Dynamic Fault Tree Models for FPGA Fault Tolerance and Reliability

Yassmeen Elderhalli[1]✉, Nahla El-Araby[2,3], Osman Hasan[4], Axel Jantsch[2], Sofiène Tahar[1]
[1]Concordia University, Montreal, QC, Canada
{y_elderh, tahar}@ece.concordia.ca
[2]TU Wien, Vienna, Austria
{nahla.el-araby, axel.jantsch}@tuwien.ac.at
[3]Canadian International College, Cairo, Egypt
[4]National University of Sciences and Technology, Islamabad, Pakistan
osman.hasan@seecs.nust.edu.pk

***Abstract*—Field Programmable Gate Arrays (FPGAs) are widely used in many safety-critical applications mainly due to their high computational efficiency and dynamic reconfiguration. Although dynamic reconfigurability is often leveraged upon to attain further flexibility and reliability, it comes with an area overhead. In this paper, we provide a methodology to analyze the trade-off between reliability and area in dynamically reconfigured FPGA systems. We mainly aim to find the lowest area overhead for a given fault recovery rate in different system modules. For this purpose, we provide a generic model for system reliability using Dynamic Fault Trees (DFTs) that considers partially reconfigurable fallback units. The experiments are performed on a fail safe Electronic Control Units (ECUs) based automotive system. We use the FPGA partial reconfiguration to replace the faulty ECU functionality. The results show that by setting a suitable threshold for the reliability enhancement, the minimum number of fallback units can be determined. This leads to an enhanced system reliability with the most optimal area overhead.**

I. Introduction

Field Programmable Gate Arrays (FPGAs) have been extensively used as an alternative to Application Specific Integrated Circuits (ASICs) for embedded applications, due to their high computational power, flexibility and low non-recurring engineering costs [1]. However, the Static Random Access Memory (SRAM) based technology, employed by most FPGAs, increases their susceptibility to different types of faults [2], especially Commercial Off-The-Shelf (COTS) FPGAs that are not radiation hardened. For many embedded systems in domains like aerospace and automotive, reliability is a very important aspect. Thus, self healing and fault recovery techniques are required to increase the reliability of those mission and safety critical systems. A wide range of fault detection and correction techniques [3], as well as fault tolerant methods analysis are available for FPGAs [4]. Moreover, many works have investigated methods for increasing the reliability and fault tolerance of FPGAs [5], [6].

Dynamic Partial Reconfiguration (DPR) technology offers a great flexibility of reconfiguring certain FPGA partitions at run-time [7]. This feature can also be used for fast and autonomous fault recovery of transient [8] and permanent faults (e.g. [9], [10]) through swapping the functionality to other partitions. This recovery provides higher availability and increased system lifetime. However, the fact that DPR can tolerate faults at run-time by providing spare Reconfigurable Partitions (RPs) that can take over the faulty functionality is accompanied by an increased FPGA resource utilization. Although FPGAs are getting larger, area is still a great limitation specially when the design cost is to be reduced. Moreover, the increased resource utilization leads to higher power consumption and longer reconfiguration time. Thus, enhancing the reliability of limited size FPGAs is still a challenge.

Optimizing reliability enhancement techniques to decrease the generated area overhead (FPGA resource utilization) is addressed in the literature in various ways (e.g., [11], [12]). In this paper, we propose to address this problem in a novel way using dynamic fault trees (DFTs) models. A DFT is a reliability model that can effectively capture the failure behavior of spare parts in a given system. We develop DFT models and derive a generic reliability expression to determine the number of fallback partitions to be included in the design. We optimally enhance the overall system reliability with the least possible area overhead. This allows us to properly model the fallback units of the system as spare parts and thus truly capture the system behavior for any number of main and fallback units.

