Enhancing the Reliability of COTS SRAM-based FPGAs with Microreconfiguration for SEU Mitigation in Space Applications

Alexandra Kourfali, Dirk Stroobandt, David Merodio Codinachs

ESTEC TEC-EDM
11/03/2016

Issue/Revision: .
Reference:
Status:

Introduction

Space Qualified Vs COTS

- Space Qualified are products specifically designed, qualified and tested for space applications.

- Commercial off the shelf (COTS) are products that are standard manufactured commercial products rather than custom products.
  - use these parts very carefully, test extensively
  - use for missions with shorter lifetime
  - use for missions with less quality constraints.

Introduction

COTS electronics

1. ASICs
   - Extensively used
2. FPGAs
   - High Flexibility
   - Good Performance
   - Runtime Reconfigurability

✦ Can we have COTS FPGA designs that are safe for space?

Introduction

Which benefits can reconfiguration bring in space applications?

1. Reconfiguration
   - change the design (if needed) in the future

2. Partial reconfiguration
   - use time & space partitioning dynamically to fit a big design in a smaller FPGA

3. Microreconfiguration
   - compress a big design by reconfiguring only small fractions of an FPGA
     - ✦ Microreconfigure one LUT on the fly to recover from a failure

Content

1. **Limitations** of COTS electronics in space applications
2. **Radiation Effects** in SRAM-based FPGAs
3. **Microreconfigurations** as a mean of SEU mitigation
4. Introduce **Micro-Scrubbing** at the FPGA flow
5. Architectural **Constraints**
    a. Logic
    b. Routing
6. **Architectural Overview** of the system
    a. Design Time
    b. Operating Time
7. Micro-scrubbing **CAD tool** for SEU mitigation
8. Conclusions

Limitations of COTS for space

1. Not extensively tested for harsh radiation environments

2. Radiation effects
   - **SEU**, SET, SEL, SEFI etc

3. Vulnerabilities
   - Volatile configuration memory prone to radiation-induced errors: Soft and Hard Errors
   - Security risk that can be inherited in the system
   - A few companies perform security reviews on every commercial application
   - Bad implementations, aging effects, attacks

✦Mitigate SEU in an FPGA to reduce probability of failure

SEU mitigation in SRAM-Based FPGAs

Single-Event Effect issues and possible mitigation solutions

	TMR	Scrub	Reconfig	Voters
BRAM	✓	✓		
CLB	✓			
IOB	✓			
Configur. memory		✓	✓	
Config. controller				✓
DSP	✓			

1. TMR- introduces specialisation overhead
2. Scrubbing - introduces time overhead. Needs resynchronisation.
3. Reconfiguration (partial or full) introduces specialisation overhead

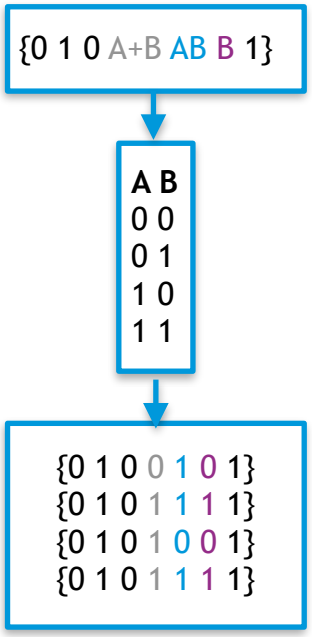
✦ Can we scrub with microreconfiguration?

Microreconfigurations for SEU Mitigation

Which benefit microreconfiguration brings in space applications?

- What is microreconfiguration?

A technique to reconfigure very small parts of the device. It only changes a small set of configuration bits only.



1. for logic resources
   - Dynamically Specialise the values of the LUTs present in FPGAs
   - Introduce logic in LUTs
   - Custom mapping for abstract primitives
   - Xilinx Tool Flow
2. for routing resources
   - Dynamically Specialise the values of the LUTs and the routing resources present in FPGAs
   - Introduce logic in LUTs and FPGA interconnects
   - Custom mapping, place and route for abstract primitives
   - VPR tool flow

Micro-Scrubbing via microreconfiguration

Which benefit microscrubbing brings in space applications?

