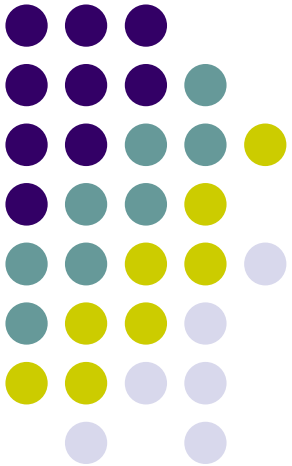
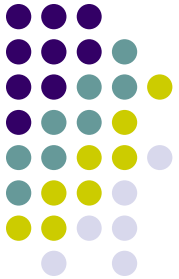
# Chapter 7 Sequential ATPG

循序電路自動輸入樣本產生器

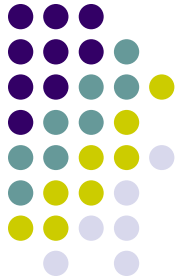


# Outline



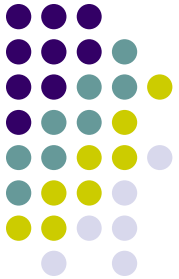
- Introduction
- Techniques for unknown initial states
  - Time-frame-based
  - Simulation-based
- Techniques for known initial states

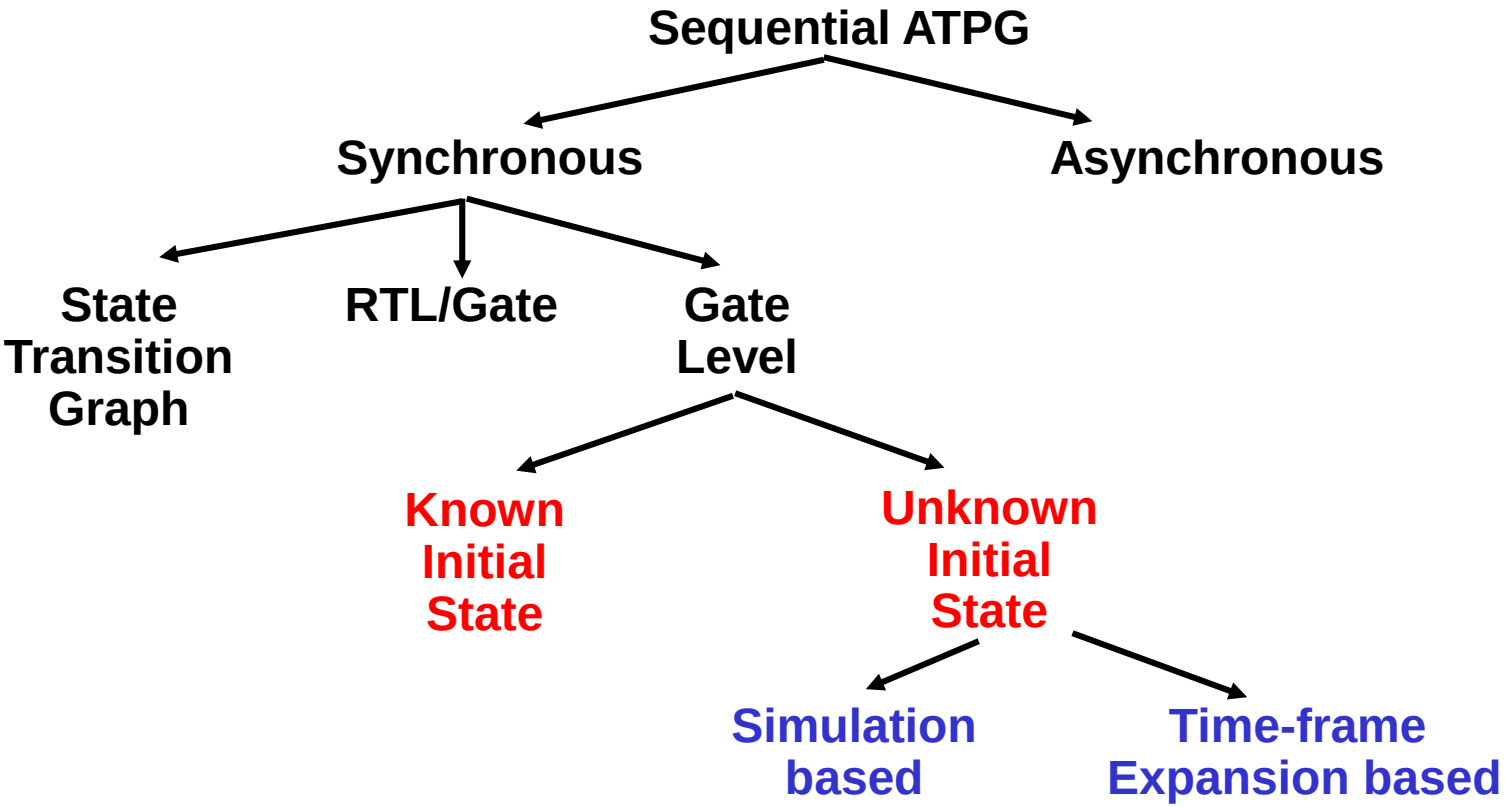
# Sequential Circuits



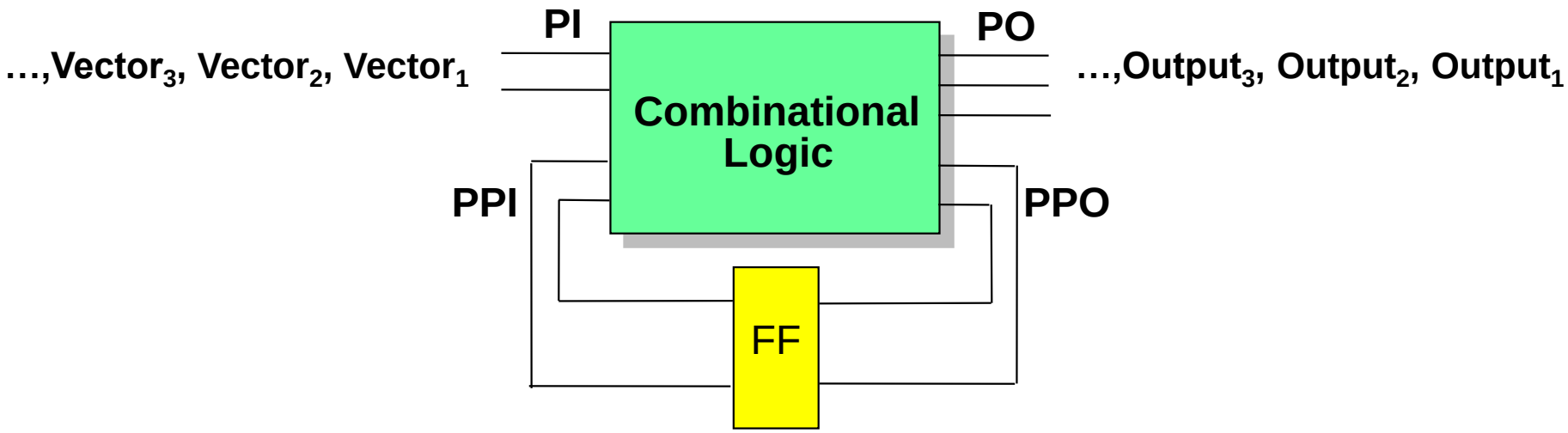
- Combination circuits + Memory elements
- Sequential circuits can be used to spread computations over time to reduce hardware
- Difficulties for test generation
  - Internal memory states unknown
  - After test applied, final states must be inferred indirectly
  - Long test sequence required
    - Initialization + fault activation and propagation + observation

# Sequential ATPG: Taxonomy





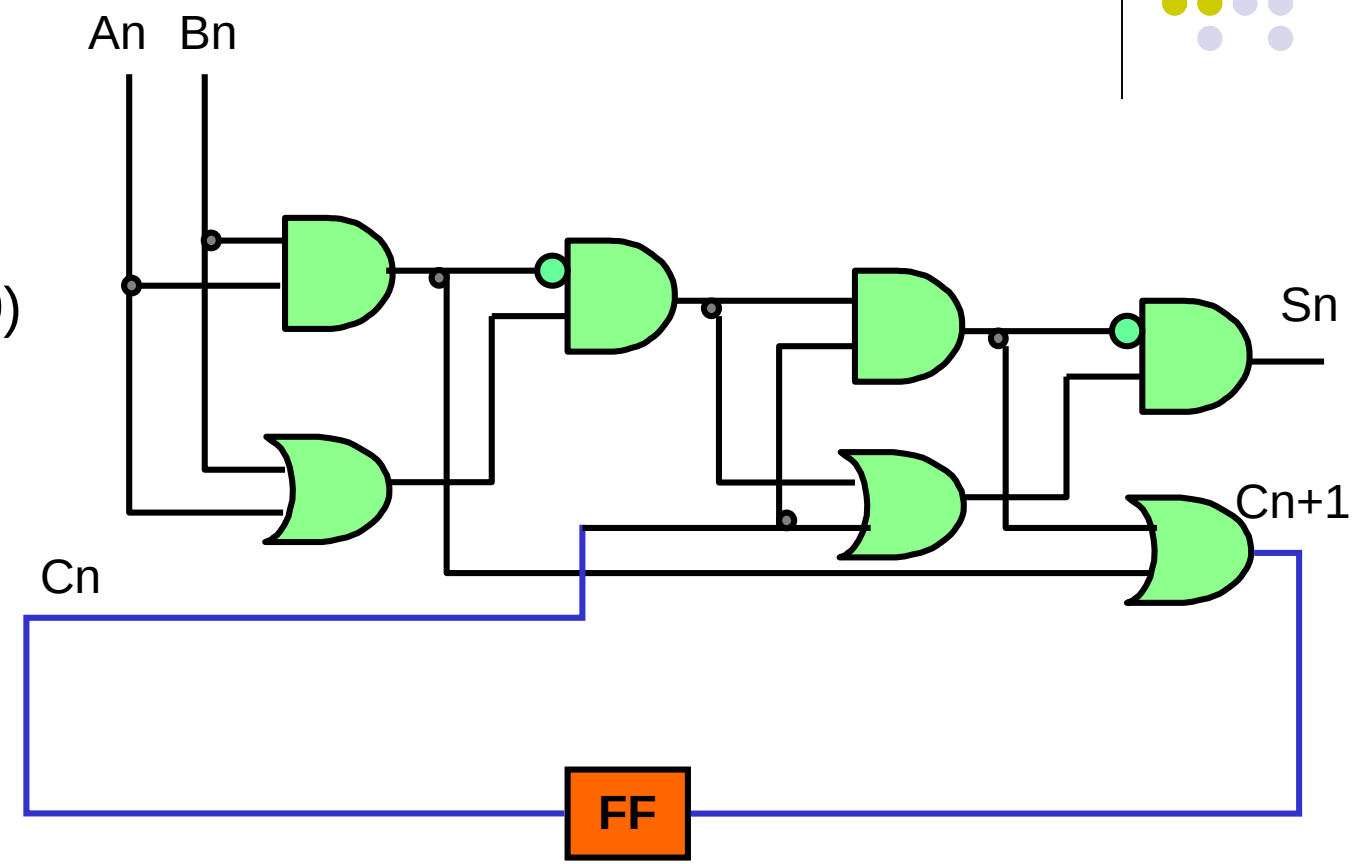
# Test Generation for Synchronous Sequential Circuits



