Fault injection for space: FT-Unshades2 updates, experiences and roadmap

Hipólito Guzmán-Miranda
María Muñoz-Quijada
Luis Sanz

SEFUW, 4th edition
9-11th April 2018

Introduction

- Fault Injection: a promising technique to compute the AVF (Architectural Vulnerability Factor) of electronic designs

- Benefit 1: AVF computation
- Benefit 2: Detect most sensitive regions
- Benefit 3: Hierarchical analysis
- Benefit 4: Verification of inserted protections
- Benefit 5: Detect collapsed TMRs
- Benefit 6: Detect defects in reset strategy
- Benefit 7: Check quality of workloads

Introduction

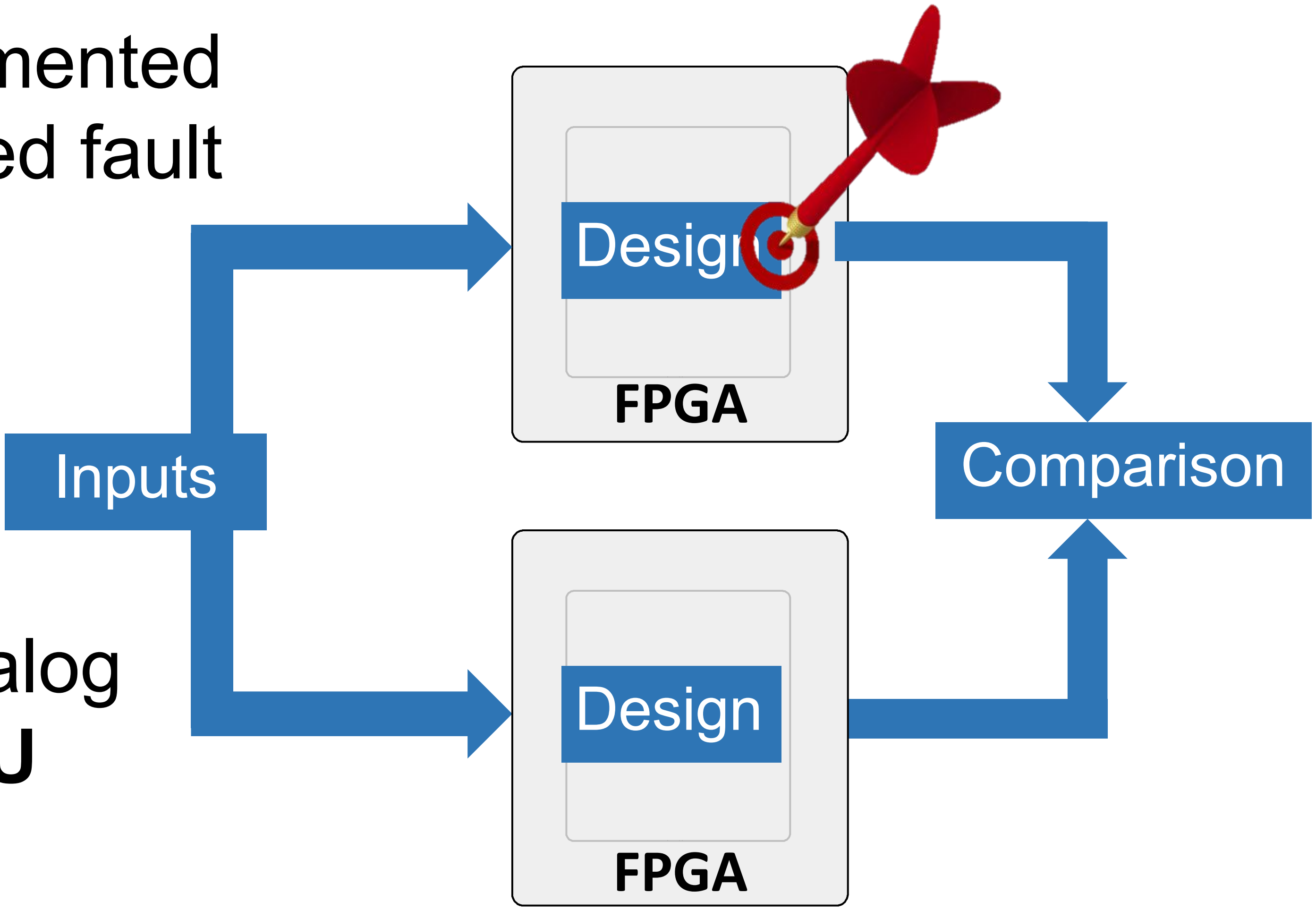
- Fault Injection: a promising technique to compute the AVF (Architectural Vulnerability Factor) of electronic designs

- Concern 1: learning curve, effort to use
- Concern 2: is it really accurate?
- Concern 3: what useful information can be extracted from the Fault Injection experiments?

Introduction

FT-Unshades2: Non-instrumented FPGA-based fault injection Emulator (SEUs).

Also an analog utility, **AFTU** (SETs).

