A Design Diversity Metric and Analysis of Redundant Systems

Subhasish Mitra, *Member*, *IEEE*, Nirmal R. Saxena, *Fellow*, *IEEE*, and Edward J. McCluskey, *Fellow*, *IEEE*

Abstract—Redundant systems are designed using multiple copies of the same resource (e.g., a logic network or a software module) in order to increase system dependability. Design diversity has long been used to protect redundant systems from common-mode failures. The conventional notion of diversity relies on "independent" generation of "different" implementations. This concept is qualitative and does not provide a basis for comparing the reliabilities of two diverse systems. In this paper, for the first time, we present a metric to quantify diversity among several designs and illustrate its effectiveness using several examples. Applications of this metric in analyzing reliability and availability of diverse redundant systems, and deriving simple relationships between diversity, system failure rate, and mission time are also demonstrated.

Index Terms—Error detection, design diversity, common-mode failures, fault-tolerant computing, dependability.

1 Introduction

The use of redundancy techniques for designing systems with high data-integrity and availability has been studied extensively [28], [37]. A duplex redundant system is an example of a classical redundancy scheme (Fig. 1). There are many commercial dependable systems from companies like Tandem, Stratus, and Sequoia using hardware duplication [28]. Hardware duplication is also used for the instruction fetch and execution units inside the IBM G5 processor [38], [45]. In a duplex system, there are two modules (shown in Fig. 1 as Module 1 and Module 2) that implement the same logic function. The two implementations can be the same or different. A comparator is used to check whether the outputs from the two modules agree. If the outputs disagree, the system indicates the presence of an error. *Data integrity* means that the system either produces correct outputs or generates an error signal when incorrect outputs are produced; in the literature on fault-tolerant computing, data integrity is also referred to as the *fault-secure property* [37]. For a duplex system, data integrity is guaranteed as long as only one module fails.

In a redundant system, *common-mode failures* (CMFs) occur when a single failure affects more than one module at the same time [11]. Examples of common-mode failure sources include electromagnetic coupling, power-supply disturbances, and radiation. Design mistakes in hardware and software systems can have very significant impact (catastrophes, major financial losses). There are many examples of design errors in hardware and software systems such as the floating point division bug in Intel's Pentium Chip [46] and the design error in Toshiba's floppy disk drive controller [47]. As pointed out in [3], although the use of redundant copies of hardware has proven to be quite effective in the detection of physical faults and subsequent system recovery, design faults are reproduced when redundant copies are made. Simple replication fails to enhance system reliability against design faults. A comprehensive survey and a characterization of common-mode failures is presented in [21].

Design diversity was proposed to protect redundant systems against common-mode failures. In [3], *design diversity* was defined as the independent generation of two or more software or hardware elements (e.g., program modules, VLSI circuit masks, etc.) to satisfy a given requirement. Design diversity was also proposed in [11] as an avoidance technique for common-mode failures. *N*-version programming [2], [15] is used to achieve diversity in software systems. Hardware design diversity is used in the Primary Flight Computer (PFC) system of Boeing 777 [30] and many other commercial systems [4]. For the Boeing 777, three different processors (from AMD, Intel, and Motorola) are used in the PFC. Tohma and Aoyagi proposed using the implementations of logic functions in true and complemented forms during duplication [42]. The use of a particular circuit and its dual was proposed in [40] to achieve diversity in order to provide protection from common-mode failures. The basic idea is that, with different implementations, common failure modes will probably cause different error effects. For example, chances of identical design errors may be minimized if two different groups of designers are asked to independently design a hardware block or a software module. A power supply dip may have different effects on two different hardware implementations of the same logic function.

While there is clear evidence that diversity can bring benefits in a redundant system, these benefits are extremely difficult to quantify with the above qualitative definition of diversity. Thus, as pointed out in [12], there is a need to answer questions such as: "what *is* diversity? Are *these* designs more diverse than *those*? *How* diverse are these two designs?" In order to quantify the effect of diversity on the reliability of a redundant system, a

• *The authors are with the Center of Reliable Computing, Computer Systems Laboratory, Department of Electrical Engineering and Computer Science, Stanford University, Room #236, MC9020, Gates Building 2A, 353 Serra Mall, Stanford, CA 94305. E-mail: {smitra, saxena, ejm}@crc.stanford.edu.*

Manuscript received 7 Jan. 2000; revised 10 May 2001; accepted 5 June 2001. For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number 111184.

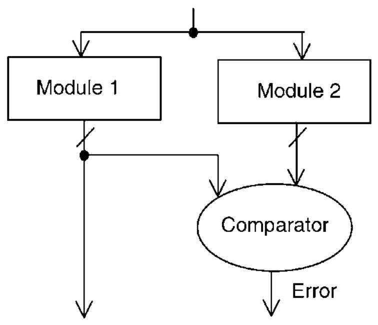

Fig. 1. A duplex redundant system.

metric is needed to *quantify diversity* among designs with the same specification [12], [40].

In addition to common-mode failures, with the high density of logic gates in a VLSI chip, multiple failures may become more frequent. For example, current research shows that multiple bit errors due to a single radiation source are common in VLSI chips [13], [29]. The classical reliability models for redundant systems are pessimistic because, in the presence of multiple module failures, they do not consider compensating effects of different faults [36]. It is interesting to find out whether design diversity also helps by achieving better compensating effects of different faults, compared to simple replication.

There are two cost components associated with diversity—design time and manufacturing cost. The design time is doubled for a duplex system design with diversity. The manufacturing cost can be avoided for systems built using reconfigurable logic elements (e.g., FPGAs). For these systems, diversity can be created by downloading different configurations instead of manufacturing different ASICs.

This work was done as part of the ROAR (Reliability Obtained by Adaptive Reconfiguration) project at the Stanford Center for Reliable Computing [33]. In the ROAR project, the system under consideration is reconfigurable and contains user-programmable logic elements (e.g., FPGAs). For such systems, faults can be detected during system operation using concurrent error detection (CED) techniques and the faulty element can be located [8]. Finally, the system can be reconfigured to operate without using the defective element [9]. Hence, for a reconfigurable system, the Field Replaceable Unit (FRU) is a programmable logic block used for implementing logic functions or a routing resource (switch module or programmable interconnect point), instead of a chip or a board as in conventional fault-tolerant systems.

Some preliminary ideas related to this work are reported in [20] and [32]. This paper's main contributions are: 1) developing a metric to quantify diversity among several designs and 2) using this metric to perform reliability and availability analysis of redundant systems. In Section 2, we introduce a design diversity metric and present reliability and availability analyses of redundant systems using this metric. We present simulation results in Section 3. Section 4 examines the effect of design diversity on the self-testing properties (the ability of a system to test itself) of a duplex system. Finally, we conclude in Section 5.

TABLE 1
Example to Illustrate Behavior of Faulty Implementations in a Duplex System (Incorrect Outputs Shown in Boldface)

Inputs	Correct Outputs	Faulty Outputs	
		N_1 (Fault f_1)	N_2 (Fault f_2)
0000	100	100	100
0110	010	**111**	010
1101	110	110	11**1**
1010	111	**0**11	1**0**1
1110	010	**1**10	**1**10

2 Design Diversity Metric and Analysis

2.1 *D*: A Design Diversity Metric

Assume that we are given two implementations (logic networks) of a logic function, a probability distribution for the primary inputs and faults f_1 and f_2 that occur in the first and the second implementations, respectively. The *diversity* $d_{i,j}$ *with respect to fault pair* (f_1, f_2) is the conditional probability that, with the faults f_1 and f_2 present, the two implementations do not produce identical errors.

Table 1 illustrates the main idea behind the above definition of $d_{i,j}$. We have two implementations N_1 and N_2 of the same logic function, and faults f_1 and f_2 affect N_1 and N_2, respectively. When the input combination 0000 is applied, both implementations produce correct outputs in the presence of the faults. When the input combination 0110 occurs, N_1 produces erroneous outputs while N_2 produces correct outputs in the presence of the faults. If a comparator is used to compare the outputs of N_1 and N_2, a mismatch will be reported and, hence, data integrity is guaranteed for this input combination. A similar situation occurs for the input combination 1101. For the input combination 1010, both N_1 and N_2 produce erroneous outputs in the presence of the faults. However, the erroneous outputs are different and a mismatch will be reported when the outputs of N_1 and N_2 are compared. Hence, data integrity is guaranteed for the input combination 1010. For the input combination 1110, both N_1 and N_2 produce identical erroneous outputs and the errors will not be detected by the comparator comparing the outputs of N_1 and N_2. Hence, data integrity is compromised for the input combination 1110.

The effects of the $d_{i,j}$ values of different fault pairs are combined to obtain a single number, D, for the diversity metric as described below. This enables easy comparison of the diversities of two diverse duplex systems.

For a given distribution function of different fault pairs, the *design diversity metric*, D, between two designs is the expected value of the diversity with respect to different fault pairs. Mathematically, we have

$$D = \sum_{(f_1,f_2)} P(f_1, f_2) d_{i,j},$$

where $P(f_1, f_2)$ is the probability of fault pair (f_1, f_2).

