Fault-Tolerance of the Iterative Cell Array Switch for Hybrid Redundancy

ROY C. OGUS, STUDENT MEMBER, IEEE

Abstract—The technique of hybrid redundancy has been used to protect those portions of a digital system which have to be made ultrareliable. Siewiorek and McCluskey have presented a new switch design for hybrid redundancy which is shown to be of less complexity than other switch designs presented in the literature.

The possibility of increasing the overall system reliability is examined, considering schemes which will allow the switch to be tolerant of a certain number of faults in the hardware. The use of fail-safe logic together with a coding scheme has been found to be an effective way to increase the fault-tolerance of the switch. The fault-tolerance could also be achieved through the use of classical protection schemes such as triple modular redundancy (TMR). The two fault-tolerant strategies are compared using reliability, cost, and speed parameters as basis for comparison. A reliability analysis of the various system configurations has indicated that it is advantageous to increase the fault-tolerance, as a significant improvement of the overall system reliability is obtained. However, an optimum fault-tolerance exists, beyond which degradation of the system results. The question of whether adding spares to a system will always improve the system reliability is studied and it is found that this is not always the case. Finally, the implementation of error-detection circuitry using self-checking checkers is discussed, together with the possibility of the construction of a fault-tolerant voter.

Index Terms—Computer reliability, error-correcting codes, fail-safe logic, fault-tolerance, hybrid redundancy, iterative cell array, mission time, self-checking checker, threshold voter, triple modular redundancy (TMR).

INTRODUCTION

The use of N-modular redundancy (NMR) together with standby sparing has resulted in a very promising technique for protecting those portions of a digital system whose continuous real-time operation is essential. This technique is known as hybrid redundancy [1], [2] and consists of using N identical copies of the original module (where N is odd), forming an NMR core, connected to a voting element which outputs a value corresponding to the majority of the input values. Also provided are a number of standby spare modules which can replace any of the modules in the core in the event of one of the latter failing. The hybrid redundancy scheme is shown in Fig. 1. Whenever a module malfunctions it is replaced by an identical spare unit, the switch being the element which carries out the replacement.

In Fig. 1 the NMR core is composed of N (N being odd) modules and there are S standby spares. The modules are connected to the switch and a disagreement detector monitors the outputs of the modules and compares them to the voter output. If the disagreement detector discovers a discrepancy then the switch will remove the disagreeing module from the NMR core and replace it by a spare.

It is shown in [1] that this scheme can be made ultrareliable if the voter, switch, and disagreement detector (VSD) are very reliable. Thus the implementation of a simple switching scheme which can be made very reliable at a reasonable cost is essential. A new switch design has been presented by Siewiorek and McCluskey [3] that is shown to be of considerably less complexity than other switch designs presented elsewhere in the literature. The detailed description and design of the iterative cell switch are given in [3] but the scheme will be briefly described here for completeness. Thereafter, the possibility of increasing the reliability of the VSD will be examined, considering schemes which will allow the VSD to be able to function correctly even in the presence of failures in its hardware.

We shall make a distinction between the terms reliability and fault-tolerance. The reliability of a network (implicitly assumed to be measured at a time T) is the probability that the network has not failed up to time T. The fault-tolerance of a network is the number of faults which can simultaneously occur in the network without causing it to function incorrectly. Thus, the term fault-tolerance is a measure of the ability of a network to perform correctly even if faults exist in its hardware. A network which has a fault-tolerance of zero will be called a simplex or irredundant network.

In this paper we shall consider two schemes to increase the fault-tolerance of the simplex VSD proposed in [3] and determine the effect on the reliability of the overall system. It will be shown that the concept of fail-safe logic (together with a coding scheme) can be used to increase the fault-tolerance of the VSD and will improve the system reliability significantly. A second scheme uses the classical technique of NMR and comparisons will be made between the two fault-tolerant VSD's and the simplex VSD based on reliability, cost, and speed parameters.

Finally, the implementation of fault-detection circuitry will be considered so that faults in the VSD may be detected and repaired before the fault-tolerance of the VSD is exceeded. Self-checking techniques can be ef-
effectively applied for this purpose. A final section discusses the possibility of implementing a fault-tolerant voter.

**ITERATIVE CELL ARRAY SWITCH [3]**

An example of the iterative cell switching scheme is shown in Fig. 2. The system consists of a triple modular redundant (TMR) core with two spares. The disagreement detector determines which modules have disagreed with the voter output and the condition flip-flops record this information. Using the condition flip-flops (C-FF) outputs an iterative cell array determines the first three nonfailed modules and assigns them, via the interconnection logic, to the three voter inputs. The interconnection logic accomplishes the actual switching, using the control lines from the iterative cell array. The switch portion of the VSD thus consists of two components, viz., the control portion (iterative cell array and condition register) and the interconnection logic portion.

