Modeling of Common-Mode Failures in Digital Embedded Systems

Lori M. Kaufman • University of Virginia • Charlottesville
Sanyukta Bhide • University of Virginia • Charlottesville
Barry W. Johnson • University of Virginia • Charlottesville

Key Words: Reliability, Fault trees, Sensitivity analysis

SUMMARY & CONCLUSIONS

This paper demonstrates how to accurately model the effects of common mode failures for digital embedded systems. By modeling the system's information flow, the integrated nature of the software and hardware components contained within such a system is represented. This modeling scheme allows for the system to be partitioned into Error Containment Regions (ECRs), which are an extension of the Fault Containment Region (FCR) concept presented in (Ref. 1). These ECRs are defined such that an error at their boundary results in system failure. If two or more ECRs produce errors at their boundaries and the underlying cause of these errors is identical, then the identification of common mode failures is achieved.

1. INTRODUCTION

As systems are required to perform increasingly complex tasks, their design complexity has increased. Design techniques such as redundancy, diversity and defense in depth are used to ensure dependability. However, there have been instances where a protection system is disabled by the occurrence of an event against which it is designed to protect (Ref. 2). Such instances occur by nullifying the assumption of component independence (Ref. 3) made by these methodologies. These failures can be caused by severe environmental conditions, design errors, calibration errors, and/or functional deficiencies (Ref. 4). Failures of this kind are known as common-mode failures (CMF).

Historically, most CMF definitions have treated hardware and software independently, with the majority of definitions pertaining to hardware. For example, any instance where multiple hardware units or components fail due to a single cause is called a CMF (Ref. 5). This definition, however, omits important information, such as the time of failure and the failure mode. Another way of defining CMFs is that they are the result of an event which, because of dependencies, causes a coincidence of failure states of components in two or more separate channels of a redundant system preventing the system from performing its intended function (Ref. 6). This definition takes into account the time of failure, but it still does not include any failure modes. Incorporating all aspects of failure, such as time of occurrence, type of cause, and mode of failure, CMFs are defined as multiple, concurrent, and dependent failures of identical equipment that fail in the same mode (Refs. 7 - 8). An example of identical equipment failing in the same mode is the stuck open or the fail as-is mode (Ref. 3). A stuck open failure mode refers to a failure caused by a break in an input, output, or intermediate logic line in a digital circuit. A fail as-is condition indicates the failure of modules in a manner such that they remain at the value they were at when the fault occurred. Hence, CMFs occur when two or more identical modules are affected by a fault in exactly the same way at exactly the same time, when there exists at least one input combination for which the outputs of two modules are erroneous. The outputs of the two modules are identical for all possible input combinations (Ref. 9). A definition of CMFs from a software perspective is a failure which occurs when two or more software versions fail in exactly the same way for the same input (Ref. 10).

In combining definitions for both hardware and software failures, the causal relationship between the two is examined. A time delay in the propagation of a fault and error from a software unit to a hardware unit or vice-versa defines this causal relationship. Hence, CMFs are related failures of redundant or separate equipment embracing all such causal relationships. The equipment may fail due to severe environments, design errors, calibration and maintenance errors, hardware and software, operating systems, and consequential failures (Ref. 11).

As definitions for CMFs evolved, this timeline is shown in

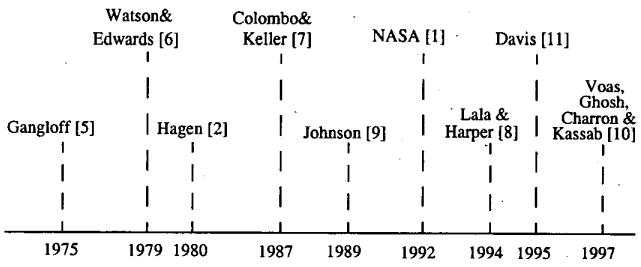

Figure 1: Timeline of the referenced definitions

0-7803-5848-1/00/$10.00 © 2000 IEEE

Figure 1, efforts were made to describe common-mode faults to aid in defining design methodologies. In developing such a definition, the concept of Fault Containment Regions (FCRs) was introduced (Ref. 1). A FCR is a collection of components, either software and/or hardware, that operates correctly regardless of any arbitrary logical or electrical fault outside the region; that is, faults inside a FCR cannot propagate outside the FCR and faults outside the FCR cannot affect it. A common-mode fault is defined as one that affects multiple fault containment regions simultaneously or nearly simultaneously (Ref. 1). It is the purpose of this paper to develop a modeling methodology for CMFs to apply to digital embedded systems derived from the definitions for FCRs and CMF as presented in (Ref. 1).

The organization of this paper is as follows: Section 2: Model Development presents the development of a CMF model for digital embedded systems. Section 3: System Analysis using the Information Flow explains the implementation of the model for an actual system and the application of the model to a prototype system. Finally, Section 4: Conclusions and Recommendations presents the conclusion of this research effort.

