Revision of Manuscript 42:

Gate-level Power Estimation Using Tagged Probabilistic Simulation

Chih-Shun Ding* Chi-Ying Tsui† Massoud Pedram‡

Abstract

In this paper, we present a probabilistic simulation technique to estimate the power consumption of a CMOS circuit under a general delay model. This technique is based on the notion of tagged (probability) waveforms, which model the set of all possible events at the output of each circuit node. Tagged waveforms are obtained by partitioning the logic waveform space of a circuit node according to the initial and final values of each logic waveform and compacting all logic waveforms in each partition by a single tagged waveform. From the tagged waveform, one can calculate the switching activity and hence the average power consumption of the circuit node. To improve the efficiency of tagged probabilistic simulation, only tagged waveforms at the circuit inputs are exactly computed. The tagged waveforms of the remaining nodes are computed using a compositional scheme that propagates the tagged waveforms from circuit inputs to circuit outputs. We obtain significant speed up over explicit simulation methods with an average error of only 6%. This also represents a factor of 2-3 improvement in accuracy of power estimates over previous probabilistic simulation approaches.

1 Introduction

With the continuing reduction in the minimum feature size, both transistor count and operating frequency of today's ICs are increasing, which in turn leads to higher power dissipation. Higher power dissipation

*Chih-Shun Ding is with Rockwell Semiconductor Systems, Newport Beach, CA 90620, USA.

†Chi-Ying Tsui is now with Department of Elec. and Elec. Engineering, H.K. Univerity of Science and Technology, Clear Water Bay, Hong Kong.

‡Massoud Pedram is with Department of Electrical Engineering - Systems, University of Southern California, Los Angeles, CA 90089, USA.

in ICs increases the packaging cost and degrades the circuit reliability. As a result, power dissipation has become an important concern in IC design. Another driver for low power is the class of portable electronic devices ranging from laptop computers to personal communicators. Here, the objective is to minimize the power consumption so as to extend the battery life.

To minimize power or to verify that power consumption in a chip meets the power budget allowed in the design specification, one needs to develop accurate and efficient power estimation tools. In addition, today's designers face increasing level of system intergation. To avoid hefty costs associated with the redesign process, the designers need to identify potential design problems as early as possible in the design process. Accurate and efficient power estimation tools are mandatory for the designers to achieve this goal.

In CMOS circuits, power is mostly consumed during charging and discharging of load capacitances. This has led to a simple, yet accurate, power consumption model at the gate level. Several gate level power estimation techniques have been proposed. They offer reasonable accuracy and excellent efficiency as compared with circuit simulation. There are two major types of approaches used in gate-level power estimation techniques: *dynamic* (or *simulative*) [1, 2] and *static* (or *non-simulative*) [3, 4, 5].

Dynamic techniques explicitly simulate the circuit under a "typical" input vector stream. Their main shortcoming is however that they are very slow. Moreover, their results are highly dependent on the simulated sequence. To produce a meaningful power estimate, the required number of simulated vectors is usually high. To address this problem, Monte Carlo simulation technique are proposed in [1, 2]. These technique use an input model based on a Markov process to generate the input stream for simulation. The main difficulty is that it is not clear how the input stream can be efficiently generated when the circuit inputs exhibit complex correlations.

The static techniques [3, 4, 5] rely on statistical information (such as the mean activity of the input signals and their correlations) about the input stream to estimate the internal switching activity of the circuit. In [3] ($CREST$), the concept of *probability waveforms* is proposed to estimate the mean and variance of the current drawn by each circuit node. During the simulation, the logic waveforms are compactly represented by probability waveforms which consist of an initial signal probability and a sequence of events occurring at different time instances. The propagation mechanism for the transition events and their associated transition probabilities is event-driven in nature. In [4] ($DENSIM$) the notion of *transition density*, which is the average number of transitions per second, is proposed. An efficient algorithm based on Boolean difference operation is proposed to propagate the transition densities from circuit inputs throughout the circuit. Although both of these two techniques can be performed efficiently, the accuracy in general is

only moderate, mainly due to the lack of efficient mechanism to account for the signal correlations among circuit nodes. An implicitly enumeration approach based on *symbolic simulation* is proposed in [5]. While the efficiency of this technique has been improved using OBDD (*Ordered Binary Decision Diagram*) [6], the space complexity is its major limitation.

In this paper, we propose an efficient power estimation technique, called *tagged probabilistic simulation* (TPS), based on the notion of tagged (probability) waveforms [7, 8]. It works under the general delay models, therefore does account for the power due to glitches. The tagged probability waveforms are formed by partitioning the waveform space such that logic waveforms produced by all members in a partition are collectively represented by a single tagged waveform. The purpose of this partitioning is to put all logic waveforms with similar properties into the same partition so that the accuracy and efficiency of simulation can be significantly improved. TPS uses the following partitioning strategy: for each node n, all input vectors that produces the same initial and final values in the logic waveforms at node n are put in the same partition and their corresponding logic waveforms are collectively represented by the same tagged waveform. Efficient waveform propagation scheme is developed to propagate the tagged waveforms from circuit inputs to circuit outputs. In particular, the correlation between the tagged waveforms at the inputs of a gate can be effectively accounted for through the use of correlation coefficients. When the inputs to the circuits are specified in terms of statistical measurements (e.g., signal probabilities, transition probabilities), TPS uses an efficient local OBDD-based technique to extract the correlation coefficients. When the inputs to the circuits are specified in terms of vector streams, a bit-parallel simulation approach is used to extract the correlation coefficients. The advantages of this technique are: 1) It is very efficient as only four tagged waveforms are simulated; and 2) Since the logic waveforms represented by each tagged waveform exhibit similar properties, issues such as signal correlations among circuit nodes and glitch generation/propagation can be effectively accounted for. In summary, TPS is intended as a combination of static approaches, which are efficient as only a small number of waveforms are simulated, and dynamic approaches in which the spatiotemporal correlation in input streams are fully captured.

The organization of the paper is as follows. In Section 2, we briefly describe the background and introduce the terminology. The notion of tagged probability waveforms in introduced in Section 3. In Section 4, the waveform computation scheme in TPS is described. The issues of computing correlation coefficients and handling complex gates are examined in Sections 5 and 6. The issue of glitch handling is discussed in Section 7, followed by the experimental results and conclusion in Section 8 and 9, respectively.

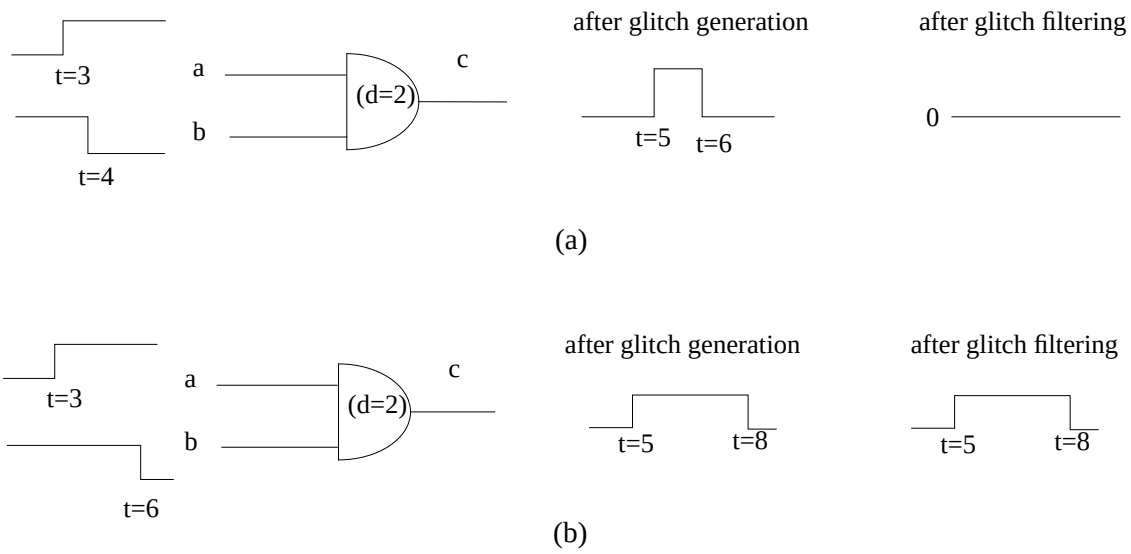


Figure 1: Glitch generation and glitch filtering.

2 Background and Terminology

2.1 Power Consumption Model

In a correctly designed CMOS circuit, the leakage current can be kept 3 to 5 orders of magnitude smaller than the "on current" of the transistors. The dominant sources are the charging/discharging current of the loading capacitance and the short-circuit current. The former depends on the loading capacitance while the latter can be modeled as the charging/discharging current of an "equivalent capacitance". Therefore the power consumption of each node n in the circuit is given by:

$$P_n = \frac{1}{2T_c} V_{dd}^2 C_n sw_n \tag{1}$$

where T_c is the clock period, V_{dd} is the supply voltage, C_n is the sum of load capacitance and the equivalent short-circuit capacitance at node n, and sw_n is the average switching activity at the output of node n (that is, the expected number of signal changes) per clock cycle. In (1), we assume that the signal transition is from 0 to Vdd or vice verse and ignore the impact of signal slew rate on the short-circuit current.

From (1), the problem of estimating the average power can be reduced to one estimating the average switching activity at each node. Therefore the focus of this paper is the estimation of average switching activities.

sw_n in (1) is related to the timing relationship (or delay) of input signals of each gate. Indeed, the output signal of a node may change more than once due to unequal signal arrival times at its inputs. The power consumed by these extra signal changes is generally referred to as the *glitch power*. Here we assume that all signal changes are between Vdd and ground and therefore do not consider partial swing. CMOS gates have inertial delays. Only glitches with adequate strength (i.e., glitch width) can overcome the gate inertia and propagate through the gate to its gate output. These two issues are referred to as the modeling of glitch generation and glitch filtering, respectively, as shown in Figure 1. In Figure 1(a) and (b), glitches are generated because of the skew in the arrival times of input transitions. However, only the glitch in Figure 1(b) remains after the glitch filering process. A delay model that accounts for gate inertia is referred to as an *inertial delay model*. In the paper, we assume an inertial gate delay model. That is, all the glitches with width less than a value specified in the cell library will be suppressed from the waveforms. For the sake of simplicity, we assume that the gate inertial is the same as the gate delay in the discussion of this paper.

