponding set of probability waveforms can then be obtained as follows. For a given input node, consider all the logical waveforms that it may experience. Then, for every time point, take the average of the values in all these waveforms, considering high to be 1 and low to be 0. When the signal is not changing, the value thus obtained is the required probability waveform value at that time. When the signal *is* changing, from low to high, then the fraction of the logic waveforms in which it makes that transition at that time gives the required transition probability in its probability waveform.

Given the input probability waveforms, an *event-driven* simulation approach, similar to what is commonly employed by logic or timing simulators, is used to propagate them throughout the circuit. The circuit is divided into gates. When a probabilistic event occurs on the input to a gate, the expected current pulse caused by the event is estimated, and the appropriate probabilistic event is created on the output node of the gate. The expected current pulses from individual gates are summed to create the expected current waveform drawn by the circuit. The program operates on a transistor description of the circuit, which it partitions into primitive gates of two major types : standard CMOS gates and pass transistor

gates. Current pulse estimation and probabilistic event propagation for each of these gate types will be further discussed in sections 3 and 4 below, respectively.

2.3 Signal dependence

A major problem with any tool dealing with a number of statistical quantities is keeping track of, and modeling, the *correlation* or *dependence* among them, CREST is no exception. Dependence between signals in CREST is of two types. The first is a dependence existing between the two values at the same node at two distinct time points, this will be referred to as *temporal dependence*. The second is the dependence existing between two signals if their nodes depend on the same fanout stem in the circuit, this will be referred to as *spatial dependence*. A feedback loop involves both spatial and temporal dependencies; in general two signals may be dependent due to either or both of these types.

CREST accounts for temporal dependence in a limited sense. The dependence between two signal values separated by a single transition edge *is* accounted for - this, in fact, is the reason transition probabilities are introduced. The program, however, does not keep track of the dependence between signal values separated by more than one transition edge. It turns out that this approach is sufficiently accurate for our purposes because the probabilities at (and on both sides of) a transition edge are enough to derive the current pulse corresponding to it. To accurately keep track of temporal dependence between the signals at any two time points in a waveform would be equivalent to a complete representation of a stochastic process (see footnote 2 above) and would be prohibitively expensive.

Spatial dependence is handled using the concept of a *supergate* [10], as explained in section 5 below. It is important to make the point now, however, that this reduces to simulating gates whose inputs are *independent*. The descriptions of the simulation of primitive gates in the following sections will therefore assume that a gate's inputs are independent.

2.4 Graph reduction

This section briefly introduces a *graph reduction* procedure that is central to the simulation algorithms to be presented, and which will be used frequently below. The appendix is devoted to a precise formulation and description of this approach.

Simply stated, the need will frequently arise to derive the probability that a mesh of transistors, joining two specified points, provides a conducting path between them. An example is given in Fig. 3, where the p part of a CMOS complex gate is shown. Typically, the probabilities at the gate nodes of the transistors are known, and the probability at the gate output is required. The general methodology will be to consider a graph that is identical to the transistor mesh (Fig. 3) in which the edges are labelled with the probabilities of the gate nodes. This graph is then *reduced* to a single edge whose label(s) provide the required probabilities. In Fig. 3, the probability that the output is high is simply the probability that the edge ξ is conducting.

Actually, the edges are labelled by both signal *and* transition probabilities, as well as by the expected conductance of the transistors. After the graph is reduced, the resultant labels on ξ determine the event at the output, and its expected conductance is used to derive the expected current. The reader is referred to the appendix for details of this reduction.

3 Standard Gate Simulation

The term *standard gate* will be used to refer to a CMOS fully complementary gate, as shown in Fig 4. The p-block or p-part (n-block or n-part) of a gate will be used to refer to the p (n) channel transistor mesh between its output node and the power supply (ground). A gate will be assumed to have spatially independent inputs. The general case is properly handled using the concept of a supergate, as described in section 2 above, with the independent-inputs-gate-solver used as a subroutine. Simulating a gate is the procedure of analyzing a gate that has certain events at its inputs to derive the corresponding output event and expected current pulse.

Given the events at the inputs of a gate at a certain time t, its output event can be easily derived as follows. Consider the p part of the gate and build a graph that represents it using the transistor gate probabilities to label the graph edges with the probabilities that each edge "is on," "was on," and "transitions from off to on." The graph reduction procedure described in the appendix is then used to reduce the graph to a single edge between V_{dd} and the output node. It is obvious that the probabilities of the single edge ξ remaining at the

end of the reduction give the required output event. The time of occurrence of this event will be derived at the end of this section.

The current estimation procedure at each gate will now be discussed. The expected gate current pulse will be modeled by a *triangular pulse* that starts with a peak of $E[I] \stackrel{\triangle}{=} E[i(t^+)]$ at time t and decays linearly to zero at time $t + \tau$. The rest of this section describes the derivation of $E[I]$ and τ.

We will focus on the *charging* current component and leave out the *direct* component which may be drawn through a path of p and n channel transistors during the transition. This policy has been adopted based on Veendrick's [11] work which suggests that if the gate is *well designed* then the direct current component may be neglected.

