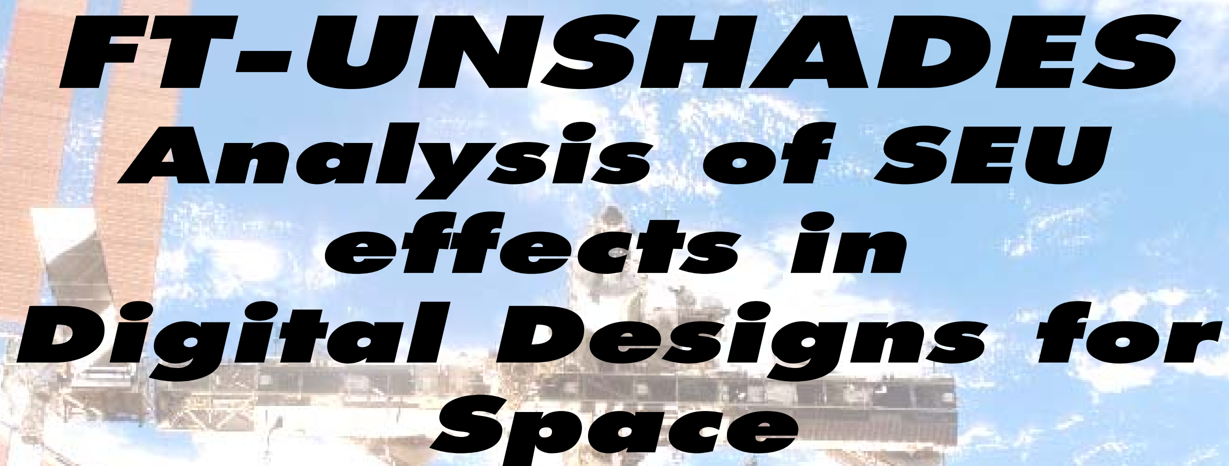
esa



# FT-UNSHADES Analysis of SEU effects in Digital Designs for Space

Gioacchino Giovanni Lucia
TEC-EDM,
MPD - 8th March
Gioacchino.lucia@esa.int
Phone: +31 71 5658482

# AGENDA

- FAULT INJECTION SYSTEM
  - Requirements
  - Classification
- FT-UNSHADES Tool
  - Features
  - Architecture
  - Software toolbox
  - Test Flow
    - FPGA TARGET
    - ASIC TARGET
- Case Study
  - LEON Processor
- Conclusions

# REQUIREMENTS

## FAULT INJECTION SYSTEM

- provide information about the behaviour of the circuit when a fault is injected while a given set of stimuli is applied to the circuit
- determine the coverage of error detection and recovery mechanisms
- evaluate the effectiveness of fault tolerance mechanism
- evaluate performance loss
- The fault model we refer to is the **single transient bit-flip**
- The bit-flip can be random or deterministic in **time** and in **space**, depending on the test objectives
- The input stimuli to be used during the analysis process are already available, and we do not deal with their generation or evaluation

# CLASSIFICATION

- **Type of SEU emulated**
  - SEUs may alter the memory elements the design embeds
  - SEUs may alter the content of the memory storing the device configuration
- **Type of fault detected**
  - Latent
  - Damage
- **Test level**
  - Software
  - Hardware
- **Cost**
  - Effort needed to prepare the design for testing
  - Rent facilities
- **Level of intrusiveness**
- **Level of information extracted**
- **Speed**

# AGENDA

- FAULT INJECTION SYSTEMS
  - Requirements
  - Classification
- FT-UNSHADES Tool
  - Features
  - Architecture
  - Software toolbox
  - Test Flow
    - FPGA TARGET
    - ASIC TARGET
- Case Study
  - LEON Processor
- Conclusions

