ESTIMATION OF MAXIMUM CURRENT ENVELOPE FOR POWER BUS ANALYSIS AND DESIGN†

S. Bobba and I. N. Hajj
Coordinated Science Lab & ECE Dept.
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801
E-mail: {bobba, i-hajj}@uiuc.edu

Abstract

In this paper we present an input pattern independent method to compute the maximum current envelope, which is an upper bound over all possible current waveforms drawn by a circuit. The maximum current envelope can be used to compute the worst-case RMS current and average current drawn by a set of gates. These current values can be used in the design of the power bus to ensure that the power bus interconnects are not susceptible to electromigration (EM) induced failure. We also present comparisons with exhaustive/long simulations for MCNC/ ISCAS-85 benchmark circuits to verify the accuracy of the method.

1 Introduction

With the significant emphasis on circuit reliability in present day VLSI design, it is essential to estimate the worst-case stress early in the design cycle. For instance, an estimate of the worst-case RMS current can be used in the design of reliable power bus interconnects for which the interconnect power dissipation is within bounds. Power dissipation in an interconnect is related to the resistance of the interconnect and the RMS current through it. Power dissipation in interconnects leads to an increase in the interconnect temperature which dramatically increases the failure rate of interconnects due to electromigration (EM). The early prediction of the worst-case current stress and design of the power bus interconnects using these values eliminates the costly redesign of interconnects that are susceptible to failure.

EM in interconnects is dependent on the average current stress and the interconnect temperature. An estimate of the RMS current can be used to determine the power dissipation and self-heating of the interconnects. There are no existing methods that can generate a tight estimate of the RMS current through a power bus interconnect. Estimates of average current obtained by using probabilistic or *Monte-Carlo* simulation based techniques are dependent on the input statistics. In this work, we generate input pattern-independent estimates of the worst-case RMS and average currents drawn by a circuit using a method that exploits the timing information and the spatio-temporal relationships in the circuit. The current estimates can be used in the design of power bus interconnects for macro-blocks. These pre-designed layouts of the macro-blocks can be reused in any circuit and are guaranteed to be reliable for *all* input statistics.

† This work was supported by SRC under contract 96-DP-109.

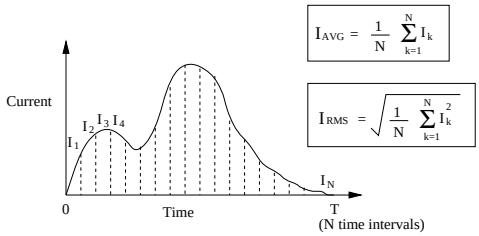

Figure 1: Maximum Current Envelope

The pattern-independent estimates of the worst-case average and RMS currents can also be used in the analysis of the power bus. Since the pattern-independent algorithms are fast and are guaranteed to generate an upper bound, they can be used to eliminate all interconnects that are not susceptible to EM induced failure. This reduces the number of interconnects to be analyzed using more complicated but accurate EM diagnosis tools. Hence, the goal in this work is to estimate the worst-case average and RMS currents.

The maximum current envelope can be used to compute the worst-case RMS current (I_{RMS}) and the worst-case average current (I_{AVG}). The maximum current envelope is a waveform whose value at an instant of time t is an upper bound over the maximum current drawn by the set of gates at the same time instant over all possible input vectors. Fig. 1 shows a maximum current envelope and the expressions for I_{RMS} and I_{AVG}. The values of I_{RMS} and I_{AVG} can be used in the design of the power bus interconnects that supply current to the logic block. For example, if a single interconnect is supplying current to a logic block, then the I_{RMS} and I_{AVG} values can be used in the design of that interconnect. The bound on the power dissipation in interconnects and the I_{RMS} value can be used in determining the upper-bound on the value of the resistance of the interconnect. The EM constraints and the I_{AVG} value can be used to obtain bounds on the width/ length of the interconnect. A number of other design parameters such as maximum instantaneous current, maximum instantaneous energy dissipation and worst-case power dissipation of the macro-blocks can be obtained from the maximum current envelope. The worst case power dissipation of the logic blocks can be used by the placement algorithm to place the logic blocks such that there are no local hot-spots. Also, the problem of estimating the maximum voltage drop in the power bus is related to the problem of estimating the maximum current envelope.

