Area, Reconfiguration Delay and Reliability Trade-offs in Designing Reliable Multi-Mode FIR Filters

Amir Hossein Gholamipour[*], Kyprianos Papadimitriou[±], Fadi Kurdahi[*], Apostolos Dollas[±] and Ahmed Eltawil[*]

[*]*University of California, Irvine*

{amirgh, kurdahi, aeltawil}@uci.edu

[±]*Technical University of Crete*

{kpapadim, dollas}@mhl.tuc.gr

***Abstract*—Wide range of digital systems from wireless devices to multi-media terminals are characterized by their multi-mode behavior. Many of these systems are deployed in high-radiation environments [5]. SRAM-based FPGAs are popular platforms to implement multi-mode systems, because of their high performance and reconfigurability. However, high susceptibility of FPGAs toward Soft Errors makes them less-than-reliable platforms. To overcome the reliability issue, various redundancy techniques have been proposed. These techniques exhibit different design and reliability characteristics. Considering the combined effect of design decisions and reliability techniques on system characteristics a coherent strategy should be devised to meet system requirements and constraints. In this work we propose a method to explore the design space for implementing a reliable multi-mode system. As we will show, different selections of design parameters and redundancy techniques generate a range of solutions which trade-off total area, reconfiguration overhead and reliability of the system. The choice of a specific solution remains a decision made by the system-designer.**

I. INTRODUCTION

An increasing number of digital systems, from wireless devices to multi-media terminals, are characterized by their multi-mode operation. This refers to the ability of a system to modify its characteristics or behavior based on user inputs or changes in the operational environment. While the fundamental algorithms to describe the functionality are the same across the modes of operation in a multi-mode system, each mode of operation is characterized by a specific set of parameters. For example in a WiMax terminal, the parameters of the MAC and PHY layers can be changed based on the current status of the wireless channel. As another example, to detect objects in Synthetic Aperture Radar (SAR) images for Automatic Target Recognition (ATR), the pattern being searched can be modified by changing the dimensions and coefficient values of an image template.

SRAM-based FPGAs are widely used to implement multi-mode systems. Their short design cycles and reconfigurability give them an advantage over ASICs. On the other hand, they are superior over general-purpose processors because of their inherent parallelism that can be used to support high data rates, and that they can be easily customized to support irregular datapaths with variable bit widths. Furthermore the support of dynamic partial reconfiguration in Xilinx FPGAs enables dynamic modification of a system during run-time.

In [4] we proposed to use the capabilities of FPGAs to implement multi-mode Finite Impulse Response (FIR) filters. These filters are commonly used in multi-mode systems due to their inherent stability and linear phase. For example, in image analysis applications, FIR filters are used for image correlation with image templates which has vast applicability in ATR. In such systems, the architectural characteristics of the FIR filters are modified based on the operational requirements of each mode.

We can design such multi-mode filters using generic structures to accommodate all the varying features of different modes. However a generic design results in large area and low performance. The generic implementation can be optimized by incorporating mode-dependent optimization techniques. Employing coefficient-specific multipliers in the implementation of an FIR filter is an example of such optimizations. Consequently another option is to implement all optimized structures on the same chip and *space-multiplex* between them. However this does not yield a scalable solution [4]. We can exploit partial reconfigurability of FPGAs and *time-multiplex* between optimized implementation of each filter. However since each filter is optimized based on specific parameters of the corresponding mode, reduced area of the optimized filter comes at the expense of making each design mode-dependent. Thus it incurs higher overhead to reconfigure the filter. To provide a balance between area and reconfiguration cost for a partially reconfigurable multi-mode filter, a previous approach proposes a spectrum of solutions that trade-off design area for reconfiguration overhead to change the mode of operation [4].

Many of the multi-mode systems mentioned earlier are often deployed in high radiation environments like space-crafts and satellites in outer space. Therefore in addition to designing the base-design we need to take measures to ensure the reliability of circuit. In this context the disadvantage of SRAM-based FPGAs is high susceptibility of the device against errors incurred due to radiation.

