A Run Time Adaptive Architecture to trade-off Performance for Fault Tolerance applied to a DVB On-Board Processor

Filip Veljković, Teresa Riesgo, Eduardo de la Torre

Raúl Regada, Luis Berrojo

Abstract— **Reliability is one of the key issues in space applications. Although highly flexible and generally less expensive than predominantly used ASICs, SRAM-based FPGAs are very susceptible to radiation effects. Hence, various fault tolerant techniques have to be applied in order to handle faults and protect the design. This paper presents a reconfigurable on-board processor capable of run-time adaptation to harsh environmental conditions and different functional demands. Run-time reconfigurability is achieved applying two different reconfiguration methodologies. We propose a novel self-reconfigurable architecture able to on demand duplicate or triplicate part of the design in order to form DMR and TMR structures. Moreover, we introduce two different approaches for voting the correct output. The first one is a traditional voter that adapts to different DMR/TMR domain positions whereas the second implies comparing the captured flip-flop values directly from the configuration memory read through ICAP. The comparison is done periodically by an embedded processor thus completely excluding the voting mechanism in hardware. The proposed run-time reconfiguration methodology provides savings in terms of device utilization, reconfiguration time, power consumption and significant reductions in the amount of rad-hard memory used by partial configurations.**

Keywords—fault tolerance, ICAP, duplex, TMR, voting, scalability, run-time partial reconfiguration, FPGAs, DVB-OBP

I. Introduction

In recent years FPGAs are progressively getting involved in space system applications. Space engineering turns to the platform as it provides high performance and high flexibility at limited costs. In contrast to Application-Specific Integrated Circuits (ASICs) or antifuse-based FPGAs, used in the majority of space digital systems, non-recurring engineering (NRE) costs are significantly lower. Moreover, design and re-spin processes of an ASIC-based system take several months of schedule time which can considerably increase the cost or even risk the entire project. On the other hand, SRAM-based FPGAs, with relatively short design cycle, offer themselves as highly convenient platforms. Modern FPGAs possess a large number of gates that make possible the implementation of a complete system in a single device. They also offer the possibility to be reprogrammed such allowing further changes and modifications of the original design. In addition, apart from the complete reconfiguration, SRAM-based FPGAs can be reconfigured partially without interrupting the operation of the rest of the design, even self-reconfiguring. This opens a whole new opportunity for space applications as a system can be reconfigured during the flight as many times as necessary. Systems may not only change their functionality during run-time, but also be able to save unnecessary portions of the FPGA area thus obtaining savings in terms of device utilization and power consumption which is one of the key issues in space systems.

Nevertheless, in space applications, there are unique environmental challenges that need to be accounted for. Despite high performance, flexibility and low design costs, volatile nature of SRAM-based FPGAs makes them highly susceptible to radiation effects. During a space flight, a charged particle may enter the substrate of a MOSFET transistor and provoke ionization. Due to low power supply and presence of configuration memory, it may invert the stored bit in a register or a memory cell. These single event effects (SEEs) can take on many forms. NASA divides SEEs into soft and hard errors [1]. Soft errors, such as single event upsets (SEUs) and single event transients (SETs), are non-destructive faults which affect both sequential and combinational logic. In contrast to hard errors, they can be repaired by rewriting the configuration memory in a process called scrubbing or refreshing the register with the correct data. On the other hand, many types of hard errors are potentially destructive and may result in high operating current or gate ruptures. Several fault tolerant techniques address these issues trying to mitigate in-flight radiation effects. As continuous technology evolution makes these occurrences more frequent even in terrestrial applications the research is mostly oriented to SETs and SEUs. The most common way to cope with these faults is to employ different redundancy techniques such as triple modular redundancy (TMR) or duplex systems [2][3]. A complete design, or only a part of it, is duplicated or triplicated so that it can still operate properly even if one redundancy domain gets struck by an SEU. Xilinx has introduced a TMR tool which, depending on the type of logic elements, triplicates the design and introduces different voter types to mask the faults and propagate the correct output [4]. Nevertheless, if more than one domain is affected, TMR systems are incapable of further masking of the fault which results in an erroneous output. Therefore, in order to prevent the system from accumulating faults, periodical scrubbing of the configuration memory is used in many applications. In [5] such system which combines scrubbing with dynamic and partial reconfiguration (DPR) is proposed. The paper also resolves the common problem that arises when scrubbing frequently uses the Internal Configuration Access Port (ICAP) thus limiting the ability to perform DPR.