Acronyms

CMF	common mode failure
DFM	dynamic flowgraph methodology
ECR	error containment region
FCR	fault containment region
VHDL	VHSIC hardware description language

2. Model Development

Some of the more popular design methods used in designing fault tolerant systems, that is systems that can continue to operate correctly in the presence of faults, are defense in depth (Ref. 12), redundancy [9, 12] and diversity (Ref. 12). Defense in depth approaches safety by implementing multiple protective echelons in a concentric nature within the design; that is, system safety is achieved by separating safety features such that the failure of one echelon does not disable the safety systems contained within other echelons. Hence, defense in depth assumes independence of the echelons. Redundancy is the addition of information, resources or time beyond what is needed for normal system operation and is implemented in either hardware or software. All types of redundancy assume component independence; that is, each component, either hardware or software, acts independently in the system while performing its task. The principle of diversity is to provide several ways of detecting and responding to a significant event by using different logic, algorithms, or means of actuation. In achieving these responses and detections, the independence of different subsystems is assumed. In any of these schemes, the simultaneous component failures, which is a CMF, violate this independence assumption and system failure can arise. Hence there is a definite need to identify potential CMFs and to develop both a qualitative and a quantitative assessment modeling methodology to assess their effects.

2.1 *Identification of CMFs within a Fault Tolerant System using FCRs*

For a digital embedded system, the fault space can be assumed to be infinite; that is, there are an infinite number of potential faults within the system software and/or the hardware. The cause of a fault may be an implementation mistake, external cause such as lightning, a specification mistake or any random cause. Once a fault manifests itself and is not detected, it then becomes an error, which is a corruption of the information that the system is processing. If the error is not detected, it then results in a system failure, The interaction of these events is shown in Figure 2.

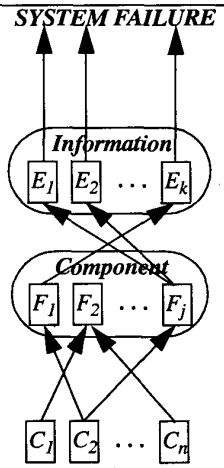

Figure 2: Fault propagation in a digital embedded system

In Figure 2, the underlying cause of a system fault is denoted by $C_1, C_2, \ldots C_n$ and will lead to a fault or faults in the software and/or hardware components, denoted by $F_1, F_2, \ldots F_j$. These faults, if undetected, can then cause one or more errors denoted by $E_1, E_2, \ldots E_k$ that will manifest themselves within the information being processed by the system. If the errors go undetected, then system failure occurs. In this example, no form of redundancy is considered, however, the construct shown in Figure 2 is applicable to such systems.

Assume that m identical or nearly identical components are used in a system to achieve any of the fault tolerant schemes discussed in Section 2: Model Development. The fault propagation model for this system is shown in Figure 3. In this model, the system fault space is the sum of the fault space for each component contained within the system; that is,

$$\Omega_{System} = \Omega_1 + \Omega_2 + \ldots + \Omega_m \qquad (2\text{-}1)$$

where Ω represents the fault space. In this model, the assumption that allows for this statistically independent behavior of the system fault space is that the fault space for each component is itself a FCR. This assumption implies that for each identical or nearly identical component the fault space, though independent, is identical and that the faults affect only those components with which they are associated via the FCR. However, the activation of faults within these components can be attributed to the same cause and potentially result in CMFs. For digital embedded systems, these faults can occur in hardware and/or software; however, it is nearly impossible to

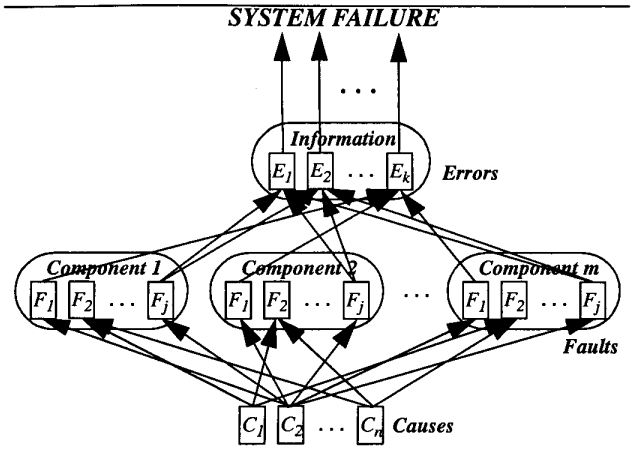

Figure 3: Fault propagation in fault tolerant digital embedded systems

determine whether the resulting failure is due to a fault in the hardware and/or the software. Hence, it is impractical to model failure events in terms of the fault space alone for digital embedded systems. By extending the concept of FCRs to informational flow of a system, then the effects of CMFs on digital embedded systems can be qualitatively and quantitatively analyzed.

