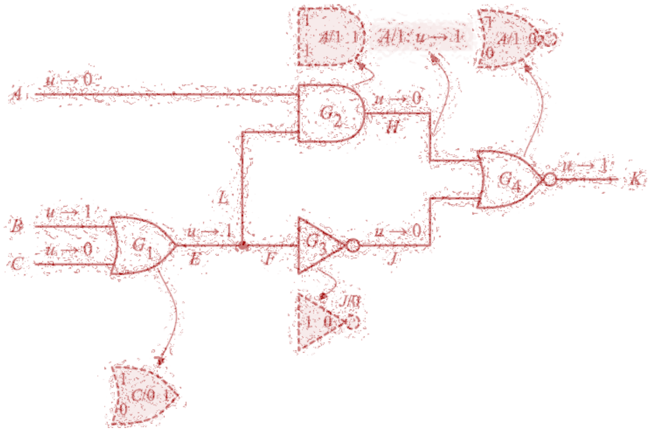
Fault Simulation

- Introduction
- **Fault simulation techniques**
  - Serial fault simulation
  - Parallel fault simulation (1965)
  - PPSFP (1985)
  - Deductive fault simulation (1972)
  - Concurrent fault simulation (1974)
  - **Differential fault simulation (1989)**
    - **Concept**
    - **Example**
- Alternatives to fault simulation
- Issues of fault simulation
- Concluding remarks



What is Fault Simulation?

- Given m faults, n patterns
  - simulate m x n 2D matrix
- G_i = Good circuit with test pattern V_i applied
- $B_{k,i}$ = Bad circuit of fault k , with test pattern V_i applied

	V_1	…	V_i	V_{i+1}	…	V_n
Good	G_1	…	G_i	G_{i+1}	…	G_n
Bad_1	$B_{1,1}$	…	$B_{1,i}$	$B_{1,i+1}$	…	$B_{1,n}$
Bad_2	$B_{2,1}$	…	$B_{2,i}$	$B_{2,i+1}$	…	$B_{2,n}$
…	…	…	…	…	…	…
Bad_k	$B_{k,1}$	…	$B_{k,i}$	$B_{k,i+1}$	…	$B_{k,n}$
Bad_{k+1}	$B_{k+1,1}$	…	$B_{k+1,i}$	$B_{k+1,i+1}$	…	$B_{k+1,n}$
…	…	…	…	…	…	…
Bad_m	$B_{m,1}$	…	$B_{m,i}$	$B_{m,i+1}$	…	$B_{m,n}$

Previous Problem (1) PPSFP

- PPSFP simulate bad circuits $B_{k,\ x}$ together
- Advantage: parallel simulation is fast

	V_1	…	V_i	V_{i+1}	…	V_n
Good	G_1	…	G_i	G_{i+1}	…	G_n
Bad_1	$B_{1,1}$	…	$B_{1,i}$	$B_{1,i+1}$	…	$B_{1,n}$
Bad_2	$B_{2,1}$	…	$B_{2,i}$	$B_{2,i+1}$	…	$B_{2,n}$
…	…	…	…	…	…	…
Bad_k	$B_{k,1}$	…	$B_{k,i}$	$B_{k,i+1}$	…	$B_{k,n}$
Bad_{k+1}	$B_{k+1,1}$	…	$B_{k+1,i}$	$B_{k+1,i+1}$	…	$B_{k+1,n}$
…	…	…	…	…	…	…
Bad_m	$B_{m,1}$	…	$B_{m,i}$	$B_{m,i+1}$	…	$B_{m,n}$

Previous Problem (1) PPSFP

- However, each simulation restarts from good circuit G_x state
- Problem : sequential circuit states not preserved
  - NOT applicable to sequential circuits

	V_1	…	V_i	V_{i+1}	…	V_n
Good	G_1	…	G_i	G_{i+1}	…	G_n
Bad_1	$B_{1,1}$	…	$B_{1,i}$	$B_{1,i+1}$	…	$B_{1,n}$
Bad_2	$B_{2,1}$	…	$B_{2,i}$	$B_{2,i+1}$	…	$B_{2,n}$
…	…	…	…	…	…	…
Bad_k	$B_{k,1}$	…	$B_{k,i}$	$B_{k,i+1}$	…	$B_{k,n}$
Bad_{k+1}	$B_{k+1,1}$	…	$B_{k+1,i}$	$B_{k+1,i+1}$	…	$B_{k+1,n}$
…	…	…	…	…	…	…
Bad_m	$B_{m,1}$	…	$B_{m,i}$	$B_{m,i+1}$	…	$B_{m,n}$

Previous Problem (2) Concurrent

- Concurrent simulate bad circuits $B_{x,\,i+1}$ from previous $B_{x,i}$
- Advantage: preserve circuit states
  - applicable to sequential circuits

	V_1	…	V_i	V_{i+1}	…	V_n
Good	G_1	…	G_i	G_{i+1}	…	G_n
Bad_1	$B_{1,1}$	…	$B_{1,i}$	$B_{1,i+1}$	…	$B_{1,n}$
Bad_2	$B_{2,1}$	…	$B_{2,i}$	$B_{2,i+1}$	…	$B_{2,n}$
…	…	…	…	…	…	…
Bad_k	$B_{k,1}$	…	$B_{k,i}$	$B_{k,i+1}$	…	$B_{k,n}$
Bad_{k+1}	$B_{k+1,1}$	…	$B_{k+1,i}$	$B_{k+1,i+1}$	…	$B_{k+1,n}$
…	…	…	…	…	…	…
Bad_m	$B_{m,1}$	…	$B_{m,i}$	$B_{m,i+1}$	…	$B_{m,n}$

Previous Problem (2) Concurrent

- Need to store **ALL states of $B_{k,i}$** so that we can simulate $B_{k,i+1}$
- Problem: **memory management difficult**

	V_1	…	V_i	V_{i+1}	…	V_n
Good	G_1	…	G_i	G_{i+1}	…	G_n
Bad_1	$B_{1,1}$	…	$B_{1,i}$	$B_{1,i+1}$	…	$B_{1,n}$
Bad_2	$B_{2,1}$	…	$B_{2,i}$	$B_{2,i+1}$	…	$B_{2,n}$
…	…	…	…	…	…	…
Bad_k	$B_{k,1}$	…	$B_{k,i}$	$B_{k,i+1}$	…	$B_{k,n}$
Bad_{k+1}	$B_{k+1,1}$	…	$B_{k+1,i}$	$B_{k+1,i+1}$	…	$B_{k+1,n}$
…	…	…	…	…	…	…
Bad_m	$B_{m,1}$	…	$B_{m,i}$	$B_{m,i+1}$	…	$B_{m,n}$

Quiz

Q: Which of the following is CORRECT?