Among several alternative approaches, redundancy techniques in combination with DPR emerge as the most reliable way of protecting the system from SEUs. Moreover, as the price when replicating the logic is often expressed in huge device occupation and high power consumption, recent research in the field of fault tolerance was mostly oriented towards run-time partial reconfiguration of faulty modules. In addition, as TMR designs are susceptible to single point of failure, reconfiguration of a faulty domain may significantly extend the life of a complete system. Authors of [6] proposed fault-tolerant architectures that use TMR with different levels of granularity to mitigate permanent SEUs in SRAM programmable cells. Redundancies are voted with a single voter or partitioned and voted after each stage using various majority voters in order to detect and mask the fault. When the fault is detected partitions are reconfigured in order to heal the circuit. TMR modules must be functionally identical, i.e. their input/output responses should match, although the way of implementing does not necessarily have to be the same. In [7] a TMR variation, the so-called design-for-diversity, is proposed where TMR modules are implemented using different techniques thus improving fault tolerance. It is shown that the diverse design improves reliability for randomly introduced faults. Three modern fault tolerant architectures based on DPR and TMR or duplex with on-line checkers as a type of Concurrent Error Detection technique are proposed in [8]. Redundancy functional units are implemented in the dynamic part of the design and replaced if an upset strikes one or more of them. Nevertheless, partial reconfiguration of the corrupted redundancy domain takes time and in many systems the rest of the design has to be offline during that period which could be unaffordable in some applications. Authors of [9] deal with this issue by implementing different TMR domains such that each domain can be reconfigured independently from the others without interrupting the circuit operation. The design is partitioned, each partition has its own upset flag and a minority voter is inserted after every majority one in order to select the fault free domain. Some of these fault tolerant approaches are already employed in various space system applications.

In this paper we present a run-time adaptive platform which trades-off performance for fault tolerance in a digital video broadcast (DVB) on-board processor (OBP) used in a satellite communications application. The reconfiguration methodology is applied to an existing demultiplexer architecture (DEMUX) included in the OBP which was previously designed as an ASIC. Our proposal consists of two different methodologies for DPR in a space qualified SRAM-based FPGA and a complete adaptability in terms of partial configuration placing which allows the implementation of different redundancy techniques on demand. The first approach is traditional DPR where, depending on configuration bits, which might be received remotely, a part of the design in charge of some task is simply substituted by another in charge of another one. In the case of the reconfigurable DEMUX, each partial configuration represents a part of a sub-band (SB) hardware and contains stages required to create a given carrier frequency. The design methodology implies rewriting the part of the configuration memory referring to the whole reconfigurable area previously dedicated to a certain SB. The second approach uses the advantage of scalable and modular design, making possible the reconfiguration of a smaller part of the hardware thus lowering the time and power consumption of the partial reconfiguration process. These scalable partial configurations can be placed one next to each other in order to create a given carrier frequency in a certain SB. In both DPR approaches, partial configurations are compatible with all of the 4 SBs and can be configured in different positions of the FPGA. These SBs can be implemented in different zones and can be modified during run-time. Moreover, an SB can be duplicated or triplicated on demand by configuring it in two or three reconfiguration zones thus forming a duplex or TMR structure. One of the important contributions of this paper is the voting mechanism. Two different voting types are implemented and compared in terms of reliability and device utilization. The first one is a traditional voter which adapts to the position of the TMR structure and propagates the correct result to the output. The second voting approach is based on the readback of the configuration memory through ICAP and comparing the captured flip-flop values of each domain outputs in software. This approach allows for a complete exclusion of a voting mechanism in hardware thus avoiding a possibility that an SEU threatens a voter which usually has serious consequences on the entire design.

One of the main design goals was to enable complete adaptability in terms of SB placement order and provide an opportunity to configure certain SB several times during run-time in order to increase fault tolerance of the entire design. Moreover, the objective was also to minimize the device utilization area by reconfiguring the device only with necessary logic at a certain point of the operation and to create the least possible memory footprint for each partial configuration as each of them should represent a "golden copy" stored in a radiation hardened, non-volatile memory. By making the design reconfigurable we achieved significant simplifications in each SB stage module and established a system with adaptive fault tolerance capabilities.

The rest of the paper is organized as follows. In section 2, the non-reconfigurable, ASIC-based DVB OBP is introduced and the motivation for making it reconfigurable is given. In section 3, the conventional and scalable DPR methodologies are described along with the architecture of the reconfigurable design. In section 4, the complete run-time adaptive system-on-chip is presented and analyzed and thereafter the on-demand creation of fault tolerant structures together with both voting mechanisms is explained in detail. Obtained results are summarized in section 5. Finally, section 6 gives perspectives and conclusion of this paper.

II. ASIC-based DVB On-Board Processor

OBP technology is still considered too expensive to be incorporated in the satellite payload and the research in OBP systems has not progressed as fast as it was initially expected.

One of the main reasons for slow adoption is the lack of flexibility in ASIC technology which is unable to evolve once in orbit. This fact, associated with the long term duration of a satellite missions which generally last more than 15 years, lead to an obsolescence of the on-board equipment which is not able to cope with the changes in modulation and codification schemes. Reconfigurable OBPs based on a reprogrammable hardware happen to be an evolution path for future challenging flexible OBP architectures. Several papers are already published considering this topic. In [10] and [11] authors presented their reconfigurable OBPs implemented in space qualified Xilinx Virtex-5QV FPGAs showing significant advances in the field of OBPs.

This work presents the reconfiguration methodology applied to an already existent DVB OBP previously implemented using several ASIC boards shown in Fig. 1. The core of the system are the multicarrier 3D (MC3D) boards which are intended to demultiplex, demodulate and decode carriers located within a transponder in order to generate a single multiplex of MPEG-2 packets following the DVB standard. Block diagram of the MC3D architecture is shown in Fig. 2. The reconfiguration methodology is applied to the DEMUX part of the system due to its modular properties. The basic functions of the DEMUX are demultiplexing, switching and TM/TC interface. It performs digital demultiplexing of the 33MHz bandwidth channel in order to extract the possible data carriers (Table I) by processing 10-bit samples coming from an ADC working at up to 90MHz. These carriers are the inputs for the demodulator-decoder (DEMDEC) part of the system which creates the packets that are later handled to the SWITCH and MUX units.

