Fault Injection Methodologies: Implementation Aspects

Miguel A. Aguirre,

Departamento de Ingeniería Electrónica

Universidad de Sevilla (SPAIN)

Grupo de Ingeniería Electrónica

Hipólito Guzmán, Javier Barrientos, Luis Sanz, Rogelio Palomo, Fernando Muñoz, Fernando Márquez

Motivation

- Fault injection is a good method for predicting the behavior of a digital circuit under radiation environment.
- It is easy, cheap and effective, and a good complement the beam testing.
- It is a design tool. The designer can assess many time and make the necessary arrangements to improve the robustness of the design.
- There is a huge amount of additional information that could be extracted to protect your design.
- SRAM FPGAs are an interesting concern in the space community. Also fault injection is a good method to improve your design.

This lesson is a compendium of procedures and methods around any fault injection platform.
We present a catalogue of all the possible analysis reports that can be extracted from a design and applications for netlist and for SRAM FPGAs

We will make the emphasis on the tool FT-UNSHADES2

Who can take benefit from FI?

- The design engineer with a predicting tool

- The test engineer with a diagnostic tool

Summary

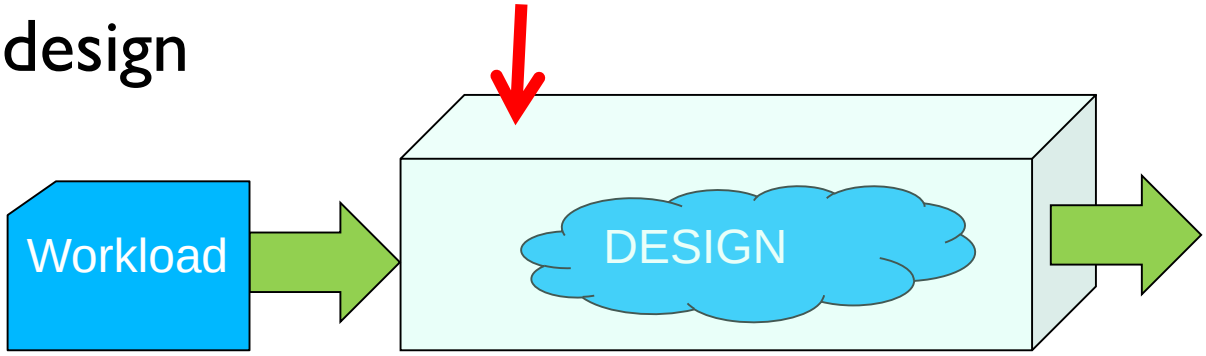
- Motivation
- Basic Concepts
- Emulation Technologies
- Advanced Fault Injection
- CASE STUDYs
- FPGA mode

1. What is expected from FI to Netlists?

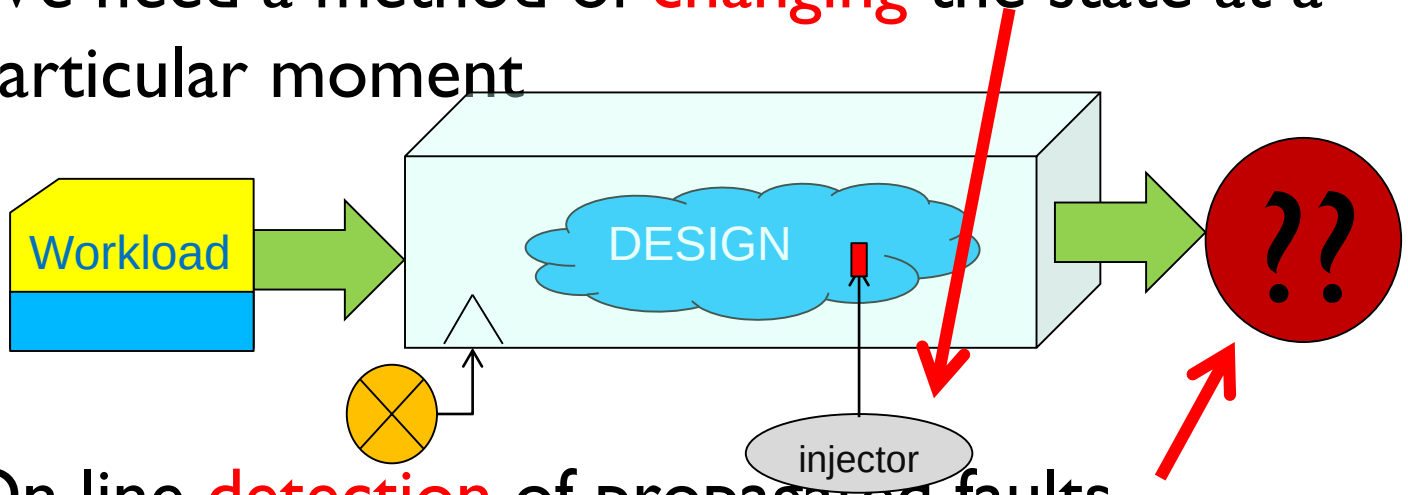
- FIN is a method of simulating of THE CONSEQUENCES of SEE in your netlist
- Characterize the vulnerability of a design, at NETLIST level, to radiation
- Check and verify the protections inserted.
- Model the spontaneous changes of the content of registers while it is being executed.
- Platform is THE INSTRUMENT for **assessing** the **robustness** and the effectiveness of your **mitigation logic**.

2. Definitions and fundamentals terms

1. We need a platform to host and execute a design



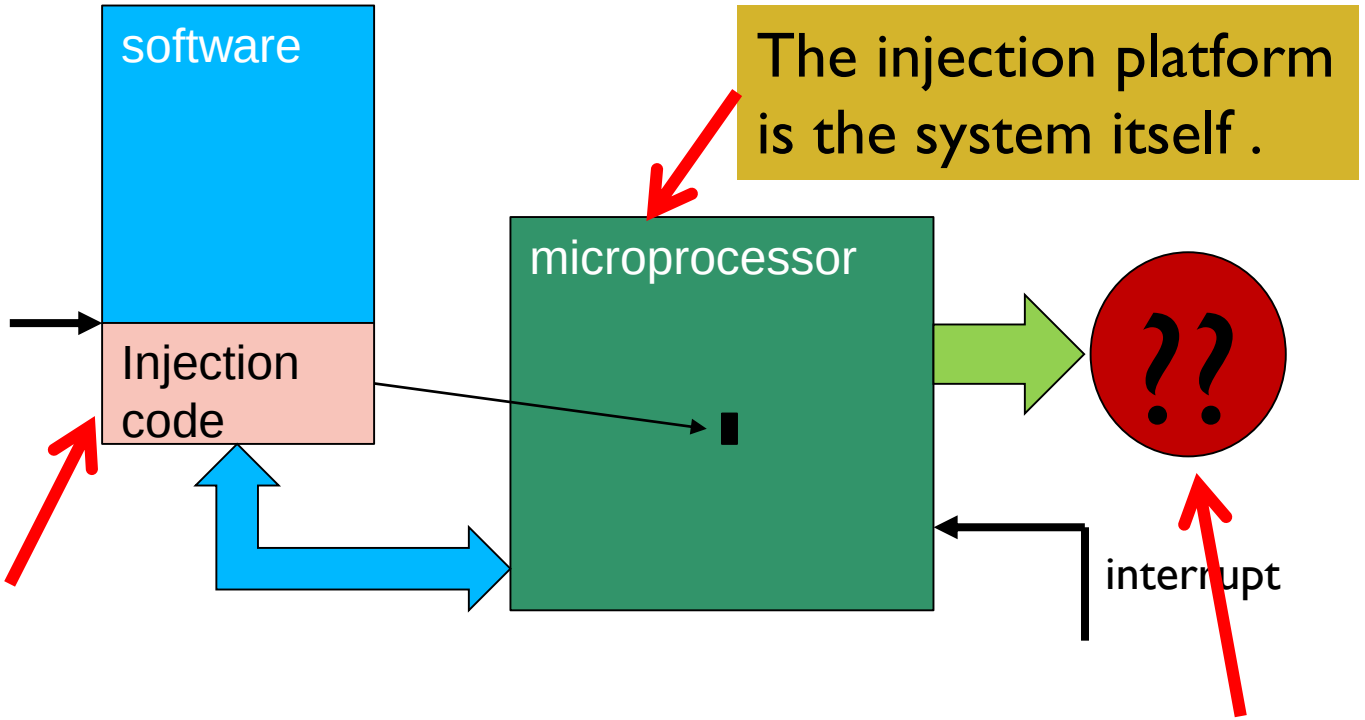
2. We need a method of changing the state at a particular moment



3. On line detection of propagated faults

Definitions and fundamentals terms

- Fault injection in microprocessor systems



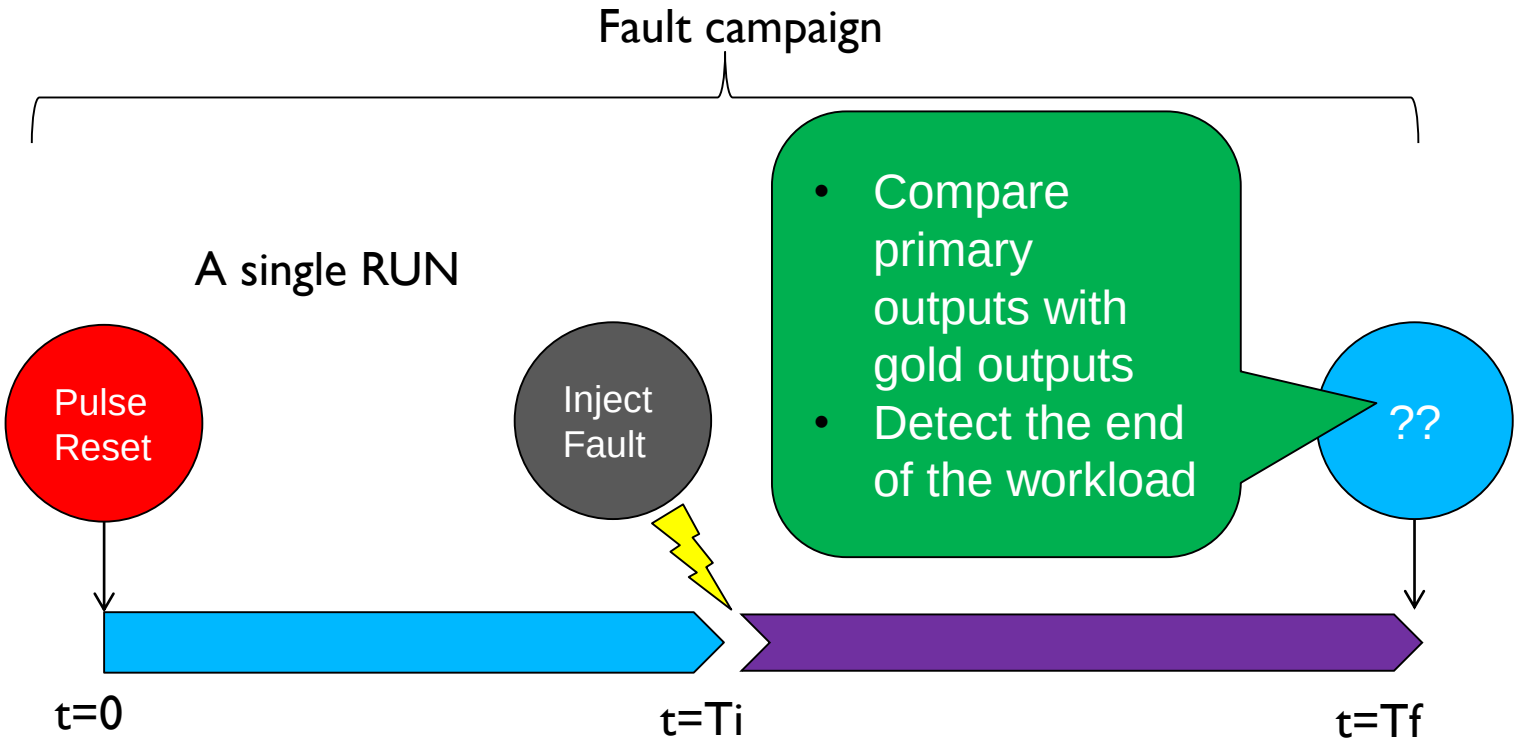
On line detection of propagated faults

Definitions and Fundamental Terms

- A design to be characterized (DUT)
- A set of stimuli (workload)
- A method of modification of the state (injector)
- Control of the injection
- A method to evaluate the effects of the fault injection. (Platform)
- *Dictionary* generation

Fault injection platform is a system that produces perturbations on the normal execution of a design, simulated or emulated, knowing a priori ***where***, ***when*** and ***how*** the injection is, and assessing the results.

- The injected fault is a pessimistic model of the physical effect under study.
- After an injection the result is recorded into a Fault Dictionary.
- The procedure is repeated a significant number of RUNS. This is called CAMPAIGN.



The environment

- Simulation

The Questa® Verification Solution

- ☺ Simple, easy, flexible
- ☺ Unlimited size
- ☺ Deep analysis
- ☺ Hardware independent (One PC is enough)
- ☹ Slow
- ☹ No FPGA mode

- Emulation

SYNOPSYS Synplify Premier

XILINX ALL PROGRAMMABLE™

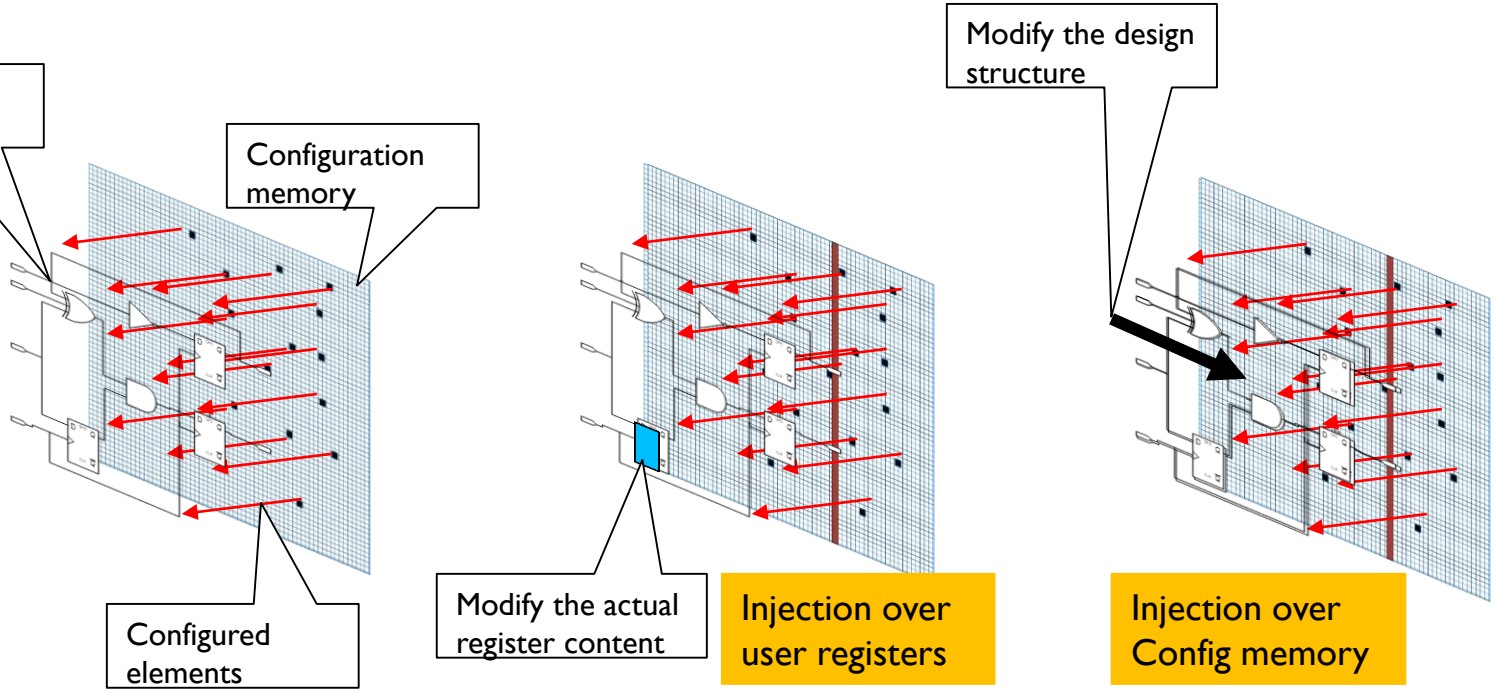
- ☺ Much faster and repetitive
- ☺ Post-synthesis
- ☺ FPGA mode
- 😐 Size independent (if it fits)
- ☹ FPGA architecture dependent

NANOXPLORE

This presentation

Fault Injection to Netlists vs. Fault Injection for FPGAs

- FIN is constrained to USER REGISTERS. Pure functional and
- FIF injects over CONFIGURATION BITS
- FIN and FIF are detected in primary outputs
- FIN and FIF depend on the stimuli set



Fault injection and radiation testing

- If we test an Integrated circuit we obtain a cross section which is the result between:

$$\#of\ errors\ detected/fluence <> Energy\ of\ particles\ (LET)$$

An error is detected when it is propagated to the primary outputs of the device: then depends on the functional architecture of the circuit.

This term is measured with fault injection

- Laser based fault injection
- Accelerators and Microprobes

Once we know about the basics of the game… let's start to play

3. Emulated Fault injection.

3.1 Gross numbers

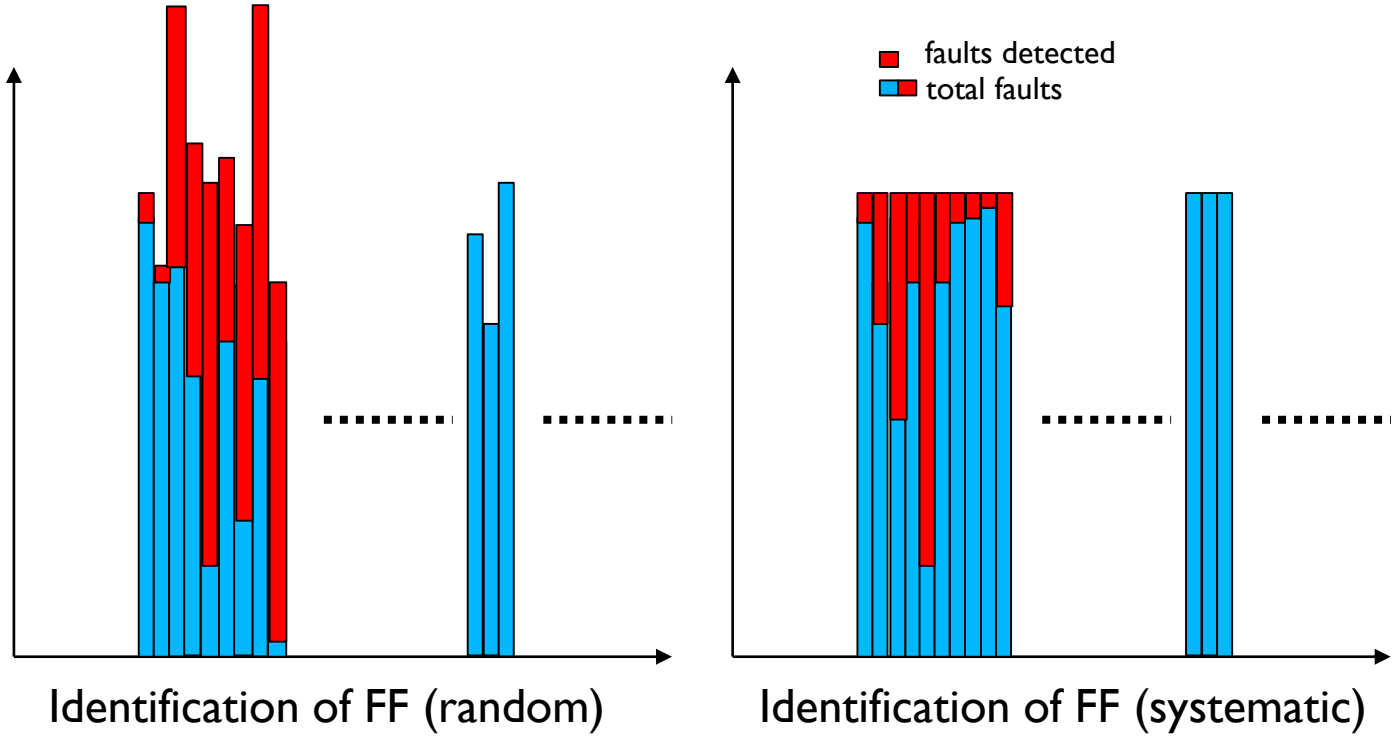
3.2 Fault models

3.3 Campaign

3.4 An example

3.1 Campaign concepts

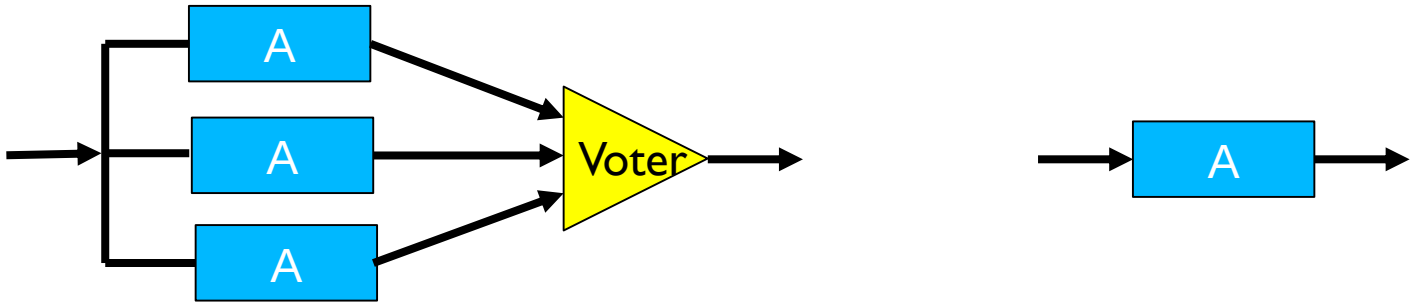
- A campaign is a huge number of FI, at different locations and different cycles
- For each FF, record the number of faults injected, and record the faults detected



Architectural defects detected

1. Collapsed TMRs

1. Redundancy is the general method to mitigate single faults
2. Redundancy may be collapsed in synthesis process

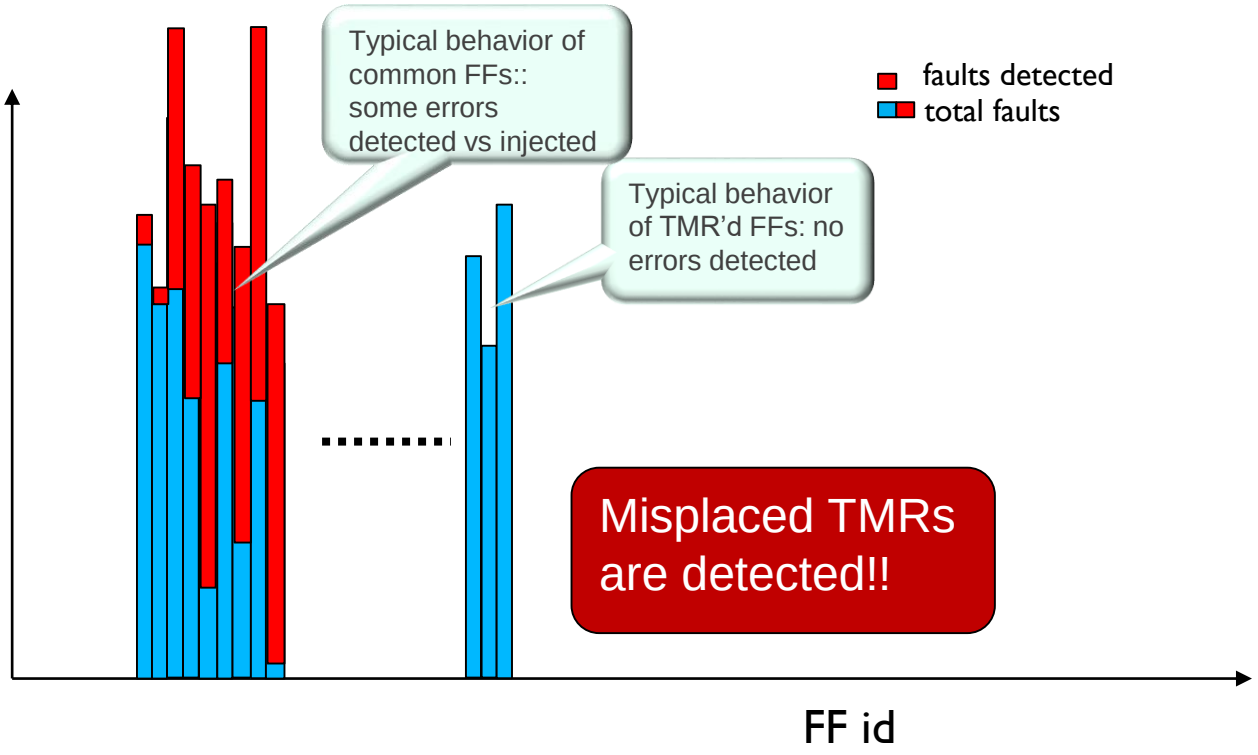


- It is convenient to check redundancy persistence in netlist at post-synthesis level. Redundant elements could be removed due to synthesis.

Architectural defects detected

- Faults are injected to well determined FFs

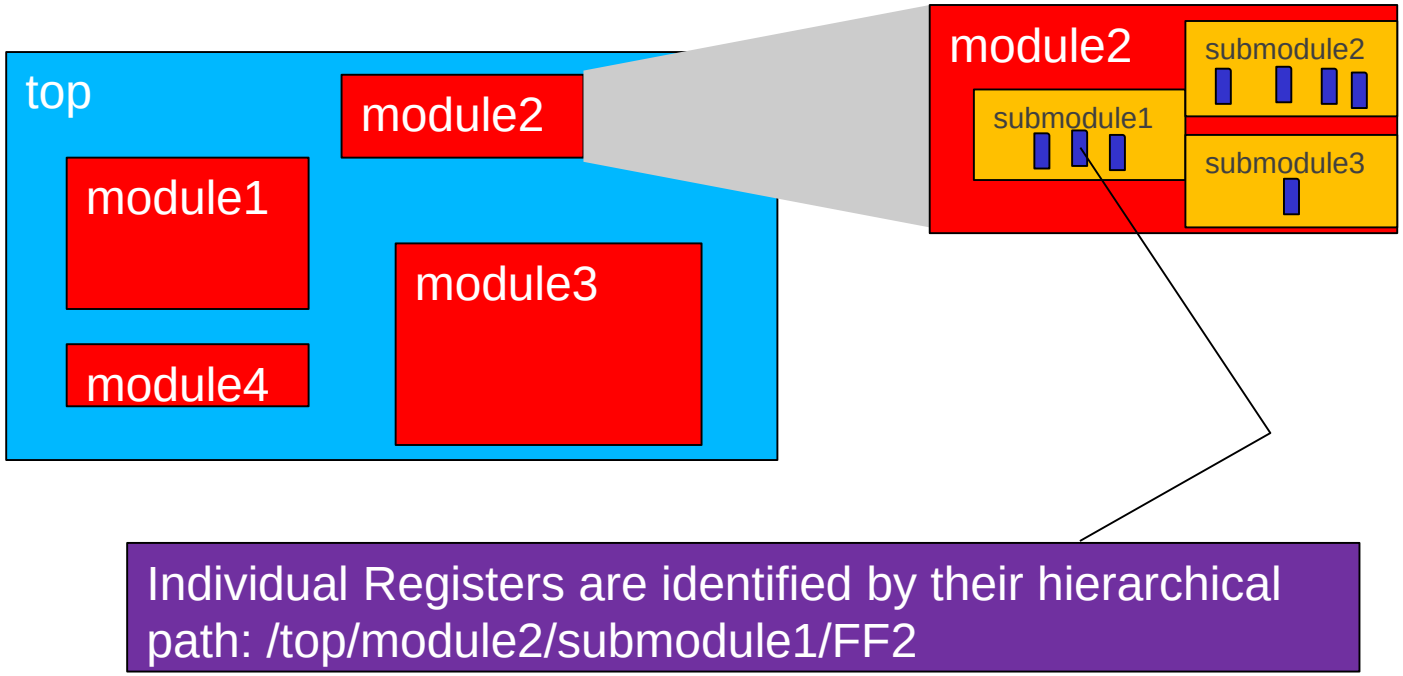
  Classification of the sensitivity to faults by FF: Extract K by FF.



Architectural properties

2. Hierarchical analysis

- FFs are identified by their name and hierarchical instantiation path.



