Hindawi Publishing Corporation
International Journal of Reconfigurable Computing
Volume 2014, Article ID 806237, 8 pages
http://dx.doi.org/10.1155/2014/806237

Research Article

Low-Cost Fault Tolerant Methodology for Real Time MPSoC Based Embedded System

Mohsin Amin, Muhammad Shakir, Aqib Javed, Muhammad Hassan, and Syed Ali Raza

Department of Electrical Engineering, COMSATS Institute of Information and Technology, Abbottabad, Pakistan

Correspondence should be addressed to Mohsin Amin; mohsinamin@hotmail.com

Received 28 July 2014; Revised 27 October 2014; Accepted 27 October 2014; Published 19 November 2014

Academic Editor: Nadia Nedjah

We are proposing a design methodology for a fault tolerant homogeneous MPSoC having additional design objectives that include low hardware overhead and performance. We have implemented three different FT methodologies on MPSoCs and compared them against the defined constraints. The comparison of these FT methodologies is carried out by modelling their architectures in VHDL-RTL, on Spartan 3 FPGA. The results obtained through simulations helped us to identify the most relevant scheme in terms of the given design constraints.

1. Introduction

There has always been an urge of high performance; subsequent to this, plenty of efforts have been made to attain higher levels in said aspect. In past, technology scaling was mainly used for performance improvement. This approach was quite helpful in handling various applications with high processing demands; however bottleneck of this approach has been reached now [1]. Continuous transistor scaling resulted in reduced nodal capacitances and low supply voltages making the devices more prone to the transient faults caused by external (alpha particles and neutrons) and internal (power supply noise and cross talk noise) noises [2].

Transient faults are more likely to occur than permanent faults [3]; that is, product monitoring shows that permanent faults rate as a result of external events is not more than 10 FIT (failure in time) whereas the transient fault rate of 1 Mbit of SRAM is around 1,000 FIT for modern process technologies. Thus transient faults have major part in disturbing the reliability of the system [4].

Methods such as rollback are considered effective in rectifying transient errors. However in hard real time systems such corrections have a huge influence on the response time of the system. A deadline miss in the task can result in the failure of the system [5]. For example, ATM machines, cell phones, and thumbprint scanners are some real time systems.

In last decade, to address high performance need there has been a shift towards multiple resource system. Different interconnect structures have been employed in such systems including point to point (P2P) and bus based [6]. Both these architectures were quite useful until we had only few processors connected through them but as the number of processing elements (P.E) increased several issues were spotted and P2P has limited scalability due to increased complexity [7] while, in buses, scalability is restrained due to the fact that by raising the number of processing elements the resulting propagation delay increases [8]. Bus based architecture can be implemented conveniently if number of processors are less than five [9].

To overcome the deficiencies of the above mentioned interconnect structures network on chip (NoC) communication architecture has been adopted. This approach has the tendency to fulfil high performance requirements and it provides the designers with several options to optimize their design according to their requirement [10].

Multiple processors system on chips (MPSoCs) connected through NoC have the ability to perform communication and computation separately.

MPSoCs featuring NoC can be used to perform various complicated tasks in parallel such as multimedia streaming, telecommunication protocols, and GPS [11]. A lot of effort has been put into the improvement of NoC's components.

A flexible and fault tolerant network interface for NoC have been developed by Saponara et al. which can accomplish various functions including safety and security [12]. Moreover, fast and redundant communication protocols like FF-LYNX are another achievement in the field of NoC based designs for implementation in harsh environments, for example, in astrophysics, nuclear medicines, and space engineering. Interfaces based on these protocols are tested and verified under different circumstances before adoption [13, 14].

The market share of embedded systems in different industries is increasing significantly over the past few years. MPSoCs based high performance embedded systems are likely to replace these current embedded systems in future. For instance, the share in the value of car contributed by electronic embedded systems is 40% [15]. And it is believed that these traditional electronics embedded systems in cars would soon be completely substituted by MPSoCs [16].

In the last few decades, there has been a need for high performance and improvements. Technology scaling is not very effective in addressing high performance demands. So, nowadays MPSoCs are being employed to perform tasks which require intense computation and communication.

Our long term objective is to design a heterogeneous MPSoC, for which the design objectives are already set in [17]. In the current work we are analysing different fault tolerance techniques on the basis of hardware overhead, performance, and reliability to conclude the optimal scheme.

The other sections of paper are summarized as follows. Section 2 presents the related work; methodology is discussed in Section 3. Experimental setup is demonstrated in Section 4, while experimental results and conclusions are discussed in Section 5 and Section 6, respectively.

