A Mid-Value Select Voter

M. D. Krstic, M. K. Stojcev*, G. Lj. Djordjevic, I. D. Andrejic

Faculty of Electronic Engineering, University of Nis, Beogradska 14, 18000 Nis, Serbia & Montenegro

Abstract

Hardware redundancy may be used in a variety of manners to achieve fault tolerance. One of the most popular techniques is a triple modular redundancy (TMR) scheme. Such a scheme has also been referred to as masking redundancy because failures those affect only if one of the three modules is masked by the majority of the nonfailed modules. Most of the published works on TMR make one crucial assumption: In fault-free operation the outputs are equal. However it is well known that the output of redundant sensor elements in fault-tolerant data acquisition systems cannot be guarantied to match even in fault-free operation. They are usually handled by a median-select or similar selection rule, so that redundant voter can pick a common value for processing by the rest of the system. This paper presents a VLSI fault-tolerant voter, with redundancy designed into the internal chip architecture. Instead of three we propose installation of four sensor elements. In order to insure that the voted value represents a correct consensus, we propose a mid-value hardware voting technique thanks to which we solve the problem of dissemination of each sensor element value to other ones. Finally, the effect of fault-tolerance on voter performance is discussed.

1. Introduction

Fault tolerant (FT) computing has been defined as the ability to execute specified algorithms correctly regardless of hardware failures and program errors [1, 2]. Computer systems based on FT principle have become essential in real-time (data acquisition) applications such as nuclear power plants, life support systems, commercial aircraft, industrial control, etc. Common to all these applications is the demand for high reliability and/or availability, and fail safe or non-stop features. These requirements are necessarily stringent because a single failure of any constituent (CPU, memory or I/O module) in these applications can lead to disaster. For example, for certain applications in commercial aircraft, no more than a 10^{-9} chance of failure over a ten-hour period is permitted [3].

Fault tolerance is generally accomplished by using redundancy in hardware, software, time, information, or combination thereof [4]. There are three basic types of redundancy in hardware and software: static, dynamics, and hybrid [5]. Hardware redundancy technique is preferable in controlling continuous processes, where events happen quickly as we meet in hard and firm real-time fault-tolerance data acquisition systems.

Triple modular redundancy (TMR) as a most common form of massive redundancy is one of the most popular fault-tolerance schemes using spatial-redundancy for fault masking. This technique triplicates hardware and performs majority vote in order to determine the output of the system. Namely, if one module becomes faulty, the remaining fault-free modules mask the result of the faulty module

* Corresponding author. Tel.: +381 18 529 660.
E-mail: stojcev@elfak.ni.ac.yu (M. Stojcev)

when the majority (two-out-of-three) vote is performed [6].

Most TMR systems make the basic assumption: In fault-free operation all three inputs to the voter are equal. In fault tolerant data acquisition systems (FTDASs) the remote sensor element (SE) is replicated at each location of the monitored technological process with goal to allow tolerance of SE failure. However, it is well known that the outputs of redundant SEs cannot be guaranteed to match even in the absence of faults. Namely, the SE readings differs due to slight differences in their calibration, physical location (of SEs) in respect to referent one, induced noise, inserted errors during data transfer between remote SEs and the central processing unit of the FTDAS, or because the system reads the respective SEs at slightly different times or of course, due to failure of one of the SEs [7]. Several TMR design solutions of SEs' data handling are based on median-select or similar selection rules [4, 7, 8] so that redundant voters can pick a common input value for processing by the rest of the FTDAS. But, the voting based on median-select is not an appropriate design choice in a case of simultaneous presence of any two different classical and non-classical failures since there is no selection rule that can pick a common input values. (We assume that SEs' output value disagreement and data transfer errors are typical non-classical failures, while hardware failures of SEs or hardware failures of input-interface-modules (IMs) are classical failures). In environment requiring extremely high reliability, such as FTDAS, voting methods must be provided which can give tolerance against two arbitrary different failure types (classical and non-classical). In order to cope with this problem an adequate solution was proposed in [9]. Here the authors propose two step processing. During the first step, a single non-classical data transfer error per each channel (channel is a serial connection between SE and IM) can be detected using single parity bit. Channel with detected transmission error is substituted with the spare channel. The second step involves a voting on three selected channels based on mid-value select scheme. In this paper we adopt similar mid-value voting mechanism, but we involve error-correction and detection capability during the first processing step. Namely, instead of parity we use Hamming coding. This possibility allows us to detect and correct all single bit errors per channel, with minimal hardware overhead.

