EEDG/CE 6303: Testing and Testable Design

Mehrdad Nourani

Dept. of ECE
Univ. of Texas at Dallas

Session 06

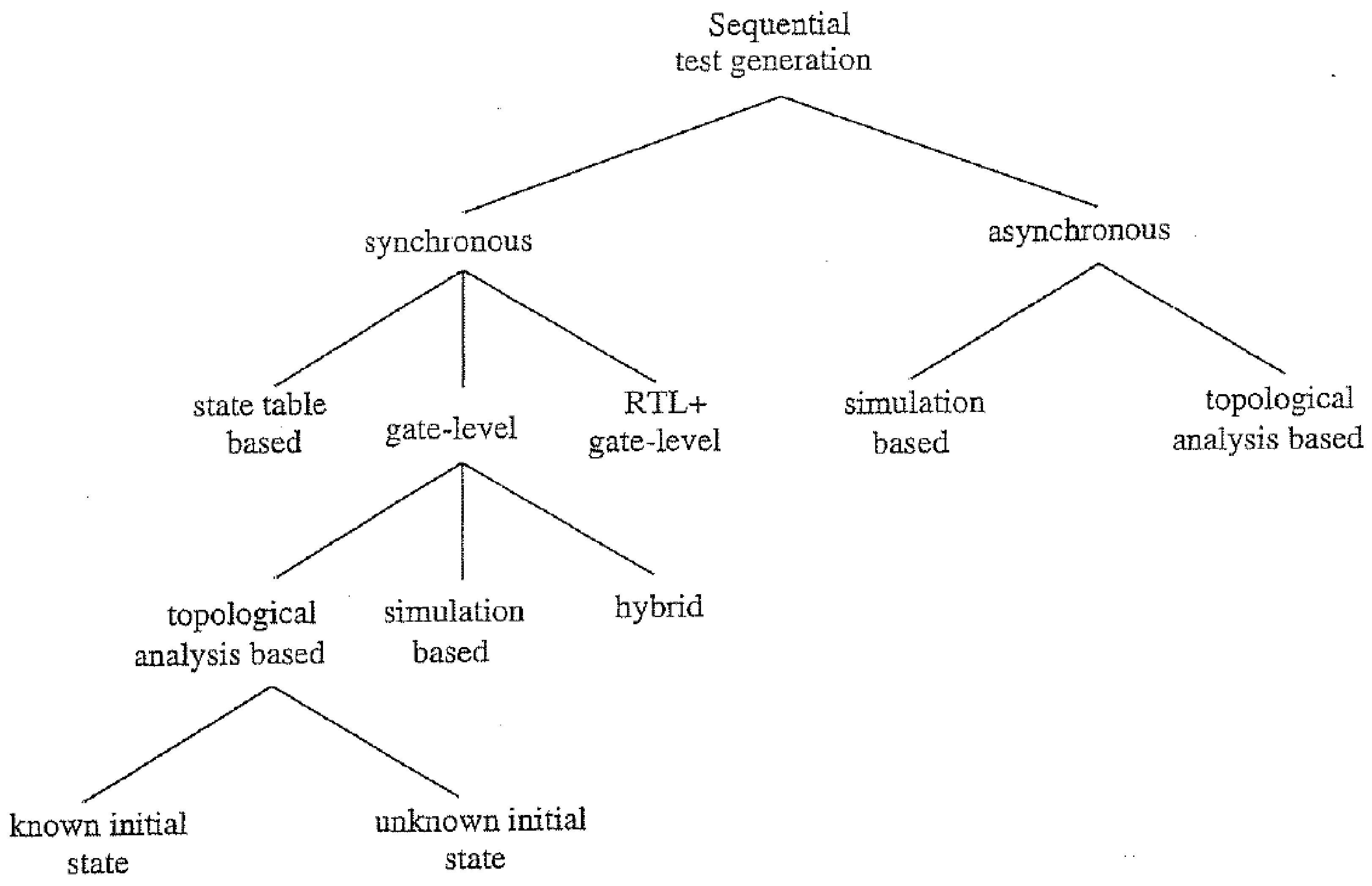
Test Generation
for Sequential Circuits

Main Concept & Strategy

Sequential Circuits

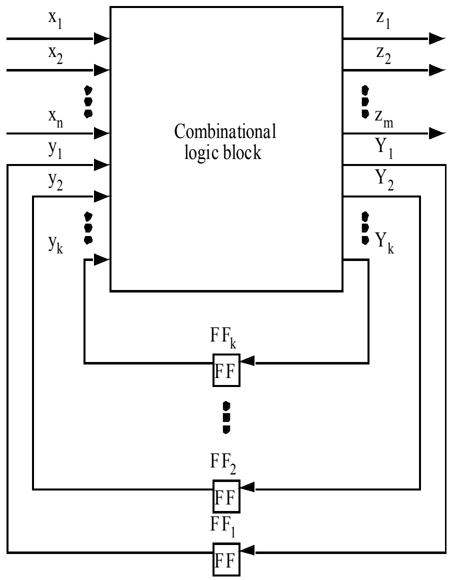
- A sequential circuit has memory (e.g. flip-flops) in addition to combinational logic.
  - As the combinational part is much larger than the memory/feedback part, most approaches (e.g. DFT techniques) deal only with the combinational part.
- Test for a fault in a sequential circuit is a sequence of vectors, which
  - Initializes the circuit to a known state
  - Activates the fault, and
  - Propagates the fault effect to a primary output
- Methods of sequential circuit ATPG
  - Time-frame expansion methods
  - Simulation-based methods

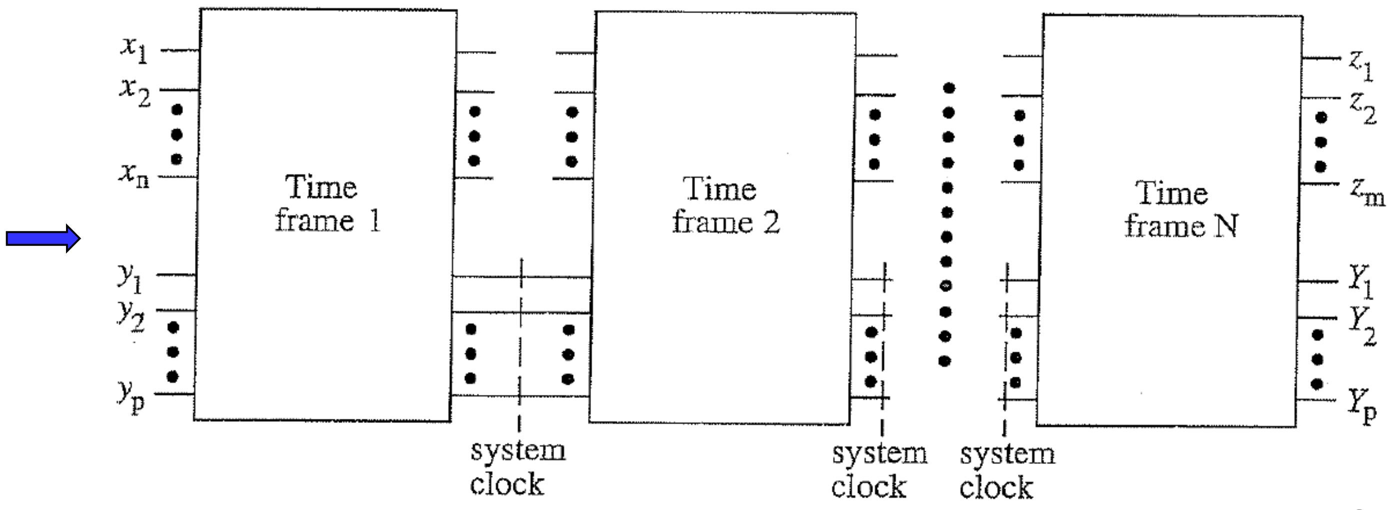
Taxonomy


Sequential test generation
synchronous
asynchronous
state table based
gate-level
RTL+ gate-level
simulation based
topological analysis based
topological analysis based
simulation based
hybrid
known initial state
unknown initial state

Concept of Time-Frames

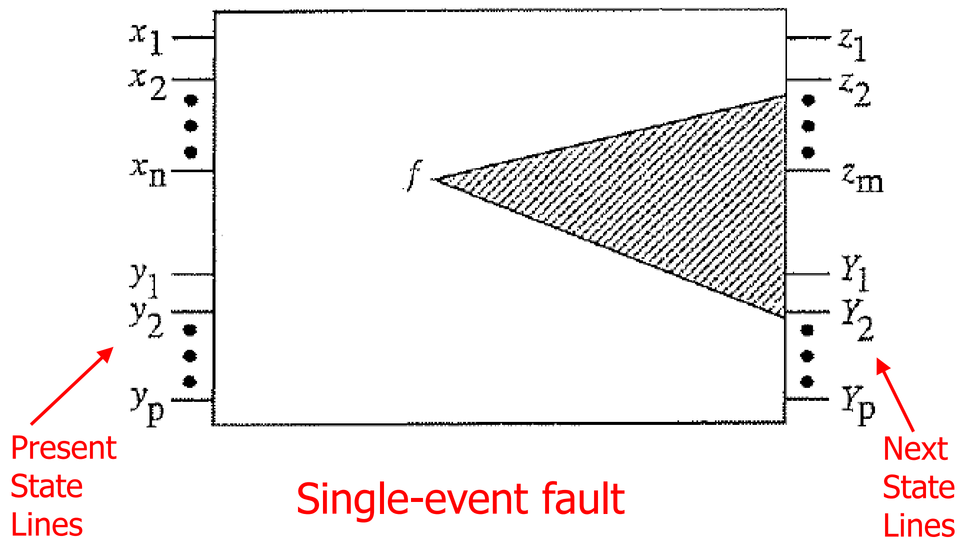
- Most sequential test generation techniques are based on a combinational iterative array model
  - Feedback signals are generated from the copies of the combinational logic in the previous **time frames**
  - A clock pulse is applied between each pair of successive time frames to update the logic values
  - A single s-a-f corresponds to the fault in all time frames



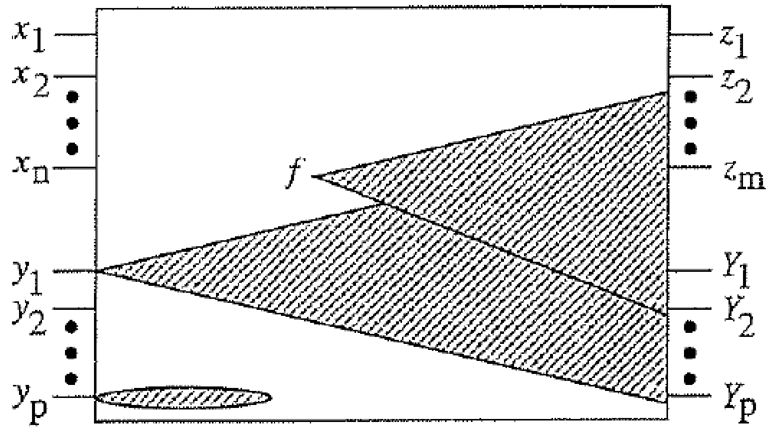


Concept of Time-Frames (cont.)

