Adaptive Reconfigurable Voting for Enhanced Reliability in Medium-Grained Fault Tolerant Architectures

Filip Veljković, Teresa Riesgo, Eduardo de la Torre

Abstract— **The impact of SRAM-based FPGAs is constantly growing in aerospace industry despite the fact that their volatile configuration memory is highly susceptible to radiation effects. Therefore, strong fault-handling mechanisms have to be developed in order to protect the design and make it capable of fighting against both soft and permanent errors. In this paper, a fully reconfigurable medium-grained triple modular redundancy (TMR) architecture which forms part of a runtime adaptive on-board processor (OBP) is presented. Fault mitigation is extended to the voting mechanism by applying our reconfiguration methodology not only to domain replicas but also to the voter itself. The proposed approach takes advantage of adaptive configuration placement and modular property of the OBP, thus allowing on-line creation of different medium-grained TMRs and selection of their granularity level. Consequently, we are able to narrow down the fault-affected area thus making the error recovery process faster and less power consuming. The conventional hardware based voting is supported by the ICAP-based one in order to additionally strengthen the reconfigurable intermediate voting. In addition, the implementation methodology ensures using only one memory footprint for all voters and their voting adaptations thus saving storing resources in expensive rad-hard memories.**

Keywords—TMR voting, ICAP-based voting, medium-grained TMR, scalable partial reconfiguration, gcapture, on-board processor, fully reconfigurable TMR, TMR with spare, adaptive voter, soft and permanent errors

I. INTRODUCTION

Over the past decades many researchers have devoted their work to increasing the reliability of digital system design. Due to constant improvements in semiconductor industry, reflected in reduced voltage, increased operational frequency and transistor scaling that faithfully follows "Moore's Law" [1], dependability emerged as one of the key concerns when designing a digital system. Special attention has been given to safety and mission-critical applications in areas such as medicine, security, defence, aviation, spaceflight and others where a single failure could lead to injures or serious consequences considering the mission they are designed for. In space applications, where once in orbit no further maintenance is possible, it is vital that a system on its own can detect the fault, diagnose it, and self-repair in order to recover its normal operation. This becomes even more essential when it comes to SRAM-based FPGAs whose impact is constantly growing in aerospace industry despite the fact that their volatile configuration memory is highly susceptible to radiation effects. Space engineering starts taking advantage of extensive computational resources they possess, short time to market and significantly lower non-recurring engineering costs compared to their antifuse-based counterparts or predominantly used ASICs. The platform also offers high flexibility reflected in the fact that it can modify or completely change its functionality through reconfiguration. In addition, recent families of FPGAs have the possibility to be partially reconfigured during runtime [2] which is one of the main assets for their breakthrough in the aerospace market.

In harsh environments the influence of radiation on semiconductor devices represents one of the main concerns. Many of the already deployed satellites, along with the International Space Station, operate in low Earth orbit (LEO) where the Single Event Upset (SEU) rate for static RAMs exceeds 10^{-6}/bit per day according to the 10 years observation study published in [3]. In addition, for SRAM-based FPGAs, this upset rate is estimated to reach up to several upsets per hour depending on the altitude, inclination and solar weather conditions. These SEUs are provoked by high energy protons or heavy ion components of solar or galactic origins which penetrate to the substrate of a transistor causing nuclear interactions or ionization. Such phenomena evoke memory cell bitflips, which are particularly threatening to the modern FPGAs as a single alteration in the configuration memory may result in bad interconnections, invalid LUT functions, wrong flip-flop values and many other undesirable scenarios. Consequently, various methodologies and design techniques have to be developed in order to cope with this issue and satisfy different reliability requirements.

The majority of today's digital systems in space use redundancy techniques in order to mask potential errors and protect its data. As a consequence, one or several parts of a circuit are duplicated or triplicated in hardware thus forming the well-known Double or Triple Modular Redundancy (DMR and TMR) structures [4], [5]. DMR performs the comparison of the domain outputs thus being able to detect the presence of a fault. However, using a simple comparator, it is impossible to determine the affected domain. Therefore, to overcome this problem, duplication with comparison is often supported with the so-called Concurrent Error Detection (CED) [5]. On the other hand, TMR structures use a majority voter at the output

of the replicas in order to propagate the correct result. As the voter is constantly fed with three domain outputs, it can be implemented not only to detect the fault and propagate the majority-voted output, but also to recognize the faulty domain and signal it to the processor or other type of controller. Consequently, the system can use that information to trigger the reconfiguration of the affected FPGA area and thus remove the fault from the design [6]-[8]. TMR is often supported with dynamic and partial reconfiguration (DPR) in order to prevent the accumulation of transient faults and therefore extend the lifetime of the structure. Along with configuration memory scrubbing [9], [10], TMR in combination with DPR represents one of the most widely used techniques for SEU mitigation in SRAM-based FPGAs.

Although modular redundancy is based upon a simple methodology, a lot of research has been done targeting its implementation, e.g. the optimal granularity of DMR/TMR domains [4], its voting mechanism or even optimal reconfiguration time after the detection of the fault [11]. Xilinx uses different approaches for implementing TMR depending on the logic type [12]. Hence, the logic is categorized into four groups: state machine logic, throughput logic, I/O logic and special features like PLLs or DSP48s. For this purpose, Xilinx developed X-TMRTool which supports the automatic generation of triple redundancy. The triplication is done at the register level thus achieving the finest granularity in contrast to the coarse-grained TMR where the entire logic is triplicated and then voted [13]. However, this introduces additional area overhead as each register output has to be checked. In addition, such fine granularity reduces the maximum frequency of the design considering that voters have to be introduced on the critical path.

