Realization of Dependable Digital Systems for Safety-Critical Computer Systems using FPGAs

F N kassab bashi1, S S Khairullah1* and C R Elks2

1College of Engineering, Computer Engineering Department, University of Mosul, Mosul, Iraq.
2College of Engineering, Virginia Commonwealth University, Richmond, USA.

*E-mail: shawkat.sabah@uomosul.edu.iq

Abstract. Digital systems built based on field-programmable-gate-arrays (FPGAs) are increasingly being used in safety-critical computer systems. These systems typically require high levels of integrity, reliability, and performance to ensure the correct operation of the critical application in which the digital systems are embedded. To achieve the safety and functional objectives, many hardware architectural scenarios based on fault tolerance techniques were proposed. This research paper presents a new dependable digital system (DDS) that consists of fault tolerant blocks (FTBs), spare functional blocks (SFBs), synchronization units, and input output units. Markov-based chain have been used to model the correct operation of the proposed hardware system that is targeted to be realized on FPGA-based technology. This system was designed to be reliable against transient faults (TF), permanent faults (PF), and hardware common cause failures (CCF). The reliability analysis of the proposed system was accomplished with Markov modeling techniques which could express the regenerative behavior of the digital system. Certain states in the system represent system failure, while others represent fault-free behavior in the presence of faults. Finally, based on the MathWorks Simulink simulation results of modeling the DDS digital system, the system does meet its functionality and reliability requirements.

1. Introduction

Field-Programmable Gate Arrays (FPGAs) are integrated circuits (ICs) that are capable of implementing custom digital designs based on the interaction between hardware and software components that could be storage elements, reconfigurable computing blocks, switch boxes, multiplexers and decoders. These embedded elements have the ability of being modified and reconfigured at runtime to execute the task of a new application [1]. In fact, many studies have indicated that FPGA’s is a viable technology for safety-critical applications such as space and avionic control systems, industrial automated systems, and nuclear power industry. The key characteristics of this technology that are beneficial for safety critical systems are; custom configurability, partitioning of functionality, true concurrency, and high performance [3].

In contemporary safety related digital systems, that use the Static random-access memory (SRAM) technology the progressive reduction in device feature size and increase in clock rates are making them more susceptible to the effects of ionizing radiation, electromagnetic interactions, and high temperatures. Safety-critical digital systems are expected to operate correctly in the presence faults, diverse failure modes that stem from external threats that could be error, fault, or attack. These faults can cause the system to generate wrong data values and go into an unsafe state in which the error must
be detected and tolerated. These faults are often classified as a single transient fault, a single permanent fault, or multiple sequential permanent faults. Transient-faults are temporary changes in the data value and can be caused by radiations. For example, Single-Event-Upsets (SEUs) are the change of state in the contents of a memory cell from a logic '1' to logical '0', or from a logic '0' to a '1'. On the other hand, Multiple-Bit-Upsets (MBUs) are examples of multiple faults affecting on the system concurrently due to the continual shrinking of the dimensions in the SRAM-based transistor [13]. Furthermore, permanent faults or hard-errors caused by system wear-out can cause changing the contents of the data permanently [2]. In recent years, numerous research works have been developed to solve the issue of soft-errors and calculate its reliability [9] [10] [11], and other works have tried to tolerate the effects of both soft-errors and hard-errors in the FPGA-based digital technology [4] [6] [14]. However, proposing a comprehensive dependable digital system that is hierarchal, deterministic, and scalable is still essential in the research field. In this paper, we aim to design a deterministic fault-tolerant digital system (DDS) that is reliable against different failure modes that could be transient faults (TFs), permanent faults (PFs), or hardware common cause failures (CCFs). MathWorks Simulink tool was used in modeling the hardware components of the proposed system and verifying the functional and safety requirements. Finally, the dependability and reliability analysis of the system was achieved by using Markov-based chains and injecting transient faults into the functional blocks was performed at different levels of the proposed system.

2. Related works
Several studies on FPGA design and fault tolerance have recently been published. However, the manufacturers are still looking for further reliable chips that are low in cost (delay, complexity, hardware, area, and so on) and have a high yield elaboration of safety-critical applications contra various types of faults like transient- faults, intermittent-faults, and permanent-faults. On the other hand, from the user's perspective, fault tolerance becomes a real requirement, especially for certain applications where the time needed for the repair process (downtime) has a critical impact on the application [5]. A brief presentation of hardware fault tolerance methods is discussed in this section.

