Exact and Approximate Methods of Switching Activity Estimation in Sequential Logic Circuits

Chi-Ying Tsui José Monteiro Massoud Pedram Alvin M. Despain
Srinivas Devadas Bill Lin

Abstract

Recently developed methods for power estimation have primarily focused on combinational logic. In this paper, 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, average switching activity estimation for sequential circuits is considerably more difficult than 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 accurately estimate the power dissipation of sequential circuits by computing the exact state probabilities in steady state. However, 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.

We describe a comprehensive framework for exact and approximate switching activity estimation in this paper. The basic computation step is the solution of a non-linear system of equations. 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.

J. Monteiro and S. Devadas were supported in part by the Defense Advanced Research Projects Agency under contract N00014-91-J-1698 and in part by a NSF Young Investigator Award with matching funds from Mitsubishi and IBM Corporation.

C-Y. Tsui, M. Pedram and A. Despain are with the Department of Electrical Engineering at the University of Southern California.

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

B. Lin is with IMEC, Belgium.

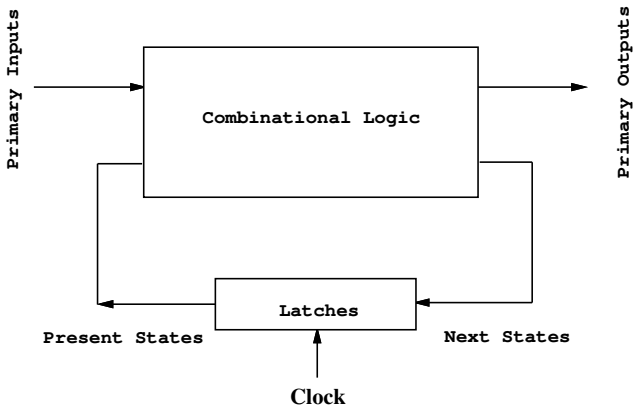

Figure 1: A Synchronous Sequential Circuit

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 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 multi-chip modules where heat dissipation is one of the biggest problems.

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

Methods for the power estimation of logic-level combinational circuits based on switching activity estimation (*e.g.* [2], [4], [8], [11]) have been presented previously. 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. At any given time instant, the probability that the circuit is in a particular state can be distinct across all the states. As an example, consider the sequential circuit of Figure 1 and the example State Transition Graph of Figure 2. Assuming that the circuit was in state **R** at time 0, and that at each clock cycle uniform 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 for the combinational logic. Power dissipation and switching activity of CMOS combinational logic is

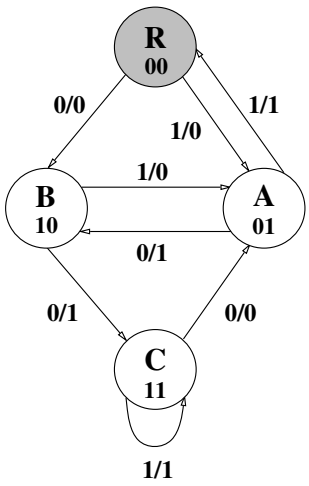

Figure 2: Example State Transition Graph

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 Figure 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 Figure 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 accurate sequential switching activity estimation.

A first attempt at estimating switching activity in logic-level sequential circuits has been 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 [7].

In this paper, we present results obtained by using the Chapman-Kolmogorov equations for discrete-time Markov Chains [10] 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 ≤ 15 flip-flops, since it requires the explicit consideration of each state in the circuit. We show that the estimates produced by purely combinational methods can greatly differ from those produced by the exact state probability method.

We next describe a comprehensive framework for exact and approximate switching activity estimation. The basic computation step is the solution of a non-linear system of equations. 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 2 we briefly review the physical model for power estimation, and summarize the combinational estimation method of [4]. In Section 3, we describe an exact switching activity estimation method for sequential circuits. In Section 4, 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 non-linear 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 5. We describe a different method to improve the accuracy based on the notion of a m-expanded network in Section 6. In Section 7 we describe methods to solve the non-linear system of equations, namely, the Picard-Peano and the Newton-Raphson methods. In Section 8, 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.

