Transition Density, A Stochastic Measure of Activity in Digital Circuits

Farid N. Najm

Semiconductor Process & Design Center
Texas Instruments Inc., MS 369
Dallas, Texas 75265

Abstract

Reliability assessment is an important part of the design process of digital integrated circuits. We observe that a common thread that runs through most causes of run-time failure is the extent of circuit *activity*, i.e., the rate at which its nodes are switching. We propose a new measure of activity, called the *transition density*, which may be defined as the "average switching rate" at a circuit node. Based on a stochastic model of logic signals, we rigorously define the transition density and present an algorithm to propagate it from the primary inputs to internal and output nodes. This algorithm may be thought of as a *simulation* of the circuit, and has been implemented in a prototype *density simulator*. We present some results of this implementation to verify the theoretical results and assess the feasibility of the approach. In order to obtain the same density information by traditional means, the circuit would need to be simulated for thousands of input transitions. Thus this approach is very efficient and makes possible the analysis of VLSI circuits, which are traditionally too big to simulate for long input sequences.

ACM/IEEE Design-Automation Conference, 1991.

1. Introduction

A major portion of the design time of digital integrated circuits is dedicated to functional verification and reliability assessment. Of these two, reliability assessment is a more recent problem whose severity has steadily increased in proportion to chip density. As a result, CAD tools that evaluate the susceptibility of a design to run-time failures are becoming increasingly important.

Chip run-time failures can occur due to a variety of reasons, such as excessive power dissipation, electromigration, hot-electron degradation, voltage drop, aging, and others. In CMOS logic circuits, the rate at which node transitions occur is a good indicator of the circuit's susceptibility to run-time failures. For example, both power dissipation and electromigration in the power lines are directly related to the power supply current which, in CMOS, is non-zero only during transitions. Hot-electron degradation is related to the MOSFETs substrate current, which, for CMOS, is also only significant during transitions. Thus, the rate at which node transitions occur, i.e., the extent of circuit *activity*, may be thought of as a measure of a failure-causing *stress*.

This paper proposes a new way of quantifying activity in digital circuits and presents a simulation technique to compute a measure of activity that we call the *transition density*. The transition density may be defined as the "average switching rate" at a circuit node; a more rigorous definition will be given in later sections.

To further motivate the notion of transition density, consider the problem of estimating the average power drawn by a CMOS gate. If the gate has output capacitance C to ground and generates a simple clock signal with frequency f, then the average power dissipated is $CV_{dd}^2 f$, where V_{dd} is the power supply voltage. In general, a node in a logic circuit may not carry a periodic signal, and the notion of frequency cannot be used to write this simple expression for average power. Instead, one may compute the power as follows. If $x(t)$ is the logic signal at the gate output and $n_x(T)$ is the number of transitions of $x(t)$ in the time interval $(\frac{-T}{2}, \frac{+T}{2}]$, then the average power is :

$$P_{av} = \lim_{T\to\infty} V_{dd} \frac{CV_{dd} n_x(T)/2}{T} = \frac{1}{2} CV_{dd}^2 \left\{ \lim_{T\to\infty} \frac{n_x(T)}{T} \right\} \tag{1.1}$$

It will be shown below (see (3.2)) that the last term in (1.1) is indeed the transition density.

If the transition densities at the circuit primary inputs are given, we will show that they can be propagated into the circuit to give the transition densities at all internal and output nodes. The propagation algorithm may be thought of as a *simulation* of the circuit, and has been implemented in a prototype *density simulator*. It performs a single-pass over the circuit to compute the transition densities at all the nodes. In order to obtain the same density information by traditional means, the circuit would need to be simulated for thousands of input transitions. Thus this approach is very efficient and makes possible the analysis of VLSI circuits, which are traditionally too big to simulate for long input sequences.

The analysis leading up to the notion of density and its propagation algorithm is best cast in a stochastic (probability) setting. Thus, in the following sections, we will present a stochastic model of logic signals, rigorously define the transition density based on that model, and then present algorithms that propagate the density into the circuit. We will also describe our implementation of this approach and present some test cases to assess its validity and feasibility.

2. Stochastic 0-1 Processes

In this section, we review some concepts from probability theory, present a model of random logic signals, define the transition density, and basically lay down the foundations for the rest of the paper. Throughout this paper, we will use **bold font** to represent random quantities.

Let $\mathbf{x}(t)$, $t \in (-\infty, +\infty)$, be a *stochastic process* [1] that takes the values 0 or 1, transitioning between them at *random* transition times. Such a process is called a *0-1 process* (see [2], pp. 38–39).

