Time and Space Redundancy Fault Tolerance Trade-offs for FPGA Based Single and Multicore Designs

By Mohamed Mahmoud Ibrahim, Kenichi Asami, and Mengu Cho

Kyushu Institute of Technology, Kitakyushu, Japan

This paper investigates the gains and losses in terms of power, area, reliability, and speed when applying time redundancy fault tolerance techniques on single core designs compared to space redundancy fault tolerance techniques applied to multi-core designs. The system is developed on the virtex5 FPGA from Xilinx, it uses 65nm technology with a relatively moderate to high static power consumption. The system consists of two design alternatives. The first is a single core embedded processing system that applies time redundancy fault tolerance through execution repetition to perform self-check pointing through consensus. The second system is built from 3 soft IP core processors which perform a space redundancy approach through Triple-Modular-Redundancy (TMR) with feedback among the processors. The performance of both systems is evaluated in terms of the execution speed and latency due to fault tolerance techniques compared to the non-fault tolerant system.

Key Words: Multicore systems, Fault Tolerance, Space Redundancy, Time Redundancy, FPGA

1. Introduction

Developing robust space avionics systems is a challenging task. As the missions diverse and increase their data processing requirements, the need for fast and reliable data processing systems emerges. Nevertheless, a balance should be hit between four main parameters: the power, the processing speed, the mass, and the reliability. It is required to optimize the design to have a reliable system with low power consumption and low mass while having high processing capabilities.

Nowadays, modern Field Programmable Gate Arrays (FPGA) provides the opportunity to develop complex digital designs with high speed and at moderate power consumption [1][2]. Single and multi-core processor systems are integrated with custom designed logic cores to serve the different design needs [3].

In developing space systems, several design techniques are commonly used to design reliable systems. Fault tolerance and fault avoidance are the common techniques [4-6]. Fault avoidance depends on preventing the faults from occurring in the functioning design. Fault tolerance depends on tolerating the effects that faults might introduce to the design in a way that keeps it functioning in an accepted performance. The concept of redundancy is the base for fault tolerant designs. Redundancy can take place in repeating the functioning design units all or in part with the same or diverse designs; in this case it is called space redundancy. A voter is used to judge between the results of the redundant units. Time redundancy is about repeating the execution of some of the program critical functions several times to reach a consensus among the results. Data redundancy is to add additional data bits to the original data where the additional bits will carry the Error Detection And Correction (EDAC) code that can be used to detect and correct faults in the data stream.

Software is a basic counterpart in developing complex system. Fault tolerant techniques are developed for software protection. N-version programming, N-copy programming, recovery blocks, and check-pointing are among the common techniques [7]. To protect the operation of a system, a hybrid of the techniques is used [8]. Fault injection and radiation testing are used to test the system robustness to bit flips

In this paper we present a comparative study between using single core processor in carrying the system tasks and using triple core redundancy. The comparison takes place in terms of power, speed, resources utilization and reliability. Both systems are implemented in a Static Random Access Memory (SRAM) based FPGA. The systems were designed using the Xilinx FPGA Virtex 5 LX50T. It is a 65nm FPGA. The Embedded Development Kit (EDK) tool from Xilinx was used to design a single processor system and a triple processor system. Bubble sort algorithm was implemented and run on both systems to detect the average speed when applying redundancy in implementing the algorithm. The power consumption and resources utilization were estimated and the total system reliability is calculated in both cases.

The objective of this work is to clearly understand the advantages and disadvantages of using space and time redundancy in 65nm FPGAs. The trade-offs in selecting either of the two techniques are: selecting a design that is economic in its power consumption, provides high reliability, performs in a high standard, and achieves reasonable utilization of the FPGA resources.

The paper proceeds as follows: Section 2 presents the different architecture alternatives when developing an embedded processor system. The systems designs are presented in section 3. The results of testing the systems are presented and discussed in section 4. The conclusion and future work are presented in section 5.

