4-1-1994

Estimation of Circuit Activity Considering Signal Correlations and simultaneous Switchirlg

Tan Li Chou
Purdue University School of Electrical Engineering

Kaushik Roy
Purdue University School of Electrical Engineering

Sharat Prasad
Texas Instruments Inc.

ESTIMATION OF CIRCUIT ACTIVITY CONSIDERING SIGNAL CORRELATIONS AND SIMULTANEOUS SWITCHING

TAN-LI CHOU
KAUSHIK ROY

TR-EE 94-13
APRIL 1994

SCHOOL OF ELECTRICAL ENGINEERING
PURDUE UNIVERSITY
WEST LAFAYETTE, INDIANA 47907-1285

Estimation of Circuit Activity Considering Signal Correlations and Simultaneous Switching

Tan-Li Chou and **Kaushik** Roy
Electrical Engineering
Purdue University
West Lafayette IN 47907-1285

Sharat Prasad
Texas Instruments Inc.
13510 N. Central Expressway
Dallas, TX 75243

Contact person: Kaushik Roy
Ph: 317-494-2361
Fax: 317-494-6440
e-mail: kaushik@ecn.purdue.edu

Estimation of Circuit Activity Considering Signal Correlations and Simultaneous Switching [1]

Abstract

This paper presents accurate estimation of signal activity at the internal nodes of combinational logic circuits. The methodology is based on stochastic model of logic signals and takes correlations and simultaneous switching of signals at logic gate inputs into consideration. In combinational logic synthesis, in order to minimize spurious transitions due to finite propagation delays, it is crucial to balance all signal paths and to reduce the logic depth [4]. As a result of balancing delays through different paths tlie inputs to logic gates may switch at approximately the same time. We have developed and implemented two algorithms to calculate signal probability and switching activity. The first technique considers signal correlations without considering the effect of simultaneous switching of inputs to logic gates, while the latter considers such switching. Experimental results for the first technique show that the switching activities of the internal notles can be off by more than 100% compared to simulation based techniques. However, the latter technique is within 5% of logic simulation results. Formal proof of correctness of our method has also been presented.

'This research was supported in part by IBM Corporation

1 Introduction

With the recent trend toward portable computing and communications systems there has been an increasing thrust toward considering power dissipation during VLSI design [4, **3,** 14, 12, 11, 13]. In order to design circuits for low power and reliability, accurate estimation of power dissipation is required. In CMOS circuits majority of the power dissipation is due to charging and discharging of load capacitance of logic gates. Such charging and discharging occurs due to signal transitions. The problem of determining when and how often transitions occur at a node in a digital circuit is tlifficult because they depend on the applied input vectors and the sequence in which they are applied. Therefore probabilistic techniques have been resorted to. All reported methods of estimating the probability of a transition at a circuit node involve estimation of *signal probability* which is the probability of a signal being logic ONE. Computing signal probabilities has attracted much research [1, 6, 7, 8]. Most of these methods trade-off accuracy for time. Research directed at estimating signal activity for combinational logic are reported in [2, 5, 9]. However, such methods fail to consider the effect of "near simultaneous" signal switching a.t logic gate inputs. SPICE simulation result for the circuit of Figure 1 shows that the spurious switching will disappear at node V_6 and is negligible at node V_5 if the two primary inputs have a rising and a.falling transition respectively, within 3ns of each other. The spurious pulses try to charge or discharge the capacitances associated with the nodes of a circuit. If such pulses are not wide enough to charge or discharge the capacitances, they disappear. The above example shows that if the inputs to a logic gate switch within a period of **At,** spurious transitions do not occur at the output. At is a function of the inertial delay of the gate and the load capacitances associated with the gate.

The effect of simultaneous switching at the inputs of a logic gate can be best understood by considering the example of Figure 2. If the signals at the inputs of a two input XOR logic gate are switching as shown in the figure, the output switching activity will be zero, even though the signal transition rates a.t the inputs are high. In this paper, we consider such effects in estimating signal activities a.t the internal nodes of a. multilevel circuit. Total ***probability theorem*** [15] is applied to consider the probability that a node of a circuit will switch if one or more inputs are switching. Experimental results show that if simultaneous

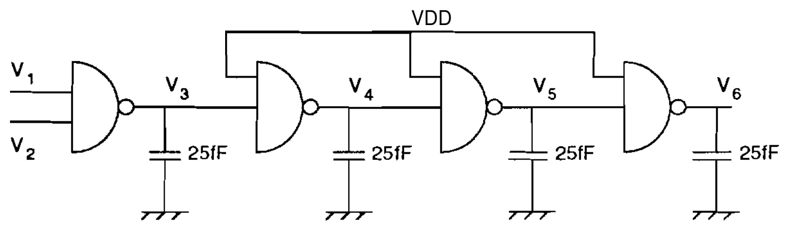

Figure 1: A Circuit for SPICE simulation

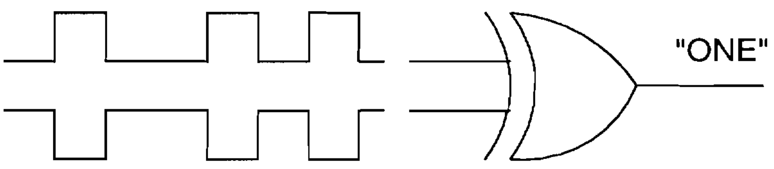

Figure 2: Example of simultaneous switching at logic inputs

switching at the inputs to logic gates is not taken into consideration, tlien the activities of the internal and the output nodes of a circuit can be off by more than 100% compared to logic simulation results. On the other hand, by applying the method proposed in this paper the estimation is within 5% of logic simulation results.

The paper is organized as follows. Section 2 gives the basic definitions and a, brief review of *signal probability*, *activity*, and *power dissipation* in CMOS logic. Section 3 considers efficient calculation of *signal probability*. Section 4 presents the formal proof of our method to accurately calculate signal activity considering signal correlations and simultaneous switching of inputs to logic gates. An approximate formula is also presented in this section. A technique to derive *activities* at the internal nodes of a circuit from its *signal probability* is given in Section 5. Section 6 gives tlie implementation details and experimental results to show that our technique is accurate and applicable for large circuits. The conclusions are given in Section 7.

2 Preliminaries and Definitions

In this section we describe the representation of millti-level circuits, followed by a brief discussion on signal probability, activity, and power dissipation in CMOS.