In general, the size of triplicated modules dictates both implementation and robustness of TMR architectures. The well-known fact is that the TMR protection of the system ends with errors present in more than one domain. That is often a consequence of an error accumulation due to insufficiently short scrub cycle, multiple bit upset (MBU) or routing errors which affect two or more domains. Furthermore, a voter itself represents a single point of failure. In other words, a fault in the voting part of the circuit may cause wrong system behavior and produce incorrect output data. Hence, the voting mechanism is also often triplicated thus introducing additional area overhead in already expensive architecture. For systems with restricted FPGA utilization area and limited power consumption additional redundancy costs may be intolerable. Therefore, recent research on this topic is focused on finding an optimal trade-off between the implementation costs and system's reliability.

In this paper we present an adaptive reconfigurable voting mechanism for both medium- and coarse-grained fault tolerant architectures. One of the main goals was to extend the DPR-supported SEU mitigation to voting partitions, which are usually the weakest points of TMR architectures. Our reconfigurable voter is capable of adapting to different types and positions of fault tolerant structures during runtime. Each adaptation is done directly through the internal configuration access port (ICAP) by reconfiguring only one frame which corresponds to LUTs, thus making possible to use a single and reduced memory footprint for all variants of voting. Moreover, that particular footprint is used for the configuration of intermediate voters in a medium-grained TMR architecture where each domain is comprised of several scalable stages. Consequently, a fully reconfigurable fault tolerant architecture is achieved giving the opportunity to correct the faults in both voter and TMR domains.

Apart from the traditional voting performed in hardware, the reconfigurable design is supported by an ICAP-based voting. More precisely, captured flip-flop values corresponding to the outputs of each stage, and stored in the configuration memory, are periodically read through the ICAP and compared by the processor. This new feature allows additional securing of the conventional hardware-based voting process. Moreover, in fault tolerant systems using so-called warm redundancy, where it is allowed to lose several clock cycles of data before correcting the fault, it can be used as the only voting mechanism thus permitting a complete exclusion of voters from hardware.

Our methodology is applied to a digital video broadcast on-board processor (DVB-OBP), used in a satellite communications application, which is able to self-adapt to harsh environmental conditions trading-off performance for fault tolerance. This adaptation is performed during runtime by duplicating or triplicating the essential part of the design. As a result, the voter itself has to be able to adapt to different types and positions of configured fault tolerant structures. Moreover, we take benefit of the modular property of the design and partition the scalable reconfigurable architecture in order to create medium-grained TMR. Consequently, we narrow down the affected area and using the information obtained by the intermediate voters, reconfigure only part of a domain in order to repair the fault. Therefore, the proposed method reduces the time needed to correct the error thus making the recovery process faster and less power consuming. In addition, the employed reconfiguration methodology does not use Xilinx DPR flow presented in [2], being able to create completely isolated TMR domains using relocatable partial configurations. Our design methodology can be applied to any system with modular properties.

The rest of the paper is organized as follows. In section 2, we present the overview of the related work focusing on medium-grained TMR architectures and give the motivation for our work. Section 3 introduces the base for our current research expressed in a runtime reconfigurable OBP capable of adapting to different operational demands and environmental conditions. The main contribution of this paper is presented in section 4. A complete implementation process along with the voting mechanism is described and the entire configuration comprised of several scalable stages is analyzed. Obtained results are summarized and discussed in section 5. Finally, section 6 gives perspectives and conclusion of this paper.

II. Related Work

The most common SEU mitigation is configuration memory scrubbing which is present in the literature for more than 20 years. It implies periodical rewriting of configuration memory with a golden copy stored in a rad-hard memory. It is performed either independently of the occurrence of an error or

when there is a mismatch between the readback of the configuration memory and the golden copy. Consequently, we can distinguish between blind scrubbing and scrubbing with readback [6]. However, these techniques introduce large overhead in terms of reconfiguration time, increase system's power consumption and often require freezing the device during the recovery process. In order to reduce these expenses, modern robust designs include redundancy-based techniques supported by DPR.

The main concept of the protection based on modular redundancy is presented in Fig. 1. The idea is to implement two or more copies of the same circuit which perform the same task in parallel. At the output of the circuits a simple comparator is implemented in order to detect and mask possible fault occurrences. Today's fault tolerant systems are mostly based on TMR performed at the register level following the concept used in X-TMRTool [12]. Therefore, the robustness is significantly increased as voting is performed over the smallest possible modules. However, such fine granularity introduces additional area overhead in an architecture that already uses 200% more logic than necessary. In addition, high number of voting instances significantly decreases the maximum frequency of a design.

On the other hand, standard coarse-grained TMR architectures [5], [13], [14] use only one voter partition in order to propagate the correct result as shown in Fig. 1b. As a voter can also be an SEU target, designers triplicate the instance and therefore add more redundant components to the system. When a fault is detected, the recovery performed through DPR takes more time in order to reconfigure large, fault-affected modules. Therefore, since the DPR process normally increases power consumption, longer reconfiguration causes higher overall energy costs. In addition, occurrence of faults in two large domains results in erroneous data at the output of the voter and completely impairs the structure.