Many techniques have been proposed to mitigate soft errors on FPGAs. These techniques exhibit different design and reliability characteristics. As an example, a classic solution proposed in the literature and used in the industry is the Triple Modular Redundancy (TMR) technique. TMR requires redundant execution of an application on the same set of data. The overhead of TMR however is not limited to triplication of the operating modules. In fact previous works in [2] and [3] have shown that the overhead of implementing TMR ranges from 4.2x up to 6.3x the **base design** (design to be protected by redundancy techniques). Therefore considering high resource usage of multi-mode systems, to use TMR we need to devise an effective strategy to manage available resources. Such a strategy can be to optimize the implementation of base design which results in smaller overall circuit size. However

978-1-4673-0469-6/11/$25.00 ©2011 IEEE

as stated earlier, optimizing the size of base design is possible at the cost of increasing reconfiguration time overhead. Considering TMR size overhead, the reconfiguration time is even higher.

Other reliability techniques have been proposed which compromise one of the design factors to improve another factor. For example in Duplication with Compare (DWC) [12] two copies of the design are processing the data at the same time, thus providing the capability of detecting a fault. However correcting the fault requires reconfiguring both copies and thus causes degradation in system's performance. On the other hand DWC + Software (DWCSW) [1] proposes to reduce the reconfiguration overhead by identifying the faulty module. To improve performance this approach compromises the reliability of the system as will be explained later.

Clearly, the choice of the base design along with the choice of the redundancy technique generates solutions with different characteristics. In specific each solution exhibits a trade-off of total system area, application reconfiguration overhead and reliability of operation. In this work we propose to combine designing the base design and the reliability technique to meet system requirements and constraints. Since modules of the base design are different in terms of area, resource requirements, etc, we explore a hybrid approach of redundancy techniques. Therefore we can implement different sections of the base design using various redundancy techniques to exploit the benefits of different approaches. We study this approach to design reliable multi-mode FIR filters. We explore the design space and show how different design decisions and reliability techniques can affect the characteristics of the system. It should be noted that while the design techniques used in this paper are specific to FIR filters, the same concept holds for any system which can benefit from flexibility in design techniques.

Our paper is divided in to 5 sections. In Section II we present the related work. In Section III, we briefly introduce our framework on designing reliable multi-mode FIR filters. We discuss the options to implement the redundancy techniques and analyze the interaction of the design choices at both levels on overall design quality. In Section IV we present our proposed system architecture and in Section V we show the result of our analyses.

II. Related Work and Motivation

FPGAs are deployed to implement many digital systems designed around fast changing standards, application requirements or specifications. For many of such applications correlation is an essential and processing-intensive part of the implementation. For example, the authors in [5] study the implementation of Automatic Target Recognition (ATR). The challenge of ATR is to analyze a digitally represented input image or video sequence in order to automatically locate and identify all objects within the scene of interest to the observer. In that project the authors proposed a reconfigurable FPGA-based implementation to analyze Synthetic Aperture Radar (SAR) images using filters.

SAR systems are often deployed in high-radiation environments ([5] and [10]). While radiation-resistant devices can be used in such environments the increasing lower cost requirements of SAR systems prompts other approaches to ensure the reliability. The low cost requirement has motivated researchers ([7] and [8]) to investigate smaller and lower cost SAR systems. In a recent work in [9] and [14] the researchers propose using SAR imaging in sensor networks where cost and size of the device are increasingly important. To undertake the data processing in the nodes of such sensor networks, FPGA-based platforms are proposed such as [10].

An important class of errors is called *soft errors* which occur due to *Single Event Upsets (SEUs)* caused by radiation. In SRAM-based FPGAs due to their programmable nature, SEUs affect the routing, logic and memory resources. A large body of research has investigated the problem of low cost approaches to ensure the reliability of FPGA-based systems. Some of these approaches target the design of a circuit in the FPGA to make it more robust against soft errors. For example in [15] the authors propose a routing paradigm to make the design wires more robust toward soft errors by reducing the number of *care bits*. Other approaches use redundancy to detect errors. This should be further accompanied by repairing the error to ensure the reliability. Repair can be done at a number of different levels. For SRAM-based FPGAs the most promising level of repair is at configuration level [11].