1. Where?
    - critical bits that are implemented as parameters

      CLBLL_X4Y82 SLICE_X6Y82

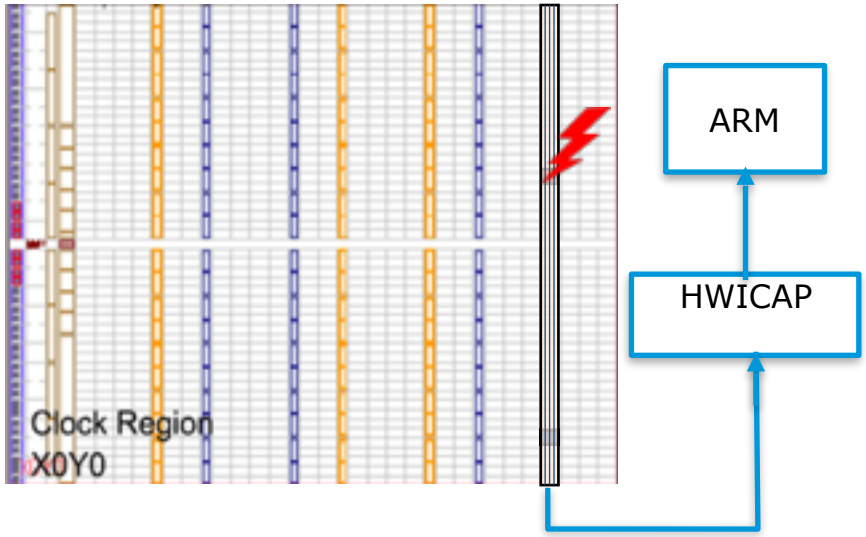
      pip INT_X4Y82 NL2MID0 -> BYP5 → - - parameter

2. When?
    - When the embedded monitors detect a SEU
    - Periodically

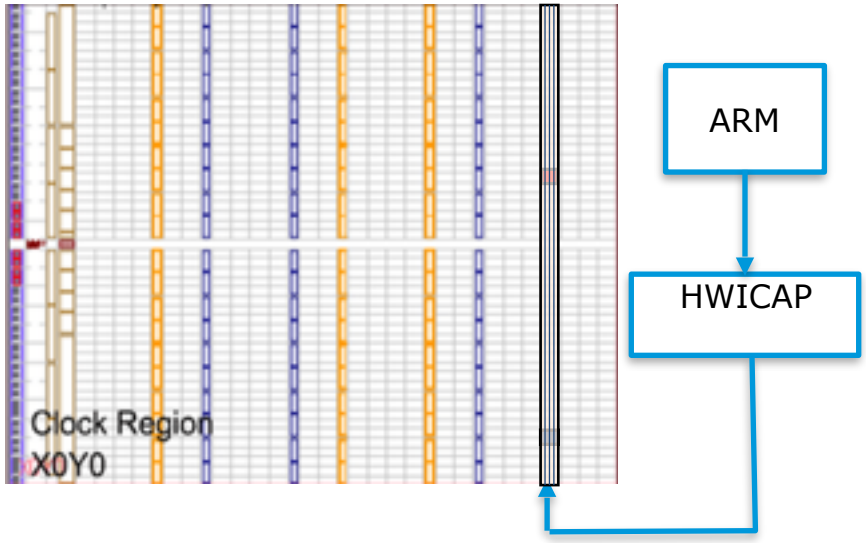
3. What?
    - Read the value (SEU)
    - Scrub 1 LUT by reseting its boolean values
    - Write back the correct value

Micro-Scrubbing via microreconfiguration

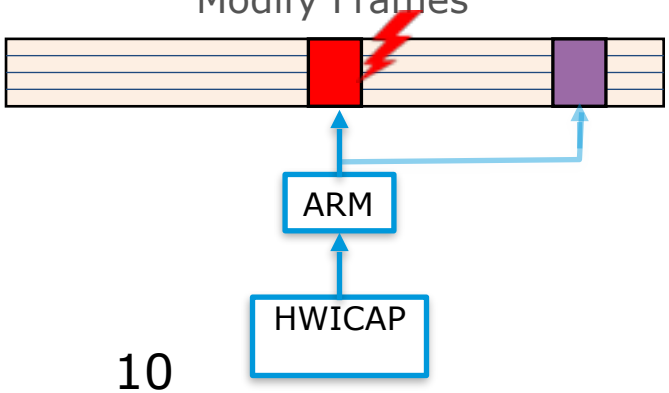
Read Frames



Write Frames



Modify Frames



- read frames: erroneous value to be scrubbed
- 1 LUT is reconfigured when we modify the frames
- write back the initial value (reset boolean functions)

