Power Estimation Methods for Sequential Logic Circuits

Chi-Ying Tsui, José Monteiro, Massoud Pedram, *Member, IEEE,* Srinivas Devadas, *Member, IEEE,* Alvin M. Despain, *Member, IEEE,* and Bill Lin

Abstract— **Recently developed methods for power estimation have primarily focused on combinational logic. We present a framework for the efficient and accurate estimation of average power dissipation in sequential circuits.**

Switching activity is the primary cause of power dissipation in CMOS circuits. Accurate switching activity estimation for sequential circuits is considerably more difficult than that for combinational circuits, because the probability of the circuit being in each of its possible states has to be calculated. The Chapman–Kolmogorov equations can be used to compute the exact state probabilities in steady state. However, this method requires the solution of a linear system of equations of size 2^N where N is the number of flip-flops in the machine.

We describe a comprehensive framework for exact and approximate switching activity estimation in a sequential circuit. The basic computation step is the solution of a nonlinear system of equations which is derived directly from a logic realization of the sequential machine. Increasing the number of variables or the number of equations in the system results in increased accuracy. For a wide variety of examples, we show that the approximation scheme is within 1–3% of the exact method, but is orders of magnitude faster for large circuits. Previous sequential switching activity estimation methods can have significantly greater inaccuracies.

I. 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 applications such as personal communication systems and 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 multichip modules where heat dissipation is one of the biggest problems.

Power dissipation of a circuit, like its area or speed, may be significantly improved by changing the circuit architecture or the base technology [3]. However, once these architectural or technological improvements have been made, it is the switching of the logic that will ultimately determine the power dissipation.

Methods for the power estimation of logic-level combinational circuits based on switching activity estimation have been presented previously (e.g., [2], [4], [7], [9], [10], [13]). Power and switching activity estimation for sequential circuits is significantly more difficult, because the probability of the circuit being in any of its possible states has to be computed. Given a circuit with N flip-flops, there are 2^N possible states. These state probabilities are, in general, not uniform. As an example, consider the sequential circuit of Fig. 1 and the example State Transition Graph of Fig. 2. Assuming that the circuit was in state **R** at time 0, and that at each clock cycle random inputs are applied, at time ∞ (i.e., steady state) the probabilities of the circuit being in state **R**, **A**, **B**, **C** are $\frac{1}{6}$, $\frac{1}{3}$, $\frac{1}{4}$, and $\frac{1}{4}$, respectively. These *state probabilities* have to be taken into account during switching activity estimation of the combinational logic part of the machine. Power dissipation and switching activity of CMOS combinational logic are modeled by randomly applied vector pairs. In the case of sequential circuits, the vector pair $\langle v_1, v_2 \rangle$ applied to the combinational logic is composed of a primary input part and a present state part (see Fig. 1), namely $\langle i_1@s_1, i_2@s_2 \rangle$. Given $i_1@s_1$, the next state s_2 is uniquely determined given the functionality of the combinational logic. For example, if i_1 happens to be 0 and the machine of Fig. 2 is in state **R**, the machine will move to state **B**. This *correlation* between the applied vector pairs has to be taken into account in order to obtain accurate estimates of the switching activity in sequential circuits.

A first attempt at estimating switching activity in logic-level sequential circuits was presented in [4]. This method can accurately model the correlation between the applied vector pairs, but assumes that the state probabilities are all uniform. Extensions of this method can produce accurate estimates for acyclic sequential circuits such as pipelines, but not for more general cyclic circuits [8].

Manuscript received June 15, 1994; revised February 20, 1995 and March 31, 1995. The work of C.-Y. Tsui and A. M. Despain was supported in part by the Advanced Research Projects Agency under Contract J-FBI-91-194. The work of M. Pedram was supported in part by the Advanced Research Projects Agency under Contract F33615-95-C-1627, and by the SRC under Contract 94-DJ-559. The work of J. Monteiro's and S. Devadas was supported in part by the Advanced Research Projects Agency under Contract DABT63-94-C-0053, and in part by a NSF Young Investigator Award with matching funds from Mitsubishi Corporation.

C.-Y. Tsui is with the Department of Electrical Engineering, Hong Kong University of Science and Technology, Hong Kong.

M. Pedram and A. Despain are with the Department of Electrical Engineering, University of Southern California, Los Angeles, CA 90089 USA.

J. Monteiro and S. Devadas are with the Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139 USA.

B. Lin is with IMEC, Belgium, France.

IEEE Log Number 9413459.

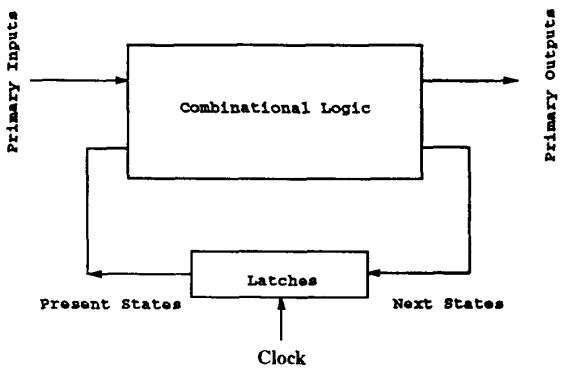


Fig. 1. A synchronous sequential circuit.

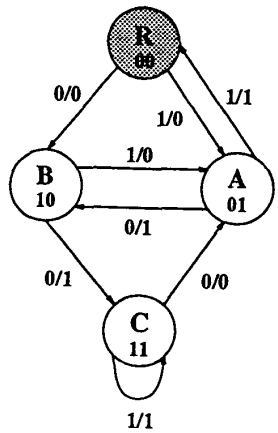


Fig. 2. Example state transition graph.

In this paper, we present results obtained by using the Chapman–Kolmogorov equations for discrete-time Markov Chains [12] to compute the exact state probabilities of the machine. The Chapman–Kolmogorov method requires the solution of a linear system of equations of size 2^N, where N is the number of flip-flops in the machine. Thus, this method is limited to circuits with relatively small number of flip-flops, since it requires the explicit consideration of each state in the circuit.

We next describe an approximate method for switching activity estimation in sequential circuits. The basic computation step is the solution of a nonlinear system of equations which is derived directly from the logic realization of the next state logic of the machine under consideration. Increasing the number of variables or the number of equations in the system results in increased accuracy. For a wide variety of examples, we show that the approximation scheme is within 1–3% of the exact method, but is orders of magnitude faster for large circuits. Previous sequential switching activity estimation methods can have significantly greater inaccuracies.

The rest of this paper is organized as follows. In Section II we briefly review the physical model for power estimation and summarize the combinational estimation method of [4]. In Section III, we describe an exact switching activity estimation method for sequential circuits. In Section IV, we first provide the basis for the approximation schemes we have developed and formulate the problem of estimating switching activity as that of solving a nonlinear system of equations. We describe a scheme based on the notion of a k-unrolled network that can be used to improve the accuracy of estimation in Section V. We describe a different method to improve the accuracy based on the notion of a m-expanded network in Section VI. In Section VII we describe methods to solve the nonlinear system of equations, namely, the Picard–Peano and the Newton–Raphson methods. In Section VIII, we show that purely combinational logic estimation methods can provide inaccurate estimates, whereas the developed approximation methods produce accurate estimates while being applicable to large circuits.

II. Preliminaries

A. A Power Dissipation Model

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 is at the output node of a CMOS gate (this capacitance includes the source–drain capacitance of the gate itself and the input capacitances of the fanout gates).
- Current is flowing either from V_{DD} to the output capacitor or from the output capacitor to ground (that is, there is no short-circuit current).
- Any change in a logic-gate output voltage is a change from V_{DD} to ground or vice-versa (that is, there are no stable intermediate voltage levels).

These assumptions are reasonably justified for well-designed CMOS gates [5] 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 output capacitance seen by the gate. 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:

$$P_{avg} = 0.5 \times C_{load} \times \left(\frac{V_{dd}^2}{T_{cyc}}\right) \times E(transitions) \qquad (1)$$

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 [9], 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 the logic function being performed and the statistical properties of the primary inputs.

