Exact and Approximate Methods for Calculating Signal and Transition Probabilities in FSMs

Chi-Ying Tsui, Massoud Pedram
and Alvin M. Despain

CENG Technical Report 93-43

Department of Electrical Engineering - Systems
University of Southern California
Los Angeles, California 90089-2562
(213)740-6006

September 1993

Exact and Approximate Methods for Calculating Signal and Transition Probabilities in FSMs

Chi-Ying Tsui, Massoud Pedram, Alvin M. Despain

15 October 1993

Abstract

In this paper, we consider the problem of calculating the signal and transition probabilities of the internal nodes of the combinational logic part of a finite state machine (FSM). Given the state transition graph (STG) of the FSM, we first calculate the state probabilities by iteratively solving the Chapman-Kolmogorov equations. Using these probabilities, we then calculate the exact signal and transition probabilities by an implicit state enumeration procedure. For large sequential machines where the STG cannot be explicitly built, we unroll the next state logic k times and estimate the signal probability of the state bits using an OBDD-based approach. We then use these estimates to approximately calculate signal and transition probabilities of the internal nodes. We show that a small value of k is sufficient to produce accurate probabilities. Our experimental results indicate that the average error of signal and transition probabilities (compared to the exact method) is only 5% when $k = 3$. This is an order of magnitude improvement in computation accuracy compared to existing approaches.

Contents

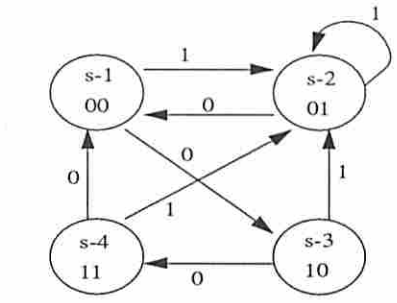
1 Introduction

As portable electronics shifting from conventional low performance products to high throughput and computationally intensive applications, power consumption becomes one of the critical criteria for IC design. Previous research has been done on low power synthesis at logic and physical levels. Low power algorithms for node minimization, kernelization, technology decomposition and mapping, state assignment and placement have been developed[11], [9],[14],[12],[10]. To evaluate the circuits generated from different low power design tools, an accurate power analysis tool is required.

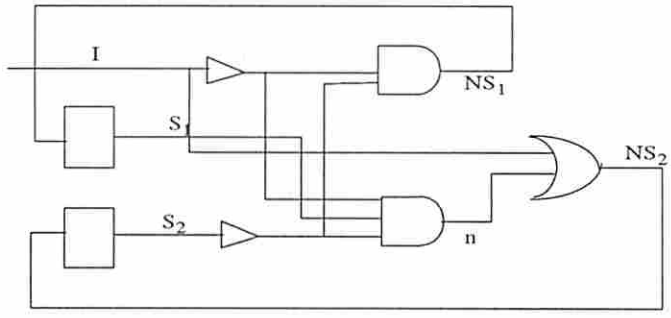
In CMOS circuits, power is consumed during charging and discharging of the load capacitance. In order to estimate the power consumption, we have to calculate the signal and transition probabilities of the internal nodes of the circuit. Signal and transition probabilities depend on the input patterns, the delay model, and the circuit structure.

Several signal and transition probabilities estimation algorithms have been developed for combinational circuit. Burch et al. [2] introduce the concept of a probability waveform. Given such waveforms at the primary inputs and with some convenient partitioning of the circuit, they examine every sub-circuit and derive corresponding waveforms at the internal circuit nodes. Najm [7] describes an efficient technique to propagate the transition densities at the circuit primary inputs into the circuit to give transition densities at internal and output nodes. These methods assume inputs to sub-circuits are independent and thus do not account for the reconvergent fanout and input correlations. Ghosh et al. [4] propose symbolic simulation in order to produce a set of Boolean functions which represent conditions for switching at each gate in the circuit. Given input switching rates, the switching probability at each gate is calculated by performing a linear traversal of the Ordered Binary Decision Diagrams (OBDDs) [1] representation of the corresponding Boolean function [7]. A general delay model which correctly computes the Boolean conditions that cause glitchings is used and correlations due to the reconvergence of input signals are taken into account. Tsui et al. [13] describe an efficient tagged probabilistic simulation approach which employs a real delay model to account for glitchings and also handles reconvergent fanout. This approach requires much less memory and runs much faster than symbolic simulation, yet achieves a very high accuracy.

The above methods assume the primary inputs to the circuit are both spatially and temporally independent, i.e. the signal value x_i of a primary input i is independent of any other primary input, and x_i at time instance t is independent of x_i at time instance $t+1$. While this assumption holds for most combinational circuits, it does not hold for finite state machines where the present state bit inputs are spatially correlated by the state encoding and temporally correlated by the state transition behavior. Figure 1 shows the STG and the gate implementation of a 4-state finite state machine. If the state bits are assumed spatially independent, the signal probability of n ($P(n)$) is equal



a) State Transition Graph



b) Gate Implementation

Figure 1: State transition graph and gate implementation of a 4-state finite state machine.

to 0.5*0.5*0.5 = 0.125. However, n will evaluate to 1 only if $s_1 s_2 = 10$ and input i is 0, hence $P(n)$ = state probability of state-3 *0.5 = 0.0721. Furthermore, if the state input bits are assumed temporally independent, the transition probability of n ($P_{1->0}(n)$) is equal to 0.125*(1-0.125) = 0.1093. However, when the present state is state-3 and the input is 0, the next state is state-4 which always forces n to 0. Therefore, the actual transition probability of n ($P_{1->0}(n)$) is equal to 0.0721.

Let $f_t(n)$ denotes the Boolean value of n at time t. The output of n will switch exactly if:

$$g(n) = f_0(n) \oplus f_t(n) \tag{1}$$

evaluates to 1. The signal probability of g is thus equal to the transition probability of n. Let PS_0 and PI_0 be the set of state bit inputs and primary inputs for $f_0(n)$ and PS_t and PI_t be the set of state bit inputs and primary inputs for $f_t(n)$. PS_0, PS_t and PI_0 are correlated by the state transition behavior of the FSM.

Based on the above observation, Ghost et al. [4] propose an approximate method, which can capture the temporal correlation between the state bits at consecutive time

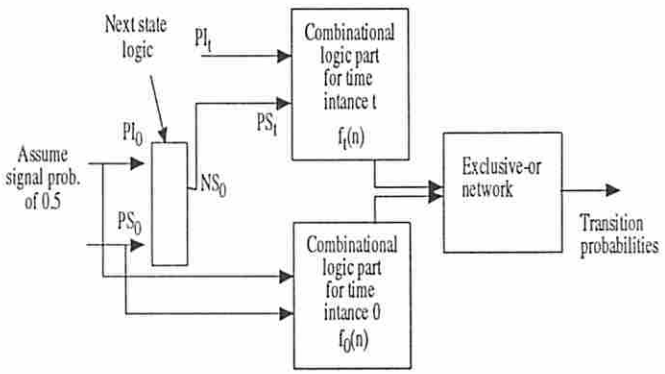


Figure 2: Capture temporal correlation of state bits by next state logic.

instances, to estimate the power consumption of an FSM. First the next state logic is duplicated and cascaded to the combinational part of the FSM. Then the exclusive-or network of the internal nodes of the combination part is built (Figure 2). Signal probabilities of the exclusive-or nodes are calculated using symbolic simulation based on the assumption that the present state probabilities are equal, i.e., the present-state lines of the next state logic are assumed to be uncorrelated and have signal probability of 0.5. This method, to some degree, captures the temporal correlation between PS_0 and PS_t and the spatial correlation among bits of PS_t. However it is inaccurate for two reasons. Firstly, the state bit signal probabilities are incorrect: Signal probabilities of PS_0 are assumed to be 0.5 and those of PS_t are calculated based on this assumption. Secondly, correlations among bits of PS_0 are not captured. Indeed, since PS_0 and PS_t are inputs and outputs of the same set of latches, their probabilistic behavior should be the same. Therefore, the signal probabilities and correlations for PS_o and PS_t should be the same.