Multi-level Logic Representation: Multilevel logic can be described by a set $\mathcal{F}$ of completely specified Boolea.1lfunctions. Each Boolean function $f \in \mathcal{F}$, maps one or more inputs and intermediate signals to an output or a new intermediate signal. A circuit is represented as a *Boolean network*. Each *node* has a Boolean *variable* and a Boolean expression associated with it. There is a directed edge to a node g from a node f, if the expression associated with node g contains in it the variable associated with f in either true or complemented form. A circuit is also viewed as a set of gates. Each gate has one or more input pins and (generally) one output pin. Several pins are electrically tied together by a signal. Each signal connects to the output pin of exactly one gate, called the driver gate.

Signal Probability and Activity:

This section briefly describes the model used in [2] for estimation of signal activity. The primary inputs to a combinational circuit are modeled as *mutually independent SSS mean-ergodic 0-1* processes. Under this assumption, tlie probability of the primary input logic signals $x_i(t), i = 1 \dots n$, assuming tlie logic value ONE at any given time t becomes a constant, independent of time, and is called tlie *equilibrium probability* of tlie random signal $x_i(t)$. This is denoted by $P(x_i)$. Tlie *activity* $A(x_i)$ a.t a. primary input x_i of the module is defined as $\lim_{T\to\infty} \frac{n_{x_i}(T)}{T}$ and equals tlie expected value of $\frac{n_{x_i}(T)}{T}$. Tlie variable n_{x_i} is the number of switching of $x_i(t)$ in the time interval $(-T/2, T/2]$. Since digital circuits can be thought of as non-linear but time-invariant systems, the signals at the internal and output nodes of the circuit are also *SSS* and *mean-ergodic*. Further, tlie Boolean functions describing the outputs of a logic module are decoupled from the delays inside the module by assuming the signals to have passed through a special delay module prior to entering tlie module under consideration. Therefore, tlie task of propagating *equilibrium probabilities* through the module is transformed into that of propagating *signal probabilities*. Also tlie *activities* $A(y_j)$ at the nodes y_j of tlie module are given by

$$A(y_j) = \sum_{i=1}^{n} P(\frac{\partial y_j}{\partial x_i})A(x_i) \tag{1}$$

Here $\partial y/\partial x$ is tlie Boolean difference of function y with respect to x and is defined by

$$\frac{\partial y}{\partial x} = y\mid_{x=1} \oplus y\mid_{x=0} = y(x) \oplus y(\bar{x}) \tag{2}$$

Though equation 1 considers signal correlations within a logic module, it does not take simultaneous switching into account. Hence, this method results in errors in estimating ac-

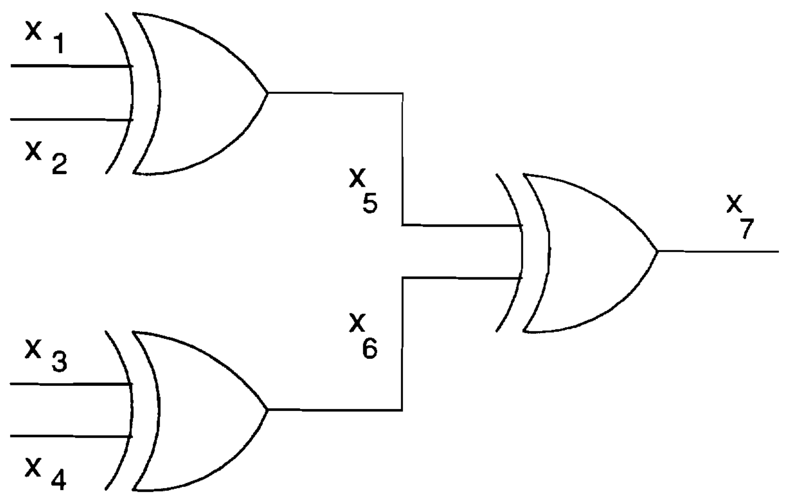

Figure 3: Example of simultaneous switching at logic inputs

tivities of a circuit. Let us consider Figure 3 as an example. Let us assume that the four inputs, $x_1, \cdots, x_4$ are mutually independent with signal probabilities of 0.5 and activities of 1.8*10" transitions per unit time. According to equation 1, the activities of the output $A(x_7)$ should be 7.2 a 10^6 transitions per unit time. However, logic simulation using 10,000 input vectors (conforming to tlie given signal probabilities and activities) show a, signal activity $A(x_7)$ of 4.16 a 10^6 transitions per unit time — a difference of almost 75%! We will observe in Section 4 that this difference is mainly due to "near simultaneous" switching of signals at the inputs to logic gates.

Power Dissipation in CMOS:

Of the three sources of power dissipation in digital CMOS circuits - switching, direct-path short circuit current, and leakage current - the first one is by far the dominant. Ignoring power dissipation due to direct-path short circuit current and leakage current, the average power dissipation in a. CMOS logic is give by

$$\text{POWER.,,.} = \frac{1}{2} V_{dd}^2 \sum_i C_i A(i)$$

where V_{dd} is the supply voltage, $A(i)$ is tlie *activity* at node i, C_i is the capacitive load at that node and is approximately proportional to the fanout at that node. The summation is taken over all nodes of tlie logic circuit. We define *normalized power dissipation measure* Φ

$$\Phi = \sum_i fanout_i a(i) \tag{3}$$

where $fanout_i$ is tlie number of fanouts at node i, and $a(i)$ is the *normalized activity* obtained by dividing activity $A(i)$ by the clock frequency. *Normalized activity* will be more formally defined in Section 4. V_{dd} is the supply voltage and is assumed to be constant. Hence, Φ is proportional to the average power dissipation of a circuit. We will use Φ as a measure to compare tlie results of different algorithms.

In this paper, we assume delay-free modules. The primary inputs are also assumed to be mutually independent.

3 Signal Probability calculation

The proposed methotlology uses signal probability measure to accurately estimate activity. Therefore, it is very important to accurately calculate signal probabilities for further use in estimating *activity.* In tlie following section we present a technique to accurately calculate signal probabilities.

