FDTC 2005

Injection of Multiple Bit-Flips for Counter Measures Validation

Karim Hadjiat, Abdelaziz Ammari, Régis Leveugle*

TIMA Laboratory – Grenoble –FRANCE

*** Partly supported by the DURACELL and VENUS Projects**

Motivation

- Recent threats related to malicious fault injections in circuits (fault-based attacks)

- Need for early analyses to evaluate the criticality of faults in the various parts of a circuit
  => Identify the real locations to be protected
  in an application-specific circuit (e. g. cryptographic IP)
  => Minimal hardware counter measures with respect to the application requirements (focus on real security assets)

- Differences between paradigms: impact on tools and fault models (cf. FDTC'04)

Overview

- Fault modeling: Multiple Bit-Flips in fault-based attacks
- Early fault effect analysis: methods and tools
- A new type of mutants for multiple bit-flips
- Experimental results
- Conclusions and perspectives

Evolution of needs in fault injection tools

- Initially was the Single Event Upset in space …
  => single bit flip modeling usually considered as accurate

- Then came Very Deep Submicron CMOS technologies
  - Smaller geometries / node capacitances
  - Lower voltages
  - Higher clock frequencies
  - Lower noise margins / increasing noise level

  → Single Event Transients

  - Decreasing TTM
  - Dependability concerns in "consumer" electronics

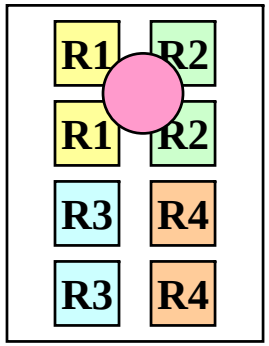
  → Early analyses (avoid long feedback, reduce costs)

And also …

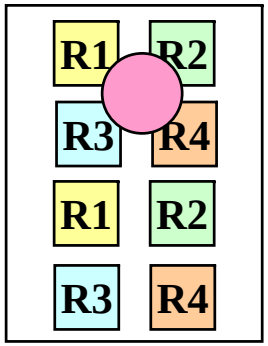
- New security threats: fault attacks
  - Cryptography primitives: DES / RSA / AES …
  - Security locks (ratification counters, …)
  - …
- Various possibilities
  - Power glitch
  - Flash light
  - Laser
  - …
- Ultimately: logic fault(s)
  - In flip-flop(s) => similar to SEU ?
  - In combinatorial logic => similar to SET ?

SEU or multiple bit-flips ??

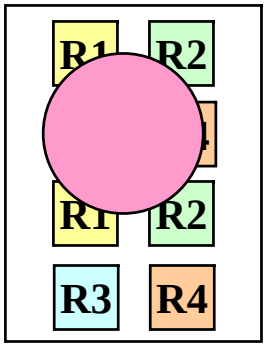
- Spatial multiplicity ("MBF") – laser attack
- Depends on
  - Laser focus
  - Placement/routing
  - Cell sensitivity
  - …



P&R-1, focus 1
=> mult. up to
2 per element,
2 elements



P&R-2, focus 1
=> mult. up to
1 per element,
4 elements



P&R-2, focus 2
=> mult. up to
2 per element,
4 elements

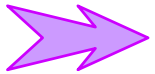
- High-level analysis: no information on P&R
  => assumptions / limitations (e.g. limited to the elements in a given register), but gives constraints on P&R for coherence

SET propagation

- A single transient can reach several outputs of the block during the latching window

=> Multiple bit errors can be expected

- The exact evaluation requires very low level data (after P&R)
  - Logic masking ?
  - Attenuation/suppression (electrical masking) ?
  - Out of latching window (temporal masking) ?



No effect …

=> Exact evaluation of effects not compatible with early evaluation

Solution: injection of multiple bit errors …

Conclusion on fault models

- Fault injection environments used for early dependability analysis can no more rely only on the classical single bit flip fault model, especially in the case of intentional faults
- Extension to multiple bit-flips (MBFs) is required and must be automated
- This work presents such an extension