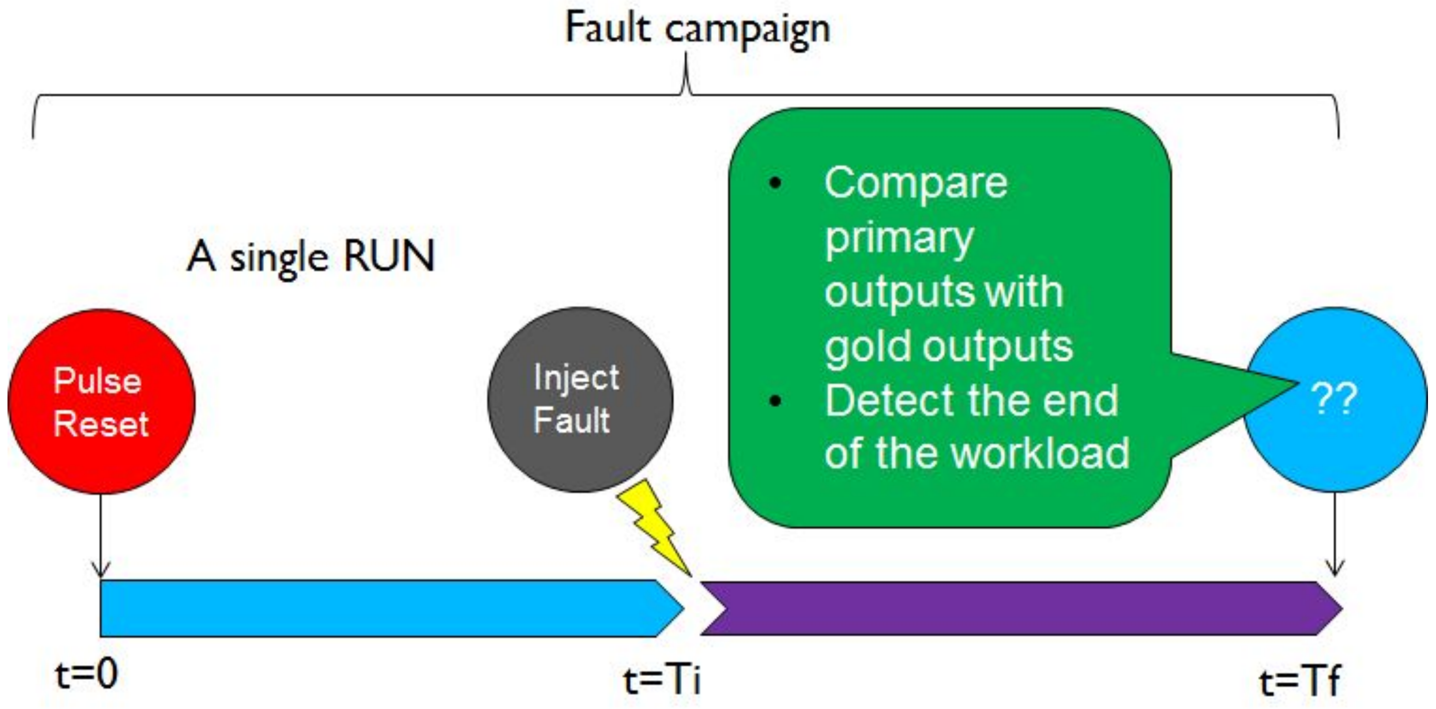


How does it work?

Design is prepared for the target FPGA (Virtex-5)

A campaign consists of multiple runs

Run: execution of test vectors + injection(s)


Fault campaign
A single RUN
Pulse Reset
Inject Fault
Compare primary outputs with gold outputs
Detect the end of the workload
??
t=0
t=Ti
t=Tf

How to use it?

Implement a design using the standard design flow considering:

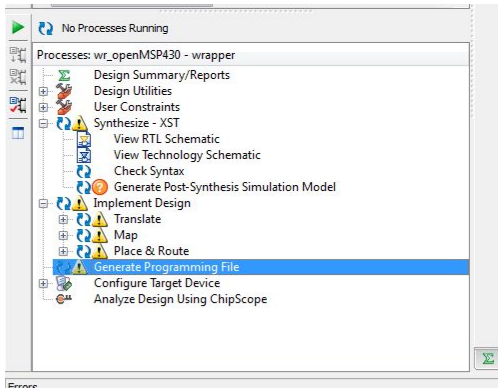
- Pinout fixed by the PCB (own tool generates .ucf file)
- Avoid DLLs and packing registers into I/Os
- Leave SelectMap port open, generate bit allocation file (.ll)
- Generate a bitstream (.bit)

Stimuli set is obtained using a standard simulator

- VCD is converted to internal I/O format

Generate design.pin

Make design.ucf

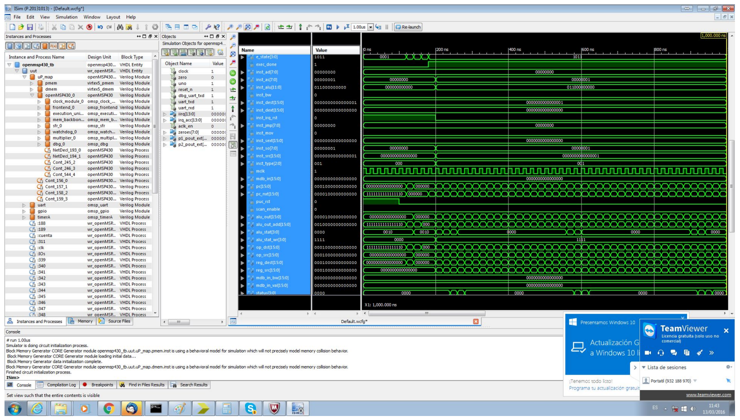


No Processes Running

Processes: wr_openMSP430 - wrapper

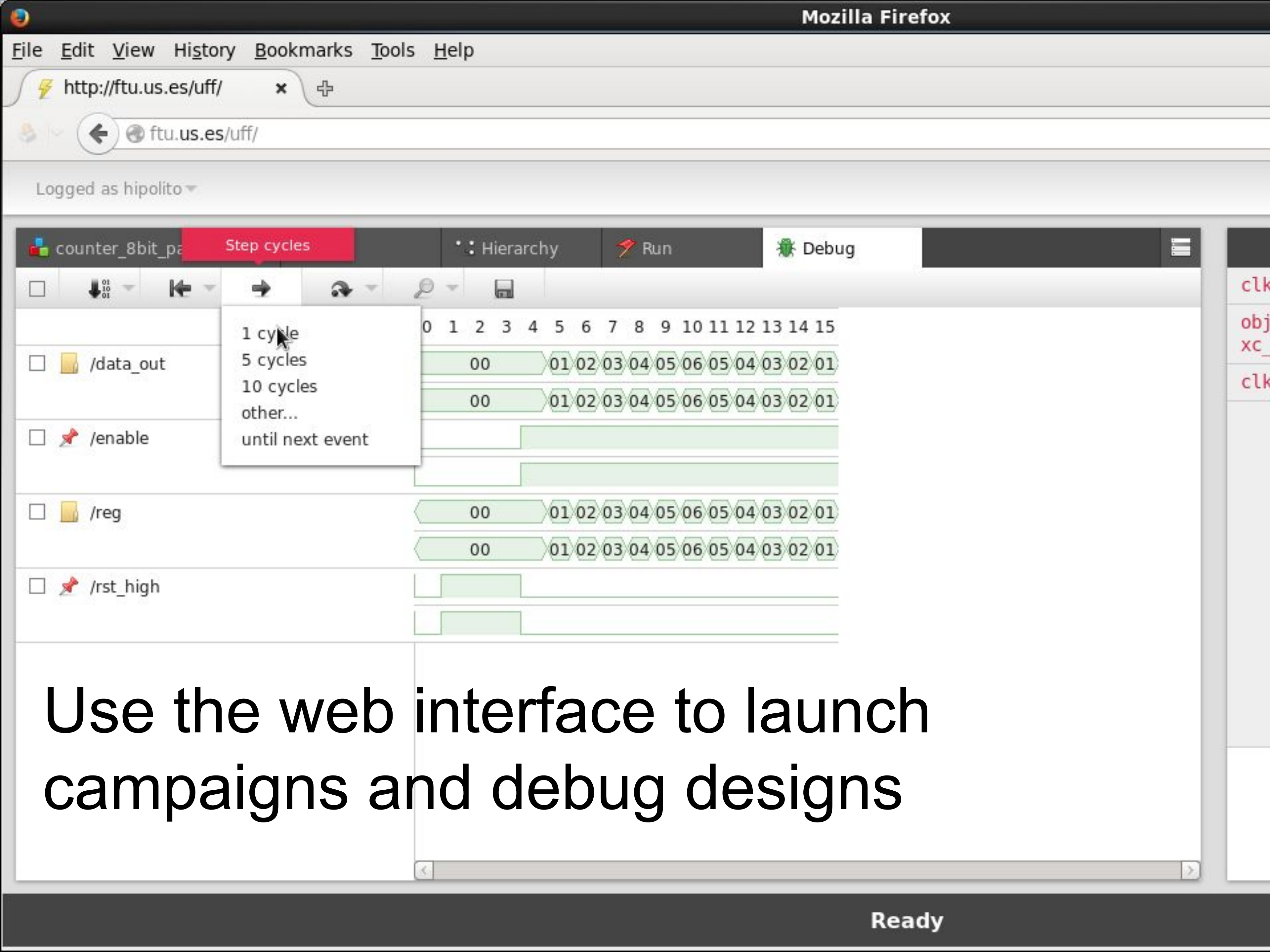
- Design Summary/Reports
- Design Utilities
- User Constraints
- Synthesize - XST
  - View RTL Schematic
  - View Technology Schematic
  - Check Syntax
  - Generate Post-Synthesis Simulation Model
- Implement Design
  - Translate
  - Map
  - Place & Route
- Generate Programming File
- Configure Target Device
- Analyze Design Using ChipScope