A new self-healing hardware architecture inspired by the concepts of biology was proposed in [1] and [3], these architectures merge between cellular-based biological concepts, operational schematics of international standard (IEC 61131-3), and traditional fault-tolerance techniques. In [1] Three lines of defence contra discovered faults was suggested by Khairullah; the first one is a hybrid unit, which is responsible for tolerating the discovered transient faults. The second one is a local healing layer, which is responsible for tolerating permanent faults and the third one is a global healing layer, which is responsible for tolerating additional permanent faults, and fault management of the critical service layer. In [3] Khairullah and et.al proposed an architecture that marshaled in two levels; the first one is a critical functions layer, which is used to provide the meant service of the implementation. And the second one is a healing layer, which regularly manages the application's proper execution and generates health syndromes in the case of failure appears in the function layer.

A horse shifting allocation method was proposed in [5], in which the apportionment of the (spare CLBs) in 2-dimensional arrays was introduced, allowing the SRAM part of the FPGA to be modified and have the ability to shift in on-chip data horizontally and vertically. When trouble is detected, it can be easily avoided by transferring user data to a nearby spare instead of the defective CLB, causing the defective CLB to become unusable. In this method, each CLB block size 3 X 3 has a maximum of two CLBs.

Miculka et al. (2013) [4], have present a fault-tolerant method in FPGA that have the ability to mitigate transient fault and sundry permanent fault that occur sequentially. To mitigate transient fault, they use partial-dynamic-reconfiguration (PDR) and for permanent faults mitigation, they use a specific FT architecture that occupies fewer resources than the already used and excludes the FPGAs faulty part. In this technique, they based on the stored precompiled configurations in external memory and they used the relocation technique to reduce the space that won't for a partial bit stream.

A Master-Slave TMR inspired technique was proposed in [6] for fault tolerance in SRAM-based FPGA, single and double faults affecting configurable logic blocks (CLBs) are being taken up. This
method depends on triple-modular redundancy (TMR) in conjunction with the master-slave technique, as well as partial reconfiguration for permanent fault tolerance. In the suggested MSU architecture, there are two kinds of CLBs, CLB-Master, and CLB-Slave. If a fault appears in CLB-Slave, a partial reconfiguration on the MSU will exchange the defective CLB-S with one of the three CLB-M, and then defective CLB-S is ignored.

An adaptable and resilient hardware architecture was proposed in [7], this architecture was designed by a classical active redundancy model, TMR with spares in SRAM-based FPGA. In this method the replacement of a triple-redundant model failed modules with spare modules will occur, a developed controller in the proposed architecture gives them a possibility to adapt between Triple-Modular-redundancy with one spare, or Triple-Modular-redundancy with two spares.

3. Overview of the proposed fault-tolerant digital system
The presented digital system below is designed to achieve high levels of reliability with respect to different fault models such as transient, permanent, and common cause hardware faults. Another objective of the design of this system is to be scalable, efficient, and deterministic. In the proposed Dependable Digital System (DDS) shown in Figure 1, four levels of the organization are discussed. Level (0) contains the function block (FB) operations, which are capable of executing four arithmetic operations, seven logic operations, and two shift operations. In level (1) a Triple-Modular-Redundancy (TMR) is embedded, which is responsible for tolerating transient faults. Furthermore, levels (2) and (3) use the concept of Reallocation Manager (RM) and take advantage of using duplication with compare (DWC) circuit to detect and correct faulty function blocks. Two spare function blocks (SFBs) were used to replace the faulty functional blocks effected by two serial permanent faults or two simultaneous common cause faults (CCF).

Moreover, in order to make easier to program from a programmability point of view, we adopted a Function Block Programming model similar to the international standard-“IEC 61131-3 Programmable Logic Controllers”.