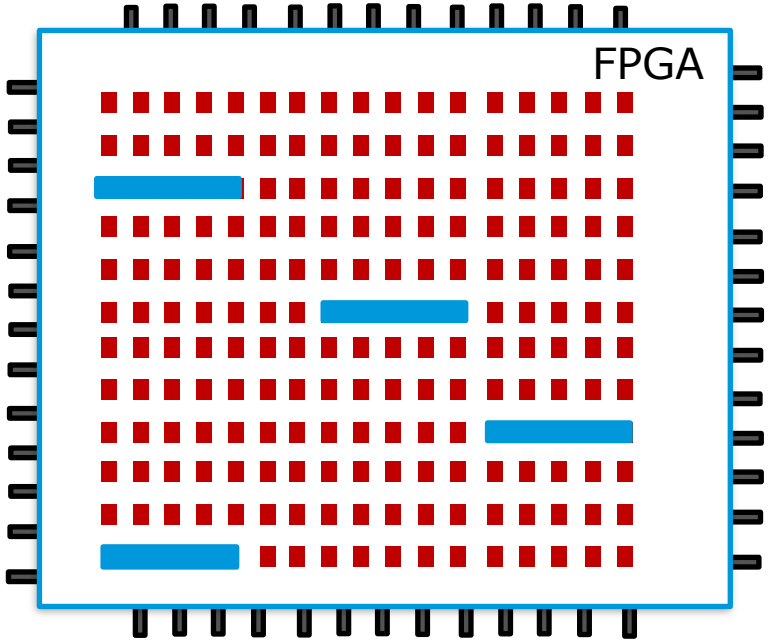
Proposed Architecture

Multiple layers that add integrated SEU mitigation

- Free resources are used

1. Design Time
    - Add monitors
    - Add virtual infrastructure
    - Enable Microreconfigurations

2. Operating Time
    - Capture SEU
    - Micro-Scrubbing



Proposed Architecture

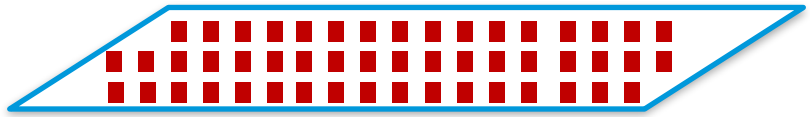
Multiple layers that add integrated SEU mitigation

- Free resources are used

1. Design Time
   - Add monitors
   - Add virtual infrastructure
   - Enable Microreconfigurations

2. Operating Time
   - Capture SEU
   - Micro-Scrubbing

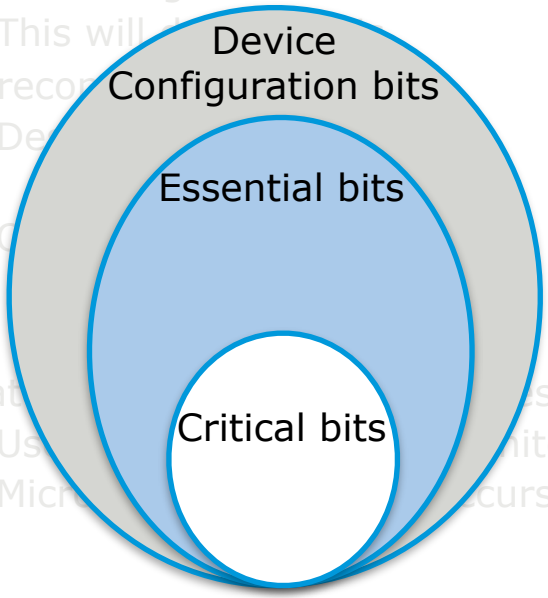
- How can I recover from SEU via scrubbing?



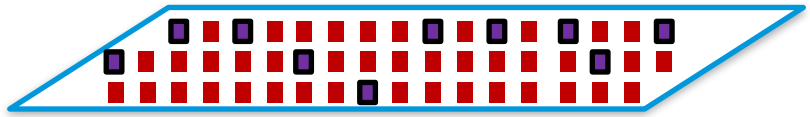
Proposed Architecture

Multiple layers that add integrated SEU mitigation
- Only a fraction of the signals are considered

1. Veriplace analysis
   - Only a subset of signals are critical



- A bit is critical if it affects a resource of the circuit and the effect can be propagated to the output