The **NMR core** is defined to be the set of **N** modules which are currently voting. In general, the number of modules in the NMR core will be \( N = 2t_e + 1 \) and the original number of spares will be \( S \). The parameter \( t_e \) is called the fault-tolerance of the core, i.e., the maximum number of modules which can simultaneously fail without resulting in an incorrect system output. (In Fig. 2, \( t_e = 1 \) and \( S = 2 \).)

The voter can be constructed in two different ways. A **threshold** element could be used and the resultant voter is called a **threshold voter**. In this case, the voter has \( 2t_e + 1 + S \) inputs and has a threshold value of \( t_e + 1 \). Since only \( 2t_e + 1 \) modules are in the NMR core at any one time, this configuration realizes the majority function for the modules selected to be in the NMR core. Fig. 2 shows the iterative cell switch with a threshold voter.

A second type of voter is a **majority voter**. This voter will realize the majority function and will vote on the \( 2t_e + 1 \) input lines. The voter therefore realizes the \( t_e + 1 \)-or-more-out-of-\( 2t_e + 1 \) function. The iterative cell array will assign the first \( 2t_e + 1 \) nonfailed modules to the voter inputs.

It is shown in [3] that the iterative cell logic for a threshold voting scheme can be made simpler because it suffices for a cell to inform a module that it is to be voted on or not. It will also be seen that the threshold voting scheme is the scheme which is suitable for the fail-safe logic protection scheme.

The above points thus favor the use of threshold voting elements in the hybrid redundancy scheme. However, the and/or realization of threshold functions with a large number of inputs is very costly with the result that, at present, the threshold voting mechanism is not practical for large values of \( t_e \) and \( S \). However, if integrated circuit versions of threshold gates become available [4], [17], large fan-in threshold functions could become economically feasible and the threshold voter method would become superior to the majority voter method.
FAULT-TOLERANCE OF THE VSD

For the overall system to be ultrareliable, it is essential that the VSD itself be very reliable. If the VSD can be realized in a fault-tolerant manner by an efficient low-cost design, then it is possible to increase the reliability of the overall system.

The concept of fail-safe logic [5] can be used to increase the reliability of the switch with an efficient usage of the extra hard-ware required. The fail-safe logic concept requires that one failure state can be considered tolerable while the other nontolerable. An example used in [5] considers a traffic light with two states, RED and GREEN. If the traffic light should fail, it would be preferable for it to fail in the RED state, otherwise a dangerous situation may arise. Thus, the fail-safe value is RED and whenever a GREEN appears, it may be assumed to be correct, while a RED may or may not be correct.

Consider now the hybrid redundancy scheme in Fig. 1. We wish that any switch failure should manifest itself as an “open” condition, i.e., resulting in a module being permanently switched out of the core. If the reverse were to occur and a module were to be permanently tied into the NMR core, then any subsequent failure of the module would result in a faulty module being permanently tied into the core. This will mean that the particular voter position will always have a faulty module voting which uses up some of the protection of the NMR core. In the former case, the switch failure would look like a module failure and a spare would be switched in its place.

By a nontolerable fault we will mean one which uses up some of the protection of the NMR core. A tolerable fault will be one which looks like a module failure.

We shall first describe how the interconnection logic can be made fault-tolerant through the use of the fail-safe logic concept. Thereafter, we will examine the other portions of the switch and show that the fail-safe logic idea can be effectively applied to a substantial portion of the rest of the switch and disagreement detector.

IMPLEMENTATION OF THE FAIL-SAFE INTERCONNECTION LOGIC

It is shown in [12] that the interconnection logic of the voting scheme using a threshold voter can be made fault-tolerant using fail-safe logic. In the majority voter i-
terconnection logic there is a bank of or-gates [3], [12] which have neither a s-a-0 or s-a-1 as tolerable faults. This voting scheme is thus unsuitable for a fault-tolerant implementation using fail-safe logic.

We now show the fail-safe implementation of the interconnection logic with a threshold voter. We first define the switch fault-tolerance, $t_s$, as the maximum number, $t_s$, such that any combination of $t_s$ simultaneous faults in the VSD will not cause the VSD to fail.

Fig. 3 depicts the construction of the interconnection logic in a fail-safe manner. Fig. 3(a) shows the simplex version of the inter-connection logic, with the threshold voter shown as a two-level AND-OR implementation. Fig. 3(b) depicts a fail-safe interconnection logic implementation for the case $t_s = 1$. This would be suitable for use with a TMR core. The control AND gates are duplicated and two copies of the control signal from the iterative cell are required. (The derivation of the duplicate control signals will be discussed in a later section.) If any single stuck-at-1 fault occurs, then the second gate will prevent the nontolerable “1” signal from propagating to and beyond the voter. The number of first level AND gate inputs in the threshold voter will thus be doubled for the case $t_s = 1$. Fig. 3(c) depicts the general case for switch fault tolerance $t_s > 1$. It is shown in [12] that up to $t_s$ stuck-at-1 faults on the inputs of the AND gates in the interconnection logic for each module will be corrected by the $t_s + 1$ gates; stuck-at-1 faults on the outputs of
the AND gates in the interconnection logic or on the inputs of the first level AND gates of the voter will be corrected by the redundant inputs to the voter. It can be seen that \( t_e \) extra inputs to the voter are required for each line in the simplex version of the interconnection logic.

