Using Triple Modular Redundant (TMR) Technique in Critical Systems Operation

Samira Abu Shernta[1], Ali A. Tamtum[1*]

[1] sss191278@gmail.com , [1*] aamtamtum@yahoo.com

[1,1*] Department of Electrical and Computer Engineering, Elmergib University, Libya
*Corresponding author email:
Received: 00 April 2018 / Accepted: 00 May 2018

ABSTRACT

Many computing systems used in applications of critical systems utilize fault tolerance criteria for normally continuing to operate. Operating in the presence of faults is required in many applications for safety and reliability such as in electric power distribution systems, telecommunications, medical life-support, nuclear reactor control, transportation, automotive, aircraft, and space vehicles. Such systems require continuity and reliability of service. One of the used techniques for meeting the severe reliability requirements inherent in certain future computer application is the use of Triple Modular Redundant (TMR) configuration. Essentially, this technique depends on voting two out of three system output levels. In this paper a fault-tolerant system is proposed using TMR configuration for processors and memory modules with spare model for - line self - reconfiguration. A voter is designed to pass reliable data and signals between processors and memory modules. The voter has the capability to analyze the error and stop the system on the proper time. The proposed system is designed at register level and tested using MATLAB simulation. A set of different faults are injected in different modules of the system in different data pater. The simulation results present the accuracy and capability of the proposed system with respect to faults as well as the ability of errors handing.

Keywords: Triple Modular Redundant (TMR); Critical systems; Voter; Computing systems.

1 Introduction

Many systems require continuity and reliability of service while operating in the presence of limited faults. The ability to deliver highest quality of service for which it is intended is very important criteria in critical systems. To fulfil these primary requirements, fault tolerant techniques are necessary to make sure these systems are fault-tolerant systems which continue to operate satisfactory in the presence of faults [1]. One of the used techniques for meeting the severe reliability requirements inherent in certain future computer application is the use of Triple Modular Redundant (TMR) configuration. The author in [2] stated that the TMR technique required tight synchronization between different units which achieved by using a single and very reliable clock to insure continuity of operation in fault tolerant systems. A fault tolerant system is a system that its behaviour is compatible with its specification in presence of faults in some of its components [3]. Faults in different operating systems are presented in many publications such as in [4] and [5] which represent detailed information regarding fault time latency and transient faults. The choice of error detection, fault handling techniques and their implementation as well as the classes of faults are presented in [6].

Multi-Version techniques based on the use of two or more versions or "variants" of a piece of software, executed either in sequence or in parallel are presented in [7]. Dynamic recovery is generally more hardware-efficient than voted systems, and it is, therefore, the approach of choice in resource-constrained systems especially in high performance scalable systems.. Its disadvantage is that computational delays occur during fault recovery where fault coverage is often low and special operating systems may be required [8].

Error detection checks that are employed in computer systems can be of different types, depending on the system and the fault of interest. Most error detection mechanisms are presented in [9]. Error coverage and mechanisms of error prediction and of latent errors are presented in [5, 10, 11]. Error detection methods such as Watchdog timers have been used since the early days of digital systems especially in embedded systems [12, 13].

The concept of redundancy implies the addition of information, resources, or time beyond what is needed for normal system operation. The redundancy can take one of four forms, including hardware redundancy, time redundancy, software redundancy, and information redundancy. The concept of hardware redundancy became more common and more practical, the cost of replicating hardware within a system is decreasing simply because the cost of hardware are decreasing. The Hardware redundancy means the addition of extra hardware, usually for the purpose either detecting errors or tolerating faults[14].

The most known hardware fault tolerance technique is triple modularity redundancy (TMR), which has been used in many fault tolerant systems. The use of TMR technique and its advantages as well as the use of multistage TMR with replicate voters are presented in [15] and [16]. In [17], a commodity chip multiprocessors (CMP) design with features for providing system-level soft error protection, is described with dual modular redundant (DMR) and triple modular redundant (TMR) systems. In [18], A hypothetical triple-modular redundant computer is subjected to a Monte Carlo program on the IBM 704, which simulates component failures. Two types of namely duplex and triple modular redundancy (TMR) systems are presented in [19]. More application and representations of TMR are presented in [20-22]

In this paper a fault-tolerant system is proposed using TMR configuration for processors and memory modules with spare model for –line self- reconfiguration. A voter is designed to pass reliable data and signals between processors and memory modules.

2 The Proposed TNR System

2.1 TMR Technique Review

The most known hardware fault tolerance technique is triple modularity redundancy (TMR), which has been used in many fault tolerant systems. The hardware unit (M) represented in Figure1 is triplicated and all three units work in parallel. The outputs of these three units are given to the voting element (V). The voting element accepts the outputs from the three modular and delivers the majority vote as output.