2 Preliminaries

2.1 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.

All of these are reasonably accurate assumptions 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 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:

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

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 [8], 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.

Eq. (1) is used by the power estimation techniques such as [4, 8] to relate switching activity to power dissipation.

2.2 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).

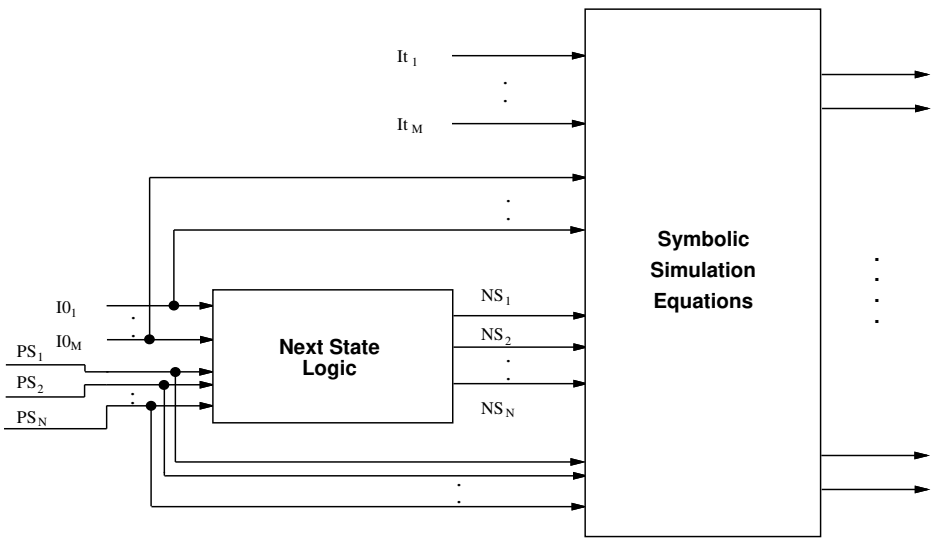

Figure 3: Taking Correlation Into Account

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.

3 The Exact Method

3.1 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 2.2. This augmentation is summarized in Figure 3.

In Figure 3, we have a block corresponding to the symbolic simulation equations for the combinational logic of the general sequential circuit shown in Figure 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 Figure 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. This configuration was originally

proposed in [4].

3.2 State Probability Computation

The static probabilities for the present state lines marked PS in Figure 3 are also correlated. We require knowledge of *present state probabilities* as opposed to present state line (PS) probabilities. The state probabilities are dependent on the connectivity of the State Transition Graph (STG) of the circuit.

These state probabilities can be computed using the Chapman-Kolmogorov equations for discrete-time Markov Chains [10]. We describe the method below.

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.

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

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

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

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 $prob(\mathbf{C}) = \frac{1}{4}$.

3.3 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 2.2, 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 Figure 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 2.2, 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 weigh these combinations according to the given probabilities. As an example, consider the 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 f can be calculated by summing the probabilities of the two product terms in f since the two terms have a null intersection. The probability of the first term 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 term 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:

$$prob(f) = \frac{1}{4} + \frac{1}{6} = \frac{5}{12}$$

The major disadvantage of this estimation method is its *average-case* exponential complexity – the probability of each state is computed, and the number of states 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.

4 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.

$$prob(ps_1 = 0) = prob(00) + prob(01) = \frac{1}{6} + \frac{1}{3} = \frac{1}{2}$$

$$prob(ps_1 = 1) = prob(10) + prob(11) = \frac{1}{4} + \frac{1}{4} = \frac{1}{2}$$

$$prob(ps_2 = 0) = prob(00) + prob(10) = \frac{1}{6} + \frac{1}{4} = \frac{5}{12}$$

$$prob(ps_2 = 1) = prob(01) + prob(11) = \frac{1}{3} + \frac{1}{4} = \frac{7}{12}$$

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. Next, given the exact state probabilities, the line probabilities

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

were determined as described above. Using the topology of Figure 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.

