Reliability Study of Combinational Circuits†

Edgar Holmann    G. Leonard Tyler    Ivan R. Linscott
Space, Telecommunications, and Radioscience Laboratory
Stanford University
Stanford, CA 94305-4055, USA

Abstract
An exact, practical implementation of reliability calculation for combinational circuits can be based on a hierarchical decomposition of the circuit into manageable sub-units, and construction of exact summary tables for each sub-unit. For a simple example of voting logic, this exact reliability analysis reaffirms that the reliability of the individual voter inputs is as important as the voter reliability.

1 Introduction
Reliability analysis of a circuit estimates the likelihood that a whole system operates correctly after a given period of time. An attractive technique of improving this reliability is through the use of hardware redundancy which ensures that correct outputs are produced even when some faults are present in the circuit structure. In these circumstances, redundancy masks faults so that many different fault combinations are not visible at the circuit's outputs.

To obtain a precise estimate of the reliability of circuits, it is desirable to model circuits at the transistor level and to consider all combinations of stuck-at and stuck-open faults in the analysis. Several investigators have noted that a stuck-at fault model is not sufficient to correctly model circuits, (Wadsack, 1978; Hajj and Saab, 1983; Bose et al., 1982; Galiay et al., 1979). For this purpose, we have developed a simulator for combinational circuits in Complementary Metal Oxide Silicon (CMOS) technology to quantify the reliability of circuits including all fault combinations noted above. This fault set is similar to the one implemented in the MOZART approach, (Cabodi et al., 1988).

Section 2 of this paper describes the process for the calculation of the reliability of combinational circuits. In Section 3, the simulation process of an implementa-

†This research was sponsored by the National Aeronautics and Space Administration under NASA Grant NGR-70249 and by the NASA Planetary Instrument Definition and Development Program under contract NASW 3989.

tion for this reliability calculation is considered. Section 4 studies the reliability of a few representative combinational circuits to illustrate the potential of this analysis and Section 5 concludes the paper.

2 Reliability Calculation of a Circuit
Calculation of the reliability of a combinational circuit gives a measure of the robustness of the design. The reliability estimate developed here does not include effects such as design flaws, fabrication defects, and software errors. Instead, our analysis is based on the premise that a correct, fully operational, design is available and that the probabilities of hardware faults developing some time in the future are known.

Two different approaches are considered for calculating the circuit reliability. The first makes use of the theorem of total reliability (Ash, 1972) to decompose the circuit into simpler blocks that are more easily analyzed. The following definition is introduced to aid in the declaration of the theorem.

Definition 1 Let A be a component of a system S and \( P_{S|A} \) be the probability of a working system S given that component A is working. Similarly, let \( P_{S|\bar{A}} \) be the probability of a working system S given that component A is not working.

The theorem then states
\[
P_S = P_A P_{S|A} + P_{\bar{A}} P_{S|\bar{A}}
\]

Example 1 To determine the reliability of the system shown in Figure 1, assume independence between all the blocks. The reliability \( P_S \) of system S is
\[
P_S = P_A P_{S|A} + P_{\bar{A}} P_{S|\bar{A}}
\]
\[
P_S = P_A P_C + P_{\bar{A}} P_B P_C
\]
\[
P_S = P_A P_C + (1 - P_A) P_B P_C
\]

Thus, a representative circuit is decomposed into elements whose probabilities and effects on the system's function are included in the analysis of the structure.
Figure 1: A simple reliability block diagram. This system \( S \) is composed of units \( A \), \( B \), and \( C \). Each of these units can be either operational or failed.

In the second approach, all states in the system are enumerated and analyzed individually. The system’s reliability is then calculated by adding the probabilities of all the states for which the system is operational.

**Example 2** Using Table 1 to classify all the states of the system shown in Figure 1, the system’s reliability is studied. Adding all the cases from this table that result in an operational system, we get:

\[
P_S = P_A P_B P_C + P_A P_B P_C + P_A P_B P_C
\]

which is the same result obtained before.

**Table 1: Tabular analysis of system.** For each valid state of the system, its probability and overall system state are listed. Units \( A \), \( B \), and \( C \) and system \( S \) are either working “\( \checkmark \)” or failed “\( \times \).”