Architectural properties

2. Hierarchical analysis

module	#faults detected	#faults injected	K
/top	21970	50000	43,6%
/top/module1	5466	18345	29,8%
/top/module2	3969	15347	25,8%
/top/module3	12355	16308	75,7%

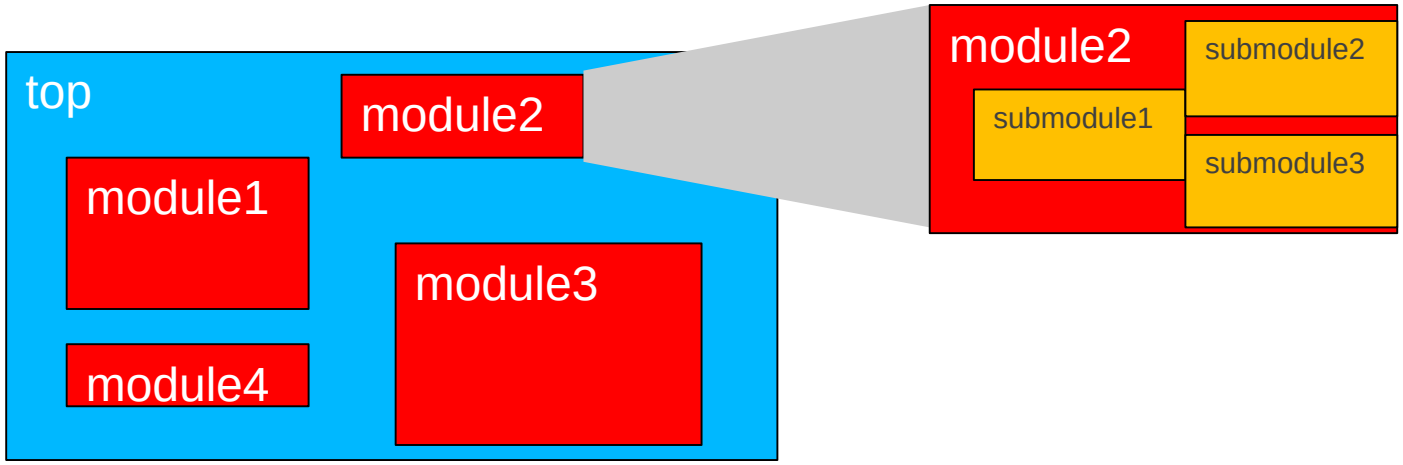
module	#faults detected	#faults injected	K
/top/module2	3969	15347	25,9%
/top/module2/submodule1	1302	5345	24,3%
/top/module2/submodule2	774	4798	16,1%
/top/module2/submodule3	1893	5204	36,4%

module	#faults detected	#faults injected	K
/top/module2/submodule1	1302	5345	24,3%
/top/module2/submodule1/FF1	533	1835	29,1%
/top/module2/submodule1/FF2	313	1356	23,1%
/top/module2/submodule1/FF3	456	2154	21,1%

Architectural properties

2. Hierarchical criticality analysis

- Criticality levels are assigned to



Let's classify all the hierarchical modules by the criticality obtained by the index. The result is, for each module, a measurement of the CONTROLLABILITY and the OBSERVABILITY of faults.

Hierarchical criticality is a method for the **ANALYSIS** of the capacity of the system Workload-Design to propagate faults to primary outputs.
During the design phase the designer can repeat several times the process and improve the internal rates.
It also provides a methodology to **selectively** introduce mitigation strategies

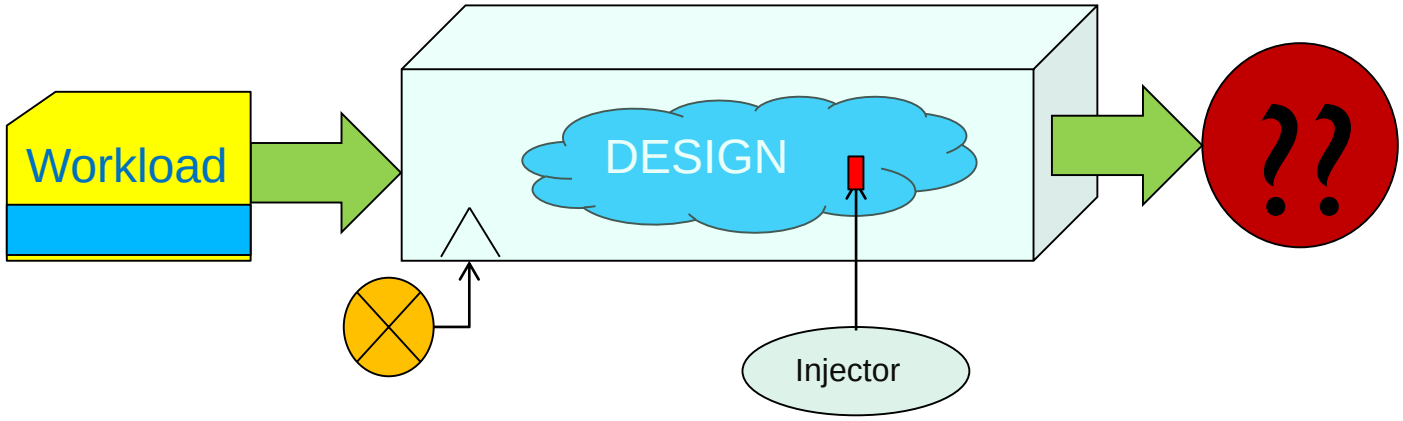
3.2 Fault models

Three relevant fault models :

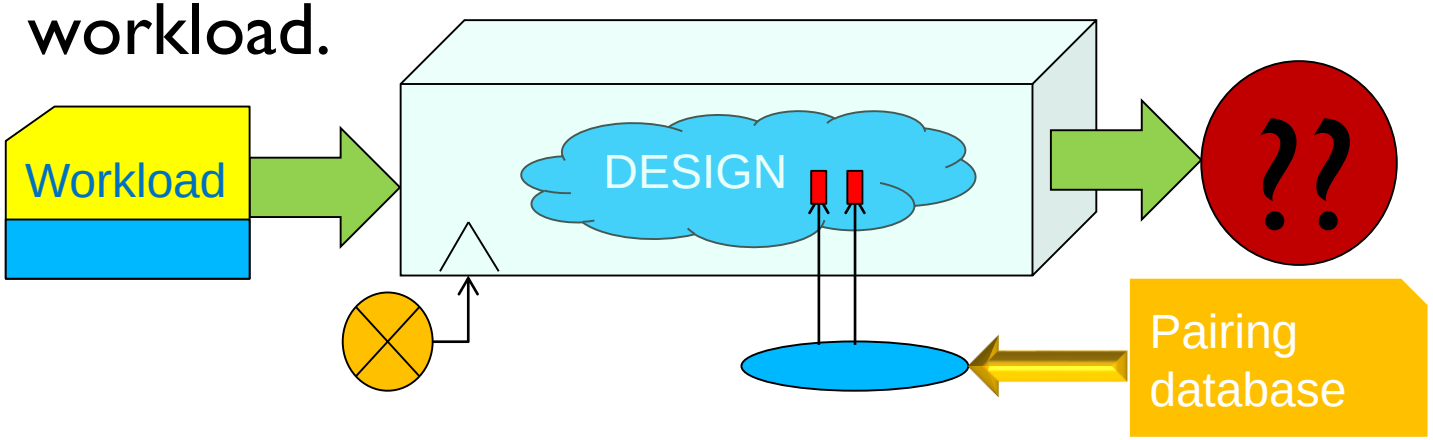
- Single Event Upset
  - Modify one single memory cell
- Multi Bit Upsets
  - The physical description has to be considered to introduce multiple modifications
  - Pairing two or more registers is a solution.
- Single Event Transients
  - Pulses depending on several factors, mainly ion impact energy
  - Propagation drives to a set of FFs paired by the logic cones.

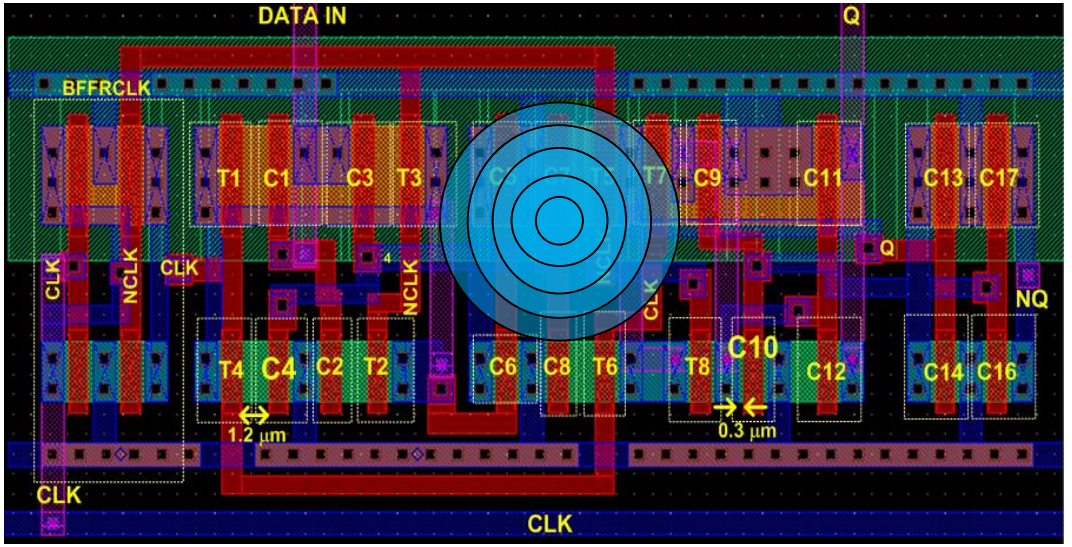
Fault models

- SOFT ERRORS that can be represented in a fault injection system are:
  - **Single Event Upset** is modeled by a single BIT-FLIP at any memory location of a circuit, at any clock cycle of the workload.



- **Multi Bit Upset** is modeled by simultaneous BIT-FLIPs at a number of adjacent memory locations of a circuit, at any clock cycle of the workload.





Single Event Transients

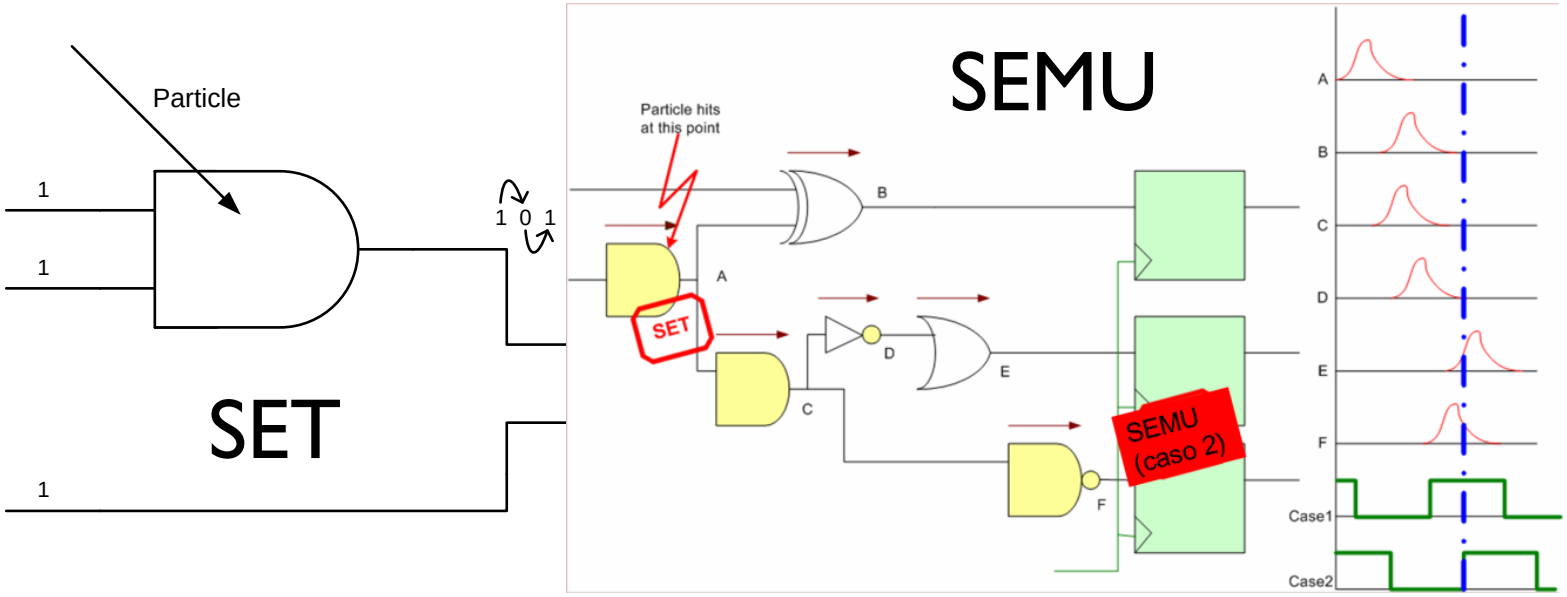
There are several approaches.

Two stages of fault injection: Logical analysis + FI.

- Single event upsets are phenomena that involve energy and LET. Pulses are propagated under three assumptions:
  - If the impact hits with **Enough energy**
  - If the internal logic avoids the **Logical filtering**
  - If the impact instant is inside the **Capture window of FFs** (clock frequency)
- As FI represents a worst case of the physical event, the logical analysis is made considering that the logical filtering allows the propagation of pulses to one or several registers.

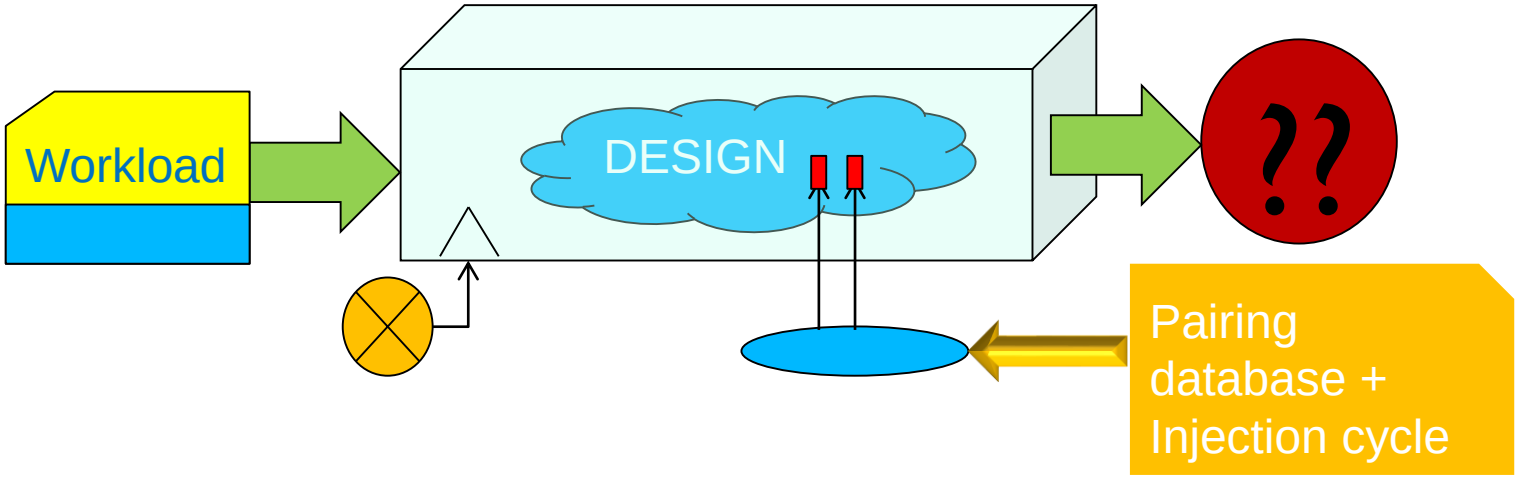
Single Event Transients

- From the logical analysis the pulses are propagated to the registers that capture the state.
- Depending on the energy of the impact the width of the pulse, the logical analysis provides different register pairings.



Motivation
Basic Concepts
Emulation Techn
Other Uses
CASE STUDIES
Fault Inj. in SRAM FPGAs

Single Event Transients



- A pairing database with injection time feed the injection system and assesses the logical effect of

MA Aguirre, V Baena, J Tombs, M Violante. "A new approach to estimate the effect of single event transients in complex circuits" Nuclear Science, IEEE Transactions on 54 (4), 1018-1024

3.3 Gross numbers.

- After a campaign…What do we obtain at a glance?

First is a gross number:

For a *design*, for a *workload*, for a set of *faults*

$$K = \frac{\#\text{faults detected}}{\#\text{faults injected}}$$

A representative K helps to the designer and the test engineer

Is this K representative of a real case?
Which are the limitations of the approach?
Can we extract more information?

- The answer is **no**, K is not representative if certain considerations are not taken.

Undetected faults can:

- Remain present in the design (latent)
- Be corrected (silent)
- … or simply unintentionally overwritten.

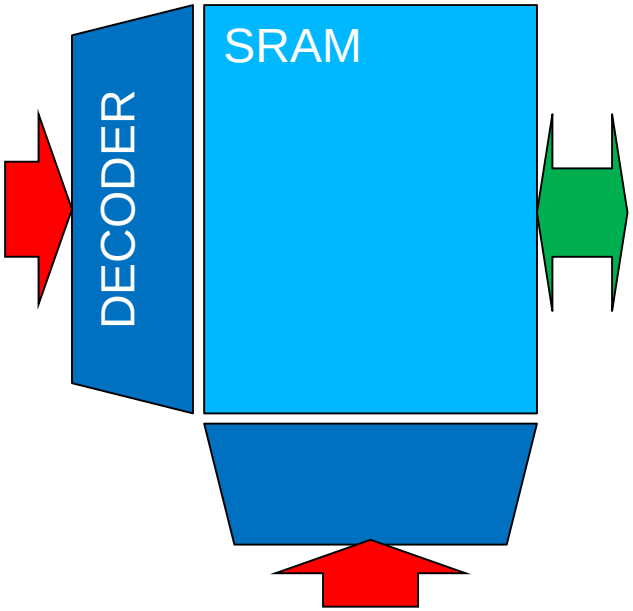
$$Kr = \frac{\#\text{faults detected} + \#\text{faults latent} + \#\text{fault overwritten}}{\#\text{faults injected}}$$

- But… in the beam there is no way to inspect the internals of one device so… **K has to be representative!!!**

Kr > K

A simple example

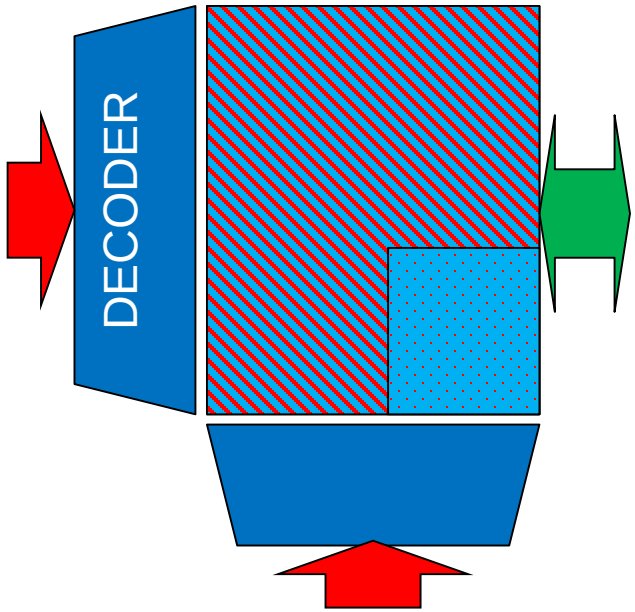
- An SRAM memory with different workloads



- An SRAM memory is a very observable device. Just read all the memory map and obtain all the effects.
- Typically all-1's is preloaded and then exposed to the beam.
- The all-0's is preloaded and then exposed to the beam.
- Discrepancies due to particle hitting are easy to collect
- Cross sections are calculated, with fluences, energies and data collected.

- Imagine that it is forgotten to read a small part of the SRAM

Particles hitting that cause SEE are not detected at the outputs



- Imagine that during the test the memory is written

Particles hitting that cause SEE are erased, being overwritten by the initialisation process,

Monitoring the primary outputs is not enough. Following this approach a latent analysis at each run has to be performed to obtain a refined K

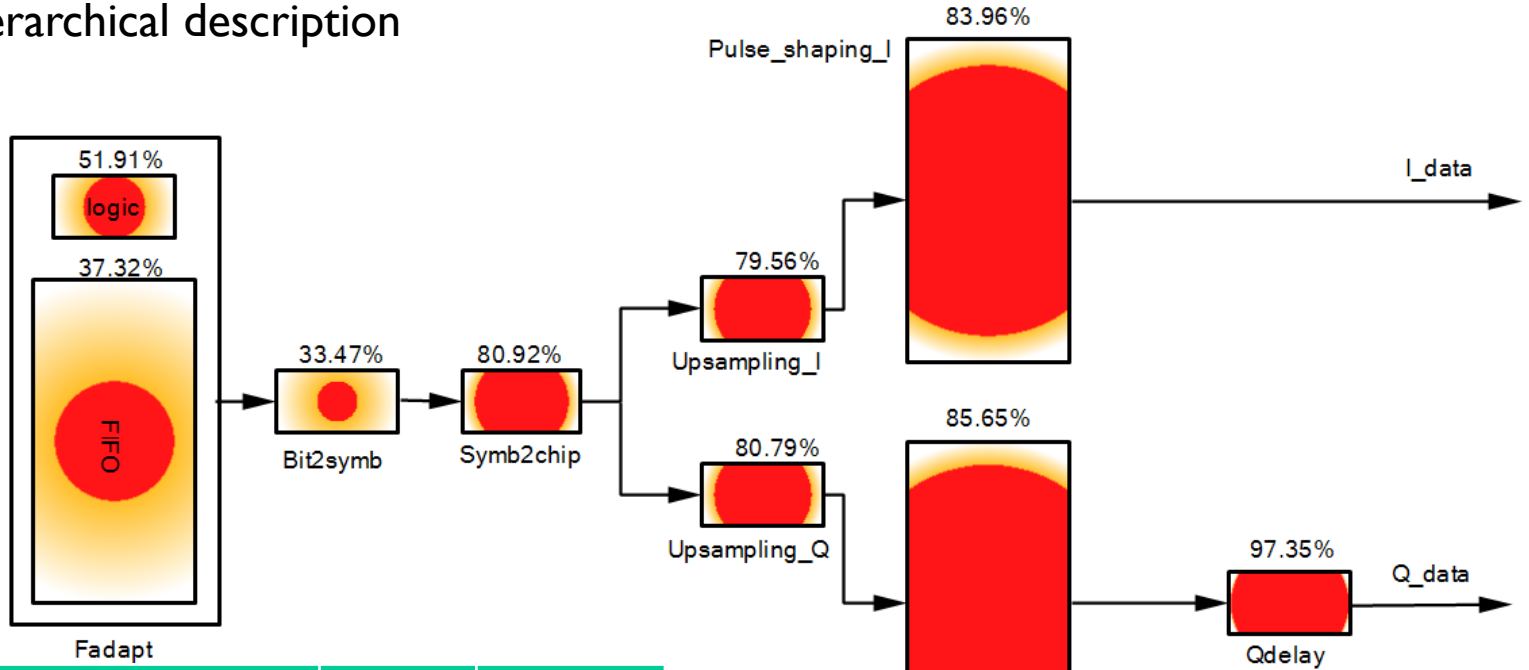
Latent faults are events that are not propagated to primary outputs due to or an absence of clear paths or due to bad stimuli set

Open questions:

1. Can the FI extract more information about the design behavior?
2. Is possible to check the architecture in depth? (mitigation techniques…)
3. Can K be improved in any way?
4. How other Single Event Effects can be modeled?

3.4 An example

Measurements of fault injection applied to a IEEE 802.15.4 transmitter
Hierarchical description



Module	Flip-Flops	% Flip-flops
All tx	388	100
fadapt (logic + fifo)	109	28.09
fadapt_logic	30	7.73
fadapt_fifo	79	20.36
bit2symb	13	3.35
symb2chip	24	6.19
upsampling_I	11	2.84
upsampling_Q	11	2.84
pulse_shaping_I	86	22.16
pulse_shaping_Q	86	22.16
Qdelay	48	12.37

Decisions

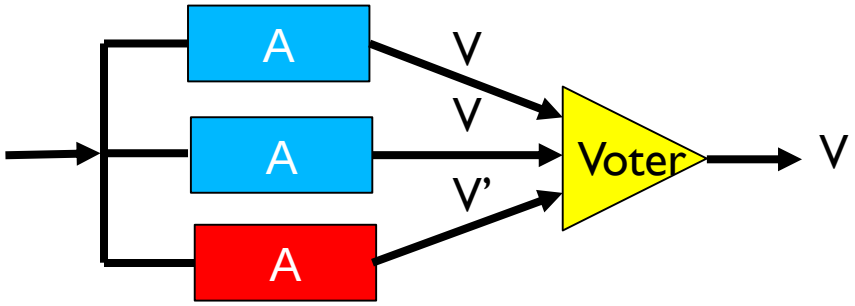
What to do with this information:

- Architectural vulnerability factor
- Latency of SEE
- Hierarchical analysis
- Selective Protections
- Observability of faults

Architectural properties

3. Mitigated or fully unprotected version of the design.

- Inject over a mitigated version:



- V' is mitigated and V is thus propagated. Mitigation structures reduces the OBSERVABILITY of the faults.
- K decreases to zero if the design is better mitigated.

Architectural properties

3. Mitigated or fully unprotected version of the design.

- But…
  1. The design that is fabricated and tested in the beam is the mitigated version.
  2. Typically the protections become ineffective over certain LETs.
  3. Failures, if existing, should be propagated to the primary outputs

The recommendation is to assess the Fault Injection campaign over **both** versions of the netlist:
-non mitigated to obtain a good assessment of the workload quality (Observability of faults)
-mitigated to detect abnormal situations due to redundancy collapsing in the synthesis process.

4.Other Uses of FIN

4.1 Relaxing the comparator (Systemic view)

4.2 Quality of Workload.

4.3 Fault diagnosis

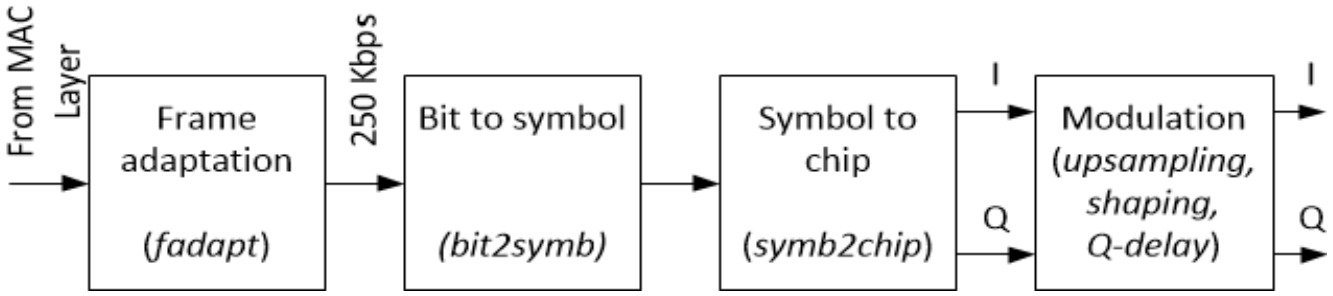
4.4 Microprocessors

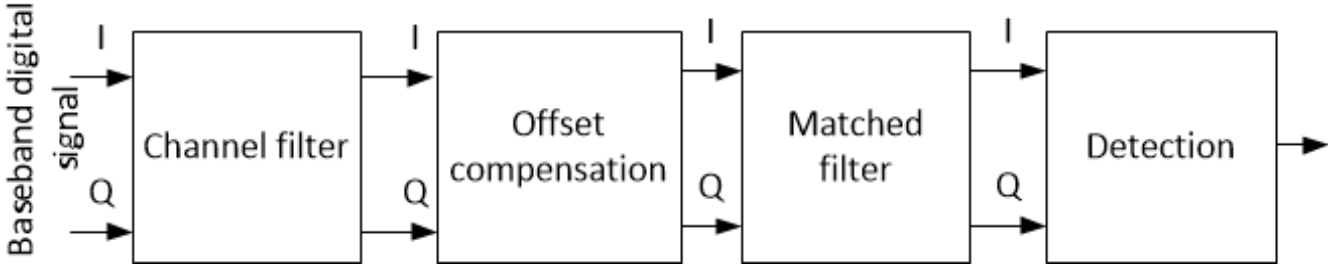
4.5 Initialization Policy

4.1 Relaxing the comparator

- It is used to detect faults at primary outputs.
- Use of a single bit comparison is very strict.
- The comparator have to be flexible enough

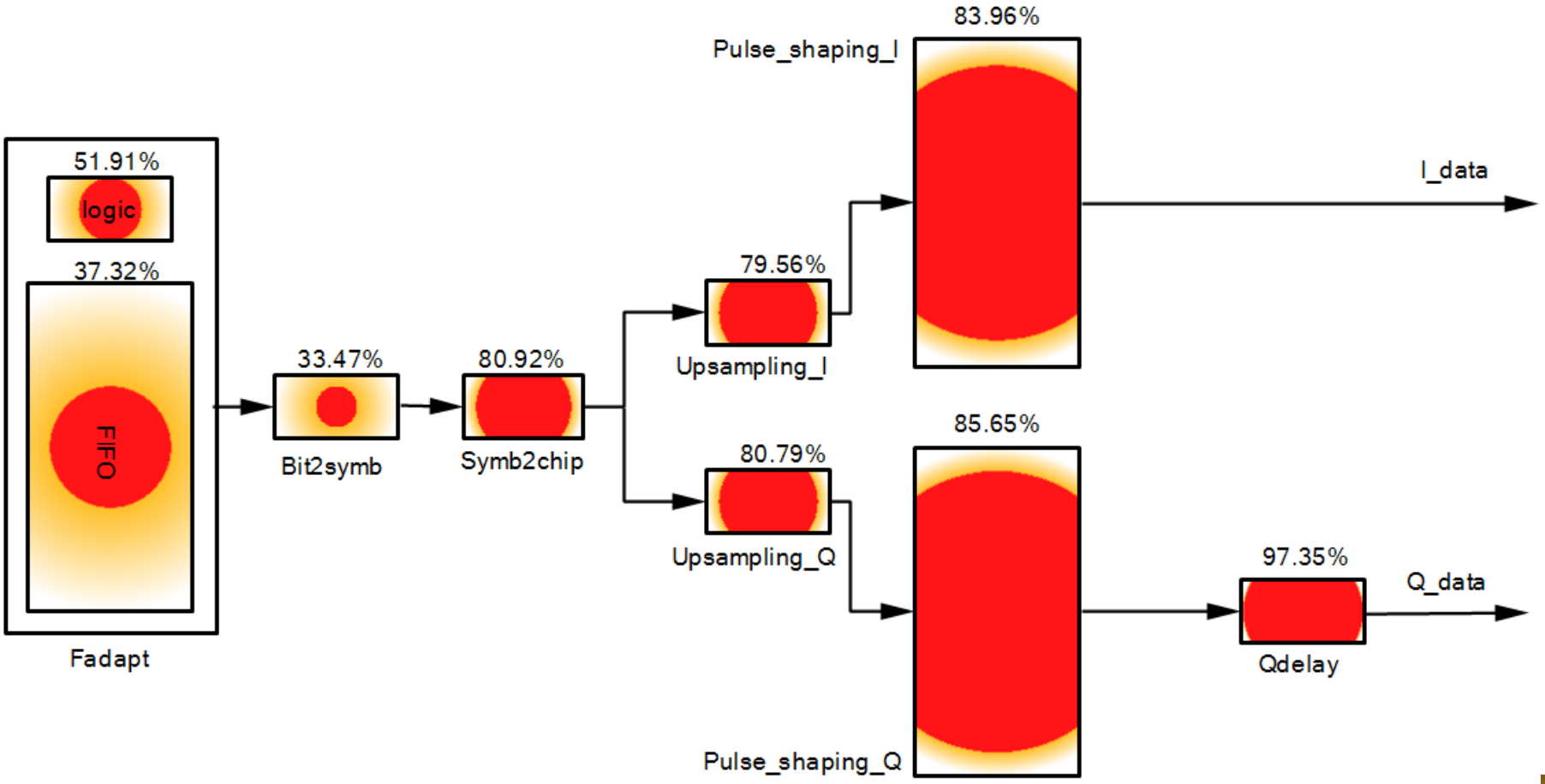
If your design car recover data you can mitigate "selectively" your design.