3.1 Definitions and Theorems

A Boolean function f, representing an internal node or an output node of a logic circuit, can always be written in a canonical sum of products of primary inputs. It has been shown in [1] that tlie signal probability *P(f'),* can be expressed as a sum of *primary input signal probability product terms* $\sum_{i=1}^{m} \alpha_i(\prod_{j\in I_i} P(s_j))$, where s_j is x_j or $\bar{x}_j$, and α_i is some integer. I_i is some index set. The sum has m product terms. For convenience, this form will be referred to as the *sum of probability products* in this paper. Assume the inputs $x_i, i = 1,...,$n, of a zero-delay logic motlule are *mutually independent SSS mean-ergodic 0-1 process.* The *signal probability* of a logic signal x_i is expressed as $P(x_i)$. Furthermore, $P(\bar{x}_i) = 1 - P(x_i)$. For primary inputs, which are mutually independent, we also define $\overline{P(x_i)}$ as $P(\bar{x}_i) = 1 - P(x_i)$.

We introduce the following definitions to explain our methodology.

Definition *1 S is an operator, which performs exponent suppression on a sum of probability* *products* $\sum_{i=1}^{m} \alpha_i(\prod_j P(s_j))$. *That is,*

$$\mathcal{S}[\sum_{i=1}^{m} \alpha_j(\prod_j P^{k_j}(s_j))] = \mathcal{S}[\sum_{i=1}^{m} \alpha_j(\prod_j P(s_j))]$$

where $k_j > 0$ *and* $s_j = x_j$ *or* $\bar{x}_j$.

We will use s_i to represent x_i or $\bar{x}_i$. ¿From the definition it follows that

$$\mathcal{S}[P(x_i)\overline{P(x_i)}] = \mathcal{S}[P(x_i)(1 - P(x_i))] = \mathcal{S}[P(x_i) - P^2(x_i)] = P(x_i) - P(x_i) = 0 \qquad (4)$$

where x_i is a primary input of the module concerned. Therefore, whenever **a** probability product term within the *suppression* operator contains both $P(x_i)$ and $\overline{P(x_i)}$, the product term equals zero and can be eliminated. Obviously,

$$\mathcal{S}[\mathcal{S}[\sum_{i}^{m} \alpha_j(\prod_j P(s_j)^{k_j})]] = \mathcal{S}[\sum_{i}^{m} \alpha_j(\prod_j P(s_j))].$$

Also it is easy to see that the following expression holds.

$$\mathcal{S}[b\sum_{i}^{m_1} \alpha_{j_1}(\prod_{j_1} P(s_{j_1})^{k_{j_1}}) + \sum_{i}^{m_2} \alpha_{j_2}(\prod_{j_2} P(s_{j_2})^{k_{j_2}})] =$$

$$b\mathcal{S}[\sum_{i}^{m_1} \alpha_{j_1}(\prod_{j_1} P(s_{j_1})^{k_{j_1}})] + \mathcal{S}[\sum_{i}^{m_2} \alpha_{j_2}(\prod_{j_2} P(s_{j_2})^{k_{j_2}})]$$

where b is a constant. By defining $\mathcal{S}^n[P(f)]$ as performing *Suppression Operation* S on $P(\mathrm{f})$, n times,we have the following lemma,:

Lemma 1 S *is linear and* $\mathcal{S}^n = \mathcal{S}$, *where* n *is a positive integer.*

Definition 2 *Let* f_1 *and* f_2 *be two arbitrary Boolean expressions and* $P(\mathrm{f,}),i = 1,2$ *be expressed in terms of sums of input signal probability products. Let* P_I *be defined as follows,*

$$P_I(f_1 + f_2) = P(f_1) + P(f_2) - P(f_1)P(f_2)$$

$$P_I(f_1 f_2) = P(f_1)P(f_2)$$

Given the above definition, the following theorem is just a generalization of [1] and can be easily proved by induction:

Theorem 1 (Suppression Theorem) *Let* f_1 *and* f_2 *be defined as above and let* $P(f_1)$ *and* $P(f_2)$ *be expressed as sums of probability products. Then,*

$$P(f_1 f_2) = \mathcal{S}[P_I(f_1 f_2)]$$

$$P(f_1 + f_2) = \mathcal{S}[P_I(f_1 + f_2)].$$

3.2 Algorithm for Calculating Signal Probability

There has been much work on bounding or estimating the signal probabilities to balance accuracy and computation time. Since correct signal probabilities is important to both algorithms we propose in tliis paper, we choose the general algorithm proposed in [1] and adopt a data structure similar to [3]. Details of tlie data representation will be given in the implementation section.

Algorithm: Compute Signal Probabilities

Inputs: Circuits, signal probabilities of all the input

Output: Signal probabilities for all nodes of tlie circuit

Step 1: For each input signal and gate output in the circuit, assign a unique variable.

Step 2: Starting at tlie inputs and proceeding to the outputs, write the expression for the output of each gate as a function(using standard expressions for each gate type for probability of its output signal in terms of its mutually independent input signals) of its input expressions.

Step **3:** Suppress all exponents in given expression to obtain the correct probability for that signal.

The following example shows tlie use of suppression theorem to calculate signal probability.

Example *1 Given $y = x_1x_2 + x_1x_3$, where $x_i, i = 1,2,3$, are mutually* ***independent. Then*** *$P(y)$ can be determined **as follows:***

By suppression theorem, $P(y) = \mathcal{S}[P(x_1x_2) + P(x_1x_3) - P(x_1x_2)P(x_1x_3)]$. But we know that $P(x_1x_2) = P(x_1)P(x_2)$ and $P(x_1x_3) = P(x_1)P(x_3)$. Hence,

$$P(y) = P(x_1)P(x_2) + P(x_1)P(x_3) - P(x_1)P(x_2)P(x_3).$$

4 Activities Considering Simultaneous Switching

When more than one primary input, say x_i and x_j, are switching simultaneously, tlie Boolean differences $\frac{\partial y}{\partial x_i}$ and $\frac{\partial y}{\partial x_j}$ are undefined at those time instants. Hence, the proof in [2] that leads to equation 1 is no longer valid for tliis situation. As we have discussed earlier, switchings of different inputs to a node or a module can happen around tlie same time. Let us assume, without any loss of generality, all primary inputs to the module switcli only at the leading edge of the clock. Let a node y of a module be observed at a point in tlie clock cycle which is separated from tlie leading edge of the clock by a long enough interval to allow tlie logic level at this node to reach its stable value. If one selects a clock cycle at random, the probability of having a switching at the leading edge of this clock cycle at node y is $A(y)/f$. Here $A(y)$ is the *activity* at this node and f is tlie clock frequency [10]. We define the *normalized activity* $a(y)$ as $A(y)/\mathrm{f}$. The following definition will be useful in understanding our approach to estimating activity.