In this paper, we address a new data-path design of a voter circuit referred as a mid-value select voter (MVSV). Section 2 briefly describes the voting principles implemented in real-time FTDAS with redundant sensing elements. In Section 3 the mid-value select voter architecture is described. Experimental results are illustrated in Section 4. Finally, conclusions are presented in Section 5.

2. Voting schemes

In real-time FTDASs with redundant sensing elements, congruency (or agreement) can not be guaranteed even in the absence of faults. For example, sensing elements, which measure a temperature, may give slightly different outputs from otherwise correctly functioning units because of inherent SE's inaccuracies. Therefore, the voting procedure will not work correctly if it is implemented on a bit-by-bit masking principle [7].

Over the past twenty-five years, several hardware/software techniques for voting, when the non-faulty SEs outputs are unequal, have been developed. A large volume of work concerning this type of voting exists, covering different types of FTDASs and methods of implementation [4, 6, 5] We will focus now, by our opinion, to short analysis of six design solutions that are more typical. The first called arithmetic middle value (AMV), is based on selection of AMV of the three as the correct value [4]. In the second approach, referred as weighted average scheme (WAS), a corresponding weight is assigned to each SE's output value according to how close the SE's output was to the selected value in the previous cycle [8], The third, called the mid-value selection (MVS) approach chooses a mid-value from the three available in the TMR system by selecting the value that lies between the remaining two [4]. In the fourth, called quorum-majority voting (QMV) a threshold defining a quorum is set such that the non-faulty tasks will always control the majority vote [10]. In the fifth, most significant bits voting (MSBV), voting is only performed on the k most significant bits of the SEs output data, while the least significant bits of the information are ignored [4]. In the sixth, referred as common voter module (CVM), the voting procedure is based on a bit-by-bit principal and out-of-range checking [7]. The AMV, WAS, MVS and QMV are software techniques, while MSBV and CVM are hardware techniques. In the software approach each processor exchanges information with all others via messages and adjusts its own value. This generally imposes a substantial time overhead on the system, with results typically

being unsuitable for real-time applications. On the other hand the hardware approach which is based on median select algorithms imposes no time overhead on the system. Consequently, it is desirable to use hardware voting for time-critical applications that need tight synchronization. However existing hardware solutions are complex [7, 8, 10, 11] and usually require special synchronization mechanism to solve the voting problem.

3. Mid-value select voter architecture

The mid-value select voter architecture provides an environment in which real-time application interfaced to life critical functions can be executed correctly in the presence of:

(a) Detected error during serial data transfer between SE and IM or temporal inability of the SE to satisfy the real-time constraints.

(b) Disagreement among SEs' output values.

(c) Incorrect operation of one redundant SE.

To accomplish the above mentioned requirements we introduce an extended mid-value selection voting scheme which is capable of handling both non-classical errors (requirement (a)) and classical errors (requirements (b) and (c)). For detection and correction of data transfer errors we use Hamming coding technique. It allows us to both detect and correct a single error in each SE's channel, and to detect double errors. Double error is treated as an unrecoverable transmission error. To overcome the situation when one double error, during data transfer, appears we have increased the number of redundant SEs to four. In this case, we propose to substitute the erroneous data with a correct data accepted from a spare channel. This kind of selection strategy allows us to continue manipulation with three SE's communication errors free data.