A) PPSFP is applicable to sequential circuits
B) concurrent fault simulation is applicable to sequential circuits
C) PPSFP wastes memory to store FF states

Differential Fault Sim. [Cheng 89]

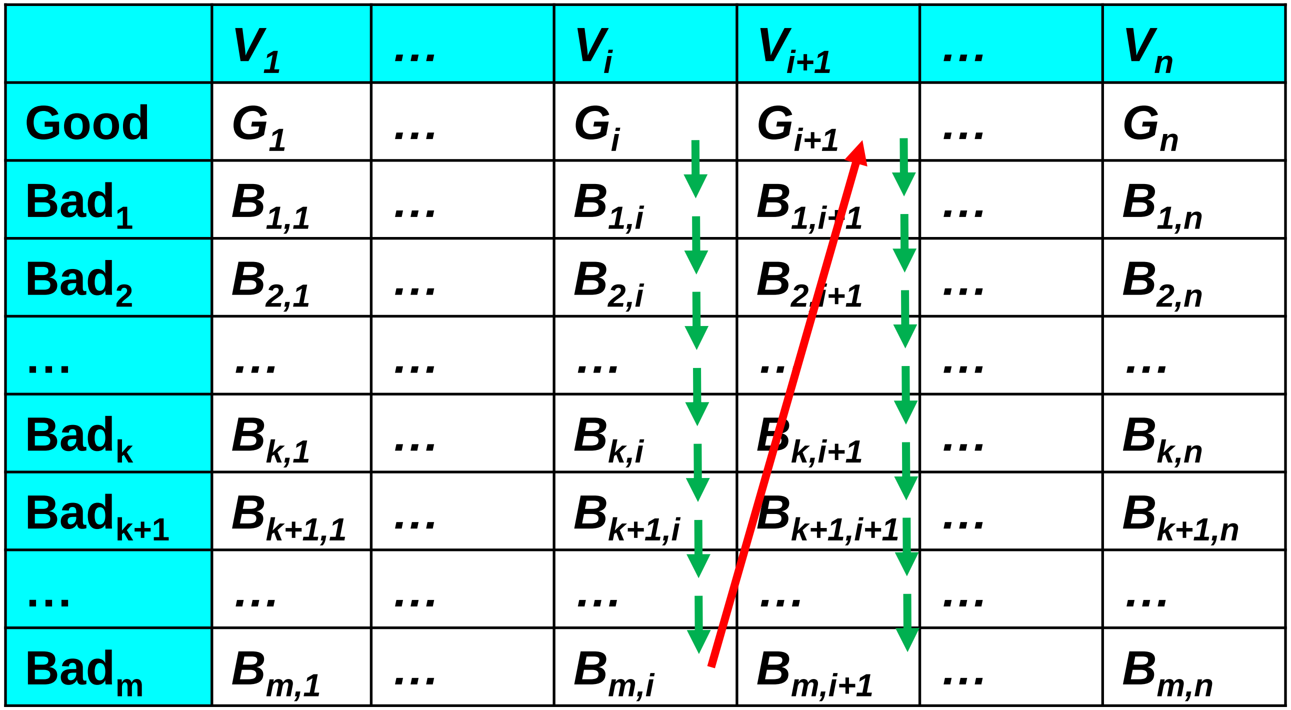
- *Differential fault simulation (DSIM)*
  - Simulate differences from the last simulated circuit

- Better than PPSFP
  - Simulate one pattern at a time
    - Preserve sequential circuit states
- Better than concurrent
  - Record circuit states DIFFERENCE between $B_{k,i}$ and $B_{k+1,i}$
    - Reduce memory
- Up to 12 times faster than concurrent, PPSFP

DSIM Combines Concurrent and PPSFP

DSIM Idea

- Simulates one pattern at a time
- First good circuit, then faulty circuits



	V_1	…	V_i	V_{i+1}	…	V_n
Good	G_1	…	G_i	G_{i+1}	…	G_n
Bad_1	$B_{1,1}$	…	$B_{1,i}$	$B_{1,i+1}$	…	$B_{1,n}$
Bad_2	$B_{2,1}$	…	$B_{2,i}$	$B_{2,i+1}$	…	$B_{2,n}$
…	…	…	…	…	…	…
Bad_k	$B_{k,1}$	…	$B_{k,i}$	$B_{k,i+1}$	…	$B_{k,n}$
Bad_{k+1}	$B_{k+1,1}$	…	$B_{k+1,i}$	$B_{k+1,i+1}$	…	$B_{k+1,n}$
…	…	…	…	…	…	…
Bad_m	$B_{m,1}$	…	$B_{m,i}$	$B_{m,i+1}$	…	$B_{m,n}$

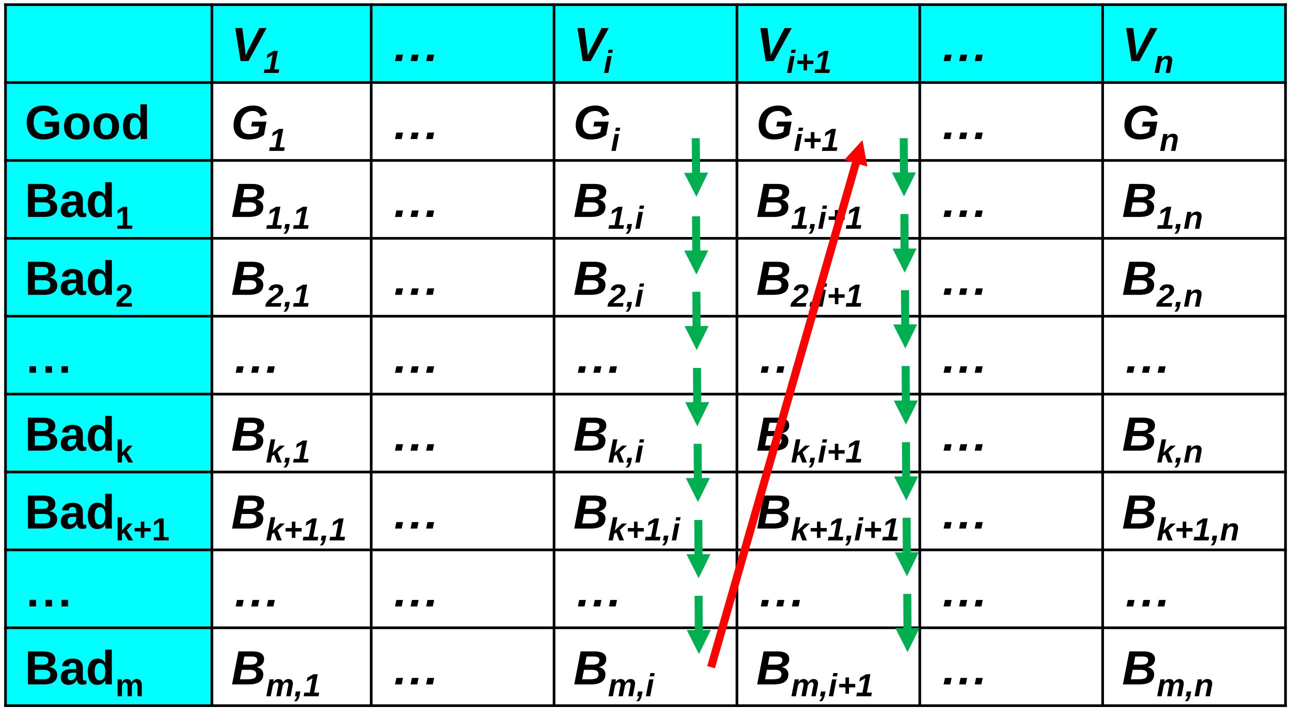
DSIM Idea (2)

- Simulates one pattern at a time
- First good circuit, then faulty circuits

	V_1	…	V_i	V_{i+1}	…	V_n
Good	G_1	…	G_i	G_{i+1}	…	G_n
Bad_1	$B_{1,1}$	…	$B_{1,i}$	$B_{1,i+1}$	…	$B_{1,n}$
Bad_2	$B_{2,1}$	…	$B_{2,i}$	$B_{2,i+1}$	…	$B_{2,n}$
…	…	…	…	…	…	…
Bad_k	$B_{k,1}$	…	$B_{k,i}$	$B_{k,i+1}$	…	$B_{k,n}$
Bad_{k+1}	$B_{k+1,1}$	…	$B_{k+1,i}$	$B_{k+1,i+1}$	…	$B_{k+1,n}$
…	…	…	…	…	…	…
Bad_m	$B_{m,1}$	…	$B_{m,i}$	$B_{m,i+1}$	…	$B_{m,n}$

DSIM Idea (3)

