# Power Estimation in Sequential Circuits†

Farid N. Najm, Shashank Goel, and Ibrahim N. Hajj

ECE Dept. and Coordinated Science Lab.
University of Illinois at Urbana-Champaign
Urbana, IL 61801

**Abstract – A new method for power estimation in sequential circuits is presented that is based on a statistical estimation technique. By applying randomly generated input sequences to the circuit, statistics on the latch outputs are collected, by simulation, that allow efficient power estimation for the whole design. An important advantage of this approach is that the desired accuracy can be specified up-front by the user; the algorithm iterates until the specified accuracy is achieved. This has been implemented and tested on a number of sequential circuits and found to be much faster than existing techniques. We can complete the analysis of a circuit with 1,452 flip-flops and 19,253 gates in about 4.6 hours (the largest test case reported previously has 223 flip-flops).**

## I. Introduction

The dramatic decrease in feature size and the corresponding increase in the number of devices on a chip, combined with the growing demand for portable communication and computing systems, have made power consumption one of the major concerns in VLSI circuits and systems design [1]. Indeed, excessive power dissipation in integrated circuits not only discourages the use of the design in a portable environment, but also causes overheating, which can lead to soft errors or permanent damage. Hence there is a need to accurately estimate the power dissipation of an IC during the design phase.

The main conceptual difficulty in power estimation is that the power depends on the input signals driving the circuit. Simply put, a more active circuit will consume more power. Thus, one straightforward method of power estimation is to simulate the design over all possible inputs, compute the power dissipated under each input, and average the results. However, such an approach is prohibitively expensive. Thus the main difficulty in power estimation, is that the power is *input pattern-dependent*.

† This work was supported in part by Intel Corp., Digital Equipment Corp., and USAF Rome Laboratory.

It is possible to overcome the pattern-dependence problem by using probabilities to describe the set of *all possible logic signals*, and then studying the power resulting from the collective influence of all these signals. This formulation achieves a certain degree of *pattern-independence* that allows one to efficiently estimate the power dissipation. Most recently proposed power estimation tools [2] are based on such a probabilistic approach, but are limited to combinational circuits. Only a few techniques have been proposed for sequential circuits, and they will be reviewed in the next section.

It is usually assumed that the circuit has the popular and well-structured design style of a *synchronous sequential circuit*, as shown in Fig. 1. In order to handle such sequential circuits, we propose a technique in which we apply a number of randomly generated input sequences to the circuit and collect statistics on the latch outputs using fast zero-delay logic simulation, or using functional simulation of a structural RTL description. Given these statistics, it is then possible to use any of the existing combinational circuit techniques to compute the total power.

## II. Background

Let $u_1, u_2, \ldots, u_m$ be the primary input nodes of a sequential logic circuit, as shown in Fig. 1, and let $x_1, x_2, \ldots, x_n$ be the *present state* lines. For simplicity of presentation, we have assumed that the circuit contains a single clock that drives a bank of edge-triggered latches. On the falling edge of the clock, the latches transfer the values at their inputs to their outputs. The inputs $u_i$ and the *present state* values determine the *next state* values and the circuit outputs, so that the circuit implements a *finite state machine* (FSM).

Most existing power estimation techniques handle only combinational circuits [2], and require information on the circuit input statistics (transition probabilities, etc). To allow extension to sequential circuits, it is therefore sufficient to compute statistics of the latch outputs (and corresponding latch power). Other existing techniques would then be applied to compute the power consumed in the combinational block.

We will briefly survey the few recently proposed techniques for estimating the power in sequential circuits. To simplify the discussion, we will assume that the sequential circuit implements a non-decomposable finite state machine. All proposed techniques that handle sequential circuits [3–6] make the simplifying assumption that the FSM is *Markov* [7], so that its future is independent of its past once its present state is specified.

Some of the proposed techniques compute only the probabilities (signal and transition) at the latch outputs, while others also compute the power. The approach in [3] solves directly for the transition probabilities on the present state lines using the Chapman-Kolmogorov equations [7], which is computationally too expensive. Another approach that also attempts a direct solution of the Chapman-Kolmogorov equations is given in [4]. While it is more efficient, it remains quite expensive, so that the largest test case presented contains less than 30 latches.

