A DESIGN DIVERSITY METRIC AND RELIABILITY ANALYSIS FOR REDUNDANT SYSTEMS

Subhasish Mitra, Nirmal R. Saxena and Edward J. McCluskey

Center for Reliable Computing (http://crc.stanford.edu)
Departments of Electrical Engineering and Computer Science
Stanford University, Stanford, California

Abstract

Design diversity has long been used to protect redundant systems against 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 to compare the reliabilities of two diverse systems. In this paper, for the first time, we present a metric to quantify diversity among several designs. Based on this metric, we derive analytical reliability models that show a simple relationship between design diversity, system failure rate, and mission time. In addition, we present simulation results to demonstrate the effectiveness of design diversity in Duplex and Triple Modular Redundant (TMR) systems. For independent multiple-module failures, we show that, mere use of different implementations does not always guarantee higher reliability compared to redundant systems with identical implementations — it is important to analyze the reliability of redundant systems using our metric. For common-mode failures and design faults, there is a significant gain in using different implementations — however, as our analysis shows, the gain diminishes as the mission time increases. Our simulation results also demonstrate the usefulness of diversity for enhancing the self-testing properties of redundant systems.

1. Introduction

The use of redundancy techniques for designing fault-tolerant systems has been studied extensively [Siewiorek 92][Pradhan 96]. Triple-Modular-Redundancy (TMR) [Von Neumann 56] is an example of a redundancy scheme where three copies of a module are replicated and the outputs are processed by a voter. Figure 1.1 shows a classical TMR system. Such a system produces correct results as long as two modules produce correct results.

In a redundant system, *common-mode failures* (CMFs) result from failures that affect more than one module at the same time, generally due to a common cause. These include operational failures that may be due to external (such as EMI, power-supply disturbances and radiation) or internal causes. In addition to these common-mode failures, with the increasing complexity of the various designs, design mistakes are becoming very significant. It has been pointed out in [Avizienis 84] that design faults are reproduced when redundant copies are made. Thus, Simple replication fails to enhance the system reliability against design faults.

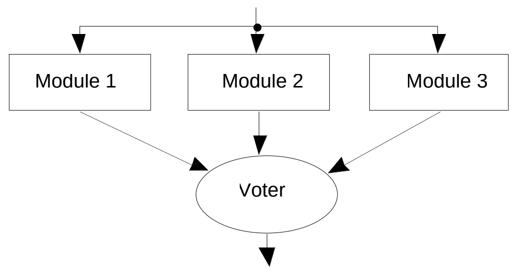

Figure 1.1. Classical Triple Modular Redundancy

Design diversity has been proposed in the past to protect redundant systems against common-mode failures. In [Avizienis 84], *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 [Lala 94] as an avoidance technique against common-mode failures. Design diversity has been applied to both software and hardware systems. *N*-version programming [Avizienis 77][Lyu 91] is an example of diversity in software systems. Hardware design diversity is used in the Primary Flight Computer (PFC) system of Boeing 777 [Riter 95] and many other commercial systems [Briere 93]. For the Boeing 777, three different processors (from AMD, Intel and Motorola) are used in the PFC. Tohma proposed to use the implementations of logic functions in true and complemented forms during duplication [Tohma 71]. The use of a particular circuit and its dual was proposed in [Tamir 84] to achieve diversity in order to handle common-mode failures. The basic idea is that, with different implementations, common-mode failures will probably cause different errors.

Design diversity can prove to be useful in the context of dependable *Adaptive Computing Systems* (ACS). The field programmability of Field Programmable Gate Arrays (FPGAs) can be utilized to achieve diversity among the different modules. In an ACS environment, we can create diversity by synthesizing and downloading different implementations into FPGAs at any time. Thus, there is no need to manufacture multiple diverse ASICs.

In order to quantify the effect of diversity on the reliability of a redundant system, a metric is needed to *quantify diversity* among designs with the same specification [Tamir 84].

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-event upsets (possibly due to a single radiation source) are common in VLSI chips [Liu 97][Reed 97]. The classical TMR reliability model is pessimistic because, in the presence of multiple module failures, it does not consider compensating effects of different faults [Siewiorek 75]. For the TMR system in Fig. 1.1, suppose that the output of Module 1 is stuck at 0 and the output of Module 2 is stuck at 1. Still, the system always produces correct outputs. This is an example of a *compensating module fault.* Earlier research in this area [Siewiorek 75][Stroud 94] has focused on predicting the effects of compensating faults on the reliability of a given TMR system. Hence, it is interesting to find out whether design diversity also helps in achieving better compensating effects of different faults, compared to simple replication.

In this paper, we address issues related to design diversity and examine their effects on the reliability of a redundant system. Some preliminary ideas related to this work were reported in [Saxena 98]. Our main contributions are: (1) develop a metric to quantify diversity among several designs; and (2) use this metric to perform reliability analysis of redundant systems. In Sec. 2, we introduce a design diversity metric and perform reliability analysis of redundant systems using this metric. Section 3 presents some preliminaries related to the stuck-at fault model and illustrates our analysis with the help of an example. We present simulation results in Sec. 4. Section 5 examines the effect of design diversity on the self-testing properties of a duplex system. Finally, we conclude in Sec. 6.

