

DESIGN OF REDUNDANT SYSTEMS PROTECTED AGAINST COMMON-MODE FAILURES

Subhasish Mitra and Edward J. McCluskey
Center for Reliable Computing

Departments of Electrical Engineering and Computer Science
Stanford University, Stanford, California
<http://crc.stanford.edu>

Abstract

Redundancy techniques like duplication and Triple Modular Redundancy (TMR) are widely used for designing dependable systems to ensure high reliability and data integrity. In this paper, for the first time, we develop fault models for common-mode failures (CMFs) in redundant systems and describe techniques to design redundant systems protected against the modeled CMFs. We first develop an input-register-CMF model that targets systems with register-files. This paper shows that, in the presence of input-register-CMFs, we can always design duplex or TMR systems that either produce correct outputs or indicate error situations when incorrect outputs are produced. This property ensures data integrity. Next, we extend the input-register-CMF model to consider systems where the storage elements of the registers are not organized in register-files; instead, the register flip-flops are placed using conventional CAD programs. For this case, we present a technique to synthesize redundant systems with guaranteed data integrity against the extended input-register-CMFs.

1. Introduction

Redundancy techniques like duplication and Triple Modular Redundancy (TMR) are widely used for designing dependable systems. There is a vast literature on various hardware redundancy techniques and reliability modeling for systems incorporating these redundancy techniques [Siewiorek 92].

The classical analysis techniques for redundant systems are optimistic because they do not consider common-mode failures [Lala 94]. CMFs result from failures that affect more than one element of a redundant system at the same time, generally due to a common cause. They may be design faults or operational faults due to external (such as EMI and radiation) or internal causes. CMFs are surveyed in [Mitra 00a]. There is no fault-model available for CMFs although it has been pointed out that analysis techniques that include CMFs do not use good CMF models. This impedes progress in the analysis and design of redundant systems in the presence of common-mode failures.

Our main contributions in this paper are the following. For the first time, we have developed quantitative fault models for CMFs. We have also developed techniques to

design redundant systems with guaranteed data integrity against the modeled CMFs. By data integrity, we mean that the system either produces correct outputs or notifies error when incorrect outputs are produced. In the literature on fault-tolerance, this property is also referred to as the fault-secure property.

Section 2 introduces TMR systems with a new voter design called the word-voter which was described in [Mitra 00b]. These voters are useful in the context of design of TMR systems protected against CMFs. In Sec. 3 we develop a common-mode fault model called input-register-CMF for redundant systems containing register-based designs. This fault model targets systems where the registers are organized in register-files. Section 4 uses an example to explain our technique for designing TMR systems protected against input-register-CMFs. In Sec. 5 we design TMR systems with guaranteed data integrity in the presence of modeled input-register-CMFs. Section 6 extends the input-register-CMF model to consider systems where the register flip-flops are not organized in register-files and presents a technique to design redundant systems protected against the modeled CMFs. In Sec. 7 we present simulation results followed by conclusions in Sec. 8.

2. TMR Systems with Word-Voter

In TMR systems, majority voting is normally performed on a bit-by-bit basis [Siewiorek 92]. For a module with n outputs, the TMR implementation has three modules and n single-bit voters. Suppose that we have a TMR system where each module has two outputs. Due to the presence of a fault in Module 1, in response to a particular input combination, the module produces an output combination 10 instead of 01. Similarly, due to the presence of a fault in Module 2, the output combination obtained from it is 11. Finally, suppose that Module 3 is working correctly and produces the expected output 01. With bit-wise voting, the voter corresponding to the first output produces a 1 and the one corresponding to the second output produces a 1. Thus, we have 11 at the system output. However, if we consider the *output word* from each module, we find that the output words from all the three modules are different. The output words from the first, second and third modules are 10, 11 and 01, respectively. This can be treated as an erroneous state for a voter because, no two output *words* are equal. Based on this observation, we can modify the classical voter design

by adding some extra circuitry that detects this error condition and produces an error signal. The details of the word-voter design are described in [Mitra 00b].