From MAC Layer
Frame adaptation
(fadapt)
250 Kbps
Bit to symbol
(bit2symb)
Symbol to chip
(symb2chip)
I
Q
Modulation
(upsampling, shaping, Q-delay)
I
Q

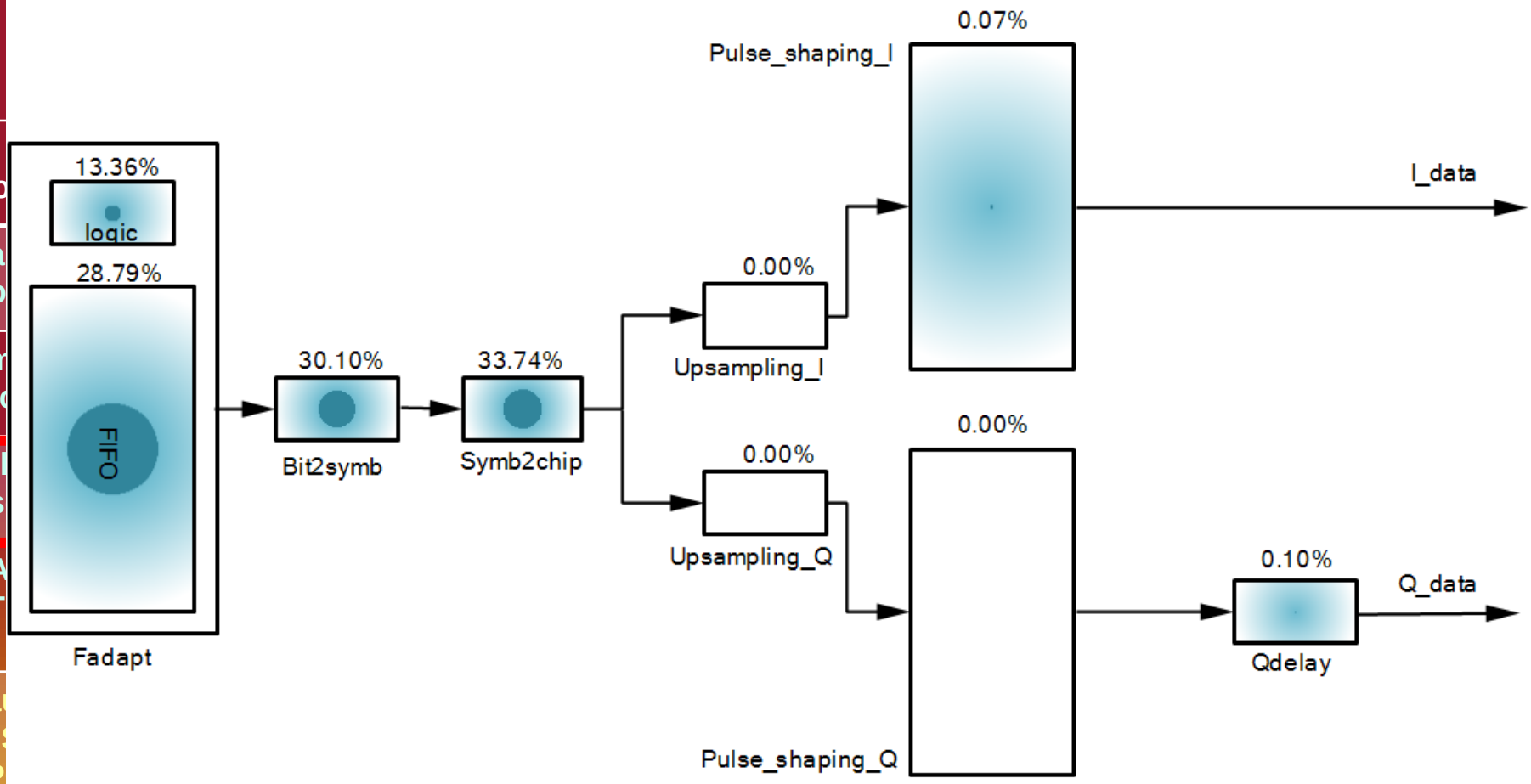

Baseband digital signal
I
Q
Channel filter
Offset compensation
Matched filter
Detection

Heat Zones

- Cycle-by-cycle comparison suggests system is very sensitive



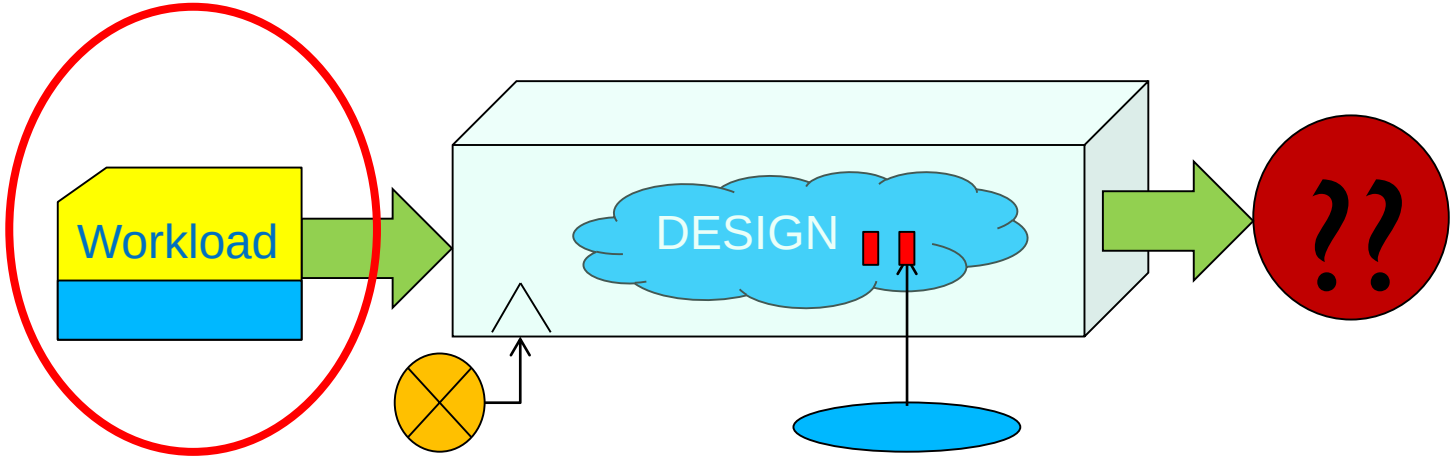
- But… can the receptor recover correct data from a "wrong" transmitted frame?



4.2 Quality of Workload.

- A Fault Injection platform can be seen using another perspective: Assess the workload.
  - The criticality analysis is a matter of controllability and observability of faults.
  - Let us think, instead on the design structure, on the set of stimuli.
  - We can make the faults more controllable and observable, if the adequate sequence of inputs are applied

Quality of Workload.

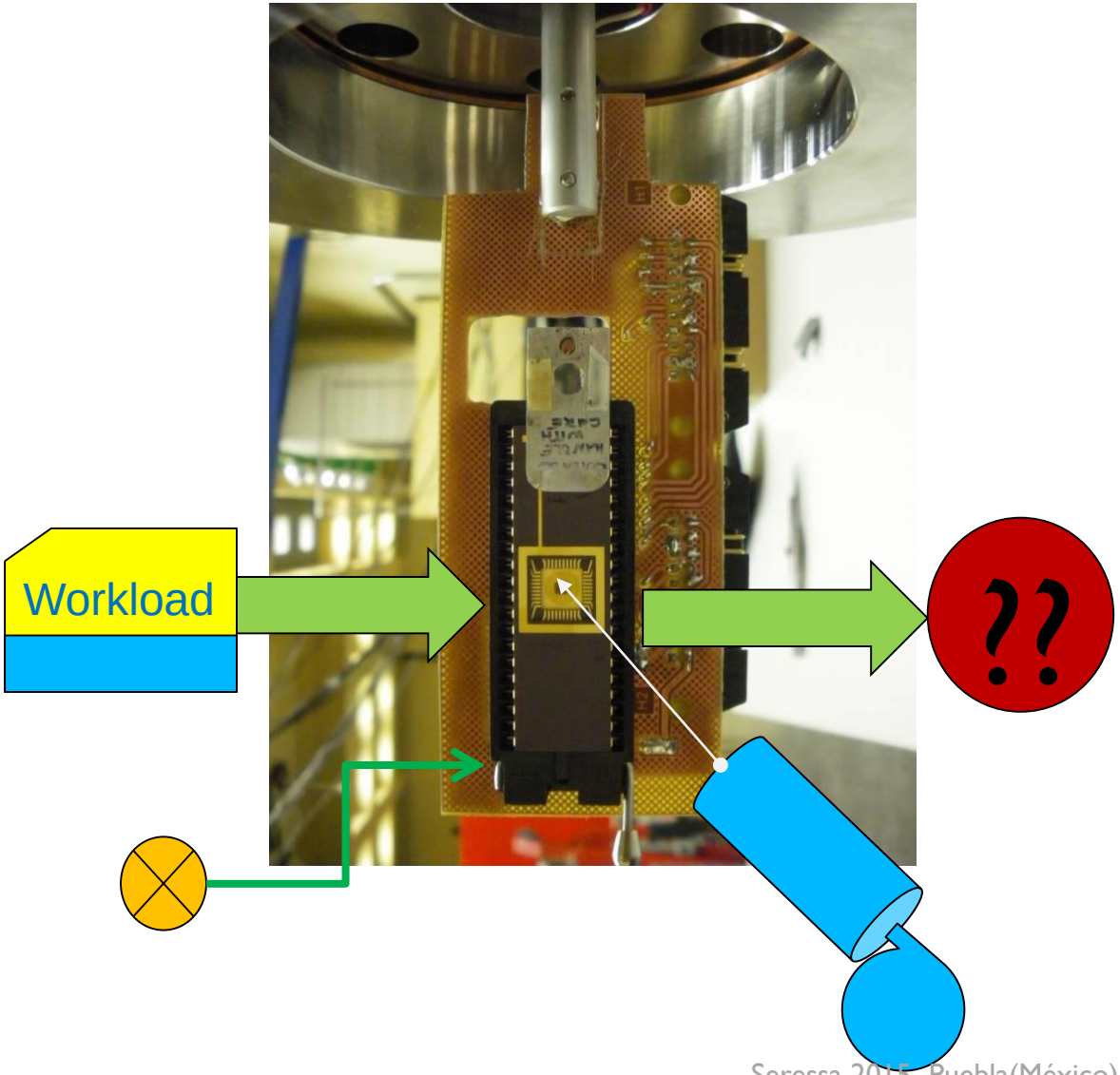


- A hierarchical analysis provide an assessment of the quality of the stimuli set. But… what for?

Let us think on the beam testing.

- The observability is reduced to monitoring the PRIMARY outputs

Quality of Workload.



- The quality of the workload is an index of capacity of detection of faults.
- The workload is part of the set-up in a dynamic beam experiment.
  - The beam experiments should be with the best capacity of control and observation of faults, but…

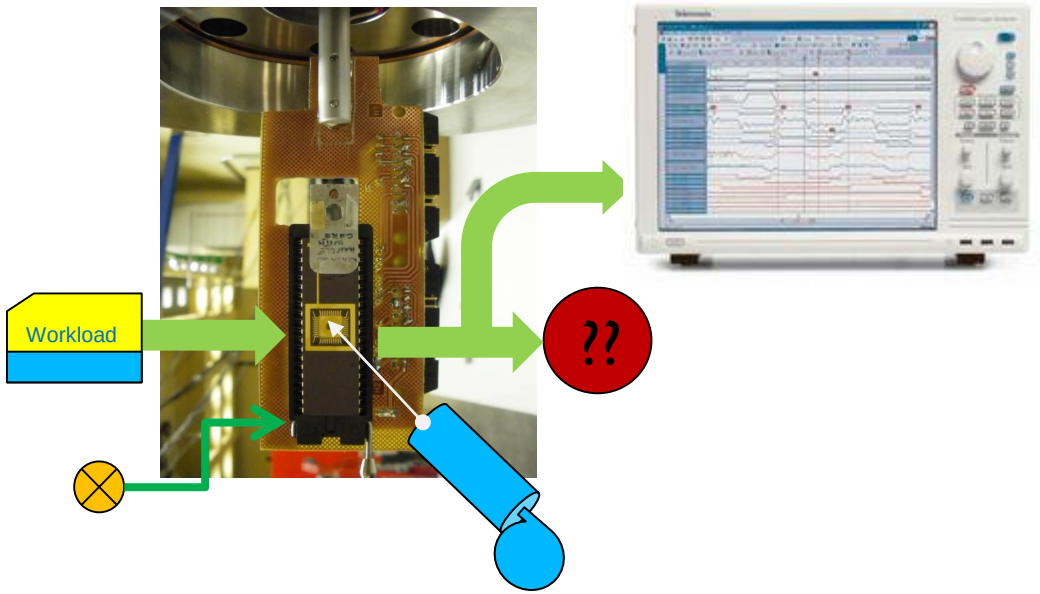
What happens if a fault is detected?
How to proceed?

4.3 Fault diagnosis

- During the beam experiment a fault is detected.
- ….
- ….
- ….
- ….
- ….
- ….
- …. We have a problem

Fault diagnosis

- The test engineer is not the design engineer.
- The beam test normally records the waveforms at the primary outputs.



- Can the Fault Injection procedures help to the beam test?

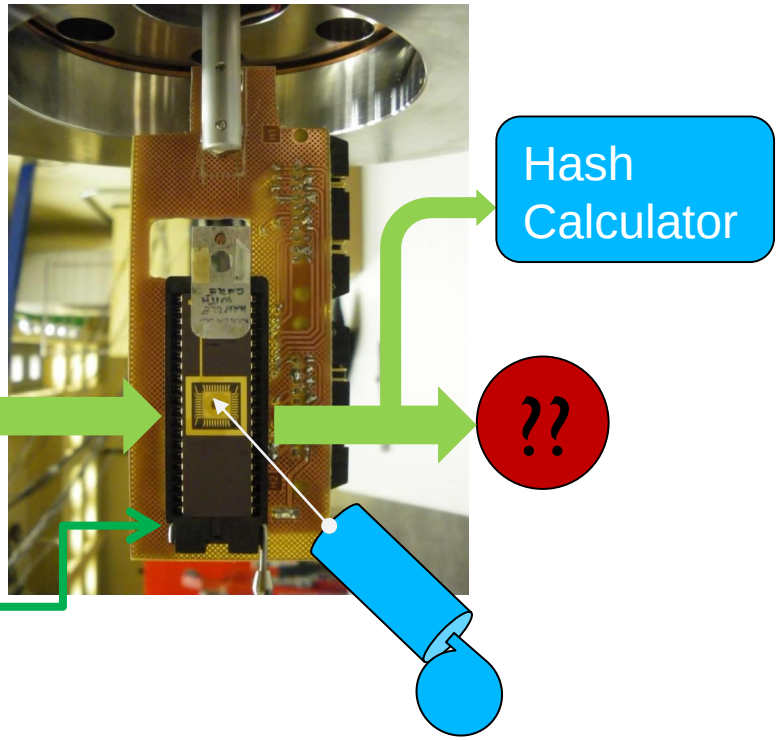
Fault Diagnosis

- A solution is to complement the FI dictionary with additional information:
  1. In-beam: record output waveform until the end of the workload, to characterize the fault
  2. In-FI: Create a database
     1. Where, When, How and **Output** waveforms
  3. Pattern matching with the in-beam outputs
- The dimension of the database of output waveforms is tremendous
- Pattern matching is not an easy task.

Fault Diagnosis

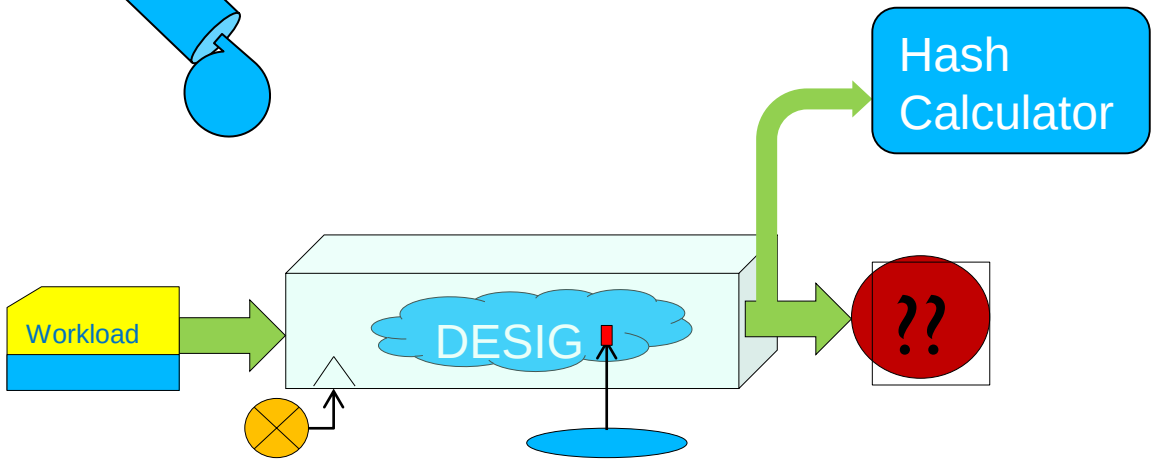
- Hash codes: A function that is used to detect failures of the consistency in a sequential system.
  - There are many formulations of hash codes.
  - Let's select one that requires, in its hardware view, one clock cycle for computing.
  - It monitorizes primary outputs. It is external, non invasive.
- Fault dictionary takes the form:
  - Where, When, How and Hash code

Fault Diagnosis



Hash Calculator is a resource that is externally connected.

Each injection is run until the end of the workload to record the hash code



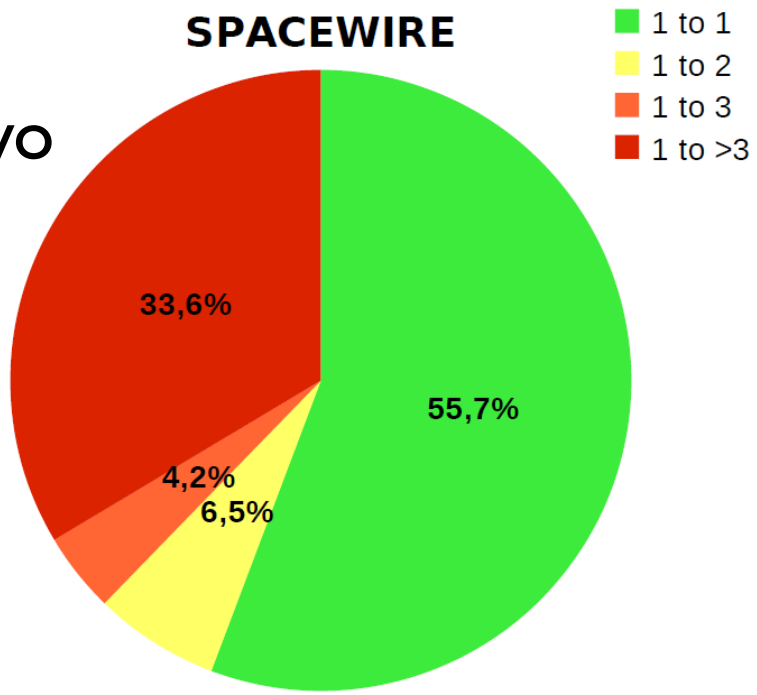
Fault Diagnosis

- Fault dictionaries relate a given signature to an internal fault

Signature	Affected Nodes	Clock Cycle
….	….	….
#F091DCE2	/DRIVER2/driver/data_reg<35>	04421
#BAD1F009	/alu/aux_reg<4>	15673
#DA09F56B	/driver/ADDRmem<13>	25670
#100DCC35	/tx_buffer/sr<31>	65389
….	….	….

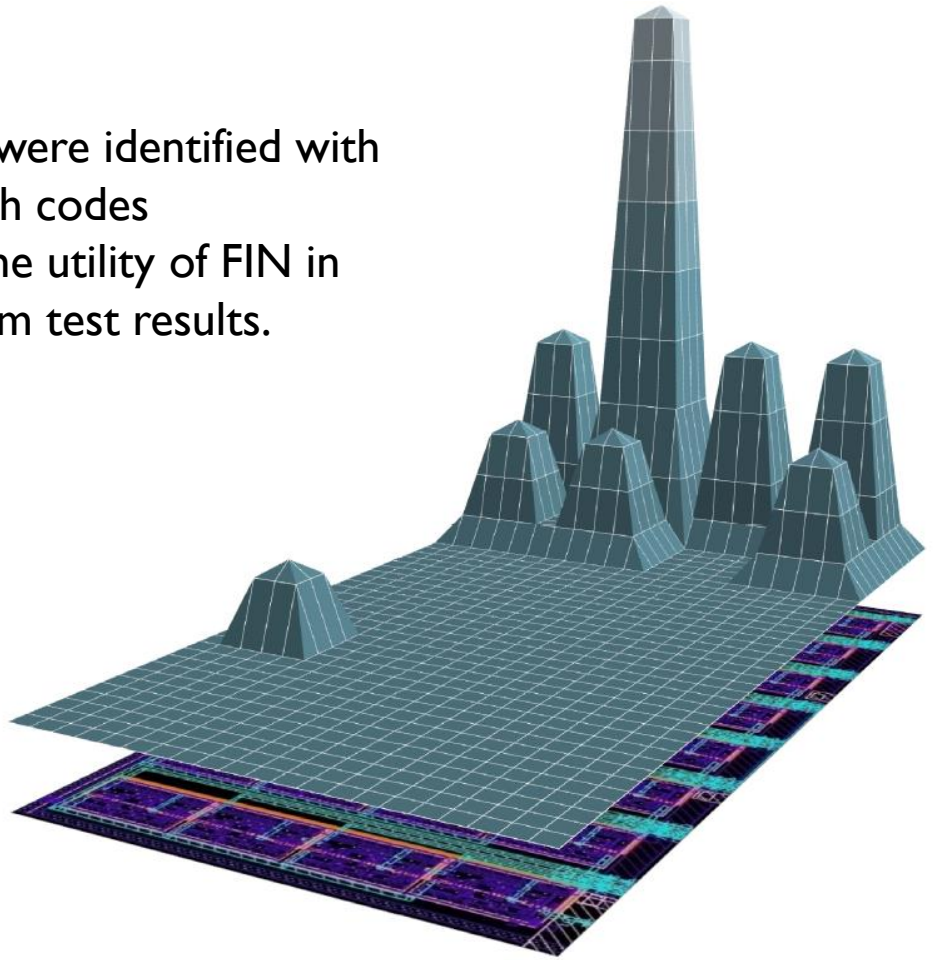
Fault Diagnosis

- SpaceWire Codec IP core
  - **56% of errors are univocal**
  - 7% of errors have two candidates
  - 37% of errors have three or more candidates



Results

All the hitting particles were identified with their corresponding hash codes
This technique shows the utility of FIN in the identification of beam test results.



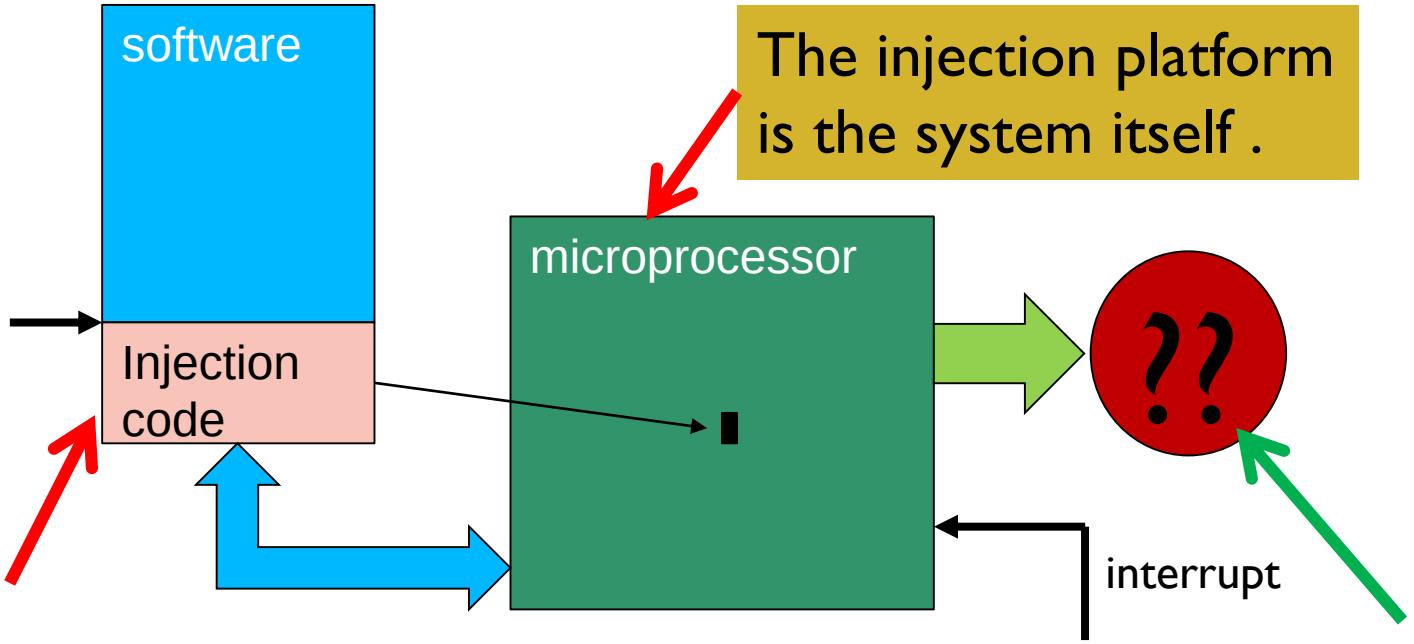
Fault diagnosis

- Radiation test results are difficult to interpret
- Hash code allows the identification of faults in radiation experiments
- Hash codes help to define candidates to study when an error is found.
- Fault dictionary should be generated over the unprotected version

A Metric for the Workload Quality is defined from the number of faults propagated, Q, but as the unique observable available are the outputs of the circuit, a second Metric is defined using the hash codes and the UNICITY of them.

4.4 Microprocessors

- Micros are devices with a very particular architecture.
- Not always the design description is available

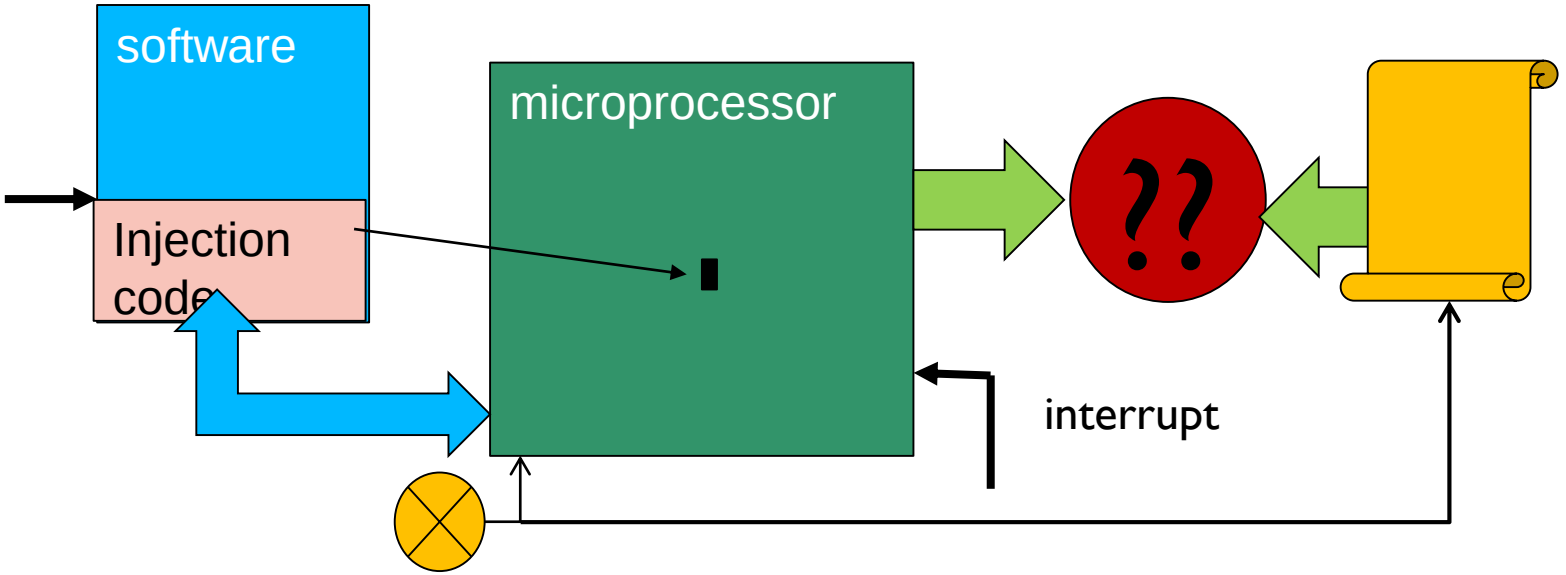


Microprocessors

- Injection is produced using pieces of code that introduce bit-flips under the execution of an event, i.e. an interrupt, and a spontaneous change in an accumulator.
- Software protections are extra instructions that produce self checks and self corrections, supervising the program flow or data flow.

