Fault Simulation

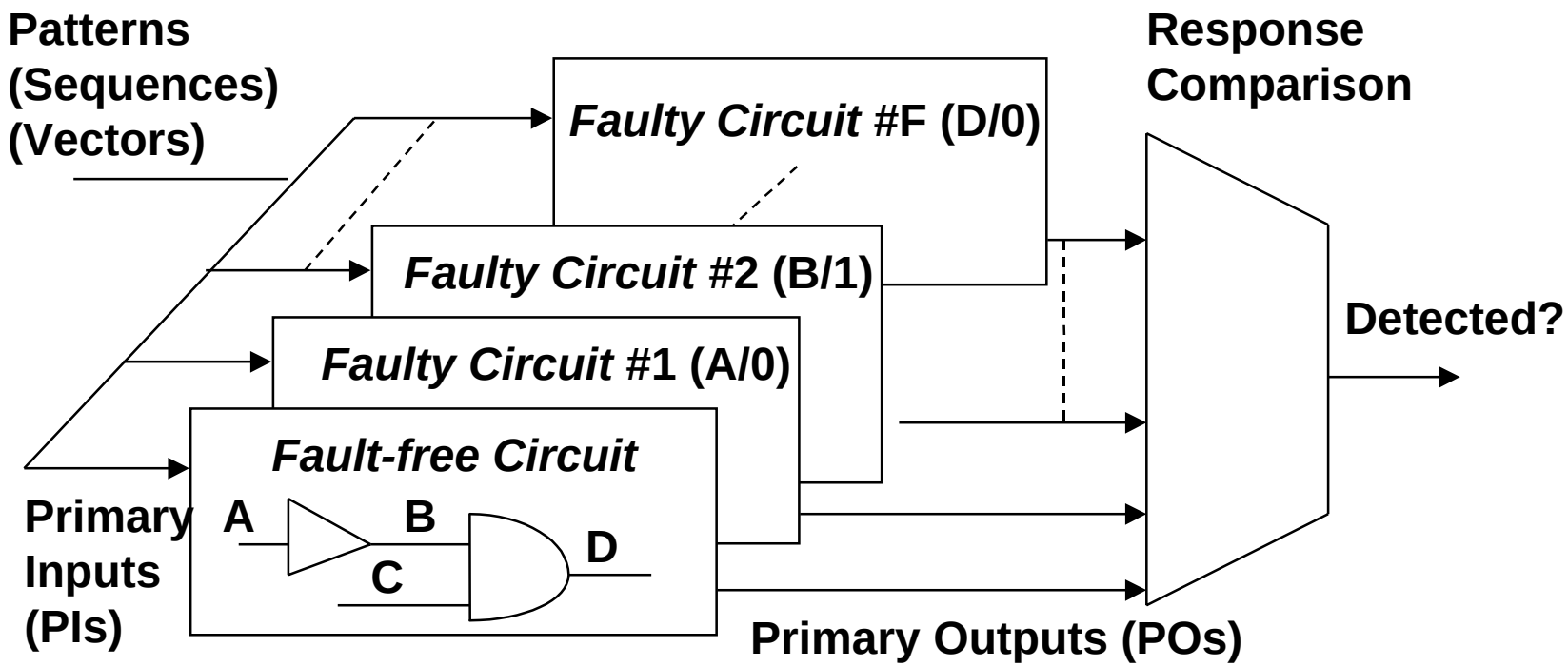
- **Introduction**
- **Classical Fault Simulation**
- **Modern Fault Simulation for Combinational Circuits**
- **Hardware Approaches to Fault Simulation**

(Source: NTU 林呈祥 教授)

Why Fault Simulation?

1. To evaluate the quality of a test set
   - usually in terms of fault coverage
2. To incorporate into ATPG for test generation
   - due to its lower complexity
3. To construct fault dictionary
   - for post-test diagnosis

Conceptual Fault Simulation

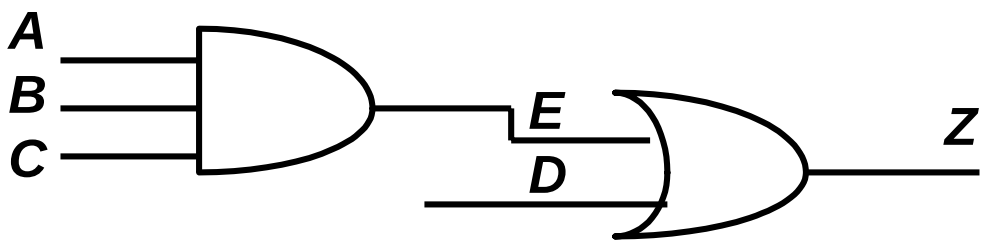


- Logic simulation on both good (fault-free) and faulty circuits

Some Basics for Logic Simulation

- For fault simulation purpose, mostly the gate delay is assumed to be zero unless the delay faults are considered. Our main concern is the functional faults.
- The logic values can be either two (0, 1) or three values (0, 1, X). For delay fault, more values will be needed.
- Two simulation mechanisms:
  - Oblivious compiled-code: circuit is translated into program and all gates are executed for each pattern.
  - Interpretive event-driven: circuit structure and gate status are stored in the table and only those gates needed to be updated with a new pattern are processed.

Oblivious Compiled Code

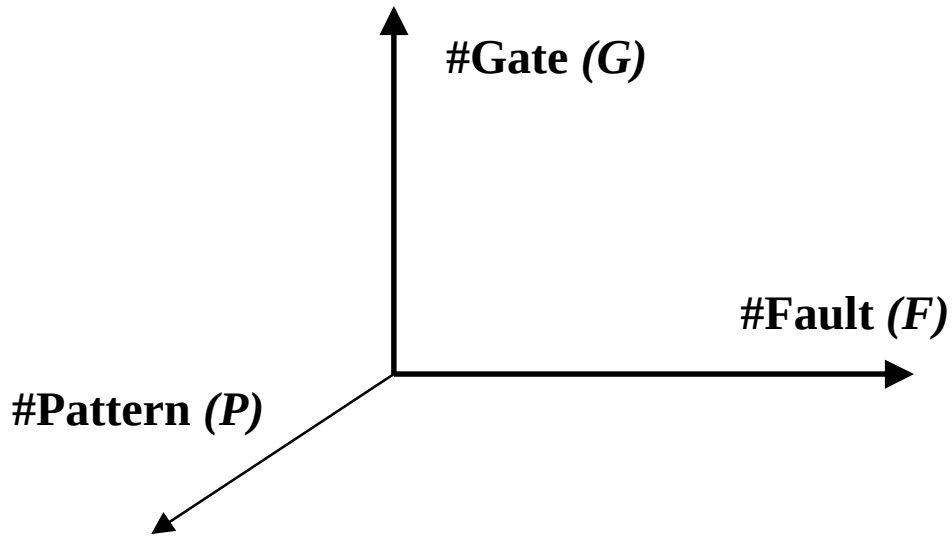


- **Compiled codes**
  - **LDA** ***A*** **/* load accumulator with value of *A* */**
  - **AND** ***B*** **/* calculate *A and B* */**
  - **AND** ***C*** **/* calculate *E = AB and C* */**
  - **OR** ***D*** **/* calculate *Z = E or D* */**
  - **STA** ***Z*** **/* store result of *Z* */**