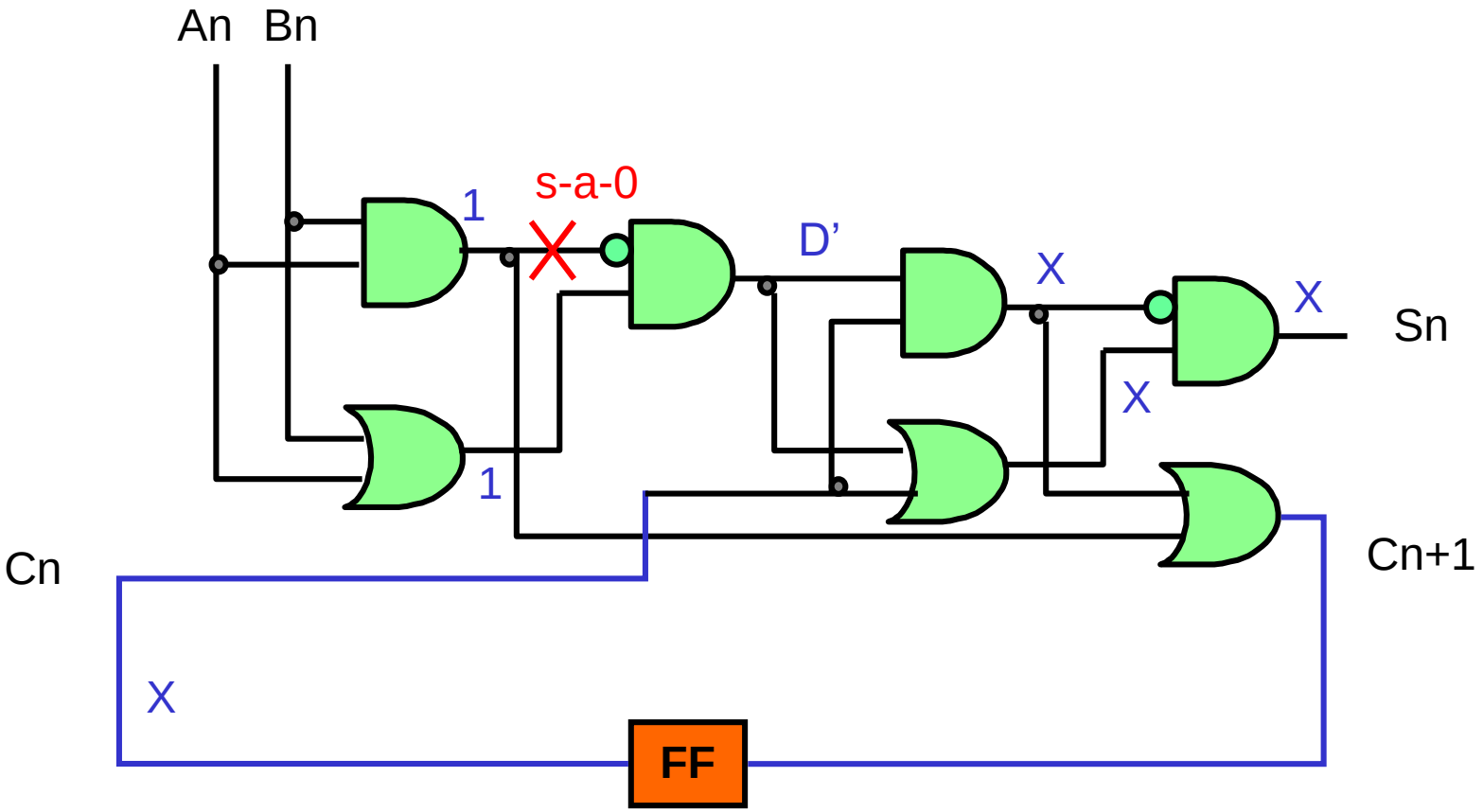

- Assumptions
  - All signal propagation times through combinational logics do not exceed the clock period
  - Flip-flops are ideal memories with an implicit clock
    - no fault at clocks and inside of FFs

# An Example: Serial Adder

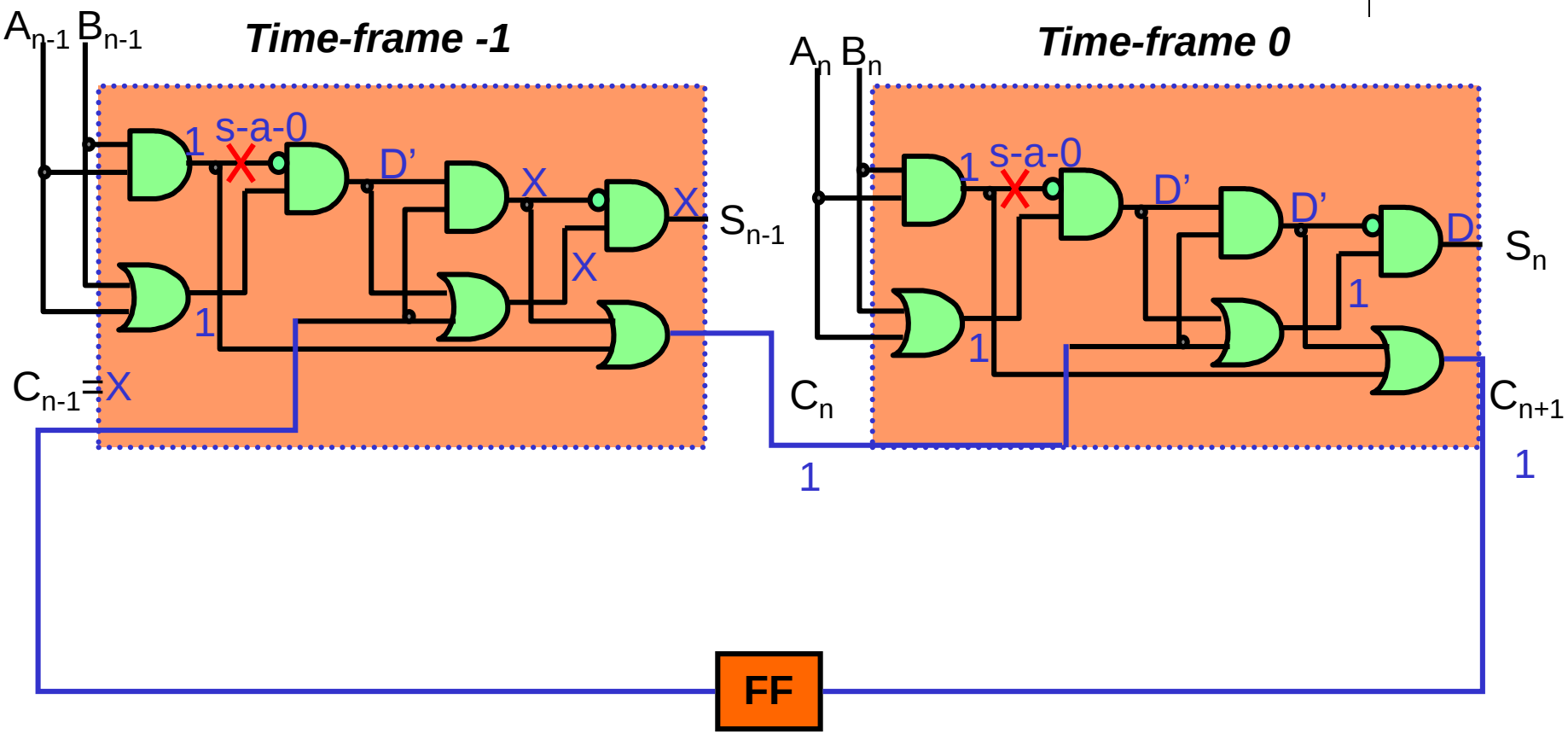
- Use one-bit adder to implement multiple-bit adder
- Initialize (A,B)=(0,0)
- (A0,B0)→S0
- (A1,B1)→S1
- ….
- (A31,B31)→S31
- (0,0)
  - S32 (carry bit)



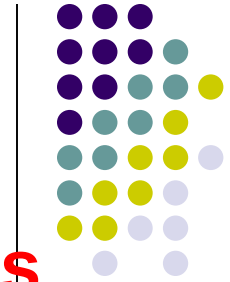
# An Example: Serial Adder



# An Example: Serial Adder



# Outline



- **For Circuits with Unknown Initial States**
  - **Time-frame Expansion Based:**
    - **Extended D-algorithm (IEEE TC, 1971)**
    - **9-V Algorithm (IEEE TC, 1976)**
    - **EBT (DAC, 1978 & 1986)**
    - **BACK (ICCD, 1988), ...**
  - **Simulation-Based:**
    - **CONTEST (IEEE TCAD, 1989)**
- **For Circuits with Known Initial States**
  - **STALLION (IEEE TCAD, 1988)**
  - **STEED (IEEE TCAD, May 1991), ...**

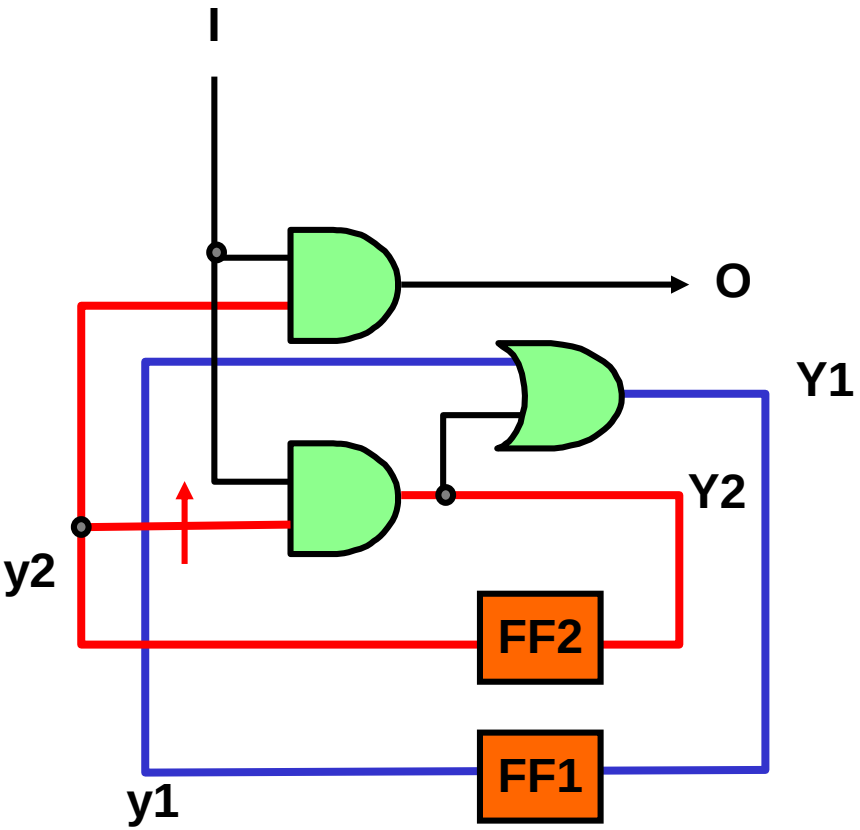
K.-T. Cheng, "Gate-Level Test Generation for Sequential Circuits", *ACM Transactions on Design Automation of Electronic System*, Vol 1, No 4, Oct. 1996

# Extended D-Algorithm [1]

(Kubo, NEC Research & Development, Oct. 1968)
(Putzolu and Roth, IEEE TC, June 1971)

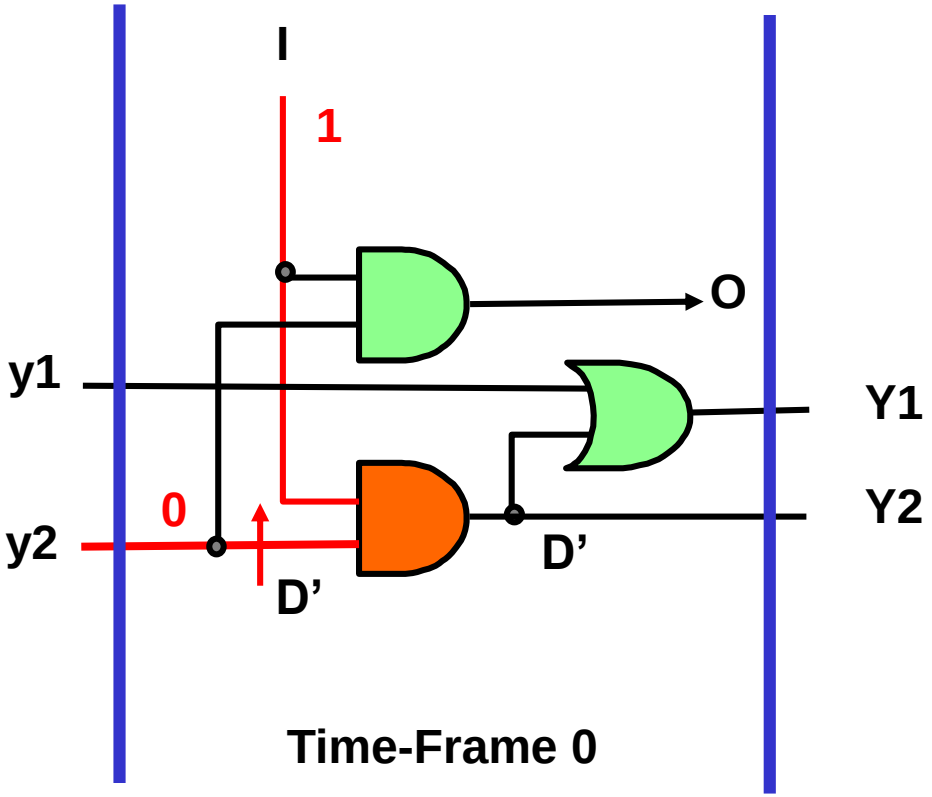
1. **Pick up a target fault f.**
2. **Create a copy of a combinational logic, set it time-frame 0.**
3. **Generate a test for f using D-algorithm for time-frame 0.**
4. **When the fault effect is propagated to the DFFs, continue the fault propagation in the next time-frame.**
5. **When there are values required in the DFFs, continue the justification in the previous time-frame.**