The fault model used in this paper is the stuck-at fault model which is widely used for digital testing of logic circuits [1]. In this model failures in a logic circuit behave as if some lines in the circuit assume constant

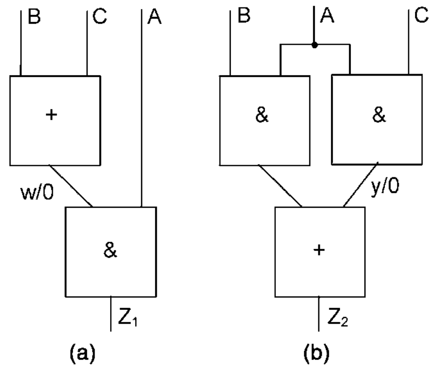

Fig. 2. Example of diversity.

logical values, either 1 or 0, independent of the logic values on other lines of the circuit. The stuck fault model is very effective in detecting defects for digital testing purposes [19]. For equivalence and dominance relationships among stuck-at faults in logic networks, please refer to [17] and [41].

For example, consider the two implementations of the logic function $Z = AB + AC$ shown in Fig. 2 with faults $f_1 = w$ stuck-at-0 (written as w/0) in the implementation of Fig. 2a and $f_2 = y$ stuck-at-0 (y/0) in the implementation of Fig. 2b. With f_1 present in N_1, the input combinations $ABC = 111$, 101, and 110 all produce errors at Z_1 (i.e., the outputs produced by the fault-free and the faulty circuits in response to these input combinations are different). A fault f_1 is detected by an input combination if and only if the outputs produced by the fault-free circuit and the faulty circuits in response to the input combination are different. Hence, input combinations $ABC = 111$, 101, and 110 detect f_1; it can be easily shown that no other input combination detects f_1 for the circuit in Fig. 2a. The only input combination that causes an error at Z_2 with f_2 present is $ABC = 101$. This is the input combination that detects f_2. If a duplex system consisting of the two implementations in Fig. 2 is affected by the fault pair (f_1, f_2), then $ABC = 101$ is the only input combination for which both implementations will produce identical errors. This erroneous output would escape detection. If we assume that all input combinations are equally likely, then the $d_{1,2}$ value for the fault pair (f_1, f_2) is $1 - \frac{1}{8} = \frac{7}{8}$.

Suppose that we are given two implementations (N_1 and N_2) of a combinational logic function with n inputs and a single output. The fault model is any fault for which a combinational circuit remains combinational in the presence of the fault.

The *joint detectability*, $k_{i,j}$, of a fault pair (f_1, f_2) is the number of input patterns that detect both f_1 and f_2. This definition follows from the idea of detectability developed by McCluskey et al. [18].

If all input combinations are equally likely, then $d_{i,j} = 1 - \frac{k_{i,j}}{2^n}$.

Next, we extend the above example to multiple-output combinational logic circuits. *For a fault pair (f_1, f_2) affecting the two implementations, we define $k_{i,j}$ as the number of input patterns in response to each of which both implementations produce identical errors*. Now, we can use the same formula as in the single output case.

TABLE 2
Behavior of Faulty Multiple Output Circuits

Inputs	Fault-free outputs	Faulty outputs (Implementation 1)	Faulty outputs (Implementation 2)
00	0 1	0 **0**	**1** 0
01	1 0	1 0	1 0
10	0 0	**1** 0	**1** 0
11	1 1	1 **0**	1 **0**

For a hypothetical combinational logic function with two inputs and two outputs (Table 2), suppose that faults f_1 and f_2 affect the first and the second implementations, respectively. The responses of the two implementations in the presence of the faults are shown in Table 2. The faulty output bits are highlighted in the third and fourth columns of Table 2. It is clear that for the calculation of $k_{i,j}$, we have to consider only the input patterns 10 and 11. Hence, $k_{i,j} = 2$ and $n = 2$ (i.e., $2^n = 4$) the value of $d_{i,j}$ is 0.5.

Fig. 3 illustrates the use of the $d_{i,j}$ values to evaluate diversity between different implementations of the same logic function. The following combinational functions are implemented: $W = A'C + BC$, $X = ABC$, $Y = BC$, and $Z = A'B + BC$. The three implementations—N_1, N_2, and N_3—shown in Figs. 3a, 3b, and 3c, respectively, use the same logic gates (AND gates $A'C$, ABC, BC, $A'B$, and two 2-input OR gates) but differ in the sharing of the logic gates among the output functions (also called fanout structures). Mitra showed that diversity in the fanout structure is important for generating diverse implementations of the same logic function [22].

We examine the following three duplex system designs:

1. A duplex system with two identical networks N_1 and N_1. Consider faults $m/1$ (m is the label of the node at the output of the AND gate ABC) in both. A CMF (e.g., a power supply dip) is expected to have identical effects on two identical designs; hence, the same fault is considered in both networks. In the presence of the m/1 fault in both implementations, the two (identical) implementations produce identical erroneous outputs for seven input combinations ($ABC = 000, 001, 010, 011, 100, 101, 110$). Hence, the $d_{i,j}$ value for this fault pair (m/1 in both implementations) is $1/8 = 0.125$ (assuming that all input combinations are equally likely).
2. The next duplex system includes networks N_1 and N_2 and is thus diverse. Consider faults $m/1$ (at the output of the AND gate ABC) in N_1 and $p/1$ (at the output of the AND gate ABC) in N_2. In the presence of this fault pair, the two implementations produce identical erroneous outputs for only two input combinations ($ABC = 010, 011$). Hence, in this case, the $d_{i,j}$ value of the fault pair (m/1 in the first network N_1 and p/1 in the second network N_2) is $6/8 = 0.75$.
3. The third duplex system is a diverse one using networks N_1 and N_3. There are faults $m/1$ (at the

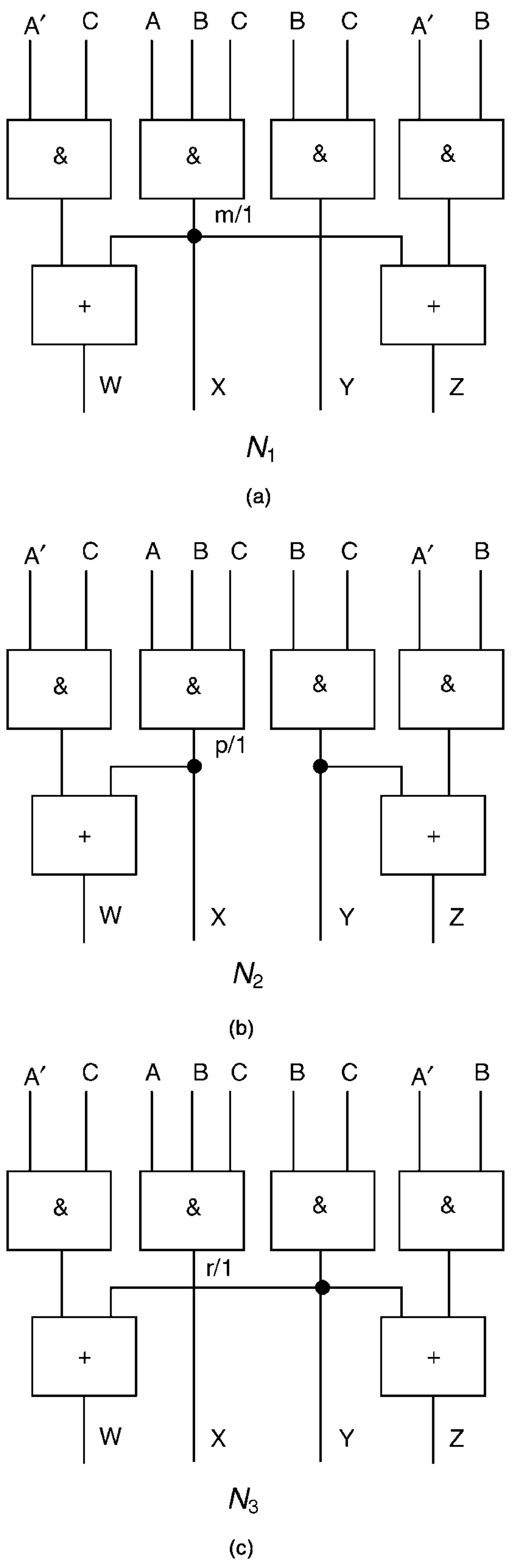

Fig. 3. Diverse implementations with diversity in fanout structures.

output of the AND gate ABC) in N_1 and $r/1$ (at the output of the AND gate ABC) in N_3. In the presence of this fault pair, the two implementations produce identical erroneous outputs for only one input combination ($ABC = 011$). Hence, in this case, the $d_{i,j}$ value of the fault pair ($m/1$ in N_1 and $r/1$ in N_3) is $7/8 = 0.875$.