As a case study, we use a fail safe electronic control unit (ECU) based automotive system to demonstrate the proposed model. In such systems, it is possible to build optimized nodes that combine ECUs and network controllers on a single FPGA chip with low power consumption and small area [13]. The system is composed of a number of ECUs, a Motor Control Unit (MCU), a Throttle Sensor (THS) and a number of Fall Back Units (FBUs). The overall system reliability is evaluated for the case of single ECU and single FBU, single ECU and multiple FBUs (with different failure rates), then for multiple ECUs and multiple FBUs. We use the generic reliability expression to evaluate the reliability enhancement for these different system configurations. A trend can be observed for the number of fallback units that can be added to enhance the system reliability. We notice that adding additional units has a positive impact on the reliability. However, the percentage of enhancement lowers as additional components are added. Therefore, we propose

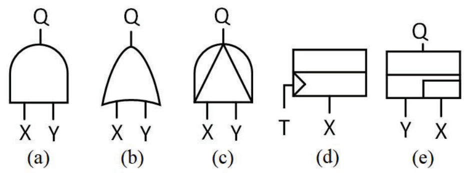

Fig. 1. Fault Tree Gates: (a) AND, (b) OR (c) PAND (d) FDEP (e) Spare

to set a threshold for the required enhancement, based on the system requirement, to minimize the number of fallback units.

The state-of-the-art literature shows that DPR was used in different ways to provide fault tolerance and enhance the reliability of FPGA based systems, also compromising reliability enhancement against area and power is a prominent problem. However, considering the interaction between system components including partially reconfigurable fall back units (replacing faulty modules) in calculating and enhancing the overall reliability is considered a novel approach.

II. Related Work

The state-of-the-art shows that FPGAs with their DPR technology are widely used for automotive ECU systems. An ECU using a fault tolerant communication controller was presented in [14], where Partial Reconfiguration (PR) is used to dynamically reconfigure a faulty controller. The authors in [15] presented an architecture for implementing a fail-safe, safety-critical ECU system on FPGAs. The FPGA is configured to monitor the control circuits and detect faults. When a fault is detected, the FPGA is dynamically reconfigured to replace the faulty modules. Partial reconfiguration was also used in other automotive technologies, like driver assistance systems presented in [16]. A scheme for implementing safety-critical ECU systems on reconfigurable hardware was proposed in [17]. Employing DPR and a custom network controller, the authors of [17] presented a scheme that is adaptable to implement an isolated fault-tolerant node or a back-up node to take over the functionality of multiple failing nodes. A short recovery time was achieved but with some area overhead. Another FPGA-based ECU system was proposed in [13] for compute-intensive non-critical functions, such as driver assistance for automotives.

The literature also investigated fault tolerance techniques for FPGA based systems through DPR, trying to optimize the area overhead and recovery time. A methodology exploiting DPR to relocate faulty modules at run-time was presented in [18] proposing a partitioning method as a solution to maximize the number of permanent faults the system can tolerate, through an algorithm that partitions the FPGA into tiles depending on the logic resources required by each tile. The number of required recovery tiles is equal to the maximum number of faulty tiles. The algorithm decreases the area overhead and increases the number of tolerable faults but with an increased computational complexity where scalability is not possible. An Algorithm Based Fault Tolerance (ABFT) approach was implemented in [19] for fault recovery using DPR and a quantitative analysis of the overheads was presented. The area overhead was shown to be 10% with an execution time penalty of 24% over the unprotected design in fault free situations.

The Maximum Empty Rectangle (MER) technique with adjacency heuristic was presented in [11], where tasks are allocated in DPR systems providing higher area utilization, higher task acceptance ratio and lower fragmentation ratio. A design flow was proposed in [6] employing a mechanism to detect and recover from permanent faults reducing the number of RPs with better utilization of FPGA resources. The design flow methodology enables relocating modules on different RPs without the need for multiple partial bitstreams. A new aging sensor was deployed adding an aging effects mitigation unit. The aforementioned techniques use DPR for enhancing the reliability. They mostly investigate partitioning algorithms for decreasing area overhead and fast recovery through smaller reconfiguration time, but are in most cases accompanied with high computational complexity and accordingly are not scalable for larger or more complex systems.

An approach for defining the suitable number of spare units needed by a system is studied in [12], where reliability and performability modeling are carried out based on Markov models. Power consumption is used as the penalty, showing that an increase in the number of spares increases the system reliability but on the other hand, leads to a poorer performability. The presented method also lacks scalability as the complexity of the Markov models significantly increases with the number of system modules. DFT models have also been used in the analysis of automotive systems. For example, in [20], DFTs are used in the Safety analysis of vehicle guidance systems. Furthermore, in [21], DFTs have been successfully used in the analysis of a drive-by-wire system with brake and throttle control units. However, to the best of our knowledge, DFTs have not been used in minimizing the area overhead incurred by adding redundant spare units. The novelty of this work lies mainly in considering the relationship between interacting system components and the effect of faulty elements on the overall system reliability to be used in determining the appropriate number of spare units for least area overhead.

