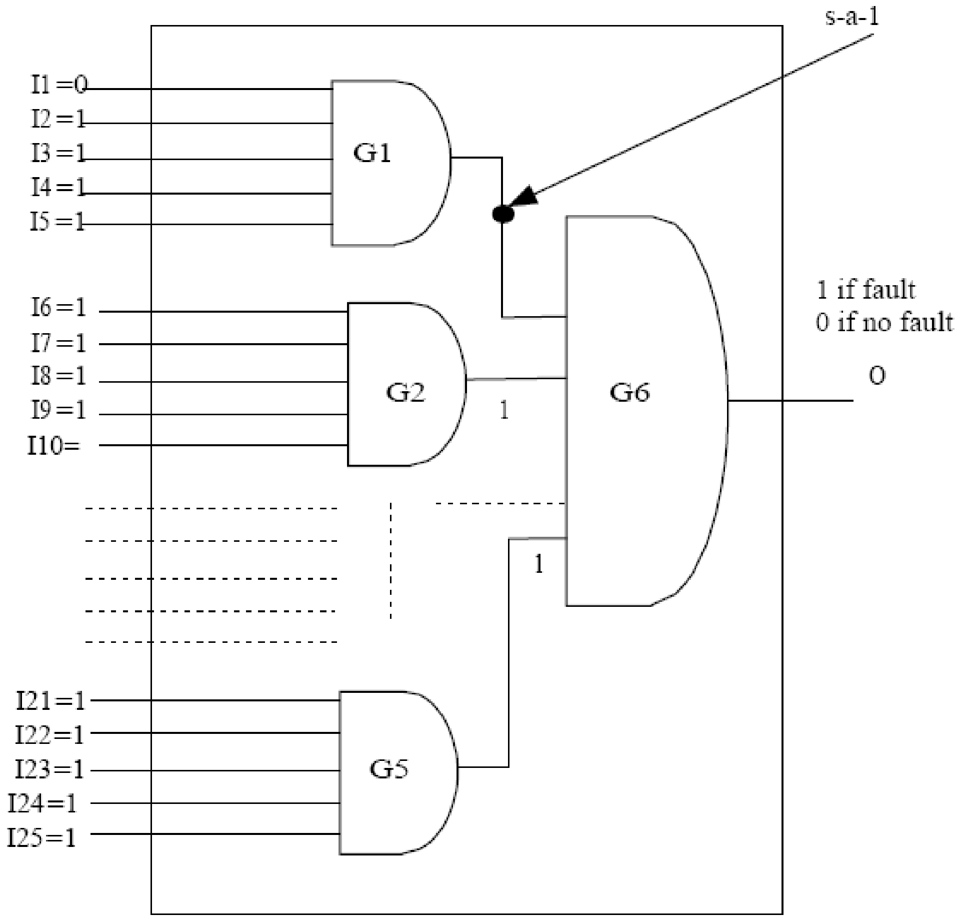
Introduction to Test Pattern Generation

- The procedure to generate a test pattern for a given a fault is called Test Pattern Generation (TPG).
- Generally TPG procedure is fully automated and called Automatic TPG (ATPG).



Introduction to Test Pattern Generation

- **Fault Sensitization**: Output net of G1 is stuck-at-1, we need to drive it to 0 to verify the presence/absence of the fault.
- **Fault Propagation**: Affect of the fault is to be propagated to a primary output (Output of G6, in this example).
- **Justification**: Determination of values at primary inputs so that Fault sensitization and Fault propagation are successful.

Introduction to Test Pattern Generation

Test Pattern No.	Test Pattern I1 I2 I3 I4 I5 I6……………………I25	Output
1	0 0 0 0 0 11111111111111111111	1 if fault 0 if no Fault
2	0 0 0 0 1 11111111111111111111	1 if fault 0 if no Fault
…………	…………………………………………	……..
2^5	1 1 1 1 0 11111111111111111111	1 if fault 0 if no Fault

TPG procedure would generate any one of the patterns given in Table 1

Introduction to Test Pattern Generation

Do we require these steps for all faults?
TPG would take significant amount of time.
However, one test pattern can test multiple faults.

