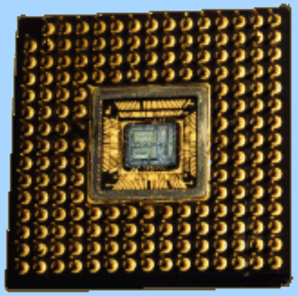
Inyección de fallos para el análisis de la sensibilidad a los errores transitorios, "soft errors", provocados por las radiaciones en circuitos integrados

Dr. Raoul Velazco

Laboratorio TIMA
Grupo «ARIS»
Architectures Robustes
of Integrated circuits and Systems
Grenoble - France
http://tima.imag.fr
raoul.velazco@imag.fr

Motivations

- The microelectronic technology is constantly changing:
  - higher density,
  - faster devices,
  - lower power



- These increase the devices' vulnerability to the effects of radiation (nuclear and space environments).
- Space Agencies favor the use of COTS technologies.
- Present and future technologies are potentially sensitive to the effects of atmospheric neutrons.

Motivations (cont'd)

- **Using commercial devices in space systems, make SEUs being a main concern.**
- **Need for qualifying processors and devices in radiation environment**
- **Radiation ground testing is expensive and time consuming**
- **The final flight application is often not available during the development phase of the project**

→ **ERROR RATE PREDICTION**

Outline

1. **Radiation effects on integrated circuits**
2. **Radiation ground testing**
3. **A two step approach for predicting SEU error rates**
4. **Applying the approach to processors and FPGAs**
5. **Conclusions and perspectives**

1. Radiation effects in integrated circuits

Space radiation

- Light particles
- Heavy ions

Effects of radiation on ICs :

- Total dose (permanent effects)
- Single Events Effects (SEE)

Radiation Effects in integrated circuits : SEU

SEUs are considered critical because they can provoke at random instants :

• modifications of crucial information

• system crashes as the result of sequencing loss (processor program counter perturbation, illegal instructions,...)

Ex: Some parts of the Hubble space telescope had to be replaced by more robust parts.

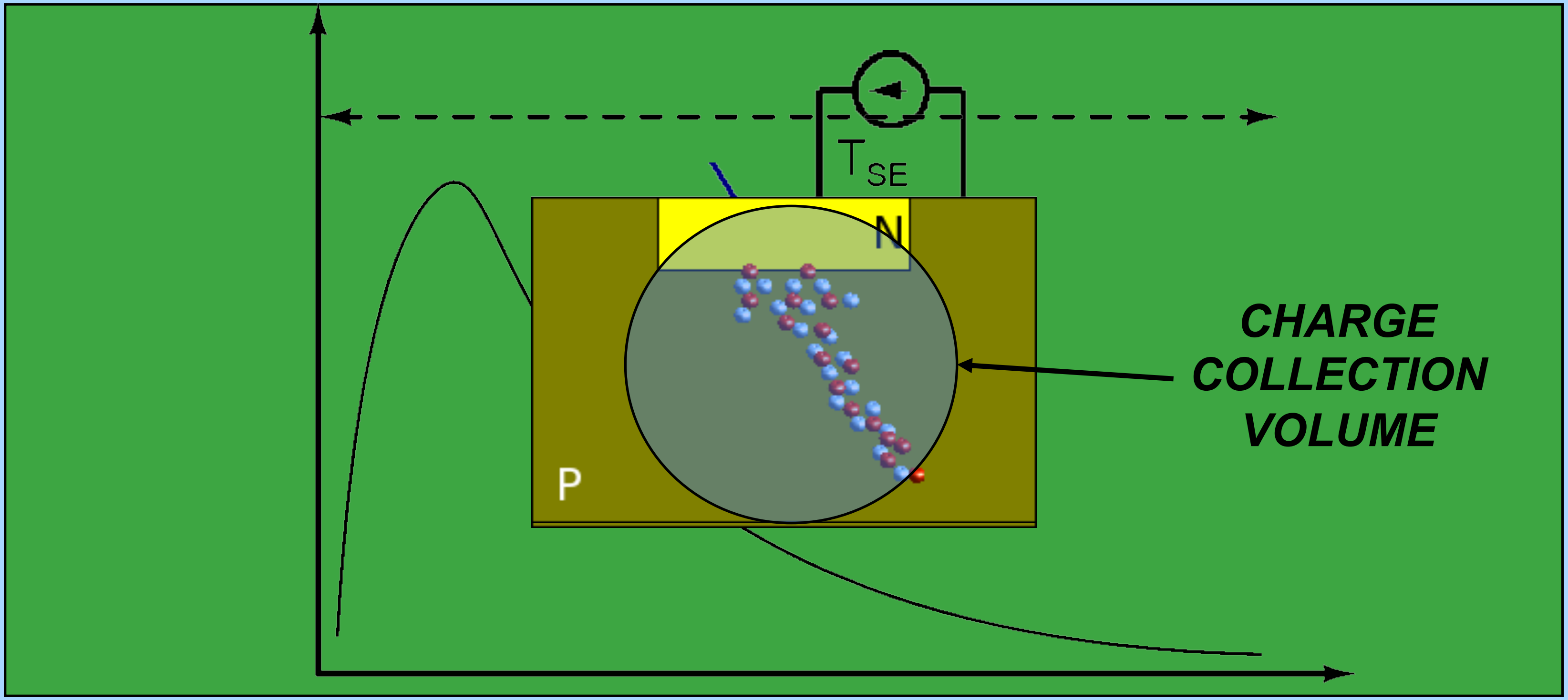
A Description of SEE's

What you always wanted to know about Single Event Effects (SEE's)

- ***What are they?:***
  One of the result of the interaction between the radiation and the electronic devices
- ***How do they act?:***
  Creating free charge in the silicon bulk that, in practical, behaves as a short-life but intense current pulse
- ***Which are the ultimate consequences?***
  From simple bitflips or noise-like signals until the physical destruction of the device

A Description of SEE's

The Physical Mechanism



The incident particle generates a dense track of electron hole pairs and this ionization cause a transient current pulse if the strike occurs near a sensitive volume.

A Description of SEE's

The Classification of SEE's

SINGLE EVENT UPSET (SEU): CHANGE OF DATA OF MEMORY CELLS

MULTIPLE BIT UPSET (MBU): SEVERAL SIMULTANEOUS SEU'S

SINGLE EVENT TRANSIENT (SET): PEAKS IN COMBINATIONAL IC's

FUNCTIONAL INTERRUPTION (SEFI): PHENOMENA IN CRITICAL PARTS

SINGLE EVENT LATCH-UP (SEL): PARASITIC THYRISTOR TRIGGER

AND OTHERS…

HARD ERRORS vs SOFT ERRORS

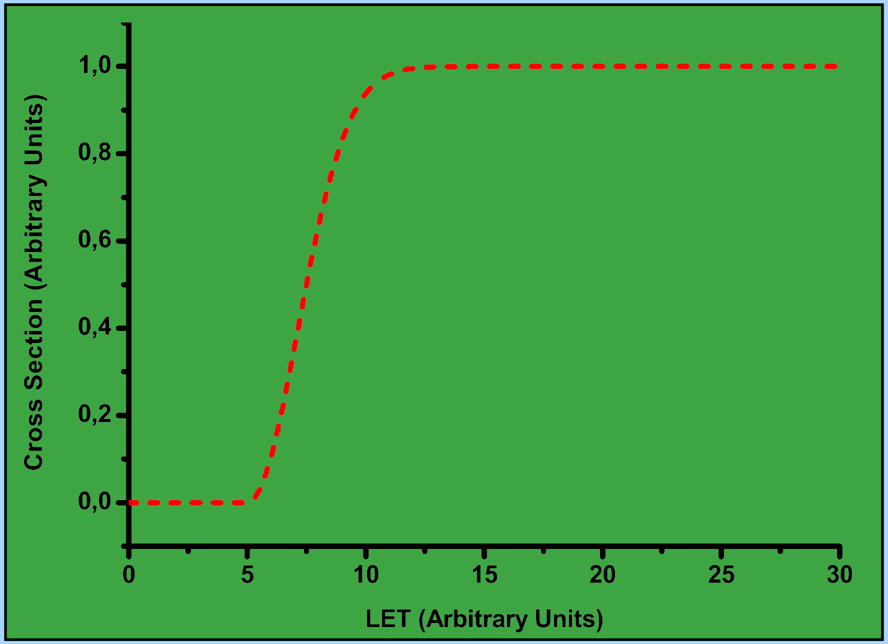
A Description of SEE's

Some Useful Definitions

LINEAR ENERGY TRANSFER (LET)

CROSS SECTION (σ)

$$\sigma_{DEV} = \frac{N_{EVENTS}}{Part.Fluence}$$



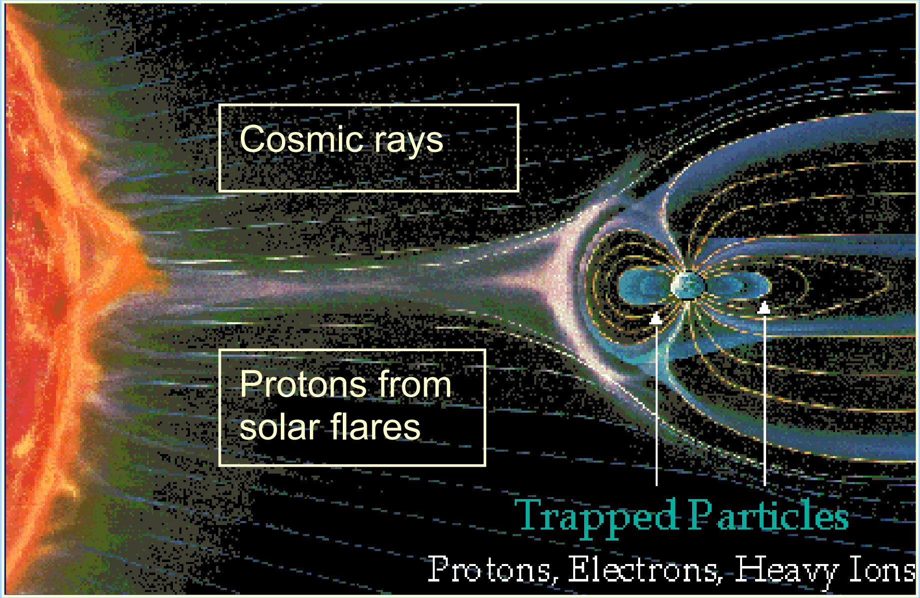
SOFT ERROR RATE: PROBABILITY OF AN ERROR AT USUAL CONDITIONS

FIT: Typical unit of SER → Probability of 1 ERROR every 10^9 h

E.g.- 180-nm SRAM: 1000-3000 FIT/Mb

Sources of SEE's

Usually, SEE's have been associated with space missions because of the absence of the atmospheric shield…



Unfortunately, our quiet oasis seems to be vanishing since the enemy is knocking on the door…

- ***Alpha particle from vestigial U or Th traces***
- ***Atmospheric neutrons and other cosmic rays***

Radiation Ground Testing: **Requirements**

Accelerated radiation ground testing are performed on-line and need:

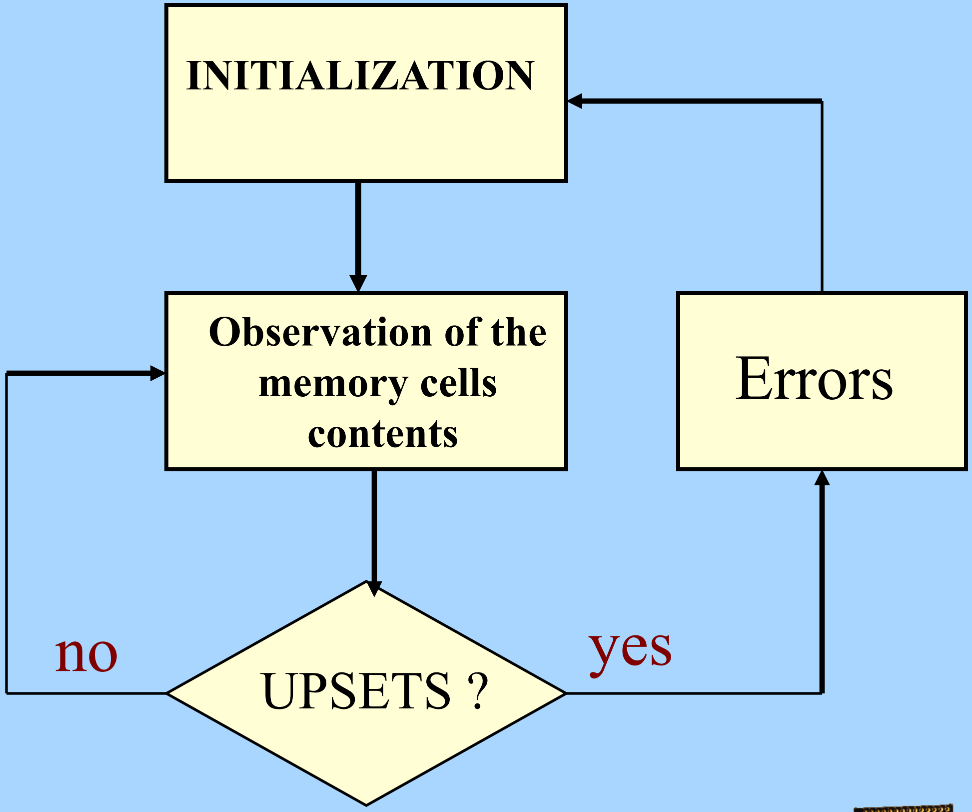
- a particle beam, which can be obtained by Radiation Facilities :
  - particle accelerators: cyclotrons, linear accelerators,...
  - equipments based on fission decay sources such as Cf^{252}
- a test methodology, defining the activity of the device under test (DUT)
- an electronic test equipment for controlling and observing the behavior of the DUT during its exposition to radiation.
- and.... A deep expertise and ...good luck

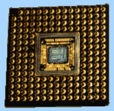
Radiation Ground Testing: **SEU test strategies**

SEU Testing

- Static test: memories, processors
- Dynamic test: more realistic
- Activate R/W sequences (memories)
- Execute a given program (processors)

Static Test





ARIS

Radiation Ground Testing:
Need for a dynamic strategy for processor's SEU testing

- The contribution to the SEU cross-section of a memory element is related with its duty periods: time between loading a value and reading it

- The cross-section of any program can be calculated as:

$$\sigma(SEU) = \Sigma\, d\,(R_i) \times \sigma_{Ri}$$

where

- $d_i\,(R_i)$ is the duty factor of memory element R_i, i.e. the sum of all the duty periods

- σ_{Ri} is the SEU cross-section of R_i, calculated from a static strategy

Radiation Ground Testing: **Need for a dynamic strategy for processor's SEU testing (cont'd)**

Cycle nb. | Program code | **Sensitive period nb.**

Cycle nb.		Program code	Sensitive period nb.
1		LD R, #3	
2	t1 (cycles 2–5)	LD A, #5	
3		Loop: ST Mem [add r], A	
4		INC addr	→ 10 cycles
5		DJNZ R, Loop	
.		.	
.		.	
.		.	
11	t2 (cycles 11–12)	LD A,#2	
12		LD R, A	
13		ADD R, #4	→ 2 cycles
14		MUL R, #10	
.		.	
.		.	

Total: 1200 cycles

➔ **Contribution of register A to the SEU error rate: d_A= 0,01**

ARIS

3. An Error Rate prediction methodology

- Radiation ground testing of complex circuits such as digital processors is usually performed with “static strategies” or with simple applications.

↓

What is the significance of derived error rates with respect to those of the final application?

➔ Strategy based on upset-like fault injection for the prediction of the SEU error-rate of microprocessor-based architectures

An Error Rate prediction methodology (cont'd)

Strategy to predict SEU the error-rate suitable for any circuit :

- Step 1: Radiation ground testing in a suitable facility:
  static SEU cross-section given in cm^2

$$\sigma_{SEU} = \#upsets / \#particles \quad (cm^2)$$

➞ How many particles to provoke an upset ?