In this paper, we present an exact method to calculate the signal and transition probabilities of the internal nodes of a FSM given its STG. The procedure is based on the notion of symbolic state support and primary input support of a node. Given a STG, the exact state probability can be derived from the Chapman-Kolmogorov equations for discrete-state discrete-transition Markov Processes [8]. We describe a fast solution by assigning initial probabilities to the states and then solving equations for the new probabilities by simple substitution. This procedure is iterated until each state probability reaches the desired bit precision. Using these probabilities, we then calculate the exact signal and transition probabilities by an implicit state enumeration procedure. We also demonstrate that accurate state bit signal probabilities can be obtained by unrolling and cascading the next state logic of the FSM. We denote this as a *k-unrolled* network. Based on the *k-unrolled* network, we present an approximate transition prob-

ability calculation procedure for large sequential machines which captures the spatial and temporal correlations among the state bits.

The rest of this paper is organized as follows. In Section 2 we describe an iterative procedure for calculating the exact state probabilities. Exact and approximate methods for calculating the signal and transition probabilities for internal nodes of the combinational parts of the FSMs are described in Sections 3 and 4, respectively. Experimental results and conclusions are presented in Sections 5 and 6.

2 Calculation of the State Probabilities

The correlation between the state bit input lines depends on the state encoding and the occurrence probability of each state which in turn is dictated by the STG. In this section, we show how to calculate the state probabilities.

2.1 Definition

A state transition graph is denoted by $G(V, E)$ where vertex $S_i \in V$ represents a state of the FSM and an edge $e_{i,j} \in E$ represents a transition from S_i to S_j. Let P_{S_i} denote the state probability, that is the probability of the machine being in S_i, and p_{ij} denote the transition probability, that is, the probability of the machine making a transition from S_i to S_j. A transition matrix Γ_m is an $M \times M$ matrix where p_{ij} is the entry in the i^{th} row and j^{th} column where M is the number of states. We further denote the n-step transition probability from S_i at time instance k to S_j at time instance $k+n$ as $p_{ij}(n)$. S_i is *transient* if there exist S_j and integer n such that $p_{ij}(n) > 0$, but $p_{ji}(l) = 0$ for all l. It means that it is possible to pass from S_i into other state, but it is no longer possible to return from that state to S_i. All other states are called *recurrent*. A recurrent state is called *periodic* if there exists an integer d with $d > 1$, such that $p_{ij}(k)$ is equal to zero for all values of k other than $d, 2d, 3d$, etc. A set of essential states forms a *single chain* if, for every pair of essential states S_i and S_j, there exists an integer r_{ij} such that $p_{ij}(r_{ij}) > 0$.

2.2 State Probability Calculation

A finite state machine is a discrete-state discrete-transition Markov process. Given 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 by the *Limiting State*

Probability Theorem[3],

$$\lim_{n->\infty} p_{ij}(n) = P_{S_j} \quad j = 1, 2, \ldots, M \tag{2}$$
$$\sum_j P_{S_j} = 1$$

The state probabilities can then be obtained by solving the Chapman-Kolmogorov equations [8] as follows:

$$\begin{aligned} P_{S_i} &= \textstyle\sum_{j \in IN\_STATE(i)} p_{ji} P_{S_j} \quad i = 1, 2, \ldots, M-1 \\ 1 &= \textstyle\sum_j P_{S_j} \end{aligned} \tag{3}$$

where $IN\_STATE(i)$ is the set of fanin states of i in the STG.

Alternatively, we could solve for S_i's as follows. Let $P_{S_i}(n)$ be the probability of S_i after n cycles. For a discrete-state discrete-transition Markov process, $P_{S_i}(n)$ and $P_{S_i}(n+1)$ are related by:

$$\begin{aligned} P_{S_i}(n+1) &= \textstyle\sum_{j \in IN\_STATE(i)} p_{ji} P_{S_j}(n) \quad i = 1, 2, \ldots, M-1 \\ 1 &= \textstyle\sum_j P_{S_j}(n+1). \end{aligned} \tag{4}$$

Given a set of initial condition $P_{S_1}(0), P_{S_2}(0), \ldots, P_{S_M}(0)$, these equations can be solved iteratively for $n = 0, 1, 2, \ldots$ to determine the state probabilities as a function of n.

This process is continued until the state probabilities converge, that is, the difference between $P_{S_i}(n+1)$ and $P_{S_i}(n)$ for all states is within a user defined parameter.

3 Exact Calculation of Signal and Transition Probabilities

In this section, we present an exact method for calculating the signal and transition probabilities of internal nodes of a FSM given its logic implementation. The state probabilities are calculated from the STG using the procedure described in Section 2.

3.1 Terminology

Let $S = S_1, S_2, \ldots, S_M$ be the set of states of the FSM, $s = s_1, s_2, \ldots, s_N$ ($N \geq \lceil log_2 M \rceil$) be the set of state bits, $PI = i_1, i_2, \ldots, i_K$ be the set of primary inputs, and n be an internal node in the combinational circuit of the FSM.

Suppose f is a disjoint cover of the function computed by n, i.e.,

$$f = \sum_{m \in Disjoint\_Cover(n)} C_m \tag{5}$$

where C_m is a cube of the disjoint cover. C_m is a function of s and PI. 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 $C_m = SS_m I_m$.

We define the symbolic state support of node n (denoted by $SS(n)$) as the union of the symbolic state supports of all the cubes in its disjoint cover. The importance of the symbolic state support of n is that if the machine is in a state S_i which is an element of $SS(n)$ then n may evaluate to 1 depending on the primary inputs. Otherwise, n will evaluate to 0. We further define the auxiliary input function of n given S_i (denoted by $AUX\_I(n|S_i)$) as a Boolean function (of the primary inputs) which forces n to 1 given that the present state of the machine is S_i. Note that $AUX\_I(n|S_i)$ is the co-factor of f with respect to the cube $(s_1, s_2, \ldots, s_i)$ which corresponds to the encoding of S_i.