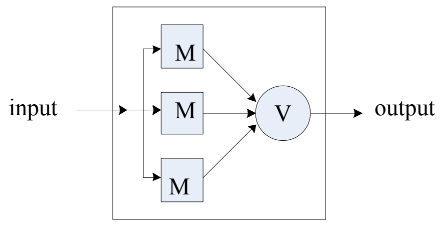

Figure 1: *Triple Modularity Redundancy (TMR) organization*

Clearly, the TMR organization can completely mask the failure of the one hardware unit. One of the features of TMR is that no explicit actions need to be performed for error detection, recovery, etc, TMR is particularly suitable for transient faults, since in the basic TMR the voter does not "remove" the faulty unit after an error occurs. This scheme cannot handle the failure of two units. In fact, once one unit fails, it is essential that both units should be work correctly (so that the voter can get a majority voted output). Due to this, the reliability of the TMR system becomes lower than a simplex system once a failure occurs.
The TMR scheme depends on the voting element. However, the voting element is typically a simple and highly reliable circuits. Another implementation aspect of TMR is that it requires tight synchronization between the different units. This has been frequently achieved by using a single clock. This requires the clock to be very reliable.

2.2 The Proposed System

Triple Modular Redundancy (TMR) configuration is the most efficient method to tolerate many types of faults and masking many types of errors at the system level. It is suitable for real time applications and online system reconfiguration where instant maintenance is not possible such as in Autopilot and unmanned space vehicles. This configuration tolerates the following set of faults:

- Faults effecting the operation of processors, memory modules and system buses.
- Faults produced from programs, compilers used to produce those programs
- Design and manufacturing faults in processors modules and memory modules.

Whereas the set of occurred errors that can be masked by this configuration includes the following classes:

- Processors internal transient errors.
- Processors internal intermittent errors.

- Data bus errors.
- Address bus Errors.
- Control and timing bus errors.
- Memory transient errors.
- Memory intermittent errors.
- Memory buses errors.

In a TMR configuration permanent errors caused by any faulty module are detected but not tolerated. Therefore the faulty module has to be replaced by a good one in order to resume system functions. Real-Time applications cause long down-time and increases Mean Time To Repair MTTR. In such a system the MTTR should be zero in order to recover from those errors and to continue system operations to achieve a high reliability.

To overcome a wide range of those errors and to tolerate that set of faults, a good configuration is proposed for a high reliable and available system with Self-Reconfiguration. In this proposed configuration, processors modules are treated separately from memory modules and the memory modules form also TMR subsystem. Another feature of this configuration is that voting is done at the signals level (data, address and control signals) between the processors modules and the memory modules.

According to this proposed configuration, the three processors (1,2,3) work in parallel and execute the same code and perform the same task. All signals outgoing from these processors are passed through a voter that compares these signals and passes the majority matched ones. If one processor does not match with the other two then the selected majority output from the voter is passed to the memory modules (or to the external I/O devices). Then that processor or its system bus is considered faulty and is given a time to recover from transient faults. If the same processors shows faulty outputs for more than a pre-specified attempts, it is considered as permanent faulty module and the whole system enters a reconfiguration procedure by bringing the spare processor to replace the faulty one.

The same process is done with the memory modules when data is read from memory to the processors. The voter is introduced with two sides: one side for the processors modules and the other side for the memories modules. The voter should also be designed in such a way to work as a comparator and by pass buffer. The voter should also have the mechanism to reconfigure the system by isolating (disconnecting) a faulty module and invoking (connecting) the spare module. The other task of the voter is to load the invoked processor with the current state of the other two processors by a roll-forward recovery procedure and resuming the system operation.

2.3 System Operation

The following assumptions are considered for the proposed system:

1 System is at start state.
2 All named model are loaded with the same copy of the program.
3 All three processors are ready to execute the same program.

4 The spare processor is physically connected but logically and electorally disconnected.

5 Give all general block diagram to used voter, three processors and three memories.

The block diagram of the voter are shown in Figure 2 which represents the composition of the Voter.