2. Related Work

Various techniques have been developed to make the system fault tolerant. However, in general, there are three major types of redundancies: hardware redundancy, information redundancy, and temporal redundancy. All these techniques have their own characteristics in terms of defined design objective. Our concern is to make the system reliable with minimum cost and it should have minimal impact on performance; therefore a suitable technique is required to meet our design constraints.

Dual modular redundancy (DMR) and triple modular redundancy (TMR) are two well-known hardware redundancy techniques. TMR is a fault masking technique that has 3.7 times hardware overhead [18] and is intended to be used in highly critical scenarios, for example, nuclear plants, space applications, and avionics [19–21]. DMR has low area overhead as compared to TMR when employed with SW based rollback mechanism [22]. DMR along with rollback can be used to perform error correction. Ziv and Bruck employed check pointing technique with DMR for error correction and determined the time consumed by a task under different fault models [23].

Information redundancy which is implemented by sending extra information with data uses different error correcting and error detecting schemes. Error correcting codes have less hardware overhead as compared to hardware redundant techniques; however the latter is considered more reliable [24]. Parity check code is one of the simplest error detecting codes that can be used to detect single bit error and odd number of bit flips [25]. Its advantage is that it can be implemented with low hardware overhead [26]. Nicolaidis employed parity prediction and double rail checking together to devise a self-checking mechanism for ALUs and adders. Furthermore, different hardware reduction techniques were used to minimize the cost [27].

Then comes temporal redundancy which requires reexecution of the same task after certain amount of time; thus there is prolonged fault recovery process as compared to hardware redundancy [28]. A system with such redundancy is only capable of detecting errors caused by transient faults while errors due to permanent faults are not traceable. However, fault tolerance can be achieved with small hardware overhead [29]. Mizan et al. purposed SITR (self-imposed temporal redundancy) for pipelined functional units (FUs); in this form of temporal redundancy there is consecutive execution of redundant tasks with reexecution and comparison being done in the same cycle. The results show that area and power overhead of SITR are less than conventional temporal redundancy [30].

In [17], two different self-checking mechanisms were utilized in MPSoC. They employed DMR in some processors and ECC coding with rollback in others to obtain results in terms of area and error coverage.

It has been observed that not much work has been done when it comes to determining a redundant scheme which provides low hardware overhead and performance at the same time. Considering this observation, this paper is intended to identify a self-checking mechanism which addresses these constraints simultaneously.

3. Methodology

The methodology adopted to practice the defined approach is as follows. We are comparing three different redundancy schemes on the basis of the results acquired by inducing these schemes in a NoC based homogeneous MPSoC. The first scheme is DMR, a hardware redundant technique in which two processes are executed concurrently and outputs are compared using comparator to check if fault has occurred or not. This technique requires additional hardware and it is meant to perform critical tasks. The second technique employed is temporal redundancy (TR) in which two similar processes are executed consecutively and then compared for fault detection. This technique has least hardware overhead but it cannot be employed in critical scenarios because of time overhead. Parity code (ECC) is the third technique, it compares the input and the generated output to detect fault. It appears to have intermediate hardware as compared to previously mentioned schemes.

The basic working and timing diagrams of DMR, TR, and ECC are illustrated in Figure 1.

All of the above techniques would be utilizing roll back mechanism to perform correction, that is, reexecution of task in case of fault.