2. Architecture Alternatives

When developing an embedded system, the main variant for the different architectures is about how the processor and memory are interfaced. The simple embedded processor system as shown in figure 1, consists of a system bus a microprocessor or microcontroller and other peripherals connected to the bus. The peripherals might contain timers, interrupt controllers, Input/Output processors, Direct Memory Access (DMA) controllers, and custom logic that implements specific functions. The memory and the system bus are crucial parts in the architecture when many processors are to be integrated together. They define how the processors access the memory for data storage and retrieval and/or instructions fetching.

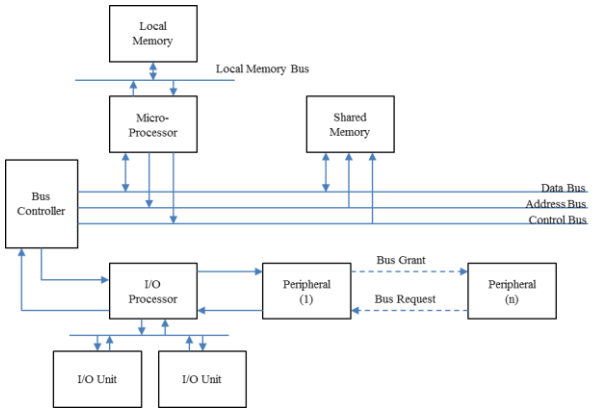


Fig. 1. Simple embedded system architecture.

In our previous paper [10] we have shown that four general architectures do exist when classifying the memory and processor interfaces:

1- Multi-Processor-Multi-Memory (MPMM)
2- Multi-Processor-Single-Memory (MPSM)
3- Single-Processor-Single Memory (SPSM)
4- Single-Processor-Multi-Memory (SPMM)

The difference among the four architectures is in the number of processors and memories which are interfaced together. The system reliability is also affected with the architecture. Assuming that (Rp) is the processor reliability and that (Rm) is the memory reliability, Table 1 [10] shows the different architectures with the estimation of the reliabilities formulas.

The MPSM makes use of the space redundancy concept where three processors operate in parallel to calculate the same operations to reach a consensus. Both systems run a bubble sort algorithm for comparing their performances. The

The calculation of the system reliability depends on the reliability of the attached memory and processors. The configuration through which the system components are connected leads to the form of the reliability formula. It is important to notice that single processor system whether connected to single memory or multiple memories is used when time redundancy in executing the software that runs on the processor will be adopted. In the case of the single memory the data can be stored in multiple buffers to provide redundancy in the storage. Multiple-Memory systems maintain an exact copy of all memory contents between the redundant units. The system can still have an additional form of redundancy by storing data in a redundant form in each memory while still having each memory repeated in a space redundancy.

The Multi-Processor systems have two conditions: the shared memory and the non-shared memory. In the shared memory systems the processors share the memories where they store and retrieve the data as well as the code memories from which they fetch the instructions. The MPSM and the MPMM with shared memories are examples of a tightly coupled multiprocessor system. The MPSM (non-shared) and the MPMM (non-shared) are examples for the loosely coupled multiprocessor systems.