Several approaches have been proposed to estimate the maximum *instantaneous* current for a circuit. In [1], the authors use a branch-and-bound algorithm for finding the input vector that generates worst-case current. The heuristic technique for scanning the input search space does not guarantee an upper bound on the maximum current. In the design for reliability paradigm, a tight upper bound on the maximum current is required. An alternate method to ob-

tain a lower bound is to use random input vector simulation. The lower bound solution generated by this method cannot be used in the design a reliable circuit. In [2], the authors propose a timed ATPG and a probability based method to generate an input vector pair for maximum instantaneous current. The proposed method is complex and it cannot be used to analyze large VLSI circuits in a reasonable amount of CPU time. In [3, 4], Kriplani et al. determine the time window during which a gate in a synchronous sequential circuit can switch by propagating uncertainty waveforms. A loose upper bound current waveform is obtained by a linear time algorithm (*iMax*). They also propose an algorithm for partial input enumeration to account for some of the signal correlations and obtain a tight bound for the maximum instantaneous current. However, for large circuits this method could become computationally expensive as a large number of inputs would have to be enumerated to tighten the upper bound on maximum instantaneous current.

Existing pattern dependent methods for maximum instantaneous current employ heuristic techniques that can underestimate the maximum current or employ search based techniques that are computationally intensive. None of the previous methods can generate a tight estimate of the RMS current. The *iMax* algorithm presented in [3] generates a loose upper bound current waveform as it ignores all signal correlations. In our method, we estimate the the maximum current envelope and not just the maximum instantaneous current using a method that accounts for some of the spatio-temporal correlations of logic signals in the combinational block of a synchronous sequential circuit.

In the next section we formulate the problem. In section 3 we define transition intervals and present a method to compute them. The gate delay model used in this work is also presented in section 3. In section 4 the constraint graph is described. The constraint graph is used for computing the maximum current envelope. In section 5 we present experimental results for the MCNC/ ISCAS-85 benchmark circuits. Finally, in section 6 we give the conclusions.

2 Problem Formulation

CMOS logic gates draw current while making logic transitions (ignoring leakage current). Charging current is drawn from the power bus through the PMOS network for a low-to-high transition. In this work, we consider the charging (dynamic) current drawn by a gate and ignore the short circuit and leakage currents. In well designed circuits the short circuit current is a small percentage of the total current [5]. The method we present is quite general and the short-circuit current can also be included in the analysis. The instant at which a gate draws current from the power bus is dependent on the gate input transitions and their arrival times. This input pattern dependence makes the problem of determining the maximum instantaneous current or the maximum current waveform for a circuit a hard problem. The exact solution to the problem requires an exhaustive search of the exponential input vector search space. Existing pattern dependent methods to find the maximum instantaneous current either explicitly simulate the circuit or implicitly search a subset of the input vector space. These methods do not generate an upper bound and are computationally expensive. Input pattern independent algorithms use the circuit and timing information to generate an upper bound. Using a loose upper bound solution results in a conservative design. Hence, the goal is to generate a tight maximum current waveform using a fast pattern independent algorithm.

In a synchronous sequential circuit the inputs are applied at the clock edge. At some time instants within the clock period logic gates in the circuit make transitions and draw current. We divide the clock period into multiple intervals and find the maximum current drawn by the circuit in each interval. The width of each interval is small enough to allow at most one transition at any gate and large enough to account for the spread of the gate current waveform around the transition instant. The accuracy of the maximum current envelope depends on the number, size and location of the intervals and the current and timing models. In this work, we use simple models and show the effectiveness of the proposed algorithm compared to other methods. Let I^k denote the maximum current drawn by the circuit in the kth interval. The problem of estimating the maximum current can be represented as:

$$I^k = \max\{I_1\alpha_1 + I_2\alpha_2 + \cdots + I_N\alpha_N\} \quad (1)$$

where, I_j is the peak current drawn by the jth gate when it's output node makes a low-to-high transition and $\alpha_j \in \{0, 1\}$. α_j is 1 when the output of the gate makes a transition that results in charging current drawn from the bus, otherwise it is 0. The problem of estimating I^k exactly is a hard problem because of the input pattern dependence.