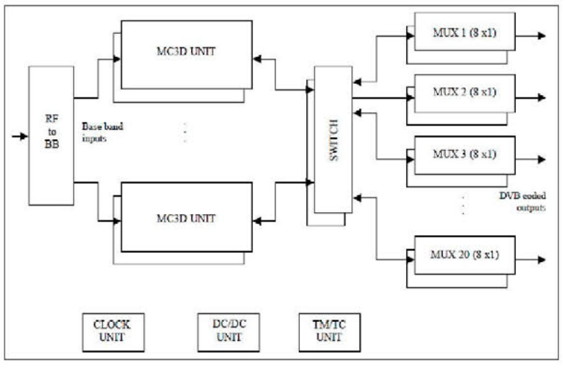

Fig. 1 Functional units of the DVB on-board processor

TABLE I. CARRIERS PRESENT IN ALL FIVE SUB-BANDS

Carrier Frequency [MHz]	Number of possible carriers in all sub-bands
8 (16Rs)	4
4 (8Rs)	9
2 (4Rs)	18
1 (2Rs)	36
0.5 (1Rs)	72

In the chosen frequency plan, useful spectrum is divided into 5 SBs, 4 of them with a bandwidth of 24Rs, and the other one with half that bandwidth. These values correspond to the bandwidth occupied by carriers of 16Rs and 8Rs respectively. The DEMUX has five possible output channels, corresponding

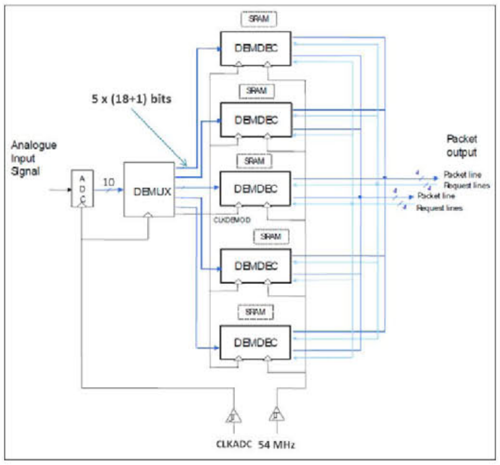

Fig. 2 MC3D architecture

to the number of SBs, connected to the five DEMDECs. The implemented DEMUX ASIC design is shown in Fig. 3. Each SB creates carriers that are always at a rate of 3*k*Rs (k = 16, 8, 4, 2, 1), whereas the demultiplexer output clock is 48Rs, which is five times slower than the input clock of the design. SB carriers with a rate lower than 16Rs are delivered multiplexed in time to the corresponding demodulator. As presented in Fig. 3, the block diagram is composed of a 10-bin and several 4-bin polyphase structures in charge of polyphase filtering. Apart from filtering they also perform decimation by 5 in the STR10 and by 4 in the STR4. The upper branch of the architecture corresponds to the 5th, 8Rs SB.

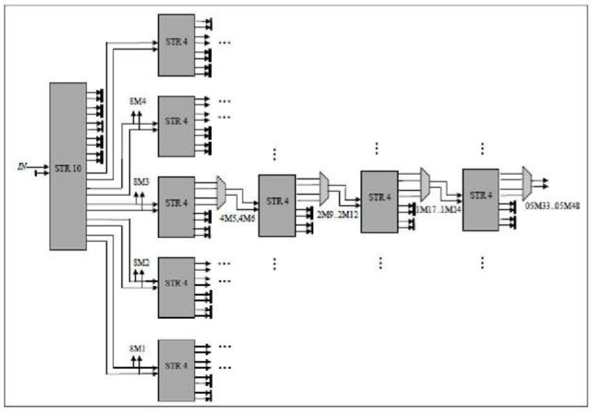

Fig. 3 Block diagram of the DEMUX ASIC architecture

As this was formerly an ASIC-based design, resources of each SB are present on the board although not all of them are necessary at all points of the operation. Hence, unnecessary device area is occupied such wasting space, power consumption and therefore cost. The following section presents the reconfigurable DEMUX architecture created by applying two different DPR methodologies.

III. THE RECONFIGURABLE DEMUX

In each SB of the non-reconfigurable DEMUX, 4, 2, 1 and 0.5 MHz carriers are constantly created although only one frequency rate is propagated to the output in a given configuration. In order to save unnecessary resources at certain points of the operation, the partitioning of the architecture described in the previous section is shown in Fig. 4. The static part consists of the 5th SB and several block modules in figure represented as STR10 that are necessary at all points of the operation. Four 24Rs SB are left in the reconfigurable part of the design in order to benefit from the modular property of each branch and create only the demanded frequency carriers. As presented in Fig. 4, the 8 MHz carriers are created in the static part unlike lower frequency carriers which are created in STR4 blocks inside the reconfigurable part of the architecture. Although named the same in Fig. 3, STR4 blocks differ going from higher to lower frequencies in the implemented architecture. Moreover, the 2 MHz frequency carriers are created after the 3rd STR4 block and a multiplexer is used after the 1st one to select the proper input for the following blocks. This is not the case in the reconfigurable DEMUX where the need for configuration bits is completely avoided inside the SBs. Using DPR techniques branches are configured on demand such that at certain point of the operation contain only the necessary logic to create carriers of the demanded frequency. Two different partial reconfiguration methodologies are developed and presented in the following subsections.