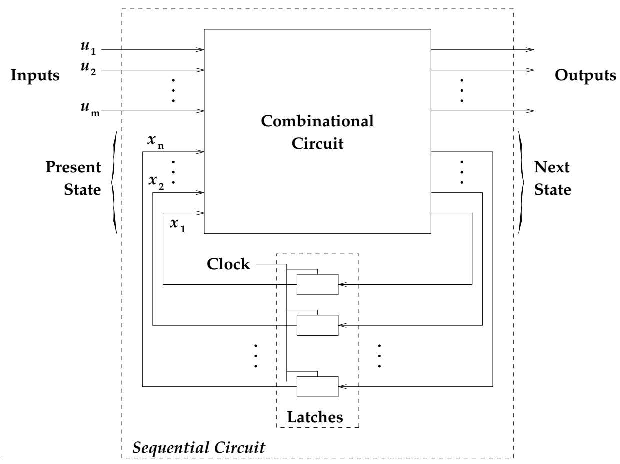


**Figure 1.** An FSM model of a sequential logic circuit.

Better solutions are offered by two recent papers [5, 6], which are based on solving a non-linear system that gives the present state line probabilities, as follows. Given probabilities $p_{u_1}, \ldots, p_{u_m}$ at the input lines, let a vector of *present state* probabilities $P_{p.s.} = [p_{x_1} \; \ldots \; p_{x_n}]$ be applied to the combinational logic block. *Assuming the present state lines are independent*, one can compute a corresponding *next state* probability vector as $F(P_{p.s.})$. The function $F(\cdot)$ is a non-linear vector-valued function that is determined by the Boolean function implemented by the combinational logic.

In general, if the next state probabilities form a vector $P_{n.s.}$, then $P_{n.s.} \neq F(P_{p.s.})$, because the latch outputs are not necessarily independent. Both methods [5, 6] make the independence assumption $P_{n.s.} \approx F(P_{p.s.})$. Finally, since $P_{n.s.} = P_{p.s.}$ due to the feedback, they obtain the state line probability values by solving the system $P = F(P)$. This system is solved using the Newton-Raphson method in [5], and using the Picard-Peano iteration method in [6].

One problem with this approach is that it is not clear that the system $P = F(P)$ has a *unique* solution. Being non-linear, it may have multiple solutions, and in that case it is not clear which is the correct one. Another problem is the independence assumption which need not hold in practice, especially in view of the feedback. Both techniques try to correct for this. In [5], this is done by accounting for $m$-wise correlations between state bits when computing their probabilities. This requires $2^m$ additional gates and can get very expensive. Nevertheless, they show good experimental results. The approach in [6] is to *unroll* the combinational logic block $k$ times. This is less expensive than [5], and the authors observe that with $k = 3$ or so, good results can be obtained. Finally, in order for the FSM to be Markov, its input vectors must be independent and identically distributed, which is another assumption that also may not hold in practice.

We offer a solution that makes no assumptions about the FSM behavior (Markov or otherwise), makes no independence assumption about the state lines, and allows the user to specify the desired accuracy and confidence to be achieved in the results. The only assumption we will make has to do with the autocovariance of the logic signals, which is mild and generally true for all but periodic logic signals, as explained below. We also assume that the user has information on the statistics of the FSM input signals.

## III. Problem Formulation

If every state of the machine is *reachable* from every other state in a finite number of cycles, then the FSM is said to be *non-decomposable*. Otherwise, it can be decomposed into a number of smaller FSMs, each of which being non-decomposable. Therefore, it is sufficient to study a single non-decomposable FSM:

**Assumption 1.** *The sequential circuit implements a non-decomposable FSM.*

Since the system is clocked, it is convenient to work with discrete time, so that the FSM inputs at time $k$, $u_i(k)$, and its present state at that time, $x_i(k)$, determine its next state, $x_i(k+1)$, and its output. In order to take into account the effect of large sets of inputs, one is typically interested in the average power dissipation over long periods of time. Therefore, we will assume that the FSM operates for all time $(-\infty < k < \infty)$. An infinite logic signal $x(k)$ can be characterized by two measures: *signal probability* $P(x)$ is the fraction of time that the signal is high, and *transition density* $D(x)$ is the average number of logic transitions per clock cycle. These measures are formally presented in the appendix, where it is also shown that $D(x) = P(t_x)$, where $t_x(k)$ is another logic signal derived from $x(k)$ so that $t_x(k) = 1$ only in those cycles where $x(k)$ makes a transition:

$$t_x(k) = \begin{cases} 1, & \text{if } x(k) \neq x(k-1); \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