Errors



Generate design.vcd

Make design.dat



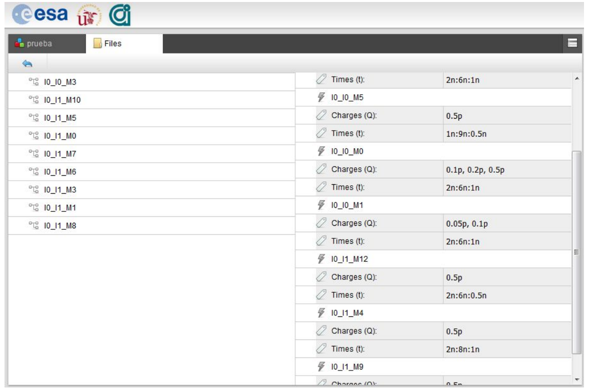
```
0  1  2  3  4  5  6  7  8  9  10 11 12 13 14 15
/data_out    00  01 02 03 04 05 06 05 04 03 02 01
             00  01 02 03 04 05 06 05 04 03 02 01
/reg         00  01 02 03 04 05 06 05 04 03 02 01
             00  01 02 03 04 05 06 05 04 03 02 01
```

Use the web interface to launch campaigns and debug designs

Updates

Extension of the injection coverage:

- Injections can now be performed now in:
  - User Flip-flops
  - **Block RAMs**
  - **Distributed RAMs**
  - Configuration bits (essential bits)
    - Almost-blind injection
- Injection in embedded RAMs important for microprocessor reliability assessment



I0_I0_M3
I0_I1_M10
I0_I1_M5
I0_I1_M0
I0_I1_M7
I0_I1_M6
I0_I1_M3
I0_I1_M1
I0_I1_M8

Times (t):	2n:6n:1n
I0_I0_M5	
Charges (Q):	0.5p
Times (t):	1n:9n:0.5n
I0_I0_M0	
Charges (Q):	0.1p, 0.2p, 0.5p
Times (t):	2n:6n:1n
I0_I0_M1	
Charges (Q):	0.05p, 0.1p
Times (t):	2n:6n:1n
I0_I1_M12	
Charges (Q):	0.5p
Times (t):	2n:6n:0.5n
I0_I1_M4	
Charges (Q):	0.5p
Times (t):	2n:8n:1n
I0_I1_M9	

Integration of the analog tool (AFTU) in the web-based user interface

Experiences

Analysis of system-level propagation of output damages

Evolve the typical cycle-by-cycle comparison model

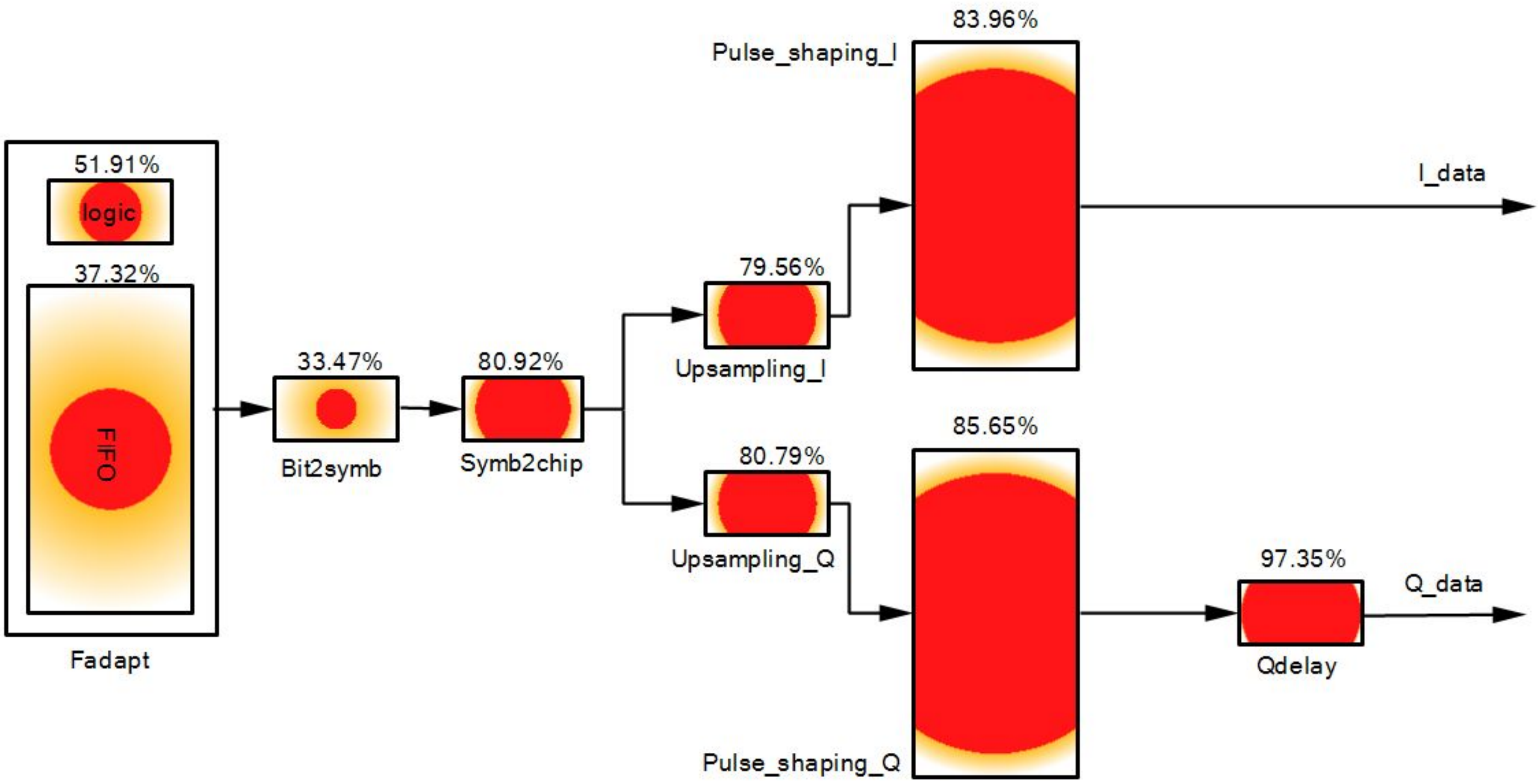
-> take into account how the erroneous outputs affect the environment

