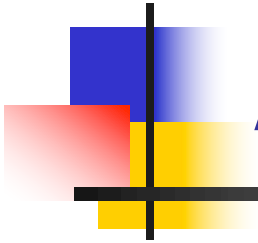
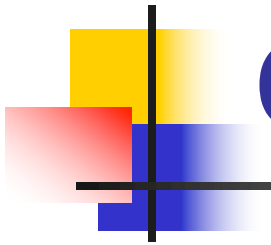


Chapter 3
Fault Simulation

Arnaud Virazel

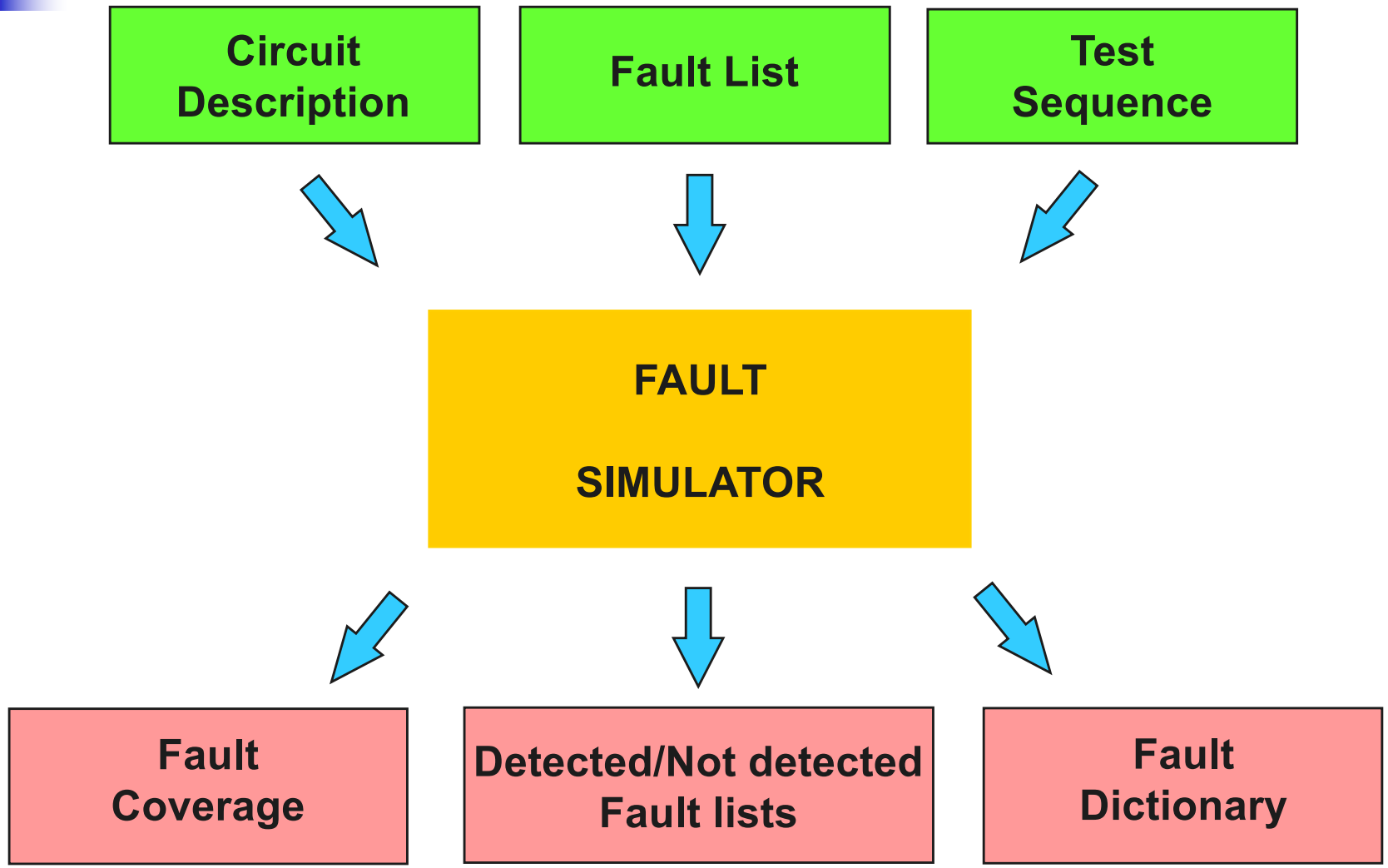
virazel@lirmm.fr

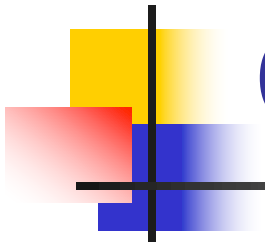


Objectives

- To determine the circuit behaviour in presence of each of the faults
- To determine the set of faults detected by a test vector sequence