Definition 3 (Generalized Boolean Difference) *Let y be a Boolean expression and $x_i, i = 1 \dots n$ be* mutually independent *primary inputs of y. We define,*

$$\frac{\partial y^k \mid_{b_{i_1}, b_{i_2}, \dots, b_{i_k}}}{\partial x_{i_1} \partial x_{i_2} \cdots \partial x_{i_k}} = y \mid_{x_{i_1}=b_{i_1}, x_{i_2}=b_{i_2}, \dots, x_{i_k}=b_{i_k}} \oplus y \mid_{x_{i_1}=\bar{b}_{i_1}, x_{i_2}=\bar{b}_{i_2}, \dots, x_{i_k}=\bar{b}_{i_k}},$$

where k is positive integer, b_{i_j} is logic value ONE or ZERO and $x_{i_j}, j = 1 \dots k$ are distinct mutually independent *primary inputs of y.*

Let us examine the above definition closely. Because tlie operator $\oplus$ is commutative,

$$\frac{\partial^k y \mid_{b_{i_1}, b_{i_2}, \dots, b_{i_k}}}{\partial x_{i_1} \partial x_{i_2} \cdots \partial x_{i_k}} = \frac{\partial^k y \mid_{\bar{b}_{i_1}, \bar{b}_{i_2}, \dots, \bar{b}_{i_k}}}{\partial x_{i_1} \partial x_{i_2} \cdots \partial x_{i_k}} \tag{5}$$

It follows from the definition that if the *generalized Boolean difference* is logic ONE, then the simultaneous transitions at $(x_{i_1}, x_{i_2}, \dots, x_{i_k})$ from $(b_{i_1}, b_{i_2}, \dots, b_{i_k})$ to $(\bar{b}_{i_1}, \bar{b}_{i_2}, \dots, \bar{b}_{i_k})$ or from $(\bar{b}_{i_1}, \bar{b}_{i_2}, \dots, \bar{b}_{i_k})$ to $(b_{i_1}, b_{i_2}, \dots, b_{i_k})$ will cause a transition at y.

Under the assumption that the primary inputs are *mutually independent* and the logic signals can be modeled as *strict-sense stationary(SSS) mean-ergodic 0-1 stochastic processes* with logic modules having zero delays, the following theorem holds.

Theorem 2 (3-inputs) *If y is a Boolean expression and $x_i, i = 1 \dots 3$ are* mutually independent *primary inputs of y, then*

$$\begin{aligned} a(y) &= \sum_{i=1}^{3} P\left(\frac{\partial y}{\partial x_i}\right)\left(a(x_i) \prod_{\substack{j \neq i \\ 1 \leq j \leq 3}} (1 - a(x_j))\right) \\ &+ 1/2\left(\sum_{1 \leq i < j \leq 3} \left(P\left(\frac{\partial^2 y \mid_{00}}{\partial x_i \partial x_j}\right) + P\left(\frac{\partial^2 y \mid_{01}}{\partial x_i \partial x_j}\right)\right)\left(a(x_i) a(x_j) \prod_{l \in \{1,2,3\} - \{i,j\}} (1 - a(x_l))\right)\right) \end{aligned}$$

$$+1/2(P(\frac{\partial^3 y \mid_{000}}{\partial x_1 \partial x_2 \partial x_3}) + P(\frac{\partial^3 y \mid_{001}}{\partial x_1 \partial x_2 \partial x_3}) + P(\frac{\partial^3 y \mid_{010}}{\partial x_1 \partial x_2 \partial x_3})$$
$$+P(\frac{\partial^3 y \mid_{011}}{\partial x_1 \partial x_2 \partial x_3}))(\prod_{l=1}^{3} a(x_l)). \qquad (6)$$

Proof: Because we assume that the module under consideration has zero-delay and the primary inputs switch only at the leading edge of the clock cycle, switching time can only be discrete time points which coincide with some leading edges of tlie clock signal.

At time t, which is some leading edge of tlie clock signal, let B_0 be tlie event tliat none of the inputs are switching. Let $B_i, i = 1,2,3$, be tlie events that only x_i is switching, $B_{i,j}, (i,j) = (1,2),(2,3),(1,3)$, be the events that only x_i and x_j are switching, and finally, $B_{1,2,3}$ be the event tliat all three inputs are switching at the same time t. According to the above tlefinitions, the union of all these events is the *sample space*. All tlie events are also *mutually exclusive* (or *disjoint)*. Therefore, they form a partition of tlie *sample space.* Because x_1, x_2, x_3 are *mutually independent*

$$P(B_0) = (1 - a(x_1))(1 - a(x_2))(1 - a(x_3)).$$

Similarly,

$$P(B_i) = a(x_i) \prod_{\substack{1 \le j \le 3 \\ i \ne j}} (1 - a(x_j)), i = 1,2,3,$$

$$P(B_{i,j}) = a(x_i)a(x_j)(1 - a(x_l)), 1 \le i \le 3, i \ne j,$$

and

$$P(B_{1,2,3}) = a(x_1)a(x_2)a(x_3).$$

Let A be the event that y is switching at time t. Using *total probability theorem* of [15], we derive

$$\begin{aligned} P(A) &= P(A \mid B_0)P(B_0) + \sum_{i=1}^{3} P(A \mid B_i)P(B_i) + \\ & \sum_{1 \le i < j \le 3} P(A \mid B_{i,j})P(B_{i,j}) + P(A \mid B_{1,2,3})P(B_{1,2,3}). \end{aligned} \qquad (7)$$