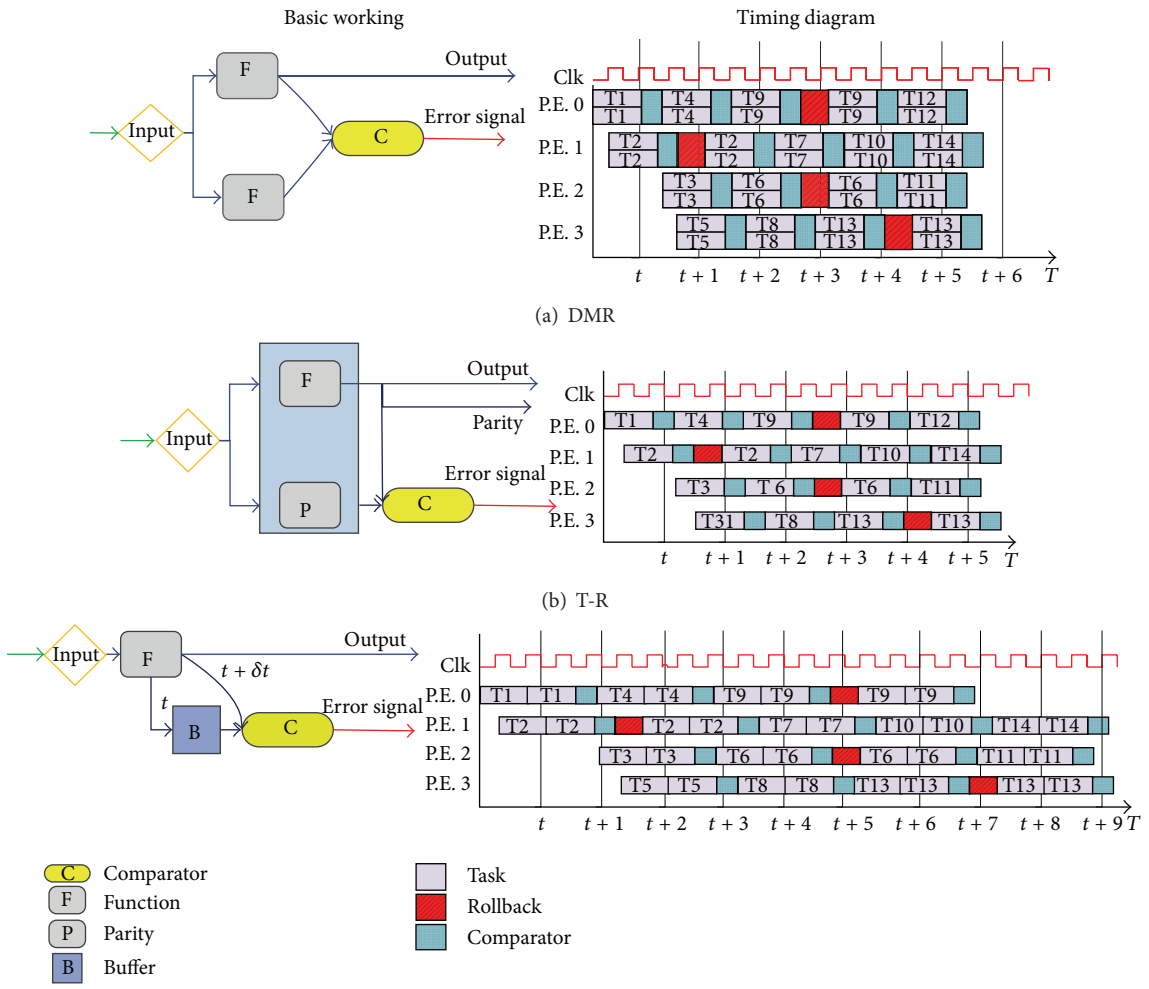

Figure 1: Basic working and timing diagram of (a) DMR, (b) TR, and (c) ECC.

4. Hypothesis

To reduce problem size in proposed application we have assumed that only transient faults can occur during computation while communication channels are fault free. The bandwidth of each channel is "1." Moreover, computations are performed through static scheduling and static mapping.

We are considering faults due to single event upset (SEU) and multiple bit upset (MBU), which are injected through bit flip.

5. Experimental Setup

Experimental setup is developed in VHDL at register transfer level (RTL) and implemented on FPGA. There is NoC based MPSoC in which we have incorporated communication intensive task graph consisting of 1200 small tasks and a computation intensive task graph which consists of 700 larger tasks. To identify working of the system "application model" is required. In application model as described earlier we considered static mapping and static scheduling. System level design is shown in Figure 2.

Time considered for each task is its worst case execution time (WCET). The failure of task is considered if it violates the overall deadline including slack time.

Since all processors are FT we considered that the errors are detected using maximum of 2 execution clocks; thereafter data available at output is fault-free. Otherwise, it will be updated as a nondetected fault in the log file.

Our interest is also to analyse impact of workload and provide fast method for execution of tasks.

An experimental setup is shown in Figure 3 in which fault generation circuitry injects fault randomly during execution of task in any P.E. If fault is detected, then rollback is called and task will be reexecuted. Reexecution has to be done within slack time. As described in previous section we are supposed to compare three FT techniques. These techniques are introduced to the P.Es and application is run to generate output results. We will implement three different error detecting methodologies:

(i) parity based ECC coding (ECC);

(ii) temporal redundancy (TR);

(iii) dual modular redundancy (DMR).

