Estimation of Average Switching Activity in Combinational Logic Circuits Using Symbolic Simulation

José Monteiro, Srinivas Devadas
MIT, EECS, Cambridge

Abhijit Ghosh
MERL, Sunnyvale

Kurt Keutzer
Synopsys

Jacob White
MIT, EECS, Cambridge

Abstract

We address the problem of estimating the average switching activity of combinational circuits, under random input sequences. Switching activity is strongly affected by gate delays and for this reason we use a variable delay model in estimating switching activity. Unlike most probabilistic methods that estimate switching activity, our method takes into account correlation caused at internal gates in the circuit due to reconvergence of input signals.

This method assumes a particular delay model, and further assumes that the primary inputs to the combinational circuit are uncorrelated. Both these assumptions can be relaxed at the cost of increased complexity. We describe extensions to handle transmission gates and inertial delays in this paper.

1 Introduction

For many consumer electronic applications low average power dissipation is desirable and for certain special applications low power dissipation is of critical importance. For personal communication applications such as hand-held mobile telephones, low-power dissipation may be the tightest constraint in the design. More generally, with the increasing scale of integration, we believe that power dissipation will assume greater importance, especially in multi-chip modules where heat dissipation is one of the biggest problems.

We address the problem of estimating the **average switching activity** in combinational circuits given random input sequences. This measure is related to the average power dissipation of a circuit, and can be used to make architectural or design-style decisions during the VLSI synthesis process. These measures can also be used to drive logic optimization methods that target low power dissipation.

Probabilistic methods for power or current estimation are attractive because statistical estimates can be obtained without recourse to time-consuming exhaustive simulation. In the past, probabilistic peak current estimation methods (e.g., [**?**]) that compute probabilistic current waveforms for combinational circuits have been developed. Estimation of worst-case power dissipation (e.g., [**?**], [**?**]) is a difficult problem requiring a branch-and-bound search and these methods have been applied to small to moderate sized circuits.

Our main contribution in this paper is to show that based on some realistic assumptions about transistor-level behavior, the problem of estimating switching activity in combinational circuits can be reduced to one of computing signal probabilities[1] of a multilevel circuit derived from the original circuit by a process of **symbolic simulation**. The work closest to ours is the transition density calculation method by Najm [**?**]. Transition densities correspond to average switching rates for gates in the circuit. In [**?**], an interconnection of combinational logic modules, each with a certain delay, makes up a circuit. Transition densities are propagated through combinational logic modules without regard to their structure. Correlations between internal lines due to reconvergence are ignored during propagation. It is possible to take into account correlations by lumping all the modules into one large module, but in this case the information regarding the delay of the individual modules is lost.

[1]The signal probability of a wire in a circuit is defined as the probability that a randomly generated input vector will produce a 1 value at the wire [**?**].

For an excellent review of power estimation methods the reader is referred to [?]. Our work on switching activity estimation, originally presented in [?], has the following important characteristics. We use a variable delay model for combinational logic in our symbolic simulation method, which correctly computes the Boolean conditions that cause *glitching* (multiple transitions at a gate) in the circuit. In some cases, glitching may account for a significant percentage of the switching activity. Symbolic simulation produces a set of Boolean functions that represent the conditions for switching at different time points, for each gate in the circuit. Given input switching rates, we can use various exact or approximate methods to compute the probability of each gate switching at any particular time point. We then sum these probabilities over all the gates to obtain the expected switching activity in the entire circuit over all the time points corresponding to a clock cycle. Our method takes into account correlation caused at internal gates in the circuit due to reconvergence of input signals (reconvergent fanout).

The rest of this paper is organized as follows. In Section 2 we first review the physical model for power estimation, introduce basic terminology and state all the key assumptions of our work. In Section 3, we describe a symbolic simulation method that computes the Boolean conditions required for a gate to switch at each time point during a clock cycle, under unit and variable delay models. We describe some extensions to this method in Section 4. In Section 5, we present results on several circuits, including those from the ISCAS-89 benchmark set. We conclude in Section 6.

2 Preliminaries and Assumptions

2.1 A Power Dissipation Model

In this section, we observe that under a simplified model of the energy dissipation in CMOS circuits, the energy dissipation of a CMOS circuit is directly related to the switching activity.

In particular the three simplifying assumptions are:

- The only capacitance in a CMOS logic-gate is at the output node of the gate.
- Either current is flowing through some path from V_{DD} to the output capacitor, or current is flowing from the output capacitor to ground.
- Any change in a logic-gate output voltage is a change from V_{DD} to ground or vice-versa.