The diverse duplex system design using N_1 and N_3 is better than the other two systems in the presence of single stuck-at faults. This fact can be verified by calculating the diversity metrics for these three systems. We calculated the diversity metrics for the above systems in two different ways:

a. If we assume that all possible single-stuck-at fault pairs are equally probable, then the values of the D metric for the first, second, and third systems are 0.96, 0.97, and 0.98, respectively. When we calculate the diversity metric assuming that all fault pairs are equally probable, we do not see a big difference because a large fraction of all fault pairs have the $d_{i,j}$ value equal to 1 for all the three systems. This is because there are very few (approx. 20-30 percent) fault pairs (f_1, f_2) such that f_1 and f_2 belong only to the same logic cone. Hence, the values of the D-metric become very close to 1 for all the three duplex designs.
b. For each single stuck-at fault f_1 in network N_1, we found the fault f_2 in the other network (N_1, N_2, or N_3) of the above duplex systems such that the $d_{i,j}$ value of the fault pair (f_1, f_2) is the minimum over all f_2s; hence, (f_1, f_2) is called a *worst-case fault pair*. These worst-case fault pairs were found through exhaustive simulation of all input combinations and all fault pairs. Finally, we calculated D as the average value of $d_{i,j}$s over the worst-case fault pairs. The D values obtained were 0.67, 0.72, and 0.75 for the first (N_1, N_1), second (N_1, N_2), and third scenarios (N_1, N_3), respectively.

The main reason behind why the third duplex system (with N_1 and N_3) is more diverse than the first duplex system (with N_1 and N_1), as indicated by the diversity metric D, is explained next. Fig. 4 shows the histogram of the worst-case fault pairs for the first (N_1 and N_1) and the third (N_1 and N_3) duplex systems. Along the X-axis we plot the number of input combinations for which the implementations produce identical errors and, hence, undetected errors. Along the Y-axis, we plot the number of worst-case fault pairs. Out of all worst-case fault pairs (f_1, f_2), we removed those fault pairs for which f_1 is equivalent to a stuck-at fault at the primary output of N_1. This is because, if f_1 is equivalent to a stuck-at fault on the primary output, the worst-case fault pair (f_1, f_2) will always have f_2 to be the same output stuck-at fault in the second implementation no matter how diverse the second implementation is.

As shown in Fig. 4, for an identical duplex system with N_1 and N_1 (the first duplex system), one out of eight input combinations produces undetected errors (so that data integrity is compromised) for seven worst-case fault pairs. For one worst-case fault pair, seven out of eight input combinations produce undetected errors. However, if the third duplex system with N_1 and N_3 is used, the diversity creates a remarkable improvement in the data integrity of

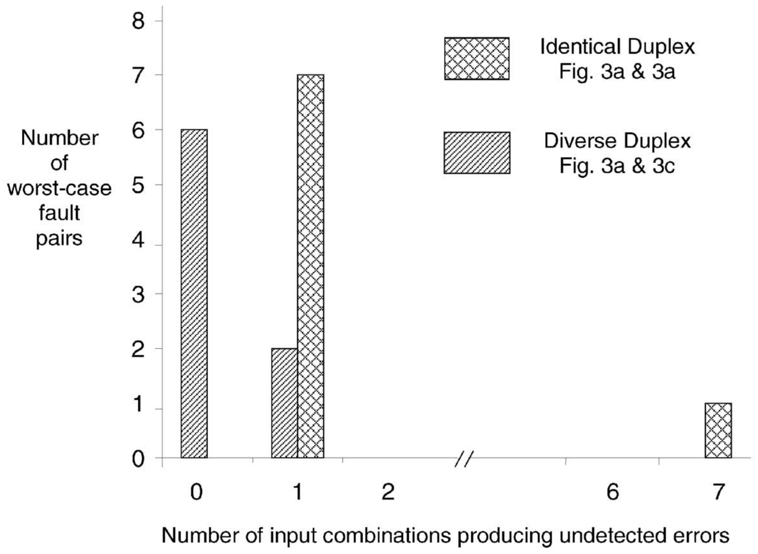

Fig. 4. Histograms illustrating the improvement in data integrity through design diversity.

the system. There is no worst-case fault pair for which seven out of eight input combinations produce identical errors. There are only two worst-case fault pairs (compared to seven in the first system) for which one out of eight input combinations produce undetected errors. Finally, there are six worst-case fault pairs for which data integrity is guaranteed because no input combination produces undetected errors.

As observed earlier, for two identical implementations of the same function, a common-mode failure (e.g., a design mistake or a power-supply dip) can be modeled as the same fault f_1 affecting the two implementations. This is because it is very likely that a failure generated by a common source will have identical effects on the two identical implementations. Let M be the number of input sequences for which f_1 produces erroneous values and, hence, both implementations produce identical error patterns at the outputs. If the second implementation is different from the first, for any fault f_2 affecting the second implementation (and f_1 affecting the first implementation), we cannot have more than M input sequences that produce identical error patterns at the outputs of the two implementations. Hence, $d_{i,i} \leq d_{i,j}$.

There are two inherent difficulties in the computation of the design diversity metric: 1) there can be very many fault pairs and 2) the problem of computing the $d_{i,j}$ value for a fault pair is NP-complete. Fast techniques for estimating diversity using a reduced list of fault pairs and an adaptive Monte-Carlo simulation technique are described in [25]. The adaptive Monte-Carlo simulation technique allows us to estimate the $d_{i,j}$ values with bounded error in the estimation; the error bound can be tuned depending on the required accuracy and available computation time. In our example of the calculation of diversity for combinational logic circuits, we assumed that all input combinations are equally likely. If we have information about the relative frequencies of various input combinations (in the form of input traces, for example), then we can incorporate this extra information (by changing weights associated with the input combinations) while calculating the value of the design diversity metric.

The design diversity metric can be extended to sequential circuits [24] and software programs used to detect hardware failures [27]. For diverse software programs used to detect or tolerate software faults or design mistakes, the idea of the design diversity metric can be used as long as we have a fault model available. There are several fault-injection techniques available in the literature [5], [10] that inject software faults in a system; hence, our diversity metric can be used in the context of these software faults.

2.2 Reliability Analysis

In this section, we calculate the reliability of duplex systems using the diversity metric described in Section 2.1. The *reliability of a duplex system* at time t is defined as the probability that the duplex systems does not produce undetected errors up to time t. Since a duplex system will be fault-secure at time t if it does not produce undetected errors up to time t (as explained in Section 1), the reliability of a duplex system at time t is also the probability that the system is fault-secure up to time t. The reliability calculation is independent of whether the redundant components are exact replicas or different implementations. For the ease of analysis, we assume a discrete time model for the system. In such a model, the time axis is broken up into discrete time cycles and we apply inputs and observe outputs at the cycle boundaries.

As shown in Fig. 5, input combination v_i is applied at the beginning of the *i*th cycle. Also, in Fig. 5, one of the modules of a duplex system becomes faulty (f_1) during cycle i and the other module of the same duplex system becomes faulty (f_2) during cycle j. We assume that the fault effects are permanent. *Let p be the probability that a particular module is affected by a fault at any cycle.* For simplicity, we assume that this probability p is the same for both modules of the duplex system at all times which is true if the sizes of the two

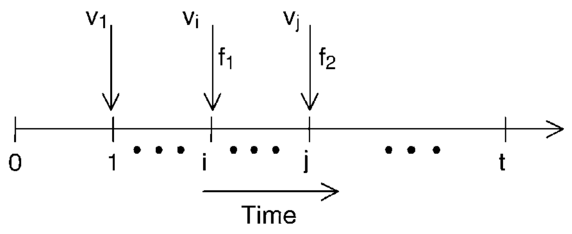

Fig. 5. A discrete time model of the system.

modules are comparable. The probability p can be looked upon as the failure rate per cycle.

For a given fault pair (f_1, f_2), there are two possible cases. In the first case, both faults appear in the same cycle. The situation is shown in Fig. 6.

In Fig. 6, faults f_1 and f_2 affect Modules 1 and 2, simultaneously at cycle i. At time 0, everything is fault-free. So, before cycle i the system will produce correct results. However, starting from cycle i, in each cycle, the system data integrity will be preserved with the probability equal to $d_{1,2}$. *The probability $s_1(f_1, f_2, t)$ that the system is fault-secure up to time t, even in the presence of the two faults f_1, f_2* is given by:

$$s_1(f_1, f_2, t) = p^2 d_{1,2} \frac{[d_{1,2}^t - (1-p)^{2t}]}{[d_{1,2} - (1-p)^2]}. \quad (1)$$

The derivation of the above expression is shown in the appendix. Next, we consider the case where f_1 and f_2 appear at different cycles.