Consider the generic CMOS gate structure shown in Fig. 4. The figure shows the p-transistor block, the n-transistor block, and the output node capacitance split into two lumped capacitors C_p to V_{dd} and C_n to V_{ss}. Similarly, each internal node n_i has two capacitances C_{in} and C_{ip}. The values of these capacitances are derived from the circuit description and the transistor model parameters. On a low-to-high transition, the currents flowing through C_n and C_p at the output node are i_{p1} and i_{p2}, respectively, as shown in the figure. The corresponding i_{n1} and i_{n2} for a high-to-low transition are also shown. The currents i_{p2} and i_{n2} are discharging currents that redistribute locally, and we are interested in $i = i_{p1} + i_{n1}$. Of course these currents are associated with the output node only, and the total gate current i_{tot} will be larger than i. However, the output current will play a central role in the derivation.

Let $i_p = i_{p1} + i_{p2}$ and $i_n = i_{n1} + i_{n2}$. It's easy to verify that $i_{p1} = i_p \times C_n/(C_p + C_n)$, and $i_{n1} = i_n \times C_p/(C_p + C_n)$. Therefore :

$$E[i(t)] = E[i_p(t)] \times \frac{C_n}{C_p + C_n} + E[i_n(t)] \times \frac{C_p}{C_p + C_n} \tag{3.1}$$

And in particular, the value at the peak is :

$$E[I] = E[I_p] \times \frac{C_n}{C_p + C_n} + E[I_n] \times \frac{C_p}{C_p + C_n} \tag{3.2}$$

The values of $E[I_p]$ and $E[I_n]$ are derived as follows. For i_p, consider the p part of the gate, and let every transistor T_k be represented by a switch of on-conductance $g_{on,k}$, where

$g_{on,k}$ is (the maximum conductance) given by :

$$g_{on,k} = \frac{\beta_k}{2V_{dd}}(V_{dd} - V_T)^2(1 + \lambda V_{dd}) \tag{3.3}$$

where V_T is the magnitude of the transistor threshold voltage, β_k is its transconductance, and λ is the channel length modulation factor. This definition of $g_{on,k}$ is used because we are interested in the peak current drawn, which occurs during saturation. The value of $g_{on,k}$ is derived from the transistor model parameters given by the user.

Now let $G_p(t)$ be the random conductance between the output node and V_{dd}. G_p is a function of the individual transistors' random conductances g_k, where g_k is 0 if the transistor is off and $g_{on,k}$ if it is on. If an event occurs at the gate at time t, then the value of $E[G_p(t^+)]$ and the previous state of the output node, $V_o(t^-)$, will determine $E[I_p]$. Formally, we have $E[I_p] = E[(V_{dd} - V_o(t^-)) \times G_p(t^+)]$, which becomes :

$$E[I_p] = V_{dd} \times E[G_p(t^+) \mid G_p(t^-) = 0] \times P(G_p(t^-) = 0) \tag{3.4}$$

where $P(A)$ is the probability of the event A, and $E[A \mid B]$ denotes the *conditional expected value* of A given B. The formula is correct because if $G_p(t^-) = 0$ (1) then $V_o(t^-) = 0$ (V_{dd}). Similarly for the n part of the gate, we get :

$$E[I_n] = V_{dd} \times E[G_n(t^+) \mid G_n(t^-) = 0] \times P(G_n(t^-) = 0) \tag{3.5}$$

If the gate inputs at t^+ are independent of their values at t^- then $E[G_p(t^+) \mid G_p(t^-) = 0] = E[G_p(t^+)]$ and the problem would be simplified. In this case the value of $E[G_p]$ (or $E[G_n]$) may be derived from the graph by considering a graph representation of the p (n) block of the gate using the conductances $E[g_k]$ of the transistors and their gate node probabilities and performing a graph reduction. However this is not true in general and the dependence between $G(t^+)$ and $G(t^-)$ should be taken into account.

To find the *conditional* expected value of G_p, $E[G_p(t^+) \mid G_p(t^-) = 0]$, we perform the graph reduction using $E[g_k(t^+) \mid G_p(t^-) = 0]$, instead of $E[g_k(t^+)]$, for every transistor; likewise for G_n. If x_k is the gate node of transistor T_k in the p-part, then it can be shown [12] that :

$$E[g_k(t^+) \mid G_p(t^-) = 0] = g_{on,k} \times \left\{ \frac{P_{x_k,ll}(t)}{P_{x_k,l}(t^-)} + \right.$$

$$\left[P_{x_k,lh}(t) \;-\; P_{x_k,l}(t^-)P_{x_k,h}(t^+)\right] \times \frac{P(G_p(t^-)=0 \mid x_k(t^-)\neq 0)}{P(G_p(t^-)=0)P(x_k(t^-)=0)}\Bigg\} \tag{3.6}$$

and, if T_k is in the n part, then :

$$E[g_k(t^+) \mid G_n(t^-)=0] = g_{on,k} \times \Bigg\{ \frac{P_{x_k,hh}(t)}{P_{x_k,h}(t^-)} +$$

$$\left[P_{x_k,lh}(t) \;-\; P_{x_k,l}(t^-)P_{x_k,h}(t^+)\right] \times \frac{P(G_n(t^-)=0 \mid x_k(t^-)=0)}{P(G_n(t^-)=0)P(x_k(t^-)\neq 0)}\Bigg\} \tag{3.7}$$

Therefore, it takes an additional graph reduction for every gate input to compute the values $P(G_p(t^-)=0 \mid x_k(t^-)\neq 0)$ and $P(G_n(t^-)=0 \mid x_k(t^-)=0)$.