At this point, it is useful to examine the actual meaning of "fault-tolerance." Using the definition of fault-tolerance given above, a system that is \( t \)-fault-tolerant is guaranteed not to fail in the presence of any combination of up to \( t \) faults within the system. This does not mean, however, that more than \( t \) faults will necessarily cause the system to fail. As an example, consider the fail-safe implementation discussed above with \( t_e = 1 \). A single stuck-at-1 fault in the interconnection logic circuitry of any voter position will be tolerated by the system and the fault will be corrected by the fail-safe logic. However, a single stuck-at-1 fault occurring in each voter position will also be tolerated; i.e., \( N + S = 2t_e + 1 + S \) faults are tolerated while \( t_e \) is only equal to 1. But, not all combinations of \( N + S \) stuck-at-1 faults will be tolerated since two or more failures in the circuitry of a single voter position would lead to system failure. Thus, the fault-tolerance, as defined earlier, is actually a lower bound on the number of faults that the system can tolerate and many cases exist where the system may actually tolerate far more.

**FAULT-TOLERANT ITERATIVE CELL ARRAY**

Thus far, we have considered the possibility of increasing the fault-tolerance of the VSD by the use of a fail-safe logic scheme for the interconnection logic. In this section we consider a method to increase the fault-tolerance of the iterative cell array. The scheme employs error-correcting codes and it will be found that the fault-tolerant iterative cell array interfaces very easily and efficiently with the rest of the VSD.

**ITERATIVE CELL ARRAYS**

An iterative cell array [6] is a cascade of identical combinational circuits, called **cells**. The \( i \)th cell (Fig. 4) receives inputs from the outside world, called **primary inputs** \( z_{j}^{i} \), as well as inputs from the cell immediately to its left, called **carry inputs** \( y_{j}^{i} \). The cell computes output functions called the **primary outputs** \( z_{j}^{i+1} \), which it transmits to the external world, and **carry outputs** \( y_{j}^{i+1} \), which are sent to the next cell to the right. The terminal behavior of the cell may be described by means of a **cell table**, which is analogous to the state table of sequential circuits. The set of values on the carry leads may be considered as **carry states**. The cell thus computes the carry output state (or next state) using the carry input state (or current state) and the primary input values. The same applies to the primary output values. An iterative cell array can thus be regarded as the spatial analog of a sequential circuit [6].

In the iterative cell array switch of Fig. 2 the primary inputs of the iterative cells emanate from the C-FF's, and the primary outputs are directed to the interconnection logic. The cell table for each cell is shown in Table I. Each cell receives as primary input the condition of the \( i \)th module (input \( c_{j}^{i} \)), i.e., failed or functional, and the current state denotes the number of modules which have been found to be functional prior to the consideration of module \( i \). The output function indicates whether the particular module is to be switched on-line or not. For example, if the current state indicates two or fewer good modules found so far and if \( c_{j}^{i} \) indicates a functional module, then the output \( z_{j}^{i} \) will be "1" and module \( i \) will be switched into the voter. The iterative cell array is described in more detail in [3].

**FAULT-TOLERANT SCHEME FOR THE ITERATIVE CELL ARRAY**

The fault-tolerant scheme envisaged for the iterative cell array makes use of error-correcting codes and is based on a method suggested by Mealy [7] and Russo [8] for sequential circuits.

A sequential circuit can be viewed as consisting of two portions, viz., a set of memory elements or flip-flops (FF's) and a combinational circuit portion. The combination of the states of the FF's is called the state of the sequential circuit. The combinational circuitry, using the inputs and present state, computes the excitation functions which cause the FF's to switch to the appropriate next state. The output functions are also realized by the combinational circuitry.

If the state of the FF's is encoded in a \( t \)-error-correcting code then the minimum Hamming distance between any two states will be at least \( 2t + 1 \). Thus, if at most \( t \) FF errors occur then it is possible to determine what the correct present state was, given the error state. The combinational circuitry can be designed to resolve the error state and implicitly correct it by realizing the excitation function and primary outputs corresponding to the correct state of the FF's.

If one considers the iterative cell as the spatial analog of a sequential circuit then the correspondence is as follows.
The outputs of the FFs correspond to the carry inputs of the iterative cell and the excitation function will now correspond to the carry outputs of the cell. The combinational circuitry comprises the actual cell logic. Since we now have an array of identical cells we can extend the definition of fault-tolerance to include the tolerance of faults in the combinational logic as well as errors on the inputs of the cell (corresponding to FF errors in the sequential circuit).

To apply the above scheme to the iterative cell we proceed as follows. Each set of inputs and outputs of each cell is encoded with an error-correcting code. If the cell is to be $t_c$-fault-tolerant, then a $t_c$-error-correcting code should be used. The subsequent description is for the case where $t_c = 1$ and single error-correcting (SEC) Hamming codes were used, but the extension to cases where $t_c > 1$ is straightforward.