- Step 2: Fault injection sessions (off-beam upset simulation):

$$\tau_{inj} = \#errors / \#upsets$$

➞ How many upsets to provoke an error in the studied application?

An error rate prediction methodology (cont'd)

- Error-rate estimation:

$\tau_{SEU} = \sigma_{SEU} * \tau_{inj}$ [errors/particle]

- Error rate in flight

τ_{SEU}*Expected particle fluency [errors/time unit]

An Error Rate prediction methodology (cont'd)

Main benefits if applied to processors

Radiation ground testing performed only once for a given processor but not for each application

Test cost and time drastically decrease

The application upset error rate can be evaluated concurrently with software developments.

Key point: How to perform "realistic" upset simulations for the chosen HW/SW application?

Time to step back for a while…

- To inject a fault, the following 3 questions must be addressed:
  - When ?
  - Where ?
  - How ?

When inject a fault to simulate SEUs ?

- Classic fault injection says:

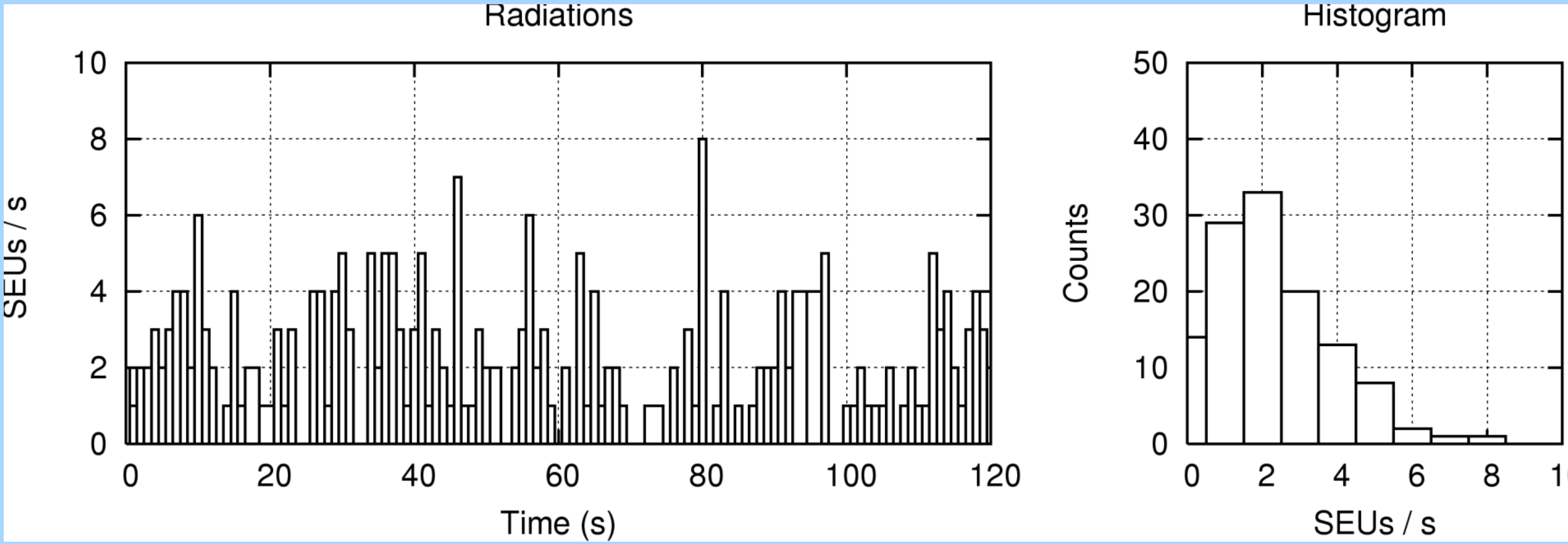
"More than one fault per execution inject you shall not. To the dark side of the force this path leads."[1]

[1] ~~Master Yoda, A long time ago in a galaxy far away.~~.. Unknown, undated.

Why inject a fault to simulate SEUs ?

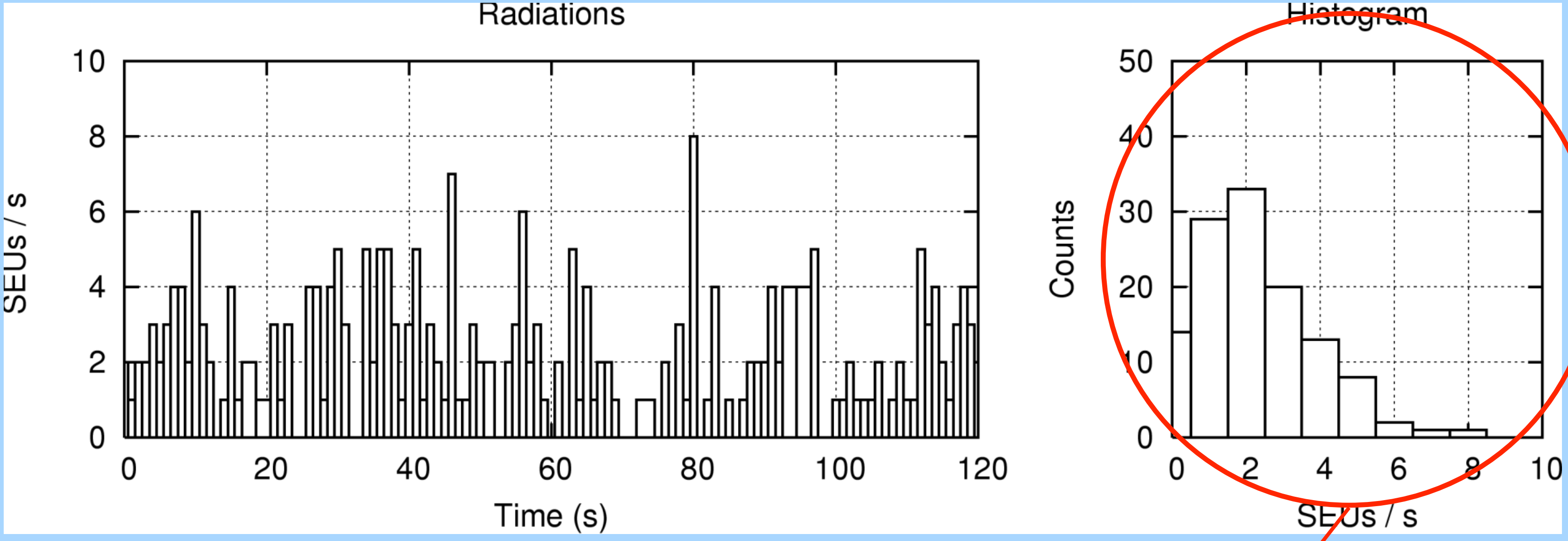
- No real reason...
- Let's look at some data.

Real upset rates: radiation ground testing of LEON processor, static strategy.



- Fact: The upset rate issued from a static test is not constant.
- There is no clear mean value.

Real upset rates (radiation ground testing)



- Fact: The upset rate of a static test is not constant.
- There is no clear mean value.

Hint: Guess the name of that distribution

More data: Bubble sort benchmark, LEON processor

Flux (particle.$cm^{-2}.s^{-1}$)	Error rate (#Errors.$particle^{-1}$)
1×10^{4}	4.48×10^{-4}
5×10^{3}	6.07×10^{-4}
2×10^{3}	7.55×10^{-4}
1×10^{3}	8.66×10^{-4}

- **Fact**: The error rate of a dynamic test **scales** with the flux…but as *the opposite of the intuition.* The higher is the flux, the higher is the probability that a first fault results in an error **masking** future faults.
- Classic fault injection (one fault injected per execution) says the error rate is 9.00×10^{-4}.
- Classic fault injection **cannot** reproduce this.

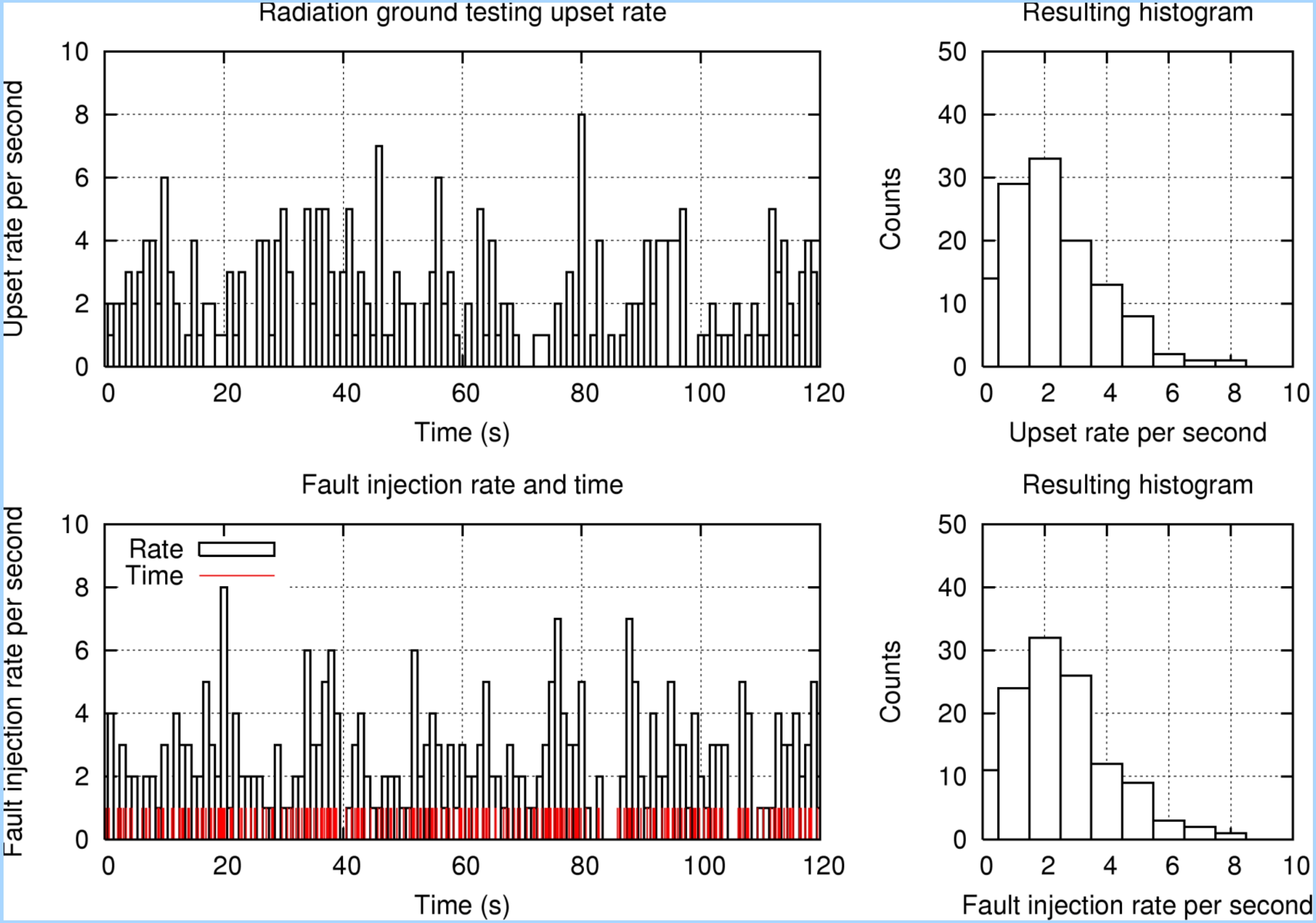
So, really, When ?

- Upsets appear following a Poisson distribution.
- If the flux is constant, the variable counting the upset rate follows an homogeneous Poisson process.
- Using this, the time interval between two upsets is exponentially distributed:
- Theory backing up this is given in:

$$P\left(N_{SEU}(t+\Delta t) = N_{SEU}(t)\right) = e^{-\sigma \times \phi \times \Delta t}$$

F. Faure "*Fault injection simulating the effects of bit-flips induced by radiation*", INPG Ph.D. Thesis, 2005.

Results on upset rates: **static strategy**



ARIS

Results on error rates: **bubble sort**

Type	Flux ($p.cm^{-2}.s^{-1}$)	Error rate ($\#Errors.s^{-1}$)
Radiations Injections	$1x10^4$	$4.48x10^{-4}$ $4.62x10^{-4}$
Radiations Injections	$5x10^3$	$6.07x10^{-4}$ $6.36x10^{-4}$
Radiations Injections	$2x10^3$	$7.55x10^{-4}$ $7.88x10^{-4}$
Radiations Injections	$1x10^3$	$8.66x10^{-4}$ $8.47x10^{-4}$

- Proposed fault injection approach can reproduce the flux scaling!

ARIS

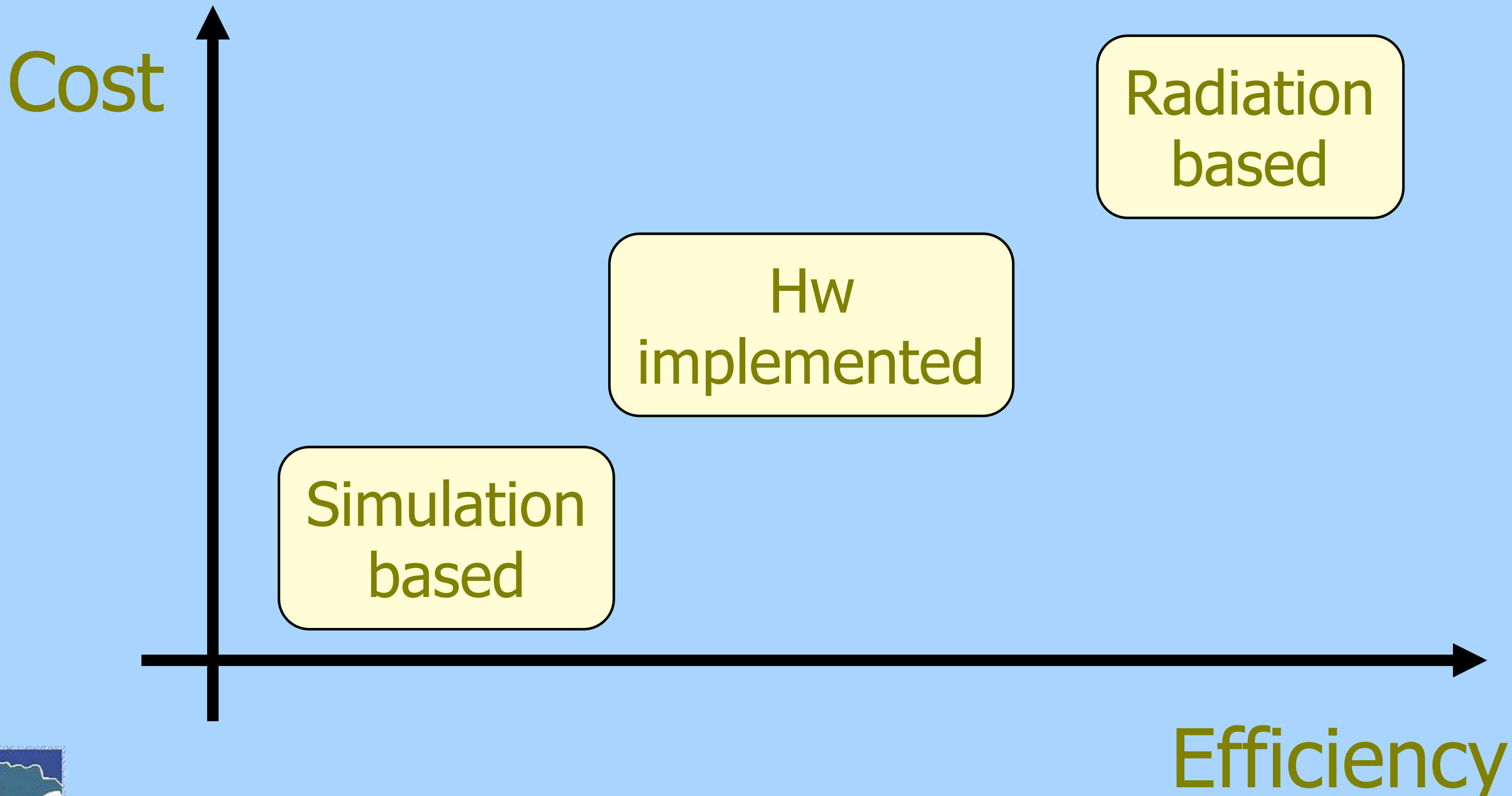
Where ?