III. Dynamic Fault Trees

Reliability expresses the probability of continuing to provide a reliable and correct service over a given period of time [22]. Several models are developed to capture the system reliability, which vary depending on the properties that they capture. Traditionally, Fault Trees (FTs) [23] and Reliability Block Diagrams (RBDs) [24] are used to model the system failure and reliability, respectively. However, these models cannot capture the failure dependencies and spares that exist in real-world systems. Dynamic reliability models, on the other hand, are introduced to model the sequences of failure and spares that affect the reliability of a given system, such as DFTs [23].

A DFT graphically models the failure behavior of a given system [23]. The modeling starts by a top event that represents the failure of a system or a sub-system. The DFT inputs represent the basic events that model the failure of the system components. The failure relationships between these components are captured using DFT gates. These gates allow capturing the failure dependencies that cannot be modeled using the traditional fault trees. The AND gate (Figure 1(a)) is a traditional gate, where its output fails with the failure of both inputs. The output of the OR gate (Figure 1(b)) fails when at least one of the inputs fail. The Priority-AND (PAND) gate (Figure 1(c)) models the failure sequences, where its output fails when both inputs fail in sequence. Failure triggers are modeled using the Functional DEPendency (FDEP) gate of Figure 1(d)). The spare gate, Figure 1(e), is used to model spare parts in a given system. For example, after the failure of the main part Y of Figure 1(e), the spare part X will be activated to replace the main part. The spare part can have three variants (hot, warm and cold) depending on its failure behavior in the dormant state. The hot spare (HSP) has the same failure behavior in both the dormant and active states, while the cold spare (CSP) cannot fail in its dormant state. The warm spare (WSP) is the general case, where the spare part can fail in its dormant state with a certain dormancy factor.

DFTs can be analyzed qualitatively to identify the sources and sequences of failure of the basic events that lead to the failure of the top event. A quantitative DFT analysis can also be conducted to express and evaluate the probability of failure of the top event. These analyses provide deep insights about the failure behavior of a given system and allow devising some strategies to enhance the system reliability when conducted at an early stage in the design process.

IV. Proposed Methodology

In this paper, we aim to minimize the number of fallback units that can be added, and thus the overall area, to enhance the system reliability. Therefore, we consider modeling the reliability of the given FPGA based system using DFTs to study the effect of the number of spare components on the system reliability. An FPGA system mainly consists of main units with dedicated functionality and fallback units that can be partially reconfigured to replace one of the main parts after failure. First, we assume a simple system with three main units ($U1-U3$) and a single fallback unit (FBU). The DFT of this system can be modeled using HSP gates, as shown in Figure 2. After the failure of one of the main parts, FBU can be reconfigured to replace the functionality of the failed main part. The system fails when at least two units fail (including the FBU). This behavior is equivalent to a 2 out of 4 voting gate, where its output fails with the failure of at least two inputs. In other words, the system continues to work if 3 out of 4 components are working. Thus, the reliability of this system can be expressed as:

$$Rel_Q(t) = \sum_{i=3}^{4} \binom{4}{i} \times R^i(t) \times (1 - R(t))^{4-i} \tag{1}$$

where Q is the output of the DFT in Figure 2, $R(t)$ is the reliability of a single component until time t.

We extend this model to capture the possibility of having m main units with n available $FBUs$, as shown in Figure 3. We need m out of $m+n$ units to be working. We express the reliability of this system as:

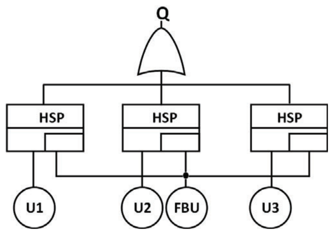

Fig. 2. DFT Model with Three Main Parts and One Spare

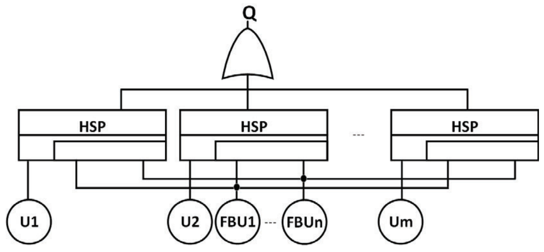

Fig. 3. Generic DFT Model

$$Rel_Q(t) = \sum_{i=m}^{m+n} \binom{m+n}{i} \times R^i(t) \times (1 - R(t))^{m+n-i} \tag{2}$$