2. Design Diversity Metric And Reliability Analysis

2.1. *D*: A Design Diversity Metric

In this section, we introduce a metric to quantify diversity among several designs. We define the metric for a system with two designs implementing the same function. The metric has application in estimating the reliability of NMR systems with masking redundancy. Before defining the diversity metric, we first define the notion of diversity between two implementations with respect to a fault pair.

For two designs implementing the same function, the *diversity with respect to a fault pair* (f_i, f_j), $d_{i,j}$, is the probability that the designs do not produce identical error patterns, in response to a given input sequence, when f_i and f_j affect the first and the second implementations, respectively.

For a given fault model, the *design diversity metric, D*, between two designs is the expected value of the diversity with respect to different fault pairs. Mathematically, we have $D = \sum_{(f_i, f_j)} P(f_i, f_j) d_{i,j}$

D is the probability that, in response to a given input sequence, the two implementations either produce error-free outputs or produce *different* error patterns on their outputs.

Example: Consider any combinational logic function with n inputs and a single output. The fault model considered is such that a combinational circuit remains combinational in the presence of the fault. Now, let us consider two implementations (N_1 and N_2) of the given combinational logic function.

The *joint detectability,* k_{ij},, of a fault pair (f_i, f_j) is the number of input patterns that detect both f_i and f_j. This definition follows from the idea of detectability developed in [McCluskey 88].

If we assume that all the input patterns are equally likely, then we can write $d_{i,j} = 1 - \frac{k_{ij}}{2^n}$.

The $d_{i,j}$'s generate a diversity profile for the two implementations with respect to a fault model. Consider a duplex system consisting of the two implementations under consideration. In response to any input combination, the implementations can produce one of the following cases at their outputs. (1) Both of them produce correct outputs. (2) One of them produces correct output and the other produces incorrect output. (3) Both of them produce the same incorrect value.

For the first case, the duplex system will produce correct outputs. For the second case, the system will report a mismatch so that appropriate recovery actions can be taken. However, for the third case, the system will produce an incorrect output without reporting a mismatch — thus, for the third case, the *integrity* of the system is lost due to the presence of faults in the two implementations. In the literature on fault-tolerance [Siewiorek 92][Pradhan 96], this system integrity has been referred to as the *fault-secure* property.

The quantity $d_{i,j}$ is the probability that a duplex system, having two implementations of the logic function under consideration, is fault-secure when faults f_i and f_j affect the first (N_1) and the second (N_2) implementations, respectively.

If we assume that all fault pairs are equally probable and there are m fault pairs (f_i, f_j), then the D metric for the two implementations is: $D = \frac{1}{m} \sum_{i,j} d_{i,j}$.

We extend the above example to consider multiple-output combinational logic circuits. *For a fault pair (f_i f_j) affecting the two implementations, we define k_{ij} as the number of input patterns, in response to each of which,* *both the implementations produce the same erroneous output pattern*. Now, we can use the same formulas as the single output case.

Table 2.1. Behaviors 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 example, consider a combinational logic function with two inputs and two outputs (Table 2.1). Suppose that, faults f_i and f_j affect the first and the second implementations, respectively. The responses of the two implementations in the presence of the faults are shown in Table 2.1. The faulty output bits are highlighted in the third and fourth columns of Table 2.1. It is clear that for the calculation of k_{ij}, we have to consider only the input patterns 10 and 11.

The above illustration of the design diversity metric can also be extended to sequential circuits and software programs. For small or medium-sized systems, the exact value of the diversity metric can be calculated manually or using computer programs. For large systems, the value can be estimated by using simulation techniques.

For two identical implementations of the same function, a common-mode failure (e.g., a design mistake) can be modeled as the same fault f_i affecting the two implementations. Let m be the number of input sequences for which these two implementations produce identical error patterns at the outputs. Now, suppose that the second implementation is different from the first. For any fault f_j affecting the second implementation (and f_i 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, $d_{i,i} \leq d_{i,j}$. This property is useful for enhancing the reliability of a redundant system against common-mode failures by using diversity.

2.2. Reliability Analysis

In this section, we calculate the reliability of duplex systems using the diversity metric described in Sec. 2.1. We define the *reliability of a duplex system* as the probability that the system is fault-secure. The reliability calculation is independent of whether the redundant components are exact replicas or different implementations. 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 only at cycle boundaries.

As shown in Fig. 2.1, input combination (vector) v_i is applied at the beginning of the ith cycle. Also, in Fig. 2.1, the first system becomes faulty (f_1) during cycle i and the second system becomes faulty (f_2) during cycle j. *Let p be the probability that a particular module is affected by a fault at cycle i.* For simplicity, we assume that this probability p is the same for all the modules in the system at all times. The probability p can be looked upon as the failure rate per cycle.