- Simple case: all memory elements have the same cross-section.
  - Randomly choose one among N.
- Complex case: several cross-sections.
  - Use the *superposition principle* (Poisson process property).

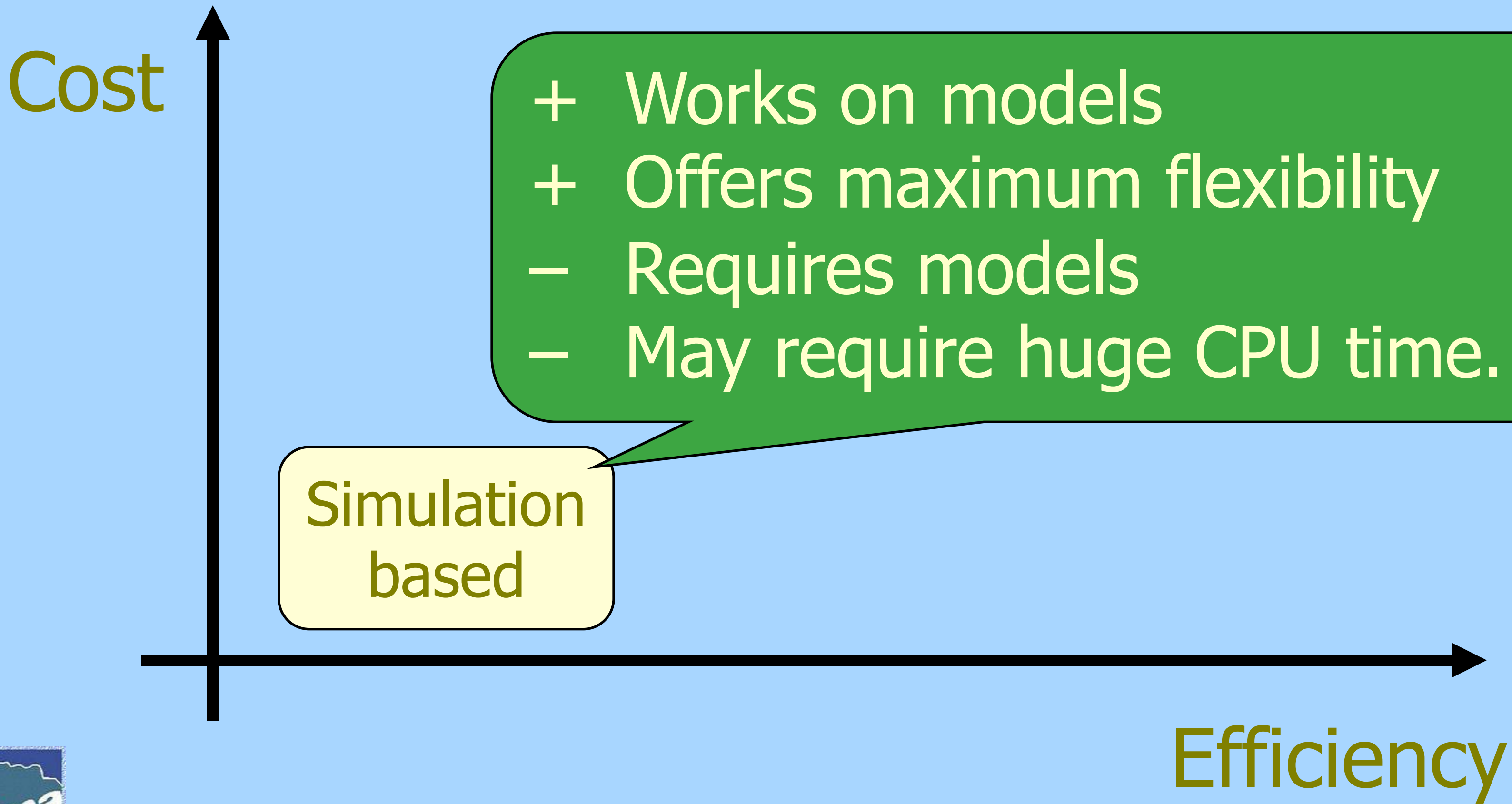
How ?

- See next slides...

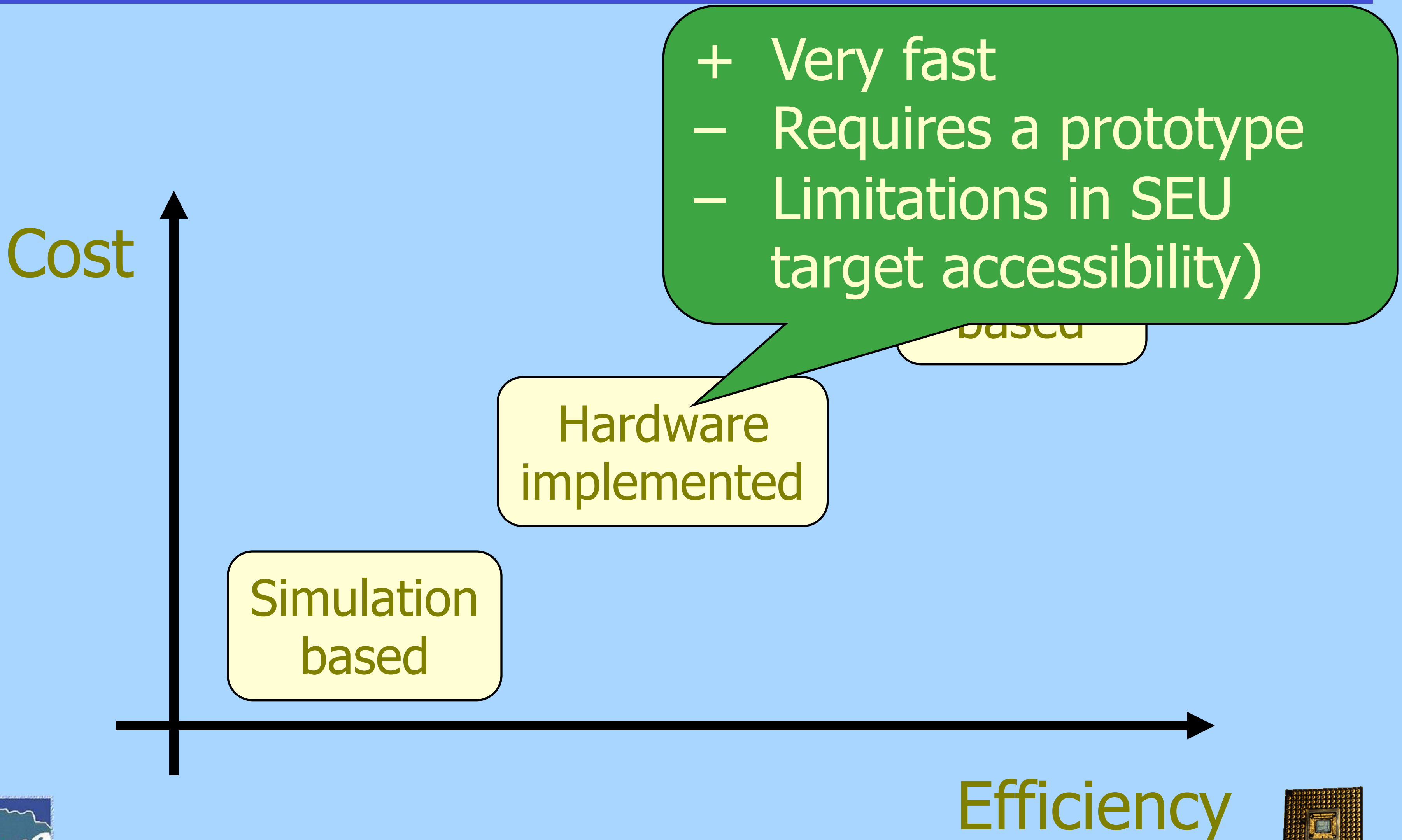
4. Available techniques for upset injection



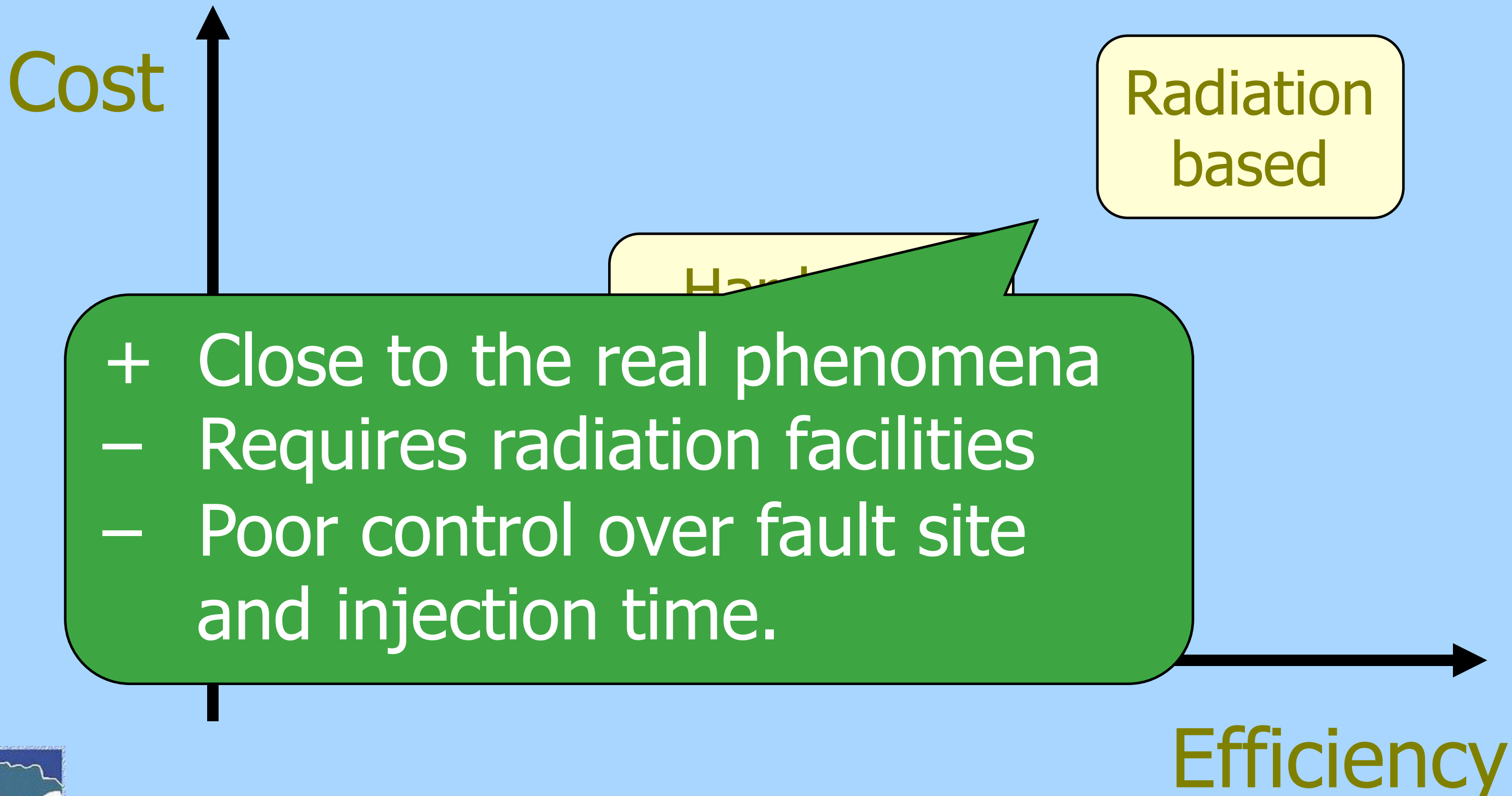
Available techniques for upset injection (cnt'd)



Available techniques for upset injection (cnt'd)



Available techniques for upset injection (cnt'd)



- \+ Close to the real phenomena
- – Requires radiation facilities
- – Poor control over fault site and injection time.

ARIS

Available techniques for upset injection (cnt'd)

Hardware based:

- Using particular processor execution modes (Trace, debugging, ...)
- Using asynchronous signals (DMA, Exceptions, Interrupts, ...)

Software based:

- Using a instruction level simulator of the processor under study
- Using a HDL (Hardware Description Language) model of the processor

A) HW-based SEU simulation for processor-like circuits: The CEU (Code Emulated Upsets) approach

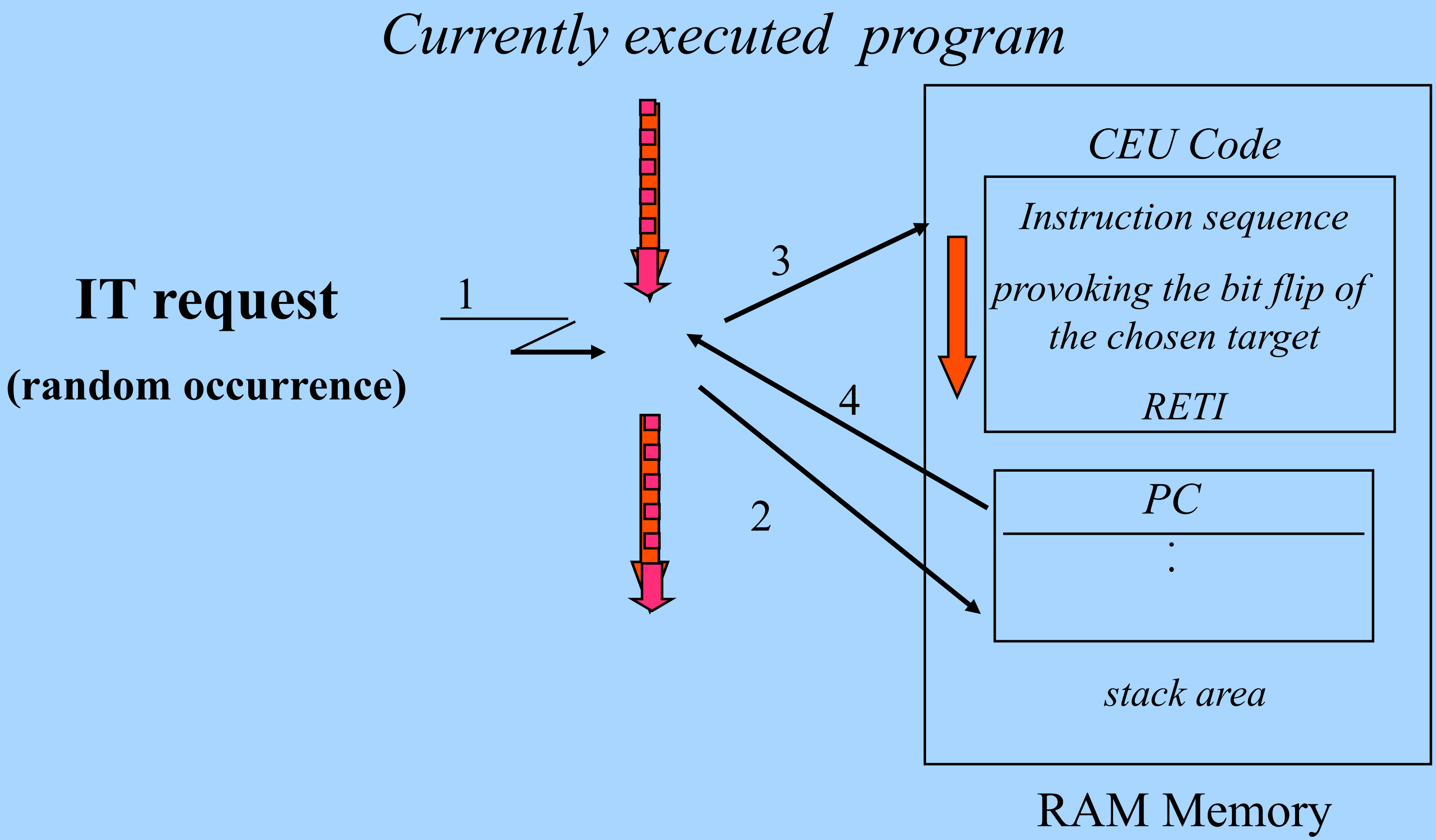
R. Velazco, S. Rezgui, R. Ecoffet., *Predicting error rate for microprocessor-based digital architectures by C.E.U. Injection*, IEEE Trans. on Nuclear Science, Vol. 47, N° 6, Dec. 2000, pp. 2405-2411.

