A Methodology for Efficient Estimation of Switching Activity in Sequential Logic Circuits

José Monteiro, Srinivas Devadas
Department of EECS, MIT, Cambridge

Bill Lin
IMEC, Leuven

Abstract

We describe a computationally efficient scheme to approximate average switching activity in sequential circuits which requires the solution of a *non-linear system of equations of size N*, where the variables correspond to state *line probabilities*. We show that the approximation method is within 3% of the exact Chapman-Kolmogorov method, but is orders of magnitude faster for large circuits. Previous sequential switching activity estimation methods can have significantly greater inaccuracies.

1 Introduction

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. But 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]) 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 modeled by randomly applied vector pairs. In the case of sequential circuits,

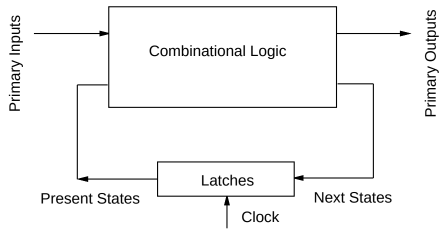

Figure 1: A Synchronous Sequential Circuit

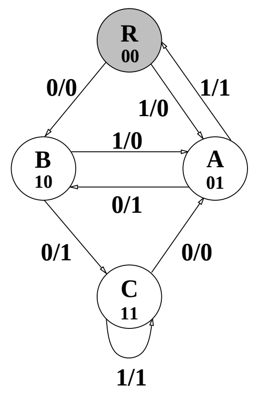

Figure 2: Example State Transition Graph

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 (*cf.* 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 [2]. 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 se-

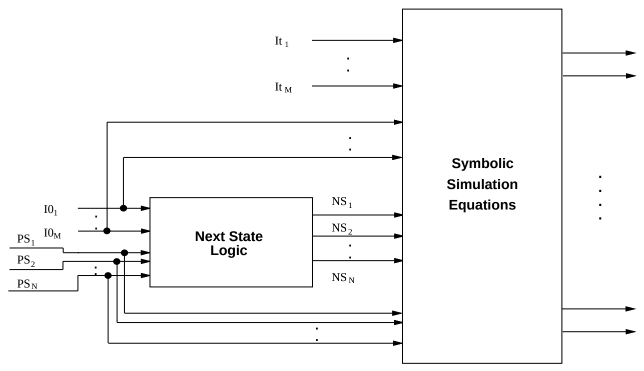

Figure 3: Taking Correlation Into Account

quential circuits such as pipelines [3], but not for more general cyclic circuits.

In this paper, we present results obtained by using the Chapman-Kolmogorov equations for discrete-time Markov Chains [6] 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 describe a computationally efficient scheme to approximate average switching activity which requires the solution of a *non-linear system of equations of size* N, where the variables are the state *line probabilities.* We show that the approximation method is within 3% of the exact method, but is orders of magnitude faster for large circuits. Previous methods that approximate switching activity for sequential circuits can have significantly greater inaccuracies.

The model for relating switching activity to power dissipation and the method for combinational logic power estimation are described in [2].

2 Modeling Correlation and Computing Exact State Probabilities

2.1 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 [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 [2].

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.

2.2 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 [6]. 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 $e.Curr$ signifying the state that the edge fans out from, $e.Next$ signifying the state that the edge fans out to, and $e.In$ signifying the input combination corresponding to the edge. Given static probabilities for the primary inputs to the machine, we can compute $prob(In)$, the probability of the combination In occurring. [1] We can compute $prob(e.In)$ using:

$$prob(e.In) \ = \ prob(e.Curr) \times prob(In)$$

For each state s_i we can write an equation:

$$prob(s_i) \ = \sum_{\forall\ e,\ e.Next\ =\ s_i} prob(e.In)$$

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:

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

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

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

In the State Transition Graph of Figure 2, the Chapman-Kolmogorov equations will produce the state probabilities, $prob(\mathbf{R}) = \frac{1}{6}$, $prob(\mathbf{A}) = \frac{1}{3}$, $prob(\mathbf{B}) = \frac{1}{4}$ and $prob(\mathbf{C}) = \frac{1}{4}$.