A stochastic process is said to be *strict-sense stationary* (SSS) if its statistical properties are invariant to a shift of the time origin [1]. Specifically, the mean of such a process is a constant, independent of time. If a constant-mean process $\mathbf{x}(t)$ has finite variance and is such that $\mathbf{x}(t)$ and $\mathbf{x}(t+\tau)$ become uncorrelated as $\tau \to \infty$, then $\mathbf{x}(t)$ is *mean-ergodic* (see [1], pp. 245–248). These conditions, which are satisfied for most regular processes [1], are sufficient but not necessary for mean-ergodicity. However, in order to abbreviate the terminology, we will, throughout this paper, simply use the term "mean-ergodic" to mean "mean-ergodic *and* satisfies the two conditions of finite variance and decaying auto-correlation."

In the sequel, we consider only 0-1 processes that are SSS and mean-ergodic. We will show in the next section that, if the circuit primary inputs are represented by SSS mean-ergodic 0-1 processes, then the 0-1 processes at all internal and output nodes are also SSS and mean-ergodic.

To show that such processes *exist* and that the assumptions of SSS and ergodicity are in fact *mild* requirements, we point out the important special case when $\mathbf{x}(t)$ has the *Markov* (memoryless) property. In that case, $\mathbf{x}(t)$ is the well-known two-state continuous-time Markov process (see [1], pp. 392–393). With an appropriate time-zero condition, such a signal becomes SSS (see [3], pp. 272–273). It also has independent inter-transition times, which makes it mean-ergodic.

Since $\mathbf{x}(t)$ is SSS, then the probability that it takes the value 1 at any given time t, denoted by $\mathcal{P}\{\mathbf{x}(t)=1\}$, which is equal to its *mean* or *expected value* $E[\mathbf{x}(t)]$ at that time, is a constant, independent of time. We refer to this value as the *equilibrium probability* of $\mathbf{x}(t)$, denoted by $P(\mathbf{x}) \stackrel{\triangle}{=} \mathcal{P}\{\mathbf{x}(t)=1\}$. Since $\mathbf{x}(t)$ is also mean-ergodic, then [1] :

$$\lim_{T\to\infty} \frac{1}{T}\int_{\frac{-T}{2}}^{\frac{+T}{2}} \mathbf{x}(t)dt \stackrel{1}{=} E[\mathbf{x}(t)] = P(\mathbf{x}) \tag{2.1}$$

where we have used the symbol "$\stackrel{1}{=}$" to denote *convergence with probability 1.* The reader is referred to [1], pp. 188–191, for a discussion of the different stochastic convergence modes.

Let $\mathbf{n_x}(T)$ denote the (random) number of transitions of $\mathbf{x}(t)$ in $(\frac{-T}{2}, \frac{+T}{2}]$. Since $\mathbf{x}(t)$ is SSS, then $E[\mathbf{n_x}(T)]$ depends only on T, and is independent of the location of the time origin. Hence $E[\mathbf{n_x}(T_1+T_2)] = E[\mathbf{n_x}(T_1)] + E[\mathbf{n_x}(T_2)]$. This necessarily means that $E[\mathbf{n_x}(T)] = kT$, where k is a positive constant. The ratio $E[\mathbf{n_x}(T)]/T$ is the expected number of transitions per unit time. We will refer to it as the *transition density* of $\mathbf{x}(t)$, to be denoted by $D(\mathbf{x}) \stackrel{\triangle}{=} E[\mathbf{n_x}(T)]/T$. The name "transition density" is inspired by the *density* of random Poisson points (see [1], page 58) : If you throw a large number of points on the time axis at random, then the "number of points in a given interval" is a random variable with a Poisson distribution whose *density* parameter λ is the "expected number of points per unit time." The *points* that we are concerned with in this paper are the time points at which transitions occur, but we make no assumption about their distribution. The above remark about Poisson points is meant only to motivate the terminology.

Let $\mathbf{z}(t) \stackrel{\triangle}{=} \frac{d}{dt}\mathbf{m}_{\mathbf{x}}(t)$ be the output of a *differentiator* (see [1], page 238), whose input is $\mathbf{m}_{\mathbf{x}}(t) =$"the number of transitions of $\mathbf{x}(t)$ in $(0, t]$," if $t \geq 0$, or "the negative of the number of transitions of $\mathbf{x}(t)$ in $(t, 0]$," if $t < 0$. Since $\mathbf{x}(t)$ is SSS, it can be shown that $E[\mathbf{m}_{\mathbf{x}}(t)]/t = D(\mathbf{x})$. The process $\mathbf{z}(t)$ is an *impulse train* that can be written as :

$$\mathbf{z}(t) = \sum_{i=-\infty}^{+\infty} \delta(t - \mathbf{t_i}) \tag{2.2}$$

where the $\mathbf{t_i}$s are the transition time points of $\mathbf{x}(t)$. Since a differentiator is a linear system, we have that :

$$E[\mathbf{z}(t)] = \frac{d}{dt}E[\mathbf{m}_{\mathbf{x}}(t)] = D(\mathbf{x}) \tag{2.3}$$