Event-Driven

- **While (event list not empty) begin**
  - ***t* = next time in the list**
  - **for every event *(i, t)* begin**
    - † **update value of gate *i***
    - † **schedule fanout gates of *i* in the event list if value changes are expected**
  - **end**
- **end**

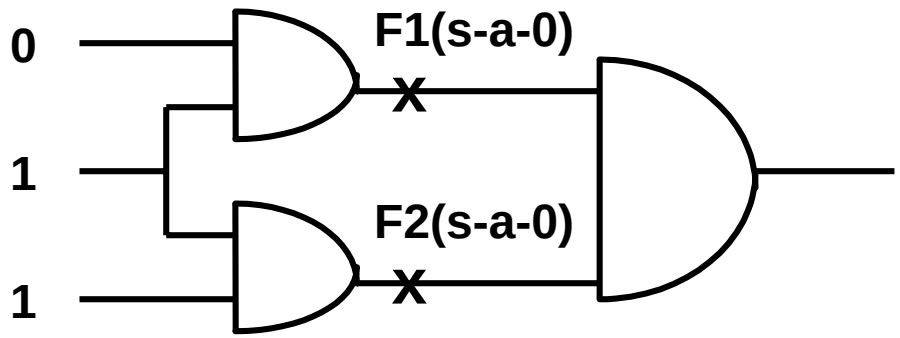
Complexity of Fault Simulation



- Complexity = $P * F * G \sim O(G^3)$ with single s-a faults
- The complexity is higher than logic simulation, $O(G^2)$, but is much lower than test pattern generation.
- In reality, the complexity is much lower due to fault dropping and advanced techniques.

Characteristics of Fault Simulation

- Fault activity with respect to fault-free circuit is often sparse both in time and in space.
- For example, F1 is not activated by the given pattern, while F2 affects only the lower part of this circuit.



- The efficiency of a fault simulator depends on its ability to exploit these characteristics.

Classical Fault Simulation Techniques

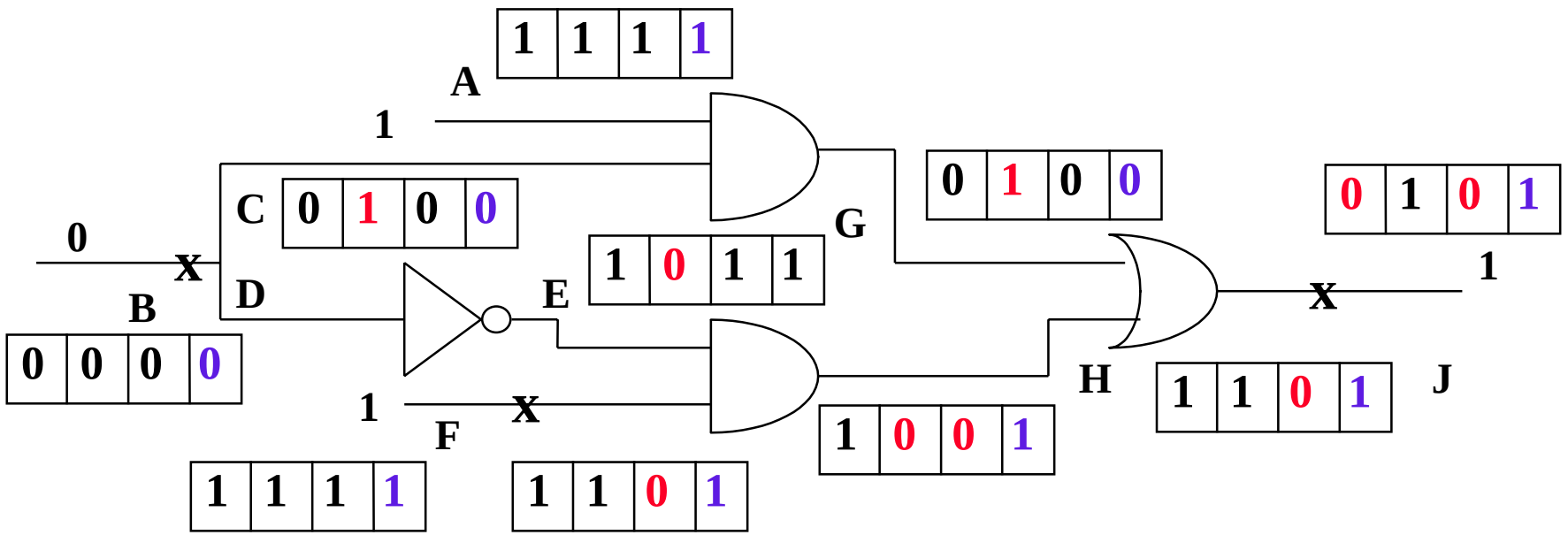
- Common Characteristics:
  - In general, no restriction on the circuit types.
  - Developed before VLSI era.
- Serial Fault Simulation
  - trivial single-fault single-pattern
- Parallel Fault Simulation
- Deductive Fault Simulation
- Concurrent Fault Simulation

Parallel Fault Simulation

- **Taking advantage of inherent parallel operation of computer words to simulate faulty circuits in parallel with fault-free circuit**
  - **the number of faulty circuits, or faults, can be processed parallelly is limited by the word length.**
- **Straightforward and memory efficient**
- **Some weaknesses:**
  - **An event, a value change, of a single fault or fault-free circuit leads to the computation of the entire word.**
  - **The fault-free logic simulation is repeated for the number of passes.**

Example of Parallel Fault Simulation

- **Consider three faults: B/1, F/0, and J/0**
  - **Bit-space:** | J/0 | B/1 | F/0 | FF | **where FF = Fault-free**



Deductive Fault Simulation

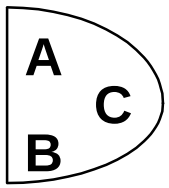
- A list of faults associated with each line containing the identifier of each fault which produces a fault effect on this line.
  - Only the faults with fault effect, or difference w.r.t. fault-free circuit, is retained in the list.
- The propagation of such lists can be based on set operation derived the gate types and values.
  - The list update is performed with each new pattern, which is not efficient in computer.
  - The list may dynamically grow is size, which incurs memory explosion problem.