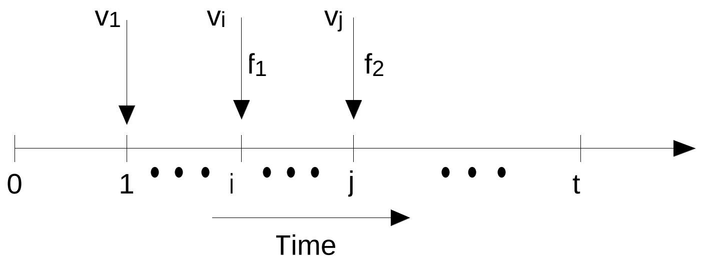

Figure 2.1. A discrete time model of the system

For a given fault pair (f_1, f_2), there are two possible cases. In the first case, both the faults appear in the same cycle. The situation is shown in Fig. 2.2.

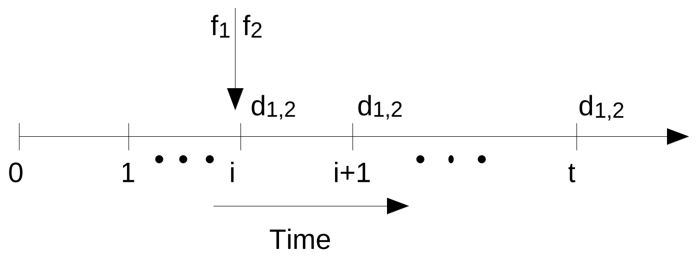

Figure 2.2. Faults affect the modules simultaneously

In Fig. 2.2, faults f_1 and f_2 affect modules 1 and 2, simultaneously at cycle i. It may be argued that if a random fault appears in a particular module, then chances are high that the second fault will also appear in that same module. However, we do not assume any such correlation in this paper. At time 0, everything is fault-free. So, before cycle i the system will produce correct results. However, starting from time i, in each cycle, the system will produce correct results 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 and f_2*, is given by:

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

The derivation of the above expression is shown in [Mitra 99]. Now we consider the case where f_1 and f_2 appear at different cycles.

As discussed earlier, in Fig. 2.1, 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 is given by the following equation.

$s_2(f_1,f_2,t)=$

$$\frac{2}{(d_{1,2}-1+p)}(1-p)p^2 d_{1,2}{}^2\frac{[d_{1,2}{}^{t-1}-(1-p)^{2t-2}]}{[d_{1,2}-(1-p)^2]}-\frac{2}{(d_{1,2}-1+p)}(1-p)^t p d_{1,2}[1-(1-p)^{t-1}]$$

The derivation of the second case is also shown in [Mitra 99]. This case is more complicated than the first case and is useful when we consider random independent faults in multiple modules. Now we have:

$$s(f_1, f_2, t) = s_1(f_1, f_2, t) + s_2(f_1, f_2, t)$$

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 now 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:

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

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 that faults f_1 and f_2 appear in modules 1 and 2, respectively.

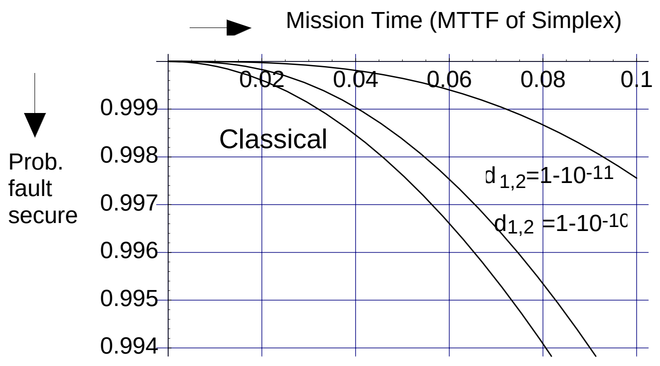

Figure 2.3. Fault-secure probability of a duplex systems with multiple independent failures

In Fig. 2.3, for a given pair of faults (f_1, f_2), we show the plots of the above expression 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 corresponds to 1 time unit. The probability that a fault appears in one cycle is 10^{-12}. Along the Y-axis, we show the probability that the duplex system is fault-secure. 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:

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

Here, p is the probability that a CMF affects the two modules. In the above expression, $z(f_1, f_2, t)$ is given by the following formula:

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

The derivation is shown in [Mitra 99]. The above expression is maximized when $d_{1,2}$ is of the order of (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 of the order of (1-p). The following observations can be derived from this relationship.

(1) When the failure rate is high, even a small diversity can help enhance the system reliability over traditional replication.

(2) If the failure 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 failure rate is 0. In that case, diversity will not buy us any extra reliability against CMFs.

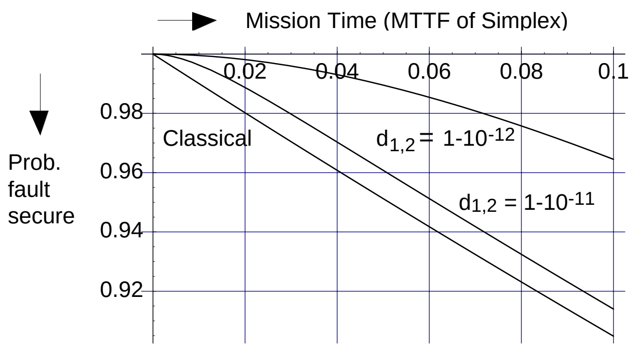

Figure 2.4. Fault-secure probability of a duplex system against common-mode failures

In Fig. 2.4, for a given pair of faults (f_1, f_2), we show the plots of the above fault-secure probability expression for the different values of $d_{1,2}$. The failure rate per cycle is 10^{-13}. It is clear that we get appreciable improvement in reliability (over classical systems) when the value of $d_{1,2}$ is very high ($1\text{-}10^{-12}$ or more). When the value of $d_{1,2}$ is less than $1\text{-}10^{-12}$, we do not see high reliability improvement.