Therefore, $\mathbf{z}(t)$ has a constant mean. In fact, since $\mathbf{z}(t) = |\frac{d}{dt}\mathbf{x}(t)|$, which is a *time-invariant* transformation, then (using [1], page 238) $\mathbf{z}(t)$ is also SSS. Since it is positive with finite mean then its variance must also be finite, and we can further argue that, since $\mathbf{x}(t)$ and $\mathbf{x}(t+\tau)$ are uncorrelated for large τ, then the same must be true for $\mathbf{z}(t)$ and $\mathbf{z}(t+\tau)$. Thus, $\mathbf{z}(t)$ is mean-ergodic and :

$$\lim_{T\to\infty} \frac{\mathbf{n}_{\mathbf{x}}(T)}{T} = \lim_{T\to\infty} \frac{1}{T} \int_{\frac{-T}{2}}^{\frac{+T}{2}} \mathbf{z}(t)dt \stackrel{1}{=} E[\mathbf{z}(t)] = D(\mathbf{x}) \tag{2.4}$$

Equations (2.1) and (2.4) are the main results of this section.

3. Modeling Random Logic Signals

Let $x(t)$ be a logic signal that transitions with zero rise and fall times between its 0 & 1 values. It is clear that $x(t)$ can be thought of as a *sample* of a stochastic 0-1 process $\mathbf{x}(t)$, i.e., $x(t)$ is one of an infinity of possible signals that comprise the family $\mathbf{x}(t)$.

If $\mathbf{x}(t)$ is SSS and mean-ergodic, then in view of (2.1) we have :

$$\lim_{T\to\infty} \frac{1}{T} \int_{\frac{-T}{2}}^{\frac{+T}{2}} x(t)dt \stackrel{1}{=} P(\mathbf{x}) \tag{3.1}$$

which means : "$P(\mathbf{x})$ is, asymptotically, the fraction of time that $x(t)$ spends in the 1 state."

Similarly, in view of (2.4), we have :

$$\lim_{T\to\infty} \frac{n_x(T)}{T} \stackrel{1}{=} D(\mathbf{x}) \tag{3.2}$$

Thus "$D(\mathbf{x})$ is, asymptotically, the average number of transitions of $x(t)$ per unit time." The two results (3.1) & (3.2) provide the essential link between physical reality (the quantities on the left hand side) and the stochastic measures $P(\mathbf{x})$ & $D(\mathbf{x})$.

We will use SSS mean-ergodic 0-1 processes to model the variety of logic waveforms that may be applied at the primary inputs of a digital circuit. We further assume that the processes at the circuit primary inputs are *mutually-independent*. Therefore, since these inputs are individually SSS, they are also *jointly* SSS. We now examine the properties of the corresponding 0-1 stochastic processes at the *internal nodes* of the circuit (we use the term internal nodes to refer to the primary output nodes as well as other proper internal circuit nodes).

One can think of a digital circuit as a nonlinear but *time-invariant* system that operates on its input waveforms to produce its internal waveforms. Naturally, if both the system and the statistics of its inputs are invariant to a shift in the time origin then so must be the statistics of its outputs (see [1], page 238). Thus internal processes are also SSS. Furthermore, since the circuit delays are finite, then the correlations of internal processes also die out for large τ, and they are also mean-ergodic. Therefore, if the primary inputs are SSS and mean-ergodic, the same is true of all internal signals. *All the above results are consequently true not only at the primary inputs but at every internal node as well.*

The next two sections deal with the issue of propagation and give a computationally efficient technique to propagate the densities and equilibrium probabilities through the circuit.

4. Propagation Through A Module

Consider a multi-input multi-output logic module M, whose outputs are Boolean functions of its inputs, as shown in Fig. 1. M may be a single logic gate or a higher level circuit block. We assume that the inputs to M are mutually-independent SSS mean-ergodic 0-1 processes.

We use a *simplified timing model* of circuit behavior, as follows. We assume that an input transition that *does* get transmitted to an output node is delayed by a *propagation delay* time of τ_p. Different propagation delays may be associated with different input-output node pairs. Implicit in this model is the simplifying assumption that the propagation delay is independent of the values at other inputs of M.