As discussed earlier, in Fig. 5, Module 1 becomes faulty during cycle i and Module 2 becomes faulty during cycle j. It is clear that up to time j, a duplex system will be fault-secure. Hence, starting from time j, the system will be fault-secure with probability $d_{1,2}$. Thus, the probability $s_2(f_1, f_2, t)$ that the system is fault-secure up to time t, in the presence of the two faults f_1 and f_2 (affecting the system at different cycles) is given by the following equation:

$$\begin{aligned} s_2(f_1, f_2, t) = & \frac{2}{(d_{1,2} - 1 + p)}(1-p)p^2 d_{1,2}^2 \frac{[d_{1,2}^{t-1} - (1-p)^{2t-2}]}{[d_{1,2} - (1-p)^2]} \\ & - \frac{2}{(d_{1,2} - 1 + p)}(1-p)^t p d_{1,2}[1 - (1-p)^{t-1}]. \end{aligned} \quad (2)$$

The derivation of the second case is also shown in the appendix. This case is more complicated than the first case

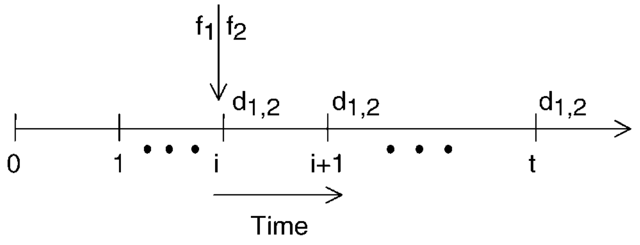

Fig. 6. Faults affect the modules simultaneously.

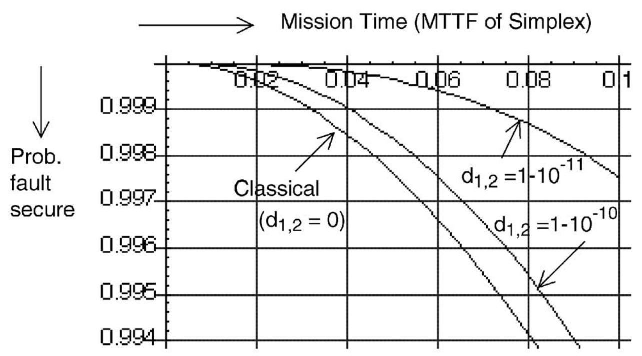

Fig. 7. Fault-secure probability of duplex systems with multiple independent failures.

and is useful when we consider random independent faults in multiple modules. We have:

$$s(f_1, f_2, t) = s_1(f_1, f_2, t) + s_2(f_1, f_2, t). \quad (3)$$

Here, *$s(f_1, f_2, t)$ is the probability that a duplex system is fault-secure up to time t, when Module 1 is affected by fault f_1 and Module 2 by fault f_2.*

We can characterize a duplex system using our diversity metric. In the following calculations, we assume that once a module becomes faulty, no other fault appears in that module. This assumption is simplistic and allows us to obtain closed-form reliability expressions. We calculate the probability that, up to time t, a duplex system is fault-secure. It is given by the following expression:

$$(1-p)^{2t} + 2(1-p)^t[1 - (1-p)^t] + \sum_{f_1, f_2} P(f_1, f_2) s(f_1, f_2, t). \quad (4)$$

The above expression follows from the fact that, in a duplex system, when none of the modules fails, the system produces correct outputs. When only one of the modules fails (due to single or multiple faults), the system is fault-secure. When both modules are faulty, then we have to consider the $d_{1,2}$ value for the fault pair (f_1, f_2) in the two modules. $P(f_1, f_2)$ is the probability of the fault pair (f_1 and f_2).

In Fig. 7, for a given pair of faults (f_1, f_2), we show the plots of (4) for different values of $d_{1,2}$. The mission time is shown along the X-axis—the MTTF (Mean Time To Failure in cycles) of a simplex system (a system with a single module) corresponds to one time unit. The probability (p) that a fault appears in one cycle (of duration 10^{-8} sec. for a 100 MHz system) is 10^{-12}. Along the Y-axis, we show the probability that the duplex system is fault-secure. It is clear from Fig. 7 that the classical analysis of duplex systems is pessimistic since it assumes that the system ceases to be fault-secure when two modules are faulty.

The above expressions can be modified for common-mode failures (CMF). The probability that a duplex system is fault-secure against common-mode failures up to time t, is given by the following expression:

$$(1-p)^t + \sum_{f_1, f_2} P(f_1, f_2) z(f_1, f_2, t). \quad (5)$$

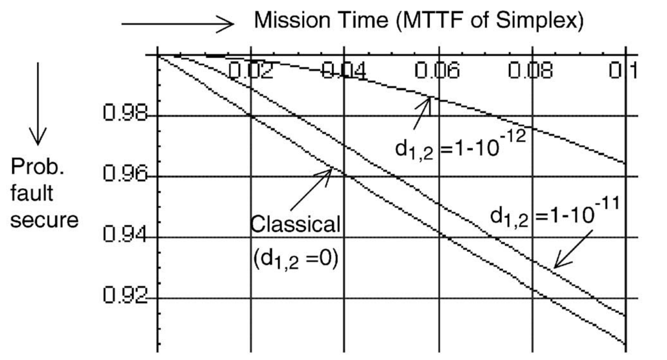

Fig. 8. Fault-secure probability of a duplex system for common-mode failures.

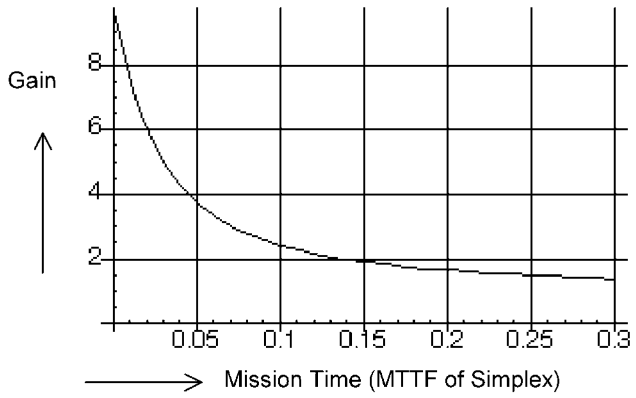

Fig. 9. Effect of diversity with mission time (for common-mode failures).

In the above expression, p is the probability that a CMF affects the two modules at any cycle. p can be looked upon as the CMF rate per cycle. In the above expression, $z(f_1, f_2, t)$ is given by the following formula:

$$z(f_1, f_2, t) = pd_{1,2}\frac{[d_{1,2}^t - (1-p)^t]}{[d_{1,2} - (1-p)]}. \quad (6)$$

The above expression is maximized when $d_{1,2}$ is greater than or equal to $(1-p)$. This suggests that, for a common-mode failure that can be modeled as fault pair (f_1, f_2), we can obtain appreciable reliability improvement over classical systems when the value of $d_{1,2}$ is by greater than or equal to $(1-p)$. The following observations can be derived from this relationship.

1. When the CMF rate is high, even a small diversity can help enhance the system reliability over traditional replication. For example, if there is a design bug, the fault is always present and any diversity in the system will be useful.
2. If the CMF rate is low, then $d_{1,2}$ must be extremely high for appreciable reliability improvement over classical systems. As a limiting case, consider the situation when the CMF rate is 0. In that case, we don't need any diversity.

In Fig. 8, for a given pair of faults (f_1, f_2), we show the plots of (5) for the different values of $d_{1,2}$. The common-mode failure rate per cycle is 10^{-13}. This is a realistic failure rate especially for radiation environments [33]. It is clear that we obtain significant improvement in reliability (over classical analysis) when the value of $d_{1,2}$ is very high ($1-10^{-12}$ or more). When the value of $d_{1,2}$ is less than $1-10^{-12}$, we do not see significant reliability improvement.

In Fig. 9, we show how the reliability improvement obtained from diversity depends on mission time. On the Y-axis of the graph in Fig. 9, we plot the ratio of the following two quantities:

1. The probability that a duplex system is not fault-secure at time i, for a fault pair (f_1, f_2) with $d_{1,2} = 1 - 10^{-11}$.
2. The probability that a duplex system is not fault-secure at time i, for fault pair (f_1, f_2) with $d_{1,2} = 1 - 10^{-12}$. The failure rate per cycle is 10^{-13}.

We call this ratio the *gain*. On the X-axis, we plot the mission time. As Fig. 9 shows, the gain diminishes with longer mission times. This analysis allows us to derive relationships between the reliability of a duplex system, the diversity incorporated to protect the system against common-mode failures, and the mission time. The relationship between diversity and mission time can also be used to determine checkpoint intervals in a redundant system. For example, referring to Fig. 9, we can checkpoint the state of the system when the gain is close to 1. Thus, our design diversity metric is a very fundamental property and can be used to understand different trade-offs associated with the design of dependable systems using redundancy.

Next, we estimate the data corruption latency using our design diversity metric. Consider a duplex system with two implementations N_1 and N_2 of the same logic function. Let us suppose that the faults f_1 and f_2 affect the two implementations at cycle c. The *data corruption latency* is defined to be the number of cycles from c after which both implementations produce the same error pattern at the output. This idea is similar to that of error latency [34]. The probability that the data corruption latency is t $(t > 0)$ is given by: $d_{1,2}^{t-1}(1 - d_{1,2})$. Here, the assumption is that $d_{1,2}$ value is strictly less than 1. If the value of $d_{1,2}$ is equal to 1, then the data corruption latency is strictly infinity and is bounded by T, the mission time. The expected data corruption latency is given by the following formula:

$$\sum_{t,(f_1,f_2),d_{1,2}\neq 1} P(f_1, f_2)td_{1,2}^{t-1}(1 - d_{1,2}) + \sum_{(f_1,f_2),d_{1,2}=1} P(f_1, f_2)T. \quad (7)$$

In the above expression, t varies from 1 to infinity. From this expression, it is clear that for long mission times (i.e., large values of t), the probability value approaches 0 when the $d_{1,2}$ value for the fault pair is less than 1. Thus, the fault pairs which have their $d_{i,j}$ values equal to 1 (i.e., the compensating fault pairs) play a dominant role in determining the data corruption latency for long mission times. Simplification of the above expression produces the following expression for the expected latency of a duplex system in terms of the $d_{i,j}$ values of different fault pairs.