For a given number of main units, we evaluate the reliability enhancement by first finding the average system reliability with a varying number of fallback units, over a given period of time. For instance, consider the generic model of Figure 3, we first fix the number of main units, and then we calculate the average reliability with one, two, three, and so on fallback units. With each additional iteration, we calculate the reliability enhancement of the two consecutive averages. The problem lies in knowing when to stop adding more fallback units. Thus, we propose to set a threshold for the reliability enhancement and determine the number of units that led to this enhancement. This means that adding additional units beyond this number will not have any significant impact on the reliability. The procedure to accomplish the proposed methodology is described in Algorithm 1.

Algorithm 1 Finding the Number of Fallback Units

```
1: $n \leftarrow 1$
2: $t \leftarrow [0, l]$
3: $th \leftarrow threshold$
4: $Rel_{enhancement} \leftarrow \infty$
5: while $Rel_{enhancement} > th$ do
6:     $Avg_{Rel_{n-1}} = \sum_{i \in t} Rel_{n-1}(i)/l$
7:     $Avg_{Rel_n} = \sum_{i \in t} Rel_n(i)/l$
8:     $Rel_{enhancement} = \frac{Avg_{Rel_n} - Avg_{Rel_{n-1}}}{Avg_{Rel_{n-1}}}$
9:     $n{+}{+}$
10: end while
11: $return\ n$
```

We first start by setting the number of fallback units n to 1 and assigning a time period $[0, l]$ over which we are interested in finding the reliability of the system. Then, we decide the threshold that will be used to determine when to stop adding spare parts. The reliability enhancement is first set to a big value, i.e., ∞, which allows us to enter the while loop for at least one time unit. We continue adding more units as long as the improvement in the reliability exceeds the threshold. After exiting the loop, n will have the number of additional units that satisfy the condition. In the following section, we describe how the proposed algorithm can be used to minimize the number of additional fallback units of a fail safe ECU based automotive system.

V. Case Study: ECU based Automotive System

Modern Automotive systems are complex distributed cyber-physical systems handling critical and non-critical functions. These system are based on distributed ECUs integrating processing elements and peripherals to implement a variety of functions. The complexity of modern automotive systems requires an increased number of ECUs and extensive in-vehicle network communication. Reliability of these systems is a prominent problem and is usually handled through incorporating redundant units increasing area and power. Accordingly we consider an ECU based automotive system as relevant use case to show how our approach can enhance the reliability with minimum redundant area overhead.

A. Case Study Description

The use case system consists of three regular separate control units connected through a communication link, as shown in Figure 4. The operating scenario reads a throttle position and controls the engine. The ECU is responsible for gathering the throttle position data, measured and provided by the THS. Then, the ECU converts the throttle position data into engine control data, and forwards this data to the MCU, which is responsible for controlling the engine. A fourth unit acts as a fallback unit FBU in case of a failure of one of the regular control units. This fallback unit-FBU is realized by using DPR. The size of the reconfigurable block used for the fall back unit supports resources to replace any of the other control units. Figure 5 shows a sequence diagram for the normal scenario. In this case, the FBU does not have to perform any active functionality, and it is only passively monitoring the data transfer on the communication link to detect possible errors.

The FBU has to monitor all transferred data on the bus and detect any failures (e.g., timeout, error flags, etc). In this system we assume fail silent as the system units are either