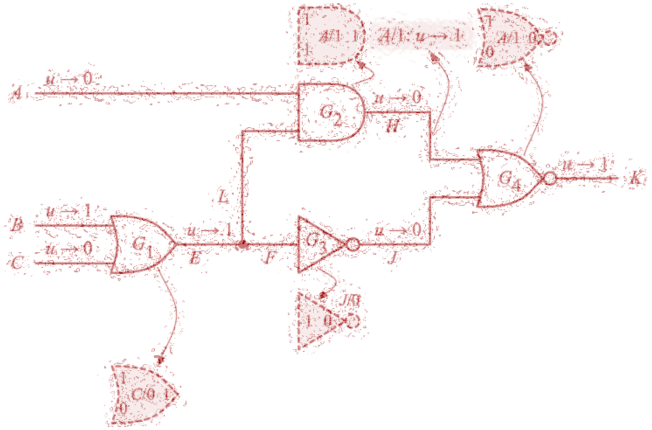
- Recover sequential circuit state
  - from **last simulated circuit**



	V_1	…	V_i	V_{i+1}	…	V_n
Good	G_1	…	G_i	G_{i+1}	…	G_n
Bad_1	$B_{1,1}$	…	$B_{1,i}$	$B_{1,i+1}$	…	$B_{1,n}$
Bad_2	$B_{2,1}$	…	$B_{2,i}$	$B_{2,i+1}$	…	$B_{2,n}$
…	…	…	…	…	…	…
Bad_k	$B_{k,1}$	…	$B_{k,i}$	$B_{k,i+1}$	…	$B_{k,n}$
Bad_{k+1}	$B_{k+1,1}$	…	$B_{k+1,i}$	$B_{k+1,i+1}$	…	$B_{k+1,n}$
…	…	…	…	…	…	…
Bad_m	$B_{m,1}$	…	$B_{m,i}$	$B_{m,i+1}$	…	$B_{m,n}$

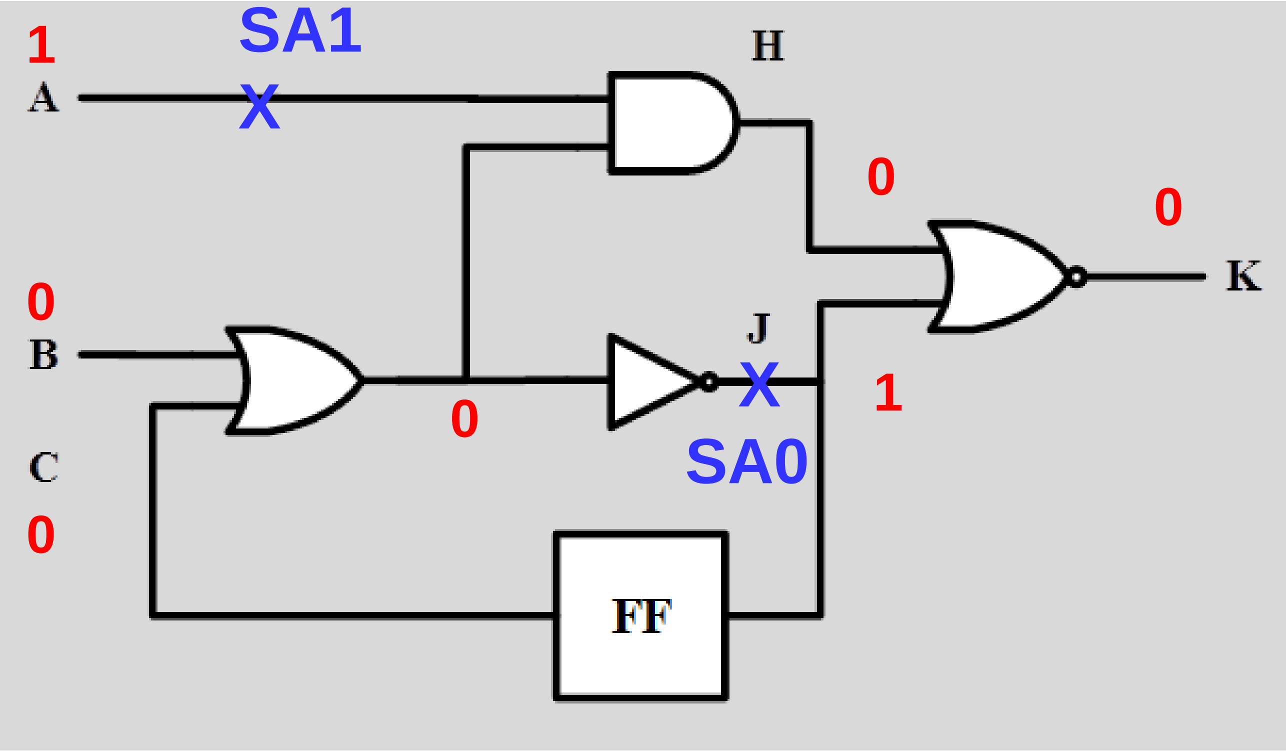
Fault Simulation

- Introduction
- **Fault simulation techniques**
  - Serial fault simulation
  - Parallel fault simulation (1965)
  - PPSFP (1985)
  - Deductive fault simulation (1972)
  - Concurrent fault simulation (1974)
  - **Differential fault simulation (1989)**
    - **Concept**
    - **Example**
- Alternatives to fault simulation
- Issues of fault simulation
- Concluding remarks



Example

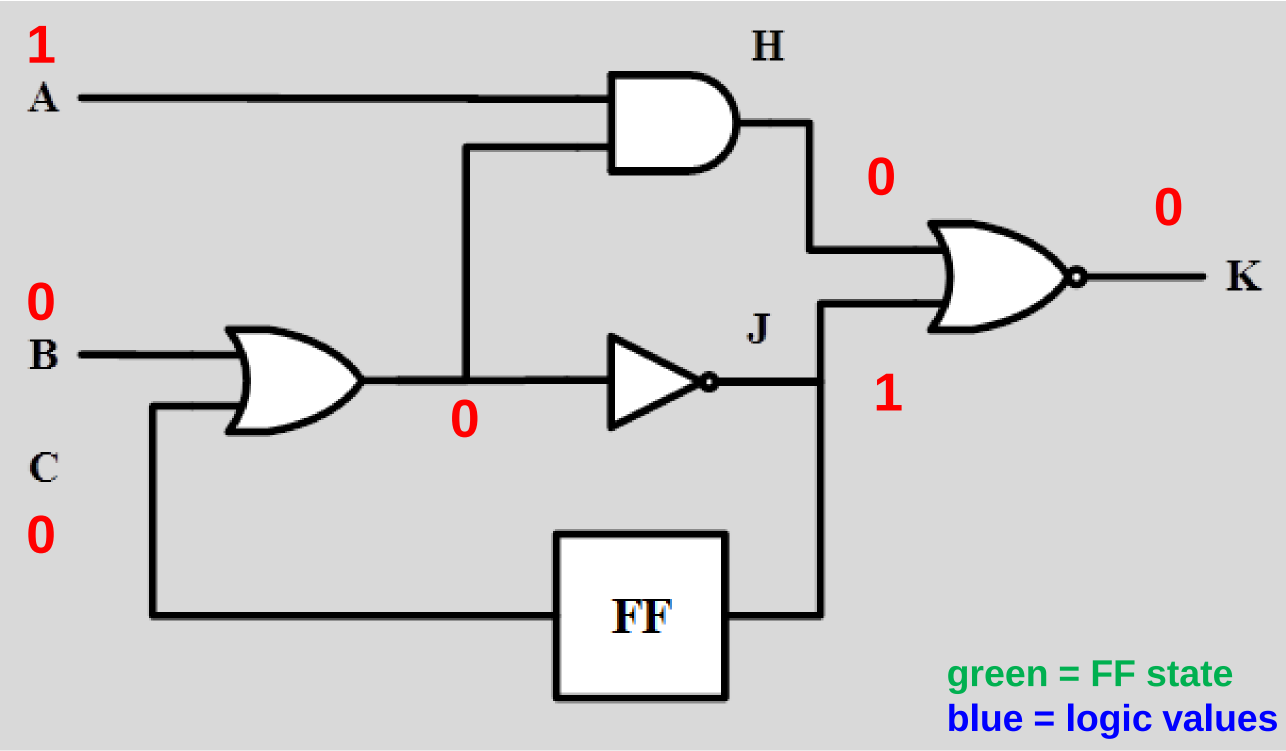
- Apply two patterns: AB=10 , 00 . Initially FF state =0
- Consider two faults: A SA1, J SA0
  - No fault dropping for demo purpose