# Example for Extended D-Algorithm



# Example: Step 1

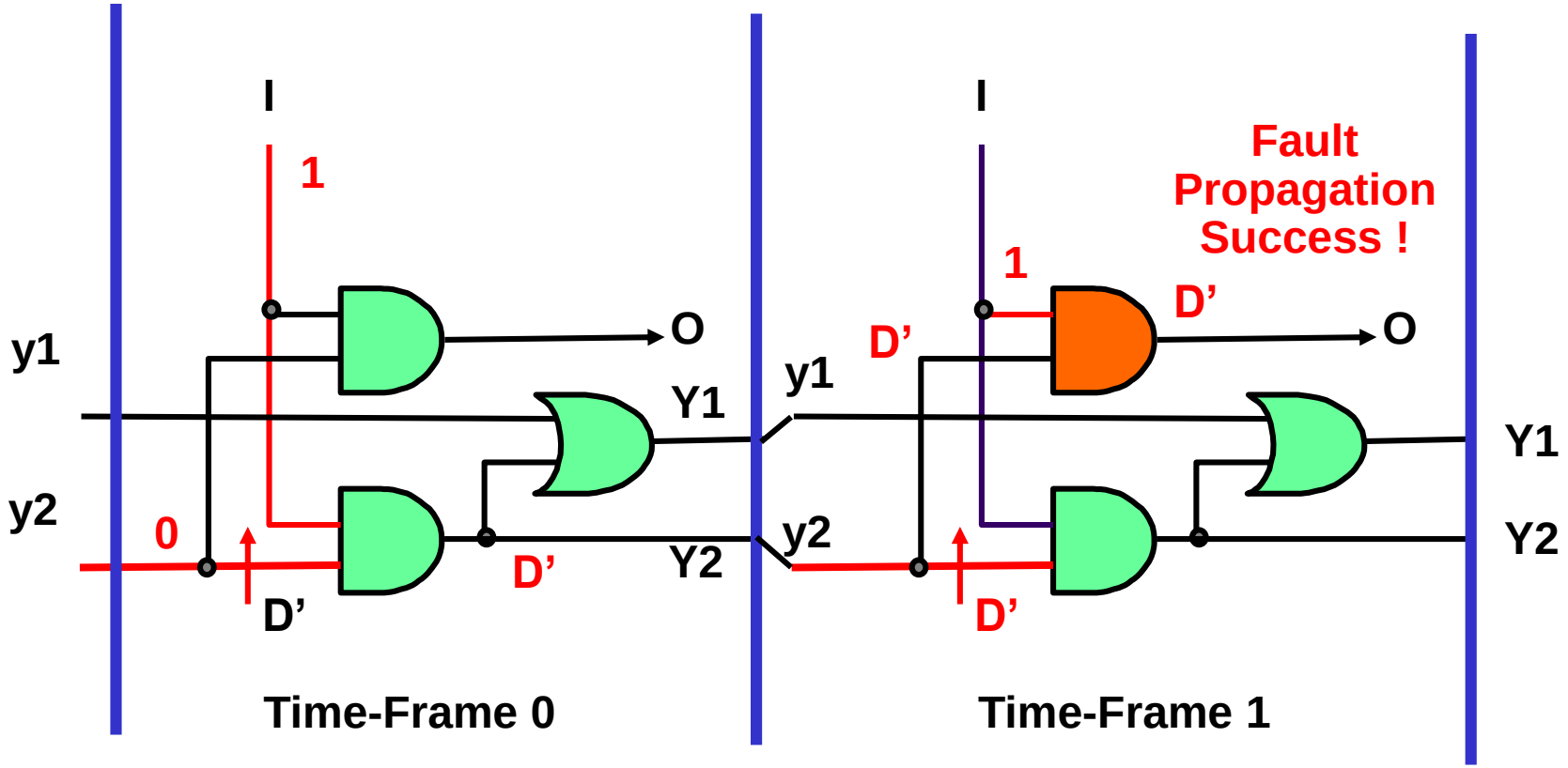
## Fault Activation




**State requirement: value assignments at present-state lines → a state-cube (y1y2 = x0)**

# Example: Step 2

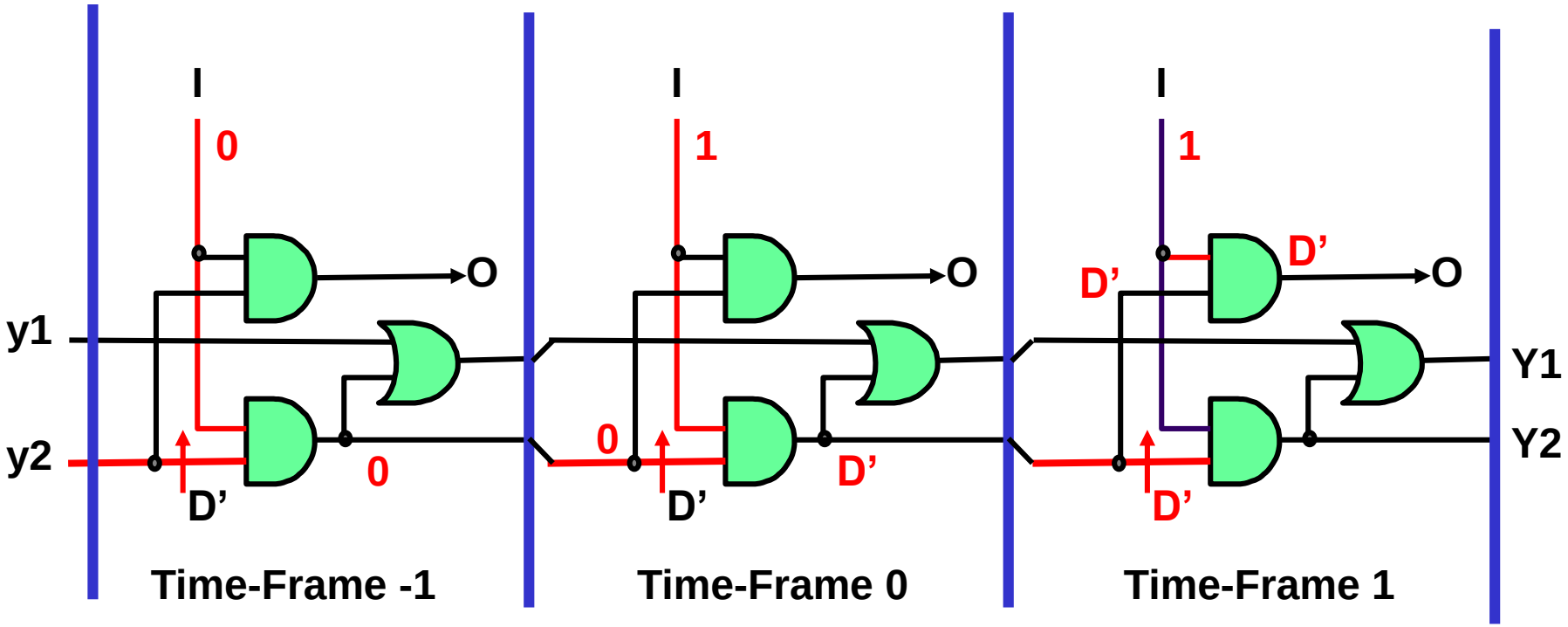
## Forward Fault Propagation



# Example: Step 3

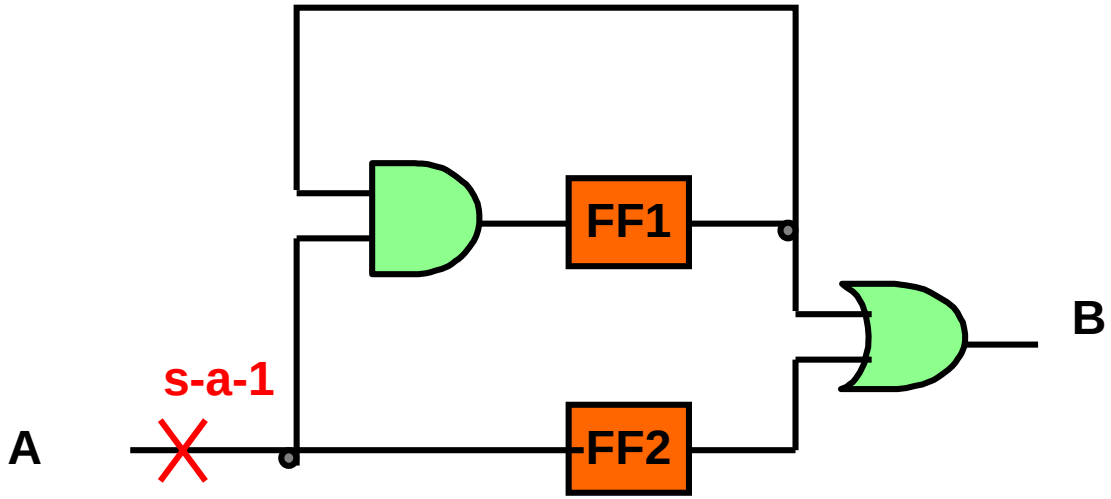
## Backward Line Justification

→ binary values at PPI's needs to be justified from PI's



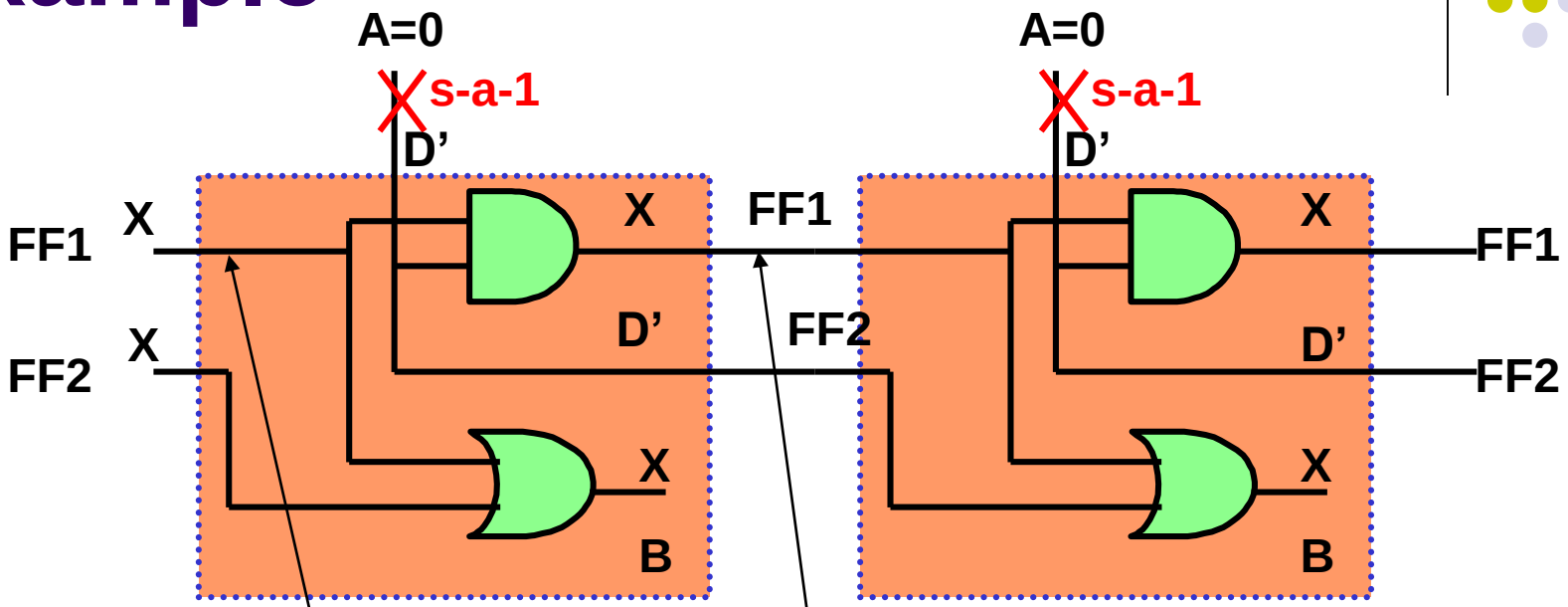

**A Test Sequence (0, 1, 1) is found !**

# Need for 9-Value Logic: An Example




- FF2 is initialized by any input at A
- FF1 is initialized by A=0

# Need for 9-Value Logic: An Example

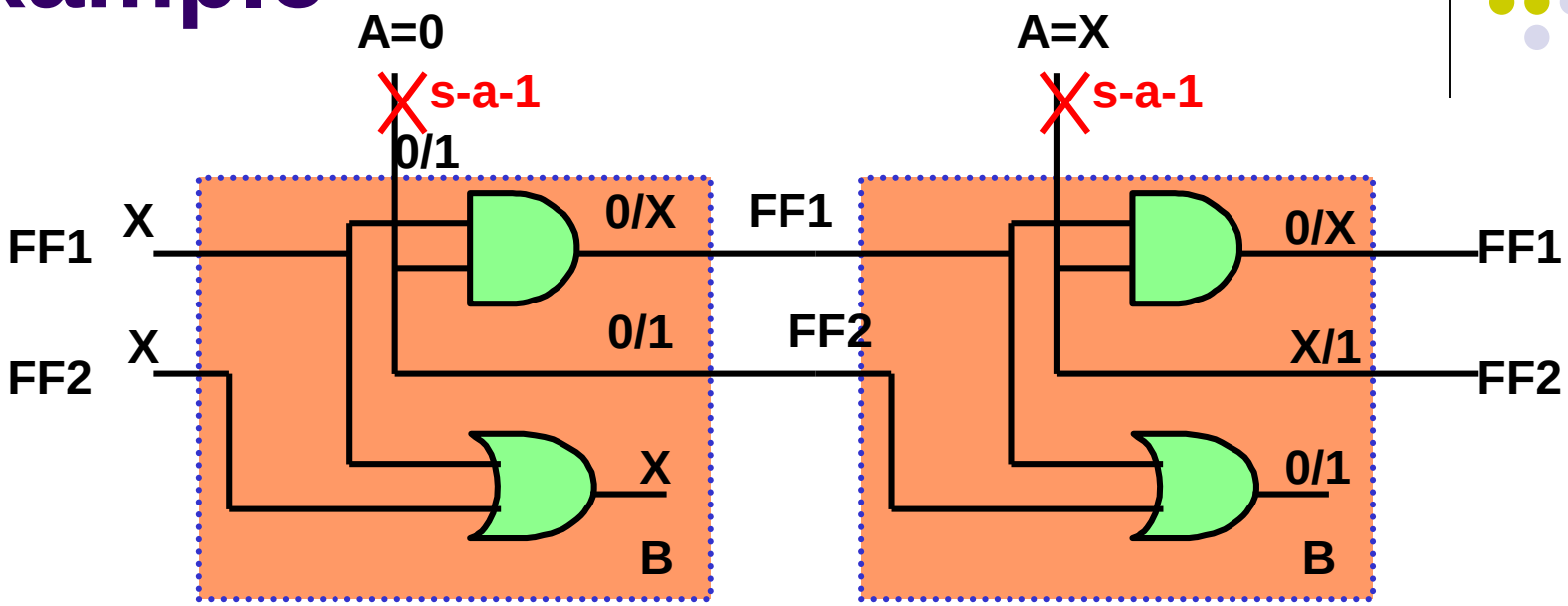