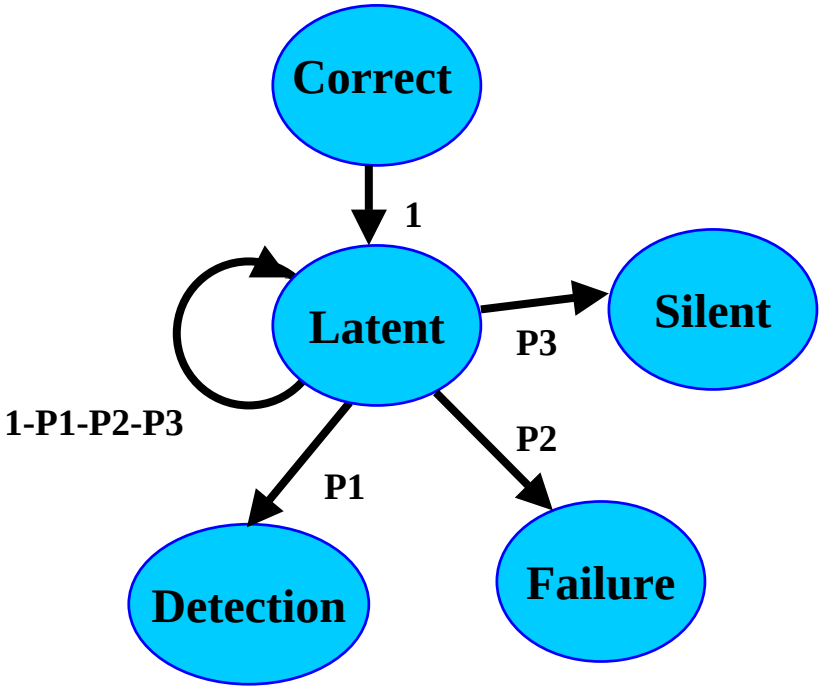
Goals of our analysis environment

- Early dependability analysis
  - RT-Level descriptions
  - Only potential knowledge of synthesis-related information (e.g. state assignment or specific synthesis procedures limiting the possible fault effects)
- Automated
- Compatible with classical up-to-date industrial design flows
- Qualitative/quantitative data usable for field failure rate prediction (representative of actual faults)
- Injection process compatible with both simulation and emulation