V_1 $AB=10$	V_2 $AB=00$
G_1 FF=0 (given)	G_2
$B_{1,1}$ A SA1	$B_{1,2}$ A SA1
$B_{2,1}$ J SA0	$B_{2,2}$ J SA0

Example (1)

- Current state FF=0. Apply AB=10 to good circuit
  - FF^{+}=1 ➔ store next FF state
  - K=0 ➔ store current PO



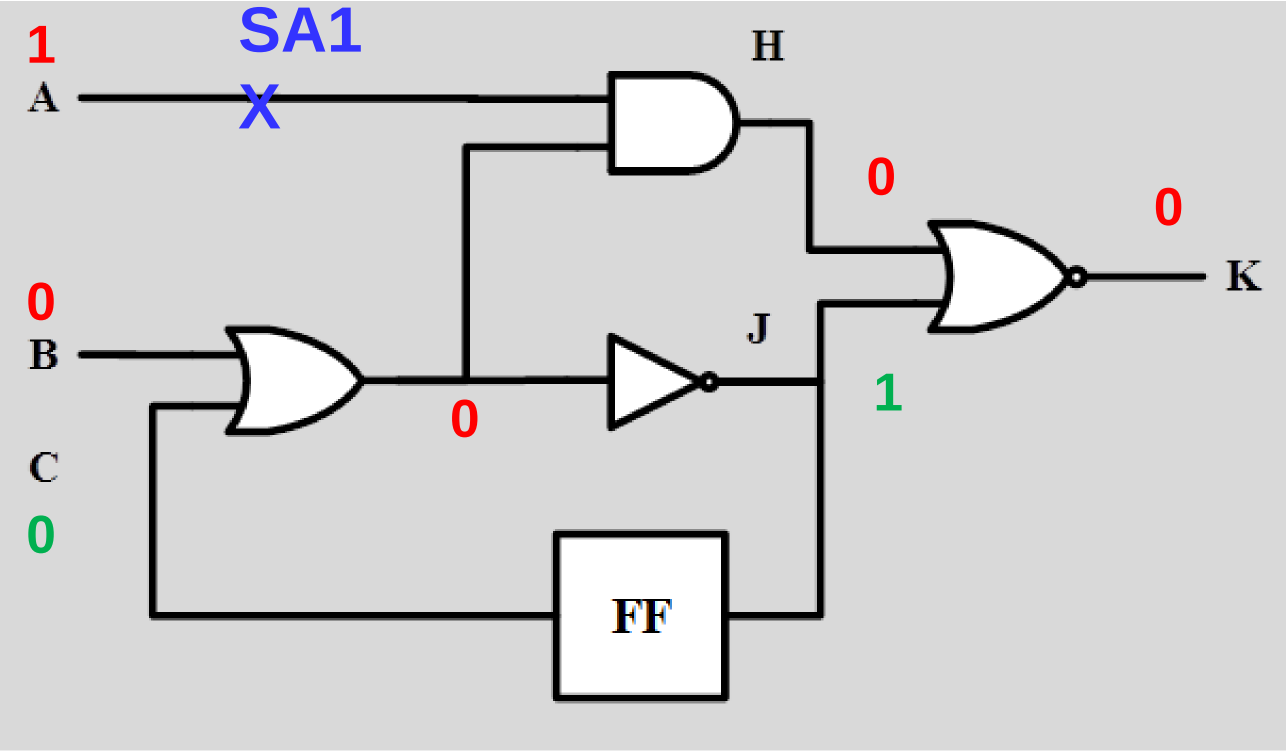
green = FF state
blue = logic values

V_1 AB=10	V_2 AB=00
G_1 $K=0$ $FF^{+}=1$	G_2
$B_{1,1}$	$B_{1,2}$
$B_{2,1}$	$B_{2,2}$

FF^{+} = next FF

Example (2)

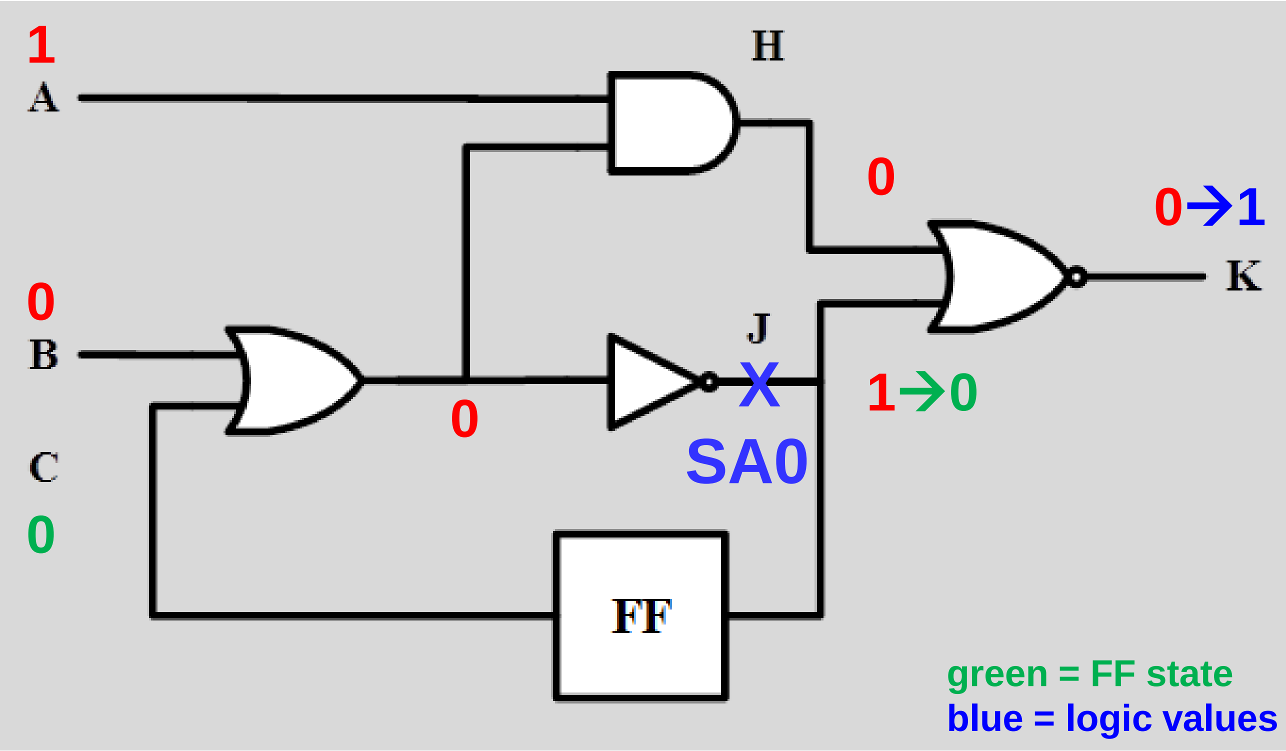
- Inject A SA1, C=0
  - FF$^+$=1 same as G_1 ➔ no need to store state difference
  - K=0 same as G_1 ➔ fault undetected