However we know that if no primary inputs are switching a.t time t, y cannot be switching. If only x_i is switching at time t, the *probability* $P(A \mid B_i)$ that y is switching is $P(\frac{\partial y}{\partial x_i})$. Since a rising transition at any node has to be followed by a falling transition and vice-versa., we have, $P(x_i =\uparrow x_j =\uparrow) = P(x_i =\downarrow x_j =\downarrow) = P(x_i =\downarrow x_j =\uparrow) = P(x_i =\uparrow x_j =\downarrow) = 1/4P(B_{i,j})$, where $\uparrow$ denotes a rising transition and $\downarrow$ a falling one. If three inputs are switching at the same time, they have similar property. Therefore, all the *conditional* probabilities are as follows:

$$P(A \mid B_0) = 0$$

$$P(A \mid B_i) = P(\frac{\partial y}{\partial x_i})$$

$$P(A \mid B_{i,j}) = 1/2(P(\frac{\partial^2 y \mid_{00}}{\partial x_i \partial x_j}) + P(\frac{\partial^2 y \mid_{01}}{\partial x_i \partial x_j}))$$

$$\begin{aligned} P(A \mid B_{1,2,3}) = \quad & 1/2(P(\frac{\partial^3 y \mid_{000}}{\partial x_1 \partial x_2 \partial x_3}) + P(\frac{\partial^3 y \mid_{001}}{\partial x_1 \partial x_2 \partial x_3}) \\ + \quad & P(\frac{\partial^3 y \mid_{010}}{\partial x_1 \partial x_2 \partial x_3}) + P(\frac{\partial^3 y \mid_{011}}{\partial x_1 \partial x_2 \partial x_3})) \end{aligned}$$

After substituting the above expressions into equation 7 and using equation 5 we obtain equation 6 of Theorem 2. □

The generalization of Theorem 2 for n-inputs is given below. The proof is very similar to the proof of Theorem 2 and is omitted for brevity.

Theorem 3 (n-inputs) *If y is a Boolean expression and $x_i, i = 1 \ldots n$ are mutually independent primary inputs of y. Then*

$$\begin{aligned} a(y) \;=\; & \sum_{i=1}^{n} P(\frac{\partial y}{\partial x_i})(a(x_i) \prod_{\substack{j \neq i \\ 1 \leq j \leq n}} (1 - a(x_j))) \\ & +1/2(\sum_{1 \leq i < j \leq n} (P(\frac{\partial^2 y \mid_{00}}{\partial x_i \partial x_j}) + P(\frac{\partial^2 y \mid_{01}}{\partial x_i \partial x_j}))(a(x_i)a(x_j) \prod_{l \in \{1,2,\ldots,n\} - \{i,j\}} (1 - a(x_l)))) \\ & \ldots \\ & +1/2(P(\frac{\partial^3 y \mid_{00\ldots0}}{\partial x_1 \partial x_2 \ldots \partial x_n}) + P(\frac{\partial^3 y \mid_{00\ldots1}}{\partial x_1 \partial x_2 \ldots \partial x_n}) \cdots \\ & +P(\frac{\partial^3 y \mid_{01\ldots1}}{\partial x_1 \partial x_2 \cdots \partial x_n}))(\prod_{l=1}^{n} a(x_l)). \end{aligned}$$

4.1 Approximate Methods Based on Theorem 3

Theorem 3 gives an exact method of determining signal activity. The computation of all the *generalized Boolean difference probabilities* grows exponentially with the number of independent inputs. If one assumes that the *probability* of having more than two inputs to a logic gate switching at the same time is very small, the higher order terms can be ignored. For such a condition the following results can be obtained.

$$\begin{aligned} a(y) &= \sum_{i=1}^{n} P(\frac{\partial y}{\partial x_i})(a(x_i) \prod_{\substack{j \neq i \\ 1 \leq j \leq n}} (1 - a(x_j))) \\ &+1/2(\sum_{1 \leq i < j \leq n} (P(\frac{\partial^2 y \mid_{00}}{\partial x_i \partial x_j}) + P(\frac{\partial^2 y \mid_{01}}{\partial x_i \partial x_j}))(a(x_i)a(x_j) \prod_{l \in \{1,2,\ldots,n\}-\{i,j\}} (1 - a(x_l)))) \end{aligned}$$

5 Derivation of Activities from Signal Probabilities

In order to calculate the activity $A(y_j)$ using equation *(1)* we need to evaluate $P(\frac{\partial y_j}{\partial x_i})$. However, one can calculate this directly from $P(y_j)$ if it is expressed as a sum of probability products $\sum_{i=1}^{m}(\alpha_i \prod_j P_{s_j})$, where $P(s_j)$ is the *probability* of the primary input signal x_i or $\bar{x}_j$. The following theorem enables us to calculate $P(\frac{\partial y_j}{\partial x_i})$ which in turn can be used to determine activity.

Theorem *4 Let y be a Boolean expression and $P(y)$, $P(y(x_i))$, and $P(y(\bar{x}_i)$ be expressed in terms of sum of probability products, where $P(y(x_i))$ and $P(y(\bar{x}_i))$ are the probabilities of the cofactors of y with respect to x_i. Then*

$$P(\frac{\partial y}{\partial x_i}) = \mathcal{S}[P(y(x_i)) + P(y(\bar{x}_i)) - 2P(y(x_i))P(y(\bar{x}_i))].$$

Proof: By Shannon's expansion:

$$y = x_i y(x_i) + \bar{x}_i y(\bar{x}_i).$$

Since $x_i \bar{x}_i = 0$,

$$P(y) = P(x_i)P(y(x_i)) + P(\bar{x}_i)P(y(\bar{x}_i)). \tag{8}$$

On the other hand $(y(x_i)\overline{y(\bar{x}_i)})(\overline{y(x_i)}y(\bar{x}_i)) = 0$. Hence, we have

$$\begin{aligned} P(\frac{\partial y}{\partial x_i}) &= P(y(x_i) \oplus y(\bar{x}_i)) \\ &= P(y(x_i)\overline{y(\bar{x}_i)} + \overline{y(x_i)}y(\bar{x}_i)) \\ &= P(y(x_i))\overline{y(\bar{x}_i)} + P(\overline{y(x_i)}y(\bar{x}_i)). \end{aligned}$$

However, by the suppression theorem we have,

$$\begin{aligned} P(y(x_i)\overline{y(\bar{x}_i)}) &= \mathcal{S}[P(y(x_i))P(\overline{y(\bar{x}_i)})] \\ &= \mathcal{S}[P(y(x_i))(1 - P(y(\bar{x}_i)))] \\ &= \mathcal{S}[P(y(x_i)) - P(y(x_i))P(y(\bar{x}_i))] \end{aligned}$$

and similarly

$$P(\overline{y(x_i)}y(\bar{x}_i)) = \mathcal{S}[P(y(\bar{x}_i)) - P(y(x_i))P(y(\bar{x}_i))].$$

Since S is linear, we can write,

$$\begin{aligned} P(\frac{\partial y}{\partial x_i}) &= \mathcal{S}[P(y(x_i)) - P(y(x_i))P(y(\bar{x}_i))] + \mathcal{S}[P(y(\bar{x}_i)) - P(y(x_i))P(y(\bar{x}_i))] \\ &= \mathcal{S}[P(y(x_i)) + P(y(\bar{x}_i)) - 2P(y(x_i))P(y(\bar{x}_i))]. \ \square \end{aligned}$$

Hence, one can calculate $P(\frac{\partial f}{\partial x_i})$ by first solving for the probabilities of the cofactors of y with respect to x_i. Then the above theorem can be applied to calculate $P(\frac{\partial y}{\partial x_i})$.

