Pattern Independent Maximum Current Estimation in Power and Ground Buses of CMOS VLSI Circuits: Algorithms, Signal Correlations and Their Resolution

Harish Kriplani,
AT&T Bell Laboratories
600 Mountain Avenue
Murray Hill, NJ 07974

Farid Najm and *Ibrahim Hajj*
University of Illinois at Urbana-Champaign
1308 W. Main Street
Urbana, IL 61801.

Abstract

Currents flowing in the power and ground (P&G) buses of CMOS digital circuits affect both circuit reliability and performance by causing excessive voltage drops. Excessive voltage drops manifest themselves as glitches on the P&G buses and cause erroneous logic signals and degradation in switching speeds. Maximum current estimates are needed at every contact point in the buses to study the severity of the voltage drop problems and to redesign the supply lines accordingly. These currents, however, depend on the specific input patterns that are applied to the circuit. Since it is prohibitively expensive to enumerate all possible input patterns, this problem has, for a long time, remained largely unsolved. In this paper, we propose a pattern-independent, linear time algorithm (`iMax`) that estimates at every contact point, an *upper bound* envelope of all possible current waveforms that result by the application of different input patterns to the circuit. The algorithm is extremely efficient and produces good results for most circuits as is demonstrated by experimental results on several benchmark circuits. The accuracy of the algorithm can be further improved by resolving the signal correlations that exist inside a circuit. We also present a novel *partial input enumeration* (`PIE`) technique to resolve signal correlations and significantly improve the upper bounds for circuits where the bounds produced by `iMax` are not tight. We establish with extensive experimental results that these algorithms represent a good time-accuracy trade-off and are applicable to VLSI circuits.

1 Introduction

A major concern in present day VLSI circuits is the design of power and ground (P&G) buses in a way that ensures design reliability and performance. Excessive currents can severely affect both circuit reliability and performance by causing excessive voltage drops in the P&G buses. Excessive voltage drops manifest themselves as *glitches* on the buses and cause erroneous logic signals (soft errors) and degradation in switching speeds. Severity of the voltage drop problems intensify with the continuing push for denser chips and finer technologies. As is known from the classical scaling theory [1], as the minimum feature size and supply voltage are scaled down, while the total power dissipation on the chip remains constant, the currents flowing in the P&G buses increase. With higher currents flowing in narrower buses, the voltage drops in the P&G buses go up and become a limiting factor in the design of VLSI chips. Furthermore, a lower supply voltage means that the noise margins [1] for the correct operation of the transistors on chip decrease. In short, in order to avoid logic errors, the circuit needs to be appropriately designed to take care of increased voltage drops and reduced noise margins. This highlights the need for efficient CAD tools to estimate voltage drops in the buses. Since worst case currents determine worst case voltage drops, our research is focused on the problem of estimating *maximum currents* in the P&G buses.

Power and ground buses deliver power to all the gates in a circuit. Points at which individual gates or cells are tied to the buses are called *contact points*. In VLSI circuits, P&G buses take up an appreciable amount of routing area, typically 20-50% or even more in some circuits. Several design methods, such as [2, 3], have appeared in literature that make use of the maximum current estimates at the contact points to redesign the buses. The output of a design optimization procedure, however, depends upon the accuracy with which maximum currents are estimated. A poor estimate of maximum currents will result in a pessimistic design and wasted silicon area. Clearly, an accurate estimation of currents at every contact point in a circuit is very crucial and is the subject of this paper.

Current drawn by a CMOS circuit depends upon the specific input pattern applied at its inputs. An *input pattern* for a circuit with n inputs is defined as a vector of n excitations, where each excitation could be any one of four possibilities : l (*low*), h (*high*), hl (*high* to *low*) or lh (*low* to *high*). For different input patterns, different transient current waveforms are drawn at the contact points. Therefore, in the presence of such input dependent and transient current waveforms, we need to define what we mean by the maximum current waveform at a contact point. Chowdhury et. al. [4] find the maximum of the peaks of various transient current waveforms at every contact point for all possible input patterns. They then use these constant peak values at the contact points to redesign the supply lines. This assumption, however, gives pessimistic results since separate sections in a circuit rarely draw their maximum currents simultaneously. In this paper, we propose a better measure of the maximum current waveform at a contact point called *maximum envelope current* (`MEC`) waveform. This maximum current estimate is discussed in section 4.

Accurate estimation of the maximum current waveform at every contact point is extremely difficult since for that we need to determine current waveforms corresponding to all possible input patterns. If a circuit has n primary inputs then we need to simulate it for 4^n input patterns, since each input can be l, h, hl or lh. This makes the problem practically impossible to handle by any of the known search procedures for large circuits. As will be shown in the next section, most previous work in this area has been based on search techniques. In this paper, we propose a pattern independent, linear time (in the number of gates) algorithm (`iMax`) that provides tight upper bounds on the `MEC` waveforms. The proposed approach represents a trade-off between execution speed and tightness of these bounds.

In order to maintain reasonable execution times, the `iMax` algorithm neglects various signal correlations that exist inside a circuit. As will be shown later, while in most cases `iMax` produces good upper bound waveforms, in some cases the loss due to signal correlations can be significant. We then propose a new *partial input enumeration* (`PIE`) algorithm that efficiently resolves these correlations and leads to significant improvement in the upper bound waveforms. The `PIE` algorithm is based on (1) intelligently selecting a few *critical* inputs and (2) enumerating a limited number of cases at these inputs to produce an overall improvement in the upper bound waveforms at the contact points. It turns out that the choice of these critical inputs is the key to improving the upper bounds. We present two heuristics for automatically selecting the critical inputs, that have shown good results in practice. While the `PIE` algorithm is slower than the simple `iMax` algorithm, we demonstrate good speed and accuracy performance results on circuits with over twenty five thousand gates. Furthermore, the algorithm has the attractive property that it does an *iterative improvement*, so that one can stop the algorithm at any time and obtain better upper bounds than the simple `iMax` results.

This paper is organized as follows. In the next section, we briefly discuss the previous and related work in this area. We then discuss various assumptions that our algorithms are based on. In section 4, we describe the proposed maximum current estimate. After that, we present the `iMax` algorithm in detail in section 5. Experimental results on several benchmark circuits using `iMax` are also provided in this section. The signal correlation problem is described in section 6. This is followed by a discussion of possible methods that can be used to resolve the signal correlations in section 7. In section 8, we present the partial input enumeration algorithm along with extensive experimental results on several benchmark circuits. Finally, in section 9, conclusions and some guidelines for future work are presented.

2 Previous Work

Several papers (such as [5, 6, 7, 8]) have appeared in literature on the estimation of P&G currents from deterministic input patterns. These methods offer significant improvement in execution times compared to SPICE, while providing acceptable accuracy in the current waveforms. These methods can be used for finding maximum currents for small circuits having a few inputs, by calculating the current waveforms corresponding to all possible input patterns. However they are not much helpful for large circuits, as they do not guide us in selecting an input pattern that leads to the maximum currents.

Chowdhury et. al. have addressed the problem of maximum current estimation in [4]. In their methodology, they divide the circuit into a set of macros, where each macro consists of a combinational interconnection of logic gates. Considering each macro separately, they use either an exact search technique (namely, branch and bound) or a heuristic technique to find the maximum of its transient currents, assuming that the macro has only one cantact point and its inputs switch simultaneously. In the analysis of the bus, to calculate maximum voltage drops,

of this assumption, their methodology overestimates the worst case currents and voltage drops. Secondly, due to the huge size of the input space, their branch and bound search technique is slow on large circuits. Furthermore, their heuristic approach does not guarantee an upper bound on the maximum currents.

Devadas et. al. have addressed a similar problem in [9]. They consider the estimation of worst case power dissipation in CMOS combinational circuits. They reduce this problem to a weighted max-satisfiability problem on a set of multi-output Boolean functions. These functions are obtained from the logic description of the circuit. The functions are appropriately weighted to account for different load capacitances. They then use either a disjoint cover enumeration algorithm or the branch and bound algorithm to solve the ($\mathcal{NP}$-complete) max-satisfiability problem. However, for a multilevel logic circuit, even under a unit gate delay assumption, the functions generated by their algorithm are fairly complex. Consequently, even for small circuits, their analysis is slow. Analysis of multi-level circuits under a general delay model was not attempted.

From this brief survey, it is clear that existing methods for the calculation of maximum current are computationally too expensive to handle large VLSI circuits. For these circuits, near linear algorithms rather than exponential, are necessary. Therefore, pattern independent algorithms become a natural choice. Hercules [10] was an initial attempt in the direction of a pattern independent approach to maximum current estimation. However, the analysis presented in [10] makes several simplifying assumptions. The approach subdivides the circuit into stages but does not discuss how information is represented at the output of each stage and how it is propagated from one stage to others. Further, the signal correlation problem is not discussed in the paper. In this paper, we present a novel approach that is able to address these problems. This approach is discussed in section 5.

3 Assumptions

In order to reduce the complexity of the problem, we focus on a specific, but very common design style, namely (edge-triggered) latch-controlled synchronous digital circuits. These circuits consist of combinational blocks separated by latches (see Fig. 1) such that all the inputs to each block switch simultaneously. As a result, we will focus the analysis, from the next section, on a

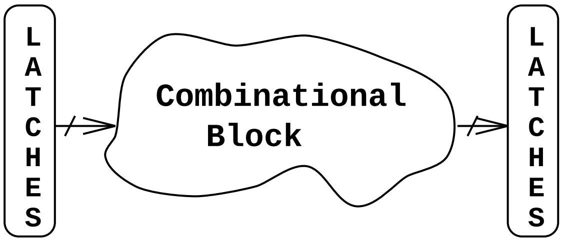


Figure 1: A latch controlled synchronous digital circuit.

single combinational block all of whose inputs switch simultaneously (if at all). This effectively eliminates the time domain uncertainty about the input transitions, and significantly simplifies the problem. This assumption has also been used by all the previous approaches.

We assume that the delay of each gate in the circuit is fixed and is calculated ahead of time. Different gates can have different delays. Further, we assume that for every transition at the output of a gate, the current waveform drawn from the power or ground bus, called the *transition current waveform*, is represented by a triangular waveform, as shown in Fig. 2. The value of the delay and various parameters of the transition current waveform, such as its duration, peak value and the time point at which the peak occurs, are calculated in a preprocessing phase from the circuit level parameters of the gate under consideration as well as of other gates that are connected to its inputs and output. This work, as well as the extension of the proposed algorithms under more general delay and current models is the subject of another paper and the interested reader is referred to [11, 12]. In this paper, due to space limitations, we focus on the algorithmic aspect of the maximum current estimation process under these simplified gate models.

