Application of Syndrome Pattern-Matching Approach to Fault Isolation in Avionic Systems

Robert Hammett and Robert Luppold

Technical Staff, The Charles Stark Draper Laboratory, Inc.

Cambridge, Massachusetts

Abstract

Future avionic systems will utilize fault-tolerant architectures and sophisticated software algorithms to provide automatic fault accommodation and dependable maintenance advisories. Currently, modular, fault-tolerant avionic hardware concepts are under development for applications such as commercial transport aircraft, military aircraft, space systems, and underwater vehicles. To maximize the potential of these types of systems, a systematic approach to the design of Fault Detection, Isolation, and Reconfiguration (FDIR) algorithms will be required. The approach must address the architectural complexity of these systems, and produce algorithms which take a more global approach to achieving precise, reliable FDIR. The key element of reliable, automatic reconfiguration of critical functions and dependable maintenance diagnostic information is the fault isolation resolution.

Introduction

This paper examines several attributes of an FDIR approach that we categorize as *syndrome pattern matching*. This approach is based on the premise that effective fault isolation algorithms must possess the capability of identifying patterns in the results of many fault detection tests, i.e., the fault syndrome. Examples of tests that comprise the syndrome are comparisons of redundant signals, hardware built-in test (BIT) results and status indications from other systems or equipment. Algorithms designed using this approach can take advantage of the high levels of interdependency present in redundant, fault tolerant architectures. The propagation of faults within the system imparts a unique pattern to the syndrome, and allows the algorithm to improve fault isolation performance. Another benefit of this approach is that the basis for determining meaningful patterns can now be traced directly to fault dependencies in the system. This allows the systematic creation of a matrix of component-failure-to-syndrome-dependency relationships that contain the meaningful syndrome patterns and exposes ambiguous failure conditions and redundant fault detection tests. Moreover, the dependency information can be derived directly from CAD/CAE representations of the system. The implementation impact of pattern-matching algorithms on embedded computer throughput is minimal and therefore, does not influence system cost, weight, or reliability.

With the isolation algorithm design task reduced to a manageable problem and the FDIR logic constrained to a conceptually manageable framework, the designer can turn his attention to further improvements in FDIR performance. Performance issues discussed in this paper include: (1) intermittent faults, (2) multiple sequential failures, and (3) unanticipated syndromes.

Background

The development of software FDIR for digital control systems evolved from BIT concepts developed for earlier analog control systems. By their nature, these analog BIT circuits tested small groups of functionally related signals and by means of comparisons deduced if failures had occurred. In some cases, this BIT was used to provide automatic reconfiguration. Often, these tests could not distinguish between failures of components within the control unit electronics and failures of sensors or effectors outside of the electronics. In addition, certain single-point failures could cause multiple BIT tests to fail, causing undesirable system reconfiguration and/or indications to the user.

The introduction of digital computers provided the opportunity to overcome these limitations by allowing virtually unlimited logical manipulation of the BIT information and by providing sophisticated reconfiguration

CH3030-4/91/0000-0056 $1.00

and indication capabilities. Early efforts simply replaced analog BIT circuits with software tests. Later efforts added more sophisticated tests and, of interest in this discussion, logic to infer single points of failure from multiple test failure indications.

The typical design process for FDIR logic is a series of interactions between system, circuit, and software designers. The usual starting point, as with analog BIT, is to devise tests to address small groups of functionally related signals. For these small groups, algorithms are developed to isolate the cause of the failure and reconfigure the system to accommodate and report the problem. Frequently it is recognized that the system has additional, useful information from other signal groups and logic is added, on an ad hoc basis, to take advantage of this information to improve fault detection or isolation for the small group. Another realization often occurs at this time that combinations of signal group failure indications can be interpreted to provide detection and isolation of "higher level" system failures (such as power loss or other common resource failures). Besides the advantage this offers in terms of being able to correctly reconfigure and report the failure, this may also expose latent failures of other signals that were undetected (uncovered) due to limitations of tests performed on them. Because this logic to identify combinations of signal group failures was appended or added on to the existing small signal group logic, it becomes a difficult software design problem to allow the added logic to override the conclusions and actions of the original small signal group logic. In addition, as systems become more complex with multiple channels of redundancy and cross-strapping of input and output resources, system-level interactions become quite complex. These complex interactions begin to challenge the designer's ability (and available time) to identify them and to devise logic to exploit (or at least to avoid undesired actions from) these interactions. Finally, as the add-on logic becomes more complex, FDIR software consumes a high percentage of the available computational resources. This results in increased system cost.