In order to study the properties of a logic signal over $(-\infty, \infty)$, it is useful to consider a random model of logic signals. We will use **bold font** to represent random quantities. We denote the probability of an event $A$ by $\mathcal{P}\{A\}$ and, if $\mathbf{x}$ is a random variable, we denote its mean by $E[\mathbf{x}]$. An infinite logic signal $x(k)$ can be viewed as a sample of a stochastic process $\mathbf{x}(k)$, consisting of an infinite set of shifted copies of the logic signal. This process, which we call a *companion process*, embodies all the details of the logic signal, including its probability and density. Details and basic results related to the companion process are given in the appendix as an extension of previous continuous-time work [8]. Specifically, the companion process is *stationary*, and for any time instant $k$, the probability that $\mathbf{x}(k)$ is high is equal to the signal probability of the logic signal:

$$\mathcal{P}\{\mathbf{x}(k)=1\} = P(x) \tag{2}$$

This result holds for *any* logic signal. If we (conceptually) construct the companion processes corresponding to the FSM signals, then we can view the FSM as a system operating on stochastic inputs, consisting of the companion processes $\mathbf{u_1}(k), \mathbf{u_2}(k), \ldots, \mathbf{u_m}(k)$, and having a stochastic state consisting of the processes $\mathbf{x_1}(k), \mathbf{x_2}(k), \ldots, \mathbf{x_n}(k)$. Given statistics of the input vector $\mathbf{U}(k) = [\mathbf{u_1}(k)\ \mathbf{u_2}(k) \ldots \mathbf{u_m}(k)]$, one would like to compute some statistics of the state vector $\mathbf{X}(k) = [\mathbf{x_1}(k)\ \mathbf{x_2}(k)\ \ldots\ \mathbf{x_n}(k)]$.

Before going on, we will need to make one mild assumption related to covariance of the process $\mathbf{X}(k)$:

**Assumption 2.** *The state of the machine at time $k$ becomes independent of its initial state at time 0 as $k \to \infty$.*

This assumption is mild because it is generally true in practice that, for all non-periodic logic signals, two values of the signal that are separated by a large number of clock cycles become increasingly unrelated. One necessary condition of this assumption is that the FSM be *aperiodic*, i.e., that it does not cycle through a repetitive pattern of states. Aperiodicity is implicitly assumed by most previous work on sequential circuits. Specifically, whenever an FSM is assumed Markov (in which case aperiodicity becomes equivalent to the above assumption), the FSM is usually also assumed to be aperiodic [5, 6].

Before leaving this section, we consider the question of exactly what statistics of $\mathbf{X}(k)$ are required in order to estimate the power. These statistics must be sufficient to compute the combinational circuit power. Most techniques for combinational circuit power estimation [2] require the signal probability and transition density at every input (for discrete-time signals, knowing the transition density is equivalent to knowing the transition probability). Since the power consumed in the latches can also be derived from $D(x_i)$, then the state line $P(x_i)$ and $D(x_i)$ are enough to compute the power for the whole circuit. However, since $D(x) = P(t_x)$, where $t_x$ is defined in (1), it will be sufficient to describe an algorithm by which $P(x_i)$ can be computed. The same algorithm can be used to compute $P(t_{x_i}) = D(x_i)$. Such an algorithm is presented in the next section.

## IV. Computing State Line Probabilities

We propose to obtain the state line probabilities by performing Monte Carlo logic simulation of the design using high-level functional description, say at the register transfer level (RTL), and computing the probabilities from the large number of samples produced. High-level simulation can be done very fast, so that one can afford to simulate a large number of cycles. However, one has to decide how long to simulate in order to get meaningful statistics. It is also important to choose the random inputs in accordance with user-specified statistics of the FSM input vector. Both of these issues are discussed below.