![Figure 1. The proposed fault-tolerant digital system](image-url)

Figure 1 illustrates the basic Functional components of our fault tolerance digital system. These are the I/O Terminal, I/O Module, Fault Tolerance Blocks (FTBs), Two Spare Function Blocks (SFBs), Sequential and Combinational Logic, Synchronization unit, and Reallocation Manager (RM). The valid execution of each function block (FB) is observed using two voting circuits and one DWC circuit comprised of XOR gates (See Figure 2(a)). Once a fault test is performed by detecting transient faults in the functional blocks and is observed in the input register of any FB, these faults will be tolerated by Triple-Modular-Redundancy (TMR) embedded in the same FB. Furthermore, if a detection of
permanent faults is observed by passive FT method; duplication with a comparison in one or two FBs, at different times or even at the same time, the Reallocation Manager (RM) (See Figure 2(b)), is responsible for ignoring this FB and establish a new connection with the new spare (SFB) as it is described in the flowchart in Figure 2(b). The RM takes the address of fault FB and makes rerouting to reach the input of spare FTFB. This strategy requires eight clock cycles to generate the resulted output date through the spare FB.

**Figure 2(a).** Scheme of PF detection and correction

**Figure 2(b).** Flow chart of PF and CCF

In our proposed architecture, we have library of functional blocks organized in a two dimensional-array that consists of 25 Fault Tolerance Function Block (FTFB) as is shown in Figure 1, highlighted with yellow colour. Every five FTFB are connected in parallel as one container in such a way that a group of containers will serially be connected. The internal structure of each FTB consists of four stages which are input, operation, voting, and output. These four stages are controlled by a state
machine-based control flow. Consequently, we need 4 clocks to activate one FTFB and compute the resulted data output as shown in Figure 3.

![Figure 3. Block diagram of state machine-based control flow](image)

Figure 3. Block diagram of state machine-based control flow

Figure 4 (a) shows a block diagram of one row or container of the proposed DDS digital system that contains different hardware components like AND gates, inverters, multiplexers, OR gates, XOR gates, FTFB, and SPARE units. One container is capable of detecting an attack of two sequential common cause faults (CCFs) that can occur in one of the following pairs of functional blocks: (FTFB1, FTFB2), (FTFB1, FTFB3), (FTFB1, FTFB4), (FTFB1, FTFB5), (FTFB2, FTFB3), (FTFB2, FTFB4), (FTFB2, FTFB5), (FTFB3, FTFB4), (FTFB3, FTFB5), (FTFB4, FTFB5) at one time. This failure mode is repaired based on two spare functional blocks SPARE1 and SPARE2. Figure 4 (b) shows the hardware implementation for Figure 4 (a) using MathWorks Simulink Software design tool.

![Figure 4(a). Block diagram of CCF detection and correction](image)

Figure 4(a). Block diagram of CCF detection and correction
Figure 4(b). Hardware implementation of CCF detection and correction

Referring to Figure 4(a), (b), if the operation of DDS flow successfully without any fault detection, 4 clock cycles are sufficient to activate every container, and $4\times5=20$ clock cycles are sufficient to activate all the containers as it is described in Table 1. In another word, in the $1^{st}$ clock, the input of the five FTFBs in the first container will be activated, in the $2^{nd}$ clock, a special operation is executed depending on the selected value of the multiplexer input lines, in the $3^{rd}$ clock, a majority voter will choose the correct output that will be flow to the output port, and finally, in the $4^{th}$ clock, DWC will test the accuracy of permanent fault or common cause fault, if there is no fault the output will flow normally.

In one scenario we suppose that we have an effect in the fifth FTFB of the second container and the fifth FTFB in the fourth container, so in the first four clocks the operation will flow normally, so we consume five outputs and no delay appear. In the $5^{th}$ clock, the input of the five FBs in the second container will enter, in the $8^{th}$ clock the device will test the accuracy of permanent fault (PF) or common cost fault (CCF), the PF or CCF appearance in the fifth FTFB will cause to enter the system in a wait state by the usage of synchronization unit. At the same time, a specific signal will activate the Reallocation Manager, the RM will change the routing of the system, by ignoring the faulty FB and activate the spare one, by connecting the inputs of faulty FTFB to the Spare FTFB, and the output of spare FTFB instead of the output of the faulty FTFB. This scenario will lead to recovering the proposed faulty component and continue to perform its task despite error occurrence at the expense of systems delay which will increase depending on wait time (we suppose 4 clock cycles) to complete the FB exchange.