3. Input-Register-CMF Model

Consider a redundant system (duplex or TMR) where there are separate input registers associated with each module. In such a system, consider the failures such that the same bit positions in two or three of the input registers are *stuck* at the same value. In classical redundant systems, these failures will have identical effects on all the implementations and will go undetected producing incorrect outputs. Hence, these failures are possible CMF candidates. We call these CMFs *Input-Register-Common-Mode-Failures* and the corresponding fault model as the *Input-Register-CMF Model (IR-CMF)*. This fault model includes transient faults from sources like radiation upsets and permanent faults caused by the operational environment. In a space environment, for example, radiation upsets can be a source of common-mode CMFs. It was shown in [Reed 97] that radiation sources can cause multiple-event upsets in sequential elements (e.g. flip-flops) of a design. Experimental results in [Liden 94] show that, in a radiation environment, 98% of the bit errors in sequential elements are caused by particles directly hitting these elements, and only 2% are caused by transients in combinational logic. Upsets from radiation sources can be modeled as bit-flips or bit-stuck-at faults [Choi 93][Rimen 94].

Consider the TMR system shown in Fig. 3.1. The system architecture is such that the registers are organized in a register file like the organization of RAM cells. The three m -bit input registers under consideration are S_1 , S_2 and S_3 . The storage elements in a register file are organized in an array structure. This architecture of register organization is common in many VLSI designs including microprocessors and ASICs like network switches and graphics chips. The register-file in Fig. 3.1 has three readout ports.

Suppose that the storage elements in a row correspond to the different bit positions of a single register. In such a system, a single source of radiation can cause upsets in adjacent storage elements. If the affected storage elements belong to a particular row, then the TMR system is protected (since only one of the registers will contain corrupt data). However, if the adjacent storage elements belong to a particular column, then identical bit positions of two or more input registers are affected by the radiation source. These CMFs cannot be detected in conventional TMR systems even using our modified voter design. It may be argued that if the registers producing inputs for the TMR system are physically placed far enough in the register-file, then the chances of upsets in two registers of the TMR system are low. However, it may not be possible to control the actual choice of the registers if compilers do not have the knowledge about the relative physical locations of the different registers.

There is evidence that design faults constitute a dominant source of CMFs in redundant systems [Lala 94]. As an example of a design fault, suppose that due to design faults, a bit-column of a register file produces very weak signals corresponding to a particular logic value. Such a design fault can manifest itself as a stuck-at fault in the same bit positions of multiple input registers.

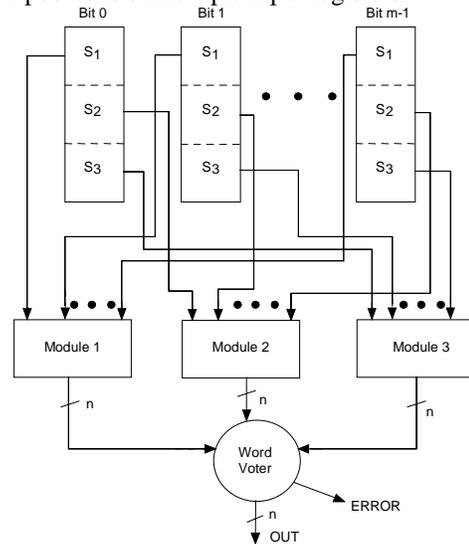


Figure 3.1. System architecture with register files

In Fig. 3.1, only one voter has been shown for the TMR. However, we can use three voters for generating three sets of outputs from the TMR. The outputs produced by the TMR are stored in three registers in the register file.

Note that, while the focus of this paper is on bit-stuck-at-faults, our techniques can be extended for bit-flips which may be realistic because of the symmetric design of the flip-flops. It may be argued that duplex or TMR systems with input registers containing parity bits are secure against the modeled CMFs. However, in this paper, we do not assume the presence of parity bits with the input registers. Later in this paper, some of our area results also demonstrate the advantages of using our technique over simple addition of parity bits. Moreover, our technique doesn't require the storage of register parity bits. In the next section, with the help of an example, we describe our technique to detect the input-register-CMFs. For duplex systems, our technique stores the actual input values in the input register of one of the modules and the complemented input values in the input register of the other module. For TMR systems, our technique stores the actual input values in the input register of the first module, the complemented input values in the input register of the second module and a "transformation" of the input values in the input register of the third module. The problem of finding the right transformation can be modeled as a Boolean Satisfiability problem and can be solved as described in [Mitra 00c].

