# Use of Triple Modular Redundancy (TMR) technology in FPGAs for the reduction of faults due to radiation in the readout of the ATLAS Monitored Drift Tube (MDT) chambers

M. Fras, H. Kroha, J. v. Loeben, O. Reimann, R. Richter, and B. Weber,,
*Max-Planck-Institut für Physik, Föhringer Ring 6, 80805 München*

***Abstract*–The Triple Modular Redundancy (TMR) technology allows protection of the functionality of FPGAs against single event upsets (SEUs). Each logic block is implemented three times with a 2-out-of-3 voter at the output. Thus, the correct logical value is available even if there is an upset bit in one location. We applied TMR to the configuration code of a Virtex-II-2000 FPGA, which serves as the on-chamber readout processor of the ATLAS monitored drift tubes (MDTs). We describe the code implementation, results of performance measurements and discuss several limitations of the method. Finally, we present a supplementary technology called "scrubbing". It permanently checks the configuration memory while the FPGA is operating, and corrects upset configuration bits when necessary.**

## I. Impact of Radiation on S-RAM based FPGAs

S-RAM based FPGAs are programmable logic devices, which at power-on receive their program code from a flash PROM or a similar device. This code is then stored in a kind of S-RAM cells in the FPGA. They determine the functionality of the device by defining the functionality of the logic primitives of the chip and the routing structure between them.

Like all electronics devices, the S-RAM based FPGAs can suffer from various radiation effects [1]. The worst of them are non-reversible and may be destructive to the device. Such are latch-ups and gate ruptures for example. Damage to the silicon lattice and changes in the doping or at p-n junctions is also possible. They can lead to severe effects like changes in transistor and timing behavior [5].

Looking at reversible effects, there can be single event upsets (SEUs) and single event transients (SETs). The SEUs result in bit-flips in registers or memories, leading to illegal states in finite state machines (FSMs) or data corruption, for instance. On the other hand, SETs produce short spikes or glitches on individual signals. Depending on the kind of signal, several effects on the circuit are possible, e.g. an asynchronous reset can be triggered, an additional clock edge can be generated, or the wrong logic state can be latched into a storage cell. Of course, there are many more possible impacts of SEUs and SETs to the user function.

Manuscript received November 8, 2010.

Markus Fras is the corresponding author. He is with the Max-Planck-Institut für Physik, Föhringer Ring 6, 80805 München, Germany (telephone: +49 (0)89 32354-311, e-mail: fras@mpp.mpg.de).

In addition to errors in the user design, which can occur to any digital device, there are types of effects only related to S-RAM base FPGAs. One of them is that the chips can suffer from radiation damages or changes to the configuration cells. As these are S-RAM like components, they are susceptible to SEUs like the flip-flops and memories in the user's design. The difference is that upsets in this region can change the functionality of the device, like if a different design was running on the device. They can change the function of logic blocks or the routing structure [2], [7]. This can lead to an unpredictable behavior like a modified logic behavior, internal driver fights or the complete malfunction of the design [10].

Finally, there are special blocks that control the overall operation of the FPGA like the configuration logic, the global reset circuit, or the JTAG state machine. A disruption or malfunction of one of these components can put the whole device out of operation. These kinds of errors are referred to as single-event function interrupt (SEFI) [1], [2], [3], [8], [11].

## II. Remedies for Radiation Effects

The destructive, non-reversible radiation damages mentioned before can only be addressed by changing the silicon layout. As this is done by the FPGA vendors, the user cannot influence it, but can only test different devices for their radiation tolerance concerning silicon effects and choose the most robust device.