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th>Probability of State</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>( P_A P_B P_C )</td>
<td>✓</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>×</td>
<td>( P_A P_B P_C )</td>
<td>×</td>
</tr>
<tr>
<td>✓</td>
<td>×</td>
<td>✓</td>
<td>( P_A P_B P_C )</td>
<td>✓</td>
</tr>
<tr>
<td>✓</td>
<td>×</td>
<td>×</td>
<td>( P_A P_B P_C )</td>
<td>×</td>
</tr>
<tr>
<td>×</td>
<td>✓</td>
<td>✓</td>
<td>( P_A P_B P_C )</td>
<td>✓</td>
</tr>
<tr>
<td>×</td>
<td>×</td>
<td>✓</td>
<td>( P_A P_B P_C )</td>
<td>×</td>
</tr>
<tr>
<td>×</td>
<td>×</td>
<td>×</td>
<td>( P_A P_B P_C )</td>
<td>×</td>
</tr>
</tbody>
</table>

Both of these approaches are conceptually simple, but unfortunately, the analysis of circuits with a large number of elements is cumbersome and impractical. However, a refinement to the second approach that uses a hierarchical analysis of circuits is effective in reducing the computational complexity making it practical to study large circuits, (Holmann et al., 1994). In this extended approach, circuits are subdivided into blocks comprising a few elements each. At the lowest level, the blocks are composed of a few transistors which are simulated exhaustively. In intermediate levels, the blocks are composed of a few sub-blocks. Statistical data gathered from the simulation of each block are used in the analysis of each level in the hierarchy to obtain exact reliability estimates for the level. Thus, using this hierarchical analysis, the exact reliability estimate can be obtained for complex circuits.

The main effect of this hierarchical analysis of circuits is the compression of all the states for each block in the analysis into a few classes that fully represent the reliability statistics of each block. When the different blocks are connected, each of these classes for each of the blocks are used to assess the reliability of the combined blocks rather than having to consider all states explicitly. Thus, the simulation complexity is reduced by orders of magnitude allowing for the analysis of all combinations of faults within a practical time period without the loss of precision.

The simulation process that implements this hierarchical analysis of CMOS circuits is described next.

### 3 Simulation Process

A switch level analysis of CMOS circuits with “stuck-at” and “stuck-open” faults is the basis of our simulation discussed in Section 2. To analyze the reliability of a circuit, it must first be represented hierarchically. The simulation time of each level in the hierarchy is directly related to the number of states in each of the blocks within each level. Thus, to reduce the total simulation time, the following should be minimized when the circuit is described: 1) number of transistors within each block, 2) number of different blocks in circuit, and 3) number of sub-blocks within each block. Of these, the first item has the highest influence on the overall simulation time.

Each block in the hierarchy is simulated by first ordering all its elements so that each signal in the structure is evaluated after all of its predecessors have been considered, (Becker et al., 1992). Through this ordering, elements closest to the primary inputs and farthest from the primary outputs are placed at the head of the simulation queue. Also, since no feedback paths exist in combinational circuits, a single simulation pass is sufficient to analyze each of the blocks in the circuit.

A program was implemented in C programming language to study the reliabilities of different circuits using this hierarchical method. This program, \( cre \), comprises 11K lines of code and is operational on DEC MIPS and DEC ALPHA architectures. The three distinct phases of the program — input, analysis, and output — are described in Sections 3.1–3.3. Section 3.4 discusses some techniques to speed up the analysis of circuits using a typical workstation environment.

#### 3.1 Circuit Description

Circuits are described hierarchically which simplifies both the description and the simulation of larger cir-
circuits. Two circuit description files are used by the simulator for the analysis: a fault and a model description files. The fault description file defines the circuit whose reliability is being studied and thus contains information for all the points that can fail in the circuit. The model description file describes the ideal logic behavior for each of the allowable input combinations. In addition, it indicates the matching relationship between the pins defined in both description files.

Five different elements are used for the circuit description: circuits, gates, summary tables, constants, and transistors. Circuits are composed of sub-circuits, gates, summary tables, and constants. Gates are composed of constants and transistors; gates would not be necessary if transistors were allowed in circuits. However, these are included in the current implementation to simplify the software coding. The only possible effect of this choice is that of increasing the number of hierarchical levels needed to describe a circuit.

The circuit structure is defined through pins which are used within circuits, gates, and summary tables. The other two elements, constants and transistors, have pointers to define their connectivity. Constants use a single pointer whereas transistors use pointers for each of their three terminals: gate, source, and drain.