We shall use upper case letters to denote the SEC encoded state variables. This is a slight deviation from normal iterative cell nomenclature but should not cause any confusion. Carry input state variables (present state) for the $i$th cell will carry the superscript $i$, while carry output state variables (next state) will carry the superscript $i + 1$. We shall also use the terminology “carry input state” and “present state” interchangeably. The same applies to “carry output state” and “next state.”

The scheme is illustrated using the cell table in Table I as example. The cell table is now modified as follows. The states and inputs are first encoded using an SEC code and the cell table is rewritten using the new SEC encoding. The new table, called the intermediate modified cell table, is shown in Table II. It should be noted that the minimum distance between code words is 3. (The minimum distance is $2t_c + 1$ for a $t_c$-error-correcting code.)

The next step is to replace each row of the intermediate table by a set of rows comprising the original row and rows corresponding to all the error states distance one away from the original valid state. Similarly, each column of the table is replaced by a set of columns representing the original column and columns for each error input distance one from the original valid input. The table entries in error rows and columns are identical to the entries in the corresponding valid (error-free) rows and columns. The final modified cell table is shown in Table III.

It is thus seen that any single error present in the carry inputs and/or the primary inputs is implicitly corrected by the modified next state and output functions. No explicit encoding or decoding operations need be done. The error-correcting mechanism is implicit in the design of the cell and no delays through extra logic levels are (theoretically) encountered. In the practical design of the cells, two-level circuits were not feasible but the coded iterative cell required fewer logic levels than comparable massive redundancy schemes such as TMR where voting has to be performed at various points.

The logic of the coded iterative cell can now be designed using the final modified cell table (Table III). The cell is purely combinational logic but the number of Boolean variables is now somewhat large so that conventional minimization methods [9] will prove cumbersome and unwieldy. However, we have developed a combinatorial algorithm to minimize the cell logic, which is an extension of an algorithm suggested by Russo [8] for counters. The algorithm makes use of the concepts of distance and intersection and is much simpler than the conventional Boolean minimization algorithms. It uses the intermediate modified cell table (Table II) rather than the final modified cell table (Table III) and thus only deals directly with the valid states. The properties of this algorithm are to be described in a subsequent report.

The modified cell is thus tolerant of single errors at the cell inputs. Any single error in the carry inputs or the primary inputs, or both, will be implicitly corrected. If the cell output functions are produced independently, then any single cell fault occurring (even if in addition to the previously mentioned input errors) can cause at most one output error which will be corrected by the next cell.

**FAULT-TOLERANT VSD CONFIGURATION**

In order to incorporate the coded iterative cell array into the switch design, the general form of the fault-tolerant iterative cells described in the previous section
need not be used. Using some of the properties of the switch, we may save some hardware by simplifying the cell design.

The iterative cell array interfaces to the condition register on the input and to the interconnection logic at the output. As we saw earlier, the interconnection logic was implemented using a fail-safe scheme which required duplicate control signals from the iterative cell array. Thus, the cell output \( z^i \) which, when encoded in an SEC code requires triplication, need only be duplicated. This results in some logic circuitry being saved in the coded cell.

Consider now the condition register, the outputs of which are fed to the primary inputs of the iterative cells. The SEC encoding of the primary input, \( c^i \), of the cell requires triplication of the condition register. However, we may observe that the condition register also meets the fail-safe requirement, viz., one failure state is tolerable whereas the other is nontolerable or dangerous. For, the failure of a C-FF to a stuck-at-1 (indicating that the particular module is functional) is a nontolerable situation, since a subsequent failure of that module will not be able to be registered in the C-FF. However, the failure of a C-FF to a stuck-at-0 is tolerable since it looks like a module failure and the “failed” module will be switched out. Thus, only a fail-safe encoding, i.e., duplication, of the condition register is necessary, resulting in a further savings in hardware. Similarly, the disagreement detectors also meet the fail-safe requirement and need only be duplicated.

In the simplex VSD design (Fig. 2) each disagreement detector compared the system output from the voter with the particular module output to detect any possible discrepancy. However, with the fail-safe implementation of the disagreement detector, the system output should be compared with each of the dual outputs of the control gates in the fail-safe interconnection logic [Fig. 3 (b)].

As mentioned earlier, stuck-at-0 faults in the fail-safe interconnection logic will not be detected by the disagreement detector as module failures if the module output itself is compared to the voter output. However, if the control gates outputs are used, this introduces the problem that spares which are not assigned to the NMR core may erroneously be recorded as failed. (This is identical to the problem which arises when power switching of modules is used instead of logical switching, as discussed in [3], and is solved in the same way.) This can be remedied by the addition of an extra AND gate at the C-FF outputs which is controlled by the iterative cell outputs. When a module is not assigned to the TMR core, the disagreement detector is inhibited and cannot influence the state of the C-FF. Note that a spare not assigned to the core may fail but its C-FF’s would not record this fact until the module is switched into the TMR core.