In the *iMax* algorithm [3] the time window in which a gate in a synchronous sequential circuit can make a logic transition is found by a static timing simulation. The logic gate is assumed to draw current in the entire time interval in which it can make a transition. The maximum current envelope is found as the sum of the currents drawn by the gates in the circuit. This ignores the spatio-temporal correlation across logic gates and results in loose upper bound on maximum current waveform. From the point of view of the problem formulation given in Eqn. (1) the *iMax* algorithm can be interpreted as a maximization problem with no constraints. The objective function of this maximization problem is formulated for each interval using static timing simulation. In the kth time interval, all the gates that can switch have the α_j values set to 1 and all other gates have the value set to 0. The maximum current value in the kth time interval is obtained as a sum of the peak current (I_j) values of the gates that can switch in that time interval ($\alpha_j = 1$). Observe that the correlations across gates are ignored as the gates are assumed to switch independent of other gates. Due to spatio-temporal correlations which relate logic signals in the circuit, some of the gates may not switch.

In this work, we use timing information and some of the spatio-temporal correlations to generate constraints between the logic gates that can switch simultaneously in a time interval. We maximize the objective function given by Eqn. (1) subject to these constraints. We only consider constraints between pairs of gates and do not account for all possible correlations. Hence, the solution we obtain is only an upper bound solution. Since the constraints are between pairs of gates they can be represented as edges in a graph called the constraint graph. In this graph, the vertices represent the logic gates in the circuit. The current values I_i can be used as weights on the vertices of the graph. Hence the problem of finding the maximum current in an interval can be transformed into an optimization problem on a graph. Fig. 2 shows the outline of our proposed algorithm. We first di-

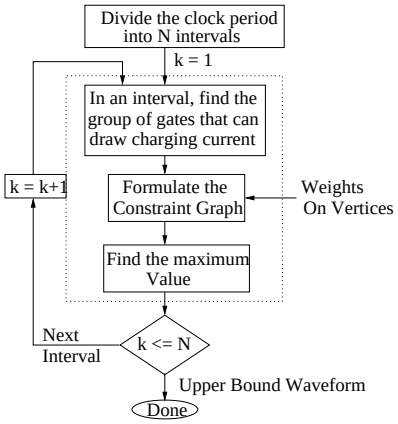

Figure 2: Flow diagram of our method

vide the clock period into a number of time intervals. The width of the time interval corresponds to the minimum pin-pin propagation delay over all gates in the circuit. Hence, the gates in the circuit can make at most one transition in a time interval. For each time interval we find the gates that can make a transition and draw charging current by a static timing simulation. These gates are used to form the constraint graph and find the maximum current value in that interval. In the next section, we describe the method to find the transition intervals at all the gates in the circuit.

3 Transition Intervals

In static CMOS circuits, due to uneven circuit delay paths, the logic gates can glitch and make multiple transitions even if all the primary inputs have a simultaneous single switching event. Transition instants are time instants at which a node can make a transition, when some primary input vector is applied at the clock edge. Transition interval is a time interval which encloses a set of transition instants of a node. A transition interval can also enclose only one transition instant, in which case the begin and the end of the transition interval have the same time value. Each node in the circuit has a set of transition intervals (one or more) that enclose *all* time instants at which the node can switch due to all possible input vectors. These intervals that enclose all transition instants can be obtained by performing a static timing analysis. The inputs to the combinational block of a latch-controlled synchronous circuit switch at the clock edges. The transition intervals associated with each node in the circuit are computed by propagating the transition intervals at the primary inputs throughout the circuit level by level. The levelization of the circuit ensures that the transition intervals at the inputs to a gate are computed before the transition intervals at the output node are calculated. In Section 2.1, the delay model is described. In Section 2.2, the propagation schemes for single and multiple transition intervals are presented.

3.1 Delay Model

A data-independent, scalable pin-to-pin delay model is used for the logic gates. It is assumed that the following parameters, α^i_{LH}, β^i_{LH}, α^i_{HL}, β^i_{HL}, are specified for each input pin i of a gate in the technology library. α^i_{LH} and β^i_{LH} are the block delay and the fanout delay for an LH event at the output of the gate due to an event at input pin i. The propagation delay of an LH output event for an input event at pin i is given by Equation (1), and it is denoted by τ^i_{LH}.

$$\tau^i_{LH} = \alpha^i_{LH} + \beta^i_{LH} * C_{LOAD} \quad (2)$$

C_{LOAD} is the total capacitance associated with the output node. τ^i_{HL} is computed using a similar expression. If the minimum and maximum pin-to-pin delays are specified, then the algorithm uses them in the propagation scheme.