Given the specific clocking scheme of the synchronous circuit, the maximum current waveforms from different combinational blocks are appropriately shifted in time depending upon the

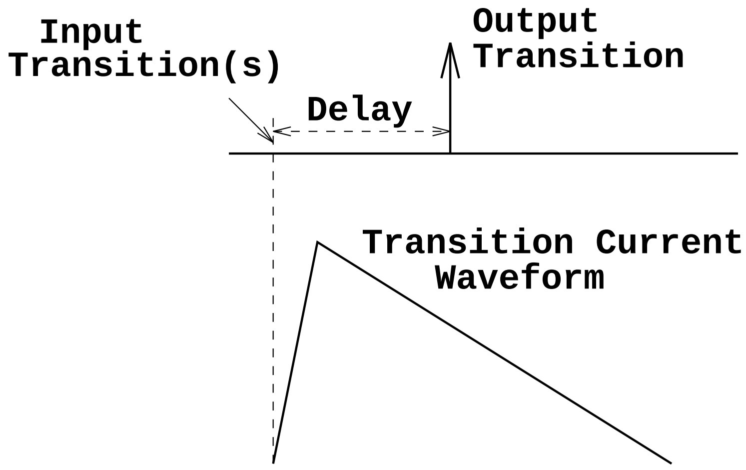


Figure 2: Gate delay and current models.

individual clock trigger, and are used to find the maximum voltage drops in the buses. Therefore, for the purposes of this paper, we will focus on the analysis of a single combinational block whose inputs switch at time zero.

4 Maximum Current Estimate

We define *excitation* at a node (or net) at any time t as the stimulus (or signal value) present at the node at that time. At any time, a node in the circuit could be either stable at *low* or *high*, or could transition from *high* to *low* or from *low* to *high*. Thus, the excitation could be any single value from the set $\mathbf{X} = \{l,\, h,\, hl,\, lh\}$.

We now describe the measure we use in our approach to represent maximum currents. For the purposes of this illustration, let us consider a specific contact point in a circuit. As mentioned in the introduction, the current drawn by a CMOS circuit is a complex function of input excitations. For each input pattern that is applied to the circuit, a different current waveform results at the contact point. Instead of representing the maximum current at the contact point by a single dc value, in our approach, we represent it by a waveform whose value at any time is the maximum current value that the circuit can draw at that time (see Fig. 3). We call this the *Maximum Envelope Current* (`MEC`) waveform.

Let us suppose that a circuit under consideration has n inputs and when an input pattern $p = (e_1, e_2, \ldots, e_n)$, where $e_i \in \mathbf{X}$, $1 \leq i \leq n$, is applied it, a transient current waveform $I_p(t)$ is drawn at the contact point. Let us denote the set of all possible input patterns that can be applied to the circuit by $U = \{(e_1, e_2, \ldots, e_n) \mid e_i \in \mathbf{X},\ 1 \leq i \leq n\}$. If the value of the `MEC`

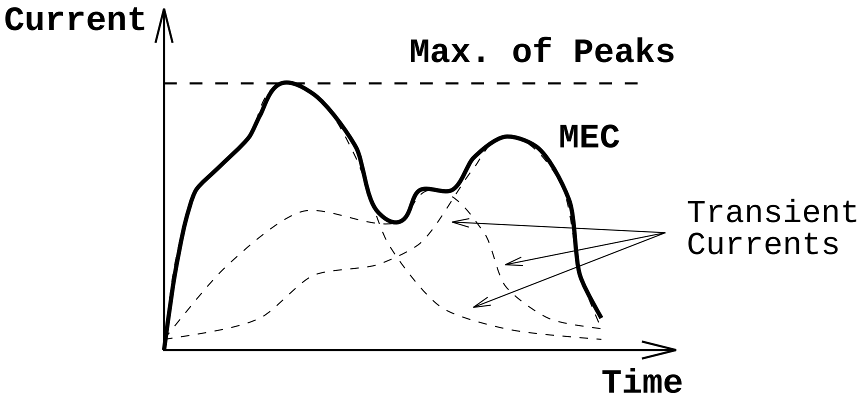


Figure 3: The Maximum Envelope Current (MEC) waveform.

waveform at any time t at the contact point is denoted by $I_{\mathtt{MEC}}(t)$, then we have the following equation:

$$I_{\mathtt{MEC}}(t) \;=\; \max_{p \in U} I_p(t) \tag{1}$$

Clearly, (also see Fig. 3) the MEC waveform is the maximum envelope of all transient current waveforms corresponding to all possible input patterns (and hence the name). There is a unique MEC waveform at every contact point.

If the power or the ground bus of a circuit is represented by an equivalent RC network, then we have the following result:

The voltage drop $\left(V_k^{\mathtt{MEC}}(t)\right)$*, occurring at any node k in the power or ground bus when the* MEC *waveforms are applied at the contact points, is an upper bound on the voltage drop* $(V_k^p(t))$ *occurring at that node when any input pattern p is applied to the circuit, i.e.,* $V_k^{\mathtt{MEC}}(t) \;\geq\; V_k^p(t)$*, for all t.*

The above result follows directly from Theorem 1 of the appendix and Eq. (1) ($I_{\mathtt{MEC}}(t) \;\geq\; I_p(t)$).

Estimating MEC waveforms at all the contact points in a circuit is an extremely difficult problem as for that we need the current waveforms corresponding to all possible input patterns. In the next section, we describe a linear time algorithm that provides tight upper bounds on the MEC waveforms at the contact points.

5 The iMax Algorithm

The proposed pattern independent, linear time algorithm operates at the gate level description of the circuit. Unless specified by the user, it assumes that nothing is known about the specific excitations at the primary inputs, except that they may transition (only) at time zero, i.e., each primary input may carry any excitation from the set $\mathbf{X}$ at time zero. We call this an *uncertainty* about these input signals. The basic idea of the proposed algorithm is to propagate this uncertainty present at the inputs inside the circuit, so that, at the output of every logic gate, we know the set of all possible excitations and their associated timing. From this, the worst case gate currents are computed, as explained below.

5.1 Signal Representation

Perhaps the first question that comes to mind is what kind of information one maintains in order to represent the signal uncertainty about internal circuit nodes. Ideally, one would like to compute the set of *all* possible transitions (along with their timing information) that occur at the output of every gate in the circuit. However, as will become clear soon, due to the uncertainty at the primary inputs and the general gate delay model used, the number of possible transitions at internal nodes grows exponentially, and quickly becomes a bottleneck. To avoid this problem, we maintain information, not about individual transitions, but about *intervals* during which the outputs of the gates *might* switch. Thus, at each node, for each of the excitations (l, h, hl, lh), we maintain a list of intervals during which the node might carry those excitations. These intervals, which might overlap, serve to describe the signal uncertainty. We call these intervals *uncertainty intervals*.

Definition 1 (Uncertainty Set $X_n(t)$) : The uncertainty set at time t for a node n defines the set of all possible excitations that the node can assume at that time. $X_n(t) \subseteq \mathbf{X}$.

Definition 2 (Uncertainty Waveform) : The uncertainty waveform describes the signal uncertainty present at a node as a function of time. At time t, the set of values taken by the waveform is the uncertainty set for the node at that time.

An example of the uncertainty waveform is given in Fig. 4. In this figure, we show an uncertainty waveform $U(t)$ represented as four sets of intervals[1] along the time axis. Thus, if $u(t)$ is a logic signal that belongs to the family $U(t)$, i.e., $u(t) \in U(t)$, then $u(t)$ will be *low* up to t_1, will switches from *low* to *high* sometime between t_1 and t_2, will then be *high* up to t_3, etc. Since the signal can switch from *low* to *high* at any time between t_1 and t_2, it can be either *high* or *low* during that interval. Notice that between t_6 and t_7 the signal may make *any* number of *low* to *high* and/or *high* to *low* transitions. At the primary inputs, signals are represented by such waveforms with a single point of *possible* transition at time 0. As internal signals are generated, the number of points at which transitions can possibly occur, increases. In order to contain the complexity, we then start to merge neighboring transition points into intervals. In general, this strategy can be stated as follows : *when the number of intervals*

[1] One set of intervals for each *low*, *high*, *hl* and *lh* excitations.

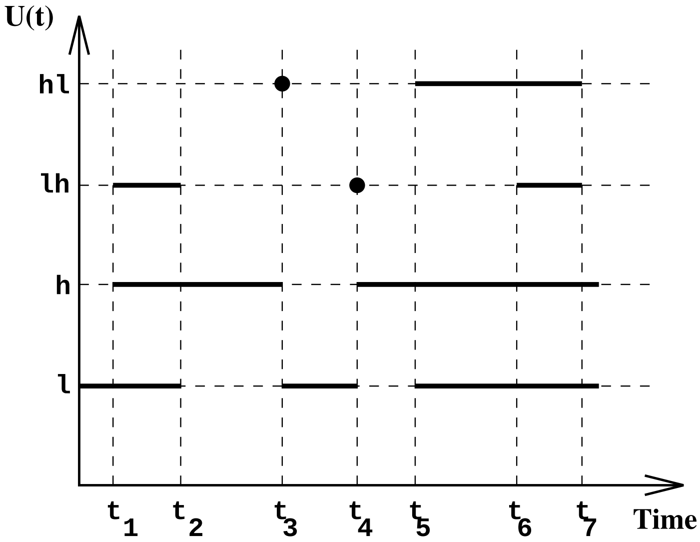


Figure 4: An illustration of uncertainty waveform.

associated with a gate corresponding to any excitation exceeds a certain user-specified threshold (`Max_No_Hops`*), we repeatedly merge closest-neighbor intervals, so as to keep their count below the threshold.*

5.2 Independence Assumption

While propagating information at a logic gate, we know the uncertainty waveforms at each of its inputs and we would like to derive the corresponding waveform at its output. However, one cannot do this accurately without knowing how some of these inputs, if any, are *correlated.* For instance, certain *combinations* of the gate input excitations may not be possible. Unfortunately, maintaining information about correlation between various circuit nodes is very expensive. We, therefore, use a *conservative* approximation, one that does *not* underestimate the `MEC` waveforms, as follows. If we assume that *all* combinations of the gate input excitations are possible, i.e., the gate inputs are *independent*, then the worst case current in that case *will be* an upper-bound on the gate current for the case when the inputs are dependent. In other words, the worst case current over *all* combinations of inputs is certainly an upper bound on the worst case current over *some*.