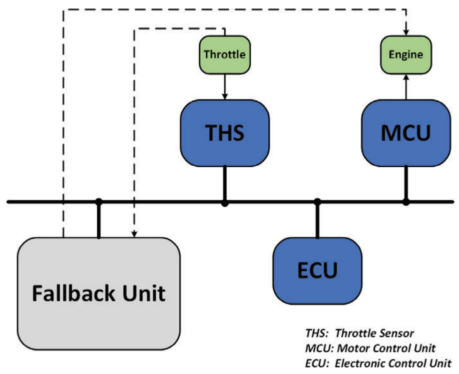

Fig. 4. Basic Concept for Failsafe ECU

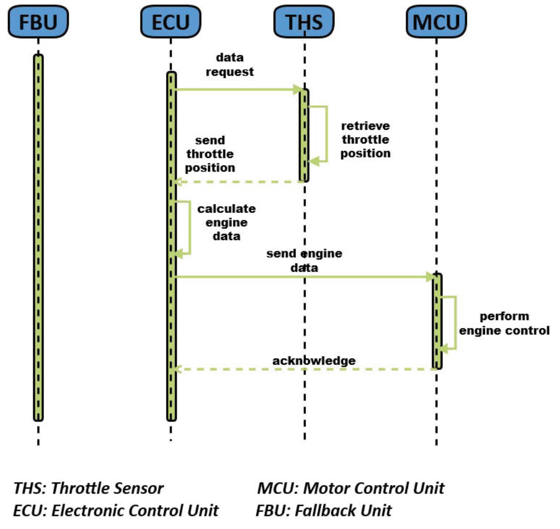

Fig. 5. Sequence Diagram for Normal Operation

functioning correctly or they stop producing output. So the bus monitor module implemented inside the FBU monitors the signal activity on the bus and can recognize a faulty module. Once an error is detected, partial reconfiguration is triggered by the bus monitor module. After the reconfiguration is done, the fallback unit completely takes over the functionality of the faulty component. Due to the fail silent assumption, the faulty device will not affect the behaviour of the system. Figure 6 shows a sequence diagram including a faulty MCU. The bus monitor detects that the MCU is running into a timeout and triggers a DPR to take over its functionality. After finishing the reconfiguration, normal operation takes over again (see Figure 5), but the fallback unit serves as MCU now.

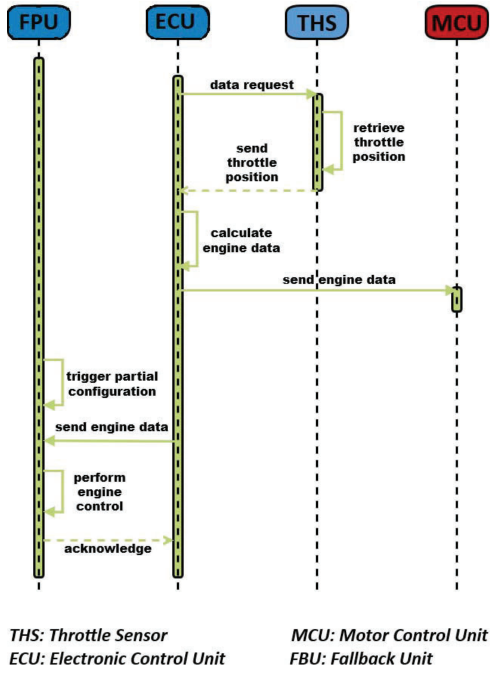

Fig. 6. Sequence Diagram for MCU Failure

The use case was implemented on a Zedboard, Zynq 7000 Evaluation Board and the ECUs are based on ARM Cortex-M1/M3 controller which is suitable core for ECU implementation due to the possibility of streamlined software development. We consider the case of 1 ECU, 1 THS, 1 MCU and 1 FBU. The design utilizes around 11% of the Zynq available hardware resources with the detailed resource utilization key provided in Table I.

In order to minimize the number of fallback units of the system, depicted in Figure 4, we need to create its DFT model. We first use the model of Figure 2, where the system has an ECU, MCU and THS with a single FBU. Then, according to our proposed algorithm, with each iteration we add an additional FBU. The system continues to work as along as there are three functioning components. These components can be the main parts, i.e., ECU, MCU and THS or their FBU replacements. We compute the reliability for up to 100 iterations, assuming exponential distributions with the same failure rate of 1×10^{-3} for all components, as shown in Figure 7(a). The reliability is enhanced with each additional component (iteration), but it is evident that this enhancement does not increase significantly when more units are added. We change the failure rates of the components to 3×10^{-3} and repeat the analysis for 50 iterations to ensure that this trend is independent of the failure rate, as shown in Figure 7(b). Clearly, the reliability trend is similar to the one of Figure 7(a), which means that with each additional spare unit the reliability improves.