The derivation of $E[G(t^+) \mid G(t^-)=0]$ outlined above makes the implicit assumption that when the probability space is restricted by the condition $G(t^-)=0$ the independence of the gate inputs is preserved. This may not always be true, and the implementation in CREST has a protection measure to safeguard against this consisting of a simple upper bound [12] on $E[G(t^+) \mid G(t^-)=0]$.

Having found $E[I]$ for the output node, the expected value of charge delivered to (or from) the output node capacitors is easily found as follows :

$$E[q] = V_{dd} \times C_n \times P_{o,lh}(t) + V_{dd} \times C_p \times P_{o,hl}(t) \tag{3.8}$$

where o is the output node. We now make the approximation that the time constant for charging or discharging the output node is the largest of the internal gate nodes. Consequently, the time span of the output node current represents the time span τ of the total gate current. By the triangular pulse approximation :

$$\tau = 2 \times \frac{E[q]}{E[I]} \tag{3.9}$$

Next, the expected value of the charge delivered by the total gate charging current, $E[q_{tot}]$ is derived using the capacitances at each internal node j as follows :

$$E[q_{tot}] \approx \sum_{j \in p \;block} V_{dd} C_{jn} P_{j,lh} + \sum_{j \in n \;block} V_{dd} C_{jp} P_{j,hl} \tag{3.10}$$

Strictly speaking the probabilities $P_{j,lh}$ and $P_{j,hl}$ are hard to find; they are in fact $\mathcal{NP}-hard$ to find in general, based on [13]. We have therefore opted to use an upper bound of these probabilities to replace them in the equation. An upper bound of $P_{j,lh}$ $(P_{j,hl})$, for a node in

the p (n) block, is the probability that the conduction state between j and V_{dd} (V_{ss}) goes from off-to-on. This is found by a graph reduction that repeats the work done to find $P_{o,lh}$ for the output node o for every internal node j. Finally, the peak total current is found as :

$$E[I_{tot}] = \frac{2E[q_{tot}]}{\tau} = \frac{E[q_{tot}]}{E[q]} \times E[I] \tag{3.11}$$

The relationship between these quantities is depicted in Fig. 5.

Having derived the expected gate current pulse, the time of the new event at the output of this gate needs to be derived, as follows. If one considers a resistor R charging a capacitor C from a power supply V, then the current and voltage at the capacitor both reach their half-point at time $.693 \times RC$. If we assume that the individual current pulses $i(t)$ are exponentially decaying, rather than linear, then the switching time at the output, t_s, is $0.693 \times (C_p + C_n)$ / G_p for a low-to-high transition, and $0.693 \times (C_p + C_n)$ / G_n for a high-to-low transition. This t_s is, again, a random variable, and one is interested in $E[t_s \mid V_o \text{ transitions}]$. Knowing that the duration of i_p (or i_n) determines the gate delay, and that these currents deliver charge to *both* C_p and C_n, then if q_p and q_n are the charges delivered, we have :

$$E[q_p] = V_{dd} \times (C_p + C_n) \times P_{o,lh} \text{ , and} \tag{3.12}$$

$$E[q_n] = V_{dd} \times (C_p + C_n) \times P_{o,hl} \tag{3.13}$$

The duration of the two pulses is derived as before, as :

$$\tau_p = 2 \times \frac{E[q_p]}{E[I_p]} \text{ , and } \tau_n = 2 \times \frac{E[q_n]}{E[I_n]} \tag{3.14}$$

Having found these values, the time delay can be shown [12] to be :

$$E[t_s \mid V_o \text{ transitions}] = 0.35 \times \frac{\tau_p \times P_{o,lh} + \tau_n \times P_{o,hl}}{P_{o,lh} + P_{o,hl}} \tag{3.15}$$

It is important to note that τ_p and τ_n are independent of the particular partitioning of $(C_p + C_n)$, which makes the timing estimate reliable.

4 Pass-Transistor Circuits

Pass-transistors present a problem because they act as memory elements. The output of a pass-transistor network depends on more than just its inputs, the previous values at its

output and internal nodes are also important. It is also no longer true that the probabilities of the path(s) from the output to V_{dd} or V_{ss} completely define the output event because the output may be disconnected from both. Current drawn through pass-transistors is a relatively small fraction of the overall current dissipated in a CMOS circuit and will be ignored. However, events must be propagated correctly through them so that the gates downstream are correctly simulated.

To reduce their complexity, we break up pass-transistor networks into two types of primitive gates : *stages* and *wires*, as shown in Fig. 6. A concrete example of this decomposition is given in Fig. 7. Stages represent conducting paths that connect two nodes; while wires are used to tie together the outputs of two or more stages. These gates can be used to build complicated pass transistor networks. This decomposition involves two simplifying assumptions : transistors are unidirectional, and memory states are significant only at the boundaries of stages and wires. The direction of each gate must be carefully assigned, and we use several levels of algorithms, including some rules from [14].

As in traditional logic simulation, the two logic values 0 and 1 are not enough to model pass-transistor circuits, and ways of representing *weak* 0 and 1 signals must be introduced. Nodes in a pass-transistor section of a circuit can have four valid states : conducting path to V_{dd} (high tied or ht), conducting path to V_{ss} (low tied or lt), charged with no conducting paths (high floating or hf), and discharged with no conducting paths (low floating or lf). An additional state is introduced : no path to either V_{dd} or V_{ss} (floating or f). Although there is redundant information, five probabilities are used to fully define the states of all nodes in a pass transistor structure : P_{ht}, P_{lt}, P_{hf}, P_{lf}, and P_f.