Pattern No.	Random Pattern I1 I2 I3 I4 I5	I6……………………I25	Faults Detected
1	1 0 0 0 1	11111111111111111111	s-a-1 at net "output of G1" s-a-1 at net "output of G6"
2	1 1 1 1 1	11111111111111111111	s-a-0 faults in all the nets of the circuit

On the other hand if we would have gone by the "sensitize-propagate-justify" approach these three steps would have been repeated 31 times.

Test pattern generation

TPG can be done in two phases

- Use random patterns, till a newly added pattern detects a reasonable number of new faults

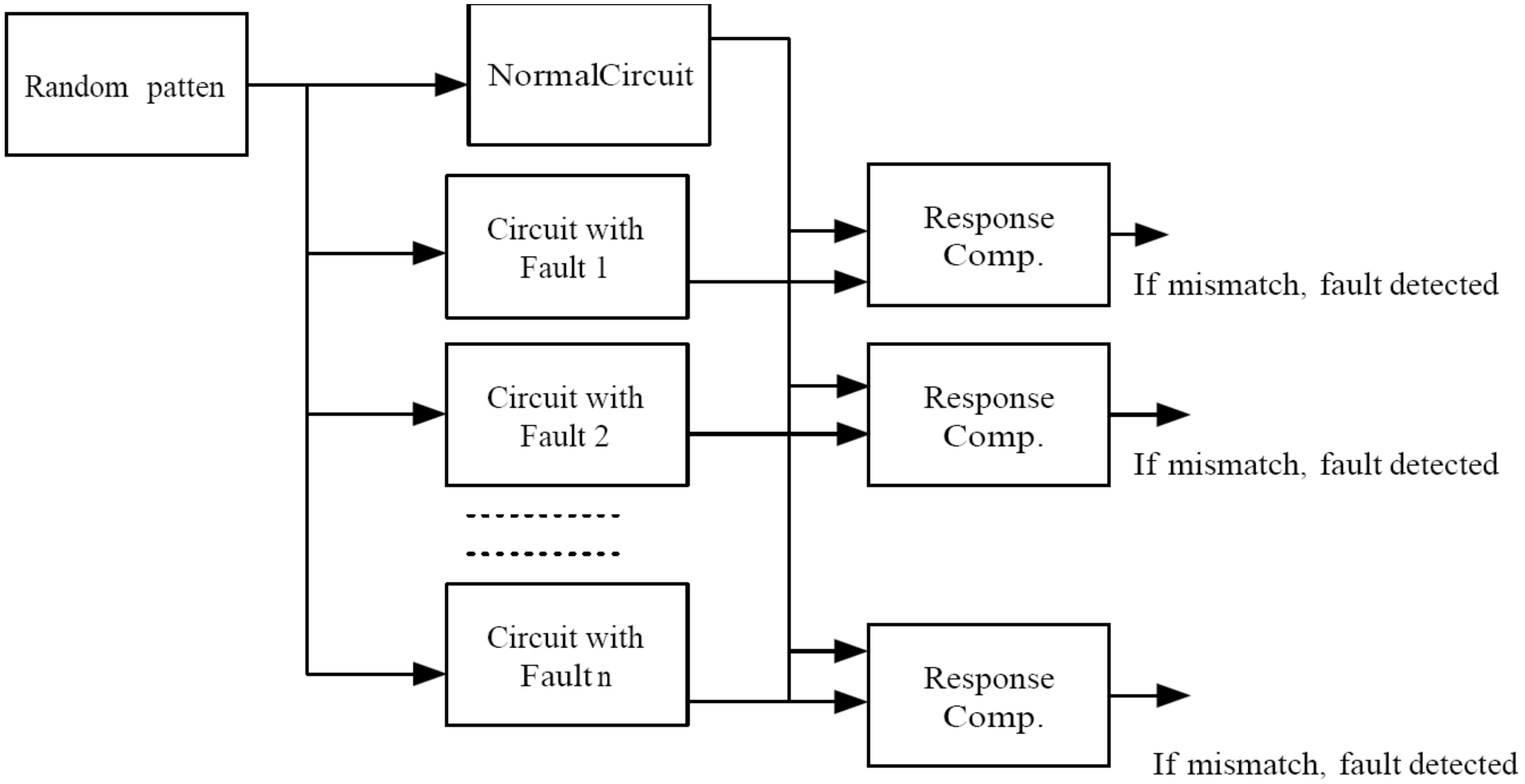
- For the remaining faults, apply “sensitize-propagate-justify” approach

