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)
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
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

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

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)

<table>
<thead>
<tr>
<th>Class</th>
<th>Description</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>No error</td>
<td>45.8%</td>
</tr>
<tr>
<td>1</td>
<td>READY stuck</td>
<td>29.8%</td>
</tr>
<tr>
<td>2</td>
<td>InterFPGA interface</td>
<td>3.9%</td>
</tr>
<tr>
<td>3</td>
<td>Register outputs</td>
<td>0.2%</td>
</tr>
<tr>
<td>4</td>
<td>READY &amp; InterFPGA interface</td>
<td>19.9%</td>
</tr>
<tr>
<td>5</td>
<td>Others</td>
<td>0.4%</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Number</th>
<th>Unmatched</th>
<th>Further observation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Yes</td>
<td>Error in stream packet data</td>
</tr>
<tr>
<td>2</td>
<td>Yes</td>
<td>None</td>
</tr>
<tr>
<td>3</td>
<td>Yes</td>
<td>None</td>
</tr>
<tr>
<td>4</td>
<td>Yes</td>
<td>Error in Hardware ID</td>
</tr>
<tr>
<td>5</td>
<td>No</td>
<td>Error in SoCP instruction</td>
</tr>
<tr>
<td>6</td>
<td>Yes</td>
<td>Error in SoCP instruction</td>
</tr>
<tr>
<td>7</td>
<td>Yes</td>
<td>None</td>
</tr>
<tr>
<td>8</td>
<td>Yes</td>
<td>None</td>
</tr>
<tr>
<td>9</td>
<td>Yes</td>
<td>Error in SoCP instruction</td>
</tr>
<tr>
<td>10</td>
<td>Yes</td>
<td>Error in address</td>
</tr>
</tbody>
</table>
Errors in category #4 (InterFPGA and READY)

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

<table>
<thead>
<tr>
<th>Number</th>
<th>Unmatched</th>
<th>Further observation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Yes</td>
<td>link restart</td>
</tr>
<tr>
<td>2</td>
<td>No</td>
<td>link restart</td>
</tr>
<tr>
<td>3</td>
<td>No</td>
<td>link restart</td>
</tr>
<tr>
<td>4</td>
<td>No</td>
<td>link restart</td>
</tr>
<tr>
<td>5</td>
<td>No</td>
<td>link restart</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Number</th>
<th>Unmatched</th>
<th>Further observation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>No</td>
<td>link restart</td>
</tr>
<tr>
<td>2</td>
<td>Yes</td>
<td>link restart</td>
</tr>
<tr>
<td>3</td>
<td>Yes</td>
<td>link restart; error in stream data</td>
</tr>
<tr>
<td>4</td>
<td>Yes</td>
<td>link restart; error in stream data</td>
</tr>
<tr>
<td>5</td>
<td>Yes</td>
<td>link restart; error in stream data</td>
</tr>
</tbody>
</table>
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
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](mailto:hguzman@us.es)

More info on our website: [ftu.us.es](http://ftu.us.es)

Check our demo!