Proposed Architecture

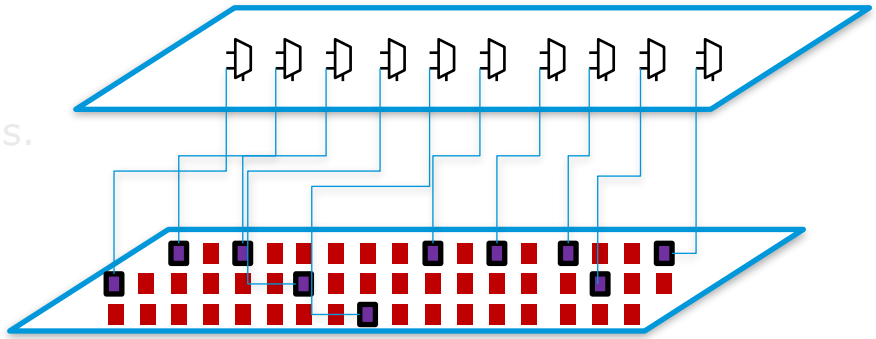
Multiple layers that add integrated SEU mitigation
- Critical Bit Parameterisation

1. Veriplace analysis
   - Only a subset of signals are critical
2. Add monitoring infrastructure
   - This will detect SEUs

Microreconfigurations
- Design Time
- Generalised Stage Operating Time
  - Detect SEU
  - Micro-Scrub

Allocate monitors at free resources
- Use spare resources to monitor SEUs.
- Micro-scrubbing if SEU occurs

- MUXs added at a virtual intermediate level that has no impact on the design



Proposed Architecture

Multiple layers that add integrated SEU mitigation
- Microreconfigurations

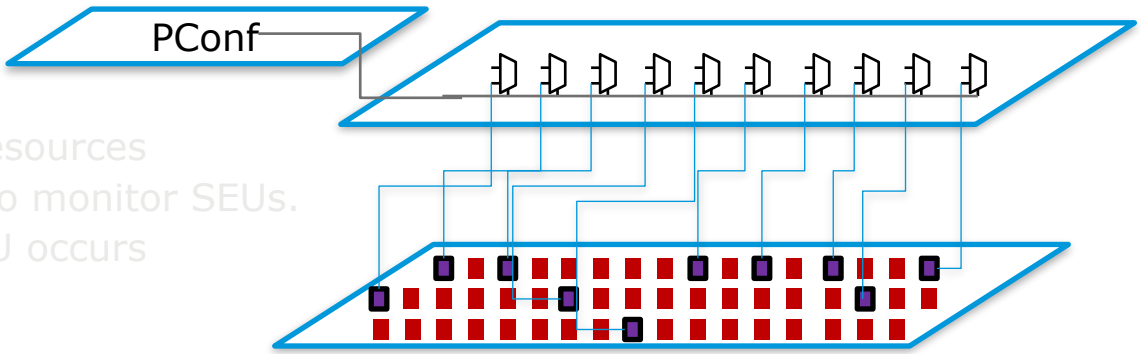
1. Veriplace analysis
   - Only a subset of signals are critical
2. Add monitoring infrastructure
   - This will detect SEUs
3. Microreconfigurations
   - Design Time
     - Generalised Stage
   - Operating Time
     - Detect SEU
     - Micro-Scrub

Allocate monitors at free resources

Use spare resources to monitor SEUs.