The desire to devise a new design approach that simplifies the design task, provides optimal FDIR performance, and limits the impact of FDIR on the computational resource utilization motivated the development of the syndrome pattern-matching approach.

Traditional FDIR Logic

Figure 1 illustrates a typical implementation of FDIR logic. It is characterized by a number of related signal failure detection tests that drive local isolation logic. This local isolation logic typically is a logical state machine, that is, combinatorial logic that makes use of both the current inputs and logical state history from previous sequences of inputs. The outputs of this local isolation logic become inputs to the global isolation logic. This global logic is intended to interpret the significance of combinations of local group failures and is typically also a state machine. With knowledge of isolated failure conditions, derived from either the local or global isolation logic, reconfiguration and reporting logic is activated to achieve the desired system response to the failure. Each local isolation logic block receives as inputs a detected failure primarily from the local group and sometimes from other groups on an ad hoc basis as defined by the designer. The local isolation logic may also make use of its previous state, the state of other local isolation logic groups, and the state of the global isolation logic.

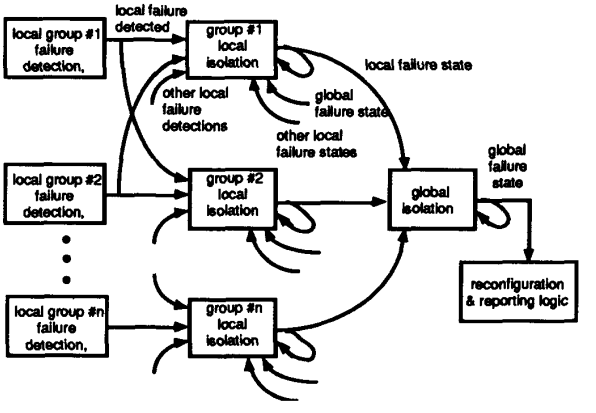

Figure 1. Typical FDIR implementation.

Syndrome pattern matching approach

In contrast to the typical approach, Figure 2 illustrates the syndrome pattern-matching concept.

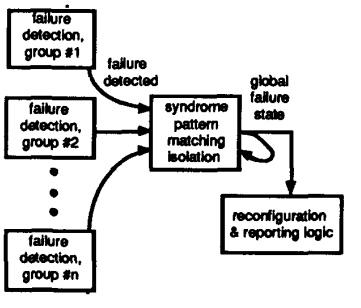

Figure 2. FDIR using syndrome pattern matching.

Note that in the figure, the local and global isolation logic of the typical approach has been replaced with the syndrome pattern-matching logic. The syndrome pattern matching logic, which is also a state machine, replaces the complex interaction of the various logic elements in the traditional approach with a centralized logic structure. The logical complexity resulting from the large number of inputs to the syndrome-matching logic (perhaps hundreds for an aircraft modular avionics system) is overcome by implementing the combinatorial logic required in the form of a pattern-matching table.

Figure 3 illustrates the concept of the pattern-matching table. It consists of a tabulation of failure detection test results that form a pattern (the syndrome), and the failure condition required to produce the pattern. The test results in the syndrome are expressed as logical true or false values. The failure condition responsible for the pattern may be a particular component or it may be an ambiguity group of components. It is important to realize that not all possible patterns are included in the table since this would result in unwieldy table sizes and many undefined and highly unlikely failure patterns. Patterns that do not match table entries can be categorized as "unanticipated failure conditions" and should result in conservative reconfiguration and reporting actions or may be subject to additional interpretation as described later in this paper.

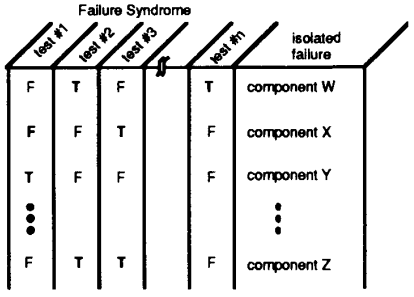

Figure 3. Example of syndrome pattern table.

The syndrome pattern-matching method outlined presents a deceptively simple approach to implementing efficient and effective FDIR logic. The hidden challenge is the analysis required to complete the table. For simple systems or parts of systems, the table can be completed by an inspection process of postulating failures and identifying which syndrome test(s) will be affected, thus determining the syndrome. For more complex systems, a more systematic analysis method is needed to define the syndrome-to-failure mapping.

Syndrome-to-Failure Analysis by Inspection

Figure 4 presents an example of a simple system that illustrates how the syndrome to failure mapping can be generated by inspection.