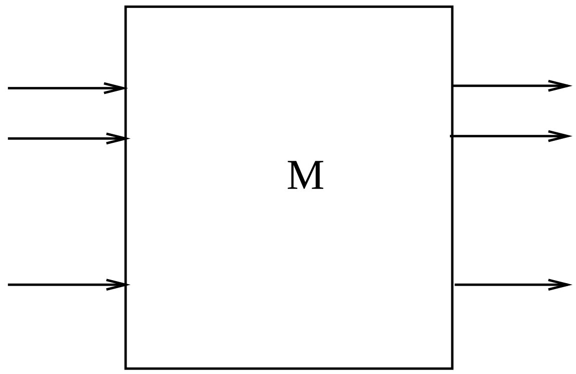

Figure 1. Logic Module M.

In effect, we decouple the delays inside M from its Boolean function description by introducing a new special-purpose delay block to model the delays between every pair of input & output nodes, as shown in Fig. 2. The block M' is a zero-delay logic module that implements the same Boolean function as M.

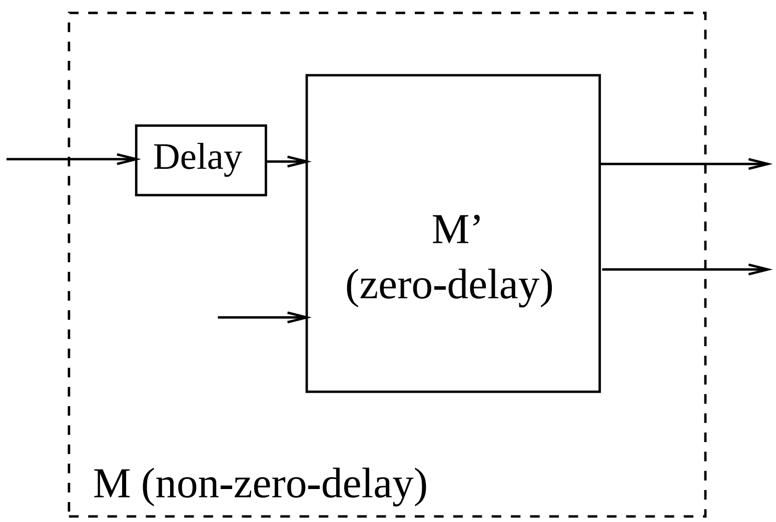

Figure 2. Decoupling of delays.

Since the signals are SSS, then the output of the delay block has the same statistics as its input, and therefore has the same probability and density. As for the zero-delay module M', we now consider the problem of propagating equilibrium probabilities and transition densities from its inputs to its outputs.

Since M' has zero delay, then the problem of propagating equilibrium probabilities through it is identical to that of propagating *signal probabilities* through logic circuits, which has been well-studied [4]–[8]. Since the internal structure of M' is not known, the problem is actually even more generic than that, and can be expressed as "given a Boolean function $f(x_1, \ldots, x_n)$ and that each x_i can be high with probability $P(\mathbf{x_i})$, what is the probability that f is high?" Any number of published techniques can be used to solve this problem. However, we have chosen (for reasons that will become clear below) to investigate a new approach based on Binary Decision Diagrams [9, 10] (BDDs) which have recently become popular in the verification and synthesis areas. The appendix describes how we use BDDs to compute the probability of a Boolean function.

We consider next the density propagation problem. Recall the concept of *Boolean Difference.* If y is a Boolean function that depends on x, then the Boolean difference of y with respect to x is defined as :

$$\frac{\partial y}{\partial x} \triangleq y|_{x=1} \oplus y|_{x=0} = y(x) \oplus y(\overline{x}) \tag{4.1}$$

where $\oplus$ denotes the exclusive-or operation. Note that, if x is an input and y is an output of M', then $\partial y/\partial x$ is a Boolean function that does *not* depend on x, but may depend on all other inputs of M'. Therefore, $\partial y/\partial x$ and x are independent. A crucial observation is that if $\partial y/\partial x$ is 1, then a transition at x will cause a (simultaneous) transition at y, otherwise *not.* Since all internal processes of a digital circuit are SSS, then $\partial y/\partial x$ is also SSS, and has an equilibrium probability, $P(\partial \mathbf{y}/\partial \mathbf{x})$. We are now ready to prove the following :

Theorem : *If the inputs $\mathbf{x_i}(t), i = 1, \ldots, n$, of a zero-delay logic module are SSS independent 0-1 stochastic processes with transition densities $D(\mathbf{x_i})$, then the densities at its outputs $\mathbf{y_j}(t), j = 1, \ldots, m$ are given by :*

$$D(\mathbf{y_j}) = \sum_{i=1}^{n} P\left(\frac{\partial \mathbf{y_j}}{\partial \mathbf{x_i}}\right) D(\mathbf{x_i}) \tag{4.2}$$