Techniques to determine faults covered by random patterns, called *fault simulation*.

Fault Simulation

- A fault simulator is like an ordinary simulator, but needs to simulate two versions of a circuit
  1. Without any fault for a given input pattern
  2. With a fault inserted in the circuit and for the same input pattern.
- If the outputs under normal and faulty situation differ, the pattern detects the fault.
- Step (ii) is repeated for all faults. Once a fault is detected it is dropped

Fault Simulation



The procedure is simple, but is too complex in terms of time required. Time required is

$$\sum_{i=i}^{no\ of\ random\ patterns} faults\ for\ i^{th}\ random\ pattren \times simulation\ time\,.$$

Improving Fault Simulation Algorithms

- Determine more than one fault that is detected by a random pattern during one simulation run

Fault Simulation

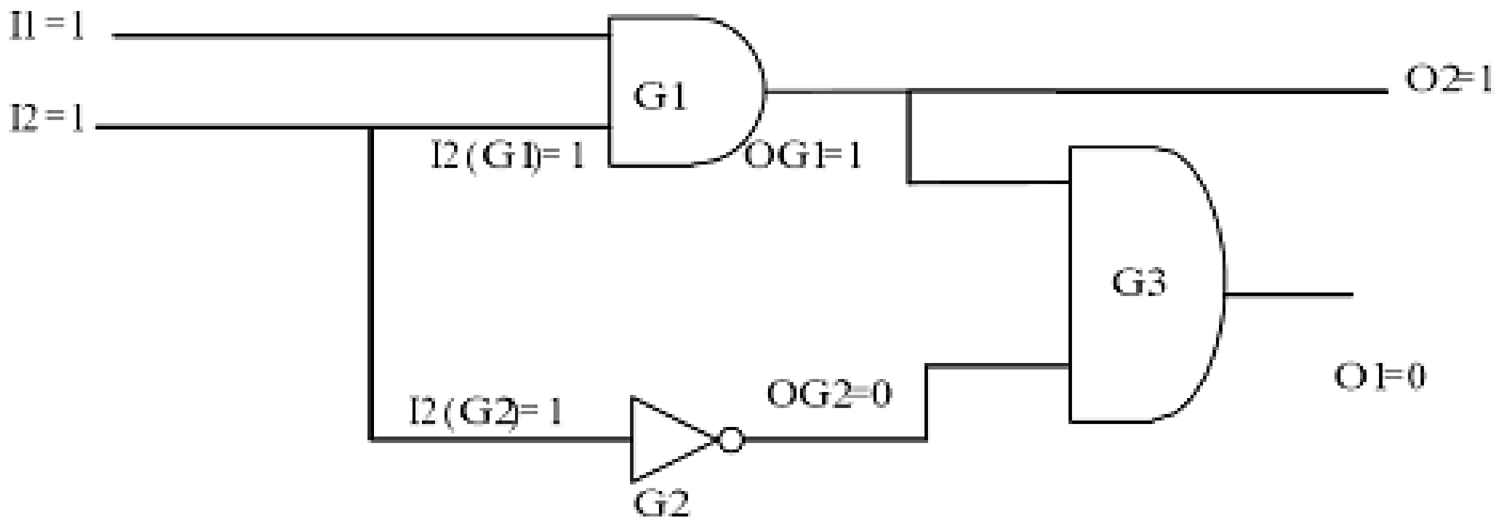
- Types of Algorithms
  - Serial
  - Parallel
  - Deductive
  - Concurrent