In order to cope with these issues, current research is directed towards an optimal implementation of redundancy-based fault tolerance. As a result, Wang in [15] performs a comprehensive study on the relation between robustness and TMR granularity. Various alternatives of partitioning the TMR are analyzed and an optimal trade-off between costs and reliability is achieved applying medium granularity. A similar research is published in [16] where a cascaded TMR is implemented by inserting alternative number of triplicated voters between the partitions. A quality comparison of a coarse- vs. medium-grained TMR robustness is presented in [17]. Authors perform partitioning in shared Wishbone architectures and categorize injected errors using fault counters for each of the domains. They present better fault mitigation results when finer, medium-grained TMR is employed. Bolchini et al. proposed TMR applications at different granularity levels ranging from system to component level [7], [18]. They support the fault mitigation using DPR of the fault-affected domains. By voting smaller TMR domains and making them independent one from each other, they achieved more than 80% of reduction in terms of average reconfiguration time when a fault is detected comparing to the typical scrubbing with readback.

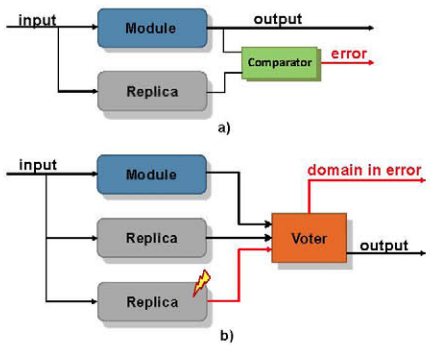

Fig. 1 a) Double Modular Redundancy - DMR; b) Triple Modular Redundancy – TMR with error localization

In this work we apply the partitioning methodology to a part of an OBP. Taking advantage of our relocatable partial configurations, the granularity and positon of the configured fault tolerant structures can be modified during runtime and on-demand. In order to make it possible, we introduce reconfigurable voters which are placed between domain partitions during runtime. These reconfigurable voters are supported by error counters corresponding to each of the configured domains. Moreover, the voting process is additionally strengthened using the ICAP-based voting [19]. It goes directly to the present state of the configuration memory and compares the predetermined FF values that correspond to the outputs of each partition in every domain. Implementing fully reconfigurable redundancy structures able to change from coarse to medium granularities during runtime we increase the robustness of an entire design and extend their lifetime. In addition, using only one memory footprint for each partition of the fault tolerant structures we can achieve significant savings in terms of resources in expensive rad-hard memories.

III. Runtime Adaptive Demultiplexer Architecure

In this chapter the base for our current research is presented summoning up the previously conducted work. It is represented in terms of a reconfigurable DEMUX which forms part of an OBP capable of adapting to different environmental conditions during runtime [19]. Two different reconfiguration strategies are applied including conventional and scalable DPR. Block diagram of the static part of the design is presented in Fig. 2, whereas the reconfigurable parts for both, conventional and scalable DPR strategies are shown in Fig. 3a and Fig. 3b, respectively. The reconfigurable DEMUX takes the 10-bit input signal coming from an analog-to-digital converter and performs series of complex operations in each of the 4 reconfigurable sub-bands (SBs) thus creating carriers of different frequencies. These carriers are later used in a Demodulator-Decoder (DEMDEC) part of the OBP for the creation of MPEG-2 packets following the DVB standard.

As shown in Fig. 2, the static part of the reconfigurable DEMUX is comprised of a *Block10*, zone selector, an adaptive voter and output interface. For simplicity of the block diagram, a series of smaller modules which perform various different operations are represented as *Block10*. The outputs of the block, SB1 to SB4, represent a set composed of 8 MHz carriers and material for the creation of lower frequency carriers in each reconfigurable SB. Therefore, 8MHz carriers are created

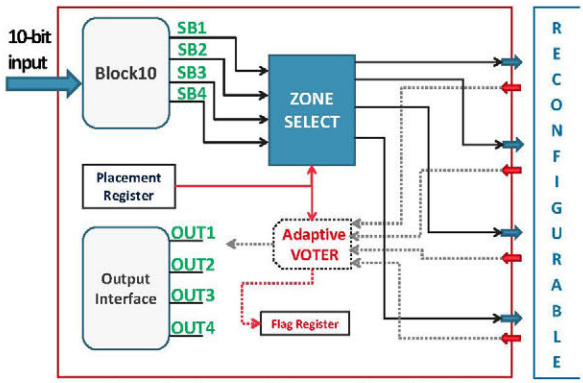

Fig. 2 Static part of the reconfigurable DEMUX

in the static part, whereas the rest of the carriers of lower frequencies are created in the reconfigurable part of the design. In spite of this, the 8 MHz carriers of all 4 SBs are conducted to the reconfigurable part. Consequently, the design is able to create and output only the demanded carriers at each point of operation.

A special block, in Fig. 2 labeled as *Zone Select*, is in charge of dispatching the *Block10* outputs to the corresponding reconfiguration zone using the information stored in a processor accessible *Placement* register. Hence, a complete flexibility in SB placement is achieved. Taking advantage of it, the design is capable of creating different fault tolerant structures in the reconfigurable portion of the chip during runtime. This is achieved by sending the data, corresponding to only one SB, to several reconfiguration zones and by reconfiguring these areas with the same bitstream. Consequently, the configured hardware will create carriers of the same frequency and return them back to the static part of the design. Depending on the number of zones that receive the same inputs, the design can configure dual or triple redundancy. In addition, since there are 4 reconfiguration zones, a SB can be triplicated in 4 different positions in the chip. As a result, the system is able to remove a configuration from a zone struck by a hard error and configure it in a spare one thus making it capable of fighting not only soft errors, but also permanent ones by simply changing the position of a configured TMR structure. This configuration is present in the literature as *TMR-with-spare* [20].