To satisfy requirements (c) and (d) we propose an error masking scheme based on mid-value selection voting algorithm. In this proposal based on one-out-of-three voting scheme, to each of the selected value status information, referred to as *congruency level*, is appended. This status indicates how all three values are close to each other. We assume that during correct operation of SEs, disagreement among their output values never excides a prespecified threshold value, ε. To characterize situations when discrepancy between sensed values excides ε, we introduce three additional levels of decreased congruency. Cases that correspond to different levels of congruency are pictured in Fig. 1**.**

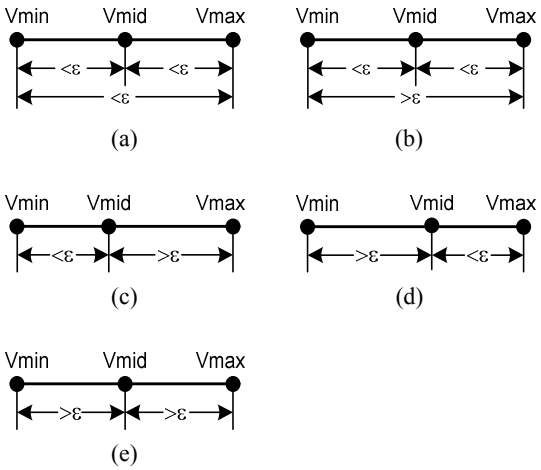


Fig. 1. Levels of congruency: (a) high; (b) decreased; (c) and (d) low; (e) failure.

Let *Vmin, Vmid* and *Vmax* represent, a minimal, mid (voted) and maximal sensed values, respectively. If all three possible differences among these three values are less then ε (Fig. 1. **Levels of congruency: (a) high; (b) decreased; (c) and (d) low; (e) failure.**

a) we conclude that high congruency among the sensed values exists. When $Vmax\text{-}Vmin>\varepsilon$ while $Vmid\text{-}Vmin<\varepsilon$ and $Vmax\text{-}Vmid<\varepsilon$ we say that a decreased level of congruency appears (Fig. 1. **Levels of congruency: (a) high; (b) decreased; (c) and (d) low; (e) failure.**

b). This case corresponds to the situation when all SEs generate output values that are within a range of the allowed full-scale inaccuracy limits (boundaries). The voting is characterized with low level of congruency when one of the difference *Vmid-Vmin* or *Vmax-Vmid* is less then ε, while the other is greater then ε (Fig. 1. **Levels of congruency: (a) high; (b) decreased; (c) and (d) low; (e) failure.**

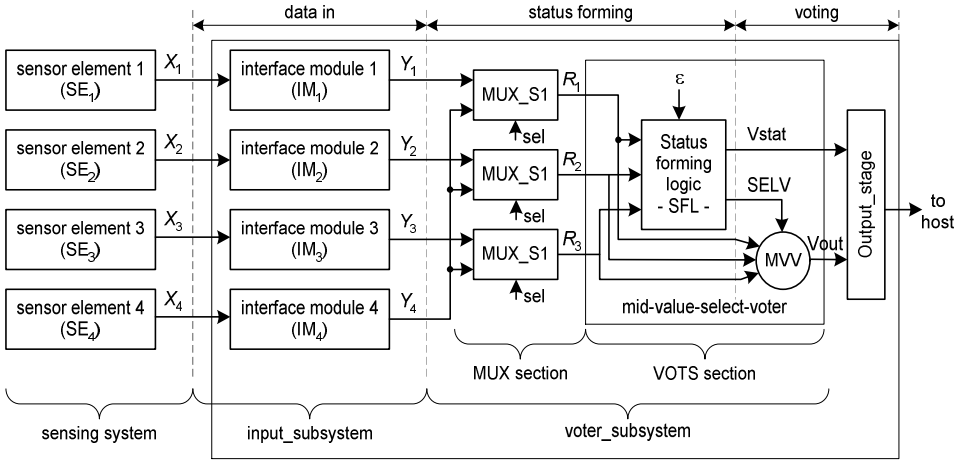
c,d)). In this case, one of the SE generates a value which is out-of-range concerning the specified full-scale inaccuracy. In a classical sense, this means that its value is outvoted as an unacceptable quantity, so some actions concerning system integrity are needed. For example, replacement of a defective SE with a correct one. If the differences *Vmid-Vmin* and *Vmax-Vmid* are greater then ε we say that a failure is detected (Fig. 1. **Levels of congruency: (a) high; (b) decreased; (c) and (d) low; (e) failure.**