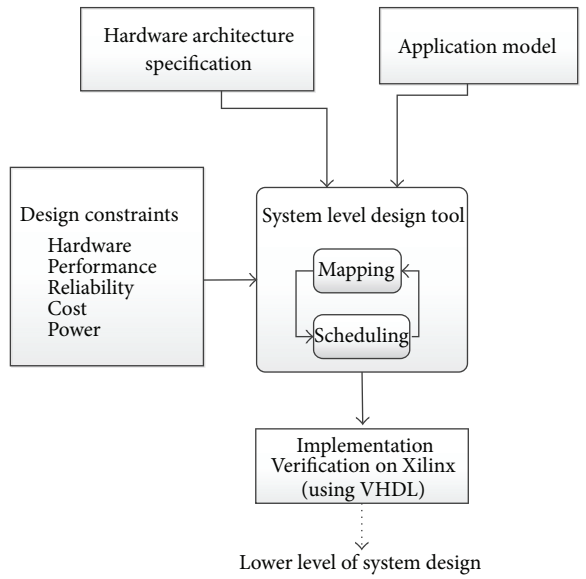

Figure 2: System level design.

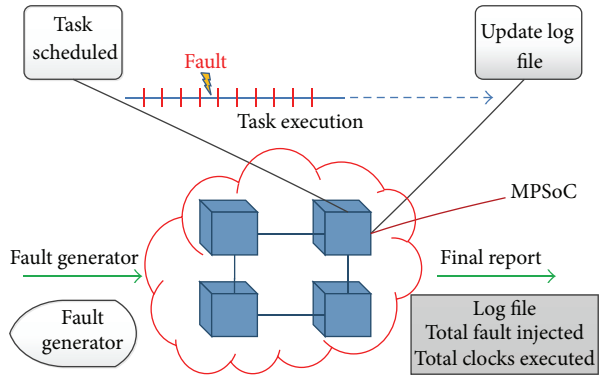

Figure 3: Experimental setup.

6. Benchmark and Application

We have designed NoC based homogenous MPSoC for real time applications to be run on the system. Automobile electronic control unit (ECU) is the considered application. ECU is divided into two basic parts: (1) engine management and control system and (2) transmission control system.

Two different types of benchmarks are designed which consist of set of general tasks in target application. The first benchmark has communication intensive tasks and second has computation intensive tasks.

(i) Benchmarks 1: in communication intensive benchmark, transmission control of ECU is considered that requires more communication and less computation.

(ii) Benchmarks 2: computation intensive benchmark considers engine management and control of ECU which is meant to perform more calculations as compared to communication.

7. Results

Overall results are divided into three categories: area overhead, error coverage, and performance and power consumption.

7.1. Hardware Analysis. All three FT mechanisms are modelled in VHDL, implemented on XILINX, and RTL file is generated to determine the hardware overhead of each mechanism.

Table 1 depicts that among all the mechanisms DMR has the highest overhead, T-MR has moderate and parity check has least hardware overhead.

7.2. Error Coverage. The error detection is monitored at different error injection rates to determine the error coverage of each scheme. The injected errors are detected by the voter while generating the output.

For instance, Figure 4 demonstrates the fault detection and correction in DMR; marker "A" depicts that values are

Table 1: Results shown by RTL file.

Selected device: Xilinx Spartan-3S200 (ft256-4)					
Device	Total	Normal	Used (% age)		
			T-R	D-MR	ECC
Slices	1920	169 (8%)	354 (18%)	435 (22%)	272 (14%)
Slice flip flops	3824	122 (3%)	292 (7%)	337 (8%)	213 (5%)
4-input LUTs	3840	311 (8%)	643 (16%)	799 (20%)	483 (12%)
IOs		13	13	13	13
Bonded IOBs	173	12 (6%)	12 (6%)	12 (6%)	12 (6%)
IOB flip flops		8	8	8	8
GCLKs	8	2	2 (25%)	2 (25%)	2 (25%)

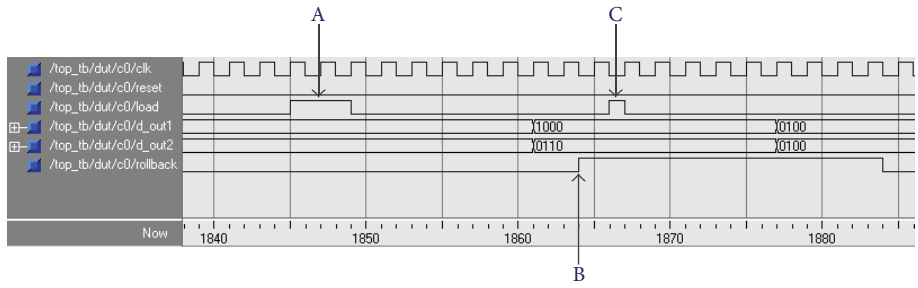

Figure 4: Fault detection and correction in DMR.

assigned to specified processor and after processing the data outputs (d_out1, d_out2) are compared; marker B suggests that mismatch has occurred at outputs and roll back is called. Thereafter similar task is reexecuted and marker C indicates reexecution, hence, making the system fault tolerant.