All of these are reasonably accurate assumptions for well-designed CMOS gates [?], and when combined imply that the energy dissipated by a CMOS logic gate each time its output changes is roughly equal to the change in energy stored in the gate's output capacitance. If the gate is part of a synchronous digital system controlled by a global clock, it follows that the average power dissipated by the gate is given by:

$$P_{avg} = 0.5 \times C_{load} \times (V_{dd}^2/T_{cyc}) \times E(transitions) \tag{1}$$

where P_{avg} denotes the average power, C_{load} is the load capacitance, V_{dd} is the supply voltage, T_{cyc} is the global clock period, and $E(transitions)$ is the *expected value* of the number of gate output transitions per global clock cycle [?], or equivalently the average number of gate output transitions per clock cycle. All of the parameters in (1) can be determined from technology or circuit layout information except $E(transitions)$, which depends on both the logic function being performed and the statistical properties of the primary inputs.

2.2 Static Probabilities

Consider the case of dynamic CMOS logic (e.g., Domino). At the beginning of each clock cycle, all the gates are precharged, and gates make transitions only if their associated Boolean functions are satisfied. For example, a three-input AND-OR gate's Boolean function might be

$$(i_1 \cdot i_2) \vee (i_2 \cdot i_3), \tag{2}$$

where i_1, i_2, and i_3 are primary inputs. In this case, the expected value of the number of transitions at this gate's output is

$$E(transitions) = 2 \times P((i_1 \cdot i_2) \vee (i_2 \cdot i_3) = 1) \tag{3}$$

where $P(x)$ is defined as probability that x is true, and the factor of two in the equation accounts for the reset transition during precharge.

To evaluate (3), it is necessary to determine the primary input probabilities. We assume that *primary inputs are uncorrelated*, and that each is a waveform in time whose value is either zero or one, changing instantaneously at global clock edges. Assuming ergodicity without further comment, the probability of a particular input i_j being one at a given point in time, denoted ${p_j}^{one}$, is given by

$${p_j}^{one} \quad = \quad \lim_{N \to \infty} \frac{\sum_{k=1}^{N} i_j(k)}{N} \tag{4}$$

where N is the total number of global clock cycles and $i_j(k)$ is the value of input i_j during clock cycle k. Clearly, the probability that i_j is zero at a given point in time, denoted ${p_j}^{zero}$ is

$${p_j}^{zero} = 1 - {p_j}^{one}$$

We refer to ${p_j}^{zero}$ and ${p_j}^{one}$ as **static probabilities**. Note that

$$P((i_1 \cdot i_2) \vee (i_2 \cdot i_3) = 1) \neq p_1^{one} p_2^{one} + p_2^{one} p_3^{one}$$

because the first and second product terms are not independent. Rather,

$$P((i_1 \cdot i_2) \vee (i_2 \cdot i_3) = 1) =$$

$$P((i_1 \cdot i_2) \vee (\overline{i_1} \cdot i_2 \cdot i_3) = 1) = p_1^{one} p_2^{one} + p_1^{zero} p_2^{one} p_3^{one}$$

where the second equality holds because $i_1 \cdot i_2$ is disjoint from $\overline{i_1} \cdot i_2 \cdot i_3$. In this example, $(i_1 \cdot i_2) \vee (\overline{i_1} \cdot i_2 \cdot i_3)$ represents a disjoint cover for the logic function, and the terms $i_1 \cdot i_2$ and $\overline{i_1} \cdot i_2 \cdot i_3$ are referred to as *cubes* in the cover [**?**]. The equivalent logical expression, $(i_1 \cdot i_2) \vee (i_2 \cdot i_3)$, does not represent a disjoint cover because $i_1 \cdot i_2 \cdot i_3$ is contained in both cubes $i_1 \cdot i_2$ and $i_2 \cdot i_3$.

In general, given a disjoint cover for a Boolean function of uncorrelated inputs described by static probabilities, it is easy to determine the probability of the function evaluating to a 1. The procedure is given in the following two theorems whose proof follows directly from elementary probability [**?**].

Theorem 2.1 : *Given any disjoint cover for a Boolean function, the probability of the function evaluating to a* 1 *is equal to the sum of the probabilities of each of the cubes in the cover evaluating to a* 1.

