RL-TR-96-77
Final Technical Report
June 1996

PROBABILISTIC TECHNIQUES FOR RELIABILITY ANALYSIS OF VLSI CIRCUITS

University of Illinois

I.N. Hajj, F.N. Najm, P-C Li, S. Goel, and V. Saxena

Rome Laboratory
Air Force Materiel Command
Rome, New York

RL-TR- 96-77 has been reviewed and is approved for publication.

APPROVED: [illegible]

MARTIN J. WALTER
Project Engineer

FOR THE COMMANDER: John J. Bart

JOHN J. BART
Chief Scientist, Reliability Sciences
Electromagnetics & Reliability Directorate

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE June 1996	3. REPORT TYPE AND DATES COVERED Final Aug 94 - Nov 95

4. TITLE AND SUBTITLE PROBABILISTIC TECHNIQUES FOR RELIABILITY ANALYSIS OF VLSI CIRCUITS	5. FUNDING NUMBERS C - F30602-94-C-0169 PE - 62702F PR - 2338 TA - 01 WU - PP
6. AUTHOR(S) I.N. Hajj, F.N. Najm, P-C Li, S. Goel, and V. Saxena	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) University of Illinois Coordinated Science Laboratory 801 S. Wright Street Champaign IL 61820	8. PERFORMING ORGANIZATION REPORT NUMBER N/A
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Rome Laboratory/ERDD 525 Brooks Rd Rome NY 13441-4505	10. SPONSORING/MONITORING AGENCY REPORT NUMBER RL-TR-96-77

11. SUPPLEMENTARY NOTES

Rome Laboratory Project Engineer: Martin J. Walter/ERDD/(315) 330-4102

12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution unlimited.	12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

This report summarizes work done on the reliability analysis and design of VLSI circuits. Many reliability issues, such as electromigration in metal lines and hot-carrier effects (HCE) induced degradation in devices, are related to the statistics of the average current flow in the circuit and devices. New techniques for current estimation in synchronous sequential circuits have been developed and implemented. Given primary input statistics, mixed statistical Monte Carlo and probabilistic methods are applied in a mixed-mode simulation approach to estimate average currents drawn by each gate or subcircuit in the design. The results are then used for reliability estimation. In the area of design for reliability, a computer-aided design system for VLSI circuit hot-carrier reliability estimation and design was developed. The system first stimulates a circuit to determine the critical transistors that are most susceptible to hot-carrier effects based on their switching frequency and current flow through them. The system then estimates the impact of HCE on circuit delay and employs a combination of design modification and optimization strategies, including signal line reordering and gate sizing along critical paths, to eliminate or reduce HCE-induced degradation on circuit performance.

14. SUBJECT TERMS Reliability estimation, Design for reliability, Current estimation, Synchronous sequential circuit, Hot-carrier effects, Time/area/reliability optimization	15. NUMBER OF PAGES 72
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UL

Contents

1 Introduction

The objectives of this research are to advance the state of the art in computer-aided techniques for assessing the reliability of VLSI microelectronic circuit designs and to detect if and where the design does not meet reliability specifications so that the design can be modified to improve its reliability.

The growing application of automation to reduce the time from system concept to manufacturing and delivery of VLSI circuits and systems that meet reliability and performance requirements has increased the need to include reliability objectives as part of the design process. These reliability considerations include electromigration, hot-carrier effects, voltage drop, oxide breakdown, among others. Existing computer-aided design systems tend to ignore reliability issues during the design process and to rely on performing reliability assurance in a post-fabrication phase. Such an approach could result in repeating the design and fabrication cycle until reliability measures are met, which increases cost and delays the time to market of the final product. Design for reliability assures that a reliable product will be produced in a single pass of design, fabrication, and evaluation.

Many reliability issues, such as electromigration in metal lines and hot-carrier-induced degradation in devices, are related to the statistics of the current flow in the circuit and devices during the operation of the design under all possible input signals. Simulating a design under all possible input patterns is extremely expensive, if not impossible, since the number of input patterns is exponential in the number of inputs. On the other hand, simulating a design in a deterministic way using a small set of "typical" inputs may lead to inaccurate and misleading results. A systematic and proven approach to estimate circuit reliability, including effects on circuit performance, in reasonable time is needed.

In previous work, we considered current estimation in CMOS digital combinational circuits. However, digital designs are sequential. In this work, we consider synchronous sequential CMOS digital circuits, which have a popular and well-structured design that consists of combinational logic blocks separated by latches. We apply mixed statistical Monte Carlo and probabilistic methods in a mixed-mode simulation approach. The Monte Carlo method is used to simulate the design at high functional-level description where a large number of inputs, based on system application and designers' specifications, can be simulated quickly to obtain signal statistics within the design, in particular, at outputs of latches. Because latches are triggered by a clock, the switching time is known. At this functional-level simulation step, the delays within the combinational blocks need not be specified accurately. The signal statistics obtained at the outputs of the latches are then used as inputs to perform probabilistic simulation of the combinational blocks where gate and transistor-level models, with accurate circuit timing and device characteristics, are used to obtain more accurate and detailed signal and current waveform statistics. In contrast to the Monte Carlo method, in probabilistic simulation, the input signals are represented by probabilistic waveforms and the circuit is simulated to obtain signal and current waveform statistics within the design in a single run rather than in multiple runs.

In the area of design for reliability, we have developed a computer-aided design system for CMOS VLSI circuit hot-carrier reliability estimation and redesign. The system first simulates a circuit to determine the critical transistors that are susceptible to hot-carrier effects based on their switching frequencies and the average currents that flow through them. It then estimates the impact of HCE on circuit delay and employs a combination of design modification strategies to eliminate HCE-induced performance degradation.

Figure 1.1 shows a block diagram of the CAD system that we are developing to implement and test our methodologies and algorithms. Given a design layout, which could be specified hierarchically, program iCHARM extracts the SPICE-file from it, including interconnect parasitics, and sorts out the CIF subfiles of the power and ground busses, with bus contacts labeled to match the corresponding node connections in the extracted SPICE file. The extracted SPICE-file has the same hierarchical structure as the original layout file.

The SPICE-file forms the input to the simulators, iPROBE-c, iPROBE-d, and iMAX. The SPICE-file can also be used as the input file to our other simulators to perform timing and fault simulations using input test vector sets; these other simulators are not indicated in Fig. 1.1. FSM is a sequential circuit statistical simulator that computes the signal statistics at the outputs of the latches. The input to FSM is RTL or functional specification of the design and primary input statistics. The output of FSM forms the input to the probabilistic simulators. iPROBE-c is a probabilistic simulator, which computes the average and variance current waveforms drawn by the combinational circuits at declared contact points. iPROBE-d estimates the average relative damage due to hot-carrier effects within the circuit devices. iMAX computes an upper bound current waveform envelope at contact points. JET is a bus extractor dedicated to extracting the RC models of the bus. The outputs of JET, iMAX and iPROBE-c form the inputs to a bus analyzer and optimizer that recommends changes to the bus line widths, if necessary, in order to meet both electromigration and voltage-drop constraints. The outputs from iPROBE-c and iPROBE-d can be used as part of a multiobjective function in a design optimization program, which we have developed, that aims at optimizing the design with respect to area, timing, and reliability measures.

In Chapter 2, we present our method for current estimation in sequential circuits. In Chapter 3, we describe our design for a hot-carrier reliability system. The system is an illustration on how reliability estimation measures can be incorporated into VLSI circuit design automation. In Chapter 4, we give conclusions and directions for future research.

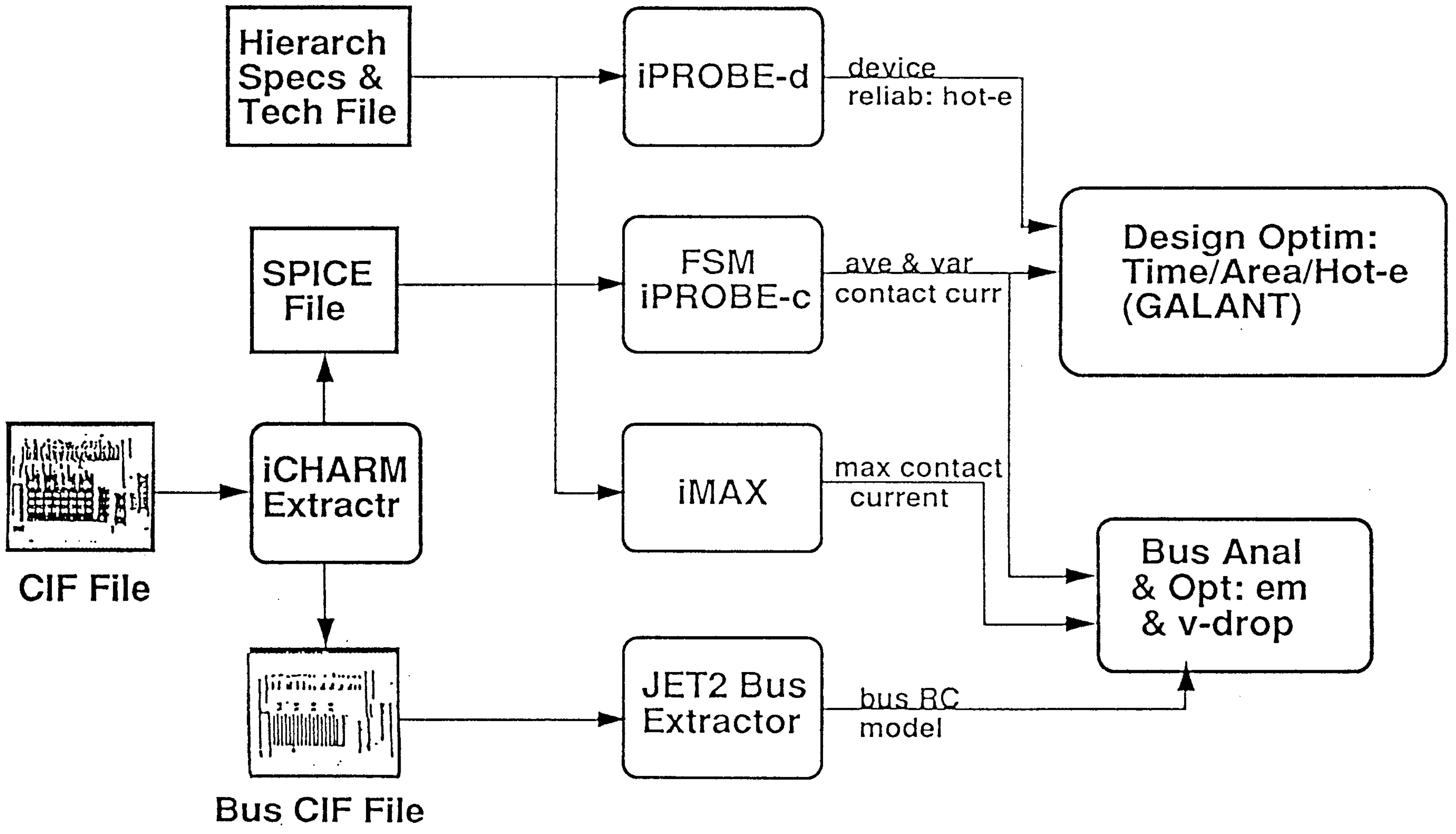

Figure 1.1. Block diagram of reliability analysis and design system.

2 Current Estimation in Sequential Circuits

2.1 Introduction

A basic and necessary step in design for reliability is an estimation step in which quantitative estimates of the reliability measures of the design under consideration are obtained. The validity and the integrity of the final design greatly depend on the accuracy of this estimation step. Traditionally, in VLSI circuit design, the emphasis so far has been on timing estimation and accurate propagation of voltage signal waveforms so that timing constraints in the design could be met. In reliability estimation, it is often necessary to obtain estimates of the current waveforms drawn by the circuit, in addition to voltage waveform propagation. For example, electromigration and average power estimation are related to the average current drawn by the gates or modules of the design, and hot-carrier effects are caused by currents flowing in the transistors during switching operation. Accurate current estimation in individual gates or subcircuits, however, requires accurate voltage waveform propagation through the circuit. A straightforward approach to obtaining average waveforms is to simulate the design over all possible inputs, which is extremely expensive or practically impossible for designs with a moderate to large number of input pins, because the number of possible input signal patterns to a digital design is exponential in the number of input nodes. Instead, one could use statistical Monte Carlo techniques in which the design is simulated over a sample of the input space, until meaningful current and voltage statistics are obtained with a certain degree of confidence. Since the size of the sample input space could be large, detailed circuit-level simulation over every input in the sample could still be prohibitively expensive.

In our previous work, we concentrated our efforts on current estimation in digital CMOS *combinational* circuits. Probabilistic techniques are applied to estimate the average current flow in each gate under all possible input signals. Given the statistics of the signals at the primary inputs, the signal statistics are propagated from the inputs of each gate to its output using probabilistic techniques, taking the delay of the gate into consideration. At the same time, the average current drawn by the gate is also obtained using the probabilistic analysis. The analysis follows an event-driven scheme starting from the primary inputs to the primary outputs of the combinational circuit. Signal correlations at nodes within the design are computed using first-order approximation techniques. These techniques have been implemented in programs, CREST [1], iProbe-c [4], and iProbe-d [5], and published in [1–7].