3.2 Circuit Analysis

A circuit’s reliability is computed by comparing the behavior of a faulty circuit with a model circuit. This comparison exercises all input combinations for each of the valid states of the faulty circuit. Of all the different elements that are used in the description of circuits, two elements are most important: transistors and summary tables. Transistors are used to describe all combinational circuits and their behavior and reliabilities are simulated exactly. Summary tables are used to substitute blocks of combinational logic in the hierarchical decomposition of circuits.

Transistor Analysis

All simulation is carried out at the logic level using four logic values: 0, 1, Z, and X. A Z, or “high impedance,” state represents a (wire) node that is not being driven to either 0 or 1. An X, or “indeterminate,” state is a node that is not at any of the other three logic values. The indeterminate or undefined state is also reached when a node is being forced to both 0 and 1 simultaneously.

Transistors are represented by ideal switches that can be conducting or non-conducting. For an nMOS transistor, the non-conducting state is active when the gate is 0, Z, or X or when the transistor is failed open. For the conducting state either the gate is 1 or the transistor must be failed closed (shorted). Table 2 lists the behavior of the source and drain of a transistor. For pMOS transistors, the difference lies in the logical control of the gate terminal. In this case, a gate with a 1, Z, or X yields a non-conducting device whereas a gate with a 0 results in a conducting transistor.

Table 2: Transistor behavior. The states of the source and drain of transistors stay constant when the transistor is non-conducting. In the conducting state, the state of each node will change as indicated.

<table>
<thead>
<tr>
<th>Transistor state</th>
<th>previous source</th>
<th>drain</th>
<th>next source</th>
<th>drain</th>
</tr>
</thead>
<tbody>
<tr>
<td>non-conducting</td>
<td>( L_s )</td>
<td>( L_d )</td>
<td>( L_s )</td>
<td>( L_d )</td>
</tr>
<tr>
<td>conducting</td>
<td>0</td>
<td>( 0, Z )</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>conducting</td>
<td>0</td>
<td>( 1, X )</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>conducting</td>
<td>1</td>
<td>( 1, Z )</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>conducting</td>
<td>1</td>
<td>( 0, X )</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>conducting</td>
<td>( Z )</td>
<td>( L_d )</td>
<td>( L_s )</td>
<td>( L_d )</td>
</tr>
<tr>
<td>conducting</td>
<td>( X )</td>
<td>( 0, 1, Z, X )</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

Summary Table Analysis

The use of summary tables in the simulation process speeds up the analysis of the whole circuit. The advantage of using tables comes from the classification of all the original states of the circuit into four different classes. For each of the valid inputs to the circuit, the table gives the probability of each of the valid outcomes. Table 3 describes a summary set of statistics for a two-input one-output circuit.

Table 3: A summary table for a two-input one-output circuit. For each valid input case, the probability of each outcome being 0, 1, Z, and X is recorded.

<table>
<thead>
<tr>
<th>( I_1 )</th>
<th>( I_0 )</th>
<th>( \text{Outcome Probabilities of} )</th>
<th>( \text{Out} = 0 )</th>
<th>( \text{Out} = 1 )</th>
<th>( \text{Out} = Z )</th>
<th>( \text{Out} = X )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>( p_{00} )</td>
<td>( p_{10} )</td>
<td>( p_{20} )</td>
<td>( p_{30} )</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>( p_{01} )</td>
<td>( p_{11} )</td>
<td>( p_{21} )</td>
<td>( p_{31} )</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>( p_{02} )</td>
<td>( p_{12} )</td>
<td>( p_{22} )</td>
<td>( p_{32} )</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>( p_{03} )</td>
<td>( p_{13} )</td>
<td>( p_{23} )</td>
<td>( p_{33} )</td>
<td></td>
</tr>
</tbody>
</table>

In Table 3, only inputs of 0 and 1 are shown whereas actual tables also include Z and X as valid inputs. A complete table for a two-input one-output circuit would thus have sixteen rows and four columns. For our specific implementation, the outcomes are presented in increasing order of likelihood. The first case to be studied is the one most probably correct according to the original circuit.

3.3 Simulator Output

At the end of a given circuit simulation, the program generates statistics for each of the output pins of the circuit to characterize their behavior. Statistics can
also be generated for each different input combination that is analyzed. These statistics list the probabilities of each of the possible outcomes: correct operation, stuck-at-0, stuck-at-1, stuck-at-Z, and uncertain.

3.4 Optimizations

The cre program allows for the cycling of the states of the faulty circuit in two different ways. The first technique is based on a single inference ordered list derived from the circuit description that leads naturally into a parallel execution of the simulation. Thus, state i in the simulation can be reached by cycling the state of the circuit from state i−1 or by setting the state of the circuit directly.