- If the effect of fault f does not propagate to the next-state line in a particular time frame, then the faulty and fault-free values of the corresponding present-state lines are the same in those time frames.
- A single event fault at one time frame may become a multiple-event fault in another time frame.



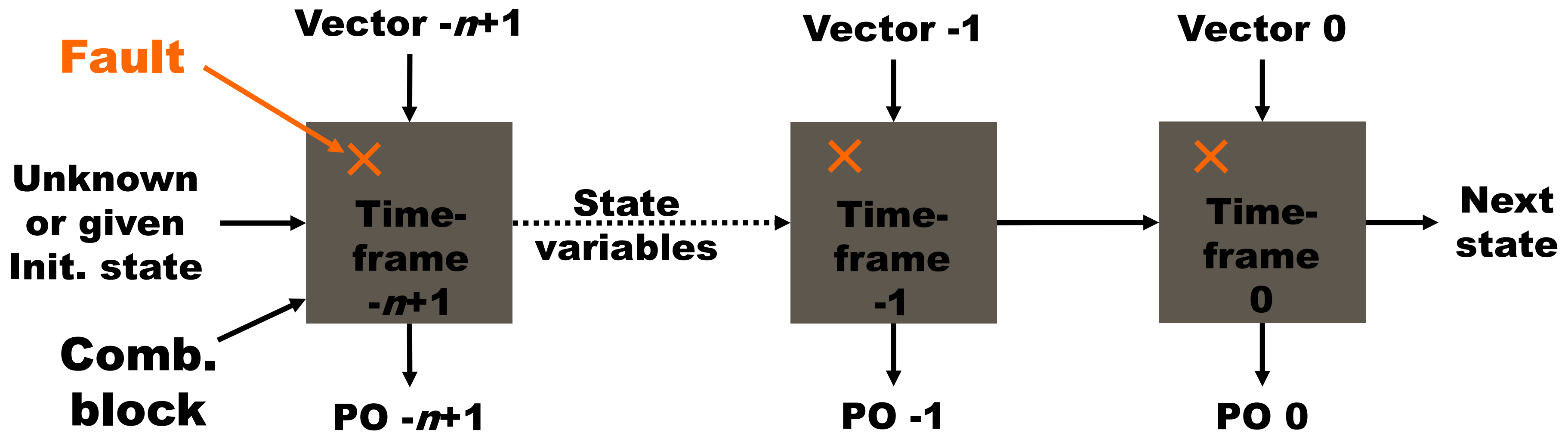
Single-event fault



Multiple-event fault

Concept of Time-Frames (cont.)

- If the test sequence for a single stuck-at fault contains *n* vectors,
  - Replicate combinational logic block *n* times
  - Place fault in each block
  - Generate a test for the multiple stuck-at fault using combinational ATPG with 9-valued logic



Review: 5- and 9-Valued Calculus

- Roth’s 5-Valued and Muth’s 9-Valued Algebra

Symbol	Meaning	Fault-Free (Good)	Faulty (Bad)
0	0/0	0	0
1	1/1	1	1
D	1/0	1	0
$\overline{D}$	0/1	0	1
X	X/X	X	X
G0	0/X	0	X
G1	1/X	1	X
F0	X/0	X	0
F1	X/1	X	1

5-Valued

9-Valued

Extended D-Algorithm

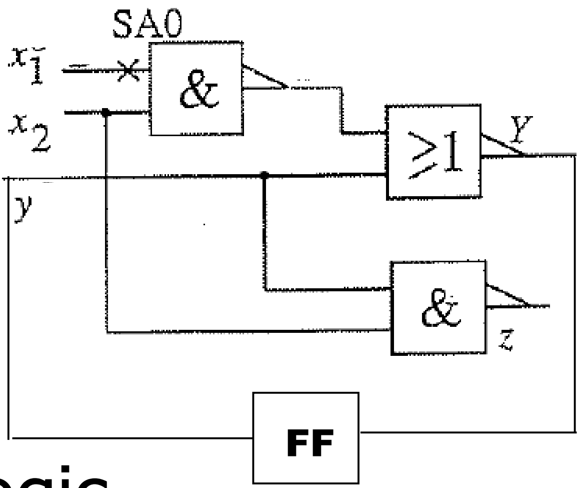
- Start with time frame 0 and use the D-algorithm to generate a test vector for this time frame
  - If the error propagates to at least one primary output in time frame 0, it's done (test vector is found)
  - If the error propagates only to the next-state lines, **a new time frame is added** as the next time frame to further propagate the error. This process continues until the error reaches at least one of the primary outputs.
  - If the test vector requires particular logic values at the present-state lines in time frame 0, **a new time frame is added at the previous time frame**. State justification is then performed backward through the previous time frame. The process continues until no specific logic values are required at the present-state lines.

Two Alternative Approaches

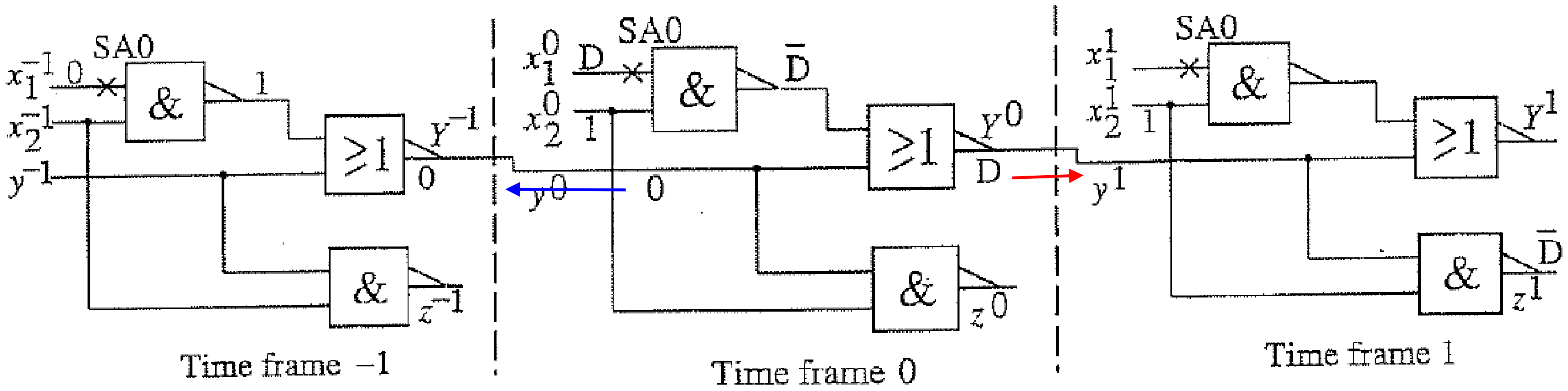
- In terms of mechanical process, we have two alternatives:
  1. Forward/Backward time frame expansion
     – Stimulate the fault and add time frames forward/backward
     – Propagating the fault effect and add time frames forward/backward
     – See Example I and II
  2. Backward time frame expansion
     – Start by assuming D (1/0) or D′ (0/1) in one of the outputs in time frame 0
     – Only work backward until you find the answer (sequence of test patterns) or you run out of options
     – See Example III

Example I

- The circuit:

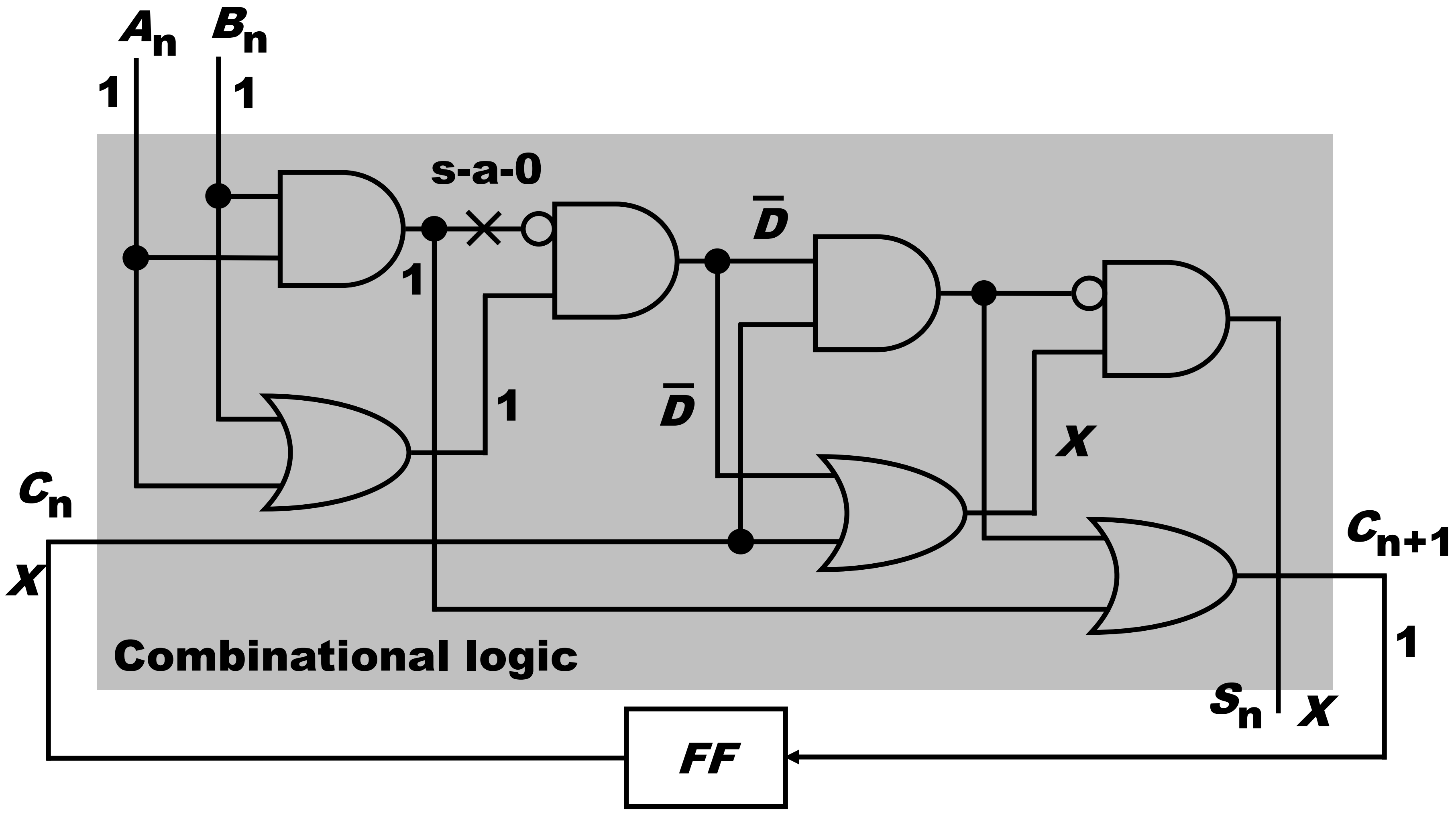


- Use 5-valued logic

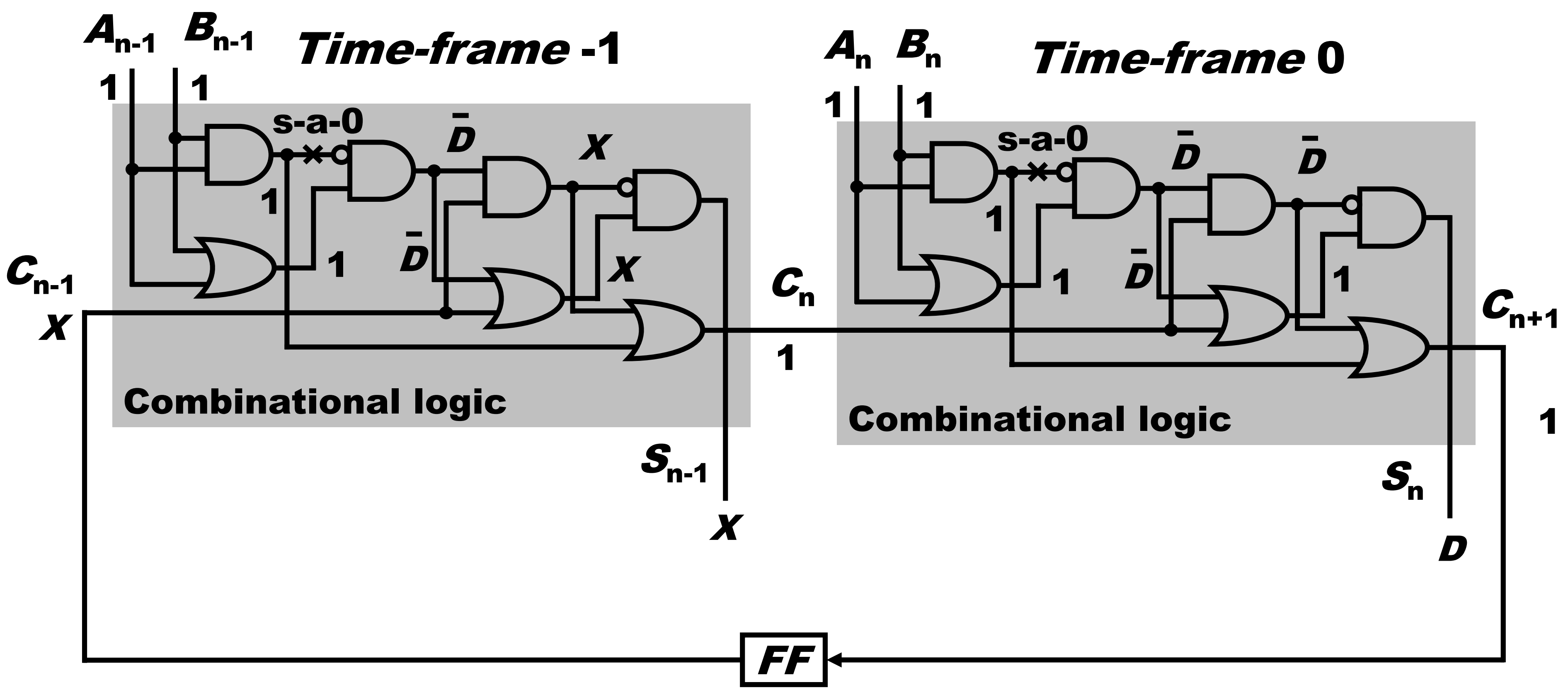


- Test Sequence: (x_1x_2)={(0x),(11),(x1)} or {(x0),(11),(x1)}

Example II: A Serial Adder



Example II

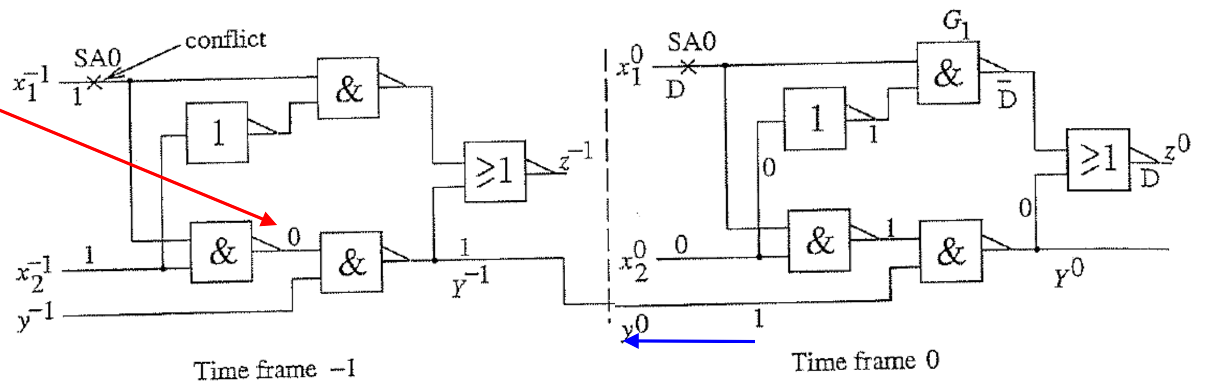


- Test Sequence: (AB)={(11),(11)}

Example - 5- vs. 9- Valued Logic

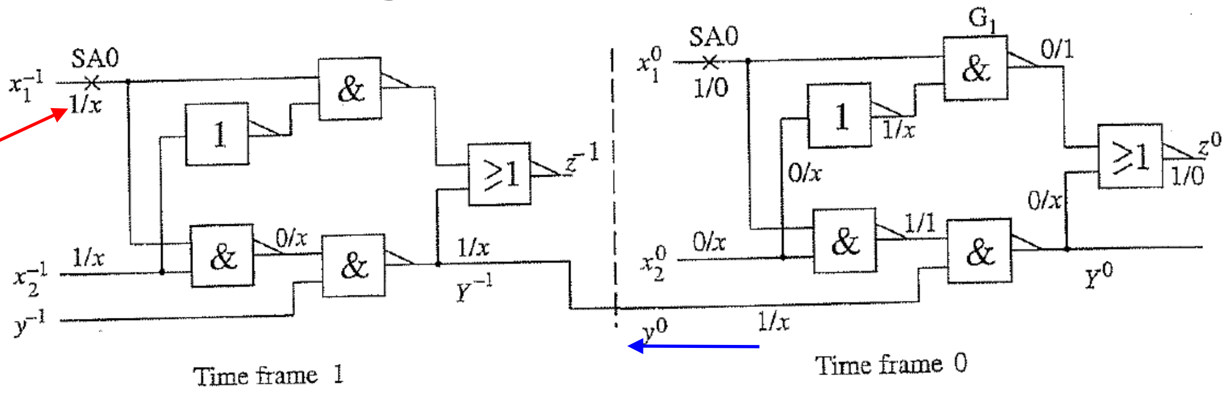
- Using 5-valued logic:

No pattern is found as “0” here is not guaranteed.



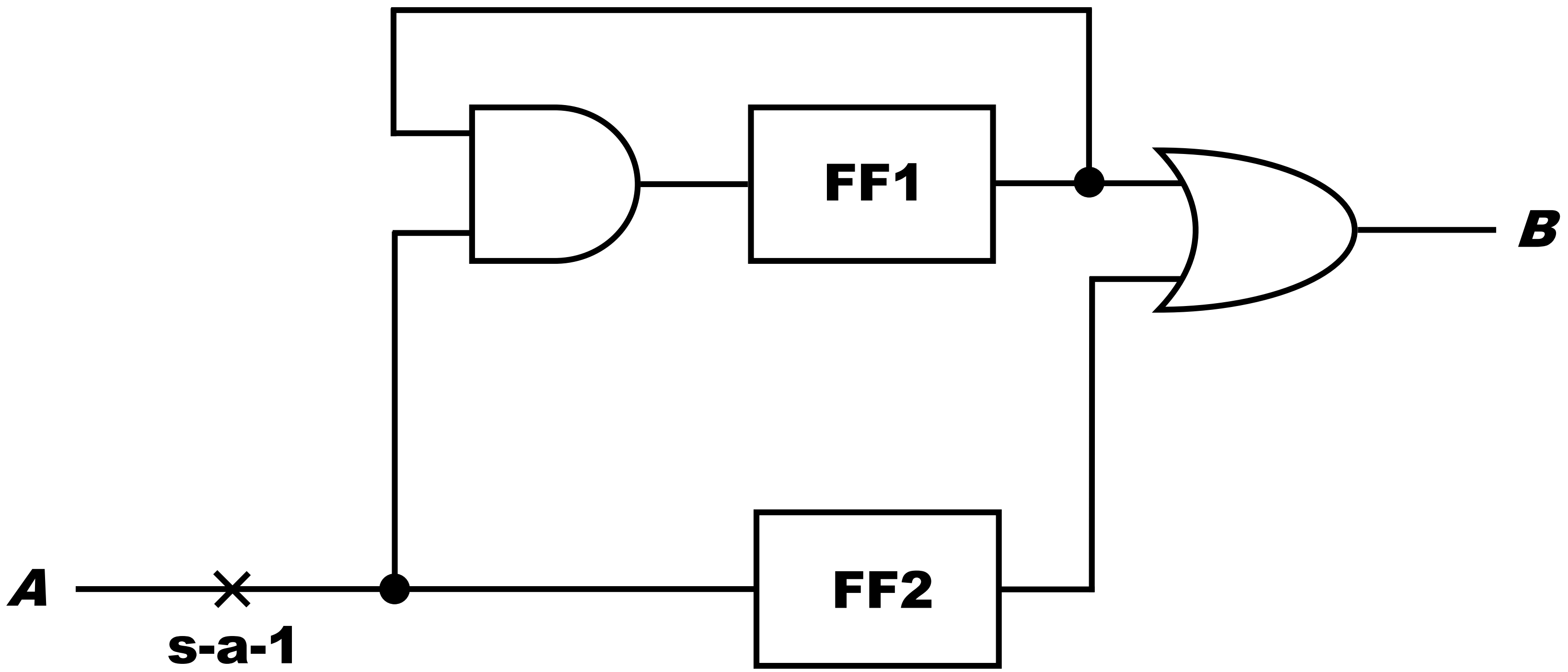
- Using 9-valued logic:

Relaxed Requirement (more information processed)

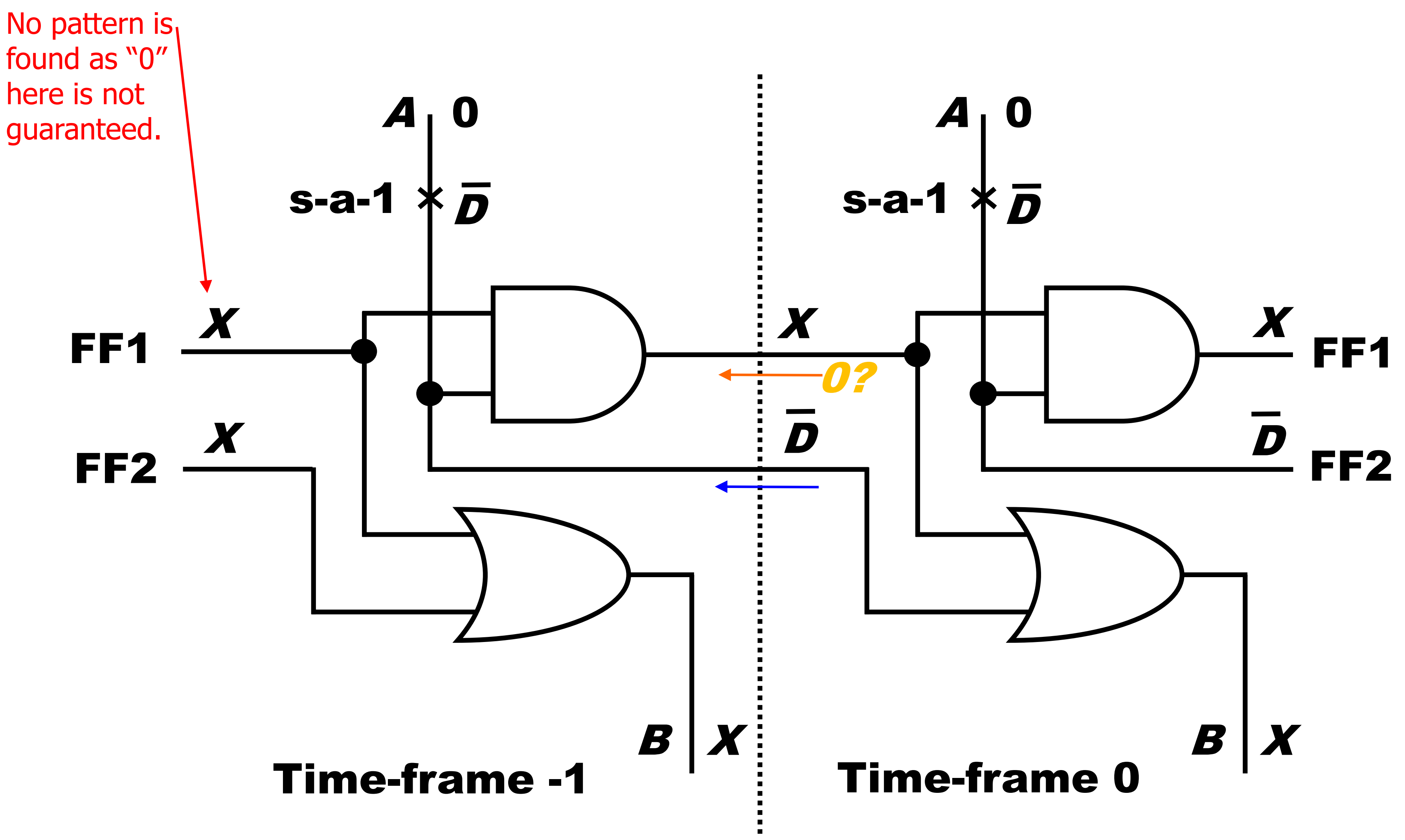


- Test Sequence: (x_1x_2)={(11),(10)}

Example for Logic Systems

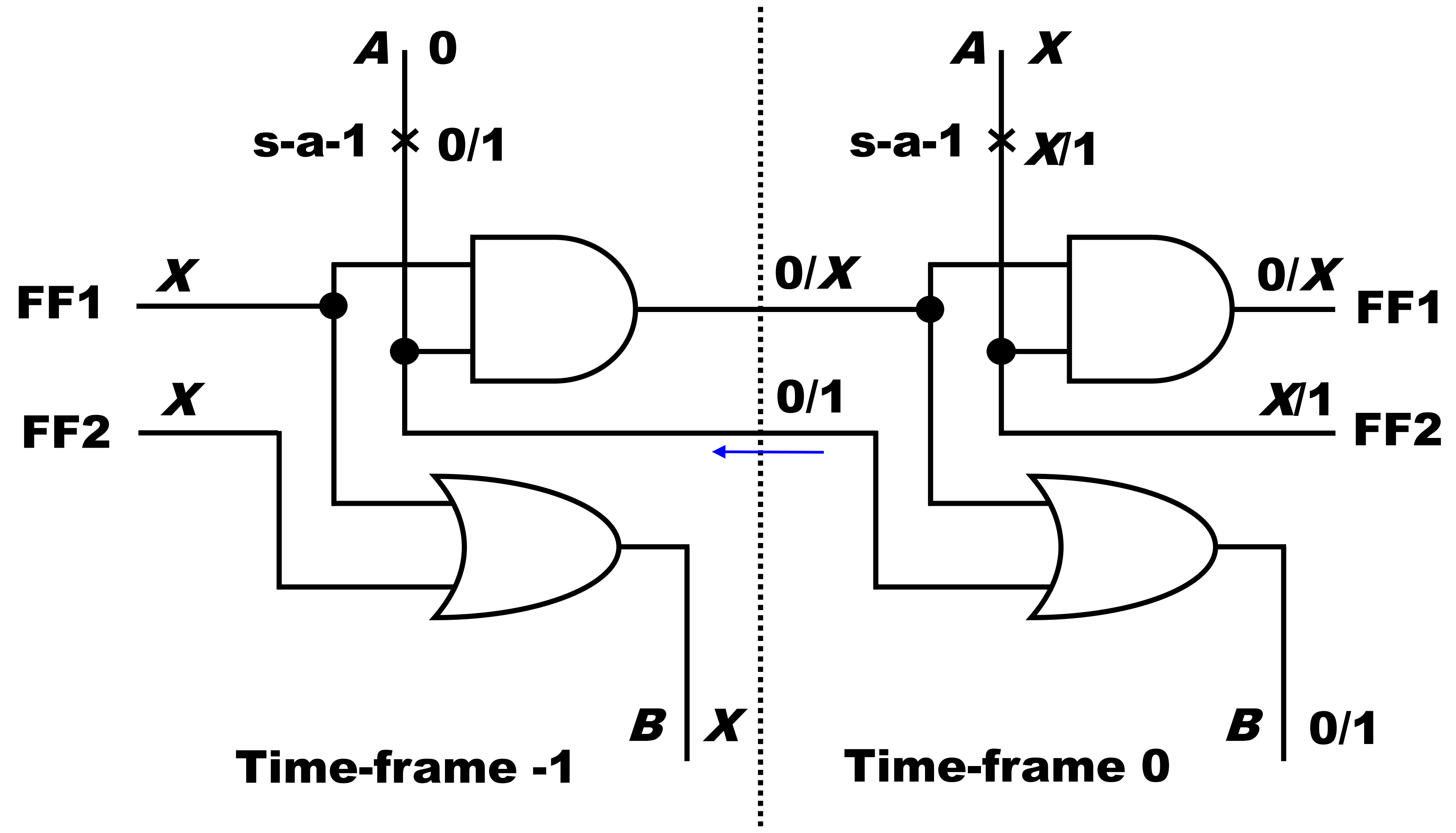


5-Valued Logic (Roth) 0, 1, D, D, $\overline{X}$



- Test Sequence: Not Found.

9-Valued Logic (Muth) 0,1,1/0,0/1,1/*X*,0/*X*,*X*/0,*X*/1,*X*



- Test Sequence: (A)={(0),(x)}

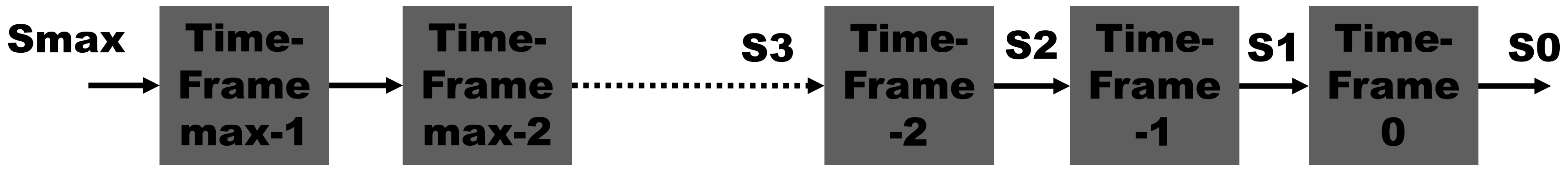
ATPG for Sequential Circuits

Implementation of ATPG

- Select a PO for fault detection based on drivability analysis (often based on CC0, CC1 and CO metrics).
- Place a logic value, 1/0 or 0/1, depending on fault type and number of inversions.
- Justify the output value from PIs, considering all necessary paths and adding backward time-frames.
- If justification is impossible, then use drivability to select another PO and repeat justification.
- If the procedure fails for all reachable POs, then the fault is *untestable.*
- If 1/0 or 0/1 cannot be justified at any PO, but 1/X or 0/X can be justified, the the fault is *potentially detectable.*

Complexity of ATPG

- Synchronous circuit -- All flip-flops controlled by clocks; PI and PO synchronized with clock:
  - Cycle-free circuit – No feedback among flip-flops: Test generation for a fault needs no more than *dseq* + 1 time-frames, where *dseq* is the sequential depth.
  - Cyclic circuit – Contains feedback among flip-flops: May need 9^{Nff} time-frames, where *Nff* is the number of flip-flops.
- Asynchronous circuit – Higher complexity!