- **Basic idea:**

  Use **Interrupt signals** to inject bit flips in processors

- **Main Steps:**
  - Selection of the CEU target (randomly or exhaustively)
  - Storage of CEU code for upset emulation in a memory zone
  - Execution of the CEU code (interruption signal assertion)
  - Comparison of obtained and expected results

HW based upset simulation: **the CEU approach (cnt'd)**



HW based upset simulation: **the CEU approach (cnt'd)**

Examples of CEU codes according to the target type

upset target	CEU code
• Register R	XOR R, mask(i) RETI
• Internal or external RAM	PUSH R LD R, Mem(@) XOR R, mask(i) ST Mem(@), R POP R RETI
• PC (program counter)	Modify the PC stored in the stack then RETI

mask(i) = 0000...10000

→ i^{th} bit

ARIS

HW based upset simulation: **CEU codes**

• **Injecting an upset in SP** requires avoiding the use of RETI to return back to the main program after injecting the fault. Idea of solution: **emulate RETI** by « writing » the code of a JUMP to the return address (which can be obtained by reading the value of PC saved in the stack).

• **The size of CEU** codes may go from a 3 bytes (for a general purpose register for exemple) to some tens of bytes (for PC).

• After CEU injection, its **effects at the program behaviour** must be observed. This can be done by comparison to expected values of:

- the content of a particular memory area where program outputs are stored
- the execution time of the whole program

B) Software based upset fault injection

<u>SEU injection</u> can be achieved by means of a SW simulator

- Easier & cheaper implementation
- The targets include a wider set of processor's memory elements
- Suitable to study the effects of SEUs on critical memory zones

B) Software based upset fault injection **(cont'd)**

GENERIC MODEL for an Upset

1. Program Execution on Processor
2. Stop Execution when time = INSTANT
3. Flip the TARGET bit among all processor bits
4. Resume execution

(*) INSTANT & TARGET are pseudo-randomly chosen

B) Software based upset fault injection (cont'd)

- Using the processor simulator to inject bit flips while executing the studied program

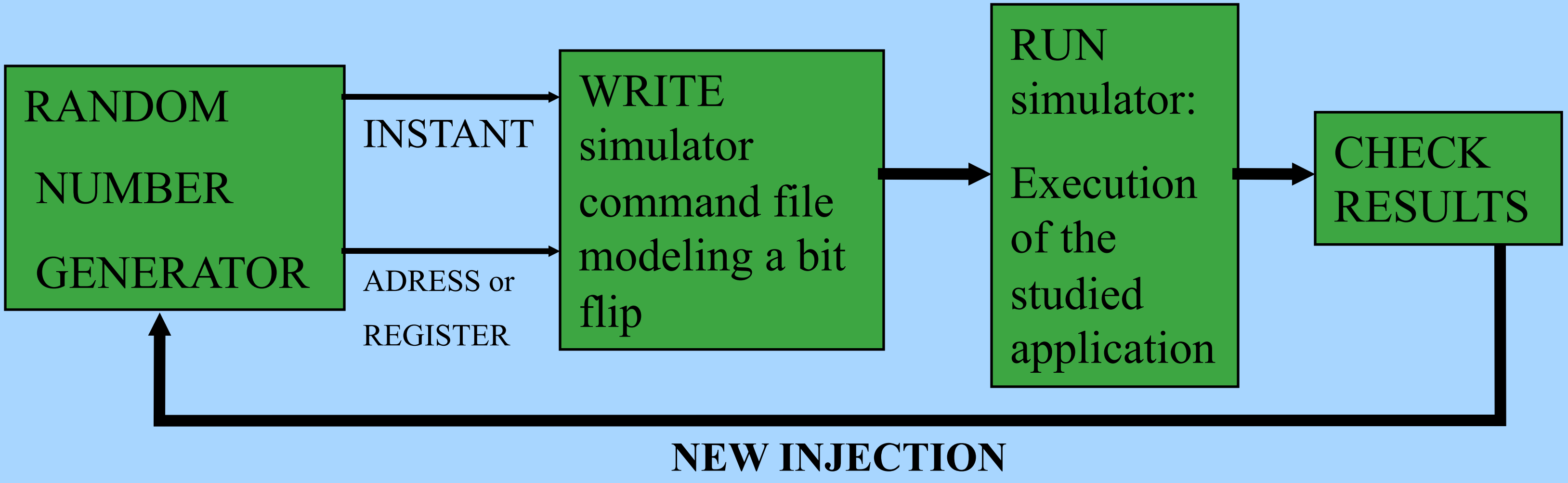
- Command file modeling a bit flip: using simulator breakpoints

```
when t>=TOTAL_TIME {"Lost Seq"; quit}

when t>=INSTANT {TARGET=TARGET^XOR_VALUE; cont}

b end_cma {Print progarm output results; quit}

run
```

- the values of INSTANT and TARGET are instanciated by a Testbench

B) Software based upset fault injection: **TestBench**

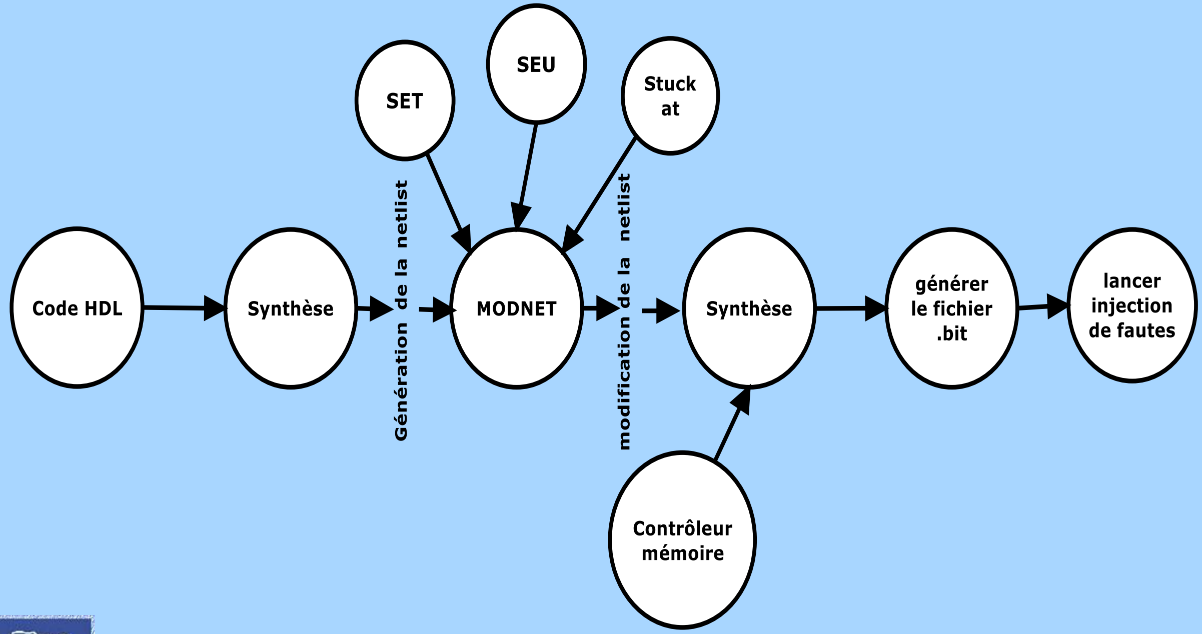
- Writing command files to inject a bit flip
- C-language Testbench:



ARIS

C) Netlist fault injection

- Automated method to emulate faults using SRAM-based FPGAs
- Based on the netlist manipulation: replacement of Xilinx built-in devices by others having same functionalities and allowing fault injection
- Allows injecting various types of faults: SEU, SET and Stuck_at
- No restriction of the size of the FPGA in which the circuit will be emulated
- Can be applied to any digital circuit



Requirements:

-HDL code

-A FPGA platform

-A PC

-"Synplify Pro" synthesis tool

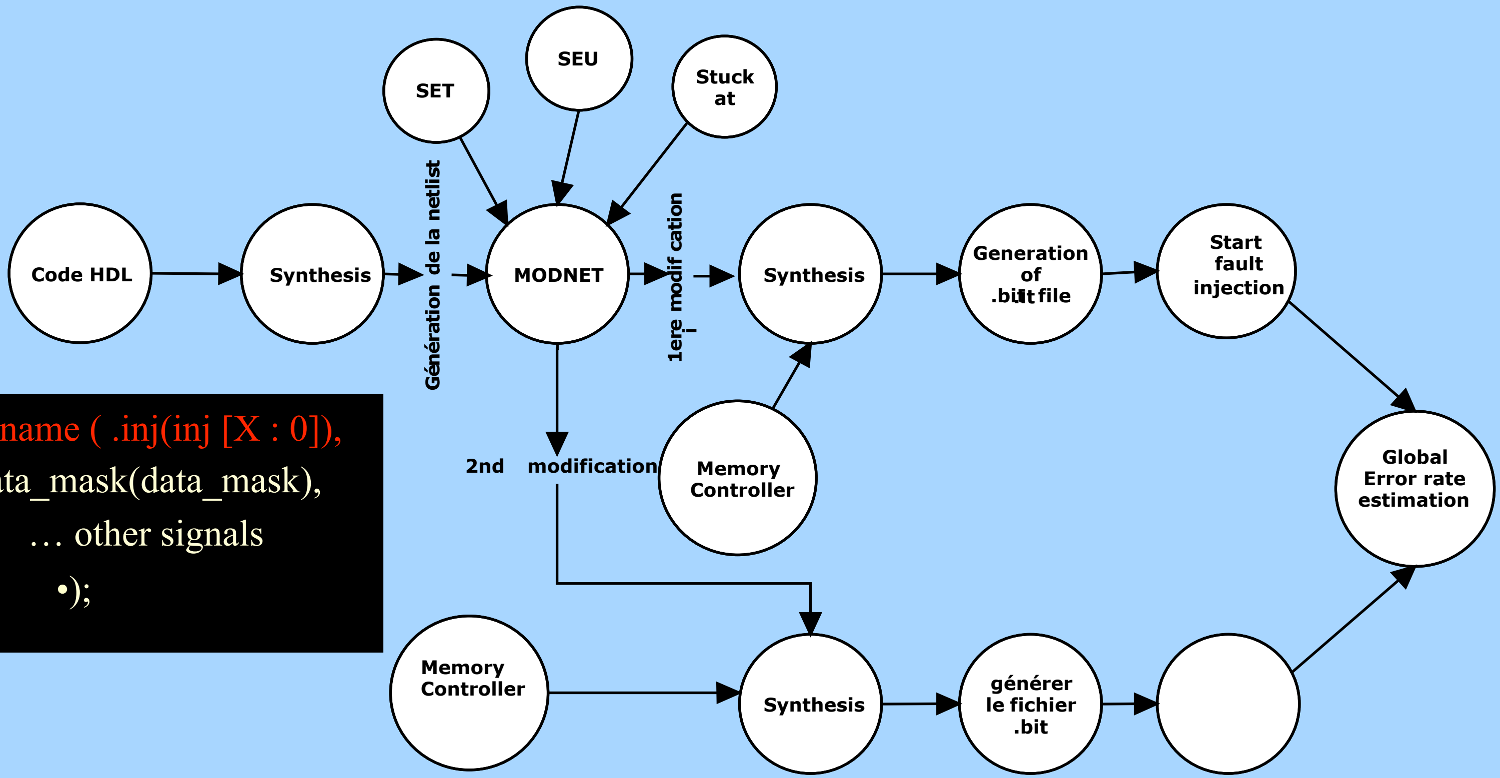
-A place &root Xilinx tool

-Impact Xilinx tool for programming the FPGA

-A *Linux-Like* tool such as Cygwin

-MODNET (MODify NETlist)

C) Netlist fault injection: The NETFI approach



- DUT DUT_name (.inj(inj [X : 0]),
  - .Data_mask(data_mask),
    - … other signals
-);

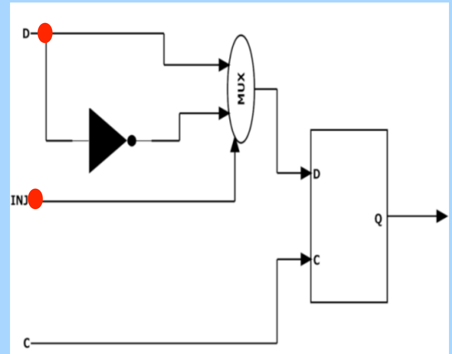
- DUT DUT_name (.inj({X/2 ‘b0 , inj [X/2 : 0]}),

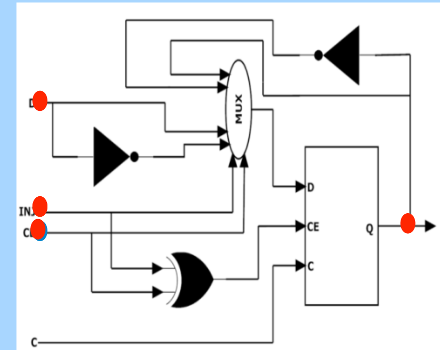
- DUT DUT_name (.inj({inj [X:X/2],X/2’b0}),

The number of faults injected in each sub-netlist will depend on their surfaces in terms of bits

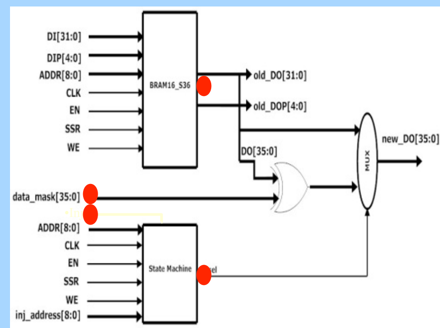
C) Netlist fault injection: The NETFI approach (cnt'd)

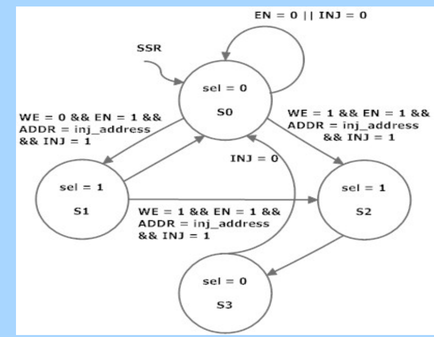
- For SEU injection:
  - Flip-Flops (with and without Enable)
  - Bloc Rams (BRAMS)
  - Shift Registers SRL_16, RAMX1D…



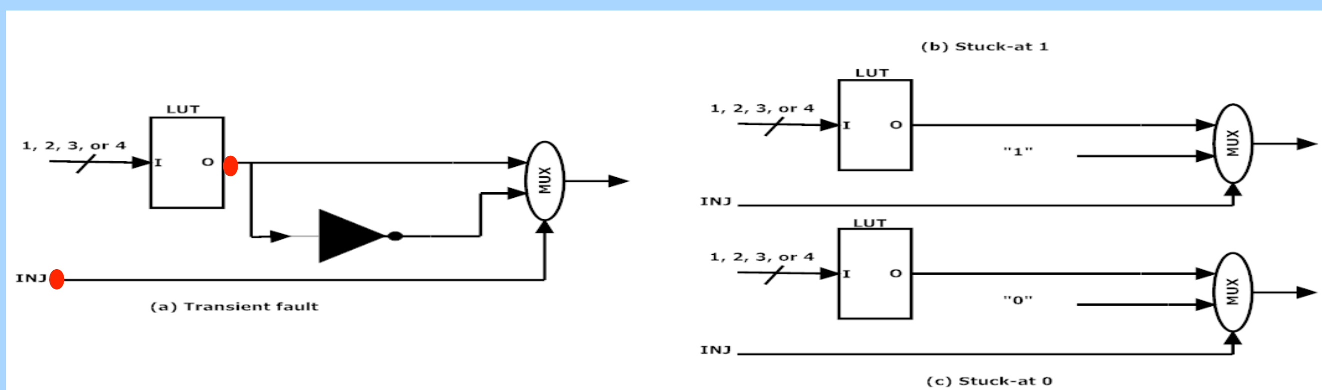


- For SET and Stuck_at fault injection:
  - LUT (Look-Up Table)
  - Logic Gates (AND2, OR2, XOR2, MUXF5..)