Using the fail-safe and coding schemes described above, the fault-tolerant version of the VSD of Fig. 2 can be
implemented and it is shown in Fig. 5. The switch-
disagreement detector portion of the VSD can be thought
of as consisting of a number of identical1 bit slices. The bit
slice is vertical and the number of bit slices, \( B \), corre-
sponds to the total number of modules, i.e., \( B = 2t + 1 + S \). The details of a bit slice are thus sufficient to
characterize the VSD design. The details of a bit slice
of the fail-safe/coded (FS/C) VSD are shown in Fig. 6(a).

The fault-tolerance of the VSD can also be increased
through the use of the classical massive redundancy
 technique of NMR. The bit slice can be divided into cells
and each cell is replicated and voters are inserted between
the replicated cells. The details of a bit slice of the VSD
protected by TMR (for the case \( t_s = 1 \)) are shown in
Fig. 6(b).

In the next section we shall analyze the hardware costs
involved and compare the FS/C VSD to the equivalent

1 Due to the occurrence of “edge effects” [3] the iterative cells
on either end of the array will not be as complex as cells which
are found in the middle of the array, but can be simplified due to
boundary conditions. However, in the subsequent analysis this
fact is ignored and all the bit slices of the VSD are considered to be
identical.

Fault-tolerant Implementation using the Massive Redundancy
Approach of TMR. Therefore, the reliability
analysis is described and the FS/C, TMR, and simplex
VSD’s are compared from a reliability viewpoint.

COST ANALYSIS

The cost model used as a basis for comparison of the
various schemes is called the unit gate model [10]. Accord-
ing to this model, the number of equivalent unit gates
(UG’s) of an integrated circuit package is defined as
follows:

\[
\text{Number of equivalent UG} = \frac{(\text{total number of gate inputs})}{2}
\cdot (\text{complexity factor}) \cdot (\text{technology factor}) \cdot \left(\frac{\text{number of packaged leads}}{14}\right)^2.
\]

For circuits consisting of NAND, NOR, AND, OR, and
INVERTER gates, i.e., small-scale integration (SSI), the
OGUS: ITERATIVE CELL ARRAY SWITCH

The cost of one bit slice of each scheme was compared, using the implementations shown in Fig. 6. It was found that the FS/C bit slice of Fig. 6(a) required 157.5 UG's while the TMR bit slice of Fig. 6(b) needed 151.5 UG's. Thus the hardware cost of the two schemes is roughly the same.

Also apparent in comparing Fig. 6(a) and (b) is the fact that far fewer interconnections are required in the FS/C bit slice compared to the TMR version which results in a simpler design.

PROPAGATION DELAY

When hybrid redundancy is used to protect the original simplex module the overall system speed will be slowed down by the propagation delay in the VSD. To compare

complexity factor is chosen as 1. The technology factor for TTL circuits is also chosen as 1. If we assume SSI TTL packages having fourteen terminals then the number of UG's reduces to:

\[
\text{number of equivalent UG} = \frac{\text{total number of inputs}}{2}
\]

Thus a simple \( n \)-input gate will be considered as equivalent to \( n/2 \) UG's. The cost of an FF was determined using data from a typical Fairchild JK FF (series 9022) [11] and it was determined that the JK FF was equivalent to fifteen UG's.

The cost of the FS/C VSD implementation was compared to the cost of the equivalent TMR version of the VSD which afforded the same fault-tolerance (\( t_s = 1 \)).

Fig. 6. Comparison of two fault-tolerant schemes for a single bit slice of the VSD. (a) FS/C approach. (b) TMR approach.

\[
\text{From system voter}
\]

\[
\text{Interconnection Logic}
\]

\[
\text{From module i}
\]

\[
\text{Interconnection Logic}
\]

\[
\text{Condition Flip-flops}
\]

\[
\text{Voter}
\]

\[
\text{Disagreement Detect...}
\]

\[
\text{Cell}
\]

\[
\text{Condition Flip-flops}
\]

\[
\text{Voter}
\]

\[
\text{Disagreement Detect...}
\]

\[
\text{Cell}
\]

\[
\text{From system voter}
\]

\[
\text{Interconnection Logic}
\]

\[
\text{From module i}
\]

\[
\text{Interconnection Logic}
\]

\[
\text{Condition Flip-flops}
\]

\[
\text{Voter}
\]

\[
\text{Disagreement Detect...}
\]

\[
\text{Cell}
\]

\[
\text{Condition Flip-flops}
\]

\[
\text{Voter}
\]

\[
\text{Disagreement Detect...}
\]

\[
\text{Cell}
\]
the speed of operation of the fault-tolerant VSD's with the simplex VSD, the propagation delay through a bit slice of each design was determined. The delays in the horizontal and vertical directions were compared.

The propagation delay was calculated in terms of \( G \), the delay through a logic gate, and \( F \), the delay through an FF. Using Fairchild data sheets for a typical FF (series 9022), it was determined that, typically, \( F = 2G \).