To propagate D' through B, require FF1=0 → Cyclic justification

To propagate D' through AND gate, require FF1=1 → Failed justification

- 5-value logic can't handle situations when faults affect initialization

# Need for 9-Value Logic: An Example

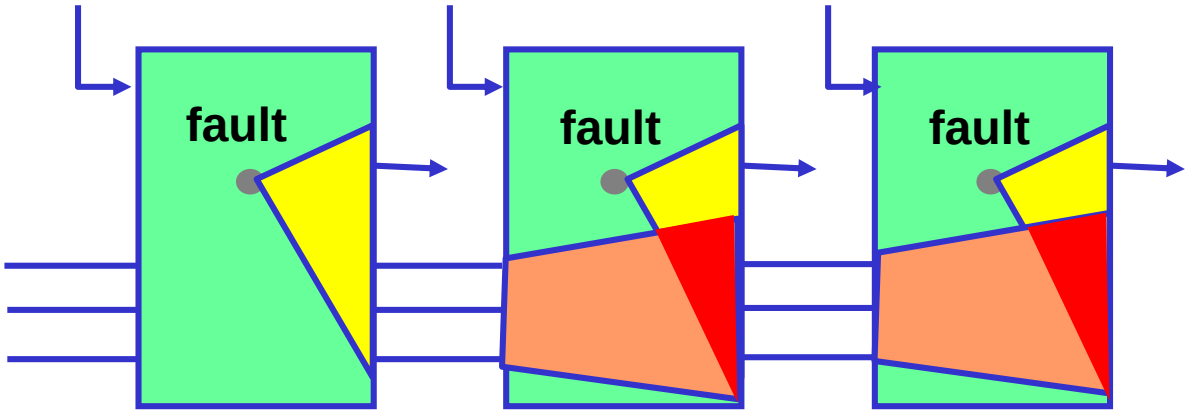



- 9-value logic finds an test $A_0$=0,$A_1$=X→ $B_1$=D' at second cycle

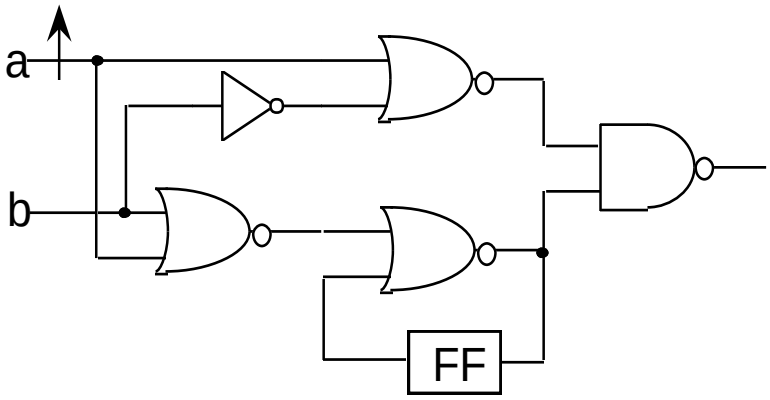
# 9-V Sequential TG [2]

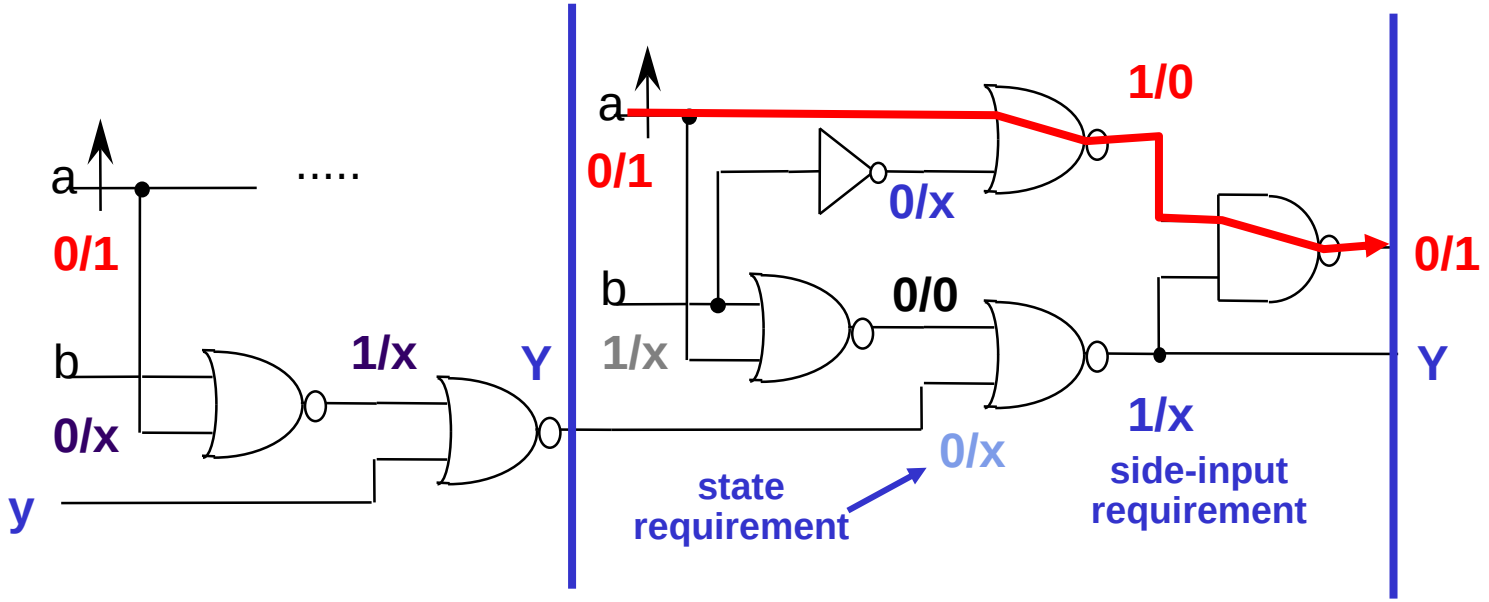
**(Muth, IEEE TC, June 1976)**

- **Extended D-algorithm is not complete.**
- **If nine-value, instead of five-value, is used, it will be a complete algorithm.**
- **Nine-value logic can consider cases where faults affect initialization**