2.3 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 [2], the symbolic equations derive 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 [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 evaluation procedure has to appropriately weight these combinations according to the given probabilities.

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.

3 Computationally Efficient State Probability Estimation

3.1 State Probabilities and Line Probabilities

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.

$$\begin{array}{lclcl} prob(PS_1 = 0) & = & prob(00) + prob(01) & = & \frac{1}{2} \\ prob(PS_1 = 1) & = & prob(10) + prob(11) & = & \frac{1}{2} \\ prob(PS_2 = 0) & = & prob(00) + prob(10) & = & \frac{5}{12} \\ prob(PS_2 = 1) & = & prob(01) + prob(11) & = & \frac{7}{12} \end{array}$$

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 $prob(00) = \frac{1}{6}$.

Extensive experiments on benchmark circuits led us to the conclusion that using line probabilities and ignoring the correlation between the present state lines caused errors of less than 10%. Therefore, 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.

3.2 Computing State Line Probabilities

We wish to *directly* determine accurate line probabilities, without recourse to State Transition Graph extraction and/or state enumeration. This will lead to a substantially more efficient sequential power estimation method.

Our method is based on solving a non-linear system of equations 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.

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

We can write:

$$\begin{array}{lcl} prob(ns_1) & = & prob(f_1(i_1,\ \cdots,\ i_M,\ ps_1,\ \cdots,\ ps_N)) \\ prob(ns_2) & = & prob(f_2(i_1,\ \cdots,\ i_M,\ ps_1,\ \cdots,\ ps_N)) \\ & & \cdots \\ prob(ns_N) & = & prob(f_N(i_1,\ \cdots,\ i_M,\ ps_1,\ \cdots,\ ps_N)) \end{array}$$

where $prob(ns_i)$ is the probability that ns_i is a 1, and $prob(f_i(i_1,\ \cdots,\ i_M, ps_1,\ \cdots,\ ps_N))$ is the probability that $f_i(i_1,\ \cdots,\ i_M,\ ps_1,\ \cdots,\ ps_N)$ 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.

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

$$\begin{aligned} y_1 &= p_1 - g_1(p_1, \cdots, p_N) = 0 \\ y_2 &= p_2 - g_2(p_1, \cdots, p_N) = 0 \\ &\cdots \\ y_N &= p_N - g_N(p_1, \cdots, p_N) = 0 \end{aligned} \quad (1)$$

where the g_i's are non-linear functions of the p_i's. We will denote the above equations as $Y(P) = 0$. 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

$$g_1 = 0.5 \cdot (p_1 \cdot (1 - p_2) + (1 - p_1) \cdot p_2) \quad (2)$$

We can solve the equation set $Y(P) = 0$ to obtain the present state line probabilities. 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.

Obtaining a solution for the given non-linear system of equations requires the use of iterative techniques such as the Newton-Raphson method. Convergence depends on the initial guess for the present state line probabilities (*cf.* Section 3.5). We use an initial guess of $p_j = 0.5$ for $1 \leq j \leq N$. In practice, for a wide variety of examples, convergence is rapidly achieved to the line probabilities that would have been obtained using the Chapman-Kolmogorov equations (*cf.* Section 5).

3.3 Solving the Nonlinear System of Equations

The Newton-Raphson method can be used to solve a non-linear system of equations given an initial guess at the solution.

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.

$$J(P^k) \times P^{k+1} = J(P^k) \times P^k - Y(P^k) \quad (3)$$

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 achieved if each entry in $Y(P^k)$ is sufficiently small.

3.4 Evaluating the Jacobian

Given the $f_i(i_1, \cdots, i_M, ps_1, \cdots, ps_N)$ functions, there exist several methods to compute $g_i(p_1, \cdots, p_N) = prob(f_i(i_1, \cdots, i_M, ps_1, \cdots, ps_N))$ for given $p_j = prob(ps_j)$'s and $prob(i_k)$'s. We briefly describe these methods in Section 3.6. The $Y(P^k)$ of Eqns. 3 can easily be evaluated using the ${p_j}^k$ values and using Eqns. 1.

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 $1 - \frac{\partial g_i}{\partial p_i}$.

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:

$$\frac{\partial g_i}{\partial p_j} = prob(f_{i\ ps_j}) - prob(f_{i\ \overline{ps_j}}) \quad (4)$$

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 3.6.

