# Multi-Architecture Fault-Tolerant Approach for In-Mission Constraint & Requirement Variability

Beatrice Shokry
*Electronics and Comm. Eng. Dept.*
*American University in Cairo*
Cairo, Egypt
beatrice@aucegypt.edu

Tarek K. Refaat
*Engineering Department*
*Cisco Systems*
Ottawa, ON, Canada
tarek.k.r@ieee.org

Hassanein H. Amer
*Electronics and Comm. Eng. Dept.*
*American University in Cairo*
Cairo, Egypt
hamer@aucegypt.edu

***Abstract*—This paper studies an FPGA-based system and shows how to switch (during runtime) between different architectures when the system is subjected to varying requirements (such as area, performance, reliability, fault security). A generic combinational module is at the center of this work. Several implementations of this combinational function must be available on chip or downloadable using Dynamic Partial Reconfiguration (DPR). A switching criterion is proposed to decide on the most suitable architecture at any point in time during system operation. Two designs are then proposed to enable the switching process. Parts of the design are implemented on Altera Cyclone IV FPGA.**

***Keywords—FPGA, SEU, fault secure, reliability, DPR***

## I. Introduction

Field Programmable Gate Arrays (FPGAs) have seen increased adoption over Application Specific Integrated Circuits (ASIC) and custom processors in various applications. Harsh environments and safety-critical systems including automotive, nuclear power, or space applications, can benefit from the customizability and flexibility of FPGAs. With the wider adoption of autonomous, hybrid and electric cars, the automotive industry is a main driver of technological innovation. Governments are offering financial incentives and often setting deadlines for going fully electric. In 2019, for example, Canada's Federal Government and Blackberry partnered to invest $350M into the autonomous vehicle research and development. Autonomous vehicles, compared to traditional vehicles, will have further reliance on inter/intra-vehicle networks, intelligence, processing, data analysis and control. FPGAs can meet these demands, with high-speed implementations of control functions and infotainment, such as antilock braking systems (ABS), video and traffic processing, lighting, traction, and safety control [1-3].

Nuclear power plants and space exploration are also industries seeing further innovation, and wider adoption. With billion-dollar investments in space exploration, and constant searches for oil-alternative energy sources, these industries can also capitalize on the customizability and flexibility of FPGAs. Due to distance constraints, space applications need to carry out on-board processing for high-speed decisions, as opposed to relying on Earth-based decision making. Data is typically only sent back for storage and analysis [4]. Similarly, with nuclear power plants being reliant on high-speed processing, FPGAs can be a suitable technology for such applications [5].

Various technologies exist for FPGA implementation [4]. These implementations are often at risk of Single Event Upsets (SEUs) due to radiation exposure in harsh environments. Single Event Upsets can manifest in the flipping of a single bit. However, FPGAs provide the multi-purpose feature: Dynamic Partial Reconfiguration (DPR) [6]. This feature offers the ability to reconfigure the on-board logic of the FPGA, in real-time, during operation [7, 8]. DPR is a useful technique, that can be utilized for fault tolerance.

Mission-critical systems, susceptible to failures, often incorporate some form of fault tolerance. Such systems should be able to continue operation, if even in a degraded mode, in the event of a failure, with a significantly decreased probability of total system failure. Many fault tolerance techniques exist and two examples (to be focused on in this paper) are Triple Modular Redundancy (TMR) and Duplication with Comparison (DWC).

Aside from fault tolerance, a system can be designed such that it is fault secure. Fault security is a notion related to fail-safe design. A fault secure system is a design technique that will ensure either operating based on verified correct outputs or producing an indication that the output is erroneous and cannot be relied upon [9].

Typical applications operate based on predictable and consistent reliability requirements. However, there are some use cases, where requirements and constraints vary during operation. There are times when tradeoffs need to be made, compromising between reliability, speed, and FPGA logic area utilization. In a space application for example, radiation intensity varies along different points in orbit [4]. This is a unique issue to be tackled in this study, where an FPGA will need dynamic reprogramming of architecture, depending on the specific stage of operation, during mission time.