Theorem 2.2 : *Given a logical function of uncorrelated inputs in the form*

$$f = i_{\alpha_1} \cdot i_{\alpha_2} \cdot \cdots \cdot i_{\alpha_M} \cdot \overline{i_{\beta_1}} \cdot \overline{i_{\beta_2}} \cdot \cdots \cdot \overline{i_{\beta_N}}$$

where the i_{α_j}'s are nonnegated inputs and the $\overline{i_{\beta_k}}$'s are negated inputs, then

$$P(f = 1) = p_{\alpha_1}^{one} \cdot p_{\alpha_2}^{one} \cdot \cdots \cdot p_{\alpha_M}^{one} \cdot p_{\beta_1}^{zero} \cdot p_{\beta_2}^{zero} \cdot \cdots \cdot p_{\beta_N}^{zero}$$

2.3 Transition Probabilities

For static CMOS combinational logic, a gate output can only change when its inputs change, and then only if the Boolean function describing the gate evaluates differently. For example, a 2-input AND gate's output will change between clock cycle t and $t+1$ if

$$(i_1(t) \cdot i_2(t)) \oplus (i_1(t+1) \cdot i_2(t+1)) \tag{5}$$

evaluates to 1, where $i_1(t)$, $i_2(t)$ and $i_1(t+1)$, $i_2(t+1)$ are the inputs to the gate at clock cycle t and $t+1$ respectively. The disjoint cover for (5) is

$$(i_1(t) \cdot i_2(t)) \cdot \overline{(i_1(t+1))} \ \vee \ (i_1(t) \cdot i_2(t)) \cdot (i_1(t+1) \cdot \overline{i_2(t+1)}) \ \vee$$
$$\overline{i_1(t)} \cdot (i_1(t+1) \cdot i_2(t+1)) \ \vee \ (i_1(t) \cdot \overline{i_2(t)}) \cdot (i_1(t+1) \cdot i_2(t+1)). \tag{6}$$

It is not possible to use Theorem 2.2 to evaluate the probability of (6), because an input at time $t+1$ is correlated to its behavior at time t (as in a sequential circuit). Instead, **transition probabilities** for the transitions $0 \to 0$, $0 \to 1$, $1 \to 0$ and $1 \to 1$ must be used. We denote these transition probabilities by ${p_j}^{00}$, ${p_j}^{01}$, ${p_j}^{10}$, and ${p_j}^{11}$, respectively, where for example ${p_j}^{10}$ is defined by:

$${p_j}^{10} = \lim_{N \to \infty} \frac{\sum_{k=1}^{N} i_j(k)\overline{i_j(k+1)}}{N} \tag{7}$$

The other transition probabilities follow similarly.

Static probabilities can be computed from transition probabilities, but *not* vice-versa, because of correlation between one time frame and the next. So in general it is necessary to specify transition probabilities. The relations between static probabilities and transition probabilities follow directly from the definitions in (4) and (7), specifically,

$$\begin{aligned} {p_j}^{zero} &= {p_j}^{00} + {p_j}^{01} \\ {p_j}^{one} &= {p_j}^{11} + {p_j}^{10} \end{aligned} \tag{8}$$

.

Both static and transition probabilities are used to compute $E(transitions)$ for static logic circuits, as can be seen from the expression for the probability that (6) evaluates to a 1,

$$p_1^{10} \cdot p_2^{one} + p_1^{11} \cdot p_2^{10} + p_1^{01} \cdot p_2^{one} + p_1^{11} \cdot p_2^{01} \tag{9}$$

. For all primary inputs, *we assume that successive input vectors are uncorrelated.*[2] A 1 or a 0 may be equally likely, in which case all transition probabilities may be assumed to be 0.25, and all static probabilities to be 0.5. It is also possible that a 1 or a 0 at a particular primary input are not equally likely resulting in non-uniform transition and static probabilities. We assume that these probabilities are provided by the user.

[2] This assumption is relaxed in the work of [?].

2.4 General Combinational Networks

The algorithm for computing the average power dissipated in a dynamic CMOS combinational network follows directly from the approach in Section 2.2. That is, for each gate g_i in a logical network, we first determine the Boolean function of the gate, f_i, in terms of the network's primary inputs. Then we find a disjoint cover for f_i, and use Theorem 2.1 and 2.2 to evaluate $P(f_i = 1)$.

The average power dissipation for the network is then

$$\begin{aligned} P_{avg} &= 0.5 \times (V_{dd}^2 / T_{cyc}) \times \sum C_i \times E(transitions\ of\ g_i) \\ &= (V_{dd}^2 / T_{cyc}) \times \sum C_i \times P(f_i = 1) \end{aligned} \tag{10}$$

where C_i is the load capacitance of the i^{th} gate and the summation is over all gates in the circuit. Note that (10) follows from the fact that the average value of a sum is equal to the sum of the average values, regardless of correlation [**?**].