Example 2 ***Let $f = a\bar{b}c + \bar{a}b$. Assume a,b,c are independent inputs. The activity $A(f)$ can be calculated as follows using equation 1.***

In order to calculate the activity at f, we first calculate $P(\frac{\partial f}{\partial a})$, $P(\frac{\partial f}{\partial b})$, and $P(\frac{\partial f}{\partial c})$ using Theorem 4. Using the suppression theorem and Shannon's expansion we can write,

$$P(f) = \mathcal{S}[\overline{P_a}P_b + P_a\overline{P_b}P_c]$$

$$= \mathcal{S}[P_a(\overline{P_b}P_c) + \overline{P_a}(P_b)] = \mathcal{S}[P_b(\overline{P_a}) + \overline{P_b}(P_aP_c)] = \mathcal{S}[P_c(P_a\overline{P_b} + \overline{P_a}P_b) + \overline{P_c}(\overline{P_a}P_b)]$$

where $\overline{P_x} = P_{\bar{x}_i} = 1 - P_x, x = a, b, c$. Therefore,

$$P(\frac{\partial f}{\partial a}) = \mathcal{S}[(\overline{P_b}P_c) + (P_b) - 2(\overline{P_b}P_c)(P_b)] = (P_b + \overline{P_b}P_c)$$

$$P(\frac{\partial f}{\partial b}) = \mathcal{S}[(\overline{P_a}) + (P_a P_c) - 2(\overline{P_a})(P_a P_c)] = (\overline{P_a} + P_a P_c)$$

$$P(\frac{\partial f}{\partial b}) = \mathcal{S}[(P_a \overline{P_b} + \overline{P_a} P_b) + (\overline{P_a} P_b) - 2(P_a \overline{P_b} + \overline{P_a} P_b)(\overline{P_a} P_b)] = P_a \overline{P_b}$$

As a result,

$$A(f) = (P_b + \overline{P_b} P_c) A(a) + (\overline{P_a} + P_a P_c) A(b) + (P_a \overline{P_b}) A(c).$$

Note here that in order to calculate $P(\frac{\partial f}{\partial b})$, we decompose $\overline{P_a} P_b$ into $\overline{P_a} P_b P_c$ and $\overline{P_a} P_b \overline{P_c}$ by using the fact that $P_c + \overline{P_c} = 1$.

The following theorem can be used to calculate $\frac{\partial^2 y|_{00}}{\partial x_i \partial x_j}$ and $\frac{\partial^2 y|_{01}}{\partial x_i \partial x_j}$ to determine activities under simultaneous switching of signals at gate inputs. The proof is similar to Theorem 4 and is omitted for brevity.

Theorem 5 Let *y be a Boolean expression. $P(y)$, $P(y(x_i, x_j))$, $P(y(\bar{x}_i, \bar{x}_j))$, ,$P(y(\bar{x}_i, x_j)$ and $P(y(x_i, \bar{x}_j))$, are expressed in terms of sum of probability products, where*

$$y(x_i x_j) = y \mid_{x_i=1, x_j=1}, \quad y(x_i \bar{x}_j) = y \mid_{x_i=0, x_j=1}, \ldots etc.$$

Then

$$P(\frac{\partial^2 y \mid_{00}}{\partial x_i \partial x_j}) = \mathcal{S}[P(y(x_i x_j)) + P(y(\bar{x}_i \bar{x}_j)) - 2P(y(x_i x_j))P(y(\bar{x}_i \bar{x}_j))].$$

$$P(\frac{\partial^2 y \mid_{01}}{\partial x_i \partial x_j}) = \mathcal{S}[P(y(x_i \bar{x}_j)) + P(y(\bar{x}_i x_j)) - 2P(y(x_i \bar{x}_j))P(y(\bar{x}_i x_j))].$$

6 The Algorithms

After having gone through all the theoretical foundations, now we are ready to present the algorithms to estimate signal activity. The first algorithm is based on Theorem 4, the derivation of which does not consider simultaneous switching. The second algorithm that is based on Theorem 5. Both algorithms use the method described in Section **3** to calculate *symbolic* probabilities.

6.1 Algorithm 1

Since this algorithm is based on Theorem 4, which uses equation 1 for activity calculation, its accuracy is the same as Najm's method [2].

Algorithm **1:** Compute Signal Probabilities **And** Activities

Input : Circuit, signal probabilities and activities of all the inputs

Output: Signal probabilities and activities for all nodes of the circuit

Step 1: Use the algorithm in section 2 to calculate the *symbolic* probabilities of a. node.

Step 2: Apply Theorem 4 to calculate activity at this node

Step **3:** evaluate the value of this *symbolic* probability and get rid of this *symbolic* probability.

Step 4: Continue Step 1 to **3** until all the nodes are calculated.

6.2 Algorithm 2

This algorithm is similar to *Algorithm* 1 except step 2. In step 2 we apply Theorem 5 instead of Theorem 4.

Algorithm 2: **Compute Signal** Probabilities And Activities

Input : Circuit, signal probabilities and activities of all the inputs

Output: Signal probabilities and activities for all nodes of the t

Step 1: Use the algorithm in section 2 to calculate the *symbolic* probabilities of a node.

Step 2: Apply Theorem 5 to calculate activity at this node

Step 3: evaluate the value of this *symbolic* probability and get rid of this *symbolic* probability. circui

Step 4: Continue Step 1 to 3 until all the nodes are calculated.