### *A. Convergence*

Suppose the FSM is known to be in some state $X_0$ at time 0. Using (2), and given assumption 2, we have for any state signal $x_i$:

$$\begin{aligned}\lim_{k\to\infty} \mathcal{P}\{\mathbf{x_i}(k)=1 \mid \mathbf{X}(0)=X_0\} &= \lim_{k\to\infty} \mathcal{P}\{\mathbf{x_i}(k)=1\} \\ &= P(x_i)\end{aligned} \tag{3}$$

For brevity, we denote the conditional probability by:

$$P_k(x_i \mid X_0) = \mathcal{P}\{\mathbf{x_i}(k)=1 \mid \mathbf{X}(0)=X_0\}$$

Our method consists of estimating $P_k(x_i \mid X_0)$ for increasing values of $k$ until convergence (according to (3)) is achieved. To achieve this, we perform repeated simulation runs of the circuit, starting from some state $X_0$, and drive the simulation with randomly generated input vectors $[u_1\ u_2\ \cdots\ u_m]$ (consistent with the statistics of $\mathbf{U}(k)$). Each run results in a logic waveform $x_i^{(j)}(k)$, $k = 0, 1, 2\ldots$, where $j$ designates the run number. If we average the results at every time $k$ we obtain an estimate of the probability at that time as follows:

$$p_i^{(N)}(k) = \frac{1}{N}\sum_{j=1}^{N} x_i^{(j)}(k)$$

From the law of large numbers, it follows that:

$$\lim_{N\to\infty} p_i^{(N)}(k) = P_k(x_i \mid X_0)$$

We do not actually have to perform an infinite number of runs. Using established techniques for the estimation of proportions [9], we can predict how many runs to perform in order to achieve some user-specified error-tolerance ($\epsilon$) and confidence ($\alpha$) levels. Specifically, it can be shown [10] that if we want $(1-\alpha)\times 100\%$ confidence that:

$$\left| p_i^{(N)}(k) - P_k(x_i \mid X_0) \right| < \epsilon \tag{4}$$

then we must perform at least $N \geq \max(N_1^2, N_2^2, N_3^2)$ runs, where:

$$N_1 = \frac{z_{\alpha/2}}{2\epsilon}, \quad N_2 = \frac{z_{\alpha/2}\sqrt{2\epsilon+0.1} + \sqrt{(\epsilon+0.1)z_{\alpha/2}^2 + 3\epsilon}}{2\epsilon},$$

$$\text{and} \qquad N_3 = \frac{\sqrt{63} + z_{\alpha/2}}{2\sqrt{\epsilon}}$$

and where $z_{\alpha/2}$ is such that the probability that a standard normal random variable is greater than $z_{\alpha/2}$ is equal to $\alpha/2$. The value of $z_{\alpha/2}$ can be obtained from the $\mathbf{erf}(\cdot)$ function available on most computer systems. For instance,

$z_{\alpha/2} = 1.96$ for 95% confidence (i.e., $\alpha = 0.05$), and $z_{\alpha/2} = 2.575$ for 99% confidence. From the above equations, it can be seen that 500 runs are enough to obtain a result with accuracy $\epsilon = 0.05$ and 95% confidence.

From user-specified $\epsilon$ and $\alpha$, the required value of $N$ can be found up-front. Given this, we initiate $N$ parallel simulations of the FSM and estimate $P_k(x_i \mid X_0) \approx p_i^{(N)}(k)$ for increasing $k$ values. The remaining question is how to determine when $k$ is large enough so that this estimate can be said to have converged to $P(x)$. We use two measures to check on this convergence, as follows. We perform two sets of simulation runs of the machine, starting from different arbitrary initial states $X_0$ and $X_1$, so as to estimate $P_k(x_i \mid X_0)$ and $P_k(x_i \mid X_1)$ for increasing $k$ values. Since both should converge to $P(x)$, we monitor both their *difference* and their *average*. When both of these measures have remained within a window of $\pm\epsilon$ for three consecutive time instants, we declare that that node has converged. The choice of three time instants is somewhat arbitrary and can be changed by the user. To speed up convergence, we actually *filter* the two waveforms $P_k(x_i \mid X_0)$ and $P_k(x_i \mid X_1)$ before checking their difference and average, using a 100 point FIR filter procedure with a cutoff frequency of 0.02 Hz. When all nodes have converged, the simulation is terminated and the average of the last available $P_k(x_i \mid X_0)$ and $P_k(x_i \mid X_1)$ values is reported as the signal probability $P(x_i)$, for each $x_i$.