Digital designs, however, are sequential rather than purely combinational; during the past year we have expanded our work to include sequential circuit designs. We have targeted synchronous sequential circuits, which are the most commonly used design styles. These circuits consist of latches driven by a common clock and combinational logic blocks whose inputs are latch outputs and whose outputs are latch inputs. For this type of circuits, the problem could be decoupled into two parts. One is the analysis of the latches that are triggered by the clock causing some of them to switch with the clock, depending on their states and inputs. The second part is the analysis of the combinational blocks. Even though the inputs to a combinational block are updated by latches in synchrony with the clock, the internal nodes of the block may switch at different times within the clock cycle and may also make several transitions before settling to their steady-state values for that clock period. These additional transitions are called hazards or glitches, and although unplanned by designers, they are not necessarily design errors; but they do contribute to the average current drawn by the gates, and thus cannot be completely ignored.

Based on the above observations, we have developed the following approach to handle sequential circuits. We first use fast high-level functional Monte Carlo statistical simulation techniques to obtain meaningful signal probabilities at outputs of flip-flops to be used in the lower-level more detailed combinational circuit simulation. The idea here is that high-level simulation could be done very fast and one can afford to perform a large number of simulations. Since latches switch at known time points determined by the clock, the computation of accurate delays in the combinational blocks is not necessary; functional simulation should be sufficient. We thus 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. After the signal statistics at the outputs of the latches are obtained, the circuit can then be analyzed in parts. The latches transition in synchrony with the clock; the signal probabilities at their outputs, which implicitly are functions of the signal probabilities at their inputs and their states, are sufficient to estimate the average current each latch draws. The signal probabilities at the output of the latches provide the input to the combinational blocks which are then analyzed using more detailed probabilistic simulation that can take into account gate delays.

To explain the relationship between the switching frequency at the output of a gate and the average current it draws, consider the CMOS inverter circuit shown in Fig. 2.1. The current flowing through the inverter consists of three components: (1) dynamic current which charges and discharges the load capacitance, C_L; (2) short-circuit current, which flows from V_{DD} to ground through the transistors, and (3) leakage current. Both dynamic current and short-circuit current flow only during gate switching. Average dynamic current can be shown to be equal to $\frac{1}{2}C_L V_{DD} f$, where f is the switching frequency of the gate. Thus the dynamic current depends on f as well as on V_{DD} and the load capacitance C_L, and is independent of the transistor characteristics. The short-circuit current, on the other hand, flows from V_{DD} to ground through the transistors during switching, and depends on the transistor characteristics, input slew rate, V_{DD}, and C_L. It should be noted that signal delay through the gate also depends on transistor characteristics, input slew rate, V_{DD} and C_L. The leakage current is independent of gate switching and is process related; it is usually very small compared to the dynamic and short-circuit currents, but could be a reliability problem when considering an entire chip, and thus has to be addressed at the process manufacturing level.

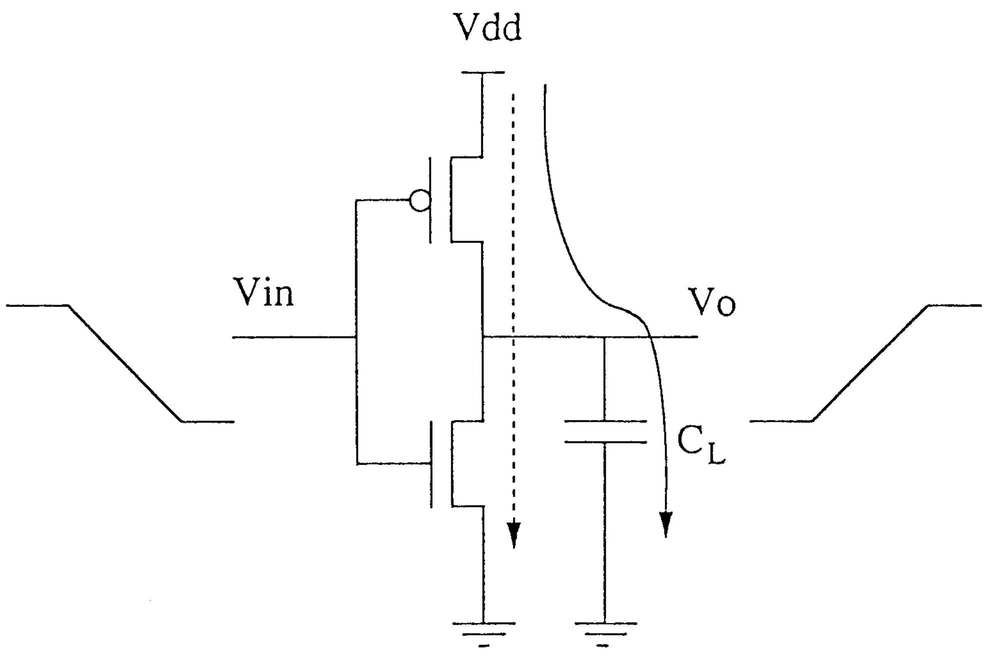

Figure 2.1. Current flow in CMOS gate during switching.

In our analysis we first estimate the switching frequency and the probability of the signal being high and low at signal lines in the design based on primary input statistics and circuit connectivity and functionality. The switching frequency, together with transistor characteristics, input slew rate, and loading, are then used to estimate the average current drawn by each gate and the gate delay. Hot-carrier effects are estimated by monitoring the time during which the transistors are in saturation during switching [3], and oxide breakdown lifetime is estimated based on the probability a transistor gate input signal is high [5].

In the next subsection, we give some preliminary explanation and definition and review other recently proposed approaches. In Section 2.3, we formulate the problem in more detail, and in Section 2.4, we present our approach. In Section 2.5, we give experimental results, and then conclude with some discussion in Section 2.6.

2.2 Preliminaries

Let $u_1, u_2, \ldots, u_m$ be the primary input nodes of a sequential logic circuit, as shown in Fig. 2.2, and let $x_1, x_2, \ldots, x_n$ be the *present state* lines. For simplicity, we assume 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).

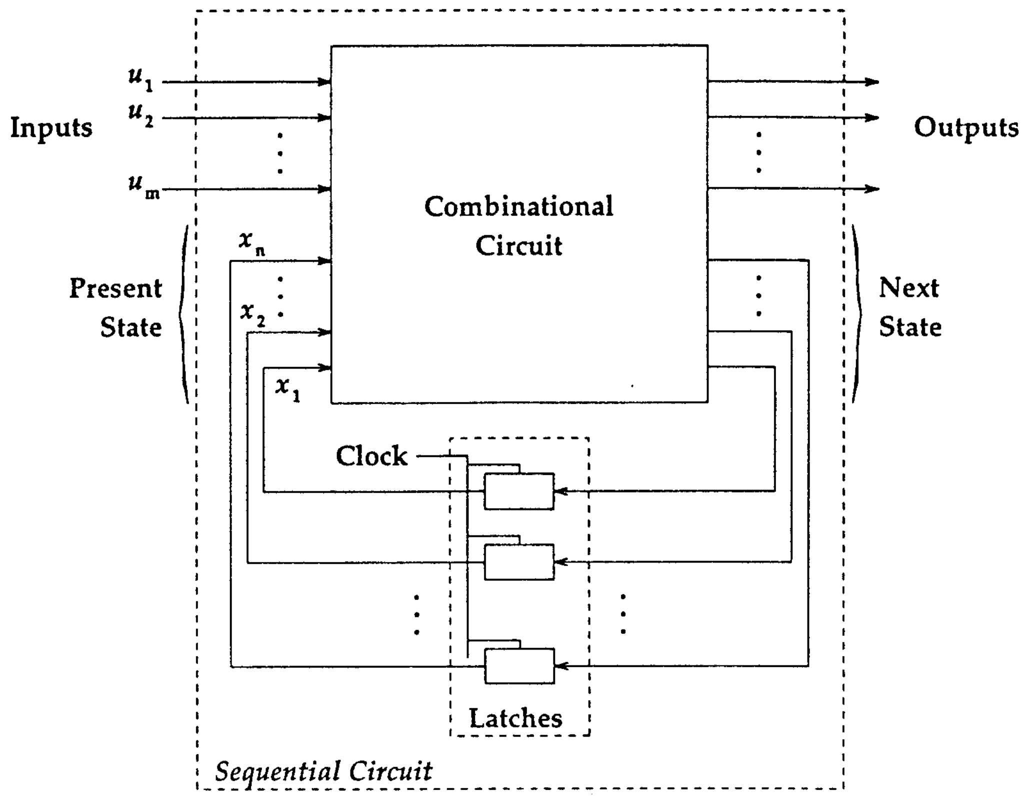

Figure 2.2. An FSM model of a sequential logic circuit.

Most existing estimation techniques handle only combinational circuits and require information on the circuit input statistics. To allow extension to sequential circuits, it is sufficient to compute the statistics of the latch outputs. Other existing techniques would then be applied to compute the currents drawn by the combinational block.

We briefly survey the few recently proposed techniques for estimating state-line statistics 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 [8–11] make the simplifying assumption that the FSM is *Markov* [12], so that its future is independent of its past once its present state is specified.

The approaches in [8] and [9] compute the transition probabilities on the present state lines using the Chapman-Kolmogorov equations [12]. Solving these equations is computationally too expensive and the largest test case presented contains less than 30 latches.

Better solutions are offered by two recent papers [10, 11], 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 $P_{p.s.} = [p_{x_1} \ \cdots \ p_{x_n}]$ 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 [10, 11] 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 [10] and the Picard-Peano iteration method in [11].

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 [10], this correction is done by accounting for m-wise correlations between state bits when computing their probabilities, which requires 2^m additional gates and can become very expensive. The approach in [11] is to *unroll* the combinational logic block k times. This is less expensive than [10], and the authors observe that with $k = 3$ or 4, 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 is in the autocovariance of the logic signals, which is 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.

2.3 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 is non-decomposable. Therefore, it is sufficient to study a single 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. To take into account the effect of large sets of inputs, one is typically interested in the average current drawn by the circuit 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. It can also be 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 [13, 14]:

$$t_x(k) = \begin{cases} 1, & \text{if } x(k) \neq x(k-1); \\ 0 & \text{otherwise.} \end{cases} \tag{2.1}$$

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 a **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. 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 [13, 14]:

$$\mathcal{P}\{\mathbf{x}(k) = 1\} = P(x) \tag{2.2}$$

This result holds for *any* logic signal. If we 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 proceeding, we will have to make one mild assumption related to the covariance of the process $\mathbf{X}(k)$:

Assumption 1. *The state of the machine at time k becomes independent of its initial state at time* 0 *as* $k \to \infty$.

This assumption is not restrictive 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., 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 [10, 11].

2.4 Computing State Line Probabilities

We propose to obtain the state line probabilities by performing Monte Carlo logic simulation of the design using a 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 obtain 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.

2.4.1 Convergence

Suppose the FSM is known to be in some state X_0 at time 0. Using (2.2), and Assumption 1, we have for any state signal x_i:

$$\lim_{k\to\infty} \mathcal{P}\{\mathrm{x_i}(k) = 1 \mid \mathbf{X}(0) = X_0\} = \lim_{k\to\infty} \mathcal{P}\{\mathrm{x_i}(k) = 1\} = P(x_i) \tag{2.3}$$

For brevity, we denote the conditional probability by

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

Our method consists of estimating $P_k(x_i \mid X_0)$ for increasing values of k until convergence (according to (2.3)) 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 [15], we can predict how many runs to perform in order to achieve some user-specified error-tolerance (ϵ) and confidence (α) levels. Specifically, it can be shown [16] that if we want $(1 - \alpha) \times 100\%$ confidence that

$$\left| p_i^{(N)}(k) - P_k(x_i \mid X_0) \right| < \epsilon \tag{2.4}$$

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

$$N_1 = \frac{z_{\alpha/2}}{2\epsilon}, \quad N_2 = \frac{z_{\alpha/2}\sqrt{2\epsilon + 0.1} + \sqrt{(\epsilon + 0.1)z_{\alpha/2}^2 + 3\epsilon}}{2\epsilon}, \quad \text{and} \quad N_3 = \frac{\sqrt{63} + z_{\alpha/2}}{2\sqrt{\epsilon}}$$

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 **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 ϵ and α, 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 the node has converged. The choice of three time instants is somewhat arbitrary and can be changed by the user. When all nodes have converged, the simulation is terminated and the average of $P_k(x_i \mid X_0)$ and $P_k(x_i \mid X_1)$ is reported as the signal probability $P(x_i)$, for each x_i.

2.4.2 Input generation

In view of assumption 1, one requirement on the applied input sequence $U(k)$ is that it not be periodic. Another condition, required for the estimation (2.4) 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 microcode from a fixed set of instructions, then every sequence $U^{(j)}(k)$ may be a piece of some microcode 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 are available, one can bias the random generation process to reproduce these correlations.

In more general situations, in which 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 the equations derived in [13], 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 [17], the total average current 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 [18], 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 explained earlier.

2.5 Experimental Results

This technique was implemented in a prototype C program, FSM, 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 [19].