4. An Example

Consider a simple combinational logic circuit, N , with three inputs x , y and z and two outputs. The two output

functions are $f_1 = xy' + yz$ and $f_2 = xy' + yz'$. For duplex systems, we can make the system fault-secure against input-register-CMFs by storing the actual input values in the register of one of the modules and storing the complemented input values in the input register of the other module. Thus, for input-register-CMFs, design of duplex systems is simple; the design of TMR systems is more challenging.

In a TMR system, we have three 3-bit registers A, B and C and three copies of the original circuit N (truth table shown in Table 4.1). In a conventional TMR system, the first copy of N gets its inputs from the 3 bits of register A; i.e., A_1 stores values corresponding to X, A_2 stores Y values and A_3 stores Z values. The same scenario holds for registers B and C. The IR-CMFs involving the first bit position are $\{A_1/1, B_1/1\}$, $\{A_1/1, C_1/1\}$, $\{A_1/1, B_1/1, C_1/1\}$, $\{B_1/1, C_1/1\}$, $\{A_1/0, B_1/0\}$, etc.

Table 4.1 Truth tables

Network N			Network N ₁			Network N ₂								
A ₁	A ₂	A ₃	f ₁	f ₂	B ₁	B ₂	B ₃	f ₁	f ₂	C ₁	C ₂	C ₃	f ₁	f ₂
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
0	0	1	0	0	0	0	1	0	1	0	0	1	1	1
0	1	0	0	1	0	1	0	1	1	0	1	0	0	0
0	1	1	1	0	0	1	1	1	1	0	1	1	1	1
1	0	0	1	1	1	0	0	1	0	1	0	0	0	1
1	0	1	1	1	1	0	1	0	1	1	0	1	0	1
1	1	0	0	1	1	1	0	0	0	1	1	0	1	0
1	1	1	1	0	1	1	1	0	0	1	1	1	1	0

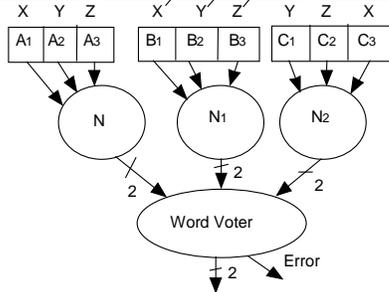


Figure 4.1. TMR implementation using our technique

Consider the TMR system of Fig. 4.1. Note that, the three registers A, B and C contain different values. The first bit of register A stores the value of variable X, the second bit stores Y's and the third bit stores Z's. The first, second and third bits of register B store values corresponding to X', Y' and Z', respectively. For register C, the first bit stores Y's, the second bit stores Z's and the third bit stores X's. The networks are implemented in such a way that for a particular combination of X, Y and Z, all the three networks produce the same outputs (so that a voter can be used). When $X = 1, Y = 0$ and $Z = 0$, all the three networks produce 11. Network N₁ (truth table in Table 4.1) may be implemented as a *dual* of the original network N [McCluskey 86], instead of replicating N and adding inverters at the inputs. This can help in protecting the system against common-mode failures affecting the

leads inside the implemented networks, instead of only the network inputs. In this system architecture the second module will have complemented inputs and will produce complemented outputs. In that case, the voter design can be changed to incorporate this architectural requirement. Note that the input register of the third module contains a rotated version of the content of the input register of the first module. Hence, network N₂ (truth table in Table 4.1) can be implemented with the same number of logic gates as network N.

Consider the CMF $\{A_1/1, B_1/1, C_1/1\}$. If we apply $X = 0, Y = 0$ and $Z = 1$, the network N sees 101 (since A_1 is stuck-at 1) at its inputs and produces $f_1 = 1$ and $f_2 = 1$. The network N₁ sees 110 at its inputs and produces 00 at its output. Network N₂ sees 110 at its input (because C_1 is stuck at 1 and $X = 0, Y = 0$ and $Z = 1$) and produces 10 at its output. Thus, when $X = 0, Y = 0$ and $Z = 1$, the three modules produce three different words at their outputs. This is an erroneous situation for the word-voter and it produces an error signal. Hence, we can detect the common-mode fault under consideration. In a similar way, it can be shown that data integrity against other common-mode faults is guaranteed in the TMR system of Fig. 4.1. Hence, the system provides error masking against independent failures and is fault-secure against the modeled IR-CMFs. It may be argued whether $\{A_1/1, C_3/1\}$ should also be called a common-mode fault (because A_1 and C_3 store the same value X). For this initial part of this paper, we are not going to consider it as an IR-CMF. In Sec. 6 we present a technique to handle these cases too.