### B. Input Generation

In view of assumption 2, one requirement on the applied input sequence $U(k)$ is that it not be periodic. Another condition, required for the estimation (4) to hold, is that the different $U^{(j)}(k)$ sequences used in different simulation runs $j = 1, \ldots, N$ be generated *independently.* Otherwise, no limitations are placed on the input sequence.

The exact way in which the inputs are generated depends on the design and on what information is available about the inputs. For instance if the FSM is meant to execute micro-code from a fixed set of instructions, then every sequence $U^{(j)}(k)$ may be a piece of some micro-code program. This method of input generation faithfully reproduces the bit correlations in $U(k)$ as well as the temporal correlation between $U(k), U(k+1), \ldots$. Alternatively, if the user has information on the relative frequency with which instructions occur in practice, but no specific program from which to select instruction sequences, then a random number generator can be used to select instructions at random to be applied to the machine. This would preserve the bit correlations, but not the temporal correlations between successive instructions. Conceivably, if such correlation data is available, one can bias the random generation process to reproduce these correlations.

In more general situations, where the machine inputs can be arbitrary, simpler random generation processes can be used. For instance, it may not be important in some applications to reproduce the correlations between bits and between successive vectors. The user may only have information on the statistics of the individual input bits, such as the probability $P(u_i)$ and density $D(u_i)$ for every input. In this case, one can design a random generation process to produce signals that have the required $P$ and $D$ statistics, as follows.

Using equations (A.3) in the appendix, one can compute from $P$ and $D$ the mean *high time* and mean *low time* of the signal. By assuming a certain distribution type for the high and low pulse widths, one can then easily generate a logic signal with the required statistics. The choice of distribution is not very important because, as observed in [11], the power is relatively insensitive to the particular distribution, rather it depends mainly on the input densities. For instance, if one uses a geometric distribution (which is equivalent to the signals $x_i$ being individually Markov), then one obtains a fixed value for the probabilities $\mathcal{P}\{\mathrm{x_i}(k) = 1 \mid \mathrm{x_i}(k-1) = 0\}$ and $\mathcal{P}\{\mathrm{x_i}(k) = 0 \mid \mathrm{x_i}(k-1) = 1\}$, as shown in [12], and generates the logic signals accordingly. Incidentally, in this case, even though the inputs are Markov, the FSM itself is not necessarily Markov.

Finally, if only the probabilities $P(x_i)$ are available for the input nodes, and if it is not important to reproduce any input correlation information, one can generate the inputs by a sequence of *coin flips* using a random number generator. In this case, the inputs are said to be i.i.d. (independent and identically distributed) and the FSM can be shown to be Markov, but the individual state bits $x_i$ may not be Markov.

Our implementation results for this approach, reported in the next section, are based on this last case of i.i.d. inputs. However, the technique is applicable to any other mechanism of input generation, as we have explained.

## V. Experimental Results

This technique was implemented in a prototype C program that accepts a netlist description of a synchronous sequential machine. The program performs a zero delay logic simulation and monitors the node probabilities until they converge. To improve the speed, we simulate 31 copies of the machine in parallel, using bit-wise operations. We have tested the program on a number of circuits from the ISCAS-89 sequential benchmark set [13].

All the results to be presented will be based on an error-tolerance of 0.05, i.e., $\epsilon = 0.05$, and 95% confidence, i.e., $\alpha = 0.05$. For initial states, we used $X_0 = [0\ 0\ \ldots\ 0]$ and $X_1 = [1\ 1\ \ldots\ 1]$. Under these conditions, a typical convergence characteristic is shown in Fig. 2. The two waveforms shown correspond to $p_i^{(N)}(k)$ starting from $X_0$ and $p_i^{(N)}(k)$ starting from $X_1$, for node X.3 of circuit s838.1 (this circuit has 34 inputs, 32 flip-flops, and 446 gates). This decaying sinusoidal convergence is typical, although in some cases the convergence is simply a decaying exponential and is much faster.

In order to assess the accuracy of the technique, we compared the (0.05, 95%) results to those of a much more accurate run of the same program (using 0.005 error tolerance and 99% confidence). Since one is interested only in steady state node values during the simulation, there is no need to use a more accurate timing simulator or a circuit simulator to make these comparisons. These highly accurate runs take a long time and, therefore, they were only performed on the circuits s1196, s1238, s713, and s1423. We then computed the difference between the probabilities $P(x_i)$ from the (0.05, 95%) run and those from the (0.005, 99%) run. Figure 3 shows the resulting error histogram for all the latch outputs from these circuits. Notice that all the nodes have errors well within the specified 0.05 error bounds.