Fault List Propagation Rule

- **Let the gate G = F(A,B) with input lists L_A and L_B and output list L_G to be updated.**
- **If in the fault-free circuit,**
  - **the value of G is 0(1), use F(F-)**
  - **the value of gate input A is 0(1), replace A in logic expression by L_A(L_A-) and A- by L_A-(L_A).**
- **Replace * by *intersection* and + by *union*.**
- **Add G/1(G/0) to the list L_G.**

Illustration of Fault List Propagation

Consider a two-input AND gate



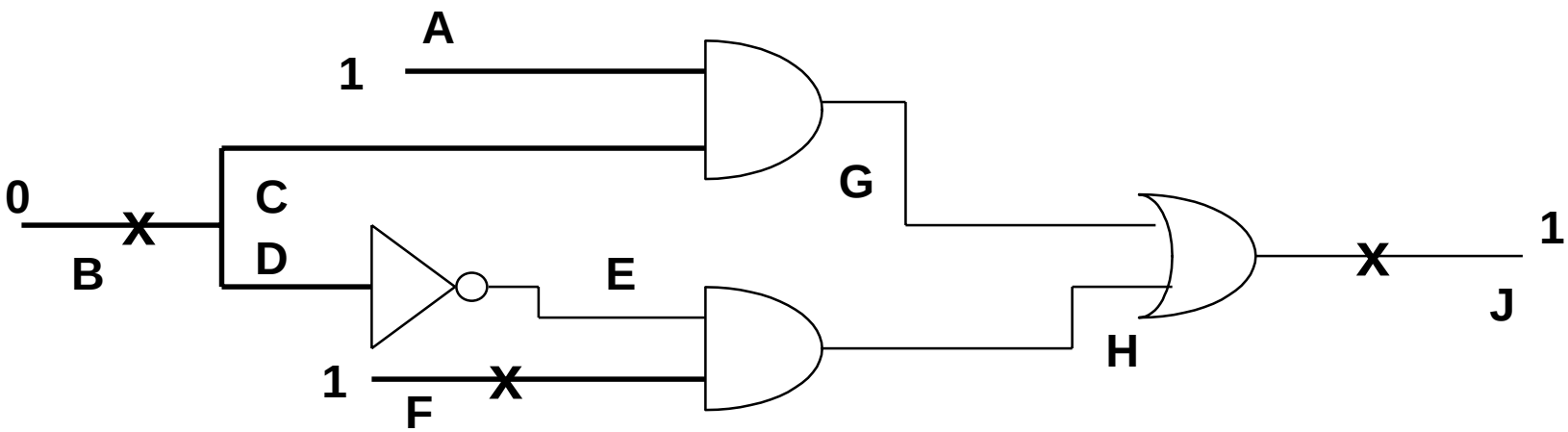
Case 1: A=1, B=1, C=1 at fault-free,
$L_C = L_A + L_B + \{C/0\}$