Four different simulated fault injection (based on bit flip) techniques are conducted. In which, campaigns from 1-to-4-bit flips are employed. The results in Figures 5, 6, 7, and 8 depict error coverage of proposed FT techniques. Error coverage is 100% when odd numbers of bits were flipped. When even numbers of bits were flipped, DMR and TR have 100% error coverage but it reduces to 52% for parity check.

7.3. Performance Analysis. Program is initiated with no fault condition to get the minimum execution time for performing specified task. Afterwards the number of faults per clock is increased and time for each execution is noted.

Performance is affected as more faults hit processor. As fault injection rate increases more rollbacks are called. Hence more time is taken by the processor to generate fault tolerant output.

As application has two types of tasks, communication intensive and computation intensive, implementation results comply with our design constraints.

Application is time-critical, we can assume a reasonable slack time; excess time to this will be considered as time-failure of application. In Figures 9 and 10, horizontal continuous line shows normal execution time and horizontal dotted line shows maximum acceptable execution time. If 0.35 of slack time is allowed, then DMR has highest performance. Parity code shows a great difference in performance as compared to TR.

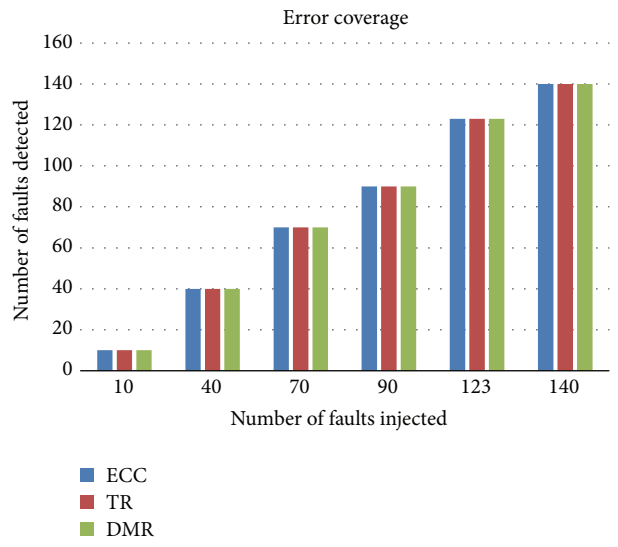

Figure 5: 1-bit injection and detection.

Figures 9 and 10 depict that for performance ECC and TR can be regarded as suitable techniques for our application.

Table demonstrate finding of our experimental setup with performance being evaluated on the basis of number of clocks and hardware on the basis of LUT. And it is close to the predicted values.

7.4. Power Consumption. Now-a-days power utilization is very important parameter, especially for modern multicore

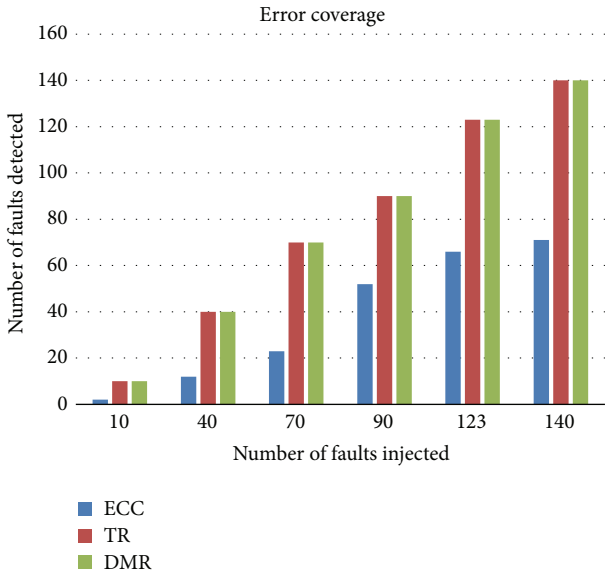

Figure 6: 2-bit injection and detection.