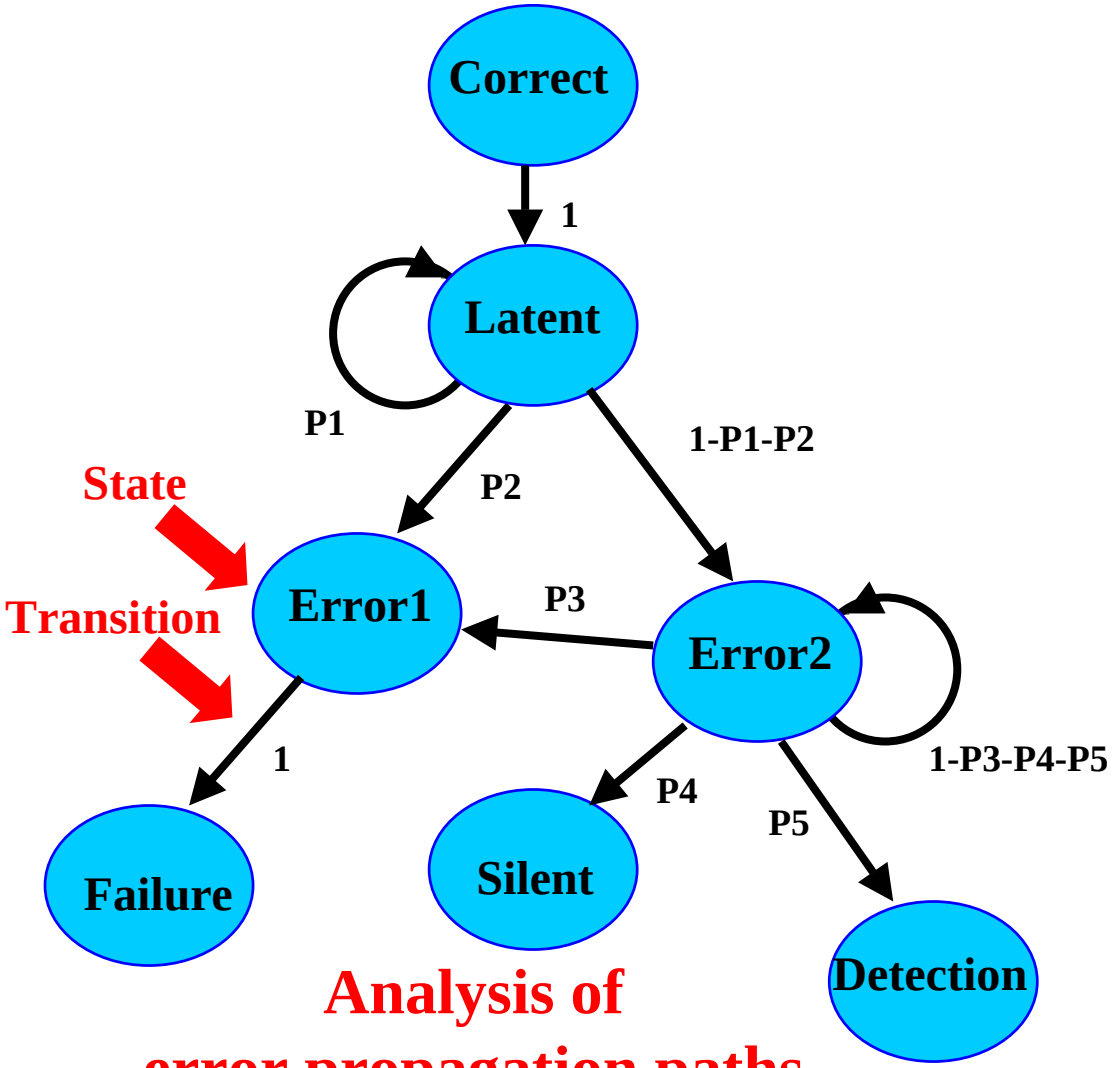
  => additional constraints

Dependability analyses: alternative results

Cycle-by-cycle comparisons

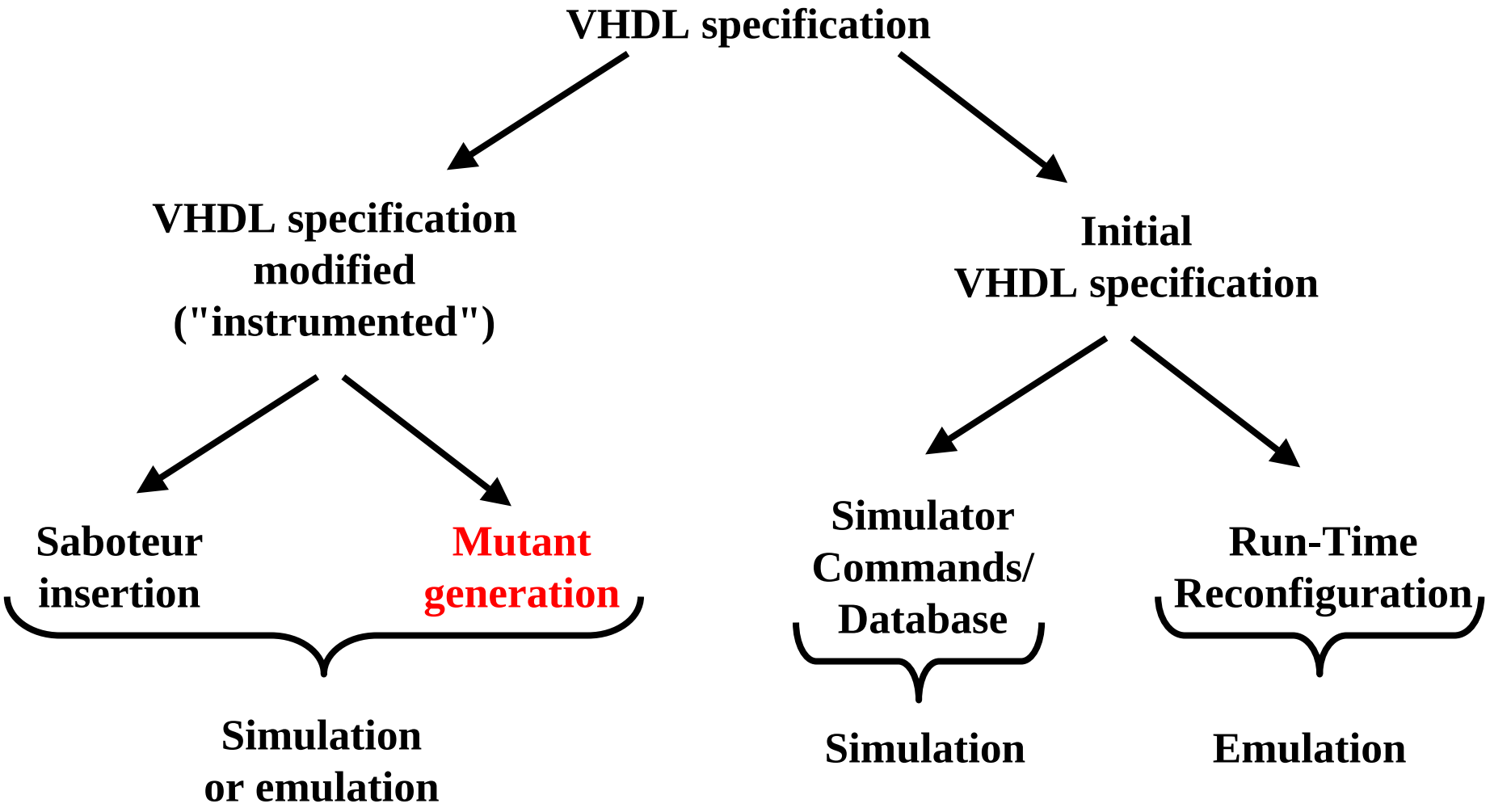


Fault classification



Analysis of error propagation paths

Alternatives for fault injection campaigns



Controlled generation of mutants

- Classical software-like mutants do not allow the targeted analyses

- "Controlled generation" of mutants implies:
  - Significant faulty behaviors
    (equivalent to the fault effects observable in the field)
  - Optimization for synthesis (compatibility with emulation)
  - Taking into account the limitations of hardware emulation systems

- Criteria for quality evaluation:
  - Number of additional I/Os (number of sub-campaigns)
  - Number of gates after synthesis (emulation hardware complexity)
  - Maximum frequency (time required for the injection campaign)