A straightforward simulative approach to power estimation would enumerate all pairs of input vectors and compute the average switching activities. This procedure will obviously be exponential in complexity. One way to cope with this complexity is through the concept of probability and tagged (probability) waveforms as described next in this and the following sections.

2.2 Probability Waveforms

Let us examine the operation of the circuit under a sequence of two input vectors V_{-1}, V_0. Let V_{-1} be the vector applied at time $-\infty$ and V_0 be the vector applied at time 0. Clearly, when vector V_0 is applied, all gates have stablized to their values under V_{-1} . During the time interval from 0 to ∞, logic value of each gate varies over time depending on the signal propagation paths from primary inputs to that gate. Therefore, there may be a large number of distinct logic waveforms to be generated at the output of a gate.

Probability waveforms [3] represent the logic waveforms of each gate collectively using probabilistic measurements. A probability waveform is a sequence of *transition edges* (or *events*) over time where each event is annotated with an occurrence probability. In a probability waveform w, two concepts are employed: *signal probability* and *transition probability*, which are defined as follows:

Definition 2.1 Signal probability *$sp_n(t)$ is defined as the probability that a node n assumes logic one at time t.*

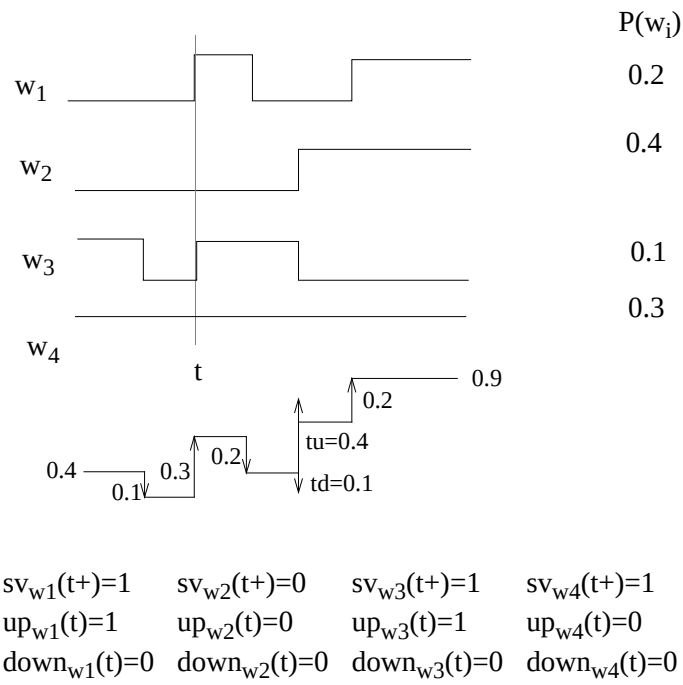


Figure 2: The logic and probability waveforms; the signal and transition probabilities.

Definition 2.2 *The transition probability of an event that changes from 0(1) to 1(0) is defined as the* **upward** *or* **rising transition probability** $tu_n(t)$ (**downward** *or* **falling transition probability** $td_n(t)$).

The probability waveform of a node is a compact representation of the set of all logical waveforms at that node under the input stimuli between two consecutive clock cycles. In this sense, it is an abstraction of the logical waveform space. If we enumerate all the distinct logic waveforms and their occurrence probabilities for a node n, its probability waveform can be easily constructed as explained below. The notations that we use are:

w_i : a logic waveform of node n,
W_n : the set of distinct w_i's of n,
$P(w_i)$: the occurrence probability of w_i,
$sv_{w_i}(t)$: the signal value of w_i at time t,
$up_{w_i}(t)$: a binary value evaluating to 1 if w_i has a $0 \rightarrow 1$ transition at time t, 0 otherwise,
$down_{w_i}(t)$: a binary value evaluating to 1 if w_i has a $1 \rightarrow 0$ transition at time t, 0 otherwise,
$sp_n(t)$: the signal probability of n at time t,

$tu_n(t)$: the $0 \rightarrow 1$ transition probability of n at time t,

$td_n(t)$: the $1 \rightarrow 0$ transition probability of n at time t.

The signal and transition probabilities are given by:

$$sp_n(t) = \sum_{w_i \in W_n} P(w_i) sv_{w_i}(t), \tag{2}$$

$$tu_n(t) = \sum_{w_i \in W_n} P(w_i) up_{w_i}(t), \tag{3}$$

$$td_n(t) = \sum_{w_i \in W_n} P(w_i) down_{w_i}(t). \tag{4}$$

Alternatively, a probability waveform w can be represented by an initial signal probability followed by a sequence of transition probabilities in temporal order, that is:

$$w = \{sp_n(0), tu_n(t_0), td_n(t_0), tu_n(t_1), td_n(t_1), \ldots, tu_n(t_m), td_n(t_m)\},$$

and

$$sp_n(t_+) = sp_n(t_-) + tu_n(t) - td_n(t)$$

Figure 2 shows an example of constructing the probability waveform from a set of logic waveforms.

As enumeration of logic waveforms is prohibitively expensive, probabilistic simulation avoids this enumeration by providing an efficient (yet approximate) waveform propagation mechanism that takes the probability waveforms at the circuit inputs and propagates them through the circuit. The main shortcoming of this simulation technique is that signal correlations caused by reconvergent signal paths in the circuit are difficult to account for. The notion of tagged waveforms is proposed here to alleviate this shortcoming.

3 Tagged Waveforms

A *tagged (probability) waveform* is obtained by partitioning a probability waveform according to the initial and final steady state logic values of the logical waveforms that have contributed to this probability waveform[1]. That is, four tagged waveforms can be defined at each node n: w_n^{00}, w_n^{01}, w_n^{10} and w_n^{11}. All

[1] This definition can be generalized to any partitioning scheme of the probability waveforms.

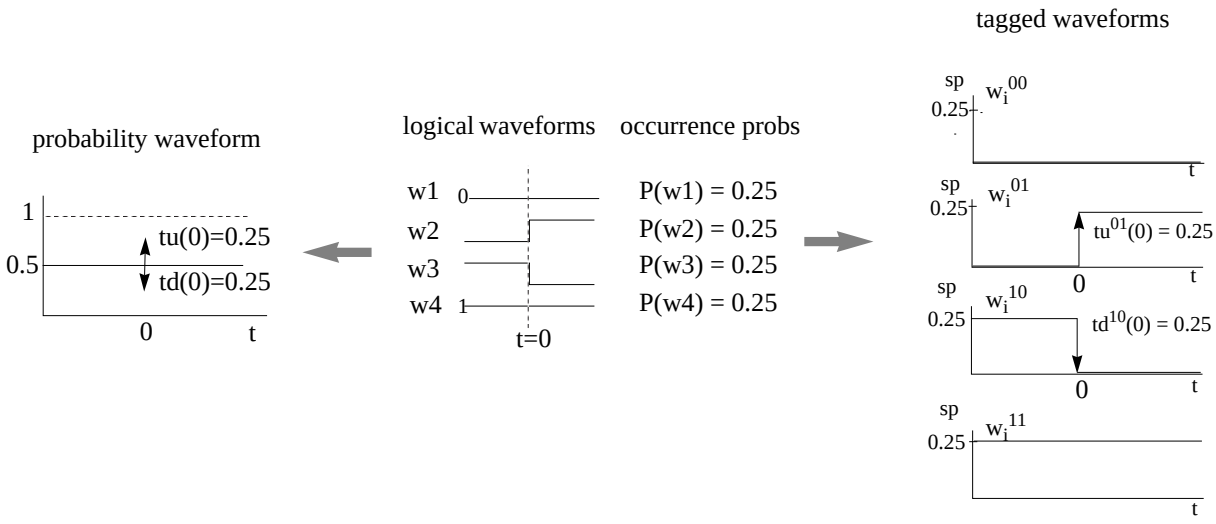


Figure 3: An example depicting the distinction between logic waveforms, probability waveform and tagged waveforms for some circuit input i.

logic waveforms at the output of node n with initial state x and final state y are compactly represented by the tagged waveform w_n^{xy}. $tu_n^{xy}(t)(td_n^{xy}(t))$ represents rising (falling) transition probability at time t in the tagged waveform w_n^{xy}. Similarly, $sp_n^{xy}(t)$ represents the signal probability at time t in tagged waveform w_n^{xy}.

Definition 3.1 Tagged waveform probability, *denoted as* $P(w_n^{xy})$, *is the sum of occurrence probabilities of all logic waveforms represented by* w_n^{xy}.

Note that the initial and final state values are not affected by the delay model. As a result, $P(w_n^{xy})$ can be computed using zero delay logic simulation. When circuit inputs are temporally uncorrelated, the tagged waveform probabilities can be computed as $P(w_n^{00}) = (1 - P_s(n))^2$, $P(w_n^{01}) = P(w_n^{10}) = P_s(n)(1 - P_s(n))$ and $P(w_n^{11}) = (P_s(n))^2$, where $P_s(n)$ is the signal probability of node n under the zero delay model. This signal probability, for example, can be obtained by an OBDD-based approach [6]. In the presence of temporal correlation, $P(w_n^{xy})$ can be computed using techniques proposed in [9, 10].

Figure 3 illustrates the logic waveforms, probability waveform, and tagged (probability) waveforms of a circuit primary input.

Let W denote the maximum number of events in any w_n^{xy} where $x, y \in \{0, 1\}$. From the tagged waveforms of node n, its power consumption can be computed as

$$P_{av}(n) = \frac{1}{2T_c} V_{dd}^2 C_n sw_n = \frac{1}{2T_c} V_{dd}^2 C_n \sum_{x=0}^{1} \sum_{y=0}^{1} \sum_{k=0}^{W} (td_n^{xy}(t_k) + tu_n^{xy}(t_k)) \tag{5}$$

where T_c, V_{dd}, C_n, and sw_n are as defined in (1).

3.1 Extremal Condition

Definition 3.2 *A tagged waveform w_c^{xy} is said to satisfy the* **extremal condition** *at time t if $sp_c^{xy}(t) = 0$ or $sp_c^{xy}(t) = P(w_c^{xy})$.*

When a tagged waveform w^{xy} satisfies extremal condition at both times t_- and t_+, there are four possible cases which will be referred to as cases $E1$ through $E4$ in this paper.