Case 2: A=1, B=0, C=0 at fault-free,
$L_C = L_{A^-} * L_B + \{C/1\}$

Case 3: A=0, B=0, C=0 at fault-free,
$L_C = L_A * L_B + \{C/1\}$

Example of Deductive Simulation Ia

- Consider 3 faults: B/1, F/0, and J/0

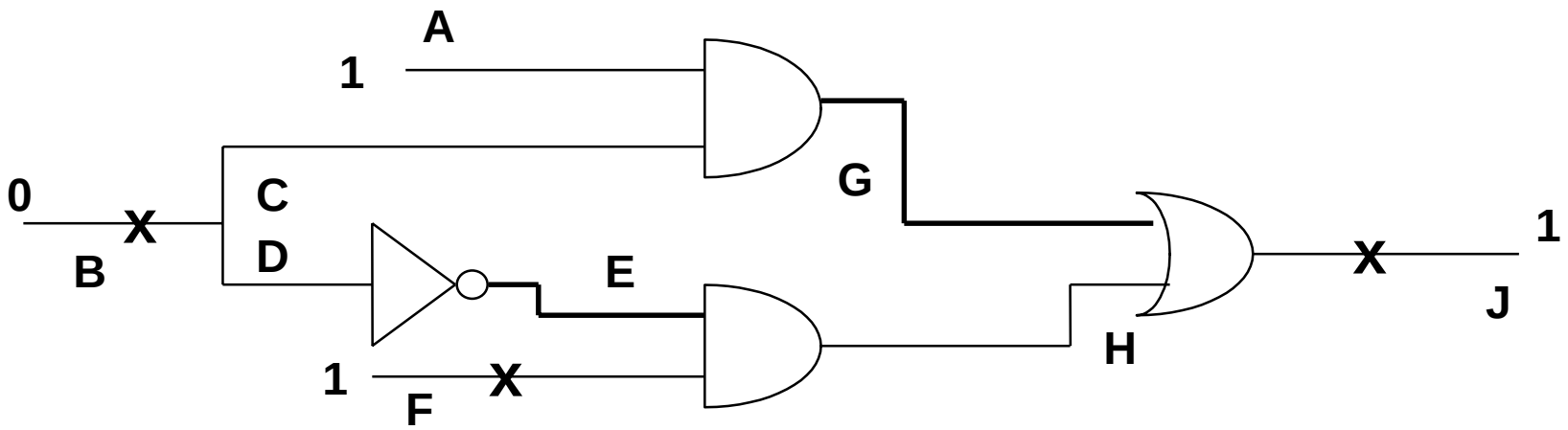


$L_B = \{B/1\}$, $L_F = \{F/0\}$, $L_A = 0$

$L_C = L_D = \{B/1\}$

Example of Deductive Simulation Ib

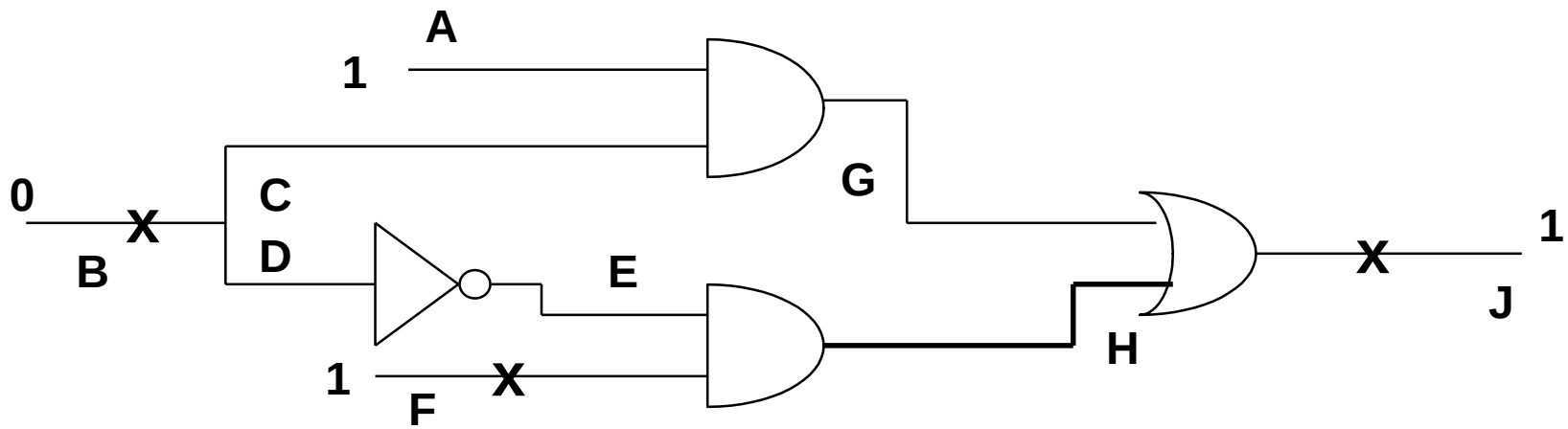
- Consider 3 faults: B/1, F/0, and J/0



$L_B = \{B/1\}$, $L_F = \{F/0\}$,
$L_C = L_D = \{B/1\}$,
$L_G = \{B/1\}$, $L_E = \{B/1\}$

Example of Deductive Simulation Ic

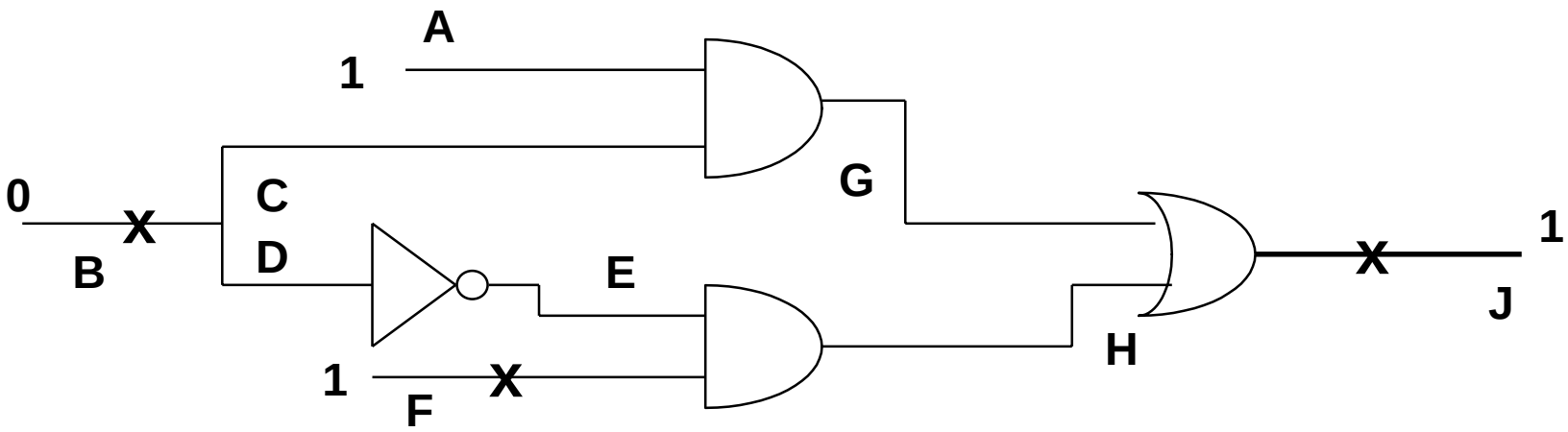
- Consider 3 faults: B/1, F/0, and J/0



$L_B = \{B/1\}$, $L_F = \{F/0\}$,
$L_C = L_D = \{B/1\}$, $L_G = \{B/1\}$,
$L_E = \{B/1\}$, $L_H = \{B/1, F/0\}$

Example of Deductive Simulation Id

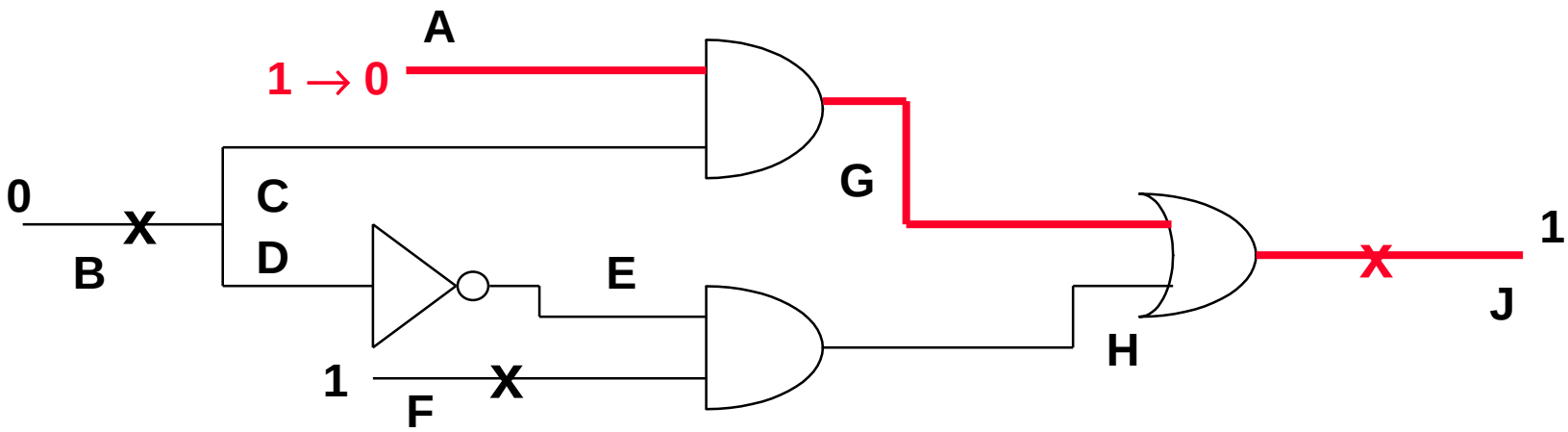
- Consider 3 faults: B/1, F/0, and J/0