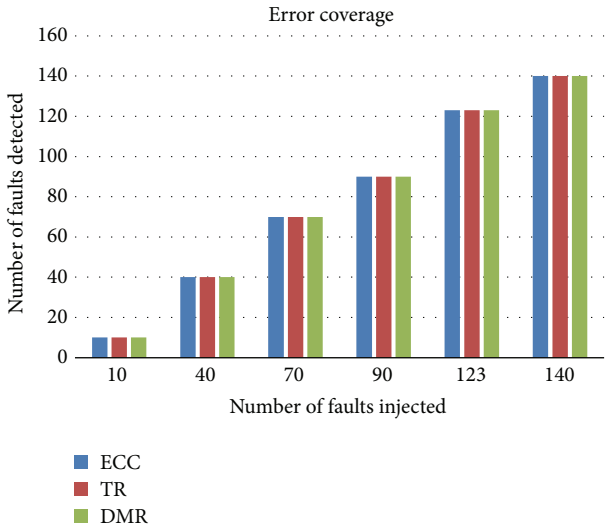

Figure 7: 3-bit injection and detection.

battery driven applications. Designing the on-chip interconnects with battery considerations and reduced power consumption is crucial. The ever increasing demand of the battery operated devices has driven the research in the field of low power system design. Saved power can be used to increase battery life cycle and reduce cooling necessities. The focus of the previous researches was on the high level analysis of NoC but they did not touch the issues on a circuit level. However, little research has been done on the circuit level issues as they were focusing on implementing a limited set of topologies. For achieving the cost efficiency, high performance, and computational speed, it is necessary to achieve energy optimization.

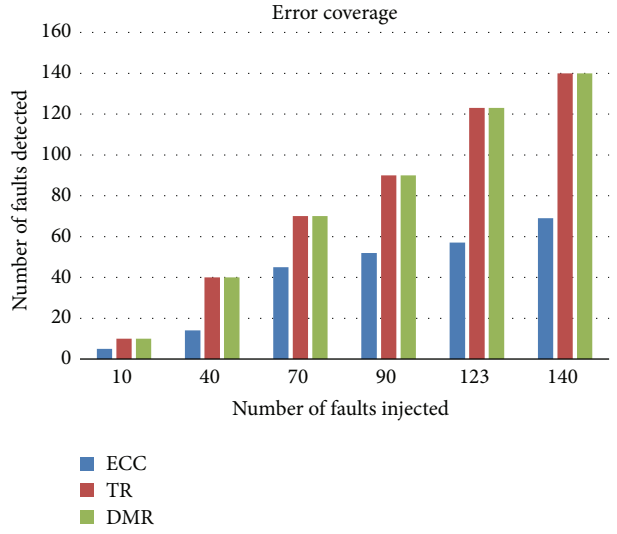

Figure 8: 4-bit injection and detection.

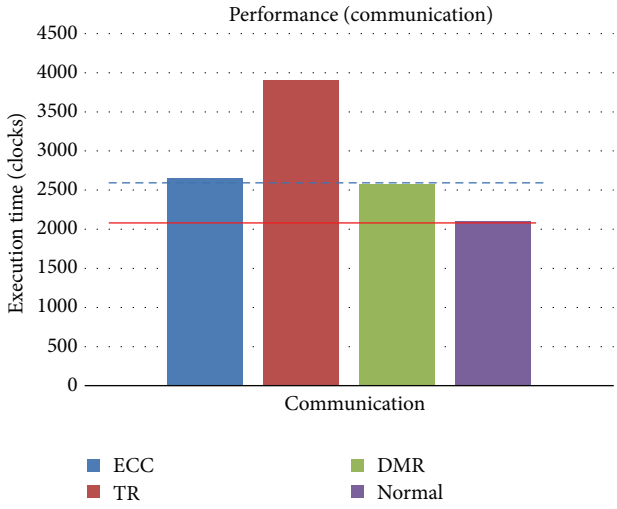

Figure 9: Communication intensive.

To estimate power consumption of specified redundant techniques, power report is generated using Xilinx ISE, XPower Analyzer (XPA) tool along with Modelsim simulator.

XPA calculates static and dynamic power using (1) and break the power into logic, signal, and output components [31]:

$$P = CV^2 f \times E, \tag{1}$$

where frequency is kept constant; that is, $f = 500$ MHz.

Results in Table 2 are extracted from XPower Analyser. The table enlists the power utilized by the corresponding components of each scheme. The total power stated is comprised of both static and dynamic power.