5. Transformation Characterization

In [Mitra 00c] we presented a formulation of the problem of finding a suitable transformation T for the inputs of the third module. However, for practical purposes, the logic complexity of the third module is dependent on the transformation chosen. Depending on the implemented function, we want to choose a *good* transformation function. This means that, for the Boolean Satisfiability Problem used to model the problem of finding a transformation T , we have to consider all possible satisfying transformations and choose the one that leads to the minimum logic implementation. There is an inherent difficulty in solving such a problem for circuits with a large number of inputs. Hence, it is important to consider some restricted transformations and examine the class of functions for which these transformations can be applied.

We characterize the left rotation transformation. By a *left rotation transform*, we mean that the content of the input register of the third module is obtained by a left rotation of that of the input register of the first module. For example, if the input register of the first module contains values of the variables x, y and z in the first, second and third bit positions, respectively, then the input register of the third module contains values corresponding to y, z and x in its first, second and third bit positions,

respectively, under the left rotation transform. This transformation does not add to the logic complexity of the third module.

Theorem 1: Any function with an *even* number (n) of input bits can be made fault-secure against IR-CMFs, by adding a maximum of one extra output, if we apply the left rotation transformation to the third module. The function corresponding to the extra output is $x_2 \oplus x_4 \oplus x_6 \oplus \dots \oplus x_n$ where x_i 's are the input bits.

For example, consider a network N implementing any arbitrary logic function with $n = 4$ inputs, x_1, x_2, x_3 and x_4 . We add another output function $g = x_2 \oplus x_4$ to N . Consider a TMR system where the input registers of the first and the second modules contain the true and complemented values of the variables, respectively, while the input register of the third module contains variable values under a left rotation transformation. Consider an input-register CMF such that the first bits of the input registers of module 1 and module 3 are stuck at 1. The first bit position of the input register of module 1 contains the x_1 value, while that of the third module contains x_2 value. Consider any input combination with $x_1 = 0$ and $x_2 = 0$. The g output of the first module is x_4 . However, the g output of the third module is x_4' . Hence, the output words from the first and the third modules can never match and the two modules affected by the CMF can never produce the same output word. Thus, the TMR either produces correct outputs or generates the *ERROR* signal when an incorrect output word is produced.

Theorem 2: A function with an odd number (n) of input bits can be made fault-secure against IR-CMFs by adding two extra outputs, if we apply the left rotation transformation to the third module. The function corresponding to the extra output can be $x_1 \oplus x_3 \oplus x_5 \oplus \dots \oplus x_n$. The second output function is equal to x_1 or x_n .

Any scheme using register parity bits needs at least around $2n$ XOR gates (including parity generation and checking) and extra register flip-flops for a function with n inputs. However, the technique used in Theorems 1 and 2 require $1.5n$ XOR gates and no extra register flip-flop. Theorems 1 and 2 demonstrate that we can always design TMR systems secure against IR-CMFs for any arbitrary logic function. These theorems have been proved in [Mitra 00c].

6. Redundancy and Extended Input-Register-CMF

In this section, we extend the IR-CMF model of Sec. 3. For simplicity and space constraints, we explain our technique for designing duplex systems that are fault-secure against the extended IR-CMFs. However, the whole discussion can be extended for TMR systems with word-voters.

Under the extended fault model, we relax the constraint that only the corresponding bit-positions of the input registers can be stuck. Thus, for a duplex system

(consisting of two modules), we can have a stuck-at fault in any bit position of the input register of the first module and another stuck-at fault in any bit position of the input register of the second module at the same time. The rationale behind this is that, the model of Sec. 3 mainly targets systems where the registers are organized in register files according to the RAM model. However, this may not be true for registers in many ASICs that are synthesized, placed and routed using conventional CAD tools. In this case, a radiation source, for example, can cause multiple upsets in different bit positions of multiple registers depending on the physical proximity of the storage elements of the different registers. This information is not usually available during the design phase. Also, it may be impractical to generate layouts on the basis of vulnerability to radiation upsets. This is because, area and performance of the final chip is strongly dependent on its layout. Moreover, even for register-file-based designs, the actual adjacencies of the different bit-positions of the registers in the layout-level may be different from the logical view of the register-file (Fig. 3.1). Hence, during design we must guarantee that the system is fault-secure against the extended IR-CMFs.