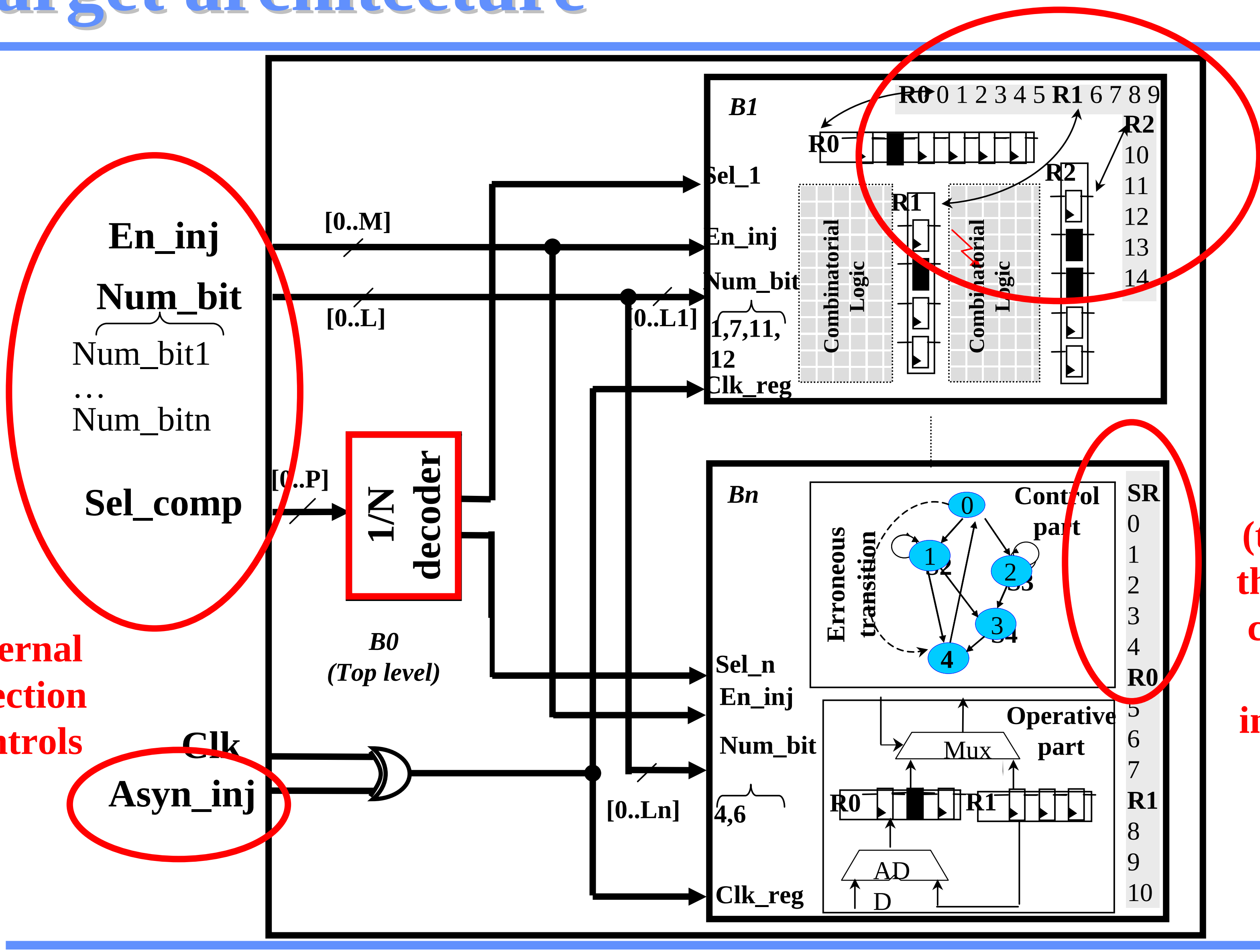
New mutant generation

- Extension of previous work
  - Multiple bit flips
  - Heterogeneous fault/error injection
    (single bit-flip, multiple bit-flip, erroneous transitions)

- Restrictions on multiplicity
  - Maximum value
  - Localization
    - No restriction (all elements selected at all time as targets)
    - Limited to a sub-block (selection of the target block)
    - Limited to a register (selection of the target register)

Limitation or not to a sub-block can give the same results in case of architectures with error confinement and optimized P&R

Target architecture



Virtual Register (targets for multiple bit injections)

Erroneous Transitions (target selection with virtual register)

Optional Decoder (control signals inputs and injection protocol)

External Injection Controls

K. Hadjiat
A. Ammari
R. Leveugle

VHDL modifications

- **Modifications of entities (external signals), hierarchy and processes (combinatorial and sequential)**
  + creation of the virtual register(s) and clock control

- **Automated for a limited synthesizable description template**

- **Trade-offs between generality and complexity: two options currently implemented (register level, sub-block level)**