From the results in table it can be observed that power consumption has a direct relation with the hardware. As perceived from the results, DMR has highest and TR has moderate while ECC has least power consumption

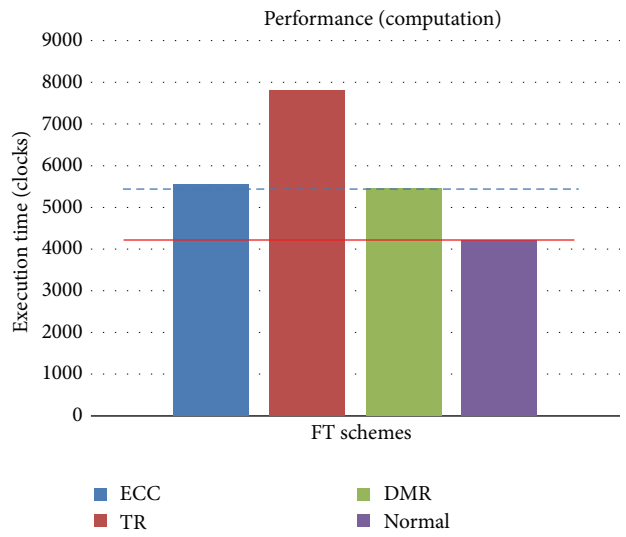

Figure 10: Computation intensive.

Table 2: Results shown by XPower Analyser.

On-chip	Power (mW)		
	DMR	TR	ECC
Clocks	81.59	75.74	40.73
Logic	1.22	1.21	0.00
Signals	2.35	2.98	0.00
IOs	57.76	53.78	50.82
Quiescent	41.84	41.79	41.53
Total	**184.77**	**175.50**	**133.08**

8. Conclusion

Our long term goal is to design a low-cost fault tolerant MPSoC and in this paper we analyzed three different FT methodologies and introduced them in NoC based homogeneous MPSoC. Thereafter, we evaluated these schemes by simulating their corresponding architectures under given set of benchmarks. For each scheme, results are obtained in terms of area overhead, error coverage, performance and power consumption.

According to the defined design objectives and benchmarks, the result of analysis shows that the parity code is most suitable technique because it has least hardware overhead along with reasonably high performance. DMR can be employed if higher performance and error coverage is required simultaneously; however it comes at the cost of increased hardware. Temporal redundancy can only be used in circumstances where we do not have hard real-time constraints.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgment

The authors wish to thank Dr. Abdul Waheed Malik for his useful suggestions to improve the paper.

References

[1] M. Bohr, "The new era of scaling in an soc world," in *Proceedings of the IEEE International Solid-State Circuits Conference (ISSCC '09)*, pp. 23–28, San Francisco, Calif, USA, February 2009.

[2] R. Vemu, *Low-Cost Assertion-Based Fault Tolerance in Hardware and Software*, UMI, 2008.

[3] M. Amin, A. Ramazani, F. Monteiro, C. Diou, and A. Dandache, "A self-checking hardware journal for a fault-tolerant processor architecture," *International Journal of Reconfigurable Computing*, vol. 2011, Article ID 962062, 15 pages, 2011.

[4] M. Nicolaidis, Ed., *Soft Errors in Modern Electronic Systems*, Springer, 2011.

[5] P. Axer, M. Sebastian, and R. Ernst, "Reliability analysis for MPSoCs with mixed-critical, hard real-time constraints," in *Proceedings of the 7th IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS '11)*, pp. 149–158, October 2011.

[6] L. Torres, P. Benoit, G. Sassatelli, M. Robert, F. Clermidy, and D. Puschini, "An introduction to multi-core system on chip—trends and challenges," *Multiprocessor System-on-Chip: Hardware Design and Tool Integration*, pp. 1–21, 2011.

[7] H. G. Lee, N. Chang, U. Y. Ogras, and R. Marculescu, "On-chip communication architecture exploration: a quantitative evaluation of point-to-point, bus, and network-on-chip approaches," *ACM Transactions on Design Automation of Electronic Systems*, vol. 12, no. 3, Article ID 1255460, 2007.

[8] C. Grecu, P. P. Pande, A. Ivanov, and R. Saleh, "Structured interconnect architecture: a solution for the non-scalability of bus-based SoCs," in *Proceedings of the ACM Great Lakes Symposium on VLSI (GLSVLSI '04)*, pp. 192–195, April 2004.

[9] L. Benini and G. de Micheli, "Networks on chips: a new SoC paradigm," *Computer*, vol. 35, no. 1, pp. 70–78, 2002.

[10] R. U. S. U. Claudia, *Multi-Level Fault-Tolerance in Networks-on-Chip*, 2010.

[11] M. Mandelli, L. Ost, E. Carara et al., "Energy-aware dynamic task mapping for NoC-based MPSoCs," in *Proceedings of the IEEE International Symposium of Circuits and Systems (ISCAS '11)*, pp. 1676–1679, May 2011.