Serial Algorithm

- Algorithm: Simulate fault-free circuit and save responses. Repeat following steps for each fault in the fault list:
  - Modify netlist by injecting one fault
  - Simulate modified netlist, vector by vector, comparing responses with saved responses
  - If response differs, report fault detection and suspend simulation of remaining vectors

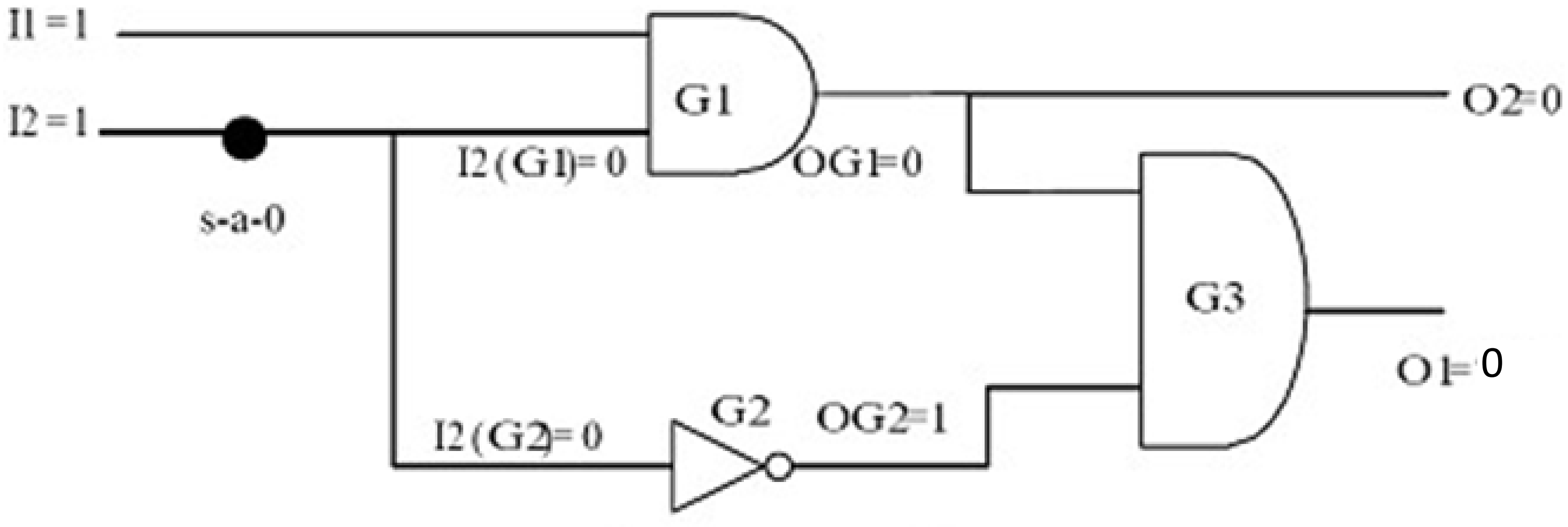
- Advantages:
  - Easy to implement; needs only a true-value simulator, less memory
  - Most faults can be simulated