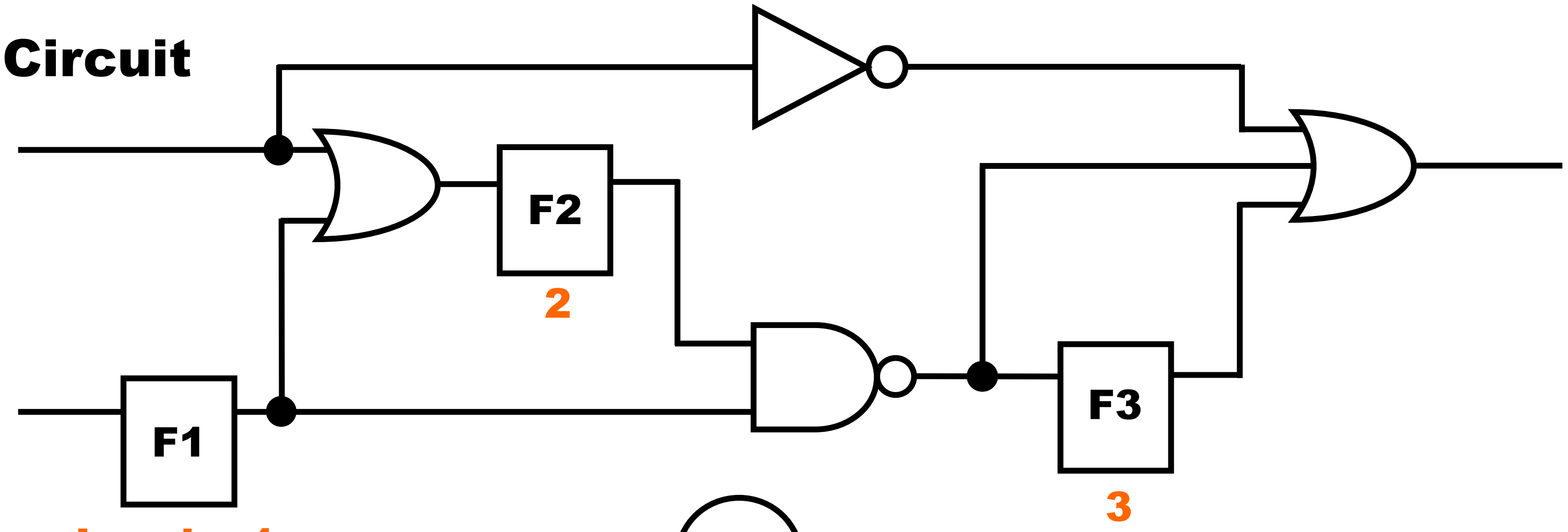
max **= Number of distinct vectors with 9-valued elements =** $\mathbf{9^{Nff}}$

Cycle-Free Circuits

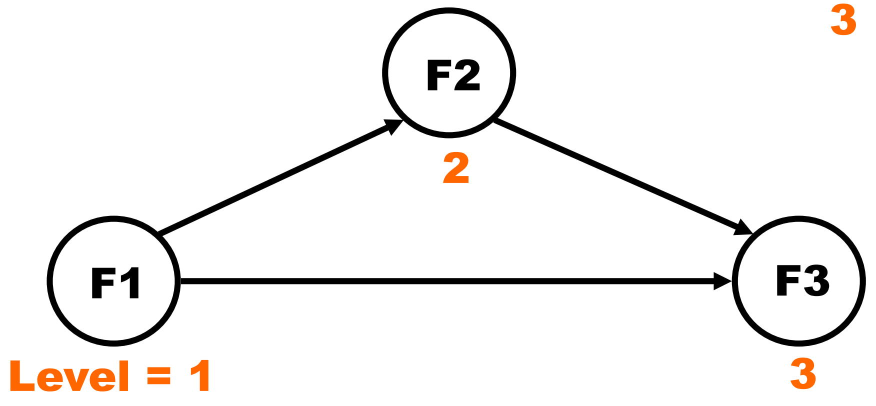
- Characterized by absence of cycles among flip-flops and a sequential depth, *dseq*.
- *dseq* is the maximum number of flip-flops on any path between PI and PO.
- Both good and faulty circuits are initializable.
- Test sequence length for a fault is bounded by ***dseq*+1**.

Cycle-Free Example

Circuit



s - graph

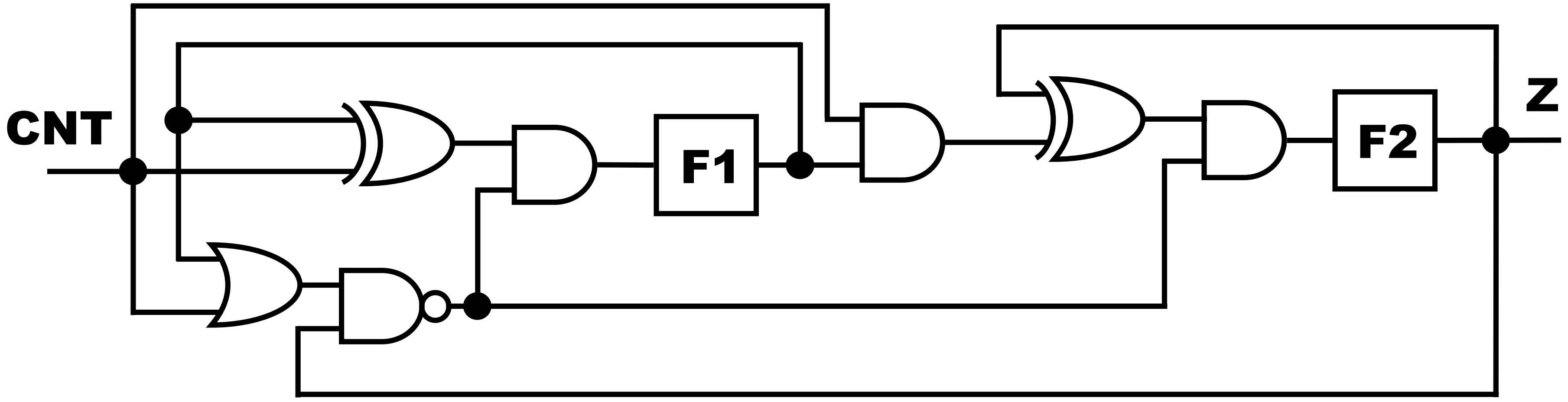


dseq = 3

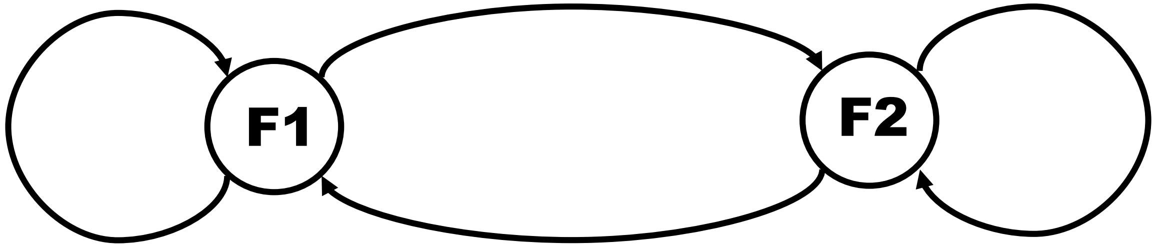
All faults are testable.

Cyclic Circuit Example

Modulo-3 counter



s - graph

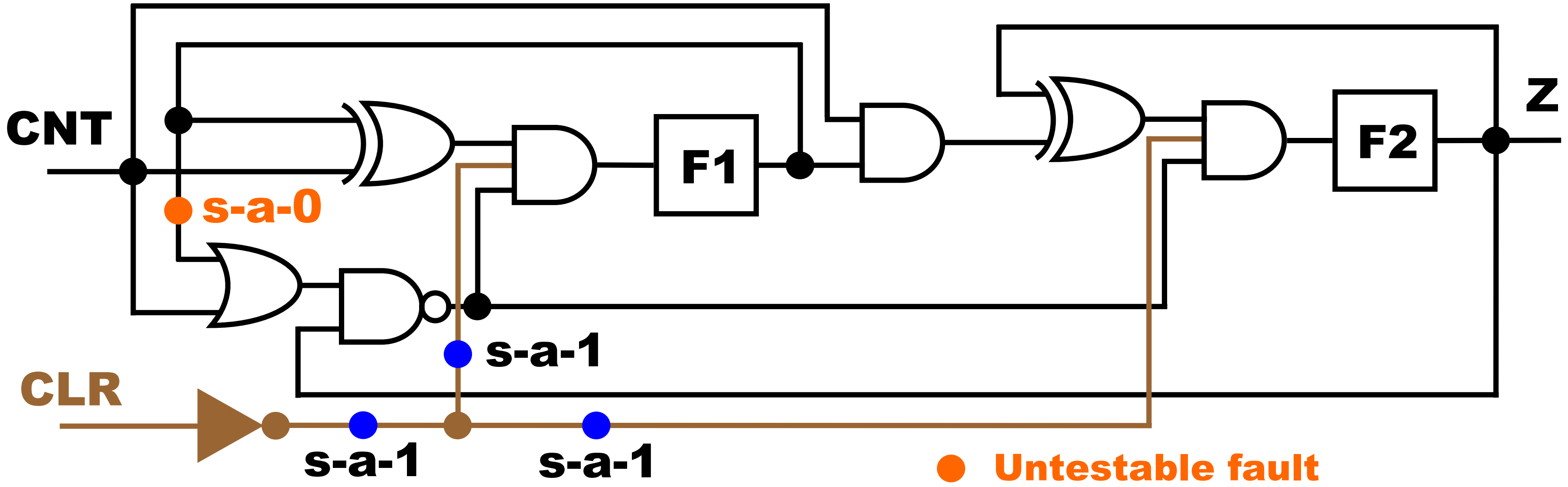


Modulo-3 Counter

- Cyclic structure – Sequential depth is undefined.
- Circuit is not initializable. No tests can be generated for any stuck-at fault.
- After expanding the circuit to $9^{Nff} = 81$, or fewer, time-frames ATPG program calls any given target fault untestable.
- Circuit can only be functionally tested by multiple observations.
- Functional tests, when simulated, give no fault coverage.

