# Dynamic Partial Reconfiguration of FPGA for SEU Mitigation and Area Efficiency

**Vijay G. Savani, Akash I. Mecwan , N. P. Gajjar**
Institute of Technology, Nirma University
vijay.savani@nirmauni.ac.in, akash.mecwan@nirmauni.ac.in , nagendra.gajjar@nirmauni.ac.in

## Abstract

The fast growing VLSI industry demands new techniques for configuring the FPGA. When it comes to defence and space application the configuration of the FPGA becomes more crucial. When it is required to configure the FPGA automatically, the need arises of more sophisticated and fast techniques for reconfiguration of FPGA. In the space application, the effect of radiation changes the bit patterns in the SRAM cells of FPGA, so it is required to put FPGA into its original condition before SEU. Considering all the facts the paper discusses the mitigation techniques for Single Event Upset (SEU) through Dynamic Partial Reconfiguration of FPGA. It is also very useful to save area of the FPGA by reconfiguration. For the proof of concept up and down sampler are developed as a reconfiguration module and then used for Dynamic Partial Reconfiguration technique. The timing and area requirement of reconfiguration using various techniques is the major focus of the paper.

***Keywords:*** *Single Event Upset (SEU), Reconfiguration, Mitigation.*

## 1. Introduction

In the modern era the FPGAs are widely use to make the prototype of any system. In the space application FPGAs are more used because the designing with FPGA is easy and fast. Many times it is also required to reprogram the chip. FPGA is very flexible to reprogram. Xilinx Virtex FPGAs offer to exploit the features of dynamic and partial run-time reconfiguration.

One of the major motivations of this paper is to give the proof of concept of DPR to mitigate the soft-errors in FPGA designs when we use it for space application. Single event upset (SEU) is defined by NASA as "Radiation-induced errors in microelectronic circuits caused when charged particles (usually from the radiation belts or from cosmic rays) lose energy by ionizing the medium through which they pass, leaving behind a wake of electron hole pairs" [10]. SEUs are soft errors, and are nondestructive. An SEU may occur in analogue, digital, optical components, or may have effects in surrounding interface circuitry. FPGAs based on SRAM can be reprogrammed an unlimited number of times, even in the end-user system. The Single Event Upset occurs when radiation affects the transistors that are part of the look up table logic of the FPGA. If the lookup table is affected by radiation can change the bit values associated with the hardware made up of the current FPGA design. SEU is a change of state caused by ions or electro-magnetic radiation striking a sensitive node (area) in a micro-electronic device (Bit-Flip) as shown in the figure 1. The state change is a result of the free charge created by ionization in or close to an important node of a logic element. In these FPGAs, a multitude of latches, also called memory cells or RAM bits, define all logic functions and on-chip interconnects. Such latches are similar to the 6- transistor storage cells used in SRAMs, which has proved to be sensitive to single event upsets caused by high-energy neutrons [11].

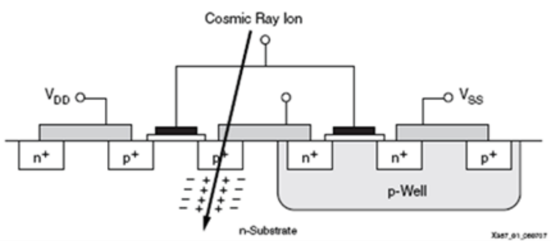


Fig 1. SEU in the FPGA

There are basically five area of the FPGA's CLB which susceptible to SEU as shown in the figure 2.

1. Upsets in the logic (LUT).
2. Upsets in the customization routing bits inside the CLB.
3. Upsets in the routing connecting CLBs and pins.
4. Upsets in the CLB flip-flops (flip-flops).
5. Upsets in BRAM.