### *A. Single-event Function Interrupts (SEFIs)*

Concerning SEFIs, an external watchdog device is mandatory. It would typically be a radiation hard ASIC, but one could also think of another FPGA or a microcontroller [8]. One can also use a set of two or more FPGAs that check each other [3]. The watchdog would monitor the most vital signals of the FPGA like the pin indicating that the program is correctly loaded and the chip is in operation mode. Checking the clock and reset nets is recommended to ensure proper operation as well as defining additional test signals like a heartbeat produced by sub-circuits. In case the watchdog detects a malfunction, it would trigger the reconfiguration of the FPGA. This will cure most of the possible reversible errors. In case the error persists, the FPGA must be power cycled, which could also be done by the watchdog, or the system would ask for user interaction. Of course, while reprogramming or cycling the power of the chip, the operation is interrupted and data will be

lost [8]. To cover this, one can use two or more individual devices with an identical functionality operating in parallel. Their outputs have to be combined in some way that ensures proper results even if one device is out of operation. This can for example be a majority voter for three individual devices [2]. Methods like this have the disadvantage that they consume more resources, a larger space on the PCB and will dissipate additional power.

### *B. Single Event Upsets in User Space*

Coping with SEUs in the logic part defining the user functionality is comparatively less demanding. In the simplest setup, one could use a single device, and triplicate all the logic blocks inside, i.e. use three copies of all design elements. Every piece of operation is carried out in three individual cells in parallel. If one of them gets upset, there are still two that provide the correct result. By inserting a majority voter at the end, the correct logical status will be determined. This method is referred to as triple module redundancy (TMR). It is very common and widely used to improve radiation and fault tolerance in digital devices. In addition, all clock and reset nets should also be tripled as well, each of them being routed to one of the three copies of logic modules. By phase-shifting the three clock lines, one can achieve better immunity against SET, because the data are latched at different times, so that a short spike would possibly affect only one of the three modules.

### *C. Single Event Upsets in Configuration Memory and Scrubbing*

As discussed in section 1, S-RAM based FPGAs can suffer from upsets in the configuration memory. It is not trivial to mitigate this kind of fault, as detailed knowledge of the configuration memory and the methods to access it is necessary. As these operations are usually not of interest for the average FPGA user, documentation and support from the manufacturer tends to be at a low level. Nevertheless, for many kinds of FPGAs there are ways to read back the configuration and check for errors while the device is in normal operating mode. For some chips, even the configuration memory can be fully or partially re-written while the user design continues to run without any disruption. This method is called "scrubbing". There are a lot of different options for implementing the scrubbing unit [1], [2].

The best and most robust one is to have an external, radiation tolerant device that performs the boot-up of the FPGA and reads back its configuration memory cyclically during operation and compares it to the original values stored in some radiation tolerant memory. In case of an upset, it would re-write the corrupted configuration memory block, thus clearing the upset bits [3].

A more simple procedure would be "blind scrubbing", which means that the configuration is written cyclically to the configuration memory without checking for upsets previously. The advantage of this is that it is easier to implement and debug. On top of this, it uses fewer resources and might even be implemented in the same FPGA that is being scrubbed. The drawback of this self-hosted scrubber is that it is less fault tolerant than an external device. Also, writing the configuration is quite critical, as it may introduce additional errors. This can for instance happen if the configuration state machine of the FPGA suffers from an upset. The safest way is to read-back and check the configuration and write to the configuration memory only when an upset is detected.

### *D. Combination of TMR and Scrubbing*

Scrubbing and TMR should be considered as complementary methods for the mitigation of radiation effects [2]. While TMR makes the device immune to single upsets of control and data processing block, scrubbing heals flipped bits in the configuration memory. It thus avoids accumulation of upsets which can lead to malfunction even if the design is TMR'd. In simple words, TMR protects against a few SEUs in the user or configuration memory, while scrubbing keeps the number of upset as low as possible [1].

## III. Implementation of TMR and Scrubbing in the MDT Readout FPGA

The processing unit of the MDT front-end readout is implemented in a Xilinx Virtex-II 2000 FPGA. It is sitting on a board called "chamber service module" (CSM). This is on the one hand a link to the slow control system, distributing setup data to the sub-units and collecting information from them. On the other hand, it receives the drift time measurement data from all tubes and multiplexes them onto a 1.6 Gbps optical link [12].

### *A. TMR Implementation for the Xilinx FPGA on the CSM board*

As the CSM board is sitting on the MDT chamber, it will be exposed to radiation, which level depends on the specific location of the chamber in ATLAS. Radiation tests showed that destructive effects will not be an issue for the CSM for an estimated period of 10 year of full luminosity LHC running. Also the amount of SEU-related malfunctions observed without any mitigation scheme applied would be at a low rate [6], [9].