This paper focuses on a generic combinational module M. During its operation, requirements are expected to change. For example, if the intensity of the radiation increases, a more reliable architecture will be needed. Therefore, four architectures will be investigated, each with its own area, performance, reliability, and fault security attributes. During runtime, requirements will be evaluated, and the appropriate architecture will be downloaded onto the FPGA. A switching criterion is proposed to help with the determination of this appropriate architecture, during runtime.

Section II next, describes the four architectures studied in this paper along with each one's attributes. In Section III, the switching criterion is developed, in order to quantitatively evaluate each architecture, as function of runtime requirements. In Section IV, a demonstrative example is presented. Finally, Section V concludes this work.

## II. Description of the Different Architectures

This Section focuses on a generic combinational module "M" and its different fault-tolerant architectures. Four such architectures will be studied in this work. Each architecture will be explained along with its attributes such as area requirements, performance, fault security and reliability. The area is going to be evaluated as being the size of the partial bit

2021 3rd Novel Intelligent and Leading Emerging Sciences Conference (NILES) | 978-1-6654-2157-7/21/$31.00 ©2021 IEEE | DOI: 10.1109/NILES53778.2021.9600514

file of the architecture. Performance will be measured by the maximum frequency of operation of the architecture.

Fault security is another aspect of system operation which is often overlooked; it is related to the concept of "fail-safe circuits". A circuit is fault secure if it either outputs correct data or if the error in the data can be recognized by the receiver [10]. When an error is detected, a commonly used technique is to stop/restart the circuit. Some systems may cause safety hazards when using incorrect data (such as industrial or automotive systems). If the system is stopped as soon as an error is detected, it can then be called fail-safe because it fails in such a way so as to prevent hazards.

Finally, reliability is the probability that the circuit is operational at time t given that it was operational at t=0. Two of the techniques used to calculate reliability are combinatorial methods and Markov models [11]. The fault model used in this work is SEUs.

The first architecture is the "SIMPLEX" architecture. No fault tolerance is added to this architecture. Let its area on the FPGA be "$A_M$", its maximum frequency of operation be "$F_M$" and its reliability $R_M(t)$. The simplex architecture is obviously not fault secure; there is no indication that the output is incorrect as a result of a failure. Let the failure rate of one bit in the partial bit file be equal to $q$. Assuming that the probability distribution of the time to failure is the exponential distribution, $q$ will be constant [11]. The reliability of one bit will be equal to $e^{-q\times t}$. With no fault tolerance and focusing on SEUs, M becomes a series system, i.e., any SEU affecting any of the bits in the partial bit file of M will produce a failure. Hence, $R_M(t) = e^{-n\times q\times t}$ where $n$ is the number of bits in M's bitfile. In summary, M has an area $A_M$, a performance $F_M$ (maximum frequency of operation), a reliability $R_M(t)$ and is not fault secure.

The second architecture uses Alternating Logic in the design of M. Let this architecture be called $M_{AL}$. Alternating logic has been used to detect errors in combinational circuits [10]. The technique consists of altering the specifications of a combinational circuit such that the output of the circuit for any input vector is the complement of the output produced when applying the inverse of that input vector. Hence, by applying any input vector followed by its inverse, the two outputs must be complementary. This technique usually requires the addition of an extra input variable. Hence, $M_{AL}$ is fault secure. As soon as the two consecutive outputs are not complementary, the error in $M_{AL}$ is detected. The obvious disadvantage of the $M_{AL}$ architecture is its low performance; any input vector has to applied twice and the frequency of operation drops to F/2. Furthermore, the introduction of alternating logic on the original design of the M module will undoubtedly increase its area and power consumption. Let the number of bits in $M_{AL}$'s partial bit file be equal to $\alpha\times n$ where $\alpha \geqslant 1$. Consequently, $R_{M_{AL}}(t) = e^{-\alpha\times n\times q\times t}$. Fig. 1 shows an error detector for the $M_{AL}$ circuit (Err-det circuit). It consists of a two-bit shift register clocked with a signal of frequency F. Every two clock cycles, the output of $M_{AL}$ followed by its complement is stored in this shift register. The contents of both bits are XORed and produce a signal *e'* which is equal to zero when both bits are identical (indicating an error). The signal *e'* is sampled at a frequency F/2. The output of the Err-det circuit (OUT) is passed to the rest of the circuit if *e'=1*. A simple hardwired processor like the one proposed in [12] along with the Internal Configuration Access Port (ICAP) can be used to handle the error detection and reconfiguration. Signal e' can be sent to this processor.