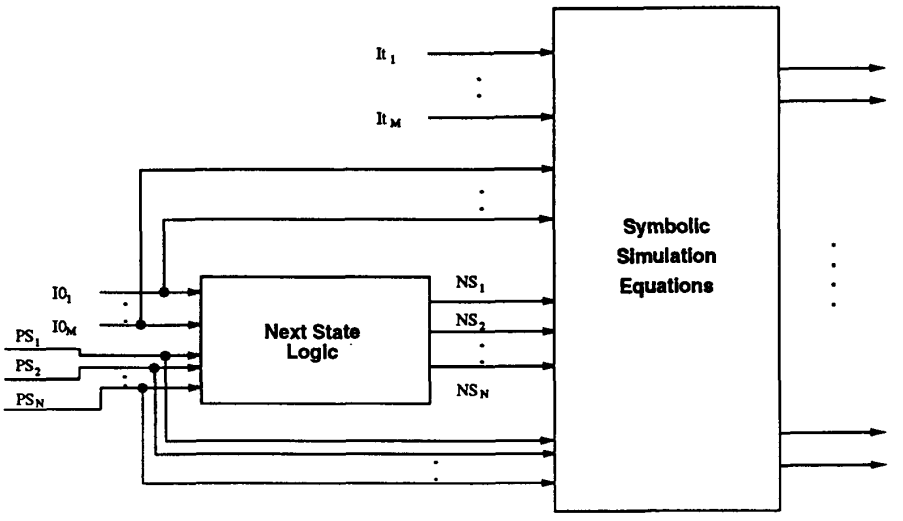


Fig. 3. Taking correlation into account.

Equation (1) is used by the power estimation techniques such as [4], [9] to relate switching activity to power dissipation.

B. Combinational Circuits

Average power can be estimated for combinational circuits by computing the average switching activity at every gate in the circuit.

It is assumed that we are given *transition probabilities* at each of the primary inputs to the circuit. That is, for every primary input the probability of the primary input staying at $0\,(0 \rightarrow 0)$, staying at $1\,(1 \rightarrow 1)$, making a $0 \rightarrow 1$ transition and making a $1 \rightarrow 0$ transition are given. Given these probabilities, the average switching activity at each gate in the circuit can be calculated.

A symbolic simulation method that performs this computation was given in [4]. Under the chosen gate delay model, the method first constructs a Boolean function representing the logical value at any gate output at each time point $\geq t$ based on the primary input variables $I0$ applied at time 0 and It applied at time t. For instance, one may compute the functions $f_i(t+1)$ and $f_i(t+2)$ for a particular gate g_i. The Boolean conditions at the inputs that correspond to a $0 \rightarrow 1$ transition on g_i between times $t+1$ and $t+2$ are represented by the function $\overline{f_i(t+1)} \cdot f_i(t+2)$. The probability of a $0 \rightarrow 1$ transition occurring between time $t+1$ and $t+2$ given the transition probabilities at the primary inputs is the probability of the Boolean function $\overline{f_i(t+1)} \cdot f_i(t+2)$ evaluating to a 1. (This probability can be evaluated exactly using Binary Decision Diagrams [1] or approximately using Monte Carlo simulation.) For each gate, probabilities of transitions occurring at any time point can be evaluated efficiently, and these probabilities are summed over all the time points to obtain the average switching activity (at each gate).

Under the zero delay, unit delay, or a general delay model (where delays are obtained from library cells), the symbolic simulation method takes into account the correlation due to reconvergence of input signals and accurately measures switching activity.

The same computation can be performed more efficiently, although not exactly, using probabilistic simulation techniques such as [10] and [13] or Monte-Carlo simulation [2]. In the remainder of this paper, whenever we need to perform the above computation, we will refer to the symbolic simulation equations (which provide the exact solution). It should however be made clear that any other solution technique (probabilistic simulation, Monte-Carlo simulation, etc.) can be used instead.

III. The Exact Method

A. Modeling Correlation

To model the correlation between the two vectors in a randomly applied vector pair, we have to augment the combinational estimation method described in Section II-B. This augmentation is summarized in Fig. 3.

In Fig. 3, we have a block corresponding to the symbolic simulation equations for the combinational logic of the general sequential circuit shown in Fig. 1. The symbolic simulation equations have two sets of inputs, namely $\langle I0, It\rangle$ for the primary inputs and $\langle PS, NS\rangle$ for the present state lines. However, given $I0$ and PS, NS is uniquely determined by the functionality of the combinational logic. This is modeled by prepending the next state logic to the symbolic simulation equations.

The configuration of Fig. 3 implies that the gate output switching activity can be determined given the vector pair $\langle I0, It\rangle$ for the primary inputs, but only PS for the state lines. Therefore, to compute gate output transition probabilities, we require the transition probabilities for the primary input lines, and the static probabilities for the present state lines. This configuration was originally proposed in [4].

B. State Probability Computation

The static probabilities for the present state lines marked PS in Fig. 3 are spatially correlated. We therefore require knowledge of *present state probabilities* as opposed to present state line (PS) probabilities in order to exactly calculate the switching activity in the sequential machine. The state probabilities are dependent on the connectivity of the State Transition Graph (STG) of the circuit.

For each state s_i, $1 \leq i \leq K$ in the STG, we associate a variable $prob(s_i)$ corresponding to the steady-state probability of the machine being in state s_i at $t = \infty$. For each edge e in the STG, we have $e.Current$ signifying the state that the edge fans out from, $e.Next$ signifying the state that the edge fans out to, and $e.Input$ signifying the input combination corresponding to the edge. Given static probabilities for the primary inputs to the machine, we can compute $prob(Input)$, the probability of the combination $Input$ occurring.[1] We can compute $prob(e.Input)$ using:

$$prob(e.Input) = prob(e.Current) \times prob(Input)$$

For each state s_i we can write an equation:

$$prob(s_i) = \sum_{\forall e \; such \; that \; e.Next = s_i} prob(e.Input)$$

Given K states, we obtain K equations out of which any one equation can be derived from the remaining $K-1$ equations. We have a final equation:

$$\sum_{i=1}^{K} prob(s_i) = 1.$$

This linear set of K equations can be solved to obtain the different $prob(s_i)$'s.

This system of equations is known as the Chapman–Kolmogorov equations for a discrete-time discrete-transition Markov process. Indeed, if the Markov process satisfies the conditions that it has a finite number of states, its essential states form a single-chain and it contains no periodic-states, then the above system of equations will have a unique solution [12].

For example, for the State Transition Graph of Fig. 2 we will obtain the following equations assuming a probability of 0.5 for the primary input being a 1.

$$\begin{aligned} prob(\mathbf{R}) &= 0.5 \times prob(\mathbf{A}) \\ prob(\mathbf{A}) &= 0.5 \times prob(\mathbf{R}) \\ &\quad + 0.5 \times prob(\mathbf{B}) \\ &\quad + 0.5 \times prob(\mathbf{C}) \\ prob(\mathbf{B}) &= 0.5 \times prob(\mathbf{R}) \\ &\quad + 0.5 \times prob(\mathbf{A}). \end{aligned}$$

The final equation is:

$$prob(\mathbf{R}) + prob(\mathbf{A}) + prob(\mathbf{B}) + prob(\mathbf{C}) = 1.$$

Solving this linear system of equations results in the state probabilities, $prob(\mathbf{R}) = \frac{1}{6}$, $prob(\mathbf{A}) = \frac{1}{3}$, $prob(\mathbf{B}) = \frac{1}{4}$, and $prob(\mathbf{C}) = \frac{1}{4}$.

[1] Static probabilities can be computed from specified transition probabilities.

C. Power Estimation Given Exact State Probabilities

We now describe a power estimation method that utilizes the exact state probabilities obtained using the Chapman–Kolmogorov method. As described in Section II-B, the symbolic equations express the exact switching conditions for each gate in the circuit under the unit or general delay models. Prepending the next state logic block as illustrated in Fig. 3 accounts for the correlation between the present and next states. Finally, computing the exact state probabilities models the steady-state behavior of the circuit.

As described in Section II-B, power estimation of a given combinational logic circuit can be carried out by creating a set of symbolic functions such that summing the signal probabilities of the functions corresponds to the average switching activity in the original combinational circuit. Some of the inputs to the created symbolic functions are the present state lines of the circuit and the others are primary input lines. Each binary combination of the present state lines is a state in the circuit and we have a number corresponding to the state probability for each state after solving the Chapman–Kolmogorov equations.