e)). This case signals to a catastrophic situation,

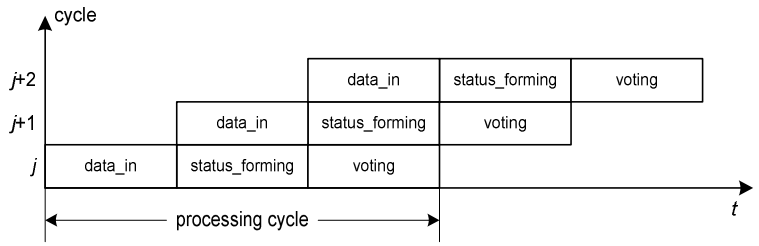
i.e. operator intervention is obligatory.

3.1 Global structure

In general, the FTDAS consists of four identical sensor elements (SEs) and a single MVSA. The SEs are located in lieu of a process. The SE picks-up a non-electrical/electrical quantity (pressure, temperature, voltage, charge, etc.), transforms it to an electrical quantity (if necessary) and after signal conditioning (filtering, amplification, etc.) converts it into a digital form (analog-to-digital conversion). The output section of the SE_i (i=1,…, 4) converts the sensed value to an encoded form using Hamming error-correcting code and transmits it as a serial data stream to the MVSA. Fig. 1a presents a block diagram of the MVSA. The MVSA is composed of two entities, denoted as Input_Subsystem and Voter_Subsystem. The Input_Subsystem consists of four interface modules (IMs). To each IM one SE is appended. The pair SE_4:IM_4 is used as a spare one. If during simultaneous serial data transfer through any of the input channels a single error appears it can be detected and corrected. Appearance of a double error in any one of the three channels X_1, X_2 or X_3 can be bypassed by substituting the erroneous data with the spare one coming from the channel X_4. The voter-subsystem logic is composed of two building blocks, Status-Forming logic (SFL) and Voter-Logic (MVV). The MVSA presents two types of information to the host, a voted_status, *Vstat*, and voted_value, *Vout*. The voted_status points to the congruency level of the accepted SEs values, and signals the unrecoverable error(s) in a corresponding channel(s). The voted_value corresponds to the mid-value selected SE input.



(a)



(b)

Fig. 1. Hardware structure of input section and processing of fault-tolerant data acquisition system.

A total time needed the data to pass from input to output of the MVSA is referred as a processing-cycle. It includes, as is sketched in Fig. 1b, three overlapping phases of equal time duration called as Data_In, Status_Forming, and Voting. Pipeline technique is used as a principle of data processing. The interface modules are involved in a data_in processing phase, while the hardware of MUX-section and mid-value-select-voter take part during Status_Forming and Voting processing phases. An

architectural organization, based on phase overlapping of all three different activities, allows us to use efficiently the MVSA's hardware resources.

The IM_i acts as an I/O controller. Its main intent is to provide correct data transfer and synchronization between constituents that do not share a common clock, the sensing system and Voter_Subsystem. Each sensed value is transferred to the IM_i as a single message. The message format consists of 16 bits of information plus 6 parity check bits. Data transfer between SE_i and IM_i is regulated by implementing a suitable handshake protocol at both bit-level and message-level. In order to avoid synchronization failures any message arriving into the IM_i is buffered. Additionally, the IM_i detects and corrects recoverable errors thanks to the installed error correcting logic. The IM outputs the 16-bit decoded value in a serial form, and two status bits that indicate to one of the following four conditions: (1) no error; (2) corrected single error; (3) double error, and (4) timing error.