5.3 Single Gate Simulation

Given the type of a Boolean gate and the independence assumption for the uncertainty waveforms at its inputs, we now describe how the uncertainty waveform at the output of the gate

AND	l	h	hl	lh
l	l	l	l	l
h	l	h	hl	lh
hl	l	hl	hl	l
lh	l	lh	l	lh

OR	l	h	hl	lh
l	l	h	hl	lh
h	h	h	h	h
hl	hl	h	hl	h
lh	lh	h	h	lh

NOT	
l	h
h	l
hl	lh
lh	hl

Figure 5: The output of a gate for AND, OR and NOT functions.

is calculated. This process is divided into the following two parts:

1. Calculation of the uncertainty set at the output of the gate at a time t.

2. Calculation of uncertainty intervals at the output of the gate.

5.3.1 Calculating Uncertainty Set :

One can calculate all possible excitations at the output of the gate at time t from the uncertainty sets at its inputs at time $t - D$, where D is the delay of the gate. Let us denote the uncertainty set at the i^{th} input of the gate at time $t - D$ by X_i. Let us further suppose that the gate has m inputs. Then the set of all possible input patterns that lead to an excitation at the output of the gate at time t can be represented by $\{(x_1, x_2, \ldots x_m) | x_i \in X_i, \ \forall 1 \leq i \leq m\}$. For each input pattern, the output of the gate can be easily determined from the Boolean equation of the gate as explained below. For simple functions, such as AND, OR and NOT, it is easy to verify that the output of the gate is as shown in Fig. 5. In fact, the set (l, h, hl, lh) along with the above definitions for AND, OR and NOT constitutes a 4-values Boolean algebra [13]. The output of a gate realizing any arbitrary Boolean function can be easily calculated by repeated applications of the above.

By calculating the output of the gate for each and every input pattern, the resulting activity (or uncertainty set) at the output of the gate at time t can be determined. This process, however, requires one to generate and evaluate $|X_1||X_2| \ldots |X_m|$ input patterns. This worst case complexity can be greatly reduced by the following observations.

1. The above input pattern generation and evaluation process can be stopped when the uncertainty set at the output of the gate becomes equal to $\mathbf{X}$. Obviously, trying out any more input patterns would not lead to any further improvement in the uncertainty set.

2. If the uncertainty sets for all of the inputs at time $t - D$ are $\mathbf{X}$s then the uncertainty set at the output of the gate at time t is also $\mathbf{X}$. It is trivial to verify this fact for simple

NAND, NOR and INVERTER gates. Because of the functional completeness of NAND, NOR and INVERTER [14], any composite gate can be represented in terms of these simple gates. Therefore, the fact holds for any gate type.

Both of these observations lead to tremendous savings in the calculation of uncertainty sets at the output of the gate and thus contribute to the speed of the algorithm.

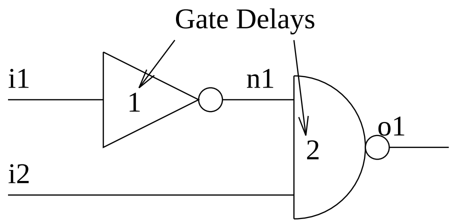
5.3.2 Calculating Uncertainty Intervals :

In `iMax`, since signals are represented in the form of uncertainty intervals at the inputs of a gate, the output of the gate would also be in the form of uncertainty intervals. An interval at the output of a gate could begin or end at time t only if an interval begins or ends at any of its inputs at time $t - D$. Between the times at inputs when an interval begins or ends, and the next interval begins or ends, the sets of excitations that the inputs can assume do not change and therefore no corresponding uncertainty interval can begin or end at the output during that time (shifted by D). Thus by calculating the uncertainty *sets* at the output of the gate at every time point at which an uncertainty interval begins or ends at any of its inputs, the uncertainty *intervals* at the output are calculated.

An example illustrating how uncertainty intervals at various circuit nodes are calculated is shown in Fig. 6. At the primary inputs, it is assumed that each of the inputs may carry any excitation from the set $\mathbf{X}$. Thus, each input may switch hl or lh at time 0, or stay at l or h for all time. Given this information at the input of the inverter and assuming its delay as 1 time unit, its output may switch lh or hl at time instant 1 or stay at l or h. Similarly, the NAND gate may switch at time 2 because of the second primary input, or at time 3 because of the output of the inverter, or stay at l or h for all time. In this fashion, uncertainty waveforms are propagated from one gate to another. From this example, we also notice that while each of the inputs to the NAND gate may switch only (at most) once, its output may switch at two time points. Thus, as the uncertainty waveforms are propagated through the NAND gate, the number of time points at which the gate can switch has doubled. This multiplicative growth of the number of intervals can potentially lead to memory bottlenecks for large circuits. In order to contain this growth, we have suggested merging neighboring intervals to form bigger intervals. Thus, if `MAX_NO_HOPS` parameter is set to 1, then we would merge intervals [2, 2] and [3, 3] to form interval [2, 3], as shown in the figure.

5.4 Current Calculation

After the uncertainty waveform at the output of a gate is known, its current contribution is calculated next. Since the output of the gate could switch at any time during an uncertainty interval, a transition current waveform could be drawn at any time during the interval (shifted



Input Description : i1, i2 $\in \{l, h, hl, lh\}$ at time 0.
Uncertainty Intervals :
i1, i2: lh[0, 0], hl[0, 0], l[0, ∞), h[0, ∞)
n1: lh[1, 1], hl[1, 1], l[0, ∞), h[0, ∞)
o1: lh[2, 2][3, 3], hl[2, 2][3, 3], l[0, ∞), h[0, ∞)
if `MAX_NO_HOPS` = 1 then
o1: lh[2, 3], hl[2, 3], l[0, ∞), h[0, ∞)
Key : Excitation[Interval Begin, Interval End]

Figure 6: An example of uncertainty waveforms calculation.

backwards by the delay of the gate) from the P&G buses, as shown in Fig.7. Hence, by taking an envelope of all possible transition current waveforms, we get the worst case current contribution of the gate due to the uncertainty interval. At every gate, there are two types of uncertainty intervals that result in some switching activity at the output and therefore, there are two possible current waveforms, one due to the hl uncertainty intervals, called `hlCurrent` and the other due to lh uncertainty intervals, called `lhCurrent`. Since at any time, the output of the gate could switch either from *high* to *low* or from *low* to *high*, by taking an envelope of the `hlCurrent` and `lhCurrent` waveforms, we get the maximum current contribution of the gate. Once all the gate currents are calculated, the current waveforms at the contact points

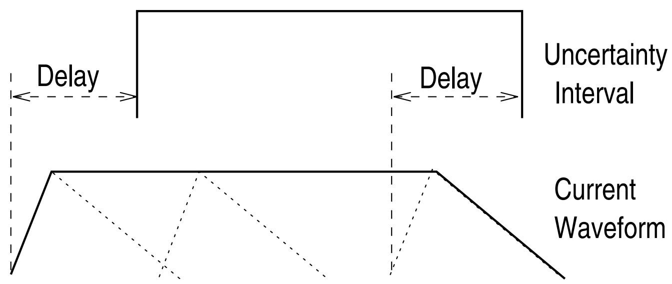


Figure 7: Current waveform due to an uncertainty interval.

are calculated by adding the individual currents (appropriately shifted in time) of those gates that are tied to it.

5.5 Calculation of Voltage Drops

It is shown in the appendix that when the equivalent network of the power or ground bus is represented by a resistive network, the vector of voltage drops appearing at its nodes (V) is related to the corresponding vector of contact point currents (I) as follows :

$$Y\,V \;=\; I \tag{2}$$

where Y is the *node admittance matrix* of the network [15]. When the bus is represented by an RC network, the following relationship holds

$$Y\,V \;=\; I - C\dot{V} \tag{3}$$

where C is the diagonal matrix of node capacitances. By calculating the LU factors of the Y matrix and quantizing time, the above equations are solved by forward and backward substitutions or by numerical integration [15]. Once the voltage drop waveforms at various nodes in the network ($V_i,\ 1 \le i \le n$) are known, the maximum voltage drop is calculated by finding the maximum of the voltage drops:

$$Max\_VD \;=\; \max_{1\le i\le n,\ t\ge 0} V_i(t) \tag{4}$$

5.6 Implementation Details

The above approach has been implemented in a program in C. In the program, the circuit is first *levelized* so that the output of a gate at level j does not feed any other gate at a level less than or equal to j. Any user-specified restrictions on certain inputs are then imposed, while all other inputs are assumed to take all possible excitations from the set $\mathbf{X}$. After this, the circuit is analyzed in a level by level fashion, starting from the lowest level, by propagating the uncertainty waveforms at the inputs of every gate to its output. From these uncertainty waveforms, we calculate the current waveforms at the contact points which are point-wise upper bounds on the corresponding `MEC` waveforms, i.e., if the current waveform calculated at any contact point by the (`iMax`) algorithm is denoted by $I_{\mathtt{iMax}}(t)$ and the corresponding `MEC` waveform by $I_{\mathtt{MEC}}(t)$, then

$$I_{\mathtt{iMax}}(t) \ge I_{\mathtt{MEC}}(t), \qquad forall\ t \ge 0. \tag{5}$$

From this equation and from Theorem 1 of the appendix, it follows that the maximum voltage drop calculated from the `iMax` algorithm ($Max\_VD_{\mathtt{iMax}}$) is an upper bound on the *worst case* *maximum voltage drop* (`WC_Max_VD`) in the bus, i.e.

$$Max\_VD_{\texttt{iMax}} \geq \max_{\forall p \in U} Max\_VD(p) = \texttt{WC\_Max\_VD} \tag{6}$$

where $Max\_VD(p)$ is the maximum voltage drop that occurs in the bus when an input pattern p is applied to the circuit.

An important property of the `iMax` algorithm is that each gate is considered exactly once in the entire analysis. Further, because of the interval merging feature, the memory space requirement per gate is fixed. Therefore, the algorithm is linear in time as well as space in the number of gates in the circuit.

5.7 Quality Assessment

In order to assess the quality of the solution obtained from the `iMax` algorithm, we need to determine how close the upper bound obtained is to the `WC_Max_VD`. One way of doing this would be to perform an exhaustive enumeration over all possible input patterns and actually calculate the `WC_Max_VD`. However, doing this is very expensive and practically impossible for circuits with more than about 10 inputs (Note: $4^{10} = 1,048,576$). Therefore, the following repeated enumeration approach is used for the verification of results. In the approach, different input patterns are repeatedly applied to the circuit. For each pattern, a logic simulator[2] is used to calculate the outputs of various gates. From these gate outputs, the current waveforms at the contact points are calculated, as in the case for `iMax`. Using these current waveforms, the maximum voltage drop in the bus is calculated, as described in Section 5.5. By repeating this process for a finite number of input patterns (say V), we obtain a lower bound on the `WC_Max_VD`, as seen from the following equation (for an input pattern p, the maximum voltage drop is denoted by $Max\_VD(p)$):

$$Max\_VD_{\texttt{iMax}} \geq \max_{p \in U} Max\_VD(p) \geq \max_{p \in V \subset U} Max\_VD(p) \tag{7}$$

Naturally, as more patterns are simulated the closer this lower bound approaches to the worst case value. In our experiments, for those cases where it is not possible to calculate the `WC_Max_VD`, we compare the `iMax` upper bound with this lower bound. The program that implements this repeated enumeration technique is called `iLogSim` (Current Logic Simulator).