L_B = {B/1}, L_F = {F/0},
L_C=L_D = {B/1}, L_G = {B/1},
L_E = {B/1}, L_H = {B/1,F/0}, L_J = {F/0,J/0}

Example of Deductive Simulation II

- When A changes from 1 to 0

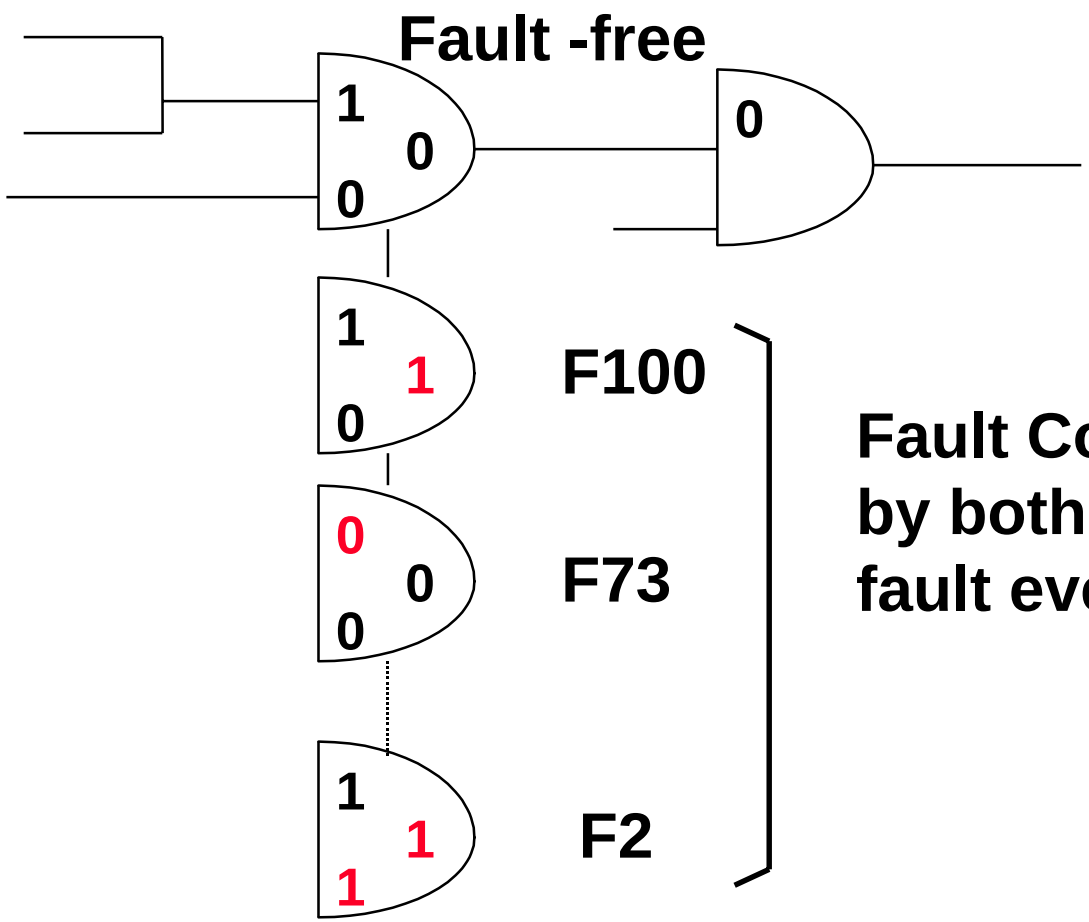


$L_B = \{B/1\}$, $L_F = \{F/0\}$,
$L_C = L_D = \{B/1\}$, $L_G = 0$,
$L_E = \{B/1\}$, $L_H = \{B/1, F/0\}$, $L_J = \{B/1, F/0, J/0\}$

Concurrent Fault Simulation

- **Each gate retains a list of fault copies each of which stores the status of a fault exhibiting difference from fault-free values.**
- **Simulation mechanism is similar to the conceptual fault simulation except that only the dynamical difference w.r.t. fault-free circuit is retained.**
- **Very versatile in circuit types and gate delays**
- **Although theoretically all faults in a circuit can be processed in one pass, memory explosion problem restricts the fault number in each pass.**