Correspondingly, the set of probabilities needed to describe an event is also enlarged, these events will be referred to as *expanded events*. Expanded events must contain the probability that the node was high tied (P_{ht}), low tied (P_{lt}), and high (P_h) before and after the transition. Additionally, transition probabilities from low to high (P_{lh}), low tied to high tied ($P_{lt\rightarrow ht}$), low tied to floating ($P_{lt\rightarrow f}$), floating to high tied ($P_{f\rightarrow ht}$), and floating to floating ($P_{f\rightarrow f}$) are needed to propagate probability waveforms through the pass-transistors.

4.1 Stage simulation

The simulation of a stage involves a graph reduction by which the stage is reduced to a single edge ξ between its input and output. This procedure (see the appendix) provides the probabilities that this edge *is on*, $P_{\xi,1}(t^+)$, *was on*, $P_{\xi,1}(t^-)$, and transitions from *off to on*, $P_{\xi,01}(t)$. Other probabilities, such as *on to off*, $P_{\xi,10}(t)$, *on to on*, $P_{\xi,11}(t)$, and *off to off*, $P_{\xi,00}(t)$, can be derived from simple probability theory. If the stage has input x and output y then the following formulas can be derived [12] :

$$P_{y,ht}(t^{\pm}) = P_{\xi,1}(t^{\pm}) \times P_{x,ht}(t^{\pm}) \tag{4.1}$$

$$P_{y,lt}(t^{\pm}) = P_{\xi,1}(t^{\pm}) \times P_{x,lt}(t^{\pm}) \tag{4.2}$$

$$P_{y,lt\to ht}(t) = P_{\xi,11}(t) \times P_{x,lt\to ht}(t) \tag{4.3}$$

$$P_{y,f\to ht}(t) = P_{\xi,01}(t) \times P_{x,ht}(t^+) + P_{\xi,11}(t) \times P_{x,f\to ht}(t) \tag{4.4}$$

$$P_{y,lt\to f}(t) = P_{\xi,10}(t) \times P_{x,lt}(t^-) + P_{\xi,11}(t) \times P_{x,lt\to f}(t) \tag{4.5}$$

$$\begin{aligned} P_{y,f\to f}(t) = {} & P_{\xi,00}(t) + P_{\xi,01}(t) \times P_{x,f}(t^+) \\ & + P_{\xi,10}(t) \times P_{x,f}(t^-) + P_{\xi,11}(t) \times P_{x,f\to f}(t) \end{aligned} \tag{4.6}$$

Notice that these formulas, while determining most of the stage output probabilities, do not finalize its simulation because the probabilities of the high floating or low floating states are not yet available. This is to be expected because the state of the floating output of a stage can be affected by other stages tied to the same wire gate. The simulation of a wire, to be discussed next, uses the quantities derived above for the stage(s) outputs to finalize their simulation and obtain the wire output probabilities.

4.2 Wire simulation

If a wire has more than two inputs, one can think of it as a cascade of several wires with two inputs each. Consequently, it is sufficient to study the simulation of a wire with only two inputs x and y, and an output z. Since a wire is basically a wired-or configuration, voltage division may arise when its inputs are $x = lt$ and $y = ht$ (or vice versa). In this case we make the assumption that the output is $z = lt$; this is based on the "weak pull-up to V_{dd}" configuration which is commonly used in CMOS and nMOS circuits. This

assumption is implicit in the derivation of the following wire simulation formulas whose (tedious) derivation [12] will not be included here :

$$P_{z,ht} = P_{x,ht} \times P_{y,f} + P_{y,ht} \times P_{x,f} + P_{x,ht} \times P_{y,ht} \tag{4.7}$$

$$P_{z,lt} = P_{x,lt} + P_{y,lt} - P_{x,lt} \times P_{y,lt} \tag{4.8}$$

$$\begin{aligned} P_{z,lt\to ht}(t) = {} & P_{x,ht}(t^+) \times [P_{y,lt\to ht}(t) + P_{y,lt\to f}(t)] \\ & + P_{y,ht}(t^+) \times [P_{x,lt\to ht}(t) + P_{x,lt\to f}(t)] \\ & + P_{x,f}(t^+) \times P_{y,lt\to ht}(t) + P_{y,f}(t^+) \times P_{x,lt\to ht}(t) \\ & - P_{x,lt\to ht}(t) \times P_{y,lt\to f}(t) - P_{y,lt\to ht}(t) \times P_{x,lt\to f}(t) \\ & - P_{x,lt\to ht}(t) \times P_{y,lt\to ht}(t) \end{aligned} \tag{4.9}$$

$$P_{z,f\to ht} = P_{x,f\to ht} \times P_{y,f\to f} + P_{y,f\to ht} \times P_{x,f\to f} + P_{x,f\to ht} \times P_{y,f\to ht} \tag{4.10}$$

$$\begin{aligned} P_{z,lt\to f}(t) = {} & P_{x,f}(t^+) \times P_{y,lt\to f}(t) + P_{y,f}(t^+) \times P_{x,lt\to f}(t) \\ & - P_{x,lt\to f}(t) \times P_{y,lt\to f}(t) \end{aligned} \tag{4.11}$$

$$P_{z,f\to f} = P_{x,f\to f} \times P_{y,f\to f} \tag{4.12}$$

When all the inputs of a wire have been considered, its simulation is finalized using the following two formulas [12] :

$$P_{z,lf}(t^+) \approx \begin{cases} P_{z,lt\to f}(t), & \text{if } P_{z,f}(t^-) = 0; \\ \frac{P_{z,lf}(t^-)}{P_{z,f}(t^-)} \times P_{z,f\to f}(t) + P_{z,lt\to f}(t), & \text{otherwise.} \end{cases} \tag{4.13}$$

$$P_{z,lf\to ht}(t) \approx \begin{cases} 0, & \text{if } P_{z,f}(t^-) = 0; \\ \frac{P_{z,lf}(t^-)}{P_{z,f}(t^-)} \times P_{z,f\to ht}(t), & \text{otherwise.} \end{cases} \tag{4.14}$$