3.2 Propagation scheme

The transition or switching events in a CMOS digital circuit are *Low* to *High* (LH) and *High* to *Low* (HL). Each node in the circuit has LH transition intervals and HL transition intervals. The transition intervals of a particular event type enclose *all* time instants at which that event can occur. The transition intervals at the output of the gate are computed using a data-independent, pin-to-pin delay model for each gate, the pin-to-pin polarity and the input transition intervals. The pin-to-pin polarity of a gate can be inverting, non-inverting or unknown. It is assumed that the technology library for the logic gates contains this information. Since the basic CMOS gates are inverting gates a LH input transition can cause only a HL output transition. Once the appropriate input transition intervals are selected, they are shifted by corresponding propagation delays and merged to get the output LH (or HL) transition intervals.

The single transition interval contains all time instants at which the gate can switch. The begin time of the single transition interval corresponds to the earliest transition instant and the end time corresponds to the last transition instant. For gates close to primary outputs, the single transition interval can be wide due to uneven circuit path delays. The single transition interval can contain some time instants at which the gate cannot switch for any input vector. This can result in loose bound on the maximum current envelope. To prevent this, one can have multiple transition intervals for each node with a fixed limit on the total number of transition intervals for each node. This fixed limit on the number of transition intervals at a node is required because the number of transition intervals could become extremely large for circuits with arbitrary reconvergent fanout. The value of this fixed limit can be used to trade-off accuracy for memory requirement and computational speed. If the number of transition intervals at the output of a node are greater than the fixed limit, then the closest intervals are repeatedly merged till the number of intervals reduces to the fixed limit. Note that it is still guaranteed that the multiple transition intervals for a gate enclose *all* the time instants at which the gate can switch. In the next section, we describe the method for creating the constraint graph and use it to obtain the maximum current envelope.

4 Constraint Graph

In the previous section, we described a method that uses static timing simulation to determine the group of gates that can switch in an interval and draw charging current. The problem we address in this section is: Can all the gates simultaneously draw charging current? A logic transition at a gate in a time interval results in charging current drawn from the bus. If the output node of a gate makes a logic transition at a time instant, it implies that the node assumes an initial logic value before the transition instant and a final logic value after the transition. When a node assumes a logic value, it can imply a logic value at other nodes in the circuit. In section 4.1, we present an algorithm to extract the logic implications in the circuit. These logic implications can be used to obtain the constraints. In this work, we obtain the constraints between every pair of gates that can

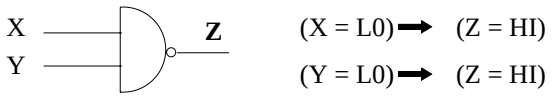

Figure 3: Two Input NAND gate

switch in an interval. A constraint exists between a pair of gates if they cannot simultaneously draw charging current from the power bus. The logic gates that can switch in an interval are denoted as vertices and the constraints between a pair of gates are represented as edges in the constraint graph. The constraints are described in section 4.2. The peak current drawn by a gate is used as the weight on the vertex that corresponds to that gate in the constraint graph. The problem of finding the maximum value of the current in an interval reduces to an optimization problem on the constraint graph. In section 4.3, we present a technique that employs the constraint graph to obtain the maximum charging current drawn by a set of gates in an interval. The constraints developed are valid for *all* the input vectors. If a simulation is performed for any input vector, the constraints relating the logic signals are always satisfied. Hence, this method always generates an upper-bound.

4.1 Logic Implications

Consider the 2-input NAND gate shown in Fig. 3. A logic assignment of LO to any one of the inputs implies a logic HI at the output, because LO is a controlling input value for a NAND gate. Hence the input node and the output node of a NAND gate in any circuit cannot have a simultaneous LH or HL transition for any input vector pair. These logic implications that account for spatial correlations across gates can be used to obtain constraint edges in the constraint graph. The construction of the constraint graph using the single logic implications will be described in the next subsection.