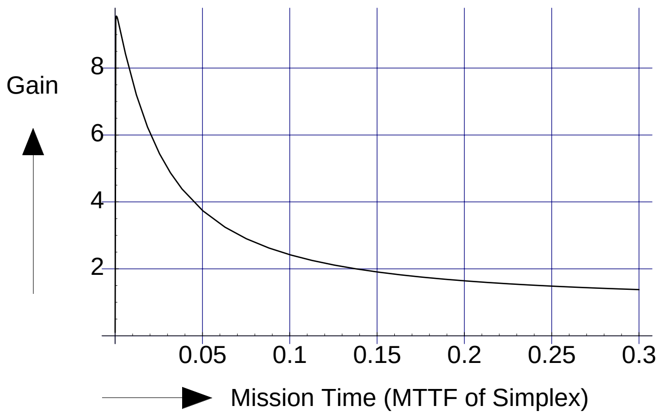

Figure 2.5. Effect of diversity with mission time (for common-mode failures)

In Fig. 2.5, we show how the reliability improvement obtained from diversity depends on mission time. On the Y-axis of the graph in Fig. 2.5, 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 $d_{1,2} = 1\text{-}10^{-11}$.

(2) The probability that a duplex system is not fault-secure at time *i*, for fault pair (f_1, f_2) with $d_{1,2} = 1\text{-}10^{-12}$. 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. 2.5 shows, the gain diminishes with longer mission times. However, it may be noted that diversity is helpful for mission times when a TMR system is most effective. This analysis allows us to derive relationships between the reliability of a redundant system, the diversity incorporated to protect the system against common-mode failures and the mission time. The graph in Fig. 2.5 helps us understand the payoffs of diversity as a function of mission time.

Now we consider design mistakes, which are special cases of common-mode failures. For these cases, the fault is *always* present. Simple analysis reveals that the probability that a duplex system is fault-secure up to time *t*, in the presence of design mistakes, is:

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

Thus, for design mistakes, for a given fault pair (f_1, f_2), the more the value of $d_{1,2}$, the more is the system reliability. 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.

For replicated systems, we can define a common-mode failure as one that produces identical faults (and hence, identical errors) in the two systems. For diverse copies, there is no such simple way to model common-mode failures. Hence, for our simulations, we choose random pairs of faults in an unbiased way. Note that, from our observations in Sec. 2.1, it follows that, in the presence of a common-mode failure, the reliability of a diverse system is never worse than that of a non-diverse system.

While diversity in hardware designs is the main focus of this paper, the above ideas can be extended to analyze diversity in software modules. For estimating the diversity metric for software modules, we need to have a fault model for the software under consideration. Considering the range of values the input variables to the software module can possibly take, it may be difficult to compute the exact value of the metric. However, the value of the metric can be estimated using simulation techniques. 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 [Eckhardt 85], [Tomek 93] and [Lyu 95].

We validate these observations using simulation data in Sec. 4. For simulation purposes we used the stuck-at fault model. In the next section, we introduce the preliminaries related to the stuck-at fault model and illustrate the calculation of our diversity metric using an example.

3. Example

Research in the area of digital testing and diagnosis of combinational and sequential logic circuits has demonstrated the effectiveness of the logical *stuck-at* fault model. In this model, the failures in a logic circuit behave as if as some lines in the circuit assume *constant* logical values, either 1 or 0, independent of the logic values on other lines of the circuit.

For the rest of this paper, we assume that all failures show up as stuck-at faults in the circuit. We also assume that the failures are permanent; i.e., if a stuck-at fault shows up at some time instant *t*, then the fault remains at all time instants greater than *t*. For circuits made from SRAM-based FPGAs, unless we re-initialize the SRAMs (reload a given configuration), a transient fault in the configuration SRAM persists. Thus, the assumption of the permanent fault behavior is reasonable.

For example, consider the network shown in Fig. 3.1. The function implemented by the network is $wx + y$. Consider a stuck-at-0 (s-a-0) fault on the line *y*, denoted by $y/0$. The function implemented by the network, in the presence of the fault, is wx. Thus, $w = 1$, $x = 0$ and $y = 1$, when applied to the input of the logic circuit causes the faulty network to produce a 0 and the fault-free network to produce a 1. Therefore, the fault *y*/0 is detected by the pattern $w = 1$, $x = 0$, $y = 1$.