A minor independence assumption must be made in the derivation of these formulas; this is necessary because of the limited sense in which temporal dependence is accounted for (as mentioned in sub-section 2.3 above).

Having derived the probabilities of the wire's output event, the only remaining unknown is the *time* of that event. The approach used by Horowitz [15] is suited to our probabilistic models. By using the expected conductance of a stage derived in the graph reduction step, and the internal capacitances of stages, a time constant can be derived for the pass-transistor network. This is used to derive the time of the output event.

5 Supergates

CREST assumes that the primary circuit inputs are *independent*. The single-gate current estimation algorithms described above require that the probability waveforms, at the inputs of each gate, be independent. This condition is violated if the circuit contains reconvergent fanout or feedback. To overcome this problem, we borrow the concept of *supergates* from [10]; a supergate is simply a subset of the circuit with independent inputs, an example is shown in Fig. 8. Supergate input nodes that are reconvergent fanout stems, or that affect internal reconvergent fanout stems of the supergate, are called *reconvergent fanout input nodes*, abbreviated *rfi* nodes. Nodes A and B in Fig. 8 are rfi nodes of that supergate.

If logic (not probability) waveforms are assigned to rfi nodes, then the signals at all internal supergate nodes become independent and the supergate can be easily simulated. Given the probabilistic events at the rfi inputs of a supergate, suppose the set of all possible logical transitions embodied by these events is generated (Fig. 9), and the supergate simulated for each of them. If the current results are summed up, weighted by the probability of each case, the required supergate currents would result.

Based on these observations, the simulation of a supergate in CREST is carried out by maintaining a set of different *simulation sub-processes*, each representing the result of a particular sequence of *logical* events at the supergate's rfi nodes. Every sub-process has a certain probability, namely the probability that the supergate has that state at that time. When a new event arrives at an rfi node it is applied to all existing sub-processes, some of which will cause the creation of new sub-processes. When two sub-processes are performing identical simulations, they are *merged* to produce a single new sub-process whose probability is the sum of their probabilities.

This approach, while acceptable for small supergates, is too expensive for larger ones. In these cases, CREST uses two heuristic parameters to reduce the performance penalties while maintaining acceptable accuracy :

(1) α_s : Limit supergate size to a user-specified *size threshold*, α_s. By limiting supergate size, this indirectly limits the number of rfi nodes and, therefore, the number of simulation sub-processes.

(2) α_p : Terminate a simulation sub-process if its probability becomes less than a user-specified *probability threshold*, α_p. This eliminates sub-processes that are likely to contribute very little to the current waveform.

The effectiveness of these heuristics will be discussed in the next section.

The formation of supergates is done as a preprocessing step, during the initial partitioning phase of the program. The details of that process are tedious and not interesting enough to warrant inclusion in this paper. Very briefly, if the circuit is combinational, then the process involves a forward and a backward sweep to discover the node dependencies and build the supergates. In a circuit with feedback, it's easy to see that every feedback loop is, strictly speaking, a single supergate. Therefore a standard strongly-connected-components algorithm is first used to group feedback loops into temporary supergates, after which the circuit appears combinational, and supergates can be easily built. Since large feedback loops can lead to huge supergates, we have used heuristics to break up large feedback blocks into smaller ones to maintain a reasonable execution time.

6 Implementation and Results

As mentioned above, the probabilistic simulation approach has been implemented in a program called CREST (CuRrent ESTimation program). The program is about 15000 lines, written in C, and has been run on a variety of circuits and computer systems. It accepts a SPICE circuit description file, and requires another file that specifies the probability waveforms at the primary circuit inputs. Excellent accuracy and speed have been achieved on real circuits.

To assess the accuracy of the results, it is important to make a fair comparison with an *expected current waveform* derived using a valid simulation tool. To do so, we have generated the expected current waveform for a variety of examples by running SPICE on every set of input voltage signals allowed by the probability vectors, weighting each resulting current waveform by the probability that the inputs producing it would occur, and summing the weighted waveforms to produce the required result. Since the number of required SPICE simulation runs grows exponentially with the number of circuit inputs, the comparisons to be presented below will necessarily be limited to medium sized circuits. There is no reason

to suspect, however, that the accuracy observed on these circuits will deteriorate on larger ones.

Figures 10 and 11 show the results for a single complex gate and an inverter chain respectively. Important features of these two figures are the accuracy of the current waveform in Fig. 10 and the good timing performance in Fig. 11. Fig. 12 shows the current pulse for a typical pass-transistor circuit. In Fig. 13 we show the result for a larger circuit - a 4-bit ALU with 154 MOSFETS involving a large number of pass-transistor gates and weak pull-up transistors. The ALU was run for logical (rather than probabilistic) inputs because it would take too long to run SPICE for all its possible inputs to allow a comparison with a probabilistic CREST run.