Single logic implications are relations between logic assignments to two nodes in a circuit. Let X and Y be two nodes in the circuit with the relation $(X = v_x) \Rightarrow (Y = v_y)$, a logic assignment of v_x to node X implies a logic value v_y to node Y. Since $v_x, v_y \in \{LO, HI\}$, there can be four types of relations between nodes corresponding to the four different logic assignments to the pair of nodes. Associated with each single logic implication is a list of time instants called the time-to-imply. The time-to-imply list associated with each implication contains the propagation time delay to the implied node. Due to reconvergent fanout, the list can contain multiple time instants. Let **I** denote the set of single logic implications in a circuit. The elements of set **I** satisfy the following rules:

- Contra-positive law: If $((X = v_x) \Rightarrow (Y = v_y)) \in \mathbf{I}$ then, $((Y = \neg v_y) \Rightarrow (X = \neg v_x)) \in \mathbf{I}$.
- Transitive law: If $((X = v_x) \Rightarrow (Y = v_y)) \in \mathbf{I}$ and $((Y = v_y) \Rightarrow (Z = v_z)) \in \mathbf{I}$ then, $((X = v_x) \Rightarrow (Z = v_z)) \in \mathbf{I}$.

The elements of set **I** are extracted from the circuit. Since logic implications in the set **I** are used to generate constraint edges, the goal is to find all or as many single logic implications as possible. The algorithm for computing the elements of the set **I** first computes the fan-in cone implications for each node using set union and intersection operations. All the fan-in cone implications for the output node of logic gates are computed by a single pass over the levelized gate net-list. Each node X in the circuit has four implication lists: L^x_L, L^x_H, H^x_L and H^x_H. Each of these lists contain nodes in the fan-in cone of node X that imply a logic value at node X. All the time delay values that correspond to the propagation delay between the node X and nodes in the fan-in cone of node X that imply a logic value at node X are present in the time-to-imply lists associated with the node X. The trivial implications at a node, $(X = LO)$ and $(X = HI)$ are included in the L^x_L and H^x_H lists respectively. The time-to-imply values for these trivial implications is set to 0. The list L^x_H contains nodes in the fan-in cone of the node X which when assigned a logic HI value imply a LO value at node X. The other implication lists have similar definitions. The gate output node implication lists are computed by set intersection and union operations on the implication lists of the gate input nodes. The primary input nodes are initialized with the trivial implications and the gate output node implication lists are evaluated starting from the first level. The gate levelization process guarantees that all the input implication lists are evaluated before a gate output node implication lists are evaluated. The gate Boolean function is used to perform the set operations (intersection or union) to compute the output node implication lists. The pin-pin delay of the logic gate is used to compute the time-to-imply values for the output node implications. The output implication lists for the two input NAND gate in Fig. 3 are obtained using the following operations:

- Intersection: $L^z_L = H^x_L \cap H^y_L$ and $L^z_H = H^x_H \cap H^y_H$
- Union: $H^z_L = L^x_L \cup L^y_L$ and $H^z_H = L^x_H \cup L^y_H$

Interchanging the intersection and union operators in the above equation gives the set-operations for computing the output node implication lists for a two input NOR gate. For gates with more than two inputs the above operations (intersection, union) are performed on all the input implication lists. If the gate is an inverter the input implication lists are duplicated and then the implication lists (L^x_L, H^x_L) and (L^x_H, H^x_H) are swapped to obtain the correct output implication lists. The output implication lists for complex CMOS gates are computed by a combination of the methods used for NAND, NOR and inverter. Contra-positive law is used to find the fan-out cone implications for each node after the fan-in cone single implications are evaluated for all the nodes. Finally, we iterate using the transitive law until no new implications are found. These logic implications are used to form the constraint graph. In the next section, we describe the method to generate the constraint graph.

4.2 Constraints

A logic transition implies a transition from an initial logic value to a final logic value. In a digital circuit, the logic value can be either low or high. A logic transition is either a LH (low-high) or a HL (high-low) transition. We now define Tmin1, Tmin0 for the output node of a gate.

Tmin1: It is the minimum input pin to output pin delay for an output HL event. This value gives the minimum time interval during which the output node remains high.

Tmin0: It is the minimum input pin to output pin delay for an output LH event. This value gives the minimum time interval during which the output node remains low.