3.2 Signal Probability Calculation

Given a disjoint cover of node n, the signal probability of n is given by:

$$P(n) = \sum_{m \in Disjoint\_Cover(n)} P(C_m). \tag{6}$$

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

$$\begin{aligned} P(C_m) &= P(I_m)P(SS_m) \\ &= P(I_m) \sum_{S_i \in SS_m} P(S_i). \end{aligned} \tag{7}$$

From equations (6) and (7), we have:

$$P(n) = \sum_{m \in Disjoint\_Cover(n)} P(I_m) \sum_{S_i \in SS_m} P(S_i). \tag{8}$$

Theorem 3.1 *Given the state probabilities, the signal probability of node n is given by:*

$$P(n) = \sum_{S_i \in SS(n)} P(S_i)P(AUX\_I(n|S_i)) \tag{9}$$

Proof equation (8) can be rewritten:

$$P(n) = \sum_{S_i \in SS(n)} P(S_i) \sum_{I_m \in I_{S_i}} P(I_m) \tag{10}$$

where I_{S_i} is the set of primary input supports of the disjoint cubes C_m such that $S_i \in SS_m$. This is, however, exactly the definition of $AUX\_I(n|S_i)$, that is,

$$\sum_{I_m \in I_{S_i}} P(I_m) = P(AUX\_I(n|S_i)). \tag{11}$$

■

Equation (9) requires explicit enumeration of the states in $SS(n)$ and is very costly. In practice, we can do the calculation by implicit enumeration of states using OBDDs.

In [7], Najm et al. propose a procedure for calculating the signal probability at the output of node n by first building an OBDD corresponding to the global function of the node. This OBDD is then traversed in postorder using

$$P(n) = P(x_i)P(f_{x_i}) + P(\bar{x}_i)P(f_{\bar{x}_i}) \tag{12}$$

where f is the global function of n, f_{x_i} and $f_{\bar{x}_i}$ are the Shannon's cofactors of f with respect to x_i and $\bar{x}_i$. The assumptions are that the signal probabilities of primary inputs are given and uncorrelated.

We modify this procedure to account for correlations of the state bit inputs. First an input variable order is derived as in [6]. The inputs are then partitioned into two groups, namely, the state bit input group and the primary input group, with the relative order within each group maintained. The final variable order is obtained by putting all the state bit variables before the primary input variables. OBDD for n is built according to this variable order (See Figure 3).

When traversal on the OBDD reaches a node m that is the first primary input variable, paths from the OBDD root to the node identify a subset SS_i of $SS(n)$ while the OBDD starting from that node corresponds to the $AUX\_I(n|S_i)$ where S_i is any element of the SS_i (Figure 3). Actually SS_i is the set of states that share the same $AUX\_I(n|S_i)$. Equation (9) is then equal to:

$$P(n) = \sum_{SS_i \subset SS(n)} P(SS_i)P(AUX\_I(n|SS_i)) \tag{13}$$

The probability of the auxiliary input function can be found by using a procedure similar to [7] as the signal probabilities of the primary inputs are given and assumed to be uncorrelated. Figure 4 shows the definitions and calculation of the signal probability for node n in Figure 1. The above procedure is summarized in the following pseudo code.

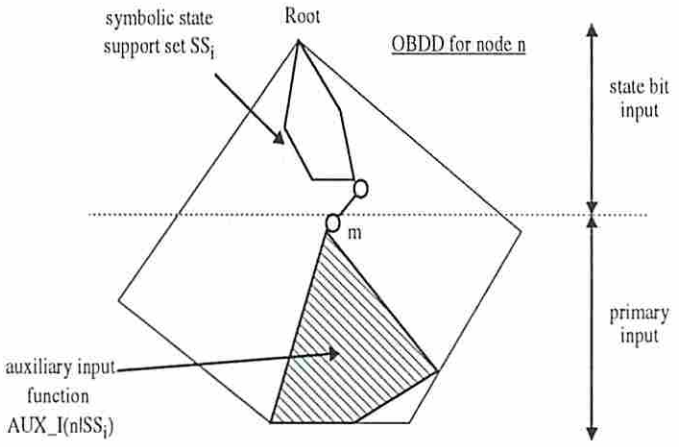


Figure 3: OBDD input variable order.

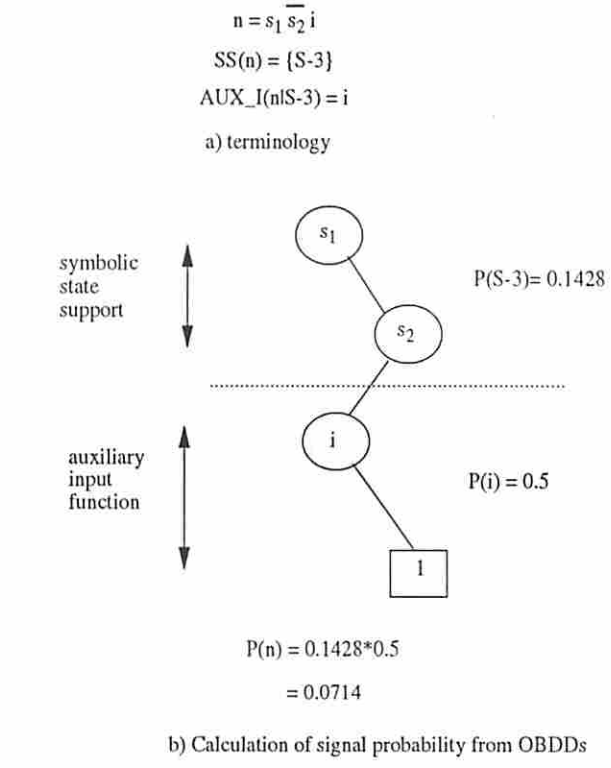


Figure 4: Terminology with an example.

```
function calc_signal_probability(FSM)
begin
   T_sp = build_state_prob_tree(FSM);
   foreach node n in the combinational logic part {
      n.signal_probability = 0;
      bdd = build_bdd(n);
      traverse the bdd in depth_first manner {
         if m is the first PI encountered in the bdd {
            states = path_to_root(m);
            state_probability = find_state_probability(T_sp,states);
            auxiliary_input_probability = find_probability_from_bdd( bdd_m);
            n.signal_probability += state_probability × auxiliary_input_probability;
         }
      }
   }
end
```

The above procedure can also be applied to FSMs where some states are assigned multiple binary codes (multicoding) as follows. Each multi-coded state is treated as a super-state and split into a number of substates with unique binary codes. Since the substates cannot be distinguished within the super-state, they have uniform state probability, i.e

$$P_{S_i^j} = \frac{P_{S_i}}{m} \tag{14}$$

where m is the number of distinct codes that S_i represents and $P_{S_i^j}$ is the state probability of the j^{th} sub-state of S_i. The above procedure will then produce the correct signal probabilities.

3.3 Transition Probability Calculation

The *1->0* transition probability of a node n is equal to:

$$P_{1->0}(n) = P(n(t) = 1)P(n(t+1) = 0|n(t) = 1) \tag{15}$$

where $P(n(t) = x)$ is the probability of n assuming x at time instance t and $P(n(t+1) = y|n(t) = z)$ is the conditional probability of n assuming y at time instance $t+1$ given it is z at time instance t.