We also consider the possibility of having more than three main units with multiple FBUs, i.e., several ECUs. Thus, we use the generic DFT of Figure 3 to model the system reliability. Figure 7(c) depicts the reliability of the system with five main units (3 ECUs, 1 MCU and 1 THS) and one FBU up to 100. We use the same failure rates as of Figure 7(a). It is noticed that the reliability of the system is lower than that of Figure 7(a), i.e., with only one ECU. Thus, this system requires more FBUs in order to guarantee a certain level of reliability enhancement. Therefore, as mentioned previously, we need to determine the number of components to be added at which the reliability enhancement will not be improved significantly. It is worth mentioning that such generic results cannot be obtained using Markov chains based tools, where the state space grows exponentially with the number of system components.

The second phase of the algorithm is to calculate the reliability enhancement with each iteration. We consider several variations of the system, i.e., for different numbers of ECUs (4 to 8 ECUs). Based on Algorithm 1, we need to calculate the reliability enhancement with each additional FBU (each iteration). Figure 8 shows the reliability enhancement calculated using the average reliability. We notice that the enhancement drops significantly with each added component and thus, adding more redundant components to the system can have a negligible effect on the reliability. According to the proposed algorithm, we need to set a threshold for the enhancement. If we assume that our target reliability enhancement threshold is 0.04, then, based on Figure 8, we need $13.75 \approx 14$ FBUs to achieve the target reliability enhancement. It can be noticed that increasing the number of main components in the system requires adding more FBUs to achieve the same target reliability enhancement.

TABLE I
RESOURCE UTILIZATION, POST-IMPLEMENTATION

Type	Used	Available	Percent [%] used
LUT	4827	53200	9.07
LUTRAM	289	17400	1.66
FF	4832	106400	4.54
BRAM	16	140	11.42
DSP	3	220	1.36
IO	25	200	12.50
BUFG	3	32	9.37
MMCM	1	4	25.00

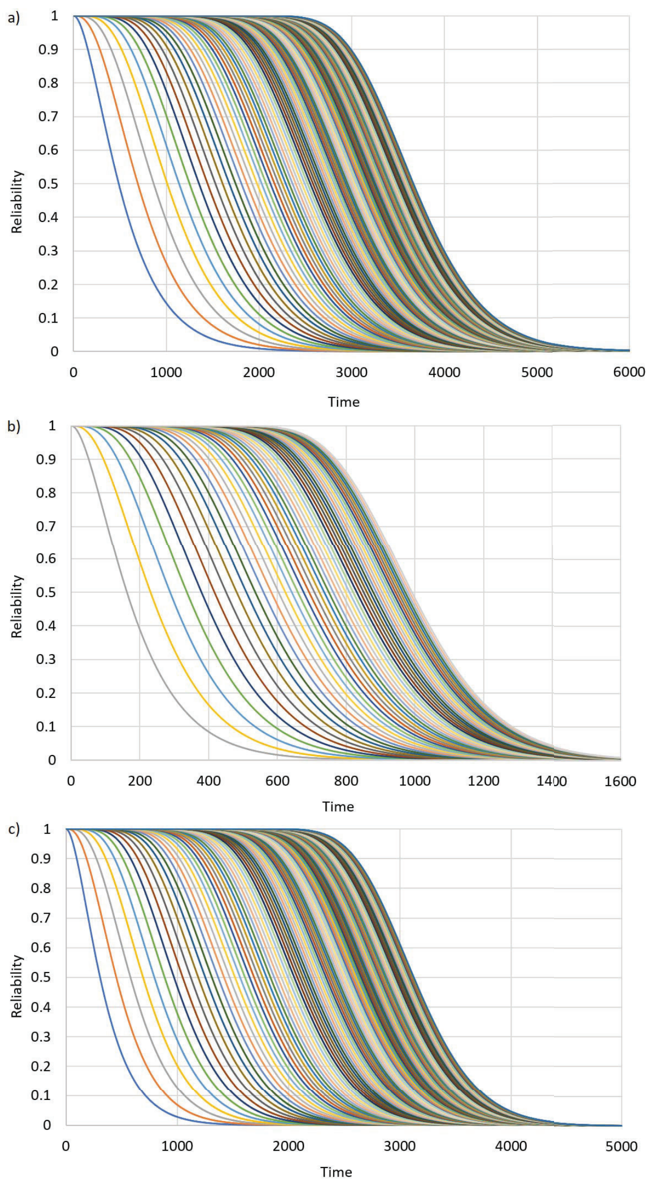

Fig. 7. System Reliability for Different Scenarios: a) Three Main Units and up to 100 FBUs with Failure Rate of 1×10^{-3} b) Three Main Units and up to 50 FBUs with Failure Rate of 3×10^{-3} c) Five Main Units and up to 100 FBUs with Failure Rate of 1×10^{-3}