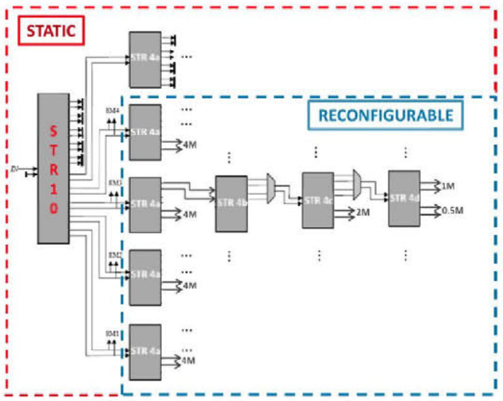

Fig. 4 Partitioning of the DEMUX architecture

A. Conventional DPR Methodology

The reconfigurable DEMUX architecture obtained applying conventional DPR methodology is presented in Fig. 5. The figure shows the current state of the architecture when 0.5, 8, 4, and 2 MHz frequency carriers are demanded from the 1st, 2nd, 3rd and 4th SB, respectively. Necessary stages for the creation of certain carriers are presented in partial bitstream blocks (PBS). Each of these blocks represents partial configuration which should be stored as a "golden copy" in an external radiation-hardened memory. The presented configuration implies taking 0.5 MHz frequency carriers from the 1st SB. Hence, the portion of the FPGA area reserved for the branch will be reconfigured using the logic referring the creation of these carriers and stored as partial bitstream 4. This is the most area consuming partial configuration as it consists of all four stages. Nevertheless, each stage is significantly simplified during the implementation optimization process as there is no need for the creation of other frequency carriers. In order to make the SBs completely reconfigurable, the 8 MHz carriers, created in the static part, are passed through the reconfigurable zone configuring a simple cover. Therefore, the entire FPGA area reserved for a certain SB is left empty thus saving the area and power consumption. The 4 MHz carriers are created and taken out to the static part of the design by configuring partial configuration 1 which consists of only one stage. Consequently, the major part of the SB area is saved. In the given configuration, the 2 MHz carriers are created in 3rd SB and taken out to the static part of the design. The partial configuration 2 consists of three stages where the 2nd is modified such that from its outputs 3rd stage creates the demanded carriers. Stage 3 is also optimized thus leaving the block without the needed logic for the following stage which is left out in this configuration.

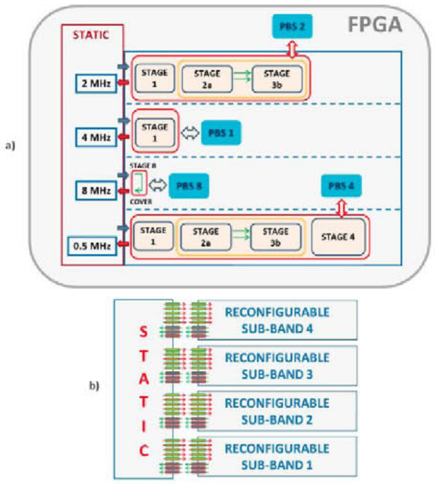

Fig. 5 Conventional DPR methodology applied to the DEMUX architecture

Partial configurations are implemented with careful considerations in the implementation stage using PlanAhead and FPGA Editor in order to constrain logic within a specific range. Each configuration is implemented such that consumes the least possible power and area. Partial configurations are additionally optimized during the synthesis process. All of them are implemented within 2 clock regions in the right part of the FPGA and their representation in FPGA Editor is given in Fig. 6. Our reconfiguration engine, capable of bitstream relocation, configures the demanded logic in reconfiguration zones placed in the right part of the FPGA where each clock region has the same architecture.

Ensuring fixed connection between the static and reconfigurable part(s) after the reconfiguration process is of the crucial importance when implementing designs with DPR. In addition, if there is more than one reconfigurable zone and they need to communicate directly one with each other, proper connection between them should also be ensured. In order to achieve it, a hard (bus) macro is created to fix each connection at a predetermined position. In the reconfigurable DEMUX, 8-bit bus macros consisting of 2 slices are placed on the border between the static and reconfigurable parts which is usually called the "cut line". As each configuration has the bus macros at the same FPGA location, when a region is reconfigured one slice is substituted with another of the same type hence ensuring permanent correct connection between the parts. The reconfigurable DEMUX uses 8 bus macros per reconfiguration zone (Fig. 5b). The lower 3 are used to pass the carriers from the reconfiguration zone to the static part whereas the higher 5 provide the inputs for the zones. Since there are 4 zones the total number of bus macros is 32.