The choice of input patterns for the above repeated enumeration process is very crucial to the goodness of the lower bound obtained. By a poor selection of input patterns, we may end up wasting cpu time without much improvement in the lower bound value. For the experimental

[2] One could use more accurate circuit simulators, such as SPICE instead, but these are extremely slow for our purposes.

Table 1: iMax and iLogSim results for 9 small circuits.

Circuit	No. Gates	No. Inputs	No. C. P.	iMax10	iLogSim	$\frac{\texttt{iMax10}}{\texttt{iLogSim}}$
BCD Decoder	18	4	1	0.05	0.05	1.00
Comparator A	52	11	2	0.27	0.27	1.00
Comparator B	54	11	5	0.44	0.44	1.00
Decoder	16	6	1	0.06	0.06	1.00
P. Decoder A	41	9	2	0.24	0.24	1.00
P. Decoder B	44	9	5	0.17	0.17	1.00
Full Adder	70	9	7	0.89	0.79	1.13
Parity	66	9	3	0.28	0.28	1.00
Alu (SN74181)	121	14	10	0.36	0.35	1.03

results, we have tried a combination of schemes such as random selection, simulated annealing and exhaustive enumeration on a reduced input space; and have reported the best lower bound obtained.

5.8 Experimental Results

In this section, we tabulate the results obtained from running the iMax and iLogSim algorithms on the power buses of several small and large circuits. Similar results can be obtained for the ground buses. For lack of real data, the bus network for each circuit was generated by randomly assigning each gate to a contact point and randomly generating links between the contact points. The network was, however, not restricted to a simple tree topology.

Table 1 lists the results of running iMax and iLogSim algorithms on nine small circuits. These circuits have number of gates ranging from 16 to 121 and number of inputs ranging from 4 to 14. The number of contact points for each circuit are also shown in the table. Under columns iMax10 and iLogSim, we report the bounds on maximum voltage drop (normalized values) obtained from the respective algorithms and their ratio is shown in the last column. The number (10) next to iMax in the table indicates the value of the Max_No_Hops parameter. For all of these circuits, the iLogSim results were obtained by trying out all possible input patterns i.e., the iLogSim results shown in the table are the exact WC_Max_VD values.[3] From the table we observe that the ratio of iMax to iLogSim results is close to one for all of the circuits. Thus, for these circuits, the iMax upper bound is very close to the WC_Max_VD.

[3] For the last circuit, the approach mentioned in the second half of this paper was used.

Table 2: `iMax` and `iLogSim` results for 10 ISCAS-85 circuits.

Circuit	No. Gates	No. Inputs	No. C. P.	Max. Vol. Drop			CPU Time	
				`iMax10`	`iLogSim`	$\frac{\texttt{iMax10}}{\texttt{iLogSim}}$	`iMax10`	`iLogSim`
c432	218	36	12	1.43	1.13	1.26	2.7s	1h 24m
c499	572	41	30	2.85	1.46	1.95	4.9s	53m 46m
c880	555	60	26	2.12	1.23	1.72	4.4s	43m 1s
c1355	636	41	28	4.64	3.49	1.33	7.2s	1h 5m
c1908	1105	33	55	3.13	2.56	1.22	13.0s	7h 32m
c2670	1799	233	81	3.75	2.71	1.38	12.8s	4h 6m
c3540	2482	50	105	5.39	2.13	2.53	20.6s	7h 42m
c5315	3552	178	167	5.67	3.32	1.71	27.8s	12h 48m
c6288	2672	32	126	8.65	6.21	1.39	46.2s	47h 32m
c7552	5066	207	247	11.52	7.49	1.54	48.4s	26h 36m

In Table 2, we report similar results for the ten ISCAS-85 benchmark circuits [16]. These circuits have number of gates ranging from 218 to 5066 and all the circuits have at least 32 inputs. In the table, in the last two columns, we document the cpu times needed by the `iMax` algorithm and the typical times needed for trying 10,000 input patterns by the `iLogSim` algorithm on a sun SPARCstation ELC. The actual `iLogSim` results were obtained after trying about 100,000 input patterns. We observe that for all the circuits, the linear time `iMax` algorithm took only a few seconds of cpu time compared to several hours of time needed by the `iLogSim` algorithm. Furthermore, for most of these circuits, the ratio of `iMax` upper bound to `iLogSim` lower bound is less than 1.72. There are two possible reasons for this mismatch. Firstly, it is quite possible that the `iLogSim` lower bound is not close to the `WC_Max_VD`. Since all the circuits have at least 32 inputs, the space of possible input patterns is huge, and the lower bound obtained after trying about 100,000 input patterns may not be very close to the `WC_Max_VD`. For circuits where the input space is not so huge (Table 1), we were able to obtain `WC_Max_VD` results and these are in good agreement with `iMax` results. The second possible source of mismatch is our conservative independence assumption for signals at various nodes. One can improve on this assumption by attempting to resolve the signal correlations, as discussed in the following sections.

We next discuss the effect of varying the `Max_No_Hops` parameter on the performance of `iMax`. Table 3 lists the `iMax` upper bound results for ISCAS-85 circuits for different values of `Max_No_Hops`. In parentheses, we also tabulate the cpu times (in sec.) needed by the algorithm. As the value of `Max_No_Hops` increases, the number of intervals being merged at every node

Table 3: `iMax` results vs. `Max_No_Hops` parameter.

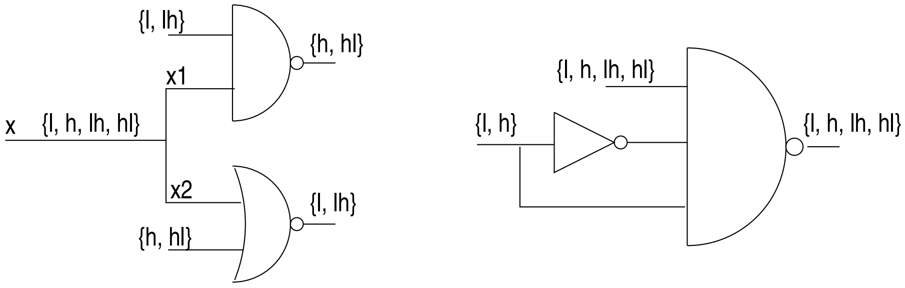
Circuit	iMax: Max_No_Hops			
	1	5	10	∞
c432	1.79 (1.1)	1.46 (2.0)	1.43 (2.7)	1.42 (12.6)
c499	2.94 (1.6)	2.92 (2.9)	2.85 (4.9)	2.85 (18.1)
c880	2.85 (1.4)	2.15 (2.8)	2.12 (4.4)	2.11 (37.0)
c1355	4.71 (1.9)	4.65 (3.8)	4.64 (7.2)	4.62 (144.1)
c1908	3.59 (5.5)	3.13 (8.8)	3.13 (13.0)	3.13 (290.7)
c2670	5.13 (4.9)	4.04 (8.9)	3.75 (12.8)	3.71 (95.7)
c3540	7.07 (7.9)	5.45 (14.1)	5.39 (20.6)	5.35 (1025.6)
c5315	7.21 (11.7)	5.76 (21.1)	5.67 (27.8)	5.66 (x503.9)
c6288	8.84 (15.1)	8.68 (28.1)	8.65 (46.2)	8.62 (x22512.2)
c7552	15.04 (21.8)	11.77 (36.1)	11.52 (48.4)	11.47 (x810.0)

decreases and therefore, the cpu time needed by the algorithm increases. This also improves the value of the upper bound, as shown in the table. However, with an increase in `Max_No_Hops` parameter, while the cpu time continues to increase, the improvement in the upper bound is not significant beyond `Max_No_Hops` = 10. A value between 5 and 10 seems to be a good choice for the parameter.

6 The Signal Correlation Problem

In general, signals at internal nodes of a circuit are *correlated* and this limits the number of transitions that can possibly occur at the outputs of the gates. Two examples of how signal correlation limits the number of transitions are illustrated in Fig. 8.

In Fig. 8(a), signal lines $x1$ and $x2$ are correlated, in this case, they carry the same signal. It is easy to verify that depending upon the specific excitation present at x, only one of the two gates can switch at a time. However, since `iMax` ignores the signal correlation between lines $x1$ and $x2$, it calculates the uncertainty sets at the outputs of the two gates as shown in the figure and thus erroneously concludes that both gates may switch at the same time. It, therefore, adds two transition current waveforms due to both gates switching simultaneously to the contact point current waveform(s). Similarly, in Fig. 8(b), the output of the inverter is correlated with its input and so the NAND gate may never switch. However, ignoring this correlation, `iMax` concludes that the NAND gate *can* switch. Thus, the `iMax` algorithm calculates more transitions than can actually occur in a circuit. It is these kinds of approximations that



(a) At most one of the two gates can switch.

(b) Correlated signals at the inputs of the NAND gate block the transition.

Figure 8: Two examples to illustrate the signal correlation problem.

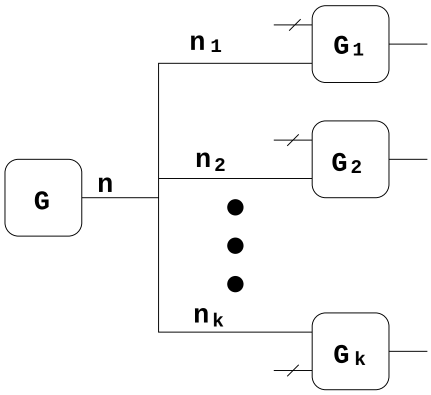


Figure 9: An example of a multiple fan-out gate.

contribute to a loose `iMax` upper bound.

As is clear from these examples, the source of the signal correlation problem, in general, is a node (the output of a gate or an input) which *fans out* to several other gates. Such nodes are called *multiple fan-out* (MFO) *nodes*. The general situation is shown in Fig. 9, where a MFO gate G with output node n fans out to nodes n_1, n_2, …, n_k that in turn feed gates G_1, G_2, …, G_k. In this figure, inputs to the gates G_1, G_2, …, G_k (which are n_1, n_2, …, n_k respectively) are correlated. Due to this correlation, even though the output of each gate (G_1, G_2, …, G_k) can assume all possible excitations as calculated by `iMax`, they may not *simultaneously* carry their worst case excitations. As one goes deeper into the circuit, where these correlated outputs reconverge and feed the same gate, the inputs of that gate become correlated (e.g., NAND gate in Fig. 8(b)). Such gates are called *reconvergent fan-out* (RFO) *gates.* With correlated signals at the inputs of a gate, the number of transitions that can possibly occur at its output is reduced. The signal correlations considered so far exist among various nodes throughout the circuit and are called *spatial correlations*.