CREST simulation times for these and other circuits are compared with SPICE in Table 1. SPICE was chosen for these comparisons because it is a generally available tool, and as such provides a common frame of reference. Since SPICE is known to be slow on large circuits, we also offer absolute measures of timing in the form of CPU seconds. The table illustrates the dramatic gains available from CREST's probabilistic analysis when an expected waveform is needed. In fact, as the number of circuit inputs increases, the speedup of CREST compared to an approach based on logical inputs increases exponentially since the size of the inputs space increases as 2^{2n}. For example, a (heuristic) CREST run on the 1839-transistor 34-bit ALU, which takes 13 seconds on a CONVEX 220, is equivalent to 2^{144} SPICE runs. The gains possible for logical inputs are illustrated in the last two examples, where logical waveforms were used since the circuits were too large to examine for all inputs in SPICE.

We finally present the results to demonstrate the effectiveness of the supergate simulation heuristics. Three circuits were used for these simulations : a 2-bit adder (Fig. 14), a 4-bit adder (Fig. 15), and a 4-bit multiplier (Fig. 16). The waveforms are compared in the figures indicated and the timing and speedup results of the different runs are shown in Table 2. The exact CREST simulation shown in Fig. 14a was 3000 times faster than SPICE. Fig. 14b shows a heuristic CREST run that is over four times faster with comparable current results. Fig. 15 shows the results of three different heuristic runs in CREST compared to a full-accuracy run. The first run merely eliminated extremely improbable supergate processes and obtained 11X speedup with virtually no accuracy loss. The second run combined both

Table 1. Execution time comparison between CREST and SPICE. Time is in CPU seconds on a SEQUENT. Size refers to the number of transistors.

Circuit	Size	CREST	SPICE	Speedup
Decode1	20	1.1	642	584X
Decode2	28	1.4	989	706X
Decode3	36	1.9	1588	836X
Invt10	20	1.0	221	221X
Invt40	80	3.3	2094	635X
Bridge	10	0.8	23237	29000X
Pass1	8	0.8	282	352X
Pass2	9	1.1	1505	1370X
4-bit ALU	154	10.9	3535	324X
34-bit ALU	1839	145.0	86152	594X

heuristics and achieved excellent accuracy with 31X speedup. The final run completely eliminated supergates and shows acceptable accuracy with a 59X speedup. Fig. 16 compares a fairly tight heuristic run and a relaxed heuristic run for a 4-bit parallel multiplier. In the tight run, supergates are allowed to grow up to nine gates and supergate simulation processes are ignored only if their probability is less than .0005. In the relaxed run, the supergate size limit is set to one gate, ie "no supergates." The relaxed heuristic ran 8 times faster with comparable results. In general, the heuristics yielded comparable results with excellent improvements in speed. This was particularly true for the most complex circuit, the 4-bit multiplier.

In all examples tested, the results were excellent. Peak currents were within 20%, average currents were within 10%, and, as clearly shown in Fig. 11, timing estimates were within 10% of SPICE.

7 Summary and Conclusions

We have discussed the electromigration problem and stressed the need for an *expected current waveform* for MTF estimation. We have presented such a technique, based on a new so-called *probabilistic simulation* approach which has been implemented in the program

Table 2. Performance of the supergate heuristics. Time is in CPU seconds on a VAX-11/780. Size refers to the number of transistors.

Circuit	Size	α_s	α_p	Time
2-bit ripple adder	54	∞	0	11.4
		1	0	2.6 (4.4X)
4-bit ripple adder	200	∞	0	332.2
		∞	.01	29.0 (11.4X)
		3	.1	10.6 (31.3X)
		1	0	5.6 (59.3X)
4-bit multiplier	648	9	.0005	2403.1
		1	0	297.5 (8.1X)

CREST. The combined effects of a variety of input patterns, each of which would require separate SPICE simulations, can be derived with a single simulation at no more expense than the simulation for a single vector. As such, our approach is pattern-independent and provides a dramatic speed-up (ranging from 200X to 29000X) over the conventional approach using SPICE. Furthermore the speedup increases exponentially with the number of circuit inputs because a single probabilistic run covers an exponentially increasing number of logical runs. The waveforms produced agree well with results obtained from SPICE : peak currents are within 20%, average currents are within 10%, and timing estimates are within 10%.

CREST can handle general CMOS circuits, allowing pass transistors, reconvergent fanout and feedback. Heuristics have been presented that help maintain speed in the presence of reconvergent fanout, without significantly affecting accuracy. The results of several CREST runs on real circuits have been presented.

The success of the heuristics leads to the conclusion that reconvergent fanout paths and the corresponding signal dependence become less important for larger circuits (eg. Fig. 16). This is supported by the intuitive observation that the effect of a reconvergent fanout node becomes negligible if the reconvergent paths contain a large number of gates.

The expected current waveform derived by CREST has an obvious immediate use in an unrelated application : the derivation of the expected instantaneous power dissipation, as well as the overall power dissipation of the circuit. Apart from this obvious application,

future work on CREST will address the simulation of circuits with truly bidirectional pass-transistors. Other extensions of the approach will also be aimed at estimating the voltage drop on the power and ground busses. This may require deriving more statistics of the current waveform, such as the waveform variance.