TMR along with partial reconfiguration has been used for handling faults in combinatorial circuits such as FIR filters and adders [2], and for sequential circuits such as FSMs and soft-core processors [3]. It is robust in the sense that if one module is corrupted the system's output is maintained in accordance to the two other replicas. TMR's robustness however comes at the expense of substantial area overhead as mentioned earlier. DWC [12] on the other hand, reduces the overhead by keeping two copies of the design at the same time, however upon detecting a fault, takes longer to correct the fault. A recent approach lying between TMR and DWC uses two hardware replicas of the application and a third model running in software (DWCSW) [1]. This scheme exploits time redundancy by running the application in software in order to locate the faulty replica, and then corrects it using partial reconfiguration. The time overhead of fault handling is shown to be smaller in this approach compared to DWC. On the other hand, the area overhead of DWCSW is comparable to that of DWC. However DWCSW is less reliable than TMR. The reason is that after error detection and while reconfiguring the faulty module, only one module is processing the data in the system which makes it impossible to detect an error, should one happen in this period. This is discussed further in Section IV.

The selection of a proper reliability scheme relies on the application requirements [11] and [13]. Present work is the first to assess different reliability schemes and combinations thereof for guarding large reconfigurable FIR filters in SAR applications running in SRAM-based FPGAs. All experimental results are based on real measurements from a real-world platform and the output of place and route tools using the Xilinx PR module-based design flow. Our main objective is to study the effect of the designer's choices on the area, performance and reliability level of the system, when guarding similar applications.

III. RELIABLE MULTI-MODE FILTER DESIGN

In this section we briefly introduce our framework for designing multi-mode filters. Also, we investigate our options in assuring reliability of this design.

A. Joint Filter Design

In [4] we proposed *Joint Filter Design (JFD)* technique which effectively produces a range of solutions that trade-off design area for reconfiguration overhead. The output of JFD is a set of reconfigurable and static modules combined together to implement the functionality of the filter. In Xilinx Partial Reconfiguration (PR) terminology [16] each of the reconfigurable modules is called *Partially Reconfigurable Module (PRM)*. PRMs of a filter should be placed on physical locations on chip called *Partially Reconfigurable Regions (PRRs)*. To reconfigure a PRM, we need to reconfigure the PRR where the PRM is placed.

JFD operates in two levels. At the first level based on a user-defined parameter called *Level of Implementation (LI)* it partitions a filter structure to *static* and *reconfigurable* sections. The implementation of reconfigurable section is specific to the coefficient set of the filter. Higher LI results in smaller static section. To decrease reconfiguration overhead, a second user-defined parameter called *Reconfiguration Factor (RF)* is used. RF determines the amount of imposed similarity in the implementation of filters in the sequence of reconfiguration. Higher RF results in filters that are more similar, thus the reconfiguration overhead becomes smaller. We present the effect of each of these knobs through an experiment.

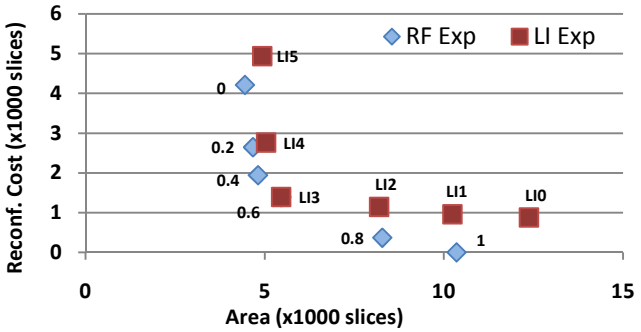

Figure 1. The tradeoff of Area and Reconfiguration Cost[1] while exploring RF and LI