esa

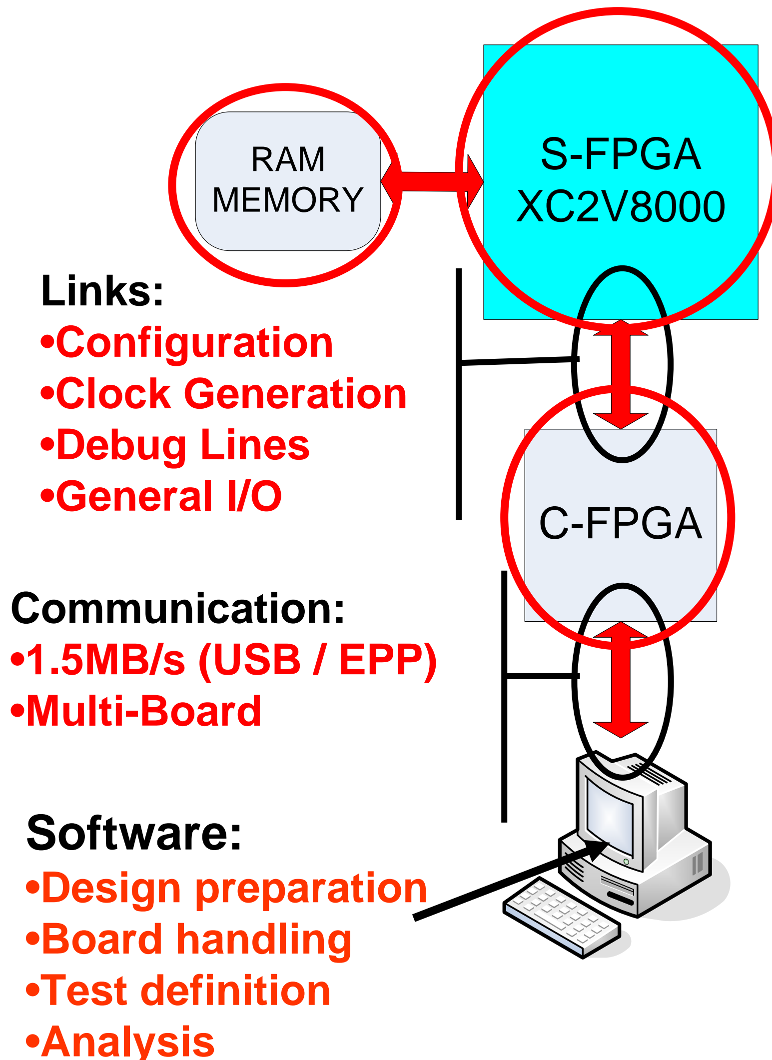
# FT-UNSHADES

- Developed by University of Seville
- Emulates **SEU** in user FFs.
- **Hardware** platform
  - Xilinx Virtex-II FPGA
    - The **capture and readback mechanism**
    - The FPGA configuration memory can be **partially read and written.**
- **Not Intrusive** technique
- **Easy** to use flow
- **Deep** and **Fast** analysis

What FT-UNSHADES is not:

- **It's not** a platform for radiation testing of FPGAs
- **It's not** designed for being inserted into a radiation environment

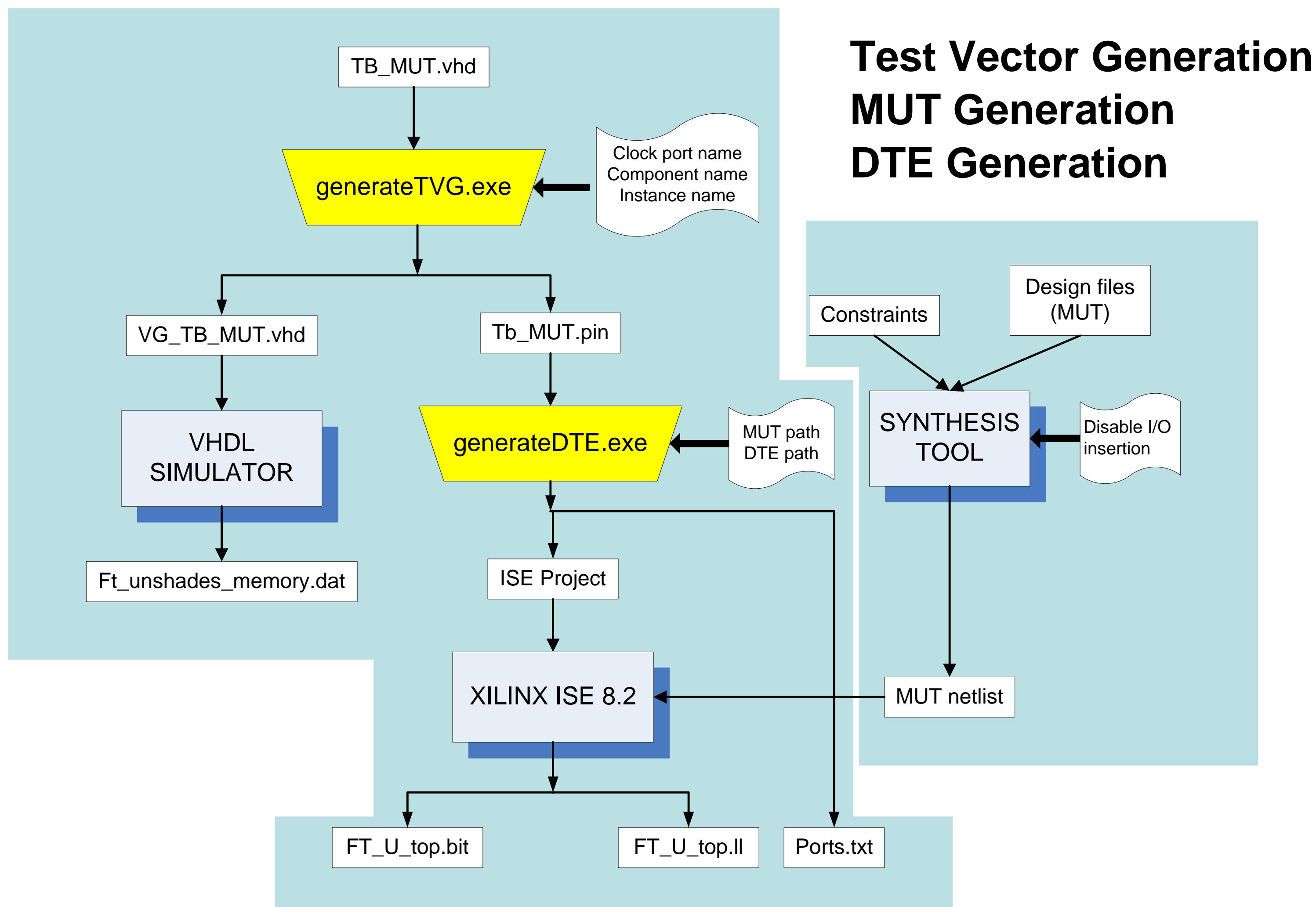
# ARCHITECTURE




- A Xilinx Virtex II called the System FPGA (S-FPGA) is used for the core emulation.
- A second FPGA (C-FPGA) is used as an extremely fast bridge between the SW and the S-FPGA.
- Large static memories are used to hold up to 2 million input test vectors, each 102 bits wide which are used to stimulate the design MUT.