Appendix : Graph Reduction

This appendix describes a graph reduction procedure that is central to the simulation algorithms to be presented. Let $\mathcal{G} = (V, E)$ be an undirected connected graph with no self loops, possibly with parallel edges, in which two arbitrary vertices $u, v \in V$ are specified. At time t, each edge $e \in E$ is labeled with three probability values $P_{e,1}(t^+)$, which is the probability that the edge *is on*, $P_{e,1}(t^-)$, the probability that the edge *was on*, and $P_{e,01}(t)$, the probability that it transitions *from off to on*. Furthermore, each edge has associated with it a random *conductance* value $g(e)$ which is either 0, if the edge is off, or $g_{on}(e)$, if it is on. The *equivalent conductance* of a network of edges is defined to be that of an identical resistive network with the same conductances. The events "is on," "was on", and "transitions from off to on", as well as the random conductances, of any two distinct edges e_1 and e_2 are assumed to be independent.

Let ψ be a random variable that is 1 if there exists at least one path of on-edges in $\mathcal{G}$ between u and v, and 0 otherwise. The probabilities $P_{\psi,1}(t^+)$, $P_{\psi,1}(t^-)$, and $P_{\psi,01}(t)$ are defined as above. Finally, let $G_{\mathcal{G}}$ be the random conductance between u and v at t^+. We are interested in finding the values of $P_{\psi,1}(t^+)$, $P_{\psi,1}(t^-)$, $P_{\psi,01}(t)$, and $E[G_{\mathcal{G}}]$.

Finding $P_{\psi,1}(t^+)$, which is perhaps the simplest to derive, is known to be $\mathcal{NP} - hard$, see [16] page 211. The algorithms to be given will, therefore, include some approximations to maintain a reasonable execution time. The general methodology will be to reduce the size of the graph by removing edges and/or vertices, so that the probabilities of ψ and $E[G_{\mathcal{G}}]$ remain invariant, until the graph reduces to a single edge ξ between u and v, as shown in Fig. 3. The labels on this edge, $P_{\xi,1}(t^+)$, $P_{\xi,1}(t^-)$, $P_{\xi,01}(t)$, and $E[g(\xi)]$ are the required answers. Hence the term "graph reduction."

We will first describe solutions for a restricted class of graphs, namely for series-parallel graphs (see [17], pp. 197–199). For such a graph, one can accurately derive the probabilities

of ψ by making parallel and series combinations of edges until a single edge remains. If e_p (e_s) is the parallel (series) combination of edges e_1 and e_2, then the following formulas apply :

$$P_{e_p,1}(t^{\pm}) = P_{e_1,1}(t^{\pm}) + P_{e_2,1}(t^{\pm}) - P_{e_1,1}(t^{\pm})P_{e_2,1}(t^{\pm}) \tag{A.1}$$

$$P_{e_s,1}(t^{\pm}) = P_{e_1,1}(t^{\pm})P_{e_2,1}(t^{\pm}) \tag{A.2}$$

$$P_{e_p,01}(t) = P_{e_1,01}(t)P_{e_2,0}(t^{-}) + P_{e_2,01}(t)P_{e_1,0}(t^{-}) - P_{e_1,01}(t)P_{e_2,01}(t) \tag{A.3}$$

$$P_{e_s,01}(t) = P_{e_1,01}(t)P_{e_2,1}(t^{+}) + P_{e_2,01}(t)P_{e_1,1}(t^{+}) - P_{e_1,01}(t)P_{e_2,01}(t) \tag{A.4}$$

The derivation of $E[G_{\mathcal{G}}]$ is not as simple. To begin with, $E[g(e)]$ is easily derived for every edge e as $g_{on}(e) \times P_{e,1}$. We then perform the series-parallel combinations using these values. In case of two edges in parallel, it is easy to see that :

$$E[g(e_p)] = E[g(e_1)] + E[g(e_2)] \tag{A.5}$$

In the series case, however, the problem is not as simple and we resort to the approximation :

$$E\left[\frac{1}{g(e)} \;\middle|\; g(e) \neq 0\right] \approx \frac{1}{E[g(e) \mid g(e) \neq 0]} \tag{A.6}$$

where $E[A \mid B]$ denotes the *conditional expected value* of A given B, to derive [12] :

$$\frac{1}{E[g(e_s)]} \approx \frac{1}{E[g(e_1)]P_{e_2,1}} + \frac{1}{E[g(e_2)]P_{e_1,1}} \tag{A.7}$$

In case of a general, non series-parallel, graph, and since the problem is $\mathcal{NP} - hard$ as pointed out above, we resort to another technique which is in line with the graph reduction methodology. This involves a *node elimination* technique which is the graph-domain operation corresponding to Gaussian Elimination on the graph's adjacency matrix [18]. It works by removing a vertex from the graph and adding new edges between each of its neighbors, as shown in Fig. 17. Suppose a vertex v has neighbors $v_1, \ldots, v_n$ along edges $e_1, \ldots, e_n$. Upon removing v, every two neighbors v_i and v_j share a new edge e_{ij} (Fig. 17) whose probabilities are derived by applying the series formulas (A.2) and (A.4) above to e_i and e_j. Using the approximation (A.6), the value of $E[g(e_{ij})]$ is derived [12] as :

$$\frac{1}{E[g(e_{ij})]} \approx \frac{1}{E[g(e_i)]P_{e_j,1}} + \frac{1}{E[g(e_j)]P_{e_i,1}} + \frac{\sum_{\substack{k=1 \\ k \neq i,j}}^{n} E[g(e_k)]}{E[g(e_i)]E[g(e_j)]} \tag{A.8}$$

As a result, certain edges are *split* into two or more new edges which may not, therefore, be independent. We however ignore this fact and proceed with the graph reduction as before. Knowing that the problem is $\mathcal{NP} - hard$, the resulting loss of accuracy is inevitable, but excellent results have been obtained in practice.