Besides the spatial correlations, there is another set of correlations that the `iMax` algorithm completely ignores. The excitation(s) assumed by a node at time t restricts the set of possible excitations that the node can assume at an earlier or a later time. For example, if a node is low at time t, then it can either stay at low or switch from high to low at time t^- and it can either stay at low or switch from low to high at time t^+. These correlations which exist in the time domain are called *temporal correlations*.

The `iMax` algorithm completely ignores all spatial and temporal signal correlations and, therefore, overestimates the supply currents. The advantage of ignoring correlations in the algorithm is its, very desirable linear time performance.

7 Resolving Signal Correlations

The upper bound produced by the `iMax` algorithm can be made exact by doing a brute-force enumeration at the inputs of the circuit. In enumeration, since unambiguous input patterns are applied to the circuit, there is no *uncertainty* present at the inputs and therefore, signal correlations do not become an issue. In a similar fashion, one can improve the results of the `iMax` algorithm by doing a *partial enumeration* at a few selected nodes in the circuit.

An example of how partial enumeration helps improve the upper bound can be seen from Fig. 8(a). In this circuit with no enumeration, `iMax` would assume that the signal lines $x1$ and $x2$ are mutually independent and therefore infer that both NAND and NOR gates can switch at the same time. However, if we do partial enumeration at signal line x, then we would generate four cases corresponding to when $x = l$, $x = h$, $x = hl$ and $x = lh$. When $x = l$ or hl, only the NOR gate switches. Similarly, when $x = h$ or lh, only the NAND gate switches. Thus, by splitting the problem into four sub-problems, we have improved the result, i.e., found that only one of the two gates may switch at any given time.

While enumerating a node, we only need to process a small subset of gates that are present in its *fanout cone*. The fanout cone (`FOC`) of a node n is defined as the set of all gates that can possibly be affected by a change in excitation at the node. Thus, a gate is in the `FOC` of a node n if the gate is either directly fed by n or is connected to the output of a gate that is in `FOC`.

One technique to partially enumerate the internal nodes of a circuit, called *Multi-Cone Analysis* (`MCA`), was reported in [17]. The motivation behind such an approach was to be able to enumerate at the MFO nodes, which are the sources of the signal correlation problem. The approach involves partitioning the circuit in a fashion such that each gate belongs to the `FOC` of at most one MFO node and then enumerating a finite number of cases at these nodes. The

Table 4: Number of inputs and MFO nodes in ISCAS-85 circuits.

Circuit	No. Inputs	No. MFO	Circuit	No. Inputs	No. MFO
c432	36	89	c2670	233	454
c499	41	155	c3540	50	579
c880	60	125	c5315	178	806
c1355	41	259	c6288	32	1456
c1908	33	385	c7552	207	1300

details of approach are given in [17]. However, from the experimental results presented in [17], it is seen that the `MCA` approach offers only a modest improvement in the upper bound. There are several reasons for this.

As shown in Table 4, there are typically several MFO nodes in a circuit and all of these nodes should be enumerated to properly resolve the signal correlation problem. From our experience with ISCAS-85 benchmark circuits, we have found that the `FOC`s of several of these nodes overlap and therefore, to properly handle signal correlations, these nodes should be enumerated *simultaneously*. Furthermore, because of the presence of glitches in a circuit, signals at internal nodes span several time points (i.e., signal transitions occur at several time points). To take care of the temporal correlation problem, the nodes should be enumerated at each of these time points. Simultaneous enumeration is an extremely expensive process specially when there are several nodes and each node needs to be enumerated at several time points. As an example, to enumerate two nodes simultaneously, the cpu time needed is the product of the times needed to enumerate each node separately. To avoid this multiplicative growth of cpu time, several simplifying assumptions were made in the implementation of the `MCA` algorithm [17]. Because of these simplifications, the algorithm led to only mild improvement in `iMax` results.

From above, it is clear that that improving the `iMax` upper bound by enumerating internal nodes is very expensive and does not offer a practical solution for large circuits. In the next section, we present an alternative partial input enumeration approach that significantly improves the `iMax` results and represents a good speed-accuracy trade-off.

8 Partial Input Enumeration (`PIE`)

As shown in Table 4, there are usually many more MFO nodes than primary inputs in a circuit. Further, as discussed in section 3, all of the inputs to a circuit switch at most once at time zero. Therefore, there is only one time point at which a primary input needs to be enumerated. This is in contrast to an internal circuit node which usually needs to be enumerated at several time points. These observations, combined with the fact that `iMax` is an extremely fast algorithm, led us to explore the following *partial input enumeration* (`PIE`) algorithm to improve the `iMax` upper bound.

8.1 The `PIE` Algorithm

Let x_1, x_2, ..., x_N be the N primary inputs of a circuit under consideration. Let X_i represent the uncertainty set for input x_i at time zero. The *input search space* for the circuit consists of the set of all valid input patterns that can be applied to the circuit. Mathematically, the input search space is $\{(e_1, e_2, \ldots, e_N) \mid e_1 \in X_1,\ e_2 \in X_2, \ldots,\ e_N \in X_N\}$. For brevity, we denote this by $(X_1, X_2, \ldots, X_N)$. We assume, without loss of generality, that for a particular input x_i, $X_i = \mathbf{X}$. Then the input search space $(X_1, X_2, .., X_i, .., X_N)$ for the circuit can be divided into four disjoint parts, namely $(X_1, X_2, .., \{l\}, .., X_N)$, $(X_1, X_2, .., \{h\}, .., X_N)$, $(X_1, X_2, .., \{hl\}, .., X_N)$ and $(X_1, X_2, .., \{lh\}, .., X_N)$. We can compute the bounds on maximum voltage drop in the bus for each of these four parts by running the `iMax` algorithm, and in each case, restricting the excitation on input x_i to the value in its respective uncertainty subset. Since the four parts combined together constitute the complete search space, by taking the maximum of the four bounds on maximum voltage drop, we can still guarantee an upper bound on the worst case maximum voltage drop in the bus. In each of the four runs of `iMax`, specific excitations are present at input x_i, therefore, signal correlations due to x_i disappear and the resulting bound on maximum voltage drop should be an improvement on the original `iMax` upper bound. In a similar fashion, the upper bounds for the individual subcases can be improved.

The set of inputs selected for enumeration has a direct influence on the quality as well as the cost of the solution obtained. If all of the inputs are selected and enumerated, then the upper bound obtained would be exact. However, doing this is practically impossible for most circuits. The extent to which an input contributes to signal correlation inside a circuit is different for different inputs. For example, in Figure 8(a), enumerating input x is more beneficial than enumerating any of the other two inputs. Similarly, in Figure 8(b), enumerating input I is better than enumerating the other input. Hence, by selecting and enumerating inputs in an intelligent fashion, we can significantly improve the `iMax` upper bound, without spending too much cpu time.

We have developed an algorithm based on *best first search* (BFS) approach [18] that is very effective in selecting and enumerating inputs and thereby improving the upper bound on maximum voltage drop. Before describing the details of the algorithm, we note that both `PIE` and the previous approaches mentioned in section 2 are based on search techniques. However, unlike previous approaches which produce a meaningful result only after a long exploration of

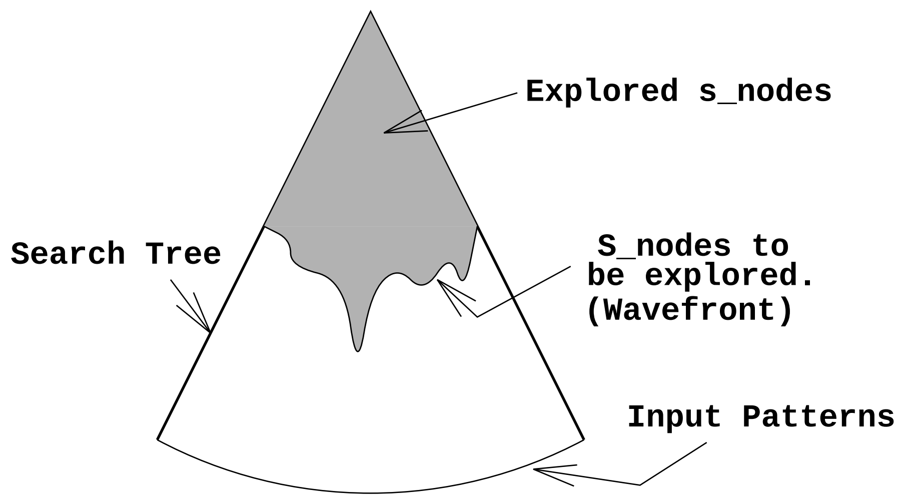


Figure 10: Exploring `s_nodes` in the search tree.

the entire input space, the `PIE` algorithm starts with resolving signal correlations due to those inputs first which contribute the most to the problem. Therefore, significant improvements in results are observed early in the search process.

The algorithm proceeds along a conceptual *search tree* in which each *node* corresponds to a partial assignment to the primary inputs, i.e., at each node, some inputs have specified excitations (e.g., say h), while others have uncertain (e.g., l, hl) excitations. We will refer to these nodes as "`s_nodes`," search nodes, to avoid confusing them with circuit nodes. The process of enumerating a primary input at a `s_node` translates, in the search tree domain, to the so-called *expansion* of the `s_node` into *children* `s_nodes`. After a `s_node` has been expanded, it is dropped from consideration and its children `s_nodes` are added to the list of `s_nodes` yet to be explored. This list of "yet to be explored" `s_nodes` is called a *wavefront*. The BFS algorithm always processes `s_nodes` which are on its current wavefront. At the start, the wavefront consists of only one `s_node`, namely the initial uncertain state $(X_1, X_2, \ldots, X_N)$. As the search progresses, this wavefront moves forward, as shown in Figure 10. At any time, the set of all the `s_nodes` on the wavefront constitutes the complete input search space for the circuit, i.e., if $W_1, W_2, \ldots, W_k$ are the `s_nodes` on the wavefront and W $= (X_1, X_2, \ldots, X_N)$ is the initial uncertain state, then

$$\begin{aligned} Input\ Search\ Space &= \{p \mid p \in W\} \\ &= \{p \mid p \in W_1 \ or \ p \in W_2, \ldots, \ or \ p \in W_k\} \end{aligned} \tag{8}$$

Thus, an input pattern leading to the `WC_Max_VD` *must* belong to one of the `s_nodes` on the wavefront.