# DESIGN PREPARATION TARGET: FPGA

**Test Vector Generation**
**MUT Generation**
**DTE Generation**



# DESIGN PREPARATION TARGET: ASIC (1/2)

- **TYPICAL SCENARIO:**
  - SEU radiation test showed functional anomalies of a submodule in a large ASIC netlist.
  - The design size doesn’t allow to test the whole design with FT-U
- **GOAL:**
  - find the causes of that functional anomalies
- **What do we need to do?**
  - Extract the module from the netlist so that the size of the design fits in FT-U
  - FT-U flow requires a VHDL testbech for the test vectors generation so we need to extract the stimuli usually from the system testbench
    - USE VCD format to dump the input of the DUT.
    - Write a VHDL testbench that parses the VCD file and feeds the DUT

# DESIGN PREPARATION TARGET: ASIC (2/2)

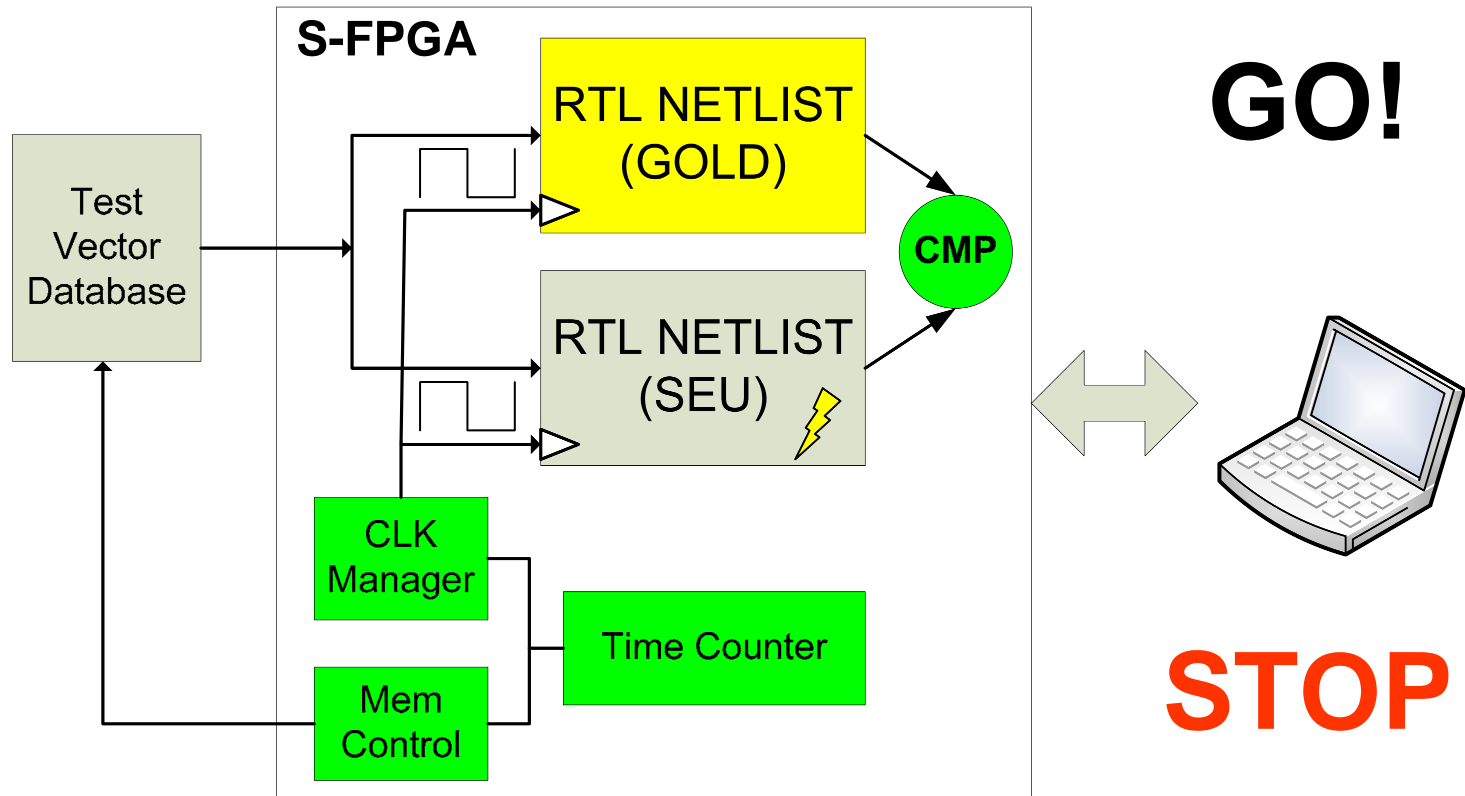
- **What do we need to do?**
  - Reliable technology remapping of the ASIC netlist to the XILINX library → **CRITICAL POINT!!!**
    - it must be verified that the translated netlist maintains **the exact same sequential logic** (FF replication or elimination must be avoided) and **100% functional equivalency** → USE A FORMAL VERIFICATION TOOL (e.g. Formality)
    - Build a simulation with the “old” and “new” netlist and compare the outputs to verify the behaviour is the same
  - FT-U can handle up to 102 input pins → Wrapper needed

# TEST CAMPAIGN DEFINITION and ...

- Configure the System FPGA and download the test vector database, define some environment variables.
- Specify the analysis type:
  - DAMAGE
  - LATENT
- Specify SEU time constraint (optional)
- Specify SEU location constraint (optional)
- Specify number of **RUNs** to perform
  - **RUN**: Complete simulation from the first to the last test vector
- Specify number of SEU per RUN to inject