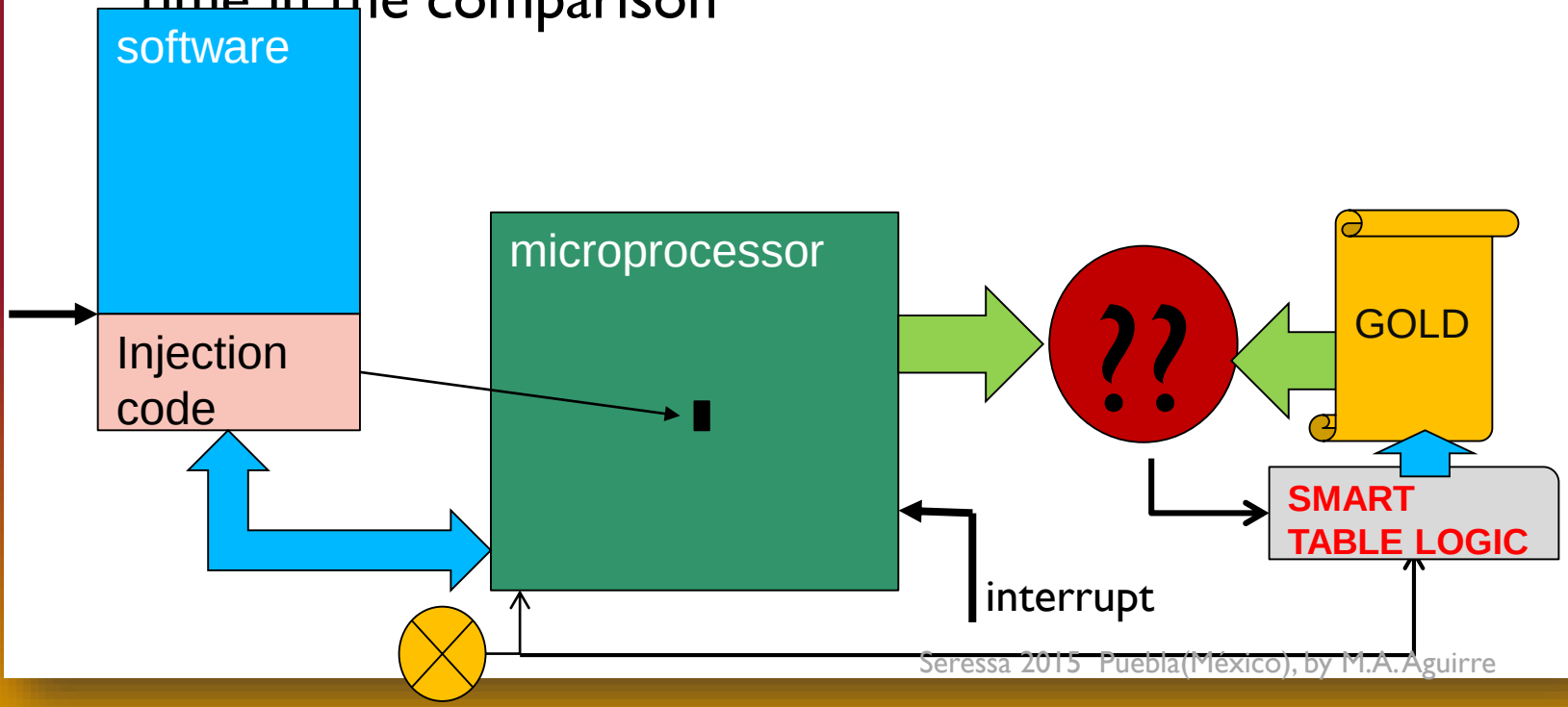
Microprocessors

- Not always the structure is completely reached
- Fault detection is made monitoring the primary outputs
  ...
- … and comparing with a "gold" execution



Microprocessors

- The comparison is not made cycle accurate.
- The overhead is to apply extra routines for correcting the faults->extra clock cycles are needed with respect to the gold table
- Introduce extra logic that takes considers the repairing time in the comparison



Microprocessors

- Due to the "smart table" the flexible comparator allows relaxed comparisons, non clock-cycle accurate
- The fault correction process takes extra clock cycles until the processor is realigned

4.5 Initialization Policy

- There are FFs WITHOUT reset signal
- Latency means the internal presence of a fault at the end of the workload,
- Then, restart for the next run
- The first stimuli should a be "*reset sequence*"-> then a latency check is performed.
- Detect if there are faults still present!!

Persistence of faults in consecutive RUNS reveals that the circuit is not re-starting from a known state.
Is the initialization policy correct?

How is it detected?

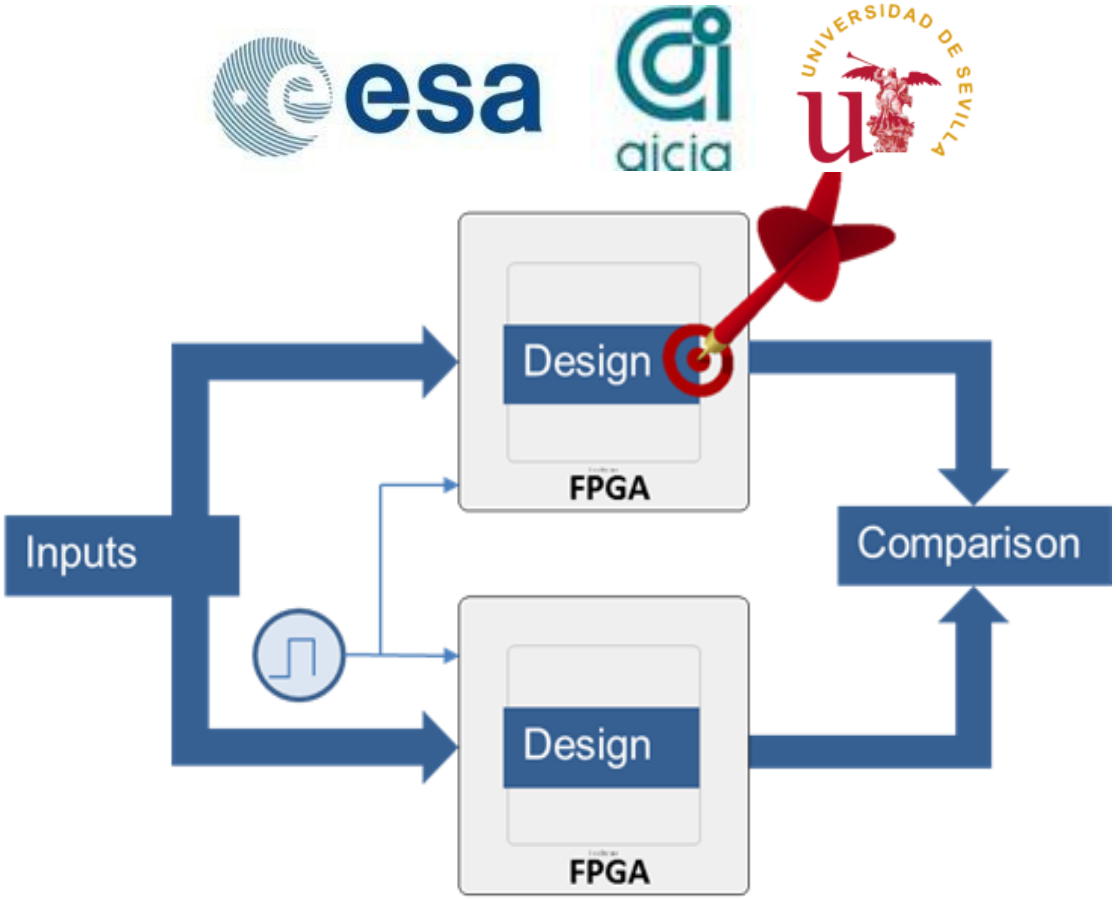
We found some faults detection in clock cycles BEFORE the injection time.
How it is possible?

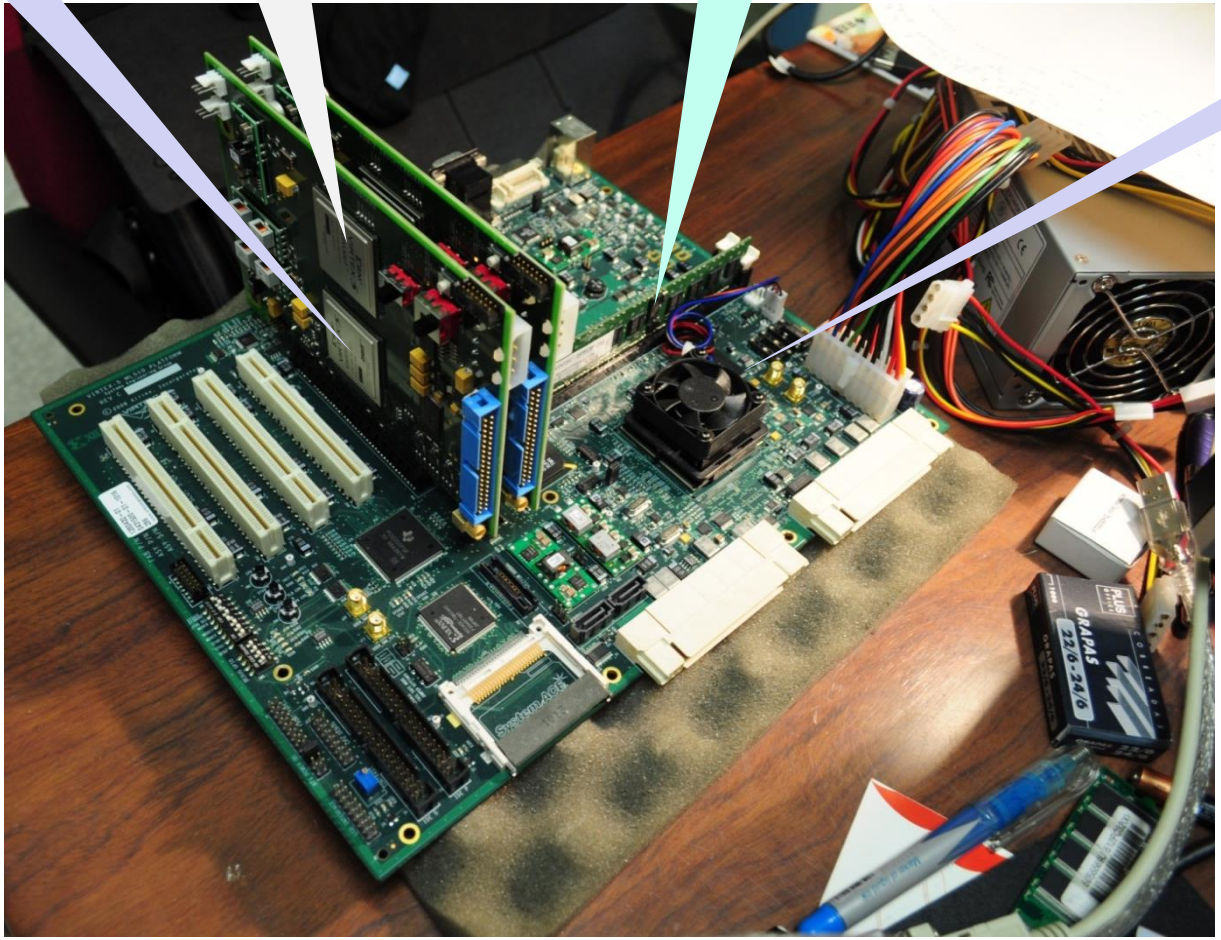
Circuit REMINDS faults propagated to non reset FFs due to PREVIOUS injections. We call this as “Memory Efffect

Take care about your INITIALIZATION policy

FIN is for digital circuits an ideal methodology to debug a design under a SEE in the earliest stages of the netlist, detect errors and take decisions, during DESIGN PHASE.

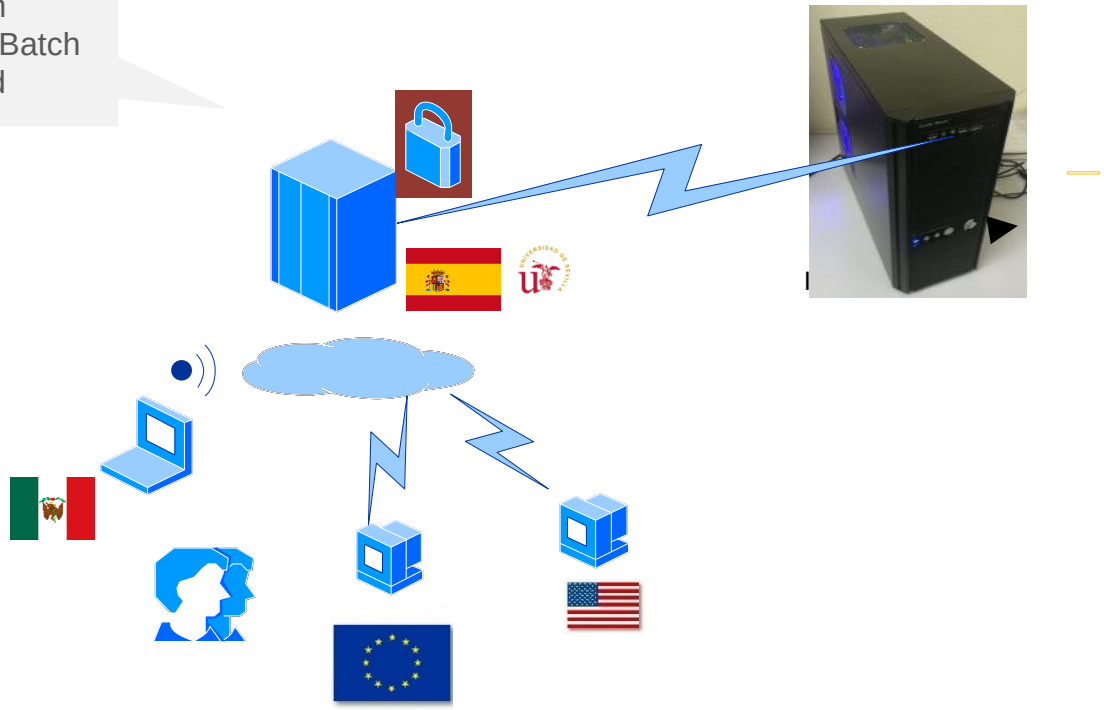
5. Case studies. Tool description.



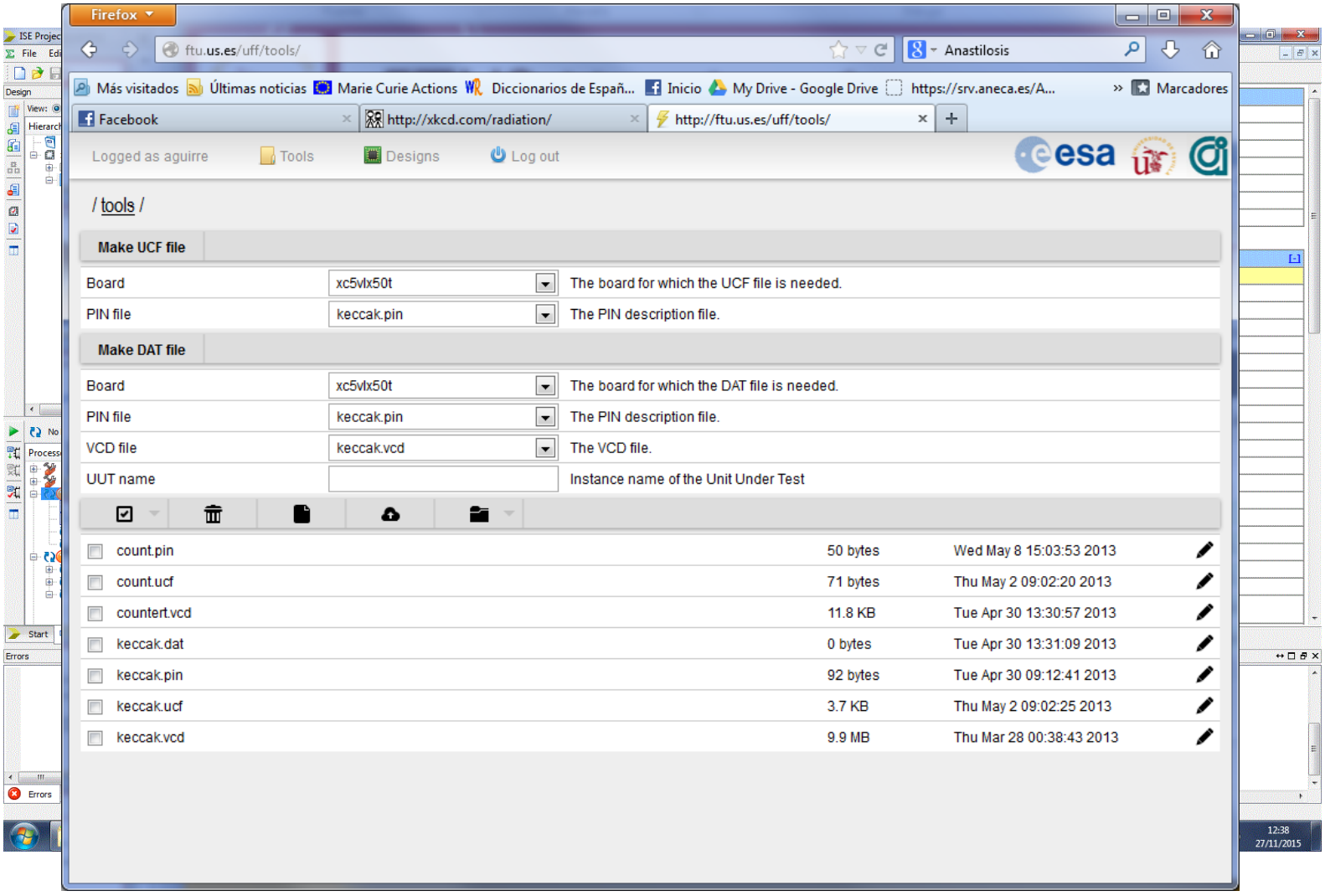

Service FPGA
Target FPGA
DRAM Memory
Control FPGA

5.0 Connecting to Universidad de Sevilla

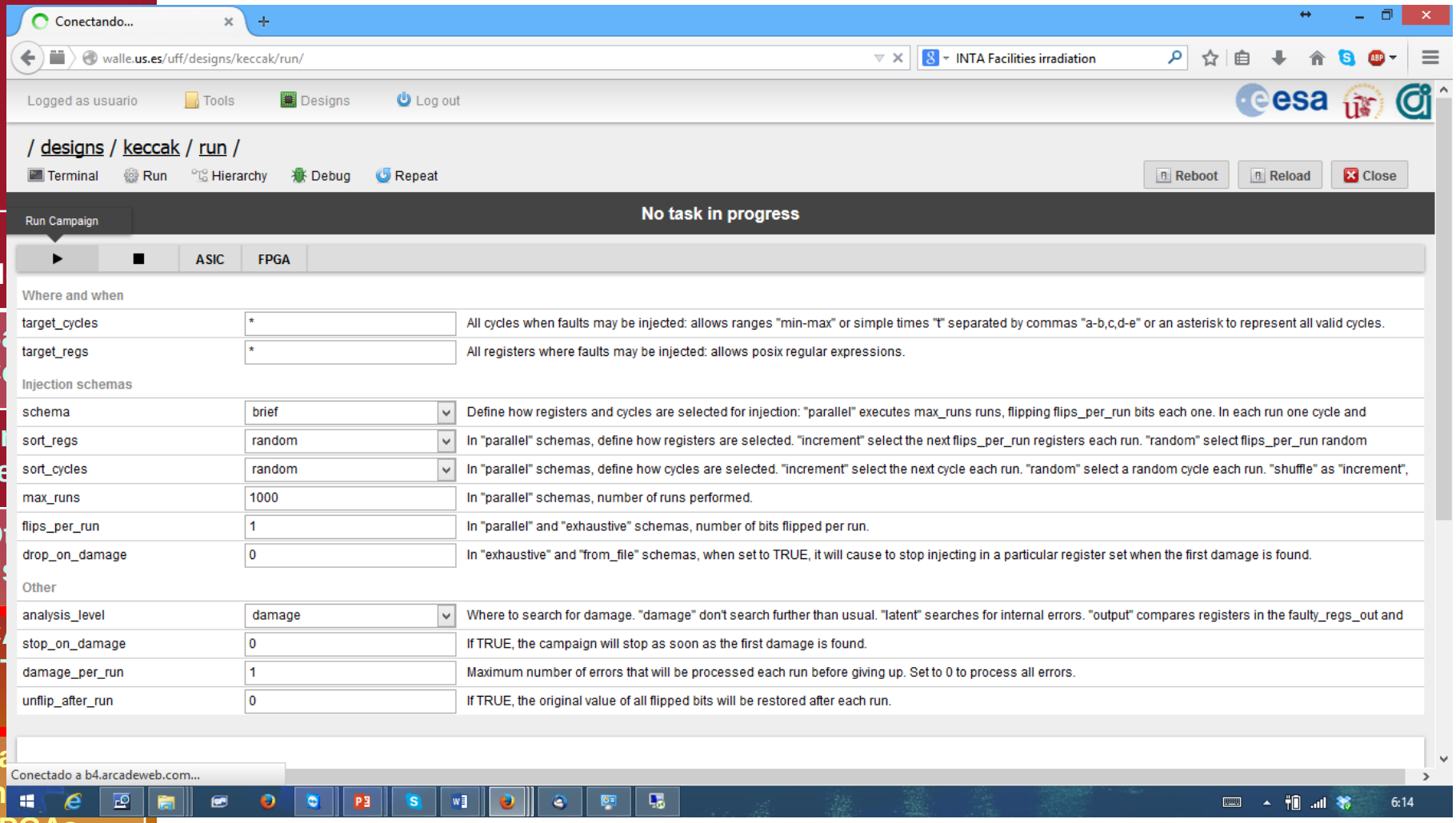
- Bitstream
- Injection Batch
- Workload

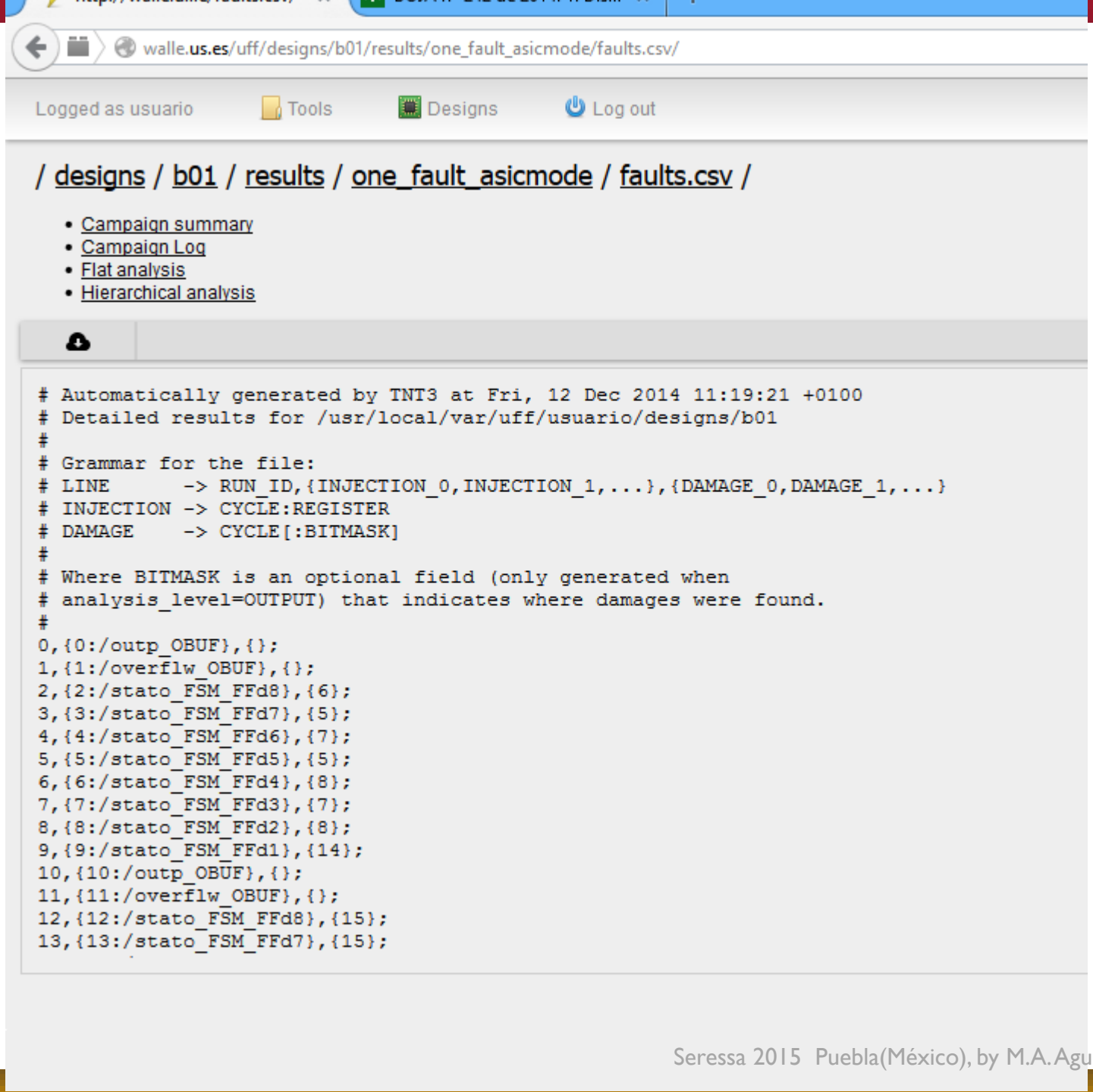


Preparation

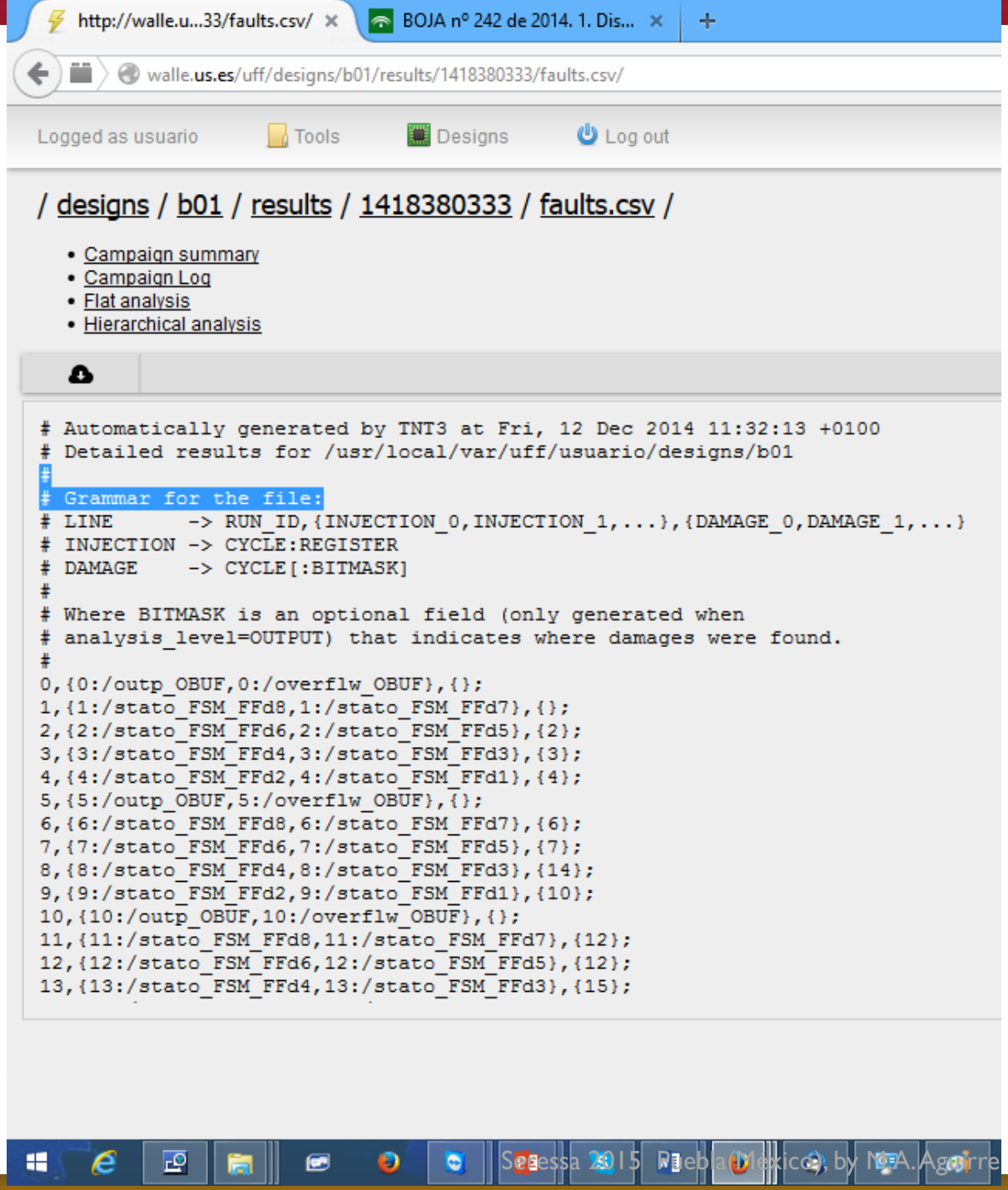


Options for campaigns in ASIC mode




walle.us.es/uff/designs/b01/results/one_fault_asicmode/faults.csv/
Logged as usuario
Tools
Designs
Log out
/ designs / b01 / results / one_fault_asicmode / faults.csv /
Campaign summary
Campaign Log
Flat analysis
Hierarchical analysis
Automatically generated by TNT3 at Fri, 12 Dec 2014 11:19:21 +0100
Detailed results for /usr/local/var/uff/usuario/designs/b01
#
Grammar for the file:
LINE -> RUN_ID,{INJECTION_0,INJECTION_1,...},{DAMAGE_0,DAMAGE_1,...}
INJECTION -> CYCLE:REGISTER
DAMAGE -> CYCLE[:BITMASK]
#
Where BITMASK is an optional field (only generated when
analysis_level=OUTPUT) that indicates where damages were found.
#
0,{0:/outp_OBUF},{};
1,{1:/overflw_OBUF},{};
2,{2:/stato_FSM_FFd8},{6};
3,{3:/stato_FSM_FFd7},{5};
4,{4:/stato_FSM_FFd6},{7};
5,{5:/stato_FSM_FFd5},{5};
6,{6:/stato_FSM_FFd4},{8};
7,{7:/stato_FSM_FFd3},{7};
8,{8:/stato_FSM_FFd2},{8};
9,{9:/stato_FSM_FFd1},{14};
10,{10:/outp_OBUF},{};
11,{11:/overflw_OBUF},{};
12,{12:/stato_FSM_FFd8},{15};
13,{13:/stato_FSM_FFd7},{15};