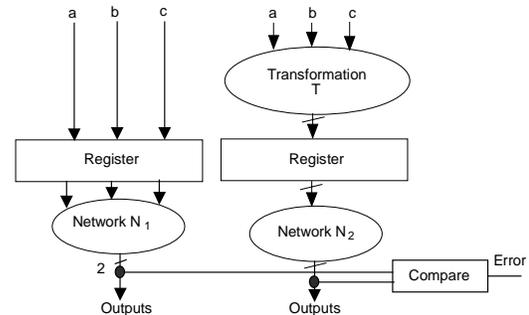


Figure 6.1. System architecture for duplex system: Network N_1 corresponds to Module 1 and network N_2 corresponds to Module 2

In this case, the overall system architecture is as follows. Each logic block is designed as a duplex system consisting of two modules. Consider a logic block A (designed as a duplex system) consisting of two modules. Each module of logic block A receives its inputs from its input register. The outputs of two modules are compared and an error is indicated when there is a mismatch. The outputs of the duplex system are stored in two registers. The outputs of the first module are directly stored in the output register corresponding to the first module – this output register acts as the input register for the first module of another logic block B (designed as a duplex system). The outputs of the second module are passed through a transformation function (shown as transformation T in Fig. 6.1), and the outputs of the transformation function are stored in the output register corresponding to the second module of logic block A – this register serves as the input register for the second module of logic block B .

Consider the example logic function of Table 4.1 (network N). Suppose we want to design a duplex system

for this logic function. Let us call the two modules of the duplex system N_1 and N_2 . The first module (N_1) of the duplex system is obtained by normal synthesis of the function — its input register contains the values of a , b and c in the three flip-flops. For the second module, we want to find a “good” transformation T of the inputs of the logic function. By a “good” transformation we mean a transformation which guarantees data integrity against IR-CMFs and has minimal area and timing impact. The outputs of the network implementing the transformation T will be connected to the input register of the second module. Finally, the second module (N_2) will take inputs from its input register and produce outputs (Fig. 6.1). The transform T can be specified in the following way. For each combination i of a , b and c , transformation T produces an output represented by a particular *symbolic output* X_i . The actual value of X_i , in terms of 1s and 0s, depends on the logic function implemented by T . Table 6.1 shows the specification of T .

If we examine Table 4.1 (Network N), we find that input combinations 000 and 001 produce the same output word (00). Hence, the transformation T can produce the same output (and hence, the same output symbol) in response to input combinations 000 and 001. Two output symbols X_i and X_j in the specification of T (Table 6.1) are *equivalent* if and only if input combinations i and j produce the same output word in the implemented function (Table 4.1, Network N).

Table 6.1. Transformation T

No equivalence		With equivalence		N ₂ Specification	
Inputs a b c	Output Symbols	Inputs a b c	Output Symbols	Input	Output
0 0 0	X ₀	0 0 0	Z ₁	Z ₂	00
0 0 1	X ₁	0 0 1	Z ₁	Z ₃	10
0 1 0	X ₂	0 1 0	Z ₂	Z ₄	11
0 1 1	X ₃	0 1 1	Z ₃		
1 0 0	X ₄	1 0 0	Z ₄		
1 0 1	X ₅	1 0 1	Z ₄		
1 1 0	X ₆	1 1 0	Z ₂		
1 1 1	X ₇	1 1 1	Z ₃		

In Table 6.1 we find that X_0 and X_1 are equivalent. Hence, we replace them by the symbol Z_1 . Similarly, X_2 and X_6 are equivalent and are replaced by the symbol Z_2 . Equivalent symbols X_3 and X_7 are replaced by the symbol Z_3 ; and, X_4 and X_5 by Z_4 .

Finding the transformation T is equivalent to encoding the Z_i 's of Table 6.1 using 1s and 0s in the presence of some constraints that will make the system fault-secure against the extended IR-CMFs. The problem can be modeled as a output encoding problem with certain constraints that must be satisfied so that data integrity against extended IR-CMFs is guaranteed. For details about the output encoding problem, the reader is referred to

[Devadas 91]. Next, we describe the constraints that should be satisfied.