# … EXECUTION

**CLOCK: 133937 REGISTER:leon0_mcore0_proc0_cx**

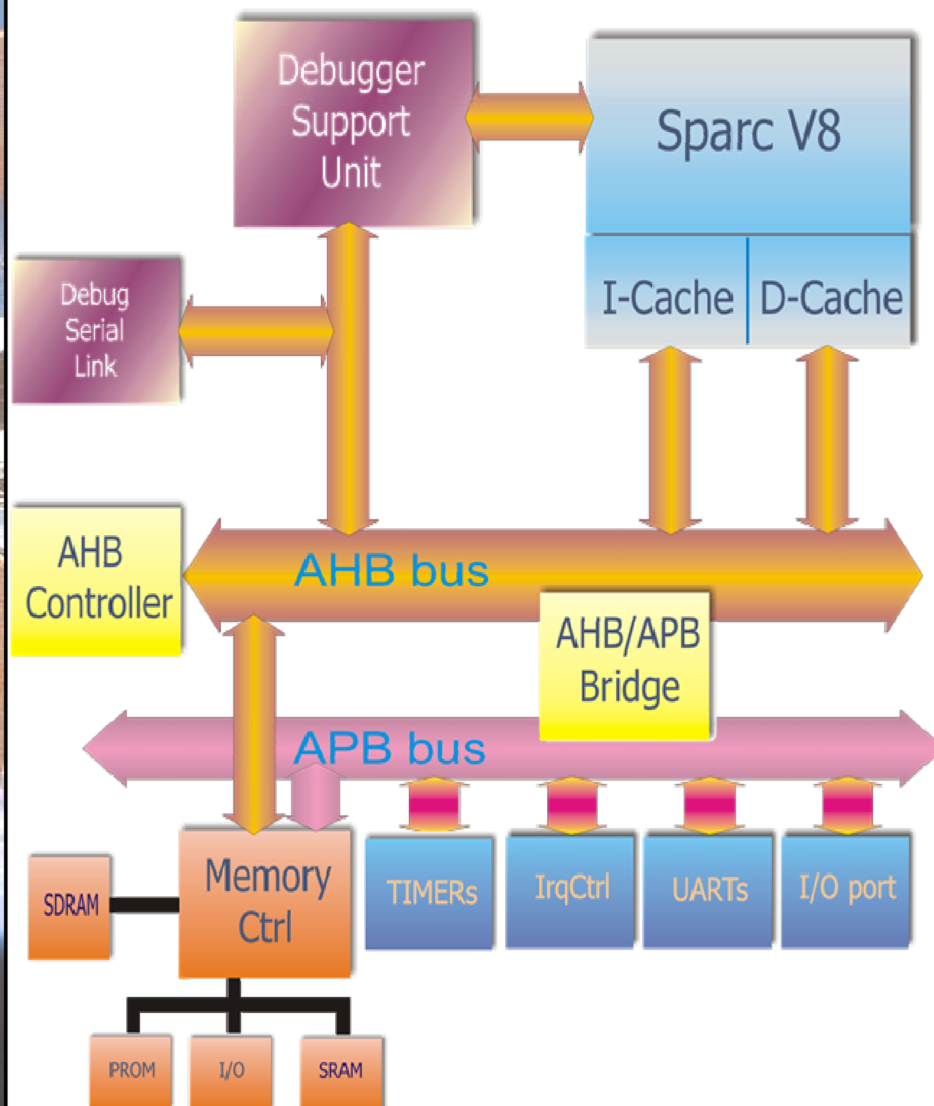



**[illegible]**
**classify fault as damage;**

# AGENDA

- FAULT INJECTION SYSTEMS
  - Requirements
  - Classification
- FT-UNSHADES Tool
  - Features
  - Architecture
  - Software toolbox
  - Test Flow
    - FPGA TARGET
    - ASIC TARGET
- Case Study
  - LEON Processor
- Conclusions

# CASE OF STUDY: LEON




Processor:

- 32 Register windows
- 2K I-Cache and D-Cache
- No FPU neither coprocessor

Peripherals:

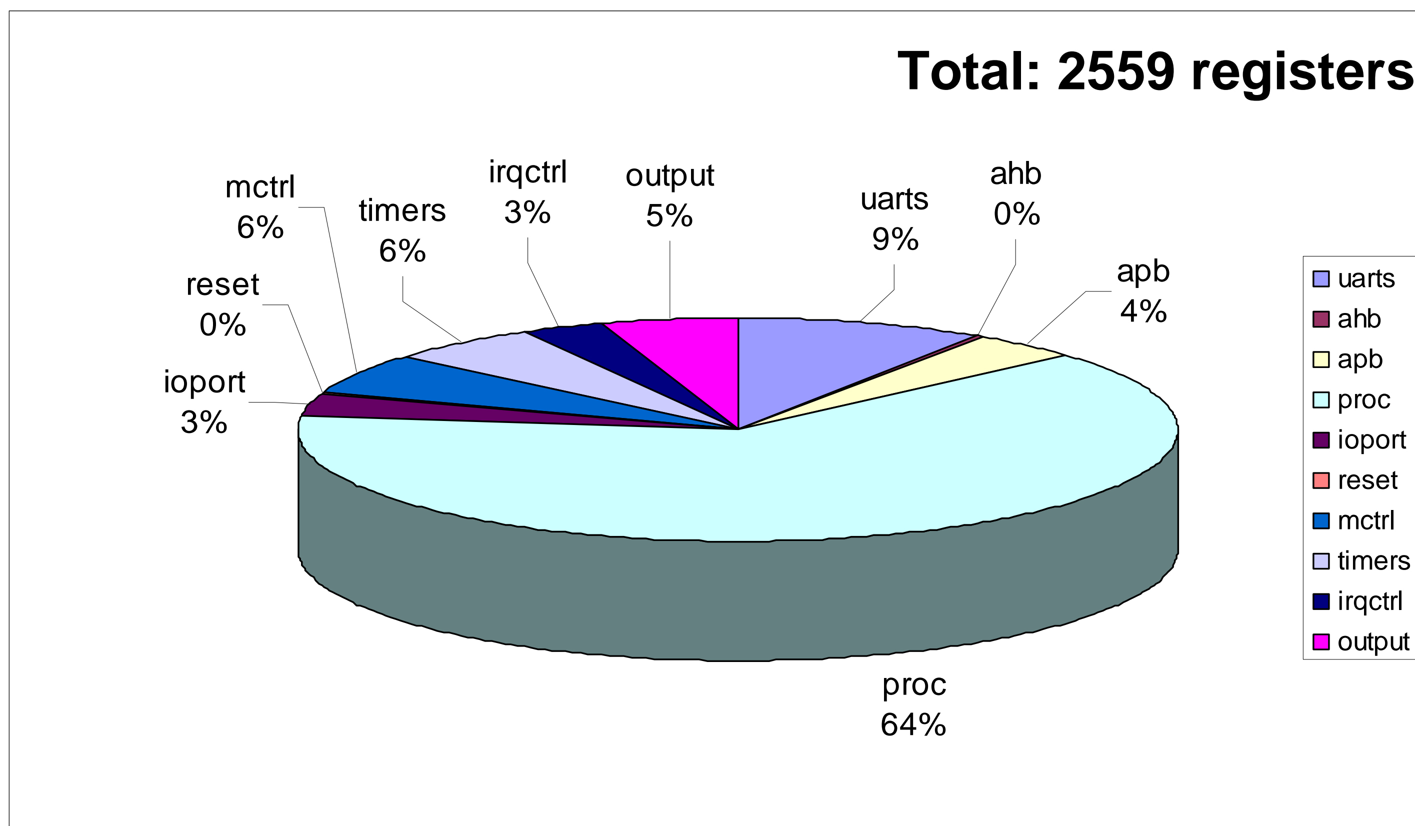
- Debug Support Unit
- 32KB AHB RAM
- 2 UARTs + Parallel I/O port
- 1 Interrupt Controller
- 2 Timers + Watchdog

# LEON: TEST CAMPAIGN DEFINITION and EXECUTION

- **How many runs ? 1, 10, 100, 100K ?**
  - Depends on the number of registers
  - Depends on the test bench used
- **What kind of Injection? Random, single register, at a given time…?**
  - It depends on what you are testing
- **When to stop the test campaign?**
  - When all the information needed are available

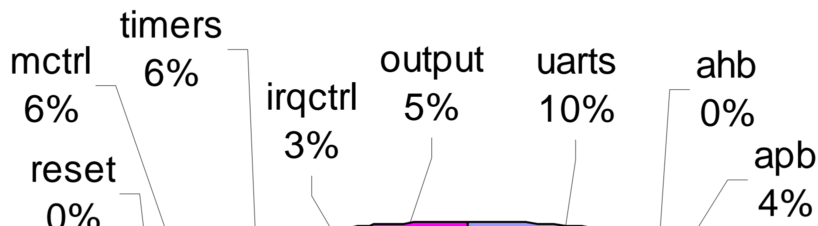
# LEON: REGISTERS DISTRIBUTION

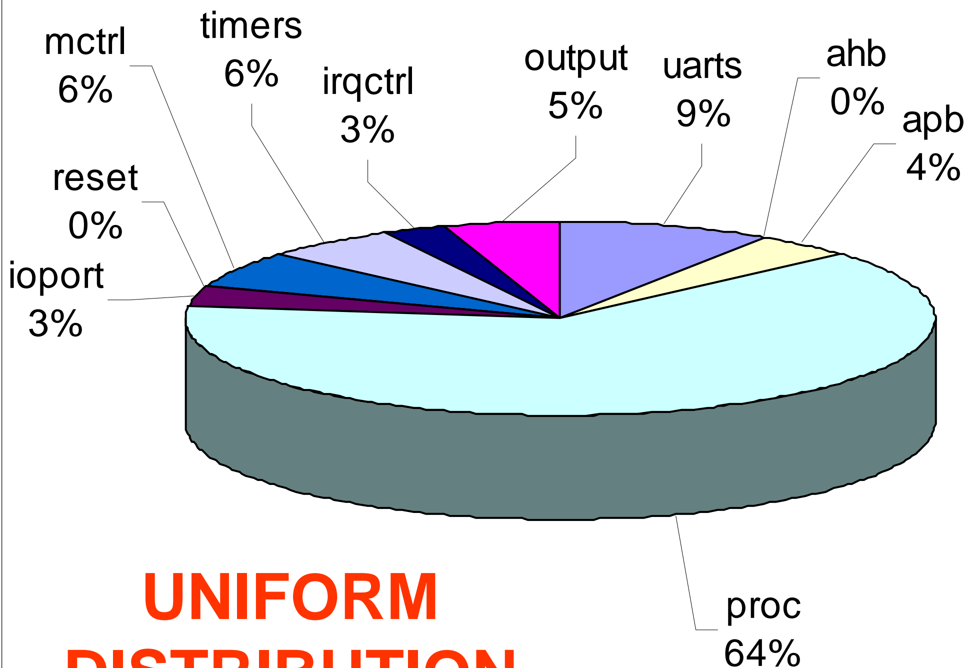
| ahb | apb | proc | ioport | reset | mctrl | timers | irqctrl | output |
|---|---|---|---|---|---|---|---|---|
| 5 | 86 | 1440 | 74 | 8 | 131 | 127 | 64 | 115 |