An upper bound value is associated with every `s_node` generated during the search. The value of this bound for a `s_node` is the upper bound on maximum voltage drop obtained from the `iMax` algorithm for the corresponding input assignment. Further, two parameters, an *upper bound* (`UB`) and a *lower bound* (`LB`), are associated with the search. The value of `UB` at any stage during the search is the current best estimate on maximum voltage drop. It is the maximum value of the upper bounds of all the `s_nodes` on the wavefront. The second parameter, `LB`, keeps track of the maximum value of the upper bound corresponding to all of the input patterns[4] seen thus far. As the search progresses, the estimates on `LB` and `UB` improve. During the search, `s_nodes` which correspond to the maximum or *best* value of upper bound are repeatedly expanded. Because of this best first strategy, there is a gradual reduction in the upper bound on maximum voltage drop (`UB`). This *iterative improvement* is a very important feature of the algorithm for large circuits where an exhaustive exploration of the input space is practically impossible. The `PIE` algorithm can be stopped at any intermediate stage and the current best `UB` can still be reported.

The `PIE` algorithm starts with the initial uncertain state and a known `LB`, which is the bound on maximum voltage drop for some input pattern. During the search, a `s_node` with the best bound on maximum voltage drop is repeatedly selected and its descendent `s_nodes` are generated by enumerating an input, as explained in the outline shown in Figure 11. Here a leaf `s_node` is one which corresponds to an input pattern. Before explaining various functions used in this outline, an example to illustrate the algorithm follows.

In Figure 12, we show how the `PIE` algorithm progresses for a circuit with three inputs. Various `s_nodes` generated during the search are shown by ovals in the figure. Upper bounds on maximum voltage drop, as obtained from `iMax`, for the `s_nodes` are shown within the ovals. We assume that the uncertainty set for each of the inputs at time zero is **X**. Then the initial uncertain state for the circuit can be denoted by (**XXX**). If the upper bound on maximum voltage drop corresponding to this `s_node` is 50, then the value of `UB` at the start of the algorithm is 50. At this time, suppose we select the second input for enumeration. Then we would generate the four children `s_nodes`, as shown in the figure. With their associated upper bound values as shown, the value of `UB` improves from 50 to 47 by this enumeration. At this stage, one could either stop with 47 as the estimate on maximum voltage drop or continue with the enumeration process. To continue, we would select `s_node` (**X**hl**X**) for enumeration, as this `s_node` has the maximum associated upper bound value. Note that by improving the upper bound value associated with this node, the overall upper bound on maximum voltage drop (`UB`) can be improved. At this `s_node`, if we select the first input for enumeration, then we would generate four more `s_nodes`, as shown. With this enumeration, the value of `UB`

[4] An input pattern in this terminology is a `s_node` with specific excitations for *all* the inputs.

Remark: *Wavefront* is an ordered list of `s_nodes`, arranged in decreasing order of upper bound values.

1. *Wavefront* $\leftarrow$ Initial uncertain state (X_1, X_2, ..., X_N).
`UB` $\leftarrow$ upper bound value of the starting `s_node`.
`LB` $\leftarrow$ bound on maximum voltage drop value for a specific input pattern, (otherwise 0.0).

2. While `Stopping Criterion` is not satisfied, do

2.1. Remove the top `s_node` from the *wavefront*.

2.2. Calculate next input number to enumerate from the `Splitting Criterion`.

2.3. Generate all ($\leq$ 4) children `s_nodes` by enumerating the input and calculate their upper bound values.

2.4. If these children are leaf `s_nodes`, then update the `LB`, else, insert them in *wavefront*, after `pruning` if any.

2.5. `UB` $\leftarrow$ upper bound value of the top `s_node` in *wavefront*.

3. Report `UB` as the current best upper bound on maximum voltage
drop found so far. STOP.

Figure 11: Outline of the `PIE` algorithm.

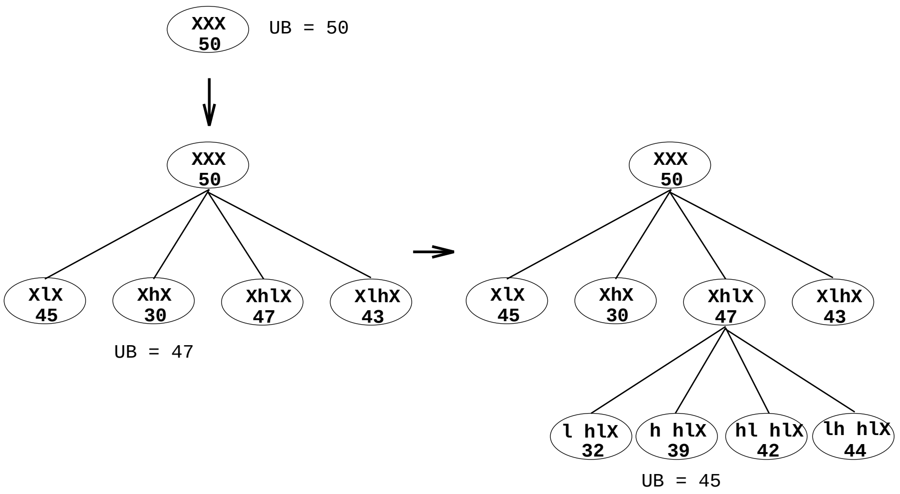


Figure 12: An example search tree for the `PIE` algorithm.

improves to 45. At this state, the current wavefront consists of the following `s_nodes`: (**X**l**X**), (**X**h**X**), (l hl**X**), (h hl**X**), (hl hl**X**), (lh hl**X**) and (**X**lh**X**). To continue further, we would select `s_node` (**X**l**X**) for enumeration and so on. We now explain the stopping, pruning and splitting criteria mentioned in the above outline.

8.2 Stopping criterion

We stop the search when any one of the following two conditions is satisfied.

(a) UB $\leq$ LB $\times$ ETF.

(b) Number of `s_nodes` generated $\geq$ User specified parameter (`Max_No_Nodes`).

The *Error Tolerance Factor* (`ETF`) is a user-specified parameter that provides control over the final desired accuracy of the algorithm. The value of this parameter is always bigger than 1. The first condition above specifies that when the `UB` value is within the `ETF` factor of some known `LB`, then the search can be terminated. In large circuits where calculating an exact solution by running the search to completion is extremely expensive, and an overestimation by 30% to 40% is often acceptable, such a parameter can be useful. The second condition puts a hard limit (`Max_No_Nodes`) on the number of `s_nodes` that are to be generated during the search.

8.3 Pruning criterion

During the search, if we come across a `s_node` for which the associated upper bound value satisfies the following condition:

$$\text{upper bound}(\texttt{s\_node}) \leq \texttt{LB} \times \texttt{ETF}$$

then such a `s_node` can be deleted from the search as its upper bound value is already acceptable. This pruning criterion deletes unnecessary `s_nodes` during the search and thus keeps the memory usage down.

8.4 Splitting Criterion and Experimental Results

The splitting criterion (`SC`) specifies the input which should be enumerated next from any `s_node` during the search. We now describe two heuristics for the `SC` that have shown good results in practice. The first heuristic selects an input which has the highest sensitivity while the second one selects an input based upon the (heuristically determined) *influence* it has inside the circuit.

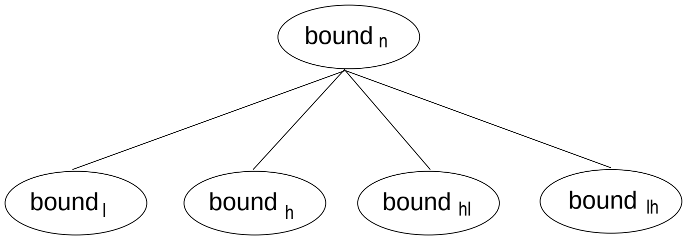


Figure 13: Enumerating a `s_node` during `PIE` algorithm.

8.4.1 H_1 heuristic

Let us suppose that during the search, we are at a particular `s_node` **n** and we select an input x_i for enumeration. Without loss of generality, we assume that the uncertainty set for x_i at time zero is **X**. Then by enumerating x_i, we would generate four children `s_nodes`, as shown in Figure 13. We assume that the upper bound associated with `s_node` **n** is denoted by $bound_{\mathbf{n}}$ and the upper bounds associated with the children `s_nodes` are denoted by $bound_l$, $bound_h$, $bound_{hl}$ and $bound_{lh}$ respectively. If we denote

$$\Delta bound_i \;=\; bound_{\mathbf{n}} - \max\{bound_l,\; bound_h,\; bound_{hl},\; bound_{lh}\}\,, \tag{9}$$

then by enumerating input x_i at `s_node` **n**, we can improve its associated bound by an amount $\Delta bound_i$. This calculation can be repeated for every input (one at a time while the uncertainty sets of other inputs are not perturbed) and an input that gives rise to the maximum improvement in the bound can be selected, i.e.,

$$Find\ k \quad such\ that \quad \Delta bound_k \geq \Delta bound_i,\;\; 1 \leq i \leq N,\;\; i \neq k. \tag{10}$$

However, if $\Delta bound_i$ is zero for all of the inputs, which occurs very often in practice, then the above selection process would not work well. For a specific input x_i, $\Delta bound_i = 0$ means that the bound associated with at least one of its children `s_nodes` is equal to $bound_{\mathbf{n}}$. However, for the remaining children `s_nodes`, the upper bound values may not be equal to $bound_{\mathbf{n}}$ and this information can be used in the selection of the best input. Based on these observations, we have come up with the following H_1 heuristic function for assigning value to an input x_i:

$$\begin{aligned} H_1(x_i) \;&=\; A \times (bound_{\mathbf{n}} - bound_1) \;+\; B \times (bound_{\mathbf{n}} - bound_2) \\ &\quad + C \times (bound_{\mathbf{n}} - bound_3) \;+\; (bound_{\mathbf{n}} - bound_4) \end{aligned}$$

where $bound_1$, $bound_2$, $bound_3$ and $bound_4$ are the bounds associated with the children `s_nodes`, generated by enumerating x_i and arranged in decreasing order. A, B and C are three constants

Table 5: Results of PIE (model m1) for 9 small circuits.

Circuit	Dynamic (H_1) Splitting Criterion			Static (H_1) Splitting Criterion		
	No. s_nodes Generated	iMax10 runs in SC	Time	No. s_nodes Generated	iMax10 runs in SC	Time
BCD Decoder	17	57	2.2s	17	17	1.1s
Comparator A	25	249	48.2s	25	45	13.6s
Comparator B	25	249	46.1s	25	45	12.3s
Decoder	25	109	3.7s	25	25	1.5s
P. Decoder A	21	177	12.6s	21	37	4.1s
P. Decoder B	21	177	13.9s	21	37	4.3s
Full Adder	37	249	53.3s	37	37	11.5s
Parity	21	177	22.9s	21	37	6.9s
Alu (SN74181)	57	477	209.9s	85	57	56.6s

such that $A \gg B \gg C \gg 1$. At any s_node during the search, we compute the heuristic values for all of the inputs and select an input with the maximum associated heuristic value. This splitting criterion is called *dynamic (H_1) splitting criterion* because at each s_node, it calculates the heuristic values for all the inputs and then selects the best input for enumeration.