2.2 *Information Flow Modeling*

In developing FCRs, the principle concept was to prevent the effects of faults in one region from interfering with another region. This concept can be extended to the information level of a system as shown in Figure 4. By isolating the information

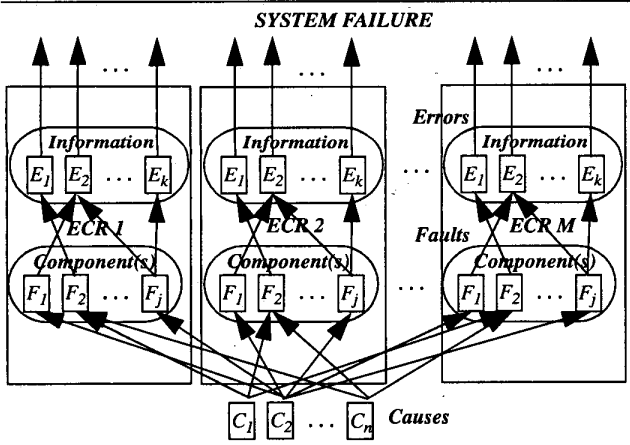

Figure 4: Error containment regions for digital embedded systems

being processed by a component or collection of components to a single region, the concept of Error Containment Regions (ECRs) is developed. In an ECR, it is assumed that any error that passes beyond the boundary of a given ECR will result in system failure. Obviously, this modeling scheme is pessimistic in nature in that it is possible that one ECR may be able to detect and to prevent an error from another ECR from causing a system failure. This pessimistic approach, however, provides a means of measuring the lower bound for system coverage [9, 14]. Since an error that crosses an ECR boundary is assumed to result in system failure, which arises from the assumed independence of each ECR, then the lower bound for the system coverage is

$$C_{System(\ lower\ bound)} = C_{ECR\ 1} \times C_{ECR\ 2} \times \ldots \times C_{ECR\ M} \quad (2\text{-}2)$$

where

$$C = \frac{\text{Number of errors detected}}{\text{Number of erors injected}} \quad (2\text{-}3)$$

However when two errors that result from the same underlying cause cross their respective ECR boundaries, then such conditions can be identified as a potential CMF. Thus, this modeling methodology provides both a qualitative basis for modeling CMFs and basis for the quantitative analysis of the effects of CMFs in digital embedded systems.

3. System Analysis using the Information Flow

In applying the concepts of ECRs as developed in Section 2.2: Information Flow Modeling, a system level modeling scheme that reflects the information flow of a system must be used. The data flow methodology (Ref. 13) provides such a technique. The system can be modeled based on its behavioral description using a hardware description language and this model maps the flow of data through the system. Hence, the data flow approach alleviates the need to separate between the hardware and software components and allows unified modeling of the digital embedded system.

In developing a data flow representation of a system, the first step is to build a graphical model of the data flow of the system. A set of modules is developed to represent both the logical and the simple arithmetic functions, which are performed by either hardware or software. The data flow representation of the various modules used in the data flow graph are shown in Figure 5. The actual data flow model is described using a

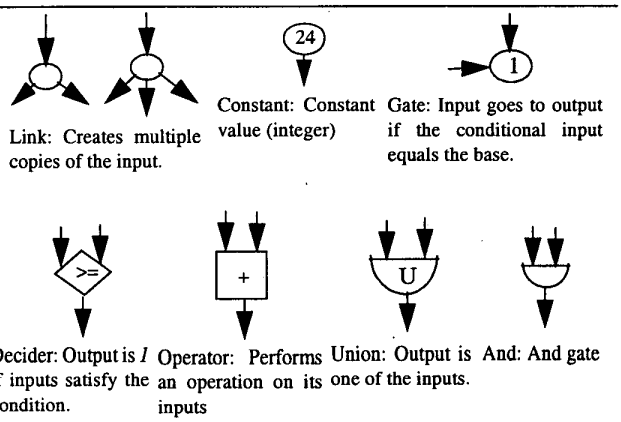

Figure 5: Building blocks of the data flow model

hardware description language, which is used for system simulation. Additionally, the data flow methodology supports error injection, which is required for analyzing the effects of CMFs. The error vectors used when triggering the error injection nodes are derived from the Dynamic Flowgraph Methodology (DFM) [15-26].

The DFM, which is derived from logic flowgraph

methodology (Ref. 27), is an integrated approach to modeling and analyzing the behavior of software driven digital embedded systems. The DFM follows the fault tree [28, 29] approach, but it incorporates the dynamic behavior of embedded systems; that is, it incorporates timing information. It utilizes a modeling framework in which system models are developed in terms of the causal relationships between physical variables and temporal characteristics of the execution of software. The DFM produces a self-contained system model from which many fault trees can be derived using an algorithmic technique. It is based on multi-valued parameter discretization and logic. Hence, the analytical insight of fault trees derived from this methodology is not limited by the constraints of binary logic used in conventional fault tree analysis. The DFM can produce timed fault trees in which timing relations between key system and parameter states are systematically and formally taken into account.

The placement of the error injection modules used in the data flow can be identified from the DFM. In addition, the DFM model allows for the verification of the system's functional behavior by obtaining the set of all possible input vectors to the various internal components contained within the system. This vector selection process is achieved by performing a deductive analysis on the system or architecture under test.