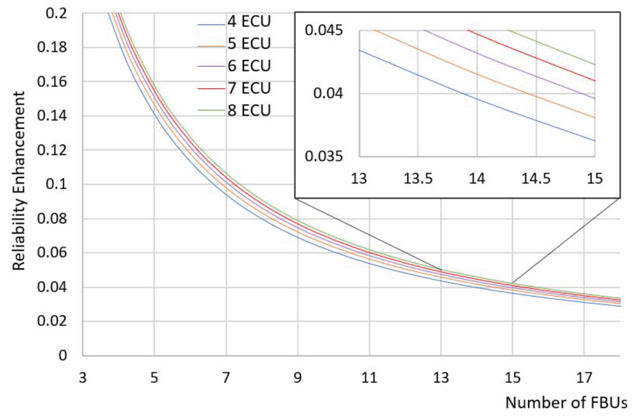

Fig. 8. Reliability Enhancement with Three Main Units and up to 100 FBUs

VI. Conclusion

In this paper, we presented a methodology to minimize the number of redundant fallback units (FBUs), i.e., spare units, in an FPGA system using dynamic fault tree (DFT) models. These FBUs replace the main faulty units in the system using FPGA dynamic partial reconfiguration. We proposed a generic DFT model that allows expressing the reliability of a given FPGA system with generic numbers of main and spare components. We proposed an algorithm that can be used to calculate the required number of FBUs based on a certain reliability enhancement threshold. This enables analyzing several scenarios to minimize the number of additional FBUs. We illustrated the efficiency of our proposed methodology using a fail safe electronic control units (ECUs) based automotive system. We showed that our DFT models can be used to calculate the reliability enhancement of several variations of the system, i.e., different numbers of ECUs and a large number of FBUs. We minimized the number of required FBUs to achieve a certain reliability enhancement. These results cannot be obtained using Markov chains tools, where the state space grows exponentially with the number of system components. The novelty of this work lies mainly in considering the relationship between interacting system components and the effect of faulty elements on overall system reliability. The developed reliability expression can be integrated into the partitioning algorithm by including the developed reliability calculations in partitioning to optimally divide the FPGA resources among the FBUs. As a future work, we plan to improve the developed reliability expression to adapt different failure rates of the system components.

Acknowledgement

The use case FPGA implementation was done through an SoC lab project at the Institute of Computer Technology, TU Wien.

References

[1] A. Hofmann, R. Wansch, R. Glein, and B. Kollmannthaler, “An FPGA based on-board Processor Platform for Space Application,” in *NASA/ESA Conference on Adaptive Hardware and Systems*. IEEE, 2012, pp. 17–22.

[2] M. Kvas, S. Valach, and P. Fiedler, “Reliability and Safety Issues of FPGA Based Designs,” *International Federation of Automatic Control Proceedings Volumes*, vol. 45, no. 7, pp. 201–206, 2012.

[3] R. R. F. L. Kastensmidt, L. Carro, *Fault-Tolerance Techniques for SRAM-Based FPGAs*. Springer, 2006.

[4] E. Stott, P. Sedcole, and P. Cheung, “Fault Tolerance and Reliability in Field Programmable Gate Arrays,” *IET Computers Digital Techniques*, vol. 4, no. 3, pp. 196–210, 2010.

[5] A. Kourfali and D. Stroobandt, “In-circuit Fault Tolerance for FPGAs using Dynamic Reconfiguration and Virtual Overlays,” *Microelectronics Reliability*, vol. 102, p. 113438, 2019.

[6] V. M. G. Martins, P. R. C. Villa, R. Travessini, M. D. Berejuck, and E. A. Bezerra, “A Dynamic Partial Reconfiguration Design Flow for Permanent Faults Mitigation in FPGAs,” *Microelectronics Reliability*, vol. 83, pp. 50–63, 2018.

[7] M. Nguyen, R. Tamburo, S. Narasimhan, and J. C. Hoe, “Quantifying the Benefits of Dynamic Partial Reconfiguration for Embedded Vision Applications,” in *International Conference on Field Programmable Logic and Applications*. IEEE, 2019, pp. 129–135.