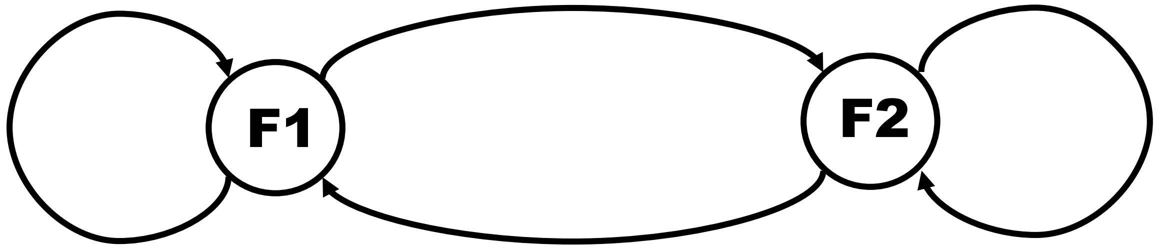
Adding Initializing Hardware

Initializable modulo-3 counter



● Untestable fault
● Potentially detectable fault

s - graph

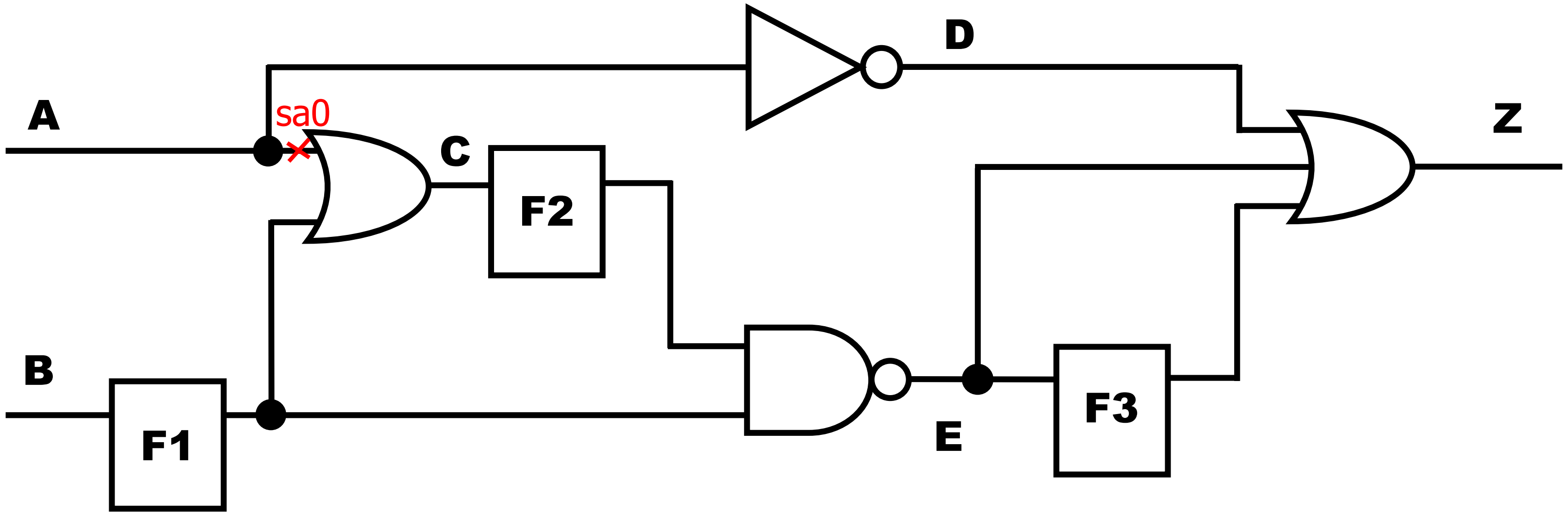


Benchmark Circuits

Circuit	s1196	s1238	s1488	s1494
PI	14	14	8	8
PO	14	14	19	19
FF	18	18	6	6
Gates	529	508	653	647
Structure	Cycle-free	Cycle-free	Cyclic	Cyclic
Seq. depth	4	4	--	--
Total faults	1242	1355	1486	1506
Detected faults	1239	1283	1384	1379
Potentially detected faults	0	0	2	2
Untestable faults	3	72	26	30
Abandoned faults	0	0	76	97
Fault coverage (%)	99.8	94.7	93.1	91.6
Fault efficiency (%)	100.0	100.0	94.8	93.4
Max. sequence length	3	3	24	28
Total test vectors	313	308	525	559
Gentest CPU s (Sparc 2)	10	15	19941	19183

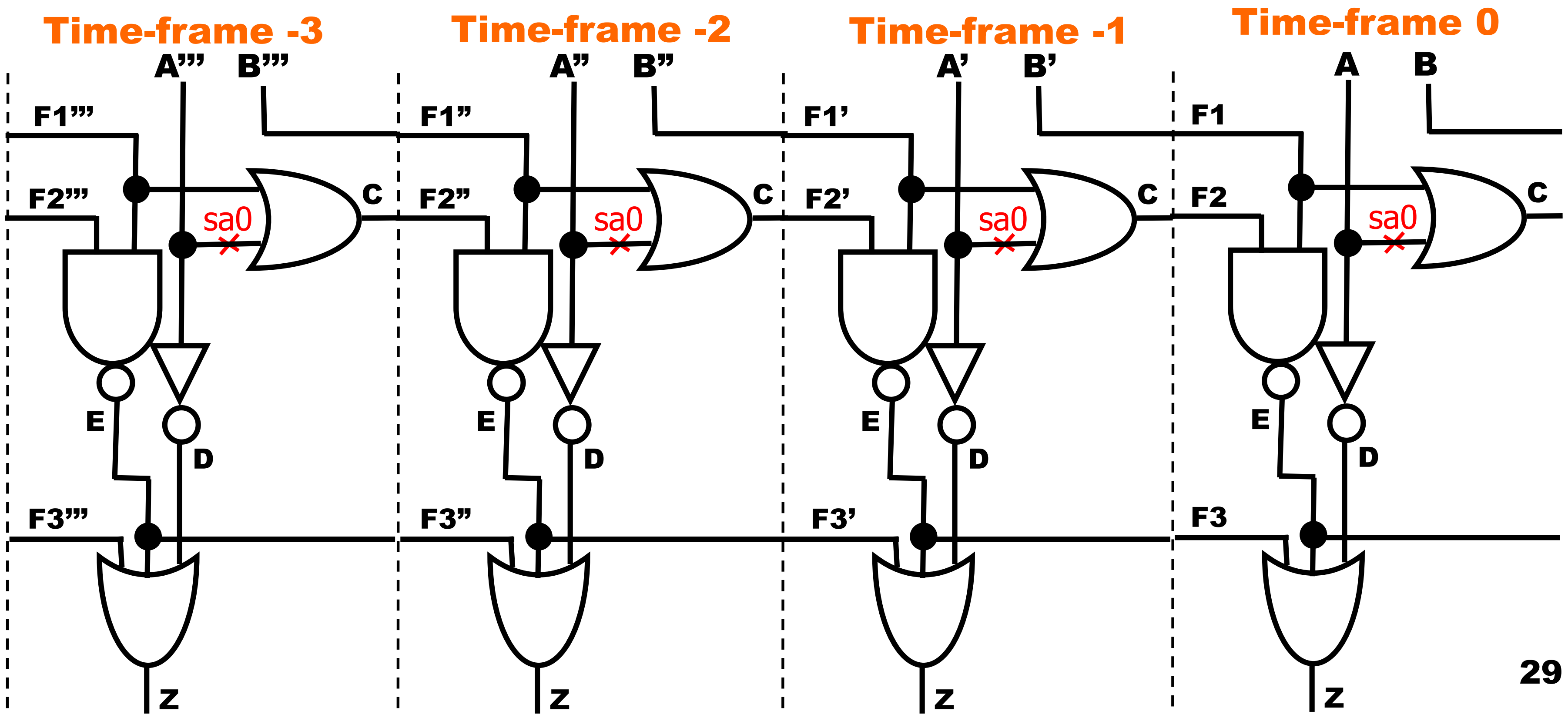
Example III: Cycle-Free Example

- We need at most 4 time-frames because dseq+1=4.
- A simple analysis tells us that the fault effect can be propagated to the output Z through two paths only:
  - C→F2→E→Z
  - C→F2→E→F3→Z
- The first path, being shorter (fewer FFs) is chosen as the primary option. Real test generators may use more complex analysis for grading the propagation paths.



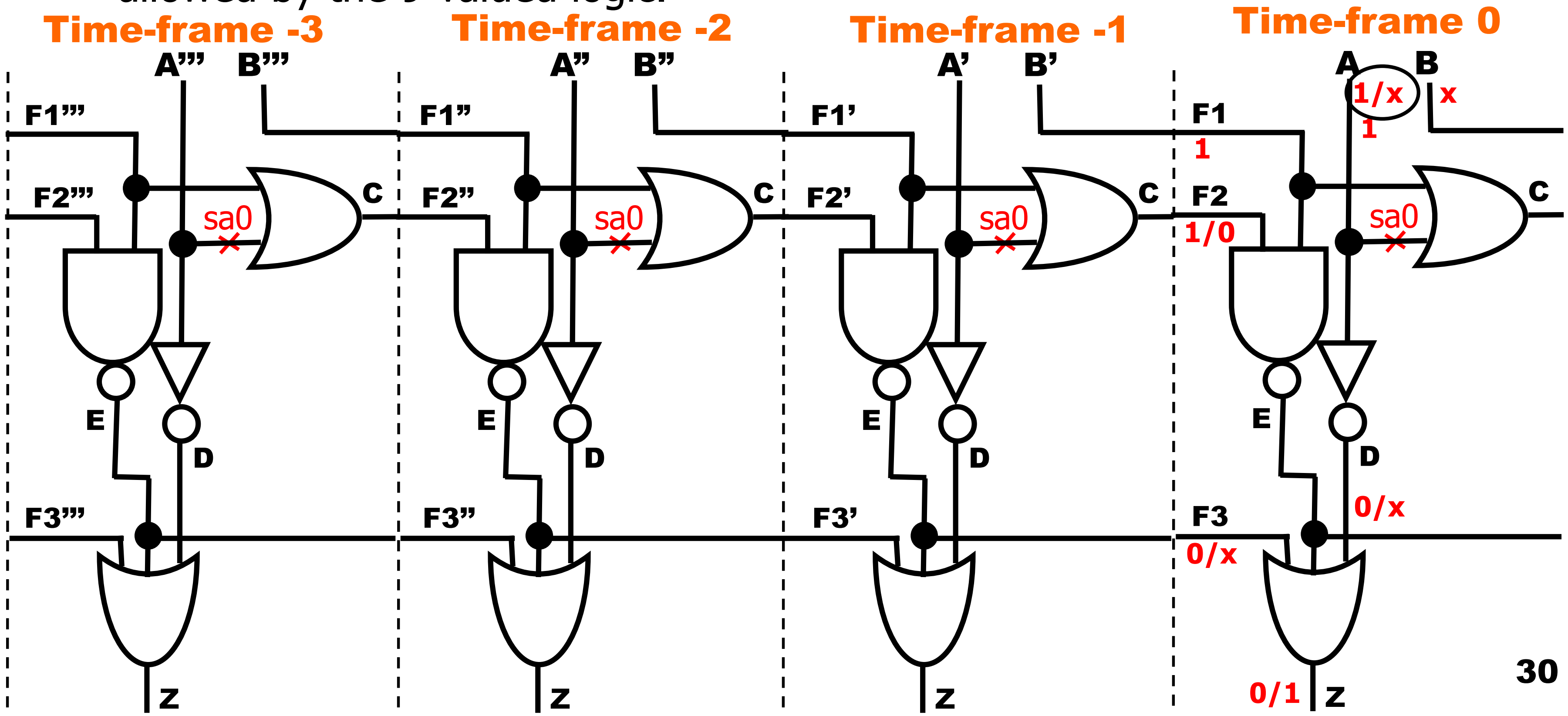
Cycle-Free Example – Step 0

- We need at most 4 time-frames because dseq+1=4.