Expected data corruption latency =

$$\sum_{(f_1,f_2),d_{1,2}\neq 1} \frac{P(f_1,f_2)}{(1-d_{1,2})} + \sum_{(f_1,f_2),d_{1,2}=1} P(f_1,f_2)T. \tag{8}$$

Consider the case of design mistakes. For these cases, the fault is always present. The probability that a duplex system is fault-secure up to time t, in the presence of design mistakes, is:

$$\sum_{f_1,f_2} P(f_1,f_2)d_{1,2}^t. \tag{9}$$

Equation (9) can be derived from (5) by substituting $p = 1$. Thus, for design mistakes, for a given fault pair (f_1, f_2), the reliability of a duplex system increases with increasing values of $d_{1,2}$. This implies that, for design mistakes, diversity among the two implementations in a duplex system helps to increase the probability that the system is fault-secure.

The previous analysis was based on the fact that once a fault pair appears, it affects the system permanently. However, this is not true for transient faults which may affect the system only for a single cycle. For common-mode failures creating transient faults that affect the duplex system only for a single cycle, the probability that the duplex system is fault-secure up to time cycle t is given by the following expression:

$$(1 - p + p\sum_{(f_1,f_2)} P(f_1,f_2)d_{i,j})^t. \tag{10}$$

The above expression can be derived from the fact that the system data integrity is not compromised up to time cycle t if either a CMF creating a transient fault pair does not appear or the input combinations applied to the system in the presence of the transient faults do not produce undetected errors.

While diversity in hardware designs is the main focus of this paper, the above ideas can be extended to analyze diversity in software modules. Note that, our observations about the relationships between diversity, mission time, and failure rate still hold for software systems. One key feature of our analysis technique is that it is powerful, but at the same time, simpler than the models in [6], [16], and [43].

2.3 Availability Analysis

In this section, we perform availability analysis of duplex systems with repair capabilities using our diversity metric. For the purpose of our analysis, we assume that p is the probability that a (common-mode) failure affects the system at any cycle. The failure produces fault f_1 and f_2 affecting Module 1 and Module 2, respectively. In our analysis, we use the following quantities, as described below.

We define $t_{1,2}$ for fault pair (f_1, f_2) as the conditional probability that, with faults f_1 and f_2 present, the two modules do not produce any errors at their outputs.

The quantity $d_{1,2} - t_{1,2}$ for fault pair (f_1, f_2) is the conditional probability that, with faults f_1 and f_2 present, the two modules produce nonidentical errors at their outputs.

The Markov chain used for our analysis is shown in Fig. 10. In the Markov chain, the system starts at the *Good* state. As long as a CMF does not appear, the system remains in the *Good* state. However, as soon as a CMF resulting in fault pair (f_1, f_2) appears, the system goes to the *Faulty Correct* state. The probability that both modules produce correct outputs, in spite of the presence of the CMF, is $t_{1,2}$. The probability that the modules produce identical errors at their outputs is $1 - d_{1,2}$. Thus, with probability $d_{1,2} - t_{1,2}$, the modules produce nonidentical erroneous outputs—this means that the presence of the CMF is detected. Once the CMF is detected, the system enters the *Repair* state. We have assumed that the expected number of cycles required to repair the system is $\frac{1}{m}$. For modeling the repair operation, we could as well use a repair rate. However, in the context of reconfigurable systems, we can have bounds on repair time (given by the fault location time and the time to reload a new configuration), which we can use during the above Markov analysis. The *availability* is given by the probability that the system is in the *Good* or the *Faulty Correct* state. In the following graph (Fig. 11), we show the dependence of availability on the values of $d_{1,2}$ and $t_{1,2}$. This analysis implies the usefulness of diversity in enhancing the self-testing property and, hence, the availability of duplex systems. The analysis can be extended for other redundant systems, e.g., Triple Modular Redundant (TMR) or N-Modular Redundant (NMR) systems.

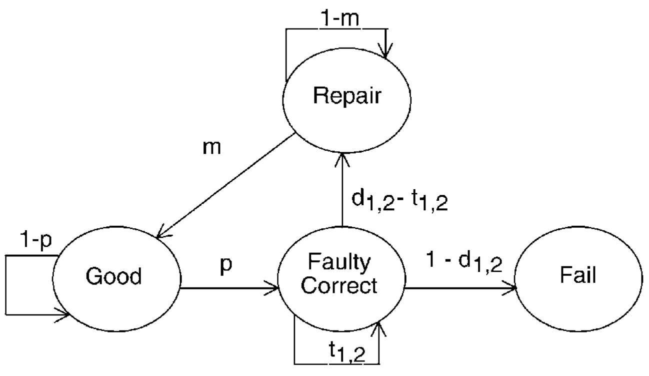

Fig. 10. Markov chain for availability analysis.

In Fig. 11, we plot the availability values for two duplex systems. The probability (p) that a fault pair appears in a particular cycle is 10^{-8}. The number of repair cycles ($\frac{1}{m}$) is 100. Both systems have the value of $d_{1,2}$ equal to $1 - 10^{-5}$ for a fault pair. However, one of the systems (shown in Fig. 11) has the value of $t_{1,2}$ equal to $d_{1,2}$ and the other one has the $t_{1,2}$ value equal to half of $d_{1,2}$. As can be seen in Fig. 11, initially, the system having $t_{1,2} = d_{1,2}$ has a higher availability (since the probability that it stays in the *Faulty Correct* state is high). However, as time increases, the availability of the system with $t_{1,2} = 0.5d_{1,2}$ decreases at a much smaller rate compared to the system with $t_{1,2} = d_{1,2}$. This is because the system with $d_{1,2}$ equal to $t_{1,2}$ never enters the Repair state in Fig. 9.

3 Simulation Results

It is difficult to completely model a complex system mathematically. Even with the stuck-at fault model, it is difficult to derive the exact reliability equation for the following reasons:

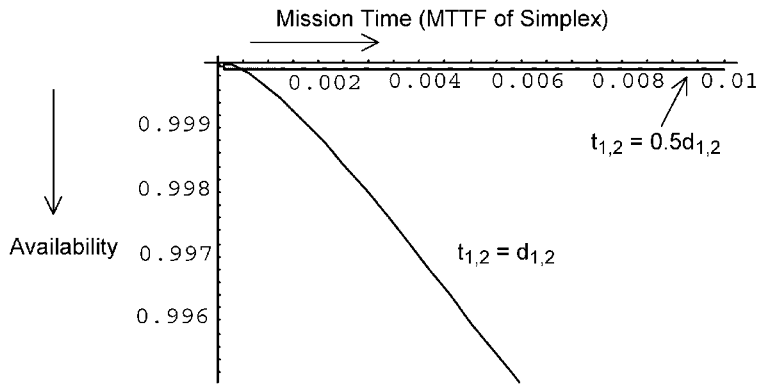

Fig. 11. Comparison of the availability of duplex systems.

1. For a given pair of faults (f_1, f_2), the calculation of $d_{1,2}$ is an NP-complete problem [7]. The problem is related to the NP-complete test generation problem.
2. If multiple stuck-at faults appear in the modules at different cycles, then the reliability expressions will become complicated. In fact, it may not be possible to obtain a closed form.

Hence, we developed a simulation environment to examine the reliability of a redundant system in the presence of multiple faulty modules.

For generating *different* designs, we minimized the truth tables corresponding to some MCNC benchmark circuits (*clip*, *inc*, *Z5xp1*, *apex4*, and *rd84*) using *espresso* available in *Sis* [35]. Then, we synthesized logic circuits after applying multilevel optimizations using the *rugged* script available in *Sis* [35]. Subsequently, we mapped the multilevel logic circuits to the LSI Logic G-10p technology library [14]. Next, we complemented the outputs in the truth tables of the benchmark circuits to generate new truth tables. We used the same synthesis procedure for these new truth tables. Finally, we added inverters at the outputs of the new designs obtained. Table 3 summarizes the characteristics of the different simulated designs.

In the fourth column of Table 3, we report the number of candidate single stuck faults for the implementations of the circuits, obtained by synthesizing the given specification. The fifth column shows the number of candidate single stuck faults for the implementations of the circuits, obtained by synthesizing the given specifications with complemented outputs.