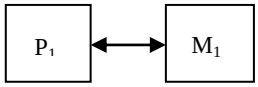
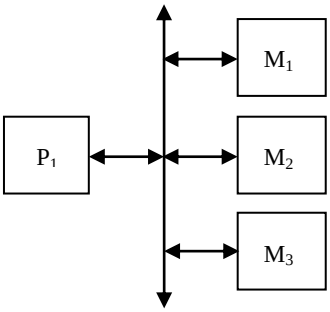
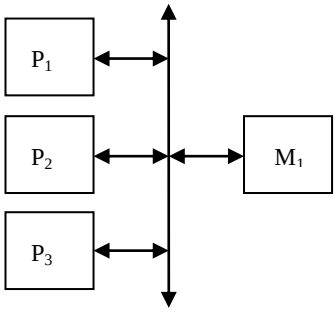
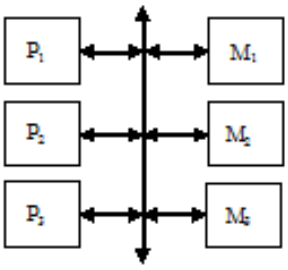
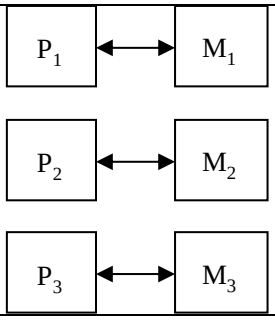
The design of a fault tolerant embedded system usually merges different techniques together. The use of redundant memories and processors adds to the reliability as well as increasing the complexity of the system and its power consumption. A system that contains reasonable number of units in space redundancy, to maintain the power consumption and reliability, while adopting time redundancy techniques, is the ultimate choice. The reliability formulas which are shown in table 1, are estimated based on the processor reliability and the memory reliability. The processor reliability is estimated based on the FPGA reliability and the Failure In Time (FIT) for the design [9]. One FIT is calculated as 1 failure per 1 billion device hours (109 hours). The FIT for the Virtex-5 family at 65nm technology is 165 FIT/Mb[9]. The calculation of the FIT depends on the occupied device slices. The design is stored in the form of binary stream in the internal SRAM of the FPGA. The size of the design multiplied by the FIT per Mbit of the specific device being used gives the total FIT for that design. The FIT can be used as the failure rate (λ) to calculate the reliability over a period of time (T).

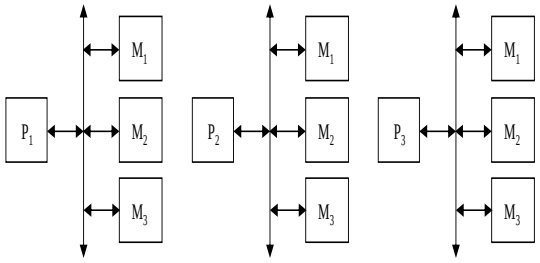
$$R = e^{-\lambda T} \qquad (1.)$$

In our designs we test the SPSM architecture and the MPSM (non-shared) architectures. The SPSM makes use of repetition of execution over the time and storage of results in extra copies.

test is run for 100 times and each time a vector of length 100 words is randomly generated. The time histogram for sorting the vector and performing the fault tolerance check of the results voting is plotted for the non-fault tolerant system, the space and time redundancy systems.

Table 1. Architecture alternatives and reliability estimations [10].

Type	Architecture	Reliability Formula
SPSM		$R = R_p R_m$
SPMM		$R = R_p\,(R_m^3 + 3R_m^2(1 - R_m))$
MPSM (shared memory)		$R = R_m\,(R_p^3 + 3R_p^2(1 - R_p))$
MPMM (shared memory)		$R = (R_p^3 + 3R_p^2(1 - R_p)\,(R_m^3 + 3R_m^2(1 - R_m)$
MPSM (non-shared memory)		$R = R_{ps}^3 + 3R_{ps}^2(1 - R_{ps})$ $R_{PS} = R_p R_m$

MPMM (non-shared memory)		$R = R_{ps}^3 + 3R_{ps}^2(1 - R_{ps})$ $R_{ps} = R_p\,(R_m^3 + 3R_m^2(1 - R_m))$