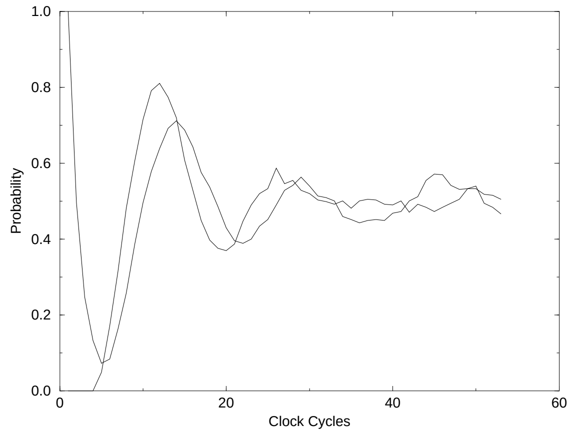


**Figure 2.** Convergence of probability for s838.1, node X.3.

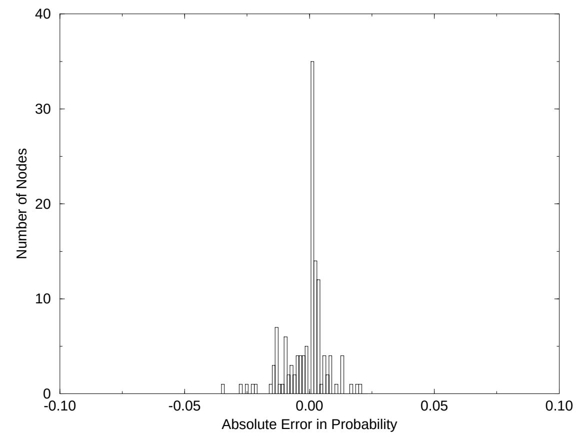


**Figure 3.** Latch probability error histogram.

We also monitored the speed of this technique and report some results in Table I. These circuits are much larger (especially in terms of flip-flop count) than the largest circuits tested in previous methods [4–6]. Furthermore, for those circuits in the table that were also tested in [4–6], this technique works much faster. Finally, since our method does not use BDDs to compute probabilities, there are no memory problems with running large circuits.

## VI. Summary and Conclusions

Most existing power estimation techniques are limited to combinational circuits, while all practical circuit designs are sequential. We have presented a new technique for power estimation in sequential circuits that is based on a statistical estimation technique. By applying randomly generated input sequences to the circuit, statistics on the latch outputs are collected, by simulation, that allow efficient power estimation for the whole design. An important advantage of this approach is that the desired accuracy of the results can be specified up-front by the user. We have implemented this technique and tested it on a number of benchmark sequential circuits, with excellent results.

**Table I.**
Execution Time on a SUN Sparc10.

| Circuit | #inputs | #latches | #gates | cpu time |
|---|---|---|---|---|
| s1238 | 14 | 18 | 508 | 24.34 sec |
| s1196 | 14 | 18 | 529 | 24.96 sec |
| s713 | 35 | 19 | 393 | 22.80 sec |
| s838.1 | 34 | 32 | 446 | 28.12 sec |
| s1423 | 17 | 74 | 657 | 55.11 sec |
| s5378 | 35 | 179 | 2,779 | 3.81 min |
| s9234.1 | 36 | 211 | 5,597 | 1.05 hrs |
| s15850.1 | 38 | 534 | 9,796 | 3.34 hrs |
| s13207.1 | 62 | 638 | 7,951 | 3.23 hrs |
| s38584 | 12 | 1,452 | 19,253 | 4.59 hrs |

## Appendix A. Discrete-Time Logic Signals

Let $\mathcal{Z} = \{\ldots, -2, -1, 0, 1, 2, \ldots\}$ be the set of all integers, and let $x(k), k \in \mathcal{Z}$, be a function of discrete time that takes the values 0 or 1, i.e., $x(k)$ is a *discrete-time logic signal*. The definitions and results below are extensions of similar concepts developed in [8]. The results are therefore given without proof.