- **The pulse duration is configurable**



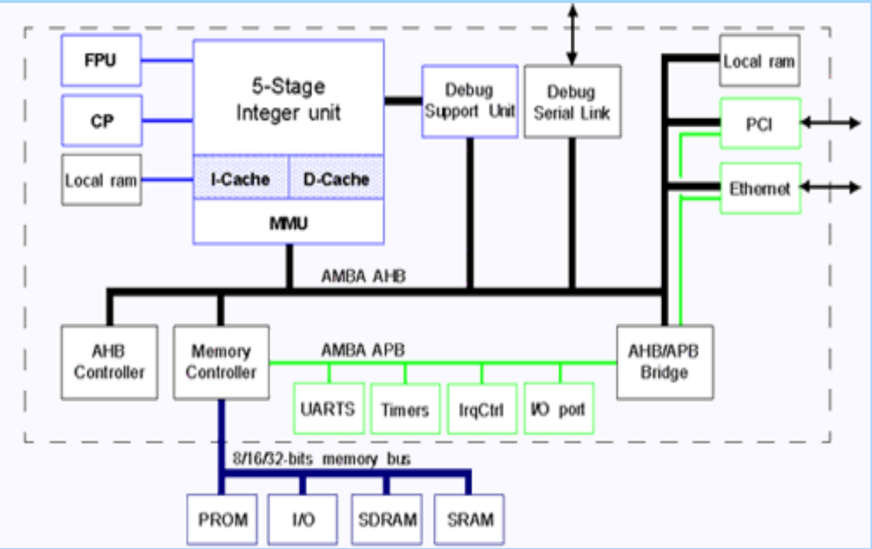
ARIS

C) Netlist fault injection: Application to a LEON 2

- 2613 Flip-Flop (pipeline + control)
- 32 BRAM4_S8
    - 16 data cache
    - 16 instruction cache
- register file: 135x32 registers
    - RAMX1D

•Targets: Flip-Flops + register files

• total: 6933 memory cells



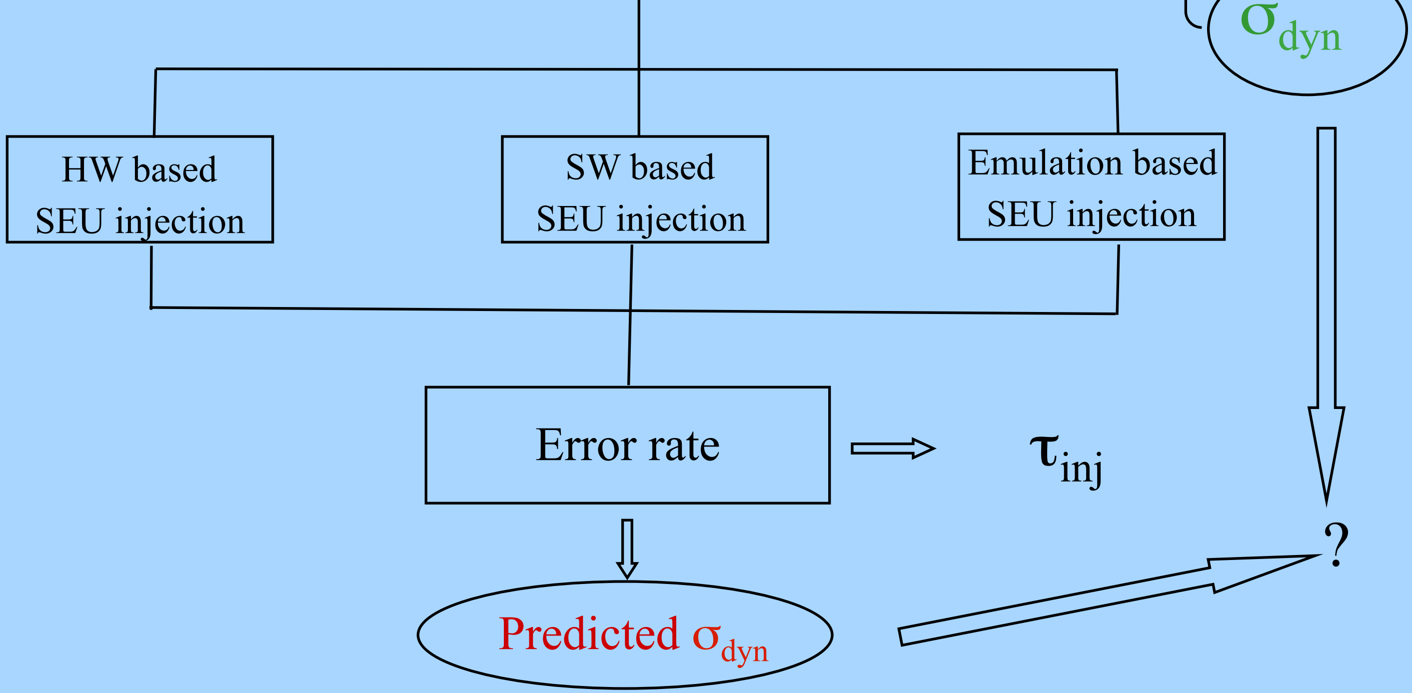
•Algorithme: *bubble sort* 1K data

Option	# injected faults	# errors	# timeouts
Cache deactivated	19474	630 (3.325%)	1832 (9.40%)
Cache activated	19867	526 (2.648%)	2125 (10.69%)

•Radiation test campaigns are scheduled end of 2014 for LEON2

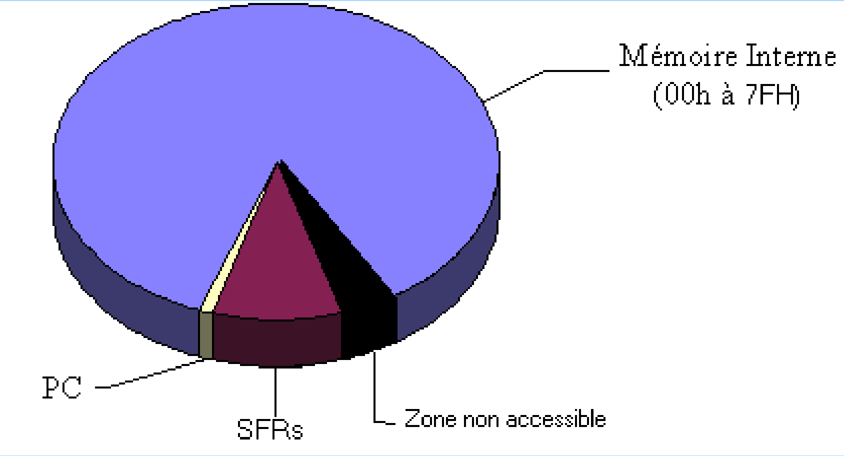
5. Combining Radiation Ground Testing with Fault Injection Sessions

The accuracy of the proposed error prediction approach must be evaluated according to the following phases:



5.A) Combining Radiation Ground Testing with Fault Injection Sessions **First case study: the 80C51 microcontroller**

- Program running during upset injection:
  - 6x6 matrix multiplication
- CEU targets:
  - all internal registers
  - internal SRAM (128 bytes)
- Observed errors:
  - sequence loss
  - single or multiple matrix result-errors



5.1) Combining Radiation Ground Testing with Fault Injection
A case study: the 80C51 microcontroller (cnt'd)

CEU Target	*CEU Code (assembly language of 8051)*	*Comments*
Accumulator	`XOR ACC, BitPos` `RETI`	Modification of one ACC register bit Return to main program

One byte of Internal or external SRAM	`PUSH ACC`
	`LOAD ACC, addr`
	`XOR ACC, BitPos`
	`STORE ACC, addr`
	`POP ACC`
	`RETI`
		Save the content of accumulator ACC
		Read the content of the target byte
		Modify the target bit
		Store the modified byte in target SRAM
		Restore ACC
		Return to main program

"addr" is the address of the SRAM byte to be perturbed.
"BitPos" is a byte having a 1 among 0s, corresponding to the position to be inverted.

5.1) Combining Radiation Ground Testing with Fault Injection
A case study: the 80C51 microcontroller (cont'd)

CEU target	CEU code	Comments
Program Counter Low	*PUSH R0* *PUSH ACC* *LOAD R0, SP* *LOAD ACC, @R0* *XOR ACC,* ***BitPos*** *STORE @R0, ACC* *POP ACC* *POP R0* *RETI*	Save the content of R0 in the stack Save the content of ACC Use R0 to point where PCL is stored Load PCL in the ACC register Flip the content of the target bit in ACC Store the modified value in PCL Restore ACC Restore R0 Return to main program
Program Counter High	*PUSH R0* *PUSH ACC* *DEC SP* *LOAD R0, @SP* *LOAD ACC, @R0* *XOR ACC,* ***BitPos*** *STORE @R0, ACC* *INC SP* *POP ACC* *POP R0* *RETI*	Save R0 Save the accumulator content Decrement the content of SP Point with R0 to the second half of PC Load the content of PCH in ACC Change the target bit content at ACC Transfer ACC content to PCH Increment SP Restore ACC Restore R0 Return to main program

CEU codes of program counter PC

ARIS

5.1) Combining Radiation Ground Testing with Fault Injection
A case study: the 80C51 microcontroller (cont'd)

12245 bit flip faults were injected while running the 6x6 matrix multiplication program.

# injected errors		***Effect-less CEUs***	***Result Errors***	***Sequence Loss***
Internal memory	10780	4890	5700	190
SFRs	1465	1227	84	154
Total	12245	6117 (49.96 %)	5784 (47.24 %)	344 (2.8 %)

The accessible targets represent 93% of the total memory cells

5.1) Combining Radiation Ground Testing with Fault Injection
A case study: the 80C51 microcontroller (cont'd)

Results for two different memory occupancy strategies of the matrix multiplication program.

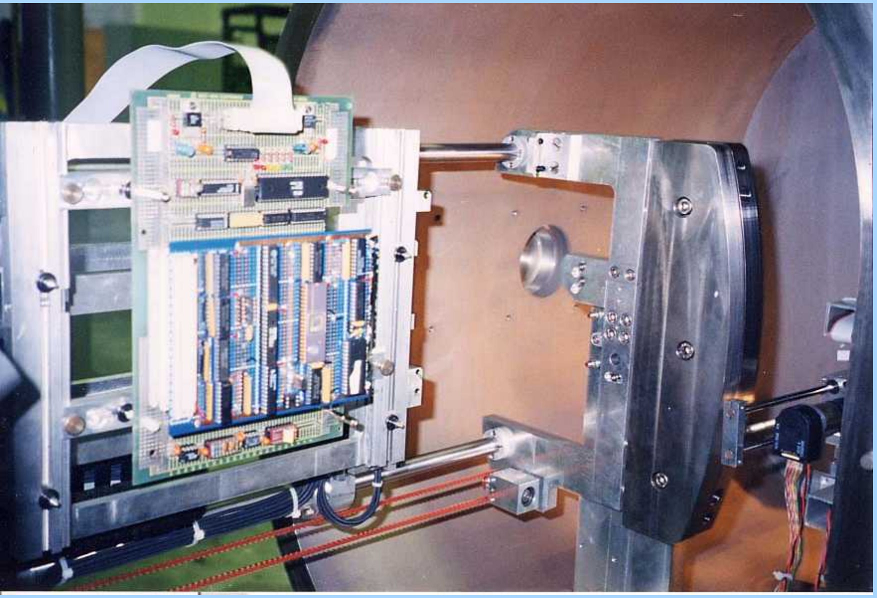
Type of error	*Matrices stored in Internal SRAM*	*Matrices stored in External SRAM*
No Error	**50%**	**94%**
Result Error	**47%**	**4%**
Sequence Loss	**3%**	**2%**

5.1) Combining Radiation Ground Testing with Fault Injection

Radiation test results for the 80C51 microcontroller

- The 8051 THESIC daughterboard was exposed to heavy ion beams while running a matrix multiplication program.
- The “Cyclone” cyclotron facility of Louvain-la-Neuve was used.
- Main Goals:
  - measure the 8051 SEU static cross-section
  - assessing the methodology of error rate prediction

5.1) Combining Radiation Ground Testing with Fault Injection
Radiation test results for the 8051 (cont'd)



The 8051 THESIC system at

the vacuum chamber of Cyclone

M/Q=5	ENERGY [MEV]	LET [MeV/mg/cm²]
$^{40}Ar^{8+}$	150	14.1
$^{20}Ne^{4+}$	78	5.85
$^{15}N^{3+}$	62	2.97
$^{10}B^{2+}$	41	1.7
$^{84}Kr^{17+}$	316	34

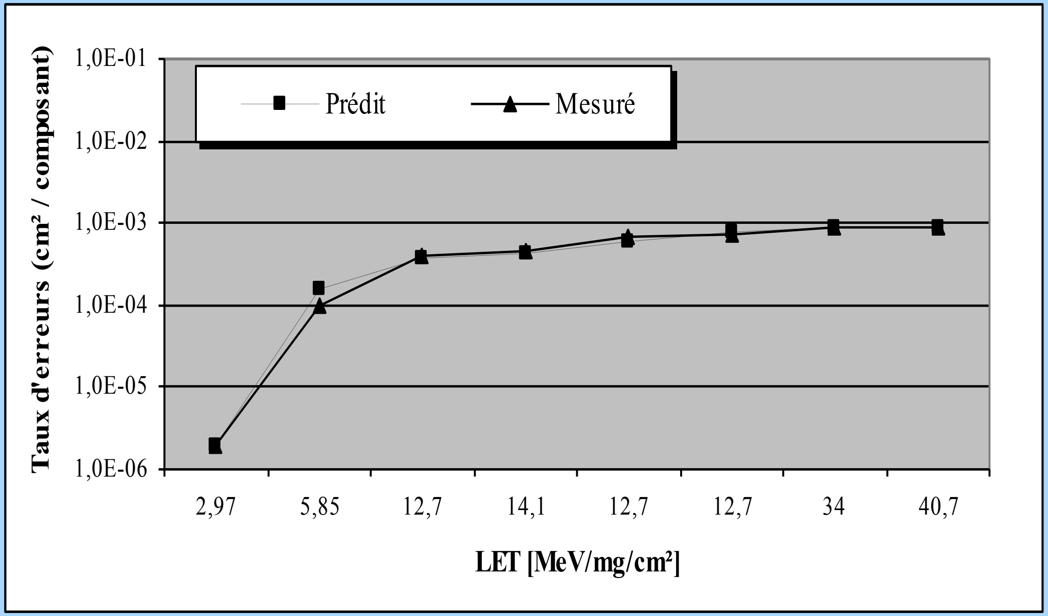
- **M** atomic mass
- **Q** ion charge state

Available beams

5.1) Combining Radiation Ground Testing with Fault Injection

Radiation test vs. Predicted measures for the 8051

Particle beam	Effectif LET [MeV/mg/cm²]	Error rate : [cm² / composant]	
		Measured	Predicted
Nitrogene (N)	2,97	$2,00\ 10^{-6}$	$2,00\ 10^{-6}$
Neon (Ne)	5,85	$1,02\ 10^{-4}$	$1,55\ 10^{-4}$
Chlorine (Cl)	12,7	$3,96\ 10^{-4}$	$3,78\ 10^{-4}$
Argon (Ar)	14,1	$4,50\ 10^{-4}$	$4,33\ 10^{-4}$
Cl (at 48°)	19,5	$6,63\ 10^{-4}$	$6,00\ 10^{-4}$
Cl (at 60°)	25,4	$7,13\ 10^{-4}$	$7,55\ 10^{-4}$
Krypton (Kr)	34	$9,12\ 10^{-4}$	$8,86\ 10^{-4}$
Bromine (Br)	40,7	$8,85\ 10^{-4}$	$9,00\ 10^{-4}$