Micro-scrubbing if SEU occurs

- PConf tool flow supports a virtual intermediate low overhead level of logic

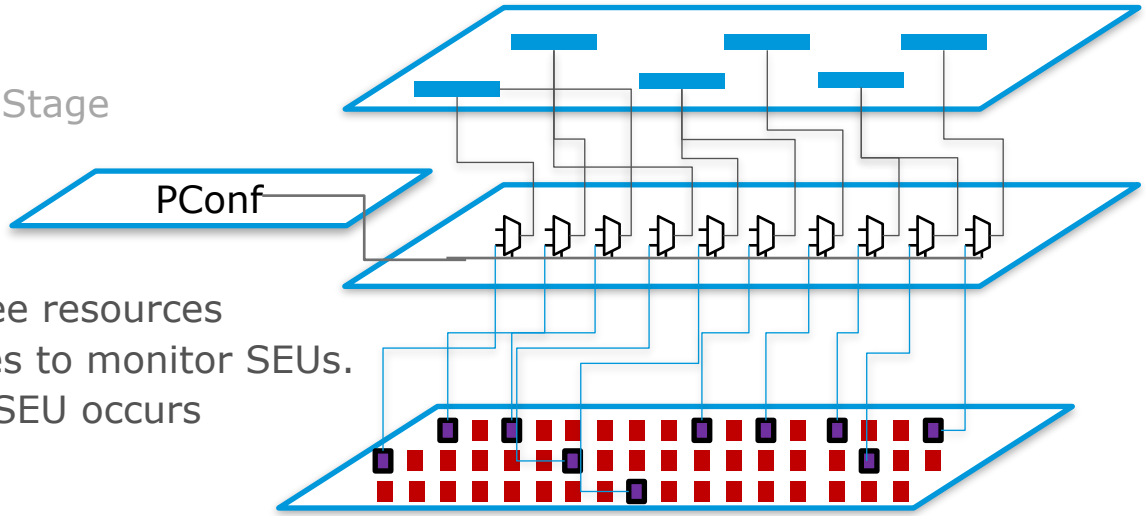


Proposed Architecture

Multiple layers that add integrated SEU mitigation
- Microreconfigurations

1. Veriplace analysis
   - Only a subset of signals are critical
2. Add monitoring infrastructure
   - This will detect SEUs
3. Microreconfigurations
   - Design Time
     - Generalised Stage
   - Operating Time
     - Detect SEU
     - Micro-Scrub
4. Allocate monitors at free resources
   - Use spare resources to monitor SEUs.
   - Micro-scrubbing if SEU occurs

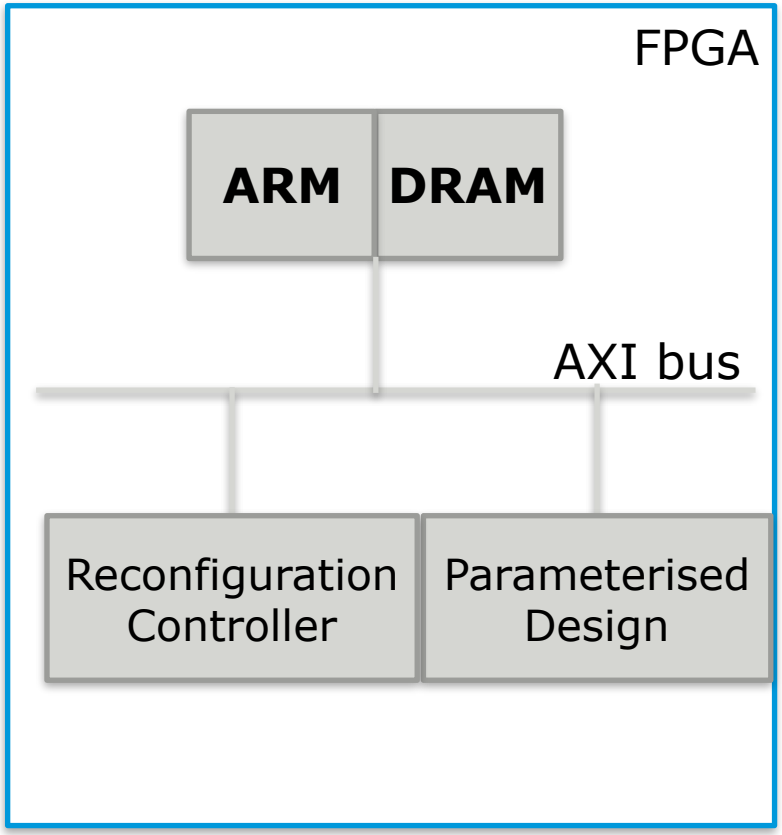
✦ use unused FPGA resources to embed monitors