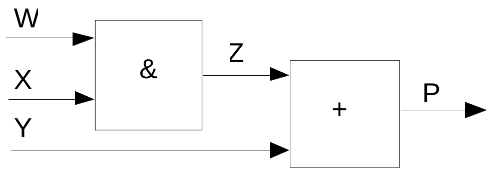

Figure 3.1. An example logic circuit

A fault is said to be functionally equivalent to another fault if and only if the output function realized by the network with only the first fault present is equal to the function realized when only the second fault is present. For example, in the network of Fig. 3.1, in the presence of the fault *x*/0, the function implemented is *y*. In the presence of the fault *z*/0, the function implemented by the network is also *y*. Hence, the faults *x*/0 and *z*/0 are functionally equivalent. The set of functionally equivalent faults forms an equivalence class. A fault f_1 *dominates* fault f_2 if and only if all input combinations that detect f_2 also detect f_1. In our example, the fault *p*/0 dominates fault *z*/0. Techniques for obtaining equivalence and dominance relationships among different fault pairs have been described in [McCluskey 71] and [To 73].

Now, we illustrate the calculation of our design diversity metric with respect to single stuck-at faults in the circuit of Fig. 3.1. There are 10 single-stuck faults associated with this network. The faults are: *w*/0, *w*/1, *x*/0, *x*/1, *y*/0, *y*/1, *z*/0, *z*/1, *p*/0 and *p*/1. The corresponding fault equivalence classes are: F_1 = {*w*/0, *x*/0, *z*/0}, F_2 = {*y*/1, *z*/1, *p*/1}, F_3 = {*w*/1}, F_4 = {*x*/1}, F_5 = {*y*/0} and F_6 = {*p*/0}. The set of vectors that detect the faults in F_1 is $V_1 = \{w = 1, x = 1, y = 0\}$. We write V_1 = {110}. Similarly, V_2 = {000, 010, 100}, V_3 = {010}, V_4 = {100}, V_5 = {001, 011, 101} and V_6 = {001, 011, 101, 111, 110}. Here, the number of inputs (*n*) is 3. Consider the fault pair (f_1, f_2) = (w/0, p/0). The set of vectors that detect *w*/0 is V_1. The set of vectors that detect *p*/0 is V_6. Now, $V_1 \cap V_6$ = {110}. Thus, the value of $d_{1,2}$ is 7/8. In this way all the $d_{i,j}$'s and the *D* metric can be calculated.

4. A Simulation-Based Approach

As we noted earlier, it is difficult to model the entire complex system mathematically. Even with the stuck-at fault model, it is difficult to derive the exact reliability equation for the following reasons:

1. For a given pair of faults (f_1, f_2), the calculation of $d_{1,2}$ is an NP-complete problem [Gary 79]. 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 TMR system in the presence of multiple faulty modules. In [Stroud 94], Stroud also used a simulation technique to obtain the survivability distribution used to calculate the reliability of the TMR system. However, our simulation approach differs from that of Stroud because, our goal is to examine the effect of diversity on the reliability of TMR systems.

Table 4.1. 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

For generating *different* designs, we minimized the truth tables corresponding to some MCNC benchmark circuits (*clip*, *inc*, *Z5xp1*, *apex4* and *rd84*) using *espresso*. Then, we synthesized logic circuits after applying multi-level optimizations using the *rugged* script available in *sis* [Sentovich 92]. We subsequently mapped the multi-level logic circuits to the LSI Logic G-10p technology library [LSI 96]. 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 4.1 summarizes the characteristics of the different simulated designs.

In the fourth column of Table 4.1, 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.

Simulation 1

We considered TMR systems (with identical and different implementations) for the benchmark circuits. In order to evaluate the effect of diversity in the presence of multiple independent failures, we conducted 100,000 experiments, each consisting of the following steps. In each experiment, we start from time instant 0, when all the three modules are fault-free. We have a binary counter for generating the input patterns and we randomly pick a seed for that counter. In each iteration, for each of the modules, we generate a uniform random variable to decide whether the module will be affected by a stuck-at fault. The probability that a particular module will be affected by some fault is proportional to the number of single stuck-at faults in that module — the constant of proportionality is 10^{-6}. If the random variable indicates that the module is going to be affected by a fault, then we randomly pick a stuck-at fault in that module. Now, we apply the content of the counter to the inputs and obtain the output. If the system output is correct (same as the output produced by the fault-free system), we proceed to the next iteration. Otherwise, we note the time instant when the failing output vector is produced and proceed to the next experiment. After completing all the experiments, we calculate the mean time to failure (MTTF) for the system by averaging the number of cycles up to which the system produced correct outputs for each experiment. The improvement in the MTTF over the classical TMR (replicated) MTTF is an indicator of the effectiveness of using a set of modules in a TMR system.