V_1 AB=10	V_2 AB=00
G_1 K=0 FF^+=1	G_2
$B_{1,1}$	$B_{1,2}$
$B_{2,1}$	$B_{2,2}$

Example (3)

- Remove A SA1. Inject J SA0
  - $FF^+=1$ different from $B_{1,1}$ ➔ store state difference from $B_{1,1}$
  - K different from G_1 ➔ fault detected



V_1 $AB=10$	V_2 $AB=00$
G_1 $K=0$ $FF^+=1$	G_2
$B_{1,1}$	$B_{1,2}$
$B_{2,1}$ $K=1$ $FF^+= 0$	$B_{2,2}$

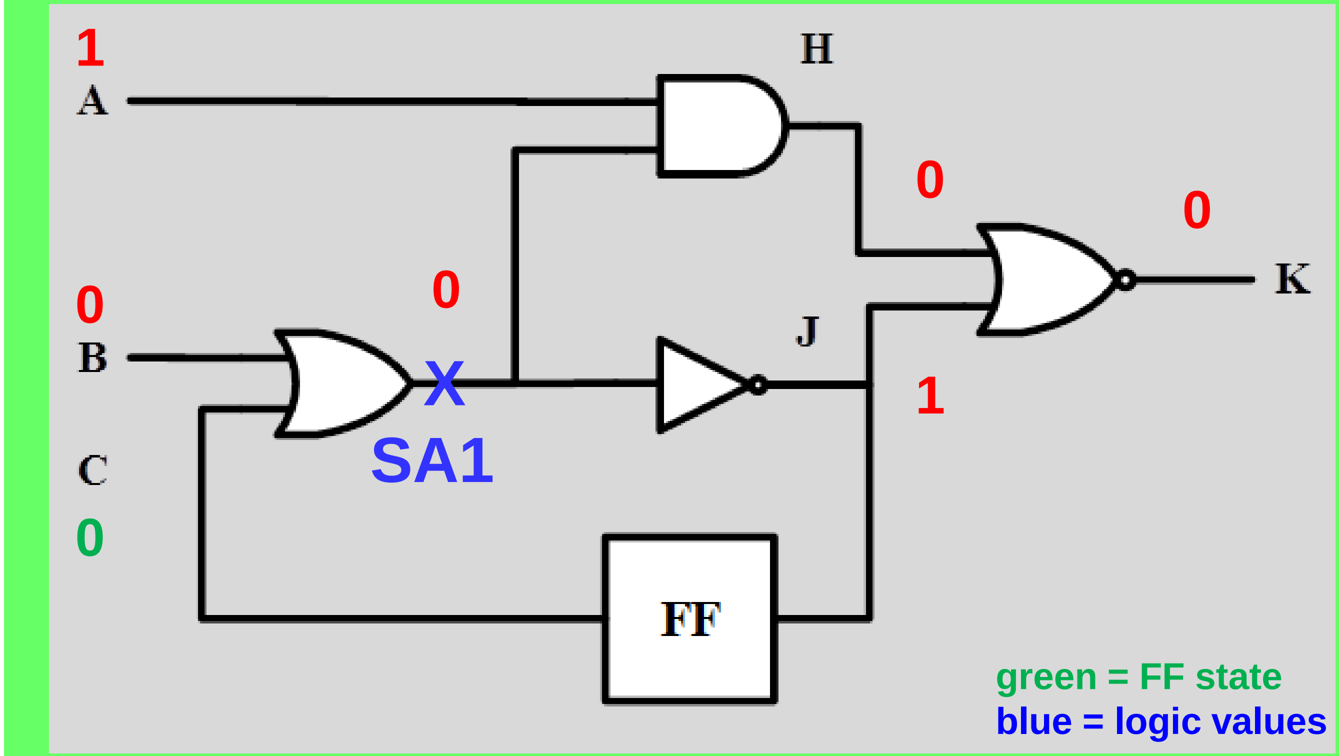
FF^+ = next FF

QUIZ

Q1: Consider a third fault, please fill in ?

Q2: What is K? Is the fault detected?

Q3: What is FF^+ ? Will this be recorded?

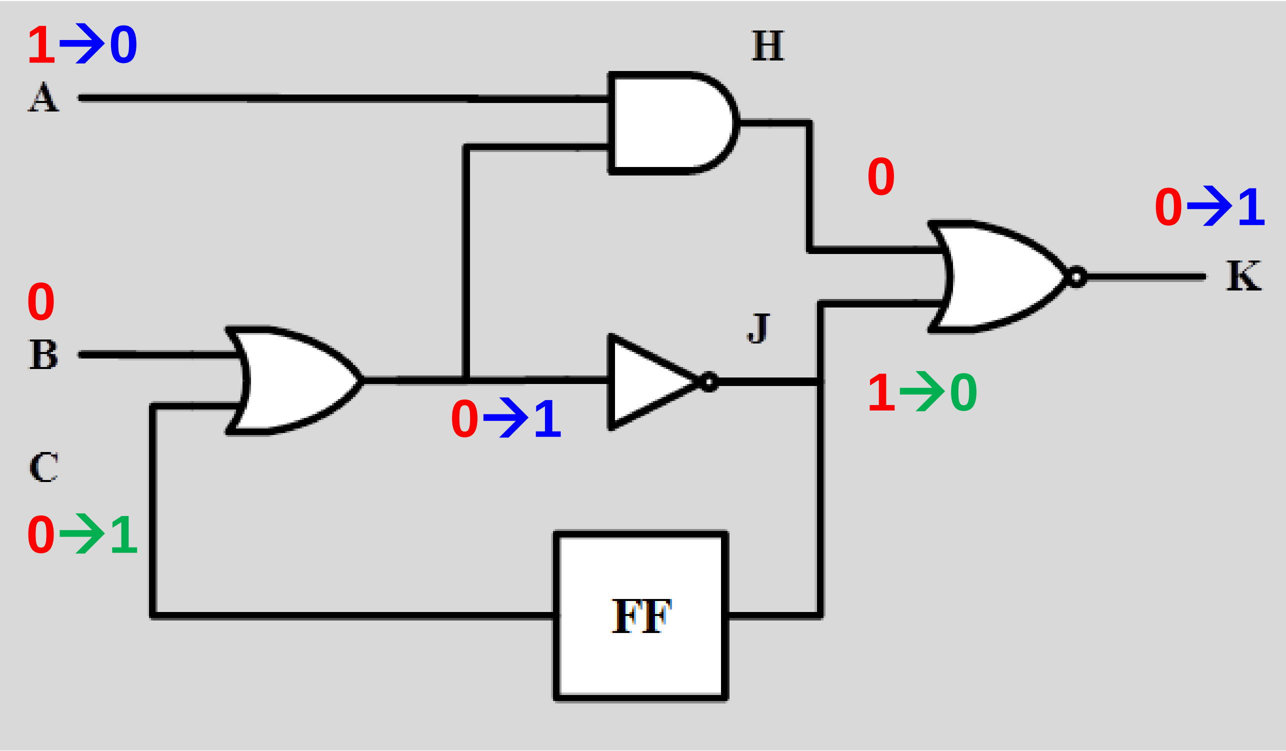


green = FF state
blue = logic values

V_1 $AB=10$	V_2 $AB=00$
G_1 $K=0$ $FF^+=1$	G_2
$B_{1,1}$	$B_{1,2}$
$B_{2,1}$ $K=1$ $FF^+= 0$	$B_{2,2}$
$B_{3,1}$ $K=?$ $FF^+=?$	

Example (4)