A similar overall approach can be used to compute the average power dissipated in a static CMOS combinational network. However, as described in Section 2.3, a two-vector input sequence is required to stimulate activity in static gates. Therefore, the Boolean function for static gate output activity is different than that for a dynamic gate, and the probability of the function being satisfied requires transition probabilities for evaluation. In particular, assuming negligible gate delays, a static CMOS logic gate's output will switch with a change in the primary input vector from $V0$ to Vt if:

$$\begin{aligned} f_i \;=\; ((h_i(V0) \;=\; 0) \;\wedge\; (h_i(Vt) \;=\; 1)) \;\bigvee \\ ((h_i(V0) \;=\; 1) \;\wedge\; (h_i(Vt) \;=\; 0)) \end{aligned} \tag{11}$$

is satisfied, where h_i is the logic function corresponding to gate g_i's output.

The average power dissipated in the static network is then computed by using (10), with f_i being given by (11), and $P(f_i = 1)$ is evaluated using both transition and static probabilities, as described in Section 2.3.

3 Symbolic Simulation to Estimate Switching Activity

As mentioned above, for static CMOS circuits, switching activity must be analyzed based on considering an input vector pair, denoted $< V0,\ Vt >$. If the gates have appreciable delays, there may be output glitches that can contribute significantly to dissipated power.

For instance, consider the circuit of Figure 1. Assuming that the delays of the INVERTER and the AND gate are both 1 time unit, if we apply the vector $i_1 = 0$, $i_2 = 0$, followed by $i_1 = 1$, $i_2 = 1$, we will obtain a glitch at the output *out*, which could cause power dissipation. Below, we present a *symbolic simulation* method that can be used to generate a multiple-output function that represents the total switching activity over any possible input vector pair, assuming a variable delay model for the gates in the circuit.

3.1 Symbolic Simulation

3.1.1 An Example

Consider the multilevel combinational circuit shown in Figure 2.

It has four primary inputs and one output and consists of four CMOS gates. Figure 3 shows the signal transitions at the primary input nodes as well as the possible transitions at the internal nodes of the network. The time points are in normalized units. The first three inputs, i_1, i_2, and i_3, switch simultaneously between time periods 0 and 1. The fourth input, i_4, is a late arriving signal that switches between the time points 5 and 6. In this example, the delays of both Gate 1 and Gate 3 are one time unit. Gate 2 has a delay of two units while Gate 4 has a delay of four units.

The e_i waveforms in Figure 3 represent the signals at the output of the i^{th}th logic gate. Each of the possible transitions, $e_{i,j}$, represents either a low-to-high or high-to-low signal transition between $[j]^{th}$ and $[j+1]^{th}$ time points. The number of all possible transitions at a gate output may equal the sum of all possible transitions at the gate inputs. These transitions are delayed by the gate's propagation delay.

3.1.2 Unit-Delay Model

Even under the idealization of a unit-delay model, the gate output nodes of a multilevel network can have multiple transitions in response to a two-vector input sequence. In fact, it is possible for a gate output to have as many transitions as levels in the network.

We construct the Boolean functions describing the gate outputs at the discrete points in time implied by the unit-delay model. That is, we consider only discrete times t, $t+1$, $\ldots$, $t+l$, where t is the time when the inputs change from $V0$ to Vt, and l is the number of levels in the network. For each gate output i, we use symbolic simulation to construct the $l+1$ Boolean functions $f_i(t+j)$,

$j \in 0, \ldots, l$ which evaluate to 1 if the gate's output is 1 at time $t + j$. Note that as we assume no gate has zero delay and that the network has settled before the inputs are changed from $V0$ to Vt, $f_i(t)$ is the logic function performed for $V0$ at the i^{th} gate output. Finally, we can determine whether a transition occurs at a boundary between discrete time intervals $t + j$ and $t + j + 1$ by exclusive-OR'ing $f_i(t + j)$ with $f_i(t + j + 1)$.

Consider the network in Figure 1. For this network,

$$f_1(t) = \overline{V0_1}$$

$$f_2(t) = \overline{V0_1} \wedge V0_2$$

Assuming both gates have unit delay,

$$f_1(t+1) = \overline{i_1(t)} = \overline{Vt_1}$$

$$f_2(t+1) = f_1(t) \wedge i_2(t) = \overline{V0_1} \wedge Vt_2.$$

Finally,

$$f_2(t+2) = f_1(t+1) \wedge i_2(t+1) = \overline{Vt_1} \wedge Vt_2$$

For this circuit there are three possible transitions: that the INVERTER changes state from t to $t + 1$, that the AND gate changes state from t to $t + 1$, and that the AND gate changes state from $t + 1$ to $t + 2$. The Boolean equations for these transitions are respectively:

$$e_{1,1} = f_1(t) \oplus f_1(t+1)$$

$$e_{2,1} = f_2(t) \oplus f_2(t+1)$$

$$e_{2,2} = f_2(t+1) \oplus f_2(t+2).$$

As an aside, note that for this example, the two-vector sequence $V0_1 = 0$, $V0_2 = 0$, $Vt_1 = 1$, $Vt_2 = 1$, simultaneously satisfies $e_{1,1}$, $e_{2,1}$ and $e_{2,2}$.