Table 4.2. Simulation 1 results

Circuit Name	TMR #	Copies	# SSFs	MTTF (cycles)
Z5xp1	1	T, T, T	1650	4711
	2	C, C, C	1830	4612
	3	T, T, C	1710	4628
apex4	4	T, T, T	28908	540
	5	T, T, C	27850	516
clip	6	T, T, T	2094	4626
	7	T, T, C	2060	4625
inc	8	T, T, T	1458	6153
	9	C, C, C	1518	5785
	10	T, T, C	1478	5979
rd84	11	T, T, T	1704	5239
	12	C, C, C	1194	6681
	13	T, T, C	1534	5380
	14	C, C, T	1364	5735

In Table 4.2, we show the Mean Time To Failure (MTTF) for the TMR systems with identical and different implementations of the same logic specification. For example, for the Z5xp1 circuit, we formed three TMR systems. For the TMR system 1, we replicated the circuits obtained by synthesizing the original truth table; hence, this TMR system is denoted by (T, T, T) (T stands for "true"). For the TMR system 2, we replicated the circuits synthesized from truth tables with complemented outputs. This system is denoted by (C, C, C) (C stands for "complement"). These two TMR systems correspond to TMRs with identical implementations. TMR 3 contains *different* implementations of Z5xp1. In the fourth column of Table 4.2, we show the total number of single stuck faults for the whole TMR system. In the fifth column, we report the MTTF (the average number of cycles after which the TMR produces incorrect outputs). It is clear from Table 4.2, that the MTTF is strongly dependent on the number of single stuck-at faults in the TMR system. Hence, the MTTF is dominated by the reliability of the individual modules in the TMR system.

Simulation 2

The experiments in Simulation 2 are similar to those in Simulation 1. The only difference is that, the probability that a particular module gets affected by any fault is *fixed, independent of the total number of possible stuck-at faults* in that module. The value of this probability is chosen to be 10^{-5}. In Table 4.3, we present the results obtained from these experiments. For the Z5xp1 example, we find that the TMR with different implementations has a higher MTTF compared to TMR with identical implementations (replication). For each of the remaining cases, there is at least one TMR with replication that has a higher MTTF than a TMR with different implementations. The case of the *inc* benchmark is interesting. The TMR with different implementations has a higher MTTF compared to one of the replicated TMRs and a lower MTTF compared to another replicated TMR.

Table 4.3. Simulation 2 results

Circuit Name	TMR #	Copies	MTTF
Z5xp1	1	T, T, T	63572
	2	T, T, C	64307
apex4	3	T, T, T	95505
	4	T, T, C	85683
clip	5	T, T, T	79594
	6	T, T, C	74883
inc	7	T, T, T	74954
	8	C, C, C	71457
	9	T, T, C	73410
rd84	10	T, T, T	73331
	11	C, C, C	66005
	12	T, T, C	67091
	13	C, C, T	64711

From the results of Simulation 1 and Simulation 2, it can be concluded that: *for independent failures in multiple modules, it is not necessarily true that a TMR system with different implementations will survive (produce correct outputs) for a longer time compared to a TMR system with identical implementations.*

Simulation 3

For Simulation 3, for each benchmark circuit, we built duplex systems with identical and different implementations. For each of these systems, we performed 100,000 experiments. In each experiment, we randomly picked up 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 a binary counter (with random seed) and calculated the *error latency* (the number of cycles after which the system ceased to be fault-secure). For more discussions on error latency, the reader is referred to [Shedletsky 76]. The expected error latency for the injected fault pairs is shown in Table 4.4. We also calculated the percentage of fault pairs for which none of the two modules produced 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 4.4, a duplex system consisting of different implementations of the Z5xp1 circuit has a higher percentage of compensating fault pairs, compared to the non-diverse version — however, that is not generally true.

For example, for the *clip* benchmark, the non-diverse duplex system has a higher percentage of compensating fault pairs. For compensating fault pairs, the error 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 error latency is dependent on the number of compensating fault pairs. This dependence can be explained using our reliability analysis in Sec. 2.1.

Table 4.4. Simulation 3 results

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

Simulation 4

It may be interesting to find how the percentage of compensating faults relates to the error latency in a TMR system. For that purpose, we performed another set of simulations, whose results are reported in Table 4.5. For each of the 100,000 experiments in Simulation 4, we considered each TMR system and randomly generated three faults in the three modules. We injected these faults into the modules and applied patterns from a binary counter (loaded with a random seed). We then computed the error latency (the number of cycles after which the system produces incorrect outputs in the presence of the fault triple). For fault triples that do not produce incorrect outputs (compensating fault triples), we assumed that the error latency is 10,000 cycles, just like Simulation 3. The results in Table 4.5 follow the same trend as those in Table 4.4.

Table 4.5. Simulation 4 results

Circuit Name	Copies	Error Latency (cycles)	% compensating fault triples
Z5xp1	T, T, T T, T, C	3396 3569	33.76 35.49
apex4	T, T, T T, T, C	396 * 373 *	0 * 0 *
clip	T, T, T T, T, C	5376 5277	53.32 52.29
inc	T, T, T T, T, C	4807 4648	47.95 46.34
rd84	T, T, T T, T, C	4772 3741	47.37 37.02

*- results from 20,000 simulations

These results from simulations 3 and 4 indicate that, *for multiple independent failures, the reliability of a redundant system is dominated by the profile of $d_{i,j}$ values of the fault pairs* (f_i, f_j). This property has been captured by our reliability analysis that has been presented in Sec. 2.2.