The values of Tmin1, Tmin0 are used to generate the constraints. The transition intervals associated with each node can be used to determine the set of gates that can switch in an interval. Let S_k denote the set of gates for the kth time

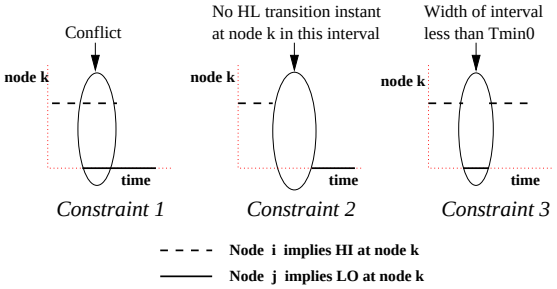

Figure 4: Constraints due to spatio-temporal correlations

interval. Each of these gates corresponds to a vertex in the constraint graph for that time interval. The constraints are checked for every pair of gates in S_k. If for a pair of gates, a constraint is violated then the two gates cannot switch simultaneously and draw charging current in kth time interval over all possible input vectors. If a pair of gates violate any of the constraints, then it is represented as an edge in the constraint graph between the vertices that correspond to the gates. Let (i, j) denote a pair of gates that belong to the set S_k. For every pair of gates, (i, j) that belong to S_k the constraints due to across gate implications and spatio-temporal correlations are checked.

Across gate implications: If one of the gates is an input of the other gate and if a logic assignment to a gate implies a complementary logic value at the other gate, then the pair of gates cannot simultaneously switch from LH (HL). In CMOS gates, if a logic value at the input of a gate implies a logic value at the output of a gate, then input and output nodes cannot have a simultaneous LH/ HL transition. This is because CMOS gates are inverting gates, and hence the logic values of input-output implications across a gate are complementary. For instance, a logic assignment of LO to an input of a NAND gate implies a logic HI value at the output. Hence, the input and output nodes of a NAND gate cannot have a simultaneous LH or HL transition.

Spatio-temporal correlations: If the gates (i, j) make a LH transition in the kth time interval, then they draw simultaneous charging current. For each of the gates (i, j), we find the logic assignment at all other gates due to the logic implication of a LH transition in a particular time interval. This can be done using the implication lists associated with the gate i or j and the time-to-imply values of each of the implications. The constraints due to spatio-temporal correlations are shown in Fig. 4 and are described below.

Constraint 1: If there exists a node k at which gates i and j imply complementary logic values at any time instant, then the output node of gates (i, j) cannot make a simultaneous LH transition.

Constraint 2: If there exists a node k at which gates i and j imply complementary logic values and there is no transition instant between the two implied values to make the logic values at node k consistent, then there is a conflict.

Constraint 3: If there exists a node k at which one of the gates implies a logic value at two different time instants and the other gate implies a complementary logic value in between the two time instants and the width of the interval between the two time instants is less than Tmin0/ Tmin1, then the node will glitch (make an incomplete transition) and this is not propagated through the fanout gates. Hence, the output nodes of gates (i, j) cannot make a simultaneous LH transition.

4.3 Maximum weight independent set

The weight on a vertex denotes the peak charging current drawn by the corresponding logic gate in the circuit. An edge between two vertices in the constraint graph implies that the corresponding vertices cannot simultaneously draw charging current from the power bus. Hence the problem of finding the maximum current drawn by a set of gates in an interval reduces to problem of finding a set of vertices in the constraint graph such that the sum of weights on the vertices is maximum and there are no edges between any pair of the vertices. This problem is exactly the problem of finding the maximum weight independent set in the constraint graph. An independent set in a graph is defined as a set of vertices with no edges between any pair of the vertices in the set. A maximal independent set is an independent set for which none of the vertices in the remaining graph can be appended to increase the size of the independent set. A maximum weight independent set in a graph is a maximal independent set for which the sum of weights on the vertices is maximum over all maximal independent sets.

The problem of determining the maximum weight independent set in an arbitrary graph is NP-Complete [6]. There exists some classes of graphs like perfect graphs, claw-free graphs for which the problem can be solved in polynomial time. Since the constraint graph does not belong to any of these special classes of graphs, it is not known if the maximum weight independent set problem can be solved in polynomial time for the constraint graph. We have implemented the recursive backtracking algorithm presented in [7] for finding the maximum weight independent set in a graph. In the next section, we present the experimental results.