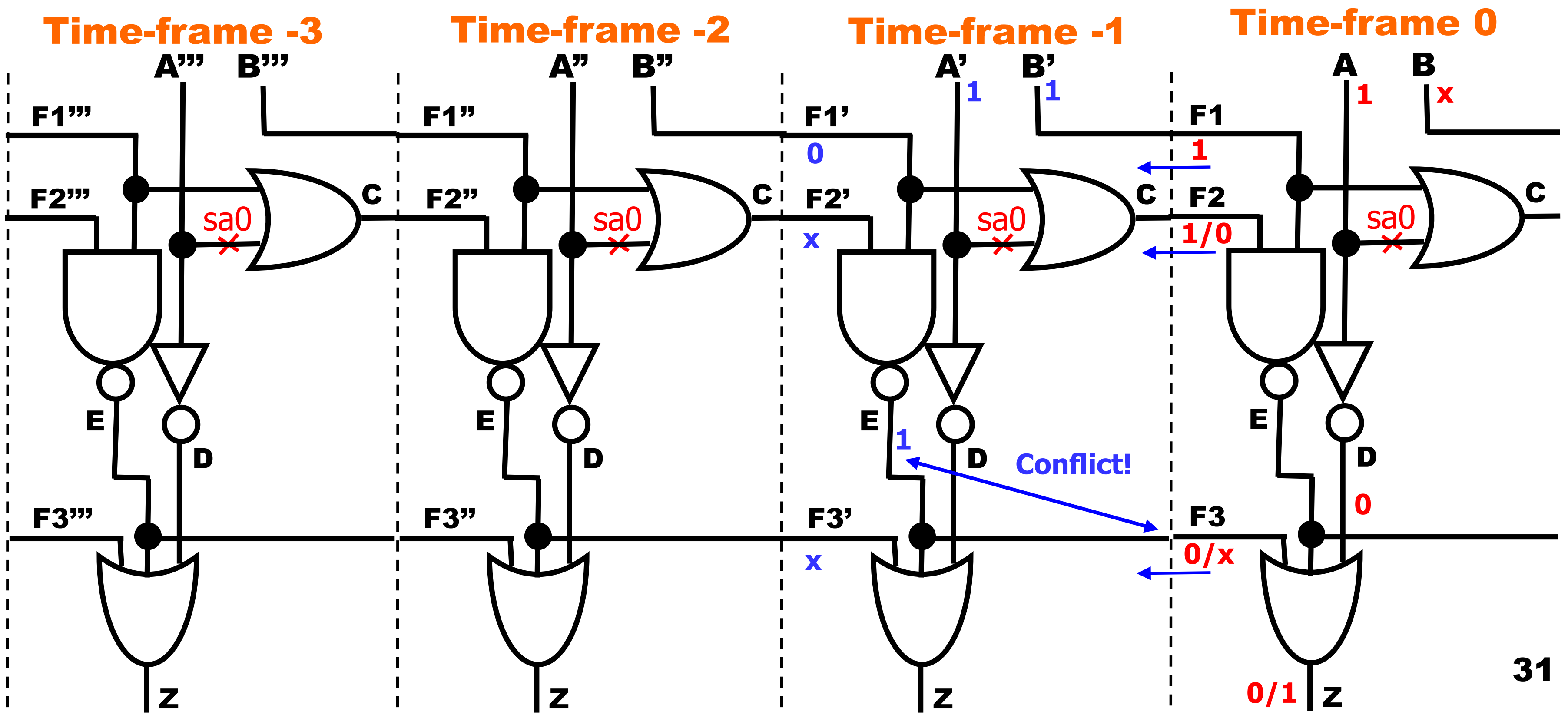
Example – Step 1

- In time-frame 0, we place a 0/1 at Z and justify it by A=1, F1=1, F2=1/0 and F3=0/x. Notice:
  - Fault effect propagation is attempted through C→F2→E→Z
  - F3 is set to 0/x (and not to 0) because all paths are simultaneously activated.
  - F3 is set to 0/x (and not to 0/1) because it is the least restrictive condition allowed by the 9-valued logic.



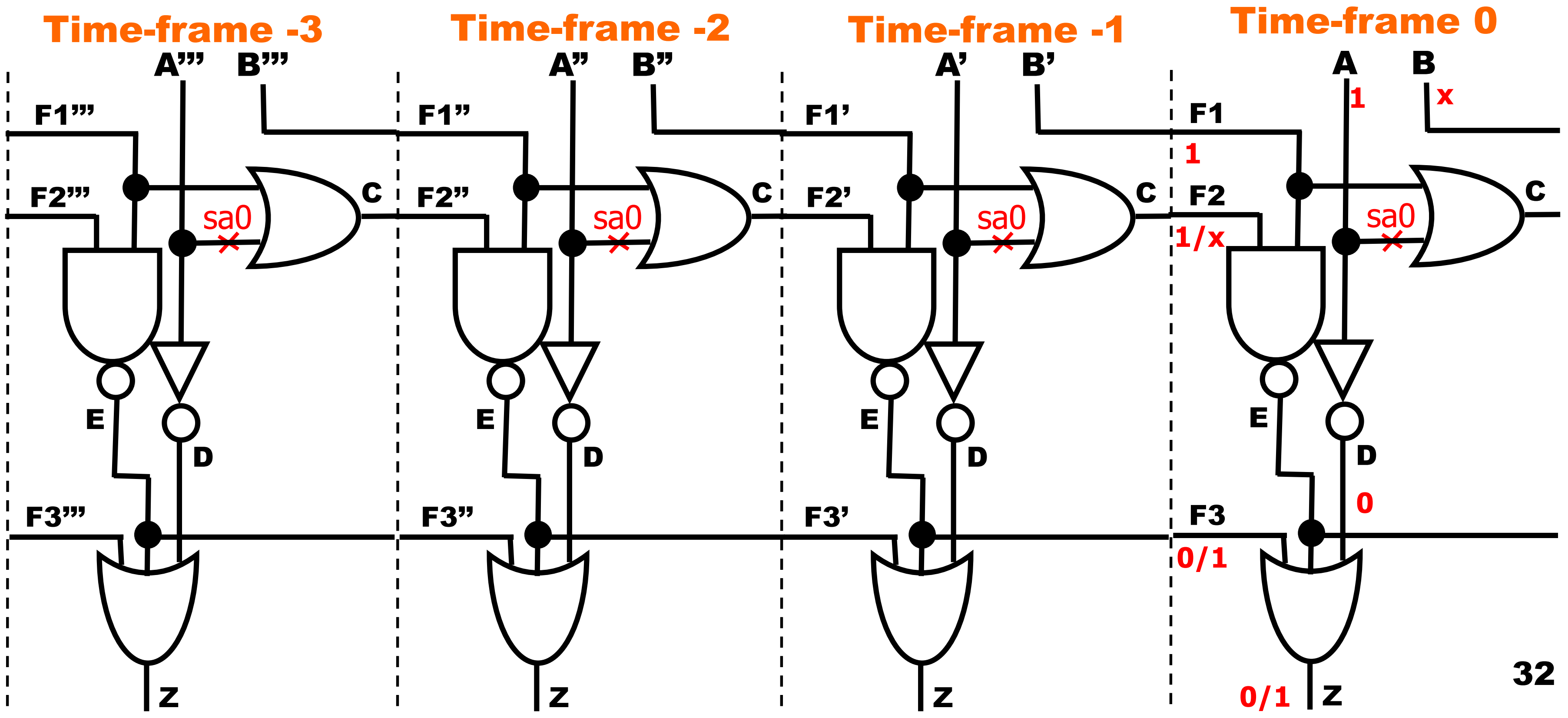
Example – Step 2

- We move to time-frame -1 to justify F1=1, F2=1/0 and F3=0/x. Setting A′=1, B′=1, F1′=0, F2′=x and F3′=x as inputs meets the first two requirement but causes a conflict on F3.
  - So, backtrack to time-frame 0.