Experiments

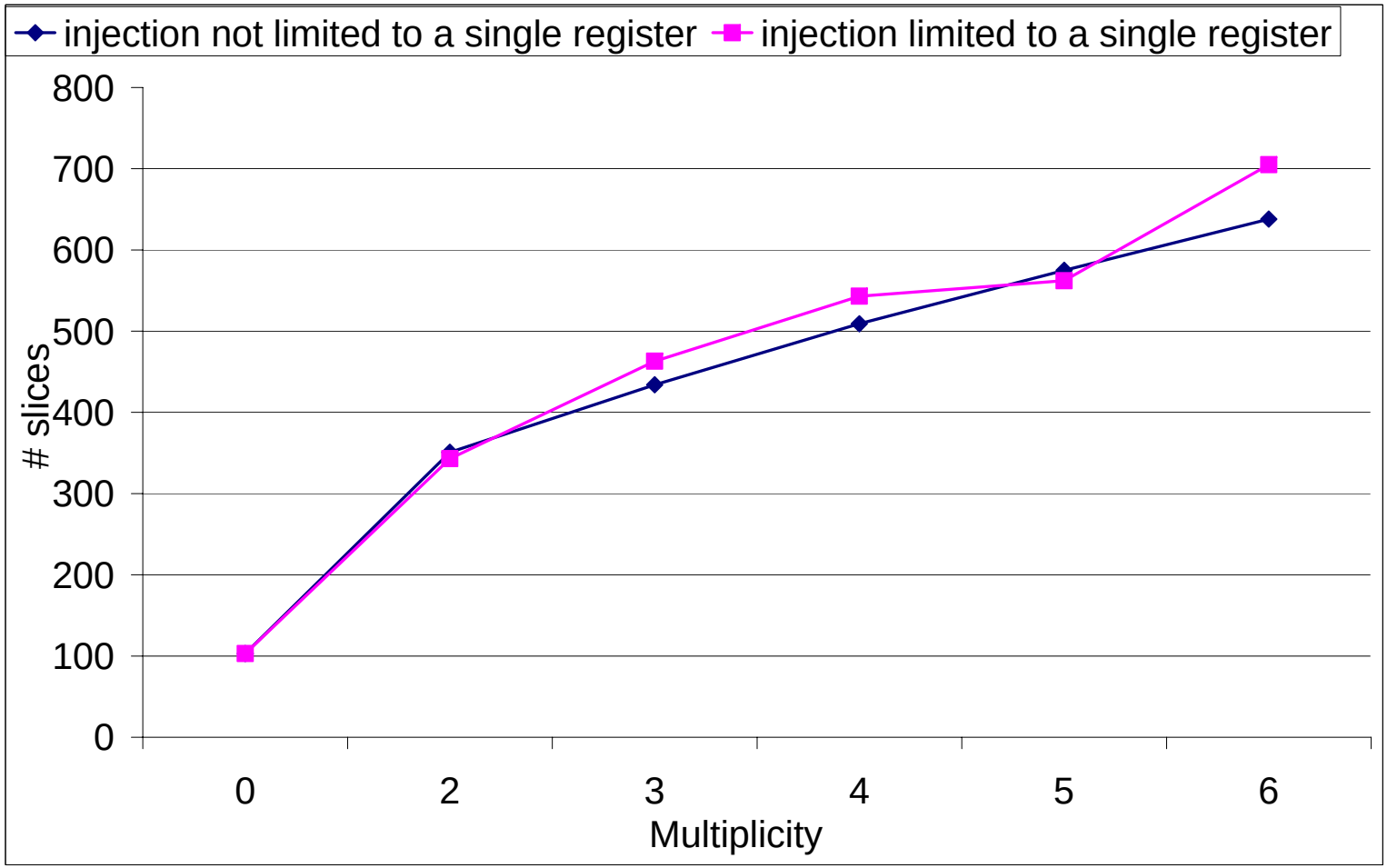
- Case study: core performing modular multiplications for the computation of RSA encryption (Montgomery)

- Two versions: initial and hardened (based on parity per 32-bit word)