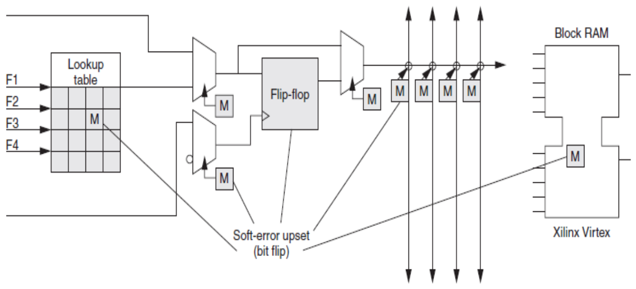


Fig 2. Area of the FPGA's CLB are affected by SEU

There are two main categories of radiation effect that are relevant for Static Random Access Memory (SRAM) Field-Programmable Gate Arrays (FPGAs) in space: Total-Dose Effects and Single-Event Effects (SEEs). Total-Dose Effects are cumulative effects that induce degradation of electrical parameters at the device, circuit, and system level. SEU falls in to SEE category. The technique of removing SEU is called mitigation. Various mitigation techniques are available to get the rid of SEU. Some of the mitigation techniques are as follows:

- Triple Modular Redundancy (TMR)
- TMR registers with Voters and Scrubbing
- Read back CRC
- Partial Reconfiguration
  - Static Partial Reconfiguration
  - Dynamic Partial Reconfiguration

Dynamic Partial Reconfiguration features open a wide area for designing systems exploiting this method, if for example run-time adaptive systems change their behavior on-demand initiated by processing input of data from external source. New approach using this feature makes it possible to use FPGAs with smaller configuration memory and consequently smaller chip size by storing configuration data on external memory. Thus it is possible to save cost and reduce power consumption because it does not actually use new modules of a system and do not allocate configuration memory and corresponding power consuming hardware.

Nevertheless power dissipation during reconfiguration has to be considered [2]. The next sections of the paper mainly concentrate on the Partial Reconfiguration of the FPGA.

## 2. Partial Reconfiguration

Partial Reconfiguration is reconfiguring only the part of FPGA. When the full FPGA is configured it requires more time to get reconfigured but when part of FPGA is reconfigured the process becomes fast. In the partial reconfiguration FPGA is divided in two regions one is static part which will never change in reconfiguration and the second is dynamic part which will change as and when require. This dynamic part can be reconfigured by two ways:

- Static Partial Reconfiguration
- Dynamic Partial Reconfiguration

In Static Partial Reconfiguration the system will stop working at the time of reconfiguration and the part which is required to be reconfigured will only be programmed again. While in the Dynamic Partial Reconfiguration the system will keep on running when the dynamic part is getting reconfigured.

In Dynamic Partial Reconfiguration, while run-time, different configurations were sent via the configuration access port to the configuration memory. Both logic elements and routing resources can be influenced and adapted to a new functionality and routing. It becomes clear that changes influence the behaviour of functions in the neighbourhood if signal lines cross the area where partial reconfiguration occurs. Figure 3 shows schematically the process of dynamic and partial hardware reconfiguration.

On the lower par of the figure the timeline with the different time steps can be seen. At t0 the reconfigurable hardware is started and contains no configuration information. After a time period until t1 the configuration data is transferred to the device. After the successful transfer of configuration data the device starts with data processing within Configuration A and B. These two configurations can be algorithms, finite state machines or other processing elements. At t2 the functionality within configuration B is stopped and substituted by another configuration. The configuration C is loaded to the device while configuration A stays in operation. After transferring the data for configuration C at time step t3, the reconfigurable hardware proceeds data processing with configuration A and C. This small example shows the benefits of dynamic and partial reconfiguration. The architecture can be adapted to the requirements of the applications while parts of the system stay operative. A system exploiting this feature is described in [1].