Proposed Architecture

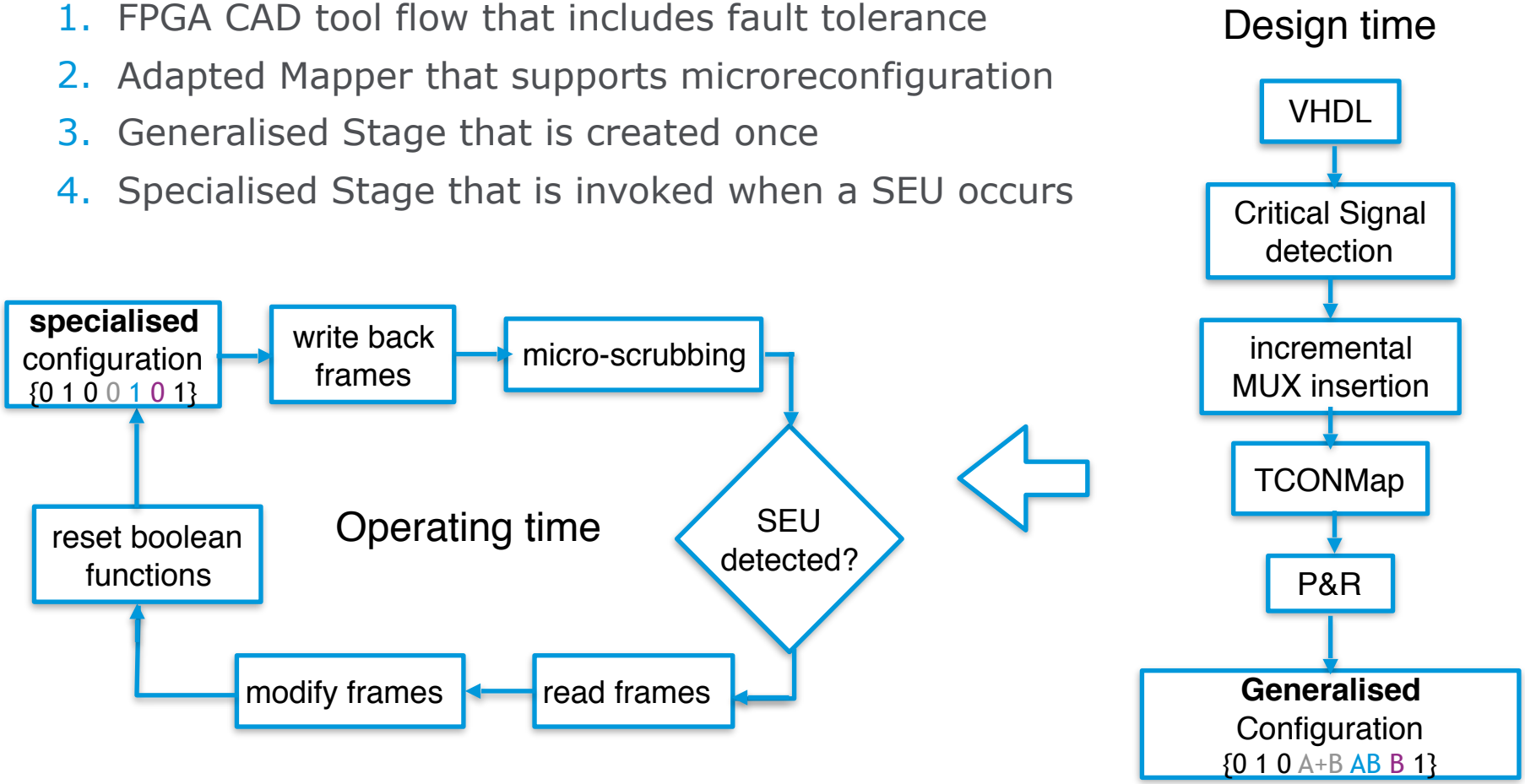
Self Reconfigurable Platform for SEU mitigation - HWICAP reconfiguration controller

1. Zynq-SoC (XC7Z020 -CLG484-1, ZedBoard)
    - COTS FPGA device
2. ARM Cortex-A9 (667 MHz)
    - Controls microreconfiguration
3. AXI bus (100 MHz)
    - Connects the system
    - Data transfer
4. DRAM
    - Stores Boolean functions
5. Parameterised Design
    - With integrated monitoring
6. Reconfiguration Controller
    - Performs micro-scrubbing



Towards a CAD tool

1. FPGA CAD tool flow that includes fault tolerance
2. Adapted Mapper that supports microreconfiguration
3. Generalised Stage that is created once
4. Specialised Stage that is invoked when a SEU occurs



Conclusions

1. Introduce a fast detect and repair system for COTS
2. System Embedded in the design. Provoked after a SEU.
3. Run time reconfigurable system. Can operate real time. No resynchronisation needed after scrubbing.
4. Gains of microreconfiguration:
   - up to 50% less area
   - up to 35% higher clock frequency
   - up to 5 orders of magnitude less generation time
   - less memory (compressed configurations)

Questions?

Enhancing the Reliability of COTS SRAM-based FPGAs with Microreconfiguration for SEU Mitigation in Space Applications

Alexandra Kourfali
alexandra.kourfali@esa.int
Thank you