Exposed Program: ***a 6x6 Matrix Multiplication***

C) Netlist fault injection: Application to a 8051

•Objective: Confrontation of predictions issued from NETFI to those issued from the CEU method

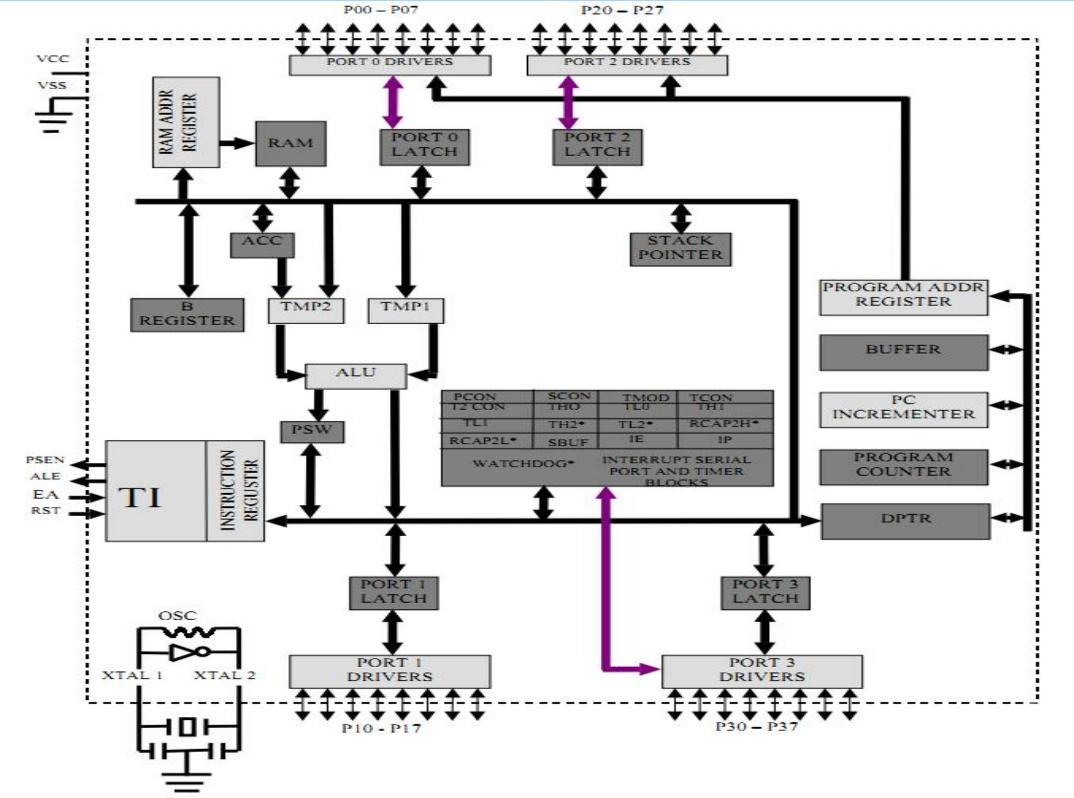
•SEU sensitive area:

- 128 bytes internal SRAM
- SFR registers
- Pipeline registers
- Program Counter
- Stack Pointer

Total : 1633 bits

• Program: 6x6 matrix multiplication

• Radiation test results are available



ARIS

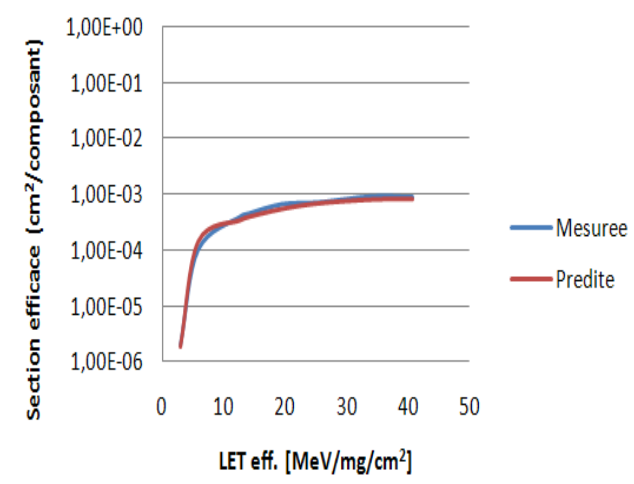
C) Netlist fault injection: Application to a 8051 (cnt'd)

NETFI vs. Radiation test data (tests done at UCL cyclotron)

- 51907 injected faults ➔ error rate: 47,09%

$$\tau_{SEU} = \sigma_{static} * \tau_{inj}$$

Ion	LET [MeV/mg/cm²]	Angle (degrés)	LET eff.	Static cross section [cm²/ device]	Dynamic cross section [cm²/composant]	
					Measured	Predicted
N	2.97	0	2.97	$4.30\ 10^{-6}$	$2.00\ 10^{-6}$	$2.03\ 10^{-6}$
Ne	5.85	0	5.85	$3.33\ 10^{-4}$	$1.02\ 10^{-4}$	$1.58\ 10^{-4}$
Cl	12.7	0	12.7	$8.12\ 10^{-4}$	$3.96\ 10^{-4}$	$3.84\ 10^{-4}$
Ar	14.1	0	14.1	$9.31\ 10^{-4}$	$4.50\ 10^{-4}$	$4.40\ 10^{-4}$
Cl	12.7	48°	19.5	$1.29\ 10^{-3}$	$6.63\ 10^{-4}$	$6.10\ 10^{-4}$
Cl	12.7	60°	25.4	$1.62\ 10^{-3}$	$7.13\ 10^{-4}$	$7.68\ 10^{-4}$
Kr	34	0	34	$1.90\ 10^{-3}$	$9.12\ 10^{-4}$	$9.00\ 10^{-4}$
Br	40.7	0	40.7	$1.94\ 10^{-3}$	$8.85\ 10^{-4}$	$9.16\ 10^{-4}$



- 2 SEU/Sec

ARIS

C) Netlist fault injection: Application to a 8051 (cnt'd)

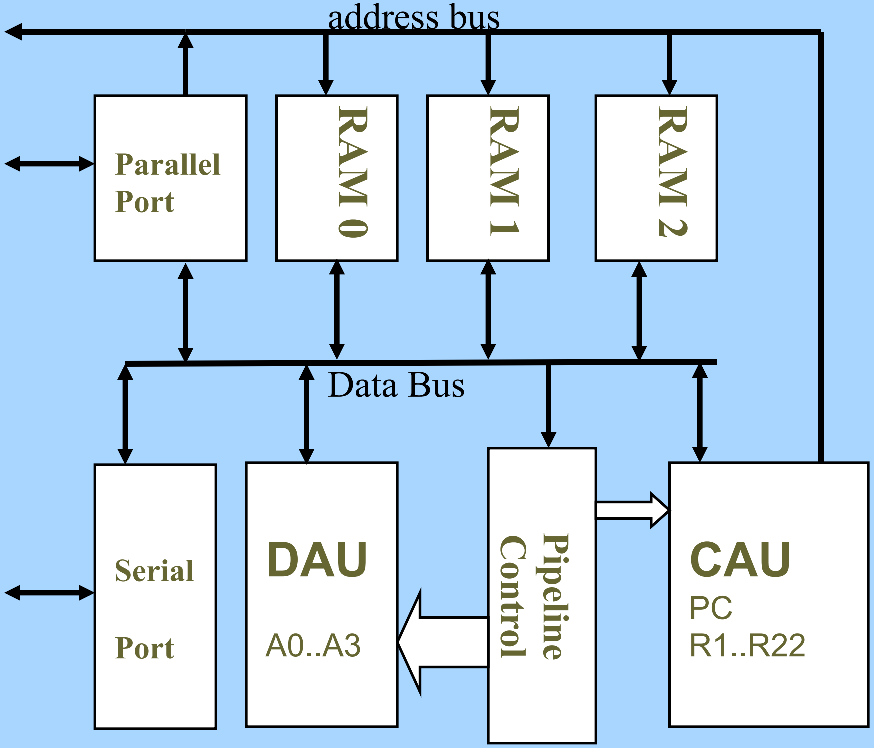
- NETFI estimation vs. CEU estimations
  - Accessible sensitive area :
    - 1633 memory cells accessible via NETFI
    - 1518 memory cells accessible via CEU (about 7% less)

ion	LET	Measured Cross Section	Cross Section predicted by CEU	Cross Section predicted by NETFI
N	2.97	2.00 10-6	$2.002\ 10^{-6}$	$2.03\ 10^{-6}$
Ne	5.85	1.02 10-4	$1.55\ 10^{-4}$	$1.58\ 10^{-4}$
Cl	12.7	3.96 10-4	$3.78\ 10^{-4}$	$3.84\ 10^{-4}$
Ar	14.1	4.50 10-4	$4.33\ 10^{-4}$	$4.40\ 10^{-4}$
Cl	12.7	6.63 10-4	$6.01\ 10^{-4}$	$6.10\ 10^{-4}$
Cl	12.7	7.13 10-4	$7.56\ 10^{-4}$	$7.68\ 10^{-4}$
Kr	34	9.12 10-4	$8.86\ 10^{-4}$	$9.00\ 10^{-4}$
Br	40.7	8.85 10-4	$9.01\ 10^{-4}$	$9.16\ 10^{-4}$

It is important to note that the NETFI approach allows exploring the SEU time/location in a better way than particle accelerators.

5.2) Combining Radiation Ground Testing with Fault Injection
Second case study: the DSP32C

- RAMs: 2KB
- Address bus: 3 Bytes
- Data bus: 4 Bytes
- CAU: Arithmetic, Logic operations & program flow control
- DAU: Floating point operations. Four stages pipelined
- ~50.000 bits in DSP



5.2) Combining Radiation Ground Testing with Fault Injection
DSP32C Target program

- **Constant Modulus Equalizer** (1280 bytes of code, 1381598 clock cycles, 133 float inputs)

- <u>Bit flip target area</u>:
- RAM0 (Output array)
- RAM1 (Global Variables, Input array)
- RAM2 (Stack)
- Registers (Rx, PC, Ax, …)

<u>Non perturbed area</u>:

- Code in the External Memory

5.2) Combining Radiation Ground Testing with Fault Injection **Upset simulation results of for the the DSP32C**

	#injected	CMA error	LS error	Halted	Total errors
RAM0	16253	1571	0	0	1571
RAM1	16262	3144	0	0	3144
RAM2	16617	85	8	0	93
Ax registers	120	!!! 0	0	0	0
Rx	572	37	32	0	69
PC	15	6	4	0	10
Other registers	249	0	0	1	1
TOTAL	50088	4843	44	1	4888

$\tau_{inj} = 0.097$ **errors / upset**

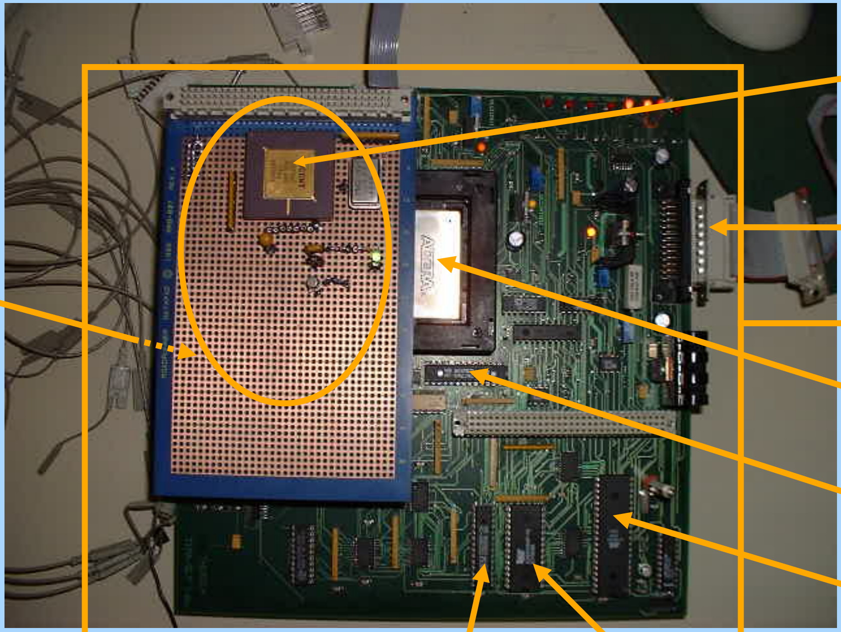
ARIS

5.2) Combining Radiation Ground Testing with Fault Injection
Radiation ground test for the DSP32C

- Vacuum chamber
  - Californium 252
  - LET: 20 - 40 MeV
  - Flux: ~ 280 Particles/s
- PC with Terminal Interface
- THESIC+ test system
- DSP32C daughterboard

5.2) Combining Radiation Ground Testing with Fault Injection
Radiation ground test experimental set up

The THESIC^{+} tester



EEPROM's for DSP32C programs

daughterboard
with DSP32C & clock

Connection to PC

motherboard

FPGA

MMI (shared memory)

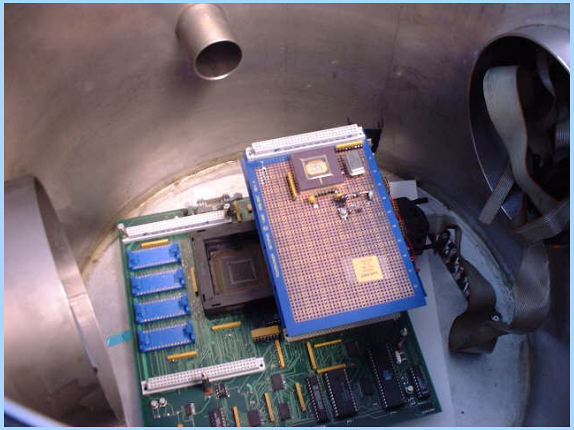
80c51

External RAM for the 80c51

EEPROM for 80c51 programs

5.2) Combining Radiation Ground Testing with Fault Injection
Predicted vs. Measured Error rates for the DSP32C

- SEU static cross-section



σ_{SEU} =2.7*10^{-3} upsets/particle

- Upset injection session

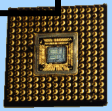
τ_{inj} = 0.097 errors/upset

- Predicted Error rate : **$\tau_{SEU} = \sigma_{SEU} * \tau_{inj} = 2.6*10^{-4}$ errors/particle**
- Measured Error rate : **$\tau_{SEU} = 3.38*10^{-4}$ errors/particle**

ARIS

5.3) Combining Radiation Ground Testing with Fault Injection
Case study 3: A complex processor The PPC4778

	PC7447A	PC7448
Architecture	32-bit implementation of the PowerPC® RISC architecture (G4) Full 128-bit implementation of Freescale AltiVec technology	
Technology	SOI 130 nm - 9 layers metal	SOI 90 nm - 9 layers metal
Transistor count	48.6 millions	90 millions
Core power supply	1.3V ± 50 mV or 1.1V ± 50 mV	1.1V ± 50 mV or 1.0V ± 50 mV
I/O power supply	1.8V ± 5% or 2.5V ± 5%	1.5V ± 5% or 1.8V ± 5% or 2.5V ± 5%
Integrated L1	2x32KB instruction and data caches with parity support	
Integrated L2	512 KB with parity support	1 MB with parity and ECC support
Registers	32 General Purpose Registers (GPR) of 32-bit each 32 Floating Point Registers (FPR) of 64-bit each 32 Vector Registers (VR) of 128-bit each	
Operating Frequency	1.167 GHz for the core 166 MHz for memory bus	1.4 GHz for the core 200 MHz for memory bus



5.3) Combining Radiation Ground Testing with Fault Injection
Case study 3: Objectives

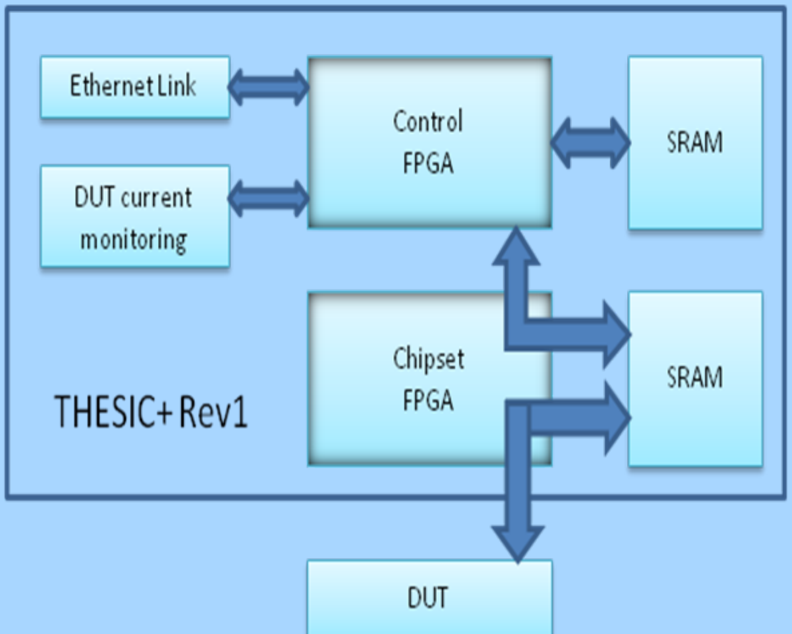
- **1- Heavy Ions tests on accelerator:**
  - To determine static cross sections of microprocessors
  - Dynamic cross section using a real space application running on PC7448

- **3- Fault Injection Session on PC7448**
  - Based on static cross sections
  - Calculation of dynamic cross sections of the application
  - Comparison with test results on accelerator

5.3) Combining Radiation Ground Testing with Fault Injection
Case study 3: Test platform for the PPC4778

Used test platform: The THESIC+ tester (see ref. [2])

Heavy ion tests done at HIF de l'UCL



[2] F. Faure, P. Peronnard, and R. Velazco, Thesic+: A flexible system for SEE testing, *Proc. of RADECS*, 2002.