As can be seen from Fig. 1a, the voter subsystem is composed of two sections, a multiplexer and voter, denoted as MUXS and VOTS, respectively. The MUXS section implements three-out-of-four data selection scheme, while the VOTS the mid-value-selection algorithm. The MUXS section is implemented as a switching network composed of three two-input multiplexers ($MUXS_1$-$MUXS_3$). Three-out-of-four selection is based on status bits generated by IMs what privides masking of one double or timing error appearing during data transfer between any SE:IM pair. If two or more double/timing errors appear in input channels then only one erroneous data with smallest index is substituted by the spare data, while the other is passed through.

The VTOS section performs two main tasks: (1) mid-value selection, and (2) estimation of the congruency level. It accepts three types of external inputs: (a) R_1, R_2 and R_3 as sensed values; (b) four pairs of two-bits signal indicating status of the individual channels, and (c) *limit-constant-signal* ε (n-bit constant (n=16) which corresponds to a maximally allowed difference between any two signal-inputs accepted from SEs). The VTOS generates two types of outputs. The first one, denoted as *Vout*, corresponds to the selected input value, while the second 15-bit signal, marked as *Vstat*, points to a congruency level of the voted value and an additional status information. In order to determine both the mid-value and congruency level, the VTOS, first, computes, in paralel, differences between any two input values, R_1, R_2 and R_3. The mid-value, *Vout*, is obtained by anlyzing signs of three differences. To estimate the congruency level, VTOS, additionally, computes magnitudes of differences and compares them with constant ε. The outcome of this comparison indicates to the congruency level, according to schema presented in Section 3.

4. Experimental results

In this section we report the results of the conducted experiments. We describe the structure of the simulation model used for the evaluation of our approach, and after that we analyze the experimental results.

4.1 Simulation model

In a concrete case we have decomposed the complete simulation procedure into two steps. The first one deals with operation of the IM and points to justification of the Hamming code implementation. Also it shows Hamming coding superiority in respect to the IM scheme which involve single parity bit encoding.

The second step relates to performance evaluation of the MVSA. It considers the overall MVSA operation taking into account both the dissemination of SEs' output values and communication errors.

4.1.1 Step one

Our simulation model treats the IM as an interconnection network with four input and three output ports. The model mimics operation of the IMs hardware structure already described in Subsection 3.1. The IM's response is simulated by injecting a specified number of errors in each set of four input coded data at arbitrary bit positions.

Figure 3. plots the expected number of correct IMs' out data, as a function of the number of injected errors, for two coding schemes: coding based on involving a single parity check bit, and coding based on Hamming-code.

As can be seen from Fig. 3, in both cases, due to implementation of a spare channel the system tolerates existence of single error. The system which uses Hamming-coding can tolerate two errors. In

both cases as the number of injected errors increases - the expected number of correctly generated results at the IMs output decreases. It is evident that implementation based on Hamming-code has superior performance in respect to single parity check bit coding. Note that for correct voting operation at least two communication error-free data are needed. For single parity check bit coding scheme this system's threshold of two correct results is reached just with two errors. By implementing Hamming coding the threshold of two correct results is moved to six errors.

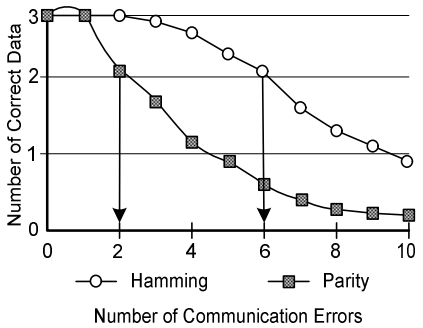


Fig. 3. Expected number of correct IMs' out data, as a function of the number of injected errors.

4.1.2 Step two

The second simulation step assumes that dissemination of IM's output data is direct consequence of:

(1) SE's full-scale inaccuracy.

(2) number of errors that appear during data transfer.

The obtained results during simulation are given in Fig. 4. The results relate to the congruency level of the values obtained at the MVSA output in term of the number of injected errors for various SEs full-scale inaccuracies.