Proposition 3.1 *If a tagged waveform w^{xy} satisfies the extremal condition at times t_- and t_+, then $tu^{xy}(t)$ and $td^{xy}(t)$ are uniquely determined as follows:*

E1) 0-holding *case: if $sp^{xy}(t_-) = sp^{xy}(t_+) = 0$, then $tu^{xy}(t) = td^{xy}(t) = 0$.*

E2) 1-holding *case: if $sp^{xy}(t_-) = sp^{xy}(t_+) = P(w^{xy})$, then $tu^{xy}(t) = td^{xy}(t) = 0$.*

E3) Rising-transition *case: if $sp^{xy}(t_-) = 0$ and $sp^{xy}(t_+) = P(w^{xy})$, then $td^{xy}(t) = 0$ and $tu^{xy}(t) = P(w^{xy})$.*

E4) Falling-transition *case: if $sp^{xy}(t_-) = P(w^{xy})$ and $sp^{xy}(t_+) = 0$, then $td^{xy}(t) = P(w^{xy})$ and $tu^{xy}(t) = 0$.*

Proof Omitted to save space. ■

As the transition probabilities in the above cases are uniquely determined and they are either 0 or the tagged waveform probability, one may ask if the transition probabilities are also exact. Later in Section 4.3 we will show that extremal conditions indeed imply the exactness of tagged waveforms.

In the next section, we describe the waveform propagation scheme in TPS.

4 Waveform Propagation

The waveform propagation mechanism used in TPS is very similar to the one used for probability waveform except that, at each node, TPS deals with four tagged waveforms, instead of a single probability waveform. Consider a two-input AND gate with inputs a and b and gate output c during the waveform propagation procedure. There are four tagged waveforms w_a^{xy} at input a and four tagged waveforms w_b^{wz} at input b, where $x, y, w, z \in \{0, 1\}$. In another words, there are sixteen tagged waveform combinations at the gate inputs. Each of these combination is referred to as the (input) *joint tagged waveform* and denoted by $w_c^{xy,wz}$. They form a disjoint partitioning of the logic waveforms produced at the gate output c. For instance, $w_c^{01,10}$ represents the logic waveforms produced at gate output c when logic waveforms at input a assume initial state 0 and final state 1 and logic waveforms at input b assume initial state 1 and final state 0.

For each joint tagged waveform, one probability waveform is produced at the gate output. The propagation of transition events from gate inputs to gate output is event-driven in nature. That is, for each joint tagged waveform in which an input event occurred at time t, an output event at time $t + d$, where d is the gate delay, is scheduled in the corresponding probability waveform produced at the gate output. For each newly scheduled transition event, we need to calculate its transition probability. After the probability waveforms of all 16 joint tagged waveforms are computed, they are combined into four tagged waveforms so that only four (*output*) *tagged waveforms* are maintained at the output of each gate.

In the following, we consider two cases: 1) tree circuits and 2) circuits with reconvergent paths to illustrate the computation of transition probabilities in TPS. In these cases, we assume no gate inertial and ignore the effect of glitch filtering, which will be discussed in Section 7.

4.1 Tree circuits

For circuits with a tree structure, signals connected to two inputs of the AND gate are uncorrelated if the circuit inputs are uncorrelated. Therefore the transition probabilities can be calculated as:

$$tu_c^{xy,wz}(t+d) = tu_a^{xy}(t)sp_b^{wz}(t_+) + tu_b^{wz}(t)sp_a^{xy}(t_+) - tu_a^{xy}(t)tu_b^{wz}(t) \quad (6)$$

$$td_c^{xy,wz}(t+d) = td_a^{xy}(t)sp_b^{wz}(t_-) + td_b^{wz}(t)sp_a^{xy}(t_-) - td_a^{xy}(t)td_b^{wz}(t) \quad (7)$$

where d is the gate delay, a and b are inputs of an AND gate, and c is the gate output. The reason to have t_+ and t_- on the righ-hand side of above equations will become clear later in the proof.

Table 1: The forcing set table for a two-input AND gate.

OUTPUT TAGS	INPUT TAGS
00	(00,00), (00,01), (00,10),(00,11),(01,00) (01,10),(10,00),(10,01),(11,00)
01	(01,01), (01,11),(11,01)
11	(11,11)
10	(10,10), (10,11), (11,10)

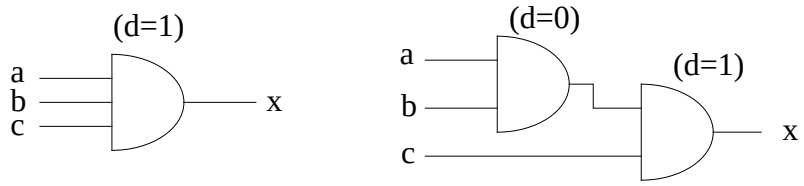
For a transition event scheduled at time $t + d$ at the gate output, at least one of the gate inputs should have a transition event at time t, which is captured by the first two terms in (6) and (7). The last terms of both equations are correction terms for simultaneous input changes. Later in this subsection, we will show that the transition probabilities are as calculated in (6) and (7) are exact when the network contains no complex gates and have a tree structure.

The procedure for combining sixteen joint tagged waveforms can be described as follows. Since each joint tagged waveform already specifies the initial and final states of the logic waveforms at each gate input, we can easily derive the initial and final states of the logic waveforms produced at the gate output. All the joint tagged waveforms that produce the same initial and final state values at the gate output are added together. After this procedure, four tagged waveforms are formed and they are propagated to the fanout gates. Table 1, which is referred to as the *forcing set table*, lists the tags of the joint tagged waveforms (for a two-input AND gate) that should be combined into each output tagged waveform.

An OR gate is equivalent to an AND gate with input and output phase inversion. The above equation can be thus applied to an OR gate with input and output probability waveform inverted. With phase inversion, we mean that w^{00} becomes w^{11}, all rising transition events become falling transition events and all falling transition events becomes rising transition events, etc.

To avoid exponential increase in the number of joint tagged waveforms, gates with more than two inputs are decomposed into subnetworks of two-input gates AND gates and inverters (c.f. Section 6). This subsection will be concluded with the following theorem.

Theorem 4.1 *If the circuit has a tree structure which consists of only simple gates and inverters and the circuit inputs are spatially uncorrelated, the transition probability calculations based on (6), (7) and phase inversion are exact.*



(a) 3-input AND gate (b) After decomposition

Figure 4: An example of decomposing a 3-input AND gate into a subnetwork of 2-input simple gates.

Proof We prove this theorem by induction on levels of nodes. Simple gates with more than two inputs are decomposed into a subnetwork of two-input simple gates. All two-input simple gates are then decomposed into 2-input AND gates and inverters, as shown in Figure 4. The delay of the root gate in each subnetwork is taken to be the same as the delay of original gate while the delays of remaining gates in the subnetwork are assigned to be 0. The new network after decomposition will still have a tree structure if the original network is a tree. In the following we only prove the case for 2-input AND gates, the proof for inverters is trivial as it only involves phase inversion.

We assume that all the circuit input nodes are at level $l = 1$. Since the tagged waveforms are computed exactly on these nodes, the theorem holds for $l = 1$. Assume that the theorem holds for $l = k$. Let c be a gate at level $l = k + 1$, with gate inputs a and b. Without loss of generality, we assume that the transition event of interest is in joint tagged waveform $w_c^{xy,wz}$ and it occurs at the output of c at time $t+d$, where d is the gate delay of c. For each input a, there are four possible combinations of values which a can assume at time t_- and t_+. Let p_a^{00}, p_a^{01}, p_a^{10}, and p_a^{11} represent the occurrence probabilities of these four combinations. From the transition probabilities $tu_a^{xy}(t)$, $td_a^{xy}(t)$, and $sp_a^{xy}(t_-)$, these four occurrence probabilities can be computed as:

$$
\begin{aligned}
p_a^{00}(t) &= 1 - sp_a^{xy}(t_-) - tu_a^{xy}(t), \\
p_a^{01}(t) &= tu_a^{xy}(t), \\
p_a^{10}(t) &= td_a^{xy}(t), \text{ and} \\
p_a^{11}(t) &= sp_a^{xy}(t_-) - td_a^{xy}(t).
\end{aligned}
$$

Similarly for input b. Since the circuit inputs are spatially uncorrelated and the circuit has a tree structure, a and b must be also spatially uncorrelated. $tu_c^{xy}(t+d)$ and $td_c^{xy}(t+d)$ are computed as:

$$tu_c^{xy,wz}(t+d) = p_a^{01}(t)p_b^{11}(t) + p_a^{01}(t)p_b^{01}(t) + p_a^{11}(t)p_b^{01}(t)$$

$$
\begin{aligned}
&= p_a^{01}(t)(p_b^{11}(t) + p_b^{01}(t)) + p_b^{01}(t)(p_a^{11}(t) + p_a^{01}(t)) - p_a^{01}(t)p_b^{01}(t) \\
&= tu_a^{xy}(t)sp_b^{wz}(t_+) + tu_b^{wz}(t)sp_a^{xy}(t_+) - tu_a^{xy}(t)tu_b^{wz}(t) \\
td_c^{xy,wz}(t+d) &= p_a^{10}(t)p_b^{11}(t) + p_a^{10}(t)p_b^{10}(t) + p_a^{11}(t)p_b^{10}(t) \\
&= p_a^{10}(t)(p_b^{11}(t) + p_b^{10}(t)) + p_b^{10}(t)(p_a^{11}(t) + p_a^{10}(t)) - p_a^{10}(t)p_b^{10}(t) \\
&= td_a^{xy}(t)sp_b^{wz}(t_-) + td_b^{wz}(t)sp_a^{xy}(t_-) - td_a^{xy}(t)td_b^{wz}(t)
\end{aligned}
$$

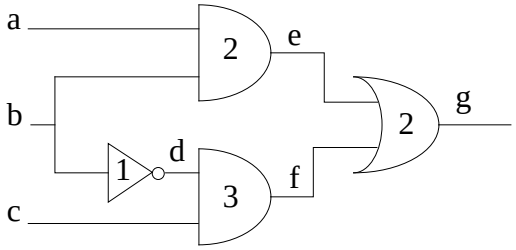
Therefore it proves the case for $l = k+1$, which in turn proves the theorem by mathematical induction. ■

4.2 Circuits with Reconvergent Fanout Paths

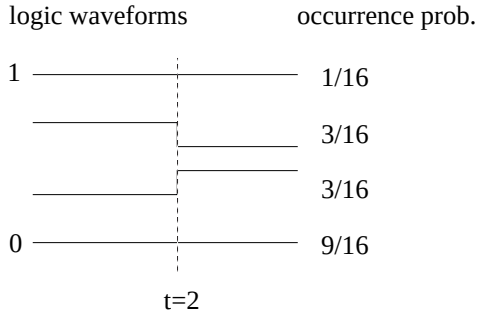
For circuits with reconvergent fanout paths, exact computation of the transition probability of an event is very difficult as explained next. A tagged waveform describes the occurrence probabilities of four different states – staying at 0, making a rising transition, making a falling transition, and staying at 1 – at any time instance. To exactly calculate the output tagged waveform from a joint input tagged waveform in a 2-input AND gate, we need to know the correlation between any two states at the same time instance of the input tagged waveforms from different gate inputs. When gate inputs are uncorrelated, so is the correlations between the two states. Therefore the occurrence probability of each joint state can be calculated by multiplying the occurrence probabilities of the corresponding states in each input tagged waveform, as demonstrated in the previous subsection. However, when the gate inputs are spatially correlated because of the reconvergent fanout paths in the circuit or other reasons (i.e., the circuit inputs are spatially correlated), correlations between states of the two different input tagged at the same time instance may become very complex. To better explain this result, we show an example in Figure 5.