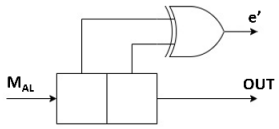


Fig. 1. Error detector circuit.

The third architecture is similar to the Duplication With Comparison (DWC) technique [11]. Its advantage over the two previous architectures is that the system can continue operating correctly and without interruption, even in the event of a failure. In certain situations, interruptions may not be tolerated. A solution to this issue is to install one M and one $M_{AL}$ ($M+M_{AL}$ architecture). The outputs of both M and $M_{AL}$ are compared by an Exclusive-Or gate (with output *d*) which, when it detects a discrepancy, indicates that one of the two modules has failed. Fig. 2 has the block diagram of this circuit.

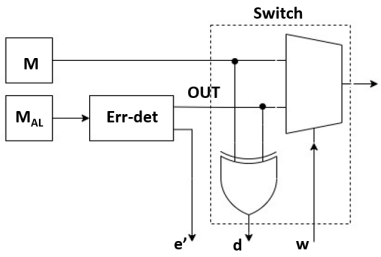


Fig. 2. $M+M_{AL}$ architecture.

The output of $M_{AL}$ is connected to the Err-det circuit. The output of M and *OUT* are applied to a two-input multiplexer to determine which output is connected to the rest of the system. If M is connected to the rest of the system and fails, the XOR will detect the discrepancy but the *e'* signal will indicate that $M_{AL}$ is functioning correctly; hence, the problem is in M. The signals *e'* and *d* are applied to the hardwired processor described above which toggles its *w* output and $M_{AL}$ is now connected to the system. The processor also initiates DPR on the M module. If an SEU affects the disconnected $M_{AL}$, both *d* and *e'* will indicate an error. The hardwired processor initiates DPR on $M_{AL}$ and does not change *w* keeping M connected to the system. Fig. 3 has the Markov model which will be used to calculate system reliability. The model has four states. The name of the state indicates which of the two modules is operating correctly. Obviously, in state "Failure", both modules will have failed. The transition rate from state ($M+M_{AL}$) to state M is $\lambda_M$ and to state $M_{AL}$ it is $\lambda_{M_{AL}}$. The transition from states ($M_{AL}$) and (M) back to state ($M+M_{AL}$) depends on the amount of time taken to DPR M and $M_{AL}$ respectively. The rate of these transitions is $\mu_M$ and $\mu_{M_{AL}}$. The Chapman-Kolmogorov equations can be used to solve the model and find the probability of not being in state Failure (=R(t)) [11].

The fourth architecture is the very commonly used Triple Modular Redundancy (TMR) architecture [11]. Three identical M modules are operated in parallel, and the three outputs are applied to a majority voter. In the context of FPGAs and SEUs, TMR reliability can be enhanced as follows: instead of the conventional majority voter, a more powerful one (similar to the one in [10]) can be used.

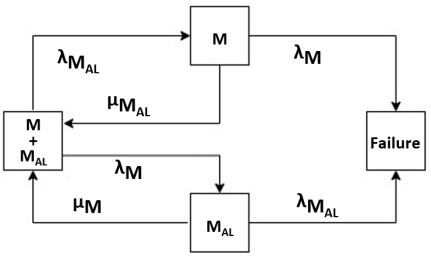


Fig. 3. Markov model for M+$M_{AL}$.