As an example consider:

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

which is exactly what we would have obtained had we differentiated Eqn. 2 with respect to p_1.

3.5 Convergence Proof

Theorem 3.1 [5] *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* $||J(P^0)^{-1}|| \leq \beta$, $\eta \geq ||J(P^k)^{-1} Y(P^0)||$ *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 3.2 *If Y is given by Eqn. 1, 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 $\frac{\partial y_i}{\partial p_j}$. Using Equation 4 we can write:

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

Differentiating with respect to p_k we have:

$$\begin{aligned} \frac{\partial^2 y_i}{\partial p_j p_k} &= prob(f_{i\ \overline{ps_j} ps_k}) - prob(f_{i\ \overline{ps_j}\,\overline{ps_k}}) - \\ & \quad prob(f_{i\ ps_j ps_k}) + prob(f_{i\ ps_j \overline{ps_k}}) \end{aligned}$$

We can write:

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

since the probabilities are between 0 and 1. ■

Condition 3 in Theorem 3.1 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.

3.6 Signal Probability Evaluation

During power estimation, we repeatedly evaluate the signal probability of a Boolean function given input probabilities, *i.e.* compute $prob(f_i(i_1, \cdots, i_M, ps_1, \cdots, ps_N))$ 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. Given a OBDD representation of a function, the signal probabilities of all of the cofactors with respect to different present state variables can be computed in a single pass over the OBDD. This implies that the Jacobian can be evaluated using a single pass over all the OBDD's for the f_i functions.

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

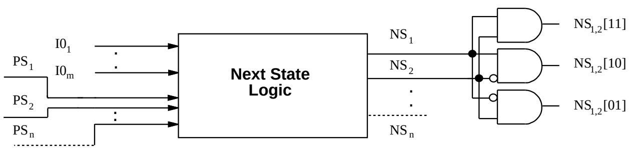

Figure 4: An m-Expanded Network with $m = 2$

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.

4 Improving Accuracy Using m-Expanded Networks

4.1 State Line Probability Computation

We describe a method to improve the accuracy of the basic approximation strategy outlined in Section 3. This method models the correlation between m-tuples of present state lines. The method is pictorially illustrated in Figure 4 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 3.6 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.

4.2 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.

5 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 datapath circuits.
- Purely combinational logic estimates result in significant inaccuracies.
- Assuming uniform probabilities for the present state line probabilities and state probabilities as in [2] can result in significant inaccuracies in power estimates.
- Computing the present state *line* probabilities using the technique presented in previous sections results in an accurate, robust and computationally efficient method 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 [2] assuming the unit delay model, **uniform-prob** corresponds to the sequential estimation method of [2] that assumes uniform state probabilities, **line-prob** corresponds to the technique of Section 3, and **state-prob** corresponds to the exact state probability calculation method of Section 2. For each method the power estimate, the percentage error with respect to the exact method, and the CPU time required to perform the estimate are reported.

The number of gates and flip-flops in each circuit is given in the first two columns of the table. 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 3 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.

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 [3]. In the fourth set are mixed datapath/control circuits from the ISCAS-89 benchmark set. Exact state probability evaluation is not possible for these circuits.

All the CPU times are in seconds on a DEC AXP 3000/500. The CPU times correspond to times required for symbolic simulation to estimate combinational switching activity using OBDD-based symbolic simulation plus the time required for the calculation of state/line probabilities. An interesting observation is that the **uniform-prob** and **line-prob** methods are very close in their CPU times, indicating that the amount of time required to solve the non-linear equations as compared to combinational power estimation is relatively small.

Once the state/line probabilities have been calculated, other methods for combinational switching activity such as [4] can be applied. However, these methods have to be extended to account for the correlation between the applied vector pairs modeled by the next state logic block in Figure 3.

The accuracy of power estimation can be improved by using the notion of a m-expanded network. For the finite state machine controller examples of Table 1, we show how increasing m results in increased accuracy in Table 2. The computed power dissipation, the percentage error with respect to the computed power dissipation using the exact state probabilities and the CPU times required for estimation are given for $m = 2$ and $m = 4$ in Table 2. The computed power dissipations using $m = 1$ (**line-prob**) and exact state probabilities (**state-prob**) for these examples were given in Table 1.