3. System Design

Two separate systems were implemented to test the trade-offs of using time and space redundancy in the Virtex 5LX50T FPGA: single processor system and Multi-Processor system. In the single processor system as shown in figure 2a, a MicroBlaze Processor is connected to a Local Memory Bus (LMB) where a local Block Random Access Memory (BRAM) is attached. The processor is connected to other peripherals through the Processor Local Bus (PLB). A watch dog timer is used to reset the system in case the processor stopped working. The processor receives an interrupt from the Interrupt Controller (INTC) that the watch dog timer finished counting and should be reinitialized. If the processor was working and did not hang up, it will respond to the watch dog timer interrupt and will reset it. In case the processor stopped working for any reason it will not respond to the watch dog timer interrupt. The watch dog timer will send a reset request to the reset module. The reset module will then reset the whole system including the processor. The system contains a system control processor which handles the correction of single bit errors that might happen in the configuration bit stream through reading the configuration frames via the Internal Configuration Access Port (ICAP) core. The communication between the system control processor and the Microblaze processor takes place through the mailbox IP core. A timer is included in the system to provide the ticks needed for an operating system to operate such as the Xilinx Kernel. Two Timers exist in the same IP core , one of them is used for measuring the execution time of the code in this experiment. The Microblaze Debug Module (MDM) port is used to debug the software application running on the Microblaze processor and the system control processor. The peripherals are connected to the PLB. Xilinx General Purpose Input Output (XGPIO) is used to input and output digital signals. It provides a control interface to the outside world. The Universal Asynchronous Receiver Transmitter (UART) is used to send and receive serial streams to and from the processor.

When three processors are used their data will be exchanged among them after each execution cycle. Therefore they should have an inter-processor communication mechanism. This mechanism is provided through the mailbox IP core. As shown in figure 2b, the three processors send the data to each other. Each processor would perform voting on its own locally generated data and the data provided by the other two processors. The voting takes place on bit level where:

$$V = A.B + A.C + B.C \qquad (2.)$$

A, B, C are binary words and the logic operations are the logical (AND) and logical (OR). (V) is the voting result. The results of the voting are then used by each processor in its operation.

This scheme we call it Cross-voting as each processor makes voting with the other processors. The operation concept of the system is shown in figure 2c, a random number stream of specified length, in this case it is 100 words, is generated by random number generation in MATALB. It is then sent to the processors through the serial ports RS232 interface. The processors save the received vector and wait for a signal to start the bubble sort algorithm. When the start signal is issued by the MATLAB script, the processors start the timers and initiate the Bubble sort algorithm after the bubble sort finishes the processors exchange the values among each other and carry on the cross voting. The voting results are sent to the MATLAB script, the test cycle continues until the required numbers of experiments are executed. The timers are stopped whenever the execution of the bubble algorithm finishes together with the data exchange between the processors and the voting mechanisms. The results are then sent to the MTALAB script for statistical analysis. In case of single processor no data exchange takes place. The processor stores three copies of the data vector in its local memory. The bubble sort function runs three times and the results are stored in three different vectors. The voting takes place between the vectors stored in the local memory. The results are then sent to the MATLAB script for statistical analysis. Figure 3, shows the flowchart of the algorithm in case of single processor and triple processors.

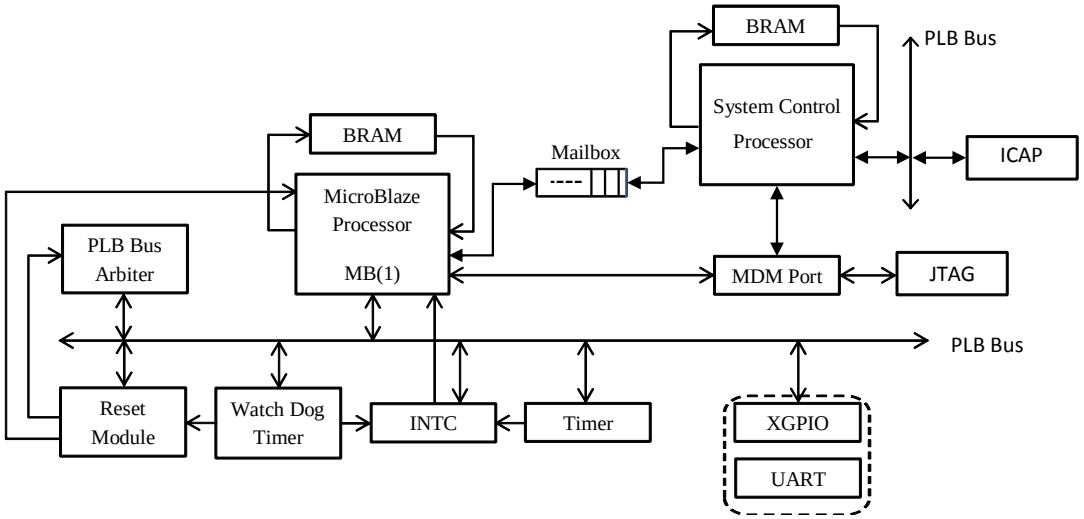


Fig. 2a. Single Processor System.