In [Sakov 87], for a given combinational logic function, the fault detectability profiles for different implementations have been reported. Further studies are needed to synthesize circuit structures with high values of $d_{i,j}$ for different fault pairs. It has been proved in [To 73] that, for fanout-free combinational logic networks, all internal single stuck-at faults are either equivalent to or dominate 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 and stuck-at faults, the diversity metric is going to achieve appreciable high values for networks synthesized in different ways, compared to simple replication. Thus, *it appears to be important to focus on achieving diverse fanout structures of different networks to obtain high values of the diversity metric for fault pairs.*

Simulation 5

Our previous simulation results mainly focused on independent faults in multiple modules of a TMR system. However, it has been observed in the literature [Avizienis 84][Lala 94], 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 possible 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 copies. Hence, for these duplicated 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. After injecting such a fault-pair, we applied input combinations from a binary counter (with the seed chosen randomly). With this setup, we found the error latency (the number of cycles, after which the duplex system ceased to be fault-secure). For duplex systems with different implementations, since we cannot establish such a one-to-one correspondence between the leads of the two copies, we performed 100,000 experiments — in each experiment we randomly chose a fault pair and calculated the error latency. Table 4.6 shows the expected error latencies we obtained from these simulations. These results show a distinct advantage of using different implementations over non-diverse designs for common-mode faults. This result can also be explained from the properties of the diversity metric discussed in Sec. 2.1. However, as we showed in Sec. 2.2, the effectiveness of diversity decreases with increasing mission times.

Table 4.6. Simulation 5 results

Circuit Name	Duplex #	Copies	Error Latency (cycles)
Z5xp1	1	T, T	15
	2	T, C	6869
apex4	3	T, T	106
	4	T, C	8094
clip	5	T, T	60
	6	T, C	7869
inc	7	T, T	12
	8	T, C	7516
	9	C, C	14
rd84	10	T, T	49
	11	T, C	6797
	12	C, C	24

5. Self-testing Property

In this section, we discuss the possible effects of having design diversity on the self-testing property of a duplicated system. A duplicated 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.

Table 5.1. Self-testing properties of diverse and non-diverse duplex systems

Circuit Name	Copies	# SSF pairs	% detected
Z5xp1	T, T	302500	99.27
	C, C,	372100	99.35
	T, C	335500	99.98
clip	T, T	487204	99.42
	T, C	463472	99.98
inc	T, T	236196	99.21
	C, C	256036	99.16
	T, C	245916	99.97
rd84	T, T	322624	99.26
	C, C	158404	98.90
	T, C	226064	99.96

For the purpose of the experiment, we assume that the failures show up as single-stuck faults in each of the two 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 [Mitra 99]. For evaluating the effects, we present some simulation results in Table 5.1. It is clear from Table 5.1 that with different implementations it is possible to achieve high self-testing properties of the designs under consideration. In fact, an interesting synthesis problem is to synthesize two copies of a given logic function such that the number of self-testable fault pairs is maximum.

6. Conclusions

In this paper, we have addressed the issue of design diversity in redundant (software or hardware) systems in order to handle common-mode failures and failures in multiple modules. In order to protect fault-tolerant systems against common-mode failures, design diversity has been used commercially. Conventionally, design diversity means “independent” generation of “different” designs. This notion of diversity is qualitative and has limitations. Hence, the need for a metric to quantify diversity between different systems has been expressed in the past.

In this paper, for the first time, we have introduced a metric to quantify diversity among different designs under a particular fault-model, and explained how to calculate the overall system reliability in terms of this metric. In our example of the calculation of diversity for combinational logic circuits (Sec. 2.1), we have assumed that all the input combinations are equally likely. In the absence of any information about the relative frequency of the different input combinations, this is a reasonable assumption. However, for a particular application, if we have information about the relative frequencies (in the form of input traces, for example), then we can appropriately modify the above expression to incorporate this extra information (by changing the weights associated with different input combinations).

We have also produced simulation results to model real-life environments that inject multiple failures in duplex and TMR system. Our theoretical and simulation results indicate that, in the presence of independent multiple module failures in redundant systems, mere use of different implementations does not guarantee higher reliability compared to redundant systems with identical implementations. It is more important to evaluate the reliability of the systems using our metric. On the other hand, for common-mode failures and design faults, there is a significant gain with different implementations. However, the gain decreases with increasing mission time. Our analysis technique can be used to derive relationships between system reliability, diversity, mission time and system failure rate and compare reliabilities of multiple diverse systems. These relationships can help understand the cost and reliability tradeoffs while designing redundant systems with diversity.

For common-mode failures, diverse systems have no worse reliability compared to replicated systems. However, there is a further need to characterize common-mode failure mechanisms in the circuit level. With a good CMF fault model, (logical or layout-level) synthesis techniques can be used to incorporate sufficient diversity to protect systems against the modeled faults.

Our simulation results demonstrate that diversity plays an important role in enhancing the self-testing property of duplex systems. This can prove to be useful if we can apply specific patterns to the system during idle cycles.

7. Acknowledgments

This work was supported by Defense Advanced Research Project Agency (DARPA) under Contract No. DABT63-97-C-0024 (ROAR project). Thanks are due to Mr. Philip Shirvani of Stanford Center For Reliable Computing.