7 Implementation and Results

The algorithms to estimate activities at the internal nodes of a CMOS logic circuit have been implemented in C under the Berkeley SIS environment. We assume that the primary input signal probabilities and activities are available to us through system level simulation of the environment that the circuit will be working in. The data structure used to represent *symbolic probabilities* in *PAS* (Probability and Activity Simulator) is memory efficient and allows us to perform the necessary operations fast. In this representation we have taken advantage of the fact that exponents have been suppressed and therefore, a signal probability expression may contain a variable (assigned to one of tlie inputs) raised to power 1 or may not contain it. So each product term may be regarded as a,set with variables as its elements. Tlie multiplication of two product terms can be achieved by taking tlie union of tlie corresponding sets. The primary inputs of the circuit under consideration are arbitrarily ordered and assigned indices. Let xj, $0 \leq j <$ M, be the primary input signals. Let p_j be tlie signal probability variable assigned to input x_j, i.e. $P(x_j = I) = p_j$. A product term Q_i is represented as a pair (a;, β_i), where both a; and β_i are integers. α_i is called the *coefficient* of the term and may be negative or positive. β_i is regarded as a. bit string. Bit. j of β_i, written β_{ij}, is 1 if and only if tlie corresponding product term contains the variable p_i and is 0 otherwise. When two product terms Q_i and Q_j are multiplied, tlie resulting product term Q_k is given by (α_k, β_k), where $\alpha_k =$ a; $* \alpha_j$, and $\beta_{kl} = \beta_{il} \wedge \beta_{jl}$. It is easy to see that we can define a full order relation on the set of all possible product terms. $Q_i < Q_j$ if $\beta_i < \beta_j$, where Lotl1 β_i and β_j are interpreted as integers. Each probability expression is represented as an ordered list of its product terms, i.e., $P(G) = (Q_1, Q_2, .., Q_{n_G})$. It is obvious that the sum of two expressions $P(G)$ and $P(H)$ can be determined in $O(n_G + n_H)$ time and the product in $O(n_G * n_H)$ time.

We present a number of test cases that show tlie accuracy and efficiency of these two algorithms. In order to asses the accuracy of tlie results, we use a logic simulator with zero-delay model. We generated 10,000 random primary inputs (conforming to the given probabilities and activities) for logic simulation to determine the activities at the internal nodes of a, circuit. The primary input signal probabilities and activities of the circuit were used in PAS to generate *probabilities* and *activities* at internal nodes using Algorithm 1 and *Algorithm* 2. Table I shows tlie detailed results of applying the algorithms to an *MCNC* benchmark example (*parity*). All inputs were assigned a. signal probability of 0.5. All primary inputs were assigned the same activity (though the inputs were all different) as shown in the table. Φ represents tlie normalized power dissipation measure introduced in Section 2 and is used to compare *Algorithm* 1 and 2 with logic simulation technique (*Sim*). The percentage error represents the deviation from tlie simulation results. The CPU times are also shown in tlie table for a SPARC 4 workstation. It can be observed that the power dissipation measure

Table 1: Detailed result for *MCNC* benchmark example *parity*

Activity	*Sim*		*Algorithm 1*			*Algorithm 2*		
	Φ	CPU (sec)	Φ	CPU (sec)	% Error	Φ	CPU	% Error
0.05	3.15	232	3.89	*0.03*	23.5	3.15	0.03	0.0
0.1	5.35	270	7.63	0.03	42.6	5.35	0.03	0.0
0.2	8.42	407	14.49	0.04	72.1	8.41	0.04	-0.12
0.3	10.59	422	20.69	0.03	95.4	10.6	0.05	0.09

Φ can be off by more than 95% for *Algorithm* 1 which calculates tlie activities based on [2], while the results for *Algorithm* 2 are remarkably close to the simulation results. This accuracy can be attributed to tlie fact that *Algorithm 2* considers simultaneous switching of signals at the input to logic gates.

In general, the problem of calculating exact *signal probability* is $NP - hard$. Without partioning the circuit into smaller modules, tlie odered list representing tlie sum of probability products can become extremely large. Even for a small circuit like *MCNC* benchmark example *parity*, which consists of 15 XOR's and has 16 inputs (a total of 31 nodes), the primary output has a 2^{15} ($\approx$ 32,000) probability product terms. This implies that the exact method of calculation may not be feasible for certain types of circuits in terms of memory space and computational time. We will continue to work on partioning to iniprove speed while maintaining accuracy. In the test cases shown in Table 1 and Table 2, we used the "lowest level partioning" [2] in which every logic gate was represented as a separate Boolean module. Results show that with such a partitioning scheme *Algorithm 2* produced accuracy within 5% of simulation results within reasonable CPU time.

Figure 4 shows the accuracy of activity calculation for the *parity* example. The x-axis represents the different nodes of tlie circuit, while the y-axis represents the *normalized activities* associated with each node. Node 0 through 15 are the primary inputs. Nodes 16 through 30 are either intermediate nodes or primary outputs. It can be easily observed that the *Algorithm* 2 closely follows simulation results, wliilr the errors introduced by *Algorithm* 1 can be large.

Table 2 sliows the result on a large number of *ISCAS* and *MCNC* benchmarks. Results

Table 2: Results on ISCAS and MCNC benchmarks

Example	Activity	*Sim*		*Algorithm 1*			*Algorithm 2*		
		Φ	CPU (sec)	Φ	CPU (sec)	% Error	Φ	CPU (sec)	% Error
C432	0.1	37.24	3386	53.82	1.38	44.5	39.56	3.05	6.2
	0.3	82.19	4283	146.25	1.66	77.9	90.63	3.77	10.2
C499	0.1	102.39	4204	250.49	0.46	144.6	106.86	0.78	4.4
	0.3	144.33	5058	679.74	0.56	370.9	148.66	0.94	3
C880	0.1	79.92	8198	86.81	2.04	8.6	73.68	4.22	-8.0
	0.3	136.84	8311	165.7	2.04	21.1	126.6	4.32	-7.5
apex6	0.1	91.43	2745	97.71	36.86	6.9	93.37	275.12	2.12
	0.3	229.31	3726	262.72	37.8	14.5	237.28	306.14	3.5
apex7	0.1	26.29	752	28.91	5.32	7.23	27.87	25.73	3.4
	0.3	69.32	876	78.96	5.39	13.4	73.44	25.8	5.9
b9	0.1	24.06	2812	25.13	1.05	4.44	24.21	2.23	0.62
	0.3	62.13	2910	68.33	1.04	9.9	62.82	2.25	1.1
i3	0.1	18.57	395	18.91	0.29	1.7	18.59	0.55	0.05
	0.3	48.92	590	51.33	0.30	4.9	49.11	0.58	0.5
i4	0.1	31.04	581	32.25	50.1	3.9	31.1	202.9	0.2
	0.3	81.07	873	87.73	50.33	8.2	81.65	205.8	0.7
i5	0.1	59.77	1143	63.12	3.2	5.6	59.7	7.6	-0.12
	0.3	153.7	1716	171.62	3.4	11.6	152.15	7.9	-1.0
i6	0.1	98.85	1859	100.18	4.47	1.3	97.11	10.4	-1.8
	0.3	261.67	2797	277.73	4.61	6.1	259.63	10.4	-0.8
i7	0.1	123.36	2320	126.20	21.0	2.3	122.05	61.2	-1.1
	0.3	326.49	3482	349.37	21.2	7.0	326.33	62.5	-0.05
x2	0.1	6.31	76.1	6.60	77.0	8.1	6.38	404.8	1.1
	0.3	16.18	170	17.81	86.4	10.07	16.70	461.3	3.2
x3	0.1	123.66	2689	128.03	13.1	3.53	124.43	35.8	0.6
	0.3	317.07	4004	344.91	12.7	8.8	326.09	34.9	2.8
x4	0.1	64.81	1257	67.23	127.2	3.7	65.57	532.5	1.2
	0.3	169.57	1939	183.84	127.0	8.4	176.30	537.3	4.0