The signal probability calculation procedure has to appropriately weight these combinations according to the given probabilities. Suppose n is a disjoint cover of the function f, i.e.,

$$f = \bigvee_{m \in \, Disjoint\_Cover(n)} C_m \tag{2}$$

where the C_m's are cubes of the disjoint cover. Each C_m is a function of the present state lines and primary inputs. We partition the inputs to C_m into two groups: the symbolic state support SS_m which includes all states s_i that have set the appropriate state bits, and the primary input support I_m which includes the PI inputs of C_m. Hence $C_m = SS_m I_m$. The signal probability of n is thus given by:

$$prob(n) = \sum_{m \in \, Disjoint\_Cover(n)} prob(C_m). \tag{3}$$

Since the primary inputs are independent of the state that the machine is currently in and states of the FSM are distinct, we can write

$$\begin{aligned} prob(C_m) &= prob(I_m) prob(SS_m) \\ &= prob(I_m) \sum_{s_i \in SS_m} prob(s_i). \end{aligned} \tag{4}$$

From (3) and (4), we have:

$$prob(n) = \sum_{m \in \, Disjoint\_Cover(n)} prob(I_m) \sum_{s_i \in SS_m} prob(s_i). \tag{5}$$

As an example, consider the following disjoint cover of a function whose signal probability is to be computed.

$$f = i_1 \wedge ps_1 \vee i_1 \wedge \overline{ps_1} \wedge ps_2.$$

Assume that the probability of i_1 being a 1 is 0.5, and state probabilities are $prob(00) = \frac{1}{6}$, $prob(01) = \frac{1}{3}$, $prob(10) = \frac{1}{4}$ and $prob(11) = \frac{1}{4}$. (The first bit corresponds to ps_1 and the second to ps_2.) The probability of the first cube is

$$\begin{aligned} prob(i_1 \wedge ps_1) &= prob(i_1) \times [prob(10) + prob(11)] \\ &= 0.5 \times (\tfrac{1}{4} + \tfrac{1}{4}) \\ &= \tfrac{1}{4}. \end{aligned}$$

Similarly the probability of the second cube is:

$$\begin{aligned} prob(i_1 \wedge \overline{ps_1} \wedge ps_2) &= prob(i_1) \times prob(01) \\ &= 0.5 \times \tfrac{1}{3} \\ &= \tfrac{1}{6}. \end{aligned}$$

Finally we have:

$$prob(n) = \tfrac{1}{4} + \tfrac{1}{6} = \tfrac{5}{12}.$$

Note that (5) requires explicit enumeration of the states and is very costly. In [14], a method which employs a partially implicit enumeration of states using OBDDs is described. The estimation method still has *average-case* exponential complexity—the probability of each state (respectively, groups of states) is computed, and the number of states (respectively, such groups) can be exponential in the number of flip-flops in the circuit. However, for the circuits that this method is applicable to, the estimates provided by the method can serve as a basis for comparison among different approximation schemes.

IV. Basis of Approximation Strategies

Consider a machine with two flip-flops whose states are 00, 01, 10, and 11 have state probabilities $prob(00) = \frac{1}{6}$, $prob(01) = \frac{1}{3}$, $prob(10) = \frac{1}{4}$ and $prob(11) = \frac{1}{4}$. We can calculate the present state line probabilities as shown below, where ps_1 and ps_2 are the first and second present state lines.

$$\begin{aligned} prob(ps_1 = 0) &= prob(00) + prob(01) \\ &= \tfrac{1}{6} + \tfrac{1}{3} = \tfrac{1}{2} \\ prob(ps_1 = 1) &= prob(10) + prob(11) \\ &= \tfrac{1}{4} + \tfrac{1}{4} = \tfrac{1}{2} \\ prob(ps_2 = 0) &= prob(00) + prob(10) \\ &= \tfrac{1}{6} + \tfrac{1}{4} = \tfrac{5}{12} \\ prob(ps_2 = 1) &= prob(01) + prob(11) \\ &= \tfrac{1}{3} + \tfrac{1}{4} = \tfrac{7}{12}. \end{aligned}$$

Note that because ps_1 and ps_2 are correlated, $prob(ps_1 = 0) \times prob(ps_2 = 0) = \frac{5}{24}$ is not equal to $prob(00) = \frac{1}{6}$.

We carried out the following experiment on 52 sequential circuit benchmark examples for which the exact state probabilities could be calculated. These benchmarks included finite state machine controllers, datapaths[2] as well as pipelines. First, the power dissipation of the circuit was calculated using the exact state probabilities as described in Section III-C. Next, given the exact state probabilities, the line probabilities were determined as described in the previous paragraph. Using the topology of Fig. 3 and the computed present state line probabilities for the PS lines, approximate power dissipations were calculated for each circuit. The average error[3] in the power dissipation measures obtained using the line probability approximation over all the circuits was only 2.8%. The maximum error for any one example was 7.3%. Assuming uniform line probabilities of 0.5 as in [4] results in significant errors of over 40% for some examples.

The above experiment leads us to conclude that if accurate line probabilities can be determined then using line probabilities rather than state probabilities is a viable alternative. We only have to determine N numbers for a N flip-flop machine, one for each present state line, rather than 2^N numbers, one for each possible state.

A. Computing Present State Line Probabilities

In our approximation framework we *directly determine line probabilities without recourse to State Transition Graph extraction*. The approximation framework is based on solving a nonlinear system of equations to compute the state line probabilities. This system of equations is given by the combinational logic implementing the next state function of the sequential circuit.

Consider the set of functions below corresponding to the next state lines.

$$\begin{aligned} ns_1 &= f_1(i_1, i_2, \cdots, i_M, ps_1, ps_2, \cdots, ps_N) \\ ns_2 &= f_2(i_1, i_2, \cdots, i_M, ps_1, ps_2, \cdots, ps_N) \\ &\cdots \\ ns_N &= f_N(i_1, i_2, \cdots, i_M, ps_1, ps_2, \cdots, ps_N) \end{aligned}$$

We can write:

$$\begin{aligned} prob(ns_1) &= prob[f_1(i_1, i_2, \cdots, i_M, ps_1, ps_2, \cdots, ps_N)] \\ prob(ns_2) &= prob[f_2(i_1, i_2, \cdots, i_M, ps_1, ps_2, \cdots, ps_N)] \\ &\cdots \\ prob(ns_N) &= prob[f_N(i_1, i_2, \cdots, i_M, ps_1, ps_2, \cdots, ps_N)] \end{aligned}$$

where $prob(ns_i)$ corresponds to the probability that ns_i is a 1, and $prob[f_i(i_1, i_2, \cdots, i_M, ps_1, ps_2, \cdots, ps_N)]$ corresponds to the probability that $f_i(i_1, i_2, \cdots, i_M, ps_1, ps_2, \cdots, ps_N)$ is a 1, which is of course dependent on the $prob(ps_j)$ and the $prob(i_k)$.

We are interested in the steady state probabilities of the present and next state lines implying that:

$$prob(ps_i) = prob(ns_i) = p_i \quad 1 \le i \le N.$$

A similar relationship was used in the Chapman–Kolmogorov (cf. Section III).

[2] We were restricted to 8-bit datapaths since the state probability computation requires explicitly enumerating the states of the machine.

[3] This error is caused by ignoring the correlation between the present state lines.

The set of equations given the values of $prob(i_k)$ becomes:

$$\begin{aligned} y_1 &= p_1 - g_1(p_1, p_2, \cdots, p_N) = 0 \\ y_2 &= p_2 - g_2(p_1, p_2, \cdots, p_N) = 0 \\ &\cdots \\ y_N &= p_N - g_N(p_1, p_2, \cdots, p_N) = 0 \end{aligned} \tag{6}$$

where the g_i's are nonlinear functions of the p_i's. We will denote the above equations as $Y(P) = 0$ or as $P = G(P)$. In general the Boolean function f_i can be written as a list of minterms over the i_k and ps_j and the corresponding g_i function can be easily derived. For example, given

$$f_1 = i_1 \wedge ps_1 \wedge \overline{ps_2} \vee i_1 \wedge \overline{ps_1} \wedge ps_2$$

and $prob(i_1) = 0.5$, we have

$$g_1 = 0.5 \cdot [p_1 \cdot (1 - p_2) + (1 - p_1) \cdot p_2]. \tag{7}$$

We can solve the equation set $Y(P) = 0$ or find a fixed point of $P = G(P)$ to obtain the present state line probabilities. We describe the use of the Picard–Peano method to obtain a fixed point of $P = G(P)$, and the use of the Newton–Raphson method to solve $Y(P) = 0$ in Section VII. The uniqueness or the existence of the solution is not guaranteed for an arbitrary system of nonlinear equations. However, since in our application we have a correspondence between the nonlinear system of equations and the State Transition Graph of the sequential circuit, there will exist at least one solution to the nonlinear system. Further, convergence is guaranteed under mild assumptions for our application.