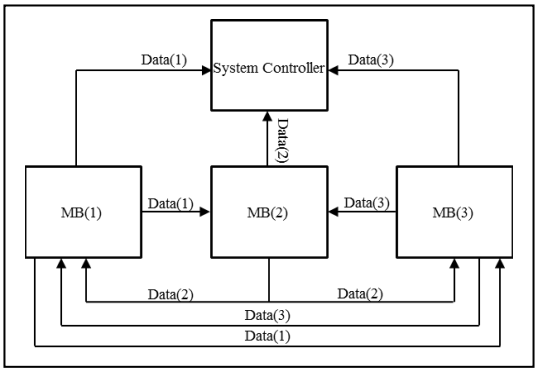


Fig. 2b. Inter-processor communication mechanism.

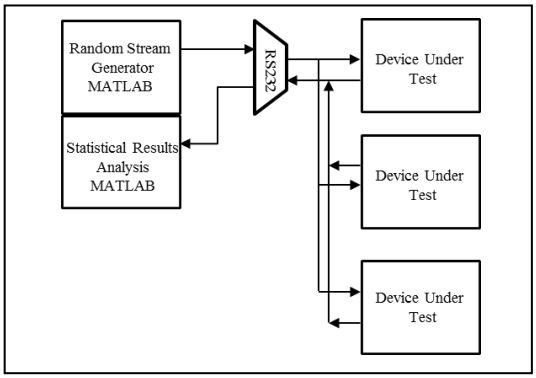


Fig. 2c. System operation concept.

Fig. 2. Single and Multi-core system design and operation concept.