Let this voter be called *en-voter*. It identifies the first module which fails by producing a two-bit output in addition to the output of M. These two outputs ad1 and ad0 equal 00 if all three inputs to the voter agree. If the first M is different from the other two, ad1:ad0=01. For a failure in the second M, they equal 10 and for a failure in the third M, they equal 11. These two signals are sent to the hardwired processor. Consequently, DPR can be applied to the failed module and the TMR system is back again with three operational modules. However, if at any point in time, the voter receives at least two incorrect M outputs, its own output will be incorrect, and this error may not be detected right away. Hence, the TMR architecture is not fault secure. In terms of performance, the maximum frequency of operation will be slightly reduced since the delay of the voter has to be added to the delay of M. For reliability calculations, a Markov model can be developed for the TMR architecture and solved to find $R_{TMR}(t)$ using the same techniques as for the reliability calculations of the M+$M_{AL}$ architecture.

Next, a switching criterion will be obtained for every architecture in order to help the system choose the appropriate architecture during runtime.

## III. Switching Criterion

Two different scenarios will be discussed in this Section. First, assume that area is an issue in a particular application; hence, the four architectures must fit in the smallest possible area on the FPGA. Sub-section A will present a design which fits this requirement. Second, if it is essential not to have any interruptions when switching from one architecture to another and area requirements are not critical, a design is presented which will not have any output interruptions when switching from any of the four architectures to another.

### *A. First Scenario*

Regarding the first scenario, three blocks are reserved for three copies of the module. One of these copies must be large enough to house $M_{AL}$. Hence, there are always two copies of M, and the third copy may be M or $M_{AL}$. The routing circuit in Fig. 4 will determine which of the four architectures will be connected to the rest of the system. The advantage of this system is that it only needs three copies of module M (which is required for TMR) and one of these copies can be replaced by $M_{AL}$. So, the required area is that of two M modules and one $M_{AL}$. The drawback of this system is that, when it is decided to switch from one architecture to another due to changes in the situation/environment, the system may have to be stopped until the new architecture is downloaded onto the appropriate location. For example, switching from TMR to $M_{AL}$ requires, stopping the system until $M_{AL}$ is downloaded and the appropriate signals are applied to the routing circuit in Fig. 4.

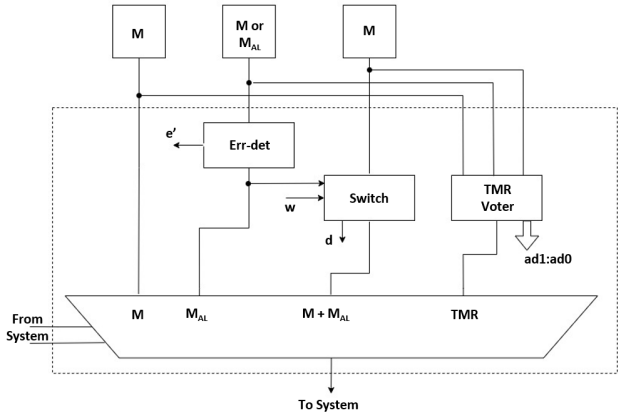


Fig. 4. Routing circuit.

In Section II, several attributes were discussed for each architecture, namely area, performance, reliability and fault security. Table I summarizes these attributes. Regarding the area, it is normalized with respect to the size of the partial bit file for module M. Introducing alternating logic to the design of M will increase its resource requirements if the function requires an additional input. Let the partial bit file for $M_{AL}$ consist of α×n bits where α⩾1. Normalizing with respect to the area A of module M, the area required for $M_{AL}$ will be α. The maximum frequency of operation is also normalized with respect to that of M (= $F_M$). For $M_{AL}$, the maximum frequency of operation is $F_M/2$ since every input must be applied twice, once as is and another time inverted.

The figure of merit $\boldsymbol{\eta}$ can be defined as:

$$\eta = \left(\frac{H1}{Ar}\right) + (H2 \times F) + \big(H3 \times R(t)\big) + H4 \times FS \qquad (1)$$

where *Ar* represents the architecture's area, *F* its maximum frequency of operation, *R(t)* its reliability and *FS* is a binary variable equal to 1 if the architecture is fault secure and 0 otherwise. The variables H1 through H4 are values between 0 and 1 determined by the designer depending on the relative importance of each attribute at any point in time. H1+H2+H3+H4=1.