B. Inaccuracy in Formulation

The above formulation does not capture the correlation between the state line probabilities. Let us consider the example State Transition Graph of Fig. 2. The equations for the next state logic are:

$$\begin{aligned} ns_1 &= i \cdot ps_1 \cdot ps_2 + \bar{i} \cdot \overline{ps_1} + \bar{i} \cdot ps_1 \overline{ps_2} \\ ns_2 &= ps_1 + i \cdot \overline{ps_1} \cdot \overline{ps_2}. \end{aligned}$$

Assuming the probability of input i being a 1 is 0.5 we obtain the nonlinear equations (after simplification):

$$\begin{aligned} n_1 &= 0.5 - 0.5p_1 - 0.5p_2 \\ n_2 &= p_1 + 0.5(1 - p_1)(1 - p_2). \end{aligned}$$

Setting $n_1 = p_1$ and $n_2 = p_2$ and solving the above equations gives us $p_1 = 0.191$ and $p_2 = 0.424$. However, if we obtain the exact line probabilities using the exact state probabilities as shown in the first paragraph of Section IV, we find that these approximate line probabilities are in error.

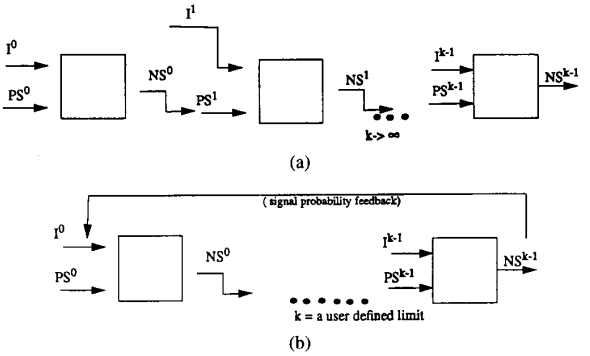


Fig. 4. k-unrolling of the next state logic.

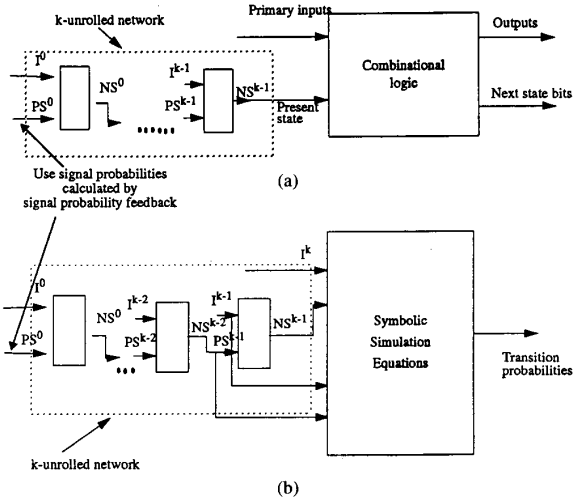


Fig. 5. Calculation of signal and transition probabilities by network unrolling.

The above example is small (4 states) and contrived, and significant errors may be obtained for such examples. The state line probabilities obtained using the approximation method of this section are on average close to the exact line probabilities, and they typically result in switching activity estimates that are close to the exact method for most real-life examples (cf. Section VIII). Nevertheless, it is worthwhile to explore ways to increasing the accuracy. We describe two such mechanisms in Section V and Section VI.

V. Improving Accuracy Using k Unrolled Networks

A. State Line Probability Computation

In the formulation of Section IV, the nonlinear equations correspond to a single stage of next state logic. Consider the *unrolled* network of Fig. 4(a). The next state logic has been unrolled k times. As illustrated in Fig. 4(b), we can construct a set of nonlinear equations corresponding to this k-unrolled network, which will partially take into account the

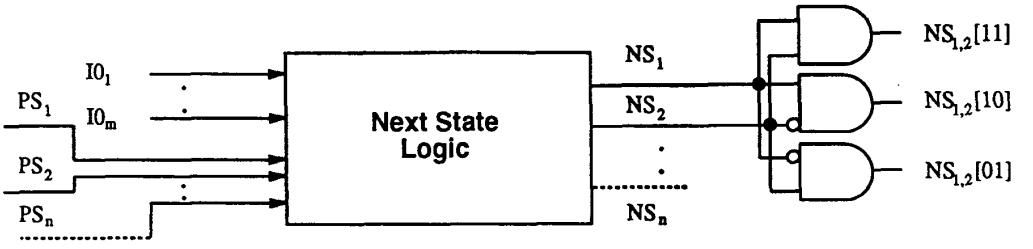


Fig. 6. An m-Expanded network with $m = 2$.

correlation between the state lines, when computing the state line probabilities.

The exact present state line probabilities can be obtained by unrolling the next state logic ∞ times (Fig. 4(a)). This is however impractical. We thus approximate the signal probabilities by unrolling the next state logic k times where k is a user defined parameter.

The equations corresponding to $k = 2$ will be:

$$\begin{aligned} ns_1^1 &= f_1(i_1^1, \cdots, i_M^1, ps_1^1, \cdots, ps_N^1) \\ &= f_1(i_1^1, \cdots, i_M^1, ns_1^0, \cdots, ns_N^0) \\ &= f_1[i_1^1, \cdots, i_M^1, f_1(i_1^0, \cdots, i_M^0, ps_1^0, \cdots, ps_N^0), \\ &\quad \cdots, f_N(i_1^0, \cdots, i_M^0, ps_1^0, \cdots, ps_N^0)] \\ &\quad \cdots \\ ns_N^1 &= f_N[i_1^1, \cdots, i_M^1, f_1(i_1^0, \cdots, i_M^0, ps_1^0, \cdots, ps_N^0), \\ &\quad \cdots, f_N(i_1^0, \cdots, i_M^0, ps_1^0, \cdots, ps_N^0)]. \end{aligned}$$

The number of equations is the same. The number of primary input variables has increased, but the probabilities for these variables are known.

Fig. 5(a) shows the method used to calculate signal probability of the internal nodes of the FSM using the k-unrolled network with signal probability feedback.

B. Switching Activity Computation

The topology of Fig. 3 was proposed as a means of taking into account the correlation between the applied input vector pair when computing the transition probabilities. This method takes one cycle of correlation into account.

It is possible to take multiple cycles of correlation into account by prepending the symbolic simulation equations with the k-unrolled network. This is illustrated in Fig. 5(b). Instead of connecting the next state logic network to the symbolic simulation equations, we unroll the next state logic network k times and connect the next state lines of the kth stage of the unrolled network, the next state lines of the $(k-1)$th stage, and the primary input of the $(k-1)$th stage to the symbolic simulation equations.

VI. Improving Accuracy Using m-Expanded Networks

A. State Line Probability Computation

We describe a different method to improve the accuracy of the basic approximation strategy outlined in Section IV. This method models the correlation between m-tuples of present state lines. The method is pictorially illustrated in Fig. 6 for $m = 2$.

The number of equations in the case of $m = 2$ is $3N/2$. We have:

$$\begin{aligned} ns_{i,i+1}[11] &= ns_i \wedge ns_{i+1} = f_i \wedge f_{i+1} \\ ns_{i,i+1}[10] &= ns_i \wedge \overline{ns_{i+1}} = f_i \wedge \overline{f_{i+1}} \\ ns_{i,i+1}[01] &= \overline{ns_i} \wedge ns_{i+1} = \overline{f_i} \wedge f_{i+1}. \end{aligned}$$

We have to solve for $prob(ns_{i,i+1}[11])$, $prob(ns_{i,i+1}[10])$, and $prob(ns_{i,i+1}[01])$ [rather than $prob(ns_i)$ and $prob(ns_{i+1})$ as in the case of $m = 1$). We use:

$$\begin{aligned} prob(ps_i \wedge ps_{i+1}) &= prob(ns_{i,i+1}[11]) \\ prob(ps_i \wedge \overline{ps_{i+1}}) &= prob(ns_{i,i+1}[10]) \\ prob(\overline{ps_i} \wedge ps_{i+1}) &= prob(ns_{i,i+1}[01]) \end{aligned}$$

in the evaluation of the $prob(f_i)$'s.

The signal probability evaluation methods of Section VII-C can be easily augmented to use the above probabilities. In the case of the OBDD-based method placing each ps_i and ps_{i+1} pair adjacent in the chosen ordering allows signal probability computation by a linear-time traversal.

The number of equations for $m = 3$ is $7N/3$. When $m = N$, the number of equations will become 2^N and the method will degenerate to the Chapman–Kolmogorov method.

The choice of the m-tuples of present and next state lines is made by grouping next state lines that have the maximal amount of shared logic into each m-tuple. Note that the accuracy of line probability estimation will depend on the choice of the m-tuples.