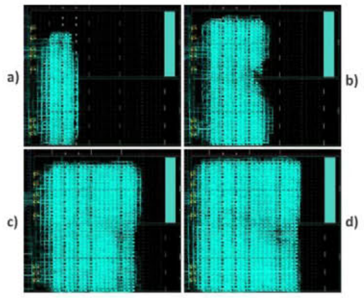

Fig. 6 Implemented partial configurations: a) PBS 1; b) PBS 2; c) PBS 3; d) PBS 4

B. *Scalable DPR Methodology*

The conventional DPR methodology presented in the previous section requires the substitution of an entire partial configuration by another one with different functionality. That implies that larger portion of the configuration memory has to be rewritten at each reconfiguration thus making the process more time and power-consuming. In order to reconfigure as less area as possible, and therefore reduce the time and power consumption, a scalable partial reconfiguration methodology that takes the benefit from the modular and scalable properties of the original architecture is proposed. The reconfiguration strategy is shown in Fig. 7.

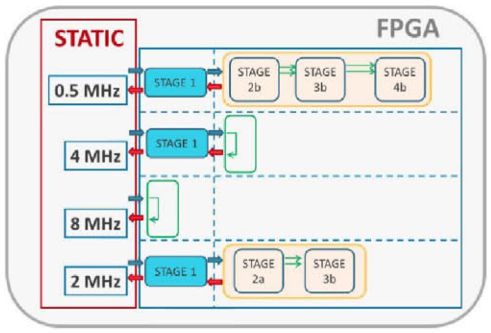

Fig. 7 Scalable DPR methodology applied to DEMUX architecture

The figure presents an example configuration where the 2, 8, 4 and 0.5 MHz frequency carriers are demanded from the SBs. Reconfigurable branches are composed of scalable configurations that generate demanded carriers by connecting blocks one next to another. Hence, when changing for instance the SB configuration from 2 to 4 MHz, stage 1 is left in SB thus reconfiguring less FPGA area. Bus macros are placed between the reconfigurable zones inside the SB thus making possible the propagation of the data to the next scalable partial configuration. Although full scalability could not be achieved due to the on-chip area limitations, scalable solution gives better results in terms of the reconfiguration time and the total size in memory of partial bitstream files comparing to the conventional DPR solution. The static part is compatible with both DPR methodologies making the decision on whether to use scalable or conventional one constantly open. The partial configurations implemented are shown in Fig. 8.

IV. RUN-TIME ADAPTIVE SYSTEM-ON-CHIP

In this section, the complete reconfigurable system will be described. The achieved adaptability in terms of SB configurations and decisions on where to place a particular SB

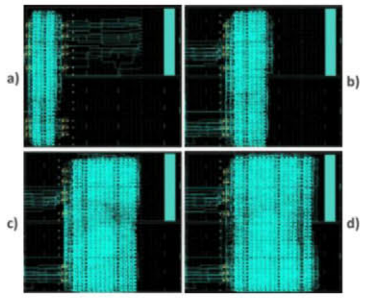

Fig. 8 Implemented scalable partial configurations: a) PBS_S 1; a) PBS_S 2; a) PBS_S 3; a) PBS_S 4;

at a given moment of operation will be presented. In addition, the run-time trade-off between the performance and fault tolerance will be explained along with the two possible voting solutions.

A. *Reconfigurable System Evaluation Platform*

A block diagram of the implemented evaluation system is presented in Fig. 9. It consists of several peripherals that make possible the run-time reconfiguration of the DEMUX architecture. In the implemented design, all the peripherals together with the STR10 block of the DEMUX are placed in the static part and therefore are present at all points of the operation. The system includes CompactFlash and DDR2 memory controllers which provide proper communication with the external memories. These memories will be replaced in the final design by the radiation hardened non-volatile memory where the full and partial configurations should be stored as the "golden copies" of the reconfigurable design. An embedded microprocessor is used to control the run-time adaptation to different operational demands and support the reconfiguration process. Moreover, along with GPIOs it is used for the evaluation and verification where the reconfigurable DEMUX is stimulated using predefined stimuli stored on the CompactFlash card. The output results are obtained through GPIO 2 and stored in the memory.

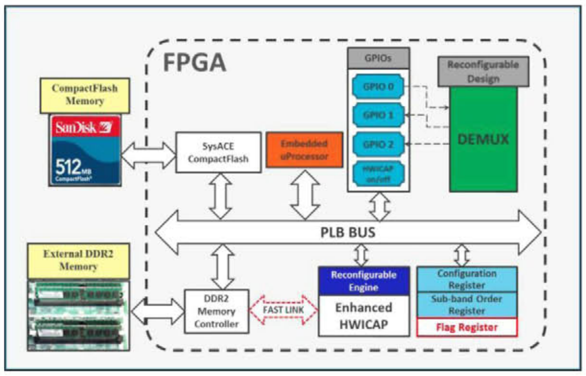

Fig. 9 Block diagram of the reconfigurable system

The reconfiguration engine is required in every reconfigurable system. Our OBP takes the benefits of the enhanced HWICAP which is based on the Xilinx HWICAP and proposed in [12]. Apart from the conventional run-time partial reconfiguration, the peripheral supports the relocation of partial configurations, core replication and run-time readback of the configuration memory. Moreover, the engine is additionally upgraded for the purpose of the system such as providing a possibility to capture the states of design flip-flops at a certain point of the operation. The enhanced HWICAP is modified in order to adapt to our FPGA platform and it is fully implemented in hardware which makes it able to achieve higher reconfiguration speeds.