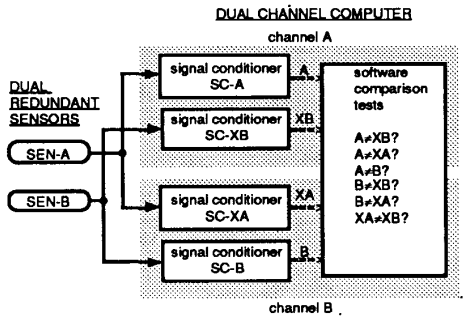

Figure 4. Example system.

The system consists of a dual channel computer and dual redundant analog output sensors which forms a dual channel sensing system. The two sensors, SEN-A and SEN-B, are measuring the same physical quantity (i.e. a temperature, pressure, position,....). The output of the sensors are converted to a digital value by signal conditioning circuits in the computers. Identical software in each computer channel can act on any one of the four converted digital sensor values. Each sensor output is processed by two signal conditioning circuits, one in each of the two channels. This is an example of sensor cross-strapping and offers two benefits: 1) Complete failure of either channel of the computer does not prevent the other channel of the computer from accessing both of the sensor's. 2) Any single signal conditioning circuit failure can be detected, isolated and accommodated. The signal conditioning circuits are referred to as SC-A for the "A" channel signal conditioning of the "A" sensor, SC-XB for the "A" channel signal conditioning for the cross-strapped "B" sensor input, SC-XA for the "B" channel signal conditioning for the cross-strapped "A" sensor input and SC-B for the "B" channel signal conditioning for the "B" sensor input. The outputs of the four signal conditioning circuits are available as software variables within the digital computers of each of the control channels. For simplicity, it is assumed that the two channel system computer operates as a fault tolerant pair and failures of the computers need not be considered. Since there are four unique inputs A, XB, XA, and B, there are six possible comparison tests between the inputs in which to detect significant deviations, A≠B, A≠XB, A≠XA, B≠XB, XA≠XB, and B≠XA. A significant deviation exists if the two inputs, when compared, exceed some allowable

threshold of difference. A "failed test" condition occurs if the comparison of the two sensors exceeds an allowable threshold. The six true or false logical test failure indications form the failure syndrome.

To derive the syndrome-to-failure table, consider the case where the "A" sensor signal conditioner (SC-A) fails. Assuming that the failure results in the "A" signal being different than the correct value, it can be concluded by inspection that the test A≠B will fail (they will be different). Similarly, all tests using the variable "A" will fail (A≠XB, A≠XA). The other tests not using "A" will pass (B≠XB, XA≠XB, B≠XA). Similarly, assuming that only the "A" sensor (A-SEN) fails, both the variables "A" and "XA" will be equally affected and all tests involving "A" or "XA" but not A≠XA will fail. As a result, the tests A≠B, A≠XB, XA≠XB, B≠XA will fail, while B≠XB and A≠XA will pass. Continuing this process for the "B" sensor and the other three signal conditioners, the fault syndrome table of Figure 5 can be completed.

Failure Syndrome

XA≠B	A≠XB	A≠XA	B≠XB	B≠XA	XA≠XB	Isolated failure
T	T	F	F	T	T	SEN-A
T	T	F	F	T	T	SEN-B
F	T	T	F	F	F	SC-A
F	T	F	T	F	T	SC-XB
T	F	T	F	T	T	SC-XA
T	F	F	T	T	F	SC-B

Figure 5. Failure syndrome table for example problem.

An inspection of the syndrome patterns in the table reveals information about the fault isolation properties of the system architecture. First, the presence of identical patterns warns that the two failures associated with the pattern cannot be differentiated from each other and a fault ambiguity situation exists. For the example, this is evident in the failure syndrome patterns of SEN-A and SEN-B, which are identical and indicate that a failure of either the "A" or "B" sensor cannot be isolated. A second observation that can be made by inspection of the table is the presence of redundant tests in the syndrome. For this example, the tests A≠XB and XA≠XB can be eliminated from the syndrome without degrading the fault isolation capability for first failures. Finally, syndromes that do not match any of the patterns associated with single failures represent either impossible combinations of test failures, unanticipated failures, or patterns that would result from multiple failure conditions. For the example, there are 2^6 or 64 possible patterns while the table identifies only 6. Excluding the no-failure condition, this leaves 57 unanticipated failure patterns. The subject of responding to these unanticipated failures is addressed later in the paper.

In this example, determining the syndrome-to-failure table by inspection was straightforward. For the complex architectures inherent in modular avionics and other fault-tolerant systems, the analysis by inspection technique is inadequate and a more systematic approach will be needed.