B. Switching Activity Computation

To estimate switching activity given m-tuple present state line probabilities, the topology of Fig. 3 is used as before. The difference is that for $m = 2$ the $prob(ps_i \wedge ps_{i+1})$, $prob(ps_i \wedge \overline{ps_{i+1}})$ and $prob(\overline{ps_i} \wedge ps_{i+1})$ values are used to calculate the switching activities.

VII. Solving the Nonlinear System of Equations

We describe two methods to solve the nonlinear system of equations obtained using k-unrolled or m-expanded networks. We will assume that the nonlinear system can be represented as $P = G(P)$ or as $Y(P) = 0$ as described in Section IV.

A. Picard–Peano Method

The Picard–Peano method is used to find a fixed point of the $P = G(P)$ system. This system is reproduced below.

$$\begin{aligned} p_1 &= g_1(p_1, p_2, \cdots, p_N) \\ p_2 &= g_2(p_1, p_2, \cdots, p_N) \\ &\cdots \\ p_N &= g_N(p_1, p_2, \cdots, p_N). \end{aligned}$$

We can start with an initial guess P^0, and iteratively compute $P^{k+1} = G(P^k)$ until convergence is reached. Convergence is deemed to be achieved if $P^{k+1} - P^k$ is sufficiently small. The above iteration is known as the Picard–Peano iteration for finding a fixed-point of a system of nonlinear equations.

We are only given the Boolean functions $f_i(i_1, i_2, \cdots, i_M, ps_1, ps_2, \cdots, ps_N)$. There exist several methods to compute $g_i(p_1, p_2, \cdots, p_N) = prob[f_i(i_1, i_2, \cdots, i_M, ps_1, ps_2, \cdots, ps_N)]$ for given $p_j = prob(ps_j)$'s and $prob(i_k)$'s. We describe these methods in Section VII-C.

Theorem 7.1: [6] If G is contractive, i.e., $|\partial g_i/\partial p_j| < 1$, for all i, j, then the Picard–Peano iteration method converges at least linearly to a unique solution P^*.

Theorem 7.2: If each next state line is a nontrivial logic function of at least two present state lines, then g_i is contractive on the domain (0, 1).

Proof: Choose any p_j. In order to perform the evaluation of $\partial g_i/\partial p_j$ we cofactor f_i with respect to ps_j.

$$f_i = ps_j \wedge f_{i\,ps_j} \vee \overline{ps_j} \wedge f_{i\,\overline{ps_j}}$$

$f_{i\,ps_j}$ and $f_{i\,\overline{ps_j}}$ are the cofactors of f with respect to ps_j, and are Boolean functions independent of ps_j. We can write:

$$g_i = p_j \cdot prob(f_{i\,ps_j}) + (1 - p_j) \cdot prob(f_{i\,\overline{ps_j}}).$$

Differentiating with respect to p_j gives:

$$\frac{\partial g_i}{\partial p_j} = prob(f_{i\,ps_j}) - prob(f_{i\,\overline{ps_j}}).$$

Since we are considering the domain (0, 1), which is not inclusive of 0 and 1, and the ns_i's are nontrivial Boolean functions of at least two present state lines for every i, this partial differential is strictly less than one, because we are guaranteed that $prob(f_{i\,ps_j}) > 0$ and $prob(f_{i\,\overline{ps_j}}) > 0$. ■

From Theorems 7.1 and 7.2, we can see that the iterated signal probability calculation is guaranteed to converge to a solution, provided some mild assumptions are made with respect to the functionality of the next state logic.

B. Newton–Raphson Method

The Newton–Raphson method can be used to solve a nonlinear system of equations given an initial guess at the solution. The advantage of the Newton–Raphson method is the quadratic rate of convergence. However, each iteration is more computationally expensive than the Picard–Peano method.

Given $Y(P) = 0$ and a column matrix corresponding to an initial guess P^0, we can write the kth Newton iteration as the linear system solve shown below.

$$J(P^k) \times P^{k+1} = J(P^k) \times P^k - Y(P^k) \tag{8}$$

where J is the $N \times N$ Jacobian matrix of the system of equations. Each entry in J corresponds to a $\partial y_i/\partial p_j$ evaluated at P^k. The P^{k+1} correspond to the variables in the linearized system and after solving the system P^{k+1} is used as the next guess. Convergence is deemed to be achieved if each entry in $Y(P^k)$ is sufficiently small.

We use the methods of Section VII-C to evaluate:

$$\begin{aligned} &g_i(p_1, p_2, \cdots, p_N) \\ &\quad = prob[f_i(i_1, i_2, \cdots, i_M, ps_1, ps_2, \cdots, ps_N)] \end{aligned}$$

for given $p_j = prob(ps_j)$'s and $prob(i_k)$'s. The $Y(P^k)$ of (8) can easily be evaluated using the p_j^k values and using (6).

We need to also evaluate $J(P^k)$. As mentioned earlier, each entry of J corresponds to $\partial y_i/\partial p_j$ evaluated at P^k. If $i \neq j$, then $\partial y_i/\partial p_j$ equals $-\partial g_i/\partial p_j$, and $\partial y_i/\partial p_i$ equals $1 - \partial g_i/\partial p_i$.

In order to perform the evaluation of $\partial g_i/\partial p_j$ we use the method in the proof of Theorem 7.2.

$$\frac{\partial g_i}{\partial p_j} = prob(f_{i\,ps_j}) - prob(f_{i\,\overline{ps_j}}).$$

We can evaluate $prob(f_{i\,ps_j})$ and $prob(f_{i\,\overline{ps_j}})$ for a given P^k using the methods of Section VII-C.

As an example consider:

$$\begin{aligned} f_1 &= i_1 \wedge ps_1 \wedge \overline{ps_2} \vee i_1 \wedge \overline{ps_1} \wedge ps_2 \\ \frac{\partial g_1}{\partial p_1} &= prob(i_1 \wedge \overline{ps_2}) - prob(i_1 \wedge ps_2) \\ \frac{\partial g_1}{\partial p_1} &= 0.5 \cdot (1 - p_2) - 0.5 \cdot p_2 = 0.5 - p_2 \end{aligned}$$

which is exactly what we would have obtained had we differentiated (7) with respect to p_1.

Theorem 7.3: [11] The Newton iterates:

$$P^{k+1} = P^k - J(P^k)^{-1} Y(P^k), \quad k = 0, 1, \cdots,$$

are well-defined and converge to a solution P^* of $Y(P) = 0$ if the following conditions are satisfied:

1) Y is F-differentiable.
2) $||J(A) - J(B)|| \leq \gamma ||A - B||, \forall A, B \in D_0$ where D_0 is the domain $0 \leq p_i \leq 1, \forall i$.
3) There exists $P^0 \in D_0$ such that $||J(P^0)^{-1}|| \leq \beta$, $\eta \geq ||J(P^0)^{-1} Y(P^0)||$ and $\alpha = \beta\gamma\eta \leq \frac{1}{2}$.

Condition 1 of the theorem is satisfied in our application because the y_i functions are continuous and differentiable. We need to prove that the parameter γ is finite to show that Condition 2 is satisfied.

Theorem 7.4: If Y is given by (6), then $\gamma \leq 2$.

Proof: In order to show that:

$$||J(A) - J(B)|| \leq \gamma ||A - B||, \forall\, A,\, B \in D_0$$

is satisfied for $\gamma = 2$, we will show that the derivative of each entry of J is less than or equal to 2.

Recall that J is a matrix with each entry corresponding to $\partial y_i / \partial p_j$. Using the equations provided in the proof of Theorem 7.2 we can write:

$$\frac{\partial y_i}{\partial p_j} = prob(f_{i\overline{ps_j}}) - prob(f_{ips_j}) \quad i \neq j.$$

Differentiating with respect to p_k we have:

$$\frac{\partial^2 y_i}{\partial p_j \partial p_k} = prob(f_{i\overline{ps_j}ps_k}) - prob(f_{i\overline{ps}_j\overline{ps_k}}) - prob(f_{ips_jps_k}) + prob(f_{ips_j\overline{ps_k}}).$$

Given that the probabilities are between 0 and 1, we have:

$$\left| \frac{\partial^2 y_i}{\partial p_j \partial p_k} \right| \leq 2.$$

■

Condition 3 in Theorem 7.3 is a constraint on the initial guess for the Newton iteration, and this initial guess can be picked appropriately, provided γ is finite. Essentially, we have to choose P^0 such that $||Y(P^0)||$ is small.