To improve immunity against SEUs, we use triple module redundancy. Xilinx offers a commercial software package called "TMRTool" that does the triplication in a semi-automatic way as an extra step between synthesis and place and route. The process is called Xilinx triple module redundancy (XTMR). The software provides the possibility to specify different XTMR type. They distinguish which parts should be tripled and how the remaining components should be attached to the tripled blocks. A voter can be inserted when interfacing a TMR'd block to a normal block, or only one of the three outputs will be used. This can be set in the TMRTool software for individual components. In addition, TMRTool replaces some critical FPGA elements which are especially problematic under radiation. One of them is the SLR16 shift register which is replaced with normal flip-flops. It also removes so-called weak-keepers or half-latches, which provide a constant 0 or 1 value to tie unused ports of registers to defined values without using routing resources [11]. These are replaced with wires connected to external or internal ground or power sources [1].

For the CSM design, only a part of the components can be triplicated due to a lack of resources. The normal design uses about 41% of the logic and 43% of the global net resources, so that the fully triplicated design would not fit into the given device. In addition, tripling the IO pins is not possible, because on the PCB every signal is routed to only one FPGA pin. So we decided to apply XTMR only to the critical parts, which are crucial for the operation of the device. These are for instance the control part, including the JTAG state machine and the interface to the clock and trigger receiver chip (TTCrx) and the gigabit optical (GOL) link ASIC. The clock that is used for all basic functions is also triplicated. In addition, parts of the data path and its flow control are TMR'd. With this partitioning we have a 92% usage of logic resources and 56% for global routing.

### *B. Test Setup for CSM Firmware*

Our findings so far are that the TMRTool flow is quite easy to handle and works well. As we have relaxed timing constraints with the fastest wide-spread clock frequency being 80 MHz, we experienced no issues in timing closure. The XTMR design was tested in our laboratory setup. There we have the possibility to inject arbitrary data directly into the CSM board using a pattern generator. In contrast, we can use analogue signals similar to those produced by the tubes. They are fed into the front-end board, which then is connected to the CSM. The output is recorded by readout software on a PC and can then be evaluated. In addition, internal signals of the FPGA can be routed to test pins and inspected in real-time with a logic analyzer. All the tests performed showed a normal behaviour of the TMR'd firmware, with no difference to the original version without TMR. The only observation was an increased power consumption of about 0.2 A (0.3 W) in the FPGA core circuitry, also resulting in a higher temperature of the FPGA and the corresponding voltage regulator.

After the tests in the laboratory with only a few channels, the TMR'd firmware was applied to a chamber of the cosmic ray test setup. It is equipped with the full set of 18 front-end boards and 432 tubes. The drift time spectra were measured with more than 107 entries. It showed a normal distribution. During the default test operation during several days, the system worked normally. Nevertheless, it had one hook-up. Its reason is uncertain, but there is a big chance that it was due to the slow control or the readout system. There will be more detailed tests in the future, before the firmware is released for the use in ATLAS.

### *C. Implementation of a Self-Hosted Scrubber*

Finally, we started to look at the possibility of configuration memory scrubbing, which would be a perfect complement for TMR [2]. As the CSM PCB was designed some years ago without an external device to perform the scrubbing process, the only choice is a self-hosted scrubber. At least the flash memory, which stores the FPGA configuration program, is considered much more radiation tolerant than the S-RAM cells in the FPGA. All configuration signals of the flash device are routed to the FPGA configuration pins as well as to FPGA user IOs on the CSM PCB. Thus, the FPGA can boot normally from the flash device. Once it has started up, a user designed circuit in the FPGA can take access over the flash memory and read out the original FPGA code. In addition, Xilinx provides a module for accessing the configuration memory from inside the FPGA, the "Internal Configuration Access Port" (ICAP). It has read and write access to the configuration memory [3]. So it is possible to create a module that reads back the configuration memory from its own FPGA through the ICAP and performs checks on it. In case of an error, it can read the original FPGA code from the flash memory and re-program the FPGA that it is running on. For Xilinx Virtex-II FPGAs there is a way of doing this without disrupting the operation. Xilinx calls this method "active partial reconfiguration". The user code will continue to run and no data will be lost during this step.