The comparison of the delays is given in Table IV where it is assumed that \( F = 2G \) and the delays are normalized with respect to the simplex scheme. It can be seen that the FS/C scheme is better than the simplex scheme in the vertical direction while only slightly slower in the horizontal direction. The TMR bit slice, however, is nearly twice as slow as the simplex scheme and is significantly slower than the FS/C scheme. As described in [3], the speed of propagation through the iterative cell array, i.e., in the horizontal direction, is important, and in [3] fast adder techniques have been borrowed to improve it.

### RELIABILITY ANALYSIS

Up until now we have considered techniques to increase the fault-tolerance of the VSD. At this point we should determine whether it is indeed advantageous to increase the fault-tolerance of the VSD. The parameter that we should consider is the reliability of the overall system.

The overall system reliability is an important parameter since it is possible to increase the fault-tolerance of a system and yet degrade its reliability due to the addition of a great amount of extra hardware.

A reliability analysis of the hybrid redundancy scheme with an iterative cell array switch has been carried out. The reliability of the scheme using simplex (fault-tolerance = 0), FS/C and TMR VSD's has been determined. The complete details are found elsewhere [12] but the pertinent results will be summarized here.

We shall first compare the systems having fault-tolerant VSD's to the system with a simplex VSD. Thereafter, the two fault-tolerant VSD's will be compared to each other. In addition, the hybrid redundancy system configurations may be compared to the original nonredundant module or to a simple TMR configuration with a single voter. Thus, the following possible system configurations may be examined: 1) the nonredundant system, 2) a simple TMR system, and 3) hybrid redundancy systems with a) a simplex VSD (H simplex), b) an FS/C VSD (H FS/C), and c) a TMR VSD (H TMR).

The reliability models used were based on the exponential failure rate of an UG and thus the overall system reliability may be examined as a function of time. A second reliability measure used was the maximum mission time for a minimum system reliability which is defined as the time that the overall system reliability takes to drop to a specified terminal value. The system reliability is assumed to be 1.0 at the start of the mission, and the terminal value was taken as 0.95.

The analysis was carried out in [12] using the UG cost model and the core fault-tolerance \( t_c \), the switch fault-tolerance \( t_s \), the number of UG's per module \( M \), the number of spares \( S \), and the reliability of a unit gate \( r_g = e^{-t} \) as parameters. The value of \( \lambda \), the failure rate of a unit gate, was taken as 0.005 percent/1000 h which is a typical value used in the industry [10].

We will briefly indicate the analysis for the hybrid redundant system with a simplex VSD. Consider the case where \( t_c = t_s = 1 \). This corresponds to a TMR core with a 1-fault-tolerant VSD.

Let the number of UG's in a bit slice of the VSD be \( M_B \) and define the bit slice cost factor, \( c \), as \( M_B/M \). Thus, the reliability of a bit slice is \( R_S = \left( r_g \right)^{M}$ and the reliability of a module \( R_M = \left( r_g \right)^{M}_B \). Therefore,

\[
R_S = \left( r_g \right)^{M} = \left( r_g \right)^{M}_B = \left( r_M \right)^{c}. \tag{1}
\]

The reliability of a hybrid redundant system with a TMR core and \( S \) spares, assuming a perfect VSD, is \[1], [12]

\[
R(3, S) = 1 - (1 - R_M)^{S+2} \left[ 1 + R_M(S + 2) \right]. \tag{2}
\]

As described earlier the switch-disagreement detector portion of the VSD consists of \( B = 2t_c + 1 + S = 3 + S \) identical bit slices. Thus, ignoring the reliability of the voter, (which can be lumped as a multiplicative constant) the reliability of the hybrid redundant system is

\[
R^*(3, S) = \left( R_B \right)^{S+2} \left[ 1 - (1 - R_M)^{S+2} \left[ 1 + R_M(S + 2) \right] \right] \tag{3}
\]

Now

\[
R_M = \exp \left( -\lambda T \right) \tag{4}
\]

where \( T \) is time. If the time function for \( R_M \) is substituted in (3) as well as the appropriate values for \( c \) and \( S \) we have the reliability as a function of time. This curve, H simplex, is shown in Fig. 7.

Using the previous definition of maximum mission time, \( T_M \), with a terminal reliability of 0.95, we get

\[
R^*(3, S) = 0.95 \text{ at } T = T_M. \tag{5}
\]

Then, using (3)–(5) we obtain

<table>
<thead>
<tr>
<th>Direction</th>
<th>H.Simplex</th>
<th>H.SF/C</th>
<th>H.TMR</th>
<th>Ratio of H.TMR:H.SF/C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vertical</td>
<td>1</td>
<td>0.89</td>
<td>1</td>
<td>2.13</td>
</tr>
<tr>
<td>Horizontal</td>
<td>1</td>
<td>1.13</td>
<td>1.75</td>
<td>1.56</td>
</tr>
</tbody>
</table>
Fig. 7. System reliability versus time for the various system configurations.
\[ R^*(3,S) = \left[ \exp(-\lambda T_M) \right]^{(3+S)} \cdot \left[ 1 - \exp(-\lambda T_M) \right]^{S+2} \]
\[ \cdot \left[ 1 + \exp(-\lambda T_M)(S + 2) \right] \]
\[ = 0.95. \]  

Solving (6) numerically we can obtain \( T_M \) as a function of the number of spares, \( S \), and this curve is plotted in Fig. 8 where the maximum mission time is plotted as a function of the number of spares.

