AIAA 79-1949R

Distributed Fault-Tolerant Switch for Use in Modular Redundancy

William C. Marmon*
Lockheed Palo Alto Research Laboratory, Palo Alto, Calif.

Abstract

THIS paper addresses the problem of selecting data from redundant processors without introducing vulnerable hardware. An original hardware-software switchover concept ensures correct system operation subsequent to any single failure, including a failure within the switch itself. A distributed output selection switch and voting mechanism are controlled by distributed software. Both hardware and software are fully specified. The concept has been implemented and tested on a microprocessor-based flight-control testbed.

Contents

A fundamental difficulty encountered in the synthesis of a fault-tolerant system is that of selecting a unique redundant module. Use of a distinct selection circuit, exemplified by a majority voter or externally controlled switch, creates a nonredundant apparatus from within which a single fault can induce terminal failure of the system. Reliability of the resulting configuration transcends that of a single module only if the switching mechanism, together with all components on which it must depend, exhibits a failure rate negligible compared to that of the total system.

The solution given here eliminates vulnerability to switch-mechanism failures in systems designed to survive only one failure. Since only one failure is to be survived, only two potential sources of the output signal are required. With only two sources, the problem is reduced to selecting which is to drive the output. A binary decision set permits an optimal vote based exclusively on a measure of the validity of the systems output; no communication among the processors is required. The decision rule for each processor can be expressed as the following: IF (AND ONLY IF) THE SYSTEM OUTPUT IS IN ERROR, CHANGE YOUR VOTE. Correct fault isolation is not necessary to guarantee correct system operation if this concept is employed.

Hardware for a fault-tolerant system protected from any single static failure is shown in Fig. 1. The system output is provided from either processor A or processor B. Only B and C provide the votes for the source of the system output, while the OR function to combine the votes is located in A. A is selected unless *both* B *and* C agree that an output error has occurred. Demonstration that a failed processor or control line cannot cause the system output to fail can be made by the following reasoning:

1) If B and C both agree that A has failed, then A has failed. Otherwise, both B and C would be in error, and the system is not designed to survive two simultaneous errors.

2) If B and C disagree on whether A has failed, then the single error has occurred in either B or C. Therefore, processor A is a correct choice by elimination, since the single error has already occurred in either B or C.

3) If B and C agree that A is correct, no error has been detected.

The above dialectics infer that each processor can identify the processor driving the output. In reality, this knowledge is not provided to the processors because the required communications links would be a source of complexity and potential failure.

Processors B and C each generate a vote based on the hypothesis that its one vote alone determines the source of the system output. Each processor decides whether its past vote resulted in a correct system output; if not, the vote is changed. This approach will result in the same operation as for the case where the processors have exact knowledge of the output source if and only if the voters are prevented from synchronizing on a periodic sequence. The required constraint is implemented by the software algorithm described in the next section.

Catastrophic failure of a processor or its physical removal from the system is included in the class of single static failures. The pull-up devices on the input to the OR gate in A ensure that A will be selected if either B or C has power removed or is physically removed. Similarly, the pull-down device in processor B on the output node of the OR gate will cause B to be selected if A has power removed or is physically absent. Transients on the output due to initial turn-on or abnormal voltages from a failed power supply are eliminated by biasing the output switch devices to fail in the safe open state if marginal voltages are present.

To summarize the features of Fig. 1:

1) The output will be driven to the active voltage through one (and only one) of the two series paths.

2) The output will be driven by either A or B. Only B and C provide votes, although the hardware to execute the decision is located in A.

3) Processor A does not vote, but is the preferred source of the system output signal. It is selected unless *both* B *and* C

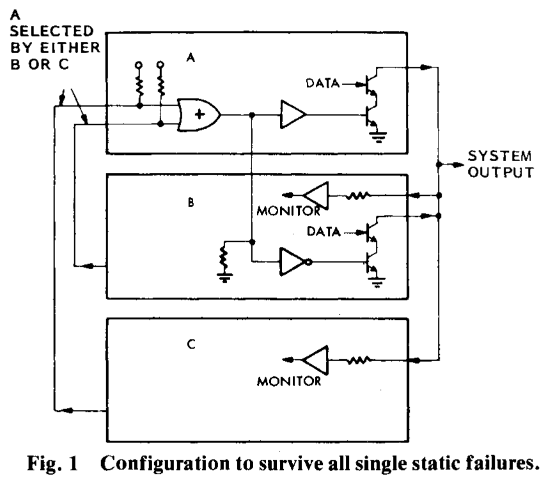

Fig. 1 Configuration to survive all single static failures.

Presented as Paper 79-1949 at the AIAA/IEEE/ACM/NASA Computers in Aerospace II Conference, Los Angeles, Calif., Oct. 22-24, 1979; submitted Dec. 10, 1979; Synoptic received June 16, 1980. Copyright © American Institute of Aeronautics and Astronautics, Inc., 1979. All rights reserved. Full paper available from AIAA Library, 555 W. 57th St., New York, N.Y. 10019; microfiche, $3.00; hard copy, $7.00. **Remittance must accompany order.**

*Research Specialist, Communication Sciences Laboratory, Dept. 52-51. Member AIAA.