-> typically involves some kind of post-processing of the faulty output

-> will vary depending on application

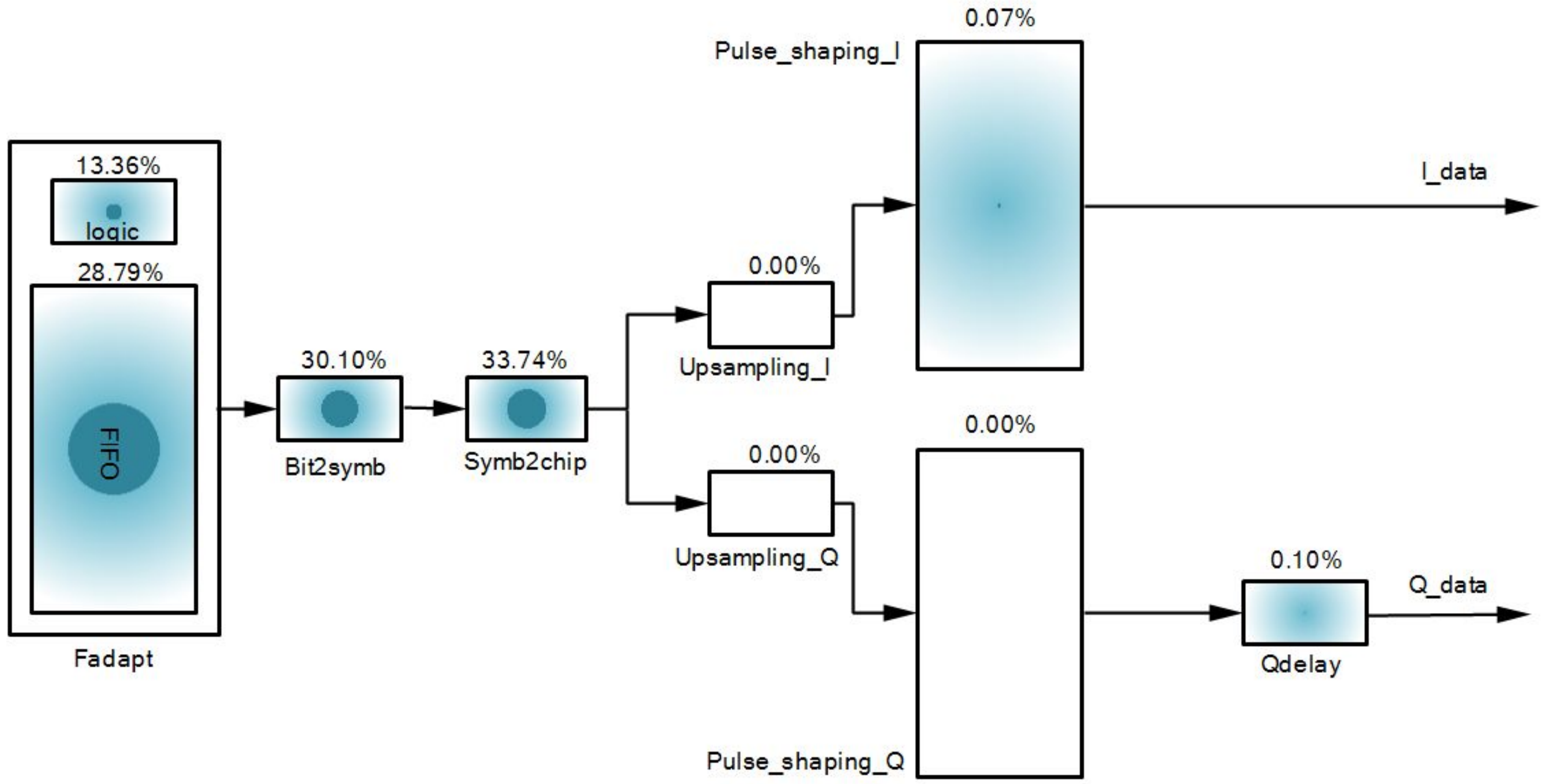
Experiences: Design of a zigbee physical layer tx

The design seemed very sensitive ...