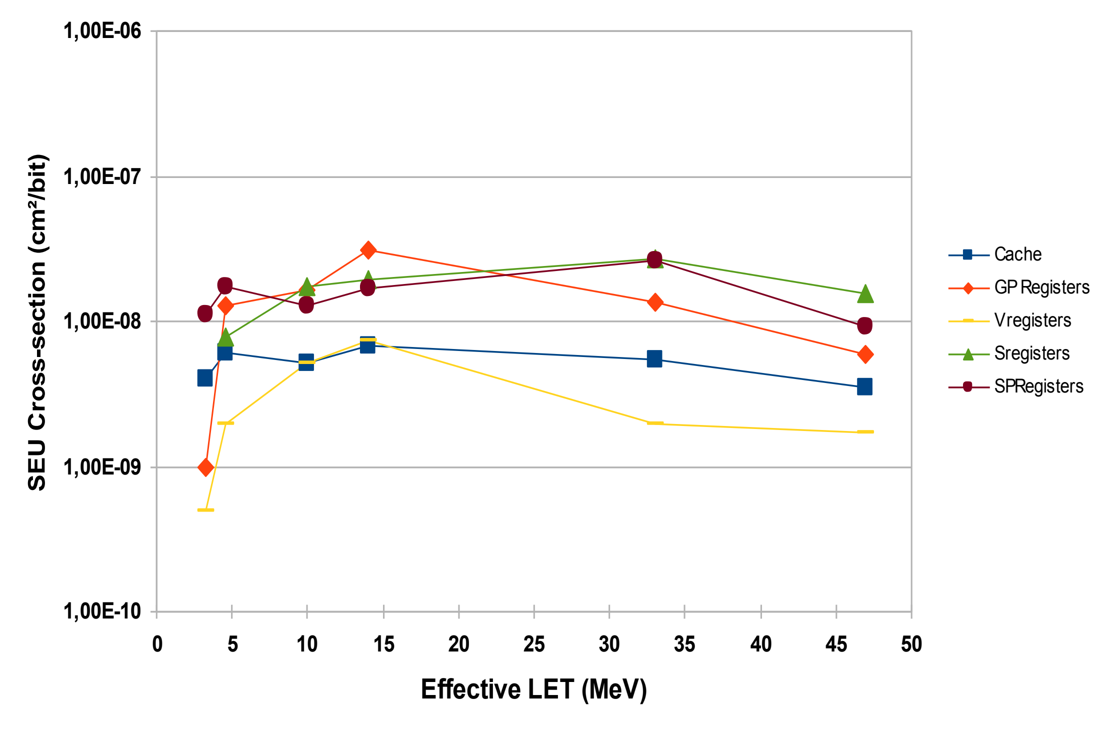
5.3) Combining Radiation Ground Testing with Fault Injection
Case study 3: test conditions and SEU targets

- Frequency fo the core: 600MHz
- Memory bus frequency: 40MHz
- Data cache L1 : 32 KB = 262144 bits
- Instruction cache L1 : 32 KB = 262144 bits
- Registers (8864 bits)

General purpose registers: 32*32 = 1024 bits
Vectorial calcularion registers: 32*128 = 4096 bits
Status registers: 16*32 = 512 bits
SP registers : 8*32 = 256 bits
Virtual memory registers = 512 bits
Floatting point registers : 32*64 = 2048 bits
ICTRL (cache memory control registers) = 32 bits
Registers for fault injection in cache memory L2 (14 registers)

5.3) Combining Radiation Ground Testing with Fault Injection
Case study 3: Static tests Heavy ion results

- **PC7448 heavy Ions Test Results:**
  - **No Latchup confirmed (SOI process)**
  - **No Multi Bit Upset (MBU) observed**
  - **Cache saturation cross-section = 6.88 E-09 cm²/bit**
  - **LETth ~ 2 MeV**

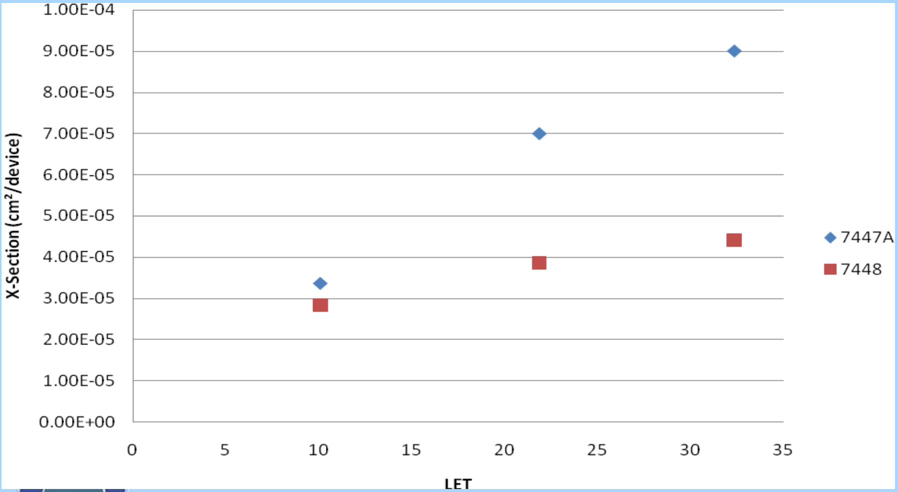


ARIS

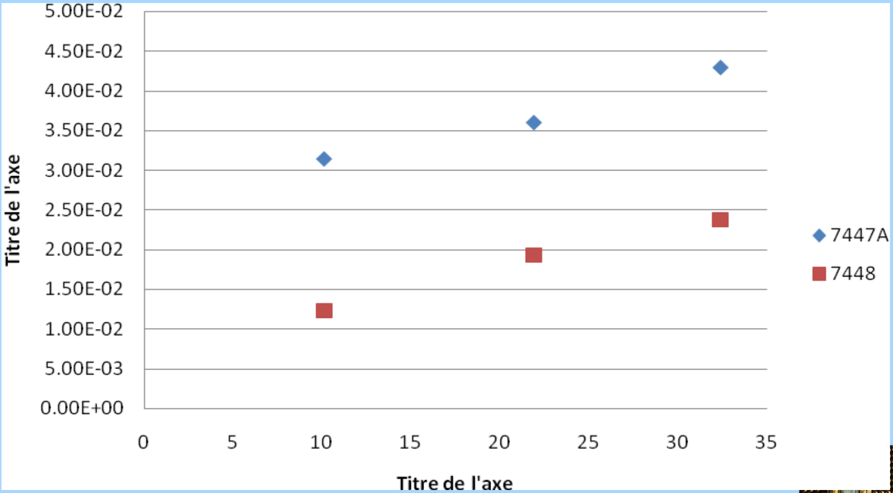
5.3) Combining Radiation Ground Testing with Fault Injection
Case study 3: SEU static cross sections

- **PC7447A vs PC7448**
  - **PC7448 Cross-sections are two times lower than PC7447A in spite of smaller process geometry (90 nm vs 130 nm)**

Registers (1 kB)



Data Cache L1 (32 kB)

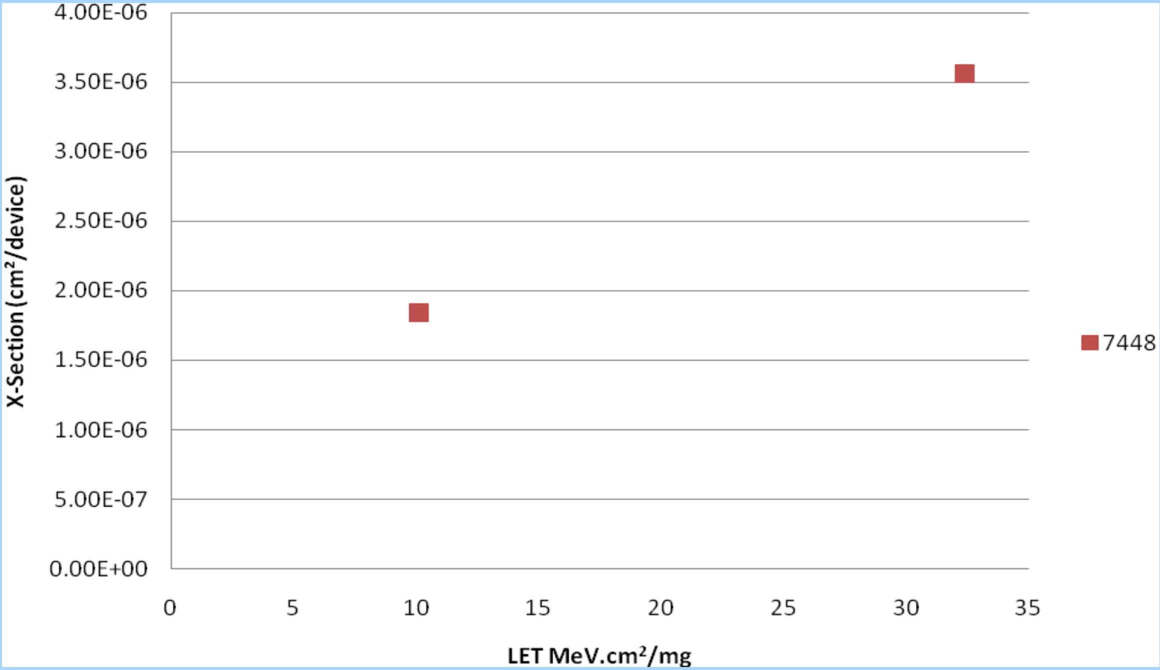


5.3) Combining Radiation Ground Testing with Fault Injection
Case study 3: SEU dynamic cross sections

- **Dynamic cross section of a real space application provided by CNES:**

ACS = software devoted to the Attitude Control of a Satellite

- **Dynamic cross section is really lower compared to static cross section:**
  - **Factor 10 compared to Registers**
  - **Factor 10 000 compared to data cache**



ARIS

5.3) Combining Radiation Ground Testing with Fault Injection

Case study 3: Error rates predicted from Fault injection

- **CEU method enables to simulate a high number of SEU**
  - **injecting 150 000 SEUs on PC 7448 registers required 2 full days**
  - **such an experiment would require 4 days of accelerator beam Cost would be exorbitant !**
- **Predicted vs. measured error rates for ACS application**

Ion	LET_{eff} Mev/mg/cm²	τ_{SEU} Predicted	τ_{SEU} Measured
Argon	10.1	2.12E-05	2.04E-05
Krypton	32.4	3.24E-05	3.17E-05

->Predictions fit very well with measurement

->No need to redo accelerator tests in case of modifications of the application

5.4) Combining Radiation Ground Testing with Fault Injection
Case study 3:Conclusions

- The predicted applications error-rates were very close to measures issued from radiation ground testing
- Absence of latchup confirmed for SOI technology
- Low sensitivity to heavy ions allows using PPC7448 for space applications where SEUs may be tolerated or mitigated.
- According to CNES, Pwer PC processors appears as good candidates to space applications requiring high calculation power.
- The low FIT of these circuits allow using them in critical avionic applications : PowerPC 7448 was selected for the on-board computer l'A350.

5.4) Combining Radiation Ground Testing with Fault Injection
Case study 4: SRAM-based FPGA

- SRAM-based FPGAs are attractive for space & avionics applications
  - low cost, high performance, fast development and on-site reconfiguration
- SRAM-based FPGAs are sensitive to radiations which can provoke:
  - Errors in the application itself
  - Errors in the configuration memory => mutation of the application
- TMR mitigation technique for SRAM-Based FPGAs has potential weaknesses
- A cooperation with NASA offered a possibility to include an experimental board in the LWS-SET project

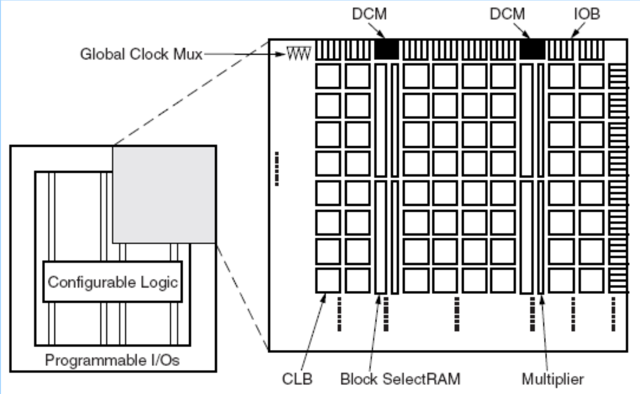
The goals of this work are:

- Validate for FPGAs a state-of-the-art error-rate prediction approach.
- Obtain preliminary experimental results about sensitivity of a TMR application using accelerated tests and HW/SW fault injections.
- Confront measures to predictions.