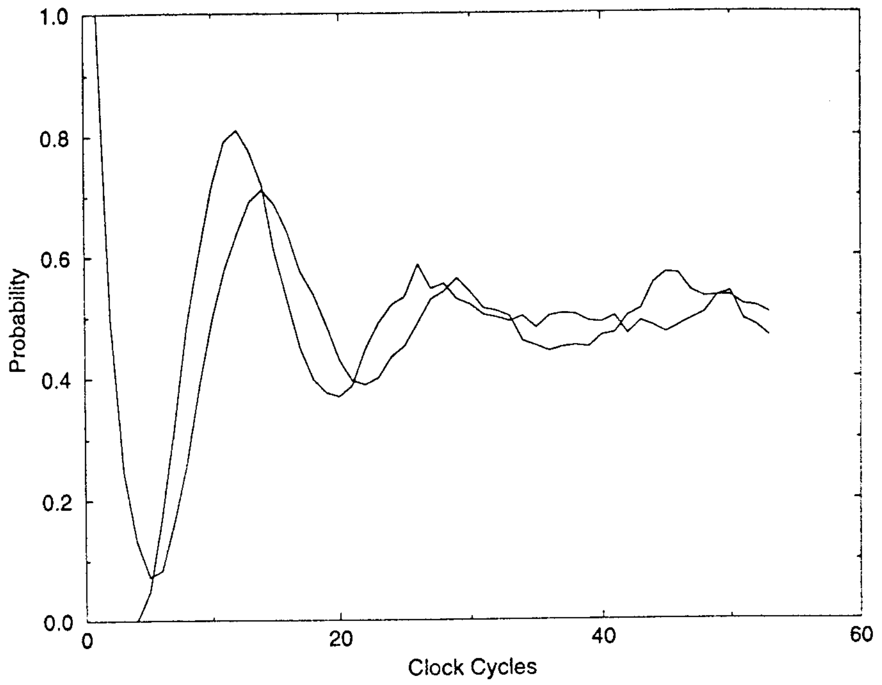

Figure 2.3. Convergence of probability for s838.1, node X.3.

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. 2.3. 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.

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, were only performed on the limited set of benchmark circuits given in Table 2.1. 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 2.4 shows the resulting error histogram for all the latch outputs from the circuits in Table 2.1. Note that all of the nodes have errors well within the specified 0.05 error bounds.

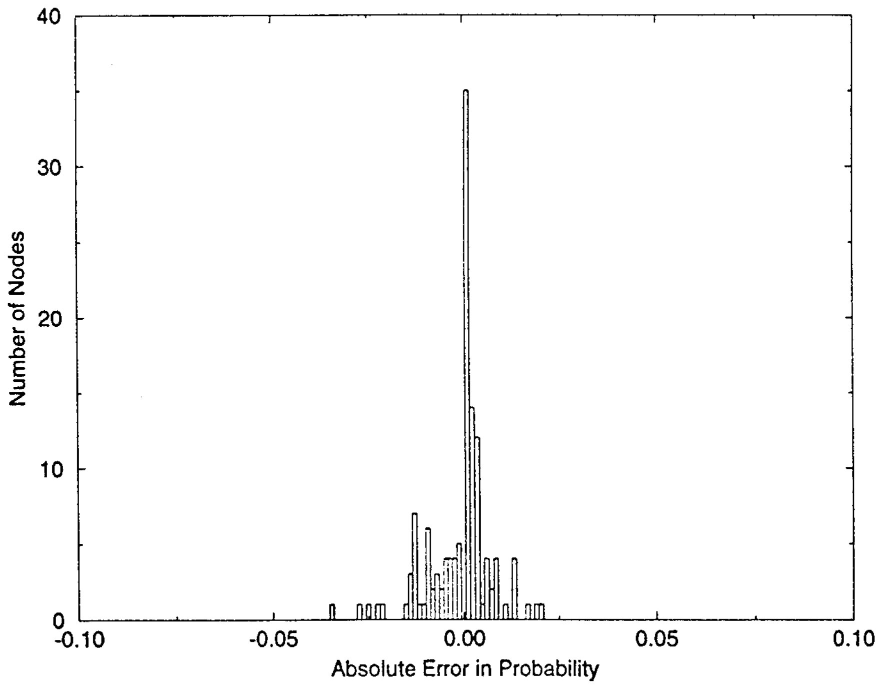

Figure 2.4. Latch probability error histogram.

Circuit	#inputs	#latches	#gates
s1196	14	18	529
s1238	14	18	508
s713	35	19	393
s1423	17	74	657

Table 2.1. A few ISCAS-89 circuits.

We also monitored the speed of this technique and report some results in Table 2.2. All simulation runs were done on a SUN Sparc10 machine. These circuits are larger (especially in terms of flip-flop count) than the largest ISCAS-89 circuits tested in previous methods [9–11]. Furthermore, for those circuits in the table that were also tested in [9–11], this technique works much faster. Since our method does not use BDDs to compute probabilities, there are no memory problems with running large circuits. As expected, however, the execution time increases on the larger circuits. One reason for this is that the larger combinational block makes the estimation of probability for a given clock cycle more time-consuming. Another reason is that the larger flip-flop count means that the machine state space is much larger and the probability of individual machine states becomes much smaller. As a result, many more cycles may be required to achieve equilibrium.

2.6 Summary

Most existing current estimation techniques are limited to combinational circuits, while all practical circuit designs are sequential. We have presented a new technique for current 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 simu-

Circuit	#inputs	#latches	#gates	cpu sec
s1238	14	18	508	2.90
s1196	14	18	529	1.85
s713	35	19	93	2.95
s838.1	34	32	446	2.24
s1423	17	74	657	7.84
s5378	35	179	2779	78.03
s9234.1	36	211	5597	431.13
s15850.1	38	534	9796	655.34
s13207.1	62	638	7951	669.63
s38584	12	1452	19253	1585.02

Table 2.2. Execution Time on a SUN Sparc10.

lation, that allow efficient current 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 sequential circuits. We verified that the accuracy specified by the user is indeed achieved. While the execution time is better than for existing techniques, we also observe that it increases for the larger circuits, due to the increased simulation cost per cycle and to the lower state probabilities in large machines.

3 VLSI Circuit Design for Hot-Carrier Reliability

3.1 Introduction

It is widely acknowledged that hot-carrier effects (HCE) are becoming a reliability concern when CMOS technology is pushed into the submicron region. HCE in an MOS transistor are caused by the processes of charge trapping in the oxide and/or interface trap generation at the Si/SiO_2 interface. These processes result in a shift in the threshold voltage, V_t, as well as degradation in the transconductance, gm, and electron mobility in the channel, μ_n. Most work on HCE has thus far concentrated on modeling the phenomenon in small test structures. This report, on the other hand, deals with the estimation of the expected long-term HCE impact on the performance of large designs and considers redesign strategies to minimize such an impact.

For current semiconductor process technologies, with effective channel lengths longer than a quarter micron, hot-carrier-induced degradation is more severe in nMOS transistors than in pMOS [20]. Based on the reliability qualification reported by various semiconductor companies, in the upper-quarter micron channel length region, the device lifetime for a pMOS is still several orders of magnitude higher than that for an nMOS. Therefore, in this report we focus our attention mostly on n-channel transistors. However, our approach is not restricted to the nMOS and can be easily applied to include the pMOS with appropriate degradation models. It has been shown that HCE in an nMOS transistor in a circuit is dominated by interface trap generation, which occurs mostly when the transistor is operating in or near the saturation region [21, 22, 23]; HCE-induced degradation depends on the transistor's substrate current I_{sub}, which is a function of the drain current I_{DS} and the drain-to-source voltage V_{DS}. The actual physical damage due to HCE can be related to the interface state generation N_{it}. The relation between N_{it}, I_{sub}, I_{DS}, and V_{DS} is given in the next section and can be found in [23]. It can also be shown that either I_{sub} or $\frac{I_{sub}^m}{I_{DS}^{m-1}}$, where $m \simeq 3$, can be used to determine *relative* damage in nMOS transistors [22].

HCE in a transistor are cumulative in nature and depend on the operation of the transistor over time within a given design. Different transistors within a design degrade at different rates, depending on their operating conditions; HCE-induced degradation in circuit performance, such as degradation in gate delay, increases at different rates in different gates, depending on the gate switching activity. We thus consider two aspects of HCE-induced degradation. One is the degradation in individual transistors within a circuit, and the other is the effect of such degradation on circuit performance and timing. We then propose redesign strategies to reduce degradation effects, again at the two levels: one at the transistor and subcircuit level, and the other at the overall circuit timing performance level. At the subcircuit level, we consider a number of new as well as previously proposed redesign techniques and compare their advantages and disadvantages. At the circuit level, we consider the incorporation of estimated HCE-induced subcircuit delay degradation into an area/timing/reliability optimization system to insure that a given design will function within its originally specified timing specifications during the expected operating lifetime of the design. Although the life cycles of new chip designs could be short, many of the systems in which these chips are embedded have to function properly and reliably for a number of years, and long-term reliability issues have to be incorporated into the design cycle, before manufacturing.

This chapter is organized as follows: In Section 3.2 we briefly review the simulation of HCE-induced degradation in transistors. In Section 3.3 we employ a two-transistor model to characterize HCE damage and its effect on circuit delay. In Section 3.4 we describe a number of redesign techniques for improving subcircuit reliability and compare their advantages and disadvantages. In Section 3.5 we present methods for including reliability as part of an area/timing/reliability optimization system. In Section 3.6 we present the simulation results and in Section 3.7 the conclusions.

3.2 HCE-degradation Estimation

Since HCE are a long-term cumulative effects, exhaustive or at least a large number of simulations using different input excitations are needed to obtain meaningful average effects in individual transistors. The estimation of the average effects can be done using either probabilistic techniques [7], or Monte Carlo-based statistical techniques, with appropriate convergence criteria [18]. In the probabilistic simulation approach which we use here, statistical descriptions of primary input signals are assumed to be given. These statistics are specified in the form of probabilities of signals being high and the probabilities of switching at specified time points. The probabilistic timing simulation approach propagates these probabilities in the design to obtain the corresponding statistical descriptions of voltage and current waveforms inside the circuit. Using the voltage and current waveform statistics and the interface trap model [23], the following equations are used to calculate the average substrate current I_{sub}, the average bond-breaking current $I_{BB} = \frac{I^m_{sub}}{I^{m-1}_{DS}}$, and the average interface state generation N_{it} in an nMOS transistor.

$$V_{DSAT} = \frac{E_{crit0} L (V_{GS} - V_T)}{E_{crit0} L + V_{GS} - V_T} \tag{3.1}$$

$$I_{sub} = \frac{A_i}{B_i} I_{DS} (V_{DS} - V_{DSAT}) \exp\left[\frac{-B_i l_c}{V_{DS} - V_{DSAT}}\right] \tag{3.2}$$

$$N_{it} + \frac{B_p X_H}{2 D_H} N_{it}^2 = \int_0^T K\left(\frac{I^m_{sub}}{I^{m-1}_{DS}}\right) dt \tag{3.3}$$

where $E_{crit0} \approx 10^4 V/cm$; $m \approx 3$; A_i and B_i are the impact ionization coefficients; D_H and X_H represent diffusion constants; K and B_p represent process-dependent coefficients; and $l_c = 0.2X_{ox}^{1/3}X_j^{1/2}$, where X_{ox} is the oxide thickness and X_j is the junction depth.

Based on the above equations and on detailed circuit simulation of typical complementary CMOS gates, we have observed the following [7]:

(a) The substrate current is exponentially proportional to $(V_{DS} - V_{DSAT})$. Therefore, if the transistor is prevented from entering the saturation region or at least the voltage across the channel is reduced, HCE degradation in the transistor can be reduced dramatically. As a consequence, the nMOS transistors in a NAND gate have less degradation than those in a NOR gate configuration, since the serial connection of nMOS transistors reduces V_{DS} across each nMOSFET, which, in turn, reduces hot-carrier generation. Thus, a design consisting mainly of NAND gates could be more reliable than a design containing mainly NOR gates, inverters, and buffer.

(b) Degradation depends on the switching frequency of the transistor, which is defined as the probability of having a gate output transition occurring as a result of the transistor under consideration switching within the gate. Since HCE occurs when the transistor is in the saturation region, which, in CMOS circuits, occurs only during transition, it follows that, on the average, the more frequently an nMOS transistor causes its gate output to go through a high-to-low transition, the more damage the transistor experiences.

(c) Degradation depends on the duration of a transistor in the saturation region, which, in turn, depends on the input slew rate and the loading capacitance. A slower slew rate causes a wider I_{sub} time span and, hence, more degradation; a higher loading capacitance also increases the time the transistor stays in saturation.

(d) Degradation also depends on the position of a transistor within a subcircuit. It has been shown by circuit simulation that *the top nMOS transistors in a gate that are directly connected to the output node have the potential of experiencing most damage if they switch last* [7]. For example, in a CMOS NAND circuit, the top nMOS transistor is usually the most susceptible to HCE.