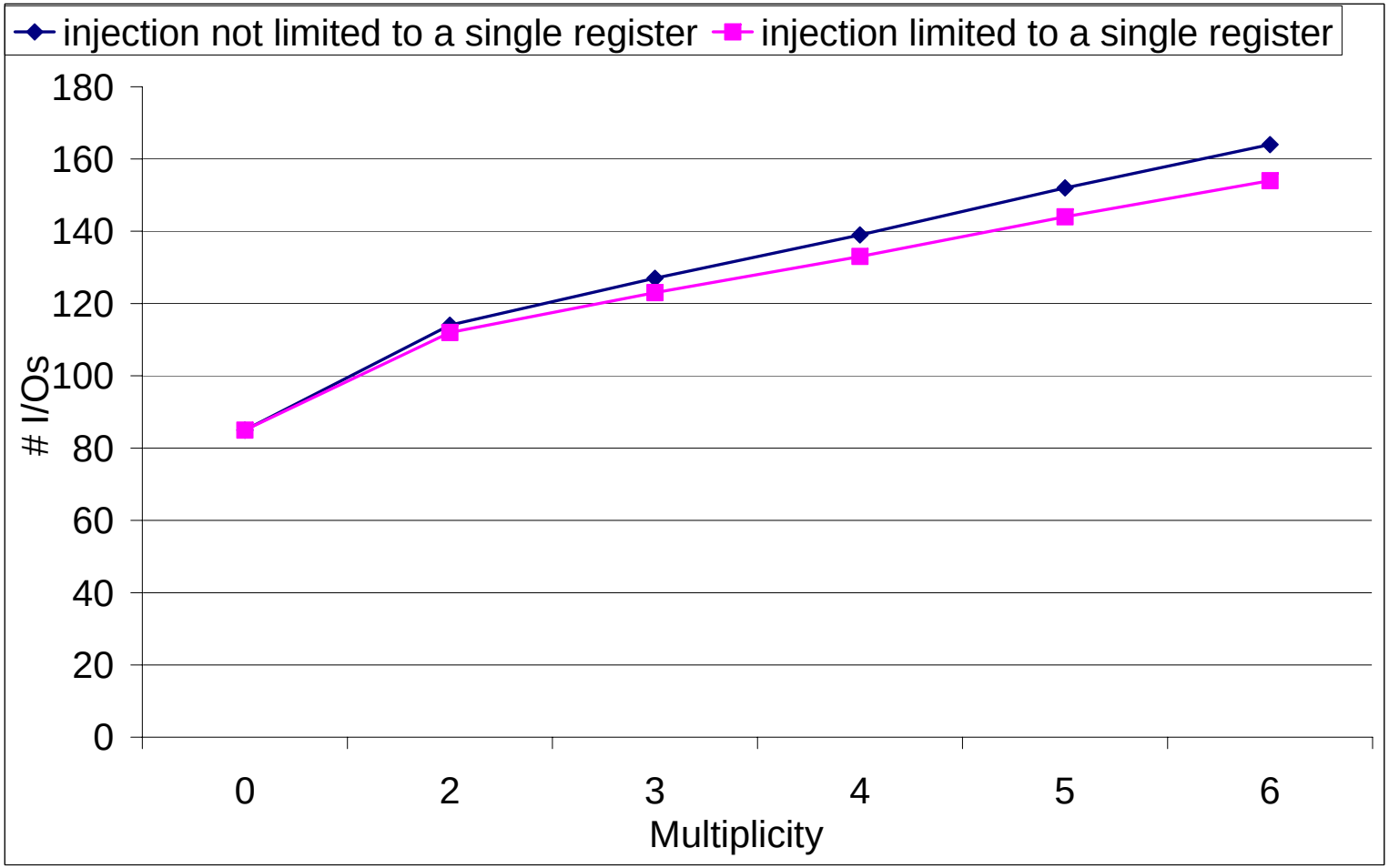
- Goals
  - Complexity analysis of the generated mutants
  - Analysis of the erroneous configurations of a set of internal/external signals and of the sequences of activations ("states" and "transitions" of the error propagation graph, detection is a terminal state)

    Injections not limited to a single register (more complex case for analytical analysis)

Results: complexity (Montgomery, Virtex II)