# LEON: INJECTIONS DISTRIBUTION

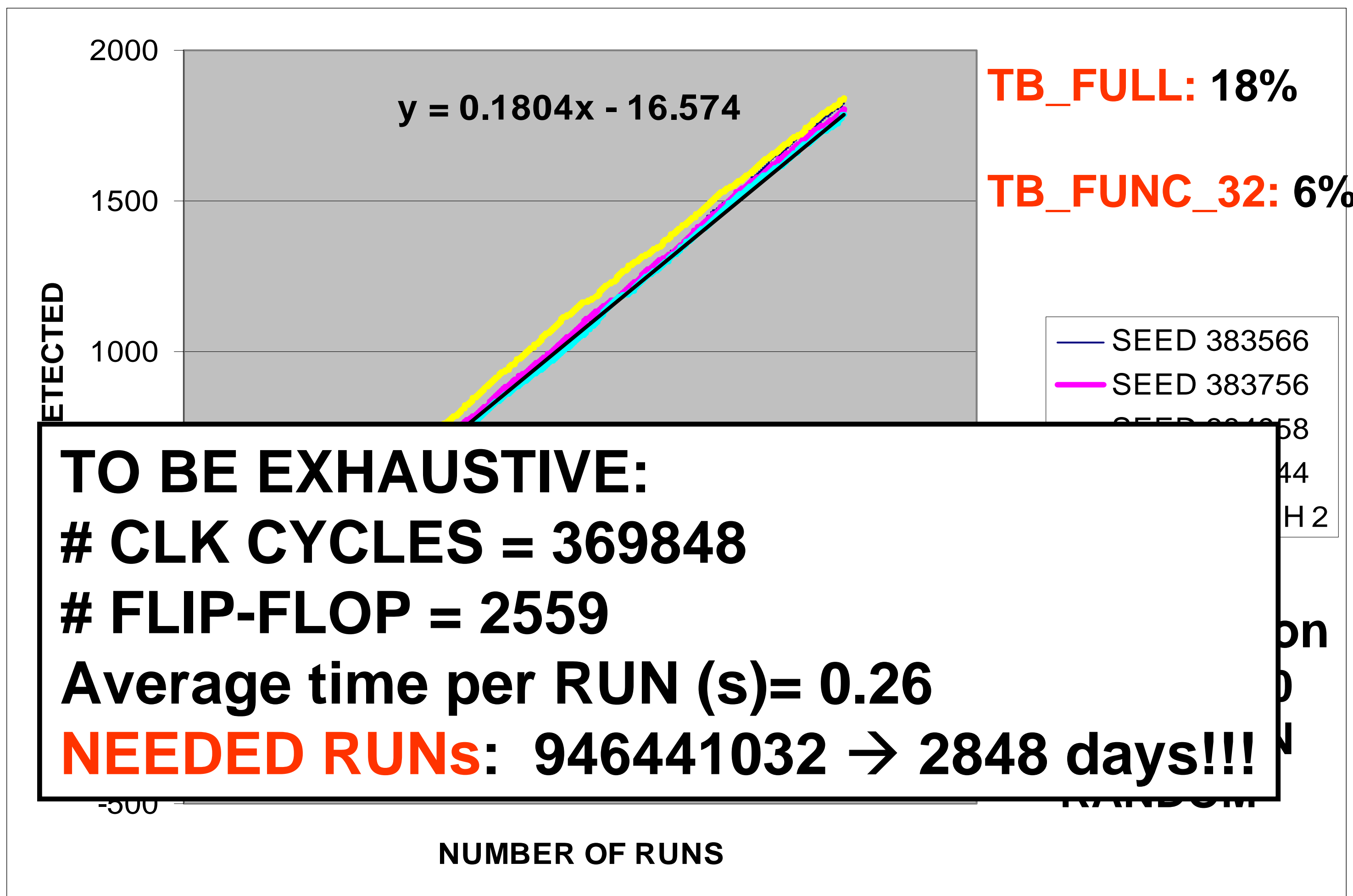
**Test Conditions**
**RUNs: 10000**
**1 SEU x RUN**
**RANDOM**