A Systematic Approach to Syndrome Analysis

For complex architectures, propagation of the effects of faults within the system can influence many tests of the syndrome. This is particularly true of failures or disruptions of resources that are shared by many functions. Examples of such resources are power sources, data busses and multiplexed I/O components. This characteristic presents both a benefit and a challenge to the designer. The benefit is that a strong correlation between failures and the resulting syndrome can be constructed. This is the source of FDIR performance improvement inherent in the syndrome-matching approach. The challenge presented by complex fault propagation is that it becomes more difficult to analyze these dependencies and the potential for error is increased. To address the complexity issue, a systematic approach to determining the syndrome table must be developed. To accomplish this, the syndrome look-up FDIR design process needs to be thoroughly understood.

Figure 6 presents an illustration of the sequence of steps in the syndrome look-up design process. The analysis process involved in generating the syndrome look-up table is highlighted and the inputs to the analysis are identified. From the figure, it is clear that the first step in systematizing the process is to understand what aspects of the design are contained in the *FDIR-oriented design description* and how these might be represented to allow for a systematic analysis. Referring to the failure analysis by inspection description, it is clear that the structure, the paths on which signals travel, is a major part of the information of interest. This structural information must be derived from both the system architecture and the details of the circuit design. Another aspect of the structure is the end points of the paths, the failure detection tests. Physical packaging is also of interest as it defines the components and interconnects of the system. Ultimately, it would seem desirable to derive the FDIR oriented design description directly from the design CAD/CAE database. In fact, the analysis itself could become a feature of the CAE environment.

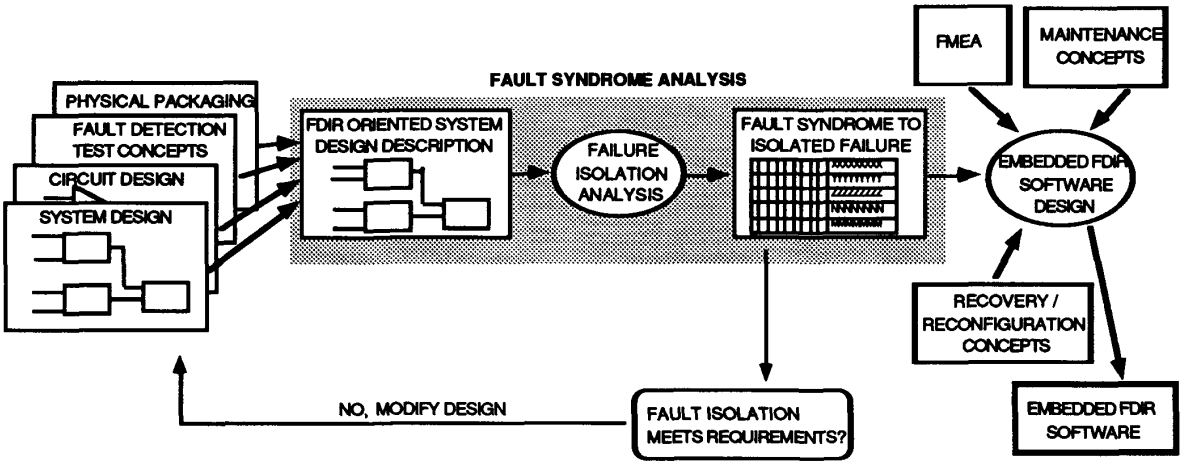

Figure 6 - Syndrome look-up design process

To reach this goal, the FDIR-oriented design description should take a form that is both man and machine readable. This machine-readable description can be thought of as a dependency model for the system. With this FDIR-oriented design description as input, many possible approaches to the analysis of the propagation of fault effects through the system and the resulting syndrome are possible. Developing an appropriate form for the design description is the topic of an ongoing research project at Draper Laboratory.

Further Improvements in FDIR - Building on the Syndrome Pattern-Matching Approach

With the isolation algorithm design task reduced to a manageable problem by syndrome matching and the FDIR logic constrained to a conceptually manageable framework, the designer can turn his attention to further improvements in FDIR performance. In particular, three topics are of immediate interest: (1) intermittent and transient faults, (2) multiple sequential failures, and (3) unanticipated syndromes.