In the $13^{th}$ clock cycle, if no fault is detected, the data input of all five FTFBs in the third container will be measured concurrently, so the output for each FTFB will flow and generate data normally at the end of the $16^{th}$ clock. In the $20^{th}$ clock, another detection of Pf will appear in the fifth FTFB of the fourth container, so the same scenario of container two will repeat and the output will flow in the $24^{th}$ clock. In the $25^{th}$ clock, the input of the five FTFBs in the fifth container will enter, no fault is detected so the output will flow normally in the $28^{th}$ clock. If we compare the delay of PF tolerance with CCF tolerance, we can see that CCF (for two parallel FTFB) detection and correction will take four clocks less than PF detection and correction, that because the latest case appears serially.

Table 1. Number of clocks and their actions

| No. Clock | FB active | Container No. | No. of I/P data | No. of memories | No. control data | No. of O/P data |
|-----------|-----------|---------------|----------------|----------------|-----------------|----------------|
| 4         | FB11      | 1             | 20             | 20             | 5               | 5              |
|           | FB12      | 1             |                |                |                 |                |
|           | FB13      | 1             |                |                |                 |                |
|           | FB14      | 1             |                |                |                 |                |
|           | FB15      | 1             |                |                |                 |                |
As it can be seen in Figure 5, we assume that one operation needs to obtain the value of the following Boolean logic expression \(((\text{NOT } 7) \text{ AND } 12) \text{ OR } (7 \text{ AND } 7)\). So firstly we must obtain the value of \(\text{NOT } 7\), then the result must AND with 12, this case will take 8 clocks because the two operations are happening serially. On the other side, to obtain the result of \(7 \text{ AND } 7\) will take 4 clocks to appear, we will need another 4 clocks to obtain the final value, so the total number of clocks will be (12 clocks). If a fault appears in one of these operations (NOT, AND1, AND2, and OR), we will need additional 4 clocks to replace the faulty FB with the spare one, ‘Figure 5’ shows that we have a fault occurrence appears at AND1 operation.

![Figure 5. Timing diagram of fault detection in AND gate FTFB](image)

Figure 5. Timing diagram of fault detection in AND gate FTFB
4. Markov Mathematical Analysis

Combinatorial mathematics approaches are used to analyze the reliability of safety-critical applications. Hazard Tree Analysis, Reliability Block Diagrams, Fault Tree Methods, and Probabilistic Risk Analysis are some of these methods. These standard reliability analysis approaches are based on stateless and time-invariant mathematics. When encountering systems with regenerative behaviors, repair, and reconfiguration features, it's important to use the more efficient Markov technique to model them [1]. We can define a Markov process that is a stochastic process whose action is solely determined by the system's current state. The order in which the device arrived at its current state has no impact on how it will behave in the future [12].

4.1 Reliability Analysis of the Proposed hardware system

To calculate the proposed hardware DDS model's reliability, a five-state Markov model was created (See Figure 6). As shown in table 2, this model has three operating states, one failing secure state, and one failing insecure state. Each of the states is in one of four situations: completely operational, fail-operational, fail-secure, or fail-insecure.

Table 2. Case and its description of markov modelled system depend on Figure 3.

| Case | Description |
|------|-------------|
| A    | completely operational System (all the FBs are operated properly and two spares are available) |
| B    | First Fail-Operational (one FB is failed by permanent fault and detected by DWC, so one spare is used for repair) |
| D    | Second Fail-Operational (second FB is failed by permanent fault and detected by DWC, so the second spare is used for repair, so no spare remains) |
| FS   | Fail-secure Operational (third FB is failed by permanent fault and detected by DWC, it cannot repair because no spares remain) |
| FI   | Fail-insecure (one, two or three FBs are failed permanently without detection) |

Figure 6. Transition state diagram for the Dependable Digital System (DDS)

The DWC system for (level 2) DDS was modelled using the Markov model that depends on Repair fault operation ($\mu_p$) and Fault coverage ($C_f$). State-A represents a case where all the FBs are operating properly and the system is completely operational. In state-B one FB is failed by permanent fault and detected by DWC, so the system is still operational but one module has failed, the transition from A to B happened at $3\lambda_p$. Furthermore, the system could return to its original state-A at repair rate ($\mu_p$) by...
using the first spare. In state-D, a second FB is failed by permanent fault and detected by DWC and the second spare is used for repair, so no spare remains. The transition from (B) to (D) happened at $2\lambda p$, furthermore; the system could return to its original state-B at repair rate ($\mu p$). In state-FS, a third FB is failed by permanent fault and detected by DWC, but it cannot repair because no spares remain. The transition from (D) to (FS) happened at $\lambda p$. State-FI represents our DDS where one, two, or three FBs are failed permanently without any detection.