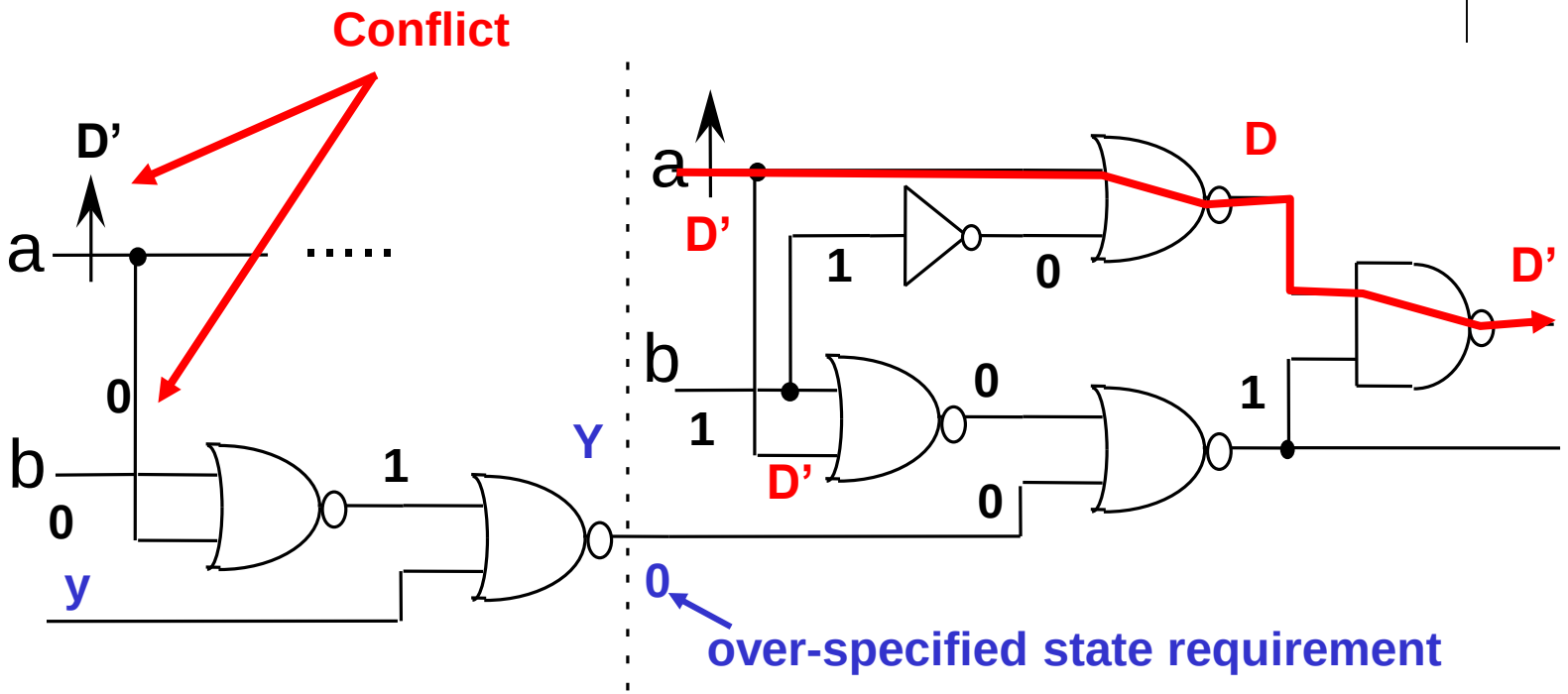


# Example: Nine-Value TG



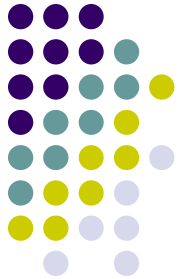
# If Five-Value TG Is Used




Time frame -1 Time frame 0

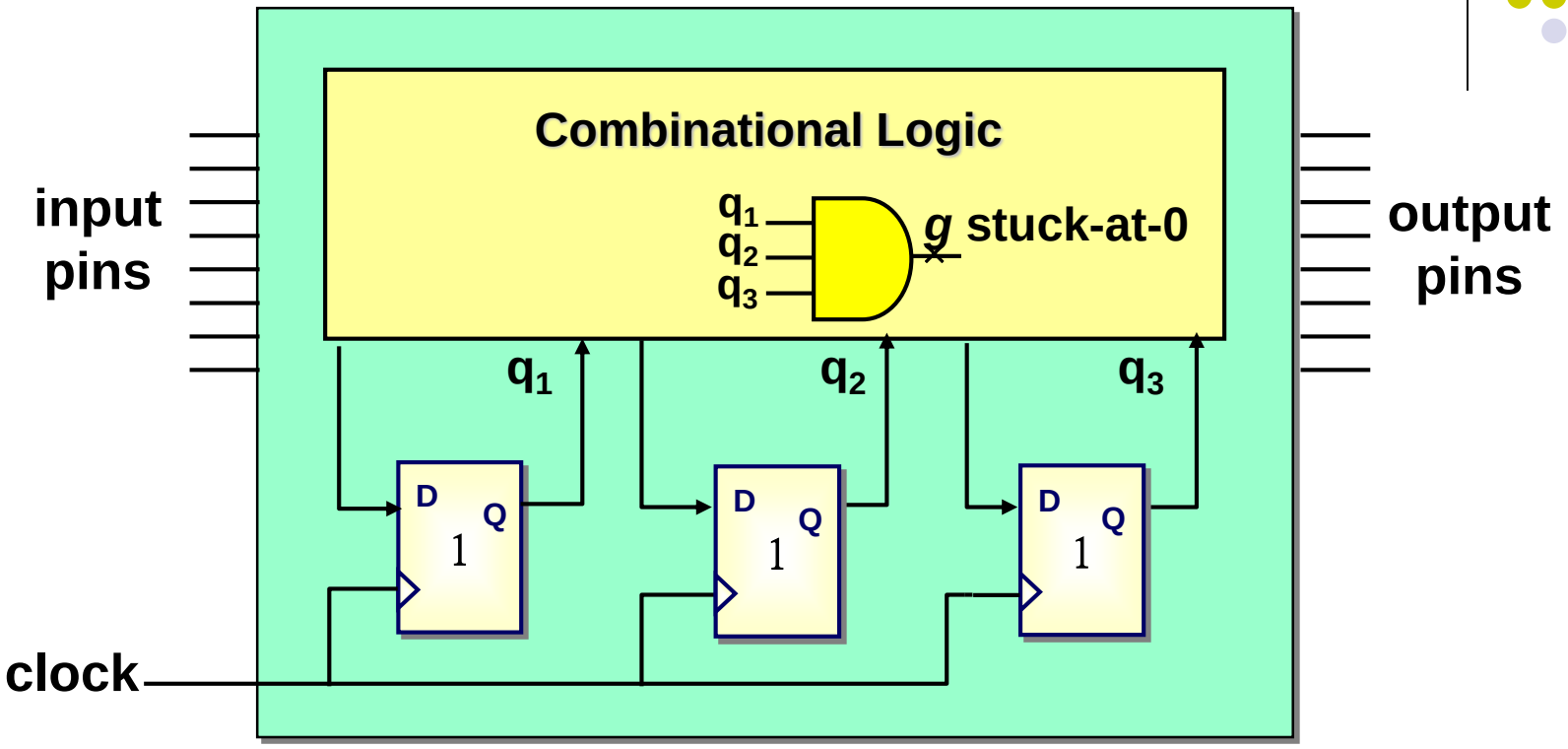
**The test can not be generated by five-value TG.**

# Problems of Mixed Forward and Reverse Time Processing Approaches



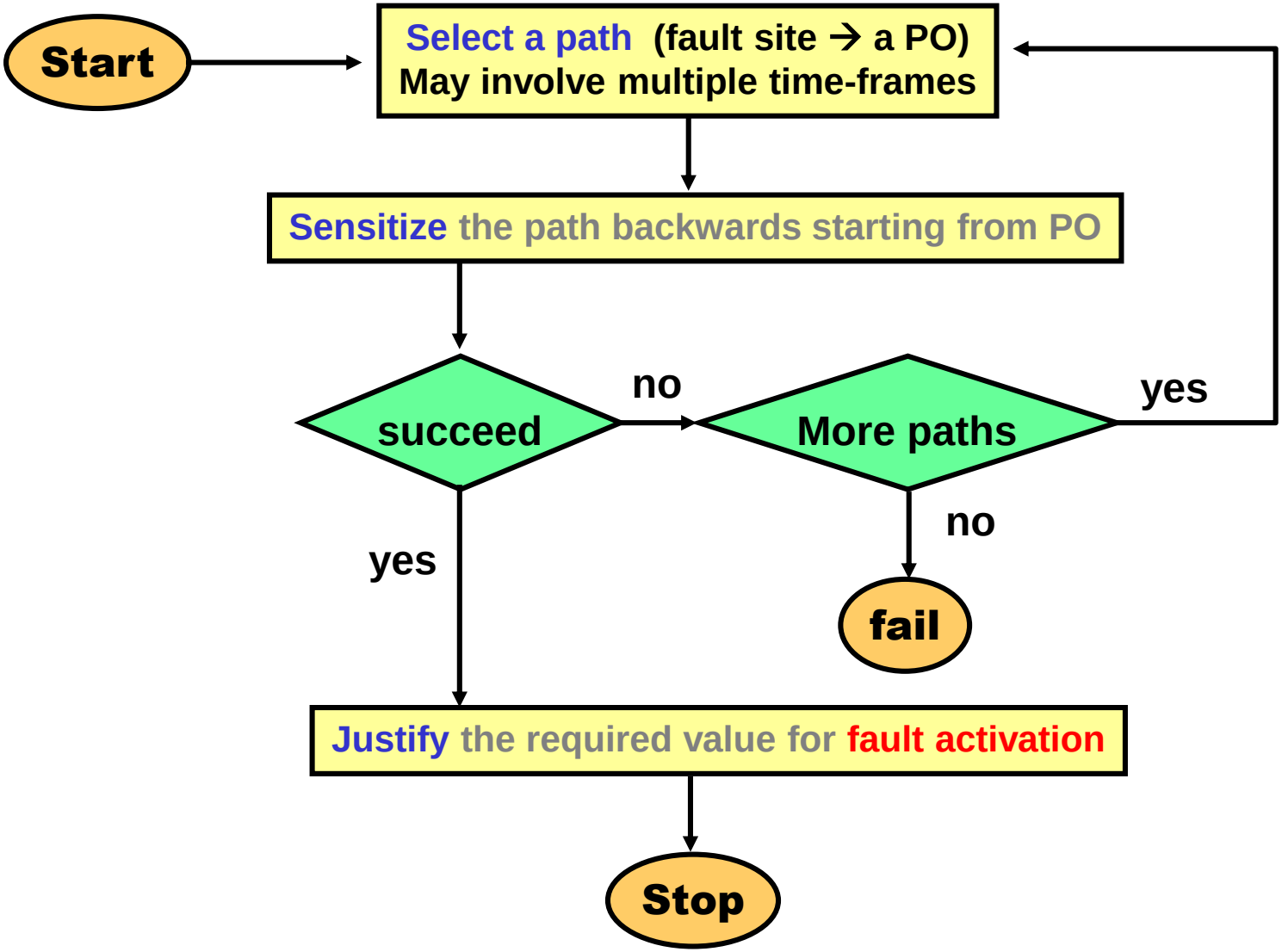
- **The requirements created during the forward process (fault propagation) have to be justified by the backward process later.**
  - **Need going both forward and backward time frames.**
  - **May need to maintain a large number of time-frames during test generation.**
  - **Implementation is complicated.**
  - **Can't detect repetition of states**

# Example: A 3-stage Counter

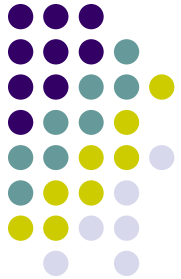



It takes 8 clock cycles to set the flip-flops to be (1, 1, 1), for detecting the target fault *g* stuck-at-0 fault
($2^{20}$ cycles for a 20-stage counter !)

# EBT (Extended Backtrace) Algorithm



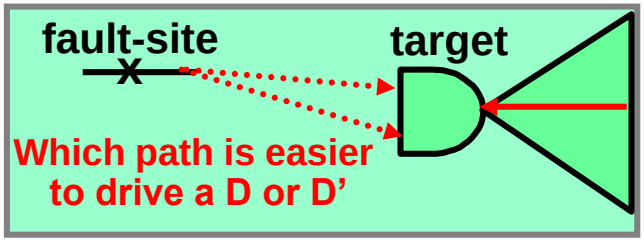
# EBT Algorithm (con't)



- Advantages
  - At any time during test generation process, only two time-frames need to be maintained:
    - Current time-frame
    - Previous time-frame
  - Easier to identify repetition of states and unjustifiable states
- Disadvantages
  - Only single-path is selected
  - The number of possible paths from fault site to POs is very large, trying path by path may not be practical

# Drivability Definition

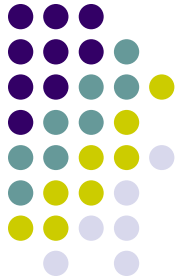
- A measure which can quantify the difficulty of driving the fault effect from the fault-site to each line toward primary outputs
  - Calculate from fault-site
  - Calculate for each faulty machine
  - Separate values for D and D’
- For searching fault propagation paths.
  - BACK algorithm



# Drivability Calculation

- SCOAP controllability for both faulty/fault free circuits
  - Faults injected with extra gates (e.g., 2-input AND for S-A-0).
  - CC0g(s), CC1g(s), CC0f(s), CC1f(s)
  - D-drive at fault site is 0 if S-A-0 is injected (D'-drive is infinity)
  - Example: 3-input AND gate G with inputs A, B and C.
  - To propagate a D from inputs to G (off-inputs of fault free circuit are 1)
    - D_drive(G)=min{ D_drive(A)+CC1g(B)+CC1g(C), D_drive(B)+CC1g(A)+CC1g(C), D_drive(C)+CC1g(A)+CC1g(B)}
  - To propagate a D' from inputs to G (off-inputs of faulty circuit are 1)
    - D'_drive(G)=min{ D'_drive(A)+CC1f(B)+CC1f(C), D'_drive(B)+CC1f(A)+CC1f(C), D'_drive(C)+CC1f(A)+CC1f(C)}

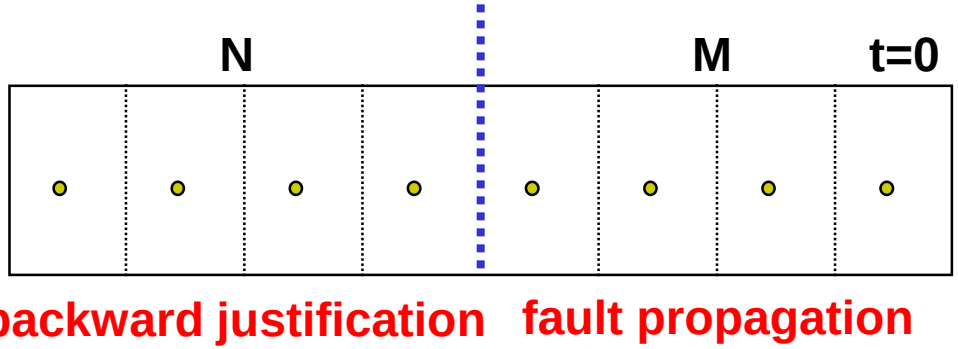
# Back Algorithm



**(Cheng, ICCD Oct. 1988)**

- An improvement of EBT algorithm
- Assign a D or D' to a primary output and justify it backwards
- Use drivability to guide the backward search from the PO to the fault site
  - To justify a D at the output of a gate, select the input with the smallest drivability as the D input
  - Note that: smaller drivability means smaller effort for fault propagation
- The sensitized paths will be created implicitly