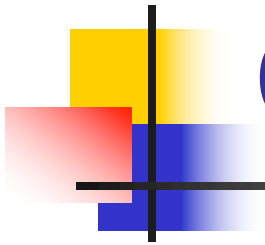
Fault Simulation flow





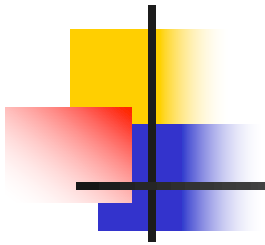
Constraints

- Circuit description
  - Functional, structural
- Fault models
  - Stuck-at, bridging, delay …
- Coding
  - Two values (0,1)
  - Three values (0, 1, X)
  - Four values (0, 1, X, Z)



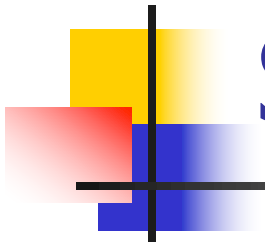
Constraints

- Timing
  - Zero-delay → No delay considered during the simulation
  - Unit-delay → A delay is associated to each gate



Fault Simulation Techniques

- Serial fault simulation
- Parallel fault simulation
- Deductive fault simulation
- Concurrent fault simulation

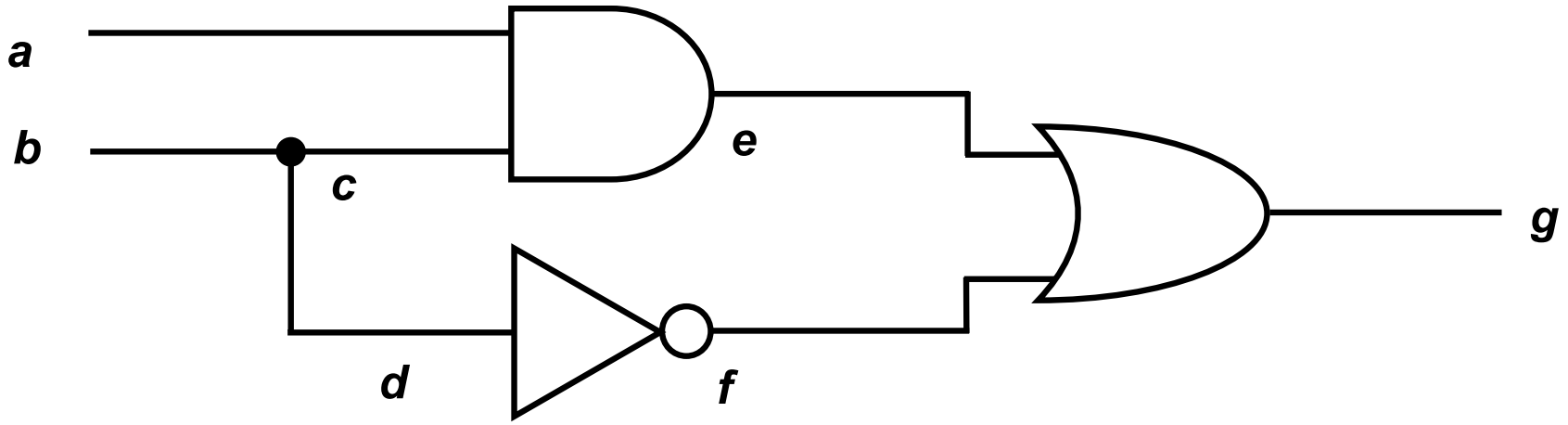


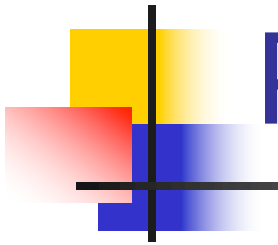
Serial Fault Simulation

- A golden circuit simulation + n independent simulations of the n faulty circuits (for n considered fault)
- Pros:
  - Simple. Does not require the development of a specialized fault simulator. Can be implemented from a classic logic simulator.
  - Able to simulate any type of fault that can be taken into account by the logic simulator.
  - Low memory space requirement
- Cons:
  - Very slow (n+1 consecutive simulations for a list of n faults)

Exercise

- Determine the faults detected by the vectors:
  V (a,b) = (1,1) et (0,0)