### *A.1 Probability and Density*

Notice that the set $\{\lfloor -K/2 \rfloor + 1, \ldots, \lfloor +K/2 \rfloor\}$ contains exactly $K$ elements, where $K > 0$ is a positive integer.

**Definition 1.** *The signal probability of $x(k)$, to be denoted $P(x)$, is defined as:*

$$P(x) = \lim_{K \to \infty} \frac{1}{K} \sum_{k=\lfloor -K/2 \rfloor + 1}^{\lfloor +K/2 \rfloor} x(k) \qquad (A.1)$$

It can be shown that the limit in (A.1) always exists.

If $x(k) \neq x(k-1)$, we say that the signal undergoes a *transition* at time $k$. Corresponding to every logic signal $x(k)$, one can construct another logic signal $t_x(k)$ so that $t_x(k) = 1$ if $x(k)$ undergoes a transition at $k$, otherwise $t_x(k) = 0$. Let $n_x(K)$ be the number of transitions of $x(k)$ over $\{\lfloor -K/2 \rfloor + 1, \ldots, \lfloor +K/2 \rfloor\}$. Therefore, $n_x(K) \leq K$.

**Definition 2.** *The transition density of a logic signal $x(k)$, denoted by $D(x)$, is defined as:*

$$D(x) = \lim_{K \to \infty} \frac{n_x(K)}{K} \qquad (A.2)$$

Notice that $n_x(K) = \sum_{k=\lfloor -K/2 \rfloor+1}^{\lfloor +K/2 \rfloor} t_x(k)$, so that $D(x) = P(t_x)$, and the limit in (A.2) exists.

The time between two consecutive transitions of $x(k)$ will be referred to as an *inter-transition time*: if $x(k)$ has a transition at $i$ and the next transition is at $i+n$, then there is an intertransition time of length $n$ between the two transitions. Let $\mu_1$ ($\mu_0$) be the average of the high (low), i.e., corresponding to $x(k) = 1$ (0), inter-transition times of $x(k)$. In general, there is no guarantee of the existence of $\mu_0$, and $\mu_1$. If the number of transitions in positive time is *finite*, then we say that there is an *infinite* inter-transition time following the last transition, and $\mu_0$ or $\mu_1$ will not exist. A similar convention is made for negative time.

**Proposition 1.** *If $\mu_0$ and $\mu_1$ exist, then:*

$$P(x) = \frac{\mu_1}{\mu_0 + \mu_1} \quad \text{and} \quad D(x) = \frac{2}{\mu_0 + \mu_1} \qquad (A.3a, b)$$

*A.2 The Companion Process*

Let $\mathrm{x}(k)$, $k \in \mathcal{Z}$, be a discrete-time *stochastic process* [7] that takes the values 0 or 1, transitioning between them at *random* discrete transition times. Such a process is called a *0-1 process*. A logic signal $x(k)$ can be thought of as a *sample* of a 0-1 stochastic process $\mathrm{x}(k)$, i.e., $x(k)$ is one of an infinity of possible signals that comprise the family $\mathrm{x}(k)$.

A stochastic process is said to be *stationary* if its statistical properties are invariant to a shift of the time origin [7]. Among other things, the mean $E[\mathrm{x}(k)]$ of such a process is a constant, independent of time, and will be denoted by $E[\mathrm{x}]$. Let $\mathbf{n}_\mathrm{x}(K)$ denote the number of transitions of $\mathrm{x}(k)$ over $\{\lfloor -K/2 \rfloor + 1, \ldots, \lfloor +K/2 \rfloor\}$. For a given $K$, $\mathbf{n}_\mathrm{x}(K)$ is a random variable. If $\mathrm{x}(k)$ is stationary, then $E[\mathbf{n}_\mathrm{x}(K)]$ depends only on $K$, and is independent of the location of the time origin. Furthermore, one can show that if $\mathrm{x}(k)$ is stationary, then the mean $E[\mathbf{n}_\mathrm{x}(K)/K]$ is constant, irrespective of $K$.

Let $\mathbf{z} \in \mathcal{Z}$ be a random variable with the cumulative distribution function $F_\mathbf{z}(k) = 1/2$ for any finite $k$, and with $F_\mathbf{z}(-\infty) = 0$ & $F_\mathbf{z}(+\infty) = 1$. One might say that $\mathbf{z}$ is uniformly distributed over the whole integer set $\mathcal{Z}$. We use $\mathbf{z}$ to construct from $x(k)$ a stochastic 0-1 process $\mathrm{x}(k)$, called its *companion process*, defined as follows.