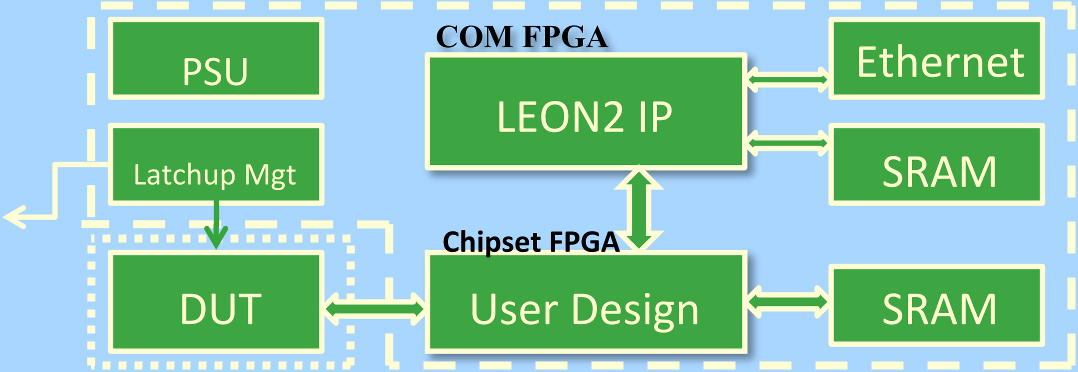
5.4) Combining Radiation Ground Testing with Fault Injection
Case study 4: SRAM-based FPGA

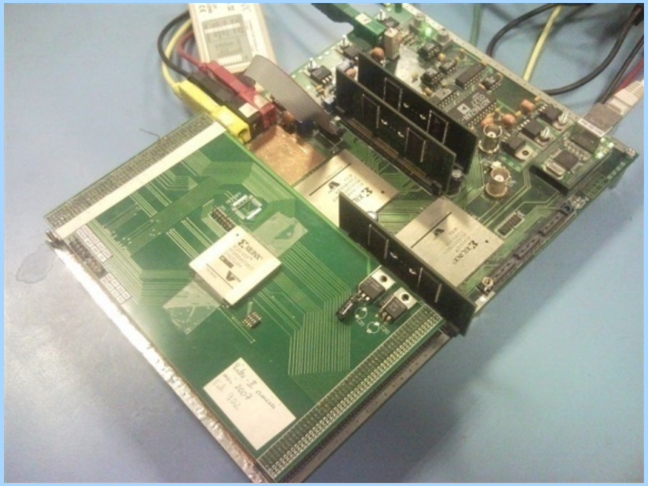
- **Xilinx Virtex-II XC2V1000-FF896**
- **0.25µm technology**
- **896 pin Flip-Chip package**
- **8 Digital Clock Manager (DCM)**
- **40 18bit x 18bit Multiplier blocks**
- **40 18Kbit SelectRAM blocks**
- **40 x 32 Configurable Logic Block (CLB) matrix**
- **432 available user I/O pins**

- **Configuration memory size : ~4Mbits**



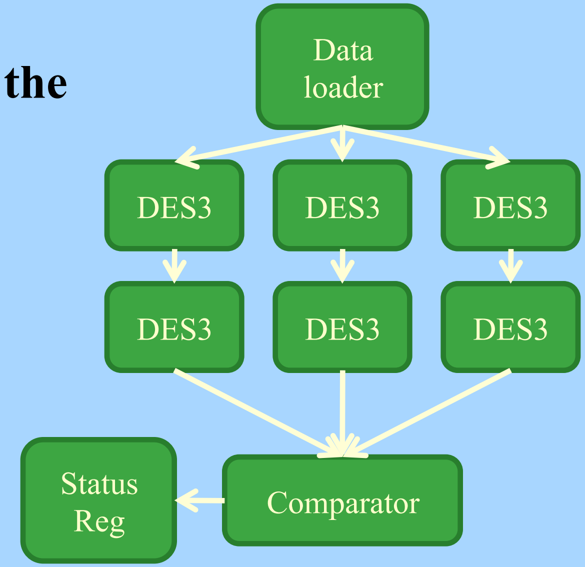
5.4) Combining Radiation Ground Testing with Fault Injection
Case study 4: Test platform THESIC+





5.4) Combining Radiation Ground Testing with Fault Injection
Case study 4: Implemented application

- **Tested application: 64-bit triple-DES (DES3) cryptocore**
- **Two DES3 are chained in order to maximize the resources used in the FPGA**
- **TMR application uses 75% of the slices**
- **A 3-bit Status Register (SR) provides information on the TMR branches behavior**



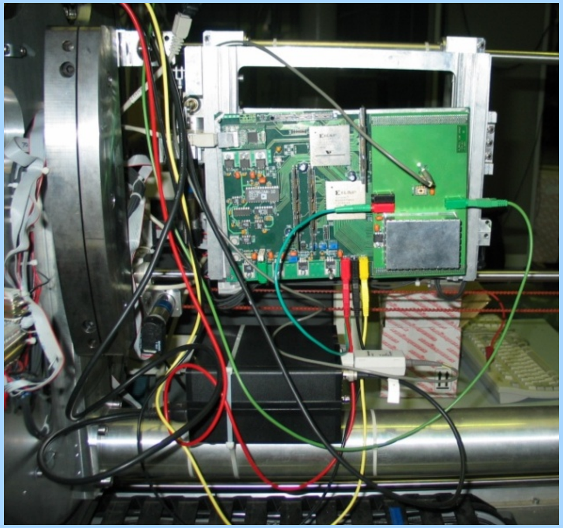
- **Status Register values:**
  - **"0" : same result on the three branches**
  - **"1", "2", "3" : branch number giving a different result**
  - **"4": three different results**
  - **"5", "6", "7" : N/A**
- **THESIC+ is used as an external comparator in order to confirm the efficiency of the error detection implemented in the applications.**

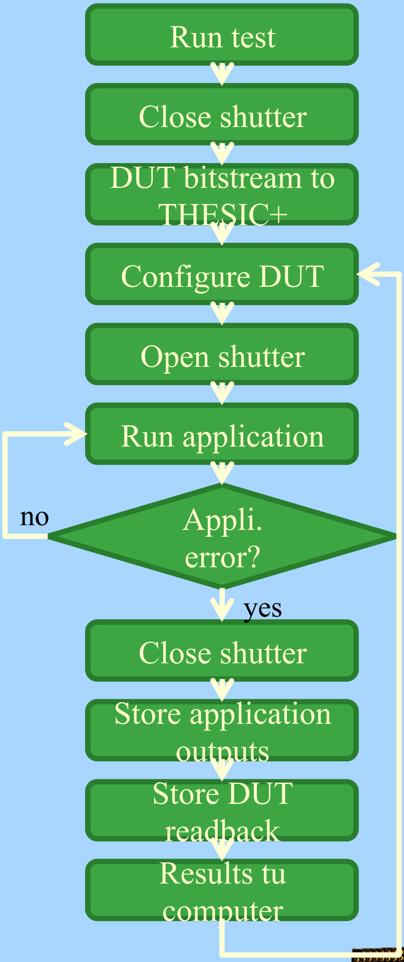
5.4) Combining Radiation Ground Testing with Fault Injection
Case study 4: Radiation ground testing

<u>Facility</u>: HIF (Heavy-Ion Facility) Louvain-la-Neuve (Belgium)

<u>Selected particles</u>: Carbon & Argon

<u>Shutter</u>: mechanical device to prevent the particle beam from hitting the DUT





Types of observed errors:

- **<u>Detected error</u>: the Status Register detects an error on one off the branches, but the TMR is able to correct it**
- **<u>Falsely detected error</u>: the SR reports a N/A value although the application result is correct**
- **<u>Undetected error</u>: the SR does not report any error, however the application result is false.**

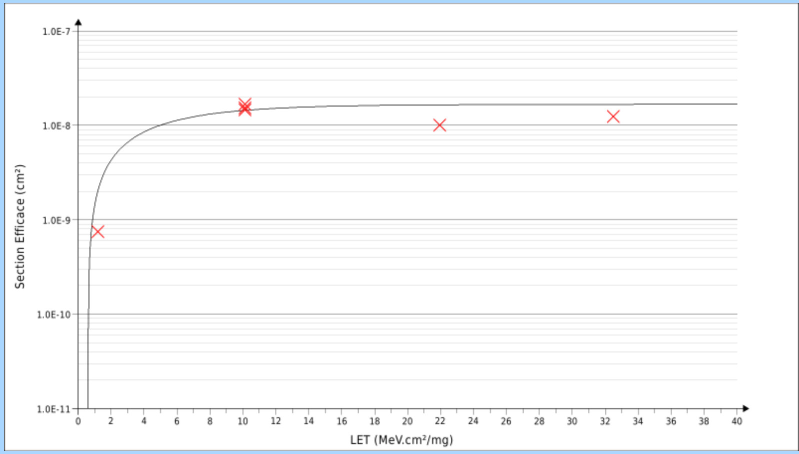
ARIS

5.4) Combining Radiation Ground Testing with Fault Injection
Case study 4: Radiation ground testing (cont'd)

Particles	LET (MeV/mg/cm²)	Detected errors	Falsely detected errors	Undetected errors	Nb. of application runs	Total fluency
Carbon	1.2	51	0	0	187,469,275	158,543
Argon	10.1	1,278	3	35	750,688,226	437,095

- **The DUT is less sensitive to Carbon than to Argon.**
- **Carbon: too few results to observe all types of errors.**
- **Argon: all types of errors observed.**

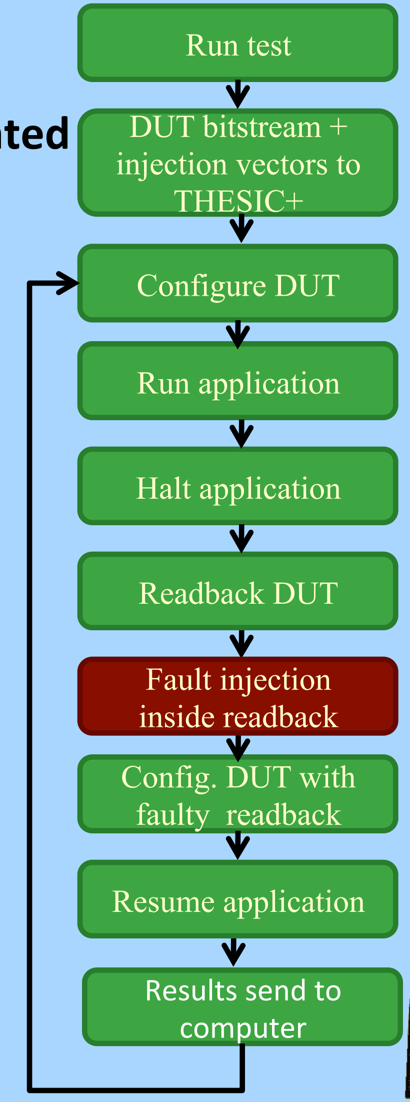
DUT static cross-section



ARIS

5.4) Combining Radiation Ground Testing with Fault Injection
Case study 4: Fault injection results

- **Injection parameters (time & location) are randomly generated**
- **Faults are injected directly in the configuration bitstream**
- **1 fault injection takes less than 1 second**
- **Number of injected faults: 426,217**



	Detected errors	Falsely detected errors	Undetected errors
# application errors	14,564 (3.41 %)	237 (0.06 %)	319 (0.07 %)
Average injections to provoke an application error	3.4×10^{-2}	5.6×10^{-4}	7.5×10^{-4}

ARIS

5.4) Combining Radiation Ground Testing with Fault Injection
Case study 4: Error rate Prediction vs. Measures

Confrontation of the measures obtained in particle accelerators and predictions made from fault injection for the TMR application

Particles	Error rate	Detected errors	Falsely detected errors	Critical errors
Carbon	Measured	1.0×10^{-4}	N/A	N/A
	Predicted	9.5×10^{-5}	1.55×10^{-6}	2.1×10^{-6}
Argon	Measured	2.8×10^{-3}	6.7×10^{-6}	7.6×10^{-5}
	Predicted	1.9×10^{-3}	3.2×10^{-5}	4.2×10^{-5}

Predictions underestimate measures by a factor 1.1 and 1.5

Measures underestimate Predictions by a factor less than 2.1

Predictions underestimate measures by a factor 1.8

- Error rates predicted are close to measure issued from particle accelerators measures.
- Differences could be explained by:
  - ❑ The little number of observed events during heavy-ion campaigns
  - ❑ Fault injection is not able to generate MBUs as this would require the knowledge of the FPGA's layout in order to generate realistic fault injection parameters

5.4) Combining Radiation Ground Testing with Fault Injection
Case study 4: Final conclusions

- Obtained results shows that SEU in the configuration memory may provoke a *mutation* of the application preventing the comparator to detect the fault.

- Measures obtained from a particle accelerator and predictions calculated from fault injections differ by a maximum ratio of 2.

Perspectives:

- To be able to simulate MBUs in the configuration memory.

- The ultimate goal is to compare measures obtained from real life experiment with measures and predictions presented in this work. The LWS-SET satellite launch is planned for October 2015.

6. Conclusions and Future Work

☺ • The proposed SEU fault injection approach can be automated for processor and FPGA based architectures

☺ • Good experimental results

☺ • Short duration of injection sessions (i.e. 1000 upsets / 3 min.)

☺ • Generic approach

☹ • Not all targets are accessible through the instruction set

☹ • Need to build a hardware prototype increases developing time and cost => use of a generic test platform!

6. Conclusions and Future Work

- SEU error-rates in processor-based architectures can be accurately predicted from automated upset injection sessions and SEU static cross-sections issued from radiation ground testing.

- Using HW-based approaches to inject upsets:
  - needs only the device data sheet
  - is a generic approach
  - short duration of injection experiments (a few minutes/10000 upset)

  but

  - needs a hardware prototype

- Using SW based approaches to inject upsets:
  - decreases implementation cost and complexity
  - allows the study of the processor critical areas

  but

  - needs a software simulator and a processor HDL model
  - entails long simulation times (a few seconds / upset)

6. Conclusions and Future Work

- Deep study of a fault injection approach using VHDL models.
- Applying the different SEU-injection approaches to the same processors and sets of programs.
- Design automated tools devoted to set up both radiation ground testing experiments and fault injection sessions.
- Validate SEU-rate predictions for FPGA based applications by confronting the results issued from radiation ground testing to those measured on orbit: LWS/SET project.

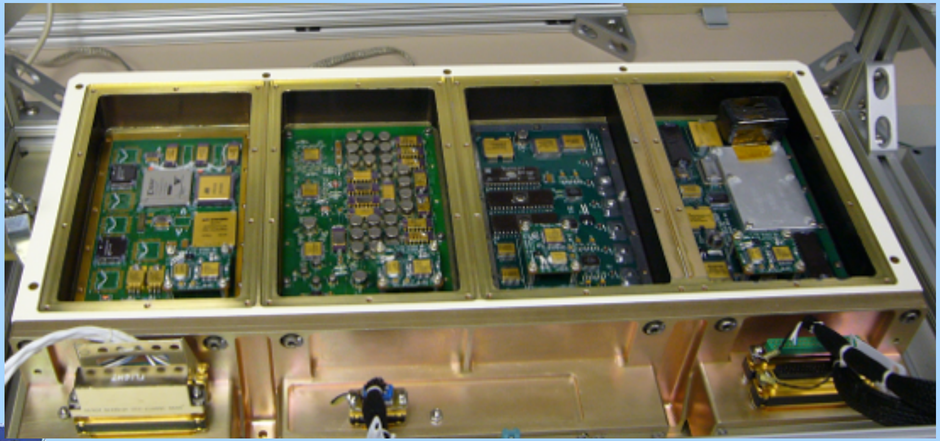
6. Conclusions and Future Work

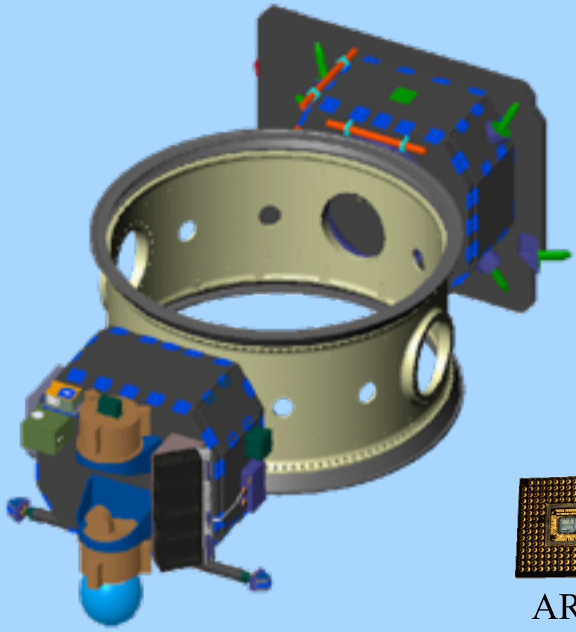
Living With a Star (LWS) aims at studying:

- **the solar activity.**
- **the impact of this activity on the Earth and on life.**

Space Environment Testbeds (SET) is the part of LWS project devoted to characterize the space environment and its impact on integrated circuits and system reliability in space.

=> A board including the DES3 implemented in and FPGA Virtex II is one of the hosted experimental boards.





Acknowledgements

e2v semiconductors

Dominique Bellin

Francoise Bezerra
Robert Ecoffet

Guy Berger

Ken Label
Michael Xapsos
Carolyn Mariano