3.1.3 General Delay Model

A gate with a large fanin may have several times the delay of an inverter. If one uses normalized time units, one can always introduce unit-delay buffers at the output of gates in a circuit, which have a delay greater than unity, in order to model differing delays among logic gates. Our symbolic simulator is able to simulate circuits with arbitrary gate transport delays without introducing any unit delay buffers in the circuit.

3.1.4 Pseudo-code for Symbolic Simulation

The pseudo-code for the symbolic simulation algorithm is presented in Figure 4. The simulator processes one gate at a time, moving from the primary inputs to the primary outputs of the circuit. For each gate g_i, an ordered list of the possible transition times of its inputs is first obtained. Then, possible transitions at the output of the gate are derived, taking into account transport delays from each input to the gate output. The processing done is similar to the "time-wheel" in a timing simulator.

3.2 Signal Probability Evaluation

Given the $f_i(t+j) \oplus f_i(t+j+1)$ functions, we can construct a disjoint cover for each of these single-output functions and compute the exact probability of a transition occurring between the boundary of $t+j$ and $t+j+1$.

Instead of enumerating disjoint covers, ordered Binary Decision Diagrams (BDDs) [?] can be used for the calculation of signal probability. It has been shown in [?] and [?] that exact signal probability calculation for a given function can be performed by a linear traversal of a BDD representation of a logic function. We have implemented methods for signal probability calculation using BDDs.

In some cases, the BDDs for the generated functions may be too large. The signal probability calculation can be done by a process of random logic simulation. For each Boolean function, $e_{i,j}$, a large number of randomly generated vectors are simulated on the network corresponding to the Boolean function till the signal probability value converges to within 0.1%. Levelized/event-driven simulation methods that simulate 32 vectors at a time can be used in an efficient probability evaluation scheme. The probabilities thus obtained are statistical approximations.

A final alternative is to use the cutting algorithm described in [?] (pp. 193-195), which computes signal probability bounds for outputs of logic gates in arbitrary combinational circuits, given input signal probabilities. The exact probability is guaranteed to lie within the computed bounds. The basic strategy requires a single traversal of the circuit from inputs to outputs and the computational complexity grows linearly with the size of the circuit. If one does not wish to underestimate the average-case power dissipation/switching activity, the upper bound on the signal probabilities of the generated Boolean functions evaluating to a 1 can be used. If one does not wish to overestimate the average-case switching activity, the lower bound on the signal probabilities of the generated

Boolean functions evaluating to a 1 can be used.

4 Extensions

We describe extensions to the basic symbolic simulation strategy of the previous section to handle transmission gates and inertial delays.

4.1 Transmission Gates

In this section we describe our power estimation scheme for the case of combinational circuits with embedded transmission gates. Symbolic simulation can be extended to estimate average switching activity in such circuits as described below.

Transmission gates have an input, an output, and a control line. When the control line is high, the outputs are identical to inputs. When the control line is low, however, the output is given by value stored in the previous time instant. It is this feature of having memory that makes transmission gate different from normal combinational gates like an AND gate. In mathematical terms, if a is the input, b the control, and x the output, then at any time instant t, the output of a transmission gate is given as

$$x(t) = b(t) \cdot a(t) + \overline{b(t)} \cdot x(t-1) \ , \tag{12}$$

where $t-1$ refers to the previous time instance.

From the switching activity estimation viewpoint, the symbolic simulation approach handles transmission gates (or transparent latches) in a straightforward manner. Since $x(t-1)$ is computed before $x(t)$ in the simulation, we create functions corresponding to the different $x(t)$'s and use them in simulating the fanout gates. We use the symbolic input $b(t)$ during symbolic simulation of $x(t)$. As the symbolic simulation proceeds, the known equations for the time points for each input are used and the logic equations corresponding to the various transitions at the output of the latch are computed. As a result, in a single pass from inputs to outputs, switching activity estimation can be carried out for an acyclic circuit.

If the initial value $x(0)$ is known it is replaced by the appropriate 0 or 1 value during symbolic simulation. If the initial value $x(0)$ is not known, it can be replaced by a Boolean variable with a signal probability of 0.5 during symbolic simulation.