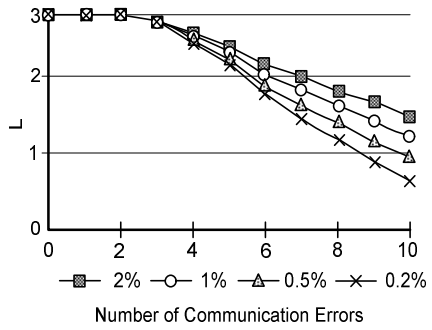


Fig. 4. Congruency level L as a function of injected errors.

In general the congruency level decreases as the number of errors increases, and full-scale inaccuracy decreases.

Finally let note that implementation of one spare channel and Hamming-coding significantly improves system tolerance to communication errors. The involved congruency level of output data, as an additional voting status attribute, improves in a great-deal the diagnostic capability of the MVSA.

4.2 Implementation

For 0,8μm CMOS technology [12] the chip encompasses a complexity of about 6850 gates (input section 4620, MUXS section 14, MVSS section 1339, output section 377, and control logic 495 (details concerning control logic implemented as a multiple FSMs (finite state machines) organized in a pipeline fashion are omitted in this paper).

5. Conclusion

The work presented here was inspired by the observations that data generated at the output of remote sensor elements in triple-modular real-time fault tolerant data acquisition system cannot be guarantied to match even in the absence of faults. This paper presents an algorithm and voter hardware structure thanks to which we implement a mid-value voting selection criterion. The voter logic is implemented as a linear interconnection structure with three processing stages connected to form a pipeline. The hardware mid-value select architecture as an ASIC has been synthesized from a pure algorithmic-level specification. In summary, the main features of the proposed mid-value select architecture are:

1. Four SEs instead of three are used in order to tolerate communication errors
2. Hamming code is used to detect/correct communication errors
3. Preliminary three-out-of-four selection is performed to eliminate sensed value corrupted by unrecoverable data transfer error.
4. Mid-value select voting is applied on three communication-error free values
5. Congruency level is determined and appended to the voted value

Additionally we have studied both the effects related to the number of injected transmission errors, between sensor elements and voter module, and the

effects of dissemination of sensor elements' output data on voter performance. Simulation results confirm the validity of implementation of both the Hamming-coding logic and voter-status logic.

References

[1] Avizienis A. Fault-Tolerant Computing - An Overview. IEEE Computer 1971; 4(1):5-8.

[2] Carter W. Fault Tolerant Computing : An Introduction and a Viewpoint. IEEE Trans Comput 1973; C-22(3): 225-29.

[3] Brilliant S, Knight J, Leveson N. Analysis of Faults in an N-version Software Experiment. IEEE Trans Software Engineering 1990; 16(2):238-47.

[4] Johnson B. Design and Analysis of Fault-Tolerant Digital Systems. Reading, MA: Addison-Wesley; 1989.

[5] Pradhan D, editor. Fault-Tolerant Computing : Theory and Techniques Vol. I, II. New Jersey: Prentice-Hall; 1986.

[6] Lala P. Fault Tolerant and Fault Testable Hardware Design. New Jersey: Prentice-Hall; 1985.

[7] Georgiev Z, Stojcev M. VLSI Common Voting Module for Fault-Tolerant TMR System in Industrial System Control Applications. International Journal of Electronics 1994; 76(2):163-205.

[8] Davis D, Wakerly JF. Synchronization and matching in redundant systems. IEEE Trans Comput 1978; C-27(6): 531-39.

[9] Stojcev MK, Djordjevic GLj, Krstic MD. A hardware mid-value select voter architecture. Microelectronics Journal 2001; 32(2):149-62.

[10] Shin K, Dolfer J. Alternative Majority-Voting Methods for Real-Time Computing Systems. IEEE Trans Reliability 1989; 38(1):58-64.

[11] S. McConnel, D. Siewiorek. Synchronization and Voting. IEEE Trans Comput 1981; C-30(2): 161-4.

[12] 2.0-Micron, 1.2-Micron, 1.0-Micron and 0.8-Micron Standard Cell Databook. Austria Mikro Systeme International 1996.