The results of partial input enumeration using the PIE algorithm and using the dynamic (H_1) splitting criterion for the nine small circuits are documented in Table 5. For the table, the iMax algorithm with Max_No_Hops = 10 was used. For all the circuits, the search was run to completion[5], i.e., until UB became equal to LB (ETF = 1). The results clearly show that the PIE algorithm is very efficient in scanning the input space. As an example, the last circuit in the table (Alu) has 14 inputs; therefore, the number of possible input patterns for this circuit is 4^{14} = 268,435,456. The PIE algorithm was able to scan the *entire* search space after generating just 57 s_nodes. Since the iMax upper bound is used to guide the search, the table also indicates that the bound produced by the iMax algorithm is *very* tight for these circuits.

As can be seen from Table 5, the number of iMax runs needed in the dynamic splitting criterion far exceeds the number of s_nodes generated. At any s_node, to calculate the H_1 heuristic value for a particular input x_i, we need to run the iMax algorithm $|X_i|$ number of times, where $|X_i|$ is the number of elements in the uncertainty set. If the s_node has k inputs which are possible candidates for enumeration (i.e., their $|X_i| > 1$), then the iMax algorithm will be run $\sum_{i=1}^{k} |X_i|$ number of times to find the best input to enumerate next. For bigger

[5]This is how the WC_Max_VD results were obtained for circuits in Tables 1

circuits, with a large number of inputs, this time will be even more dominant rendering the PIE algorithm prohibitively expensive. Therefore, we have experimented with other less expensive alternatives.

Instead of calculating the (H_1) heuristic values for all the inputs at every s_node during the search, the heuristic value for every input is calculatedat the beginning of the search. All of the inputs are arranged in the decreasing order of their heuristic values. During the search, at every s_node, inputs are selected in this *fixed* order. This criterion is called the *static (H_1) splitting criterion*. The amount of time spent in the static splitting criterion is fixed and is equal to $\sum_{i=1}^{N} |X_i|$ runs of the iMax algorithm for a circuit with N inputs. The results of the PIE algorithm using the static (H_1) splitting criterion are also summarized in Table 5. With the static splitting criterion, the number of runs of the iMax algorithm required in the splitting criterion goes down, but the number of s_nodes generated during the search goes up for some circuits. However, for all of the circuits, an overall reduction in the cpu times required by the algorithm to finish is observed.

8.4.2 H_2 heuristic

The number of gates that are affected by a change in excitation at an input is a good heuristic measure of how much influence the input has on the upper bound current waveforms at the contact points and thus on the maximum voltage drop. Inputs which affect more gates (i.e., which have larger FOCs) should be enumerated before others. This leads us to another (static) splitting criterion H_2, in which the sizes of the FOC associated with all the inputs are calculated. As with H_1, all of the inputs are arranged in the decreasing order of H_2 values (i.e., FOC values) and during the search, at every s_node, inputs are selected in this fixed order. We will show that, while both static H_1 and H_2 give good results in practice, H_2 is much better in terms of speed and has accuracy comparable to H_1.

The results of the PIE algorithm using both H_1 and H_2 static splitting criteria for the ISCAS-85 benchmark circuits are shown in Table 6. In the tables, under various iMax and PIE columns, we show the ratio of the respective upper bound to the lower bound obtained from iLogSim. The numbers in parentheses under the PIE columns indicate the number of s_nodes that were generated before stopping the search (i.e., the Max_No_Nodes parameter; 1k stands for 1000). Total cpu times required by the algorithm on a SUN SPARCstation ELC (with Max_No_Nodes=100) are also shown in the tables. From the tables, we note that for most of the circuits, the PIE algorithm leads to some improvement in results, as is reflected by the ratio. This ratio can be further improved by running the PIE algorithm for longer durations. We emphasize that, since we can only compare the upper bound to a *lower bound*, the numbers in the table are only upper bounds on the error. It is prohibitively expensive to measure the

Table 6: Results of PIE for 10 ISCAS-85 circuits.

Circuit	iMax10		PIE : Static (H_1) SC			PIE : Static (H_2) SC		
	Ratio	Time	Ratio (100)	Ratio (1k)	Time (100)	Ratio (100)	Ratio (1k)	Time (100)
c432	1.26	2.7s	1.23	1.19	9m 48s	1.26	1.26	4m 32s
c499	1.95	4.9s	1.89	1.82	16m 40s	1.95	1.93	7m 6s
c880	1.72	4.4s	1.66	1.63	20m 22s	1.70	1.66	6m 38s
c1355	1.33	7.2s	1.33	1.33	25m 58s	1.33	1.33	11m 22s
c1908	1.22	13.0s	1.15	1.14	42m 26s	1.18	1.17	22m 2s
c2670	1.38	12.8s	1.36	1.36	3h 9m	1.38	1.38	20m 14s
c3540	2.53	20.6s	1.78	1.72	1h 25m	1.88	1.70	26m 25s
c5315	1.71	27.8s	1.58	1.57	5h 17m	1.70	1.69	43m 39s
c6288	1.39	46.2s	1.39	1.39	2h 44m	1.39	1.39	1h 25m
c7552	1.54	48.4s	1.47	1.44	10h 41m	1.47	1.46	1h 13s

true error.

While the improvement over the original `iMax` algorithm is not large in all the cases, in those cases where the `iMax` bound *was* loose, such as c3540, the new `PIE` algorithm with H_1 or H_2 heuristic leads to *significant* improvement: the ratio of 2.53 (maximum over-estimation by 1.53) is now 1.70 (maximum over-estimation by 0.70) with H_2, a reduction in the maximum over-estimation by about 54%.

We also emphasize the following attractive property of the algorithm : a significant amount of improvement in the upper bound occurs in the first few `s_nodes` (about 50-100) of the algorithm. This is shown in Figure 14 for c3540, where the ratio of the upper bound to lower bound is plotted as a function of cpu time for the first 1000 `s_nodes`. The figure also indicates that our heuristics are working well to select the most critical `s_nodes` first. Similar behavior is observed for most other circuits.

The cpu time needed for generating the input list by the H_2 splitting criterion is negligible compared to the time needed by the H_1 criterion. For VLSI circuits with several hundred inputs, where the time needed by the H_1 criterion may be large, the H_2 criterion may be used instead. As can be seen from Table 6 (also see Table 7), the results produced by using either splitting criterion are quite comparable, especially for those circuits where `iMax` did not produce a good upper bound.

In order to demonstrate the applicability of the `PIE` algorithm for large circuits with several thousand gates, we have also experimented with the ISCAS-89 benchmark circuits [19]. For

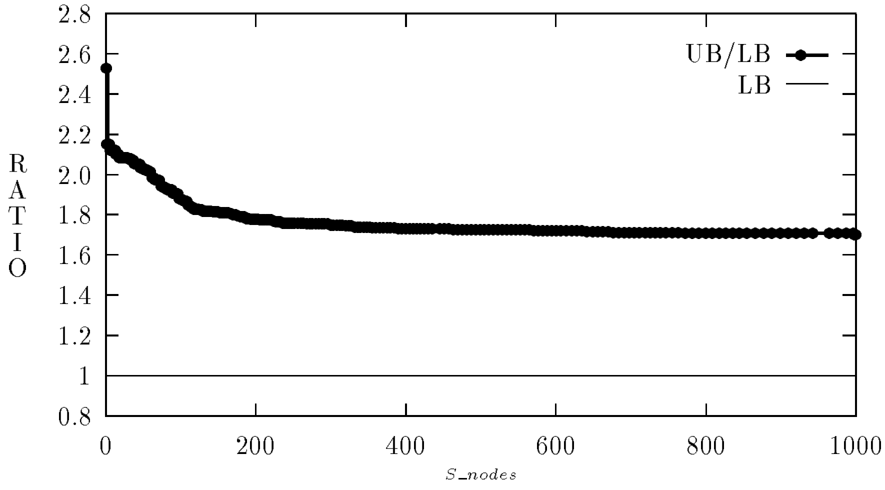


Figure 14: $\frac{Upper\ Bound}{Lower\ Bound}$ vs. Time plot for c3540.

these synchronous sequential circuits, we have extracted the combinational blocks by deleting the flip-flops. These combinational blocks have gate counts ranging up to 27,400 and number of inputs (primary inputs and D-flip-flops) ranging up to 1750. The results of the `PIE` algorithm on these circuits using both H_1 and H_2 splitting criteria are summarized in Table 7. Similar improvements in results, as for those of Table 6, are observed. It is clear from the table that even for circuits of this size, our algorithms show good speed and accuracy performance.

9 Conclusions and Future Work

In this paper, we have proposed a linear time algorithm (`iMax`) that computes maximum currents in the supply lines. Most of the previous algorithms on maximum current estimation suffer from exponential complexity and are not adequate for large circuits. Our approach avoids exponential complexity by adopting a pattern independent approach. The results produced by the algorithm are within acceptable bounds for most circuits. We have also presented a new *partial input enumeration* algorithm that partially resolves the signal correlations and further improves the upper bound obtained from `iMax`. The algorithm is based on the *best first search* (BFS) technique and represents a good time-accuracy trade-off. The `PIE` algorithm involves a search procedure, but this search need not be carried too deep to obtain good results. The algorithm is quite applicable to VLSI circuits, as is demonstrated by the experimental results on circuits with up to 27,400 gates. In our future research, we plan to extend the study to

Table 7: Results of PIE for ISCAS-89 circuits (combinational parts).

Circuit	No. Gates	iMax	Static H1 SC			Static H2 SC		
			PIE (100)	PIE (1k)	Time (100)	PIE (100)	PIE (1k)	Time (100)
s1423	991	1.35	1.32	1.29	37m 22s	1.35	1.34	7m 43 s
s1488	1203	2.21	1.40	1.08	5m 32s	1.41	1.06	2m 49s
s1494	1205	2.18	1.37	1.06	5m 35s	1.39	1.05	2m 51s
s5378	3018	1.38	1.29	1.25	2h 23m	1.30	1.23	13m 21s
s9234	6983	1.76	1.51	1.47	7h 24m	1.56	1.56	37m 18s
s13207	9577	1.37	-	-	-	1.30	1.26	36m 53s
s15850	12101	1.81	-	-	-	1.64	1.57	1h 11m
s35932	21249	1.66	-	-	-	1.56	1.56	2h 6m
s38417	26559	1.73	-	-	-	1.72	1.68	2h 46m
s38584	27390	1.45	-	-	-	1.39	1.37	2h 15m

include better gate delay and current models and to identify troublesome voltage drop sites in supply lines, using RC models, from the maximum current estimates.