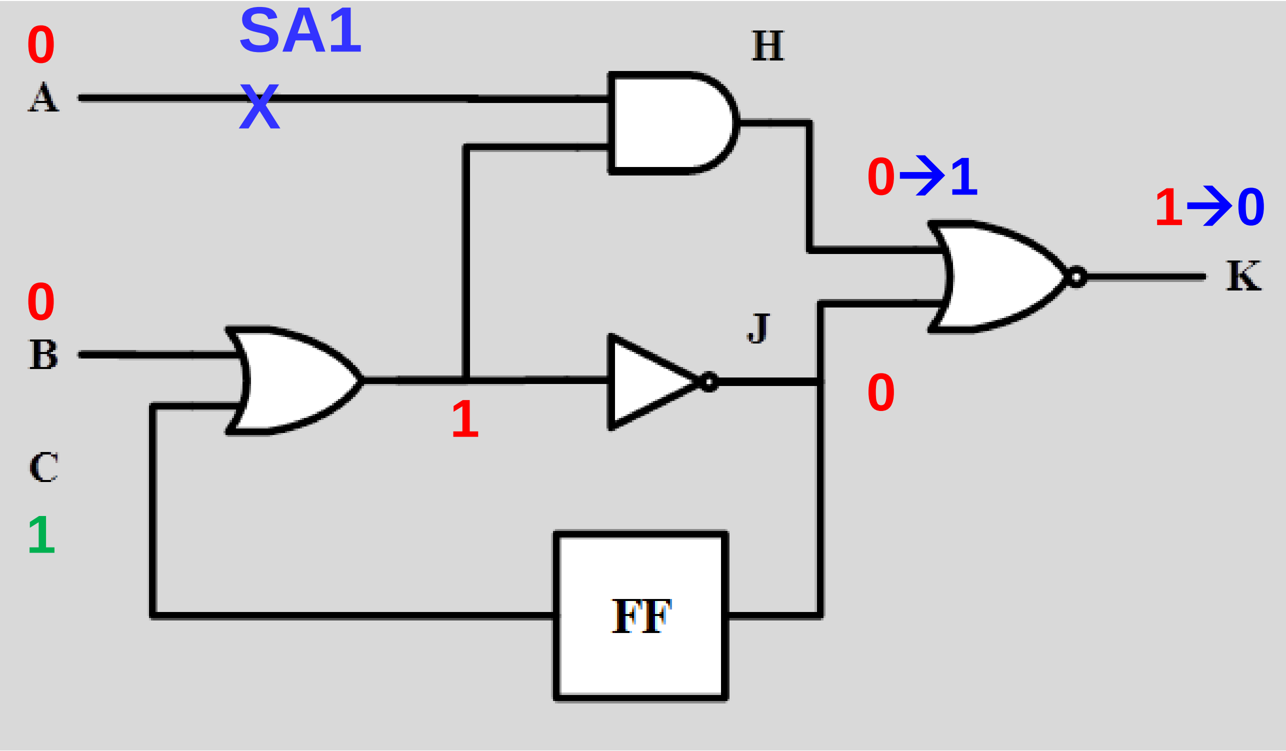
- **Remove J SA0. Apply V_2: A=1→0. Restore good state: C=0→1**
  - FF$^+$=0 → store next FF state
  - K=1 → store current PO



V_1 AB=10	V_2 AB=00
G_1 K=0 FF$^+$=1	→ G_2 K=1 FF$^+$=0
$B_{1,1}$	$B_{1,2}$
$B_{2,1}$ FF$^+$=0	$B_{2,2}$

Example (5)

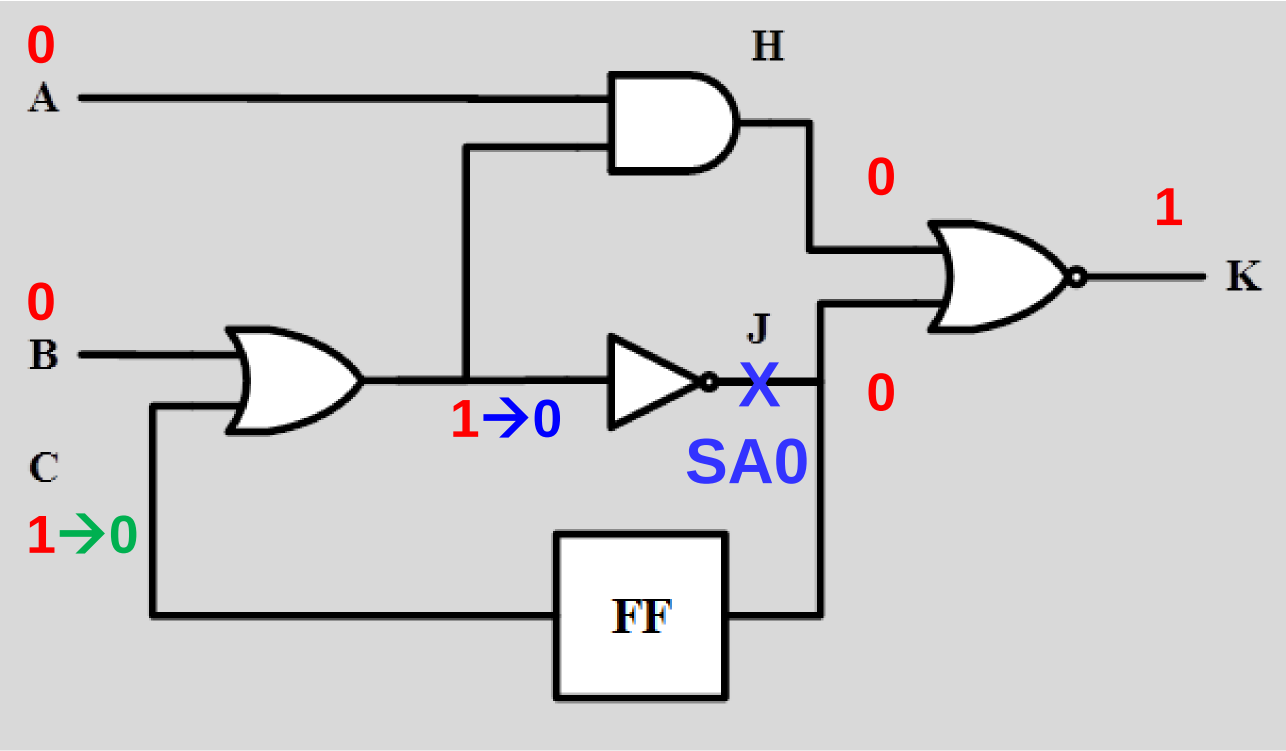
- Inject A SA1. A=0, C=1 (same as good)
  - FF$^+$=0 same as G_2 ➔ no need to store FF state difference
  - K different from G_2 ➔ fault detected



V_1 $AB=10$	V_2 $AB=00$
G_1	G_2 $K=1$ $FF^+=0$
$B_{1,1}$	$B_{1,2}$ $K=0$
$B_{2,1}$ $FF^+=0$	$B_{2,2}$

Example (6)

- Restore state from $B_{2,1}$: C=1→0
  - FF$^+$=0 same as $B_{1,2}$ ➔ no need to store FF state difference
  - K same as G_2 ➔ fault undetected