Figure 5(a) shows a simple circuit with reconvergent fanout paths. All logic waveforms at nodes e and f and their occurrence probabilities are shown in Figure 5(b) and (c) assuming 0.5 signal and transition probabilities at nodes a, b, and c. Figure 5(d) shows the joint tagged waveform $w_{ef}^{01,10}$. Figure 5(d) shows joint logic waveforms with non-zero occurrence probability that are represented by $w_{ef}^{01,10}$. To compute the output tagged waveform at g under this joint tagged waveform, we need to know the correlations of the following joint states: 1) at time 2, the joint state in which e makes a rising transition and f stays at 1, and 2) at times 3 and 4, the joint state in which e stays at 1 and f makes a falling transition. Although the joint states under consideration at times 3 and 4 are the same, their correlations are different as shown in Figure 5(d). This example indicates that the correlation of the same joint state may change with time. Unless all joint logic waveforms exhibit the same correlation, it is difficult to accurately estimate the correlations of the joint states. These possibly time-variant correlations will be referred to as *microscopic*

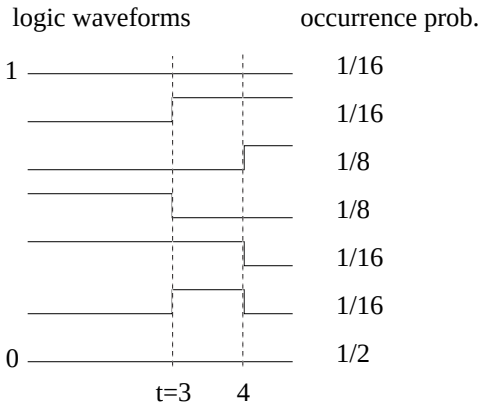
The delay of each gate is written inside the gate



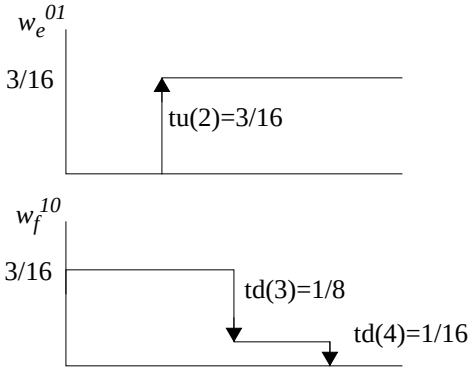
(a) An example circuit.



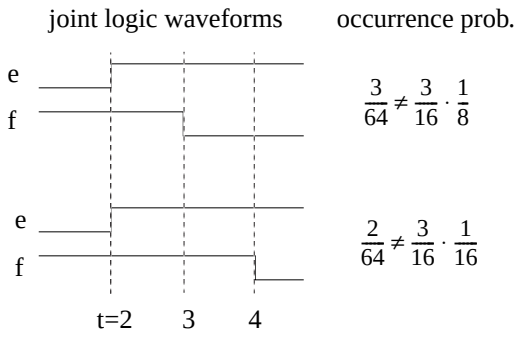
(b) Logic waveforms of node e.



(c) Logic waveforms of node f.

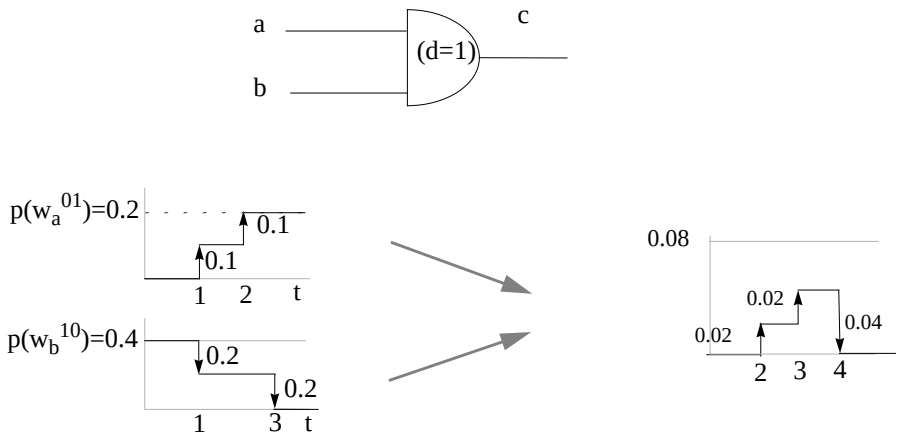


(d) Joint tagged waeform $w_{ef}^{01,10}$.

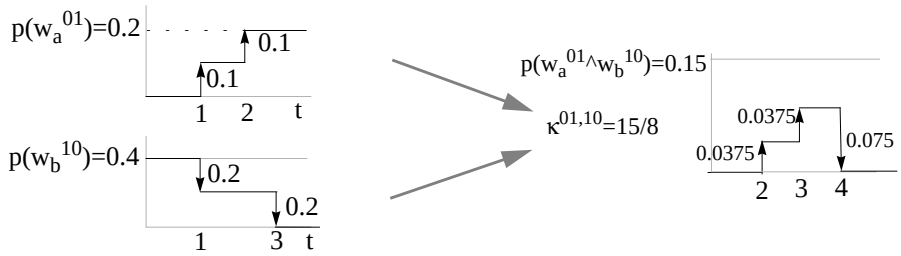


(e) The joint logic waveforms represented by $w_{ef}^{01,10}$.

Figure 5: An example.



(a) The circuit is a tree circuit



(b) There are reconvergent fanout paths in the circuit

Figure 6: An example of waveform computation.

correlations. More precisely, microscopic correlations refer to the correlations between joint events which are present on different logic waveforms. As the time changes, these correlations may change because the joint event under consideration may move from one set of logic waveforms to another. Modeling of microscopic correlations is very difficult for circuits with reconvergent paths. In contrast, *macroscopic correlations* are the correlations between the tags of the input tagged waveforms that constitute the joint tagged waveforms. These macroscopic correlations are encapsulated by *correlation coefficients*, which are defined as, for w_a^{xy} and w_b^{wz}:

$$\kappa^{xy,wz} = \frac{P(w_a^{xy} \wedge w_b^{wz})}{P(w_a^{xy})P(w_b^{wz})} \tag{8}$$

where w, x, y, and $z \in \{0, 1\}$.

Figure 6 shows the difference in waveform computation between a tree circuit and a circuit with reconvergent fanout paths.

TPS uses macroscopic correlations to approximate microscopic correlations. That is, we essentially assume that correlations between any element from set $\{tu_a^{xy}(t), td_a^{xy}(t), sp_a^{xy}(t)\}$ and any element from set $\{tu_b^{wz}(t), td_b^{wz}(t), sp_b^{wz}(t)\}$ is equal to $\kappa^{xy,wz}$. Therefore, the the transition probabilities can be computed as:

$$tu_c^{xy,wz}(t+d) = \kappa^{xy,wz}\{tu_a^{xy}(t)sp_b^{wz}(t_+) + tu_b^{wz}(t)sp_a^{xy}(t_+) - tu_a^{xy}(t)tu_b^{wz}(t)\} \tag{9}$$

$$td_c^{xy,wz}(t+d) = \kappa^{xy,wz}\{td_a^{xy}(t)sp_b^{wz}(t_-) + td_b^{wz}(t)sp_a^{xy}(t_-) - td_a^{xy}(t)td_b^{wz}(t)\} \tag{10}$$

where d is the gate delay, $w, x, y, z \in \{0, 1\}$, a, b are the inputs of an AND gate and c is the output.

The rest of the propagation scheme is exactly the same as in the case of tree circuits. While the above scheme is only an approximation to model the complex microscopic correlations among logic waveforms, it has a few desirable properties. First, the correlation coefficient $\kappa^{xy,wz}$ can be computed under the zero delay model using approaches proposed in [9, 10] which can account for both spatial and temporal correlations in circuit inputs. In particular, $\kappa^{xy,wz}$ is identical to $TC^{xy,wz}$ defined in [9].

Second, if the circuit inputs are free of any temporal correlations, calculation of correlation coefficients can be greatly simplified to no more than that of calculating the signal probabilities under the zero delay model. Under the temporal independence assumption, (8) can be rewritten as:

$$\kappa^{xy,wz} = \frac{P(a = x \wedge b = w)P(a = y \wedge b = z)}{P(a = x)P(b = w)P(a = y)P(b = z)}. \tag{11}$$

$P(a = x \wedge b = w)$ can be calculated directly from signal probabilities of a, b and c using a tabular method [8].

Third, the scheme is consistent in the sense that initial and final signal probabilities of each tagged waveform are exactly calculated if $\kappa^{xy,wz}$'s are exact. Moreover, $td_c^{xy,wz}(t+d)$ can be also obtained from $sp_c^{xy,wz}(t_- + d)$, $sp_c^{xy,wz}(t_+ + d)$, and $tu_c^{xy,wz}(t+d)$ as shown below:

$$sp_c^{xy,wz}(t+d) = \kappa^{xy,wz} sp_a^{xy}(t) sp_b^{wz}(t) \quad (12)$$

$$td_c^{xy,wz}(t+d) = sp_c^{xy,wz}(t_+ + d) - sp_c^{xy,wz}(t_- + d) + tu_c^{xy,wz}(t+d) \quad (13)$$

The results of (13) and (10) are identical.