The second technique is based on a double reference articulation of the states which allows for a computation that is incremental in accuracy. First the number of existent (out of all the possible) faults in the circuit is defined and then this number of faults is cycled through all the elements in the circuit. Thus, a full simulation first analyzes the fault-free state, then all the single-fault states, then all the dual-fault states, and so on until all the states have been considered. This approach allows for the early identification of critical elements for the reliability of the circuit.

The current implementation of the program takes on the order of 25μsec per transistor for each simulation cycle on a DEC 5000/25 with 24MB of RAM. Each cycle is specified by a state and an input combination.

4 Experimental Reliability Analysis

Three representative combinational circuits are studied to illustrate the use of this simulator as an effective tool for reliability analysis: 1) a standard CMOS two-input NAND gate; 2) a four-bit ripple adder modified from an LSI database, (LSI Logic, 1988); and 3) a subset of some highly redundant fault tolerant gates proposed by Takefuji and Ikeda (1980). An analysis of the latter two circuits that includes all fault combinations in all the internal elements is not possible without the use of a hierarchical approach as implemented in the simulator. All simulations are done on the DEC 5000/25.

4.1 Analysis of two-input NAND gate

A standard four-transistor CMOS NAND gate, implemented as shown in Figure 2, is analyzed for two different cases. Both cases consider failure modes for the three power sources: \( Vcc1 \), \( Vcc2 \), and \( Gnd \); and for two pins: the “Out” pin and the “I” pin that connects the two nMOS transistors. Of these two cases studied, the second one also considers failures for all the four transistors. The objective of this study is to assess the effectiveness of a pure stuck-at model versus an extended model that also includes transistor failures.

Table 4 lists the resulting probability estimates for the NAND gate. In this study, the operational probability of all elements with failure modes are set to a constant value given by \( p_{ok} \) and the failure modes of each node are uniformly distributed. Thus, for wire nodes, the stuck-at-0 and stuck-at-1 failure probabilities are set to \( p_{0} = p_{1} = \frac{1}{2}(1 - p_{ok}) \). Similarly, transistors are assigned probabilities \( p_{open} = p_{closed} = \frac{1}{2}(1 - p_{ok}) \), and sources are assigned \( p_{fail} = 1 - p_{ok} \).

Table 4: Reliability analysis of two-input NAND gate.

<table>
<thead>
<tr>
<th>Outcome</th>
<th>&quot;Out&quot; pin</th>
<th>Reliability estimates for</th>
</tr>
</thead>
<tbody>
<tr>
<td>Correct</td>
<td></td>
<td>( p_{ok} = .999 ) ( p_{ok} = .9999999 ) ( p_{ok} = .99999999 )</td>
</tr>
<tr>
<td>Stuck at 0</td>
<td>8.7462e-04</td>
<td>8.74996e-6</td>
</tr>
<tr>
<td>Stuck at 1</td>
<td>4.9957e-5</td>
<td>4.99994e-6</td>
</tr>
<tr>
<td>Stuck at Z</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Uncertain</td>
<td>6.2387e-4</td>
<td>6.24998e-6</td>
</tr>
</tbody>
</table>

b. All transistors can also fail.

Table 4 presents the effects that the probability values of each element of the NAND gate have on the overall reliability estimate. Decreasing the probability of failure of all the elements in the gate by a factor of 100 reduces each of the different failure modes by an equal factor. This result is true regardless of whether transistors can fail or not. When transistors are also allowed to fail, the stuck-at-0 and stuck-at-1 reliability estimates change by less than 0.02%: however, the
probability of an uncertain output increases by a factor of two and the probability of a high impedance output changes to a nonzero value. In general, a stuck-at failure model is not sufficient to correctly characterize the reliability of CMOS combinational circuits.

Additionally, note that reliability estimates based on fully operational circuitry are pessimistic since some fault combinations are not observable at the output pin for some of the input combinations. A detailed analysis as developed here accounts for such cases when estimating the reliability of the circuit.

### 4.2 Analysis of four-bit adder

The reliability of a four-bit ripple adder is analyzed to illustrate that summary tables are effective for calculating the reliability of larger circuits. Figure 3 depicts the single-bit full adder that is cascaded four times to generate the four-bit adder. This adder circuit comprises 200 transistors, 100 wires nodes, and 156 sources. Using CRE, all fault combinations of all transistors, wires, and sources are considered when the reliability of the adder is calculated. Again, three different sets of probabilities are used for all the elements. Figure 4 illustrates the reliability estimates for the adder with $p_{ok}$ set to 0.999, 0.99999, and then 0.9999999. The failure modes are presented in Section 4.1.