The designed system’s Markov model can be written in matrix form as:

$$P_{\text{system}} (t+\Delta t) = T_{\text{system}} * P_{\text{system}} (t)$$

Where;

$T_{\text{system}}$: represents the state transition matrix,

$P_{\text{system}} (t)$: denotes the likelihood of being in the corresponding state at time $t$.

$P_{\text{system}} (t+\Delta t)$: is the possibility about being in the identical state at $(t + \Delta t)$ time.

Furthermore; we expect that every functional cell follows the exponential failure rule and has a fix failure rate $\lambda$ at all times when we calculate the reliability of the Markov model. So, we can compute the system’s probability of failure at a time $(t + \Delta t)$ as follows:

$$P (t + \Delta t) = 1 - e^{-\lambda \Delta t} = \lambda \Delta t$$

$$P_{A} (t + \Delta t) = (1 - 3\lambda p Cf \Delta t) P_{A} (t) + \mu p \Delta t P_{B} (t) + \mu p \Delta t P_{D} (t)$$

$$P_{B} (t + \Delta t) = [1 - (2 \lambda P + \mu p) Cf \Delta t] P_{B} (t)$$

$$P_{D} (t + \Delta t) = [1 - (\lambda P + \mu p) Cf \Delta t] P_{D} (t)$$

$$P_{FS} (t + \Delta t) = \lambda P Cf \Delta t P_{D} (t) + P_{FS} (t)$$

$$P_{FI} (t + \Delta t) = 3\lambda p (1 - Cf) \Delta t) P_{A} (t) + 2 \lambda P (1 - Cf) \Delta t P_{B} (t) + \lambda P (1 - Cf) \Delta t P_{D} (t) + P_{FI} (t)$$

The reliability can be computed as follows:

$$R (t) = 1 - P_{FI} (t) = P_{FS} (t) + P_{D} (t) + P_{B} (t) + P_{A} (t)$$

Where:

$$PA (t + \Delta t) - PA (t) = -3\lambda p Cf \Delta t P_{A} (t) + \mu p P_{B} (t) + \mu p P_{D} (t)$$

$$PB (t + \Delta t) - PB (t) = -2 \lambda P + \mu p \Delta t P_{B} (t)$$

$$PD (t + \Delta t) - PD (t) = -(\lambda P + \mu p) \Delta t P_{D} (t)$$

$$PFS (t + \Delta t) - PFS (t) = \lambda P Cf \Delta t P_{D} (t)$$

$$PFI (t + \Delta t) - PFI (t) = 3\lambda p (1 - Cf) \Delta t P_{A} (t) + 2 \lambda P (1 - Cf) \Delta t P_{B} (t) + \lambda P (1 - Cf) \Delta t P_{D} (t)$$

A Markov model’s two-dimensional state transition matrix would look like this:

$$P_{\text{system}} (t + \Delta t) = \begin{bmatrix}
PA (t + \Delta t) \\
PB (t + \Delta t) \\
P_{FS} (t + \Delta t) \\
P_{FI} (t + \Delta t)
\end{bmatrix}$$
Using Laplace Transform, we obtain the following equations:

S PA (S) - PA (0) = - 3λp Cf PA (S) + μp PB (S) + μp PD (S)
S PB (S) - PB (0) = -(2 λP + μp) Cf PB (S)
S PD (S) - PD (0) = -(λP + μp) Cf PD (S)
S PFS (S) - PFS (0) = λP Cf PD (S)
S PFS(S) - PFI (0) = 3λp (1 - Cf) PA (S) + 2λP (1 - Cf) PB (S) + λP (1 - Cf) PD (S)

We suppose that the system begins in a perfect state at time t=0 in the analysis, so:
PA (0) = 1, PB (0) = 0, PD (0) = 0, PFS (0) = 0, PFI (0) = 0.
So, the Reliability is the probability of being in either State A, State B, State D or FS State R (t) = PA (t) + PB (t) + PD (t) + PFS (t).