3.1 *System Analysis*

In order to demonstrate the application of ECRs in the assessment of CMF in digital embedded systems, a sample system reflective of the architecture used in a nuclear power plant's reactor protection system was analyzed. This architecture, shown in Figure 6, contains four channels, each

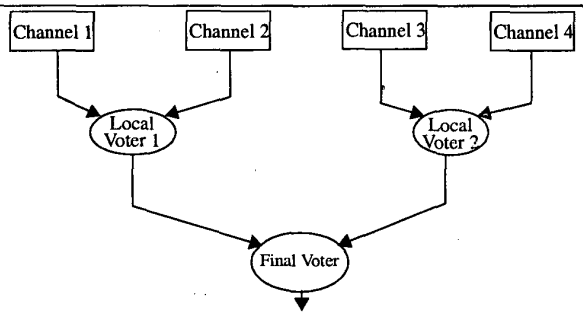

Figure 6: Block diagram of the *Quad* architecture

having identical inputs, divided into two groups of four and is referred to in the nuclear industry as a *Quad*. The channels within each group are compared using a local voter. The outputs of the two local voters are then compared by the final voter. During actual operations, one of the channels is always off-line (for maintenance purposes) and the system operation is referred to as a 2/4 system; that is, the output is correct as long as 2 of the 4 channels are working properly.

3.1.1 *Data Flow Model*

The data flow model of the *Quad* system is shown in Figure 7. From this graphical model, the information flow through the system and the resulting ECRs are defined. To enable the data flow model to perform error injection, special modules called

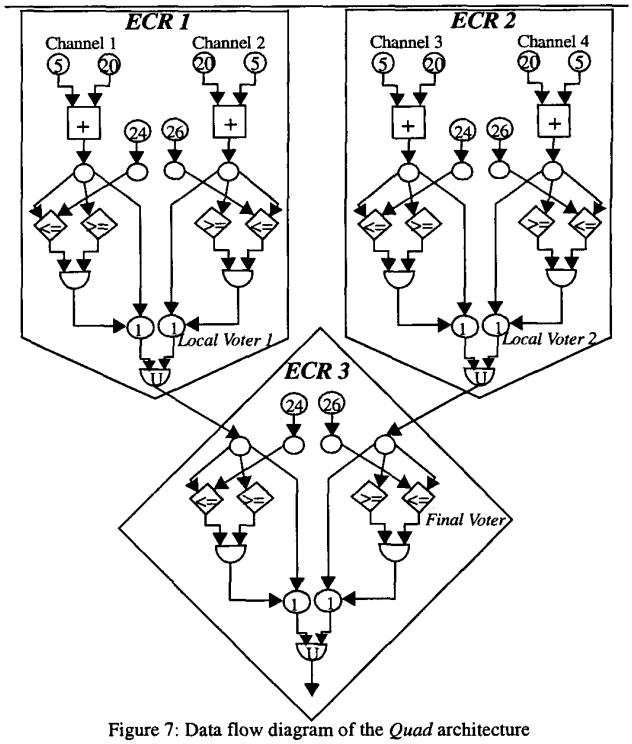

Figure 7: Data flow diagram of the *Quad* architecture

error injection modules need to be added to the data flow model. The placement of these modules is obtained from the DFM analysis of the system under consideration. Once the data flow model with the error modules is built, the error injection modules can then be exercised with the appropriate error information.

3.1.2 *DFM Analysis*

In order to define the error vectors associated with the various components and to determine where the data flow error injection modules need to be placed, a DFM was generated, which is shown in Figure 8. The DFM reflects the behavior of the *Quad* system via a collection of transition boxes and their interconnections. Decision tables describing the behavior of the transition boxes are developed as part of the analysis. These tables provide a mechanism to determine the value of the output of the box for a given set of inputs as well as the inputs which will cause a certain output to be obtained. The resulting decision tables for the *Quad* system are shown in Table 1.

Using the generated lookup table from the DFM analysis, a fault tree for the entire system can be developed. Since the DFM uses a multi-valued logic, the procedure used to generate the fault tree differs from the conventional top-down approach. First, the transition box corresponding to the top level is identified, and for the *Quad* system, the $T9_{ECR3}$ transition box of the final voter is the top level. Next, the decision table of the $T9_{ECR3}$ transition box is checked. From this check, it is determined that a faulty top level output has an OR behavior. This process is repeated until the fault tree for the entire system is generated. This fault tree is shown in Figure 9.