In the case of pMOS transistors, the time integral of the transistor gate current, I_{gate}, has been found to be an appropriate measure of its HCE-induced degradation [24, 25]. The computation of an average I_{gate} in pMOS transistors in a given design can be easily included as part of the statistical or probabilistic simulation process. Note that I_{gate} will flow whenever the transistor gate signal switches. Thus, average I_{gate} depends on the transistor gate signal switching frequency, as well as on the transistor characteristics and its operating conditions. In contrast to nMOS transistors where HCE-induced degradation causes a decrease in the transistor's current driving capabilities, HCE-induced degradation in a pMOS transistor *increases* its current driving capabilities. In addition, in current technologies, HCE-induced degradation in pMOS transistors is much less severe than in nMOS transistors. Thus, no subcircuit redesign strategies have been proposed yet for reducing HCE in pMOS transistors under subcircuit operating conditions, as has been done for nMOS. However, one obvious way to reduce HCE in a pMOS transistor is to minimize its gate signal switching frequency, which can most appropriately be done by optimization techniques at the functional level during the logic synthesis stage.

3.3 HCE Gate Delay Degradation Model

As mentioned above, HCE cause a shift in the affected nMOS transistor's threshold voltage V_t and decrease its transconductance, g_m, as will be described below. This, in turn, causes an increase in the delay of the subcircuit in which the transistor is embedded. In this section we describe a method for estimating the increase in subcircuit delay in terms of the estimated degradation in the nMOS transistors comprising the subcircuit.

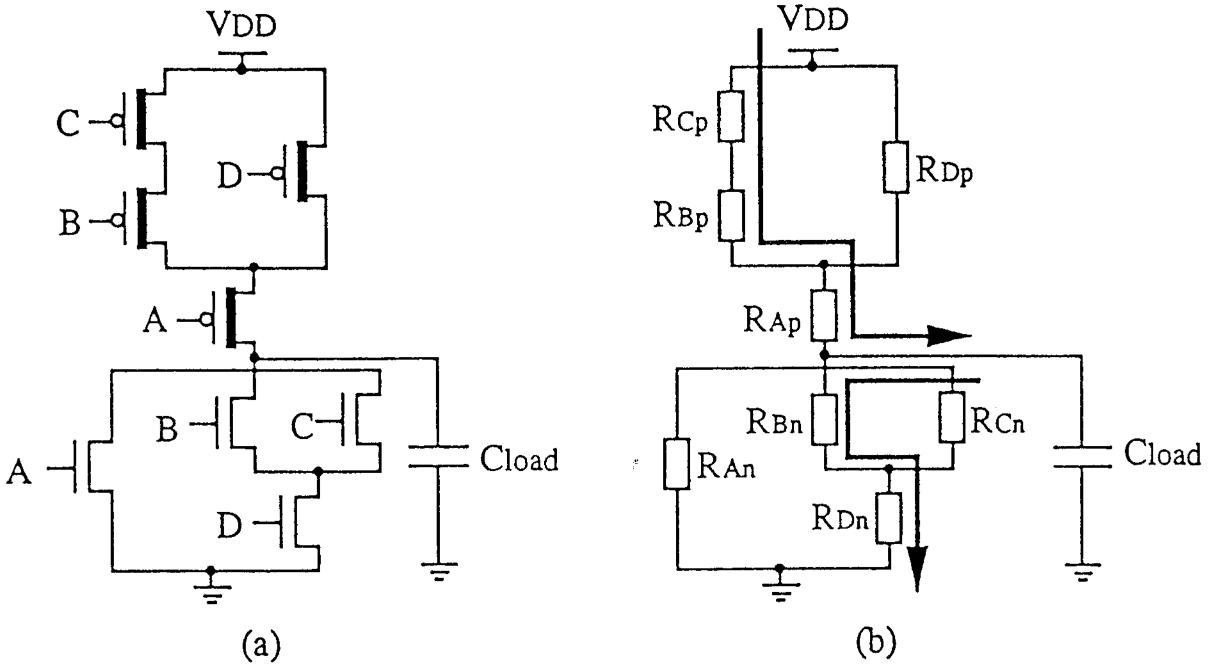

Figure 3.1. (a) A CMOS circuit and(b) its equivalent RC network and worst-case delay paths.

In estimating the circuit delay, we employ a linearized RC model. We assume that the circuit is first partitioned into channel-connected subcircuits or gates. In each subcircuit, the transistors are replaced with equivalent linear resistors [26, 27]. For simplicity, we will lump the subcircuit capacitors together, including fanout capacitance, at the subcircuit's output node, although this is not necessary. As an illustration, Fig. 3.1 shows a CMOS circuit and its equivalent RC network.

Worst-case delay estimation is considered, based on the maximum resistive path between the output node and V_{DD} and ground, respectively. In this case, the worst-case low-to-high (LH, rise) and high-to-low (HL, fall) transition propagation delays of the circuit, based on the Elmore delay [28], are

$$t_{PLH} = (R_{A_p} + R_{B_p} + R_{C_p}) * C_{load} \tag{3.4}$$

$$t_{PHL} = (R_{B_n} + R_{D_n}) * C_{load} \tag{3.5}$$

assuming that

$$R_{A_n} = R_{B_n} = R_{C_n} = R_{D_n}, R_{A_p} = R_{B_p} = R_{C_p} = R_{D_p} \tag{3.6}$$

In (3.4), t_{PLH} is defined as the difference between the time when the low-to-high output signal crosses $V_{DD}/2$ and the time at which the input signal, a step function, switches. A similar definition holds for t_{PHL}. The expression for the equivalent resistance can be derived from the propagation delay of a CMOS inverter driving a loading capacitance. Consider the inverter shown in Fig. 3.2(a). The equivalent RC circuit during the fall transition of the inverter is shown in Fig. 3.2(b). From [29], the high-to-low propagation delay of the inverter is given by

$$t_{PHL} = \frac{C_{out}}{\beta_n(V_{DD} - V_{Tn})} \left\{ \frac{2V_{Tn}}{(V_{DD} - V_{Tn})} + ln[\frac{4(V_{DD} - V_{Tn})}{V_{DD}} - 1] \right\} \tag{3.7}$$

and the fall propagation delay of the equivalent circuit is

$$t_{PHL} = R_n \times C_{out} \tag{3.8}$$

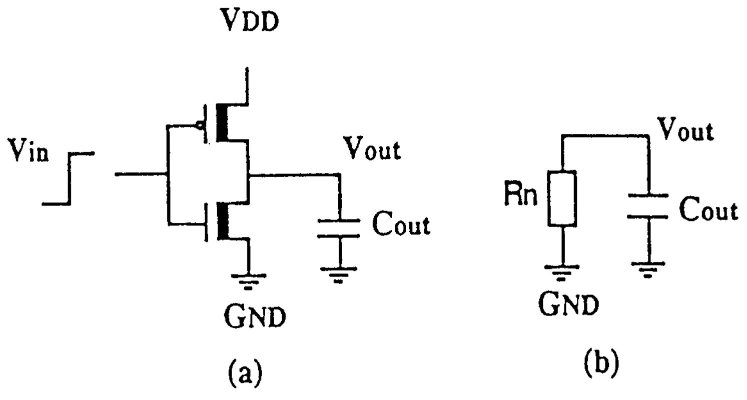

Figure 3.2. (a) A CMOS circuit and(b) its equivalent RC network and worst-case delay paths.

Comparing Eqs. (3.7) and (3.8), the equivalent resistance of an MOS transistor can be expressed as

$$R_{eq} = \begin{cases} \frac{1}{\beta_n(V_{DD}-V_{Tn})}\left\{\frac{2V_{Tn}}{(V_{DD}-V_{Tn})} + ln[\frac{4(V_{DD}-V_{Tn})}{V_{DD}} - 1]\right\} & \text{for nMOS} \\ \frac{1}{\beta_p(V_{DD}-|V_{Tp}|)}\left\{\frac{2|V_{Tp}|}{(V_{DD}-|V_{Tp}|)} + ln[\frac{4(V_{DD}-|V_{Tp}|)}{V_{DD}} - 1]\right\} & \text{for pMOS} \end{cases} \quad (3.9)$$

The equation above shows that HCE may affect the nMOS equivalent resistance, which depends on both V_T and μ_n, but the capacitance is not affected. Earlier work on HCE assumed that the oxide-interface charge ($Q_{it} = N_{it} \times q$) is uniformly distributed over the entire channel region; hence, the change of threshold voltage is linearly proportional to the generated interface charge [30]. However, experiments have shown that the damage caused by HCE is localized in the channel near the drain node (for nMOS) and the uniform model cannot correctly characterize the damaged device. A one-dimensional model has been proposed for nMOS transistors with hot-carrier induced oxide damage [31, 32]. In this model, the transistor channel is divided into two regions, the *damaged* and *undamaged*. The damaged region is located within 0.1 μm of the drain node and the interface charge is uniformly distributed in this region. The rest of the channel remains undamaged with no generated interface charge.

We adopt the concept of this model and use a two-transistor degradation model, as shown in Fig. 3.3, in which a damaged nMOS transistor is considered as two transistors connected in series, with different threshold voltages and mobilities. The channel lengths of the two transistors are 0.1 μm and $L - 0.1$ μm, respectively, where L is the channel length of the original transistor, and the transistor with the 0.1 μm channel length is considered to be the damaged one. The model parameters of the two transistors are calculated with the following equations [32]:

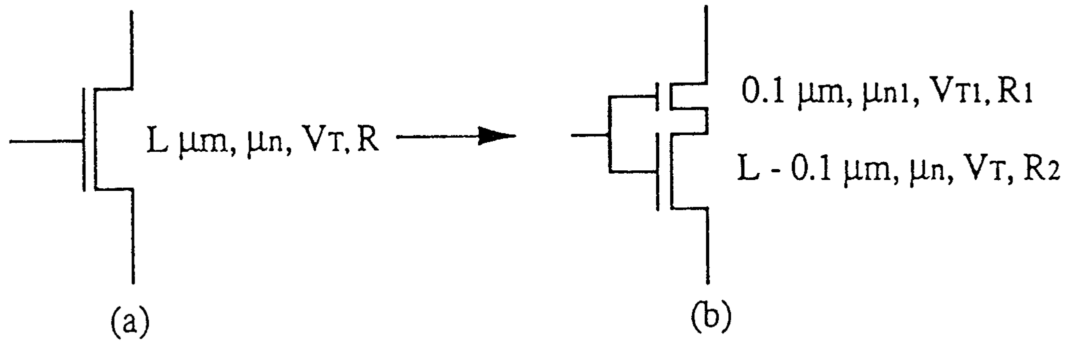

Figure 3.3. An nMOSFET (a) before and (b) after HCE degradation.

$$\mu_{ox} = 1000 \times \frac{3 \times 10^{11}}{N_{it}} \times \frac{T}{80} \quad [cm^2/Vsec] \tag{3.10}$$

$$\frac{1}{\mu_{n_1}} = \frac{1}{\mu_n} + \frac{1}{\mu_{ox}} \tag{3.11}$$

$$V_{T_1} = V_T - \frac{Q_{it}}{C_{ox}} \tag{3.12}$$

where T is the temperature in the oxide. The values of μ_n, μ_{n_1}, V_T, and V_{T_1} are then applied to Eq. (3.9) to calculate the equivalent resistances R_1 and R_2 of Fig. 3.3. The values of the equivalent resistances of the damaged transistors in each gate are then used to estimate the increase in the fall delay of the gate.

The RC delay model mentioned above is based on a step input assumption, which considers that only one of the nMOS and pMOS transistors, in an inverter, would conduct during either the rise or fall transition. In [33], an improved delay model, taking into account the effect of input slope, is developed. The fall propagation delay, $t_{PHL_{nonstep}}$, of a CMOS inverter with respect to a nonstep low-to-high input transition

$$V_{in} = \begin{cases} 0 & t < t_0 \\ V_{DD} * (\frac{t-t_0}{t_{LH}}) & t_0 \leq t \leq t_0 + t_{LH} \\ V_{DD} & t > t_0 + t_{LH} \end{cases} \tag{3.13}$$

is given by

$$t_{PHL_{nonstep}} = t_{PHL_{step}} + \frac{t_{LH}}{6}\left[1 + \frac{2V_{Tn}}{V_{DD}}\right] \tag{3.14}$$

where

V_{Tn} = threshold voltage of nMOS transistor

V_{DD} = supply voltage

$t_{PHL_{step}}$ = inverter transition delay under a step input excitation

$t_{PHL_{nonstep}}$ (or $t_{PLH_{nonstep}}$) is defined as the difference between the times when the output and input signals cross $V_{DD}/2$, as shown in Fig. 3.4. At the primary input nodes, the slew rate, t_{HL} or t_{LH}, is provided by the user; while at the output node of a gate, the output slew rates, t_{HL} and t_{LH}, are approximated with the following equations because of the difficulty of estimating output slew rates accurately as functions of input slew rates. The equations below are used to calculate the output slew rate of a gate with step function input, which we found to be a very good approximation.

$$t_{HL} = 2 \times t_{PHL_{step}} \tag{3.15}$$

$$t_{LH} = 2 \times t_{PLH_{step}} \tag{3.16}$$

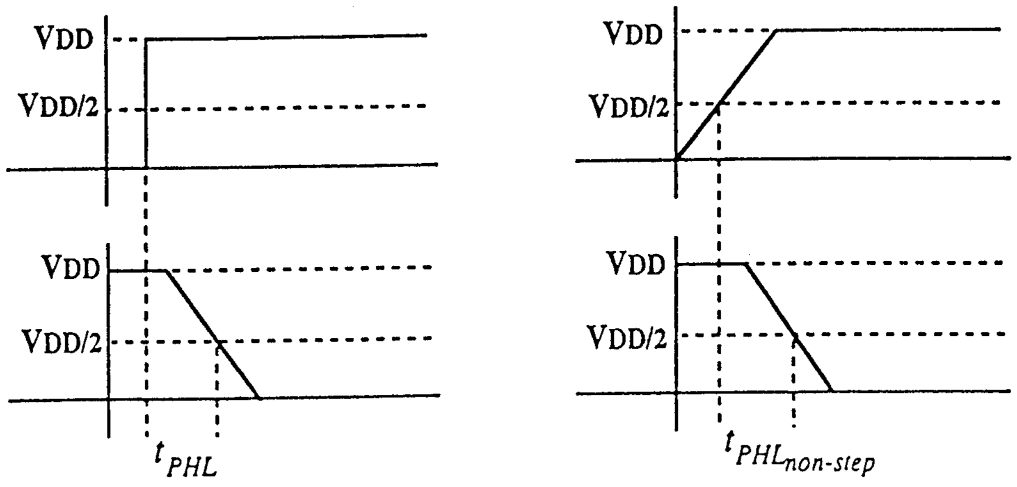

Figure 3.4. Graphical illustrations of $t_{PHL_{step}}$ and $t_{PHL_{nonstep}}$.