The value of n at $t+1$ depends on the state of the machine at $t+1$ which is determined by the state and primary input vectors at t. Given state S_i and input minterm PI_i at

t, if $S_i \in SS(n)$ and $PI_i \subset AUX\_I(n|S_i)$, n evaluates to 1 while the next state S_j, is determined from the STG. The conditional probability of n being 1 if the state of the machine is S_j is actually the probability of $AUX\_I(n|S_j)$. Given that the machine state and primary inputs at t are S_i and PI_i, the conditional probability of n being 0 at time $t+1$ is given by:

$$\begin{aligned} P(n(t+1)=0|S_i(t) \wedge PI_i(t)) &= P(n=0|S_j) \\ &= 1 - P(AUX\_I(n|S_j)) \end{aligned} \tag{16}$$

where S_j is the next state of S_i under input PI_i.

Let $NS(S_i, AUX\_I(n|S_i))$ be the set of states that can be reached from S_i under the input function $AUX\_I(n|S_i)$, and $TR\_I(S_i, S_j)$ be the Boolean function (of primary inputs) that causes the transition from S_i to S_j.

Given the state probabilities for the FSM, the 1->0 transition probability [1] of a node n is given by:

$$\begin{aligned} P_{1->0}(n) = & \sum_{S_i \in SS(n)} P(S_i) P(AUX\_I(n|S_i)). \\ & \sum_{S_j \in NS(S_i, AUX\_I(n|S_i))} (1 - P(AUX\_I(n|S_j))). P(TR\_I(S_i, S_j)|AUX\_I(n|S_i)) \end{aligned} \tag{17}$$

Unfortunately, equation (17) cannot be implemented by an implicit state enumeration procedure as in equation (13) since different states in SS_i may go to different next states S_j under the same input function $AUX\_I(n|SS_i)$.

The transition probability calculation can, however, be reduced to a signal probability calculation based on equation (1) as shown in Figure 2. g is a function of PI_0,PI_t and PS_0. As these input vectors are assumed to be uncorrelated, Theorem 3.1 and the *calc_signal_prob* procedure can be used to obtain the exact signal probability of g, thus, the exact transition probability of n.

4 Approximate Calculation of Signal and Transition Probabilities

As the number of states is exponential to the number of flip flops, for sequential machines having large number of flip flops, we cannot explicitly build the STG, and thus the exact method cannot be applied. To calculate the signal and transition probabilities of the internal nodes, we have to use the signal and transition probabilities of the state bits.

[1] A similar expression can be written for the *0->1* transition probability.

The state bits are correlated and hence their signal probabilities are not 0.5 (which is the probability of a random input).In the following, we describe an approximate method for calculating the signal and transition probabilities using finite network unrolling followed by iterative solution of a system of non-linear equations.

4.1 Signal Probability Calculation

Given the state probabilities and the state encoding, the signal probability of state bit s_i is given by:

$$P(s_i) = \sum_{S_m \in EN_S(i)} P(S_m) \tag{18}$$

where $EN_S(i)$ is the set of states whose encodings have the i^{th} bit equal to 1.

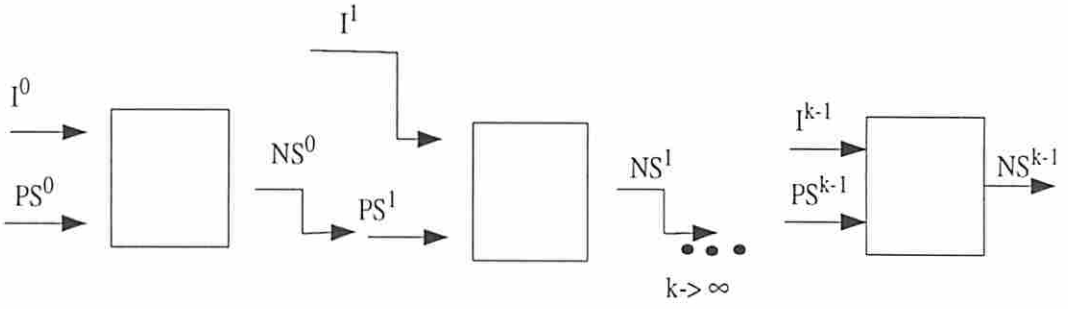
The state bit signal probabilities can be derived without explicitly calculating the state probabilities which is very costly for sequential machines with a large number of flip flops.

The transition behavior of the STG is implicitly captured by the next state logic of the FSM. Assuming the present state bits are uncorrelated and their signal probabilities are given, the characteristics (i.e. signal probabilities and correlations) of the next state bits can be obtained from the OBDD for the next state logic. If we unroll and cascade the next state logic k times to form a *k-unrolled* network as shown in Figure 5, we can then calculate the signal probabilities of the state bits at the k^{th} cycle given the state bit probabilities at the 0^{th} cycle.

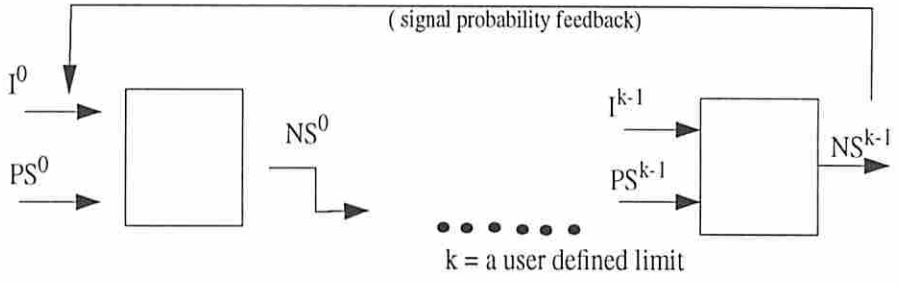
Theorem 4.1 *Given that the state bits (s_i) at time instance 0 are uncorrelated and have signal probability of 0.5, the state bit signal probabilities obtained by recursively applying equation (12) on the OBDD for the k-unrolled network, are identical to those calculated from equation (18) where the state probabilities are obtained by solving equation (4) k times, with initial conditions $P_{S_i} = \frac{1}{M}$.*

Proof Given a specific state encoding, a set of state probabilities maps to a unique set of state bit singal probabilities and correlations. Since the initial conditions for both methods are equivalent (uncorrelated state bits with signal probabilities equal to 0.5 imply that each state has uniform probability of $\frac{1}{M}$), the state bit characteristics captured by the OBDDs of the k-unrolled network maps to the set of the state probabilities obtained by solving equation (4) k times, and hence the signal probabilities obtained by both methods are the same. ∎

The exact state bit signal probabilities can be obtained by unrolling the next state logic ∞ times (Figure 5a). This is however impractical. We thus approximate the signal



(a)



(b)

Figure 5: Unrolling of the next state logic k times.

probabilities by unrolling the next state logic k times where k is a user defined parameter. We then feed back the signal probability values of the state bits at the output to the state bits at the input of the network and iterate until the signal probabilities converge (Figure 5b). We denote this method as the *signal probability feedback* method. Indeed, we still unroll the next state logic as many times as necessary to achieve convergence, but after every k stages, we only pass forward the signal probability values and disregard the correlations of the state bits.

The above iteration is known as the Picard-Peano iteration for finding a fixed-point of a system of non-linear equations, in particular, $x = f(x)$. The Picard-Peano method starts with an initial guess for x_0 and calculates $x_i = f(x_{i-1})$ until convergence is reached.