TABLE I. Architectures' Attributes

| Architecture | Area | Max Freq | R(t) | FS |
|---|---|---|---|---|
| M | 1 | 1 | 0.942 | 0 |
| $M_{AL}$ | α=1.5 | β=0.5 | 0.914 | 1 |
| M+$M_{AL}$ | 1+α=2.5 | β=0.5 | 1 | 1 |
| TMR | y | γ=1 | 1 | 0 |

where
α = area for $M_{AL}$/area for M
β = max. frequency of $M_{AL}$/max. frequency of M
γ = max. frequency TMR/max. frequency M
y = area for (TMR+voter)/area for M

As an example of the use of Table I and η, let the size of module M's bitfile be 16KB (= n). Furthermore, assume that the size of the bitfile for $M_{AL}$ is 24KB, i.e., α=1.5 (as in Section IV below). For simplicity, the delays of the routing circuit have been neglected; hence β=0.5 and γ=1. R(t) is determined using the equations and Markov models in Section III and calculated after one month of continuous operation. The value of q (=$1.52\times10^{-8}$/hour) was taken from [13]. The values of $\mu_M$ and $\mu_{M_{AL}}$ are also based on data from [13]. For example, if the

situation is such that reliability and fault security are more important than area and performance, let (H1H2H3H4) = (0,0,0.5,0.5). The M+$M_{AL}$ architecture will have the highest η=1.

### B. Second Scenario

As mentioned above, there may be situations where switching from one architecture to another requires downloading (using DPR) one or more partial bit files. There are applications where such interruptions are not allowed. Therefore, a new system is proposed next which guarantees no interruptions when switching to a new architecture. This system requires four regions for M as opposed to just three as in the system described in sub-section III.A. One of these regions must be large enough to house $M_{AL}$. Hence, this system has larger area requirements than the previous one.

Since there are four architectures considered in this work, all $12 = P_2^4$ cases are studied and any switch from one architecture to the other does not cause any system interruption. For example, in the first scenario, switching from TMR to $M_{AL}$ required downloading $M_{AL}$ instead of one of the M modules in the previous system; hence, the system had to be interrupted. Here, since there are four regions, $M_{AL}$ can be downloaded into the fourth region while TMR is still operating in the first three regions. When $M_{AL}$ is ready and operational, its output is connected to the rest of the system while simultaneously disconnecting the TMR output.

## IV. Demonstrative Implementation

In this section, an example for a module M and its corresponding $M_{AL}$ module are studied. A normal 2-bit multiplier (N-MUL) and a 2-bit multiplier with alternating logic (AL-MUL) are implemented on Altera Cyclone IV E EP4CE22F17C6 FPGA. As shown in Fig. 5, A and B are the inputs for the N-MUL while OUT is the output of the multiplication operation. To realize the alternating logic version of N-MUL (AL-MUL), the Control input is added.

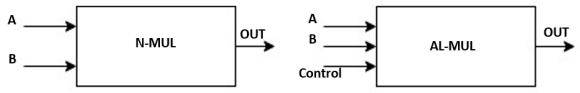


Fig. 5. Block diagram of the Normal and Alternating Logic versions of a 2-bit multiplier.

TABLE II. Resource Utilization of a 2-bit Multiplier with and without Alternating Logic

| Architecture | % LUT |
|---|---|
| N-MUL | 0.018 |
| AL-MUL | 0.027 |

TABLE III. Resource Utilization of the Multiplexer

| Architecture | % LUT |
|---|---|
| Multiplexer | 0.036 |

Table II compares the resource utilization of both architectures and shows that AL-MUL has 50% resource overhead. Additionally, the multiplexer needed to select the desired output (bottom of Fig. 4) was described in RTL VHDL and implemented on the aforementioned FPGA. Table III shows its resource utilization, assuming that it selects a 4-bit output from a set of four possible architectures and hence, uses a 2-bit selector.