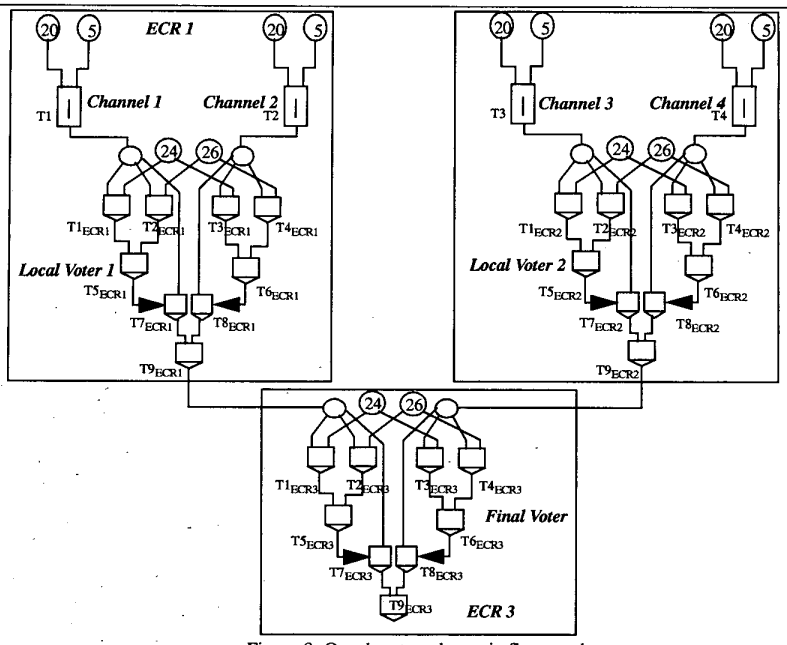

Figure 8: *Quad* system dynamic flowgraph

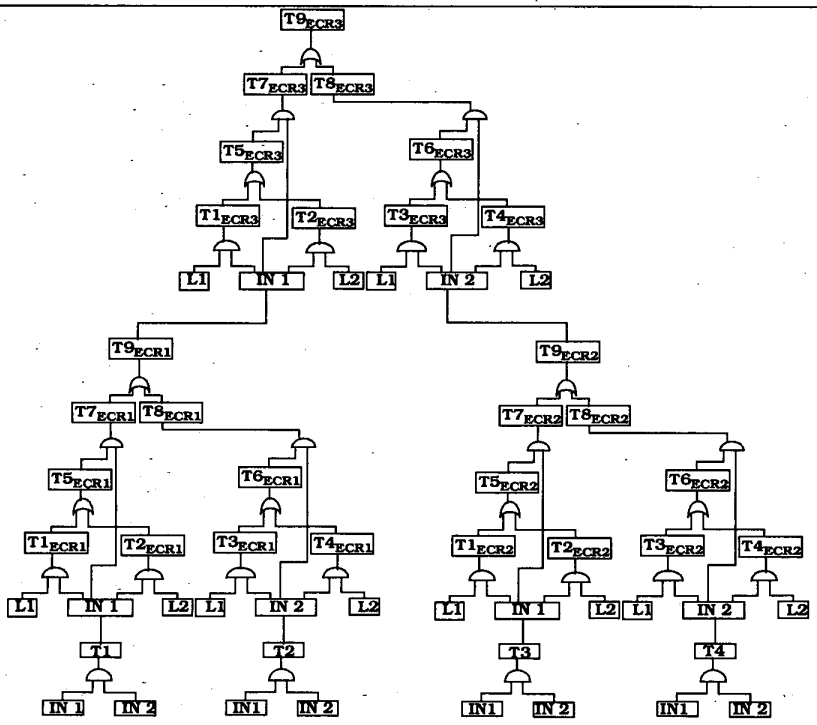

Figure 9: Fault tree for the *Quad* system

By giving specific values to the top level of the fault tree, the fault vector for a given top output can be obtained using deductive analysis. In this deductive analysis, a top event is selected and the decision tables are traversed. From this analysis, the vectors for the final voter with a top level output '*Z*' are obtained. The results of this analysis are presented in Table 2

From this DFM analysis, the $T1_{ECR}$,$T2_{ECR}$,$T3_{ECR}$ and $T4_{ECR}$ transition boxes in each of the local voters and the transition boxes in each of the channels are identified as candidates for injecting errors. This result arises because the $T1_{ECR}/T2_{ECR}$ and the $T3_{ECR}/T4_{ECR}$ transition box pairs serve as *deciders*; that is, they compare the two inputs and decide which input is the greater or the smaller of the two. A pair of *deciders* establishes whether the input from the channel is between the two allowable limits. Since each local voter has

Table 1: Transistion box decision tables

I: Correct Value, X unknown/null, Z: Faulty not null value																		
T1/T2/T3/T4				$T1_{ECR}/T2_{ECR}$			$T3_{ECR}/T4_{ECR}$			$T5_{ECR}/T6_{ECR}$			$T7_{ECR}/T8_{ECR}$			$T9_{ECR}$		
In_1	In_2	F	Out	F	In_2	Out	F	In_2	Out	In_1	In_2	Out	In_1	In_2	Out	In_1	In_2	Out
I	I	I	I	I	I	I=1	I	I	I=1	I=1	I=1	I=1	I=1	I	I			
I	Z	I	Z	I	X	I=0	I	X	I=0		I=0	I=0		X	X			
X	I	I	X	I	Z	I=0	I	Z	I=0		Z=1	Z=1		Z	Z			
X	Z	I	X	Z	X	Z=1	Z	X	Z=1		Z=0	Z=0	I=0	I	X	I	I	I
I	I	Z	Z	Z	X	Z=0	Z	X	Z=0	I=0	I=0	I=0		X	X	X	I	I
I	Z	Z	Z	Z	Z	Z=1	Z	Z	Z=1		Z=1	I=0		Z	X	Z	I	Z
X	I	Z	Z	Z	I	Z=0	Z	I	Z=0		Z=0	I=0	Z=1	I	Z	Z	I	I
X	Z	Z	Z							Z=1	Z=1	Z=1		X	Z	X	X	X
											Z=0	I=0		Z	Z	Z	X	Z
													Z=0	I	X	Z	Z	Z
														X	X			
														Z	X			