So far, we have considered the effects of degradation in nMOS transistors on gate delay (t_{PLH}). In order to include the effects of degradation in pMOS transistors on gate delay, it is necessary to relate the change (decrease) in the effective resistance of the pMOS transistor to the time integral of I_{gate} and the physical parameters of the transistors. Such an explicit relation does not yet exist. Nevertheless, it is possible to use experimental data and curve-fitting techniques [24] to derive a transistor degradation model, which can then be used to estimate the change (decrease) in t_{PLH}, taking into account the switching frequency of the transistor.

In summary, our program accepts the slew rates, provided by the user, at the primary input nodes; the program then calculates the propagation delay at each gate using Eqs. (3.4)-(3.9), for both fresh and degraded circuits, assuming step function input. Therefore, each gate has three propagation delay values, the fresh t_{PLH}, the fresh t_{PHL}, and the degraded t_{PHL}. The delay values are then used to approximate output slew rates with Eqs. (3.15) and (3.16), and finally the propagation delay is modified with Eq. (3.14) for nonstep input. If pMOS degradation is taken into consideration, then each gate will have four propagation delay values, with one more for degraded t_{PLH}.

3.4 Redesign for Subcircuit Reliability

As mentioned above, the factors that influence HCE degradation in a transistor include its feature size, switching frequency, position in the circuit, the loading capacitance, and input slew rate. Several strategies, based on these factors, have been previously proposed to improve circuit reliability. However, these proposed improvements have been suggested on small circuit structures in isolation, without taking into consideration the operation of the small circuit within a large design, as we do here.

In this section, we will describe five subcircuit redesign strategies. The first two have been previously proposed; the others are new. We also compare their advantages and disadvantages.

(1) To reduce the damage in the top nMOS transistor, a dummy transistor could be added to the affected subcircuit [34]. By adding a self-bootstrapping nMOS transistor, as shown in Fig. 3.5(a), the peak substrate current in transistor NM1 can be reduced by as much as 3.9 times [34] and the interface-state generation can be suppressed by one to two orders of magnitude. In addition, this structure also has shorter rise and fall times compared to those for a conventional NAND logic circuit. A disadvantage of this structure is an increase in the circuit area; therefore, it is preferred to add a small self-bootstrapping nMOS transistor. In addition, this structure cannot be used to improve NOR gates.

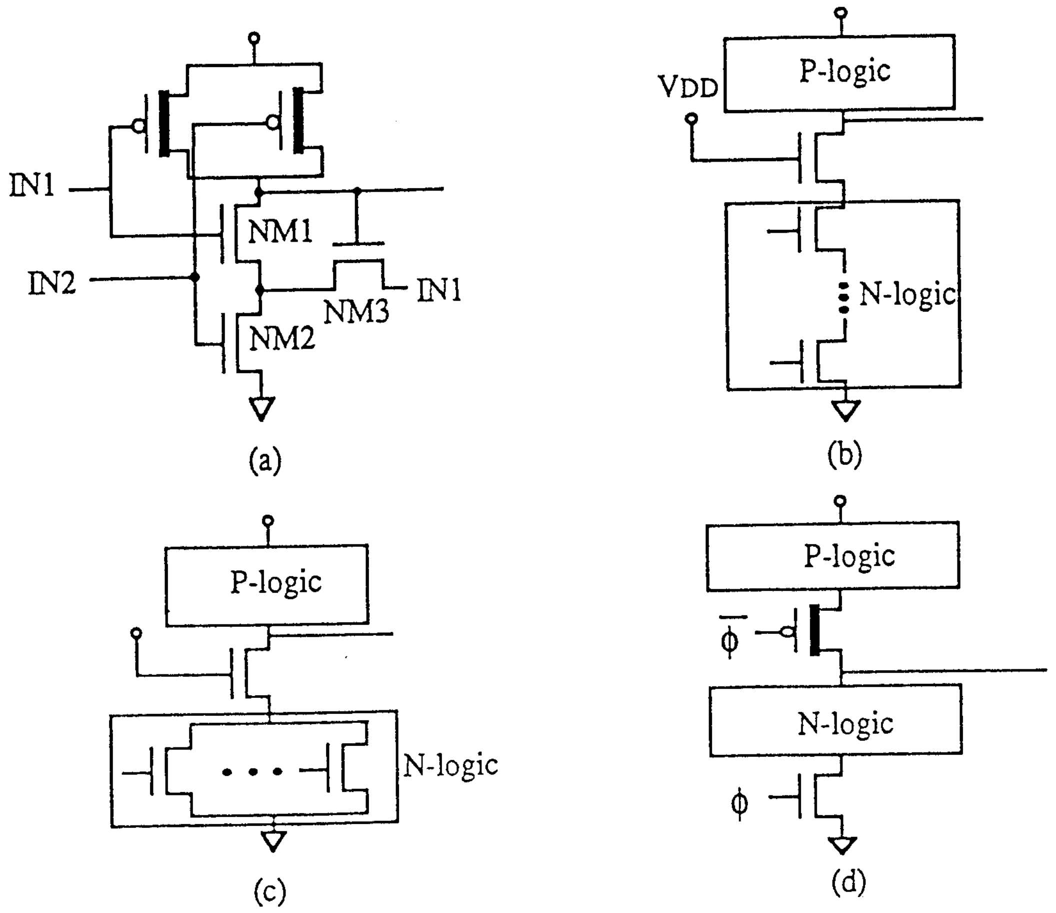

Figure 3.5. Several HCE resistant circuit design strategies.

(2) A second HCE suppressing circuit redesign technique, shown in Fig. 3.5(b), adds a normally-on nMOS transistor at the top of the nMOSFETs in a CMOS circuit, as proposed in [35]. Since the added transistor is always on, the logic function of the circuit is not affected. In addition, the added transistor is always operating in the linear region and, hence, has negligible hot-carrier generation. The rest of the nMOS transistors are not directly connected to the output node and, thus, are relatively safe from HCE degradation. This technique can also be applied to NOR gates, as shown in Fig. 3.5(c). Its disadvantage is the increase

in layout area as well as in the fall transition propagation delay; therefore, it should be used selectively.

(3) In clocked CMOS (C^2MOS) circuits in which the clock signal arrives after all the inputs settle, we recommend that the control nMOS transistors be located anywhere but on the top. With this restriction, the top nMOS transistors never switch last, i.e., their switching does not cause output transitions; hence, HCE degradation can be dramatically reduced. Figure 3.5(d) illustrates an HCE resistant C^2MOS design. The same strategy can also be applied to the dynamic CMOS design.

(4) A CMOS circuit design, shown in Fig. 3.6, has been proposed for testability in [36], where it is proposed that, with the four combinations (00, 01, 10, 11) of the control signals ϕ_1 and ϕ_2, all stuck-open and stuck-on faults of the circuit can be detected. The circuit operates as follows: When $\phi_1 = 0$ and $\phi_2 = 1$, the circuit operates normally and implements the same function as the one without the transistors N_ϕ and P_ϕ (i.e., shorted). When $\phi_1 = 1$ and $\phi_2 = 0$, both N_ϕ and P_ϕ are off, and the output state does not change even if the input signals have transitions. When $\phi_1 = 0$ and $\phi_2 = 0$, only P_ϕ is on and all the stuck-open and stuck-on faults of the pMOSFET network can be detected. Finally, when $\phi_1 = 1$ and $\phi_2 = 1$, only N_ϕ is on and all the stuck-open and stuck-on faults of the nMOSFET network can be detected. This structure can be employed for improving reliability as well. There are two situations: (i) If the circuit is in normal operation most of the time, i.e., the control signals rarely switch, then this design is similar to the structure in Fig. 3.5(b). (ii) If the system clock is used as the control signal, then the design has to be modified by moving

N_ϕ to the bottom of the nMOSFET network, similar to the case of C^2MOS design. This design style, although it improves testability as well as reliability, does increase the area and the clock routing.

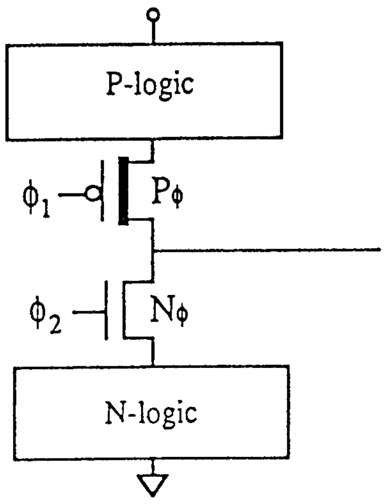

Figure 3.6. A CMOS circuit structure for testability.

(5) It has been proposed that by increasing the width of the pMOS pull-up block of a gate, the hot-carrier damage of the nMOS transistors of its fanout gates can be reduced, since the pMOS block determines the input slew rate of its fanout in a typical CMOS circuit. However, such an increase in the widths of pMOS transistors could increase the degradation of the fanin gate by increasing its short-circuit current during switching, and possibly increase the degradation of its own fanin gates by increasing the load capacitance of these gates. Such an approach has to be included as part of an overall circuit transistor sizing optimization problem rather than only as a local subcircuit redesign strategy.

	NAND1 top	NAND1 bottom	NAND2 top	NAND2 bottom	NAND3 top	NAND3 bottom
connection1						
Nit	1.695	-	0.875	-	3.485	-
Switchingprobability	0.3	0.12	0.08	0.12	0.216	0.168
connection2						
Nit	1.072	-	0.875	-	3.073	-
Switchingprobability	0.12	0.3	0.08	0.12	0.168	0.216

Table 3.1. The damage $N_{it}(x10^{11}\, cm^{-2})$ of all the nMOS transistors in the circuit shown in Figs. 3.7(c) and (d).

(6) Finally, we propose an input signal line reordering to reduce HCE in a gate. Since the damage in a transistor depends on its switching frequency as well as its position in the circuit, the damage can be reduced by connecting nMOS transistors with the least switching frequency to the output node. This is done by reconnecting the circuit input wiring so that the top nMOS transistor always has the least switching probability. For example, Fig. 3.7(a) shows the waveform description of four input nodes A, B, C, and D. The circuit under test is a three-NAND-gate circuit with the detailed gate structure shown in Fig. 3.7(b). When the circuit is connected as in Fig. 3.7(c), the HCE degradation is listed in row 1 of Table 3.1, as well as the switching probability of each nMOS transistor. Note that the bottom transistors of gates NAND1 and NAND3 have lower switching probability values. After reconnecting the inputs of these two gates as in Fig. 3.7(d), the HCE degradation is reduced, according to the results in Table 3.1. The advantage of this method is that the circuit area is not increased. However, this method cannot be applied to inverters and NOR gates in which all the nMOS transistors are at the top, but it can be applied to complex gates. It is interesting to note that input line reordering has been also proposed in delay optimization

[37] and low-power design [38, 39, 40]. In fact, reducing subcircuit power improves its reliability since both power and reliability depend directly on circuit currents.