Figure 1 shows the tradeoff between area and reconfiguration overhead of 10 filters designed using JFD. Each filter represents an image template and has 400 taps. In Figure 1, the X axis represents the area occupied for each filter (in thousands of slices) and the Y axis shows the average reconfiguration overhead (in thousands of slices). Each point represents one multi-mode filter designed to support 10 modes of operation (corresponding to the number of image templates). The scope of Figure 1 is twofold. First, it aims at exploring the impact of Reconfiguration Factor (RF) on the area and the reconfiguration overhead. This is illustrated with the points marked as *"RF Exp"* (RF Exploration) varied from 0 to 1, for a given Level of Implementation (LI_3). Secondly, it aims at exploring the impact of Level of Implementation (LI) on the area and the reconfiguration overhead. This is illustrated with the points marked as *"LI Exp"* (LI Exploration) varied from LI_0 to LI_5, for a given Reconfiguration Factor (RF = 0.6).

As observed in Figure 1, for the "RF Exp" experiment, by increasing RF from 0 to 1, the average reconfiguration overhead for the design reduces from around 4200 slices to a negligible size. The reason is that the designs become similar in implementation. However, this comes at the expense of increasing the size of the design. For the "LI Exp" experiment, increasing the LI from 0 to 5 (LI_5 results in a design without static section) results in a considerable decrease of the size of circuit. This is because the circuit becomes more optimized based on the coefficient set of the filter. At the same time, the average reconfiguration cost is increasing since the filters become coefficient specific and less similar.

Table 1 shows the size of different implementations of our proposed multi-mode filter. Column 1, shows the LI at which the filter is implemented for RF = 0.6). Columns 2 and 3, show the size of the static and the reconfigurable section of each filter respectively. Column 4 shows the amount of PRMs needed for each implementation. Column 5 reports the average size of the PRRs for each implementation while Column 6 shows the range of the sizes of the PRRs. As can be observed, by increasing the LI there are less number of PRMs that are generated. However in average the sizes of corresponding PRRs are larger. It is worth mentioning that the size of generic implementation for the same filter is 21,533 slices which is orders of magnitude larger than the designs implemented with JFD.

Table 1. The area of multi-mode filter and size of PRRs to implement PRMs

Base Design	Total Size (# Slices)		#PRM	Avg PRR Size (# Slices)	Range of PRR Sizes (# Slices)
	Static	Reconf.			
LI0	5900	6476	400	16	6-30
LI3	900	4562	50	91	12-151
LI4	136	4902	7	700	283-861
LI5	0	4935	1	4935	4935

It should be noted that due to popularity of using FIR filters in a multitude of applications, many FPGA vendors have included DSP units in their high-end devices, providing a power and area efficient implementation of FIR filters and similar structures. While implementing an FIR filter using DSP blocks seems to be the natural choice, in our framework we propose to implement filters using LUTs for three reasons:

- *Portability:* Different families of FPGAs offer different number of DSP blocks, ranging from a few to thousands of blocks in Xilinx Spartan to Virtex FPGAs for example. This results in a wide variance in capabilities across families and manufactures. Thus implementing filters using LUTs leads to a vendor and device-independent design which could easily be ported to other platforms.

[1] The terms reconfiguration cost and reconfiguration overhead are used interchangeably, corresponding to the amount of hardware that needs to be reconfigured in order to implement the new filter.

- *Cost:* Since FPGAs carrying many DSP units are more expensive than FPGAs lacking such resources, this way we can manage the cost of implementing the system.
- *Resource Management*: Many of the applications for which we use FIR filters include DSP intensive processing for other components of the design as well. Thus depending on the availability of resources the designer can greatly exploit the flexibility in designing filters in absence or partial availability of these structures.

Our framework however, does not preclude the use of DSP blocks for the devices that include such resources. In [17] we proposed an approach to implement a multi-mode filter fully or partially using available DSP blocks, thus giving the designer the flexibility to integrate filter designs at system-level.

B. Reliability

We design a multi-mode filter using JFD technique introduced earlier in Section III.A. To assure the reliability of such system we propose to use a hybrid approach of TMR and DWCSW. By "a hybrid approach" we are proposing to implement reconfigurable section of the filter using one redundancy approach and the static section using another approach. In specific we focus on TMR and DWCSW as they provide different characteristics in terms of area overhead, reliability and fault handling efficiency. A hybrid approach however does not preclude using one redundancy technique for the entire design. We can extend this scheme to implement each individual reconfigurable module using TMR or DWCSW techniques, however as this complicates the placement of modules it is beyond the scope of this paper.