4.1 Computing Present State Line Probabilities

In our approximation framework we *directly determine line probabilities without recourse to State Transition Graph extraction.* Moreover, we provide mechanisms for improved accuracy by using the notions of k-unrolled and m-expanded networks, respectively. The approximation framework is based on solving a non-linear 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.

$$ns_1 = f_1(i_1,\ i_2,\ \cdots,\ i_M,\ ps_1,\ ps_2,\ \cdots,\ ps_N)$$

$$ns_2 = f_2(i_1,\ i_2,\ \cdots,\ i_M,\ ps_1,\ ps_2,\ \cdots,\ ps_N)$$

$$\cdots$$

$$ns_N = f_N(i_1,\ i_2,\ \cdots,\ i_M,\ ps_1,\ ps_2,\ \cdots,\ ps_N)$$

We can write:

$$prob(ns_1) = prob(f_1(i_1,\ i_2,\ \cdots,\ i_M,\ ps_1,\ ps_2,\ \cdots,\ ps_N))$$

$$prob(ns_2) = prob(f_2(i_1,\ i_2,\ \cdots,\ i_M,\ ps_1,\ ps_2,\ \cdots,\ ps_N))$$

$$\cdots$$

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

$$prob(ns_N) \;=\; prob(f_N(i_1,\; i_2,\; \cdots,\; i_M,\; ps_1,\; ps_2,\; \cdots,\; ps_N))$$

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:

$$prob(ps_i) \;=\; prob(ns_i) \;=\; p_i \quad 1 \leq i \leq N$$

A similar relationship was used in the Chapman-Kolmogorov equations (*cf.* Section 3).

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

$$y_1 = p_1 - g_1(p_1,\; p_2,\; \cdots,\; p_N) \;=\; 0$$

$$y_2 = p_2 - g_2(p_1,\; p_2,\; \cdots,\; p_N) \;=\; 0$$

$$\cdots$$

$$y_N = p_N - g_N(p_1,\; p_2,\; \cdots,\; p_N) \;=\; 0 \tag{2}$$

where the g_i's are non-linear 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

$$g_1 = 0.5 \cdot (p_1 \cdot (1 - p_2) + (1 - p_1) \cdot p_2) \tag{3}$$

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 7. The uniqueness or the existence of the solution is not guaranteed for an arbitrary system of non-linear equations. However, since in our application we have a correspondence between the non-linear system of equations and the State Transition Graph of the sequential circuit there will exist at least one solution to the non-linear system. Further, convergence is guaranteed under mild assumptions for our application.