[12] S. Saponara, T. Bacchillone, E. Petri, L. Fanucci, R. Locatelli, and M. Coppola, "Design of an NoC interface macrocell with hardware support of advanced networking functionalities," *IEEE Transactions on Computers*, vol. 63, no. 3, pp. 609–621, 2014.

[13] G. Magazzù, G. Borgese, N. Costantino, L. Fanucci, J. Incandela, and S. Saponara, "Design exploration and verification platform, based on high-level modeling and FPGA prototyping, for fast and flexible digital communication in physics experiments," *Journal of Instrumentation*, vol. 8, no. 2, Article ID P02021, 2013.

[14] N. Costantino, G. Borgese, S. Saponara, L. Fanucci, J. Incandela, and G. Magazzu, "Development, design and characterization of a novel protocol and interfaces for the control and readout of front-end electronics in high energy physics experiments," *IEEE Transactions on Nuclear Science*, vol. 60, no. 1, pp. 352–364, 2013.

[15] G. Scuro, *Automotive Industry: Innovation Driven by Electronics*, 2013, http://embedded-computing.com/articles/automotive-industry-innovation-driven-electronics/.

[16] S. Aust and H. Richter, "Real-time Processor Interconnection Network for FPGA-based Multiprocessor System-on-Chip (MPSoC)," in *Proceedings of the 4th International Conference on Advanced Engineering Computing and Applications in Sciences*, pp. 47–52, October 2010.

[17] M. Amin, M. Tagel, G. Jervan, and T. Hollstein, "Design methodology for fault-tolerant heterogeneous MPSoC under real-time constraints," in *Proceedings of the 7th International Workshop on Reconfigurable and Communication-Centric Systems-on-Chip (ReCoSoC '12)*, pp. 1–5, July 2012.

[18] Dagan White, "Managing single event effects in FPGAs, ASICs and processors (Part 2)," 2011.

[19] N. Desai, "The Tricon Turbine Control System," 2010.

[20] R. Yuan, "Triple Modular Redundancy (TMR) in a configurable fault tolerant processor (CFTP) for space applications," December 2003.

[21] Y. C. Yeh, "Triple-triple redundant 777 primary flight computer," in *Proceedings of the IEEE Aerospace Applications Conference*, vol. 1, pp. 293–307, February 1996.

[22] M. Rebaudengo, M. S. Reorda, and M. Violante, "A new approach to software-implemented fault tolerance," *Journal of Electronic Testing*, vol. 20, no. 4, pp. 433–437, 2004.

[23] A. Ziv and J. Bruck, "Analysis of checkpointing schemes for multiprocessor systems," in *Proceedings of the 13th Symposium on Reliable Distributed Systems*, pp. 52–61, October 1994.

[24] S. Mukherjee, *Architecture Design for Soft Errors*, Elsevier, 2008.

[25] G. Bertoni, L. Breveglieri, I. Koren, P. Maistri, and V. Piuri, "A parity code based fault detection for an implementation of the advanced encryption standard," in *Proceedings of the 17th IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems (DFT '02)*, pp. 51–59, 2002.

[26] G. Di Natale, M. Doulcier, M. L. Flottes, and B. Rouzeyre, "A reliable architecture for parallel implementations of the advanced encryption standard," *Journal of Electronic Testing: Theory and Applications (JETTA)*, vol. 25, no. 4-5, pp. 269–278, 2009.

[27] M. Nicolaidis, "Carry checking/parity prediction adders and ALUs," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 11, no. 1, pp. 121–128, 2003.

[28] A. K. Somani and N. H. Vaidya, "Understanding fault tolerance and reliability," *Computer*, vol. 30, no. 4, pp. 45–50, 1997.

[29] P. Agrawal, "Fault tolerance in multiprocessor systems without dedicated redundancy," *IEEE Transactions on Computers*, vol. 6, no. 2, pp. 358–362, 1988.

[30] E. Mizan, T. Amimeur, and M. F. Jacome, "Self-imposed temporal redundancy: an efficient technique to enhance the reliability of pipelined functional units," in *Proceedings of the 19th International Symposium on Computer Architecture and High Performance Computing*, pp. 45–53, October 2007.

[31] M. Glesner, P. Zipf, and M. Renovell, *Field-Programmable Logic and Applications: Reconfigurable Computing is Going Mainstream*, vol. 2438 of *Lecture Notes in Computer Science*, Springer, Berlin, Germany, 2002.