The design decision at each level (base-design and redundancy technique) is not independent from each other. As mentioned earlier the quality of base design in terms of overall area and reconfiguration overhead is determined by the selection of LI and RF. With regard to redundancy, the number and size of the modules guarded using TMR super-linearly increases the overhead of implementation [3]. On the other hand, the system remains unreliable for longer period if the circuit guarded by DWCSW is larger, as this scheme exploits time dimension to detect the faulty module.

The interdependency of the design choices made at the two levels of reliable multi-mode filter design, prompts that we need to explore the design space and select the design which best suits the system requirements. In Section IV we introduce the system architecture. In Section V we show the effect of each parameter on the design quality in terms of area, reconfiguration overhead and reliability. The choice of the setup that best meets the system requirements remains a system-level design decision.

IV. SYSTEM ARCHITECTURE

This section discusses the architectural details of the system guarding the FIR filters. We discuss TMR and DWCSW schemes separately to examine how the filter design fits into each of them. Also, a qualitative comparison between the two schemes in terms of the resource requirements and response in fault handling is described.

A. DWCSW-based architecture

The System architecture using DWCSW scheme is depicted in Figure 2 [1]. Two separate PRRs are allocated in the FPGA fabric for the FIR filter cores. The FIFO, which is implemented with BRAM blocks, resides at the input of the system and is connected to the filter cores and the PowerPC (PPC). The FIFO controller is guarded using the TMR scheme. The partial bitstreams are loaded from a compact flash to a DDR memory at the power up of the system.

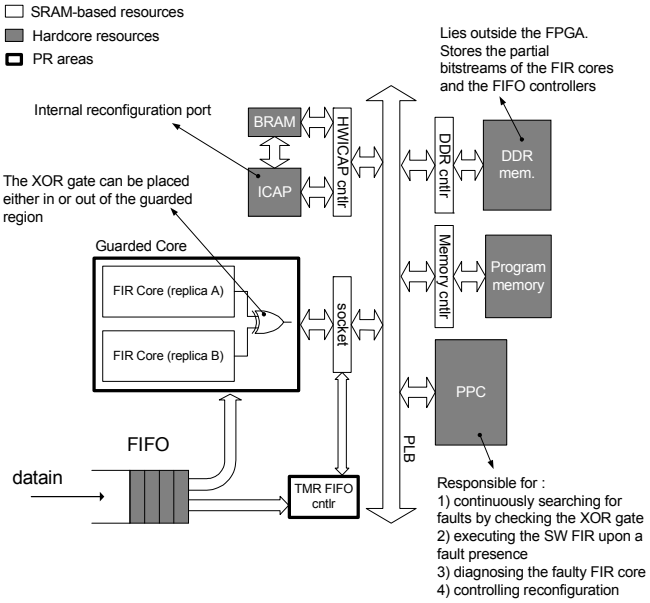

Figure 2. DWCSW scheme. Two HW cores co-exist with one SW model of the filter; the SW model is executed on the processor once a fault occurs.

During normal operation both HW filter cores process the same input. Once the PPC detects a mismatch, the processing in the HW cores is halted. Then, the PPC starts executing the SW counterpart of the filter on a specific amount of the next input data (i.e. equal to the amount of filter taps) and during this time both HW cores process simultaneously the same input data. The results are stored in registers accessible by the PPC, and the latter compares them in order to identify the HW core that disagrees with the SW model. Then, the PPC instructs the uncorrupted HW core to resume its operation, while it reconfigures the PRR carrying the corrupted HW core. Once reconfiguration is ended, the system enters its normal operation. Therefore during the reconfiguration period of the corrupted core, there is just one copy of the hardware processing the data which results in the system being unreliable for that period.