B. Signal Probability Evaluation

In the nonlinear equation solver, regardless of whether we are using the Picard–Peano method or the Newton–Raphson method, we have to repeatedly evaluate the signal probability of a Boolean function given input probabilities, i.e., compute $prob[f_i(i_1, i_2, \cdots, i_M, ps_1, ps_2, \cdots, ps_N)]$ given the $prob(i_k)$'s and the $prob(ps_j)$'s.

There exist several methods to evaluate signal probability. An exact method corresponds to using Ordered Binary Decision Diagrams (OBDD's) [1]. If an OBDD can be created for f_i, then $prob(f_i)$ can be evaluated in linear time in the size of the OBDD for f_i. OBDD's can be cofactored in linear time, allowing for the efficient evaluation of the Jacobian entries.

An alternative is to use Monte Carlo simulation. Approximate signal probabilities can be computed using random logic simulation on the multilevel network corresponding to f_i. Our experience has been that the signal probabilities quickly converge to the exact results obtained using OBDD's. In order to evaluate a particular Jacobian entry, the appropriate input to f_i has to be set to 0 (1) and random simulation is performed on the remaining inputs.

VIII. Experimental Results

In this section we present experimental results that illustrate the following points:

- Exact and explicit computation of state probabilities is possible for controller type circuits. However, it is not viable for data path circuits. Purely combinational logic estimates result in significant inaccuracies.
- Assuming uniform probabilities for the present state line probabilities and state probabilities as in [4] can result in significant inaccuracies in power estimates.
- Computing the present state *line* probabilities using the technique presented in the previous sections results in 1) accurate switching activity estimates for all internal nodes in the network implementing the sequential machine; 2) accurate, robust and computationally efficient power estimate for the sequential machine.

TABLE I
Comparison of Sequential Power Estimation Methods

Circuit Name	#lit	#ff	Combinational			Uniform Prob.			Line Prob.			State Prob.	
			power	err	cpu	power	err	cpu	power	err	cpu	power	cpu
cse	132	4	610.0	58.7	1s	578.1	50.3	7s	380.3	1.0	9s	384.4	11s
dk16	180	5	1077.5	3.1	1s	1097.2	5.0	10s	1045.0	0.0	13s	1044.8	15s
dfile	119	5	923.2	32.5	1s	701.5	0.6	7s	701.4	0.6	8s	696.8	10s
keyb	169	5	749.8	43.3	1s	724.9	38.6	12s	517.6	1.0	14s	523.0	15s
mod12	25	4	245.2	21.7	0s	195.9	2.7	1s	199.1	1.1	1s	201.4	1s
planet	327	6	1640.6	2.5	2s	1709.4	1.5	17s	1685.9	0.1	24s	1683.9	28s
sand	336	5	1446.0	33.1	2s	1165.5	7.2	24s	1078.2	0.7	27s	1086.4	34s
sreg	9	3	127.5	1.4	0s	129.4	0.0	0s	129.4	0.0	0s	129.4	1s
styr	313	5	1394.8	45.3	2s	1208.2	25.8	22s	996.9	3.8	28s	959.9	30s
tbk	478	5	1958.1	24.1	4s	1903.6	20.7	48s	1538.2	2.4	52s	1577.0	71s
accum4	45	4	360.9	3.5	0s	374.3	0.0	2s	374.3	0.0	2s	374.3	5s
accum8	89	8	720.6	4.2	1s	752.6	0.0	7s	752.6	0.0	8s	752.6	875s
accum16	245	16	1521.2	-	2s	1596.3	-	234s	1596.3	-	239s	unable	
count4	19	4	256.2	20.1	0s	213.3	0.0	1s	213.3	0.0	1s	213.3	2s
count7	35	7	474.2	12.2	0s	422.6	0.0	2s	422.6	0.0	3s	422.6	5s
count8	40	8	560.1	10.2	0s	507.9	0.0	3s	507.9	0.0	4s	507.9	8s
cbp32.4	489	223	8719.1	12.2	15s	8731.9	12.3	45s	7745.4	0.3	119s	7769.1	84s
add16	214	98	3772.3	5.1	3s	3780.5	5.4	13s	3568.0	0.5	22s	3586.5	23s
mult8	176	87	5985.6	22.8	12s	5962.6	22.4	82s	4866.9	0.1	110s	4871.1	344s
s953	418	29	762.4	76.8	1s	672.7	56.0	10s	438.7	1.7	12s	431.1	15s
s1196	529	18	2557.6	-	4s	2538.4	-	484s	2293.8	-	488s	unable	
s1238	508	18	2709.4	-	4s	2688.3	-	156s	2439.2	-	151s	unable	
s1423	657	74	6017.1	-	251s	4734.2	-	271s	7087.1	-	289s	unable	
s5378	4212	164	12457.4	-	74s	12415.1	-	455s	6496.0	-	478s	unable	
s13207	11241	669	37842.1	-	5m	27186.4	-	11m	10572.7	-	338m	unable	
s15850	13659	597	40016.2	-	8m	23850.7	-	14m	10534.1	-	167m	unable	
s35932	28269	1728	122131.2	-	20m	118475.3	-	36m	62292.0	-	152m	unable	
s38584	32910	1452	112705.6	-	24m	85842.1	-	44m	63995.1	-	922m	unable	

In Table I, results are presented for several circuits. In the table, *combinational* corresponds to the purely combinational estimation method of [4] and *uniform-prob* corresponds to the sequential estimation method of [4] that assumes uniform state probabilities. The column *line-prob* corresponds to the technique of Section IV and using the Newton–Raphson method with a convergence criterion of 0.0001% to solve the equations. These equations correspond to $k = 0$ or $m = 1$. Finally, *state-prob* corresponds to the exact state probability calculation method of Section III. The zero delay model was assumed, however, any other delay model could have been used instead.

The first set of circuits corresponds to finite state machine controllers. These circuits typically have the characteristic that the state probabilities are highly nonuniform. Restricting oneself to combinational power dissipation (*combinational*) or assuming uniform state probabilities (*uniform-prob*) results in significant errors. However, the line probability method of Section IV produces highly accurate estimates when compared to exact state probability calculation.

The second set of circuits corresponds to datapath circuits, such as counters and accumulators. The exact state probability evaluation method requires huge amounts of CPU time for even the medium-sized circuits, and cannot be applied to the large circuits. For all the circuits that the exact method is viable for, our *line-prob* method produces identical estimates. The *uniform-prob* method does better for the datapath circuits—in the case of counters for instance, it can be shown that the state probabilities are all uniform, and therefore the *uniform-prob* method will produce the right estimates. Of course, this assumption is not always valid.

The third set of circuits corresponds to pipelined adders and a pipelined multiplier. For pipelined circuits, exact power estimation is possible without resort to Chapman–Kolmogorov equation solving . The fourth set corresponds to mixed data-path/control circuits from the ISCAS-89 benchmark set. Exact state probability evaluation is not possible for these circuits.

The CPU times in the table corresponds to seconds (s) or (m) on a SUN SPARC-2. The CPU times correspond to times required for symbolic simulation to estimate combinational activity plus the time required for the calculation of state/line probabilities. For all the circuits BDD's were used to obtain the line probabilities. However, Monte-Carlo simulation was used for combinatorial activity estimation for the large ISCAS-89 circuits.

In Table II, present state line probability estimates for the benchmark circuits are presented. The error value provided in each column shows the absolute error (i.e., absolute value of the difference between exact and approximate values) of the signal probabilities averaged over all present state lines in the circuit. The exact values were calculated by the method described in Section III. (We could not generate the exact values for circuits in Groups 3 and 4, as the size of Chapman–Kolmogorov system of equations becomes too large.) It is evident from these results that the error averaged over all benchmark circuits is well below 0.05 (see the *line-prob* column entries which correspond to the method described in Section IV). Note that this error is due to ignoring correlation as exemplified in Section IV-B, and not due to convergence error of the Newton–Raphson method. The convergence criterion for line probabilities was set to 0.0001% to generate these results.

We present the switching activity errors for the benchmark circuits in Table III. Again, the error value provided in each column represents the absolute error averaged over all internal nodes in the circuit. It can be seen that this error is quite small. These two tables demonstrate that the approximate procedure provided in Section IV leads to very accurate estimates for both the present state line probabilities and for the switching activity values for all circuit lines.