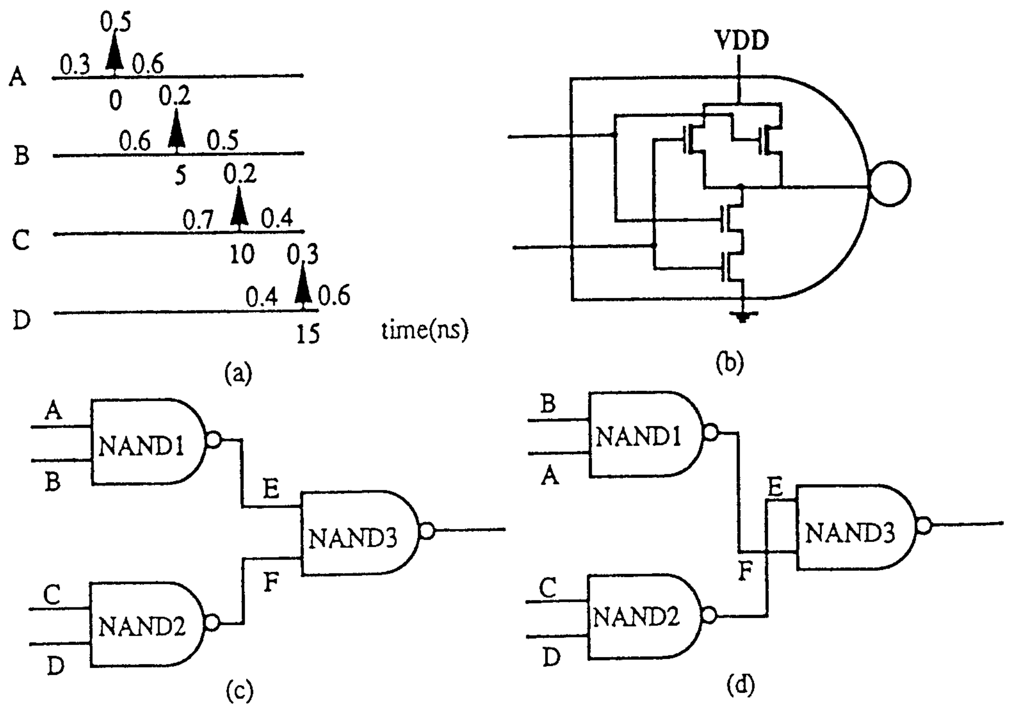

Figure 3.7. (a) The probabilistic voltage waveforms at the four input nodes, A, B, C, and D. t_{HL} and t_{LH} are both 0.2 ns for each input event.
(b) The detailed structure of a CMOS NAND gate.
(c) Original connection of a three-NAND-gate circuit.
(d) A better connection of the three-NAND-gate circuit for reliability.

Although the above techniques are very efficient in suppressing HCE degradation, note that except for the input signal reordering method, they cause an increase in the circuit area and, hence, should be applied selectively, based on the gate switching frequency and HCE degradation of the individual transistors. Thus to minimize the increase in circuit area, only subcircuits that contain critical transistors have to be redesigned; a critical transistor is defined as one that receives HCE degradation higher than a specified threshold.

In addition to the increase of circuit area, another drawback of the above redesign strategies is that they may cause the circuit delay to increase. Adding a normally-on transistor on top of the n-part of a CMOS gate, even though it eliminates HCE degradation, could severely degrade circuit performance. Since both HCE and HCE-resistant redesigns could increase circuit delay, after any of the redesign strategies have been applied to the circuit, special efforts have to be made to improve circuit performance, but at the same time not increasing HCE degradation in transistors and subcircuits that have not been redesigned. In the next section, we consider area/timing/reliability optimization in a unified way. The aim is to improve reliability without degrading performance and increasing area unnecessarily.

3.5 Design for Area, Timing, and Reliability

In high-speed MOS digital circuit design, delays often have to be reduced to obtain faster response times, with a minimum penalty in area. A typical synchronous digital circuit consists of multiple stages of combinational logic blocks separated by latches that are clocked by system clock signals. The worst-case delays of the combinational blocks must be below a certain specification so that valid signals reach a latch in time for a transition in the signal clocking the latch. The aim of timing/area optimization is to ensure that the longest delay path in each combinational block is below a specific bound, while keeping the area to a minimum.

When including reliability as part of the design objective function, our aim here is to ensure that the design will function at its initially designed speed, without sacrificing area, in spite of an expected delay degradation caused by HCE after a specified period of time. Our aim thus is to ensure that the longest path in the circuit *after* HCE-induced degradation over a certain time period meets timing specifications. Since our concern is only with longest paths, gates that do not lie on any longest path (after degradation) need not be redesigned since any increase in their delays could be tolerated without affecting the reliability and functionality of the design. Thus, the number of transistors and gates that have to be redesigned could be a small fraction of the total number of affected transistors and gates.

For a combinational circuit, the sizing problem is formulated as

$$\text{Minimize} \quad \text{Area} \tag{3.17}$$

$$\text{Subject to } \mathit{Delay} \leq T_{spec}$$

where *"Delay"* is the longest path or critical path delay in the circuit, i.e., the largest delay path among all paths between a primary input and a primary output, and T_{spec} is a time specification, usually slightly less than the clock period. A number of algorithms have been proposed for identifying longest paths in combinational circuits [41, 42, 43]. Initially, we applied the Program Evaluation and Review Technique (PERT) described in [41]. Given a circuit, partitioned into a cluster of gates/subcircuits, a topological sort based on the fanin/fanout connections is performed. After all of the subcircuits are scheduled, the delay at the output of each subcircuit is calculated assuming that the latest arriving input signal of the subcircuit controls its output transition. After all of the subcircuits are processed, the latest arrival time at the primary output nodes is defined as the circuit delay. A trace-back method is then applied to identify the critical path or paths.

However, PERT does not eliminate false paths, and thus could lead to very conservative results. We subsequently applied the algorithm described in [43], which eliminates false paths in the process of identifying the longest paths. In this algorithm every time a new subcircuit is encountered during the process of tracing the longest paths from the primary inputs to the primary outputs, the logic functions, which are constructed in the form of Binary Decision Diagrams (BDD), of all the side inputs to the subcircuit are used to check if certain input combinations exist which can activate the path. If not, a different path is chosen. This is referred to as path sensitization. This algorithm eliminates most false paths and gives more accurate results compared to PERT, at the price of additional computation.

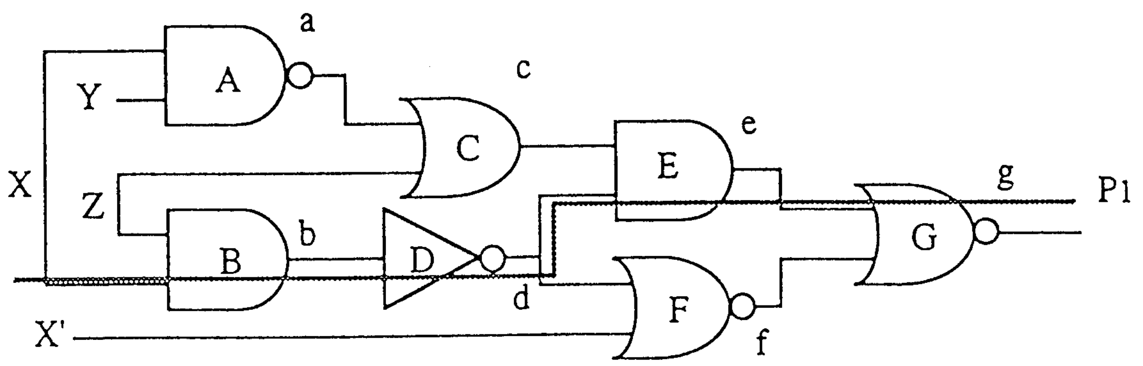

Figure 3.8. Longest path in a logic circuit example.

Most previous work on critical path analysis, including the one proposed in [43], makes the assumption that each subcircuit has only one delay value associated with it and, hence, each path has one delay. However, in general, the rise delay of a subcircuit is different from its fall delay; therefore, there are two delay values associated with a given path. For example, the path P_1 in Fig. 3.8 has two delay values:

$$t_{B_{PLH}} + t_{D_{PHL}} + t_{E_{PHL}} + t_{G_{PLH}} \tag{3.18}$$

and

$$t_{B_{PHL}} + t_{D_{PLH}} + t_{E_{PLH}} + t_{G_{PHL}} \tag{3.19}$$

For XOR gates in which the output may switch either way when the input has a transition, we use the larger one of rise and fall delays to make sure the path delay is not underestimated.

Timing and area optimization problems in VLSI design can be broadly classified into two types. One is suitable for full custom design and considers individual transistor sizing where the size of each transistor, as measured in terms of its channel width, is a design parameter that controls both delay and area [44, 45, 46]. To incorporate HCE-induced degradation effects into the transistor-sizing optimization process, it is necessary first to estimate the change in the conductance of each transistor due to HCE, as described in Section 3.3. To procure this information, an initial design could be obtained by applying a timing/area optimization procedure, such as the one described in [46], using the "fresh" model of the devices. The HCE-induced degradation can then be estimated using this initial design. A second timing/area optimization pass can then be carried out using the degraded transistor models. In this second pass, only transistors lying along the paths whose delays after degradation exceed T_{spec} have to be considered. It is expected that the number of paths that violate the timing constraints after degradation is small. In addition, one may safely assume that the estimate of transistor degradation using the initial design is not very sensitive to changes in some transistor sizes and in gate delays caused by HCE, even though changes in gate delay may affect glitching, which, in turn, may affect the average current flows in the transistors. Otherwise, the HCE-degradation and optimization procedures can be repeated, if necessary, at extra cost in computation.

The second type of timing and area optimization problems targets standard-cell circuit design styles. A standard-cell library typically contains several versions of any given gate type, each of which has a different gate size. The optimization problem is concerned with choosing optimal gate sizes from the library to minimize a cost function, typically total circuit area, while meeting the timing constraints [47, 48].

In the remainder of this section, we consider the standard-cell design problem and follow the approach described in [48], where we incorporate estimated HCE-induced gate delay degradation into the optimization problem [49]. In [48], the gate-sizing problem is solved using a three-phase scheme: (1) Formulation of the optimization problem as a linear programming (LP) problem, which is solved using an available LP package [50]; (2) solution of the LP problem is then mapped onto a permissible set; and (3) based on the permissible set, the gate sizes are adjusted to satisfy timing constraints.

As explained in [48], the delay of a gate in a standard-cell library can be characterized by

$$\begin{aligned} d(x) &= R \times C_{load} + \tau \\ &= \frac{R_u}{x} \times C_{load} + \tau_1 x + \tau_2 \end{aligned} \tag{3.20}$$

where R is the gate equivalent resistance, C_{load} is the gate load capacitance, τ is the gate intrinsic delay, R_u represents the on-resistance of a unit transistor, and x_i is called the nominal size of gate g_i. Thus, the size of each gate can be parameterized by a number, x_i, referred to as the nominal gate size. We consider the worst-case delay, which can be expressed in terms of the worst-case equivalent resistance of the gate, which, in turn, is related to the highest resistive path between the output node and the power node, as pointed out in Section 3.3.

The output load capacitance, C_{load}, of a gate is calculated by summing the gate terminal capacitances of its fan-out gates and interconnect wiring capacitance (interconnect resistance is ignored), assuming the layout information is available. The output load capacitance of gate i

driving gate j can be approximated by a convex piecewise-linear ($pw\ell$) function of x_j, the size of gate j [48]. By also approximating $\frac{1}{x_i}$ by a convex $pw\ell$ function of x_i, the delay function of all the gates put together is a convex $pw\ell$ function.

$$\hat{\mathbf{D}}(\mathbf{x}) = \mathbf{A}\mathbf{x} + \mathbf{b} \tag{3.21}$$

In addition, the cell area of gate i can be expressed as a convex $pw\ell$ function of x_i:

$$area(i) = \gamma_i x_i + \epsilon_i \tag{3.22}$$

The worst-case signal arrival time, m_i, which corresponds to the worst-case delay from the primary inputs to gate i, is given by

$$m_i = max\{m_j + d_i\} \geq (m_j + d_i) \quad \forall j \epsilon fanin(i) \tag{3.23}$$

where d_i is the delay of gate i as defined in Section 3.3. Putting all the constraints together, an LP is formulated as in [48].

$$\begin{array}{lll} \text{Minimize} & \sum_{i=1}^{n} \gamma_i \cdot x_i & \\ \text{Subject to} & For\ all\ gates\ i,\ i = 1, ..., N & \\ & m_j + d_i \leq m_i & \forall j \epsilon fanin(i) \\ & m_i \leq T_{spec} & \forall\ gate\ i\ at\ primary\ outputs \\ & d_i \geq \hat{D}_i(x_i, \mathbf{x}_{i,fo(i)}) & \\ & minsize_i \leq x_i \leq maxsize_i & \end{array} \tag{3.24}$$

where $\mathbf{x}_{i,fo(i)}$ are the sizes of the gates to which gate i fans out, and $Minsize(i)$ and $Maxsize(i)$ are the minimum and maximum sizes of gate i in the library, respectively.

Equation (3.24) is a very sparse LP in the variables x_i, d_i, and m_{ij}. The solution is started with all gates at minimum size so that a minimum number of gates have to be increased in size in order to satisfy the constraints. The solution of gate sizes obtained from the LP is, in general, not in the available set of discrete sizes in the cell library. Thus, a cell-mapping algorithm, followed by a delay constraint adjusting algorithm, are invoked to obtain a permissible solution that satisfies all the constraints, as described in [48].

In incorporating HCE-induced degradation into the optimization process, an initial design is obtained using the "fresh" device models. This initial design is used to estimate the increase in gate delays. We assume that the transistor description of each gate in the cell library is available. Following that, the LP described in (3.24) is reformulated using the new values of d_i and m_i after estimated degradation, and solved starting from the initially optimized design. One option we have tried is to skip the mapping and the adjusting algorithms in the second pass of the optimization process and choose the sizes of all the gates selected by the LP solution to be the closest larger permissible sizes in relation to their sizes obtained from the optimized "fresh" design. As we show in Section 3.6, for all benchmark examples considered, only a very small number of gates have to be adjusted in size during the second pass of the optimization algorithm, and the increase in area of each benchmark circuit for compensating HCE-induced degradation is less than 1%.