Figure 3 depicts the design of the multi-mode filter for two different LIs using DWCSW as redundancy technique. Figure 3.a shows the case where the size of static section is 0 (largest LI possible). Thus the whole filter design is composed of one large reconfigurable module which should be reconfigured across different modes of operation. The outputs of the two replicas are connected to an XOR circuitry in order to monitor any mismatches. Once a mismatch occurs, the PPC starts executing the SW model of the module to locate the corrupted module and reconfigure it. The advantage of this setup is that the overall area is small since we implemented the base design

at higher LI. However the shortcoming is its large reconfiguration overhead. Furthermore fault handling of the system becomes inefficient as the run-time of the SW model of the module becomes too large. In particular, SW run-time increases linearly with the filter size and as we consider large filters, developing the entire filter in software would be inefficient. During fault handling this would possibly cause the FIFO to overflow resulting to loss of input data. On the other hand, we can implement the base design at lower LI as shown Figure 3.b.

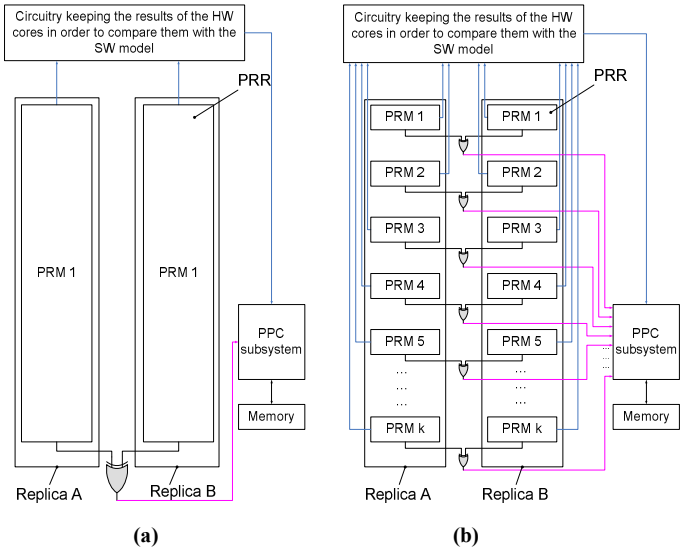

Figure 3. DWCSW implementation of the proposed multi-mode filter. Wires connecting the PRMs in (b) are omitted for clarity.

Figure 3.b presents the approach of having many small reconfigurable modules that compose the filter design. Each module is formed as a PRM lying on a different PRR. Pairs of the corresponding modules between the two filter cores are formed, and their outputs are connected to XOR circuitries in order to monitor any mismatches. Once a mismatch is detected, the PPC executes the SW model of the corresponding PRM only - and not of the entire filter as in Figure 3.a-, to locate the corrupted PRM and reconfigure it. As well reconfiguration overhead for this design is smaller. A study is needed to evaluate the demands in hardware resources (i.e. additional logic and routing) and the way the modules are implemented in software. This modification comes at the expense of physical resources such as more XOR circuitries on module pairs and additional routing to pass the module outputs to the circuitry in order to compare their result against the corresponding SW model. A previous study in [2] shows that the increase in hardware resources is super-linear for the increased number of modules. Also, the PPC memory should be programmed with the software model of each module, i.e. all PRMs of Figure 3.b should have their SW counterpart. This induces some additional area overhead but, it is insignificant considering the delay that the PPC would cause if it had to execute the entire filter.

B. TMR-based architecture

If the system is built with TMR then all the components required in the first scheme except the FIFO are mandatory. In particular, just like in the DWCSW, the circuitry for the partial reconfiguration, i.e. the PPC, the ICAP subsystem, the memory controllers, and the DDR controller are needed. In addition, a third replica of the FIR core needs to be instantiated, which is implemented as a new PRM in a separate PRR. Finally, the majority and the minority voters along with the feedback logic need to be implemented. The benefit of TMR over DWCSW is that it masks errors, thus once a mismatch occurs the two FIR cores maintain the output. During that time, the corrupted core is corrected through partial reconfiguration.