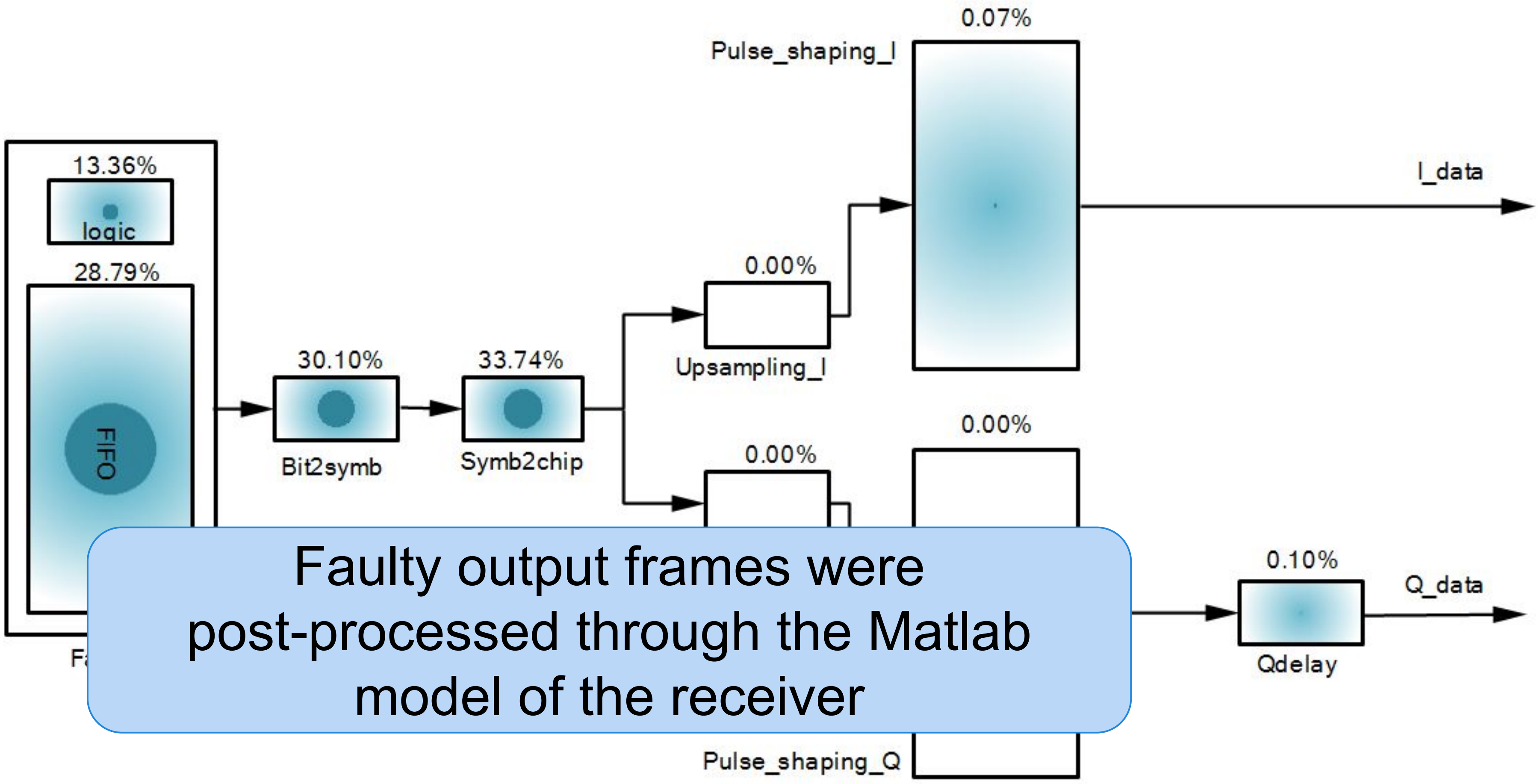
Experiences: Design of a zigbee physical layer tx

… but, can the rx recover from faulty frames?



Experiences: Design of a zigbee physical layer tx

… but, can the rx recover from faulty frames?



Faulty output frames were post-processed through the Matlab model of the receiver

Experiences: soPHI NoC

Fault analysis and classification of the Network-on-Chip of the space instrument soPHI (Solar Orbiter’s Polarimetric and Helioseismic Imager)

Collaboration with T.U. Braunschweig

Acknowledgement to:

H. Michel, H. Michalik, A. Dörflinger

H. Michel, H. Guzmán-Miranda, A. Dörflinger, H. Michalik and M. A. Echanove, "SEU fault classification by fault injection for an FPGA in the space instrument SOPHI," *2017 NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*, Pasadena, CA, 2017, pp. 9-15.

PHI on Solar Orbiter

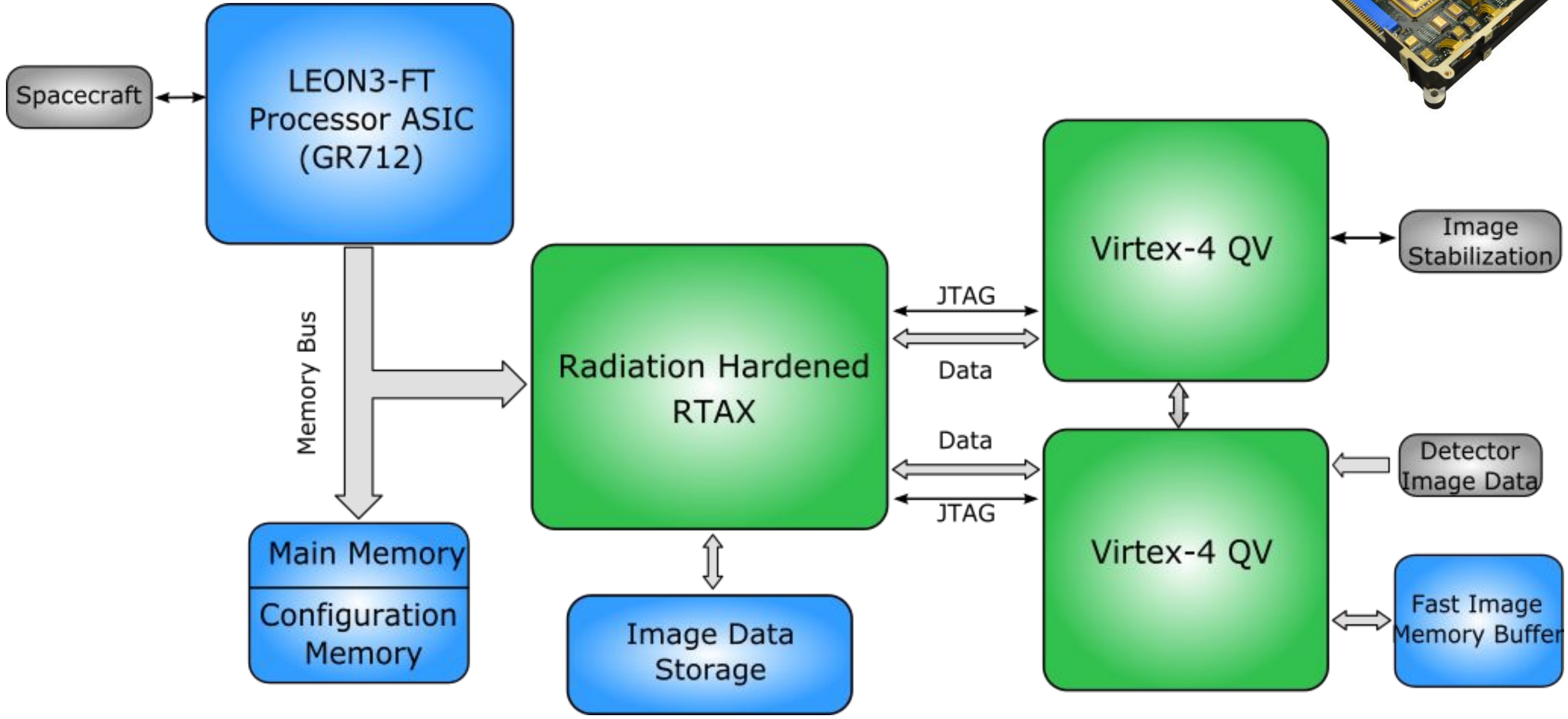
Solar orbiter
- ESA mission for sun observation
- Will orbit at 0.28 AU of the sun
- Launch scheduled for 2018

PHI instrument:
- Acquires 2k*2k images
- At different wavelengths and polarizations

Computes:
- Map of magnetic field vector
- Line-of-sight velocity in solar photosphere
- Very computation intensive tasks!