In order to control the reconfiguration process and make the system capable of various run-time adaptations, a custom peripheral containing configuration, sub-band order (SBO) and flag registers is included in the architecture. The configuration register provides the information about the current state of the SBs and is constantly used by the reconfiguration engine. The information about where to configure a particular SB is given in the SBO register. Contents of these registers should be received remotely and be available at all points of the operation. The flag register is a 4-bit register where each bit represents a reconfiguration zone. If a fault occurs, the bit referring to the zone is set high. Communication between the peripherals is done over processor local bus (PLB).

When the configuration register bits are changed and therefore the demands from the SBs, required partial bitstreams saved on the CompactFlash card are copied to the DDR2 in order to obtain faster reconfiguration. The enhanced HWICAP rewrites a portion of the configuration memory referring to the certain zone with the corresponding partial configuration. The reconfiguration engine is able to reconfigure the required zone using the DDR2 interface highlighted as the "fast link" in Fig. 9. The reconfiguration process can be done at the rate of 200 MHz although the power consumption is higher comparing to the case when the processor interface is used. As the reconfiguration speed is not of the crucial importance for the application, the Enhanced HWICAP takes the benefit of the second option.

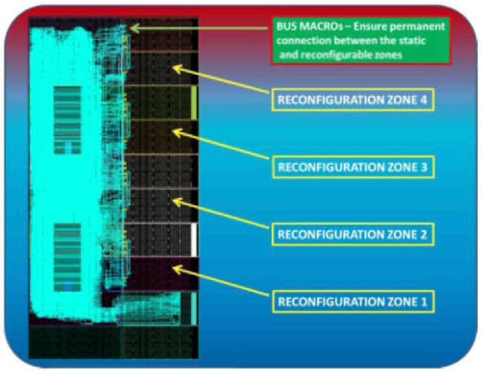

Fig. 10 Implemented static part of the design

The implemented static part of the design is presented in Fig. 10. Four reconfiguration zones reserved for four SBs are implemented in the right part of the FPGA. Each SB is implemented within 2 clock regions corresponding to the are occupation of the largest partial configuration. Five and six different configurations are created and stored in the memory for both, conventional and scalable DPR methodologies, respectively. At the very same border between the static and reconfigurable zones 32 bus macros are implemented (highlighted in yellow in the given figure). The reconfiguration zones are completely separated one from another so that no overlapping is possible. As can be seen, the entire right part of the FPGA is left with no logic and made completely reconfigurable thus making a significant step toward a fault tolerant system.

B. The Sub-band Placement Adaptability

The initial reconfigurable DEMUX solution implied fixed positions for each of the 4 reconfigurable SBs. In order to implement a completely adaptive system capable of taking SB carriers from different reconfiguration zones to every DEMDEC block, the SB placement should be completely flexible. Therefore, a hardware module able to select the zone for a particular SB is implemented in the static part of the design. It uses the remotely received configuration data stored in the SBO register and propagates the SB inputs to the corresponding zone. It also takes the created carriers from each configured SB providing the data properly to the output module.

A block diagram of the described process is implemented in Fig. 11. In an example configuration where the SB order in zones from 1 to 4 states 2nd, 3rd, 1st and 4th, respectively, the selector module properly propagates the input data created in the STR10 block (SB1…SB4) to the corresponding reconfiguration zone. It then takes the output carriers from zones 1, 2, 3 and 4 and forwards them to the SB2_out, SB3_out, SB1_out and SB4_out inputs of the interface module, respectively. The selector module allows all possible order combinations and enables run-time placement adaptability. The 32 bus macros that ensure proper connection after the reconfiguration process are presented on the border between the static and reconfigurable zones.

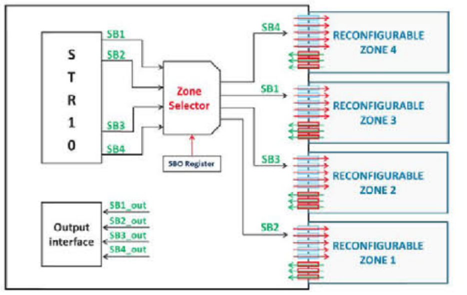

Fig. 11 Adaptive sub-band placing for an example SBO configuration

C. The Run-Time trade-off: Performance vs. Fault Tolerance

In the previous sub-section, the run-time adaptability in the SB placement which allows for implementing reconfigurable branches in different reconfiguration zones was presented. Another property of the selector block is the ability to forward particular SB inputs to several reconfiguration zones. Hence, SB logic can be duplicated or triplicated thus forming the well-known duplex or TMR structures used in various fault tolerant systems. Moreover, as the selector uses the remotely received configuration from the SBO register, these structures can be implemented with great flexibility during run-time. In other words, structures can be configured in any possible zone position thus making the system tolerant not only to soft errors but also to certain types of hard errors as a particular logic can be implemented in other, undamaged part of the FPGA.

In the DVB OBP presented in section II, the DEMUX constantly provides carriers to each of the five DEMDEC blocks. However, in many cases MPEG-2 packets are created using only some of them. When a particular SB is duplicated or triplicated in hardware, some other SBs have to be left from the configuration. The selector module enables the proposed trade-off thus providing additional protection to the system. Nevertheless, as structures based on hardware redundancy require selection of the undamaged domain outputs, an adaptive voting mechanism able to recognize the structure position had to be implemented.