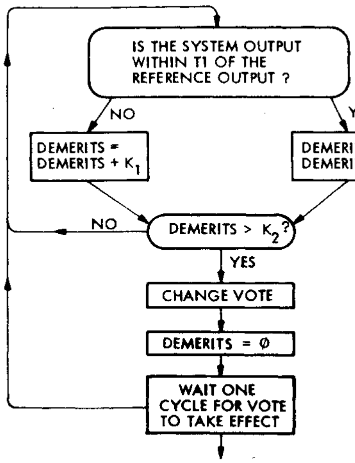
IS THE SYSTEM OUTPUT WITHIN T1 OF THE REFERENCE OUTPUT ?
NO
YES
DEMERITS = DEMERITS + K1
DEMERITS = DEMERITS - 1
NO
DEMERITS > K2?
YES
CHANGE VOTE
DEMERITS = 0
WAIT ONE CYCLE FOR VOTE TO TAKE EFFECT

Fig. 2 Software voting algorithm.

agree that votes for A have resulted in an output error. Processor B cannot unilaterally assume control without the concurrence of C.

4) Hardware devices ensure that A is selected if either B or C is unpowered or physically removed from the system. Similarly, B is selected if A is unpowered or removed.

5) A static failure in the selection circuit will force an arbitrary selection of either A or B. Either choice is correct, since the single failure has occurred in the selection mechanism. Components used in the selection circuit are separated from those affecting the processor data. This renders validity to the assumption of mutually exclusive failures in the data and selection mechanisms.

6) Votes are determined by correctness of the system output rather than by an effort to diagnose the precise source of a system failure.

An illustration of the value of the final feature above occurs if B, because of a switch failure, continues to drive the system output even though A is also driving the output. The output is the superposition of two nonsynchronous sources. Processor A has not failed, but the decision to select A has resulted in an incorrect output. Processor B is selected immediately without an attempt to verify the "failure" of processor A.

Required Software

Software for each processor must receive information on the status of the system output, compare this with its best estimate of the "ideal output," and change its vote if the output is not acceptable. This process is shown in Fig. 2 along with two features to prevent unnecessary switching.

Threshold T_1 is a design parameter to prevent voting changes based on negligible noise or normal phase differences between nonsynchronous processors.

Minor transients should be ignored, since unnecessary switching introduces its own transient. However, frequent transients indicate a long-term problem requiring a reconfiguration. Constants K_1 and K_2 in Fig. 2 constitute a filter coefficient set to implement these features based on a "demerit system": If the system output is found to be in error for a given cycle, K_1 demerits are assigned to the source of the current system output. If the system output is correct, one demerit is removed from the current output source to allow that source to redeem itself by a period of correct behavior. Finally, if the demerit count exceeds K_2, the current source of the output is deemed to have failed. In this case, a vote to change the configuration is made, the demerit count against the potentially new system configuration is reset, and one cycle is allowed for the reconfiguration to be completed before retesting the output.

An oscillation problem arises if voting processors are synchronized or loosely synchronized and use identical voting algorithms. An erroneous selection can remain uncorrected if the two voting processors fall into a mode cancelling each other's vote for a change. For example, assume that A in Fig. 1 has failed. *Both* B and C must *simultaneously* agree to inhibit the processor A output. If B and C are of opposite phase in their voting commands, at any given time one of them can vote for A while the other votes against it. This will result in A remaining permanently in control, even though both B and C detect a consistently erroneous output.

This problem is solved by selection of K_1 and K_2 in Fig. 2. If T_1 is assigned the same value by both voting units, and the cycle times of the two voting units are equal, the following is a sufficient condition to eliminate the problem:

$$|K_{2C}/K_{1C} - K_{2B}/K_{1B}| > 2$$

where K_{ij} is the value of K_i used by unit j. The worst-case uncertainty in the measurement of the output for the one cycle following a configuration change is considered by this condition. Unequal values of T_1 in the two voting units would not allow expression of a similar stability condition independent of the system output waveform.

Implementation of the above algorithm, with the specified stability condition satisfied, eliminates transient selection reversals. Following the appearance of an error on the system output, one single positive configuration change is made. No further switching action will occur.

Discussion of the Configuration

The configuration just described will survive *any* single static failure other than a fault on the final output node. It is directly expandable to as many output nodes as necessary. No restriction has been placed on the degree of synchronization among the processors. No communication link among the processors is required. The fault-tolerant organization has been developed without reference to the function of the processors, so it can be used in a wide range of applications.

This approach is applicable to a system with any number of two-state output waveforms. The only requirement is that all processors be able to determine correctness of the system outputs; criteria for correctness will vary with the application and type of waveform.

An implementation of this system is a readily understood fault-tolerant mechanization with no known eccentricities or critical parameters. It has been constructed, tested, and demonstrated as a fault-tolerant flight control system.