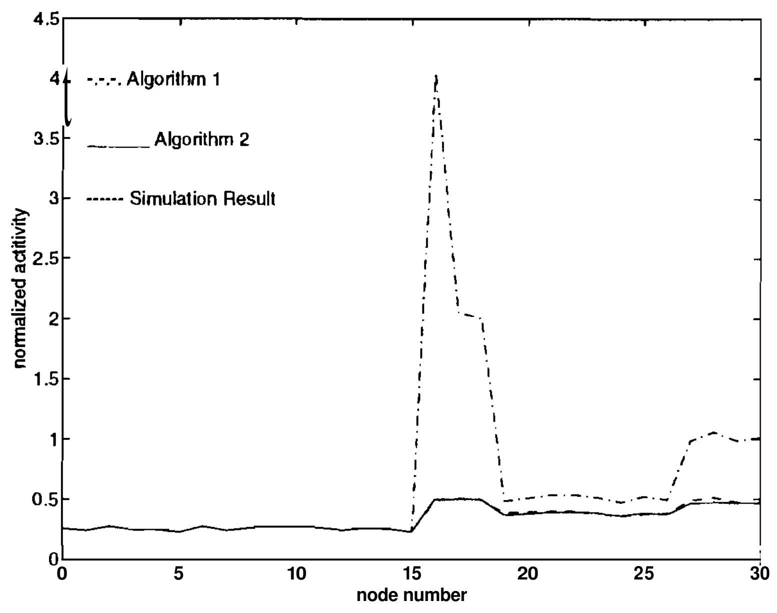

Figure 4: Node Activities for example *parity* (input activities = 0.3)

show that the activity or power dissipation measure Φ determined by *Algorithm 2* is within 5% of logic simulation results.

8 Conclusions

In this paper we have shown that the activities at the internal nodes of a CMOS circuit can be estimated accurately by considering signal correlations and "near simultaneous" switching of inputs to logic gates. Results show that the activity measures are remarkably close to the simulation results. We have aso shown that if such switchings are not considered, activities at the internal nodes can be off by more than 100%. The formal proof our estimation technique has also been presented in the paper.

By partitioning the circuit properly, it is possible to achieve large speed-up in the estimation algorithm. Hence, this technique can be efficiently used in a synthesis environment to estimate power dissipation.

References

[1] K. P. Parker and E. J. McCluskey, "Probabilistic Treatment of General Combinatorial Networks," IEEE *Trans. on Computers.*, vol. C-24, Jun. 1975, pp. 668-670.

[2] F.N. Najm, "Transition Density, A Stochastic Measure of Activity in Digital Circuits," *ACM/IEEE Design Automation Conf.*, 1991, pp. 644-649.

[3] K. Roy and S. Prasad, "Circuit Activity Based Logic Synthesis for Low Power Reliable Operations," IEEE *Trans. on VLSI Systems*, Dec. 1993, pp. 503-513.

[4] A.P. Chandrakashan, S. Sheng, and R.. Brodersen, "Low Power CMOS Digital Design," IEEE *Trans. on* Solid-State *Circuits.*, vol. 27, No. 4, April, 1992, pp. 473-483.

[5] A. Ghosh, S. Devadas, K. Keutzer, and J. White, "Estimation of Average Switching Activity in Combinational and Sequential Circuits," *ACM/IEEE D*esign *Automation Conf.*, 1992, pp. 253-259.

[6] J. Savir, G. Ditlow, and P. Bardell, "Random Pattern Testability," IEEE *Trans.* on *Computers*, Jan. 1984, pp 79-90.

[7] B. Krishnamurthy and I. G. Tollis, "Improved Techniques foi Estimating Signal Probabilities," IEEE *Trans. on Computers.*, vol. C-38, Jul. 1989, pp. 1245-1251.

[8] R.. Kapur, and M. R.. Mercer, "Bounding Signal Probabilities for Testability Measuremerit Using Conditional Syndromes," *SRC* Pub *C92152*, Mar. 1992.

[9] J. Monteiro, S. Devadas, B. Lin, C-Y. Tsui, M. Pedram, and A. Despain, "Exact and **Ap**proximate Methods of Switching Activity Estimation in Sequential Logic," *Intl.* Workshop *on Low Power Design*, Napa Valley, 1994.

[10] S. Prasad and K. Roy, "Circuit Optimization for Minimization of Power Consumption under delay Constraint," *Intl. Workshop on Low Power Design*, Napa Valley, 1994.

[11] C. Tsui, M. Pedram, and A. Despain, "Technology Decomposition and Mapping Targeting Low Power Dissipation," *ACM/IEEE Design Automation Conf.*, 1993, pp. 68-73.

[12] V. Tiwari, P. Ashar, and S. Malik, "Technology Mapping foi Low Power," *ACM/IEEE Design Automation Conf.*, 1993, pp. 74-79.

[13] J. Monteiro, S. Devadas, and A. Ghosh, "Retiming Sequential Circuits for Low Power," IEEE *Intl. Conf. on Computer-Aided-Design*, 1993, pp. 398-402.

[14] B. Lin and H. de Man, "Low-Power Driven Technology Mapping under Timing Constraint," *Intl.* Workshop on Logic *Synthesis,*, 1993 pp. 9a.1-9a.16.

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