Accommodating Intermittent and Transient Faults

Intermittent and transient fault conditions account for a high percentage of the failure detection events that the FDIR will encounter. An intermittent failure is an actual failure, requiring a repair to correct, that does not prevent the correct functioning of the affected component under all operating conditions. A transient condition is one in which the system is disturbed by external or usual events; correct operation returns once the disturbance is removed and no repair or permanent recovery action is required. It is widely accepted that transient failures are far more frequent than permanent failures and are a major source of avionic system false alarm failure indications. These false alarms result in unnecessary maintenance actions and are a major concern of the avionics maintenance community. Complicating the situation is the fact that many actual failures which the system must correctly detect, isolate, and recover from are intermittent. A major challenge to any FDIR design is to differentiate between transient and intermittent failure and act accordingly. The traditional approach to this problem is to employ counters or timers that measure the persistence of a detected failure then initiate recovery and reporting only in response to persistent conditions. In traditional designs, problems can occur when comparatively long events such as power interruptions are first detected as failures of individual sensors or signals and the system reacts to these conditions before the actual condition becomes apparent. With the syndrome pattern matching approach, it is feasible to identify the evolution of a syndrome from one type to another and reverse an earlier decision. In a traditional design, the software complexity required to perform this would discourage its incorporation. This ability to observe syndrome evolution also greatly simplifies dealing with conditions where all tests that form the syndrome do not operate continuously and must either be activated in response to some event or will only execute when the system is in a particular mode of operation.

An added benefit of the syndrome-matching logic is that the transient fault counting and timing logic is greatly simplified, resulting in software that is more efficient, easier to validate and more deterministic in it's operation. Another advantage is that the failure state of the

FDIR is more compact, thus it can more readily be stored or reset.

Accommodating Multiple, Sequential Faults

Another area of interest is that of multiple, sequential failures. A generally accepted ground rule for the design of FDIR is that *simultaneous*, multiple failures are unlikely and need not be addressed in the design. This ground-rule remains unchanged for syndrome-matching systems. The case of *sequential*, multiple failures (i.e.,two or more failures, at different times) is a potential but generally unlikely condition with today's systems. It is anticipated that these sequential, multiple failures will be a relatively common occurrence with new modular avionic systems. The reason for this anticipated increase is that new systems will use fault tolerance not only to allow for continued safe flight and landing, but also to make possible deferring maintenance to a more convenient time. As a result, these system will be routinely operated with existing failure(s).

The syndrome-matching approach is well suited to multiple, sequential failures as the effects of previous failures can be eliminated from the syndrome by masking out inactive or meaningless tests. Also, since the failure state of the system is defined by the syndrome and syndrome history, the system can be reset to reflect partial repairs without re-enabling components that were previously found failed and were not repaired.

Unanticipated Syndromes

A final area of interest is that of unanticipated syndromes. Designers of FDIR logic are frequently humbled when systems find modes of failure that they had not anticipated. An example of such a condition is when signals fail in such a way that some but not all, of the tests designed to detect this condition indicate failure. Although it does not seem possible to design systems that are totally immune, the syndrome-matching approach does make it possible to contemplate steps to correctly isolate such failures. One such step is to look for partial pattern matches and, on the basis of the closeness of the match and the criticality of the potential failure condition, infer what the actual condition is. Methods for accommodating unanticipated conditions are currently being investigated.

Conclusions

A methodology known as the syndrome pattern-matching technique has been developed that can improve the performance of FDIR in complex fault-tolerant control systems such as modular avionics. This approach offers several advantages over traditional methods for the design and implementation of FDIR. These advantages include the following:

- The embedded software FDIR designed by this method can fully realize the fault isolation potential inherent in the system design.
- The embedded software FDIR is efficient and will minimize the use of system computational resources for performing FDIR functions.
- The design process is systematic and directly related to the architecture of the hardware. The potential exists to automate the process in a CASE tool.
- The logic structure of the embedded software is less complex and more deterministic than traditional FDIR, which makes the software less costly to design, test, and maintain.
- The logic structure inherent in the approach allows for adding sophisticated new techniques for dealing with intermittent and multiple failures and unanticipated failure scenarios.

Acknowledgement

The authors would like to acknowledge the contributions of William Weinstein who initially advocated the benefits of alternative approaches to FDIR and the merits of table-based logic structures.

References

(1) A. Aliphas, A. Wei, and B. Musicus, "A 16-Processor Prototype for a Fault Tolerant Parallel Digital Signal Processor," C.S. Draper Technical Paper, November 14, 1990.

(2) Hill, F. and Petterson, G. *Introduction to Switching Theory and Logical Design*, 3rd edition, John Wiley & Sons.

(3) Spitzer, C., *Digital Avionic Systems*,"Prentice Hall.