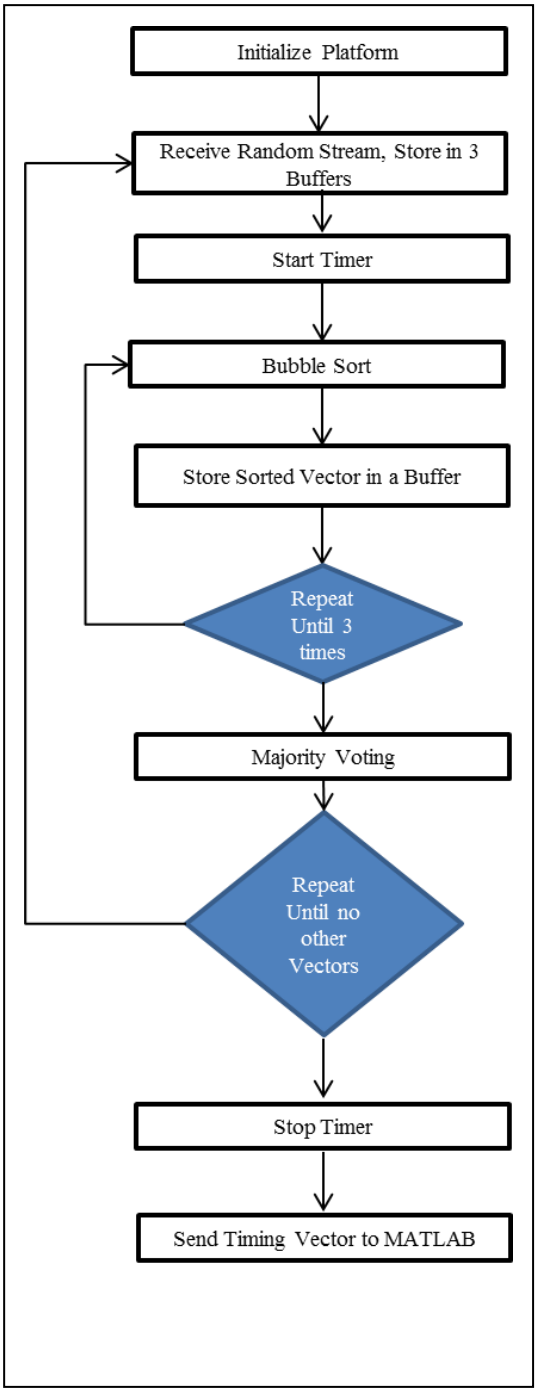


Fig. 3a. Flow chart of operation in case of 1 processor system.

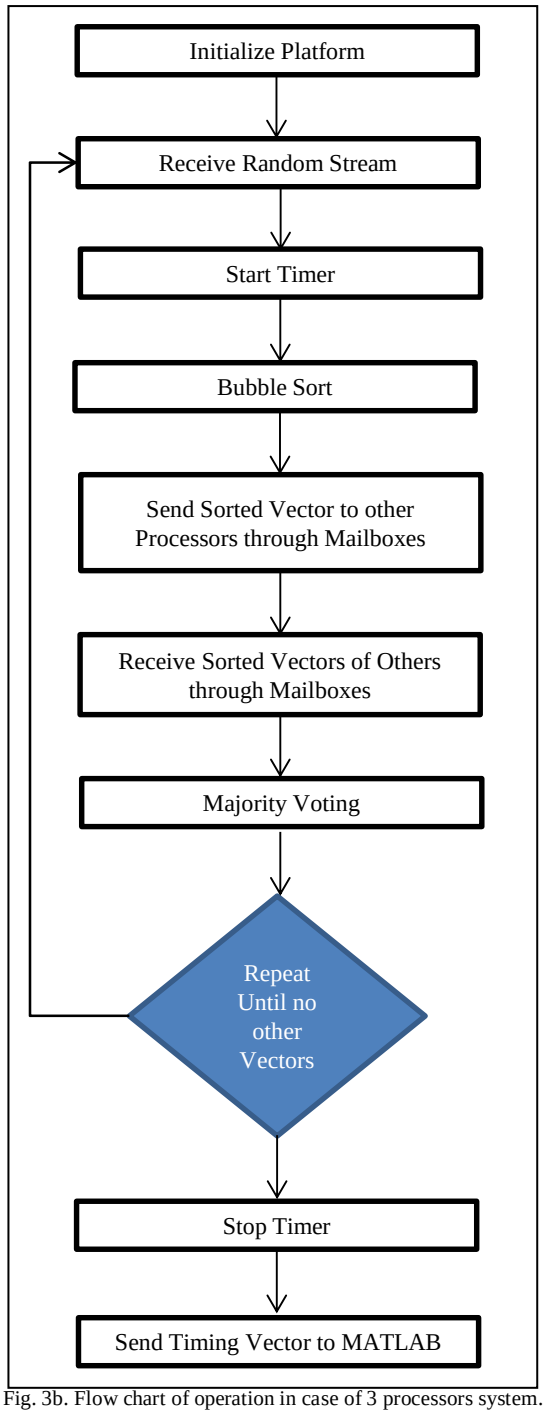


Fig. 3b. Flow chart of operation in case of 3 processors system.

Fig. 3. Operation flow chart.

4. Results and Discussion

The systems were tested using vectors of randomly generated data words. The time span of execution was collected for the non-fault tolerant system, the time redundant system and the space redundant system. Figure 4 shows the execution times histograms of the three cases.

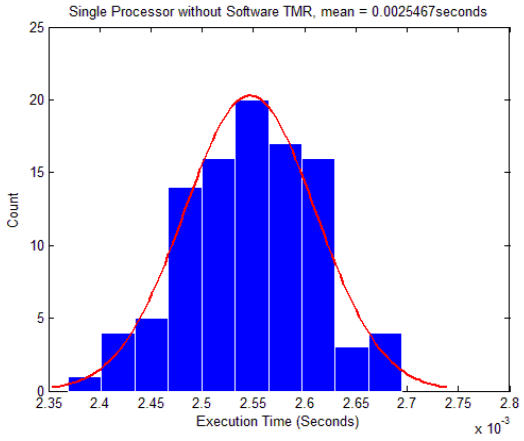


Fig. 4a. Single processor time histogram without Software TMR.