Consider an extended IR-CMF, due to which, the first bit of the input register of the first module is stuck-at 1 and any arbitrary bit of the input register of the second module is stuck at a particular value. When input combination 000 is applied, the first module has 100 at its input and produces 11 (Table 4.1, Network N). For the system to be fault-secure, the second module must not produce 11. Hence, for the second module (N_2), the code corresponding to Z_1 must not get changed to the code of Z_4 due to the presence of a single stuck-at fault at any bit position of the input register of the second module. This can be written as a *Hamming distance* constraint: $d(Z_1, Z_4) > 1$. Here, $d(Z_1, Z_4)$ is the notation for the Hamming distance between Z_1 and Z_4 . This is illustrated in Figure 6.2.

Module 1		
	Input Comb.	Output
Fault-free	000	00 = N ₂ (Z ₁)
Bit 1 stuck-at 1	100	11 = N ₂ (Z ₄)

Module 2		
	Input Comb.	Output
Fault-free	T(000) = Z ₁	N ₂ (Z ₁)
Any single bit stuck	Anything except Z ₄	Anything except N ₂ (Z ₄)

Figure 6.2. Generation of distance constraints

Algorithm: Generate Distance Constraints

For each extended IR-CMF and each input combination i
 Let i change to combination j with the CMF present
 If $Z_i = Z_j$ no distance constraint generated
 else Distance constraint $d(Z_i, Z_j) > 1$ is generated

For our current example, the distance constraints generated are: $d(Z_1, Z_2) > 1$, $d(Z_1, Z_3) > 1$, $d(Z_1, Z_4) > 1$, $d(Z_2, Z_3) > 1$ and $d(Z_2, Z_4) > 1$. The existence of an encoding satisfying all the constraints is always guaranteed by using an extra parity bit. The constrained output encoding can be accomplished by modifying conventional encoding techniques [Lin 90]. Table 6.2 shows an encoding that satisfies the distance constraints.

In Table 6.2, we used 3 bits to encode four symbols. However, there are 8 possible binary combinations with the 3 bits — hence, there are 4 *don't care* combinations. We must ensure that the fault-secure property does not get violated because of these don't cares. For example, suppose we apply input combination 000 to the system. The register corresponding to the second module should contain 001. Consider a CMF, due to which the first bits of the input registers of the two modules are stuck-at 1. In response to input combination 000, the first module sees 100 at its inputs and produces output 11. Network N_2 sees 101 at its input. However, 101 is a *don't care* combination. We must ensure that N_2 does not produce 11 in response to 101. Since $d(Z_1, Z_2) > 1$, $d(Z_1, Z_3) > 1$, $d(Z_1, Z_4) > 1$ had to be satisfied, network N_2 must not produce 11, 01 or 10

in response to any combination of c_1 , c_2 and c_3 at a distance 1 from the code of Z_1 . 101 is an input combination which is at unit distance from 001, the code of Z_1 . However, 101 is at unit distance from the code of Z_4 (100) and since $d(Z_4, Z_1) > 1$, network N_2 must not produce 00 in response to 101. Hence, we need an extra output combination. Since there are two output bits, there can be a maximum of 4 output words. So we need an extra output bit. We can perform similar computations for the remaining don't care combinations. The truth table of N_2 is shown in Table 6.2. Note that, for this particular example we did not have enough choices for the output words that the don't care combinations on the input variables can produce. It is not true that we always have to add an extra output.

Table 6.2. Encoding satisfying distance constraints

Encoding			Final truth table for N_2			
Symbol	x_1	x_2	x_3	o_1	o_2	o_3
Z_1	0	0	1	0	0	- - 1
Z_2	0	1	0	0	0	1
Z_3	1	1	1	0	1	0
Z_4	1	0	0	0	1	1
				1	0	1
				1	0	- - 1
				1	1	- - 1
				1	1	1

If we want to make the above system self-testing, instead of fault-secure, against IR-CMFs, we can accomplish the encoding using only two bits [Mitra 00c].