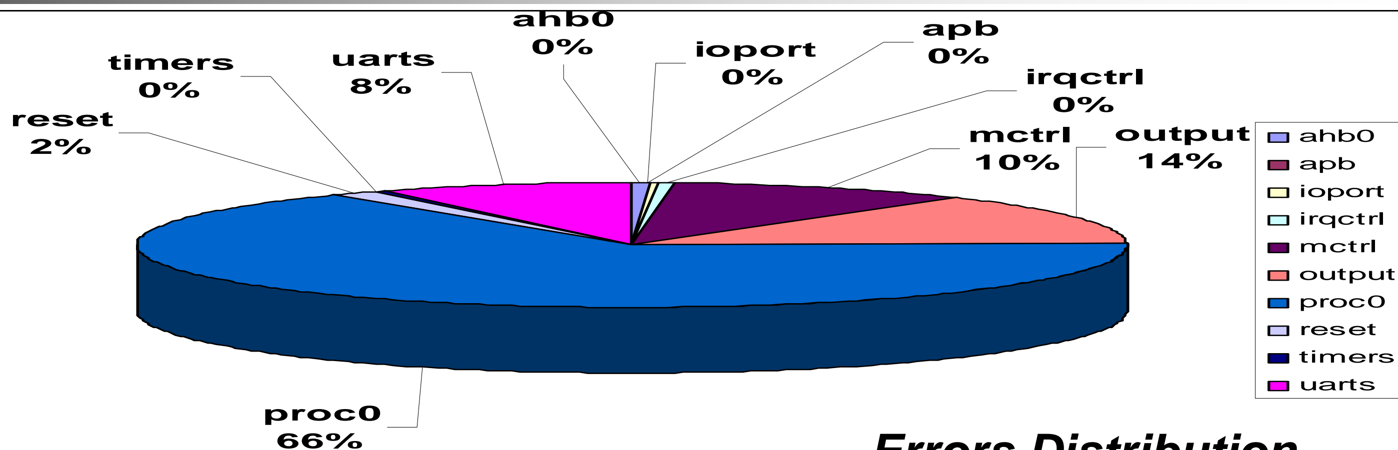




**UNIFORM DISTRIBUTION**

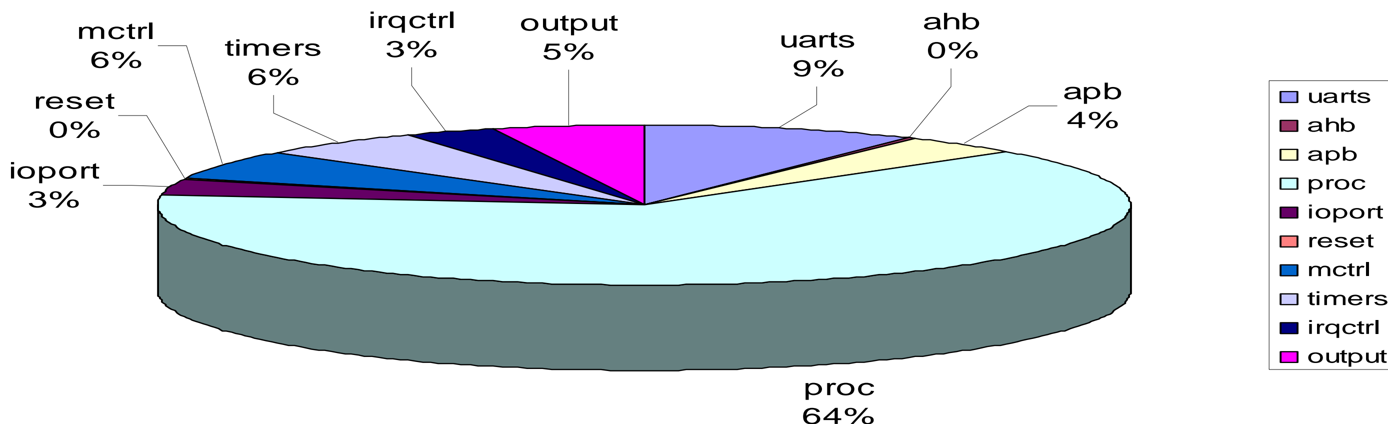
# TESTBENCH SEU COVERAGE



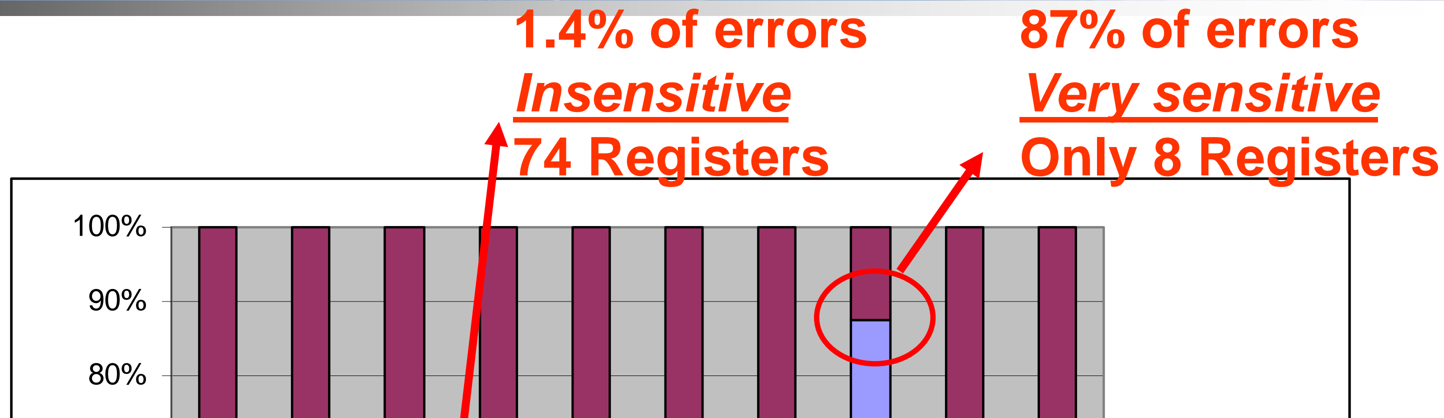
# Errors Distribution vs Injections Distribution