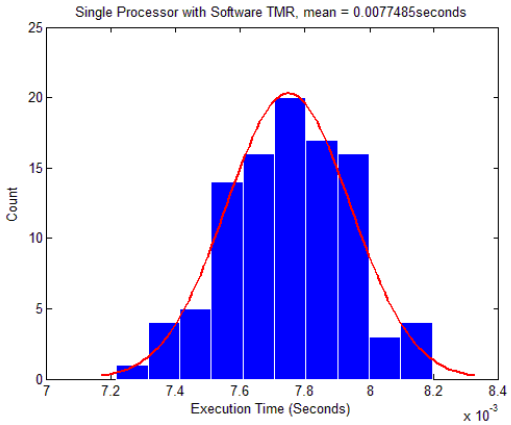


Fig. 4b. Single processor time histogram with Software TMR.

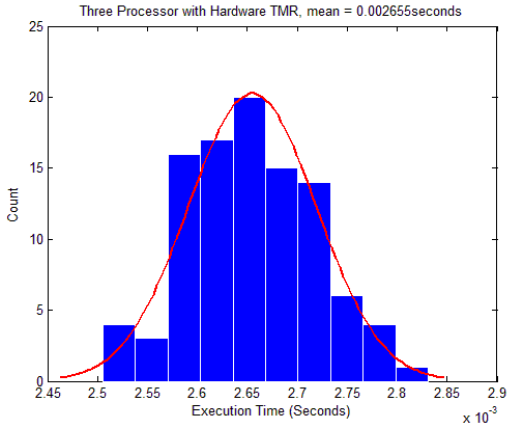


Fig. 4c. Single processor time histogram with Hardware TMR.

Fig. 4. Execution times histograms for single and multicore systems .

Each time a random sequence vector was generated based on the clock seed of the computer system running MATLAB to avoid repeating patterns of random sequences. The length of each vector was 100 words. The numbers of vectors applied during the test were 100 vectors. A normal distribution fit was applied to the histograms in figure 4. The statistical parameters of each histogram are shown in Table 2.

Table 2. Single processor without software TMR statistics.

Parameter	No-Fault Tolerance	Time Redundancy	Space Redundancy
Min	0.002354	0.007171	0.002463
Max	0.002739	0.008326	0.002847
Mean	0.002547	0.007749	0.002655
Median	0.002547	0.007749	0.002655
Mode	0.002354	0.007171	0.002463
Std	0.0001129	0.0003386	0.0001127
Range	0.0003852	0.001156	0.0003847

The mean execution time for the single processor without software TMR is about 2.55ms. It is almost the same as the mean execution time of the hardware TMR, space redundancy, which is 2.65ms. The mean execution time of the single processor with software TMR, time redundancy, is 7.75ms. almost 3 times higher than the non-fault tolerant and the space redundancy fault tolerance. This means that hardware redundancy is better in terms of execution time as it is as fast as the non-fault tolerant system which contains no overheads. However, we have to carefully notice that adding complicated data exchange protocols between the processors will add an execution overhead hence increasing the execution time significantly. If too much time is spent in handling the communication between the processors then the execution time might be close to the time redundancy case.

The power consumption varies between the single core and the multicore systems as shown in Table 3. The power consumption was estimated using the XPower Analyzer from Xilinx. The total power consumed by the multi-core system is 1.26Watt which is only about 28% higher than the power consumed by the single core system. About 0.46Watt is dissipated in the form of leakage. . The leakage power is mainly due to the leakage current in the static power consumption of the FPGA transistors [11]. As the feature size of the transistors is minimized, the leakage current increases.