Proof : Let $\mathbf{t_{ik}}, k = 1, 2, \ldots, \mathbf{n_{x_i}}(T)$, be the sequence of transition time points of $\mathbf{x_i}(t)$ in $(\frac{-T}{2}, \frac{+T}{2}]$. Consider the sequence of random variables $\frac{\partial \mathbf{y_j}}{\partial \mathbf{x_i}}(\mathbf{t_{ik}}), k = 1, 2, \ldots, \mathbf{n_{x_i}}(T)$, defined for every input-output pair $(\mathbf{x_i}, \mathbf{y_j})$ of M'.

Since $\frac{\partial \mathbf{y_j}}{\partial \mathbf{x_i}}(t)$ is SSS and independent of $\mathbf{x_i}(t)$, then $\mathcal{P}\left\{\frac{\partial \mathbf{y_j}}{\partial \mathbf{x_i}}(\mathbf{t_{ik}}) = 1\right\} = P\left(\frac{\partial \mathbf{y_j}}{\partial \mathbf{x_i}}\right)$ is the same for any k. Therefore, this is a sequence of identically-distributed (not necessarily independent) random variables, with mean $P\left(\frac{\partial \mathbf{y_j}}{\partial \mathbf{x_i}}\right)$.

Since $\frac{\partial \mathbf{y_j}}{\partial \mathbf{x_i}}(\mathbf{t_{ik}}) = 1$ if and only if the kth transition of $\mathbf{x_i}(t)$ is transmitted to $\mathbf{y_j}(t)$, then the number of transitions of $\mathbf{y_j}(t)$ in $(\frac{-T}{2}, \frac{+T}{2}]$ is given by :

$$\mathbf{n_{y_j}}(T) = \sum_{i=1}^{n} \sum_{k=1}^{\mathbf{n_{x_i}}(T)} \frac{\partial \mathbf{y_j}}{\partial \mathbf{x_i}}(\mathbf{t_{ik}}) \tag{4.3}$$

Taking the expected value of both sides gives :

$$E\left[\mathbf{n_{y_j}}(T)\right] = \sum_{i=1}^{n} E\left[\sum_{k=1}^{\mathbf{n_{x_i}}(T)} \frac{\partial \mathbf{y_j}}{\partial \mathbf{x_i}}(\mathbf{t_{ik}})\right] \tag{4.4}$$

Since $\frac{\partial \mathbf{y_j}}{\partial \mathbf{x_i}}(t)$ is independent of $\mathbf{x_i}(t)$, and if n is some positive integer, then :

$$E\left[\frac{\partial \mathbf{y_j}}{\partial \mathbf{x_i}}(\mathbf{t_{ik}}) \;\middle|\; \mathbf{n_{x_i}}(T) = n\right] = E\left[\frac{\partial \mathbf{y_j}}{\partial \mathbf{x_i}}(\mathbf{t_{ik}})\right] = P\left(\frac{\partial \mathbf{y_j}}{\partial \mathbf{x_i}}\right) \tag{4.5}$$

Using [1], p. 183, these facts lead to :

$$E\left[\mathbf{n_{y_j}}(T)\right] = \sum_{i=1}^{n} P\left(\frac{\partial \mathbf{y_j}}{\partial \mathbf{x_i}}\right) E\left[\mathbf{n_{x_i}}(T)\right] \tag{4.6}$$

which, dividing by T, leads to (4.2). ■

If the Boolean difference is available, then evaluating $P\left(\frac{\partial \mathbf{y_j}}{\partial \mathbf{x_i}}\right)$ is no more difficult than evaluating the probability of a Boolean function knowing those of its inputs. Note that if M is a 2-input AND gate, with inputs x_1 & x_2, and output y, then $P\left(\frac{\partial \mathbf{y}}{\partial \mathbf{x_1}}\right) = P(\mathbf{x_2})$. In more complex situations, the "compose" and "xor" functions of the BDD package [10] can be used to evaluate the Boolean difference using equation (4.1). The BDD-based algorithm given in the appendix (for computing the probability of a Boolean function) can then be used to compute $P\left(\frac{\partial \mathbf{y_j}}{\partial \mathbf{x_i}}\right)$.

5. Overall Simulation Strategy

The propagation algorithm through a module presented above may be thought of as a *simulation* of the module that yields the density and probability at its outputs, given those at its inputs. Using this algorithm, one can simulate each module in the circuit, starting from the primary inputs, and compute the density and probability at every circuit node.