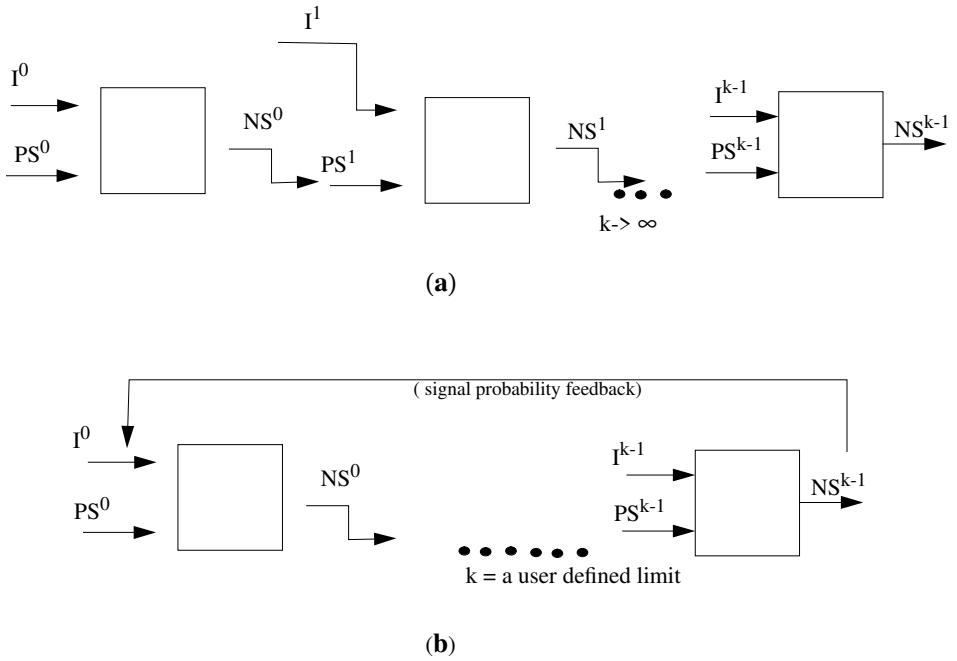

Figure 4: k-unrolling of the next state logic

4.2 Inaccuracy in Formulation

The above formulation does not capture the correlation between the state line probabilities. While the state line probabilities obtained using the above method will result in switching activity estimates that are close to the exact method, it is worthwhile to explore ways to increasing the accuracy. We describe two such mechanisms in Section 5 and Section 6.

5 Improving Accuracy using k-Unrolled Networks

5.1 State Line Probability Computation

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

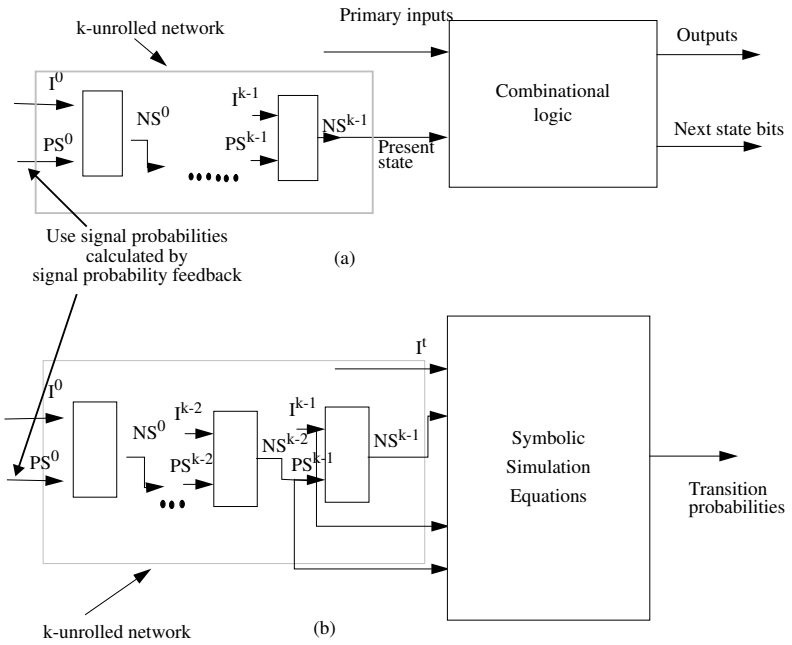

Figure 5: Calculation of signal and transition probabilities by network unrolling

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

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

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

Figure 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.

5.2 Switching Activity Computation

The topology of Figure 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 Figure 5(b). Instead of

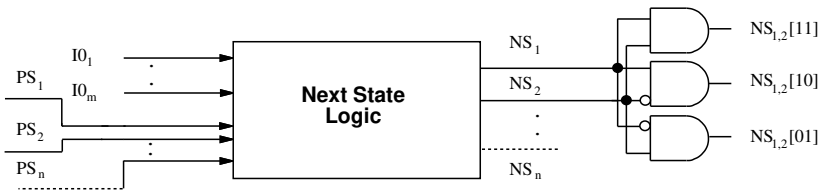

Figure 6: An m-Expanded Network with $m = 2$

connecting the next static logic network to the symbolic simulation equations, we unroll the next static logic network k times and connect the next state lines of the k^{th} 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.

6 Improving Accuracy Using m-Expanded Networks

6.1 State Line Probability Computation

We describe a method to improve the accuracy of the basic approximation strategy outlined in Section 4. This method models the correlation between m-tuples of present state lines. The method is pictorially illustrated in Figure 6 for $m = 2$.