3.1 Simulation 1

For Simulation 1, for each benchmark circuit, we designed duplex systems with identical and different implementations. For each of these systems, we performed 100,000 experiments. In each experiment, we randomly picked a single stuck-at fault pair (f_1, f_2) such that the fault f_1 affects Module 1 and f_2 affects Module 2. We injected these faults into the modules, applied input patterns from an LFSR (with random seed) and calculated the data corruption latency. The expected data corruption latency for the injected fault pairs is shown in Table 4. We also calculated the percentage of fault pairs for which the modules did not produce the same erroneous outputs at the same time (compensating fault pairs). These are the fault pairs (f_1, f_2) that have $d_{1,2}$ equal to 1.

As shown in Table 4, a duplex system consisting of different implementations of the *Z5xp1* circuit has a higher percentage of compensating fault pairs, compared to the nondiverse version—however, that is not generally true. For example, for the clip benchmark, the nondiverse duplex system has a higher percentage of compensating fault pairs. For compensating fault pairs, the data corruption latency is strictly infinity—we assumed the value to be 10,000 cycles for our experiments. This is because the number of inputs of the benchmark circuits under consideration lie between 7 and 9. Thus, the total number of input patterns is between 128 and 512. Note that, the expected data corruption latency is dependent on the number of compensating fault pairs. This dependence of data corruption latency on the number

TABLE 3
Characteristics of Simulated Designs

Circuit	# Inputs	# Outputs	# SSF (T)	# SSF (C)
Z5xp1	7	10	550	610
apex4	9	19	9636	8578
clip	9	5	698	664
inc	7	9	486	506
rd84	8	4	568	398

TABLE 4
Simulation 1 Results

Circuit Name	Copies	Data corruption latency (cycles)	% compensating fault pairs ($d_{i,j} = 1$)
Z5xp1	T, T	6733	66.96
	T, C	6869	68.76
apex4	T, T	8594	85.71
	T, C	8094	80.51
clip	T, T	7951	79.24
	T, C	7869	78.44
inc	T, T	7666	76.54
	T, C	7516	75.08
	C, C	7512	74.90
rd84	T, T	7638	76.23
	T, C	6797	67.73
	C, C	7051	70.40

TABLE 5
Simulation 2 Results

Circuit Name	Duplex Type	Worst data corruption latency (cycles)	
		Theoretical Prediction	Simulation
Z5xp1	T, T	10	13
	T, C	1711	1725
	C, C	14	12
Clip	T, T	35	42
	T, C	372	368
	C, C	48	45
Inc	T, T	16	19
	T, C	1645	1638
	C, C	17	15
rd84	T, T	35	40
	T, C	301	310
	C, C	21	25

of compensating fault pairs has been explained earlier in Section 2.1.

3.2 Simulation 2

Our previous simulation results mainly focused on independent faults in multiple modules of a duplex system. However, it has been observed in the literature [3], [11], that design diversity is useful for handling correlated failures and common-mode failures. Since we did not find any data on common-mode failure mechanisms, we performed the following sets of experiments to estimate the effect of diversity in the presence of common-mode failures.

In a duplicated system with identical implementations, we can find a one-to-one correspondence between the leads of the two logic networks. Hence, for these duplex systems, we injected fault pairs (f_1, f_2) such that f_1 and f_2 affect lead i of Module 1 and Module 2, respectively. Note that, in the presence of f_1 and f_2, the two modules behave exactly in the same way. Hence, they can be called common-mode faults. With this setup, we found the data corruption latency for these common-mode faults. For duplex systems with different implementations, we cannot establish such a one-to-one correspondence between the leads of the two copies. Hence, for each fault f_1 in Module 1, we found the fault f_2 in Module 2 with the minimum value of $d_{1,2}$ using exhaustive simulation of all input combinations. Thus, for f_1 affecting Module 1, we have the least data corruption latency when f_2 affects Module 2. Hence, the fault pair (f_1, f_2) is called the worst-case fault pair with the worst-case latency. Then, we averaged the worst-case latencies over all the worst-case fault pairs—this number is reported in the fourth column of Table 5. For the chosen benchmark circuits, it was possible to perform exhaustive simulation of all fault pairs and all input combinations.

The results in Table 5 show a distinct advantage of using different implementations over nondiverse designs for common-mode faults. This is because the worst-case data corruption latency of a common-mode fault in a duplex system with different implementations is at least an order of magnitude larger than the data corruption latency of a common-mode fault in a duplex system with identical implementations.

In order to bring into perspective the significance of this increased data corruption latency, we consider the execution of an application that uses the *Z5xp1* circuit of Table 5. If the mission time of the application is of the order of hundreds of cycles, then the system with two identical implementations will produce corrupt outputs *in the presence of CMFs*. However, a system with two different implementations of *Z5xp1* will preserve data integrity, on an average, *in the presence of CMFs*. Finally, if the mission time is of the order of thousands of cycles, then in the presence of CMFs, none of these systems will be able to preserve data integrity. This result can also be explained from the properties of the diversity metric discussed in Section 2.1. The relationship of this result with the CMF rate and mission time is explained in Section 2.2.

Suppose that we have a system for which the common-mode failures affect only the inputs. In this case, the duplex systems with different implementations are *not diverse so far as the inputs are concerned*. Thus, such systems do not provide any extra protection against the common-mode failures of interest (affecting only the inputs) compared to systems with identical implementations. This argument motivates research in developing common-mode fault models and designing redundant systems with sufficient diversity against the modeled common-mode faults [26].

In [31], for a given combinational logic function, the fault detectability profiles for different implementations have been reported. It has been proven in [41] that, for fanout-free combinational logic networks, all internal single stuck-at faults are either equivalent to [17] or dominate [41] single stuck-at faults on the primary inputs of the network. Thus, if we want to implement two diverse fanout-free networks implementing the same function, the $d_{i,j}$ values of the different fault pairs will be strongly dependent on the input combinations detecting the single stuck-at faults on network inputs and outputs. For both the networks, the set of patterns that detect the input or output stuck-at faults is independent of the network structure and is directly determined by the function the networks are implementing. Thus, chances are low that, for fanout-free networks, the diversity metric is going to achieve appreciable high values for networks synthesized in different ways, compared to simple replication. Hence, it is important to focus on achieving diverse fanout structures of different networks to obtain high values of the diversity metric for fault pairs. This point has also been highlighted in Section 2.1 (Fig. 3) and also in [22] where the diversity metric is used as a cost function for synthesizing diverse implementations of combinational logic circuits. There is further need to characterize common-mode failure mechanisms in the circuit level and develop CMF fault models. With a good CMF fault model, logical or layout-level design techniques (like the ones described in [26]) can be used to incorporate sufficient diversity to protect redundant systems from modeled CMFs.

4 Self-Testing Property

In this section, we discuss the possible effects of having design diversity on the self-testing properties of duplex systems. A duplex system is called *self-testing* with respect to a fault pair (f_1, f_2) (f_1 affecting Module 1 and f_2 affecting Module 2) if and only if there exists an input combination for which the two modules produce different outputs in the presence of the faults.

For the purpose of the experiment, we assume that failures manifest as single-stuck faults in each of the two

TABLE 6
Self-Testing Properties
of Diverse and Nondiverse Duplex Systems

Circuit Name	Copies	# SSF pairs	% escapes
Z5xp1	T, T	302,500	0.73
	C, C	372,100	0.65
	T, C	335,500	0.02
Clip	T, T	487,204	0.58
	T, C	463,472	0.02
Inc	T, T	236,196	0.73
	C, C	256,036	0.84
	T, C	245,916	0.03
rd84	T, T	322,624	0.74
	C, C	158,404	1.1
	T, C	226,064	0.04

modules under consideration. The self-testing property ensures that, in the presence of failures that affect the two modules under consideration, we can detect the presence of the failures. This detection is important for the system to take corrective action and directly affects the system availability as shown in Section 2.3. The fourth column of Table 6 shows the number of non-self-testable fault pairs in duplex systems with identical and different implementations (obtained through exhaustive simulation of all input combinations and all fault pairs). It is clear from Table 6 that with different implementations it is possible to achieve highly self-testable duplex systems for the simulated designs. This property is extremely useful if we apply specific patterns to the system during idle cycles in order to detect CMFs [23].

5 Conclusions

This paper demonstrates, for the first time, that it is possible to quantify diversity between different designs using a metric. This metric can be used to evaluate different dependability parameters such as reliability, availability, and data-integrity of a diverse redundant system. Our analysis using the diversity metric and simulation results show that there are situations when we can obtain significant improvement in dependability by using diversity in redundant systems. However, there are other situations in which the pay-offs of using diversity are not very significant. The design diversity metric and the analysis techniques presented in this paper help understand these cost and dependability tradeoffs while designing redundant systems with diversity.