Last, but not the least, TPS calculates the power consumption due to functional activity exactly, since only the glitch power is approximated. Therefore, the power estimates obtained by TPS are never less than those obtained from the zero delay logic simulation. Note that without using tagged waveforms, there is no assurance that the estimated power will be greater than the zero delay power.

In the next subsection, we will show two results: 1) If a tagged waveform satisfies one of the conditions of $E1$ through $E4$ at time t, the tagged waveform will also be exact at time t; and 2) When the estimated switching activity from TPS on a node equals to its functional (i.e. zero-delay) activity, the estimated activity is exact (even though the tagged waveforms may not).

4.3 Exactness Conditions of TPS

We first show that if an output tagged waveform w^{xy} satisfies extremal condition at time t, signal probability $sp^{xy}(t)$ is exact.

Lemma 4.2 *Let c be a 2-input AND gate with gate inputs a and b.*[2] *In tagged waveform w_c^{xy}, where $x, y \in \{0,1\}$, the signal probability $sp^{xy}(t+d)$ is exactly calculated using (12) when w_c^{xy}'s satisfy the extremal condition at time $t+d$, and κ's are calculated exactly for all transitive fanin gates of c.*

Proof We prove this lemma by induction on levels of nodes l. We assume that all the circuit input nodes are at level $l = 1$. Since extremal conditions are true on these nodes at all time other than time 0 and the tagged waveforms are computed exactly on these nodes, the theorem is true for $l = 1$. Assume that the theorem is true for $l = k$. Let us now consider a node c at level $l = k+1$ with gate inputs a

[2] We assume that the network has been decomposed into 2-input AND gates and inverters. The theorem applies to the network after decomposition.

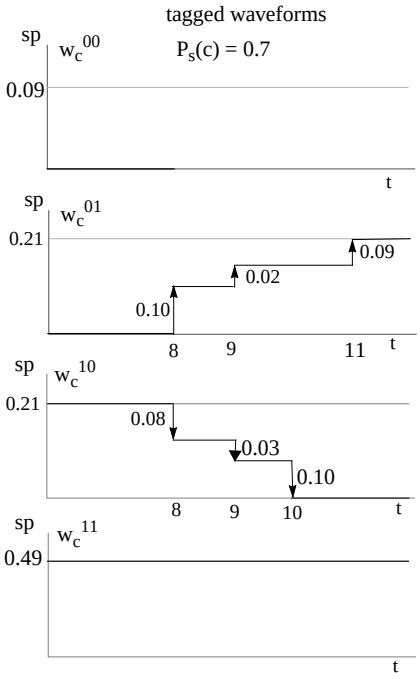


Figure 7: An example of $sw_c = 2 \cdot P(w_c^{01})$.

and b. First, consider the case when $sp_c^{xy}(t+d) = 0$. Note that this implies that each of the joint tagged waveforms $w_c^{pq,rs}$'s that are combined into w_c^{xy} will have

$$sp_c^{pq,rs}(t+d) = 0.$$

From (12), it implies that either $\kappa^{pq,rs} = 0$ or at least one of $sp_a^{pq}(t)$ and $sp_b^{rs}(t)$ is zero. For the former case, $sp_c^{pq,rs}(t+d)$ is exact. For the later case, Without loss of generality, we assume $sp_a^{pq}(t) = 0$. Since node a is at level k or less and $sp_a^{pq}(t) = 0$ satisfies extremal condition, tagged waveform w_a^{pq} is exact at time t and all logic waveforms represented by w_a^{pq} assume 0 at time t. As a result, no logic waveforms at c represented by $w_c^{pq,rs}$ can produce logic 1 at time $t+d$ and $sp_c^{pq,rs}(t+d) = 0$ is exact. The proof for the case when $sp_c^{xy}(t+d) = P(w_c^{xy})$ is similar.

We only prove the case for AND gates as inverters only invert the tagged waveforms and phase inversion do not affect the exactness of the signal probability calculation.

It completes the proof for $l = k+1$. The theorem is true by mathematical induction. ∎

Corollary 4.3 *Let c be a 2-input AND gate with gate inputs a and b. In tagged waveform w_c^{xy}, where $x, y \in \{0,1\}$, the transition probabilities of transition events at time $t+d$ are exactly calculated using (9)*

and (10), when w_c^{xy}'s time $t+d$ satisfy one of the conditions described in Proposition 3.1, and κ's are calculated exactly for all transitive fanin gates of c.

Proof Omitted to save space. ■

An interesting case is when the total switching activity sw_c of a node c estimated from TPS equals to its functional switching activity, e.g. $sw_c = 2 \cdot P(w_c^{01})$. Figure 7 shows an example Note that w_c^{01} and w_c^{10} may not be exact. However, we only need to show there is no 1 to 0 (0 to 1) transition at any time in the logic waveforms represented by w_c^{01} (w_c^{10}), as in the following Theorem, to prove sw_c is exact.

Corollary 4.4 *Let c be a 2-input AND gate with gate inputs a and b. If $sw_c = 2 \cdot P(w_c^{01})$, then sw_c is exact provided that κ's are calculated exactly for all transitive fanin gates of c.*

Proof Omitted to save space. ■

5 Computing the Correlation Coefficient

The correlation coefficient $\kappa^{xy,wz}$ in (8) can be computed exactly or approximately as described next.

5.1 Exact Techniques

Probability Calculation using Global OBDDs

Global OBDDs refer to the OBDDs which represent the logic function of a circuit node in terms of the variables associated with circuit inputs. Using the technique proposed in [11], the temporal correlation of circuits can be exactly accounted for. [9] gives an exact technique that accounts for temporal and pairwise spatial correlations among circuit inputs. However, no exact technique is available to account for general spatial correlations among circuit inputs. Moreover, the space complexity of global OBDDs limits these techniques to circuits of small size.

Bit-parallel Simulation

Very often the input stimuli are specified by a vector stream (e.g., from a simulation results obtained at RT level or higher). In this case, bit-parallel simulation technique under the zero delay model [13] can be

employed to calculate the correlation coefficients. On a SUN SS-20 machine, the run time of a bit-parallel algorithm that computes all sixteen correlation coefficients can run as fast as 600K gate-vector/sec. Bit-parallel simulation can directly capture both temporal and spatial correlations in the circuit inputs without the accuracy loss that will be incurred if we use a procedure to extract input statistics followed by an approximation technique that models the complex signal correlations among the circuit inputs. However, the main drawback is that it cannot handle extremely long streams in a reasonable time.

In the following subsection, we describe two improved techniques that offer good accuracy vs. efficiency trade-off.

5.2 Approximate Technique

Local OBDDs refer to OBDDs which represent the logic function of a circuit node in terms of the variables associated with some set of intermediate variables (associated with nodes in the fanin cone of the node in question). When the signal probability is computed from a local OBDD, the OBDD variables are assumed to be spatially uncorrelated. Several researchers [14, 9, 8] have concluded that the error caused by this assumption is negligible as long as the OBDD variable support set is selected far away (in terms of the level distance) from the node for which the local OBDD is built. This is mainly because signal correlations are result of reconvergent paths in the circuit; these correlations are stronger for short reconvergent paths compared to long reconvergent paths. By selecting a variable support set that is sufficiently far from the target node, short reconvergent paths can be easily captured. However, each local OBDD has its own variable support set which reduces the computation caching and sharing of OBDD. Therefore could be a significant penalty if the local OBDD-based approach is not implemented carefully.

In [15], a technique for computing the signal probabilities using local OBDDs is proposed. The idea is to break the circuit nodes into disjoint parts so that the number of input variables (in the support set) to each part is less than or equal to some user-specified value. However, this approach compromises the accuracy on the nodes at the first few levels of each part since these nodes are too close to the support set. As a result, the effect of short reconvergent paths that pass through the input variables in the support set is not properly captured for the nodes at the first few levels. In summary, this partitioning technique is proposed to limit the size of support set and it overlooks the impact of short reconvergent paths on the signal correlations.

In our scheme [8], nodes in the network are first levelized. Then the OBDD variables for each local OBDD (associated with some node n) are selected from the transitive fanins of n that are at least l levels

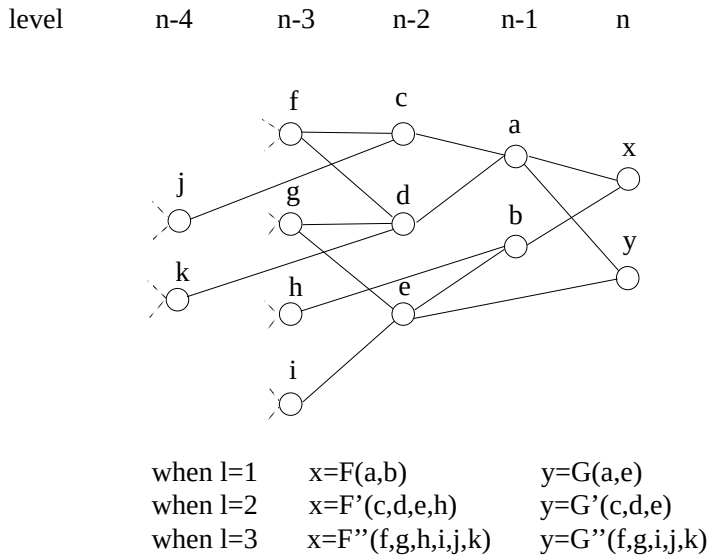


Figure 8: Selection of support sets in the local OBDD approach.

away from n, where l is a user-supplied number, as shown in Figure 8. Nodes in the circuit are processed level by level. The advantages of this approach are: 1) It maximizes the computation caching (sharing of intermediate results) of OBDDs, as the nodes on the same level will share the same set of variables in the proposed scheme and they can be constructed by the same set of OBDD variables and the same variable ordering; and 2) It addresses the signal correlation problem as the function of a node is built in terms of variables which are at least l levels away from it and thus the short reconvergents are always taken into account and the computation accuracy is roughly the same for all the nodes in the circuit.

In practice, we have found that $l = 6$ (on the decomposed network, cf. Section 6) works very well for most of the circuits that we have tested; at the same time, the OBDD size is not an issue as long as $l \leq 10$. Increasing l beyond 6 does not improve the accuracy of total network power estimate by much.