4.2 Inertial Delays

Logic gates require energy to switch state. The energy in an input signal to a gate is a function of its amplitude and duration. If its duration is too small, the signal will not force the gate to switch. The minimum duration for which an input change must persist in order for the gate to switch states is called the **inertial delay** of an element and is denoted by δ (cf. [?], p. 187).

Inertial delay is usually modeled at the inputs to gates, however, for our purposes it is more convenient to model it at the gate output. We will assign an integer [3] $\delta_i \geq 0$ to each gate g_i and require that any pair of output transitions at g_i be separated by at least a duration δ_i.

The symbolic simulation proceeds assuming that $\delta_i = 0$ for all g_i as described in Section 3. The functions $f_i(t), \ldots,\ f_i(t+l)$ are computed. If the specified $\delta_i > 0$, then if there is a transition between time t and $t+1$ we cannot have a transition between $t+1$ and $t+\delta_i$. The functions that would be created for $\delta_i \leq 1$ are

$$e_{1,1} = f_1(t) \oplus f_1(t+1)$$

$$e_{1,2} = f_1(t+1) \oplus f_1(t+2)$$

$$e_{1,3} = f_1(t+2) \oplus f_1(t+3)$$

$$e_{1,4} = f_1(t+3) \oplus f_1(t+4)$$

and so on for each $e_{1,j}$. For $\delta_i = 2$, the functions created are

$$\begin{array}{ll} \alpha_{1,1} = e_{1,1} & \qquad E_{1,1} = \alpha_{1,1} \cdot \overline{e_{1,2}} \\ \alpha_{1,2} = \overline{\alpha_{1,1}} \cdot e_{1,2} & \qquad E_{1,2} = \alpha_{1,2} \cdot \overline{e_{1,3}} \\ \alpha_{1,3} = \overline{\alpha_{1,2}} \cdot e_{1,3} & \qquad E_{1,3} = \alpha_{1,3} \cdot \overline{e_{1,4}} \end{array}$$

and so on for each $E_{1,j}$. The $\alpha_{1,j}$ functions evaluate to 1 if the transition at time point j has not been canceled with the transition at time point $j-1$. This ensures that any pair of transitions are separated by at least 2 time units.

The signal probabilities of the $E_{1,j}$ are evaluated to obtain the switching activities.

5 Power Estimation Results

Throughout this section, we will be measuring the average power dissipation of the circuit by using Eqn. (1) summed over all the gates in the circuit. The $E(transitions)$ values are computed for the

[3] This number is obtained from process and device parameters like propagation delay.

gates in the circuit under different delay models. Since the circuits are technology-mapped circuits, the load capacitance values of the gates are known.

The statistics of the examples used are shown in Table 1. All of the examples except the last two belong to the ISCAS-89 Sequential Benchmark set. Example **add16** is a 16-bit adder and **max16** is a 16-bit maximum function.

All the circuits considered are technology-mapped static CMOS circuits. For all the circuits, we assumed uniform static (0.5) and transition (0.25) probabilities for the primary inputs. Note, however, that user-provided non-uniform probabilities could just as easily been used.

We focus on estimating switching acivity and power dissipation in the *combinational logic* of the given circuits. In Table 2, the effects of various delay models on the power estimate are illustrated. In the zero delay model, all gates have zero delay and therefore they switch instantaneously. In the unit delay model, all gates have one unit delay. Using the zero delay model ignores glitches in the circuit, and therefore power dissipation due to glitches is not taken into account. The unit delay model takes into account glitches, but a constant delay value is assumed for all gates. The variable delay model uses different delays for different gates, thus is the most realistic model.

Only the times required to obtain the power estimate for the variable delay model are shown in the last column. The variable delay computations are the most complex and therefore power estimation under this model takes the most time. The CPU times correspond to a DEC 3000/900 with 256Mb of memory, and are in seconds. The signal probability calculation was done using two different methods. The column BDD corresponds to exact signal probability evaluation of the $e_{i,j}$ functions of Section 3 using ordered Binary Decision Diagrams. The orderings were obtained automatically using a heuristic strategy. It is possible to speed up the power estimation by 2X to 5X by using hand orderings.

Using random logic simulation to evaluate signal probabilities required substantially less CPU time for the large examples as shown in the column LOGIC. Random logic simulation was carried out until the signal probability of each $e_{i,j}$ function converged to within 0.1%. This required the simulation of between 1000-50,000 vectors for the different examples.

The power measures obtained using the two methods BDD and LOGIC are identical. These estimates, in microWatts, have been obtained assuming a clock frequency of 20MHz and a supply voltage of 5V.