The number of equations in the case of $m = 2$ is $\frac{3N}{2}$. 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 7.3 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 $\frac{7N}{3}$. 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.

6.2 Switching Activity Computation

To estimate switching activity given m-tuple present state line probabilities, the topology of Figure 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.

7 Solving the Non-Linear System of Equations

We describe two methods to solve the non-linear system of equations obtained using k-unrolled or m-expanded networks. We will assume that the non-linear system can be represented as $P = G(P)$ or as $Y(P) = 0$ as described in Section 4.

7.1 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.

$$p_1 = g_1(p_1,\ p_2,\ \cdots,\ p_N)$$

$$p_2 = g_2(p_1,\ p_2,\ \cdots,\ p_N)$$

$$\cdots$$

$$p_N = g_N(p_1,\ p_2,\ \cdots,\ p_N)$$

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 non-linear 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 7.3.

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

Theorem 7.2 *g_i is contractive on the domain [0,1].*

Proof. Choose any p_j. In order to perform the evaluation of $\frac{\partial g_i}{\partial p_j}$ 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:

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

If there are no redundant state lines, that is, $ns_i \neq ps_i$ for every i, then this partial differential is strictly less than one. ■

From Theorems 7.1 and 7.2, we can see that the iterated signal probability calculation is guaranteed to converge to a unique solution, provided P^* is such that no present state line has a probability of 0 or 1 in P^*.

7.2 Newton-Raphson Method

The Newton-Raphson method can be used to solve a non-linear system of equations given an initial guess at the solution. The advantage with 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 k^{th} Newton iteration as the linear system solve shown below.

$$J(P^k) \times P^{k+1} \;=\; J(P^k) \times P^k - Y(P^k) \tag{4}$$

where J is the $N \times N$ Jacobian matrix of the system of equations. Each entry in J corresponds to a $\frac{\partial y_i}{\partial p_j}$ 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 7.3 to evaluate:

$$g_i(p_1,\ p_2,\ \cdots,\ p_N) \;=\; prob(f_i(i_1,\ i_2,\ \cdots,\ i_M,\ ps_1,\ ps_2,\ \cdots,\ ps_N))$$

for given $p_j = prob(ps_j)$'s and $prob(i_k)$'s. The $Y(P^k)$ of Eqns. 4 can easily be evaluated using the ${p_j}^k$ values and using Eqns. 2.

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

In order to perform the evaluation of $\frac{\partial g_i}{\partial p_j}$ 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 7.3.

As an example consider:

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

$$\frac{\partial g_1}{\partial p_1} \;=\; prob(i_1 \wedge \overline{ps_2}) - prob(i_1 \wedge ps_2)$$

$$\frac{\partial g_1}{\partial p_1} \;=\; 0.5 \cdot (1 - p_2) - 0.5 \cdot p_2 \;=\; 0.5 - p_2$$

which is exactly what we would have obtained had we differentiated Eqn. 3 with respect to p_1.

Theorem 7.3 [9] *The Newton iterates:*

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

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 Eqn. 2, then $\gamma < 2$.*

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 $\frac{\partial y_i}{\partial p_j}$. Using the equations provided in the proof of Theorem 7.2 we can write:

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

Differentiating with respect to p_k we have:

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

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

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

■

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.

7.3 Signal Probability Evaluation

In the non-linear solve, 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.

8 Experimental Results on Power Estimation

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 datapath 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 accurate, robust and computationally efficient methods for sequential power estimation.

In Table 1, 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 4 and using the Newton-Raphson method 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 3. The zero delay model was assumed, however, any other delay model could have been used instead.

The first set of circuits correspond to finite state machine controllers. These circuits typically have the characteristic that the state probabilities are highly non-uniform. 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 4 produces highly accurate estimates when compared to exact state probability calculation.

The second set of circuits correspond 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.

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

Table 1: Comparison of sequential power estimation methods

The third set of circuits correspond to pipelined adders and a pipelined multiplier. For pipelined circuits, exact power estimation is possible without resort to Chapman-Kolmogorov equation solving [7]. The fourth set to mixed datapath/control circuits from the ISCAS-89 benchmark set. Exact state probability evaluation is not possible for these circuits.