6 Handling Complex Gates

Complex gates are first decomposed into a subnetwork of simple gates, then each simple gate is further decomposed into two-input AND gates and inverters. Caution must be exercised during decomposition as (9) and (10) are only exact for reconvergent fanout free networks. For example, Figure 9 shows two possible decompositions for an And-Or-Invert gate: $o = !(a + bc)$. The factored form decomposition in

Figure 9: Different forms of decomposition for an AOI $o = !(a + bc)$.

Figure 9(a) does not introduce any reconvergent fanout paths (compare this with the SOP decomposition in Figure 9(b), which indeed introduces short reconvergent fanout paths). If the network itself is reconvergent fanout free, this type of decomposition ensures that (9) and (10) are exact.

Not all complex gates can be decomposed into subnetworks without reconvergent fanout paths. On the other hand, only a few gate types in a cell library give rise to short reconvergent fanout paths after decomposition into simple gates. Two-input XOR, XNOR and two-input multiplexor (two data inputs and one select input) are such examples. For the first two gates, we derive efficient closed-form equations similar to (9) and (12). For the third gate we propose a more general approach that can be applied to any gate with greater than two inputs.

XOR gates

Using the analysis approach described in Theorem 4.1 and the correlation coefficient $\kappa^{xy,wz}$ in Section 4.2, the rising transition probability $tu_c^{xy,wz}(t+d)$ at the output of an XOR are calculated as:

$$\begin{aligned} tu_c^{xy,wz}(t+d) &= \kappa^{xy,wz}\{tu_a^{xy}(t)(1-sp_b^{wz}(t_-)) + td_a^{xy}(t)sp_b^{wz}(t_-) + \\ &\quad tu_b^{wz}(t)(1-sp_a^{xy}(t_-)) + td_b^{wz}(t)sp_a^{xy}(t_-) - \\ &\quad 2(tu_a^{xy}(t)tu_b^{wz}(t) + td_a^{xy}(t)td_b^{wz}(t))\} \end{aligned} \tag{14}$$

$td_c^{xy,wz}(t+d)$ is calculated as in (13) with $sp_c^{xy,wz}$ computed as:

$$sp_c^{xy,wz}(t+d) = \kappa^{xy,wz}\{sp_a^{xy}(t)(1-sp_b^{wz}(t)) + sp_b^{wz}(t)(1-sp_a^{xy}(t))\} \tag{15}$$

2-1 Multiplexor gates

The general approach for calculating transition probabilities at the output of a gate at time $t+d$ is based on symbolic simulation using OBDDs as described in [17]. We illustrate this technique on a two-input

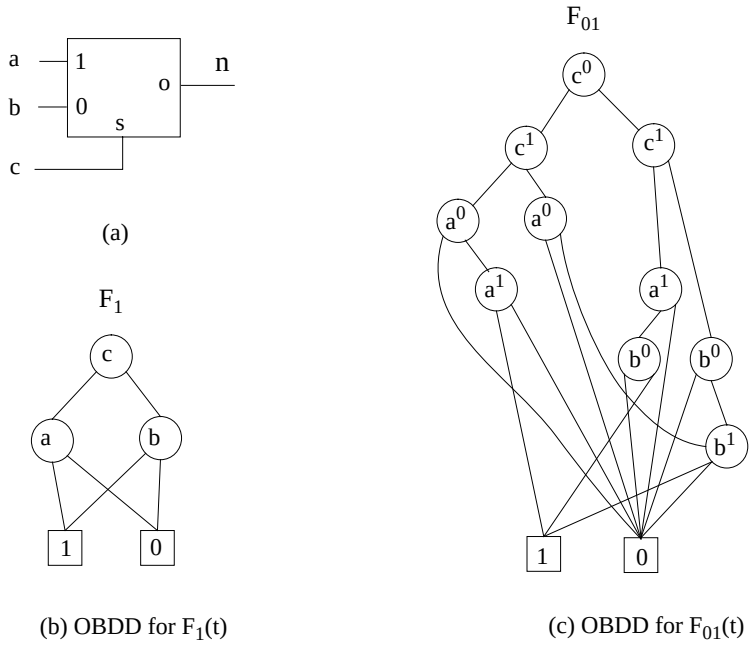


Figure 10: A MUX ($n = ac + b\overline{c}$) and its OBDD representation of $F_1(t)$ and $F_{01}(t)$.

MUX gate n shown in Figure 10(a). Let the inputs of n be a, b, and c. At the output of n, we build two symbolic formulae: $F_{01}(t)$ which describes all input combinations (a, b, and c) that cause n to assume zero and one at time t_- and t_+, respectively; and $F_1(t)$ which describes all input combinations that cause n to assume one at time t. Figure 10(b) and (c) show the OBDDs of $F_1(t)$ and $F_{01}(t)$ for n, respectively (the left edge of a node represents the IF clause). For $F_1(t)$, the variables denote the logic value of each gate input at time $t - d$, where d is the delay of the MUX. For $F_{01}(t)$, the ordering in the OBDD is chosen so that two copies of the same variable (one copy for the values at time $t_- - d$ and another copy for those at time $t_+ - d$) are placed adjacent to one another. $F_{01}(t)$ essentially gives the transition probability $tu_n(t)$. Consider joint tagged waveform w^{opq_irst}, $tu_n^{opq_irst}(t+d)$ is calculated from $P(F_{01}^{opq_irst}(t+d))$ as follows. Let a^0 and a^1 be OBDD variables corresponding to the first and second copies of variable a. The four transition probabilities from time t_- to t_+ in an input tagged waveform w_a^{or} are calculated as:

$$
\begin{aligned}
p_a^{00}(t) &= p(\overline{a^0} \wedge \overline{a^1}) = 1 - sp_a^{or}(t_-) - tu_a^{or}(t), \\
p_a^{01}(t) &= p(\overline{a^0} \wedge a^1) = tu_a^{or}(t), \\
p_a^{10}(t) &= p(a^0 \wedge \overline{a^1}) = td_a^{or}(t), \text{ and} \\
p_a^{11}(t) &= p(a^0 \wedge a^1) = sp_a^{or}(t_-) - td_a^{or}(t).
\end{aligned}
$$

Similar probabilities are also calculated for b and c. There are four paths from the root of $F_{01}(t)$ to the leaf node 1. Therefore, $P(F_{01}^{opq,rst}(t+d))$ is calculated as:

$$\begin{aligned} P(F_{01}^{opq,rst}(t+d)) &= p_c^{11}p_a^{01} + p_c^{10}(p_a^{00}+p_a^{01})(p_b^{01}+p_b^{11}) + \\ & \quad p_c^{01}(p_b^{00}+p_b^{01})(p_a^{01}+p_a^{11}) + p_c^{00}p_b^{01} \\ &= (sp_c^{qt}(t_-) - td_c^{qt}(t))tu_a^{or}(t) + td_c^{qt}(t)(1-sp_a^{or}(t_-))sp_b^{ps}(t_+) + \\ & \quad tu_c^{qt}(t)(1-sp_b^{ps}(t_-))sp_a^{or}(t_+) + (1-sp_c^{qt}(t_-) - tu_c^{qt}(t))tu_b^{ps}(t) \end{aligned}$$

$P(F_{01}^{opq,rst}(t+d))$ will replace the clause within the curly bracket on the right hand side of (9). That is,

$$\begin{aligned} tu_n^{opq,rst}(t+d) &= \kappa^{opq,rst}\{(sp_c^{qt}(t_-) - td_c^{qt}(t))tu_a^{or}(t) + td_c^{qt}(t)(1-sp_a^{or}(t_-))sp_b^{ps}(t_+) + & (16) \\ & \quad tu_c^{qt}(t)(1-sp_b^{ps}(t_-))sp_a^{or}(t_+) + (1-sp_c^{qt}(t_-) - tu_c^{qt}(t))tu_b^{ps}(t)\} & (17) \\ & & (18) \end{aligned}$$

$td_n(t+d)$ is computed from (13) with

$$sp_n^{opq,rst}(t+d) = \kappa^{opq,rst}\{sp_c^{qt}(t)sp_a^{or}(t) + (1-sp_c^{qt}(t))sp_b^{ps}(t)\} \tag{19}$$

To compute the correlation coefficients among more than two inputs, one can use the approximation proposed in [10] in terms of pair-wise correlations. That is, for three-input gates, we have[3]:

$$\kappa^{opq,rst} = \kappa^{or,ps}\kappa^{or,qt}\kappa^{ps,qt} \tag{20}$$

The above scheme can be easily extended to complex gates with more than 3 inputs.

7 Dealing with the Glitches

Glitch filtering refers to the process of "adjusting the transition probabilities" in a joint tagged waveform to account for the fact that short glitches in the logic waveforms do not pass through gates due to the inertial delay of these gates. From our experience as well as that reported in [19], the power dissipation of some datapath circuits (e.g., multipliers) without any glitch-filtering can be overestimated by as much as 200%. Glitch filtering is a complicated task as two transitions in a probability waveform that constitute the same glitch may be correlated. Figure 11 shows two sets of logic waveforms represented by the same

[3] Alternatively, one can use $\kappa^{opq,rst} = \sqrt[3]{\kappa^{or,ps}\kappa^{or,qt}\kappa^{ps,qt}}$ as proposed in [9].

Figure 12: The relation of input and output glitch width implemented in the glitch filtering block.

probability waveform. It is thus impossible to identify the glitches just from transition probabilities in a probability waveform (since the mapping from logic waveforms to probability waveforms is many-to-one).

Another importance issue in glitch filtering is to determine the minimum glitch width that enables it to propagate through a gate. This information can be obtained from the propagation delay and rise/fall times of a gate. In general, we found that using gate delay as the minimum glitch width yields satisfactory results. That is, we implement a glitch propagation mechanism which is based on the relation between input and output glitch width as shown in Figure 12.

The proposed glitch filtering scheme is based on observations from logic simulations. Consider a two-input AND gate with inputs a, b and output c. Without loss of generality, we assume a single $1 \rightarrow 0$ or $0 \rightarrow 1$ transition at each of a and b. Furthermore, we assume that the arrival time of the transition at input a is earlier than that at input b and the skew between the two transitions is smaller than the gate inertial delay. Out of the four possible transition combinations, only the case where input a has a rising transition and its arrival time is earlier than that of a falling transition at input b requires filtering.