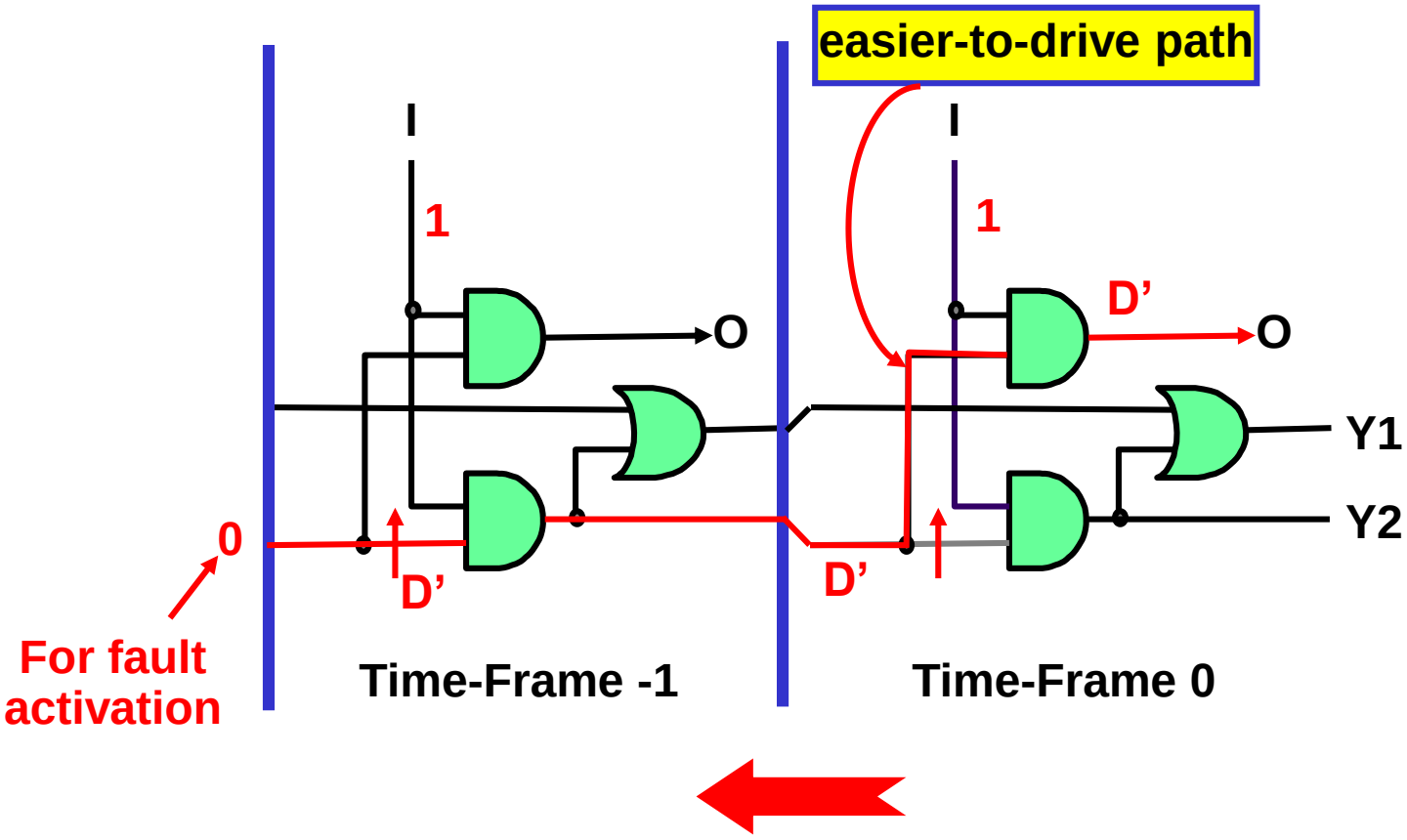
# Back Algorithm




- **Good machine controllability calculation**
- Pick a fault and calculate bad machine controllability and drivability
  - Pick a primary output to have a sensitized value and set t=0
    - Justify all the values required at the current time frame
    - If state matched, test found
    - Else set T=T-1 and continue another time-frame processing

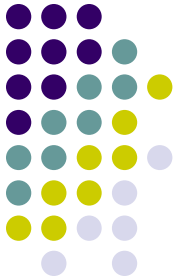
# Example: BACK Algorithm

**Step 1: Drive a D or D' from a PO to the fault site**

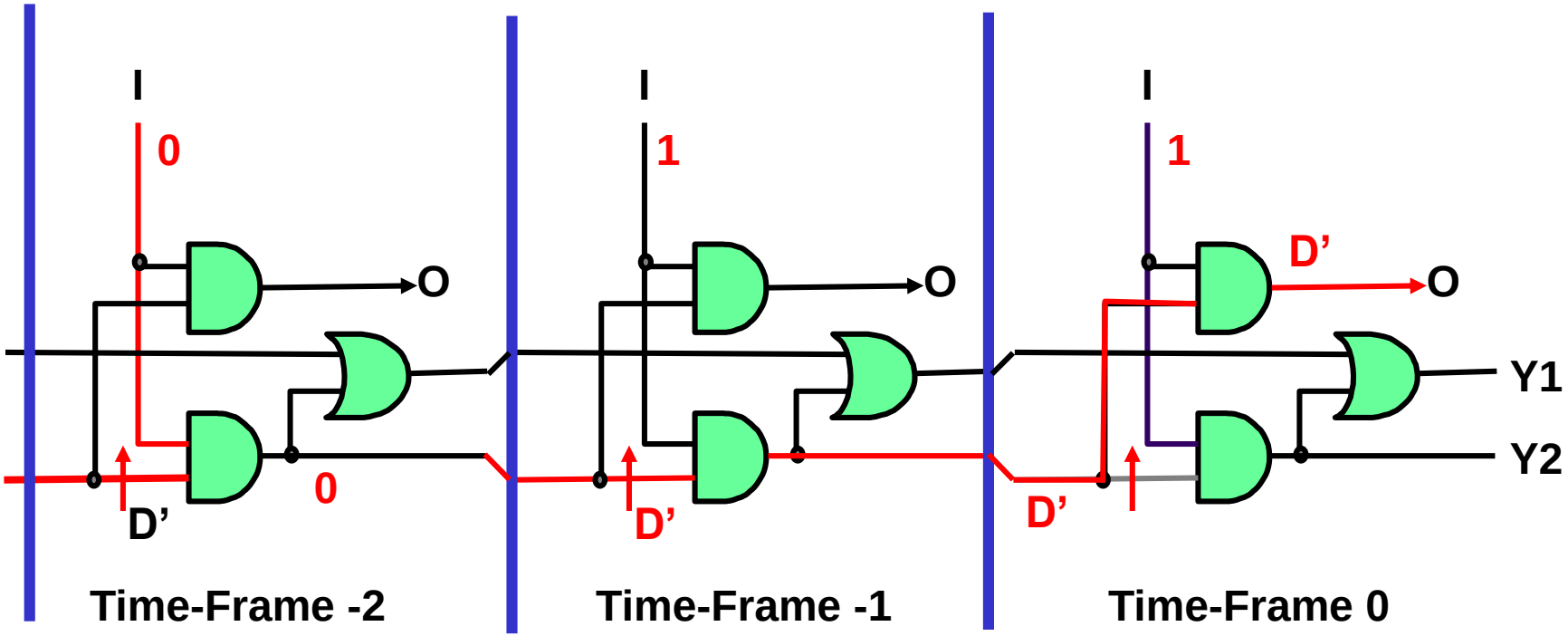



**BACK algorithm involves only reverse-time processing**

# Example: BACK Algorithm



## Step 2: Backward Line Justification

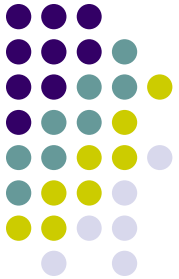



**A Test Sequence (0, 1, 1) is found !**

# The BACK Algorithm

- Main activities:
  - Pre-select a primary output to have value D or D’ guided by drivability
  - Backward line justification
  - Implication
- Memory requirement:
  - Two snapshots only
- Scheduling of justification requirement:
  - Simple

# Outline



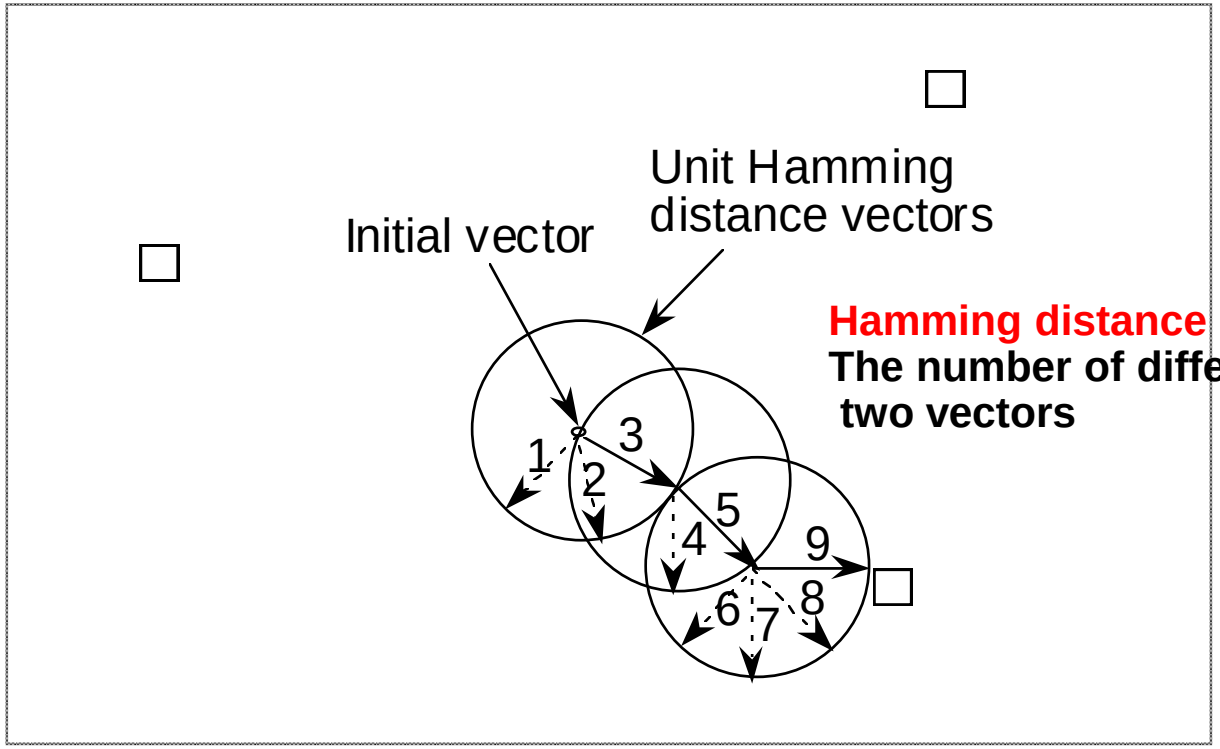
- For Circuits with Unknown Initial States
  - Time-frame Expansion Based:
    - Extended D-algorithm (IEEE TC, 1971)
    - 9-V Algorithm (IEEE TC, 1976)
    - EBT (DAC, 1978 & 1986)
    - BACK (ICCD, 1988), ...
  - **Simulation-Based:**
    - **CONTEST (IEEE TCAD, 1989)**
- For Circuits with Known Initial States
  - STALLION (IEEE TCAD, 1988)
  - STEED (IEEE TCAD, May 1991), ...

# CONTEST: A Concurrent Test Generator for Sequential Circuits [3,4]

**(Agrawal and Cheng, IEEE TCAD, Feb. 1989)**

- Simulation-based test generation.
- It is subdivided into three phases :
  - Initialization
  - Concurrent fault detection
  - Single fault detection
- For different phases, different cost functions are defined to guide the searching for vectors.