The carriers coming from the reconfigurable zones are taken by a voter which, depending on the present value of the placement register, adapts to different types and positions of configured redundancies. It takes the outputs coming from all four zones, compares (DMR), votes (TMR) and finally forwards them to the output interface. Any detected error in the voting process is stored in the flag register setting one of the 4 bits corresponding to the area that has been affected. A bit set in the flag register triggers the reconfiguration of the affected domain. Therefore, the system is able to self-recover from an SEU appeared as a bitflip in the configuration memory.

Two DPR strategies applied to the architecture are presented in Fig. 3. The first one is the conventional reconfiguration where one module is substituted by another at each change of demand. An example configuration, where 0.5, 1, 2 and 4 MHz are demanded from the SBs, is presented in

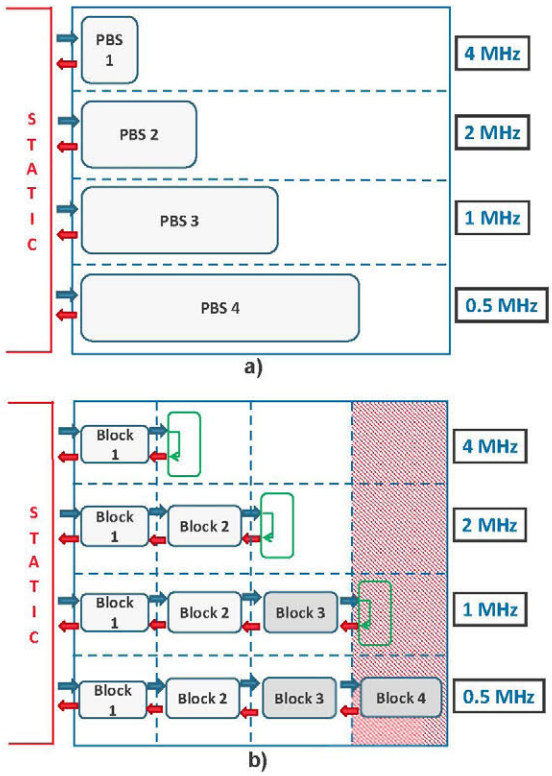

Fig. 3 Reconfigurable part of the DEMUX architecture: a) Conventional DPR strategy; b) Scalable DPR strategy

Fig. 3a. In order to reconfigure less area and such obtain a faster and less power consuming DPR process, we take advantage of the modular property of the SB logic, partition the SB and create runtime scalable partial configurations. The same configuration example for scalable DPR is presented in Fig. 3b. It can be noted that when using the first approach, 4 reconfiguration zones are available, corresponding to 4 SBs. On the other hand, when using scalable configurations, each of the SBs is additionally partitioned thus creating 3 sub-zones. Due to the lack of resources in the current chip, the 4th block cannot be configured. Therefore, when 0.5 MHz carriers are demanded from a SB in the second approach, a partially scalable configuration containing blocks 2, 3 and 4 together is connected to the *Block1*. Using scalable configurations, the obtained savings reach up to 43% in terms of storage resources in an expensive rad-hard memory. Moreover, we made an important step towards the possibility to further increase the fault tolerance of the design.

IV. Fully Reconfigurable Fault-Tolerance

A vast majority of today's fault tolerant structures employed in SRAM-based FPGAs use the redundancy approach. However, not as many support it with DPR to mitigate the occurring faults and stop their accumulation. Those designs that do use DPR often focus only on redundancy domains thus completely neglecting the importance of the voter. More precisely, when an SEU appears and gets detected in one of the replicated modules, the system triggers the reconfiguration process and reconfigures the logic that corresponds to the affected domain. However, when a fault appears in a voting partition it completely destroys the

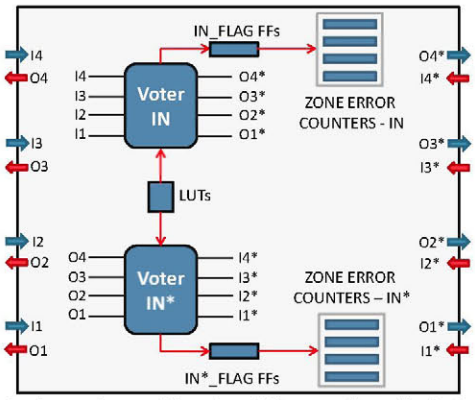

Fig. 4 Adaptive voting partition for a fully reconfigurable fault tolerance

structure. A solution which is often found in the literature implies triplication of the voter as well. That might be too much for a system with strict area and power constraints or design in which triplicated modules possess huge number of signals that have to be voted. Moreover, even if triplicated, a voter losing one of its replicas stops being properly secured since any subsequent error in the two remaining replicas would corrupt the entire composition. Therefore, our approach takes a step further and introduces adaptive reconfigurable voting which brings the opportunity to correct not only faults present in domain replicas but also those which appear in the voting partition.

A. Runtime Adaptive Reconfigurable Voting Partition

In order to enable reconfigurable voting for our fault tolerant structures a voting partition comprised of two voter units is implemented and shown in Fig. 4. One voter takes the SB material coming from the static part of the design, whereas the other one compares the created carriers coming from reconfigurable modules. They are able to adapt to both non-redundancy and redundancy modes of operation. In the former case, voters perform a simple propagation of the inputs. When a SB is duplicated it adapts to the DMR position, compares the inputs coming from the duplicated logic and, if equal propagates them along with the 2 remaining inputs. On the other hand when a SB is triplicated, voters adapt to the TMR position and act as majority voters. The remaining input is simply propagated to the output. It is important to note that in this composition, the voter, as well as the entire structure, is constantly in a *TMR-with-spare* mode. Therefore, if a permanent error appears in one of the reconfiguration zones, the system can stop using the affected part of the hardware and move the logic to the spare zone. Consequently, the system is able to mitigate not only soft errors in the configuration memory but also permanent errors which cannot be repaired via DPR. The price that has to be paid in that case is the inability to configure another SB along with the triplicated one.