Glitch filtering on joint tagged waveform $w_c^{xy,wz}$ of an AND gate c with delay d can be described as follows. For a rising transition event $tu_a^{xy}(t_1)$, all of the falling transition events $td_b^{wz}(t_2)$ that 1) come from the other gate input and 2) $t_1 < t_2 \leq t_1 + d$, where d is the gate delay, are subject to glitch-filering. By glitch filtering, we mean that $\kappa^{xy,wz} tu_a^{xy}(t_1) td_b^{wz}(t_2)$ is subtracted from both $tu_c^{xy,wz}(t_1 + d)$ and $td_c^{xy,wz}(t_1 + d)$. In essence, we assume that $tu_a^{xy}(t_1)$ and $td_b^{wz}(t_2)$ are also correlated by macroscopic correlations of the joint tagged waveforms. This scheme is applied to each joint tagged waveforms before they are merged into four or six tagged waveforms using the forcing set table. Note that this scheme does not perform any filering on short glitches that come from the same gate input. In practice, we found that it produces reasonably good results.

Compared to other probabilistic techniques [3] which only uses one probability waveform for each node, TPS offers better opportunity for glitch-filtering due to the disjointness property of the joint tagged waveforms. To be more precise, during waveform propagation, the sixteen joint tagged waveforms (in the four-tag scheme) form a disjoint partition of the logic waveform space, only the transitions in the same joint tagged waveform are subject to glitch-filtering.

Table 2: Circuit profiles

circuit	pi	po	gate	level
apex6	135	99	588	18
dalu	75	16	722	23
des	256	245	2666	21
i8	133	81	835	15
i10	257	224	1920	51
pair	173	137	1144	25
t481	16	1	619	18
C432	36	7	206	32
C1355	41	32	409	19
C1908	33	25	407	29
C2670	233	140	619	22
C3540	50	22	949	46
C5315	178	123	1221	30
C6288	32	32	2536	108
C7552	207	108	1699	37

8 Experimental Results

The proposed algorithm has been implemented under the *SIS* environment [21]. TPS can be divided into three phases: network decomposition, correlation coefficient calculation, and waveform propagation. During correlation coefficient calculation phase, one can use either local OBDD or bit-parallel simulation approaches.

In the experiments, we compare the power estimates of TPS to those obtained from gate-level logic simulation. Circuits used in the experiments are taken from ISCAS85 and MCNC91 benchmarks and are first mapped to *lib2.sis* library. The delay of each gate is calculated using the delay calculator in SIS [21]. The glitch filtering scheme in both logic simulator and TPS is such that all glitches with width smaller than the gate delay is filtered out. Table 2 shows the circuit profiles. The experiments are performed on two type of sequences. The first sequence – an input sequence of 40,000 vectors – is generated randomly by assuming 0.5 signal and transition probabilities at circuit inputs. The second sequence of length $4,000$ is a non-random one obtained from industry and will be referred to as the biased sequence in this section.

Table 3: Node-by-node error comparison (random sequences).

ckts	TPS_NC			TPS_BDD(l=6)			TPS_BP(N=40,000)		
	sw≥0.1		sw< 0.1	sw≥0.1		sw< 0.1	sw≥0.1		sw< 0.1
	avg	rms	avg	avg	rms	avg	avg	rms	avg
apex6	9.0%	17.5%	69%	4.8%	9.9%	2.2%	3.0%	6.9%	1.5%
dalu	16.7%	26.9%	52%	4.8%	10.0%	25.6%	3.2%	6.7%	14.1%
des	10.8	17.6%	51%	4.4%	9.2%	12.6%	2.9%	6.1%	8.1%
i8	8.9%	20.6%	182%	6.1%	16.2%	146.0%	1.9%	4.0%	20.9%
i10	15.4%	24.1%	57%	9.4%	18.2%	34.5%	6.3%	13.1%	13.8%
pair	8.5%	18.5%	51%	4.8%	10.3%	8.7%	3.0%	8.0%	5.5%
t481	31.5%	47.3%	88%	4.9%	12.1%	7.9%	2.0%	4.8%	2.5%
C432	11.0%	15.8%	47%	5.9%	11.6%	7.9%	1.4%	3.2%	3.9%
C880	8.6%	15.4%	80%	5.4%	10.2%	7.1%	2.1%	6.1%	5.6%
C1355	8.9%	16.5%	68%	5.0%	10.3%	4.6%	3.6%	7.5%	9.2%
C1908	17.3%	28.6%	125%	12.0%	19.1%	47.2%	7.1%	12.7%	37.8%
C2670	8.3%	14.7%	55%	4.4%	9.5%	7.2%	2.9%	6.2%	1.3%
C3540	27.8%	47.7%	88%	12.7%	29.3%	19.0%	7.8%	16.9%	7.2%
C5315	14.2%	21.3%	44%	9.0%	14.7%	35.4%	5.5%	10.0%	6.1%
C6288	27.2%	34.2%	NA	20.1%	27.9%	NA	12.5%	21.3%	NA
C7552	13.2%	22.2%	39%	9.9%	17.5%	37.8%	7.7%	14.4%	10.7%
avg	14.8%	24.3%	73%	7.7%	14.7%	26.8%	4.5%	9.2%	9.8%

Table 4: The run time and total power estimation error for benchmark circuits (random sequences).

circuit	logic sim.		TPS_NC(l=6)			TPS_BDD(l=6)				TPS_BP(N=40,000)			
	power (mW)	run time	power (mW)	run time	error (%)	power (mW)	run time	error (%)	speed-up	power (mW)	run time	error (%)	speed-up
apex6	10.52	146	11.02	0.1	4.7	10.70	0.8	1.7	182	10.48	7.6	0.4	19
dalu	10.68	178	11.58	0.2	8.4	10.68	1.9	0.1	93	10.86	9.5	1.7	18
des	48.38	861	49.10	0.5	0.6	47.76	6.5	1.3	132	48.54	38.6	0.3	22
i8	15.25	205	16.17	0.2	6.0	15.76	1.6	3.3	128	14.96	10.6	1.9	19
i10	37.75	559	32.66	0.6	13.0	36.07	6.8	4.4	82	37.35	25.2	1.0	22
pair	24.16	385	24.72	0.2	2.3	24.16	3.1	0.1	124	24.08	16.3	0.4	23
t481	4.46	112	5.60	0.1	25.5	4.52	1.2	1.3	93	4.48	6.7	0.4	17
C432	3.90	44	4.01	0.1	2.8	3.87	0.6	0.7	73	3.91	2.3	0.2	19
C880	7.15	105	7.45	0.1	4.2	7.30	0.9	2.1	116	7.16	3.9	0.1	27
C1355	8.94	127	7.80	0.1	12.7	8.22	0.8	8.7	158	8.47	5.3	5.2	24
C1908	9.80	134	7.88	0.2	19.6	8.36	1.4	17.2	95	8.94	5.4	8.7	25
C2670	14.61	214	13.44	0.2	8.0	14.10	1.4	3.5	152	14.67	8.4	0.4	25
C3540	25.41	263	23.54	0.7	7.3	25.38	3.7	0.1	71	25.61	12.6	0.7	21
C5315	37.01	452	31.24	0.7	15.6	33.22	3.4	11.4	132	35.23	18.2	5.0	25
C6288	245.62	865	169.07	5.0	30.9	271.90	22.2	10.4	38	262.23	50.2	6.7	17
C7552	57.52	641	50.36	1.5	12.4	55.02	5.7	4.3	112	57.32	23.3	0.3	27
avg					10.8			4.4	111			2.1	22

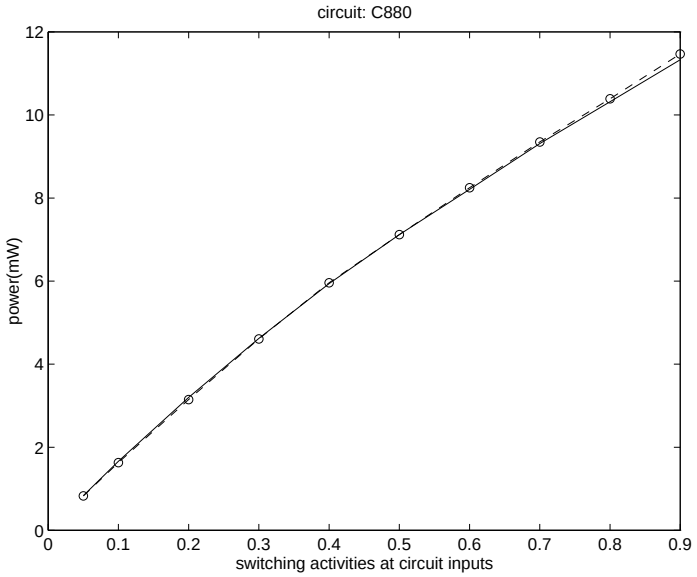


Figure 13: Impact the switching activity at circuit inputs on total power estimates for random sequences

The average bit activity for this sequence is 0.32.

We first assess the node-by-node accuracy of three different approaches on the random sequence:

1) TPS_NC (no correlation): the transition probabilities at each gate are calculated directly from the transition probabilities at gate inputs. Correlations between gate inputs are ignored. This approach is similar to the one used in $CREST$, which is a probabilistic simulation technique.

2) TPS_BDD: TPS using local OBDDs with $l = 6$.

3) TPS_BP: TPS using bit-parallel simulation scheme.

We do not use TPS with global OBDDs due to excessive memory requirement of this approach. The power estimates from the above three approaches are compared against those obtained from logic simulation over the entire vector sequence.