Appendix

Reliability Calculation

In this section, we derive an expression for the probability that a duplex system is fault-secure up to time t when two modules are affected by faults f_1 and f_2, respectively. As explained in Section 3, there are two cases that must be considered. In the first case, the faults f_1 and f_2 appear simultaneously. Let p be the probability that a module is affected by a fault at any time cycle. Also, we assume that only a single stuck fault can affect a particular module. With these assumptions, the probability that the fault pair appears at time cycle i is given by the following expression:

$$(1-p)^{2(i-1)}p^2. \qquad (11)$$

The above expression follows from the fact that the two modules are fault-free up to time $(i-1)$ and become faulty at time cycle i. After the fault pair (f_1, f_2) arrives at time cycle i, the probability that the duplex system will produce correct outputs or indicate error signal (i.e., fault-secure) from cycle i up to cycle t is obtained by multiplying $d_{1,2}(t-i+1)$ times to obtain the following expression:

$$(1-p)^{2(i-1)}p^2 d_{1,2}^{t-i+1}. \qquad (12)$$

In the above expression, i can vary from 1 to t. Thus, we have the following summation:

$$\sum_{i=1}^{t}(1-p)^{2(i-1)}p^2 d_{1,2}^{t-i+1}. \qquad (13)$$

The above summation evaluates to the following expression (by summation of Geometric Progression series):

$$s_1(f_1, f_2, t) = p^2 d_{1,2}\frac{[d_{1,2}^{t} - (1-p)^{2t}]}{[d_{1,2} - (1-p)^2]}. \qquad (14)$$

Next, we consider the case where the faults f_1 and f_2 do not appear simultaneously. First, we consider the case where fault f_1 appears earlier. The probability that faults f_1 and f_2 appear at times i and j, respectively, $(j > i)$ is given by:

$$(1-p)^{i-1}(1-p)^{j-1}p^2. \qquad (15)$$

In a duplex system, as long as two modules are working correctly, correct outputs are produced by the system. However, once fault f_2 affects Module 2, the probability that the duplex system is fault-secure starting from cycle j to cycle t can be obtained by multiplying the above expression by $d_{1,2}$ $(t-j+1)$ times as shown below:

$$(1-p)^{i+j-2}p^2 d_{1,2}^{t-j+1}. \qquad (16)$$

Here, i can vary from 1 to $t-1$ and j can vary from $i+1$ to t. Thus, we have the following expression:

$$\sum_{i=1}^{t-1}\sum_{j=i+1}^{t}(1-p)^{(i+j-2)}p^2 d_{1,2}^{t-j+1}. \qquad (17)$$

In the above discussion, we assumed that f_1 appears before f_2. In order to consider the other case, we can multiply the above expression by 2 to obtain the following expression:

$$s_2(f_1, f_2, t) = \frac{2}{(d_{1,2}-1+p)}(1-p)p^2 d_{1,2}^2\frac{[d_{1,2}^{t-1} - (1-p)^{2t-2}]}{[d_{1,2} - (1-p)^2]} - \frac{2}{(d_{1,2}-1+p)}(1-p)^t p d_{1,2}[1-(1-p)^{t-1}]. \qquad (18)$$

$s(f_1, f_2, t) = s_1(f_1, f_2, t) + s_2(f_1, f_2, t)$ is the probability that the system produces correct outputs when Module 1 is affected by fault f_1 and Module 2 is affected by fault f_2.

We can extend the above derivations for common-mode failures. For that purpose, we assume that p is the probability that a given pair of modules get affected by a CMF at any cycle. Let f be a common-mode failure that affects both the modules such that faults f_1 and f_2 appear in Module 1 and Module 2, respectively. The probability that the CMF appears at time cycle i is given by the following expression:

$$(1-p)^{i-1}p. \tag{19}$$

After the CMF f arrives at time cycle i, the probability that the duplex system will be fault-secure from time cycle i up to time cycle t is obtained by multiplying $d_{1,2}(t-i+1)$ times to obtain the following expression:

$$(1-p)^{i-1}pd_{1,2}^{t-i+1}. \tag{20}$$

Here, i can vary from 1 to t. Thus, we obtain the following expression:

$$z(f_1, f_2, t) = pd_{1,2}\frac{[d_{1,2}^t - (1-p)^t]}{[d_{1,2} - (1-p)]}. \tag{21}$$

As mentioned in Section 2.2, since we are considering CMFs, we do not consider the case where f_1 and f_2 arrive at different times.

Acknowledgments

This work was supported by the US Defense Advanced Research Project Agency (DARPA) under Contract No. DABT63-97-C-0024 (ROAR project). The authors would like to thank Dr. Philip Shirvani of Stanford Center for Reliable Computing for his comments.

References

[1] M. Abramovici, M. Breuer, and A. Friedman, *Digital Systems Testing and Testable Design.* IEEE Press, 1990.

[2] A. Avizienis and L. Chen, "On the Implementation of N-Version Programming for Software Fault-Tolerance During Program Execution," *Proc. Int'l Computer Software and Applications Conf.,* pp. 149-155, 1977.

[3] A. Avizienis and J.P.J. Kelly, "Fault Tolerance by Design Diversity Concepts and Experiments," *Computer,* pp. 67-80, Aug. 1984.

[4] D. Briere and P. Traverse, "Airbus A320/A330/A340 Electrical Flight Controls: A Family of Fault-Tolerant Systems," *Proc. Int'l Symp. Fault-Tolerant Computing (FTCS),* pp. 616-623, 1993.

[5] J. Christmansson, M. Hiller, and M. Rimen, "An Experimental Comparison of Fault and Error Injection," *Proc. Int'l Symp. Software Reliability Eng.,* pp. 369-378, 1998.

[6] D.E. Eckhardt and L.D. Lee, "A Theoretical Basis for the Analysis of Multi-Version Software Subject to Coincident Failures," *IEEE Trans. Software Eng.,* vol. 11, pp. 1511-1517, Dec. 1985.

[7] M. Garey and D. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W.H. Freeman and Company, 1979.

[8] W.-J. Huang, S. Mitra, and E.J. McCluskey, "Fast Fault Location in Dependable FPGA Applications," *Proc. Int'l Symp. Defect and Fault Tolerance,* pp. 206-214, 2001.

[9] W.-J. Huang and E.J. McCluskey, "Column-Based Precompiled Configuration Technique for FPGA Fault Tolerance," *Proc. IEEE Symp. Field Programmable Custom Computing Machines,* 2001.

[10] J. Hudak, B. Suh, D. Siewiorek, and Z. Segall, "Evaluation and Comparison of Fault-Tolerant Software Techniques," *IEEE Trans. Reliability,* vol. 42, no. 2, pp. 190-204, June 1993.

[11] J.H. Lala and R.E. Harper, "Architectural Principles for Safety-Critical Real-Time Applications," *Proc. IEEE,* vol. 82, no. 1, pp. 25-40, Jan. 1994.

[12] B. Littlewood, "The Impact of Diversity Upon Common Mode Failures," *Reliability Eng. and System Safety,* vol. 51, no. 1, pp. 101-113, 1996.

[13] J. Liu et al., "Heavy Ion Induced Single Event Effects in Semiconductor Device," *Proc. Int'l Conf. Atomic Collisions in Solids,* 1997.

[14] *G10-p Cell-Based ASIC Products Databook.* LSI Logic, May 1996.

[15] M.R. Lyu and A. Avizienis, "Assuring Design Diversity in N-Version Software: A Design Paradigm for N-version Programming," *Proc. Int'l Conf. Dependable Computing for Critical Applications (DCCA),* pp. 197-218, 1991.

[16] M. Lyu, *Handbook of Software Reliability Engineering.* CS Press, 1995.

[17] E.J. McCluskey and F.W. Clegg, "Fault Equivalence in Combinational Logic Networks," *IEEE Trans. Computers,* vol. 20, no. 11, pp. 1286-1293, Nov. 1971.

[18] E.J. McCluskey, S. Makar, S. Mourad, and K.D. Wagner, "Probability Models for Pseudo-Random Test Sequences," *IEEE Trans. Computers,* vol. 37, no. 2, pp. 160-174, Feb. 1988.

[19] E.J. McCluskey and C.W. Tseng, "Stuck-Fault Tests vs. Actual Defects," *Proc. Int'l Test Conf.,* pp. 336-343, 2000.

[20] S. Mitra, N.R. Saxena, and E.J. McCluskey, "A Design Diversity Metric and Reliability Analysis for Redundant Systems," *Proc. Int'l Test Conf.,* pp. 662-671, 1999.

[21] S. Mitra, N.R. Saxena, and E.J. McCluskey, "Common-Mode Failures in Redundant VLSI Systems: A Survey," *IEEE Trans. Reliability* (special section on Fault-Tolerant VLSI Systems), vol. 49, no. 3, pp. 285-295, Sept. 2000.

[22] S. Mitra and E.J. McCluskey, "Combinational Logic Synthesis for Diversity in Duplex Systems," *Proc. Int'l Test Conf.,* pp. 179-188, 2000.

[23] S. Mitra, N.R. Saxena, and E.J. McCluskey, "Fault Escapes in Duplex Systems," *Proc. IEEE VLSI Test Symp.,* pp. 453-458, 2000.

[24] S. Mitra and E.J. McCluskey, "Design Diversity for Concurrent Error Detection in Sequential Logic Circuits," *Proc. IEEE VLSI Test Symp.,* pp. 178-183, 2001.