3.6 Simulation Results

3.6.1 HCE-degradation simulation results

The algorithms and strategies described above have been implemented into a simulator, *iProbe-d*, and used to simulate and redesign a set of benchmark circuits. Table 3.2 shows the results of applying probabilistic simulation on ISCAS85 combinational benchmark circuits [51]. These benchmark circuits have been converted to *SPICE* input files with complementary CMOS

ISCAS85 Circuits	# of nMOS transistors in each damage level (Normalized D_n)						CPU time (sec)
	< 1.0	1.0 - 2.0	2.0 - 3.0	3.0 - 4.0	4.0 - 5.0	> 5.0	
C432	232	37	16	65	9	53	1.00
C499	576	0	80	0	0	226	1.65
C880	443	62	40	39	52	265	3.37
C1355	776	176	56	32	0	114	4.43
C1908	653	30	29	65	86	760	7.60
C2670	971	209	151	80	125	1146	8.49
C3540	1450	340	225	171	143	1423	17.12
C5315	2147	274	374	261	192	2383	18.37
C6288	512	92	495	417	156	3384	92.08
C7552	2767	332	861	262	191	3285	26.84

Table 3.2. HCE damage on ISCAS85 benchmark circuits and the computation time.

transistor circuit implementation of each gate. We assign an event, occurring at time 0, at each primary input node with probability 0.5 being high before and after switching, and probability 0.25 to switch from low-to-high and high-to-low. The input signals are assumed to repeat every 50 ns (100 ns for C6288), and the damage is extrapolated to 10 years of continuous operation. The simulation results are based on the following parameter values: $A_i = 9 \times 10^6\ cm^{-1}$, $B_i = 1.7 \times 10^6\ V/cm$, $l_c = 2 \times 10^{-5}\ cm$, $K = 5 \times 10^{15}$, $\frac{B_p X_H}{D_H} = 10^{-8}\ cm^2$, and $m = 2.9$. For different process/device technologies, the values of these parameters may vary.

After HCE degradation in each nMOS transistor is estimated, the critical path analysis algorithm is employed to check the global degradation criterion. Table 3.3 shows the circuit delay before and after ten years of HCE degradation. From this table, we see that the delay increase for most circuits is negligible, with C6288 having the largest delay increase, merely 1.09%. The results almost assure that HCE has little effect on circuit performance, given $V_{DD} = 5$ V and transistor effective channel length $\geq 1\ \mu m$. These results are obtained assuming degradation only in the nMOS transistors and ignoring degradation in the pMOS transistors. As mentioned

Circuit	#gates	#nodes	CKT delay before HCE	CKT delay after HCE	delay increase
C432	160	198	20.88	20.95	0.34%
C499	202	245	16.46	16.50	0.24%
C880	383	445	19.02	19.05	0.16%
C1355	546	589	20.11	20.15	0.20%
C1908	880	915	27.71	27.98	0.97%
C2670	1193	1428	31.38	31.57	0.61%
C3540	1669	1721	37.87	38.13	0.69%
C5315	2307	2487	34.69	34.98	0.84%
C6288	2416	2450	80.04	80.93	1.09%
C7552	3512	3721	28.78	29.01	0.80%

Table 3.3. The circuit delay of several benchmark circuits before and after HCE degradation. $V_{DD} = 5$ V.

earlier, after degradation, the nMOS transistor current driving capability decreases and the pMOS transistor current driving capability increases. However, since nMOS degradation is more severe than pMOS degradation, any speedup caused by the pMOS transistors is overshadowed by the slowdown in the nMOS transistors. Thus, the results shown in Table 3.3 are slightly on the conservative side.

However, with the decrease in feature size and the corresponding increase in circuit density on a chip, capacitive coupling, which may cause voltage overshoot, can no longer be ignored [52]. In addition, many IC manufacturers allow 5 - 10% margin tolerance for supply voltage. We therefore simulate the same set of circuits with 5.5 V as the supply voltage. The same input signal statistics are applied; that is, one event is assigned to each primary input, with 50 ns as the clock cycle (100 ns for C6288). After a simulated ten years of continuous operation, the circuit delay values are shown in Table 3.4. With the 0.5 V increase in supply voltage, after ten years of continuous operation, some circuits experience a significant delay increase. For example,

Circuit	#gates	#nodes	delay (ns) before HCE	delay (ns) after HCE	delay (ns) increase
C432	160	198	17.98	18.21	1.28%
C499	202	245	14.18	14.30	0.85%
C880	383	445	16.40	16.61	1.28%
C1355	546	589	17.32	17.49	0.98%
C1908	880	915	23.90	25.54	6.86%
C2670	1193	1428	27.07	27.84	2.84%
C3540	1669	1721	32.67	33.89	3.73%
C5315	2307	2487	29.93	31.13	4.01%
C6288	2416	2450	69.19	77.81	12.46%
C7552	3512	3721	24.83	25.69	3.46%

Table 3.4. The circuit delay (longest path (l.p.)) of several benchmark circuits before and after HCE degradation. $V_{DD} = 5.5$ V.

C1908 posts a 6.86% increase in circuit delay and the delay increase for C6288 is almost 12.5%. With this amount of delay increase, the effect of HCE on circuit performance can no longer be ignored. In the next subsection, we show the results of applying the redesign strategies and the optimization algorithms described above and compare their abilities in suppressing HCE-induced degradation and enhancing circuit performance.

3.6.2 Simulation results of redesign strategies

The results of redesign improvements are shown in Table 3.5, which shows the impact of applying three strategies, namely, increasing transistor widths, adding normally-on transistors, and interchanging input lines, on benchmark circuit C3540. We set the criteria of local redesign to be normalized N_{it} equal to 5.0 for individual transistor damage and 0.05 ns for subcircuit delay increase. We also assume the equivalent resistance of the normally-on transistor to be one-fourth that of a normal nMOSFET. As can be seen from this table, increasing transistor

C3540	# of nMOS transistors in each damage level						Circuit delay (ns) after HCE
	< 1.0	1.0 - 2.0	2.0 - 3.0	3.0 - 4.0	4.0 - 5.0	> 5.0	
initial	959	177	142	118	189	2167	33.89
(1)	959	177	142	177	451	1846	32.84
(2)	3211	176	138	72	155	0	33.26
(3)	967	199	150	112	215	2109	33.87

Table 3.5. (1) Increasing the width of critical transistors by 75%.
(2) Adding a normally-on transistor. (3) Interchanging input lines.

widths can improve circuit delay but have less improvement on transistor damage. The reason is that HCE damage is a weak function of device width. Adding a normally-on transistor, on the other hand, does eliminate HCE degradation in all the critical transistors, while the circuit delay is improved compared to that for the damaged circuit. However, both strategies increase the total circuit area by 20%, assuming the normally-on transistor occupies the same area as one regular nMOSFET.

The strategy of interchanging input lines has little effect on circuit area. However, its ability to enhance circuit performance of this particular circuit is limited. As shown in the table, 58 of the 2167 transistors in the highest damage level moved to lower damage levels after interchanging the input lines of 192 NAND-type gates. Some other transistors may also experience less degradation, but still remain at the same damage level. The circuit delay, however, does not improve to a significant degree. The reason is that interchanging inputs, although it increases reliability, does not improve the performance of the gates on the critical path when the gate is a NOR type, or when the top nMOS transistor in a gate already has the least switching frequency. If more than half of the gates on the critical path are in either situation, which is the case in this example, then we could not expect the strategy to have any significant effect on circuit performance enhancement.

As pointed out above, increasing the width of all the critical transistors can enhance the circuit performance at the expense of circuit area. Since many of the most susceptible transistors may not be on the critical path, it is not necessary to resize these transistors. In the following we present the results of applying the cell-based gate sizing algorithms [49] to area/timing/reliability optimization of the benchmark circuits. We use the circuit delay of the fresh circuit after the first pass area/timing optimization as the timing constraint that the damaged circuits have to satisfy after ten years of continuous operation.

Table 3.6 demonstrates the application of gate sizing on several benchmark circuits to compensate for the delay increase caused by HCE. Three delay values are associated with each circuit. They represent the circuit delay of the initial circuit, the damaged circuit after ten years of operation, and the optimized circuit delay after ten years of operation. The supply voltage is chosen as 5.5 V. As can be seen in the table, only a small number of the gates have to be resized. For every circuit under test, less than a 1% increase in circuit area is observed. The results justify our usage of the simplified gate sizing algorithm in the redesign phase, as explained in Section 3.5.

Among these strategies, it is shown that gate sizing is the most efficient method. This conclusion is based on the assumption that minimizing circuit-level performance degradation (in this case, delay increase) is more vital to the circuit designer than the degradation of individual devices, provided the circuit area increase is minimal. Although adding a dummy transistor, increasing the width of the critical transistor and increasing the width of pMOS pull-up block may improve HCE reliability, their impact on circuit area should not be ignored. Our conclusion is that the best strategy for improving hot-carrier reliability is first, to improve process technology and device structure so that the transistors are more HCE robust; second, to reorder gate input

Circuit	#gates	#nodes	longest path delay (ns)			# gates resized	area increase
			initial	after HCE	optimized		
C432	160	198	17.98	18.21	17.85	1	0.12%
C499	202	245	14.18	14.30	14.08	2	0.45%
C880	383	445	16.40	16.61	16.29	3	0.16%
C1355	546	589	17.32	17.49	17.10	8	0.35%
C1908	880	915	23.90	25.54	23.65	16	0.71%
C2670	1193	1428	27.07	27.84	26.70	5	0.18%
C3540	1669	1721	32.67	33.89	31.93	13	0.29%
C5315	2307	2487	29.93	31.13	29.41	15	0.20%
C6288	2416	2450	69.19	77.81	-	-	-
C7552	3512	3721	24.83	25.69	24.44	9	0.09%

Table 3.6. The circuit delay of several benchmark circuits before HCE, after HCE, and after gate sizing. $V_{DD} = 5.5$ V.

signal lines based on signal switching frequency; and third, to selectively add dummy transistors and increase widths for the most critical transistors, only when absolutely necessary; and, finally to employ either transistor or gate sizing for delay optimization.

3.7 Summary

A system for circuit HCE reliability estimation and redesign has been presented in this report. This system applies probabilistic simulation techniques to find critical nMOS transistors that might have excessive HCE degradation and to provide redesign guideline information on minimizing HCE effects on long-term circuit performance. The simulation results show that this approach is very efficient for long-term performance enhancement of CMOS combinational circuit blocks in synchronous sequential logic designs. The focus has been on design optimization at the physical layout design level. It is also clear from this study that reliability and low power go hand in hand since they both depend on current flows in the circuit. At the individual logic gate level, reliability as well as power are directly related to the average switching activity of the gate. Thus design for reliability has to be carried out at all levels of the design process, from architectural level to layout and device levels, together with a design optimization for area, timing, and low power [53].

4 Summary and Conclusions

In this report we have described our recent work on reliability estimation and design for reliability of integrated circuits. The report summarizes the work done on two tasks: one is current and reliability estimation of sequential CMOS VLSI circuits, and the other is circuit design for hot-carrier reliability.

Many reliability issues in an integrated circuit, such as electromigration and hot-carrier-induced degradation, are directly related to the average current flow in the design over time. The average current drawn by a CMOS digital gate embedded within a design, which determines collectively with currents drawn by other gates in the design the average current flow in the lines. The average current that flows in the devices within a gate, determines the HCE on the device. The average current drawn by a gate depends on the switching frequency of its output signals under expected operating conditions, as well as device characteristics, loading, and gate input slew rates. A first step in estimating the average current flow in a gate is thus an accurate estimation of the switching frequency of the signal at its output.

Most existing current estimation techniques, including our previous work, have been limited to combinational circuits, while practical designs are sequential. Our approach, which we have described in this report, considers synchronous sequential circuits, and applies a combination of statistical and probabilistic techniques to estimate the current in a hierarchical fashion. First, simulation-based statistical Monte Carlo techniques are applied to a functional or RTL-level description of the design to obtain the statistics of the signals at the outputs of the latches. These statistics are then used to estimate the currents drawn by the latches themselves as well as form inputs to the combinational blocks. The combinational blocks are then analyzed using probabilistic techniques with more detailed models of the circuit, including device characteristics, loading and delay calculation.

A number of new techniques have been developed as part of our approach. First, the convergence limits of the Monte Carlo are preset based on user-specified level of confidence and error bound criteria. Second, in order to determine when convergence occurs, the design is simulated starting from two different initial states and the signal statistics are monitored as the simulation progresses until they converge to each other. This convergence detection criterion is based on the fact that the final signal statistics are independent of the initial states. Third, to speed up convergence, filtering techniques are applied to remove noise in the results as the simulation is progressing. Moreover, parallel simulation techniques are used to speed up the simulation. As a result of all these techniques, we have been able to analyze circuits containing up to 20,000 gates with more than 1400 latches in less than 5 hours on a SUN SPARC10 workstation. However, we have observed that in all the circuits we have simulated, very few latches have very low switching activity, and thus slow down the convergence unnecessarily. By applying a different, more relaxed convergence criterion to the low activity latches, our preliminary results show that we can reduce simulation time for the largest circuit from 5 hours to less than 30 minutes. We are currently investigating this selective convergence criterion in more detail.