![Graph showing maximum mission time versus number of spares](image)

Fig. 8. Maximum mission time versus number of spares.

The corresponding analysis for the systems with fault-tolerant VSD's as well as a more detailed analysis of the simplex system is found in [12]. Some of the pertinent results are presented here.

Fig. 7 shows the overall system reliability plotted as a function of time. As expected, the simple TMR system shows an improvement in reliability over the original nonredundant module. The hybrid redundancy scheme with a simplex VSD is also more reliable than the non-redundant module but is initially less reliable than the simple TMR configuration. The important result to be seen in Fig. 7 is the fact that the systems with fault-tolerant VSD's show a significant improvement in reliability, and that the extra hardware introduced to achieve the fault-tolerance has upgraded the system rather than degraded it. Thus, the earlier question as to the benefits of increasing the fault-tolerance of the VSD has been answered in the affirmative. The reliabilities of the two fault-tolerant schemes are very close in value, with the H.FS/C configuration being initially more reliable than the H.TMR system, but eventually dropping off at a slightly faster rate.

The improvement in reliability can also be seen in terms of the maximum mission time (Fig. 8). Here the maximum mission time is plotted as a function of the number of spares. The values for the TMR and non-redundant systems are indicated for reference although they are obviously independent of \( S \). As expected, the
hybrid redundancy schemes have higher maximum mission times than the TMR or nonredundant systems and the systems with fault-tolerant VSD's show a significant improvement over the system with a simplex VSD.

Another interesting phenomenon is apparent from examination of Fig. 8. Previous reliability analyses of hybrid redundancy systems [1], [2], [13] have either ignored the effect of the VSD reliability, or have lumped it into a product term which modifies the reliability equations of systems with perfect VSD's. The result of this practice is that the reliability of hybrid redundancy systems is a monotonically increasing function of S. Fig. 8 indicates that this is certainly not the case and that the unreliability of the VSD can have a profound effect on the overall system reliability. In actual fact, after an optimum number of spares, the overall system reliability begins to decrease as more spares are added. In fact, Fig. 8 shows that the system with a simplex VSD reaches a maximum reliability with only two spares, and that the addition of more spares will degrade the system. The systems with fault-tolerant VSD's also exhibit this phenomenon but the optimum number of spares is greater, resulting in a higher maximum system reliability. The reason for this is the fact that the VSD complexity increases with the number of spares and after a point, the extra hardware of the VSD begins to have a significant effect on the overall system reliability and results in its degradation.

The effect of varying the switch fault-tolerance $t_s$ was also examined and the results for the H.TMR system are shown in Fig. 9. As can be seen, increasing the fault-tolerance to $t_s = 2$ for the given parameters results in a decrease in the overall system reliability. This is due to the fact that again the VSD hardware complexity becomes significant compared to the module complexity and results in making the system more unreliable. This effect is shown [12] to decrease as the module size becomes much larger than a bit slice of the VSD.

The above two situations (Figs. 8 and 9) were for the case where $M = 1000$. This is perhaps a more typical situation where the hybrid redundancy protection scheme would be used. The suitability of the iterative cell array switching scheme was demonstrated in [3] for the case of the arithmetic and logic unit (ALU) of a popular minicomputer which required 1300 equivalent gates. The modules were complete ALU's and thus $M$ was of order $10^6$. The above results would indicate the number of spares and fault-tolerance of the VSD which should be used for such a configuration.

In summary, the reliability analysis of the various
hybrid redundancy systems has produced some significant results. It was found that increasing the fault-tolerance of the VSD does indeed result in a higher overall system reliability and provides justification for the extra hardware required. However, increasing the fault-tolerance beyond a point where the VSD complexity becomes significant compared to the module complexity will further degrade the system. Thus, depending on the module size, an optimum fault-tolerance exists. Finally, an optimum number of spares also exists and further spares will degrade the system reliability. For the simplex system this optimum value is fairly low and is about two for a typical situation.

IMPROVEMENTS

Two other improvements to the fault-tolerant VSD's are discussed in [12]. They will be briefly mentioned here.

A. Detection of Faults in the FS/C VSD

The use of fail-safe logic networks in the VSD will have the effect that some VSD failures will cause good modules to be regarded as failed and consequently switched out of the TMR core. To further increase the reliability of the H.FS/C system it would be necessary to be able to recover the good switched out modules. The mechanism of retrying [3] is especially suitable for the recovery of the good modules. Thus, detection of the faults in the VSD is necessary.