Serial Fault Simulation: Example



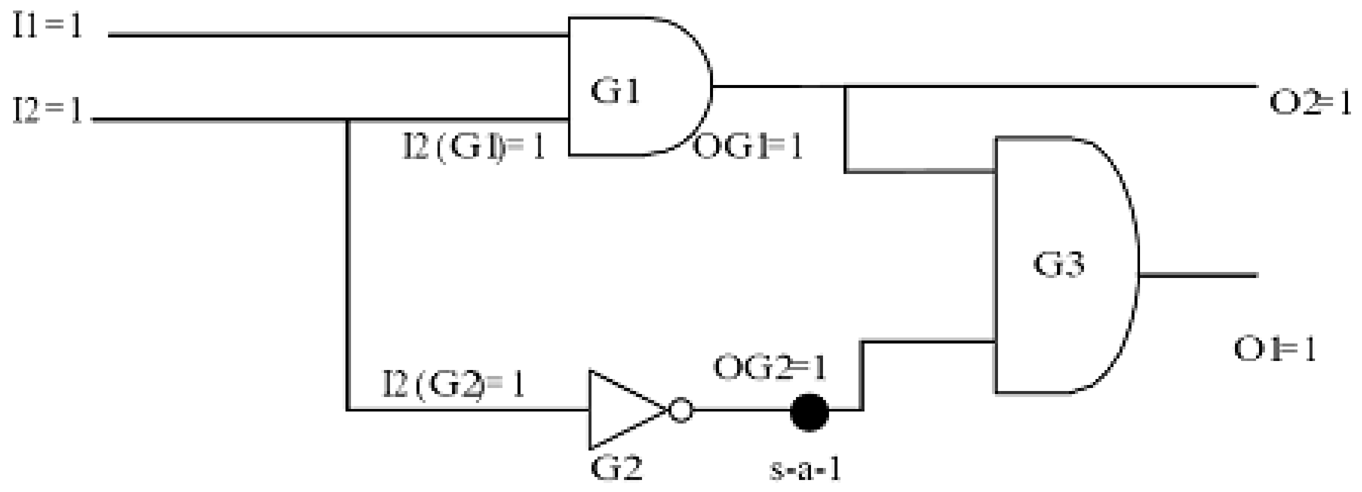
Normal Condition

Serial Fault Simulation: Example



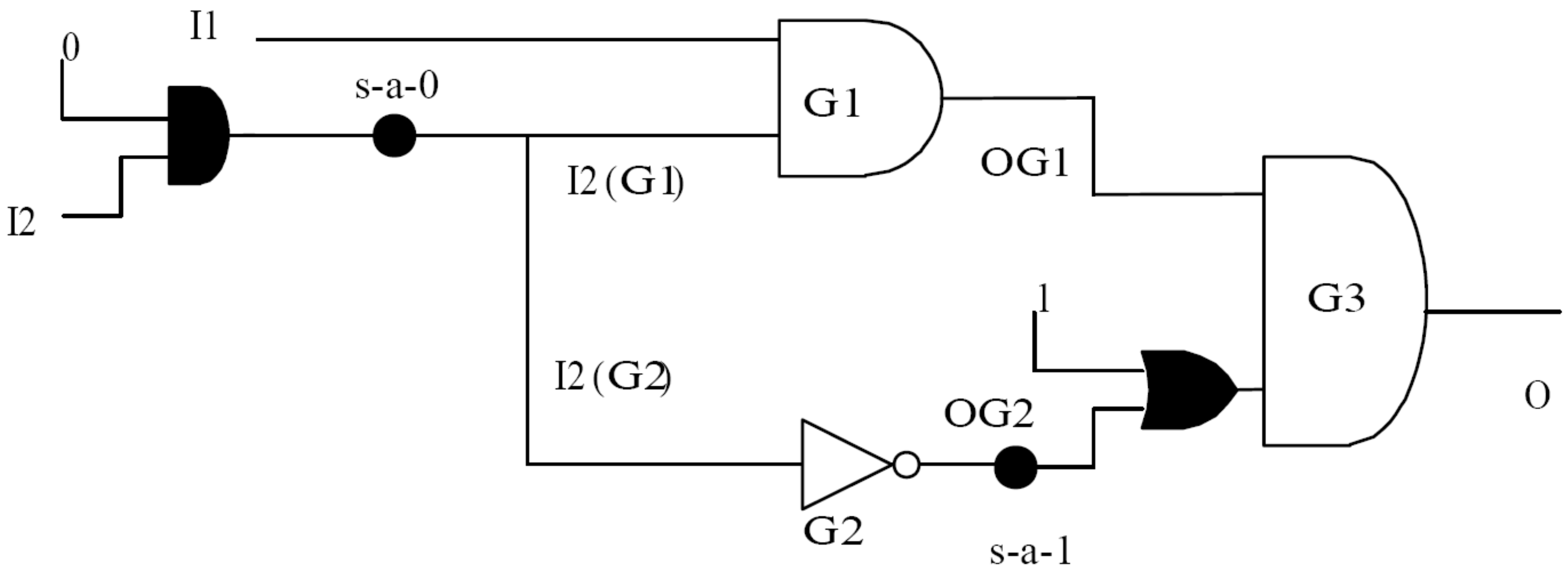
s-a-0 fault in I2 Condition

Serial Fault Simulation: Example



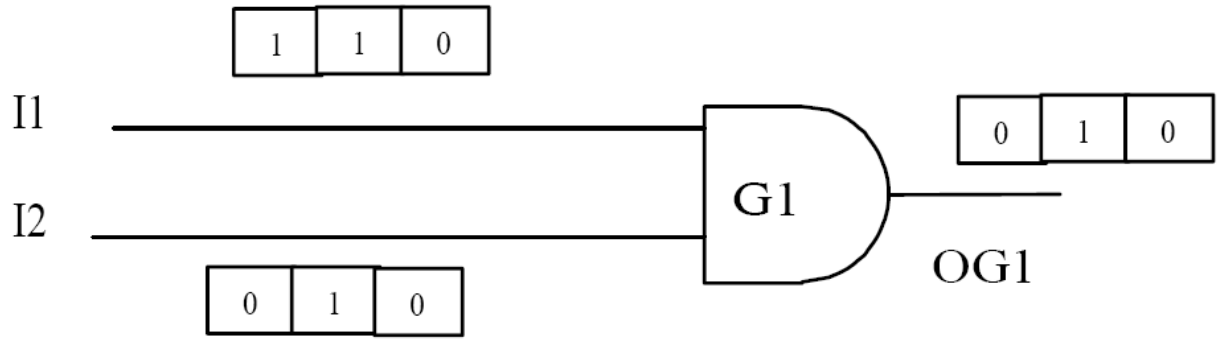
s-a-1 fault in OG2 Condition