5 Experimental Results

In this section, we present the experimental results for the ISCAS-85 and MCNC benchmark circuits. Each of these combinational multilevel circuits were optimized by *script.rugged* and mapped with *lib2.genlib* as the library of gates, using SIS [8]. The load capacitance of the output node of a gate and the delay parameters specified in the technology library for the gate were used to compute the pin-to-pin delay values of a gate using Eqn. (2). The peak current value for each gate was chosen (arbitrarily) to be 1.0 unit and it is used as the weight associated with the corresponding vertex in the constraint graph. The limit on the total number of transition intervals per node was set to 15. The run time values are in CPU seconds on a UltraSparc2 workstation.

In this paper, we present a comparison of the accuracy of the maximum current envelope generated using *iMax* algorithm, the constraint graph based algorithm described in this paper and exhaustive/random input vector pair simulation. Exhaustive simulation is performed for circuits with small number of primary inputs. Logic simulations for 100000 randomly generated input vector pairs is performed for larger circuits. The logic simulator used in the simulations had the same gate delay and current models described in this paper. Random input vector simulation was used for the ISCAS-85 benchmark circuits and exhaustive simulation was used for MCNC benchmark circuits.

Fig. 5 shows a plot of the maximum current envelope generated using *iMax* algorithm, the constraint graph based method and exhaustive input vector pair simulation for MCNC benchmark circuit Z4ML. It can be seen that the constraint graph based method generates a tight upper bound

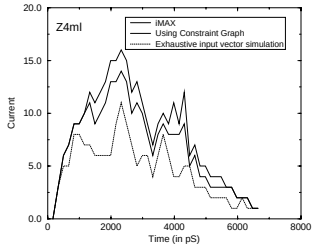

Figure 5: Maximum current envelope for Z4ML

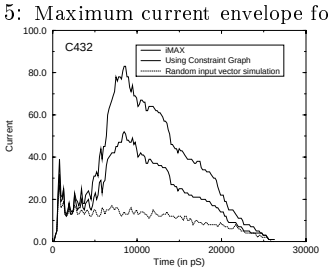

Figure 6: Maximum current envelope for C432

on the maximum current waveform. The *iMax* algorithm uses the timing information and ignores all signal correlations. The constraint graph based method uses the timing information and accounts for pair-wise correlations. The exhaustive input vector pair simulation accounts for all possible correlations (simulation based) and scans the entire input vector space. Hence, it is natural that the constraint graph based method generates results that lie between the results generated by *iMax* algorithm and exhaustive input vector pair simulation. Fig. 6 shows a plot of the maximum current envelope generated using *iMax* algorithm, the constraint graph based method and simulations with 100000 random input vector pairs for ISCAS-85 benchmark circuit C432. The difference between the current waveforms generated using the constraint graph based method and random input vector simulation can be attributed to one or all of the following reasons:

1. In the constraint graph based method we account for pair-wise correlations and not all possible correlations.
2. The random input vector pair simulation for 100000 covers only a small fraction of the entire search space. C432 has 32 primary inputs and the number of possible input vector pairs is 4^{32}.
3. The transition intervals at a gate obtained using the static timing simulation can include time instants at which the gate cannot make a transition.

Table 1 shows the results for MCNC benchmark circuits. In this we present the I_{RMS}, I_{AVG} values computed using *iMax* algorithm, the constraint graph based method and exhaustive input vector pair simulation for MCNC benchmark

Table 1: Results for MCNC benchmark circuits

	iMax			Constraint graph			Exhaustive	
circuit	I_{rms}	I_{avg}	Run Time	I_{rms}	I_{avg}	Run Time	I_{rms}	I_{avg}
9symml	21.59	15.81	0.14	11.77	9.34	0.26	8.73	7.24
alu2	35.01	27.44	6.55	16.19	12.77	28.34	9.27	7.69
alu4	20.01	14.97	12.28	11.98	8.19	73.23	7.47	5.69
b1	2.34	2.05	0.02	2.18	1.88	0.08	1.87	1.50
cm138a	7.48	7.08	0.07	6.14	4.97	0.13	4.28	3.84
cm151a	7.22	6.27	0.07	4.61	3.97	0.15	4.53	3.75
cm42a	8.46	7.80	0.08	4.50	3.46	0.13	3.06	2.77
cm82a	4.92	4.42	0.04	3.23	2.96	0.11	2.97	2.68
cm85a	9.45	8.45	0.11	4.97	4.29	0.21	3.85	3.23
decod	6.56	4.84	0.06	3.41	2.67	0.11	2.36	1.56
f51m	20.97	16.68	0.49	12.18	9.76	0.59	7.59	6.27
majority	1.84	1.66	0.01	1.77	1.57	0.07	1.54	1.36
x2	9.57	7.94	0.24	6.27	5.32	0.35	5.00	4.38
z4ml	8.67	7.03	0.16	7.94	6.32	0.26	5.35	4.67