The adaptation is performed through ICAP by modifying only 4 bits that control the outputs of 4 implemented LUTs. These LUTs are in charge of controlling the voters' configuration so that the voting can adapt to different types and positions of the configured structures. They are placed in a single slice and configured such that the bits corresponding to their outputs reside in only one CLB frame. Therefore, a faster adaptation is achieved since only one frame needs to be reconfigured. Apart from the control LUTs, two 4-bit flag registers have been implemented to indicate the presence of the fault in one of the reconfigurable domains. If a mismatch occurs in the voting process a bit corresponding to the affected zone is set high. In addition, eight counters, four per voter, are implemented in order to count possible faults in 4 different branches.

B. Implementation methodology

When the Xilinx DPR flow is not used, the implementation of reconfigurable partitions requires slightly different approach. In order to design completely independent and therefore relocatable modules, each partial configuration is treated as a separated module. Once the logic is synthesized, it is floorplanned in PlanAhead such that occupies the least possible area on chip. After the implementation step, a *Netlist Circuit Description* (NCD) file, which is a Xilinx proprietary binary format to internally describe the implemented design, is created. In order to contain the routing within strict boundaries on chip, a user can open the file using FPGA Editor and try to reroute the conflicting nets. Although helpful when compact configuration is not a must, the lack of routing flexibility implies searching for other solutions in applications with limited FPGA resources.

The first step in our design methodology implies converting the NCD file to *Xilinx Description Language* (XDL) [21] using the *-ncd2xdl* command in Command Prompt of the ISE Design Suite. Therefore, a human-readable equivalent version of the NCD is created which allows design modifications exploiting an open source java based set of RapidSmith [22], [23]. We modified the RapidSmith based router published in [24] and such created our own rerouter which contains all nets of the design within the predetermined area on chip. When the XDL file is repaired by the rerouter, it is reconverted to NCD using the - *xdl2ncd* command. Consequently, a programming file is generated containing a full configuration of the FPGA. Since only part of it is useful information, we cut the part of the programming file corresponding to the portion of FPGA area where the logic is implemented and store it as a partial configuration file in our external RAM memory. Finally, we obtain the smallest possible memory footprint and on-chip relocatable partial configuration.

Each partial configuration in our design is implemented following the above set of steps. Therefore, compact, reliable and, most importantly, relocatable partial configurations are obtained. Consequently, they can be used in any other part of the chip with the same architecture. This implementation methodology is also applied to the voting partition thus enabling flexible voter insertion for medium-grained fault tolerant structures.

C. Scalable Medium-Grained Fault Tolerant Structures

The implementation of the runtime adaptive voting partition allows the employment of fully reconfigurable TMR structures. Such configuration is presented in Fig. 5 where the 3rd SB is configured along with the triplicated 2nd SB. In this example configuration, 1 MHz carriers are demanded from the TMR-protected 2nd SB, whereas 8 MHz carriers are demanded from the 3rd one. When using conventional partial configurations for the creation of fault tolerant structures, only

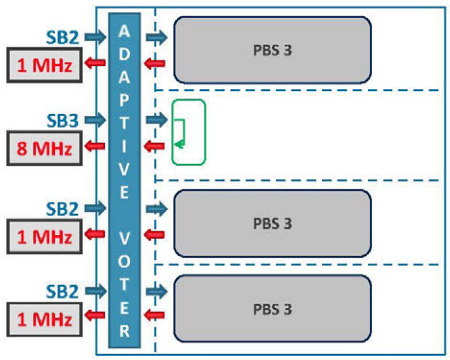

Fig. 5 An example of a coarse-grained TMR configuration – Demanded 1 MHz carriers from the 2nd SB and 8 MHz carriers from the 3rd SB

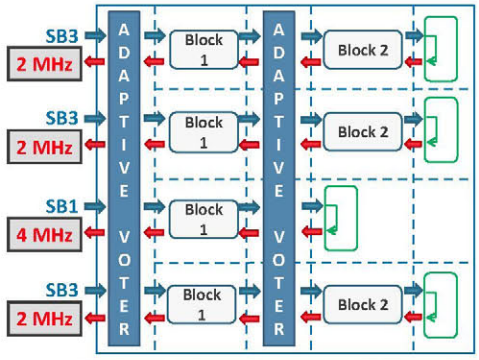

Fig. 6 An example of a scalable medium-grained TMR configuration – Demanded 2 MHz carriers from the 3rd SB and 4 MHz carriers from the 1st SB

one reconfigurable voter is used to vote and propagate the error free carriers to the static part of the design. It is able to mask errors present in one of the TMR domains. Moreover, a possible SEU affecting the voting partition can be eliminated simply by reconfiguring its on-chip reconfiguration zone. However, the drawback of such configuration is still the same as in all coarse-grained fault tolerant structures. It becomes evident when errors accumulate or a MBU appears affecting two TMR replicas. We prevent the accumulation by instantly reconfiguring the domain in error, i.e. when a bit of the flag register signals a mismatch in a voting process. However, due to large size of the replicas, the probability of MBU corrupting two domains is high. Therefore, we take benefit of our scalable partial configurations and place in between our adaptive voter partition thus making possible the on-demand configuration of medium-grained DMR/TMR structures.