Multibit, MBUS



5. Case studies

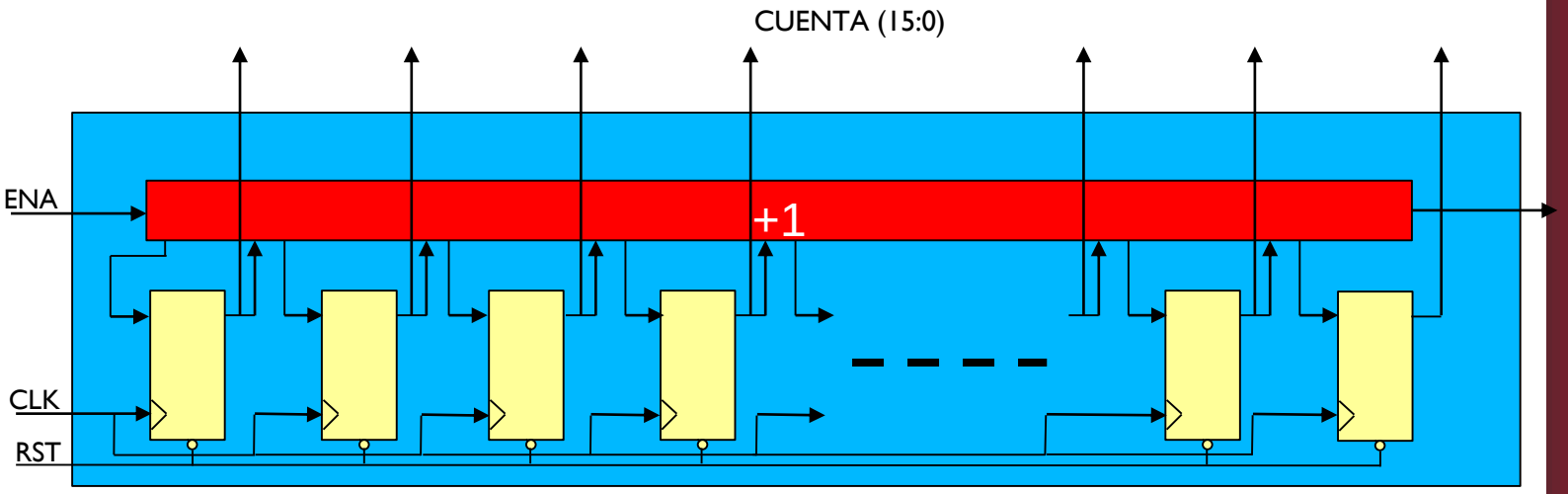
5.1 Fully implemented 16 bits counter

5.2 Half protected 16bits counter

5.3 openmsp430 with peripherals

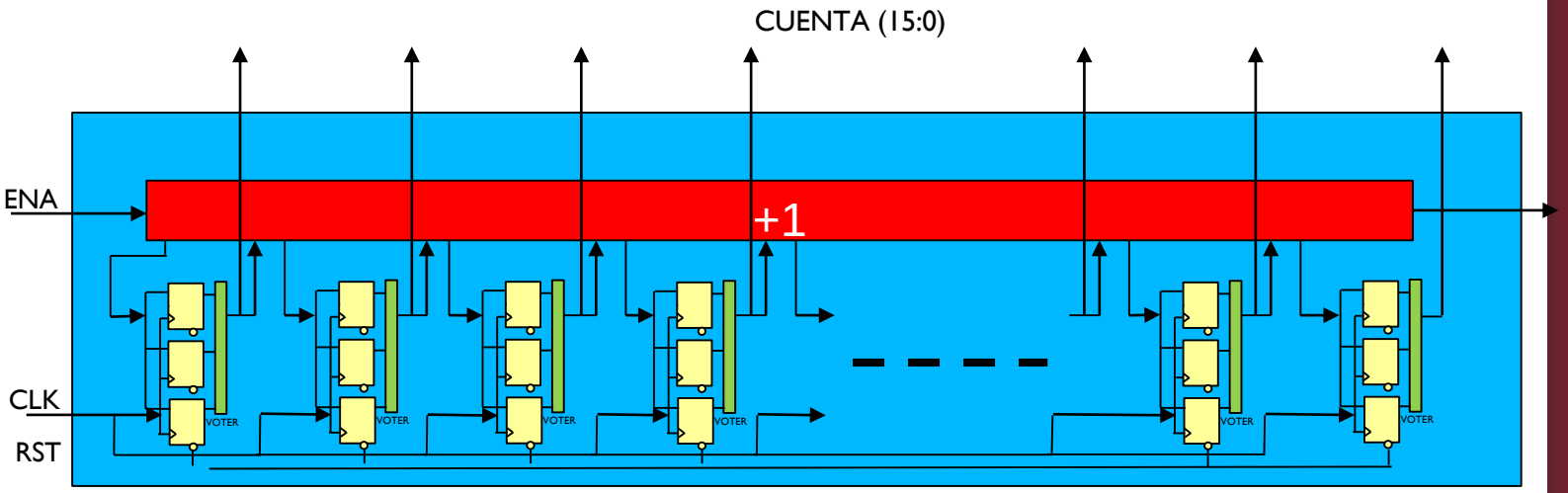
5.4 openmsp430 selective injection

5.1 Fully implemented 16 bits counter



See in FT-UNSHADES2 the results

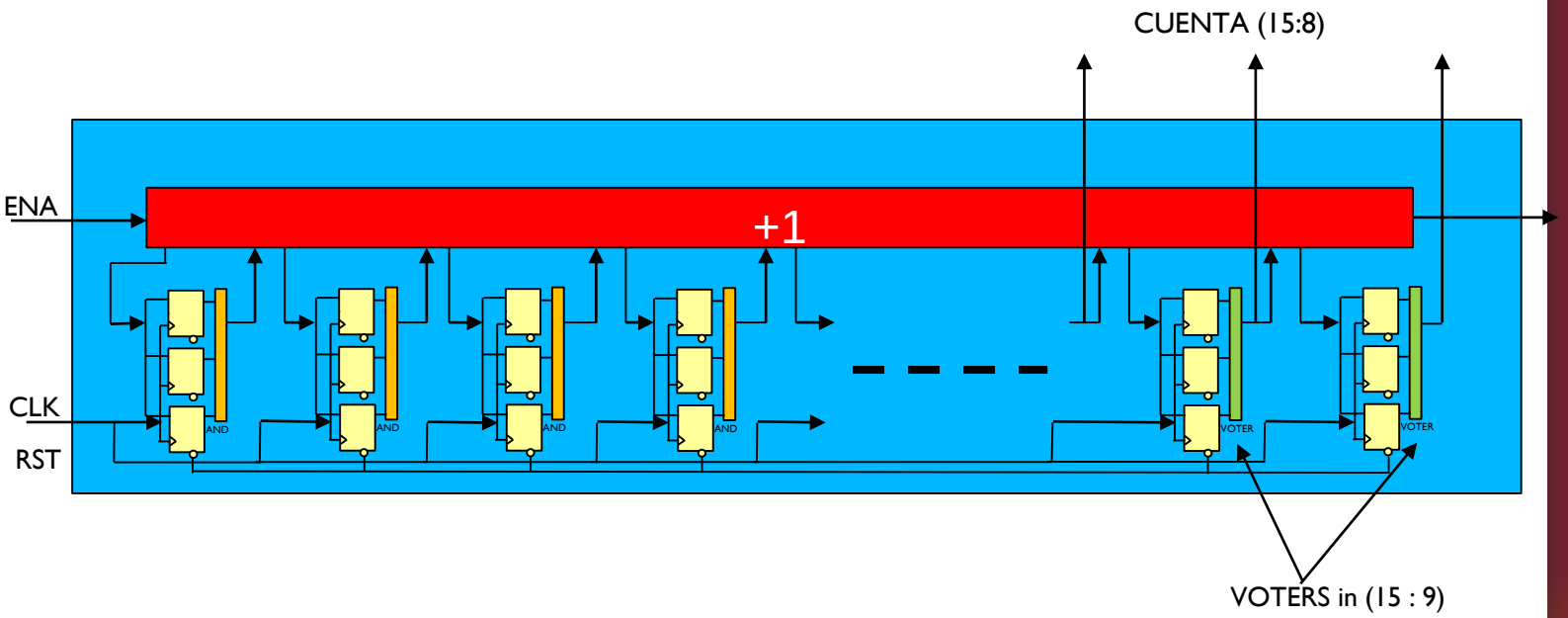
Why all the faults are visible?



See in FT-UNSHADES2 the results

Can we preserve the redundancies during the synthesis?

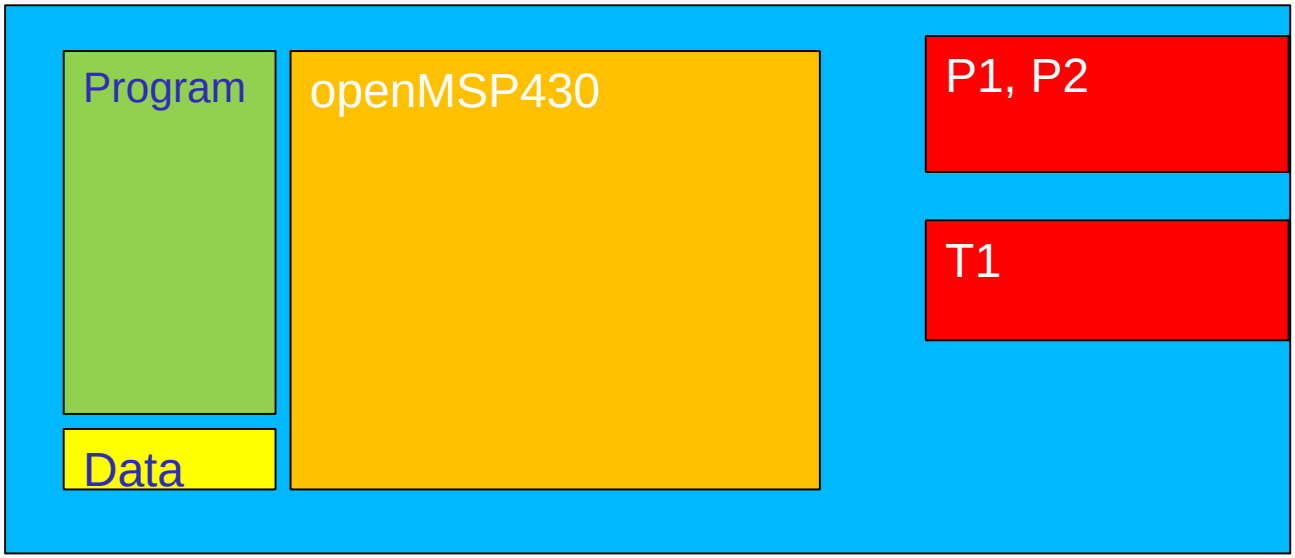
5.2 Half-protected 16bits counter



See in FT-UNSHADES2 the results
Can we see the propagation of faults?

5.3 openMSP430 with peripherals

- 16 bits microprocessor
- Cycle accurate with commercial MSP430
- Written in Verilog with "open style"



See in FT-UNSHADES2 the injection results
Can we see the reset problem?

5.4 openmsp430 selective injection

- Same design and application
- Selective injection

See in FT-UNSHADES2 the injection results
Can we see the reset problem?
Can we say anything about the current application?

Part II: Fault injection in SRAM-FPGAs

- Antifuse FPGAs are treated as ASICs.
- We want to study the design to be used in THAT SRAM-FPGA
- Currently the target is XC5VFX70T

Open questions:

1. Other device of the same family
2. Other family of the same vendor

- SRAM FPGAs related injection platform
- In spite of a 100% occupation, only a small amount of resources are related to the design.
- Particle hits to unrelated resources do not affect to the design behavior
- Particle hits to related parts can affect to the design behavior

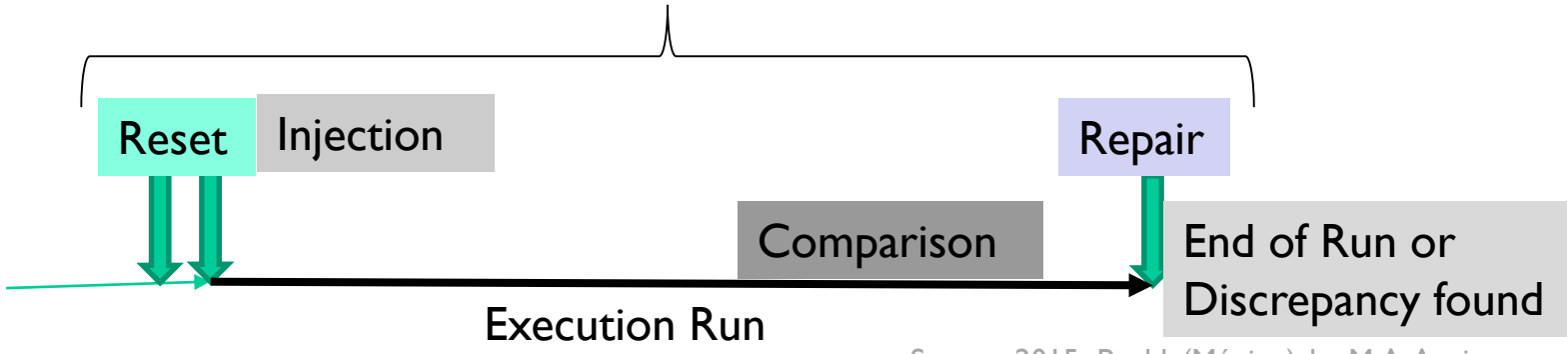
FT-UNSHADES2 approach

- Reuse the FT-UNSHADES2 hardware platform
- The principles:

A fault on one configuration bit will not propagate to any other confguration bit (Is it totally true?)

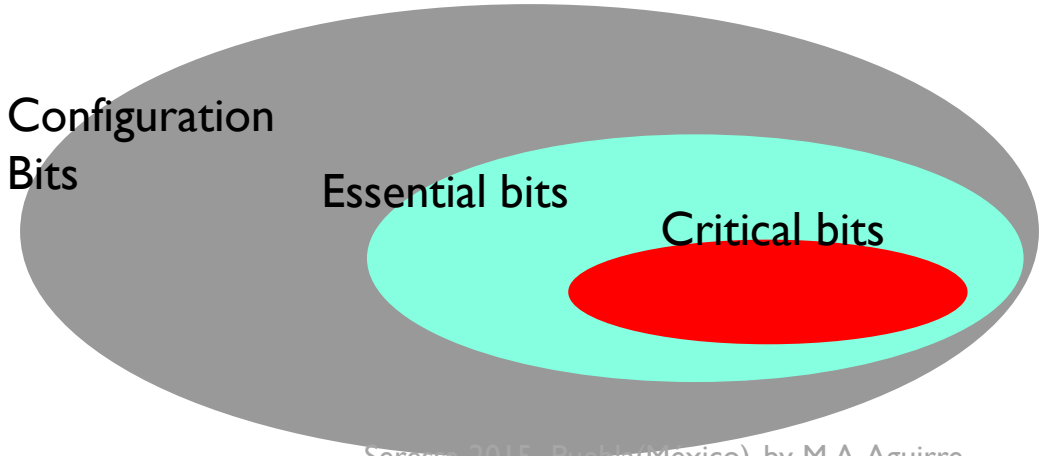
A fault configuration bit can propagate to the circuit logic

- Make the DYNAMICAL injection
- Technique: inject and repair (fast scrubbing)



Essential bits

- *Essential bits are defined here as those bits associated with the circuitry of the design, and are a subset of the device configuration bits.*
- *If an essential bit is upset, it changes the design circuitry. However, the upset might not affect the function of the design.*
- They are calculated from BitGen utility (v 13.4)
- From the essential bits file we calculate the sentitive bits map



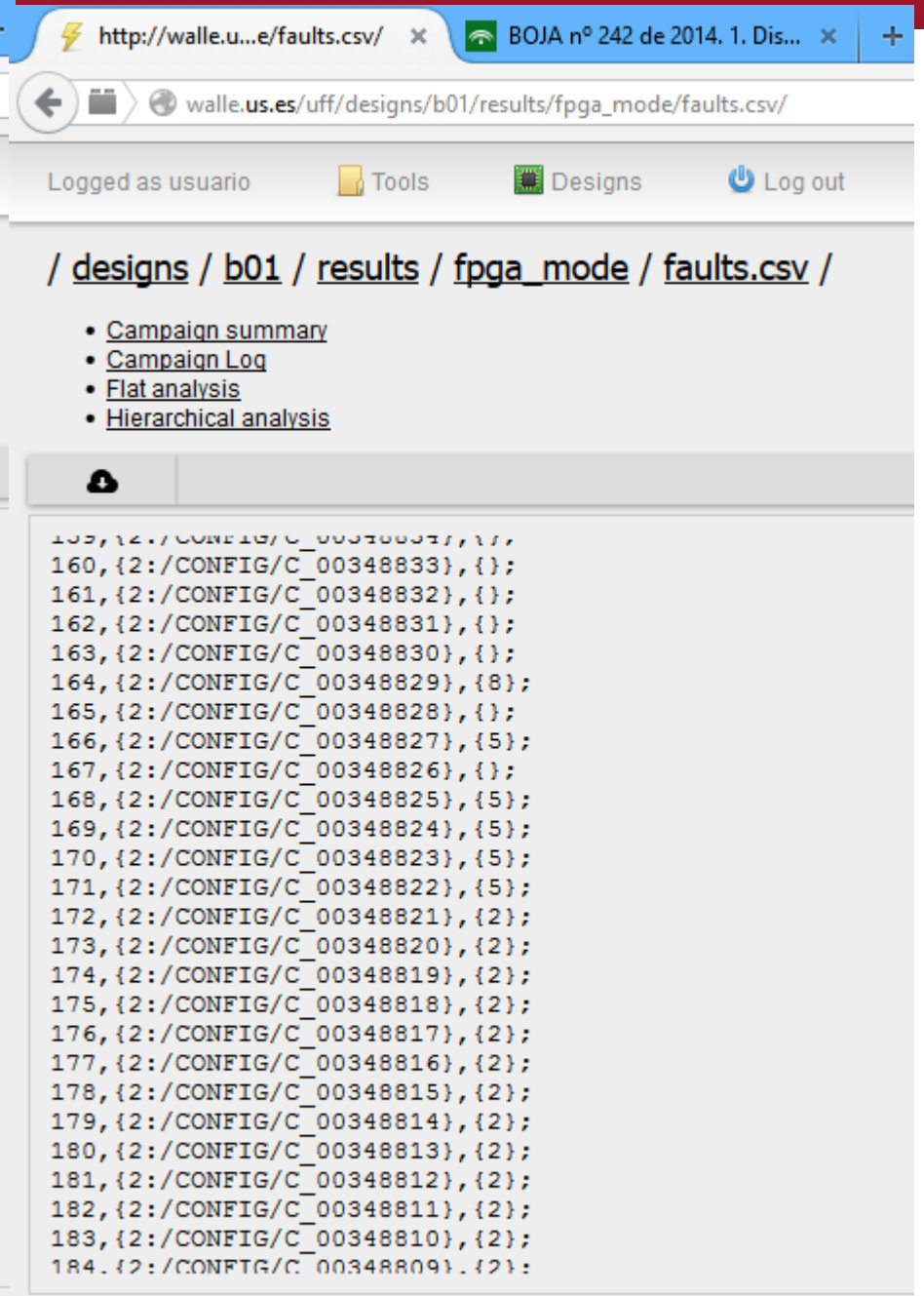
Possible fault classification

- **Unrelated bits**: They do not affect to the user circuit
- **Essential bits**:
  - **Electrical faults**: Affect to capacities and resistivity of wires
  - **Functional faults** (critical): Affect to functional structures
  - **"Destroy" faults**: Affect to other configuration bits. Cannot be repaired.
- **Content faults**: Affect to potentially writeable structures (LUT, UR, BRAM…)
- **Structural faults**: Affect to elements of the basic configuration of the FPGA.

FPGA mode

- Campaings over configuration bits
- ESSENTIAL bits campaings
- Discrimination between critical and electrical faults
- "Destroyer" configuration bits


http://walle.u...e/faults.csv/
BOJA nº 242 de 2014. 1. Dis...
walle.us.es/uff/designs/b01/results/fpga_mode/faults.csv/
Logged as usuario
Tools
Designs
Log out
/ designs / b01 / results / fpga_mode / faults.csv /
Campaign summary
Campaign Log
Flat analysis
Hierarchical analysis
22,{2:/CONFIG/C_00355519},{10};
23,{2:/CONFIG/C_00355518},{7};
24,{2:/CONFIG/C_00355517},{};
25,{2:/CONFIG/C_00355516},{};
26,{2:/CONFIG/C_00355515},{};
27,{2:/CONFIG/C_00355514},{};
28,{2:/CONFIG/C_00355513},{};
29,{2:/CONFIG/C_00355512},{};
30,{2:/CONFIG/C_00355511},{16};
31,{2:/CONFIG/C_00355510},{29};
32,{2:/CONFIG/C_00355509},{};
33,{2:/CONFIG/C_00355508},{76};
34,{2:/CONFIG/C_00355507},{};
35,{2:/CONFIG/C_00355506},{};
36,{2:/CONFIG/C_00355505},{};
37,{2:/CONFIG/C_00355504},{6};
38,{2:/CONFIG/C_00355503},{};
39,{2:/CONFIG/C_00355502},{};
40,{2:/CONFIG/C_00355501},{};
41,{2:/CONFIG/C_00355500},{48};
42,{2:/CONFIG/C_00355019},{};
43,{2:/CONFIG/C_00355018},{5};
44,{2:/CONFIG/C_00355017},{};
45,{2:/CONFIG/C_00355016},{};
46,{2:/CONFIG/C_00355015},{};


http://walle.u...e/faults.csv/
BOJA nº 242 de 2014. 1. Dis...
walle.us.es/uff/designs/b01/results/fpga_mode/faults.csv/
Logged as usuario
Tools
Designs
Log out
/ designs / b01 / results / fpga_mode / faults.csv /
Campaign summary
Campaign Log
Flat analysis
Hierarchical analysis
160,{2:/CONFIG/C_00348833},{};
161,{2:/CONFIG/C_00348832},{};
162,{2:/CONFIG/C_00348831},{};
163,{2:/CONFIG/C_00348830},{};
164,{2:/CONFIG/C_00348829},{8};
165,{2:/CONFIG/C_00348828},{};
166,{2:/CONFIG/C_00348827},{5};
167,{2:/CONFIG/C_00348826},{};
168,{2:/CONFIG/C_00348825},{5};
169,{2:/CONFIG/C_00348824},{5};
170,{2:/CONFIG/C_00348823},{5};
171,{2:/CONFIG/C_00348822},{5};
172,{2:/CONFIG/C_00348821},{2};
173,{2:/CONFIG/C_00348820},{2};
174,{2:/CONFIG/C_00348819},{2};
175,{2:/CONFIG/C_00348818},{2};
176,{2:/CONFIG/C_00348817},{2};
177,{2:/CONFIG/C_00348816},{2};
178,{2:/CONFIG/C_00348815},{2};
179,{2:/CONFIG/C_00348814},{2};
180,{2:/CONFIG/C_00348813},{2};
181,{2:/CONFIG/C_00348812},{2};
182,{2:/CONFIG/C_00348811},{2};
183,{2:/CONFIG/C_00348810},{2};

Analysis of individual faults

There are mechanisms that can do:

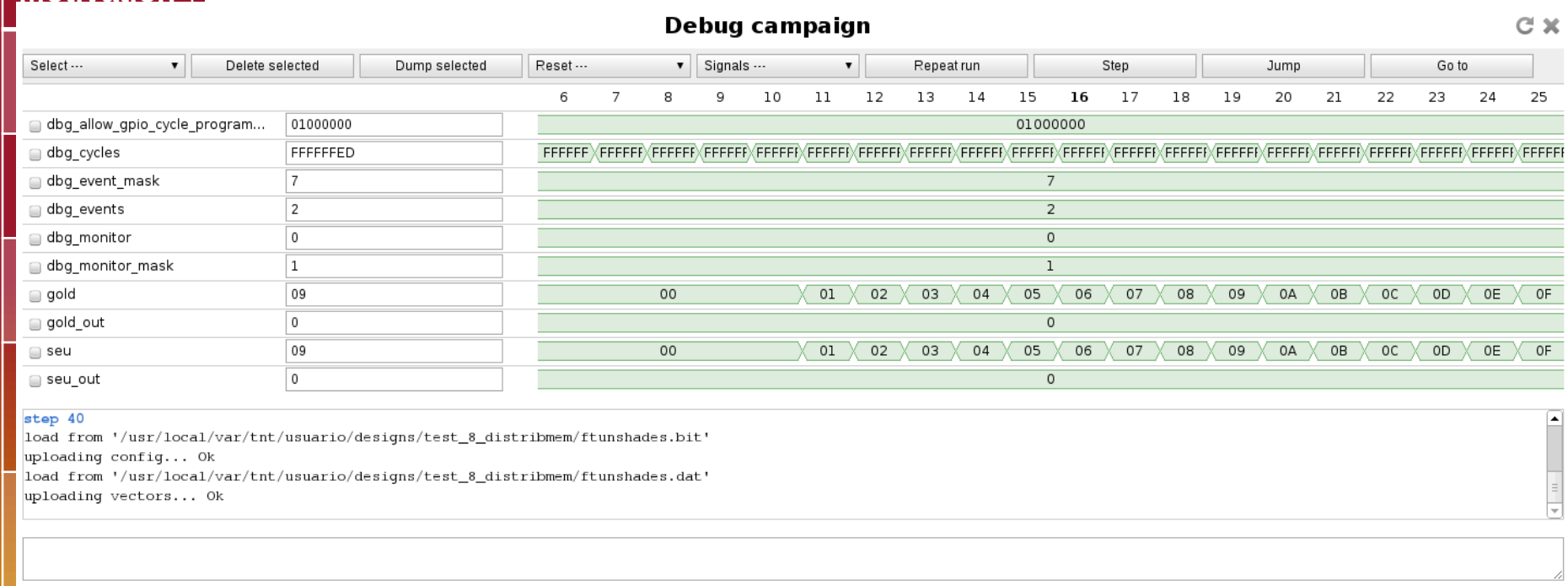
1. Step by step propagation
2. Inspect internal values of the memory elements at any clock cycle

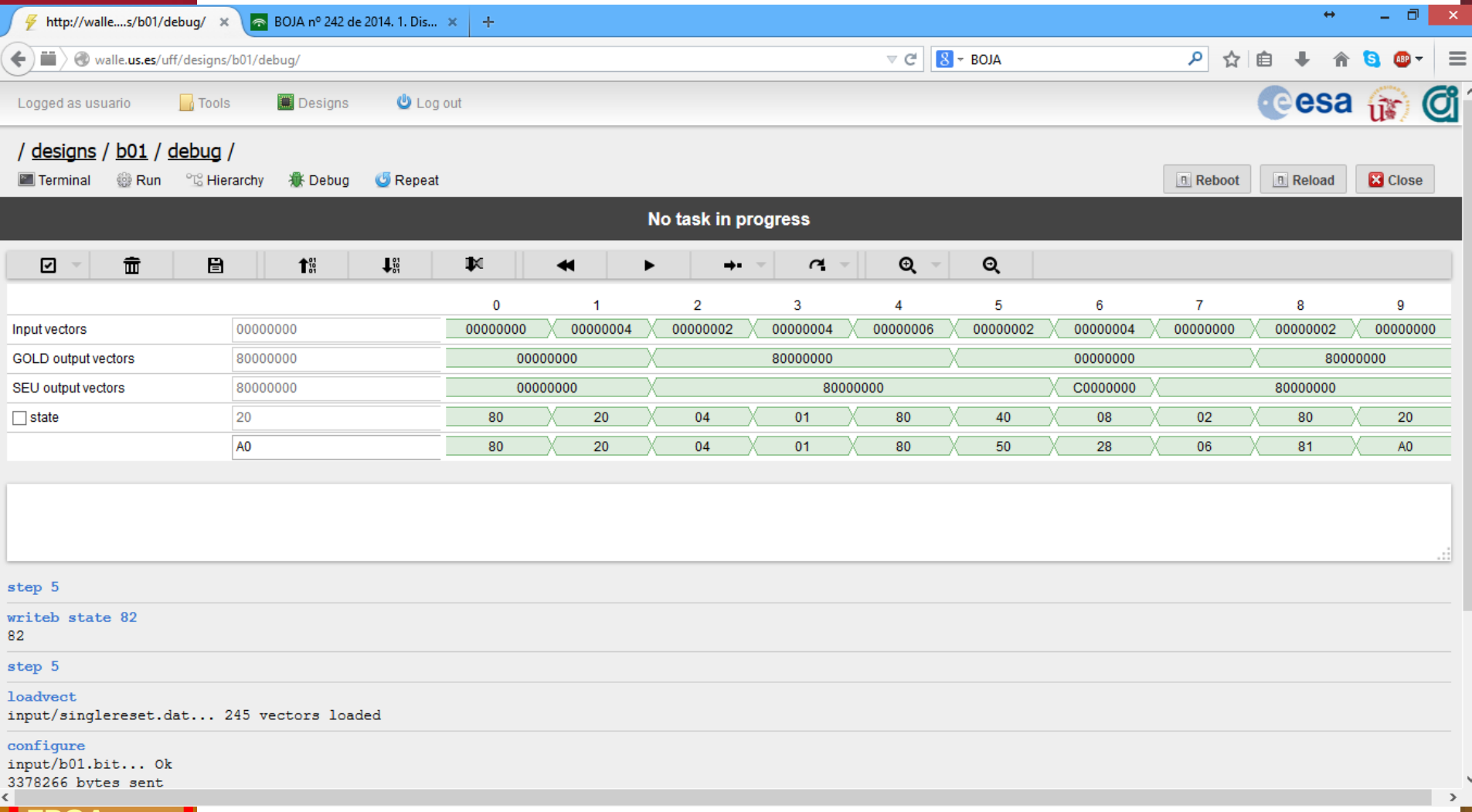
Observe in detail:

- How the faults are propagated through the netlist.
- Latent faults

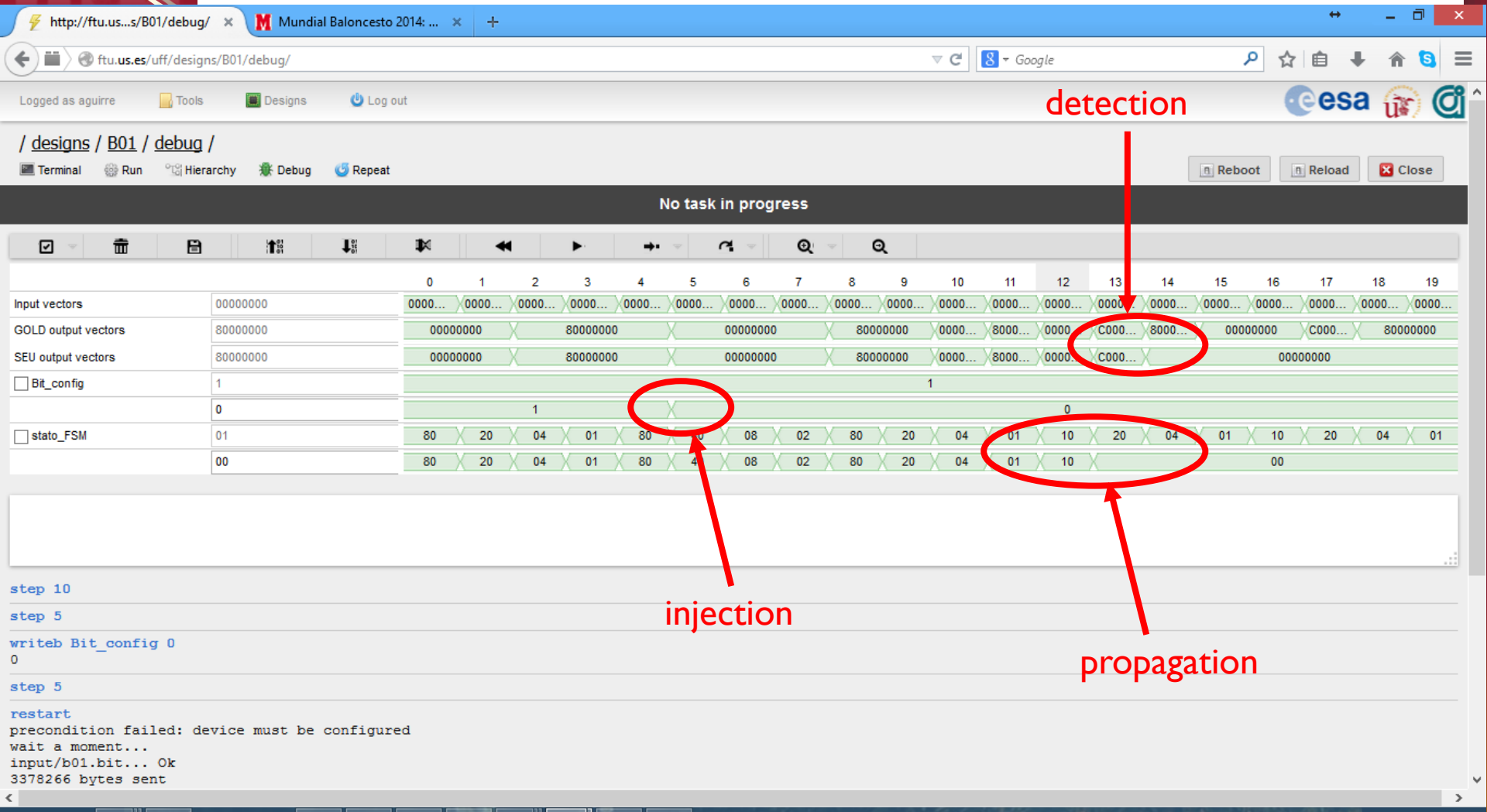
Analysis, fault propagation

- Memory elements are identified by their respective hierarchical path -> check if internal values are assigned properly
- Memory elements are internally read related to clock cycles -> a waveform is created showing how the injected fault is propagated.




http://walle....s/b01/debug/
BOJA nº 242 de 2014. 1. Dis...
walle.us.es/uff/designs/b01/debug/
BOJA
Logged as usuario
Tools
Designs
Log out
esa
/ designs / b01 / debug /
Terminal
Run
Hierarchy
Debug
Repeat
Reboot
Reload
Close
No task in progress
0 1 2 3 4 5 6 7 8 9
Input vectors 00000000 00000000 00000004 00000002 00000004 00000006 00000002 00000004 00000000 00000002 00000000
GOLD output vectors 80000000 00000000 80000000 00000000 80000000
SEU output vectors 80000000 00000000 80000000 C0000000 80000000
state 20 80 20 04 01 80 40 08 02 80 20
A0 80 20 04 01 80 50 28 06 81 A0
step 5
writeb state 82
82
step 5
loadvect
input/singlereset.dat... 245 vectors loaded
configure
input/b01.bit... Ok
3378266 bytes sent

FPGA mode: Fault in configuration propagated to the outputs

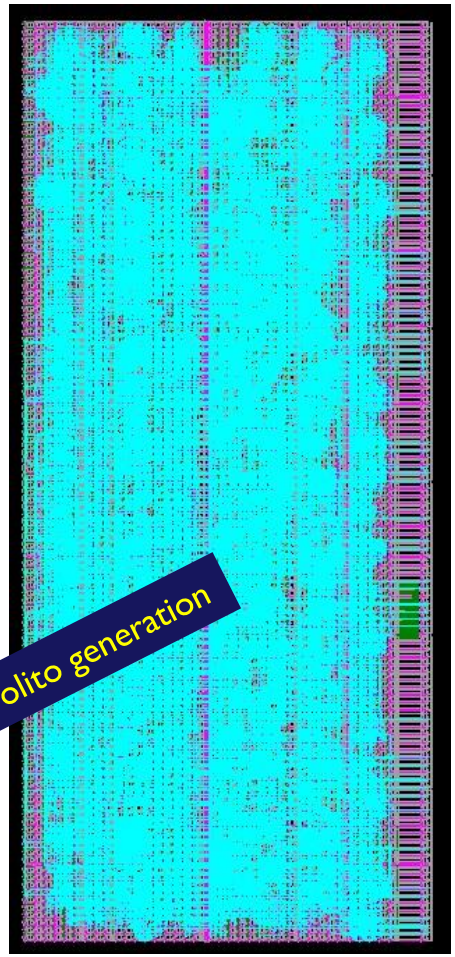


Mitigation

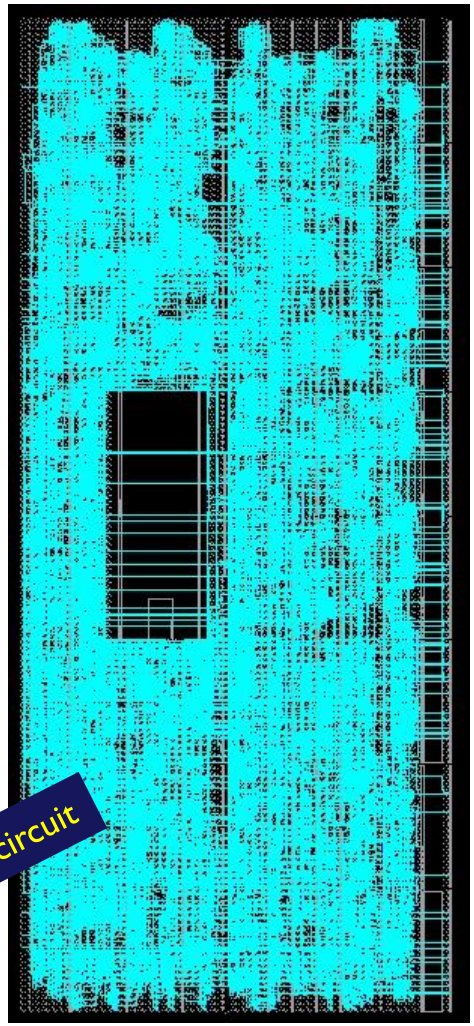
- Scrubbing: “Soft” overwriting of configurations
- Clock rate far from the theoretical limit
- TMR, EDACs,…
- TMR with rules for clock domains
- See Melanie, Fernanda, Stephen, Luca… recomendations in SERESSA 2015

Comparison between real testing and FIF

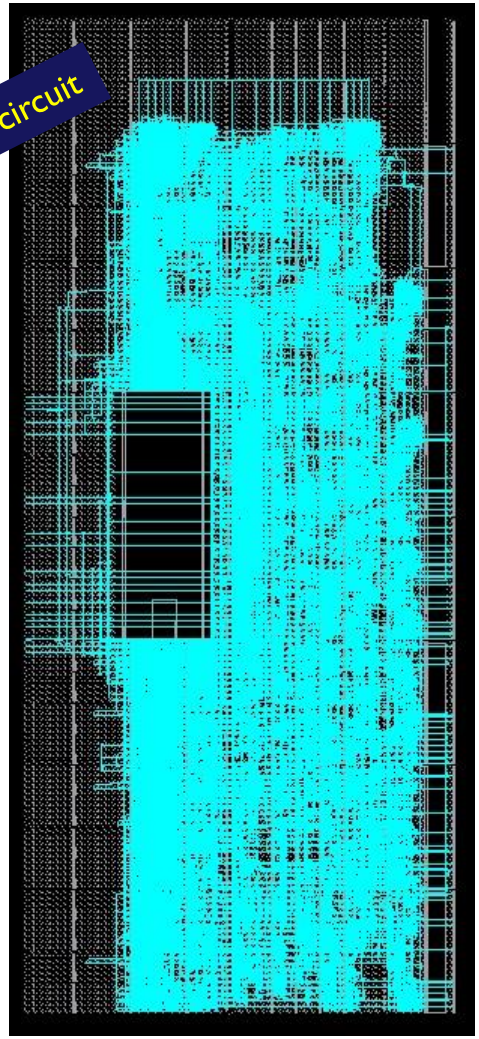
Design as was imported from Politecnico.
B13, x30 replicas, XTMRed, LX50T



Design implemented in FX70T with same IO constraints than Politecnico.
B13, x30 replicas, XTMRed, FX70T

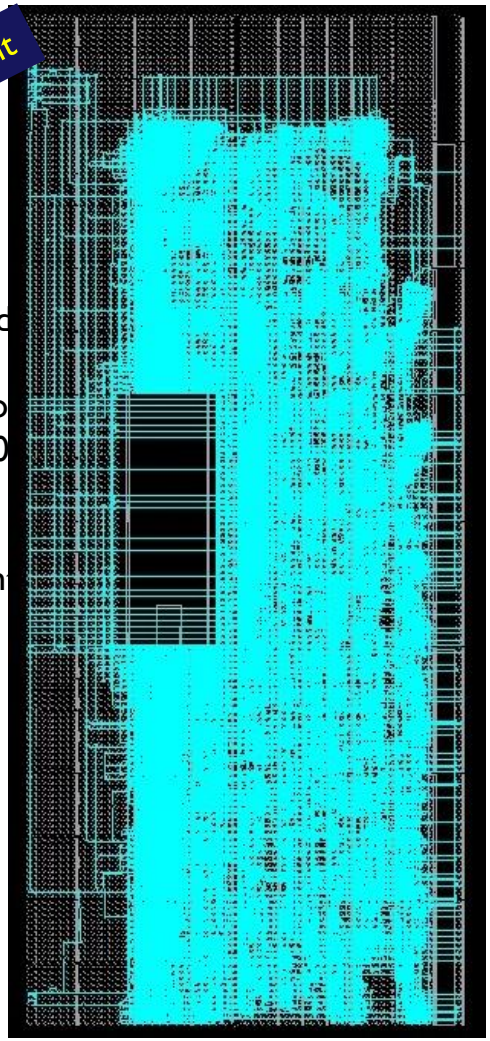


Polito circuit

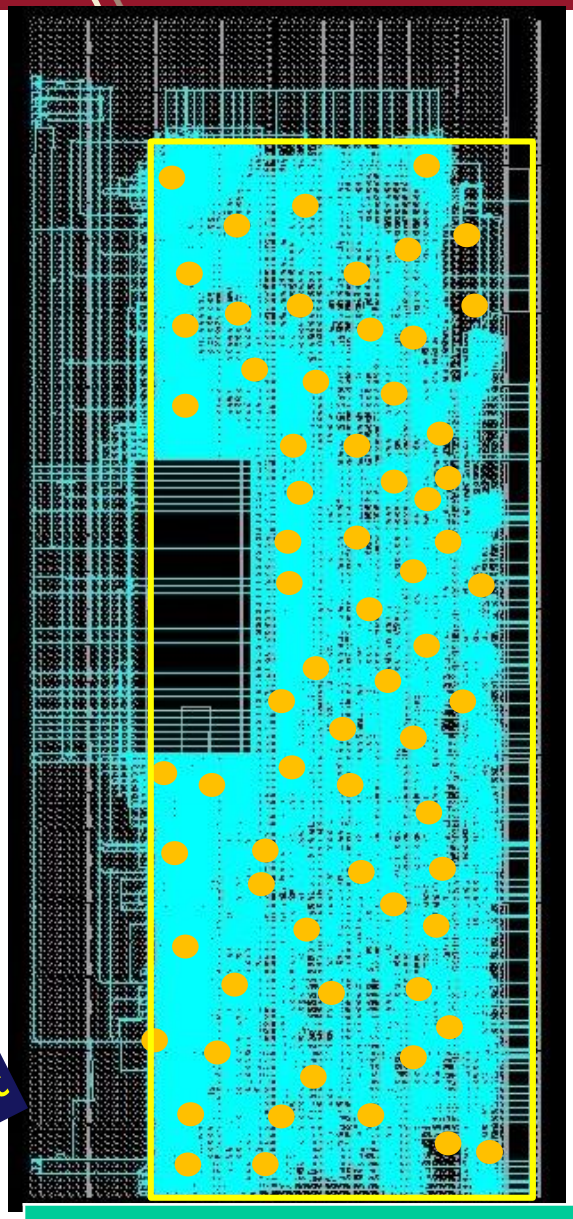


Design implemented in FX70T with same IOs than Politecnico. B13, x30 replicas, XTMRed, FX70T, but introduced an area constraint equivalent to LX50T

FTU2 circuit



Design implemented in FX70T with IOs manually changed to fit in FTU2. B13, x30 replicas, XTMRed, FX70T, an area constraint equivalent to LX50T, and IOs for FTU2.



FTU2 circuit

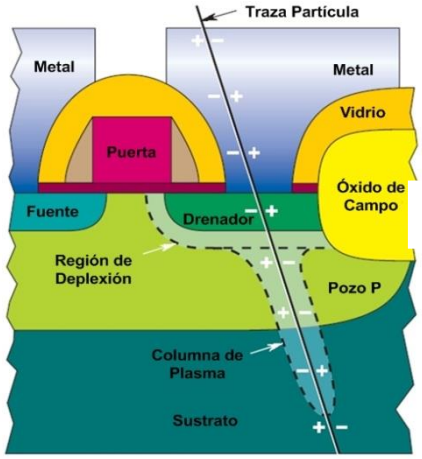
Each FPGA model is different
Careful constraints can make it quite
Similar to the one tested
LX50 vs LX70
Same circuit
Same technology
Different size
Different layout
Different I/Os

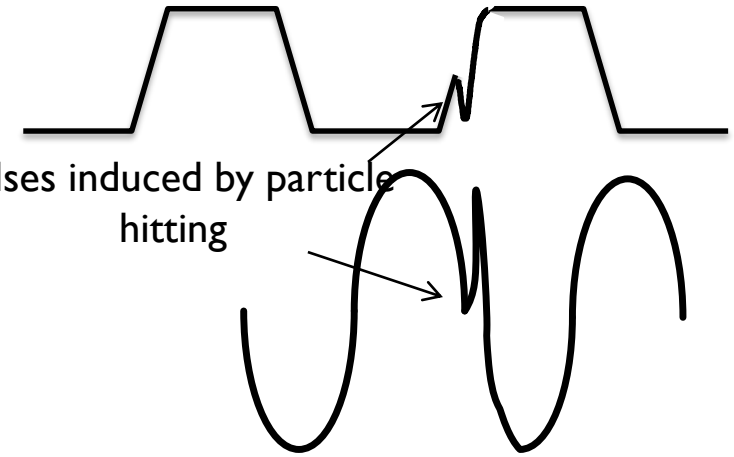
Design and conditions	Device	Essential Bits	Total Essential Bits
B13_X30_XTMR (Polito circuit)	FX70T	1.912.920	18.936.096
B13_X30_plain (FTU2's circuit)	FX70T	1.922.272	18.936.096

- FAULT INJECTION for FPGA is a still open problem.
- There is a high necessity of using SRAM FPGAs is space systems
- The industry need solutions

Analog fault injection

- Environments for high radiation:
  - Particles of high energy affects to the integrity of electronic systems.





- Technology scale→ most critical effects(SET)
  - Higher frequencies and smaller dimensions→ same effects with less charge
  - Increment in complexity if error nature→ specific analysis in analog domain

Analisis tool: AFTU

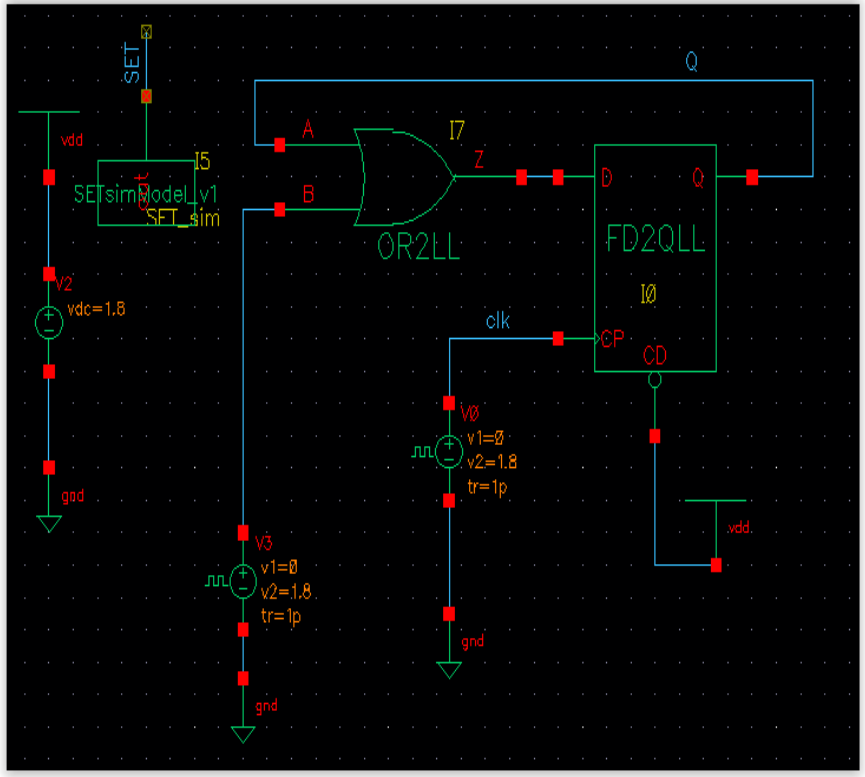
- AFTU* is a software tool for análisis of particle hitting in analog and mixed design
- *¿How does AFTU work?*

*F. Márquez, F. Muñoz, L. Sanz, F.R. Palomo, and M. A. Aguirre. "*AFTU, an Analog Single Event Effects automatic analysis tool*" 5th International Workshop on Analogue and Mixed-Signal Integrated Circuits for Space Applications (AMICSA '14), Ginebra, July 2014

* **A**nalog **FTU**, desarrollada mediante proyecto FT-UNSHADES2 (**F**ault **T**olerant **UN**iversity of **S**eville **HA**rdware **DE**bbugging **S**ystem) financiado por ESA

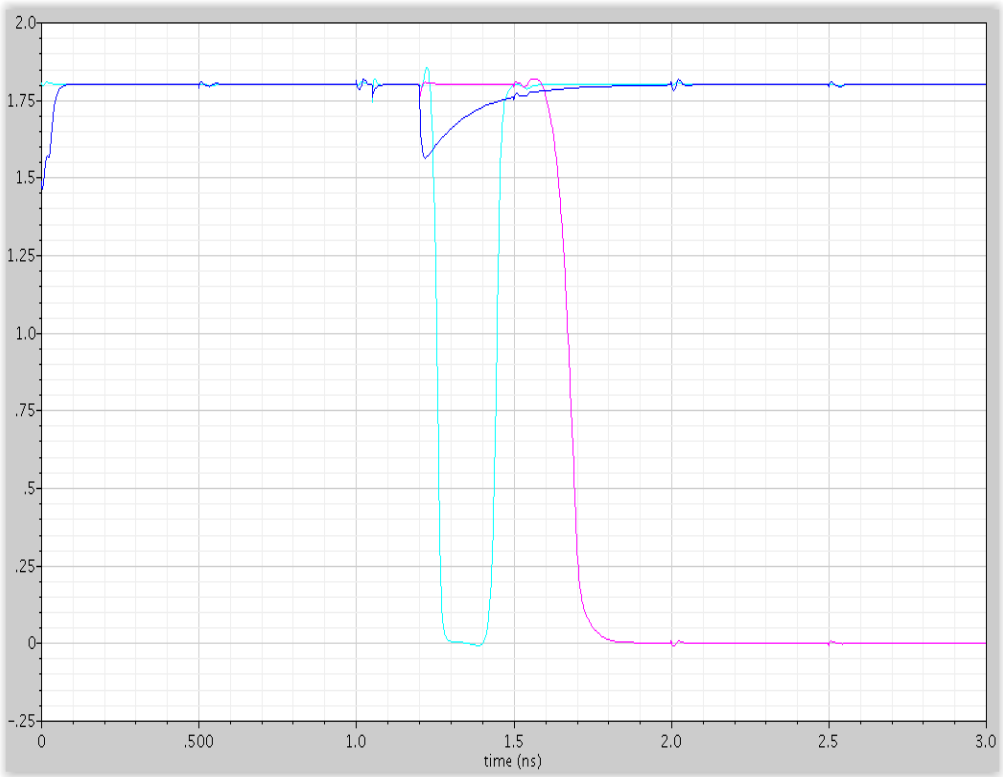
AFTU

- AFTU starts from a test-bench SPECTRE for the designer:



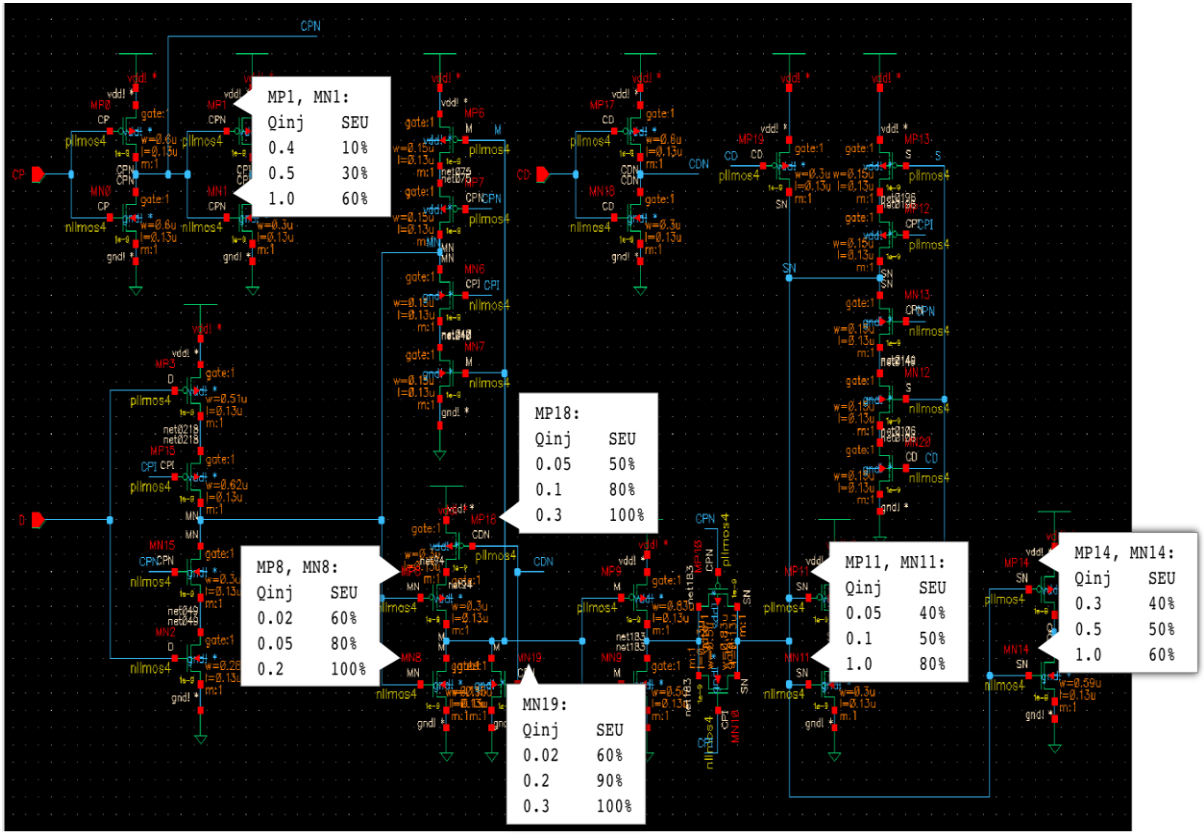
AFTU

- Emulates the radiation conditions:



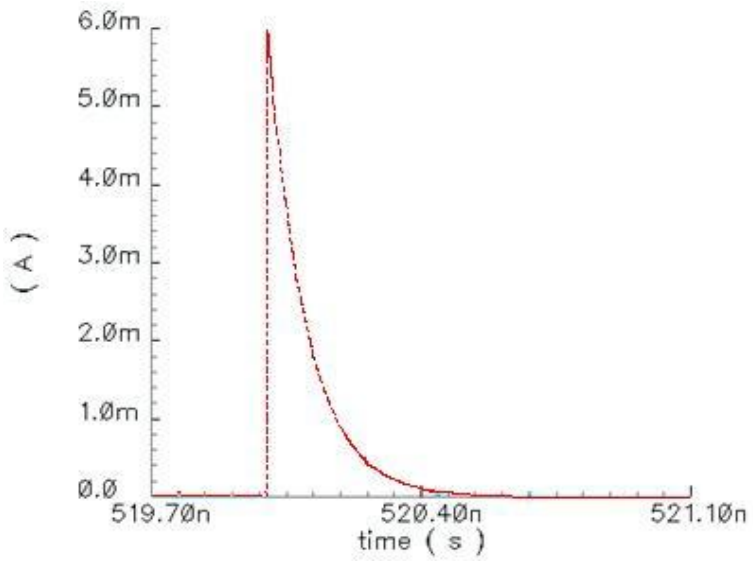
AFTU

- Explores the vulnerabilities of the circuit under test:



AFTU: impact model

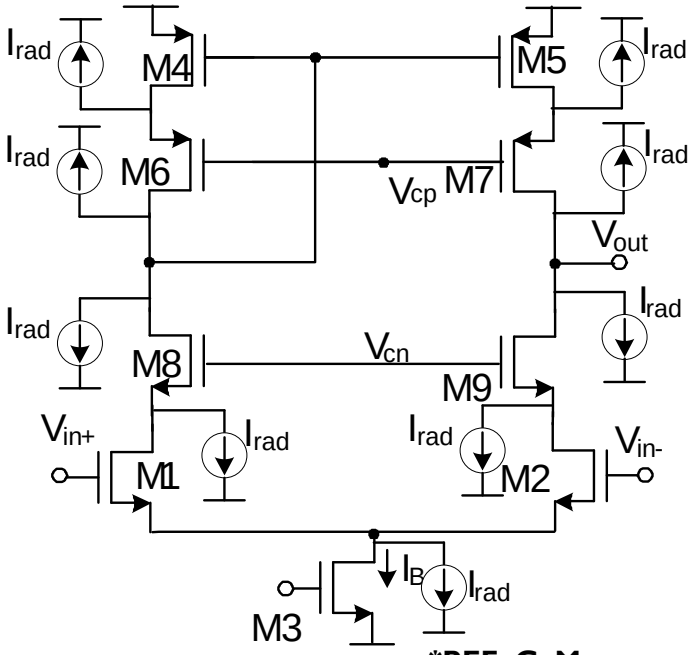
- Model based on charge injection:
  - Use of configurable parameters (AHDL/VerilogA)



$$I_{rad} = \frac{Q_c}{\tau_d - \tau_r}\left(e^{-\frac{\tau}{\tau_d}} - e^{-\frac{\tau}{\tau_r}}\right)$$

$$Q_c = \frac{\rho \cdot LET \cdot d}{3.6}$$

***REF: G. Messenger,** "Collection of Charge on junction nodes from ion tracks", *IEEE Transactions on nuclear science*, vol.29, n° 6, Dec. 1982



$$I_{rad} = \frac{Q_c}{\tau_d - \tau_r}\left(e^{-\frac{\tau}{\tau_d}} - e^{-\frac{\tau}{\tau_r}}\right)$$