The results on random sequences are summarized in Tables 3 and 4. In Table 3, we report average error on nodes with switching activity (sw) greater than or equal to 0.1 calculated as $Error_{avg} = \sum_i \frac{|sw_i^{ls} - sw_i^{tps}|}{sw_i^{ls}}$ and normalized sum error on nodes with switching activity less than 0.1 calculated as $Error_{avg} = \frac{\sum_i |sw_i^{ls} - sw_i^{tps}|}{\sum_i sw_i^{ls}}$, where sw_i^{ls} and sw_i^{tps} are the activities obtained from logic simulation and TPS, respectively. This is because the average error on low activity nodes is not a good measure. Note

Table 5: Results for biased sequences.

circuit	logic sim.		TPS_BP(N=4000)						
				total power		node-by-node			
						sw≥0.1		sw<0.1	
	power (mW)	run time	run time	power	%error	avg	rms	avg	speedup
apex6	6.78	14.3	2.7	6.85	1.0	3.7%	11.0%	26.0%	5.3
dalu	4.26	15.5	3.8	4.22	1.0	1.2%	3.6%	8.1%	4.0
des	28.05	82.4	15.0	27.99	0.1	3.9%	10.1%	15.3%	5.5
i8	9.29	19.2	4.1	9.22	0.1	1.9%	4.6%	10.3%	4.7
i10	18.09	51.0	11.9	17.73	2.0	7.0%	15.7%	29.4%	4.3
pair	16.08	37.1	6.8	16.10	0.1	4.9%	12.7%	38.7%	5.5
t481	2.01	10.7	2.5	2.01	0.1	1.2%	4.7%	4.9%	4.3
C432	2.11	4.3	0.9	2.26	6.8	8.1%	21.0%	4.2%	4.7
C880	4.11	7.9	1.8	4.18	1.6	4.1%	10.1%	8.7%	4.3
C1355	5.13	10.6	2.0	4.91	4.2	4.2%	9.6%	11.5%	5.3
C1908	6.32	9.8	2.5	6.06	4.1	5.7%	9.8%	40.7%	4.0
C2670	8.31	20.8	3.4	8.58	3.2	5.2%	18.5%	35.7%	6.1
C3540	15.18	24.6	6.5	15.49	2.0	8.8%	17.0%	25.0%	3.8
C5315	20.71	42.3	7.7	20.06	3.2	7.5%	14.5%	40.1%	5.5
C6288	65.53	67.6	25.4	72.10	10.3	25.2%	37.3%	55.1%	2.6
C7552	34.32	58.3	11.5	35.00	1.9	11.3%	28.6%	50.6%	5.0
avg					2.6	6.4%	13.9%	25.0%	4.7

that C6288 has no low activity nodes. The TPS_BP approach simulates all 40,000 vectors. The errors for C6288 circuit are high. This is because very high glitch power is observed in this circuit (5 times the zero-delay power).

Table 4 shows run times and accuracy in estimating the total circuit power consumption. The run times in seconds are reported on a SUN Ultra-Sparc2. The operating frequency of the circuits is 20MHz and Vdd is set to 5 volts. Not surprisingly, the error is much smaller than the node-by-node error due to the averaging effect.

To investigate the impact of changing switching activity at circuit inputs on overall power estimates, we generate random sequences with different input switching activities and with a length of 2,000 vectors. We applied these sequences to circuit C880 and plot the results in Figure 13 where the solid line shows the results of logic simulations and the dashed line shows the result of TPS_BP. This plot shows that TPS is very accurate over a wide range of input switching activities. Similar results are obtained for other benchmark circuits.

For the biased sequence (Table 5), only the results from TPS_BP are reported. All errors of circuit power estimates are smaller than 10% except for C6288. Again, the glitch power in this circuit is still very high (258% of the zero-delay power), even under the biased sequence. The speedup in TPS is about 5, except C6288, since only 4,000 vectors are simulated. The speedup will be higher if longer vectors are simulated as the run time of waveform computation in TPS is relatively insensitive to the vector length. As the vector length increases, the run time of bit-parallel simulation becomes dominant.

To investigate the impact of sequence length of the biased sequence on the estimation accuracy, we compared results of logic simulation and TPS_BP on the first N clock cycles with N varying from 93 to 2000. The results are plotted in Figure 14 where the solid line shows the results of logic simulations. This figure shows that TPS is robust even for short input sequences.

Although the run times of TPS are higher than the probability waveform approach [3] and transition density approach [4] that consider no gate input correlations, TPS has higher accuracy and the ability to account for the spatiotemporal correlations in the input streams.

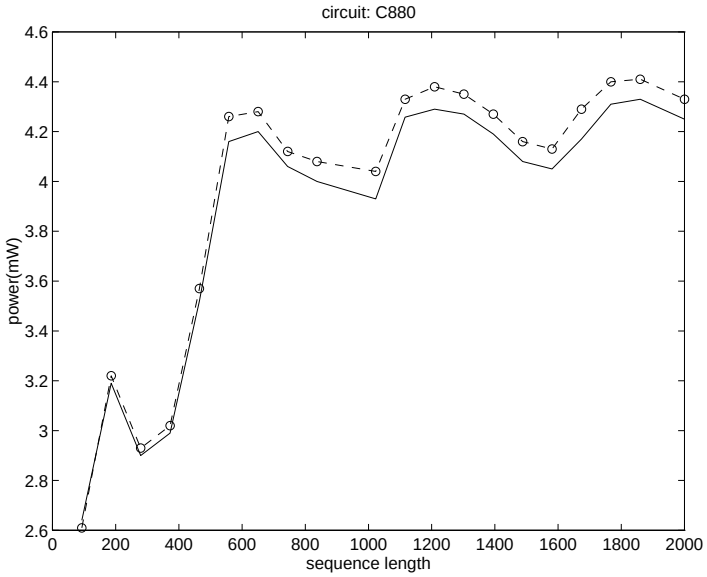


Figure 14: Impact of the simulation length on total power estimates for the biased sequence

9 Conclusion

Efficient power estimation techniques have become very important in today's EDA environments. While this issue can be addressed at different abstraction levels of design, gate-level techniques provide good balance between accuracy and efficiency. In this paper, we presented tagged probabilistic simulation (TPS) as an accurate and efficient gate level power estimation technique. TPS is based on the notion of tagged waveforms which divide the logic waveform space into a small number of disjoint parts and then represents all the logic waveforms in a part by a probability waveform. The advantage of this simulation strategy is that the correlations among circuit internal nodes can be effectively accounted for. Due to the disjointness of the tagged waveforms, an effective filtering scheme was developed and used in TPS.

We described the fundamental properties and the waveform propagation schemes of tagged waveforms. We also presented necessary conditions for the transition probability calculation to be exact. Lastly, the experimental results showed that TPS is more accurate than a probabilistic simulation without considering gate input correlations and a factor of 10 more efficient than explicit gate-level simulation. TPS is also space-efficient and can easily handle circuits of large size.

While in this paper we do not discuss the issues of applying TPS to sequential circuits, TPS can be easily extended to do so. The estimation procedure consists of two tasks: 1) estimate the signal and transition probabilities of state lines, and 2) estimate the power consumption of the combinational kernel and flip-

flops (or registers) based on the probabilities computed in 1). Techniques such as those proposed in [20] can be used to perform task 1.

References

[1] R. Burch, F. N. Najm, P. Yang, and T. Trick. A Monte Carlo approach for power estimation. IEEE *Transactions on VLSI Systems*, 1(1):63–71, March 1993.

[2] M. Xakellis and F. Najm. Statistical estimation of the switching activity in digital circuits. In *Proceedings of the 31st Design Automation Conference*, pages 728–733, 1994.

[3] F. N. Najm, R. Burch, P. Yang, and I. N. Hajj. Probabilistic simulation for reliability analysis of cmos circuits. IEEE *Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 9(4):439–450, April 1990.

[4] F. N. Najm. Transition density: A new measure of activity in digital circuits. IEEE *Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 12(2):310–323, February 1993.

[5] 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.

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

[7] C-Y. Tsui, M. Pedram, and A. M. Despain. Efficient estimation of dynamic power dissipation under a real delay model. In *Proceedings of the* IEEE *International Conference on Computer Aided Design*, pages 224–228, November 1993.

[8] C.-S. Ding and M. Pedram. Tagged probabilistic simulation provides accurate and efficient power estimates at the gate level. In *Proc. of the Sym. on Low Power Electronics*, pages 42–43, September 1995.

[9] D. Marculescu R. Marculescu and M. Pedram. Switching activity analysis considering spatiotemporal correlation. In *Proceedings of the* IEEE *International Conference on Computer Aided Design*, pages 294–299, June 1994.

[10] S. Ercolani, M. Favalli, M. Damiani, P. Olivo, and B. Ricco. Estimate of signal probability in combinational logic networks. In *First European Test Conf.*, pages 132–138, 1989.

[11] P. Schneider and U. Schlichtmann. Decomposition of boolean functions for low power based on a new power estimation technique. In *Proceedings of 1994 International Workshop on Low Power Design*, pages 123–128, April 1994.

[12] R. Marculescu, D. Marculescu, and M. Pedram. Logic level power estimation considering spatiotemporal correlations. In *Proceedings of the* IEEE *International Conference on Computer Aided Design*, pages 294–299, November 1994.

[13] P. H. Schneider. PAPSAS: A fast switching activity simulator. In *Proc. of PATMOS*, pages 350–360, 1995.

[14] A. Majumdar and S. Sastry. Probabilistic characterization of controllability in general homogeneous circuits. *Computer-Aided Design*, 25(2):76–93, February 1993.

[15] B. Kapoor. Improving the accuracy of circuit activity measurement. In *Proceedings of the 31st Design Automation Conference*, pages 734–739, 1994.

[16] C.-S. Ding, C.-T. Hsieh, Q. Wu, and M. Pedram. Stratified sampling for power estimation. In *Proceedings of the* IEEE *International Conference on CAD-96*, November 1996. To appear.

[17] J. Monteiro, S. Devadas, and A. Ghosh. Estimation of switching activity in sequential logic circuits with applications to synthesis for low power. In *Proceedings of the 31st Design Automation Conference*, page , June 1994.

[18] C.-S. Ding and M. Pedram. Efficient symbolic simulation under the extended bounded delay model. In *Proceedings of Tau'93: Int'l Workshop on Timing Issues in the Specification and Synthesis of Digital Systems*, September 1993.

[19] F. N. Najm. Low-pass filter for computing the transition density in digital circuits. IEEE *Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 13(9):1123–1131, September 1994.

[20] C-Y. Tsui, J. Monteiro, M. Pedram, S. Devadas, A. M. Despain, and B. Lin. Power estimation in sequential logic circuits. IEEE *Transactions on Computer-Aided Design of Integrated Circuits and Systems*, September 1995.

[21] E. M. Sentovich, K .J. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P. R. Stephan, R. K. Brayton, and A. L. Sangiovanni-Vincentelli. SIS: A system for sequential circuit synthesis. Technical Report UCB/ERL M92/41, Electronics Research Lab, University of California at Berkeley, 1992.