Using smaller modules for triplication and voting their outputs within the intersections increases the overall robustness of the design. More precisely, the probability of two errors affecting two different domains decreases as the same, smaller block has to be struck by an error in both of the replicas. An example of a medium-grained TMR configuration is presented in Fig. 6. The 3rd SB which creates 2 MHz carriers is triplicated and configured along with the 1st SB which creates 4 MHz carriers. As can be seen, two voting partitions are placed between the static part and *Block1* and between *Block1* and *Block2*. These blocks are in charge of creating 4 and 2 MHz carriers, respectively, along with the material for lower frequency carriers creation used in the subsequent blocks. The 4 MHz carriers are discarded in the 3rd SB, whereas in the 1st SB they are propagated through the 2nd voting partition and returned back to the static part using a simple cover.

The advantage of using medium-grained redundancy over coarse-grained one can be explained using this example configuration. If two faults occur in two replicas of the 3rd SB, where one of them affects *Block1* and the other *Block2*, the structure will still be able to propagate correct data back to the output interface. Moreover, an instant repair of the blocks through reconfiguration will be triggered thus extending the lifetime of the structure. On the other hand, if the traditional partial configurations are used for the presented composition, the occurrence of faults in two *PBS 2* configurations would corrupt the entire structure. In addition, when only one domain is affected, the recovery process takes longer since larger domain needs to be reconfigured.

The presented medium-grained redundancy can tolerate error accumulation and MBUs to a higher extend. Moreover, the design is capable of changing the granularity level on demand. Therefore, it is allowed to make the decision online on whether to triplicate smaller or larger modules. Furthermore, when using the scalable modules, DMR and TMR structures can grow and decrease during runtime at each change of demand from a triplicated SB. This is achieved by reconfiguring smaller reconfiguration zones (Fig. 6) which implies faster and less power consuming adaptation.

In both coarse- and medium-grained redundancies, a soft error affecting the voting partition can be mitigated by reconfiguring the affected area. In order to further protect the voting process we support it with the ICAP-based voting which enters the configuration memory and compares the actual FF values corresponding to the domain outputs.

D. The ICAP-based voting: A support for the conventional hardware-based voting

In order to additionally strengthen the voting process, we go directly to the present state of the configuration memory and compare the predetermined FF values that correspond to the outputs of each partition used in TMR/DMR compositions. To do so, during the floorplanning step in our implementation methodology, using LOC and BEL constraints we fix the registers corresponding to the outputs of the module to particular slices and FFs within those slices. In this way the bits corresponding to their initial values (INIT0/INIT1) in the configuration memory reside within only one *CLB frame* in order to accelerate the voting process. This is done for all partial configurations, including the voter partition where, instead of the outputs, flag registers and error counters are constrained. Consequently, using the logic allocation file (*.ll), which can be created together with the programing file (*.bit), we can locate the constrained FF bits within a frame.

When a particular module is triplicated, the location of the bits in the configuration memory that correspond to the outputs of DMR/TMR domains is known. Although the domains belong to different clock regions, they reside within the same frame and have the same position within that frame. Taking advantage of it, the *GCAPTURE* command is periodically issued to the ICAP using our modified enhanced HWICAP ([19], [25]) thus capturing the present state of the configuration memory. Hence, the actual *complement* value of the outputs is

written to their INIT0/INIT1 bits in the configuration memory. Consequently, a readback of the CLB column is performed where the output FFs are placed. We repeat the process for each of the three TMR replicas and extract the bits of interest from the read configuration. By placing them in only one, 31st frame of the 36 frame long readback, a significantly faster extraction process is achieved. Hence, the floorplanning has to be performed carefully. The extracted bits are then compared by a processor and if there is a mismatch, two different scenarios are possible.

The first scenario takes into account the priority of the conventional hardware-based voter, i.e. verifies whether the mismatch is recognized by the voter checking the INIT0/INIT1 bits corresponding to the flag and error counter registers. Therefore, an error present in the voting partition can be detected. In the second one, the ICAP-based voting takes precedence and by changing the LUT function through reconfiguration directs the carriers from the non-corrupted domain to the output. The second scenario can be used either to detect the error that occurred in the voting partition or to completely exclude the voter from hardware in applications where it is allowed to lose several clock cycles of data before masking or correcting the fault. In the latter case, multiple input signature registers (MISRs) could be implemented to record the data coming from each of the domains between two consecutive captures.

V. Results and Discussions

In order to evaluate the proposed architecture, the entire system is implemented on a Xilinx Virtex-5 XC5VFX130T FPGA. The implementation of the static part of the design, along with all partial configurations, is done in ISE Design Suite 14.2 following the methodology presented in this paper. These implemented partitions are presented in Fig. 7. The results in terms of device utilization and size in memory for the voting partition and partial modules used for triplication are given in Table I. We can scale our runtime adaptable DMR and TMR structures in area such that create carriers in the range from 8 MHz to 2 MHz. The increased reliability of the system is achieved at an increased cost in terms of the on-chip area overhead. In the use case, we lost the opportunity to go up to 1 MHz due to the limitations in terms of the FPGA resources. Nevertheless, the same principle can be used in every design with modular properties where reliability is the main concern.