The assumption was made at the beginning of the previous section that the inputs to a module are *independent*. Even if this is true at the primary inputs (as assumed in section 3), it may not be true for internal nodes. Circuit topologies that include reconvergent fanout and feedback will cause internal nodes to be correlated, and destroy the independence property. This problem is central to any circuit analysis based on a statistical representation of signals, and can usually be taken care of by using heuristics that trade-off accuracy for speed [4-8].

Based on our experience with the propagation of probability waveforms [11], we have found that, if the modules are large enough so that tightly coupled nodes (such as in latches or small cells) are kept inside the same module, then the coupling outside the modules is sufficiently low to justify an independence assumption. Of course, one can model the whole circuit as a single module, but then performance would be sacrificed because the BDD can become too large. The next section will investigate this speed-accuracy trade-off.

6. Implementation and Results

We have implemented the above ideas in a prototype *density simulator* that takes a description of a circuit in terms of its boolean modules and gives the transition density at every node. The simulator requires values for the transition density and equilibrium probability at the primary inputs. The units of density are arbitrary; they may be "transitions per second" or "million transitions per second," as long as one is consistent. We present below the results of two test cases that we have used to investigate the *validity* of the technique and the speed-accuracy *trade-off* mentioned above. All execution times are for a SUN Sparcstation 1.

To study the validity of this approach, we have devised a test by which randomly generated sequences of inter-transition times are fed to the circuit primary inputs and propagated into the circuit (by logic simulation based on the BDD). From these, we estimate the number of transitions per unit time. For a large number of input transitions, this number should converge to the transition density, according to equation (3.2). We also estimate the percentage of time that the signal spends in the high state and check if that converges to the equilibrium probability, in accordance with (3.1). As a test case, we constructed a logic module with 8 inputs $(A, B, \ldots, H)$ and one output (Z), that implements the boolean function : $Z = ABFD + CFD + ABHD + CHD + ABFG + CFG + ABHG + CHG + AFE + ADE + CFE + CDE$. The results, showing the correct convergent behavior at the output Z, are shown in Fig. 3. The run took only 0.38 seconds, was based on input values of $P = 0.5$ and $D = 2.0$, and resulted in $P(Z) = 0.476562$ and $D(Z) = 3.71875$.

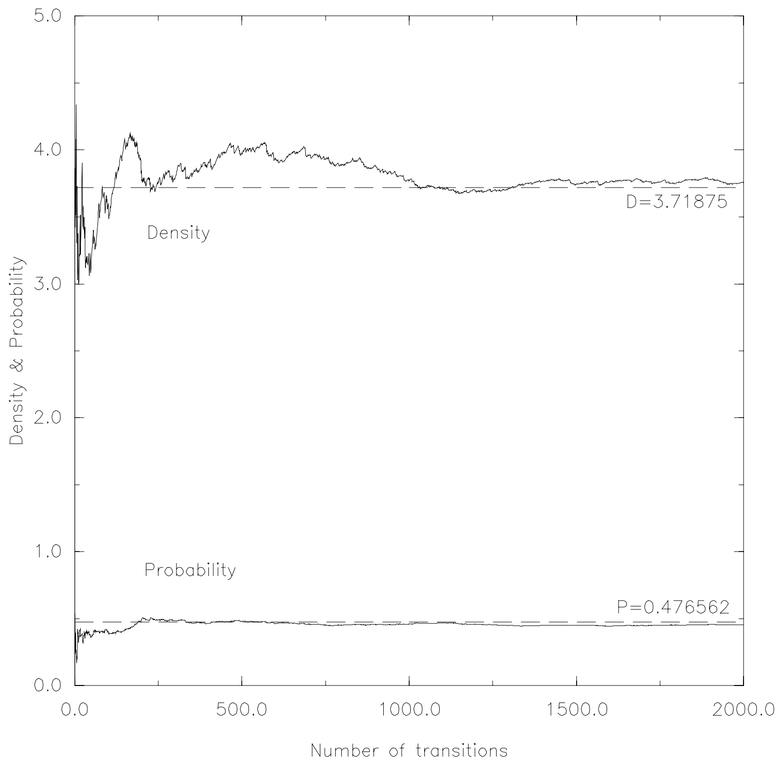

Figure 3. Density and probability convergence plot.