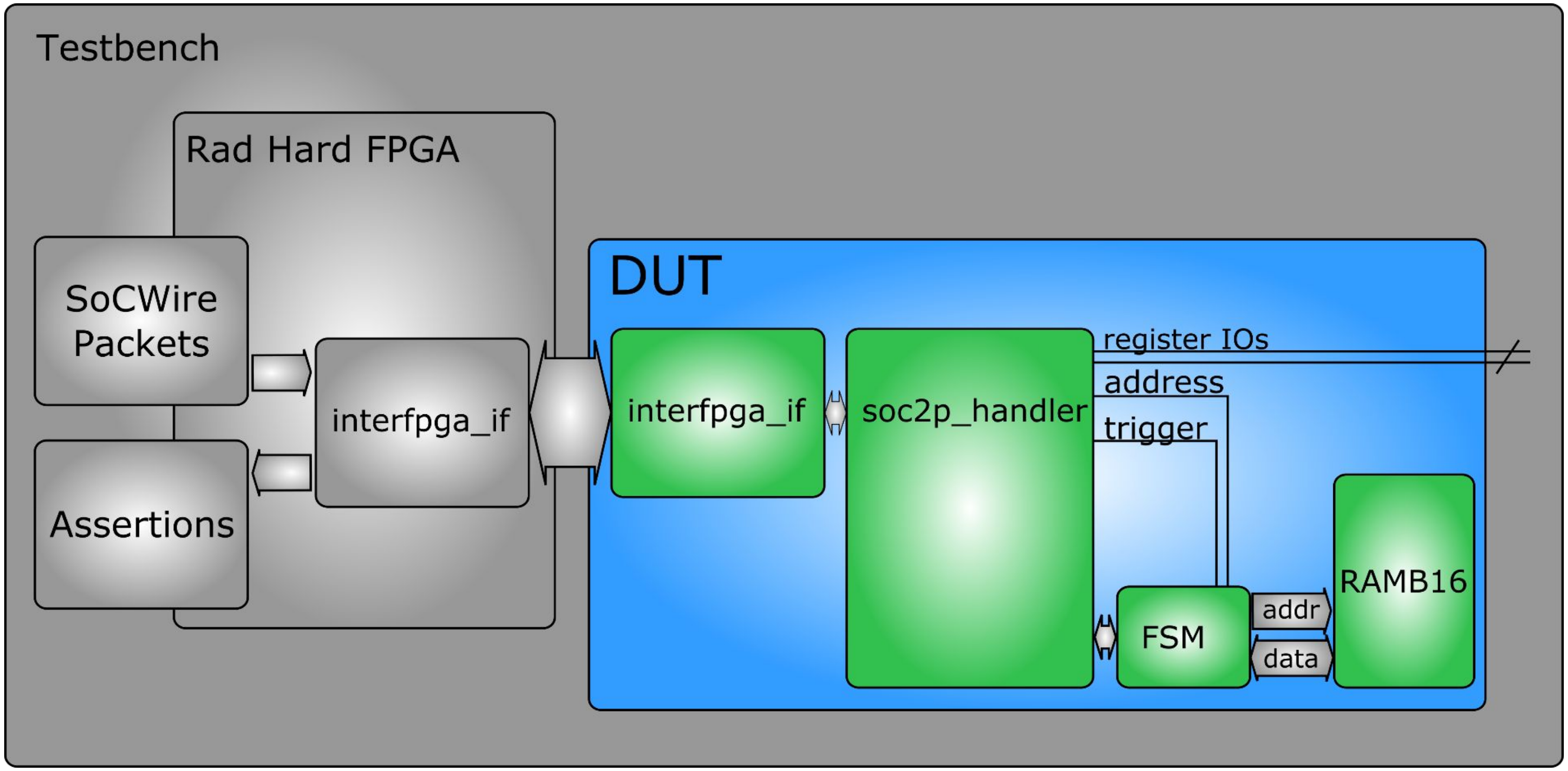
Data Processing Unit of PHI on Solar Orbiter

Image processing is done in the two Virtex-4 FPGAs
Full TMR cannot be applied because of limited capacity