Circuit	Picard-Peano			Newton-Raphson		
Name	#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

Table 2: Comparison of Picard-Peano and Newton-Raphson

The CPU times in the table correspond to seconds on a SUN SPARC-2. The CPU times correspond to times required for symbolic simulation to estimate combinational switching activity using BDD-based symbolic simulation plus the time required for the calculation of state/line probabilities.

Next, we present results comparing the Picard-Peano and Newton-Raphson methods to solve the non-linear equations of Section 4. These results are summarized in Table 2. The number of iterations required for the Picard-Peano and Newton-Raphson methods are given in Table 2 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.

The convergence criterion allowed a maximum error of 1% in the line probabilities. In this case, the Picard-Peano method outperforms 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 4) is more than 1% 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 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 3, we present results that indicate the improvement in accuracy when k-unrolled or $m-$expanded networks are used. Results are presented for the finite state machine circuits of Table 1 for $0 \leq k \leq 2$ and $1 \leq m \leq 4$. The percentage differences in power from the exact power estimate are given. In general, if $k \rightarrow \infty$, the error will reduce to 0%, however, increasing k when k is small is not guaranteed to reduce the error. The m-expansion-based method behaves more predictably for this set of examples, however, no guarantees can be made regarding the improvement in accuracy 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.

Circuit	Initial	k-Unrolled Error		m-Expanded Error	
Name	Error	$k = 1$	$k = 2$	$m = 2$	$m = 4$
cse	1.06	0.33	0.02	0.53	0.01
dfile	0.67	0.20	0.20	0.56	0.32
keyb	1.02	0.02	0.04	0.51	0.42
mod12	1.13	0.85	0.30	1.1	0.0
planet	0.11	0.15	1.72	0.10	0.05
sand	0.76	0.61	0.29	0.64	0.27
styr	3.85	0.16	0.41	1.82	0.89
tbk	2.46	1.52	0.12	2.21	0.50

Table 3: Results of estimation based on k-unrolled and m-expanded networks

9 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 non-linear 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.

Ongoing work includes the application of estimation methods to drive sequential logic synthesis methods that target low power dissipation.

References

[1] R. Bryant. Graph-Based Algorithms for Boolean Function Manipulation. *IEEE Transactions on Computers*, C-35(8):677–691, August 1986.

[2] R. Burch, F. Najm, P. Yang, and T. Trick. McPOWER: A Monte Carlo Approach to Power Estimation. In *Proceedings of the Int'l Conference on Computer-Aided Design*, pages 90–97, November 1992.

[3] A. Chandrakasan, T. Sheng, and R. W. Brodersen. Low Power CMOS Digital Design. In *Journal of Solid State Circuits*, pages 473–484, April 1992.

[4] A. Ghosh, S. Devadas, K. Keutzer, and J. White. Estimation of Average Switching Activity in Combinational and Sequential Circuits. In *Proceedings of the* 29^{th} *Design Automation Conference*, pages 253–259, June 1992.

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

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

[7] J. Monteiro, S. Devadas, and A. Ghosh. Retiming Sequential Circuits for Low Power. In *Proceedings of the Int'l Conference on Computer-Aided Design*, pages 398–402, November 1993.

[8] F. Najm. Transition Density, A Stochastic Measure of Activity in Digital Circuits. In *Proceedings of the* 28^{th} *Design Automation Conference*, pages 644–649, June 1991.

[9] J. M. Ortega and W. C. Rheinboldt. *Iterative Solution of Nonlinear Equations in Several Variables*. Academic Press, Inc., Boston, MA, 1970.

[10] A. Papoulis. *Probability, Random Variables and Stochastic Processes*. McGraw-Hill, 3^{rd} edition, 1991.

[11] C. Y. Tsui, M. Pedram, and A. Despain. Efficient Estimation of Dynamic Power Dissipation under a Real Delay Model. In *Proceedings of the Int'l Conference on Computer-Aided Design*, pages 224–228, Nov 1993.