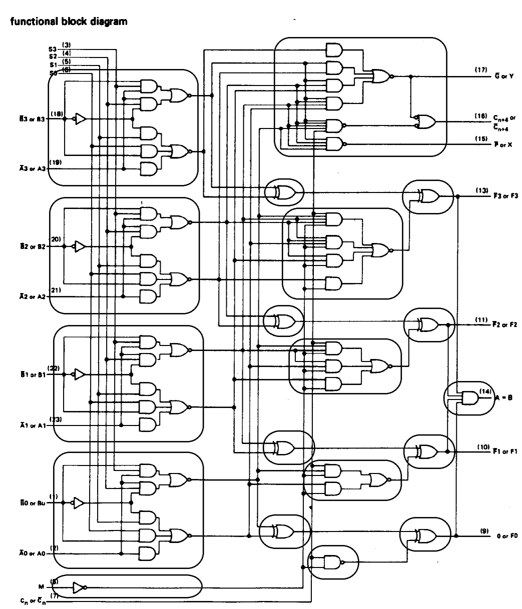

Figure 4. ALU / function generator example.

To investigate the speed-accuracy trade-off, we considered an ALU circuit (SN54181) from the TI TTL Data Book which has 75 logic gates, shown in Fig. 4. We simulated this circuit in two ways : (1) using a single module to model the whole circuit (this took 15 seconds), and (2) using the 19 modules indicated in the figure (this took 1.0 seconds). The density results are compared in Fig. 5 for all nodes that are module outputs. By "accurate density," we mean the density from the first run, while "approximate density" refers to that of the second run. The error resulting from the independence assumption in the second run is reflected in the deviation from the dashed diagonal line. In this case there was a "$< 20\%$" loss in accuracy for a gain of 15X in speed.

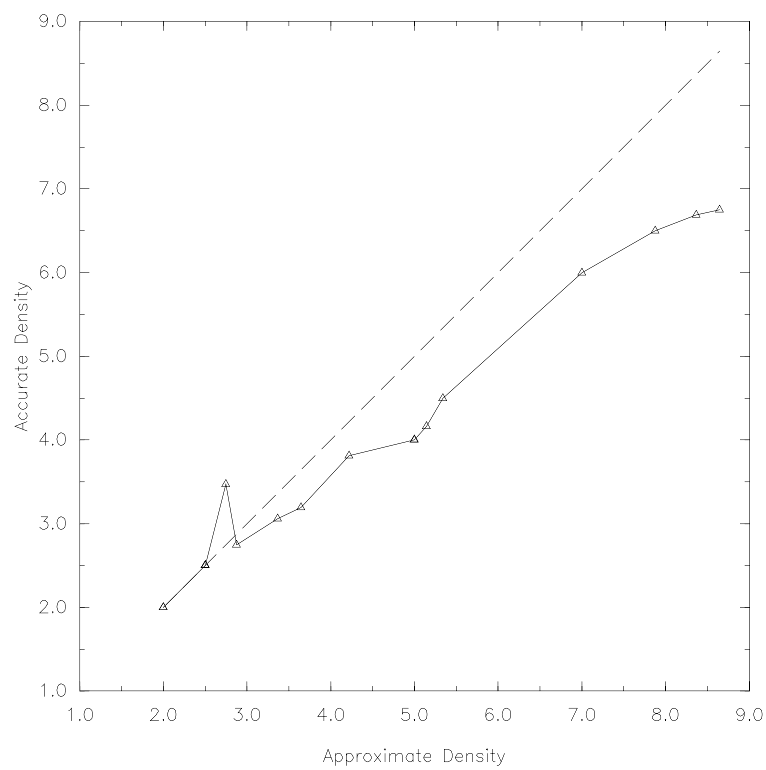

Figure 5. Speed-accuracy trade-off in the ALU.

7. Summary and Conclusions

To summarize, we have observed that a common thread that runs through most causes of run-time failure is the extent of circuit *activity*, i.e., the rate at which its nodes are switching. Based on a stochastic model of logic signals, we have defined a new measure of circuit activity, called the *transition density*. We have also presented an algorithm to propagate the density from the primary inputs to internal nodes. This algorithm may be thought of as a *simulation* of the circuit and has been implemented in a prototype *density simulator*. We have presented some results of this implementation that verify the theoretical results and establish the feasibility and efficiency of the approach.

Appendix : Using BDDs for Probability Propagation

We will briefly review the concept of a Binary Decision Diagram [9, 10] (BDD) and then present a new application for BDDs as tools for computing the probability of a boolean function.

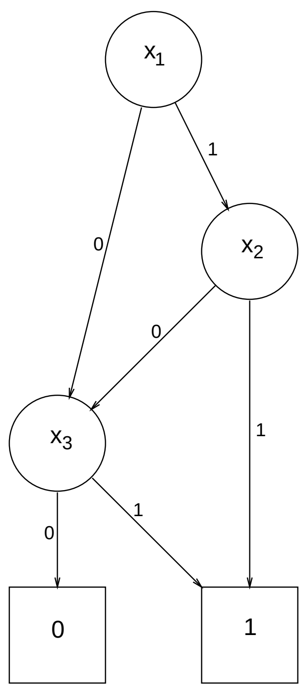

Figure 6. Example BDD representation.