## V. Conclusion

Nowadays, FPGAs are often used in Industrial, Automotive and Space applications among others. Circuit and/or environmental conditions can change during operation. Therefore, it is important to be able to switch architectures on the fly to adapt to the new requirements. This paper focuses on four different implementations of a generic combinational block regardless of its size. These implementations differ in terms of area, performance, reliability and fault security; only SEUs are considered. FPGA-based designs are presented to enable switching between these four implementations depending on the required circuit/environment characteristics at the time. One design focuses on reducing overall area but may require system interruption during the switching process. The second design guarantees no system interruption but requires a larger area. A switching criterion is developed to determine the most appropriate implementation at any given time. A demonstrative example of a combinational function with and without alternating logic (for fault security) is implemented on Altera Cyclone IV E EP4CE22F17C6 FPGA.

## References

[1] S. Shreejith, S.A. Fahmy and M. Lukasiewycz, “Reconfigurable computing in next-generation automotive networks,” IEEE Embedded Systems Letters, March 2013, 5, 12-15G.

[2] M. Gabrick, R. Nicholson, F. Winters, B. Young and J. Patton, “FPGA considerations for automotive applications,” SAE Technical Paper, 2006.

[3] P. Rech, J-M Galliere, P. Girard, A. Griffoni, J. Boch, F. Wrobel, et al, “Neutron-induced multiple bit upsets on two commercial SRAMs under dynamic-stress,” IEEE Transactions on Nuclear Science, August 2012, 59, 893-899.

[4] F. Siegle, T. Vladimirova, J. Ilstad and O. Emam. "Mitigation of radiation effects in SRAM-based FPGAs for space applications," ACM Computing Surveys (CSUR) vol. 47, no. 2, 2015, pp. 37.

[5] M. Farias, R. Martins, P. Teixeira and P. Carvalho, "FPGA-based I&C systems in nuclear plants", Chemical Engineering Transactions, vol. 53, pp. 283-288, September 2016.

[6] Xilinx, "Xilinx Partial Reconfiguration user guide," UG702, April 2013.

[7] J.J. Rodríguez-Andina, M.D. Valdés-Peña and M.J. Moure, "Advanced features and industrial applications of FPGAs-A review," IEEE Trans. on Industrial Informatics, vol. 11, no. 4, pp. 853-864, August 2015.

[8] C. Bernardeschi, L. Cassano and A. Domenici, "SRAM-based FPGA systems for safety-critical applications: a survey on design standards and proposed methodologies," Journal of Computer Science and Technology, pp. 373-390, 2015.

[9] T.K. Refaat, H.H. Amer, G.I. Alkady, R.M. Daoud and H.M. ElSayed, “Machine operating speed, fault security and fault tolerance for performability analysis in industrial automation.” Proceedings of the Intern. Symp. on Industrial Electronics ISIE, Kyoto, Japan, June 2021.

[10] D.G. Mahmoud, G.I. Alkady, H.H. Amer, R.M. Daoud, I Adly, Y Essam, H.A. Ismail and K.N. Sorour, “Fault secure FPGA-based TMR voter”, Proceedings of the Mediterranean Conference on Embedded Computing MECO, Budva, Montenegro, June 2018

[11] D.P. Siewiorek and R.S. Swarz, “Reliable Computer Systems Design and Evaluation,” A K Peters, Natick, Massachusetts, 1998.

[12] G.I. Alkady, R.M. Daoud, H H. Amer, I Adly, H.H. Halawa and M.B. Abdelhalim, “Highly reliable controller implementation using a network-based fully reconfigurable FPGA for industrial applications”, Proc. of the IEEE Intern. Conf. on Emerging Technologies and Factory Automation ETFA, Limassol, Cyprus, September 2017.

[13] B. Shokry, G.I. Alkady, H.H. Amer, R.M. Daoud, I. Adly and H.M. ElSayed, "Error detection and recovery in FPGA-based pipelined architectures," Proceedings of the Novel Intelligent and Leading Emerging Sciences Conference NILES, Cairo, Egypt, October 2020.