V_1 AB=10	V_2 AB=00
G_1	G_2 K=1 FF$^+$=0
$B_{1,1}$	$B_{1,2}$
$B_{2,1}$ FF$^+$=0	$B_{2,2}$

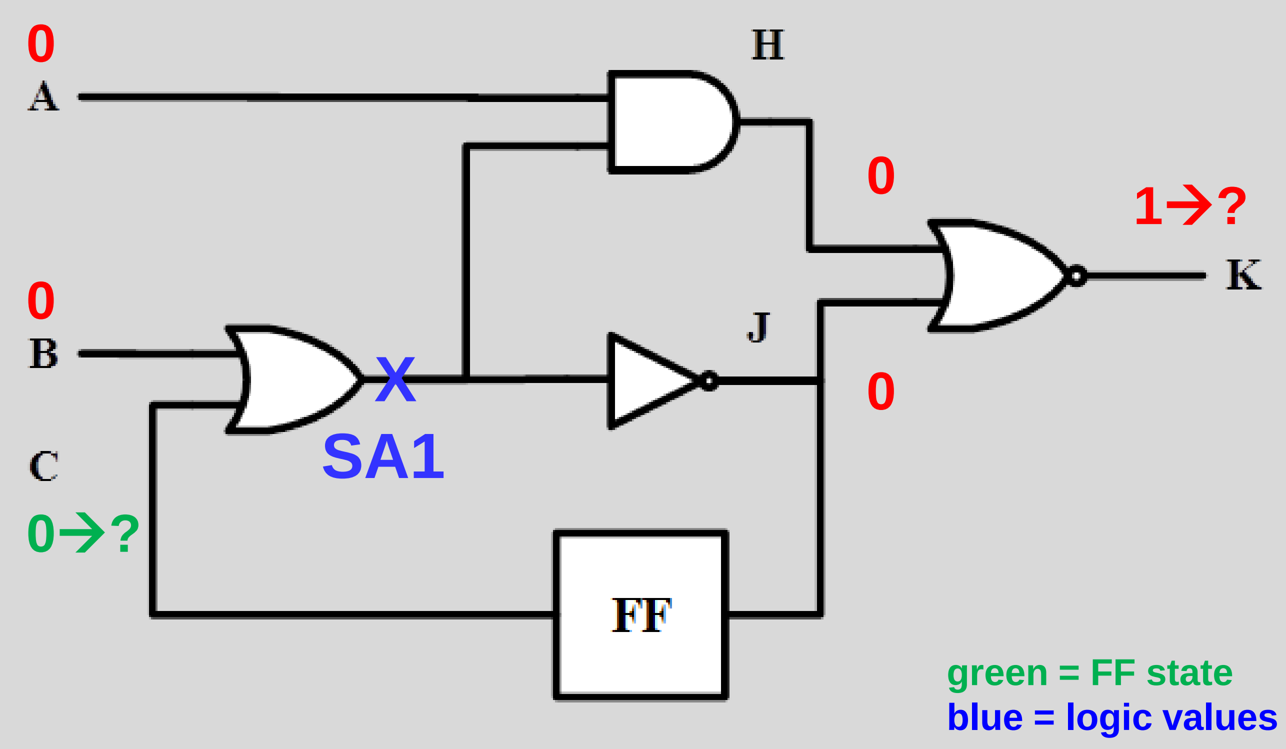
Very Small Memory Required

QUIZ

Q1: Consider a third fault, please fill in ?

Q2: What is K? Is the fault detected?

Q3: What is FF^+ ? Will this be recorded?

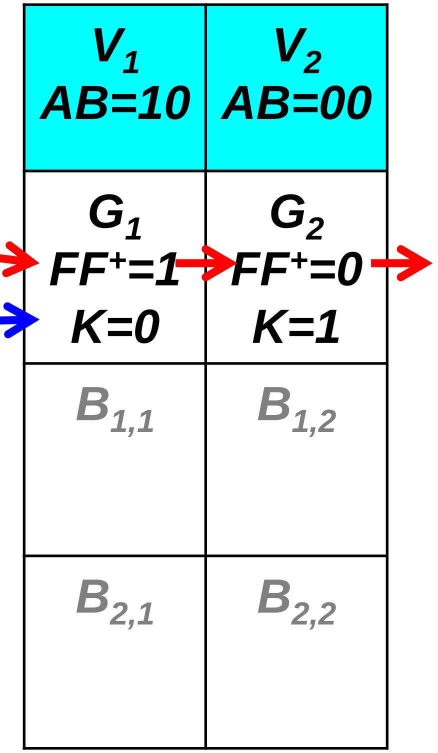


green = FF state
blue = logic values

V_1 AB=10	V_2 AB=00
G_1	G_2 K=1 FF^+=0
$B_{1,1}$	$B_{1,2}$
$B_{2,1}$ FF^+=0	$B_{2,2}$
$B_{3,1}$	$B_{3,2}$ K=? FF^+=?

DFSIM Algorithm

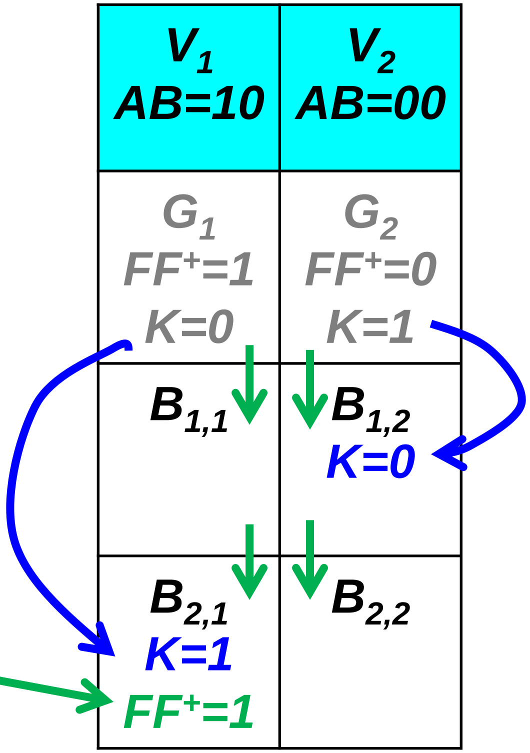
```
foreach test vector $V_i$
   if ($V_i$ is first vector)
       initialize ckt states
   else
       remove previously injected fault
   recover current states
   set $V_i$ pattern at primary inputs
   event-driven simulation
   store PO
   sensitized output counter =0
   foreach undetected faulty machine $B_j$
      remove previous injected fault
      recover current states
      inject current fault
      event-driven simulation
      if sensitized output counter >0
           drop the fault
```



V_1 AB=10	V_2 AB=00
G_1 $FF^+=1$ $K=0$	G_2 $FF^+=0$ $K=1$
$B_{1,1}$	$B_{1,2}$
$B_{2,1}$	$B_{2,2}$

DFSIM Algorithm

```
foreach test vector $V_i$
  if ($V_i$ is first vector)
     initialize ckt status
  else
     remove previously injected fault
  recover current states
  set $V_i$ pattern at primary inputs
  event-driven simulation
  store PO
  sensitized output counter =0
  foreach undetected faulty machine $B_j$
    remove previous injected fault
    recover current states
    inject current fault
    event-driven simulation
    if sensitized output counter >0
        drop the fault
```