A real case implementation of the example composition shown in Fig. 6 is presented in Fig. 7. Each module occupies the height of 2 clock regions. As presented, the voting partition, put in between the blocks, has the height of 8 clock regions corresponding to the height of 4 SBs. Its logic is implemented within one CLB column. The fault injection is performed in the right part of the chip where the structures are configured during runtime. In order to accelerate the process the faults are injected in frames related to the used LUTs to ensure that it will affect the operation of the design. As expected, when traditional partial configurations are used to configure the TMR structures, the injection of two errors in different reconfiguration zones provided erroneous data at the output. When injecting the faults in the same positions in the medium-grained TMR architectures, those faults that affected

Table I.
The Device Utilization and Size of the Memory Footprints of the Partial Configurations used in Fully Reconfigurable Fault Tolerant Structures

*	Voting Partition	PBS 1	PBS 2	Block 1	Block 2	Cover
Occ. Slices	253 (1%)	280 (1%)	661 (3%)	333 (1%)	424 (2%)	8 (1%)
Size [KB]	177	132	211	88	120	12

Table II.
The Reconfiguration Time for each of the Partial Configurations used in Fully Reconfigurable Fault Tolerant Structures

*	Voting Partition	PBS 1	PBS 2	Block 1	Block 2
Recovery time [μs]	119	103	192	60	90

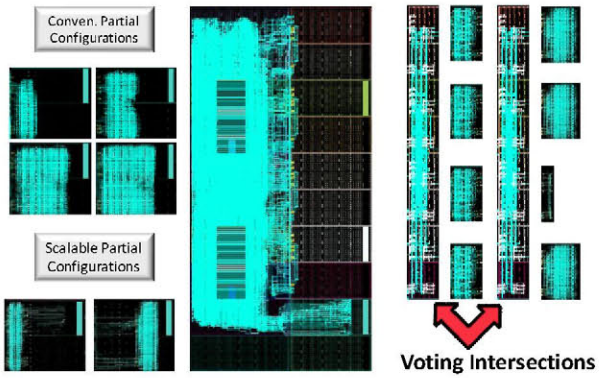

Fig. 7 The implemented partial configurations, the static part of the design and a real case of the example composition presented in Fig. 6

two domains, where Block1 is affected in one and Block2 in the other, were masked by our reconfigurable voting intersections. As a result, fault-free data was detected at the output. The recovery is determined by the time it takes to reconfigure the affected domain. The time necessary for the reconfiguration of each of the modules is presented in Table II. When faults are injected in the composition presented in Fig. 7, the recovery times are 60 μs and 80 μs depending on the block that is affected. On the other hand, when the faults are injected in the same locations in its large-grained counterpart, which uses the conventional partial configuration PBS2, the recovery time exceeds 192 μs. Therefore, by narrowing down the fault affected area in the use case, we achieved the reduction in recovery time that ranges from 58% to 69%. Moreover, the size in rad-hard memory of the partial modules given in Table I also favors medium-grained structures as fewer resources have to be reserved for storing their domain configuration. In the case when the injected fault affects the voting partition, the ICAP-based voting is in charge of detecting the fault and triggering the reconfiguration process. The recovery is determined by several tasks: GCAPTURE command to the ICAP; readback of a CLB column; check the INIT0/INIT1 bits that correspond to the flag registers; and reconfiguration of the partition. Therefore, the design is able to detect and

consequently mitigate errors present in the voting partition at the cost of losing several clock cycles of the operation.

In warm redundancy applications, the voter could be completely excluded from hardware leaving the voting partition only with a set of MISRs to record the output changes between two periodical issues of the ICAP-based voting. However, the optimal capture frequency should be calculated considering the environmental conditions, requirements and sensitivity of the design.

VI. Conclusions

In this paper we have presented a reconfigurable voting mechanism able to adapt to different modes of operation and different types of configured fault tolerant structures during runtime. Taking advantage of the implementation methodology, one memory footprint can be used for the insertion of voting partitions between the scalable modules thus creating fully reconfigurable medium-grained redundancy structures. The design can take advantage of both coarse- and medium-grained redundancies using conventional and scalable partial configurations. Results show that when the voting is performed over smaller, scalable modules the robustness of the entire design is increased. Moreover, significant savings in terms of recovery time are obtained using the medium-grained composition. The performed fault injection provided expected results, also favoring the finer granularity. Nevertheless, certain performance capabilities are lost due to the partitioning thus directly pointing to the price that has to be paid when increasing the reliability by decreasing the granularity level. The methodology performed in our runtime adaptable DVB-OBP can be applied to any other design with modular properties.

Acknowledgments

This work was supported in part by Thales Alenia Space Spain. We gratefully acknowledge the provided equipment.

References

[1] Moore, G.E., "Cramming More Components Onto Integrated Circuits," Proceedings of the IEEE , vol.86, no.1, pp.82,85, Jan. 1998 doi: 10.1109/JPROC.1998.658762

[2] Xilinx Inc., Partial Reconfiguration User Guide, UG702 (v14.5) April 26, 2013

[3] Underwood, C.I., "The single-event-effect behaviour of commercial-off-the-shelf memory devices-A decade in low-Earth orbit," Nuclear Science, IEEE Transactions on , vol.45, no.3, pp.1450,1457, Jun 1998

[4] Kastensmidt, F.L.; Sterpone, L.; Carro, L.; Reorda, M.S., "On the optimal design of triple modular redundancy logic for SRAM-based FPGAs," Design, Automation and Test in Europe, 2005. Proceedings , vol., no., pp.1290,1295 Vol. 2, 7-11 March 2005

[5] de Lima Kastensmidt, F.G.; Neuberger, G.; Hentschke, R.F.; Carro, L.; Reis, R., "Designing fault-tolerant techniques for SRAM-based FPGAs," Design & Test of Computers, IEEE , vol.21, no.6, pp.552,562, Nov.-Dec. 2004

[6] Heiner, J.; Sellers, B.; Wirthlin, M.; Kalb, J., "FPGA partial reconfiguration via configuration scrubbing," Field Programmable Logic and Applications, 2009. FPL 2009. International Conference on , vol., no.,pp.99,104,FPL2009

[7] C. Bolchini, A. Miele, M. D. Santambrogio, "TMR and Partial dynamic Reconguration to mitigate SEU faults in FPGAs", 22nd IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems, DFT, 2007.