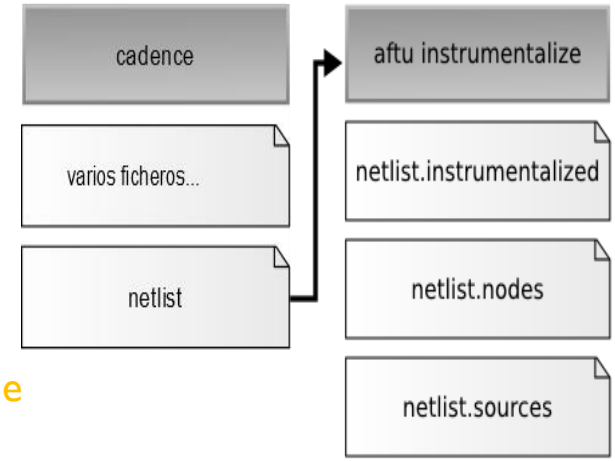
$$Q_c = \frac{\rho \cdot LET \cdot d}{3.6}$$

***REF: G. Messenger,** "Collection of Charge on junction nodes from ion tracks", *IEEE Transactions on nuclear science*, vol.29, n° 6, Dec. 1982

AFTU: tool chain

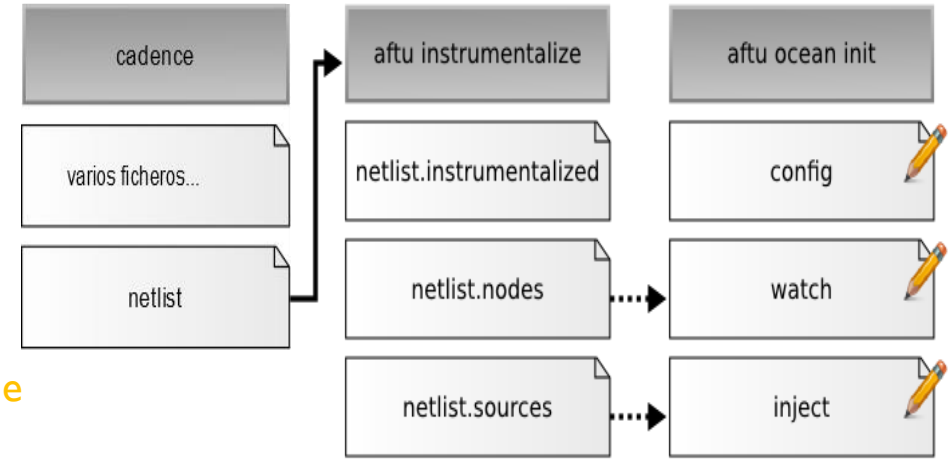
- Before AFTU:
  - Design using CADENCE
  - Simulation with a *test-bench*
  - We need to generate the *netlist*

AFTU: *instrumentalize*



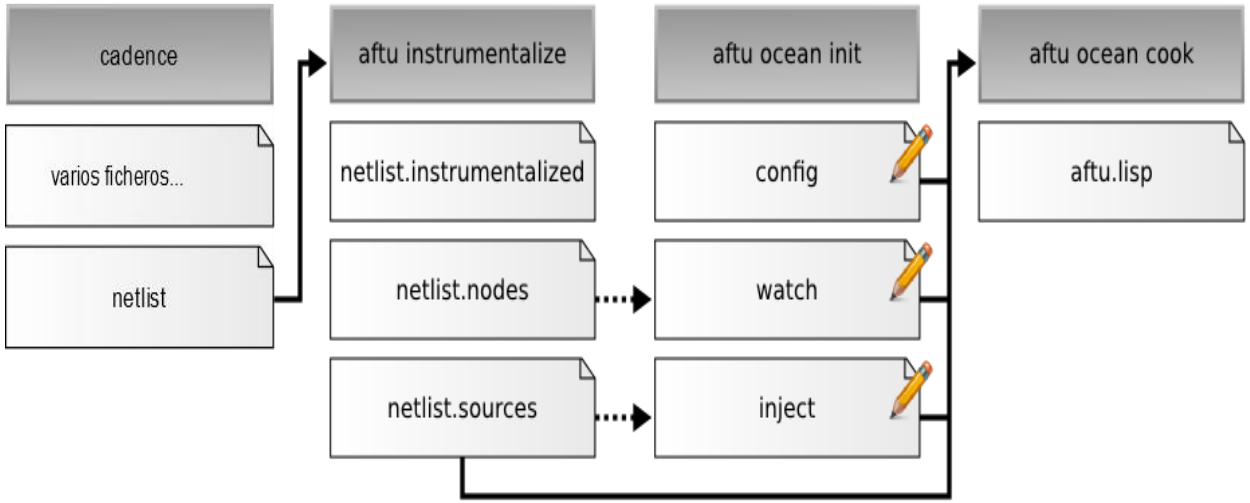
- Using *instrumentalize* a parser is implemented for a simulator based on SPECTRE:
  - The native netlist is substituted by other functionally identical but with the radiation effects emulated
  - *Netlist.nodes* includes all the nodes to be observed
  - *Netlist.sources* included all the nodes to be hitted.

AFTU: *ocean init*



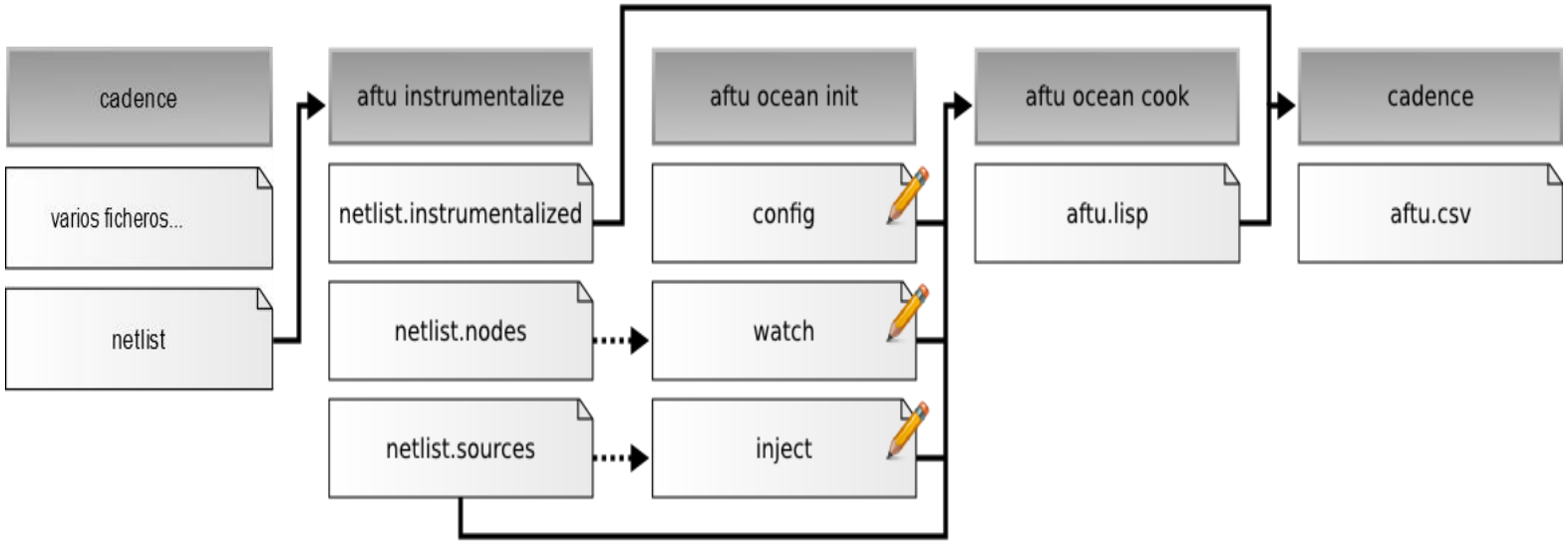
- User can config a AFTU project:
  - ***Config*** includes all the routes that are input to the circuit
  - ***Watch*** defines all the nodes to be monitored.
  - ***Inject*** dine WHAT, WHEN and HOW the injection is going to be.

AFTU: *ocean cook*



- Defines how to analize the results of the campaign

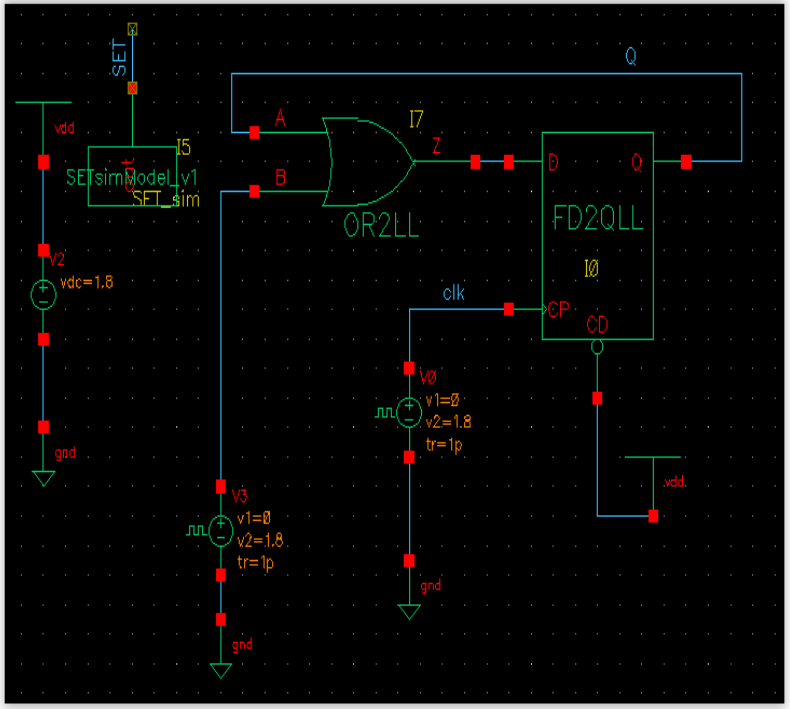
AFTU: test campaign



- Script given by CADENCE.
- results clasified.

AFTU: case study

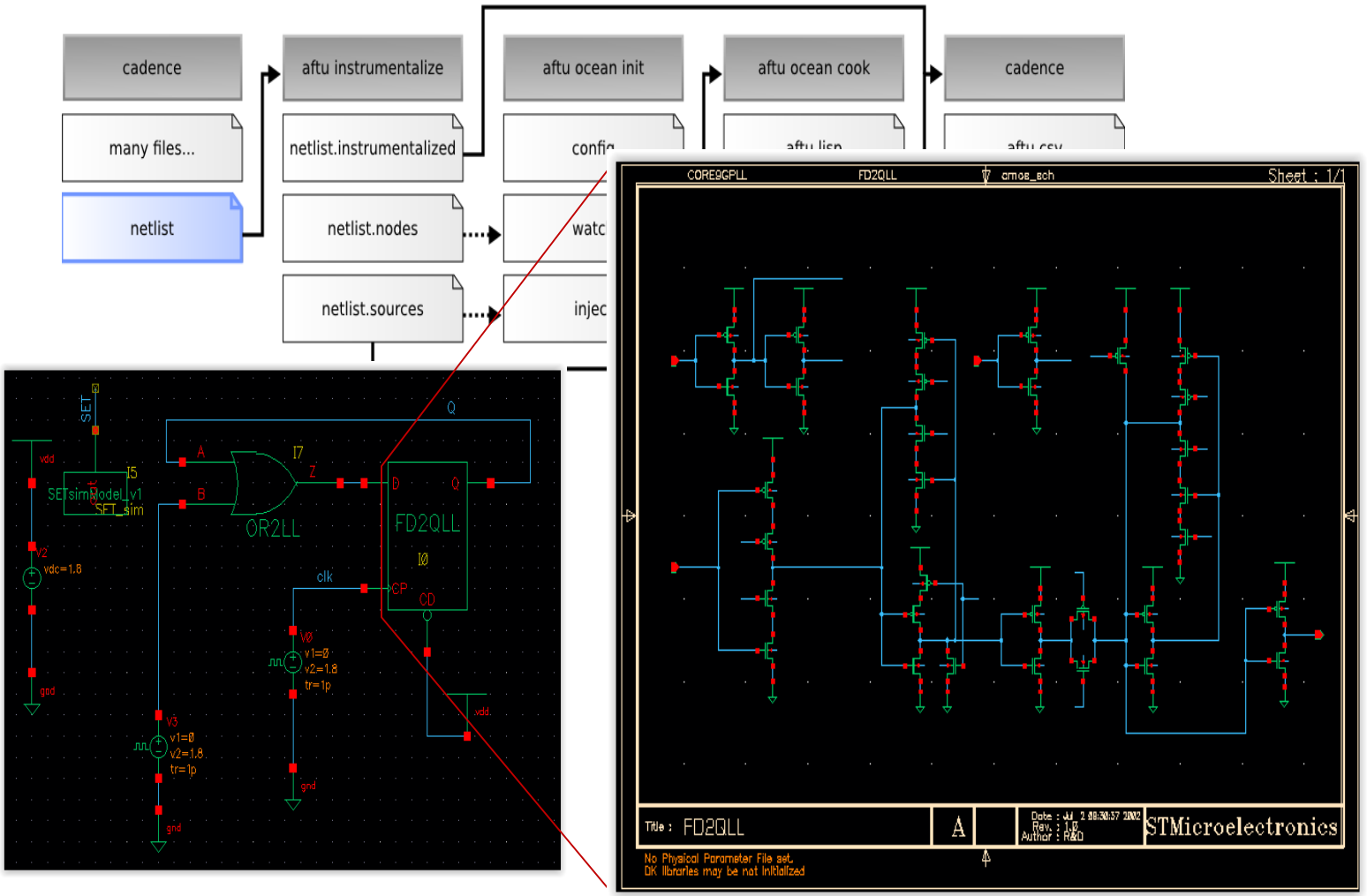
- Latch in 130 nm from ST Microelectronics.



- ¿Which is the critical charge that coult porvoque a SET and its ocurrence probability?

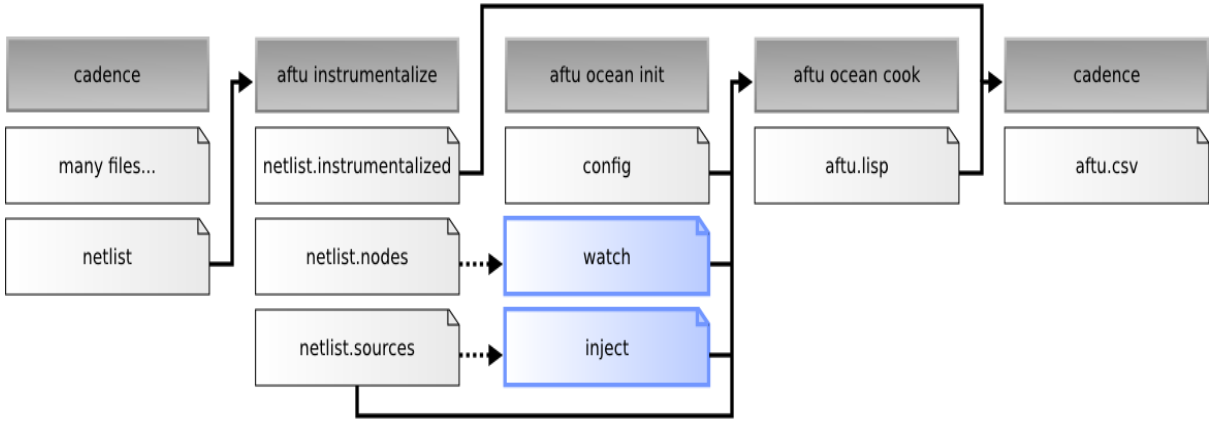
AFTU: case study

- Latch in 130 nm from ST Microelectronics.



AFTU: case study

- Latch in 130 nm from ST Microelectronics.




```
watch Q = /Q :
        threshold = 0.975 ;
```

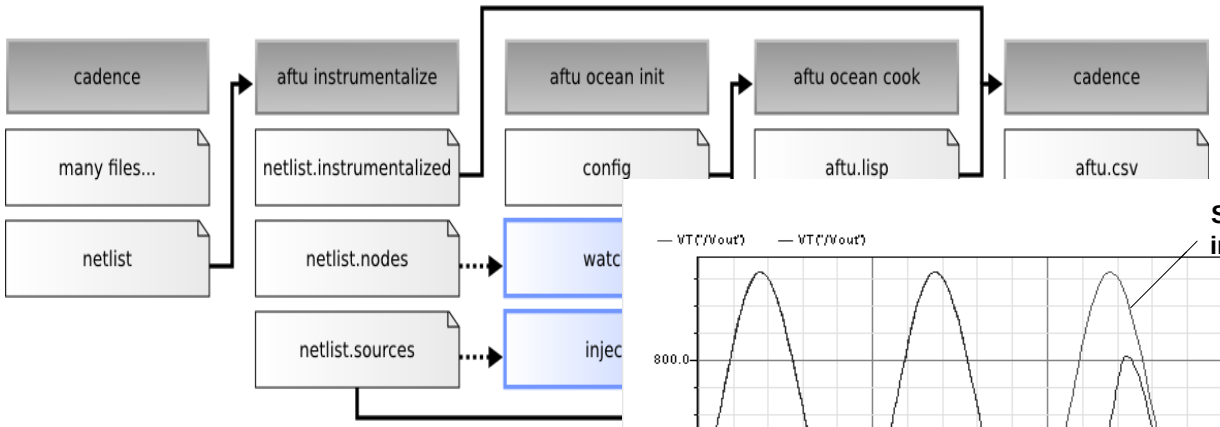
```
inject I0_MP19:
    Q = .025p, .05p, 0.1p, .2p, .5p, .75p, 1p, 1.5p;
    t = 1.0n, 1.1n, 1.2n, 1.3n, 1.4n, 1.5n, 1.6n,
1.7n, 1.8n, 1.9n;

inject I0_MN19:
    Q = .025p, .05p, 0.1p, .2p, .5p, .75p, 1p, 1.5p;
    t = 1.0n: 1.9n: 0.1n;

inject I0_MP18, I0_MN12, I0_MN7:
    Q = .025p, .05p, 0.1p, .2p, .5p, .75p, 1p, 1.5p;
    t = 1.0n, 1.1n, 1.2n, 1.3n, 1.4n, 1.5n, 1.6n,
1.7n, 1.8n, 1.9n;
```

AFTU: case study

- Latch in 130 nm from ST Microelectronics.

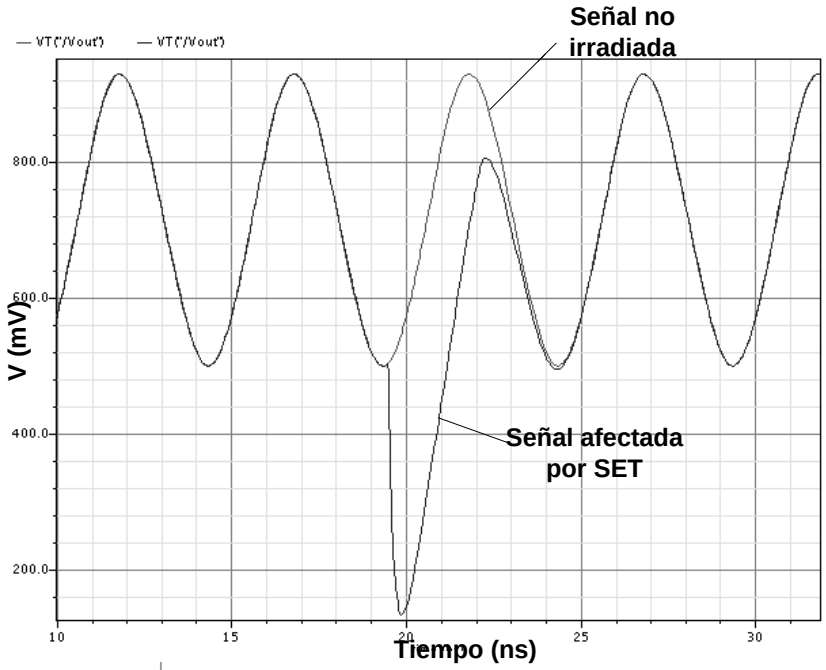



```
watch Q = /Q :
        threshold = 0.975 ;
```

```
inject I0_MP19:
    Q = .025p, .05p, 0.1p, .2p, .5p, .75p,
    t = 1.0n, 1.1n, 1.2n, 1.3n, 1.4n, 1.5n,
    1.7n, 1.8n, 1.9n;

inject I0_MN19:
    Q = .025p, .05p, 0.1p, .2p, .5p, .75p,
    t = 1.0n: 1.9n: 0.1n;

inject I0_MP18, I0_MN12, I0_MN7 :
    Q = .025p, .05p, 0.1p, .2p, .5p, .75p, 1p, 1.5p;
    t = 1.0n, 1.1n, 1.2n, 1.3n, 1.4n, 1.5n, 1.6n,
    1.7n, 1.8n, 1.9n;
```

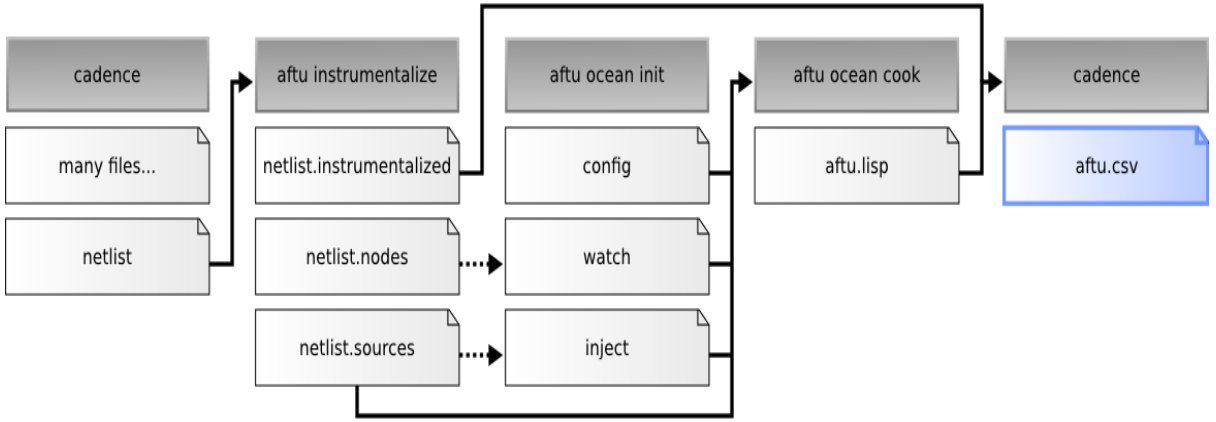


Heurística empleada:

Deviation_recovery

AFTU: case study

- Latch in 130 nm from ST Microelectronics.

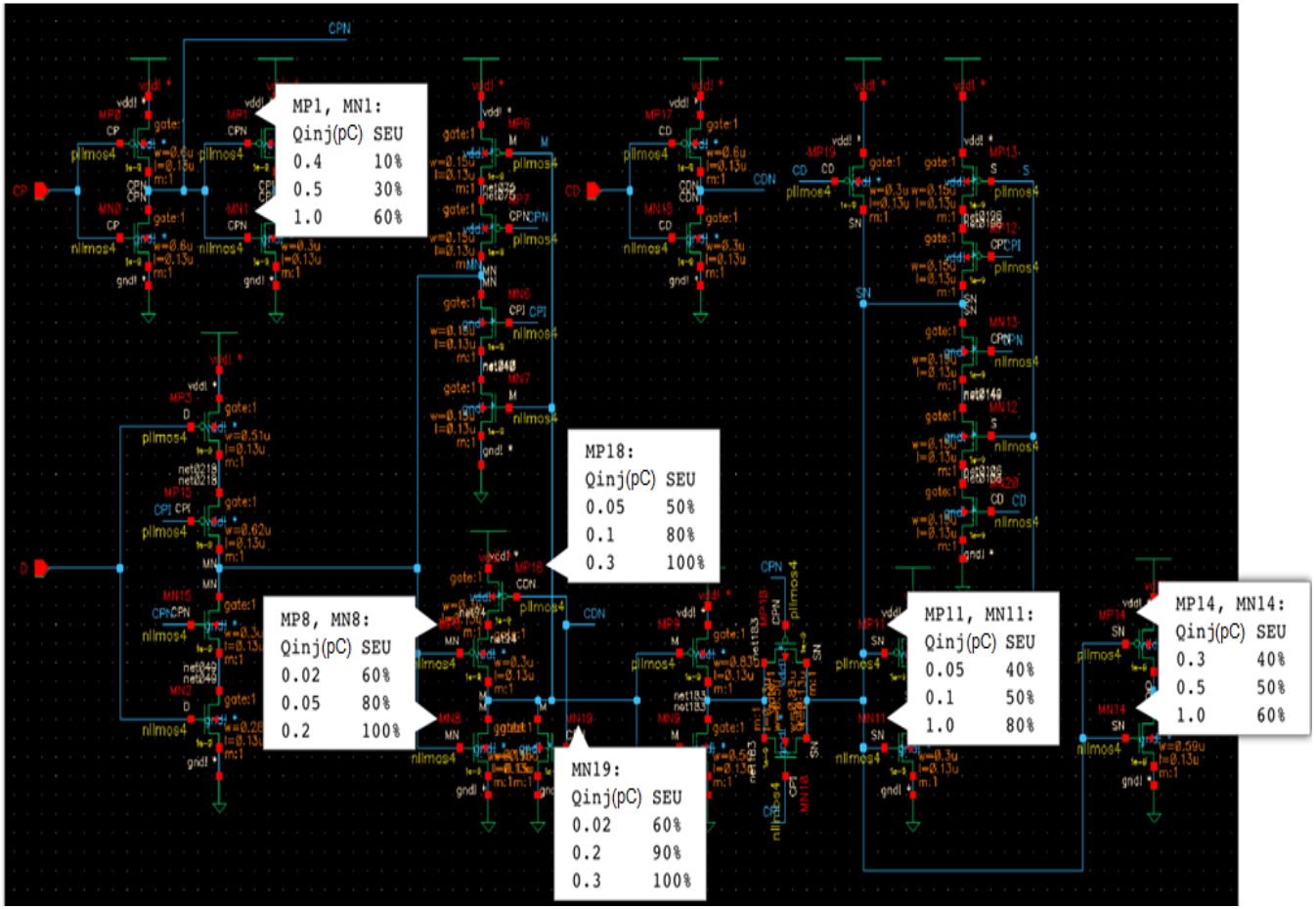


Output	ImpactNode	Qinj	Timp	Trec	Vmax
V_Q	I0_MP18	2.5e-14	1e-09	0.000000	0.001941
V_Q	I0_MP18	2.5e-14	1.1e-09	0.000000	0.00219
...					
V_Q	I0_MP18	2.5e-14	1.9e-09	0.000000	0.003664
V_Q	I0_MP18	5e-14	1e-09	2.000000	1.807421
V_Q	I0_MP18	5e-14	1.1e-09	1.910000	1.807422
V_Q	I0_MP18	5e-14	1.2e-09	1.800000	1.807426
V_Q	I0_MP18	5e-14	1.3e-09	1.700000	1.807384
...					
V_Q	I0_MP18	5e-14	1.8e-09	0.000000	0.519730
V_Q	I0_MP18	5e-14	1.9e-09	1.100000	1.803023
...					

Output	ImpactNode	Qinj	Timp	Trec	Vmax
V_Q	I0_MN11	2.5e-14	1e-09	0.000000	0.006827
...					
V_Q	I0_MN11	2.5e-14	1.9e-09	0.000000	0.016568
V_Q	I0_MN11	5e-14	1e-09	0.000000	0.017084
...					
V_Q	I0_MN11	5e-14	1.3e-09	0.000000	0.005371
V_Q	I0_MN11	5e-14	1.4e-09	1.610000	1.806680
V_Q	I0_MN11	5e-14	1.5e-09	1.500000	1.806814
V_Q	I0_MN11	5e-14	1.6e-09	1.400000	1.805740
V_Q	I0_MN11	5e-14	1.7e-09	1.300000	1.802925
V_Q	I0_MN11	5e-14	1.8e-09	0.240000	1.404223
...					