Parallel Fault Simulation

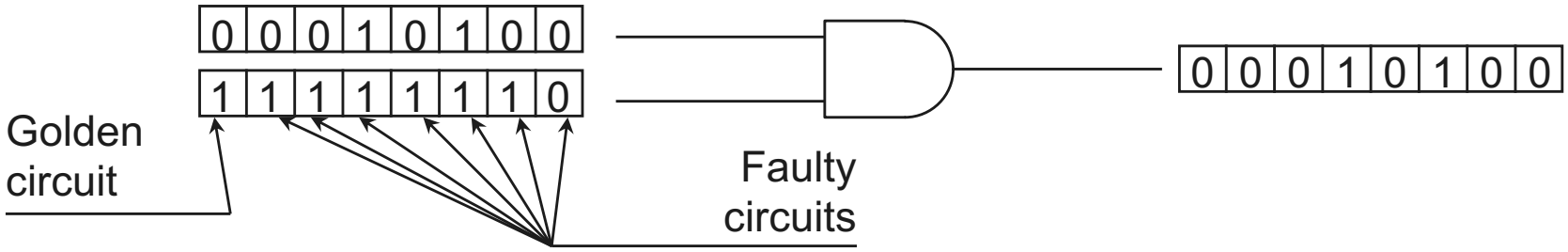
- Principle
  - Takes advantage of parallel (per-word) processing of binary information in computers
    - If the host computer manipulates n-bit words, the golden circuit and the n-1 faulty circuits can be simulated in parallel

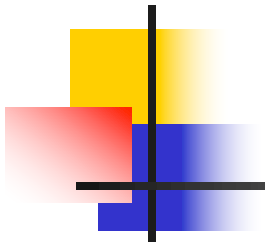
    $\Rightarrow$ The number of faults considered during a simulation depends on the length of the computer words

    $\Rightarrow$ Fault simulation performed in some phases

  - Each circuit net is associated with a n-bit words
    - The first bit represents the logic state of the golden circuit
    - Each other bit represents the state of a faulty circuit

Example


0 0 0 1 0 1 0 0
1 1 1 1 1 1 1 0
Golden circuit
Faulty circuits
0 0 0 1 0 1 0 0

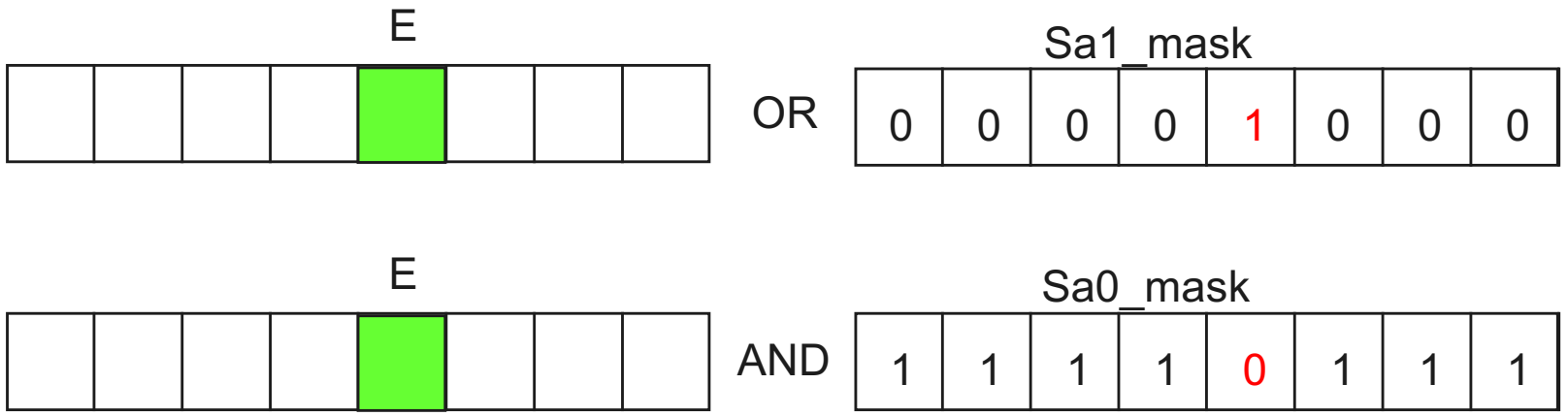


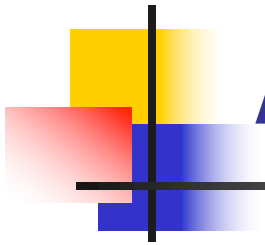
Methodology

- Faults are taken into account (or injected) via logic masks
  - A Sa1 at net E is injected by “ORing” the word of E and the Sa1_mask
  - A Sa0 at net E is injected by “ANDing” the word of E and the Sa0_mask

Sa0 and Sa1 Masks

- Fault injection masks are taken into account when evaluating the output of a gate