Next, we present results comparing the Picard–Peano and Newton–Raphson methods to solve the nonlinear equations of Section IV. These results are summarized in Table IV. The number of iterations required for the Picard–Peano and Newton–Raphson methods are given in Table IV under the appropriate columns, as are the CPU times per iteration and the total CPU time. Newton–Raphson typically takes fewer iterations, but each iteration requires the evaluation of the Jacobian and is more expensive than the Picard iteration. The results obtained by the two methods are identical, since the convergence criterion used was the same.

TABLE II
ABSOLUTE ERRORS IN PRESENT STATE LINE PROBABILITIES AVERAGED OVER ALL PRESENT STATE LINES

Circuit Name	Combinational err	Uniform Prob. err	Line Prob. err
cse	0.427	0.427	0.00788
dk16	0.0782	0.0782	0.0125
dfile	0.075	0.075	0.047
keyb	0.414	0.414	0.0133
mod12	0	0	0.03
planet	0.031	0.031	0.09
sand	0.12	0.12	0.044
sreg	0	0	0
styr	0.3138	0.3138	0.0357
tbk	0.2614	0.2614	0.026
accum4	0	0	0
accum8	0	0	0
accum16	0	0	0
count4	0	0	0
count7	0	0	0
count8	0	0	0
cbp32.4	-	-	-
add16	-	-	-
mult8	-	-	-
s953	-	-	-
s1196	-	-	-
s1238	-	-	-

To generate the results in Table IV, the convergence criterion allowed a maximum error of 1% in the line probabilities. In this case, the Picard–Peano method out performs the Newton–Raphson method for virtually all the examples. If the convergence criterion is tightened, e.g., to allow for a maximum error of .01%, the Picard–Peano method requires substantially more iterations than the Newton–Raphson and in several examples, the Newton–Raphson method outperforms the Picard–Peano method. However, since the error due to ignoring correlation (cf. Section IV-B) can be more than 1%, in practice it does not make sense to tighten the convergence criterion beyond a 1% allowed error.

In some pathological examples, where the conditions of Theorem 7.1 are not satisfied, the Picard–Peano method may exhibit oscillatory behavior, and will not converge. In these cases, the strategy we adopt is to use Picard–Peano for several

TABLE III
ABSOLUTE ERRORS IN SWITCHING ACTIVITY AVERAGED OVER ALL CIRCUIT LINES

Circuit Name	Combinational err	Uniform Prob. err	Line Prob. err
cse	0.402	0.053	0.003
dk16	0.354	0.020	0.010
dfile	0.268	0.019	0.015
keyb	0.363	0.067	0.009
mod12	0.387	0.149	0.156
planet	0.375	0.034	0.034
sand	0.400	0.015	0.010
sreg	0	0	0
styr	0.415	0.058	0.022
tbk	0.423	0.020	0.008
accum4	0.084	0	0
accum8	0.086	0	0
accum16	0.096	0	0
count4	0.169	0	0
count7	0.189	0	0
count8	0.192	0	0
cbp32.4	-	-	-
add16	-	-	-
mult8	-	-	-
s953	-	-	-
s1196	-	-	-
s1238	-	-	-

TABLE IV
COMPARISON OF PICARD–PEANO AND NEWTON–RAPHSON

Circuit Name	Picard-Peano			Newton-Raphson		
	#iter	cpu/iter	total cpu	#iter	cpu/iter	total cpu
cse	5	0.1	0.5	3	1	3
dk16	4	0.18	0.7	3	1	3
dfile	5	0.12	0.6	2	1.5	3
keyb	10	0.07	0.7	6	0.33	2
mod12	3	0.03	0.1	2	0.1	0.2
planet	11	0.13	1.4	3	2.33	7
sand	6	0.22	1.3	3	1	3
sreg	1	0.1	0.1	1	0.1	0.1
styr	7	0.2	1.4	3	2	6
tbk	4	0.5	2.0	3	1.33	4
accum4	1	0.1	0.1	1	0.1	0.1
accum8	1	0.3	0.3	1	1	1
accum16	1	1.0	1.0	1	6	6
count4	1	0.1	0.1	1	0.1	0.1
count7	1	0.2	0.2	1	1	1
count8	1	0.2	0.2	1	1	1
cbp32.4	3	0.8	2.4	4	18.5	74
add16	3	0.3	0.9	3	3	9
mult8	2	3.25	6.5	4	9.25	37
s953	30	0.04	1.1	4	0.5	2
s1196	2	1.1	2.2	2	2	4
s1238	2	1.15	2.3	2	2.5	5

iterations, and if oscillation is detected, the Newton–Raphson method is applied. The Newton–Raphson method does not require the domain to be contractive, however, the initial guess has to be "close" to the solution P^* in a manner quantified by Theorem 7.3.

In Table V, we present results that indicate the improvement in accuracy in power estimation when k-unrolled or m-expanded networks are used. Results are presented for the finite state machine circuits of Table I for $0 \leq k \leq 2$ and $1 \leq m \leq 4$.[4] The percentage differences in power from the exact power estimate are given. In general, if $k \to \infty$, the error will reduce to 0%, however, increasing k when k is small is not guaranteed to reduce the error in total power estimates (e.g., consider styr). This phenomenon can be explained as follows. The total power estimate is obtained by summing power consumptions of all nodes in the circuit. The individual power estimates may be under- or over-estimated, yet when they are added together, the overall error may become small due to error cancelation. Increasing k improves the accuracy of power estimates for individual nodes (see Table VI), but does not necessarily improve the accuracy of power estimate for the circuit due to the unpredictability of the error cancelation during the summing step. The m-expansion-based method behaves more predictably for this set of examples, however, again no guarantees can be made regarding the improvement in accuracy (of total power estimates) on increasing m, except that when m is set to the number of flip-flops in the machine, the method produces the Chapman–Kolmogorov equations, and therefore the exact state probabilities are obtained. The Newton–Raphson method with a convergence criterion of 0.0001% was used to obtain the line probabilities in Tables V and VI.

The CPU times for power estimation are in seconds on a SUN SPARC-2. These times can be compared with those listed in Table I under the "Line Prob." column as those times correspond to $k = 0$ and $m = 1$. Based on these results, we conclude that $k = 1$ and $m = 2$ provide a good compromise between accuracy and run-time.

During the synthesis process, we often want to know the switching activity of individual nodes instead of a single power consumption figure. Table VI presents the percentage error in

[4]The initial error for dk16 and sreg benchmarks is 0, thus, there is no need to improve the accuracy by using larger values of k and m.

TABLE V
RESULTS OF POWER ESTIMATION BASED ON k-UNROLLED AND m-EXPANDED NETWORKS

Circuit Name	Initial Error	k-Unrolled Error				m-Expanded Error			
		$k=1$		$k=2$		$m=2$		$m=4$	
		err	cpu	err	cpu	err	cpu	err	cpu
cse	1.06	0.33	18	0.02	51	0.42	10	0.00	10
dfile	0.67	0.20	16	0.20	29	0.23	9	0.17	10
keyb	1.02	0.02	44	0.04	53	1.01	14	0.32	14
mod12	1.13	0.85	2	0.30	3	1.13	1	0.00	2
planet	0.11	0.15	40	1.72	45	0.10	25	0.08	25
sand	0.76	0.61	64	0.29	109	0.64	28	0.43	30
styr	3.85	0.16	67	0.41	113	0.58	29	0.52	29
tbk	2.46	1.52	207	0.12	597	2.17	58	0.12	59

TABLE VI
PERCENTAGE ERROR IN SWITCHING ACTIVITY ESTIMATES AVERAGED OVER ALL NODES IN THE CIRCUIT

Circuit Name	average % error					
	$k=0$	$k=1$	$k=2$	$m=1$	$m=2$	$m=4$
cse	6.79	2.26	0.57	6.79	3.40	0.00
dfile	14.05	5.37	3.10	14.05	4.82	3.56
keyb	7.18	1.68	0.70	7.18	7.09	2.25
mod12	10.24	6.36	5.00	10.24	10.05	0.00
planet	43.08	30.22	28.97	43.08	41.26	35.22
sand	16.65	12.20	11.78	16.65	14.02	9.42
styr	43.51	12.99	6.31	43.51	6.55	5.97
tbk	18.04	4.48	2.95	18.04	15.91	1.88

individual node's switching activity from the exact values as a function of k and m, averaged over all the nodes in the circuit. It is seen that the accuracy of switching activity estimates consistently increases with the value of k and m. For example, the error in switching activity estimates for styr decreases from 13% to 6.3% when k increases from 1 to 2 and from 6.6% to 6.0% when m increases from 2–4. A similar trend exists with respect to the maximum error and the root-mean-squared error criteria.