Table 3. Power consumption in single and multi-core systems.

On-Chip	**Single Core Power(W)**	**Multi-core Power(W)**
Clocks	0.170	0.347
Logic	0.003	0.007
Signals	0.005	0.014
BRAMs	0.010	0.096
DSPs	0.000	0.000
PLLs	0.263	0.263
DCMs	0.068	0.068
IOs	0.001	0.002
Leakage	0.458	0.460
Total	0.978	1.256

The FPGA resources are almost utilized by the multi-core system while about less than one-third is utilized by the single core system as shown in Tables 4 and 5.

Table 4. Resources utilization in single core system.

Slice Logic Utilization	Used	Available	Utilization
Number of Slice Registers	5,932	28,800	20%
Number of Slice LUTs	6,866	28,800	23%
Number of Occupied Slices	3,337	7,200	46%
Number of BRAM/FIFO	18	60	30%
Total Memory Used (KB)	648	2,160	30%

Table 5. Resources utilization in multi-core system.

Slice Logic Utilization	Used	Available	Utilization
Number of Slice Registers	16,362	28,800	56%
Number of Slice LUTs	18,763	28,800	65%
Number of Occupied Slices	7,076	7,200	98%
Number of BRAM/FIFO	42	60	70%
Total Memory Used (KB)	1,512	2,160	70%

The multi-core system makes better use of the resources and it executes at almost triple the speed of the single core system while its power consumption is only 28% higher than it. The multi-core which uses the space redundancy for implementing fault tolerance in the 65nm Virtex 5 FPGA through repeating redundant hardware processor cores is more efficient and effective than the single core design. In terms of reliability the TMR design is better than the single core design as indicated in Table 1 [10].

5. Conclusion and future work

This paper studied the effects of using hardware redundancy and software redundancy on the resources utilization, power consumption and execution speeds in single and multi-core designs of the Virtex-5 FPGA. It is recommended according to the obtained results to make use of space redundancy approaches when designing digital systems using the Xilinx Virtex5 65nm FPGA. This is valid due to the fact that considerable portion of the power consumed is dissipated in the form of leakage power. Adding extra logic did not add much to the total power consumed. The reliability of the space redundant system is higher and its execution speed is better as far as the communication protocol between the cores does not add much overhead. Resources are better to be utilized in the FPGA device rather than wasting them. The space redundant system makes higher utilization of the FPGA resources.

We recommend repeating the work in this paper on different algorithms and applying a more time consuming communication protocol. A comparative study between the 65nm FPGA and other families such as the 28nm Virtex7 would be beneficial as well.

References

1) Xilinx website: http://www.xilinx.com
2) Actel website: http://www.actel.com
3) V. Asokan, " Designing Multiprocessor Systems in Platform Studio, White Paper," Xilinx White Paper, WP262, vol. 262, 2007, pp. 1-18.
4) Israel Koren, Mani Krishna, "Fault Tolerant systems", ElSEVIER 2007.
5) philip P. Shirvani, "Fault-Tolerant Computing for Radiation Environments," Ph.D thesis, Center for reliable Computing, Stanford University, June, 2001.
6) F.D. Lima and L. Carro, "Fault Tolerance Techniques for SRAM-based FPGAs," , Springer, 2006.
7) Laura L. Pullum, "Software fault tolerance techniques and implementation, " Artech House, London, 2001.
8) Olga Goloubeva, "software implemented hardware fault tolerance", Springer 2006.
9) XILINX, "Device Reliability Report First Quarter 2012," XILINX UG116 (v9.0), 2012.
10) Ibrahim, M., Asami, K., and Cho, M., "Fault Tolerant Architecture Alternatives for Developing Nano-Satellites Embedded Computers", Online Proc. of AIAA SPACE 2012 Conference & Exposition, Pasadena, California, USA, September 11-13, 2012.
11) Derek Curd, "Power Consumption in 65 nm FPGAs," Xilinx Corporation, WP246, vol. 246, February 2007.