Theorem 4.2 *[5] If f is contractive, i.e., $|\frac{\partial f}{\partial x}| < 1$, then the Picard-Peano iteration method converges at least linearly.*

In the following, we derive the system of non-linear equations which is solved when calculating the state bit probabilities. Clearly,

$$\begin{aligned} \mathcal{N}_1^k &= \mathcal{F}_1(PI, \mathcal{P}_1^0, \mathcal{P}_2^0, \ldots) \\ \mathcal{N}_2^k &= \mathcal{F}_2(PI, \mathcal{P}_1^0, \mathcal{P}_2^0, \ldots) \\ &\cdots \\ \mathcal{N}_n^k &= \mathcal{F}_n(PI, \mathcal{P}_1^0, \mathcal{P}_2^0, \ldots) \end{aligned} \tag{19}$$

where $\mathcal{N}_i^k$ and $\mathcal{P}_j^0$ denote the i^{th} next state bit at the output and the j^{th} present state bit at the input of the k-unrolled network, respectively and $\mathcal{F}_l$'s are Boolean functions. Alternatively,

$$\begin{aligned} ns_1^k &= f_1(pi, ps_1^0, ps_2^0, \ldots, ps_n^0) \\ ns_2^k &= f_2(pi, ps_1^0, ps_2^0, \ldots, ps_n^0) \\ &\cdots \\ ns_n^k &= f_n(pi, ps_1^0, ps_2^0, \ldots, ps_n^0) \end{aligned} \tag{20}$$

where ns_i^k and ps_j^0 denote the state bit probabilities of the i^{th} next state bit at the output and the j^{th} present state bit at the input of the k-unrolled network, respectively and f_l's are nonlinear algebraic functions. Since we want to find the steady state bit probabilities, hence (21) becomes

$$\begin{aligned} ps_1 &= f_1(pi, ps_1, ps_2, \ldots, ps_n) \\ ps_2 &= f_2(pi, ps_1, ps_2, \ldots, ps_n) \end{aligned} \tag{21}$$

$$\ldots$$

$$ps_n = f_n(pi, ps_1, ps_2, \ldots, ps_n)$$

which is a system of non-linear equations.

Theorem 4.3 *f_i is contractive on the domain [0,1].*

Proof As $f_i = ps_j P(\mathcal{F}_i(\mathcal{P}_j = 1)) + (1 - ps_j)P(\mathcal{F}_i(\mathcal{P}_j = 0)$, we can write

$$\frac{\partial f_i}{\partial x_j} = P(\mathcal{F}_i(\mathcal{P}_j = 1)) - P(\mathcal{F}_i(\mathcal{P}_j = 0)).$$

If there are no redundant bits, that is, $\mathcal{N}_i \neq \mathcal{P}_i$ for every i, then this partial differential is strictly less than one. ∎

From theorems 4.2 and 4.3, we can see that the iterated signal probability calculation for the k-unrolled network is guaranteed to converge.

Note that the signal probabilities at the next state bit lines of the next state logic in Figure 2 are actually the result of signal probability calculation for a 1-unrolled network without any signal probability feedback. We will refer to this method as *1-unrolled, 0-stage feedback*, or simply *1u_0f*. Figure 6a shows the method used to calculate signal probability of the internal nodes of the FSM using the k-unrolled network with signal probability feedback.

4.2 Transition Probability Calculation

We enhance the transition probability calculation procedure of Figure 2 by using the concept of k-unrolled next state logic network. Instead of connecting the next static logic network to the exclusive-or network, we unroll the next static logic network k times and connect the next state bits of the k^{th} stage of the unrolled network, the next state bits of the $k - 1^{th}$ stage, and the primary input of the $k - 1^{th}$ stage to the combinational logic parts which are then exclusive-ored together (Figure 6b) [2]. By doing so, accurate state bit signal probabilities are used and the spatial and temporal correlations between PS_0 and PS_t are captured.

[2]This method can be improved as follows. The signal probability feedback is taken from of the next state output of the combinational part (time instance t version) of the FSM. In effect, this adds one more unrolling to the k-unrolled network and hence is more accurate.

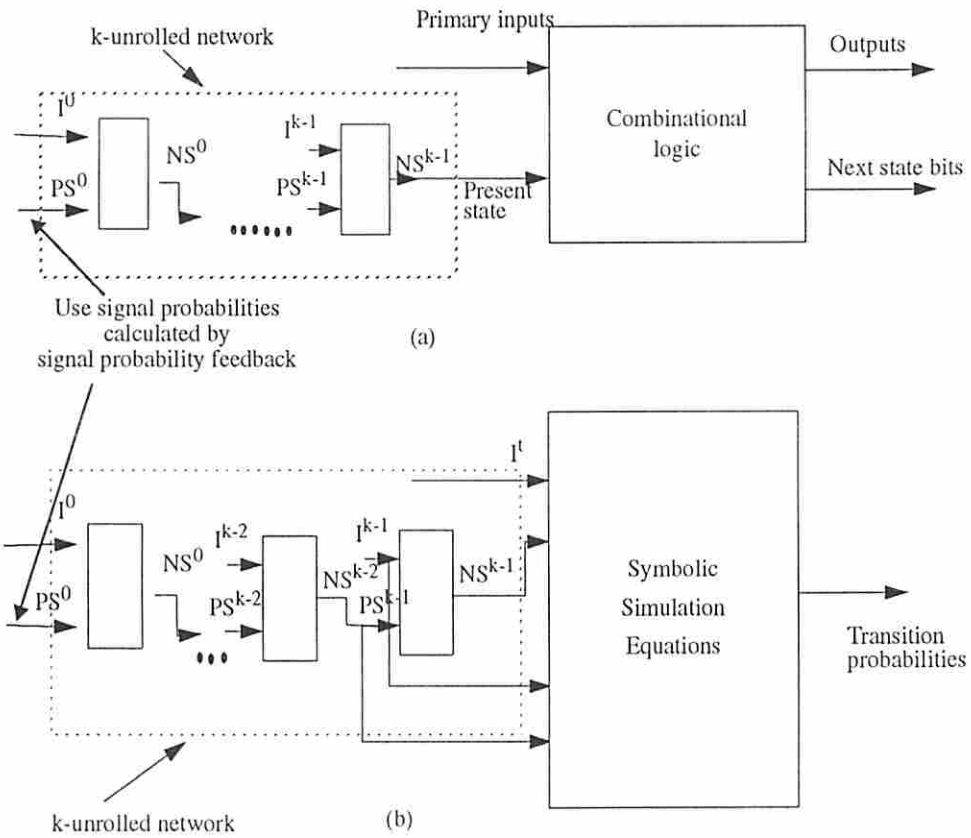


Figure 6: Calculation of signal and transition probabilities using k-unrolled network.