D. Adaptive Voting Methodologies

The first voting method implies configuration of an adaptive majority voter capable of adjusting to the structure position in order to take the created carriers from different redundancy domains and forward the correct ones to the output interface. An example of voting the carriers created in the reconfigurable TMR domains is presented in Fig. 12. The 3rd SB is configured in zones 1, 2 and 4. Zone 3 is configured with the 2nd SB logic. The voter adapts to the position of the redundancy domains and forwards the output of the undamaged circuit replica. In addition, it can detect the faulty domain and set high the flag register bit referring to the affected reconfiguration zone. When a fault is detected, the information given in the flag register triggers the enhanced HWICAP to execute the reconfiguration of a faulty domain. The selector is in charge of taking the carriers from reconfiguration zone 3 to the SB2_out input of the interface.

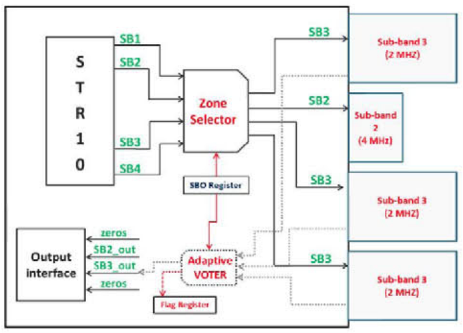

Fig. 12 Reconfigurable TMR domains and adaptive carrier voting

In conventional TMR, majority voters are used to mask errors that occur in any single circuit replica. Nevertheless, the voter logic can also be affected by an SEU thus completely corrupting the TMR structure. Therefore, voters are also triplicated in order to provide better protection. However, in complex applications where various domain outputs have to be compared, this methodology additionally increases the area overhead of the system. In order to avoid this problem another voting methodology is proposed which completely excludes the voting mechanism from hardware and performs voting directly through ICAP. The proposed methodology takes the benefit of the modified enhanced HWICAP which is capable of capturing the present state of the configuration memory and performing the readback of the captured condition. In order to vote the output from the undamaged domain and detect the faulty one, the outputs of each reconfiguration zone are registered and carefully constrained in PlanAhead. Four output registers corresponding to four reconfiguration zones are contained in areas with the width of one CLB column. The enhanced HWICAP periodically sends the GCAPTURE command to the ICAP thus capturing the current flip-flop state of the design. Depending on the position of the configured fault tolerant structure, different portions of the configuration memory are read through the ICAP and stored in DDR2 memory. At each readback 36 frames are stored corresponding to a CLB column representation in the Virtex-5 configuration memory. Xilinx does not reveal detailed information about the frame composition. However, the logic allocation file, which can be generated in ISE along with the programming file, indicates the bitstream position of latches, flip-flops, BRAMs, LUT programming and IOBs. Using this information we are able to locate the relevant flip flop bits in the performed readback. Moreover, it is observed that these bits belong to the 30th and 31st frame which is the case for all CLB flip flops of the FPGA. In order to vote the correct result, embedded processor performs a periodical comparison of the bits referring to the domain outputs. It scans only two frames in captured configurations in order to read the bits and perform the comparisons thus making the voting process faster. If a fault is detected, the processor flips a spare bit in the SBO register to select the correct result for the interface module. In addition, the number of the affected reconfiguration zone is provided to the enhanced HWICAP which performs partial reconfiguration of the area thus eliminating the upset. In order to detect even transient faults, the present value of the zone output is combined with the previous one. These feedback registers keep the track of the output changes and are implemented in the same CLB column to take the benefit of the same readback. A transient fault is detected, if in a captured state, domain outputs match, whereas the feedback register value of one replica differs from others.

V. RESULTS AND DISCUSSIONS

The proposed architecture is implemented on a Xilinx Virtex-5 XC5VFX130T FPGA. Implementation of the full and partial configurations is done in ISE Design Suite 14.2. To achieve the desired flexibility, each configuration is carefully constrained using PlanAhead. Three GPIOs are implemented in order to evaluate the architectures. Five different stimuli files, stored on the CompactFlash card, are applied for 5 possible SB configurations. Files containing the outputs after each reconfiguration and stimulation are saved on the card and compared to the corresponding files obtained in the evaluation of the original design. The comparisons showed no difference thus proving the reliability of proposed DPR methodologies. The non-reconfigurable and reconfigurable systems are analyzed in terms of device utilization and the amount of memory used by the configurations. The analysis is done also for the DEMUX architecture alone and presented in Table II. Implemented static part of the reconfigurable DEMUX architectures use approximately 3/4 less resources than the non-reconfigurable one. Hence, the reduction of almost 75% is obtained in terms of the logic that is always present in the device. When comparing the entire systems, savings drop to approximately 60% due to the presence of the enhanced HWICAP in the reconfigurable system.