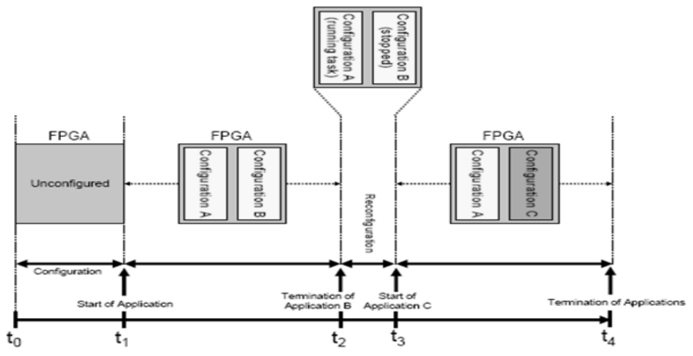


Fig 3. Dynamic Partial Reconfiguration

## 3. Implementation

In Partial Reconfiguration the portion of the design which gets reconfigured, is called Reconfiguration Module (RM). To explore the efficiency of partial reconfiguration we have chosen one reconfiguration region and two RM, one is the digital up-sampler and another is down-sampler. They were designed and configured using static and dynamic partial reconfiguration. Figure 4 shows the RTL design of Up-Sampler and figure 5 shows the RTL design of Down-Sampler.

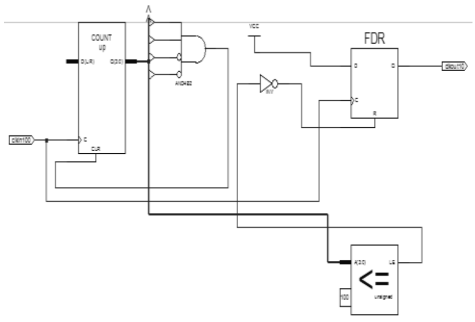


Fig 4. RTL of Up –Sampler Design

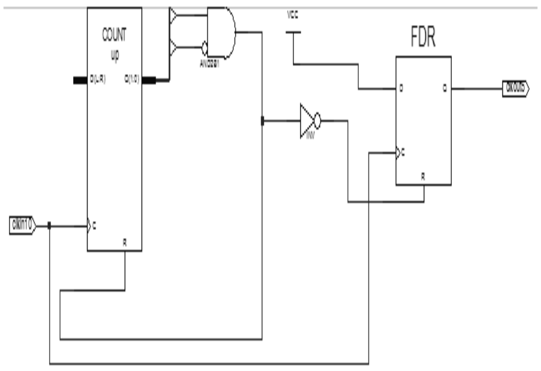


Fig 5. RTL of Down –Sampler Design

The static part of the design consists of inputs to the system, start and stop control and a 100 MHz clock signal. While the dynamic part consist of up and down sampler as shown in figure 4 & 5. Systems implementing partial self-reconfiguration are still uncommon in most FPGA applications due to the relatively large overhead required for the initial analysis phase to find a working tool flow. Xilinx has recently been attempting to make partially reconfigurable systems easier to implement. Xilinx has added more documentation to their website and has produced a more user-friendly floor planning tool used for partial reconfiguration, but much of this information is currently limited to an Early Access area that requires approval for access. A dynamic partial reconfiguring system on a Xilinx Virtex FPGA is implemented by making use of the Internal Configuration Access Port (ICAP). A partial bitstream is written to the ICAP, which then reconfigures the specified portions of the FPGA with the new logic. Communication with the ICAP can be implemented through soft core processor or through a custom VHDL logic design. Using the custom logic or by writing the custom routine in the soft core processor we can frequently reconfigure the portion of the design which is very critical and we can mitigate or remove the accumulation of single bit-flip error which we called as SEU.

These two reconfiguration modules are configured in the FPGA Xilinx Virtex5 XC5VLX110T-1ff1136 using static and dynamic partial reconfiguration. The designs are compared for reconfiguration timing. The total time of configuring the full FPGA is also compared. Figure 6 shows the dynamic reconfigurable portion in the final design.