5. Conclusion
In this research paper, a novel FPGA based fault tolerant architecture for safety critical systems is proposed and developed. The important aspects of the proposed architecture are it is adaptive resilience to a wide class of faults while maintaining good performance and uninterrupted service. The basic fault detection and tolerance patterns build upon, duplication with comparison, and triple modular redundancy, but extend these concepts with respect to the system level organization of fault tolerance and fault resilience. The proposed architecture system shows significant promise in meeting high levels of fault resilience, according to the simulation results in MathWorks Simulink software when modeling the DDS digital system verified against its specifications. To increase the dependability levels of the proposed system, future works will include design and implementation of more robust digital systems inspired by biological concepts. In addition, MathWorks Simulink Design verifier tool will be used to check the right execution of our proposed design as verification & validation methods (V&V).
6. References

[1] Khairullah S 2018 Toward biologically-inspired self-healing, resilient architectures for digital instrumentation and control systems and embedded devices (Virginia: Virginia Commonwealth University: Theses Diss) pp 1-144.

[2] Di Carlo S, Gambardella G, Prietto P, Rolfo D, Trotta P and Vallero A 2014 A novel methodology to increase fault tolerance in autonomous FPGA-based systems 2014 IEEE 20th International On-Line Testing Symposium (IOLTS) (Spain, Platja d’Aro: IEEE) pp 87–92.

[3] Khairullah S and Elks C 2020 Self-repairing hardware architecture for safety-critical cyber-physical-systems IET Cyber-Phys. Syst. Theory Appl. vol 5(1) (England Wales: IET Digital Library), pp 92–99.

[4] Miculka L, Straka M, and Kotasek Z 2013 Methodology for fault tolerant system design based on FPGA into limited redundant Area 2013 Euromicro Conference on Digital System Design (USA, CA, Los Alamitos: IEEE) pp 227–234.

[5] Kshirsagar R and Sharma S 2012 Fault tolerance in FPGA through horse shifting 2012 Fifth International Conference on Emerging Trends in Engineering and Technology (Japan, Himeji: IEEE) pp 228–232.

[6] Lahrach F, Doumar A, Châtelet E, and Abdaoui A 2010 master-slave TMR inspired technique for fault tolerance of SRAM-based FPGA 2010 IEEE Computer Society Annual Symposium on VLSI (Greece, Lixouri: IEEE) pp 58–62.

[7] Farias M, Nedjah N and de Carvalho P 2020 Active redundant hardware architecture for increased reliability in FPGA-based nuclear reactors critical systems 2020 23rd Euromicro Conference on Digital System Design (DSD) (Slovenia, Kranj: IEEE) pp 608–615.

[8] Chmiel M, Kulisz J, Czerwinski R, Krzyzyk A, Rosol M, and Smolarek P 2016 An IEC 61131-3-based PLC implemented by means of an FPGA Microprocess. Microsyst. vol 44 (Netherlands: Elsevier) pp 28–37.

[9] Luo P and Zhang J 2011 SEU mitigation strategies for SRAM-based FPGA Vol 8196 81960N-1 (China: Beijing) pp 1–10.

[10] Zheng M, Wang Z, and Li L 2015 DAO: Dual module redundancy with AND/OR logic voter for FPGA hardening 2015 First International Conference on Reliability Systems Engineering (ICRSE) (China, Beijing: IEEE) pp 1–5.

[11] Afzaal U, and Lee J 2018 A self-checking TMR voter for increased reliability consensus voting in FPGAs IEEE Trans. Nucl. Sci vol 65(5), IEEE pp 1133–1139.

[12] Dubrova E 2012 Fault-Tolerant Design (New York: Springer), pp 1–194.

[13] Shaker M, Hussien A, Alkady G, Amer H, and Adly I 2019 Mitigating the effect of multiple event upsets in FPGA-based automotive applications 8th Mediterranean Conference on Embedded Computing (MECO) (Montenegro, Budva: IEEE) pp 1–4.

[14] Bolchini C, Miele A, and Sandionigi C 2013 Autonomous fault-tolerant systems onto SRAM-based FPGA platforms Journal of Electronic Testing vol 29 (New York: Springer), pp 779–793.