Acknowledgements

Part of this research was performed while the first author was at Texas Instruments during the summers of 1991 and 1992. The authors are thankful to the technical staff members of the Semiconductor Process and Design Center at TI Dallas, especially Dr. Ping Yang and Dr. Jue-Hsien Chern for providing valuable discussions and guidance. Thanks are also due to Prof. Tamer Basar and Prof. Janak Patel of University of Illinois for their help. The authors would like to thank an anonymous reviewer for numerous critical and helpful remarks. This research is supported by Texas Instruments Inc. and the Semiconductor Research Corporation.

Appendix (Voltage Drops in Buses)

To calculate voltage drops occurring at various nodes in the power or ground bus, we have to extract the equivalent resistive or RC network of the bus. Various circuit extraction algorithms, such as the one described in [20], can be used for this purpose. In the RC network, capacitances are assumed to be lumped between various nodes and ground. No floating capacitors are allowed in this formulation. Let's assume that the equivalent network of the bus has n nodes. If the node voltage at node i is denoted by v_i, then the voltage drop (with respect to the reference voltage) occurring at this node when it is part of a power bus is $V_{DD} - v_i$. Similarly, when the node is part of the ground bus network, the voltage drop is v_i. Let us denote the vector of voltage drops at various nodes in the equivalent network of the power or ground bus by V. Let us further denote the vector currents entering/leaving various nodes of the network by I, with the following convention for its sign : current leaving a node in the power bus and entering the node in the ground bus is taken as positive. With these notations, it is straightforward to show (see [15]) that when the equivalent network of the bus is represented by a resistive network, the following relationship exists between V and I:

$$YV = I \tag{11}$$

where Y is the *node admittance matrix* of the network. For the case in which the equivalent network of the bus is represented by an RC network, let us assume that the capacitor present at node i is denoted by c_i. Since the current through the capacitor at node i is $-c_i\dot{v}_i$, Eq. (11) can be written as

$$\begin{aligned} YV &= [I_i - c_i\dot{v}_i] \\ &= I - C\dot{V} \end{aligned} \tag{12}$$

where I_i is the current at node i and C is a diagonal matrix of node capacitances. The *ith* diagonal entry of C is equal to c_i. The following lemma holds between the node voltages and the applied currents:

Lemma 1 *If nonnegative current waveforms are injected at various nodes of a resistive or an RC network, then all of the node voltages will be nonnegative.*

Proof : The proof is quite easy for the case of a resistive network. Equation (11) can be written as

$$V = RI \tag{13}$$

where R is the resistance matrix of the network. The lemma follows from the fact that R is an element-wise positive matrix [21].

For the case of the RC network, the current voltage equation for the system is given by Eq. (12):

$$C\dot{V}(t) = I(t) - YV(t) \qquad\qquad V(0) = 0 \tag{14}$$

where time zero is taken as the power on time for the circuit when all of the voltage drops are zero. To prove that the node voltage $v_i(t)$ at any node i is nonnegative for all time (i.e., $v_i(t) \geq 0 \; t \geq 0$), all that has to be proven is that whenever the node voltage becomes zero

($v_i(\tau) = 0$), its time derivative at that time is nonnegative ($\dot{v}_i(\tau) > 0$); therefore, the node voltage never becomes negative. In the beginning ($t = 0$), $V(0) = 0$; therefore,

$$C\dot{V}(0) = I(0) \geq 0 \tag{15}$$

Therefore, the time derivatives of all the node voltages are nonnegative at time 0.

At some arbitrary time $t > 0$, let's assume that the node voltage at node i is zero ($v_i(t) = 0$), while all other node voltages are <u>nonnegative</u> ($v_j(t) \geq 0,\ 1 \leq j \leq n,\ j \neq i$). The differential equation corresponding to node i can be written as

$$c_i \dot{v}_i(t) = I_i(t) - \sum_{j=1}^{n} y_{ij} v_j(t) \tag{16}$$

Since $v_i(t) = 0$,

$$\dot{v}_i(t) = \frac{1}{c_i}\left[I_i(t) - \sum_{\substack{j=1 \\ j \neq i}}^{n} y_{ij} v_j(t) \right] \tag{17}$$

Now, $c_i > 0$, $I_i(t) \geq 0$, $v_j(t) \geq 0$ for $j = 1, 2, \ldots n$, $j \neq i$ and y_{ij}, $i \neq j$ being the off-diagonal terms of the node admittance matrix Y are all negative [22]. Hence $\dot{v}_i(t) \geq 0$. ■

In the following theorem, for vectors X and Y, we use the notation $X \leq Y$ to mean that vector X is element-wise less than or equal to Y, i.e., $x_i \leq y_i,\ 1 \leq i \leq n$.

Theorem 1 *If the vectors of node voltages of a resistive or an RC network in the two cases when nonnegative current waveforms $I_1(t)$ and $I_2(t)$ are injected into various nodes of the network are denoted by $V_1(t)$ and $V_2(t)$, respectively, and if $I_1(t) \leq I_2(t)$, then*

$$V_1(t) \leq V_2(t) \tag{18}$$

Proof : Note that each of $I_1(t)$ and $I_2(t)$ consists of a set of n nonnegative current waveforms. Let $\Delta V(t)$ denote the vector of voltage drops at various nodes when $I_2(t) - I_1(t)$ current waveforms are applied at the nodes. Since $I_2(t) \geq I_1(t)$, $I_2(t) - I_1(t) \geq 0$. Hence, from Lemma A.1, we conclude that $\Delta V(t) \geq 0$. Note that

$$I_2(t) = I_1(t) + (I_2(t) - I_1(t)) \tag{19}$$

Therefore, from the linearity of the resistive or RC network, we have

$$V_2(t) = V_1(t) + \Delta V \tag{20}$$

Since $\Delta V \geq 0$, $V_2(t) \geq V_1(t)$. ■

References

[1] C. Mead and L. Convey, *Introduction to VLSI Systems.* Reading, MA: Addison-Wesley, 1980.

[2] R. Dutta and M. Marek-Sadowska, "Automatic sizing of power/ground (p/g) networks in VLSI," in *Proceedings of 26th ACM/IEEE Design Automation Conference*, pp. 783–786, June 25-29, 1989.

[3] S. Chowdhury, "Optimum design of reliable IC power networks having general graph topologies," in *Proceedings of 26th ACM/IEEE Design Automation Conference*, pp. 787–790, June 25-29, 1989.

[4] S. Chowdhury and J. S. Barkatullah, "Estimation of maximum currents in MOS IC logic circuits," *IEEE Transactions on Computer-Aided Design*, vol. 9, no. 6, pp. 642–654, June 1990.

[5] F. Rouatbi, B. Haroun, and A. J. Al-Khalili, "Power estimation tool for sub-micron CMOS VLSI circuits," in *Proceedings of IEEE/ACM International Conference on Computer-Aided Design*, pp. 204–209, Santa Clara, CA, November 8-12, 1992.

[6] A. Nabavi-Lishi and N. Rumin, "Delay and bus current evaluation in CMOS logic circuits," in *Proceedings of IEEE/ACM International Conference on Computer-Aided Design*, pp. 198–203, Santa Clara, CA, November 8-12, 1992.

[7] U. Jagau, "SIMCURRENT – an efficient program for the estimation of the current flow of complex CMOS circuits," in *Proceedings of IEEE International Conference on Computer-Aided Design*, pp. 396–399, Santa Clara, CA, November 11-15, 1990.

[8] A.-C. Deng, Y.-C. Shiau, and K.-H. Loh, "Time domain current waveform simulation of CMOS circuits," in *Proceedings of IEEE International Conference on Computer-Aided Design*, pp. 208–211, Santa Clara, CA, November 7-10, 1988.

[9] S. Devadas, K. Keutzer, and J. White, "Estimation of power dissipation in CMOS combinational circuits using Boolean function manipulation," *IEEE Transactions on Computer-Aided Design*, no. 3, pp. 373–383, March 1992.

[10] A. Tyagi, "Hercules: a power analyzer for MOS VLSI circuits," in *Proceedings of IEEE International Conference on Computer-Aided Design*, pp. 530–533, Santa Clara, CA, November 9-12, 1987.

[11] H. Kriplani, F. Najm, and I. Hajj, "Improved delay and current models for estimating maximum currents in CMOS VLSI circuits," in *Proceedings of International Symposium on Circuits and Systems*, p. to appear, London, England, 30 May - 2 June, 1994.

[12] H. Kriplani, "Worst case voltage drops in power and ground buses of CMOS VLSI circuits," Ph.D. dissertation UILU-ENG-93-2248, DAC-43, University of Illinois at Urbana-Champaign, November 1993.

[13] Z. Kohavi, *Switching And Finite Automata Theory.* New York, NY: McGraw-Hill Publishing Co. Ltd., 2nd ed., 1992.

[14] K. H. Rosen, *Discrete Mathematics and Its Applications.* McGraw-Hill Inc., 2nd ed., 1991.

[15] L. O. Chua and P.-M. Lin, *Computer-Aided Analysis of Electronic Circuits: Algorithms and Computational Techniques.* Englewood Cliffs, NJ: Prentice-Hall, 1975.

[16] F. Brglez and H. Fujiwara, "A neutral netlist of 10 combinational benchmark circuits and a target translator in fortran," in *Proceedings of International Symposium on Circuits and Systems*, pp. 695–698, June 1985.

[17] H. Kriplani, F. Najm, and I. Hajj, "Maximum current estimation in CMOS circuits," in *Proceedings of 29th ACM/IEEE Design Automation Conference*, pp. 2–7, Anaheim, CA, June 8-12, 1992.

[18] J. Pearl, *Heuristics – Intelligent Search Strategies for Computer Problem Solving.* Reading, MA: Addison-Wesley, 1984.

[19] F. Brglez, D. Bryan, and K. Koźmiński, "Combinational profiles of sequential benchmark circuits," in *Proceedings of International Symposium on Circuits and Systems*, pp. 1929–1934, May 1989.

[20] H. Cha, "Current density calculation using rectilinear region splitting algorithm for very large scale integration metal migration analysis," M.S. thesis, UILU-ENG-90-2223, University of Illinois at Urbana-Champaign, August 1990.

[21] P. K. Chan and M. D. F. Schlag, "Bounds on signal delay in RC mesh networks," *IEEE Transactions on Computer-Aided Design*, vol. 8, no. 6, pp. 581–589, June 1989.

[22] C. A. Desoer and E. S. Kuh, *Basic Circuit Theory.* New York, NY: McGraw-Hill, 1969.