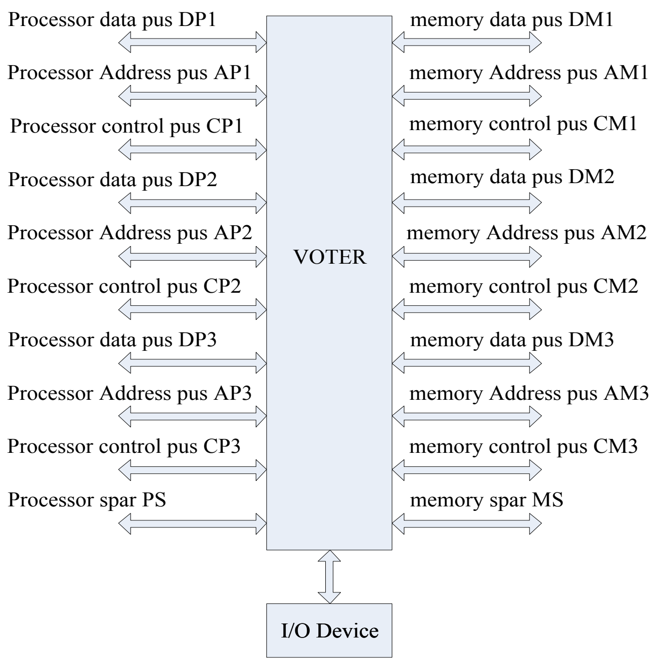

Figure 2: *Block Diagram of the voter*

In the input side of the Voter there are three processors (Data, Address, and Control) as well as a spare processor. Similarly, in the output of the Voter there are three memories (Data, Address, and Control) as well as a spare memory. Data will be transferred to the I/O devices in case of data saving fail.

Figure 3 represents operating flow chart in which the system starts working by applying either Read or Write command. The system is then tested whether it is working or not. If the system working, a check is made on Address, Control and Data. If not, processors are added to the system and the system is tested again. Then, the three processors are tested. In case of error detection, the damaged processor is specified and repaired and the system continues working. Then data writing and saving in the memory is done. On the other side the process of reading data from the memory is running. Then a test is made. In case of an error is detected, error is located and repaired. This process continues until finishing the desired job.

The operation starts in " write cycle" by entering data to pr1,pr2 and pr3. After that ,the operation is tested to know if there is an error or not?