Similar observations to those made for the two-input NAND gate also apply to the four-bit adder. Namely, decreasing the probability of failure of each of the elements in the circuit by a factor of 10 reduces all the different failure modes by an approximately equal factor. Also, the probability of correct operation of each of the outputs is larger than the value obtained from just considering fully operational circuitry.

Using CRE we are able to consider the contribution of all the states of the four-bit adder ($\sim 3^{300} \times 2^{156} = 2^{631.49}$ states) for each of the 512 different input values in less than 6 hours of CPU time.

#### Figure 3: A one-bit full adder. Two one-bit variables and a carry-in are added to generate a sum and a carry-out. The carry-out is generated through the majority “MAJ” circuitry.

#### Figure 4: Reliability analysis of four-bit adder. The probability of the different outcomes for the sum and carry out of an adder are depicted. Each bar illustrates the lower bound reliability estimate calculated from a fully functional circuit and the actual reliability estimate calculated by the simulator. The reliability estimate improves by a factor of 100 when the individual element reliabilities are improved accordingly.

### 4.3 Analysis of Fault Tolerant Gates

Six different fault tolerant gates (FTGs) are simulated to characterize the robustness of their design for tolerating faults. All gates modeled have two-inputs and include: AND, NAND, OR, NOR, XOR, and XNOR gates. These gates are composed of a triplicated redundant structure that takes triplicated input signals and generates triplicated outputs. (Takefuji and Ikeda, 1980).

To simplify the analysis here, the outputs of the replicated redundant structure (RS) are collapsed into a single value with a voter (VT). Further, we will assume that only single input values are available so that each value drives all three lines in each RS. Each two-input FTG thus looks as shown in Figure 5.

#### Figure 5: Structure to Analyze Fault Tolerant Gates. The reliability of an FTG is studied by collapsing the triplicated outputs into a single value. The output pin is correct whenever two or more correct values are generated from the FTG.

The use of a voter to combine the three outputs of the FTG makes it straightforward to flag a correct
operation of the gate. When three values are available, as long as two or more of them yield the correct result, the circuit is operational. Since errors in the voter itself affect the overall circuit reliability, two sets of simulations are done. The first set of simulation uses a fault-free voter (VTOK) and the second set considers faults in all the elements of the voter (VTFL). Both sets of simulations include all faults in all the elements of the FTG structure. Figure 6 illustrates the reliability estimates for the six FTGs that are studied. The failure modes are presented in Section 4.1.

This analysis shows that the reliability of the FTGs are indeed very high. The values obtained are higher than those achievable from the standard gates that they replace. However, for the fault-set modeled here, the gain comes from the triplicated nature of the circuitry rather than from the suggested redundancy in the structure that replaces each of the gates. That is, the reliability of each RS is lower than the reliability of a standard gate. Once three RSs are combined, then a higher reliability is obtained. A further study that uses triplicated standard gates followed by a voter shows that even higher reliabilities than those achieved from the FTGs are possible. This is true regardless of whether the voter is fault-free or not.

Several factors that lead to these results that should be analyzed further include: 1) the use of a CMOS rather than an nMOS design, 2) the effect of triplicated inputs and how errors in the input pins affect the reliability estimates, and 3) the fault-set used. All these factors represent differences from the FTGs that were originally proposed.

5 Conclusion

A simulator has been developed to support the computation of reliability of combinational circuits of medium complexity. The effectiveness of the simulator in obtaining the reliability estimate was demonstrated through the analysis of representative combinational circuits. Based on our analysis method and results, the need for a model that goes beyond a stuck-at fault-set is affirmed.

By quantifying the reliability of a circuit, it becomes easier to compare the robustness of a design which will lead to the design of circuits with higher reliabilities. Further analysis that can be done with this tool is the study of the effect of assigning different reliabilities to each of the elements individually. Ultimately, the reliabilities of each element should be indicative of its expected physical behavior and this tool has the capability of allowing as many different probability values as desired.

Figure 6: Reliability analysis of Fault Tolerant Gates. Six FTGs are studied to analyze their reliabilities. Each bar illustrates the result corresponding to a fully operational voter and one that also exhibits faults. The inclusion of faults in the voter during the analysis has a higher consequence when the individual element reliabilities are higher.

References