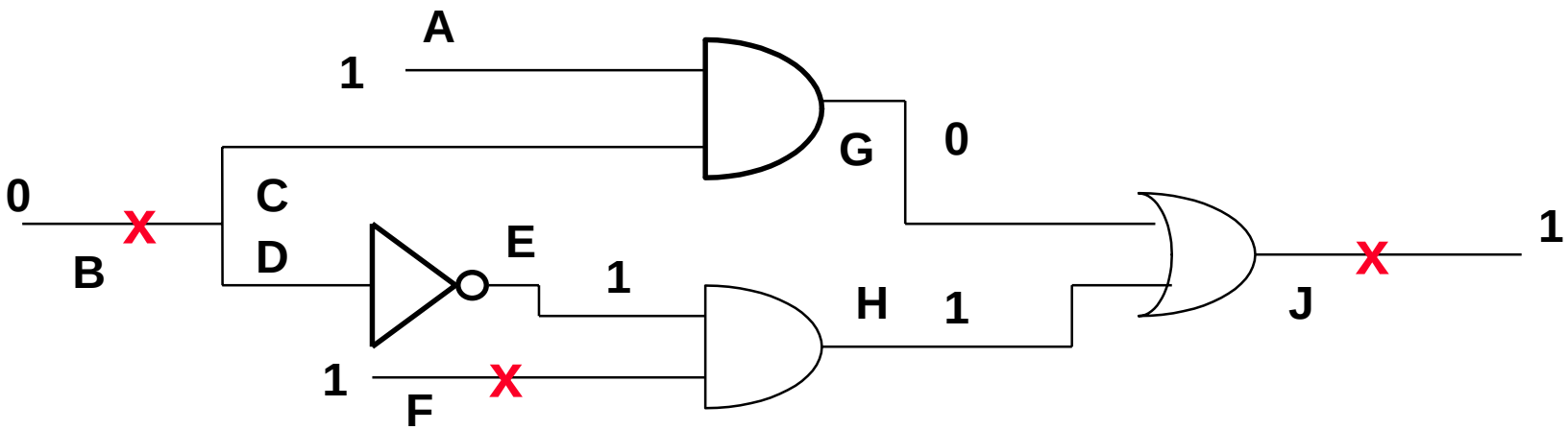
Concurrent Fault Simulation



Fault Copies: updated by both logic event and fault events

Example of Concurrent Simulation I

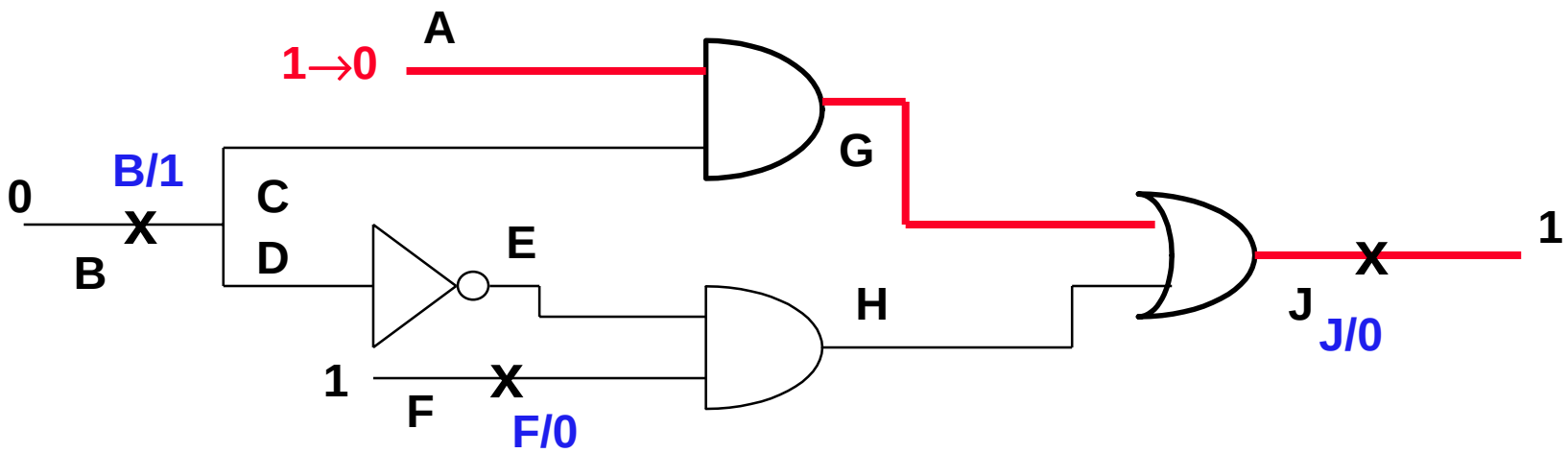
- Consider 3 faults: B/1, F/0, and J/0



L_G = {10_0, B/1:11_1} L_E = {0_1, B/1:1_0}
L_H = {11_1, B/1:01_0, F/0:10_0}
L_J = {01_1, B/1:10_1, F/0:00_0, J/0:01_0}

Example of Concurrent Simulation II

- When A changes from 1 to 0



L_G = {00_0, B/1:01_0} L_E = {0_1, B/1:1_0}

L_H = {11_1, B/1:01_0, F/0:10_0}

L_J = {01_1, B/1:00_0, F/0:00_0, J/0:01_0}

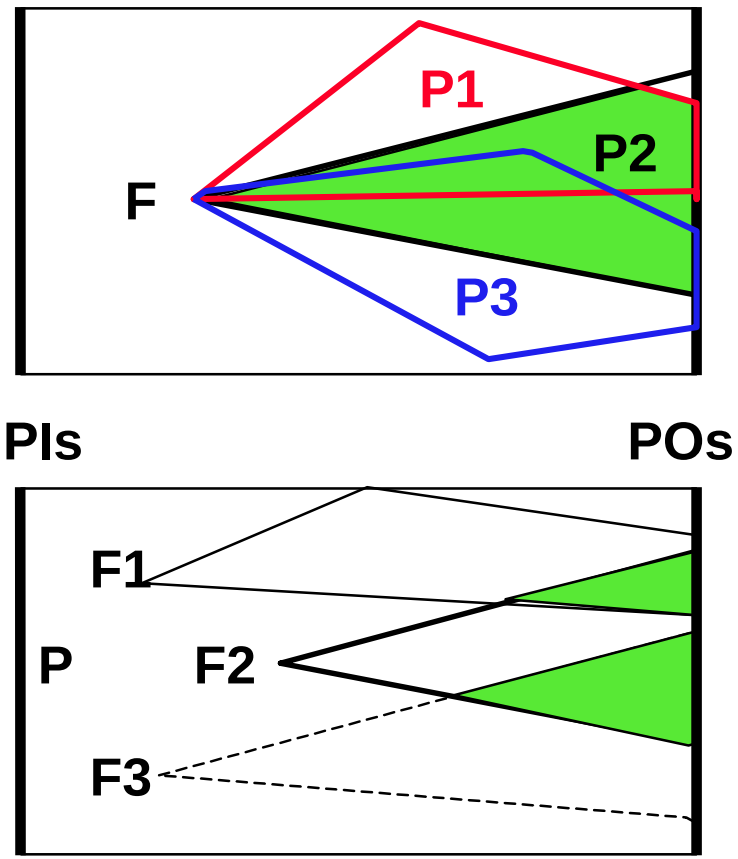
Modern Combinational Simulation Techniques

- **Parallel pattern**
- **Critical path tracing**
- **Other sophisticated techniques**

Parallel-Pattern Single-Fault Propagation (PPSFP)

- Many patterns are simulated in parallel for both fault-free and faulty circuits. The number of patterns is a multiple of computer word length.
- Coincident logic events of fault-free circuit from these patterns can be simulated in parallel.
  - reduction of logic event simulation time
- Coincident fault events of faulty circuits from these patterns can also be simulated in parallel.
  - reduction of fault event simulation time
- Simple and extremely efficient
  - basis of all modern combinational fault simulators

Comparison with Parallel-Fault



- Fault event maps for parallel-pattern (upper) and parallel-fault (lower)
- Parallel-Pattern case:

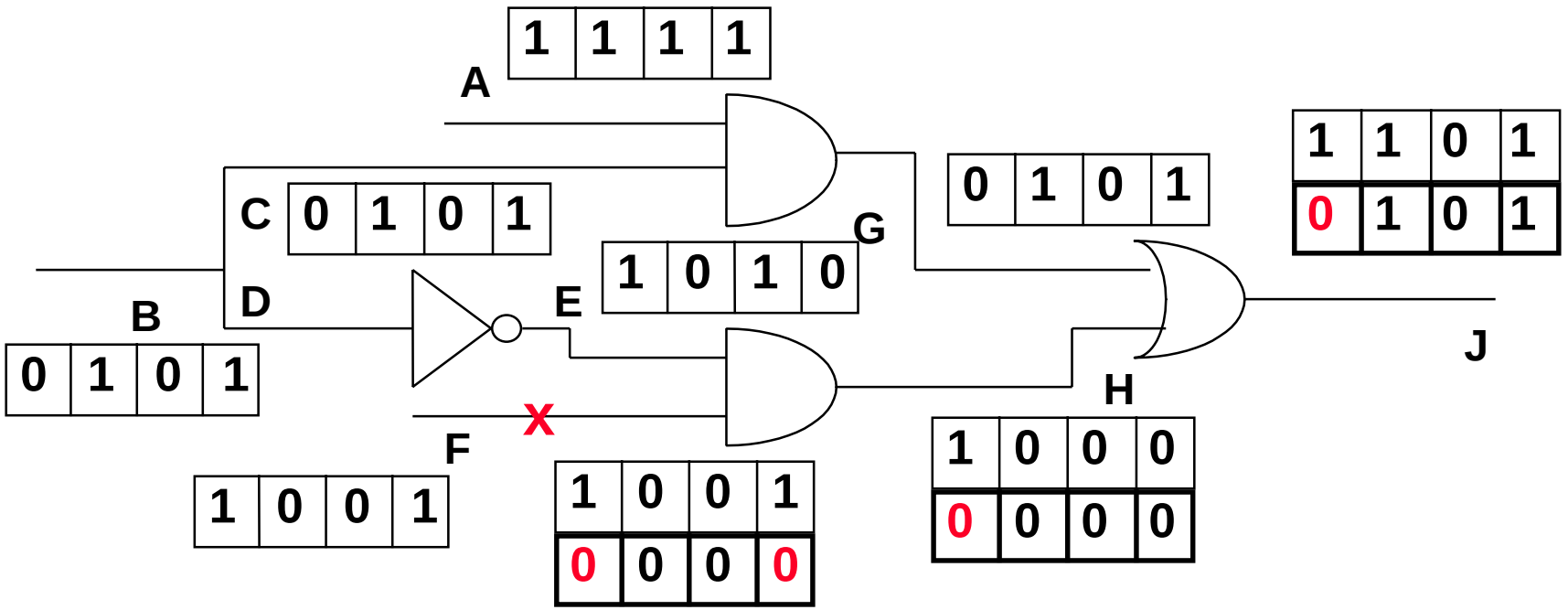
  Consider three patterns:

  P1, P2, P3.
- Parallel-Fault case:

  Consider three faults: F1, F2, F3.
- The overlapping of fault events is inherently much higher in parallel-pattern simulation, and hence more event can be done at the same time.