A first simple version of the scrubber has been implemented and tested. It was designed as a finite state machine. As this implementation method tends to use many resources, it works as simple blind scrubber only, i.e. it constantly re-programs the FPGA with the data from the flash without reading back and checking the configuration memory beforehand. The problem with this variant is that it relies on a completely working configuration interface of the FPGA. It assumes that the address counter is incremented correctly from the beginning of the stream to the end. According to experts from Xilinx, this mechanism tends to fail under radiation. Their recommendation is to write only single frames and check the access to the configuration registers before writing the next frame. Unfortunately, this procedure seems too complicated to be implemented as a state machine. The resulting code would use a lot of logic resources. To overcome this issue, a soft-processor can be used to control the scrubbing. There is an example design provided by Xilinx for the Virtex-5 FPGA using the 8-bit PicoBlaze soft microprocessor [4]. The design could be adapted to the CSM. The drawback of the soft-processor is that its memory will be susceptible to SEUs and there is no proper way of checking and healing it. Of course, the scrubbing unit will always be implemented in TMR, as this is the most critical part the device.

So far, no radiation tests have been carried out with the TMR'd firmware and the blind memory scrubbing. We are planning to schedule tests for the next year, which will give a better view of the effectiveness of the methods. This will then lead to our final decision and a firmware version that is robust enough for the use in the ATLAS experiment at nominal luminosity.

## ACKNOWLEDGMENT

It is a pleasure to thank all my colleagues for the good work and the friendly and helpful spirit in our institute. Special thanks to Robert Richter and Bradley Weber for the delighting and rich discussions about several topics, including FPGAs, radiation damages and their remedies. The nice layout of the poster was done by my dear colleague Agnes Rudert. Thank you very much!

## REFERENCES

[1] Xilinx, "TMRTool User Guide", UG156 (v2.2) September 12, 2007.
[2] Brendan Bridgford, Carl Carmichael, and Chen Wei Tseng, "Single-Event Upset Mitigation Selection Guide", XAPP987 (v1.0), March 18, 2008
[3] Carl Carmichael, and Chen Wei Tseng, "Correcting Single-Event Upsets with a Self-Hosting Configuration Management Core", XAPP989 (v1.0), April 2, 2008
[4] Ken Chapman, "SEU Strategies for Virtex-5 Devices", XAPP864 (v2.0) April 1, 2010
[5] Federico Faccio, "Radiation effects in deep submicron CMOS technologies", ESE seminar, Jan 2010
[6] R. Ball, T.Dai, J.Chapman, "Preliminary Radiation Testing Results Of the ATLAS CSM", March 31, 2005
[7] Paul Graham, Michael Caffrey, Jason Zimmerman, D. Eric Johnson, "Consequences and Categories of SRAM FPGA Configuration SEUs", Military and Aerospace Programmable Logic Devices International Conference, Washington DC 9/9-9/11/2003
[8] David R. Czajkowski, Praveen K. Samudrala, and Manish P. Pagey, "SEU Mitigation for Reconfigurable FPGAs", Aerospace Conference, 2006 IEEE, page 7 pp.
[9] Hiemstra, D.M. Chayab, F. Szajek, "Dynamic single event upset characterization of the Virtex-II and Spartan-3 SRAM field programmable gate arrays using proton irradiation", Radiation Effects Data Workshop, 2004 IEEE, page 79-84
[10] R. Katz, K. LaBel, J.J. Wang, B. Cronquist, R. Koga, S. Penzin, and G. Swift, "Radiation Effects on Current Field Programmable Technologies", IEEE TNS/NSREC 1997
[11] Michael Caffrey, Paul Graham, Eric Johnson, Michael Wirthlin, Carl Carmichael, "Single-Event Upsets in SRAM FPGAs", Military and Aerospace Applications of Programmable Logic Devices (MAPLD), Laurel MD, USA, September 2002
[12] Y. Arai,g B. Ball, j M. Beretta,d H. Boterenbrood, et al., "ATLAS Muon Drift Tube Electronics", 2008 JINST 3 P09001