Table 2: Vectors for final voter for output 'Z'

In_{ECR3}: $T9_{ECR1}$	F1	T1 ECR3	F2	T2 ECR3	T5 ECR3	T7 ECR3	In_{ECR3}: $T9_{ECR2}$	F3	T3 ECR3	F4	T4 ECR3	T6 ECR3	T8 ECR3	T9 ECR3
X	Z	Z=1	Z	Z=1	Z=1	Z	X	I	I=0	I	I=0	I=0	X	Z
							X	I	I=0	Z	Z=1	I=0	X	
							X	I	I=0	Z	Z=0	I=0	X	
							X	Z	Z=0	Z	Z=0	I=0	X	
							Z	I	I=0	I	I=0	I=0	X	
							Z	I	I=0	Z	Z=1	I=0	X	
							I	I	I=1	Z	Z=0	Z=0	X	
Z	Z	Z=1	Z	Z=1	Z=1	Z	X	I	I=0	I	I=0	I=0	X	
							X	I	I=0	Z	Z=1	I=0	X	
							X	I	I=0	Z	Z=0	I=0	X	
							X	Z	Z=0	Z	Z=0	I=0	X	
							Z	I	I=0	I	I=0	I=0	X	
							Z	I	I=0	Z	Z=1	I=0	X	
							I	I	I=1	Z	Z=0	Z=0	X	
X	Z	Z=1	Z	Z=1	Z=1	Z	X	Z	Z=1	Z	Z=1	Z=1	Z	
							Z	Z	Z=1	Z	Z=1	Z=1	Z	
Z	Z	Z=1	Z	Z=1	Z=1	Z	X	Z	Z=1	Z	Z=1	Z=1	Z	
							Z	Z	Z=1	Z	Z=1	Z=1	Z	
X	Z	Z=1	Z	Z=1	Z=1	Z	I	I	I=1	I	I=1	I	I	
Z	Z	Z=1	Z	Z=1	Z=1	Z								

two pair of *deciders*, each voter has four potential internal causes of error. The T1 box in each channel is implemented as an addition function which could be performed incorrectly. Thus, each channel has a potential internal cause of error.

Additionally, the DFM analysis identified a set of 77 possible input vectors for the final voter, where 7 of them allowed for normal operation, 20 vectors led to unsafe failure and 50 vectors led to safe failure. It is the set of 70 vectors that led to failure that need to be analyzed to determine if they result in a CMF. From this analysis, various scenarios for CMFs can be identified. For example, there is a potential for a CMF between ECR 1 and ECR 2 if the local voter contains an undetected error and at least one of the inputs channels contains an error as shown in Table 3 and Table 4. In the case illustrated, the two local voters have the same transition box, $T2_{ECR}$, failing in the same manner and the system output is corrupt, which is a CMF.

Table 3: Final voter condition

In_{ECR3}: $T9_{ECR}$:	F1	T1 ECR3	F2	T2 ECR3	T5 ECR3	T7 ECR3	In_{ECR3}: $T9_{ECR2}$	F3	T3 ECR3	F4	T4 ECR3	T6 ECR3	T8 ECR3	T9 ECR3
Z	Z	Z=1	Z	Z=1	Z=1	Z	I	I	I=1	I	I=1	I	Z	Z
X	I	I=0	I	I=0	I=0	X	X	I	I=0	I	I=0	I=0	X	X

Table 4: Local voter 1 and local voter 2 condition (same vector)

In_{ECR}	F1	T1 ECR	F2	T2 ECR	T5 ECR	T7 ECR	In_{ECR}	F3	T3 ECR	F4	T4 ECR	T6 ECR	T8 ECR	T9 ECR
I	I	I=1	Z	Z=0	Z=0	X	I	I	I=0	I	I=0	I=0	I	I

3.1.3 *Simulation Results*

By applying the results of the DFM analysis to the information flow model of the *Quad* system, the system coverage (Ref. 9) can be estimated using a VHDL model of the data flow. This estimation includes the effect of the CMFs as identified by the DFM analysis. By injecting a set of 100,000 random errors that included those errors identified in the DFM analysis as a potential cause of CMFs, the coverage estimate for each ECR was found to be $\hat{C}_{ECR\,1} = 0.9933$, $\hat{C}_{ECR\,2} = 0.9933$ and $\hat{C}_{ECR\,3} = 1$. The estimate for ECR 3 assumes perfect coverage because no errors were injected into ECR 3; that is, the inputs were assumed to be perfect. The resulting coverage estimate for the *Quad* system using the ECRs coverage levels was

$$\hat{C}_{System\ ECRs} = 0.9933 \times 0.9933 \times 1 = 0.9866 \qquad (3\text{-}1)$$

The coverage estimate for the entire system without consideration to the ECRs was

$$\hat{C}_{System} = 0.9995 \qquad (3\text{-}2)$$

This coverage estimate using ECRs is pessimistic in nature because, as stated in Section 2.2: Information Flow Modeling, system failure is assumed whenever an error crosses an ECR boundary. However, it provides the designer with an indication as to the potential problems that CMFs may pose to the system and it provides a lower bound for the estimate of the system coverage.