Example of Parallel Pattern Simulation

- **Consider one fault F/0 and four patterns: P3,P2,P1,P0**
  - **Bit-space:** | P3 | P2 | P1 | P0 | **faulty value in red**



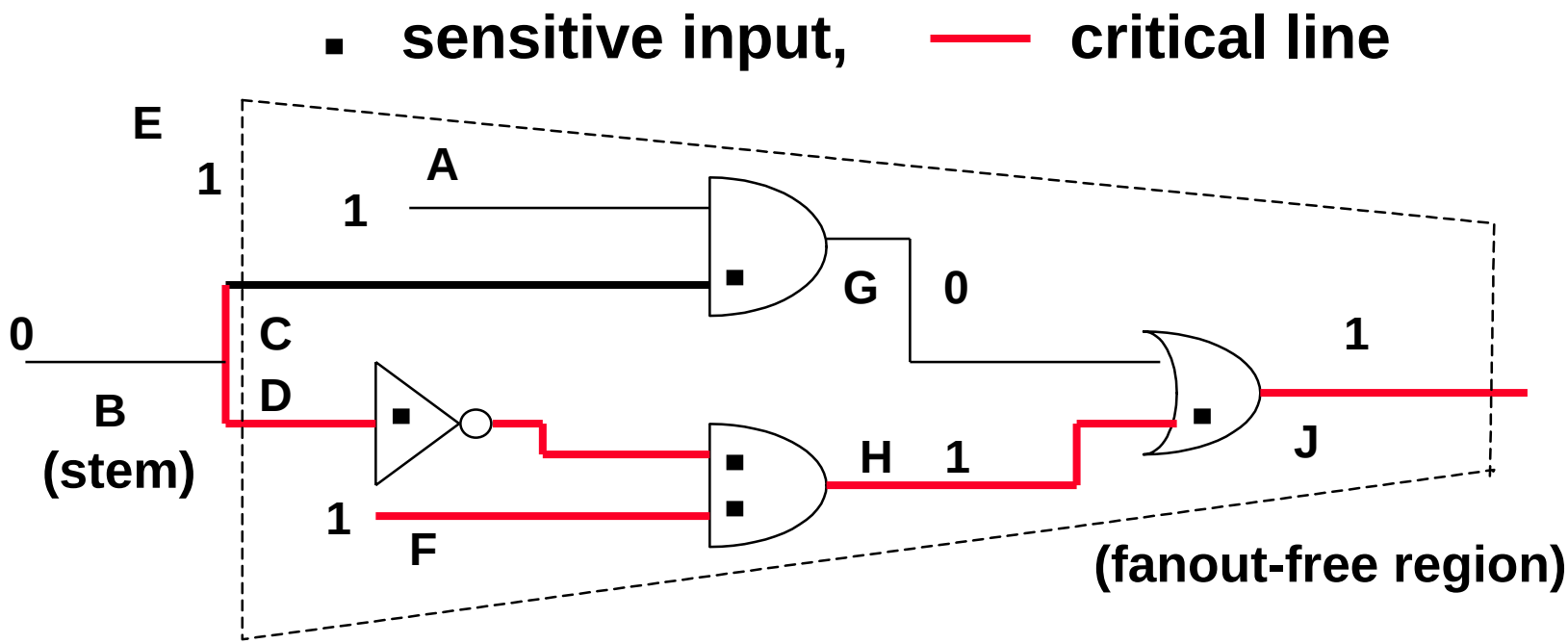
Critical Path Tracing

- **Two-step Procedure:**
  - **Fault-free simulation and identification of the sensitive gate inputs**
  - **Backtracing to identify the critical lines (critical path tracing)**
- **O(G) complexity for fanout-free circuits --- very rare in practice**
- **However, it becomes effecitve in fanout-free region when stem faults are simulated by parallel-pattern fault simulator.**

Basics of Critical Path Tracing

- A line *l* is ***critical*** w.r.t. a pattern *t* iff *t* detects the fault *l/v.*
- Paths of critical lines are critical paths.
- A gate input *i* is *sensitive* if complementing the value of *i* changes the value of the gate output.
- A gate input *i* is ***critical*** w.r.t. a pattern *t* if the gate output is critical and *i* is sensitive w.r.t. *t*.
- In a fanout-free circuit, the criticality of all lines can be determined by backward traversing the successive sensitive gate inputs from POs, in linear time.