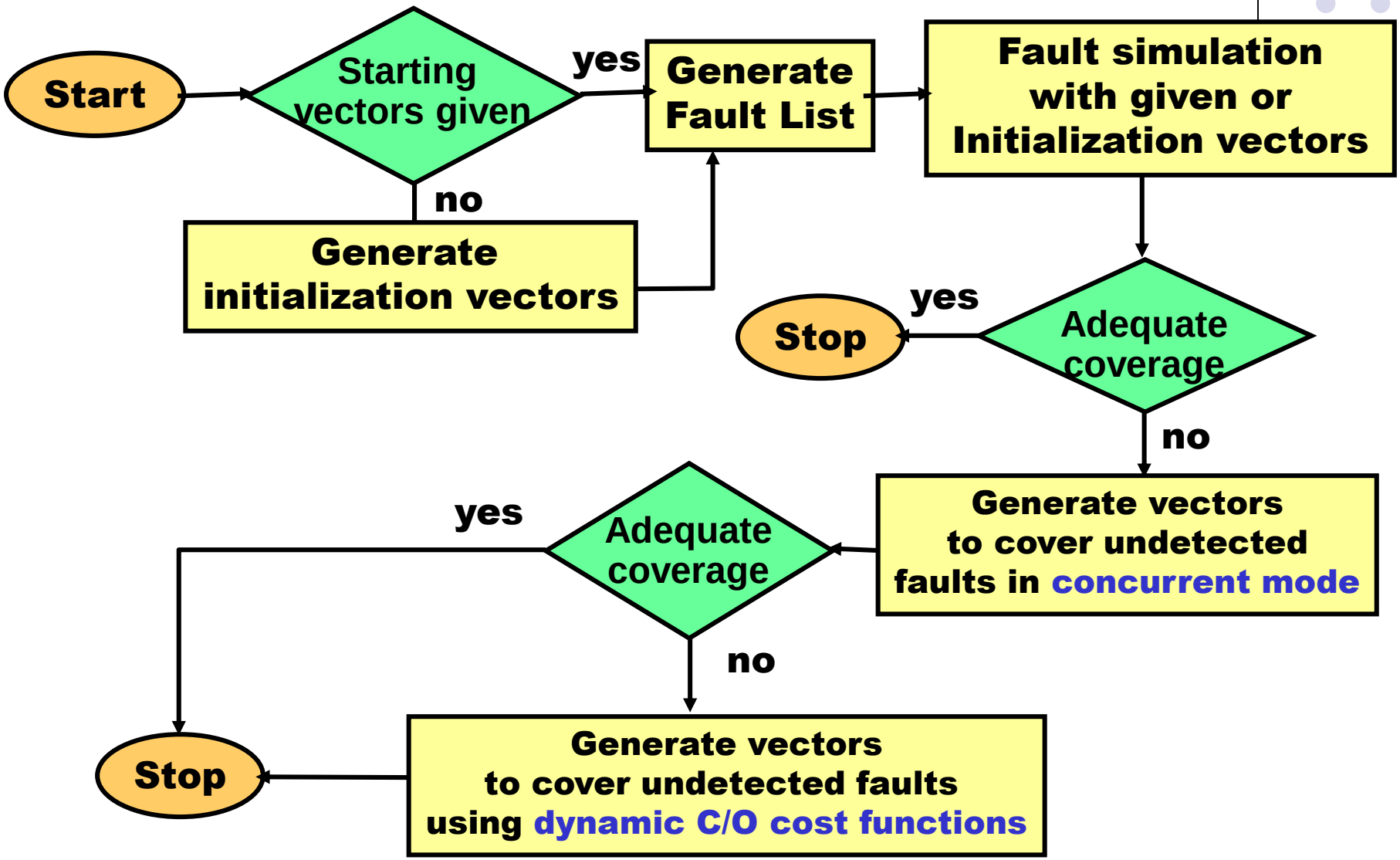
# Directed-Search in Input Vector Space



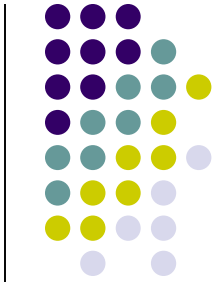

**Hamming distance of two vectors:**
**The number of different bits in these two vectors**

- the possible tests for the given target fault
- the trial vectors that are simulated but not accepted because of their higher costs.
- the accepted moves

# Flow Chart of CONTEST



# Phase 1 : Initialization

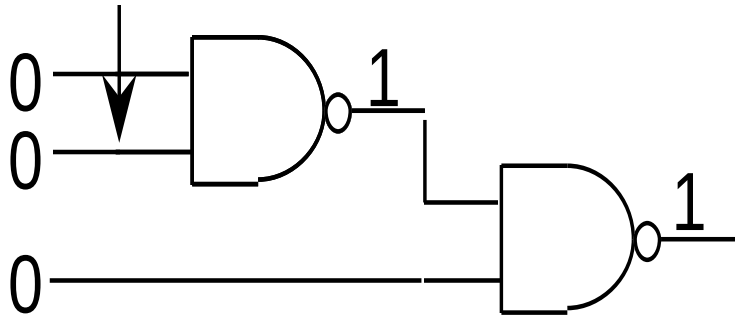


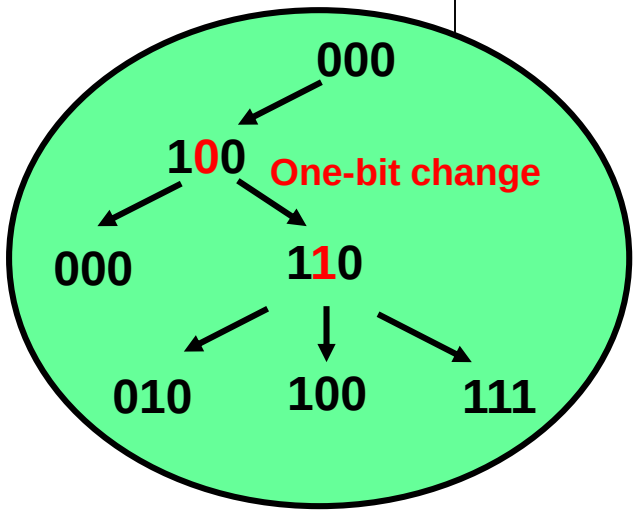
- Start with arbitrary vector.
- Start with all flip-flops in unknown state.
- Use only true-value simulation
  - Cost = number of flip-flops in unknown state.
  - Generate new vectors to reduce the cost by one-bit change in the present vector.
- Stop when the cost drops below desired value.

# Phase 2 : Concurrent Fault Detection

- Start with initialization vectors.
- Fault simulate the vectors and remove detected faults.
- Compute cost function for the last vector.
  - Cost of an undetected fault = minimum distance of its effect from a PO.
  - Cost of the vector = Sum of costs of some undetected faults (e.g., 10% of lowest cost faults).
- Generate new vectors by accepting only those one-bit changes that reduce vector cost.
- Switch to phase 3 when no cost can be reduced.

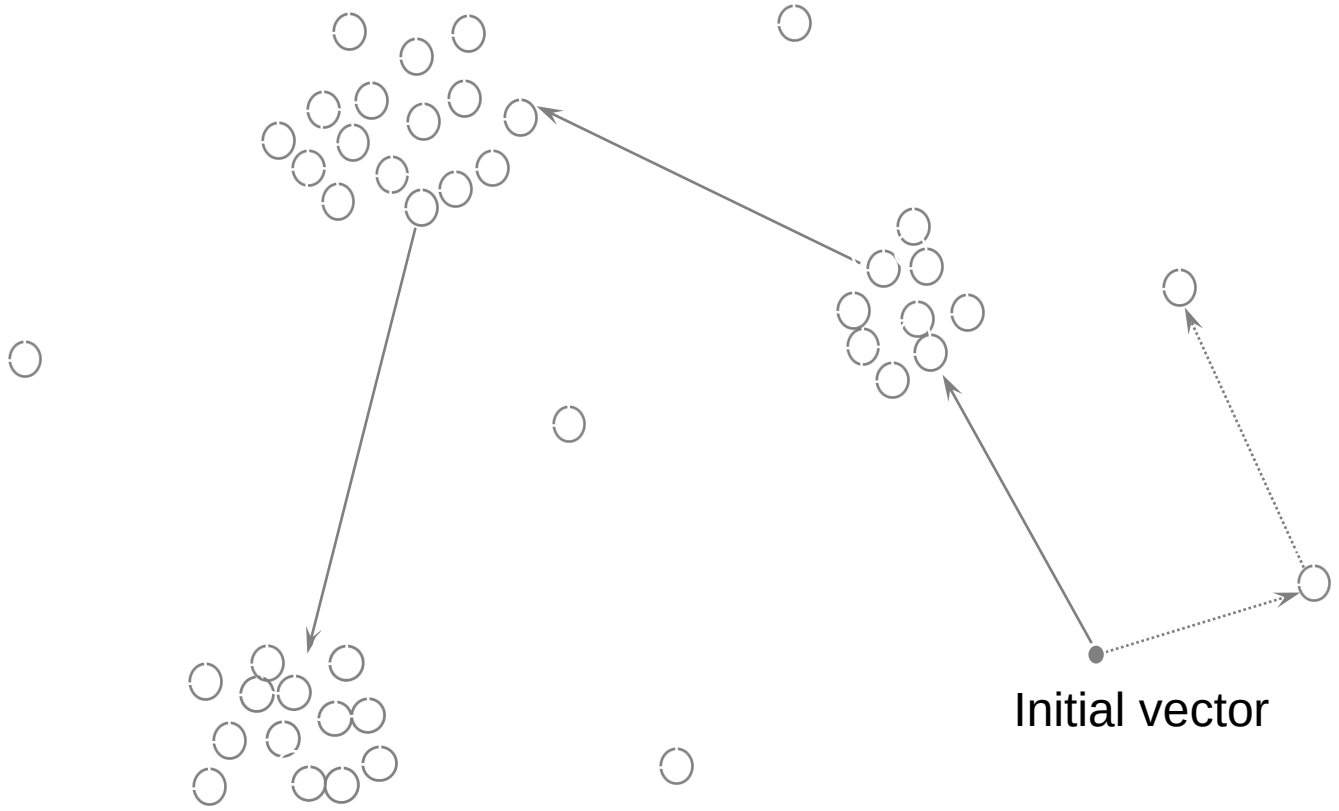
# Example : Distance Cost Function




| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Trial vectors | 0 | 1 | 0 | 1 | 0 | 1 | 1 |
| | 0 | 0 | 0 | 1 | 1 | 0 | 1 |
| | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| Distance | ∞ | 2 | ∞ | 1 | ∞ | 2 | 0 |

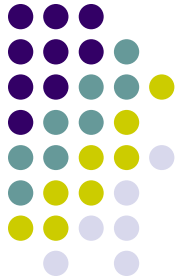
# Multi-Target Guidance




**Search directed by a single target fault**

**Search directed concurrently by a set of faults**

# Phase 3 : Single Fault Detection



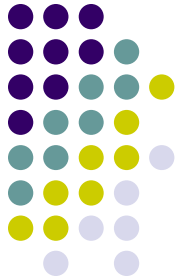
- **Start with the last vector obtained from Phase 2.**
- **Cost(F) = K x AC(F) + PC(i) when F i-s-a-v not activated**

  **= min PC(Nf) when faulty effects of F at Nf**
  - **AC: Activation cost is the dynamic controllability (SCOAP) of faulty line.**
  - **PC: Propagation cost is the minimum (over all paths) dynamic observability (SCOAP) of the faulty line.**
  - **Dynamic testability measures are calculated based on both the circuit structure and the signal states**

  **- K is a large weighting factor.**
- **Generate new vectors by accepting only those one-bit changes that reduce vector cost until the fault is detected or the search is abandoned.**

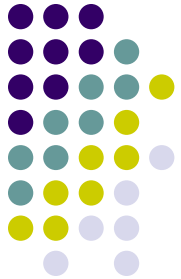
# Genetic Algorithms

(CRIS, GATEST, GATTO, STRATEGATE)



- Improve simulation-based algorithm by "learning" past success of vectors
- Define fitness functions
  - Should involve fault simulation data
- Create an initial population of vectors
- Create new generations of vectors by *crossover, mutation, selection*
  - Crossover: combine bits from two vectors
  - Mutation: manipulate bits of a vector
  - Selection: select the best-fit vectors

# Simulation-Based Approaches



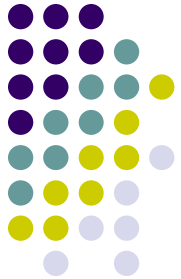
**Advantages:**

- **Timing is considered.**
- **Asynchronous circuits can be handled.**
- **Can be easily implemented by modifying a fault simulator.**

**Disadvantages:**

- **Can not identify undetectable faults.**
- **Hard-to-activate faults may not be detected.**