5 Experimental Results

To demonstrate the effectiveness of the signal and transition probability calculation procedures, we carried out several experiments using the full MCNC-91 and subsets of the ISCAS-89 sequential benchmark sets. The circuits were generated using the SIS mapper and an industrial gate library. Zero delay model was assumed. All experiments were carried out on a Sparc 2 workstation with 64MB memory. Power consumption measurement is based on the following model:

$$P_{avg} = 0.5 \times C_{load} \times \frac{V_{dd}^2}{T_{cycle}} \times E(transitions). \tag{22}$$

Table 1 shows the accuracy of calculating the state probabilities by iteratively solving equation (4). Results are compared to the exact values obtained by solving the Chapman-Kolmogorov equations for the STG. The process was terminated when the error was smaller than 0.5% or the number of iterations was greater than 100. The error was calculated as the sum of the absolute value of % error for each state probability divided by the number of states. It is seen that the average number of iterations needed to achieve the required accuracy is about 20.

Table 2 shows the accuracy of state bit signal probability calculation by unrolling the next state logic. Results are compared to the exact signal probability generated from equation (18) where the state probabilities were obtained by solving the Chapman-Kolmogorov equations. The values generated from the *1-unrolled* network without *signal probability feedback* (i.e. *1u_0f*) are also presented. For each FSM, the error is calculated by summing the absolute value of % error on all state bit signal probabilities divided by the number of state bits. As expected, error decreases when k increases and *signal probability feedback* is used. The average error is about 6% when k is equal to 3.

Table 3 shows the error in transition probability values using the network unrolling method. Results are compared with those obtained from the exact method described in Section 3.3. The results obtained from the method of Figure 2 are also presented. The error is calculated by summing the absolute value of % error in transition probability for each node divided by the total number of nodes in the network. It is seen that if inaccurate state bit signal probabilities are used and the present state bit correlations are not taken into account, the transition probability calculation can be very inaccurate. The network unrolling method produces more accurate results as k increases. From the experimental results, when k is equal to 3 the average error is only 4.6 % which confirms that a small value of k is sufficient to produce accurate results.

Table 4 contains run times for various approaches reported in Table 3. As expected, the computation time increases with k. To obtain the effect of transition probabilities on the power estimation, we also compare the power consumption values estimated by

different methods. Table 5 summarizes these results. The % deviation from the exact power consumption falls to 1% for $k = 3$.

Finally, we applied our method on larger sequential machines from the ISCAS benchmark set. Power consumption estimates for these circuits using different estimation methods are summarized in Table 6.

6 Concluding Remarks

We have presented exact and approximate algorithms for estimating the signal and transition probabilities for FSM. In particular we showed how to calculate state probabilities given the STG for a FSM by iteratively solving the Chapman-Kolmogorov equations. Based on the state probabilities, exact procedure to find the signal and transition probabilities of the internal nodes of a FSM was described. Also we introduced the notion of *k-unrolled* network to correctly estimate the signal probabilities of the state bits of a FSM given its logic implementation. An approximate algorithm for obtaining the transition probabilities based on the notion of *k-unrolled* network was also presented for large sequential machines of which STG cannot be explicitly built.

The signal and transition probabilities of a FSM depends on the state probabilities and the state encoding. It is still an open problem to find a state encoding such that the total transition probabilities of the network is minimized. Further research needs to be done to address this issue.

References

[1] R. Bryant. Graph-based algorithms for Boolean function manipulation. *IEEE Transactions on Computers*, C-35:677–691, Aug. 1986.

[2] A. R. Burch, F. Najm, P. Yang, and D. Hocevar. Pattern independent current estimation for reliability analysis of CMOS circuits. In *Proceedings of the 25th Design Automation Conference*, pages 294–299, June 1988.

[3] A. W. Drake. *Fundamentals of Applied Probability Theory.* McGraw-Hill.

[4] A. A. Ghosh, S. Devadas, K. Keutzer, and J. White. Estimation of average switching activity in combinational and sequential circuits. In *Proceedings of the 29th Design Automation Conference*, pages 253–259, June 1992.

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

[6] S. Malik, A. R. Wang, R. Brayton, and A. Sangiovanni-Vincentelli. Logic Verification using Binary Decision Diagrams in a Logic Synthesis Environment. In *Proceedings of the IEEE International Conference on Computer Aided Design*, pages 6–9, November 1988.

[7] F. Najm. Transition density, a stochastic measure of activity in digital circuits. In *Proceedings of the 28th Design Automation Conference*, pages 644–649, June 1991.

[8] A. Papoulis. *Probability, Random Variables and Stochastic Processes.* McGraw-Hill, 1984.

[9] S. C. Prasad and K. Roy. Circuit activity driven multilevel logic optimization for low power reliable operation. In *Proceedings of the European Design Automation Conference*, pages 368–372, February 1993.

[10] K. Roy and S. Prasad. SYCLOP: Synthesis of CMOS logic for low power application. In *Proceedings of the International Conference on Computer Design*, pages 464–467, October 1992.

[11] A. A. Shen, A. Ghosh, S. Devadas, and K. Keutzer. On average power dissipation and random pattern testability of CMOS combinational logic networks. In *Proceedings of the IEEE International Conference on Computer Aided Design*, November 1992.

[12] V. Tiwari, P. Ashar, and S. Malik. Technology mapping for low power. In *Proceedings of the 30th Design Automation Conference*, pages 74–79, June 1993.

[13] C. Y. Tsui, M. Pedram, and A. Despain. Efficient estimation of dynamic power dissipation with an application. In *Proceedings of the IEEE International Conference on Computer Aided Design*, Nov 1993. to appear.

[14] C. Y. Tsui, M. Pedram, and A. Despain. Technology decomposition and mapping targeting low power dissipation. In *Proceedings of the 30th Design Automation Conference*, pages 68–73, June 1993.

circuits	% of abs. error in state prob.	no. of iterations
bbara	0.00	11
bbsse	0.00	11
bbtas	0.10	30
beecount	0.02	7
cse	0.00	15
dk14	0.00	3
dk15	0.07	3
dk16	0.00	6
dk17	0.03	5
dk27	0.00	9
dk512	0.00	9
donfile	0.00	10
ex1	0.00	14
ex2	0.02	18
ex4	0.02	43
ex5	0.05	6
ex6	0.13	6
keyb	0.00	6
kirkman	0.01	16
lion	0.07	25
mark1	0.06	10
mc	0.13	11
opus	0.00	8
planet	4.33	100
s1	0.00	7
s1488	0.00	26
s27	0.00	3
s386	0.00	12
s510	9.44	100
s820	0.00	19
s832	0.00	19
sand	0.05	92
scf	0.00	20
sse	0.00	11
styr	0.00	26
tbk	0.00	3
train4	0.10	7
average	**0.04**	**19.65**

Table 1: State probabilities.