Regardless of the scheme chosen to guard the system, there is some constant resource overhead. This is the circuitry for partial reconfiguration, which also needs to be guarded. The cost of guarding this circuitry is identical for both cases, and as it does not affect the comparison of the two schemes in terms of the total area cost, we do not examine it. Moreover, it is worth noting that in the new radiation-hardened Virtex-5 devices, the reconfiguration control logic is hardened with embedded triple modular redundancy [14].

V. EXPERIMENTAL RESULTS

As part of the experiments, we have developed a tool which implements JFD technique mentioned in Section III.A. For each filter the tool optimizes the filter structure and generates a hardware implementation of the optimized structure. We implement 10 filters representing 10, 20x20 pixel image templates. Each pixel is 8 bits. Therefore, each filter has 400 coefficients (400-tap filter) and each coefficient is 8 bits wide. The filters are reconfigured in an arbitrary given sequence. It should be noted that the results of our experiments have been collected using values from actual platforms and place and route using the Xilinx PR design flow

Table 2. Area of the implemented system for different setups of redundancy technique and base design implementation

Base Design	Total System Size (Static/Reconfigurable) in # Slices			
	TMR/TMR	TMR/ DWCSW	DWCSW/ TMR	DWCSW/ DWCSW
Generic	90,438	n/a	n/a	52,947
LI0	51,979	43,096	44,190	31,886
LI3	22,940	17,694	24,651	15,984
LI4	21,159	15,267	24,322	15.008
LI5	20,727	14,772	20,727	14,772

Table 2 shows the size of the system after applying the redundancy techniques to each part separately. In this scheme we assume that we implement the entire reconfigurable or static section of the filter using TMR or DWCSW. Therefore, for example, for the third column which is TMR/DWCSW, we are implementing the static section of the filter using TMR technique while the entire reconfigurable section is implemented using DWCSW.

Table 3 shows the reconfiguration overhead for changing the mode of operation i.e adjusting the system to the pattern being searched. In Table 3, Column 2 shows the reconfiguration overhead for changing the mode when the reconfigurable section of the filter is implemented using TMR. Column 3, on the other hand shows the reconfiguration overhead, when the reconfigurable section is implemented using DWCSW. The

reconfiguration overhead of TMR is higher than DWCSW as we triplicate each reconfigurable module.

Table 3. Average reconfiguration time to change mode of operation

Base Design	Application Reconf. Time (in ms)	
	TMR	DWCSW
Generic	0	0
LI0	12.36	8.24
LI3	19.57	13.05
LI4	38.90	25.94
LI5	69.51	46.34

Table 4 shows the period of time when the system is unreliable. As mentioned earlier, if we use DWCSW to implement part of a system, during reconfiguration of the faulty module, the incoming data is processed on only one instance of the module. Thus the system would not be able to detect an error until the reconfiguration completes. This inherently means that the system is unreliable. The numbers reported in Columns 3,…, 6 of Table 4 represent the average reconfiguration time of a PRR when implemented using DWCSW.

Table 4. Reconfiguration overhead to change the mode of operation and unreliable period

Base Design	Unreliable period (in ms) (Static/Reconfigurable)				
	TMR/ TMR	TMR/ DWCSW	DWCSW/ TMR	DWCSW/ DWCSW	
				Static	Reconf.
Generic	0	n/a	n/a	101.06	
LI0	0	0.09	27.7	27.7	4.12
LI3	0	0.43	4.23	4.23	6.52
LI4	0	3.3	0.65	0.65	12.97
LI5	0	23.17	0	0	23.17

We assume TMR technique to be reliable the entire execution time. This is because during the time that one faulty module is being reconfigured, we have two instances of the module running at the same time, which makes it possible to detect an error in this period.

To assess the reconfiguration time, we assume the reconfiguration process presented in [6]. We consider the case where the partial bit-streams are stored in DDR memory. Furthermore the PPC processor is using both an I-Cache and a D-Cache to fetch and store the data required for reconfiguring the device from DDR. The processor and the memory are running at 100 MHz and 32 bits of configuration data is fetched at every memory access. This scheme gives the fastest reconfiguration time as computed in [18] as part of our experiments.