For each circuit, we compared the power estimates obtained by our method to that obtained by timing simulation. A power estimate was obtained by timing simulation using randomly generated input vectors. *The same power dissipation model and the same delay model was used in the timing simulation and in symbolic simulation.* In order to get within 2% of the BDD-based symbolic simulation estimate, timing simulation sometimes required substantially larger CPU times.

Note that under our delay model assumption it is the timing simulation estimate that is in error, since the symbolic simulation takes into account correlation in internal gates, and the BDD-based method computes exact signal probabilities. If random timing simulation is run for a very long time (e.g., 5 hours for the large circuits in Table 1) the results obtained become identical to the symbolic simulation results.

Timing simulation can easily take into account effects such as rise times and fall times. This is more more difficult to do in symbolic simulation, and increases the complexity of the approach.

6 Conclusions, Limitations and Ongoing Work

We have presented algorithms for probabilistically estimating the switching activity in combinational logic circuits, implemented either as dynamic or as static circuits. Results indicate that our algorithms are applicable to circuits of moderate size. The most desirable feature of our algorithm is that correlation in internal signals is taken into account under a variable delay model.

In order to perform exact signal probability evaluation, we use ordered Binary Decision Diagrams. Ordered BDDs cannot be built for large multipliers (≥ 16 bits) and for very large circuits. Approximate techniques (cf. Section 3.2) have to be used in these cases. Our experience with random logic simulation for signal probability evaluation has been favorable.

Given the delay model chosen, our BDD-based method of estimating switching activity is exact. The assumptions stated in Section 2 are required in order for the switching activity of a CMOS circuit to reflect the power dissipation of the circuit. These assumptions and the delay model assumptions can be relaxed, but this will result in more complex symbolic simulation methods. For example, rise and fall times can be modeled by using a multiple-valued algebra. Under the variable delay model, BDD-based symbolic simulation is usually significantly more efficient as well as being more accurate than random timing simulation. However, this is not true for more sophisticated delay models.

Correlation between primary inputs exists when a given combinational circuit is embedded in a larger sequential circuit. The techniques described have to be augmented to handle sequential circuits and primary input correlation, and efforts in this direction are presented in [**?**, **?**].

7 Acknowledgements

We thank Mike Bershetyn, Jan Rabaey and S. Murai for helpful comments and suggestions and many interesting discussions on power estimation. This work was supported in part by the Defense Advanced Research Projects Agency under contract N00014-91-J-1698, and in part by NSF under the NYI program with matching funds from Mitsubishi and IBM corporation.

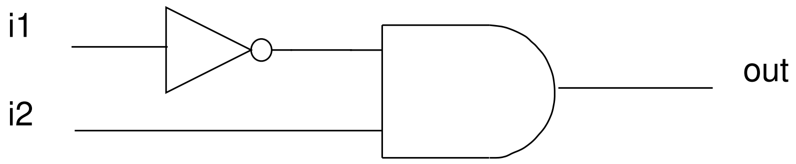

Figure 1: Glitching in a Static CMOS Circuit.

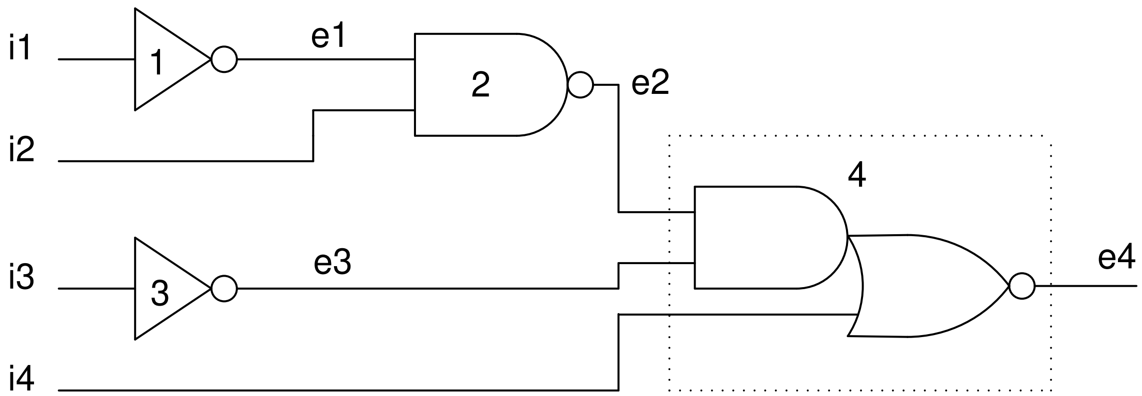

Figure 2: A Multilevel Combinational Circuit.