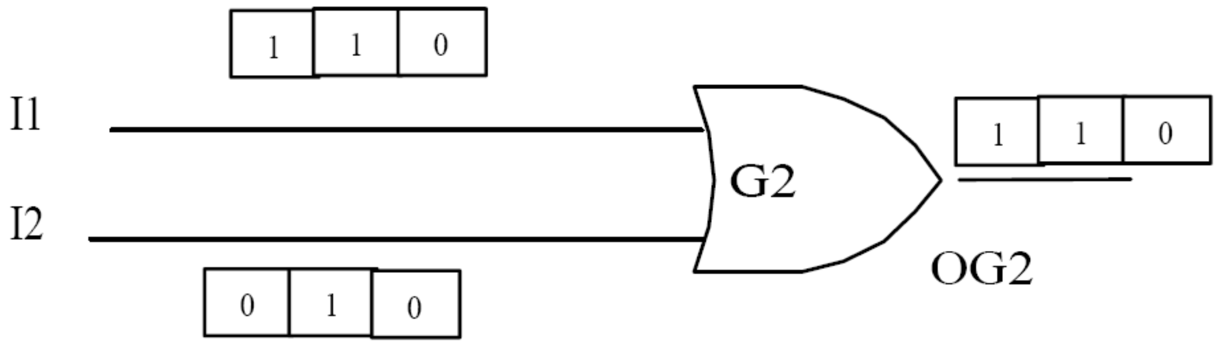
Insertion of faults in the circuit fault simulation in event driven simulator



Parallel Fault Simulation

- Parallel fault simulation can processes more than one fault in one pass of the circuit simulation.
- Uses bit-parallelism of a computer.