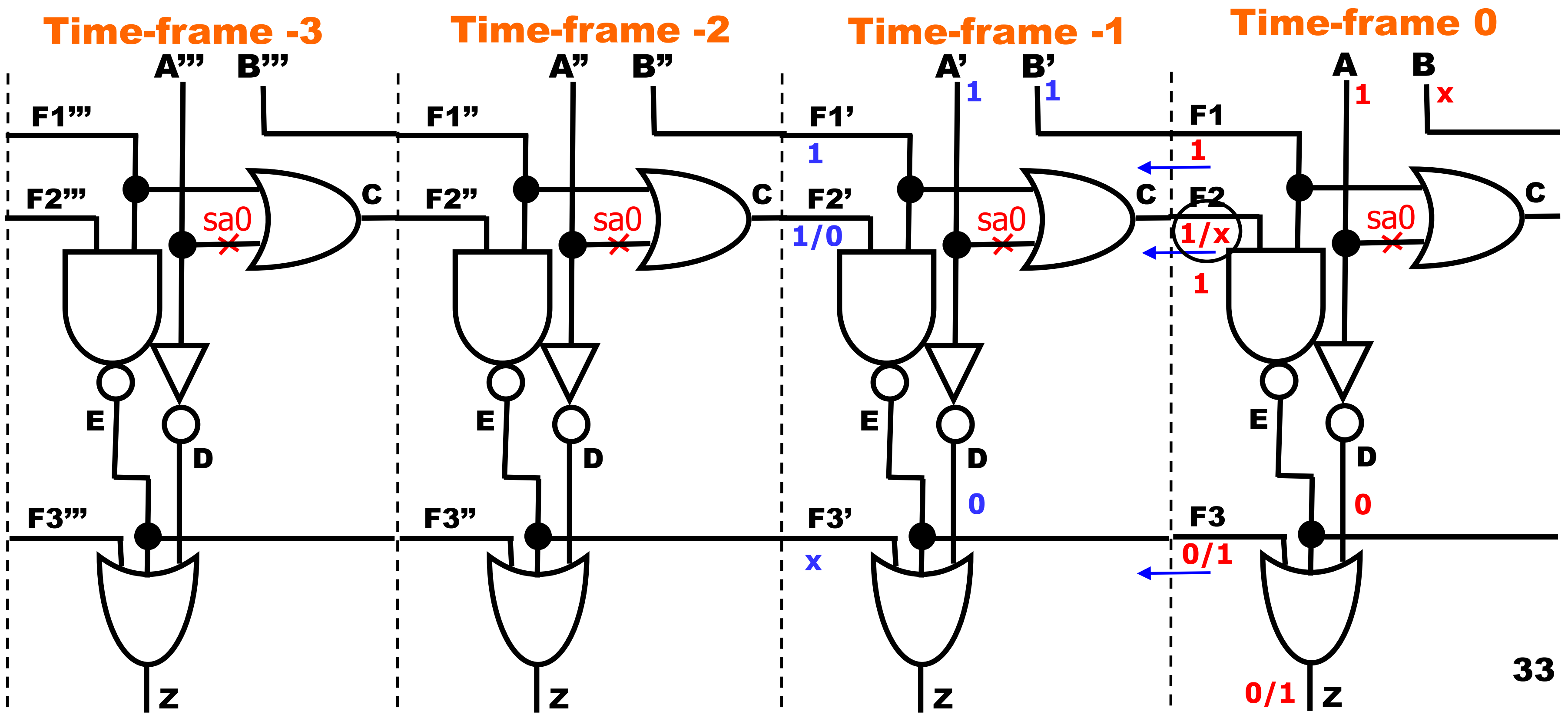
Example – Step 3

- The two path objectives are reversed. Z=0/1 is obtained by F2=1/x and F3=0/1.
- Fault effect propagation is attempted through C→F2→E→F3→Z


Time-frame -3
Time-frame -2
Time-frame -1
Time-frame 0
A’’’ B’’’ F1’’’ F2’’’ C sa0 E D F3’’’ Z
A’’ B’’ F1’’ F2’’ C sa0 E D F3’’ Z
A’ B’ F1’ F2’ C sa0 E D F3’ Z
A B F1 F2 C sa0 E D F3 Z
1
x
1
1/x
0
0/1
0/1

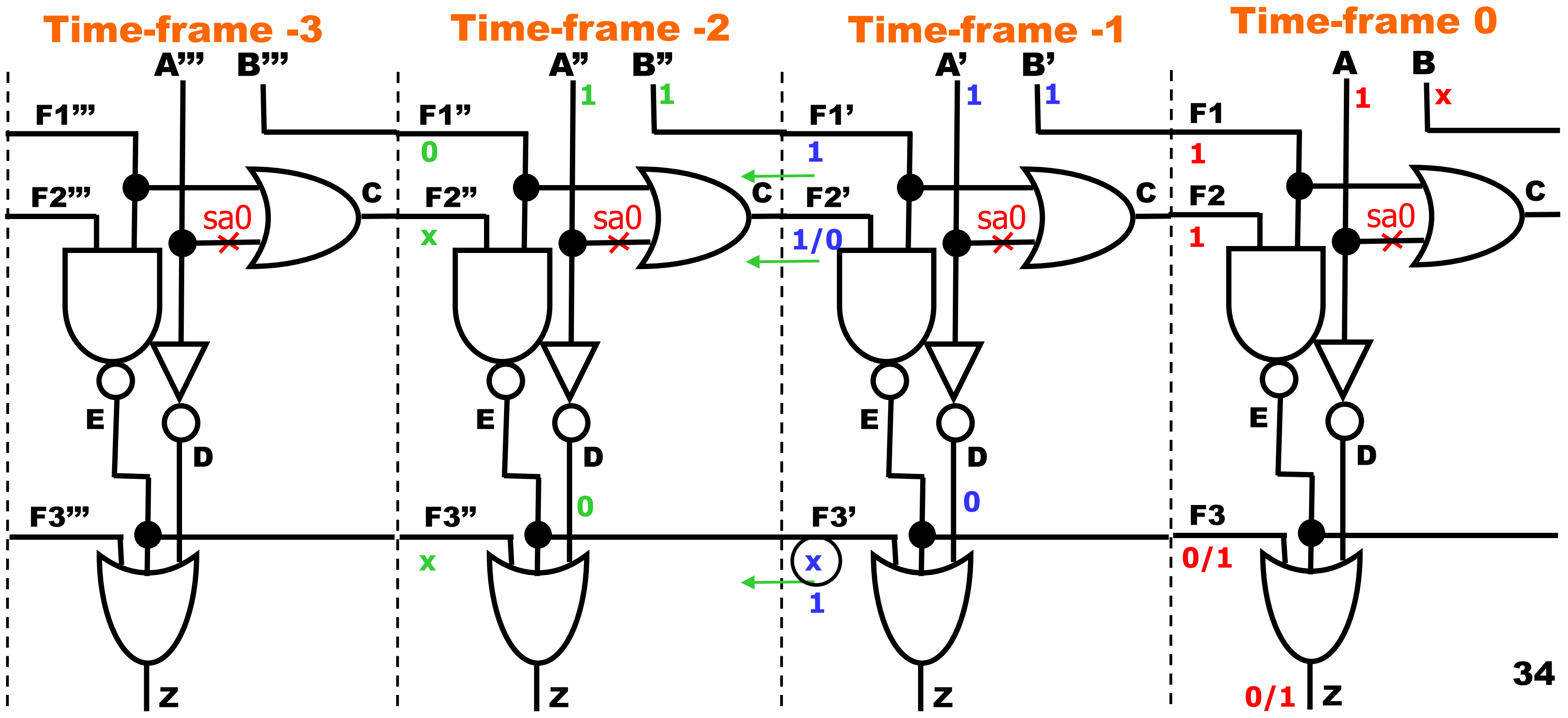
Example – Step 4

- In time-frame -1, we justify F1=1, F2=1/x and F3=0/1, by A′=1, B′=1, F1′=1, F2′=1/0 and F3′=x. This changes F2 to 1 but that is not a conflict with 1/x.



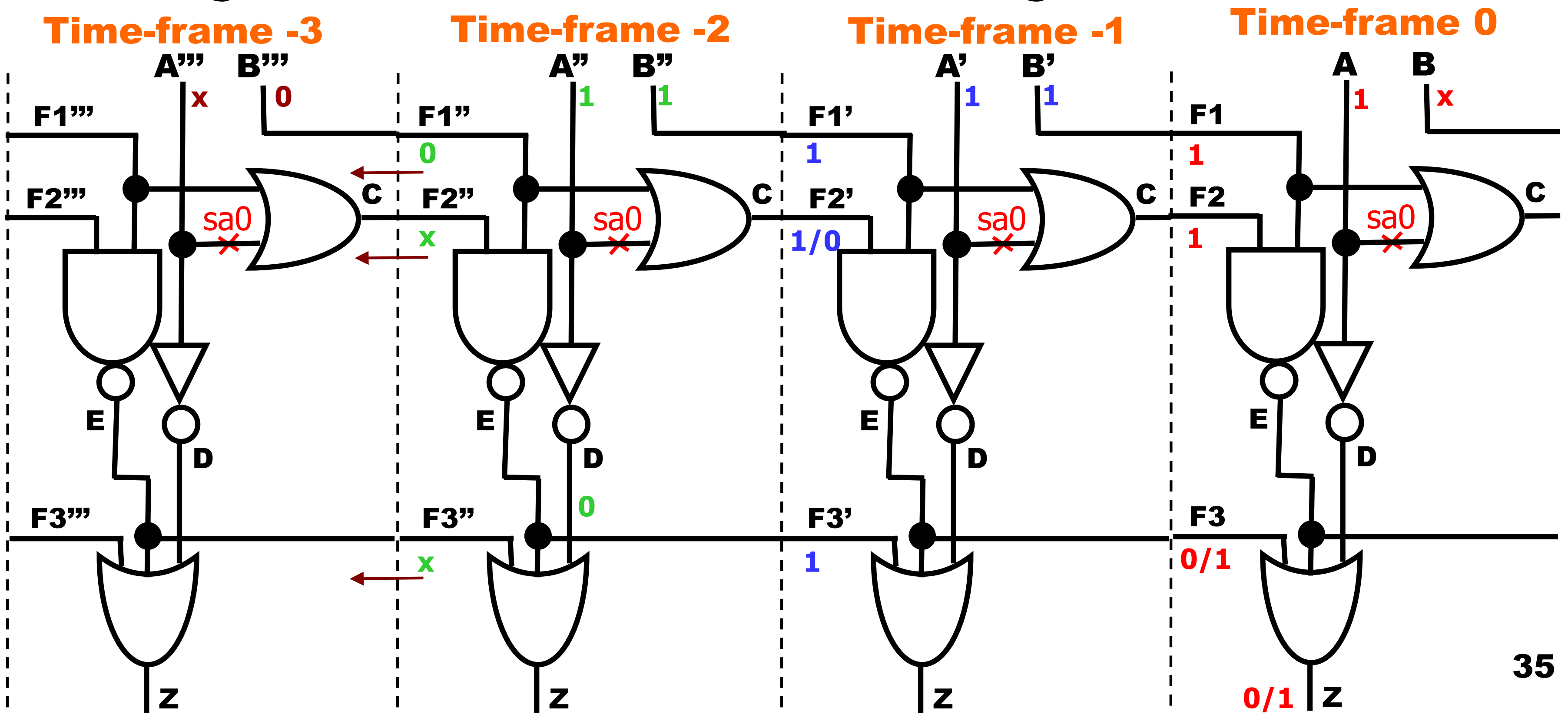
Example – Step 5

- In time-frame -2, we justify F1′=1, F2′=1/0 and F3′=1, by A″=1, B″=1, F1″=0, F2″=x and F3″=x.



Example – Step 6

- In time-frame -3, we justify F1″=0, F2″=F3″=x, by A‴=x, B‴=0.
  - Thus, a 4-vector test is found: (AB)={(x0),(11),(11),(1x)} which produces a 0/1 output on the last vector.
- A full test generation produces 17 vectors for detecting all 21 single s-a-f's in the combinational logic of this circuit.



Time-frame -3
Time-frame -2
Time-frame -1
Time-frame 0
A‴
B‴
x
0
F1‴
F2‴
F3‴
C
E
D
Z
sa0
A”
B”
1
1
F1”
0
F2”
x
F3”
x
0
F1’
1
F2’
1/0
F3’
1
A’
B’
1
1
A
B
1
x
F1
1
F2
1
F3
0/1
0/1