[8] A. Jacobs, G. Cieslewski, A. D. George, A. Gordon-Ross, und H. Lam, „Reconfigurable Fault Tolerance: A Comprehensive Framework for Reliable and Adaptive FPGA-Based Space Computing", ACM Trans. Reconfigurable Technol. Syst., Bd. 5, Nr. 4, S. 21:1–21:30, Dez. 2012.

[9] Ostler, P.S.; Caffrey, M.P.; Gibelyou, D.S.; Graham, P.S.; Morgan, K.S.; Pratt, B.H.; Quinn, H.M.; Wirthlin, M.J., "SRAM FPGA Reliability Analysis for Harsh Radiation Environments," Nuclear Science, IEEE Transactions on , vol.56, no.6, pp.3519,3526, Dec. 2009

[10] Sari, A.; Psarakis, M., "Scrubbing-based SEU mitigation approach for Systems-on-Programmable-Chips," Field-Programmable Technology (FPT), 2011 International Conference on , vol., no., pp.1,8, 12-14 Dec. 2011

[11] Sterpone, L.; Ullah, A., "On the optimal reconfiguration times for TMR circuits on SRAM based FPGAs," Adaptive Hardware and Systems (AHS), 2013 NASA/ESA Conference on , vol., no., pp.9,14, 24-27 June 2013

[12] Carmichael, "Triple module redundancy design techniques for Virtex FPGAs," In Xilinx Application Notes XAPP197 July 2006.

[13] Schweizer, T.; Schlicker, P.; Eisenhardt, S.; Kuhn, T.; Rosenstiel, W., "Low-Cost TMR for Fault-Tolerance on Coarse-Grained Reconfigurable Architectures," Reconfigurable Computing and FPGAs (ReConFig), 2011 International Conference on , vol., no., pp.135,140, Nov. 30 2011-Dec. 2 2011

[14] Legat, U.; Biasizzo, A.; Novak, F., "Self-reparable system on FPGA for single event upset recovery," Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC), 2011 6th International Workshop on , vol., no., pp.1,6, 20-22 June 2011

[15] Xin Wang, "Partitioning Triple Modular Redundancy for Single Event Upset Mitigation in FPGA," E-Product E-Service and E-Entertainment (ICEEE), 2010 International Conference on , vol., no., pp.1,4, 7-9 Nov. 2010

[16] Anwer, J.; Platzner, M.; Meisner, S., "FPGA Redundancy Configurations: An Automated Design Space Exploration," Parallel & Distributed Processing Symposium Workshops (IPDPSW), 2014 IEEE International , vol., no., pp.275,280, 19-23 May 2014

[17] Kretzschmar, U.; Astarloa, A.; Lazaro, J.; Garay, M.; Del Ser, J., "Robustness of different TMR granularities in shared wishbone architectures on SRAM FPGA," Reconfigurable Computing and FPGAs (ReConFig), 2012 International Conference on , vol., no., pp.1,6, 5-7 Dec. 2012

[18] Bolchini, C.; Miele, A.; Sandionigi, C., "A Novel Design Methodology for Implementing Reliability-Aware Systems on SRAM-Based FPGAs," Computers, IEEE Transactions on , vol.60, no.12, pp.1744,1758, Dec. 2011

[19] Veljkovic, F.; Riesgo, T.; de la Torre, E.; Regada, R.; Berrojo, L., "A run time adaptive architecture to trade-off performance for fault tolerance applied to a DVB on-board processor," Adaptive Hardware and Systems (AHS), 2014 NASA/ESA Conference on , vol., no., pp.143,150, 14-17 July 2014

[20] Navas, B.; Oberg, J.; Sander, I., "The upset-fault-observer: A concept for self-healing adaptive fault tolerance," Adaptive Hardware and Systems (AHS), 2014 NASA/ESA Conference on , vol., no., pp.89,96, 14-17 July 2014

[21] C. Beckhoff,D. Koch, and J. Torresen,"The Xilinx Design Language (XDL): Tutorial and use cases," in ReCoSoC 2011 6th International Workshop, 2011

[22] http://rapidsmith.sourceforge.net/

[23] C.Lavin, M. Padilla, J. Lamprecht, P. Lundrigan, B. Nelson and B. Hutchings, "RapidSmith: Do-It-Yourself CAD Tools for Xilinx FPGAs,"inFPL'11, Greece, 2011

[24] Wei He; Otero, A.; de la Torre, E.; Riesgo, T., "Automatic generation of identical routing pairs for FPGA implemented DPL logic," Reconfigurable Computing and FPGAs (ReConFig), 2012 International Conference on , vol., no., pp.1,6, 5-7 Dec. 2012

[25] Otero, A.; Morales-Cas, A.; Portilla, J.; de la Torre, E.; Riesgo, T., "A Modular Peripheral to Support Self-Reconfiguration in SoCs,"Digital System Design: Architectures, Methods and Tools (DSD), 2010 13th Euromicro Conference on, vol., no., pp.88,95, 1-3 Sept. 2010