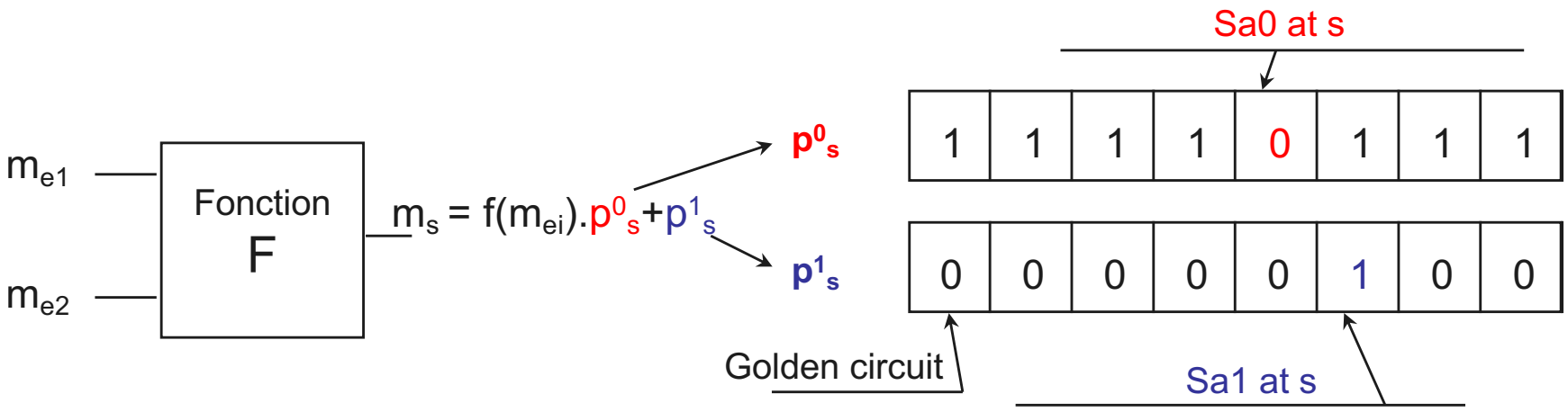




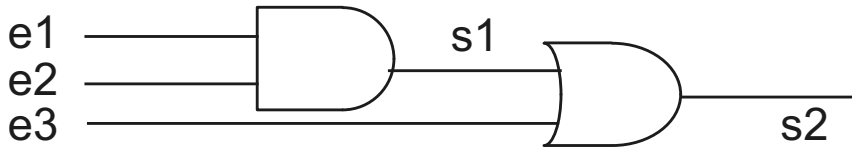
AND Gate Example

- *ma* and *mb* are the two words assigned to the AND gate inputs (a, b)
- *p0* and *p1* are the two masks corresponding to Sa0 and Sa1 faults on output c of the AND gate
- Computation of word *mc* (output of the AND gate)
  - mc = ma . mb for the golden gate
  - mc = ma . mb . p0 if the Sa0 on c is considered only
  - mc = ma . mb + p1 if the Sa1 on c is considered only
  - mc = ma . mb . p0 + p1 if both SaF are considered

General Case



Example (1)



- A 5-bit word

Golden	S1 Sa0	S1 Sa1	S2 Sa0	S2 Sa1

Example (2)

- Masks
  - s1 Sa0 p0(s1) = 10111 s2 Sa0 p0(s2) = 11101
  - s1 Sa1 p1(s1) = 00100 s2 Sa1 p1(s2) = 00001
- Test vector
  - V (e1, e2, e3) = 110
    m(e1) = 11111, m(e2) = 11111 et m(e3) = 00000
- Parallel fault simulation
  - m(s1) = m(e1) . m(e2) . p0(s1) + p1(s1) = 10111
  - m(s2) = (m(s1) + m(e3)) . p0(s2) + p1(s2) = 10101

Example (2)

- Masks
  - s1 Sa0 p0(s1) = 10111 s2 Sa0 p0(s2) = 11101
  - s1 Sa1 p1(s1) = 00100 s2 Sa1 p1(s2) = 00001
- Test vector
  - V (e1, e2, e3) = 110
    m(e1) = 11111, m(e2) = 11111 et m(e3) = 00000
- Parallel fault simulation
  - m(s1) = m(e1) . m(e2) . p0(s1) + p1(s1) = 10111
  - m(s2) = (m(s1) + m(e3)) . p0(s2) + p1(s2) = 10101
- Fault simulation result
  - Golden circuit response → s2 = 1
  - V = 110 detects Sa0 faults on s1 et s2

Exercise

- Test vector → V (a,b) = (1,1)