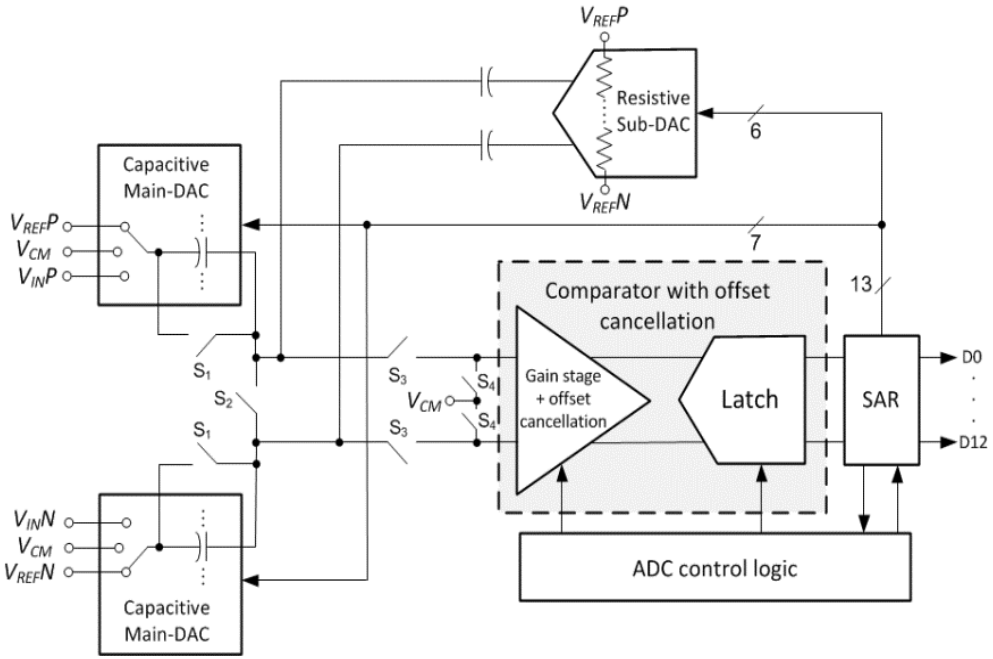
AFTU: case study

- Latch in 130 nm from ST Microelectronics.
- Sensitivity mapSEE:



AFTU: case study

- A/DC SAR using tecnology 250 nm SiGe from IHP.

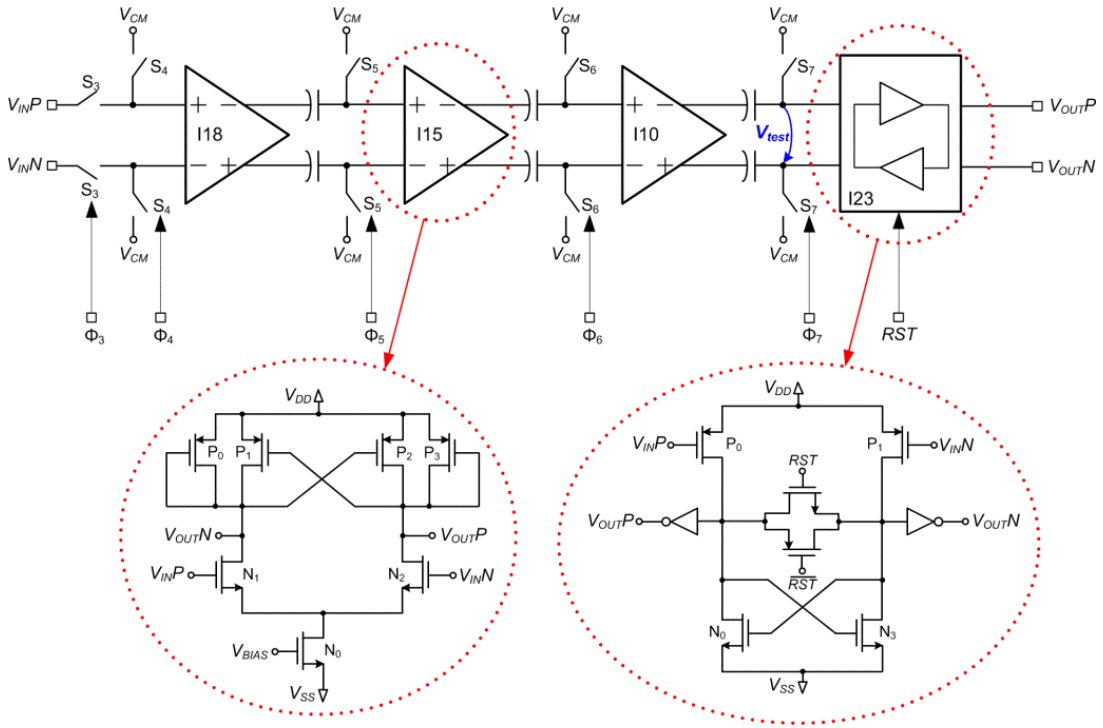


- Critical component: *comparator*

***F. Márquez**, F. Muñoz, F.R. Palomo, L. Sanz, E. López-Morillo, M.A. Aguirre, A. Jiménez
"Automatic Single Event Effects sensitivity analysis of a 13-bit Successive approximation ADC"
Nuclear Science, IEEE Transactions on , vol.62, no.4, pp.1609-1616, Aug. 2015

AFTU: case study

- A/DC SAR using tecnology 250 nm SiGe from IHP.



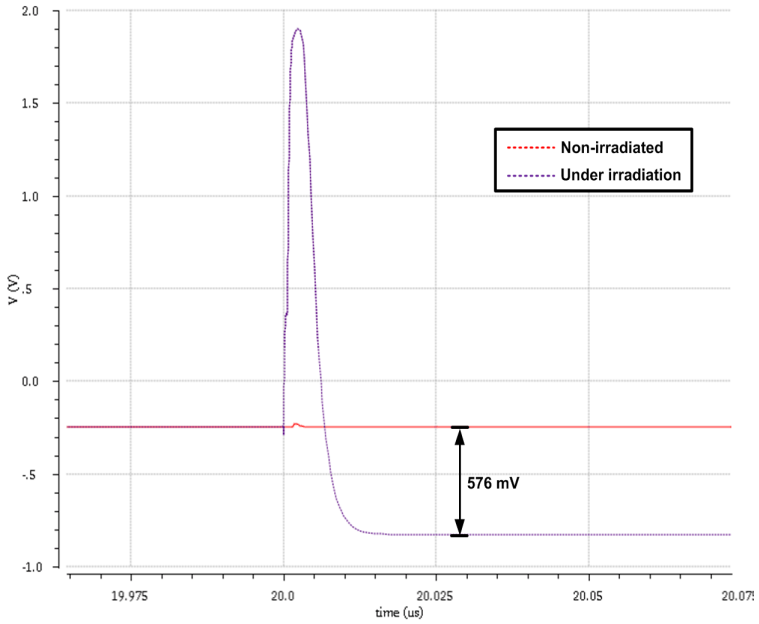
AFTU: case study

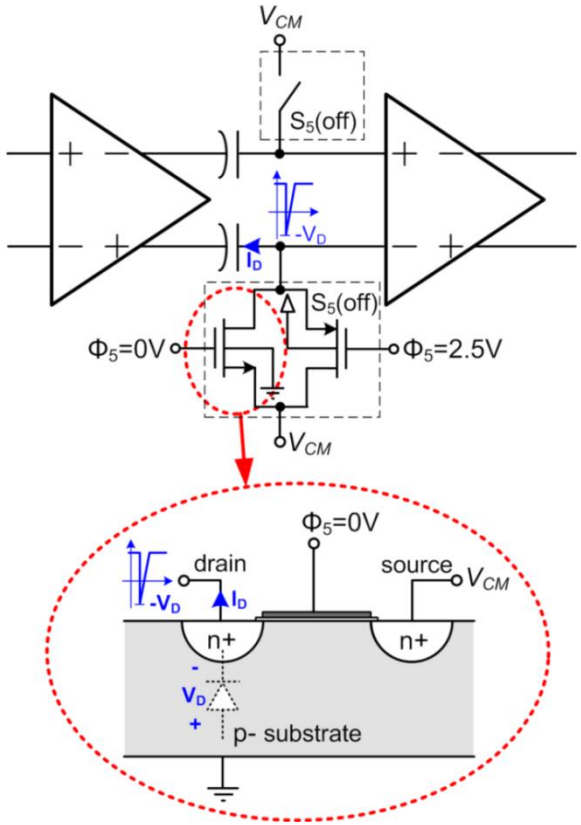
- A/DC SAR using tecnology 250 nm SiGe from IHP.

Transistor	Carga inyectada (C)	Tiempo de impacto (seg)	Trec (nseg)	Vmax(V)
.....	.....	.....	.....	.....
I15_P3	5,00E-13	2,00E-05	6,2	1,322
I15_P3	2,00E-13	2,00E-05	7,2	1,497
I18_N0	5,00E-13	2,00E-05	2	0,067
I18_N0	2,00E-13	2,00E-05	0	0,061
I18_N1	5,00E-13	2,00E-05	15000*	1,667
I18_N1	2,00E-13	2,00E-05	10,4	1,676
I18_N2	5,00E-13	2,00E-05	15000	2,131
I18_N2	2,00E-13	2,00E-05	290,1	2,143
I18_P0	5,00E-13	2,00E-05	15000	1,668
I18_P0	2,00E-13	2,00E-05	10,4	1,676
I18_P1	5,00E-13	2,00E-05	15000	1,668
I18_P1	2,00E-13	2,00E-05	10,4	1,676
I18_P2	5,00E-13	2,00E-05	15000	2,131
I18_P2	2,00E-13	2,00E-05	290,1	2,143
I18_P3	5,00E-13	2,00E-05	15000	2,131
I18_P3	2,00E-13	2,00E-05	290,1	2,143
I23_N0	5,00E-13	2,00E-05	0	0,0015
.....	.....	.....	.....	.....

AFTU: case study

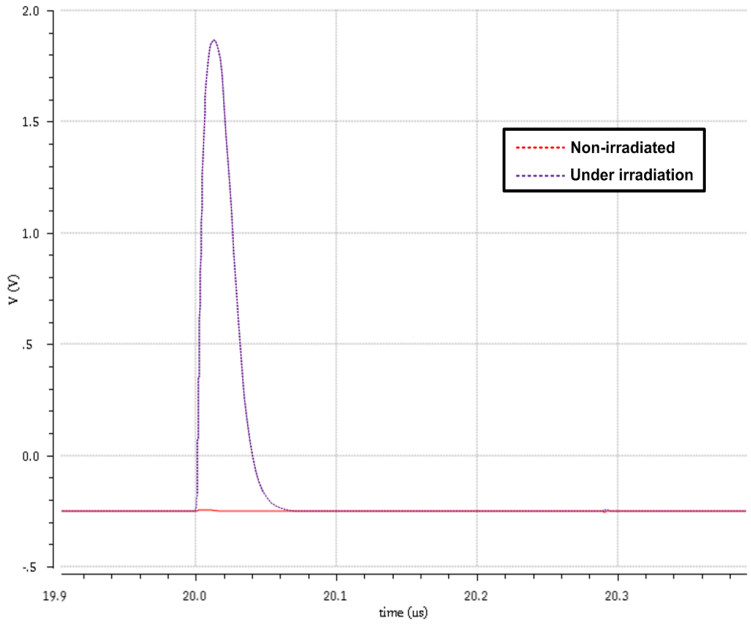
- A/DC SAR using tecnology 250 nm SiGe from IHP.

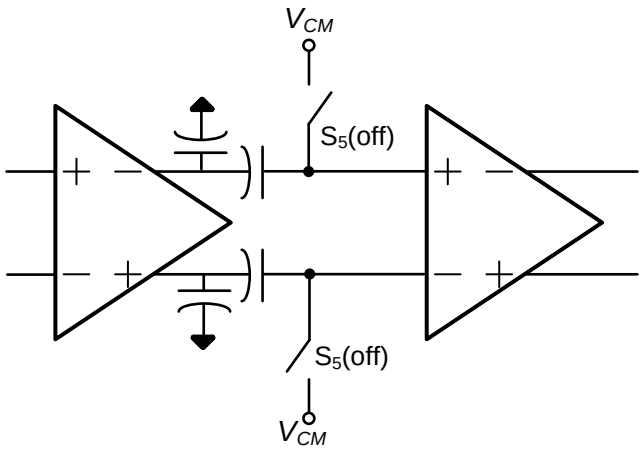




AFTU: case study

- A/DC SAR in 250 nm technology SiGe from IHP.





Database is automatically generated

Output	Impact	Qc	Timp	Trec	Vmax
/I0/net32	I0_M5	5,00E-13	1,00E-08	5.100.000	0.109233
/Vout	I0_M5	5,00E-13	1,00E-08	6.400.000	0.225535
/I0/net32	I0_M5	5,00E-13	2,00E-08	5.100.000	0.109186
/Vout	I0_M5	5,00E-13	2,00E-08	6.400.000	0.225493
/I0/net32	I0_M7	5,00E-13	1.3e-08	2.300.000	0.426629
/Vout	I0_M7	5,00E-13	1.3e-08	3.600.000	0.183520
/I0/net32	I0_M7	5,00E-13	2.6e-08	2.300.000	0.370461
/Vout	I0_M7	5,00E-13	2.6e-08	3.000.000	0.147083
/I0/net32	I0_M10	2,50E-13	1,00E-08	2.800.000	0.205726
/Vout	I0_M10	2,50E-13	1,00E-08	3.900.000	0.186725
/I0/net32	I0_M10	2,50E-13	2.0e-08	3.300.000	0.143872
/Vout	I0_M10	2,50E-13	2.0e-08	4.600.000	0.231184
			.		
			.		
			.		

- Analog Fault Injection is a new methodology of inspection of, in this case, analog circuits.
- This kind of tools predict the behavior of analog cells at topology level.
- These tools are a good assistant for designers.

Acknowledgement:

- Andalusian excellence Project:

EDELWEISS: "Diseño de un Sistema de Comunicaciones Inalámbricas Intra-Satélite de Alta Eficiencia"

P11-TIC-7095

Secretaría General de Universidades Investigación y Tecnología. Junta de Andalucía

Gracias

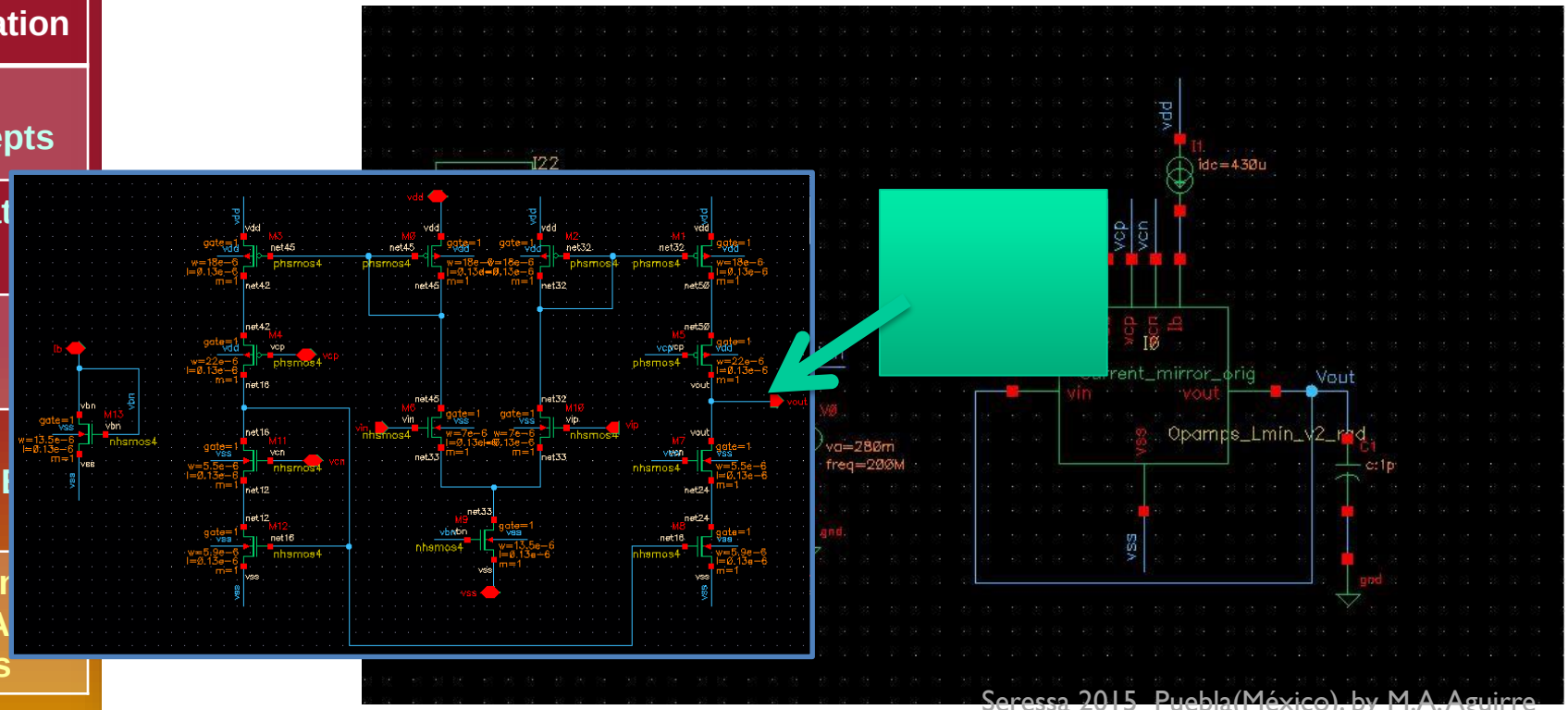
- ¿Preguntas?
- Questions?

Aguirre@gie.esi.us.es

- Extra slides!!

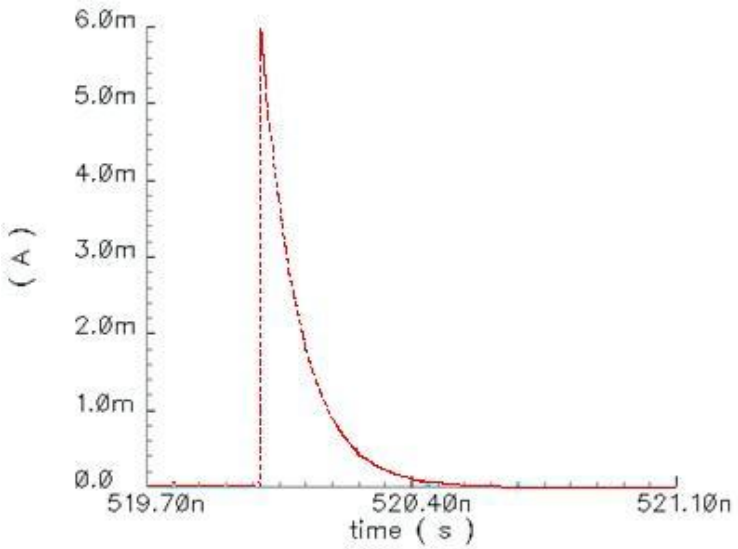
Fault Injection in analog designs

- Analog designs are subjected to a similar radiation effects than digital. SEEs have to be simulated using CADENCE spectre.



SET emulation

- SET emulation based on charge injection models:
  - Current sources with double exponential dynamics*
  - Use of configurable parameters (AHDL implementation)
  - **Open to alternative charge injection model improvements**



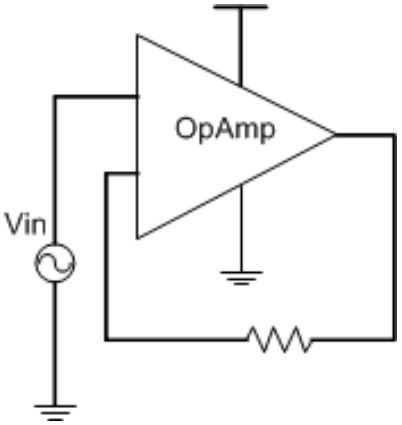
$$I_{rad} = \frac{Q_c}{\tau_d - \tau_r}\left(e^{-\frac{\tau}{\tau_d}} - e^{-\frac{\tau}{\tau_r}}\right)$$

$$Q_c = \frac{\rho \cdot LET \cdot d}{3.6}$$

***REF: G. Messenger,** "Collection of Charge on junction nodes from ion tracks", *IEEE Transactions on nuclear science*, vol.29, n° 6, Dec. 1982

The analysis of several analog topologies has been performed:

- Test-bench:



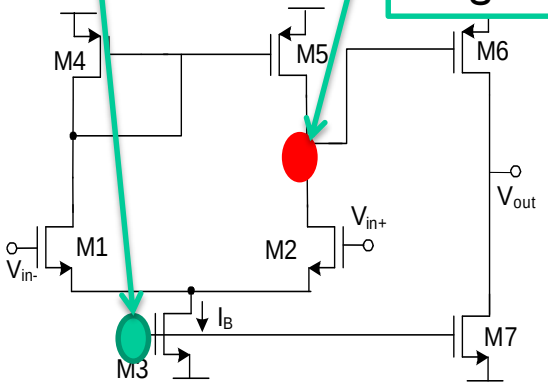
- Input signal:
  - *A= 400mV*
  - *Fin= 200MHz (T=5ns)*

Longest Trec= 5.1ns

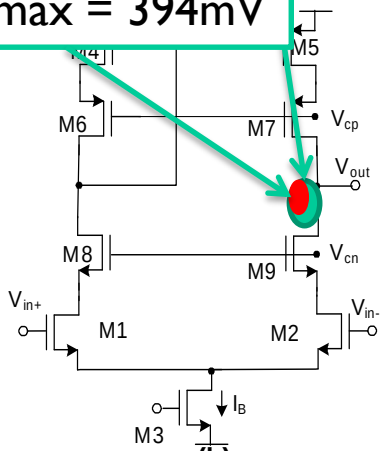
Largest Vmax = 290mV

Longest Trec= 4.4ns

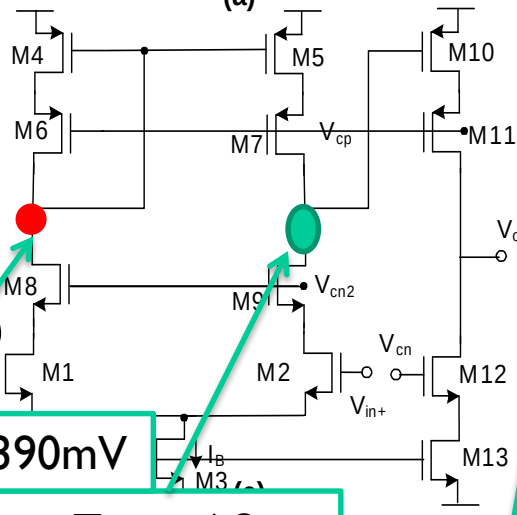
Largest Vmax = 394mV

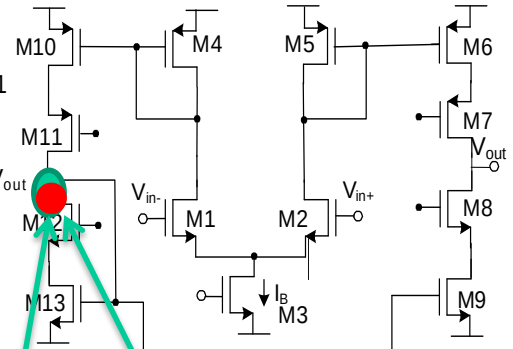


(a)



(b)





Largest Vmax = 390mV

Longest Trec= 6.3ns

Largest Vmax = 390mV

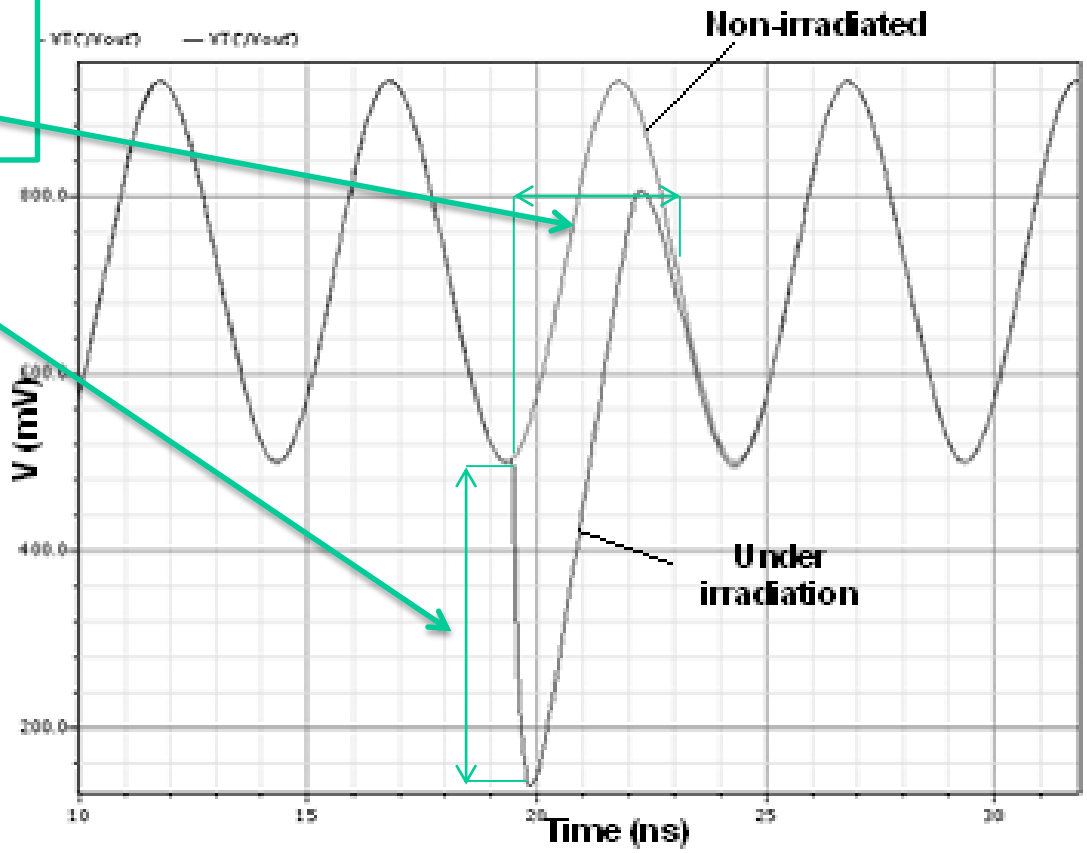
Longest Trec= 6.4ns

Example: SET Sensitivity for 130 nm analog cells

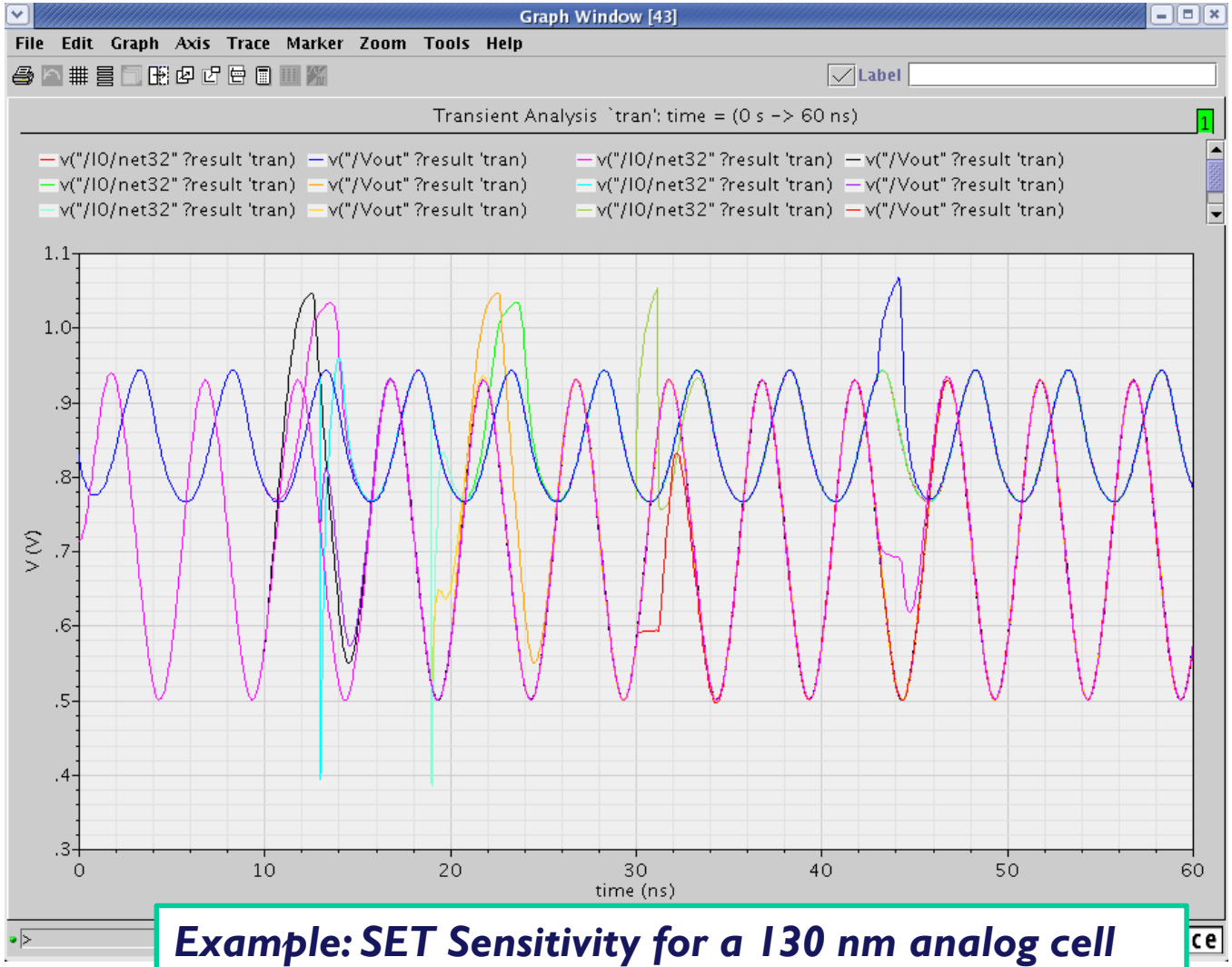
Outputs comparison

Two main transient parameters have been considered:

- Recovery time
- Voltage deviation



Example: SET Sensitivity for a 130 nm analog cell



Example: SET Sensitivity for a 130 nm analog cell

Remarks

- Fault Injection, regardless of the platform, is a methodology of inspection of any kind of circuits, thinking on radiation, but at design level
- It should be integrated in the design flow as a mandatory stage before fabrication.
- It is cheap and effective