8. References

[Avizienis 77] Avizienis, A. and L. Chen, "On the implementation of N-version programming for software fault-tolerance during program execution," *Proc. Intl. Computer Software and Appl. Conf.*, pp. 149-155, 1977.

[Avizienis 84] Avizienis, A. and J. P. J. Kelly, "Fault Tolerance by Design Diversity: Concepts and Experiments," *IEEE Computer*, pp. 67-80, August, 1984.

[Briere 93] Briere, D. and P. Traverse, "Airbus A320/A330/A340 Electrical Flight Controls: A family of fault-tolerant systems," *Proc. FTCS*, pp. 616-623, 1993.

[Eckhardt 85] Eckhardt, D. E. and L. D. Lee, "A theoretical basis for the analysis of multi-version software subject to coincident failures," *IEEE Trans. Software Engg.*, Vol. SE-11, pp. 1511-1517, Dec. 1985.

[Gary 79] Gary, M. and D. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman and Company, 1979.

[Lala 94] Lala, J. H. and R. E. Harper, "Architectural principles for safety-critical real-time applications," *Proc. of the IEEE*, vol. 82, no. 1, pp. 25-40, January, 1994.

[Liu 97] Liu, J., et. al., "Heavy ion induced single event effects in semiconductor device," *Proc. Intl. Conference on Atomic Collisions in Solids*, 1997.

[LSI 96] *G10-p Cell-Based ASIC Products Databook*, LSI Logic, May 1996.

[Lyu 91] Lyu, M. R. and A. Avizienis, " Assuring design diversity in N-version software: a design paradigm for N-version programming," *Proc. DCCA*, pp. 197-218, 1991.

[Lyu 95] Lyu, M., *Handbook of Software Reliability Engineering*, Computer Society Press, 1995.

[McCluskey 71] McCluskey, E. J. and F. W. Clegg, "Fault Equivalence in combinational logic networks," *IEEE Trans. On Computers*, Vol. C-20, No. 11, pp. 1286-1293, Nov. 1971.

[McCluskey 88] McCluskey, E. J., 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.

[Mitra 99] Mitra, S., N. Saxena and E. J. McCluskey, "Design Diversity For Redundant Systems," *CRC TR* 99-4, Center For Reliable Computing, Stanford Univ., 1999.

[Pradhan 96] Pradhan, D. K., *Fault-Tolerant Computer System Design*, Prentice Hall, 1996.

[Reed 97] Reed, R., et. al., "Heavy ion and proton-induced single event multiple upset," *IEEE Trans. on Nuclear Science*, Vol. 44, No. 6, pp. 2224-2229, July 1997.

[Riter 95] Riter, R., "Modeling and Testing a Critical Fault-Tolerant Multi-Process System," *Proc. FTCS*, pp. 516-521, 1995.

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

[Saxena 98] Saxena, N.R., and E.J. McCluskey, "Dependable Adaptive Computing Systems," *Proc. IEEE Systems, Man and Cybernatics Conf.*, San Diego, pp. 2172-2177, 1998.

[Shedletsky 76] Shedletsky, J.J., and E.J. McCluskey, "The Error Latency of a Fault in a Sequential Digital Circuit," *IEEE Trans. Computers*, C-25, No. 6, pp. 655-659, June 1976.

[Sentovich 92] Sentovich, E. M., *et. al.*, "SIS: A System for Sequential Circuit Synthesis," *ERL Memo. No. UCB/ERL M92/41*, EECS, UC Berkeley, CA 94720.

[Siewiorek 75] Siewiorek, D. P., "Reliability modeling of compensating module failures in majority voted redundancy," *IEEE Trans. Comp.*, vol. 24., no. 5, pp. 525-533, 1975.

[Siewiorek 92] Siewiorek, D. P. and R. S. Swarz, *Reliable Computer Systems: Design and Evaluation*, Digital Press, 1992.

[Stroud 94] Stroud, C. E., "Reliability of Majority Voting Based VLSI Fault-Tolerant Circuits," *IEEE Trans. on VLSI*, vol. 2, no. 4, pp. 516-521, December, 1984.

[Tamir 84] Tamir, Y. and C. H. Sequin, "Reducing common mode failures in duplicate modules," *Proc. ICCD*, pp. 302-307, 1984.

[To 73] To, K., "Fault Folding for Irredundant and Redundant Combinational Circuits," *IEEE Trans. Comp.*, Vol. C-22, No. 11, pp. 1008-1015, Nov. 1973.

[Tohma 71] Tohma, Y. and S. Aoyagi, "Failure-tolerant sequential machines with past information," *IEEE Trans. Computers*, Vol. C-20, No. 4, pp. 392-396, April 1971.

[Tomek 93] Tomek, L. A., J. K. Muppala and K. S. Trivedi, "Modeling Correlation in Software Recovery Blocks," *IEEE Trans. Software Engg.*, Vol. 19, No. 11, pp. 1071-1086, Nov. 1993.

[Von Neumann 56] Von Neumann, J., "Probabilistic Logics and the synthesis of reliable organisms from unreliable components," *Automata Studies, Ann. of Math. Studies*, no. 34, pp. 43-98, 1956.