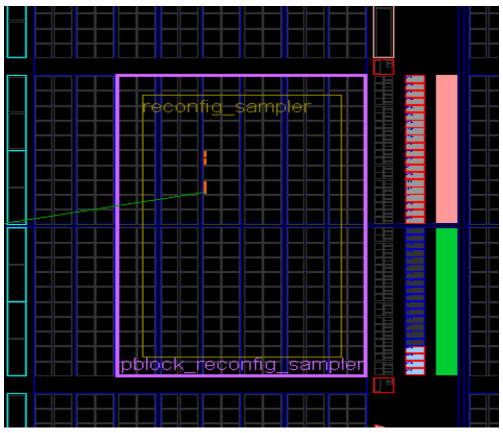
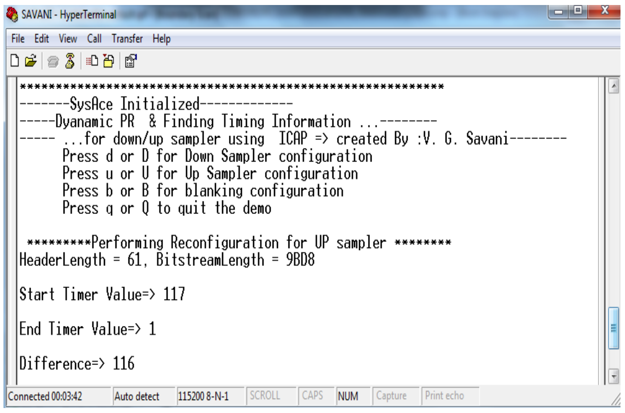


Fig 6. Pblock area of Dynamic part of Design

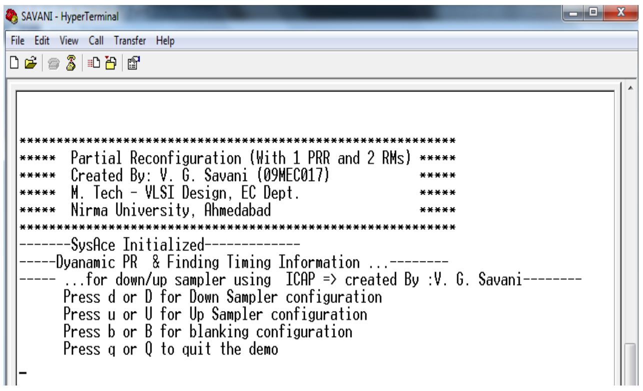
## 4. Results

The figure 7 shows the reconfiguration menu which the part of routine written in the soft core processor and which having the UART interface enable for the user interaction with core and ICAP to do the reconfiguration. The figure 8 shows the timing requirement for reconfiguration of the RM into the design.



```
************************************************************
-------SysAce Initialized-------------
-----Dyanamic PR  & Finding Timing Information ...--------
----- ...for down/up sampler using  ICAP => created By :V. G. Savani--------
      Press d or D for Down Sampler configuration
      Press u or U for Up Sampler configuration
      Press b or B for blanking configuration
      Press q or Q to quit the demo

 *********Performing Reconfiguration for UP sampler ********
HeaderLength = 61, BitstreamLength = 9BD8

Start Timer Value=> 117

End Timer Value=> 1

Difference=> 116
```

Fig 7. Partial Reconfiguration of RM Module



```
************************************************************
*****  Partial Reconfiguration (With 1 PRR and 2 RMs) *****
*****  Created By: V. G. Savani (09MEC017)            *****
*****  M. Tech - VLSI Design, EC Dept.                *****
*****  Nirma University, Ahmedabad                    *****
************************************************************
-------SysAce Initialized-------------
-----Dyanamic PR  & Finding Timing Information ...--------
----- ...for down/up sampler using  ICAP => created By :V. G. Savani--------
      Press d or D for Down Sampler configuration
      Press u or U for Up Sampler configuration
      Press b or B for blanking configuration
      Press q or Q to quit the demo
-
```

Fig 8. Timing requirement for the configuration of RM

Table 1 is total device utilization summary when we combine the module into single design without reconfiguration. Table 2 is area comparisons for the up and down converter and shows the % saving in area when we use only one module into the design and doing the reconfiguration of the required module into the design.