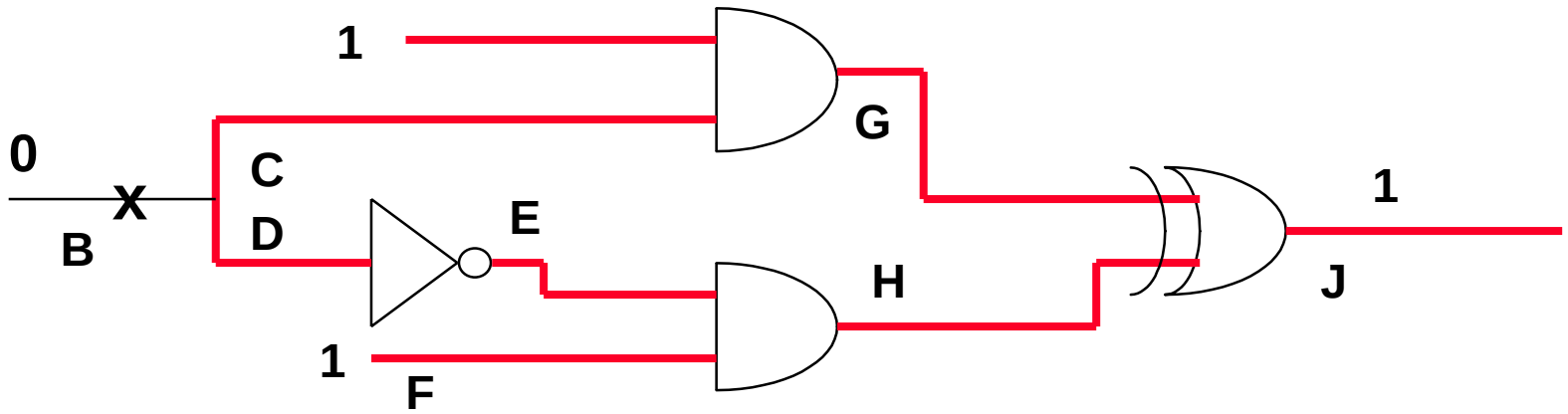
Example of Critical Path Tracing



Detected faults in the fanout-free region:
{J/0, H/0, F/0, E/0, D/1}

Anomaly of Critical Path Tracing

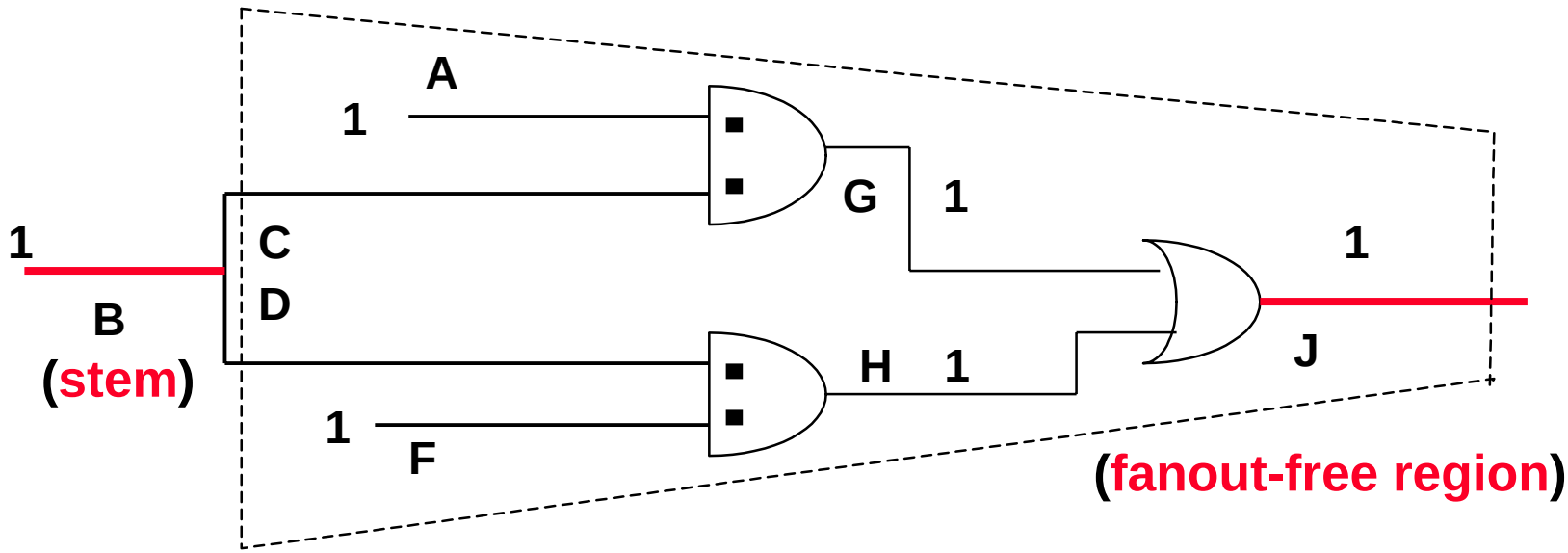
- Stem criticality is hard to infer from branches.
  E.g. is B/1 detectable by the given pattern?



- It turns out B/1 is not detectable even though both C and D are critical, because its effects cancel each other out at gate J.
- There is also multiple path sensitization problem.

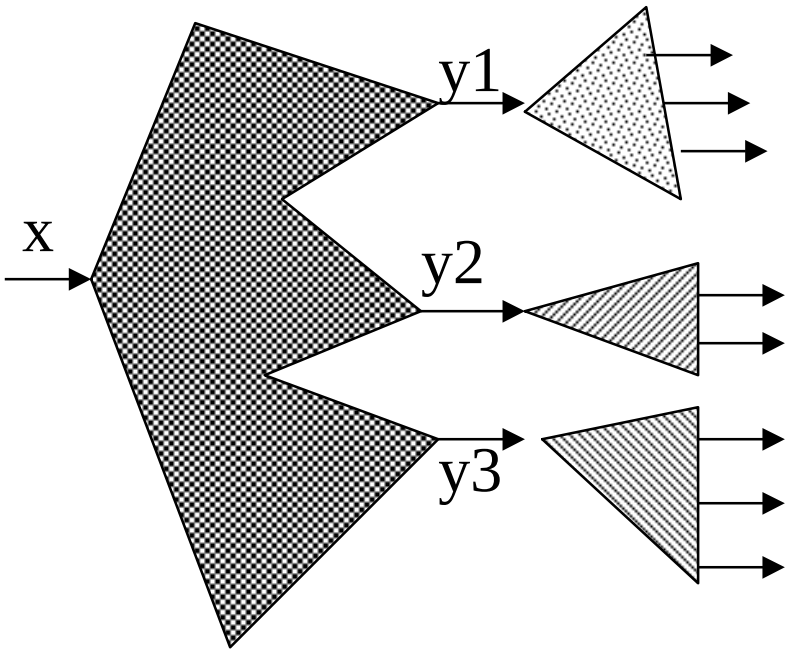
Multiple Path Sensitization

▪ sensitive input, ——— critical line



Both C and D are not critical, yet B is critical and B/0 can be detected at J by multiple path sensitization.

Other Sophisticated Techniques



d(x) = d((x-y1) d(y1) + d(x-y2) d(y2) + d(x-y3)d(y3)

- **Stem Region Methods such as Exit Lines**
  - **in the right figure, d(x) is determined from its exit lines y1, y2, and y3**
- **Multiple Packet Simulation**
  - **use more memory space to improve speed**
- **Selective or Demand-driven Fault-free Simulation**
  - **reduce unnecessary fault-free simulation as undetected faults become less**

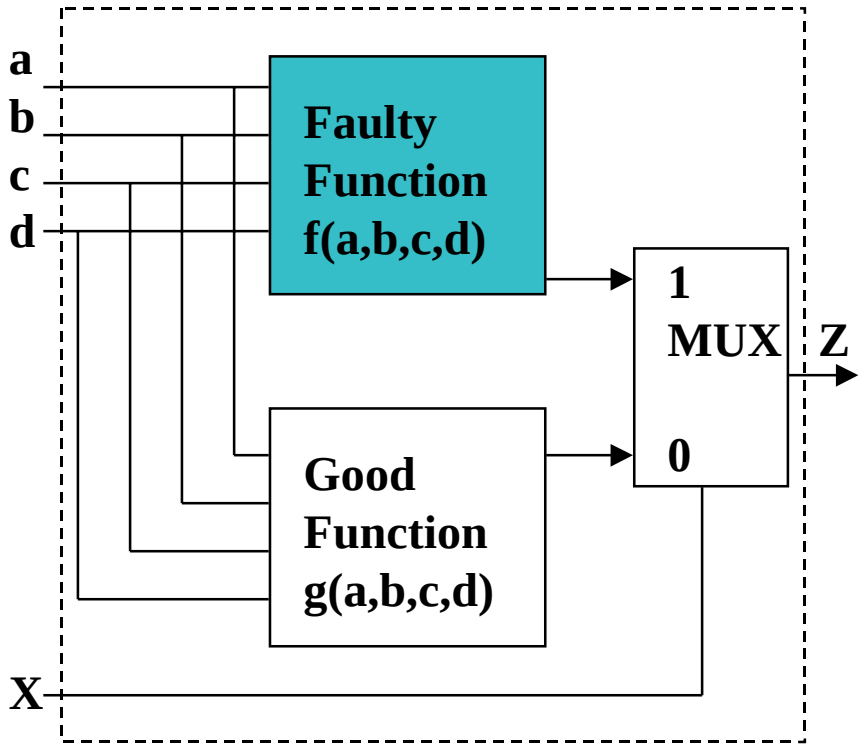
Hardware Approaches to Fault Simulation

- Commercialized Dedicated Accelerators:
  - IKOS & ZYCAD
- Other Hardware Ideas
  - Associative Memory
  - Cellular Automata
  - Field-Programmable Gate Array

Cellular Automata

- **The cellular automata is a 2D array of very simple processors with interconnection to 4 immediate neighbors. It can be regarded as a massively parallel pipelined computer.**
- **[CA95] shows both gates and interconnects of a circuit can be mapped into the cellular automata and the CA can execute logic and fault simulation with additional comparison.**
- **Mainly restricted to combinational circuits.**

Field Programmable Gate Array



A CLB with a dynamic fault injected (activated with x=1)

- FPGA is a reconfigurable gate array which can be mapped from any logic circuits and emulated much faster than software simulation.
- The fault insertion process is slow. One way to minimize the insertion is to have both good gates and faulty gates within FPGA as shown left [FPGA95].