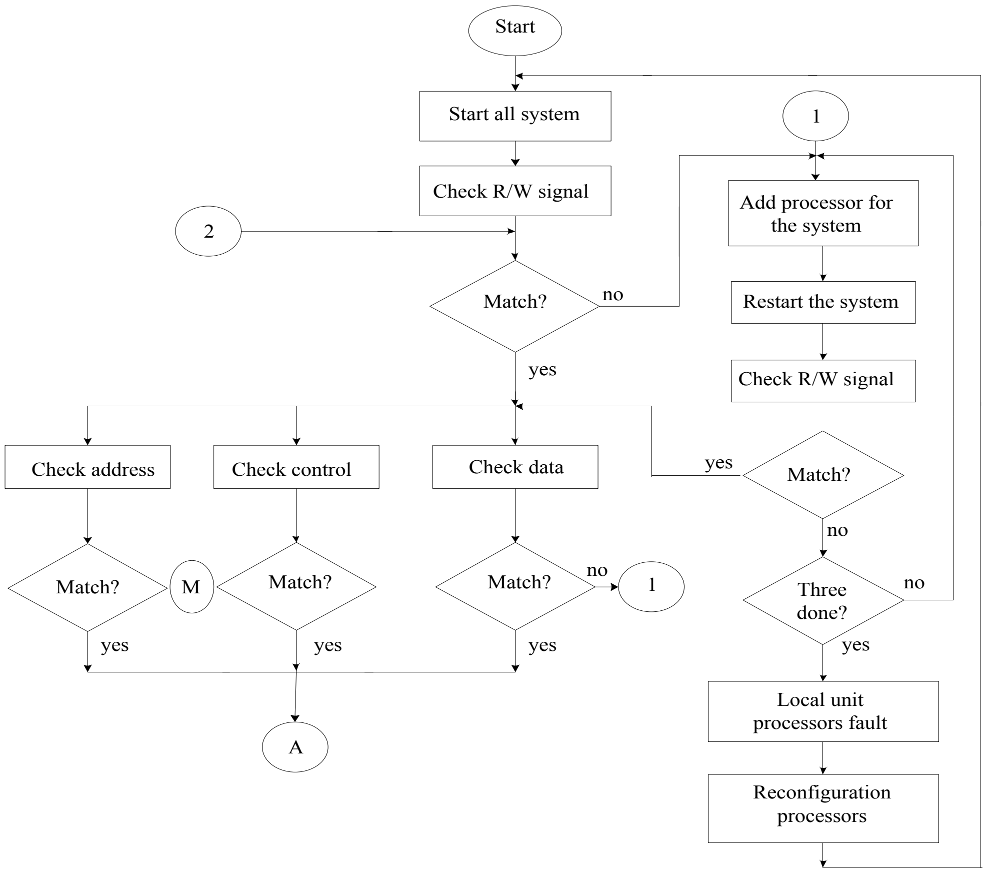

Figure 3: *Operating flow chart*

There are two cases "Yes" or "No".
"yes" means there is an error and another test will start to know whether the error is permanent or not. The voter will know in which process the error occurs.
" no" means that the error is transient and it may be regain by doing the operation again. Once the faulty process is known, it will be changed by a spare process to continues the operation .
Case II: If "NO" we need to know where the results will be sent. There are two options. Either the results will be sent to the memory and the data will be saved into "me1,me2 and me3 " or the data will be sent into the input ,output devices.
Regarding read cycle, the data will be read from the memory and the operation will continue to the end of the cycle in similar manner.

3 Numerical results

The results can be summarized in Tables 1, 2, 3,4, 5 and 6 which represent Processor Data Bus, Memory Data Bus, processor address Bus, memory address Bus, Control processor Bus and Control memory Bus including total time latency.

Table 1: *Processor Data Bus*

Module	Injected faults	Detected faults	Time Latency (Second)	Coverage	System Recovery (Y,N)
Dp1	0	0	2.7375e-005	0	N
Dp2	0	0	2.7375e-005	0	N
Dp3	4	3	2.7375e-005	75%	Y
Total	4	3	8.3991e-005	75%	Y

The system considered healthy when DP3 recorded failure two consecutive times. When number of errors =3, a permanent error in DP3 is recorded and changed with the spare one (DPs).

Table 2: *Memory Data Bus*

Module	Injected faults	Detected faults	Latency (Second)	Coverage	System Recovery (Y,N)
DM1	2	2	2.9241e-005	100%	Y
DM2	0	0	2.9241e-005	0	N
DM3	0	0	2.9241e-005	0	N
Total	2	2	8.7724e-005	100%	Y

A permanent error in DM1 is recorded with latency time = 2.9241e-005 sec; and total latency time = 8.7724e-005 sec; The error in DM1 is temporary and a 100% recovered.

Table 3: *processor address Bus*

Module	Injected faults	Detected faults	Latency (Second)	Coverage	System Recovery (Y,N)
Ap1	0	0	3.2352e-005	0	N
Ap2	3	3	3.2352e-005	100%=100%	Y
Ap3	0	0	3.2352e-005	0	N
Total	3	3	9.7055e-005	100%	Y

The number of injected errors equal to 3, and the error in AP2 is permanent and changed with the spare one (Aps).

Table 4: *memory address Bus*

Module	Injected faults	Detected faults	Latency (Second)	Coverage	System Recovery (Y,N)
AM1	0	0	2.8152e-005	0	N
AM2	0	0	2.8152e-005	0	N
AM3	6	3	2.8152e-005	3/6*100%=50%	Y
Total	6	3	8.4457e-005	50%	Y

Number of errors =3 and permanent error in AM3. The spare AMs replaces the mean AM3 with 50% coverage.

Table 5: *Control processor Bus*

Module	Injected faults	Detected faults	Latency (Second)	Coverage	System Recovery (Y,N)
Cp1	2	2	2.8774e-005	2/2*100%=100%	Y
Cp2	0	0	2.8774e-005	0	N
Cp3	0	0	2.8774e-005	0	N
Total	2	2	8.6323e-005	100%	Y

The error in CP1 is temporary with 100% coverage.

Table 6: *Control memory Bus*

Module	Injected faults	Detected faults	Latency (Second)	Coverage	System Recovery (Y,N)
CM1	0	0	2.9863e-005	0	N
CM2	3	3	2.9863e-005	3/3*100%=100%	Y
CM3	0	0	2.9863e-005	0	N
Total	3	3	8.9589e-005	100%	Y

The number of errors = 3 and the error in CM2 is permanent. The spare CMs is utilized instead of using the mean CM2.

4 Conclusions

The principles and concepts of fault tolerance were introduced and investigated. The analysis was devoted to the online error detection and mainly focused on the use triplication techniques. According to the outcome from the survey of the online error detection techniques and investigation of some previous systems, a TMR system configuration was proposed to increase system reliability and availability for self – reconfigurable application. In this system both the processor and memory module are triplicated with one spare module. A voter was designed to pass reliable data and singles between processors module and memory modules. The voter has the capability to stop the system and analysis the error. It enters the system for roll- back procedure in case of transient error or it replaces the faulty module with the spare one in case of permanent error. Thus system is recovered and resumes its operation on line which achieves the target objective. To verify the capabilities and the behaviour of proposed system and voter design, the system is simulated using MATLAB package. A set of faults are injected in different information paths and the response of the system was monitored.