References

[1] F. M. d'Heurle, "Electromigration and failure in electronics : an introduction," *Proceedings of the IEEE*, vol. 59, no. 10, October 1971.

[2] J. R. Black, "Electromigration failure modes in aluminum metallization for semiconductor devices," *Proceedings of the IEEE*, vol. 57, no. 9, September 1969.

[3] J. E. Hall, D. E. Hocevar, P. Yang, and M. J. McGraw, "SPIDER - a CAD system for modeling VLSI metallization patterns," *IEEE Transactions on Computer-Aided Design*, vol. CAD-6, pp. 1023–1031, Nov. 1987.

[4] L. W. Nagel, "SPICE2 : a computer program to simulate semiconductor circuits," Ph.D. dissertation, Dept. of Electrical Engineering, University of California, Berkeley, 1975.

[5] J. W. McPherson and P. B. Ghate, "A methodology for the calculation of continuous dc electromigration equivalents from transient current waveforms," *The Electrochemical Society, Proc. Symp. on Electromigration of Metals*, New Orleans, LA, pp. 64–74, Oct. 7–12, 1984.

[6] R. Burch, F. Najm, P. Yang, and D. Hocevar, "Pattern-independent current estimation for reliability analysis of CMOS circuits", *ACM/IEEE* 25^{th} *Design Automation Conference*, pp. 294–299, June 12–15, 1988.

[7] F. Najm, R. Burch, P. Yang, and I. Hajj, "CREST - a current estimator for CMOS circuits," *IEEE International Conference on Computer-Aided Design*, Santa Clara, CA, pp. 204–207, November 7–10, 1988.

[8] A. Papoulis, *Probability, Random Variables, and Stochastic Processes*. New York, NY: McGraw-Hill Book Co., 1984.

[9] K. P. Parker and E. J. McCluskey, "Probabilistic treatment of general combinational networks," *IEEE Transactions on Computers*, pp. 668–670, June 1975.

[10] S. C. Seth, L. Pan, and V. D. Agrawal, "PREDICT – probabilistic estimation of digital circuit testability," *IEEE* 15^{th} *Annual International Symposium on Fault–Tolerant Computing*, Ann Arbor, MI, pp. 220–225, June 1985.

[11] H. J. M. Veendrick, "Short-circuit dissipation of static CMOS circuitry and its impact on the design of buffer circuits," *IEEE Journal of Solid-State Circuits*, vol. SC-19, no. 4, August 1984.

[12] F. Najm, "Probabilistic simulation for reliability analysis of VLSI circuits," Ph.D. thesis, Dept. of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, June 1989.

[13] F. Najm and I. Hajj, "The complexity of test generation at the transistor level," Report # UILU-ENG-87-2280, Coordinated Science Lab., University of Illinois at Urbana-Champaign, December 1987.

[14] N. P. Jouppi, "Derivation of Signal Flow Direction in MOS VLSI," *IEEE Transaction on Computer-Aided Design*, vol. CAD-6, pp. 480–490, May 1987.

[15] M. Horowitz, "Timing Models for MOS Pass Networks," *Proceedings of the IEEE International Symposium on Circuits and Systems*, 1983.

[16] M. R. Garey and D. S. Johnson, *Computers and Intractability, A Guide to the Theory of NP-Completeness*. New York, NY: W. H. Freeman and Co., 1979.

[17] F. Harary, "Combinatorial problems in graphical enumeration," in *Applied Combinatorial Mathematics*, E. F. Beckenbach, Editor. New York, NY: John Wiley and Sons, Inc., 1984.

[18] S. Parter, "The use of linear graphs in gauss elimination," *SIAM Review*, vol. 3, no. 2, pp. 119–130, April 1961.

Figure Captions :

Figure 1: Four actual current waveforms (dashed) and the corresponding expected current waveform (solid).

Figure 2: A probability waveform (bottom) represents four logical waveforms (top).

Figure 3: A typical graph reduction.

Figure 4: A generic CMOS gate.

Figure 5: Relationship between gate output current and gate total current pulses.

Figure 6: Stages and wires.

Figure 7: The decomposition of a typical pass-transistor circuit into stages and wires.

Figure 8: A supergate schematic showing two *rfi* nodes A and B.

Figure 9: The decomposition of a single event into logical transitions.

Figure 10: Complex gate current estimate.

Figure 11: A 40-stage inverter chain current estimate.

Figure 12: Current results for a typical pass-transistor circuit.

Figure 13: Current results for a 4-bit ALU circuit with 154 MOSFETS.

Figure 14: Current waveforms for the 2-bit ripple adder in Table 2.

Figure 15: Current waveforms for different heuristic CREST runs compared to the full-accuracy CREST run on the 4-bit ripple adder in Table 2.

Figure 16: Current waveforms for a fairly tight heuristic run and a relaxed heuristic run for the 4-bit multiplier in Table 2.

Figure 17: A generic node elimination step.