Acknowledgements:

This research was supported in part by the Advanced Research Projects Agency under contract N00014-91-J-1698, in part by the Portuguese "Junta Nacional de Investigação Científica e Tecnológica" under project "Ciência" and in part by a NSF Young Investigator Award with matching funds from Mitsubishi and IBM Corporation.

Circuit Name	#gate	#ff	Combinational			Uniform Prob.			Line Prob.			State Prob.	
			power	err	cpu	power	err	cpu	power	err	cpu	power	cpu
cse	132	4	739.8	55.8	6	671.0	41.3	5	470.1	1.0	5	475.0	6
dk16	180	5	1344.9	3.3	7	1359.6	4.4	6	1309.3	0.5	6	1302.5	7
donfile	119	5	1105.4	34.5	3	832.5	1.3	3	832.3	1.3	3	821.7	4
keyb	169	5	867.6	40.4	13	806.1	30.5	8	614.8	0.5	8	617.9	8
modulo12	25	4	285.5	23.7	0	220.1	4.7	0	225.3	2.5	0	230.9	0
planet	327	6	2421.4	9.1	32	2641.6	0.8	18	2470.4	7.3	18	2664.0	19
sand	336	5	1939.2	38.1	36	1539.7	9.6	23	1371.4	2.3	23	1404.3	24
shiftreg	9	3	140.6	3.4	0	145.6	0.0	0	145.6	0.0	0	145.6	0
styr	313	5	1798.5	48.5	30	1535.5	26.8	19	1239.5	2.3	19	1211.5	19
tbk	478	5	2369.1	21.0	76	2184.3	11.6	46	1883.0	3.8	46	1957.6	54
accum4	45	4	412.8	4.2	1	431.0	0.0	1	431.0	0.0	1	431.0	2
accum8	89	8	859.6	6.6	3	920.1	0.0	17	920.1	0.0	17	920.1	246
accum16	245	16	1521.2	-	4	1596.3	-	33	1596.3	-	33	**unable**	
count4	19	4	286.3	19.0	0	240.5	0.0	0	240.5	0.0	0	240.5	0
count7	35	7	551.3	17.0	1	471.4	0.0	1	471.4	0.0	1	471.4	2
count8	40	8	657.4	17.2	1	561.0	0.0	1	561.0	0.0	1	561.0	3
cbp32.4	489	223	9906.9	7.7	48	10127.1	10.1	56	9002.7	2.1	65	9197.1	72
addrpl16	214	98	4132.5	2.7	5	4199.1	4.3	5	3954.8	1.8	6	4025.1	8
mult8	176	87	8759.1	43.5	140	8468.7	38.7	187	6674.2	9.3	190	6104.2	271
s953	418	29	2105.3	-	21	2267.6	-	24	1125.8	-	25	**unable**	
s1196	529	18	3149.7	-	241	3117.3	-	82	2859.0	-	82	**unable**	
s1238	508	18	3328.4	-	68	3292.9	-	167	3032.1	-	167	**unable**	
s1423	657	74	6017.1	-	251	4734.2	-	271	7087.1	-	289	**unable**	

Table 1: Comparison of sequential power estimation methods

Circuit Name	$m = 2$			$m = 4$		
	power	err	cpu	power	err	cpu
cse	472.7	0.5	6	475.0	0.0	6
dk16	1308.6	0.5	6	1304.2	0.1	6
dfile	830.1	1.0	4	827.8	0.7	4
keyb	614.9	0.5	8	620.3	0.4	8
mod12	226.2	2.1	0	230.9	0.0	0
planet	2555.9	4.1	19	2569.2	3.6	20
sand	1372.8	2.2	23	1407.5	0.2	23
sreg	145.6	0.0	0	145.6	0.0	0
styr	1233.4	1.8	18	1201.6	0.8	19
tbk	1902.9	2.8	50	1947.9	0.5	51

Table 2: Results using m-expanded networks

References

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

[2] 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.

[3] 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.

[4] F. Najm. Transition Density: A New Measure of Activity in Digital Circuits. *IEEE Transactions on Computer-Aided Design*, 12(2):310–323, February 1993.

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

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