circuit	Average absolute % error in signal prob.					
	Method	*k-unrolled* with *k-stage feedback*				
	1u_0f	k=1	k=2	k=3	k=4	k=5
bbara	168.20	20.20	8.66	6.72	5.51	4.53
bbsse	105.90	17.12	2.82	0.99	0.13	0.03
bbtas	21.21	16.43	12.59	9.25	7.30	5.20
beecount	10.76	6.99	2.05	0.73	0.23	0.08
cse	75.10	15.09	3.84	0.87	0.20	0.04
dk14	2.36	0.22	0.16	0.03	0.00	0.00
dk15	5.66	3.10	0.24	0.03	0.00	0.00
dk16	4.63	3.18	0.91	1.06	0.42	0.08
dk17	6.78	4.85	0.37	0.26	0.03	0.03
dk27	4.08	4.03	2.79	2.87	0.82	0.55
dk512	8.70	2.70	4.10	2.30	1.10	0.60
donfile	12.71	10.25	3.14	1.60	0.76	0.37
ex1	35.88	29.89	15.57	7.30	4.27	2.30
ex2	10.05	6.30	4.76	1.22	1.18	0.40
ex4	18.81	17.88	10.76	9.21	6.00	7.20
ex5	33.38	5.41	1.63	0.58	0.34	0.09
ex6	13.74	9.00	1.80	0.60	0.13	0.06
keyb	4708.00	637.00	2.40	0.38	0.03	0.01
kirkman	137.00	19.88	5.74	6.18	2.24	1.44
lion	9.11	10.81	6.45	4.20	2.90	2.11
mark1	26.50	11.55	4.48	1.83	1.48	0.54
mc	10.83	15.00	9.00	4.90	1.89	2.00
opus	21.26	15.90	5.84	3.71	0.82	0.14
planet	17.85	15.00	18.50	15.30	31.40	7.60
s1	15.57	16.42	8.28	4.00	1.20	0.45
s1488	811.00	447.70	144.90	87.20	41.70	1.50
s27	8.56	8.50	3.00	1.20	0.50	0.20
s386	44.32	12.45	3.74	0.60	0.16	0.04
s510	19.59	12.41	10.09	9.35	8.39	8.16
s820	157.04	24.90	12.70	6.30	2.90	1.40
s832	157.04	24.90	12.71	6.29	2.90	1.41
sand	3.90	9.00	5.50	4.90	4.80	4.64
scf	132.80	100.73	15.40	8.01	5.73	2.07
sse	105.90	17.12	2.82	0.99	0.13	0.03
styr	219.53	47.38	11.75	4.69	3.61	1.98
tbk	26.07	12.70	2.94	1.38	1.01	0.70
train4	20.83	0.04	0.02	0.00	0.00	0.00
average	**194.32**	**44.12**	**9.8**	**5.87**	**3.84**	**1.57**

Table 2: State bit signal probabilities.

circuit	Exact total trans. prob	Average absolute %error over exact method					
		Method *1u_0f*	*k-unrolled* with *k-stage feedback*				
			k=1	k=2	k=3	k=4	k=5
bbara	8.17	55.85	9.16	7.31	6.11	4.86	3.89
bbsse	20.67	71.23	15.18	5.27	1.13	0.30	0.08
bbtas	5.21	34.11	23.04	14.38	10.95	7.96	5.49
beecount	12.04	29.25	7.39	2.42	0.64	0.24	0.08
cse	16.09	98.55	8.10	2.31	0.53	0.12	0.02
dk14	23.20	19.88	2.88	0.30	0.05	0.01	0.01
dk15	19.61	11.47	5.99	1.04	0.04	0.01	0.01
dk16	38.43	23.18	9.52	3.79	1.45	0.68	0.22
dk17	16.42	19.66	8.28	2.24	1.16	0.23	0.09
dk27	9.54	8.98	3.33	2.26	0.85	0.57	0.27
dk512	15.33	9.12	5.45	3.69	2.00	1.31	0.32
donfile	23.56	18.63	8.83	5.10	2.90	1.42	0.66
ex1	28.01	48.08	35.91	19.16	9.60	4.92	2.58
ex2	29.65	41.74	34.88	11.43	4.96	1.56	1.01
ex4	24.13	23.60	20.75	8.06	7.50	8.94	11.63
ex5	13.61	51.20	9.17	3.97	1.25	0.46	0.22
ex6	21.66	25.64	11.33	2.99	1.59	0.66	0.22
keyb	26.53	50.56	6.11	1.55	0.43	0.15	0.01
kirk	34.32	122.00	12.96	5.80	3.42	2.49	1.71
lion	4.14	5.00	9.59	2.07	1.41	1.02	0.75
mark1	19.40	74.26	23.29	18.78	6.84	3.38	1.79
mc	7.69	12.86	7.60	4.07	2.04	2.34	1.91
opus	16.59	42.79	40.23	24.11	5.11	0.93	0.33
planet	69.98	42.93	40.76	28.10	26.80	28.40	21.35
s1	41.29	34.56	38.10	14.69	6.05	3.01	1.34
s1488	69.58	241.14	119.05	34.79	14.60	6.00	3.40
s27	5.00	19.89	4.41	0.96	0.42	0.18	0.08
s386	25.57	377.13	12.20	5.80	1.33	0.40	0.07
s510	42.82	39.31	14.56	1.48	9.04	8.47	0.60
s820	43.55	201.45	14.22	7.80	3.88	1.95	0.92
s832	45.11	211.84	14.65	7.93	4.01	2.03	0.95
sand	47.13	24.88	16.62	12.02	11.43	11.47	11.43
scf	59.22	102.57	79.31	38.51	18.64	7.13	2.64
sse	20.67	71.23	15.18	5.27	1.13	0.30	0.09
styr	35.47	166.25	43.16	14.83	7.62	3.53	1.69
tbk	79.46	39.86	17.98	3.74	2.58	31.75	1.26
train4	5.00	18.23	0.36	0.16	0.00	0.00	0.00
average		**67.27**	**20.26**	**8.87**	**4.85**	**3.22**	**2.14**

Table 3: Transition probabilities.

circuit	CPU time in seconds						
	Exact Method	Method *1u_0f*	*k-unrolled* with *k-stage feedback*				
			k=1	k=2	k=3	k=4	k=5
bbara	7.08	3.15	2.63	4.18	8.15	14.70	23.10
bbsse	7.39	3.76	3.28	5.16	8.86	13.76	18.99
bbtas	1.32	0.72	0.59	0.68	0.83	1.13	1.45
beecount	2.40	1.43	1.05	1.45	2.08	3.11	4.25
cse	11.86	6.29	5.31	10.71	19.80	31.70	48.00
dk14	6.30	3.62	2.71	4.98	8.13	11.20	11.20
dk15	3.86	2.31	1.63	2.36	3.20	4.43	4.43
dk16	18.20	9.73	8.43	17.68	30.65	48.50	62.58
dk17	3.60	2.00	1.55	2.24	3.28	4.78	6.35
dk27	1.26	0.65	0.61	0.65	0.78	0.86	1.06
dk512	2.80	1.38	1.16	1.44	1.92	2.60	3.41
donfile	26.30	6.83	5.40	11.20	21.13	32.11	44.80
ex1	21.25	10.25	9.05	18.91	30.78	44.78	61.58
ex2	12.41	4.75	4.22	8.00	15.40	23.00	32.50
ex4	5.61	2.50	2.10	2.20	2.58	3.00	3.56
ex5	3.28	1.82	1.45	2.11	3.18	4.76	6.83
ex6	5.46	3.12	2.51	4.30	6.82	10.00	14.33
keyb	34.48	11.34	8.98	20.24	36.04	60.68	87.66
kirk	44.88	7.42	6.26	7.83	9.94	12.26	15.63
lion	0.88	0.48	0.45	0.56	0.68	0.86	1.08
mark1	4.60	2.31	2.10	2.60	3.11	3.69	4.76
mc	1.36	0.75	0.66	0.80	0.95	1.22	1.45
opus	6.48	2.48	2.26	3.90	5.96	8.10	10.90
planet	144.54	18.00	15.51	22.68	31.80	43.52	59.74
s1	46.64	20.11	17.35	50.28	105.70	179.54	252.77
s1488	89.89	32.54	35.23	66.44	109.48	170.60	227.35
s27	0.91	0.45	0.43	0.50	0.61	0.78	1.08
s386	26.30	5.80	5.16	10.00	16.23	0.78	29.85
s510	127.94	9.08	7.95	12.55	18.33	24.64	33.40
s820	33.51	22.15	16.06	42.50	84.62	130.07	191.90
s832	33.70	21.50	17.06	41.83	83.20	133.61	195.10
sand	45.29	23.26	18.94	42.39	84.50	140.46	221.25
scf	294.50	31.30	26.48	40.09	56.40	74.68	100.49
sse	7.08	3.73	3.23	5.16	8.78	13.53	18.24
styr	38.78	23.03	19.19	43.23	84.60	138.31	215.00
tbk	127.14	74.56	60.34	209.60	905.00	1471.00	1625.00
train4	0.95	0.53	0.46	0.56	0.72	0.93	1.18