[8] F. Ghaffari, O. Romain, and B. Granado, *Mitigation Transient Faults by Backward Error Recovery in SRAM-FPGA*. Springer, 2019, pp. 249–276.

[9] G. I. Alkady, N. A. El-Araby, M. B. Abdelhalim, H. H. Amer, and A. H. Madian, “Dynamic Fault Recovery using Partial Reconfiguration for Highly Reliable FPGAs,” in *Mediterranean Conference on Embedded Computing*. IEEE, 2015, pp. 56–59.

[10] J. Espinosa, D. de Andrés, J. Carlos Ruiz, and P. Gil, “Tolerating Multiple Faults with Proximate Manifestations in FPGA-based Critical Designs for Harsh Environments,” in *International Conference on Field Programmable Logic and Applications*. IEEE, 2012, pp. 292–299.

[11] G. Wang, S. Liu, and J. Sun, “A dynamic Partial Reconfigurable System with Combined Task Allocation Method to Improve the Reliability of FPGA,” *Microelectronics Reliability*, vol. 83, pp. 14–24, 2018.

[12] G. I. Alkady, N. A. El-Araby, H. H. Amer, and M. B. Abdelhalim, “Using Power Consumption in the Performability of Fault-Tolerant FPGAs,” in *Mediterranean Conference on Embedded Computing*. IEEE, 2016, pp. 55–58.

[13] S. Shanker, “Enhancing automotive embedded systems with FPGAs,” Ph.D. dissertation, Nanyang Technological University, Singapore, 2016.

[14] H. Pham, S. Pillement, and D. Demigny, “Reconfigurable ECU Communications in Autosar Environment,” in *International Conference on ITS Telecommunications*. IEEE, 2009, pp. 581–585.

[15] N. Chujo, “Fail-safe ECU System Using Dynamic Reconfiguration of FPGA,” *R&D Review of Toyota CRDL*, vol. 37, no. 2, 2002.

[16] C. Claus, J. Zeppenfeld, F. Muller, and W. Stechele, “Using Partial-Run-Time Reconfigurable Hardware to accelerate Video Processing in Driver Assistance System,” in *Design, Automation Test in Europe Conference*. IEEE, 2007, p. 498–503.

[17] S. Shreejith, K. Vipin, S. A. Fahmy, and M. Lukasiewycz, “An Approach for Redundancy in FlexRay Networks using FPGA Partial Reconfiguration,” in *Design, Automation & Test in Europe Conference*. IEEE, 2013, pp. 721–724.

[18] S. Di Carlo, G. Gambardella, P. Prinetto, D. Rolfo, P. Trotta, and A. Vallero, “A novel methodology to increase fault tolerance in autonomous fpga-based systems,” in *International On-Line Testing Symposium*. IEEE, 2014, pp. 87–92.

[19] J. J. Davis and P. Y. K. Cheung, “Achieving low-overhead Fault Tolerance for Parallel Accelerators with Dynamic Partial Reconfiguration,” in *International Conference on Field Programmable Logic and Applications*. IEEE, 2014, pp. 1–6.

[20] M. Ghadhab, S. Junges, J. Katoen, M. Kuntz, and M. Volk, “Model-based Safety Analysis for Vehicle Guidance Systems,” in *Computer Safety, Reliability, and Security*, ser. LNCS 10488. Springer, 2017, pp. 3–19.

[21] Y. Elderhalli, O. Hasan, and S. Tahar, “A Methodology for the Formal Verification of Dynamic Fault Trees Using HOL Theorem Proving,” *IEEE Access*, vol. 7, pp. 136 176–136 192, 2019.

[22] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, “Basic Concepts and Taxonomy of Dependable and Secure Computing,” *IEEE Transactions on Dependable and Secure Computing*, vol. 1, no. 1, pp. 11–33, 2004.

[23] E. Ruijters and M. Stoelinga, “Fault Tree Analysis: A Survey of the State-of-the-art in Modeling, Analysis and Tools,” *Computer Science Review*, vol. 15-16, pp. 29–62, 2015.

[24] O. Hasan, W. Ahmed, S. Tahar, and M. S. Hamdi, “Reliability Block Diagrams based Analysis: A Survey,” in *International Conference of Numerical Analysis and Applied Mathematics*, ser. AIP 1648, 2015, pp. 850 129.1–4.