4. Conclusions and Recommendations

By developing both a qualitative and a quantitative information level model of a digital embedded system, it can be viewed as an integrated entity; that is, there is no need to distinguish between the hardware and the software components. In extending the concept of FCRs to information flow modeling via ECRs, CMFs can be modeled in a coherent fashion without knowing if their underlying cause is in either the hardware or the software. The use of DFM provides a quantitative approach for determining where to inject errors within a given ECR and what errors may lead to CMFs.

In this analysis, it is assumed that whenever an error crosses the boundary of an ECR, then system failure occurs. This modeling approach is pessimistic in nature because it is possible for another ECR to prevent an error that is undetected in another ECR from manifesting itself as a failure. Hence, it provides a lower bound for coverage estimation. However when two errors that result from the same underlying cause cross their respective ECR boundaries, then such conditions can be identified as a potential CMF.

ACKNOWLEDGEMENTS

This work was sponsored by the United States Nuclear Regulatory Commission under grant NRC-04-97-067.

REFERENCES

1. NASA Contractor Report 189632, Volume II, "Advanced Information Processing System: The Army Fault Tolerant Architecture Conceptual Study," pp. 8.1-8.36, July 1992.

2. Hagen, E.W., "Common-mode/Common-cause Failure: A Review," *Nuclear Engineering and Design*, vol. 59, pp. 423-431, 1980.

3. National Research Council, *Digital Instrumentation and Control Systems in Nuclear Power Plants: Safety and Reliability Issues*, National Academy Press, pp. 43-51, 1997.

4. Dhillon, B.S., *Reliability Engineering in Systems Design and Operation*, Van Nostrand Reinhold Company, pp. 70-71, 1983.

5. Gangloff, W.C., "Common-mode Failure Analysis," *IEEE Transactions on Power Apparatus and Systems*, vol. PAS-94, no. 1, pp. 27-30, January - February 1975.

6. Watson, I.A., and G.T. Edwards, "Common-mode Failures in Redundant Systems," *Nuclear Technology*, vol. 46, no. 2, pp. 183-91, November - December 1979.

7. Colombo, A.G., and A. Z. Keller, editors, *Reliability Modeling and Applications: Proceedings of the ISPRA Course, held at Joint Research Centre, Ispra, Italy*, D. Reidel Publishing Company, pp. 1-32, 1987.

8. Lala, J.H., and R.E. Harper, "Architectural Principles for Safety-critical Real-time Applications," *Proceedings of the IEEE*, vol. 82, no. 1, pp.25-40, January 1994.

9. Johnson, B.W., *Design and Analysis of Fault Tolerant Digital Systems*, Addison Wesley, New York, 1989.

10. Voas, J.A. Ghosh, F. Charron, and L. Kassab, *Reducing Uncertainty about Common Mode Failures*, http://www.rstcorp.com.

11. Davis, C., "Common Mode Failures in Information Systems," *SciTech Journal*, July-August 1995.

12. NUREG/CR- 6303, *Method for Performing Diversity and Defense-in-depth Analyses of Reactor Protection Systems*, prepared by G. G. Preckshot, Lawrence Livermore National Laboratory.

13. Choi, C.Y., *Dependable System Hardware/Software Codesign using Data Flow Models*, Doctoral Dissertation, University of Virginia, 1997.

14. Kaufman, Lori M. and Barry W. Johnson, "The Importance of Fault Detection Coverage in Safety Critical Systems," Proceedings of the Twenty-Sixth Water Reactor Safety Information Meeting, Volume 2, U.S. Nuclear Regulatory Commission, NUREG/CP-0166, October 1998, pp. 5-28.

15. Garrett, C., S.B. Guarro and G.E. Apostolakis, "The Dynamic Flowgraph Methodology for Assessing the Dependability of Embedded Software Systems," *IEEE Transactions on systems, man and cybernetics*, vol. 25, no. 5, May 1995, PP 824-40.

16. Cosgrove, J., S. Guarro, G. Romanski, M. Yau, "Dynamic Modeling and Verification of Safe set Architectures," *Proceeding of Wescon/96*, IEEE, no.ix+682, pp. 528-540.

17. Garrett, C., M. Yau, S. Guarro and G. Apostolakis, "Assessing the Dependability of Embedded Software Systems using the Dynamic Flowgraph Methodology," *Dependable Computing for Critical Applications 4* (J. F. Meyer and R. D. Schlichting eds.), Springer Verlag, New York, pp. 161-184, 1995.

18. Garrett, C., M. Yau, S. Guarro and G. Apostolakis, "The Dynamic Flowgraph Methodology for Assessing the Dependability of Embedded Software Systems," *IEEE Transaction on Systems, Man and Cybernetics*, vol 25, no. 5, pp. 824-840, May 1995.

19. Guarro, S., M. Yau, "Analysis of control software in Advanced reactors using the dynamic flowgraph methodology(DFM)," *Proceedings of the 1996 American Nuclear Society International Topical Meeting on Nuclear Plant Instrumentation, Control And Human Interface Technologies*, vol. 2, 1996,pp. 1025-32.