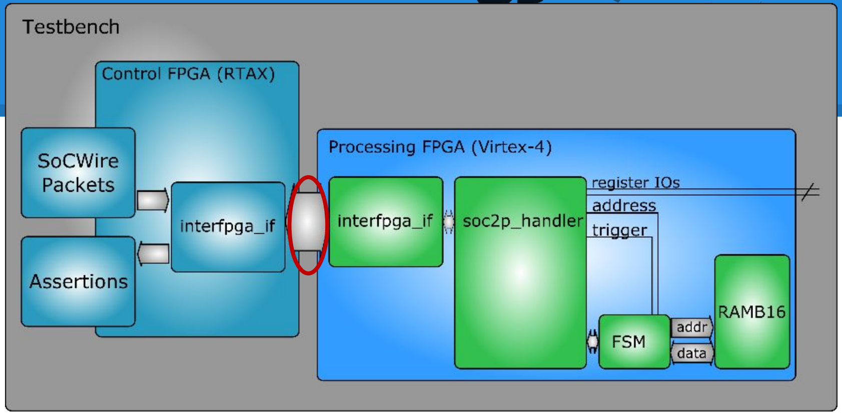


SoPHI Design in Fault Injection Tests

- We are testing the part of the NoC that goes into the Virtex-4 FPGAs

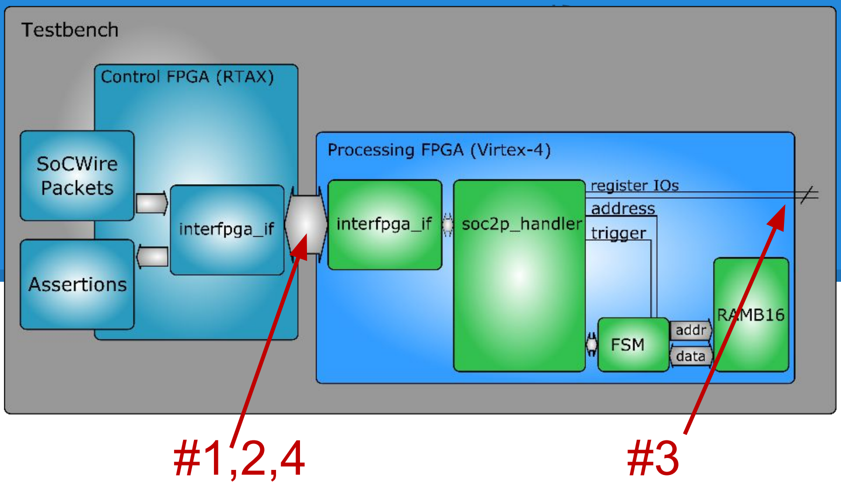


InterFPGA Interface



- READY signal indicates more data can be sent
- CTRL_CHR signal: control character
  - Null: no data transmitted in this clock cycle
  - End of Packet
  - Erroneous End of Packet
- 4 DATA signals (serialization of 16 bit network-on-chip)
  - Receiver can detect, if a packet does not add up to 16 bit words (*unmatched condition*)
- PARITY signal over all other signals
- SpaceWire like link initialization and restart
  - Restart on *unmatch*, parity error, time-out of READY signal

Fault Injection Results



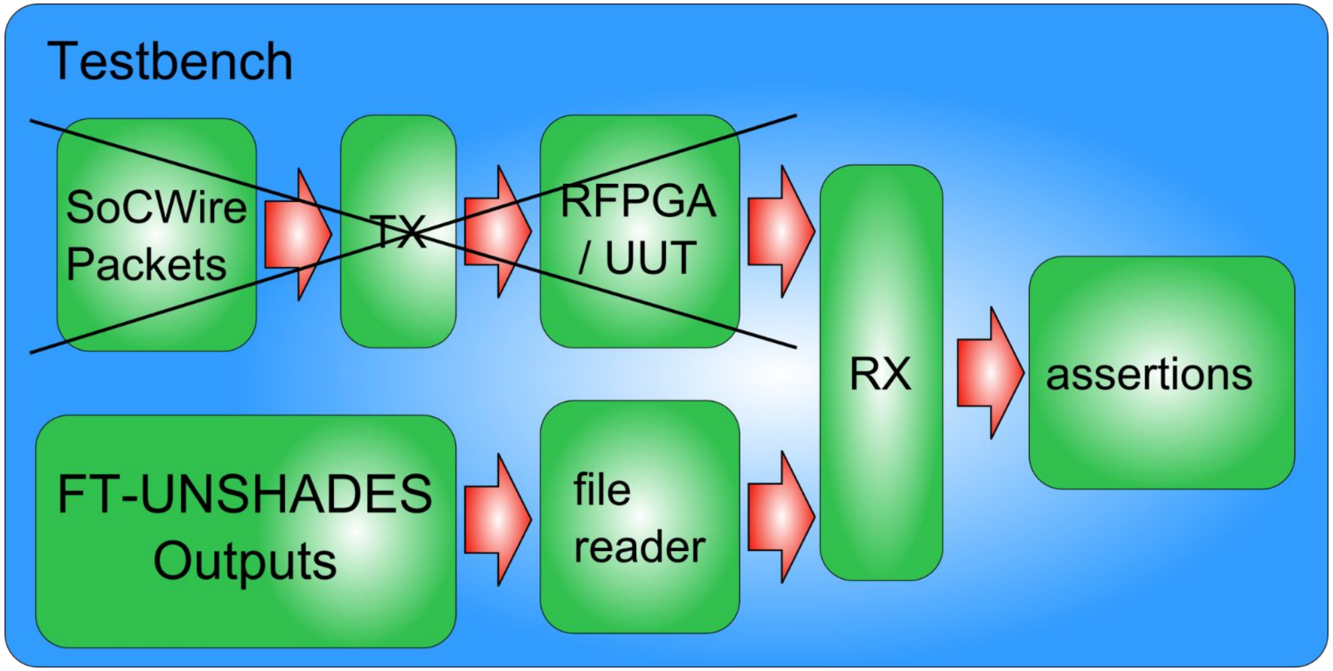
- 45% of injected faults into *essential bits* lead to no error at all
- Classification of errors according to their observed output behavior
- InterFPGA interface
  - READY signal stuck (#1)
  - Other error on InterFPGA interface (#2)
  - READY signal stuck and other error on InterFPGA interface (#4)
- Register Output (#3)

Class	Description	%
0	No error	45.8%
1	READY stuck	29.8%
2	InterFPGA interface	3.9%
3	Register outputs	0.2%
4	READY & InterFPGA	19.9%
5	Others	0.4%

Fault Classification by Fault Injection Results in Simulation

Test-setup for detailed analysis of errors in categories #2, #4, #5:

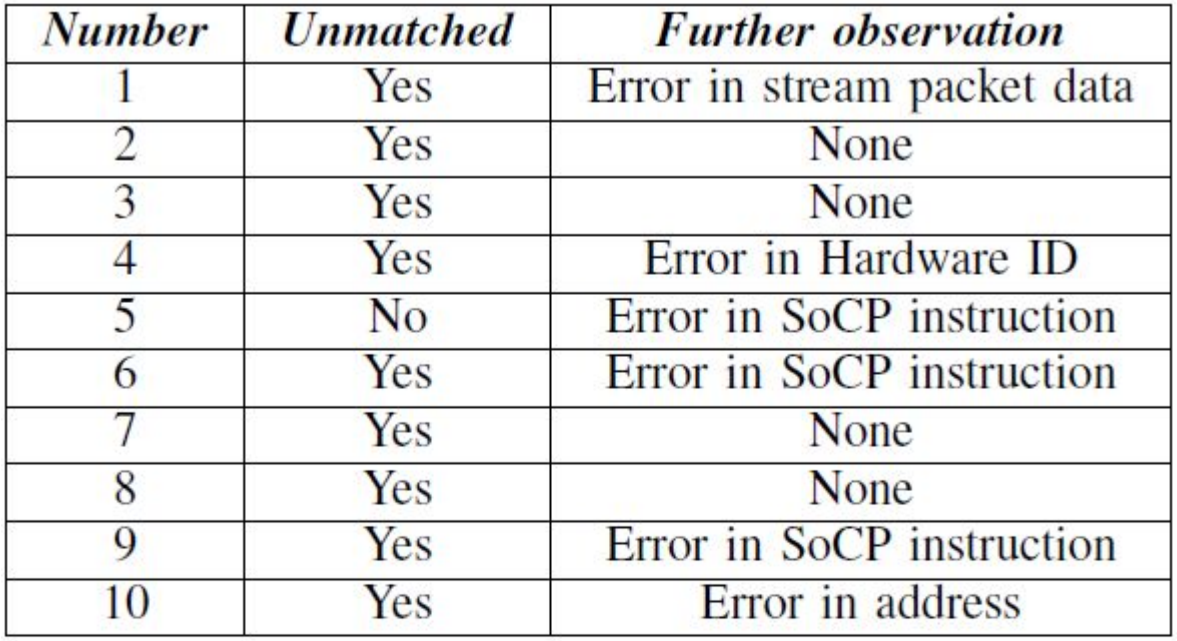
Erroneous outputs generated by FT-UNSHADES are used as stimuli for simulating RX interface behavior in Control FPGA.