From the results shown in this section, we can observe that each setup of base design and redundancy technique exhibits a different characteristic of design in terms of area, reconfiguration time for the base design and reliability of the system. To find the optimal point of implementation considering the requirements of the system, all the data points shown in Table 2 and Table 4 should be considered. Careful analysis of the results might reveal setups that are clearly superior over some other configurations. For example, implementing LI_3 using TMR/TMR is more area efficient than using DWCSW/TMR. This is because the overhead of extra circuitry needed for DWCSW (FIFO controller, etc) results in a larger overhead than implementing using TMR. Besides these special cases in the design space, the other design points remain as viable options in designing a reliable multi-mode FIR filter. Selecting the proper setup is a system-level decision to fit the system requirements.

VI. CONCLUSIONS

In this work we proposed a framework to implement reliable multi-mode FIR filters. We showed that by changing the design parameters, a range of system designs is possible. In this spectrum of solutions, each design has a different characteristic in terms of system area, mode reconfiguration overhead and reliability of the system.

VII. REFERENCES

[1] A. Ilias, K. Papadimitriou and A. Dollas, "*Combining Duplication, Partial Reconfiguration and Software for On-line Error Diagnosis and Recovery in SRAM-based FPGAs*", FCCM 2010

[2] C. Bolchini, A. Miele and M. D. Santambrogio, "*TMR and Partial Dynamic Reconfiguration to mitigate SEU faults in FPGAs*", DFT 2007

[3] Y. Ichinomiya, S. Tanoue, M. Amagasaki, M. Iida, M. Kuga and T. Sueyoshi, "*Improving the Robustness of a Softcore Processor against SEUs by Using TMR and Partial Reconfiguration*", FCCM 2010

[4] A. H. Gholamipour, H. Eslami, A. Eltawil, F.J. Kurdahi, "*Size-Reconfiguration Delay Tradeoffs for a Class of DSP Blocks in Multi-mode Communication Systems*", FCCM 2009

[5] J. Villasenor, et al, "*Configurable Computing Solutions for Automatic Target Recognition*", FCCM 1996

[6] K. Papadimitriou, A. Anyfantis, A. Dollas, "An Effective Framework to Evaluate Dynamic Partial Reconfiguration in FPGA Systems", IEEE TIM 2010.

[7] http://www.mers.byu.edu/yinsar/index.html

[8] G. L. Charvat. "*Build a high resolution synthetic aperture radar imaging system in your backyard*," MIT Haystack Observatory, May 2010.

[9] http://teams.eas.muohio.edu/CCLIEM/radar/

[10] http://www.artemis-smart.eu/home.aspx

[11] E. Stott, P. Sedcole, P. Cheung, *"Fault Tolerant Methods for Reliability in FPGAs"*, FPL 2008.

[12] J. Johnson, W. Howes, M. Wirthlin, D. McMurtrey, M. Caffrey, P. Graham and K. Morgan, "*Using Duplication with Compare for On-line Error Detection in FPGA-based Designs*", IEEE Aerospace Conference 2008.

[13] M. Wang and G. Bolotin, "*IBM PowerPC 405 SEU Mitigation Using Processor Voting Techniques in Xilinx Virtex-II Pro FPGA*", MAPLD 2004.

[14] http://www.spacedaily.com/reports/Xilinx_Launches_First_High_Density_Rad_Hard_Reconfigurable_FPGA_For_Space_Apps_999.html

[15] S. Golshan, E. Bozorgzadeh, *"Single-Event-Upset (SEU) Awareness in FPGA Routing"*, DAC 2007

[16] Lysaght P., et al, "*Invited Paper: Enhanced Architectures, Design Methodologies and CAD Tools for Dynamic Reconfigurable of Xilinx FPGAs*", FPL 2006

[17] A. H. Gholamipour, F. Kurdahi, A. Eltawil, "*Heterogeneous Mapping to Minimize Resource Usage Under Maximal Spatial Reuse Constraints for FIR-like Structures*", CECS technical report # TR11-01

[18] PRCC, Partial Reconfiguration Cost Calculator, users.isc.tuc.gr/~kpapadimitriou/prcc.html, 2010