References

[1] N. Kim and S. Gupta "Testing of Digital Systems" , *Cambridge University press 2003.*

[2] Chris Weaver , Todd Austin, "A Fault Tolerant Approach to Microprocessor Design" *Advanced Computer Architccture Laboratory University of Michigan*, July 2001.

[3] Teijo Lehtonen , Juha Plosila, Jouni Isoaho , "On Fault Tolerance Techniques towards Nanoscale Circuits and Systems " *Turku Center for Computer Science*, TUCS Technical Report, August2005..

[4] Ali H. Maamar , Asma y. Elhawadi, "Self Checking Register file" *Computer Department Higher Institute of Electronics, Beni- Waled*, APRIL 1999.

[5] Lisboa,C.A. Erigson, M.I. and Carro, l.and carro ,L, "System level approaches for mitigation of long duration transient fanlts in future technologies", *12th IEEE European Tcst Symposium (ETS,07),* 2007.

[6] Januu Sosnowski, "Transient fault Tolerance in Digital System", *Warsaw University of technology, IEEE* , 1994.

[7] Subhasish Mitra, "Diversity Techniques for Concurrent Error Detection" *Technical Report , Center for Reliable Computing* , may 2000.

[8] Parg K Lala "Sef- checking and fault –Tolerance Digital Design", *Morgan Kaufmann Publisher*,2001.

[9] Manoj Franklin , "A Study of Time Redundant Fault Tolerance Techniques for Superscalar processors", *Departmcnt of Electrical &computer Engineering , Clemson Universty ,Clemson, USA*, 1995 IEEE.

[10] Stanislaw J.Piestrak," Design of fast self -testing checkers a Class of Berger Codes", *IEEE Transaction on Computer*, MAY 1987.

[11] Kim and K. G. Shin , "Evaluation of Fault Tolerance Latency from Real -Time Application 's Perspectives", *IEEE Transactions on Computers*, vol . 49 No 1, Jan. 2000.

[12] Robert Redinbo, " Generalized Algorithm-Based Fault Tolerance: Error Correction via Kalman Estimation", *IEEE Transactions on Computers*, Vol. 47, No. 6, June 1998.

[13] Robert Redinbo, " Generalized Algorithm-Based Fault Tolerance: Error Correction via Kalman Estimation", *IEEE Transactions on Computers,* Vol. 47, No. 6, June 1998.

[14] Constantinescu, "Teraflops Supercomputer : Architecture and Validation of the Fault Tolerance Mechanisms", *IEEE Transactions on Computers*, Sep 2000.

[15] Karri, K. Kim, and M. Potkonjak, " Computer Aided Design of Fault-Tolerant Application Specific Programmable Processors", *IEEE Transactions on Computers*, Nov 2000.

[16] Dutt and N. R. Mahapatra, "Node-Covering, Error Correcting Codes and Multiprocessors With Very High Average Fault Tolerance*", IEEE Transactions on Computers*, Sept 1997.

[17] James E. Smith, " Motivating Commodity Multi-Core Processor Design For System-Level Error Protection", *Kewal K. Saluja* 2006.

[18] George W. Grosline, "The Use of Triple-Modular Redundancy to Improve Computer Reliability", *IBM Journal*, April 1962.

[19] Rami Melhem, "Energy-Efficient Duplex and TMR Real-Time Systems Appeared in the IEEE Real-Time Systems Symposium", *Computer Science Department, University of Pittsburgh*, Dec 2002.

[20] Dmitry Burlyaev, Pascal Fradet, Alain Girault, "Verification-guided voter minimization in triple-modular redundant circuits", *Automatin & Test in Europe Conference & Exhibition (DATE),* Year: 2014.

[21] Jeffrey Prinzie, Michiel Steyaert, Paul Leroux, Jorgen Chrisiansen, Paulo Moreira, "A single-event upset robust, 2.2 GHz, to 3.2 GHz, 345fs jitter PLL with triple-modular redundant phase detector in 65 nm CMOS", *IEEE Asian Solid-State Circuits Conference (A-SSCC),* Year: 2016..

[22] Pang Zh, Qi Zheng, Zhankui Zeng, Liman Yaung, "The single integrity design and simulation of triple-modular redundant (TMR) computer", *IEEE International Conference on Cybernetics and Intelligent Systems (CIS) and , IEEE Conference on Robotics, Automation and Mechatronics (RAM),* Year: 2017.