[25] S. Mitra, N.R. Saxena, and E.J. McCluskey, "Techniques for Calculating Design Diversity for Combinational Logic Circuits," *Proc. Int'l Conf. Dependable Systems and Networks,* pp. 25-34, 2001.

[26] S. Mitra and E.J. McCluskey, "Design of Redundant Systems Protected Against Common-Mode Failures," *Proc. IEEE VLSI Test Symp.,* pp. 190-195, 2001.

[27] N.S. Oh, S. Mitra, and E.J. McCluskey, "ED^{4}I: Error Detection by Diverse Data and Duplicated Instructions," *IEEE Trans. Computers,* vol. 51, no. 2, pp. 180-199, Feb. 2002.

[28] D.K. Pradhan, *Fault-Tolerant Computer System Design.* Prentice Hall, 1996.

[29] R. Reed et al., "Heavy Ion and Proton-Induced Single Event Multiple Upset," *IEEE Trans. Nuclear Science,* vol. 44, no. 6, pp. 2224-2229, July 1997.

[30] R. Riter, "Modeling and Testing a Critical Fault-Tolerant Multi-Process System," *Proc. Int'l Symp. Fault-Tolerant Computing,* pp. 516-521, 1995.

[31] J. Sakov and E.J. McCluskey, "Functional Test Pattern Generation for Random Logic," CRC Technical Report 87-1, Center For Reliable Computing, Stanford Univ., 1987.

[32] N.R. Saxena and E.J. McCluskey, "Dependable Adaptive Computing Systems," *Proc. IEEE Systems, Man and Cybernatics Conf.,* pp. 2172-2177, 1998.

[33] N.R. Saxena, S. Fernandez-Gomez, W.-J. Huang, S. Mitra, S.-Y. Yu, and E.J. McCluskey, "Dependable Computing and On-Line Testing in Adaptive and Reconfigurable Systems," *IEEE Design and Test of Computers,* vol. 17, no. 1, pp. 29-41, Jan-Mar. 2000.

[34] J.J. Shedletsky and E.J. McCluskey, "The Error Latency of a Fault in a Sequential Digital Circuit," *IEEE Trans. Computers,* vol. 25, no. 6, pp. 655-659, June 1976.

[35] E.M. Sentovich et al., "SIS: A System for Sequential Circuit Synthesis," ERL Memo. No. UCB/ERL M92/41, EECS, UC Berkeley.

[36] D.P. Siewiorek, "Reliability Modeling of Compensating Module Failures in Majority Voted Redundancy," *IEEE Trans. Computers,* vol. 24., no. 5, pp. 525-533, May 1975.

[37] D.P. Siewiorek and R.S. Swarz, *Reliable Computer Systems: Design and Evaluation.* Digital Press, 1992.

[38] L. Spainhower and T.A. Gregg, "S/390 Parallel Enterprise Server G5 Fault Tolerance," *IBM J. Research Development,* vol. 43, pp. 863-873, Sept.-Nov. 1999.

[39] C.E. Stroud, "Reliability of Majority Voting Based VLSI Fault-Tolerant Circuits," *IEEE Trans. VLSI,* vol. 2, no. 4, pp. 516-521, Dec. 1984.

[40] Y. Tamir and C.H. Sequin, "Reducing Common Mode Failures in Duplicate Modules," *Proc. Int'l Conf. Computer Design (ICCD),* pp. 302-307, 1984.

[41] K. To, "Fault Folding for Irredundant and Redundant Combinational Circuits," *IEEE Trans. Computers,* vol. 22, no. 11, pp. 1008-1015, Nov. 1973.

[42] Y. Tohma and S. Aoyagi, "Failure-Tolerant Sequential Machines with Past Information," *IEEE Trans. Computers,* vol. 20, no. 4, pp. 392-396, Apr. 1971.

[43] L.A. Tomek, J.K. Muppala, and K.S. Trivedi, "Modeling Correlation in Software Recovery Blocks," *IEEE Trans. Software Eng.,* vol. 19, no. 11, pp. 1071-1086, Nov. 1993.

[44] J. Von Neumann, "Probabilistic Logics and the Synthesis of Reliable Organisms from Unreliable Components," *Automata Studies, Annals of Math. Studies,* no. 34, pp. 43-98, 1956.

[45] C.F. Webb and J.S. Liptay, "A High Frequency Custom S/390 Microprocessor," *IBM J. Research and Development,* vol. 41, no. 4/5, pp. 463-474, 1997.

[46] "Intel Confirms Latest Pentium Glitch," *EE Times,* 10 Nov. 1997.

[47] "Vendor Settles Suit over Alleged Problems in Floppy Disk Drives," *PC World,* 10 Nov. 1999.

Subhasish Mitra received the BE degree (1994) in computer science and engineering from Jadavpur University, Calcutta, India, the MTech degree (1996) in computer science and engineering from the Indian Institute of Technology, Kharagpur, and the PhD degree (2000) in electrical engineering from Stanford University, Stanford, California. Professor E.J. McCluskey was his PhD thesis adviser. Dr. Mitra is currently working at Intel Corporation. He is also a consulting assistant professor at the Electrical Engineering Department of Stanford University and an assistant director of the Center for Reliable Computing (CRC), Stanford University. At Intel, Dr. Mitra works on Design for Testability, Reliability, and Manufacturability. At Stanford CRC, he supervises PhD students and is currently involved with the Stanford CRC test chip experiment project. Before that, he was the project leader of the Stanford CRC ROAR (Reliability Obtained by Adaptive Reconfiguration) project. Dr. Mitra also provides part-time consulting in various areas of VLSI design and test. During 2000-2001, he consulted at Agilent Technologies in their System Chip Testing project. He spent a summer at Ambit Design Systems (now part of Cadence Design Systems) to integrate a special synthesis algorithm developed by him into Ambit's BuildGates tool. Dr. Mitra's research interests include digital testing, fault-tolerant computing, VLSI synthesis, and computer architecture. He has published several papers in these areas in leading conferences and journals. He is also a coinventor of patents on VLSI synthesis algorithms, fault-tolerant computing and VLSI test. Dr. Mitra received gold medals for being the top student in the School of Engineering in the undergraduate and M. Tech levels. He is a member of the IEEE and the IEEE Computer Society.

Nirmal R. Saxena received the BE degree in electronics and communication engineering from Osmania University, India, in 1982, the MS degree in electrical engineering from the University of Iowa in 1984, and the PhD degree in electrical engineering from Stanford University in 1991. He is currently with Chip Engines, Inc., and is also a consulting faculty at the Center for Reliable Computing, Stanford University. His research interests include instruction and packet processing architecture, fault-tolerant computing, combinatorial mathematics, probability theory, and VLSI design/test. He is a fellow of the IEEE.

Edward J. McCluskey received the AB degree (summa cum laude, 1953) in mathematics and physics from Bowdoin College, and the BS (1953), MS (1953), and ScD (1956) degrees in electrical engineering from M.I.T. The degree of Doctor Honoris Causa was awarded in 1994 by the Institut National Polytechnique de Grenoble. He worked on electronic switching systems at the Bell Telephone Laboratories from 1955 to 1959. In 1959, he moved to Princeton University, where he was professor of electrical engineering and director of the University Computer Center. In 1966, he joined Stanford University, where he is professor of electrical engineering and computer science, as well as director of the Center for Reliable Computing. He founded the Stanford Digital Systems Laboratory (now the Computer Systems Laboratory) in 1969 and the Stanford Computer Engineering Program (now the Computer Science MS Degree Program) in 1970. The Stanford Computer Forum (an Industrial Affiliates Program) was started by Dr. McCluskey and two colleagues in 1970 and he was its director until 1978. Dr. McCluskey developed the first algorithm for designing combinational circuits—the Quine-McCluskey logic minimization procedure as a doctoral student at MIT. At Bell Labs and Princeton, he developed the modern theory of transients (hazards) in logic networks and formulated the concept of operating modes of sequential circuits. His Stanford research focuses on logic testing, synthesis, design for testability, and fault-tolerant computing. Professor McCluskey and his students at CRC worked out many key ideas for fault-equivalence, probabilistic modeling of logic networks, pseudoexhaustive testing, and watchdog processors. He collaborated with Signetics researchers in developing one of the first practical multivalued logic implementations and then worked out a design technique for such circuitry. Dr. McCluskey served as the first president of the IEEE Computer Society. His most recent honors include election to the National Academy of Engineering, 1998, the 1996 IEEE Emanuel R. Piore Award, and IEEE Computer Society Golden Core Member. In 1984, he received the IEEE Centennial Medal and the IEEE Computer Society Technical Achievement Award in Testing. In 1990, he received the EURO ASIC 90 Prize for Fundamental Outstanding Contribution to Logic Synthesis. The IEEE Computer Society honored him with the 1991 Taylor L. Booth Education Award. He is a fellow of the IEEE, AAAS, and ACM. He has published several books and book chapters as well as hundreds of papers. His most recent book is *Logic Design Principles with Emphasis on Testable Semicustom Circuits*, published in 1986 by Prentice-Hall. He has supervised 64 PhD students.