20. Milici, A., J.-S. Wu and G. Apostolakis, "The Use of the Dynamic Flowgraph Methodology In Modeling Human Performance and Team Effects," *Proceedings of the 1996 American Nuclear Society International Topical Meeting on Nuclear Plant Instrumentation, Control And Human Interface Technologies*, vol. 2, 1996,pp. 653-59.

21. NUREG/CR-6465, *Development of Tools for Safety Analysis of Control Software in Advance Reactors*, prepared by S. Guarro, M. Yau and M. Mohamed, ASCA, Inc.

22. Yau, M., S. Guarro, R. Mulvihill, T. Milici, "Application of Dynamic Flowgraph Techniques for Safety Analysis and Testing of Space Systems Software," ASCA Inc., Final Report prepared for NASA, Lewis Research Center.

23. Yau, M., S. Guarro, G. Apostolakis, "Demonstration of the Dynamic Flowgraph Methodology using the Titan II Space Launch Vehicle Digital Flight Control System," *Reliability Engineering and System Safety*, vol. 49, pp. 335-353, 1995.

24. Yau, M., Apostolakis, G., Guarro, S., "The Use of Prime Implicants in the Dependability Analysis of Software Controlled Systems," ACSA Inc. (Accepted for Publication in *Reliability Engineering and System Safety*).

25. Yau, M., S. Guarro, S., R. Mulvihill and T. Milici, *Application of Dynamic Flowgraph Techniques for Safety Analysis and testing of Space Systems software*, ASCA, Inc., Final report prepared for National Aeronautics and Space Administration, Lewis Research Center.

26. Yau, M., S. Guarro and G. Apostolakis, "Demonstration of the Dynamic Flowgraph Methodology using the Titan II Space launch Vehicle Digital Flight Control System", *Reliability Engineering and System Safety*, vol. 49, 1995, pp. 335-353.

27. Guarro, S. and D. Okrent, "The Logic Flowgraph: A New Approach to Process Failure Modeling and Diagnosis for Disturbance Analysis Applications," Nuclear Technology, vol. 67, 00. 11-22, pp. 348-359.

28. Dugan, J. B., S. J. Bavuso and M. A. Boyd, "Dynamic Fault Tree Models for Fault Tolerant Computer Systems," *IEEE Transaction on Reliability*, vol. 41, no. 3, pp. 363-376, June, 1992.

29. Modarres, M., What Every Engineer Should Know about Reliability and Risk Analysis, Marcel Dekker, New York, 1993.

BIOGRAPHIES

Lori M. Kaufman, *Ph.D.*
Department of Electrical Engineering
Thornton Hall
University of Virginia
Charlottesville, VA 22903-2442 USA

Internet (e-mail): lmk2q@virginia.edu

Lori M. Kaufman is currently a post-doctoral Research Associate in Electrical Engineering at the University of Virginia. She received the B.S., the M.S. and the Ph.D. degrees in Electrical Engineering from the University of Virginia in 1987, 1993 and 1997 respectively. From 1987-1991, Ms. Kaufman worked in industry. Her current research interests include software/hardware reliability engineering, fault tolerant computing, and analytical modeling using fault trees, Markov models, Petri nets, statistics of the extreme and simulation. Ms. Kaufman is member of Eta Kappa Nu, Tau Beta Pi and the IEEE.

Sanyukat Bhide.
Department of Electrical Engineering
Thornton Hall
University of Virginia
Charlottesville, VA 22903-2442 USA

Internet (e-mail): sanyukta@virginia.edu

Sanyukta Bhide is currently a graduate student in Electrical Engineering at the University of Virginia. She received the B.S. degree in Electrical Engineering from the University of Mumbai in 1997. She has been working on her M.S. since Fall '97. Her research interests include hardware/software reliability engineering, fault tolerant computing and hardware design. Ms. Bhide is a member of the IEEE.

Barry W. Johnson, *Ph.D.*
Department of Electrical Engineering
Thornton Hall
University of Virginia
Charlottesville, VA 22903-2442 USA

Internet (e-mail): bwj@virginia.edu

Barry W. Johnson received the B.S., M.E., and Ph.D degrees in electrical engineering from the University of Virginia, Charlottesville, Virginia, in 1979, 1980, and 1983, respectively. Dr. Johnson is currently a Professor in the Department of Electrical Engineering at the University of Virginia. He is a co-founder and the director of the Center for Semicustom Integrated Systems, a Technology Development Center of Virginia's Center for Innovative Technology. He is also a co-founder and co-director of the Center for Safety-Critical Systems. He is a Fellow of the IEEE and is active in the IEEE as a member of the IEEE Board of Directors, a member of the IEEE Computer Society Executive Committee, and a member of the IEEE Computer Society Board of Governors. Previously, he has served as the IEEE Computer Society's President, Vice President for Publications, Vice President for Conferences and Tutorials, Vice President for Press Activities, and Vice President for Membership Activities.He is also a member of Tau Beta Pi, and Eta Kappa Nu.