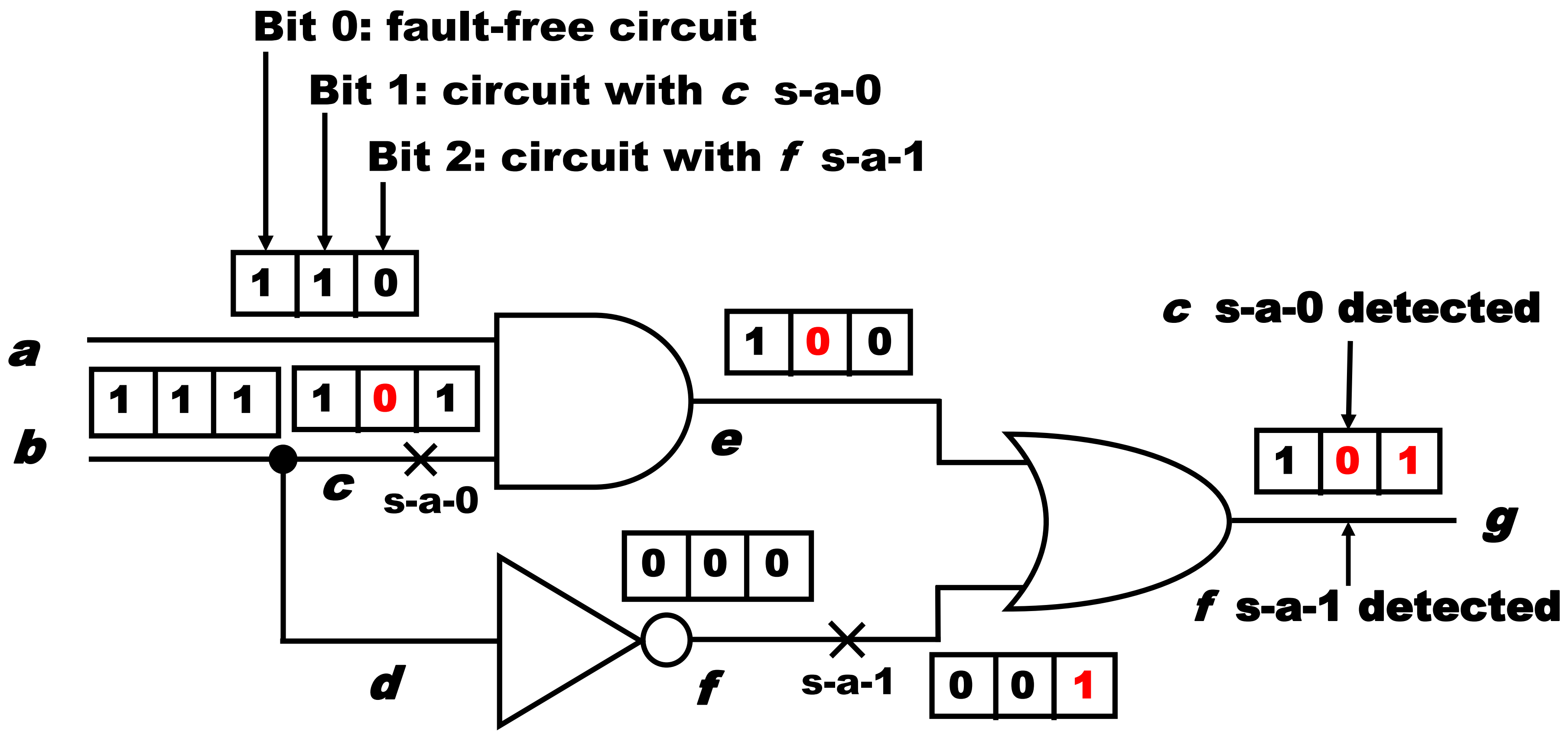




Parallel Fault Simulation

– Input lines for any gate comprise binary words of length *w* (instead of single bits) and output is also a binary word of length *w.*

Parallel Fault Simulation - Example


Bit 0: fault-free circuit
Bit 1: circuit with *c* s-a-0
Bit 2: circuit with *f* s-a-1
1 1 0
a
1 1 1
b
1 0 1
c
s-a-0
1 0 0
e
d
0 0 0
f
s-a-1
0 0 1
c s-a-0 detected
1 0 1
g
f s-a-1 detected