Currently, the high-level Monte Carlo statistical simulator and the probabilistic combinational circuit simulator are separate and not directly linked. We are in the process of combining the simulators into one, so that the user will not have to perform two separate simulation runs.

The second task we have included in this report is circuit design for hot-carrier reliability. HCE in a device depends on its operation within a given design. Different devices within a design degrade at different rates, depending on their operating conditions. HCE-induced degradation in circuit performance, such as degradation in gate delay, increases at different rates in different gates, depending on their switching activities. We have applied the results of the current estimation task to estimate the HCE on the performance of combinational circuits embedded within a sequential design. In estimating HCE-induced degradation in a gate, device characteristics, input slew rate, and output loading, in addition to the switching frequency, are needed to obtain good estimation. The results have been implemented in iProbe-d.

The main thrust of the work on HCE described in this report is on developing design strategies. to reduce degradation effects on circuit performance. We have considered two design strategies: one at the transistor and subcircuit level and the other at the overall circuit timing level. At the subcircuit level, we developed a gate input signal line ordering technique to reduce HCE without increasing the area of the gate. This is done by connecting the input signal line with the highest switching activity to transistors that are connected away from the output mode of a gate, whenever possible.

At the circuit level, we consider synchronous sequential circuits where the combinational blocks are usually designed to meet certain timing specifications. In these circuits, the longest or critical paths in the combinational circuit blocks are designed to be less than the clock period. Timing/area optimization procedures are usually applied to accomplish this performance objective. In our work, we use the results of the probabilistic simulation results, together with available transistor HCE degradation models, to predict the degradation in gate delays and find their effects on the timing of the longest paths in a given design. A timing/area/reliability optimization procedure is then used to ensure that the design meets timing specifications in the long run, including expected HCE-induced degradation. Using this design approach, only a few gates lying along longest paths have to be redesigned to ensure that long-term timing constraints are being met. This is done with a minimal increase in area.

The device size we used in our CAD experiments was $1\mu m$, with a corresponding HCE-induced degradation model. In order to apply the performance/reliability optimization approach to state-of-the-art submicron designs, a submicron HCE degradation model similar to the one we used for the $1\mu m$ model is needed for use in the design optimization system, such as ours. In fact, with the push towards the deep submicron regime and with the continued shrinking of feature sizes, reliability at the device, circuit, and interconnect levels is becoming increasingly important. Our research direction is aimed towards design for reliability of deep submicron VLSI circuit designs where HCE, oxide breakdown, electromigration in metal lines, cross-talk and voltage-drop are becoming critical reliability issues to be addressed at the VLSI circuit level. In this respect, accurate three-dimensional extraction techniques have to be developed. Moreover, reliability estimation and design for reliability have to start at the architectural and functional levels before the actual physical level design is implemented. To accomplish this, high-level functional reliability modeling and estimation techniques have to be developed based on low-level circuit, device, and process reliability measures, and incorporated into the design at an early stage in the design process. This, of course, will not eliminate the need for design for reliability at the physical design level.

5 References

[1] F. Najm, R. Burch, P. Yang, and I. N. Hajj, "Probabilistic simulation for reliability analysis of CMOS VLSI circuits," *IEEE Transactions on Computer-Aided Design*, vol. 9, no. 4, pp. 439–450, April 1990 (Best Paper Award).

[2] F. N. Najm, I. N. Hajj, and P. Yang, "An extension of probabilistic simulation for reliability analysis of CMOS VLSI circuits," *IEEE Transactions on Computer-Aided Design*, vol. 10, no. 11, pp. 1372–1381, November 1991.

[3] P. C. Li, G. I. Stamoulis, and I. N. Hajj, "A probabilistic timing approach to hot-carrier effect estimation," *Int. Conf. on Computer-Aided Design (ICCAD-92)*, Santa Clara, CA, pp. 210–213, November 1992.

[4] G. Stamoulis and I. N. Hajj, "Improved techniques for probabilistics simulation including signal correlation effects," *30th ACM/IEEE Design Automation Conference*, Dallas, TX, pp. 379–383, June 1993.

[5] P.-C. Li, G. I. Stamoulis, and I. N. Hajj, "iProbe-d: A hot-carrier and oxide reliability simulator," *International Reliability Physics Symposium*, San Jose, CA, pp. 274–279, April 1994.

[6] G. I. Stamoulis and I. N. Hajj, "Slope considerations in probabilistic simulation," *Custom Integrated Circuits Conference*, San Diego, CA, pp. 425–428, May 1994.

[7] P.-C. Li, G. I. Stamoulis, and I. N. Hajj, "A probabilistic timing approach to hot-carrier effect estimation," *IEEE Transactions on Computer-Aided Design*, vol. 13, no. 10, pp. 1223–1234, October 1994.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

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

[13] 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.

[14] F. N. Najm, S. Goel, and I. N. Hajj, "Power estimation for sequential circuits," *ACM/IEEE Design Automation Conference (DAC'95)*, San Francisco, CA, pp. 635-640, June 1995.

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

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

[17] 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.

[18] 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.

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

[20] P. Yang and J.-H. Chern, "Design for reliability: the major challenge for VLSI," *IEEE Proceedings*, vol. 81, no. 5, pp. 730–744, May 1993.

[21] W. Weber, L. Risch, W. Krautschneider, and Q. Wang, "Hot-carrier degradation of CMOS-inverters," *IEDM Technical Digest*, pp. 208–211, 1988.

[22] W. Weber, "Dynamic stress experiments for understanding hot-carrier degradation phenomena," *IEEE Transactions on Electron Devices*, vol. 35, pp. 1476–1486, September 1988.

[23] C. Hu, S. Tam, F. C. Hsu, P. K. Ko, T. Y. Chan, and K. W. Terrill, "Hot-electron-induced MOSFET degradation - model, monitor and improvement," *IEEE Transactions on Electron Devices*, vol. ED-32, pp. 375–384, February 1985.

[24] B.S. Doyle and K.R. Mistry, "The characterization of hot-carrier damage in p-channel transistors," *IEEE Transactions on Electron Devices*, vol. ED-40, no. 1, pp. 152–156, 1993.

[25] W. Sun, E. Rosenbaum, and S.M. Kang, "Fast timing simulation for submicron hot-carrier degradation," *Proc. of International Reliability Physics Symposium*, pp. 65–71, 1994.

[26] R. Kao and M. Horowitz, "Timing analysis for piecewise linear rsim," *IEEE Transactions on Computer-Aided Design*, vol. 13, no. 12, pp. 1498–1512, December 1994.

[27] J. Qian, S. Pullela, and L. Pillage, "Modeling the effective capacitance for the RC interconnect of CMOS gates," *IEEE Transactions on Computer-Aided Design*, vol. 13, no. 12, pp. 1526–1535, December 1994.

[28] J. Rubenstein, P. Penfield, and M. Horowitz, "Signal delay in RC tree networks," *IEEE Transactions on Computer-Aided Design*, vol. CAD-2, pp. 202–211, July 1983.

[29] J. P. Uyemura, *Fundamentals of MOS Digital Integrated Circuits.* Addison Wesley, 1988.

[30] F. Hsu and S. Tam, "Relationship between MOSFET degradation and hot-electron-induced interface-state generation," *IEEE Electron Device Letters*, pp. 50–52, February 1984.

[31] C. Lombardi, P. Olivo, B. Ricco, E. Sangiorgi, and M. Vanzi, "Hot electrons in MOS transistors: lateral distribution of the trapped oxide charge," *IEEE Electron Device Letters*, pp. 215–217, July 1982.

[32] Y. Leblebici and S. M. Kang, "A one-dimensional MOSFET model for simulation of hot-carrier induced device and circuit degradation," *Proc. IEEE International Symposium on Circuits and Systems*, pp. 109–112, May 1990.

[33] N. Hedenstierna and K. O. Jeppson, "CMOS circuit speed and buffer optimization," *IEEE Transactions on Computer-Aided Design*, pp. 270–281, March 1987.

[34] H. J. Park, K. Lee, and C. K. Kim, "A new CMOS NAND logic circuit for reducing hot-carrier problems," *IEEE Journal Solid-State Circuits*, vol. 24, pp. 1041–1047, August 1989.

[35] T. Sakurai et al., "Hot-carrier generation in submicrometer VLSI environment," *IEEE Journal Solid-State Circuits*, vol. SC-21, pp. 187–192, February 1986.

[36] L. Bruni, G. Buonanno, and D. Sciuto, "Transistor stuck-at and delay faults detection in static and dynamic CMOS combinational gates," *Proc. IEEE International Symposium on Circuits and Systems*, pp. 431–434, May 1992.

[37] B.S. Carlson and C.Y.R. Chen, "Performance enhancement of CMOS, VLSI circuits by transistor reordering," *ACM/IEEE 30th Design Automation Conference*, pp. 361–366, June 1993.

[38] S.C. Prasad and K. Roy, "Circuit optimization for minimization of power consumption under delay constraint," *Proc. International Workshop on Low Power Design*, Napa Valley, CA, pp. 15–20, April 1994.

[39] C.H. Tan and J. Allen, "Minimizing of power in VLSI circuits using transistor sizing, input ordering, and statistical power estimation," *Proc. International Workshop on Low Power Design*, Napa Valley, CA, pp. 15–20, April 1994.

[40] A.L. Glebov, D. Blaauw, and L.G. Jones, "Transistor reordering for low power CMOS gates using an SP-BDD representation," *Proc. of International Symposium on Low Power Design*, Dana Point, CA, pp. 161–166, April 1995.

[41] T. I. Kirkpatrick and N. R. Clark, "PERT as an aid to logic design," *IBM Journal Research and Development*, pp. 135–141, March 1966.

[42] P. McGeer and R. K. Brayton, "Efficient algorithms for computing the longest viable path in a combinational network," *ACM/IEEE Design Automation Conference*, pp. 561–567, July 1989.

[43] Y. C. Ju and R. A. Saleh, "Incremental techniques for the identification of statically sensitizable critical paths," *ACM/IEEE Design Automation Conference*, pp. 541–546, June 1991.

[44] J.P. Fishburn and A.E. Dunlop, "TILOS - a posynomial programming approach to transistor sizing," *IEEE International Conference on Computer-Aided Design*, Santa Clara, CA, pp. 326–328, November 1985.

[45] J. Shyu, J.P. Fishburn, A.E. Dunlop, and A.L. Sangiovanni-Vincentelli, "Optimization-based transistor sizing," *IEEE Journal Solid-State Circuits*, pp. 400–409, April 1988.

[46] S.S. Sapatnekar, V.B. Rao, P.M. Vaidya, and S.M. Kang, "An exact solution to the transistor sizing problem for CMOS circuits using convex optimization," *IEEE Transactions on Computer-Aided Design*, vol. 12, no. 11, pp. 1621–1634, November 1993.

[47] S. Lin, M. Marek-Sadowska, and E.S. Kuh, "Delay and area optimization in standard-cell design," *ACM/IEEE Design Automation Conference*, pp. 349–352, June 1990.

[48] W. Chuang, S.S. Sapatnekar, and I.N. Hajj, "Timing and area optimization for standard-cell VLSI circuit designs," *IEEE Transactions on Computer-Aided Design*, vol. 14, no. 3, pp. 308–320, March 1995.

[49] P.-C. Li and I. N. Hajj, "Computer-aided redesign of VLSI circuits for hot-carrier reliability," *IEEE Transactions on Computer-Aided Design* (Accepted).

[50] M.R.C.M. Berkelaar, *Lp_Solve - Solve (Mixed Integer) Linear Programming Problem.* Eindhoven University of Technology, June 1992.

[51] F. Brglez and H. Fujiwara, "A neutral netlist of 10 combinational benchmark circuits and a target translator in Fortran," *Proc. IEEE International Symposium on Circuits and Systems*, June 1985.

[52] K.R. Mistry, R. Hokinson, B. Gieseke, T.F. Fox, R.P. Preston, and B.S. Doyle, "Voltage overshoots and N-MOSFET hot carrier robustness in VLSI circuits," *Proc. International Reliability Physics Symposium*, pp. 65–71, 1994.

[53] S. Devadas and S. Malik, "A survey of optimization techniques targeting low power VLSI circuits," *ACM/IEEE Design Automation Conference*, San Francisco, CA, pp. 242–247, June 1995.

☆U.S. GOVERNMENT PRINTING OFFICE: 1996-710-126-20234

MISSION
OF
ROME LABORATORY

Mission. The mission of Rome Laboratory is to advance the science and technologies of command, control, communications and intelligence and to transition them into systems to meet customer needs. To achieve this, Rome Lab:

a. Conducts vigorous research, development and test programs in all applicable technologies;

b. Transitions technology to current and future systems to improve operational capability, readiness, and supportability;

c. Provides a full range of technical support to Air Force Materiel Command product centers and other Air Force organizations;

d. Promotes transfer of technology to the private sector;

e. Maintains leading edge technological expertise in the areas of surveillance, communications, command and control, intelligence, reliability science, electro-magnetic technology, photonics, signal processing, and computational science.

The thrust areas of technical competence include: Surveillance, Communications, Command and Control, Intelligence, Signal Processing, Computer Science and Technology, Electromagnetic Technology, Photonics and Reliability Sciences.