Consider the boolean function $y = x_1 \cdot x_2 + x_3$, which can be represented by the BDD shown in Fig. 6. The boolean variables x_i are *ordered*, and each *level* in the BDD corresponds to a single variable. Each level may contain one or more BDD nodes at which one can branch in one of two directions, depending on the value of the relevant variable. For example, suppose that $x_1 = 1$, $x_2 = 0$, and $x_3 = 1$. To evaluate y, we start at the top node, branch to the right since $x_1 = 1$, then branch to the left since $x_2 = 0$, and finally branch to the right since $x_3 = 1$ to reach the terminal node "1". Thus the corresponding value of y is 1.

The importance of the BDD representation is that it is *canonical*, i.e., that it does not depend on the Boolean expression used to express the function. In our case, if the function was expressed as $y = x_3 + x_1 \cdot (x_2 + x_3)$ (an equivalent representation), it would have the same BDD. BDDs have been found to be an efficient representation for manipulating Boolean functions, both in terms of memory and execution time. For example, checking if a boolean function is satisfiable can be done in time that is linear in the size of the BDD.

Let $y = f(x_1, \ldots, x_n)$ be a Boolean function. We will show that, given signal probabilities for the variables x_i, and that these variables are independent (random variables), then the probability of the function f can be obtained in *linear time* (in the size of its BDD representation). By Shannon's expansion :

$$y = x_1 f_{x_1} + \overline{x_1} f_{\overline{x_1}} \tag{A.1}$$

where $f_{x_1} = f(1, x_2, \ldots, x_n)$ and $f_{\overline{x_1}} = f(0, x_2, \ldots, x_n)$ are the *cofactors* of f with respect to x_1. Since $x_1 \overline{x_1} = 0$, then :

$$P(y) = P(x_1 f_{x_1}) + P(\overline{x_1} f_{\overline{x_1}}) \tag{A.2}$$

Since the cofactors of x_i do not depend on x_i, and since all variables are independent, then :

$$P(y) = P(x_1)P(f_{x_1}) + P(\overline{x_1})P(f_{\overline{x_1}}) \tag{A.3}$$

This equation shows how the BDD is to be used to evaluate $P(y)$. The two nodes that are descendants of y in the BDD correspond to the cofactors of f. The probability of the cofactors can then be expressed in the same way, in terms of their descendants. Thus a depth-first-traversal of the BDD, with a post-order evaluation of $P(\cdot)$ at every node is all that is required. We have implemented this using the "scan" function of the BDD package [10].

References

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

[2] E. Parzen, *Stochastic Processes*, San Francisco, CA: Holden-Day Inc., 1962.

[3] D. R. Cox and H. D. Miller, *The Theory of Stochastic Processes*, New York: John Wiley & Sons Inc., 1968.

[4] K. P. Parker and E. J. McCluskey, "Probabilistic treatment of general combinational networks," *IEEE Trans. Computers*, pp. 668-670, June 1975.

[5] S. C. Seth, L. Pan, and V. D. Agrawal, "PREDICT - probabilistic estimation of digital circuit testability," *IEEE 15th Annual International Symposium on Fault-Tolerant Computing*, Ann Arbor, MI, pp. 220-225, June 1985.

[6] J. Savir, G. S. Ditlow, and P. H. Bardell, "Random pattern testability," *IEEE Trans. Computers*, pp. 79–90, January 1984.

[7] G. Markowsky, "Bounding signal probabilities in combinational circuits," *IEEE Trans. Computers*, pp. 1247–1251, October 1987.

[8] S. Ercolani, M. Favalli, M. Damiani, P. Olivo, and B. Riccó, "Estimate of signal probability in combinational logic networks," *1989 IEEE European Test Conference*, pp. 132–138, 1989.

[9] R. E. Bryant, "Graph-based algorithms for Boolean function manipulation," *IEEE Trans. Computer-Aided Design*, pp. 677–691, August 1986.

[10] K. S. Brace, R. L. Rudell, and R. E. Bryant, "Efficient implementation of a BDD package," *27th ACM/IEEE Design Automation Conference*, pp. 40–45, June 1990.

[11] F. N. Najm, R. Burch, P. Yang, and I. N. Hajj, "Probabilistic simulation for reliability analysis of CMOS VLSI circuits," *IEEE Trans. Computer-Aided Design*, pp. 439–450, April 1990 (Errata in July 1990).