Table 1. Device Utilization Summary

(Device Used: Virtex5, 5vlx110tff1136-1)

| **XILINS VIRTEX5 XC5VLX110T-1FF1136** | | | |
|---|---|---|---|
| **LOGIC UTILIZATION** | **USED** | **AVAILABLE** | **UTILIZATION** |
| NUMBER OF SLICE REGISTERS | 3255 | 69120 | 4% |
| NUMBER OF SLICE LUTS | 3245 | 69120 | 4% |
| NUMBER OF FULLY USED BIT SLICES | 1328 | 5172 | 25% |
| NUMBER OF BONDED IOBS | 22 | 640 | 3% |
| NUMBER OF BLOCK RAM/FIFO | 18 | 148 | 12% |
| NUMBER OF BUFG/BUFGCTRLS | 5 | 32 | 15% |
| NUMBER OF DCM_ADVS | 1 | 12 | 8% |
| NUMBER OF DSP48ES | 3 | 64 | 4% |

Table 2. Area Comparison Up/Down Sampler

| **Device Used: Virtex5 5vlx110tff1136-1** | **Device utilization summary:** | **Available** | **Up sampler alone** | **Down sampler alone** | **Combine up and down sampler** | **Average % of Saving** |
|---|---|---|---|---|---|---|
| **Slice Logic Utilization** | **Number of Slice Registers** | 69120 | 3 | 8 | 11 | **45.46 %** |
| | **Number of Slice LUTs** | 69120 | 4 | 9 | 12 | **50.00 %** |
| | **Number used as Logic** | 69120 | 4 | 9 | 12 | **50.00 %** |
| **Slice Logic Distribution** | **Number of LUT Flip Flop pairs used** | -- | 7 | 17 | 12 | **8.34 %** |

Table 3 shows the comparisons of the time required loading the Reconfiguration Modules into the FPGA and the time required to load the entire design as a whole.

Table 3. Time Requirement for Reconfiguration

| **Reconfigure Module** | **Size of bit file** | **Time Require to Reconfigure (Using No of Count)** | **No of Processor Clock Cycle** |
|---|---|---|---|
| Down Sampler | 43.379 Kbytes | 0.99 µsec (990 nsec) | 1 Cycle |
| Up Sampler | 39.993 Kbytes | 1.16 µsec (1160 nsec) | 1 Cycle |
| Blank | 39.863 Kbytes | 0.97 µsec (970 nsec) | 1 Cycle |

## 5. Conclusion

This method is extremely useful for the space application as the SEU has to be mitigated. The basic conclusion that follows from the paper is that the time requirement in case of dynamic partial reconfiguration is very less compare to the full reconfiguration or the static partial reconfiguration. Also and small FPGA can also be used for larger design by dividing the design in the modules and we load the module into the FPGA as and when required by the application.

**Reference**

[1] M. Ullmann, M. Huebner, B. Grimm, J. Becker: “An FPGA Run-Time System for Dynamical On-Demand Reconfiguration”, RAW04, Santa Fee.

[2] M. Hübner, J. Becker: “Exploiting Dynamic and Partial Reconfiguration for FPGAs - Toolflow, Architecture and System Integration”

[3] Cristiana Bolchini, Davide Quarta, Marco D. Santambrogio: “SEU Mitigation for SRAM-Based FPGAs through Dynamic Partial Reconfiguration”

[4] Davin Lim and Mike Peattie: Two types for partial reconguration: Module based or small bit manipulations, xapp290 (v1.0). May 2002.

[5] Davin Lim and Mike Peattie: Difference-based partial reconguration, xapp290 (v2.0). Dec 2007.

[6] Xilinx. Virtex 5 FPGA Configuration user guide, UG191 (v3.8), August 14, 2009.

[7] Xilinx. Virtex 5 FPGA user guide, UG190, November 5, 2009.

[8] http://trs-new.jpl.nasa.gov/dspace/bitstream/2014 /40763/1/08-09.pdf

[9] http://www.sti.nasa.gov/thesfrm1.