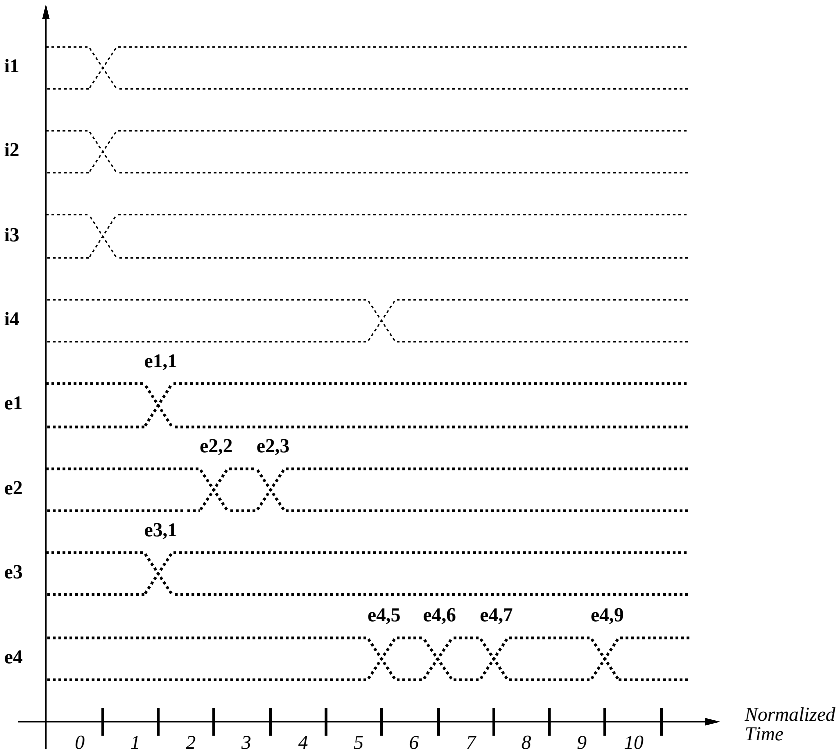

Figure 3: Signal Waveforms for the Primary Inputs and the Outputs of the Logic Gates.

```
Gates = Topological_Sort( Network ) ;
for each $g_i$ in Gates {
    $\Delta$ = delay of $g_i$ ;
    TimePoints = NIL(LIST) ;
    for each input $g_j$ of $g_i$ ( $g_{i_1}, \cdots, g_{i_m}$ ) {
        if $g_j$ is a primary input
            then TimePoints = InsertInOrder ( TimePoints, { $(0, V0_j)$, $(t, Vt_j)$ } ) ;
            else for each time point $(k, f_j(k))$ of $g_j$ {
                    TimePoints = InsertInOrder ( TimePoints, $(k, f_j(k))$ ) ;
                }
    }
    /* $\tilde{g}_i$ is the Boolean function of gate $g_i$ with respect to its immediate inputs */
    $f_i(0) = \tilde{g}_i(f_{i_1}(0), \cdots, f_{i_m}(0))$ ;
    $l = 0$ ;
    for each new time point $k$ in TimePoints {
        $f_i(k + \Delta) = \tilde{g}_i(f_{i_1}(k), \cdots, f_{i_m}(k))$ ;
        $e_{i,k+\Delta} = f_l \oplus f_i(k + \Delta)$ ;
        $l = k + \Delta$ ;
    }
}
```

Figure 4: Pseudo-code for the Symbolic Simulation Algorithm.

CKT	Inputs	Outputs	Latches	Gates
s27	4	1	3	10
s298	3	6	14	119
s349	9	11	15	150
s386	7	7	6	159
s420	19	2	16	196
s510	19	7	6	211
s641	35	24	19	379
s713	35	23	19	393
s838	35	2	32	390
s1238	14	14	18	508
s1494	8	19	6	647
add16	33	17	16	288
max16	33	16	16	154

Table 1: Statistics of Examples.

CKT	Zero Delay Power	Unit Delay Power	Variable Delay Power	Time	
				BDD	LOGIC
s27	82	93	93	0.1	0.2
s298	922	1033	1069	5.2	2.3
s349	777	1094	1110	9.7	6.1
s386	1070	1183	1250	9.2	4.9
s420	877	940	958	12.0	5.2
s510	993	1236	1331	11.2	5.5
s641	1228	1594	1665	62.6	36.3
s713	1338	1847	1932	151.6	92.6
s838	1727	1822	1847	52.6	16.9
s1238	2394	3013	3158	115.1	43.7
s1494	3808	4762	5045	68.9	32.2
add16	1258	1725	1741	10.4	6.1
max16	599	713	713	4.2	1.6

Table 2: Power Estimation for Combinational Logic.