**Definition 3.** *Given a logic signal $x(k)$ and a random variable $\mathbf{z}$, uniformly distributed over $\mathcal{Z}$, define a 0-1 stochastic process $\mathrm{x}(k)$, called the companion process of $x(k)$, given by:*

$$\mathrm{x}(k) = x(k + \mathbf{z}) \qquad (A.4)$$

For any given $k = k_1$, $\mathrm{x}(k_1)$ is the random variable $x(k_1 + \mathbf{z})$ - a function of the random variable $\mathbf{z}$. Intuitively, $\mathrm{x}(k)$ is a family of shifted copies of $x(k)$, each shifted by a value of the random variable $\mathbf{z}$. Thus, not only is $x(k)$ a sample of $\mathrm{x}(k)$, but one can also relate statistics of the process $\mathrm{x}(k)$ to properties of the logic signal $x(k)$, as follows.

**Proposition 2.** *The companion process $\mathrm{x}(k)$ of a logic signal $x(k)$ is stationary, with:*

$$E[\mathrm{x}] = \mathcal{P}\{\mathrm{x}(k) = 1\} = P(x), \quad \text{and} \quad E\left[\frac{\mathbf{n}_\mathrm{x}(K)}{K}\right] = D(x) \qquad (A.5a, b)$$

## REFERENCES


[1] R. W. Brodersen, A. Chandrakasan, S. Sheng, "Technologies for personal communications," *1991 Symp. on VLSI circuits*, Tokyo, Japan, pp. 5-9, 1991.

[2] F. Najm, "A survey of power estimation techniques in VLSI circuits," *IEEE Transactions on VLSI Systems*, pp. 446–455, Dec. 1994.

[3] A. A. Ismaeel and M. A. Breuer, "The probability of error detection in sequential circuits using random test vectors," *Journal of Electronic Testing*, vol. 1, pp. 245–256, January 1991.

[4] G. D. Hachtel, E. Macii, A. Pardo, and F. Somenzi, "Probabilistic analysis of large finite state machines," *31st ACM/IEEE Design Automation Conference*, San Diego, CA, pp. 270–275, June 6–10, 1994.

[5] J. Monteiro and S. Devadas, "A methodology for efficient estimation of switching activity in sequential logic circuits," *ACM/IEEE 31st Design Automation Conference*, San Diego, CA, pp. 12–17, June 6–10, 1994.

[6] C-Y Tsui, M. Pedram, and A. M. Despain, "Exact and approximate methods for calculating signal and transition probabilities in FSMs," *ACM/IEEE 31st Design Automation Conference*, San Diego, CA, pp. 18–23, June 6–10, 1994.

[7] A. Papoulis, *Probability, Random Variables, and Stochastic Processes*, 2nd Edition. New York, NY: McGraw-Hill Book Co., 1984.

[8] F. Najm, "Transition density: a new measure of activity in digital circuits," *IEEE Transactions on Computer-Aided Design*, vol. 12, no. 2, pp. 310-323, February 1993.

[9] I. R. Miller, J. E. Freund, and R. Johnson, *Probability and Statistics for Engineers*, 4th Edition. Englewood Cliffs, NJ: Prentice-Hall Inc., 1990.

[10] F. Najm, "Statistical estimation of the signal probability in VLSI circuits," University of Illinois at Urbana-Champaign, Coordinated Science Lab. Report #UILU-ENG-93-2211, DAC-37, April 1993.

[11] R. Burch, F. Najm, P. Yang, and T. Trick, "A Monte Carlo approach for power estimation," *IEEE Transactions on VLSI Systems*, vol. 1, no. 1, pp. 63–71, March 1993.

[12] M. Xakellis and F. Najm, "Statistical estimation of the switching activity in digital circuits," *31st ACM/IEEE Design Automation Conference*, San Diego, CA, pp. 728–733, 1994.

[13] F. Brglez, D. Bryan, and K. Koźmiński, "Combinational profiles of sequential benchmark circuits," *IEEE International Symposium on Circuits and Systems*, pp. 1929–1934, 1989.