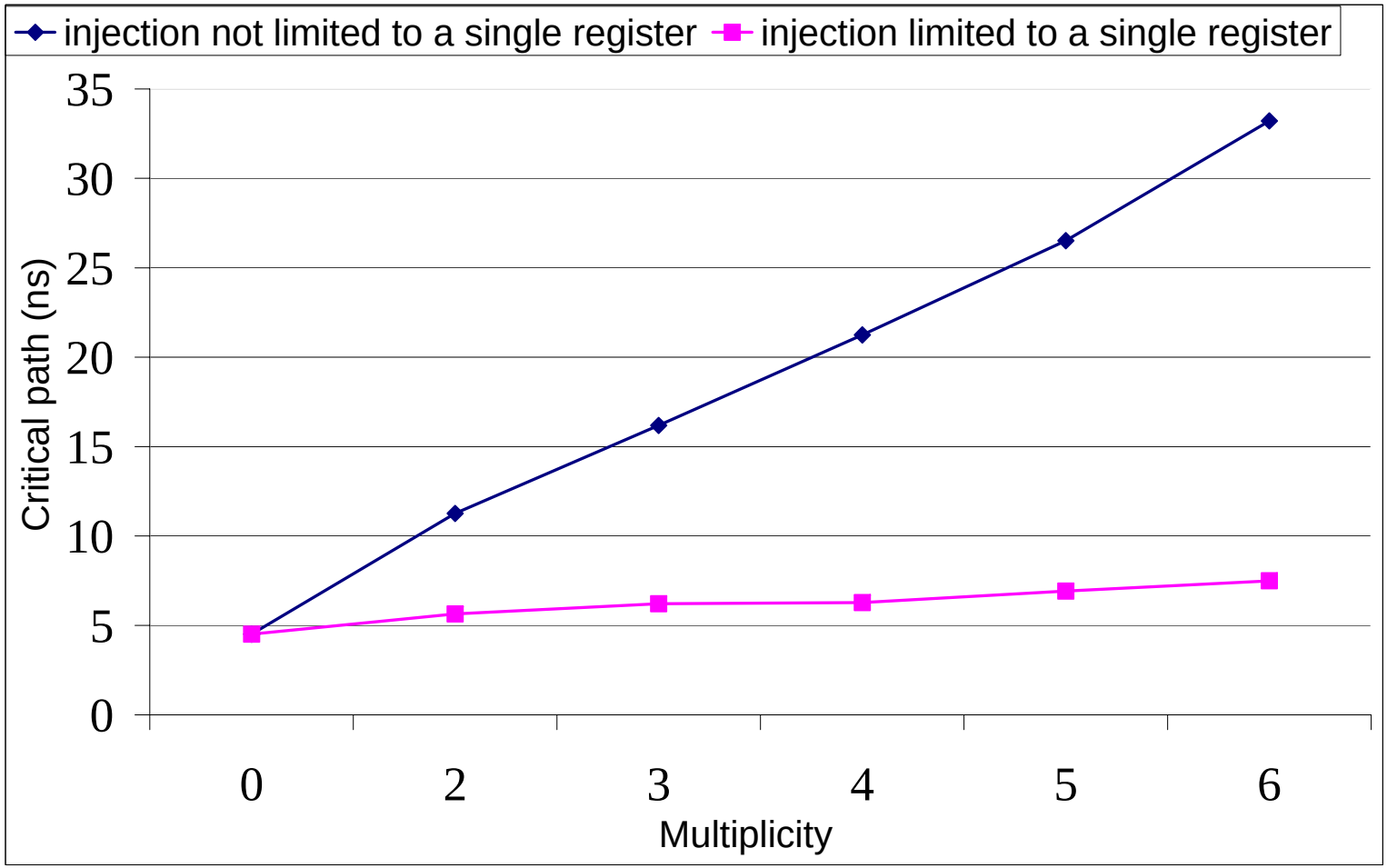


Results: I/Os (Montgomery, Virtex II)



Reduced I/Os => Impact on - prototyping requirements (platform complexity)
- length of experiments (#bits to be transferred)

Results: frequency (Montgomery, Virtex II)



Results: impact on the analysis efficiency (Montg)

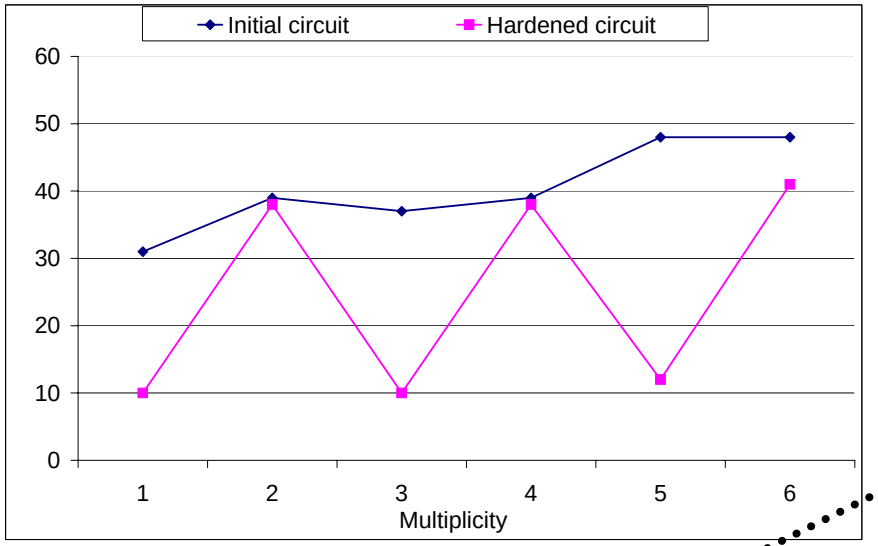
Initial circuit (before hardening):

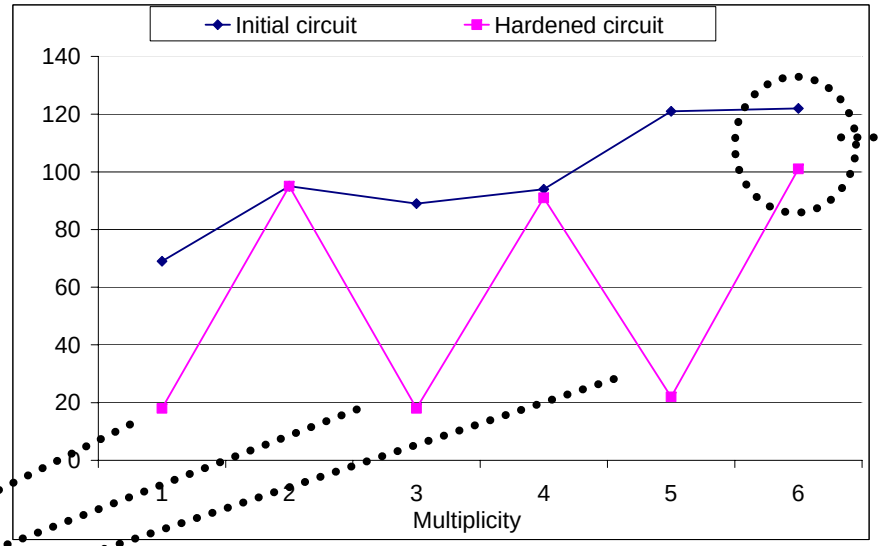
Multiplicity	States/Transitions	Common States/Transitions	Specific States/Transitions	Specific States/Transitions (SEU)
1	**31/69**	--	--	--
Multiple fault injection not limited to a single register				
2	39/95	30/67	9/28	1/2
3	37/89	30/65	7/24	1/4
4	39/94	30/63	9/31	1/6
5	48/121	31/65	17/56	0/4
6	48/122	31/65	17/57	0/4