Reductions in the size of the memory footprints are of great importance as they represent the "golden copies" stored in a rad-hard non-volatile memory. A full configuration of the Virtex-5 XC5VFX130T FPGA has the size of 5.86 MB. The compressed bitstream of the non-reconfigurable and reconfigu-

Table II. THE CHIP UTILIZATION AND SIZE OF THE MEMORY FOOTPRINTS

*	DEMUX Architecture	Static part - DEMUX Architecture	Non-Reconfigurable System	Reconfigurable System
Occupied Slices	11892 (55%)	3079 (15%)	15465 (75%)	5510 (26%)

TABLE III.
THE CHIP UTILIZATION, THE MAXIMUM RECONFIGURATION TIME AND THE SIZE IN MEMORY OF THE CONVENTIONAL PARTIAL CONFIGURATIONS

*	PBS1	PBS2	PBS3	PBS4	Cover	
Occupied Slices	280 (1 %)	661 (3 %)	1125 (5 %)	1305 (6 %)	16 (1%)	
Reconf. Time [us]	103.3	191.9	280.44	280.44	29.52	Total
Size [KB]	132	211	295	295	24	957

TABLE IV.
THE CHIP UTILIZATION, THE MAXIMUM RECONFIGURATION TIME AND THE SIZE IN MEMORY OF THE SCALABLE PARTIAL CONFIGURATIONS

*	PBS_ S1	PBS_ S2	PBS_ S3	PBS_ S4	Cover	
Occupied Slices	340 (1 %)	484 (<2%)	944 (<4%)	1104 (<5%)	16 (1%)	
Reconf. Time [us]	88.2	103.3	191.9	236.16	29.52	Total
Size [KB]	88	132	207	242	24	693

rable system occupy 3.91 MB and 2.23 MB, respectively. The device utilization for each partial configuration of the conventional and scalable methodology and the corresponding size in memory are presented in Table III and Table IV, respectively. It can be seen that the important reduction in the memory is achieved when the scalable partial configurations are used instead of the traditional ones. In addition, the reconfiguration time for each configuration is given in the table also favoring the scalable DPR methodology. The static part of the reconfigurable system utilizes less than 30% of the FPGA resources. The rest of the chip is either completely reconfigurable or left empty which increases the possibility to mitigate upsets. The self-reconfigurable system is able to adapt to harsh environmental conditions during run-time by duplicating or triplicating a critical part and voting the outputs. Faults are injected by changing certain bits in the partial configurations. Upsets that affected the design outputs were detected and corrected in 100% of the cases by the adaptive system thus recovering the proper functionality. In the case of the voting through ICAP methodology, systems may produce erroneous outputs during several clock cycles before the subsequent memory capture. Therefore, the frequency of the captures should accommodate to the environmental conditions, requirements and sensitivity of the design.

VI. CONCLUSIONS

In this paper we have presented a self-reconfigurable DVB OBP that adapts to the remotely received demands during run-time. Two different DPR methodologies are proposed and implemented. We have obtained compact and low memory consuming partial configurations for both reconfiguration approaches. Implementation results have been given for the Virtex-5 XC5VFX130T FPGA which is compatible with the space qualified, Virtex-5QV, recently introduced by Xilinx. With a single memory footprint an entire half of the FPGA can be reconfigured which significantly affects the size of the rad-hard memory. The proposed architecture is able to duplicate a part of its logic during run-time in order to create fault tolerant structures at the price of some performance capabilities. This property makes the system adaptive to different environmental conditions. We have developed two different methods to detect the faults and vote the correct output coming from the structures that may be implemented in several different FPGA positions. Detected faults trigger the reconfiguration process performed by an advanced ICAP controller thus eliminating the present upset and avoiding fault accumulation in the redundant domains. In addition, an ICAP-based voting is proposed which makes possible a complete exclusion of a voting mechanism in hardware thus preventing eventual upsets of the most critical part of TMR structures.

REFERENCES

[1] http://radhome.gsfc.nasa.gov/radhome/papers/seespec.htm

[2] 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 doi: 10.1109/MDT.2004.85

[3] Pratt, B.; Caffrey, M.; Graham, P.; Morgan, K.; Wirthlin, M., "Improving FPGA Design Robustness with Partial TMR," Reliability Physics Symposium Proceedings, 2006. 44th Annual., IEEE International , vol., no., pp.226,232, 26-30 March 2006

[4] Carmichael, "Triple module redundancy design techniques for Virtex FPGAs," Xilinx Corporation, Tech. Rep., November 1, 2001, xAPP197(v1.0).

[5] 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

[6] 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.

[7] Ashraf, R.A.; Mouri, O.; Jadaa, R.; DeMara, R.F., "Design-for-Diversity for Improved Fault-Tolerance of TMR Systems on FPGAs,"Reconfigurable Computing and FPGAs (ReConFig), 2011 International Conference on, vol., no., pp.99,104, December 2011

[8] Straka, M.; Kastil, J.; Kotasek, Z., "Modern fault tolerant architectures based on partial dynamic reconfiguration in FPGAs,"Design and Diagnostics of Electronic Circuits and Systems (DDECS), 2010 IEEE 13th International Symposium on, vol., no., pp.173,176, April 2010

[9] 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 doi: 10.1109/AHS.2013.6604220

[10] Hofmann, A.; Wansch, R.; Glein, R.; Kollmannthaler, B., "An FPGA based on-board processor platform for space application," Adaptive Hardware and Systems (AHS), 2012 NASA/ESA Conference on , vol., no.,pp.17, 22, 25-28 June 2012; doi: 10.1109/AHS.2012.6268653

[11] Rittner, F.; Glein, R., "Implementation of an initial-configuration based on self-reconfiguration for an on-board processor," Adaptive Hardware and Systems (AHS), 2013 NASA/ESA Conference on , vol., no., pp.55,62, 24-27 June 2013 doi:10.1109/AHS.2013.6604226

[12] 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