IX. CONCLUSIONS AND ONGOING WORK

We presented a framework for sequential power estimation in this paper. In this framework, state probabilities can be computed using the Chapman–Kolmogorov equations, and present state line probabilities are computed by solving a system of nonlinear equations. We have shown that the latter is significantly more efficient for medium to large circuits, and does not sacrifice accuracy.

Given the present state line probabilities, the switching activity and power dissipation of the circuit can be accurately computed. Any combinational logic estimation method that can accurately model the correlation between the applied input vector pairs can be used.

REFERENCES

[1] R. Bryant, "Graph-based algorithms for Boolean function manipulation," *IEEE Trans. Comput.*, vol. C-35, pp. 677–691, Aug. 1986.

[2] R. Burch, F. Najm, P. Yang, and T. Trick." McPOWER: A Monte Carlo approach to power estimation," in *Proc. Int. Conf. Computer-Aided Design*, Nov. 1992, pp. 90–97.

[3] A. Chandrakasan, T. Sheng, and R. W. Brodersen, "Low power CMOS digital design," in *J. Solid State Circuits*, pp. 473–484, Apr. 1992.

[4] A. Ghosh, S. Devadas, K. Keutzer, and J. White, "Estimation of average switching activity in combinational and sequential circuits," in *Proc. 29th Des. Automat. Conf.*, June 1992, pp. 253–259.

[5] L. Glasser and D. Dobberpuhl, *The Design and Analysis of VLSI Circuits*. Reading, MA: Addision-Wesley, 1985.

[6] H. M. Lieberstein, *A Course in Numerical Analysis*. New York: Harper & Row, 1968.

[7] R. Marculescu, D. Marculescu, and M. Pedram, "Logic level power estimation considering spatiotemporal correlations," in *Proc. Int. Conf. Computer-Aided Design*, Nov. 1994, pp. 294–299.

[8] J. Monteiro, S. Devadas, and A. Ghosh, "Retiming sequential circuits for low power," in *Proc. Int. Conf. Computer-Aided Design*, Nov. 1993, pp. 398–402.

[9] F. Najm, "Transition density, A stochastic measure of activity in digital circuits," in *Proc. 28th Des. Automat. Conf.*, June 1991, pp. 644–649.

[10] F. N. Najm, R. Burch, P. Yang, and I. Hajj, "Probabilistic simulation for reliability analysis of CMOS VLSI circuits," *IEEE Trans. Computer-Aided Design*, pp. 439–450, Apr. 1990, vol. 9.

[11] J. M. Ortega and W. C. Rheinboldt, *Interative Solution of Nonlinear Equations in Several Variables*. Boston, MA: Academic, 1970.

[12] A. Papoulis, *Probability, Random Variables and Stochastic Processes*. New York: McGraw-Hill, 1991, 3rd ed.

[13] C. Y. Tsui, M. Pedram, and A. Despain, "Efficient estimation of dynamic power dissipation under a real delay model," in *Proc. Int. Conf. Computer-Aided Design*, June 1993, pp. 224–228.

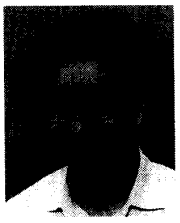
[14] ——, "Exact and approximate methods for calculating signal and transition probabilities in FSMs," Elec. Eng.-Syst. Dept., Univ. So. CA, Tech. Rep. CNEG 93-42, Oct. 1993.

Chi-Ying Tsui received the B.S. degree in electrical engineering from the University of Hong Kong, and the M.S. and Ph.D. degrees in computer engineering from the University of Southern California in 1989 and 1994, respectively.

He is an Assistant Professor of Electrical and Electronic Engineering at the Hong Kong University of Science and Technology. He is working on VLSI design and CAD algorithms for high performance low-power microprocessor design. His research interests include power analysis and optimization for CMOS circuits, and hardware/software codesign for high-performance low-power processors.

José Monteiro received the Engineer's and Master's degrees in electrical and computer engineering in 1989 and 1994, respectively, from Instituto Superior Técnico at the Technical Univeristy of Lisbon.

He is currently working towards the Ph.D. degree at the Massachusetts Institute of Technology in the area of power estimation and synthesis for low power of VLSI circuits.



Massoud Pedram (M'90–S'90–M'91) received the B.S. degree in electrical engineering from the California Institute of Technology in 1986, and the M.S. and Ph.D. degrees in electrical engineering and computer sciences from the University of California, Berkeley in 1989 and 1991, respectively.

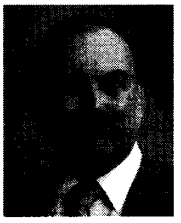
He is an Assistant Professor of Electrical Engineering—Systems at the University of Southern California. His research interests span many aspects of design and synthesis of VLSI circuits, with particular emphasis on layout optimization, logic synthesis and behavioral optimization, layout-driven synthesis, and design for low power.

Dr. Pedram is a recipient of the National Science Foundation's Research Initiation Award in 1992 and the Young Investigator Award in 1994. His research has received a number of awards including one ICCD Best Paper Award and a Distinguished Paper Citation from ICCAD. He has served on the technical program committee of a number of conferences and workshops, including the Design Automation Conference. He was the co-founder and General Chair of the 1994 International Workshop on Low Power Design, and the General Chair of the 1995 International Symposium on Low Power Design. He has given several tutorials on low power design at major CAD conferences and forums including, ICCAD and DAC. He is a member of the ACM.

Srinivas Devadas (S'87–M'88) received the B. Tech degree in electrical engineering from the Indian Institute of Technology, Madras in 1985, and the M.S. and Ph.D. degrees in electrical engineering from the University of California, Berkeley, in 1986 and 1988, respectively.

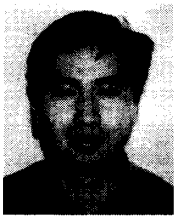
Since August 1988, he has been at the Massachusetts Institute of Technology, Cambridge, and is currently an Associate Professor of Electrical Engineering and Computer Science. He held the Analog Devices Career Development Chair of Electrical Engineering from 1989 to 1991. His research interests span all aspects of synthesis of VLSI systems.

Dr. Devadas has received five Best Paper Awards at CAD conferences and journals, including the 1990 IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN Best Paper Award. In 1992, he received the NSF Young Investigator Award. He has served on the technical program committees of several conferences and workshops including the International Conference on Computer Design, and the International Conference on Computer-Aided Design. He is a member of the ACM.



Alvin M. Despain (S'58–M'65) received the B.S., M.S., and Ph.D. degrees in electrical engineering from the University of Utah in 1960, 1962, and 1966, respectively.

He is the Powell Professor of Computer Engineering at the Univeristy of Southern California (USC), and a Professor in the Computer Science and Electrical Engineering Systems Departments. He has been an Assistant Research Professor at the University of Utah, an Associate Professor at Utah State University, a Visiting Associate Professor at Stanford University, a Professor at the University of California at Berkeley, and has been at USC since 1989. He is a pioneer in the study of high-performance computer systems for symbolic calculations. His research group builds experimental software and hardware systems including compilers, custom VLSI processors, and multiprocessor systems. Their goal is to determine principles for the design of high-performance computer systems. Despain's research interests include computer architecture, multiprocessor and multicomputer systems, logic programming, and design automation.



Bill Lin received the B.Sc., M.S., and Ph.D. degrees in electrical engineering and computer sciences from the University of California, Berkeley, in 1985, 1988, and 1991, respectively.

Since graduating from Berkeley, he has been working in the VLSI Systems Design Methodologies division of the Inter-University Micro-Electronics Center (IMEC) in Leuven, Belgium. At IMEC, he is currently heading the System Control and Communications group, which is mainly focusing on system design technology for embedded hardware-software systems. This group also has a major effort in asynchronous design methods and high-speed telecom ATM-based network applications. He has been work-package leader in several E.C. sponsored Esprit projects an Belgian Flemish government sponsored IWT projects. Previously, he has worked at the Hewlett Packard Corporation, the Hughes Aircraft Company, and the Western Digital Corporation. He has authored or co-authored more than 70 scientific publications in the area of CAD methods for VLSI design.

Dr. Lin has served on the program committee of several international conferences. In 1987, he received a Best Paper Award at the 24th Design Automation Conference, Miami, FL. In 1989 and 1990, respectively, he received a Distinguished Paper Citation at the IFIP VLSI conference in Munich, Germany, and at the ICCAD conference in Santa Clara, CA. In 1994, he received a best paper nomination at the ACM Design Automation Conference, San Diego, CA.