Table 4: CPU times for transition probability calculation.

circuit	Power consumption(% error)						
	Exact Method	Method *1u_0f*	*k-unrolled* with *k-stage feedback*				
			k=1	k=2	k=3	k=4	k=5
bbara	3633	3939(8.4)	3667(0.9)	3656(0.6)	3649(0.4)	3644(0.3)	3640(0.2)
bbsse	5072	6326(24.7)	5035(0.7)	5037(0.7)	5061(0.2)	5072(0)	5072(0)
bbtas	981	1224(24.7)	1145(16.7)	1080(9.1)	1053(7.3)	1028(4.8)	1005(2.4)
beecount	2204	2422(9.9)	2179(1.1)	2172(1.4)	2197(0.3)	2201(0.1)	2203(0)
cse	4144	5842(41)	4141(0)	4144(0)	4147(0)	4145(0)	4144(0)
dk14	4913	5186(5.6)	4939(0.5)	4916(0)	4913(0)	4913(0)	4913(0)
dk15	3755	3793(1)	3751(0.1)	3748(0.2)	3755(0)	3755(0)	3755(0)
dk16	11947	12487(4.5)	11928(0.1)	11962(0.1)	11972(0.2)	11969(0.2)	11945(0)
dk17	3934	4151(5.5)	4008(1.9)	3945(0.3)	3936(0)	3938(0.1)	3938(0.1)
dk27	2144	2235(4.2)	2142(0.1)	2133(0.5)	2156(0.5)	2140(0.2)	2145(0)
dk512	4485	4374(2.5)	4307(4)	4466(0.4)	4466(0.4)	4499(0.3)	4481(0.1)
donfile	7787	7872(1.1)	7773(0.2)	7806(0.2)	7812(0.3)	7801(0.2)	7792(0)
ex1	6460	8299(28.5)	7918(22.5)	6971(7.3)	6597(2.1)	6515(0.8)	6502(0.6)
ex2	8695	7963(8.4)	7816(10.1)	8377(3.8)	8539(1.8)	8591(1.2)	8603(1)
ex4	4539	4880(7.5)	4851(6.8)	4578(0.8)	4628(1.9)	4451(1.9)	4462(1.7)
ex5	3051	3500(14.7)	3037(0.4)	3012(1.3)	3001(1.6)	3006(1.4)	3004(1.5)
ex6	4518	4866(7.7)	4655(3)	4567(1)	4529(0.2)	4522(0.1)	4522(0.1)
keyb	7712	9176(19)	7638(1)	7704(0.1)	7728(0.2)	7711(0)	7713(0)
kirk	9753	11966(22.7)	9646(1.1)	9651(1)	9694(0.6)	9699(0.5)	9733(0.2)
lion	650	644(0.9)	658(1.2)	655(0.7)	653(0.4)	652(0.3)	652(0.3)
mark1	4191	4562(8.8)	4033(3.7)	4191(0)	4062(3.1)	4102(2.1)	4110(1.9)
mc	1232	1341(8.8)	1253(1.7)	1207(2)	1204(2.2)	1212(1.6)	1223(0.7)
opus	4080	4277(4.8)	4238(3.8)	4049(0.7)	4095(0.3)	4102(0.5)	4085(0.1)
planet	19497	19905(2.1)	19778(1.4)	19483(0)	19489(0)	19888(2)	19760(1.3)
s1	13220	14421(9)	14207(7.4)	13200(0.1)	13157(0.4)	13246(0.2)	13227(0)
s1488	22250	52481(137)	31707(43)	26735(16)	23679(6.4)	23072(3.7)	22572(1.4)
s27	514	549(6.8)	518(0.7)	512(0.4)	513(0.2)	514(0)	514(0)
s386	7058	14749(109)	6763(4.1)	7037(0.3)	7076(0.2)	7058(0)	7058(0)
s510	8709	10816(24)	9261(6.3)	8785(0.8)	8745(0.4)	8635(0.8)	8696(0.1)
s820	18078	31557(75)	18001(0)	17878(1)	18116(0)	18109(0)	18092(0)
s832	18537	32425(75)	18485(0.4)	18349(1)	18584(0)	18573(0)	18553(0)
sand	13259	14161(6.8)	13150(0.8)	13147(0.8)	13187(0.5)	13192(0.5)	13192(0.5)
scf	14309	18323(28)	15434(8)	14351(0.3)	14342(0.2)	14317(0)	14276(0)
sse	5072	6326(25)	5035(0.7)	5037(0.7)	5061(0.2)	5072(0)	5072(0)
styr	13020	16106(24)	13367(2.7)	12953(0.5)	13045(0.2)	13054(0)	13040(0.1)
tbk	74528	87964(18)	73328(1.6)	73337(1.6)	74505(0)	74418(0.1)	74418(0.1)
train4	1015	1073(5.7)	1015(0)	1015(0)	1015(0)	1015(0)	1015(0)
avg % error		**21.86**	**4.31**	**1.56**	**0.92**	**0.68**	**0.42**

Table 5: Power consumption estimation.(The values written in parenthesis denote the absolute value % error compared to the exact method)

circuit	No. of latches	Power consumption			
		Method *1u_0f*	k-unrolled network method		
			k=1	k=2	k=3
s298	14	8976	4494	4318	4164
s344	15	7256	5791	5750	5707
s349	15	7368	5849	5807	5765
s382	21	10099	3980	3992	3976
s400	21	10640	4063	4075	4060
s420	16	4028	4028	4028	4028
s444	21	10174	3890	3897	3881
s526	21	14202	5778	5798	5762
s641	19	9500	8149	8111	8098
s713	19	9848	8566	8539	8522
s838	32	5934	5934	5934	5934
s953	29	16440	12381	11856	12367
s1196	18	28935	27889	27890	27890
s1238	18	31946	30911	30912	30912

Table 6: Power consumption estimation for machines with larger number of flip flops (the estimation accuracy increases as k increases).