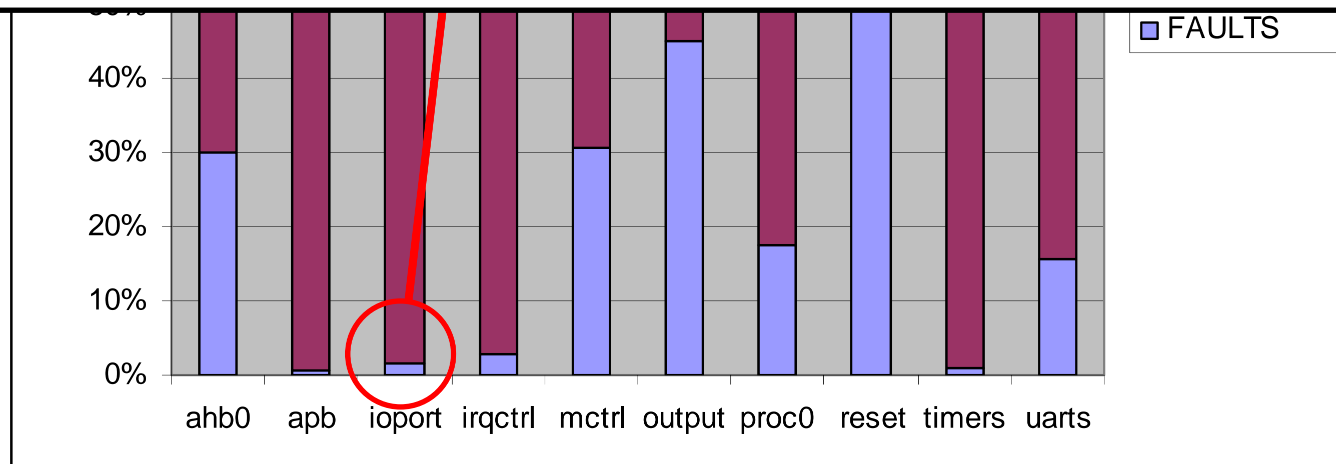

**TIMERS:** 6% of the Design FFs (127)
565 injections
Only 3 ERRORS detected



# NORMALIZED SENSITIVITY




**IT IS POSSIBLE TO IDENTIFY WHICH PARTS OF THE DESIGN ARE "MORE SENSITIVE TO SEUs"**




**Average: 20 injections per register**

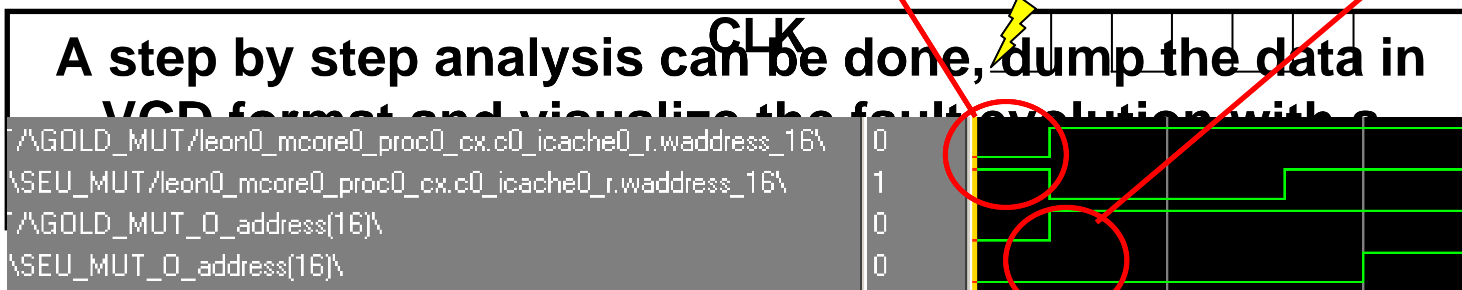
# STEP BY STEP ANALYSIS

```
#RUN 1
Selected clk cycle for SEU insertion: 133937
Selected reg for SEU insertion: SEU_MUT/leon0_mcore0_proc0_cx.c0_icache0_
ess_16
OK
Output error detected in port: address
Damage detected 1 clk cycle after SEU insertion
Total elapsed time: 0.062501
Target size: 1, registers
Total FPGA Cycles: 0x0,00020B32 (133938)
```

**CLOCK:** 133937
**REGISTER:** leon0_mcore0_proc0_cx.c0_icache0_r.waddress_16
**DAMAGE DETECTED:** YES
**LATENCY:** 1 CLK
**PORT:** address

Error Detected
After one clock cycle

Fault Inj.



# NEXT STEPS

- **NEW DESIGNS TO TEST**
  - CTM (CCSDS Time Manager)
    - Unprotected
    - XTMR_V1
    - XTMR_V2 (to be produced)
  - PCI interface of the AT697E
  - SpW CODEC/Router
  - LEON with specific TB aiming to reach a higher coverage
- **FT-UNSHADES →The future**
  - Faster processing
    - Insert a frame processor in the C-FPGA to reduce the USB traffic
  - Injection in memory blocks
  - Better handling of bidirectional pins
  - Larger design capacity
  - On-line access to the test board

# Conclusions (1/2)

- A tool for verification of the design protections
- Automatic search of weak/unprotected points in your design BEFORE place and route and fabrication
- Identify which areas are more sensitive in the design (selective protection)
- Understand/reproduce rad test results
- Verify the correctness of a new implementation of a fault tolerance mechanism

esa

# Conclusions (2/2)

**Factors affecting the probability of detecting “SEU sensitivity” or “SEU protection errors”:**

- “quality” of the test bench to expose SEU effects. This can be quantified by FT-U
- How exhaustive the test campaign is (how many possible SEU cases are covered)
- “Faulty SEU protections” can have different probabilities of being detected by an FT-U test

# THANK YOU