7. Simulation Results

In this section, we present simulation results. We applied the technique described in Sec. 6 to duplex systems for some MCNC benchmark functions. For solving the encoding problem, we modified JEDI [Lin 90], an existing FSM state encoding tool, to incorporate the distance constraints that are generated by our technique. We performed encoding with two design objectives: (1) Duplex systems that are *fault-secure* from extended IR-CMFs. (2) Duplex systems that are *self-testing* against extended IR-CMFs. For synthesizing the logic circuits, we used the *Sis* tool [Sentovich 92]. For technology mapping purposes, we used the LSI Logic G10p library [LSI 96]. In Table 7.1, for each benchmark, we present a comparison of the areas of conventional duplex systems obtained through direct replication and the systems synthesized using our technique presented in Sec. 6.

Table 7.1. Area Comparison

Circuit Name	# i/p, # o/p	Area of Duplex System		
		Normal	Fault-secure	Self-testing
example1	15, 9	500	480 (-4%)	428 (-14%)
c17	5, 2	80	107 (33%)	92 (15%)
clip	9, 5	846	1002 (18%)	752 (-11%)
misex1	8, 3	340	307 (-10%)	284 (-16%)

It may be noted, that for some of the cases (with negative overheads), the duplex systems which provide protection against the extended IR-CMFs consume *less*

area compared to a conventional duplex system with two replicated modules.

8. Conclusions

It is a well-known fact that redundant systems are prone to common-mode failures. This paper shows, for the first time, that it is possible to develop fault-models for common-mode failures (CMFs) in redundant systems. The CMF models provide new opportunities to design redundant systems so that data-integrity is maintained during system operation in the presence of the modeled CMFs. This paper presents one such design technique. Our design technique gives rise to new and very interesting problems that may prove to be useful for logic synthesis.

9. Acknowledgment

This work was supported by DARPA under Contract No. DABT63-97-C-0024. Thanks to Nirmal Saxena and Philip Shirvani of Stanford CRC for their comments.

10. References

- [Choi, 93] Choi, G. S., R. Iyer and D. Saab "Fault Behavior Dictionary for Simulation of Device-level Transients," *Proc. ICCAD*, pp. 6-9, 1993.
- [Devadas 91] Devadas, S., and A. R. Newton, "Exact Algorithms for Output Encoding, State Assignment and Four-Level Boolean Minimization," *IEEE Trans. on Computer-Aided Design*, Vol. 10, No. 1, pp. 13-27, January 1991.
- [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.
- [Liden 94] Liden, P., et al., "On Latching Probability of Particle-Induced Transients in Combinational Networks," *Proc. FTCS*, pp. 340-349, 1994.
- [Lin 90] Lin, B. and A. R. Newton, "Synthesis of Multiple Level Logic from Symbolic High-Level Description Languages," *VLSI 89*, pp. 187-196, Elsevier Science Publishers, 1990.
- [LSI 96] *G10-p Cell-Based ASIC Products Databook*, LSI Logic, May 1996.
- [McCluskey 86] McCluskey, E. J., *Logic Design Principles with Emphasis on Testable Semicustom Circuits*, Prentice-Hall, Englewood Cliffs, NJ, USA, 1986.
- [Mitra 00a] Mitra, S., N. Saxena and E. J. McCluskey, "Common-Mode Failures in Redundant VLSI Systems: A Survey," *IEEE Trans. Reliability*, To appear, 2000.
- [Mitra 00b] Mitra, S. and E. J. McCluskey, "Word-Voter: A New Voter Design for Triple Modular Redundant Systems," *Proc. VLSI Test Symposium*, pp. 465-470, 2000.
- [Mitra 00c] Mitra, S., and E. J. McCluskey, "Design of Redundant Systems Protected Against Common-Mode Failures," *Technical Report, CRC-TR-00-2*, Stanford Univ., 2000 (<http://crc.stanford.edu>).
- [Reed 97] Reed, R., et al., "Heavy ion and proton-induced single event multiple upset," *IEEE Trans. Nuclear Science*, Vol. 44, No. 6, pp. 2224-2229, July 1997.
- [Rimen 94] Rimen, M., J. Ohlsson, J. Torin, "On Microprocessor Error Behavior Modeling," *FTCS*, pp. 76-85, 1994.
- [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 92] Siewiorek, D. P., R. S. Swarz, *Reliable Computer Systems: Design and Evaluation*, Digital Press, 1992.