=> More complex error propagation paths when the multiplicity increases

Results: impact on the analysis efficiency

Comparison of results obtained with the two versions of the Montgomery core





States **Transitions**

Odd multiplicity: only transient erroneous states are recorded – 100% detection
Otherwise: ~50% detection, simpler error propagations only for large multiplicity

Results very similar in terms of erroneous configurations (states) and propagation paths (transitions)

Conclusions

- Multiple bit soft errors is an increasing concern …
- … and existing analysis environments and methodologies must be extended to automatically take into account spatial (and temporal ?) multiplicity
- A new generation of mutants has been reported targeting heterogeneous fault/error models
- Practical results show that considering only single bit flips can lead to optimistic conclusions or non optimal protections
- Prototypes currently fabricated (ST HCMOS9 technology)
- Experiments using laser-based attacks scheduled in September to compare with simulation results (actual probability to detect an attack ?…)

Thank you !

Any questions ?

VHDL modifications: examples (1)

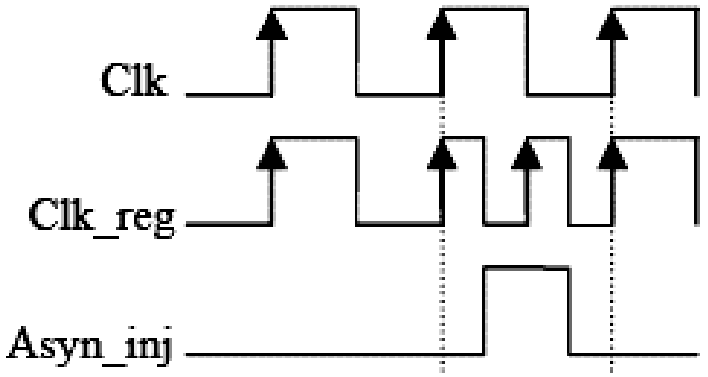
Insertion of the injection control signals in the entity definition:

```
entity My_entity is
  port
    ( -- initial inputs/outputs
    clk, reset : in std_logic ;                    -- Clock and initialization signals
    …                                              -- Other signals
     -- control  inputs  insertion
    En_inj : in std_logic_vector(i downto 0);      -- Number of bits to invert
    En_asyn : in std_logic;                        -- Asynchronous injection control
    num_bit1 : in integer range a to b;            -- Index of 1st bit to modify
    num_bit2 : in integer range a to b;            -- Index of 2nd bit to modify
    …
    num_bitn : in integer range a to b             -- Index of nth bit to modify
     ) ;
end My_entity;
```

VHDL modifications: examples (2)

Process added to modify the circuit clock:

```
seq: process(Clk, Asyn_inj)
begin
    if(Asyn_inj='1') then
            Clk_reg <=  not(Clk) ;
    else
            Clk_reg <= Clk;
    end if;
end process ;
```



The signal Asyn_inj is used to asynchronously determine the injection times; an extra clock edge is sent to the target registers at injection times.

VHDL modifications: examples (3)

Basic process defining the modification in the virtual register and the register values at injection time :

```
process (En_inj, num_bit1, …,  num_bitn, Elemt0,…, Elemti)
variable All_reg_tmp : std_logic_vector(l  downto 0);
 begin
      All_reg_tmp := Elemti …&Elemt1&Elem0;
      if En_inj="one" or  En_inj="two" or ... En_inj="n" then
            All_reg_tmp(num_bit1) := not All_reg_tmp(num_bit1);
      end if;
      if En_inj="two" or ... En_inj="n" then
            All_reg_tmp(num_bit2) := not All_reg_tmp(num_bit2);
      end if;
       …
      if En_inj="n" then
            All_reg_tmp(num_bitn) := not All_reg_tmp(num_bitn);
      end if;
      Elemti_inj <= All_reg_tmp(l downto l-li);
      …
      Elemt1_inj <= All_reg_tmp(n downto n-l1);
      Elemt0_inj <= All_reg_tmp(n-l1-1 downto 0);
end process;
```

Results: impact on the analysis efficiency (Montg)

Hardened circuit:

Multiplicity	States/Transitions	Common States/Transitions	Specific States/Transitions	Specific States/Transitions (SEU)
1	**10/18**	--	--	--
Multiple fault injection not limited to a single register				
2	38/95	9/15	29/80	1/3
3	10/18	7/12	3/6	3/6
4	38/91	9/10	29/81	1/86
5	12/22	6/10	6/12	4/8
6	41/101	10/10	31/91	0/8

=> Few states/transitions when the multiplicity is odd
(only transient erroneous configurations before 100% detection)
=> Similar complexity of error propagation paths in the other cases

Results: case of crash/detection states (Montg)

Multiplicity	%Crash Initial version	%Crash Hardened version	%Detection Hardened version	%Crash+Detection Hardened version
1	2.5	2.45	97.55	100
Multiple fault injection not limited to a single register				
2	6.6	6.6	47.77	54.37
3	9.3	9.29	90.71	100
4	12.3	12.34	50.10	62.44
5	15.4	15.38	84.62	100
6	18	18.29	47.73	66.02

=> Crash situations during RT-Level simulations increase with multiplicity