Table 2: Results for ISCAS-85 benchmark circuits

	iMax			Constraint graph			Random	
ckt	I_{rms}	I_{avg}	Run Time	I_{rms}	I_{avg}	Run Time	I_{rms}	I_{avg}
c432	42.66	35.56	1.15	26.63	22.59	9.01	11.54	10.55
c499	57.29	46.65	1.77	30.03	26.91	16.62	17.72	16.84
c880	54.01	42.25	2.61	31.93	25.17	30.34	14.14	12.21
c1908	61.54	52.77	2.26	26.71	20.89	33.77	15.94	14.15
c2670	79.45	69.11	6.97	47.23	41.57	86.75	21.30	18.12
c3540	76.89	62.71	4.54	38.17	29.63	71.28	22.69	20.00
c5315	91.61	72.97	5.81	47.22	37.99	38.51	27.91	23.14
c6288	232.63	179.61	10.77	119.72	87.21	362.13	56.21	40.77
c7552	169.26	131.19	7.12	82.11	64.93	112.69	37.00	31.45

circuits. It can be seen that the constraint graph based method generates tight upper-bound values. Table 2 shows the results for ISCAS-85 benchmark circuits. In this we present the I_{RMS}, I_{AVG} values computed using *iMax* algorithm, the constraint graph based method and simulations with 100000 random input vector pairs for ISCAS-85 benchmark circuits. It can be seen that the constraint graph based method generates a tighter bound compared to the *iMax* algorithm. The run time values for constraint graph based method are reasonable even for large circuits.

6 Conclusions

We have presented an input pattern independent algorithm for computing the maximum current envelope for a circuit. The algorithm uses circuit functionality, some of the spatio-temporal correlations, and timing information to generate a tight bound on the maximum current envelope. The values of I_{RMS}, I_{AVG} are computed using the maximum current envelope and they can be used in the design and analysis of power bus. The method we presented is quite general and it can be applied to asynchronous circuits or circuits where the primary input signal arrival times are not known. We have presented comparisons with results obtained by exhaustive/long random input vector simulations to show that the constraint graph based method generates tight upper bound results. The algorithm for computing the maximum current envelope and the I_{RMS}, I_{AVG} values is fast and it requires only a few minutes of CPU time for the largest circuit.

References

[1] S. Chowdhury and J. Barkatullah, "Estimation of maximum currents in MOS IC logic circuits," *IEEE Trans. on CAD*, vol. 9, no. 6, pp. 642–654, June 1990.

[2] A. Krstic and K. Cheng, "Vector generation for maximum instantaneous current through supply lines for CMOS circuits," in *Proc. of 34th DAC*, pp. 383–388, June 9-13, 1997.

[3] H. Kriplani, F. Najm, and I. Hajj, "Maximum current estimation in CMOS circuits," in *Proc. of 29th DAC*, June 8-12, 1992, pp. 2–7.

[4] H. Kriplani, F. Najm, P. Yang, and I. Hajj, " Resolving signal correlations for estimating maximum currents in CMOS combinational circuits," in *Proc. of 30th DAC*, pp. 384–388, June 14-18, 1993.

[5] 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. 19, no. 4, pp. 468-473, August 1984.

[6] M. R. Garey and D. S. Johnson, *Computers and Intractability*. New York, NY: W. H. Freeman and Company, 1979.

[7] E. Loukakis and C. Tsouros, "An algorithm for the maximum internally stable set in a weighted graph," *Int. Journal of Comp. Math.*, vol. 13, no. 2, pp. 117–129, 1983.

[8] R. Brayton, G. D. Hachtel, and A. L Sangiovanni-Vincentelli, "Multilevel logic synthesis," in *Proc. of the IEEE*, vol. 78, no. 2, pp. 264–300, February 1990.