V_1 AB=10	V_2 AB=00
G_1 $FF^+=1$ K=0	G_2 $FF^+=0$ K=1
$B_{1,1}$	$B_{1,2}$ K=0
$B_{2,1}$ K=1 $FF^+=1$	$B_{2,2}$

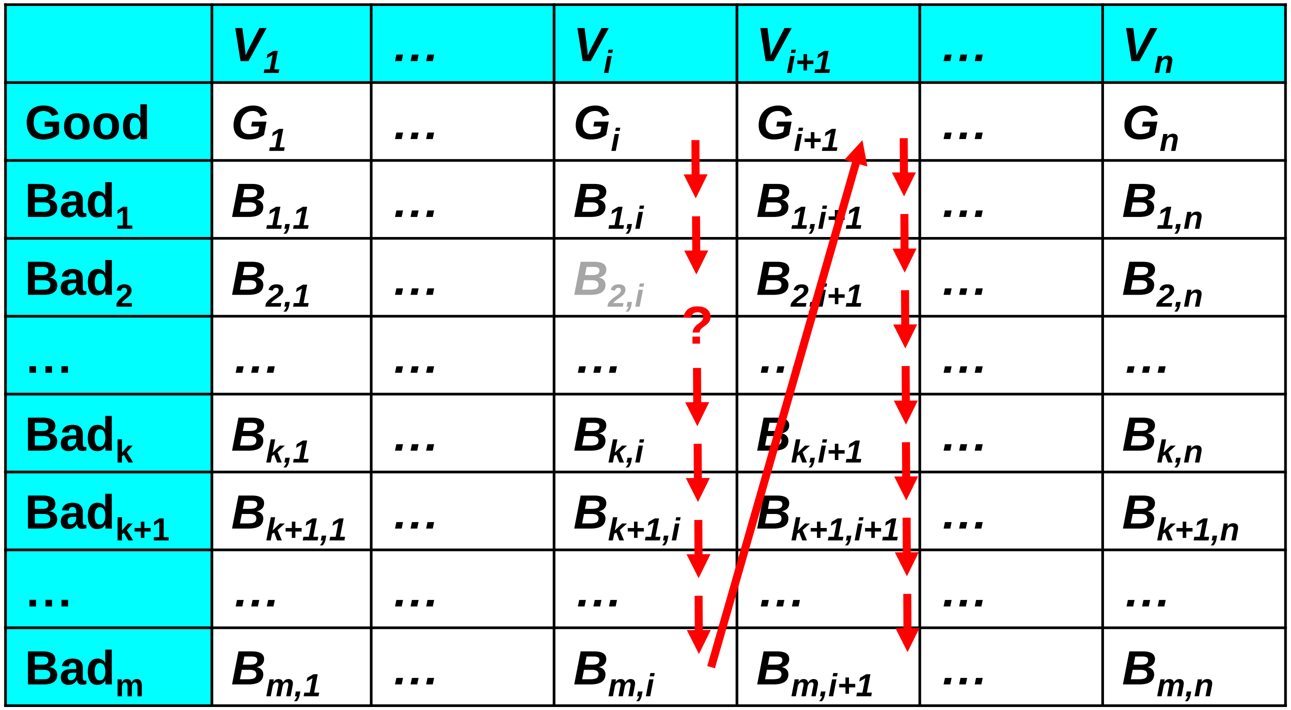
Summary

- Differential fault simulation
  - ♦ Simulate one vector, one circuit at a time
  - ♦ Restore state difference from previous circuit simulated
- Advantages
  - ☺ Applicable to sequential circuits
    - ∗ Better than PPSFP fault simulation
  - ☺ Require very small memory
    - ∗ Better than concurrent fault simulation
  - ☺ Event-driven simulation
    - ∗ Handles delay faults

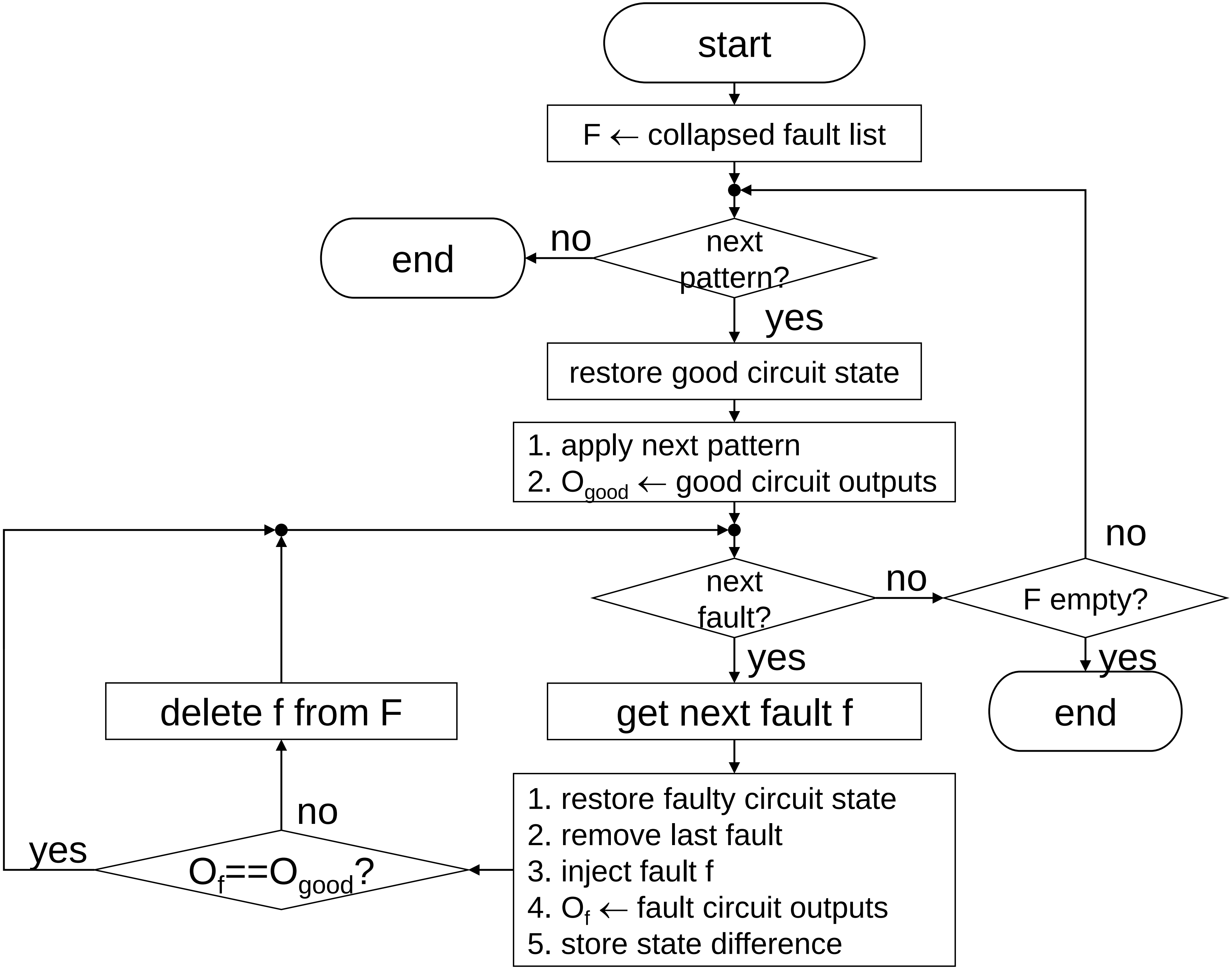
DSIM Very Useful for Seq. Ckt.

FFT

- Q: If a fault is dropped, how can we restore FF state of next fault?
  - why do not we record FF state of all Good FF



	V_1	…	V_i	V_{i+1}	…	V_n
Good	G_1	…	G_i	G_{i+1}	…	G_n
Bad$_1$	$B_{1,1}$	…	$B_{1,i}$	$B_{1,i+1}$	…	$B_{1,n}$
Bad$_2$	$B_{2,1}$	…	$B_{2,i}$	$B_{2,i+1}$	…	$B_{2,n}$
…	…	…	…	…	…	…
Bad$_k$	$B_{k,1}$	…	$B_{k,i}$	$B_{k,i+1}$	…	$B_{k,n}$
Bad$_{k+1}$	$B_{k+1,1}$	…	$B_{k+1,i}$	$B_{k+1,i+1}$	…	$B_{k+1,n}$
…	…	…	…	…	…	…
Bad$_m$	$B_{m,1}$	…	$B_{m,i}$	$B_{m,i+1}$	…	$B_{m,n}$


start
F ← collapsed fault list
next pattern?
no
end
yes
restore good circuit state
1. apply next pattern
2. O$_{good}$ ← good circuit outputs
next fault?
no
F empty?
no
yes
end
yes
get next fault f
1. restore faulty circuit state
2. remove last fault
3. inject fault f
4. O$_f$ ← fault circuit outputs
5. store state difference
O$_f$==O$_{good}$?
yes
no
delete f from F