The fact that there is a very close relationship between self-checking circuits and fail-safe circuits is discussed in [12]. It is shown that using a self-checking checker [14] in each bit slice, and feeding the fail-safe signals to the checker, faults can be detected in the VSD. The status of each bit slice, “bit slice good” or “bit slice faulty,” can be displayed in an error register, which can be used to aid diagnostic procedures. Since the detection circuitry is itself self-checking, faults in this circuitry would also be indicated in the error register. In addition, the error register itself could be made fail-safe by duplication.

Thus, the close relationship that exists between self-checking circuitry and fail-safe circuits allows the simple interconnecting of a self-checking checker to each bit slice of the fail-safe VSD. Once the faults are detected and the circuitry replaced or repaired, any good modules could be recovered by retrying the failed modules.

B. Fault-Tolerant Voter

The set of circuits which has to function continuously to ensure the correct operation of an entire network is called the hard core of the network. By increasing the fault-tolerance of a network we are, in effect, reducing the amount of hard core. In the original scheme proposed in [3] for hybrid redundancy, the entire VSD constituted the hard core of the system. However, observing Fig. 5, we notice that the entire VSD is now fault-tolerant with the exception of the system threshold voter. The voter alone is the hard core of the system. It can be seen from Fig. 4(a) that the fail-safe logic concept cannot be applied to the threshold voter. Any stuck-at-1 fault on the output of a first level voter and gate will result in the or gate being stuck-at-1 on its output. This situation arises because both stuck-at-0 and stuck-at-1 faults are nontolerable faults in the voter.

A possible method of making the voter fault-tolerant is to apply NMR to the voter and interface the redundant voter outputs to the rest of the VSD. This can be done by modifying the fail-safe disagreement detectors so that each independently computes the majority of the voter outputs and uses this value for comparison with the module outputs.

If the system output is fed to another portion of the machine which can accept and use the redundant voter signals then we have successfully reduced the amount of hard core in our system to 0, and the system is completely 1-fault-tolerant. However, if a single system output is required then we are back to where we started since we would need another voter to perform this decision. Since the disagreement detectors become somewhat costly in the above scheme, it may be better to retain the original single voter as the hard core and concentrate on making it as internally reliable as possible.

CONCLUSIONS

The use of a hybrid redundancy scheme is a very promising technique to achieve ultra-reliable digital systems. The success of the scheme, however, depends on having a very reliable VSD. This implies that the design of these units must be as simple as possible since a complex design requiring a great amount of hardware will degrade the overall system reliability.

Siewiorek and McCluskey [3] have presented a new switch design which is less complex than other designs in the literature and we have examined methods to increase the reliability of the VSD using the iterative cell array switch. Using fail-safe implementations of portions of the VSD, viz., the interconnection logic, condition register and disagreement detectors, and a coded implementation of the iterative cell array, we have increased the fault-tolerance of the VSD. A second scheme achieved a similar fault-tolerance through the use of TMR.

In order to increase the fault-tolerance of a system we have to add additional hardware to the system. The question then arises whether we have not, in actual fact, degraded the system reliability as a result of the extra circuitry. However, we have shown that the fail-safe scheme, as well as the scheme using TMR, has improved the system reliability by a significant amount, thus justifying the extra hardware cost incurred. Also, the speed of operation has not been significantly degraded.

A second result indicated that further increasing the
fault-tolerance of the VSD beyond a point resulted in a decrease in the system reliability, the extra hardware finally having an effect. Thus, an optimum fault-tolerance exists for the VSD, given a set of system parameters.

Another interesting phenomenon discovered is the fact that an optimum number of spares also exists and that, contrary to previous analyses, increasing the number of spares beyond this point does not further improve the system reliability, but rather degrades it. In other words, the system reliability is not a monotonically increasing function of the number of spares as alluded to by other researchers [1].

Finally, noting the close relationship between fail-safe and self-checking circuits, we have indicated how a set of self-checking checkers can easily be interfaced to the VSD to achieve error detection.

The last piece of hard core in the system is the system voter and, if required, it can be made fault-tolerant by triplication and modification of the disagreement detectors.

ACKNOWLEDGMENT

The author would like to acknowledge the valuable advice and guidance of Prof. E. J. McCluskey received during the course of this work.

REFERENCES


Roy C. Ogus (S'71) was born in Pretoria, South Africa on December 3, 1946. He received the B.Sc.(electrical engineering) (cum laude) degree in electrical engineering from the University of Pretoria, Pretoria, in 1968 and the M.S. degree from Stanford University, Stanford, Calif., in 1971. He is currently working towards the Ph.D. degree in the Digital Systems Laboratory, Stanford University.

From 1969 to 1970 he held the position of Research Officer in the Automation Division of the Council for Scientific and Industrial Research, South Africa, where he was engaged in research and development work on real-time computer systems. Since 1972 he has been a Research Assistant in the Digital Systems Laboratory, Stanford University, working on fault-tolerant computing. His current interests include fault-tolerant computing, reliability evaluation and modeling, computer architecture, and real-time computer systems.

Mr. Ogus is a member of the IEEE Computer Society, the Association for Computing Machinery, the South African Institute of Electrical Engineers, and Sigma Xi.