Errors in category #2 (InterFPGA)

- For almost all examined errors, the receiver detected an *unmatched* condition
  → Errors are detectable and Application Software can react appropriately

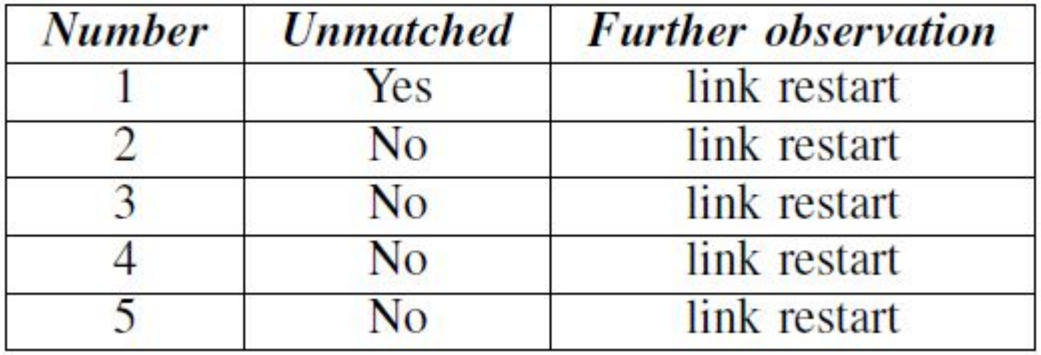
- There were also errors in the SocWire Packet including its header



Number	*Unmatched*	*Further observation*
1	Yes	Error in stream packet data
2	Yes	None
3	Yes	None
4	Yes	Error in Hardware ID
5	No	Error in SoCP instruction
6	Yes	Error in SoCP instruction
7	Yes	None
8	Yes	None
9	Yes	Error in SoCP instruction
10	Yes	Error in address

Errors in category #4 (InterFPGA and READY)

- All examined errors resulted in a link restart
  → Errors are detectable and Application Software can react appropriately

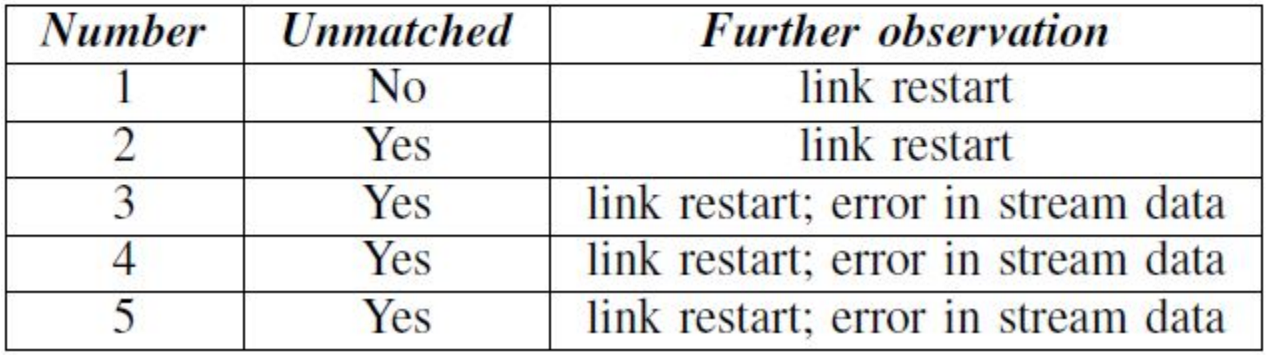


Number	*Unmatched*	*Further observation*
1	Yes	link restart
2	No	link restart
3	No	link restart
4	No	link restart
5	No	link restart

Errors in category #5 (other)

- All examined errors resulted in a link restart
  → Errors are detectable and Application Software can react appropriately

- For some cases, errors in stream data have been observed



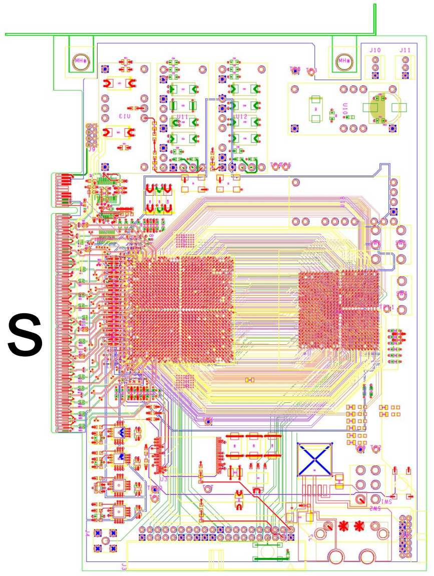
Number	*Unmatched*	*Further observation*
1	No	link restart
2	Yes	link restart
3	Yes	link restart; error in stream data
4	Yes	link restart; error in stream data
5	Yes	link restart; error in stream data

Future work: FTU-VEGAS

New architecture and daughterboard targeting NanoXplore NG-MEDIUM FPGA

- Full PCIe functionality
- Onboard SRAMs for faster emulation & microprocessor analysis

(Under development)



Part of h2020 project VEGAS

European Commission | Horizon 2020 European Union funding for Research & Innovation

Conclusions

- Non-instrumented injection increases the accuracy of the technique
  - Pending more comparisons with radiation experiments
- Web-based interface, documentation and email list for support reduces learning curve and effort
- System-level fault propagation analysis allows to extract very useful information for designers
  - -> real heat zones + how to better mitigate the faults

Conclusions

We want **YOU** to use it!

Just ask us! -> hguzman@us.es

More info on our website: ftu.us.es

Check our demo!