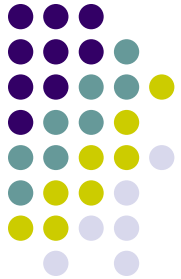
# Results for Simulation-based Approach [8]



| Circuit | Total faults | HITEC [7] | | GATEST [15, 26] | | CRIS [12] | | GATTO [16] | | DIGATE | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Det | Vec | Det | Vec | Det | Vec | Det | Vec | Det | Vec | Dist |
| s298 | 308 | **265** | 306 | 264 | 161 | 253 | 476 | - | - | 264 | 239 | 7 |
| s344 | 342 | 328 | 142 | **329** | 95 | 328 | 115 | - | - | **329** | 109 | 8 |
| s382 | 399 | **363** | 4931 | 347 | 281 | 273 | 246 | - | - | **363** | 581 | 6 |
| s400 | 426 | **383** | 4309 | 365 | 280 | 357 | 758 | - | - | 382 | 3369 | 11 |
| s444 | 474 | 414 | 2240 | 405 | 275 | 397 | 519 | - | - | **420** | 1393 | 9 |
| s526 | 555 | 365 | 2232 | 417 | 281 | 428 | 692 | - | - | **446** | 2867 | 9 |
| s641 | 467 | **404** | 216 | **404** | 139 | 398 | 628 | - | - | **404** | 180 | 17 |
| s713 | 581 | **476** | 194 | **476** | 128 | 475 | 1124 | - | - | **476** | 147 | 17 |
| s820 | 850 | **813** | 984 | 517 | 146 | 451 | 1381 | - | - | 621 | 465 | 9 |
| s832 | 870 | **817** | 981 | 539 | 150 | 370 | 1328 | - | - | 606 | 703 | 9 |
| s1196 | 1242 | **1239** | 453 | 1232 | 347 | 1180 | 2744 | 1226 | 5202 | 1236 | 549 | 13 |
| s1238 | 1355 | **1283** | 478 | 1274 | 383 | 1229 | 4313 | 1274 | 4672 | 1281 | 504 | 12 |
| s1423 | 1515 | 776 | 177 | 1222 | 663 | 1167 | 2696 | 1265 | 3394 | **1393** | 4044 | 30 |
| s1488 | 1486 | **1444** | 1294 | 1392 | 243 | 1355 | 1960 | 1344 | 631 | 1378 | 542 | 5 |
| s1494 | 1506 | **1453** | 1407 | 1416 | 245 | 1357 | 1928 | 1277 | 912 | 1354 | 581 | 6 |
| s5378 | 4603 | 3238 | 941 | 3175 | 511 | 3029 | 1255 | 3277 | 1132 | **3447** | 10500 | 94 |
| s35932 | 39094 | 34902 | 240 | 35009 | 197 | 34481 | 1525 | 32943 | 563 | **35100** | 386 | 301 |
| am2910 | 2391 | 2164 | 874 | 2163 | 745 | - | - | - | - | **2195** | 2206 | 71 |
| mult16 | 1708 | 1640 | 273 | 1653 | 204 | - | - | - | - | **1664** | 915 | 32 |
| div16 | 2147 | 1665 | 189 | 1739 | 634 | - | - | - | - | **1802** | 4481 | 3 |
| pcont2 | 11300 | 3354 | 7 | 6826 | 272 | - | - | - | - | **6837** | 3452 | 0 |
| piir8o | 19920 | 14221 | 347 | 15013 | 531 | - | - | - | - | **15072** | 506 | 0 |
| piir8 | 29689 | 11131 | 31 | - | - | - | - | - | - | **18140** | 603 | 0 |

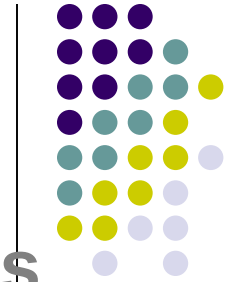
Det: number of faults detected  Vec: test set length  Dist: number of distinguishing sequences generated
Highest numbers of detections are highlighted

# Results for Time-Frame-Based Approach [9]



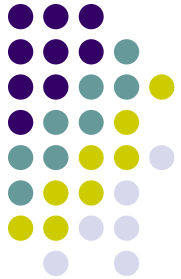
| Circuit | ATOMS | | | | HITEC | | | |
|---|---|---|---|---|---|---|---|---|
| | Det | Unt | Vect | Time | Det | Unt | Vect | Time |
| s298 | 265 | 30 | 312 | 5.5s | 265 | 26 | 322 | 16.2m |
| s344 | 329 | 11 | 127 | 24.9s | 324 | 11 | 115 | 8.1m |
| s349 | 335 | 13 | 118 | 23.9s | 332 | 13 | 128 | 7.7m |
| s382 | 364 | 18 | 5169 | 2.9m | 301 | 9 | 1463 | 1.5h |
| s386 | 314 | 70 | 328 | 3.2s | 314 | 70 | 286 | 7.3s |
| s400 | 384 | 27 | 3868 | 3.2m | 341 | 17 | 1845 | 1.2h |
| s444 | 424 | 29 | 4433 | 5.5m | 373 | 25 | 1761 | 1.5h |
| s510 | 0 | 564 | 0 | 1.0s | – | – | – | – |
| s526 | 451 | 18 | 6552 | 29.0m | 316 | 23 | 436 | 5.8h |
| s641 | 404 | 63 | 184 | 2.0s | 404 | 63 | 209 | 4.8s |
| s713 | 476 | 105 | 172 | 2.4s | 476 | 105 | 173 | 6.7s |
| s820 | 814 | 36 | 997 | 19.9s | 813 | 37 | 1115 | 3.5m |
| s832 | 818 | 52 | 998 | 19.4s | 817 | 53 | 1137 | 5.8m |
| s953 | 89 | 990 | 13 | 6.2s | – | – | – | – |
| s1196 | 1239 | 3 | 373 | 2.6s | 1239 | 3 | 435 | 5.5s |
| s1238 | 1283 | 72 | 409 | 3.2s | 1283 | 72 | 475 | 8.2s |
| s1423 | 1095 | 10 | 570 | 40.6m | 723 | 14 | 150 | 13.9h |
| s1488 | 1444 | 42 | 1101 | 1.1m | 1444 | 41 | 1170 | 16.5m |
| s1494 | 1453 | 53 | 1162 | 36.7s | 1453 | 52 | 1245 | 9.6m |
| s5378 | 3379 | 148 | 839 | 34.2m | 3231 | 217 | 912 | 18.4h |
| s35932 | 34,908 | 3984 | 272 | 1.8h | 34,901 | 3984 | 496 | 4.7h |

# Outline



- For Circuits with Unknown Initial States
  - Time-frame Expansion Based:
    - Extended D-algorithm (IEEE TC, 1971)
    - 9-V Algorithm (IEEE TC, 1976)
    - EBT (DAC, 1978 & 1986)
    - BACK (ICCD, 1988), ...
  - Simulation-Based:
    - CONTEST (IEEE TCAD, 1989)
- **For Circuits with Known Initial States**
  - **STALLION (IEEE TCAD, 1988)**
  - **STEED (IEEE TCAD, May 1991), ...**

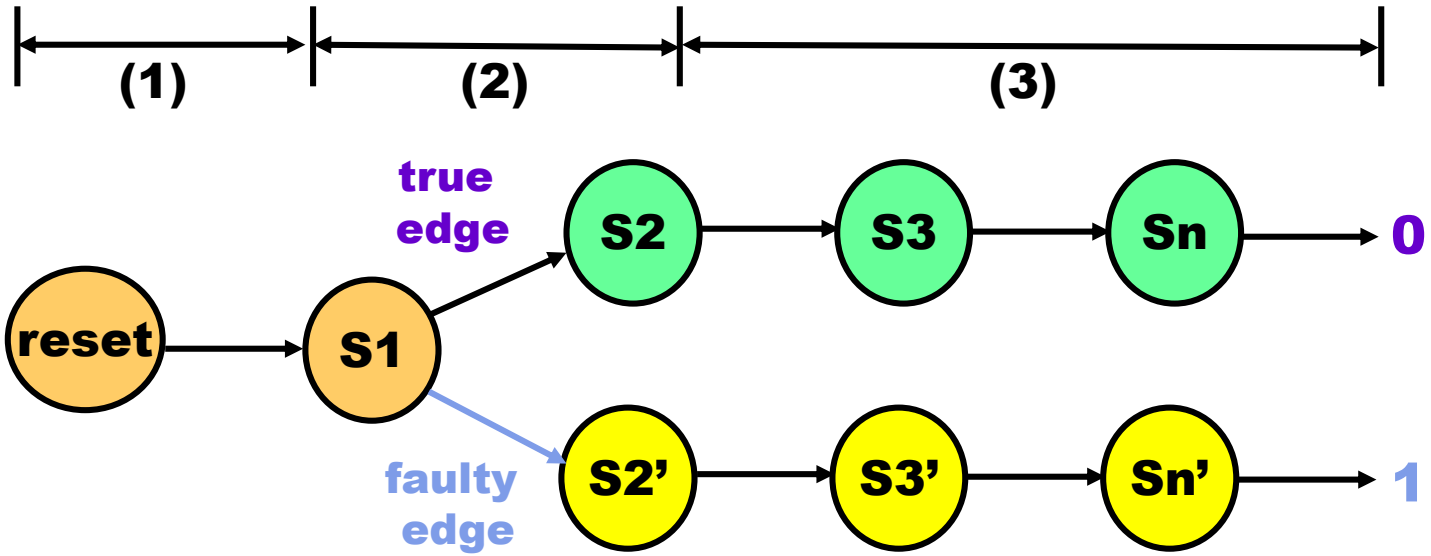
# Test Generation Assuming A known Initial State



- Initialization is avoided.
- Valid for pure controllers that usually have a reset mechanism (reset PI, resetable flip-flops, ...)

# Test Generation for State-Transition Fault

- A test sequence for a state-transition fault has
  - (1) initialization sequence
  - (2) input label of the faulty transition
  - (3) state pair differentiating sequence between good and faulty states

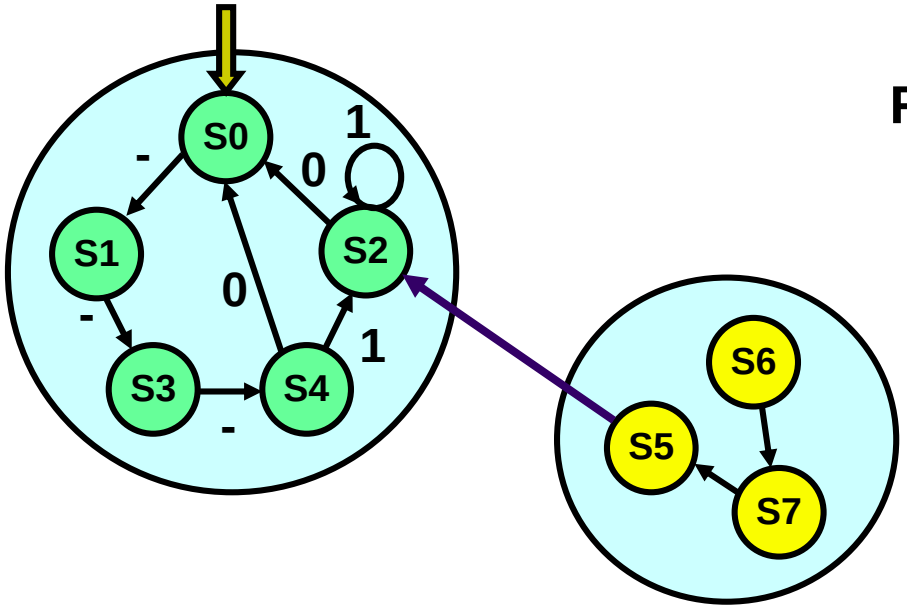


# STALLION [5]

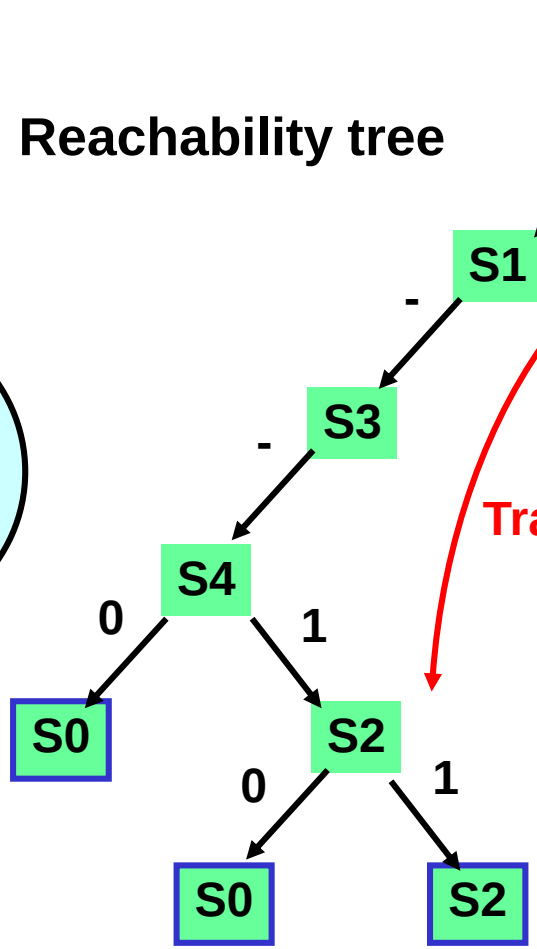
**(Ma et al, IEEE TCAD, Oct. 1988)**

1. Generate State Transition Graph (STG) for fault-free circuit
2. For the fault, find an activation state S and a fault propagation sequence T to a PO using PODEM and the time-frame expansion models
3. When there are values required in PPI's, say state S, find a transfer sequence T0 from initial state S0 to S using the STG
4. Fault simulate sequence T0+T. If it is not a valid test, go to step 2 or 3 to find another sequence

# Finding Transfer Sequence




Reachability tree




Transfer sequence
From S0 to S2
(x,x,x,1)

# STALLION

## Advantages:

- Transfer sequences are easily derived from STG.
- Good performance for controllers whose STG can be extracted easily.

## Disadvantages:

- Fault-free transfer sequence may not be valid.
  - Approximation only
- Extraction of STG may not be feasible for large circuits.

## Heuristics:

- Construct partial STG only.
- If the required transfer sequence can not be derived from partial STG, augment the partial STG.

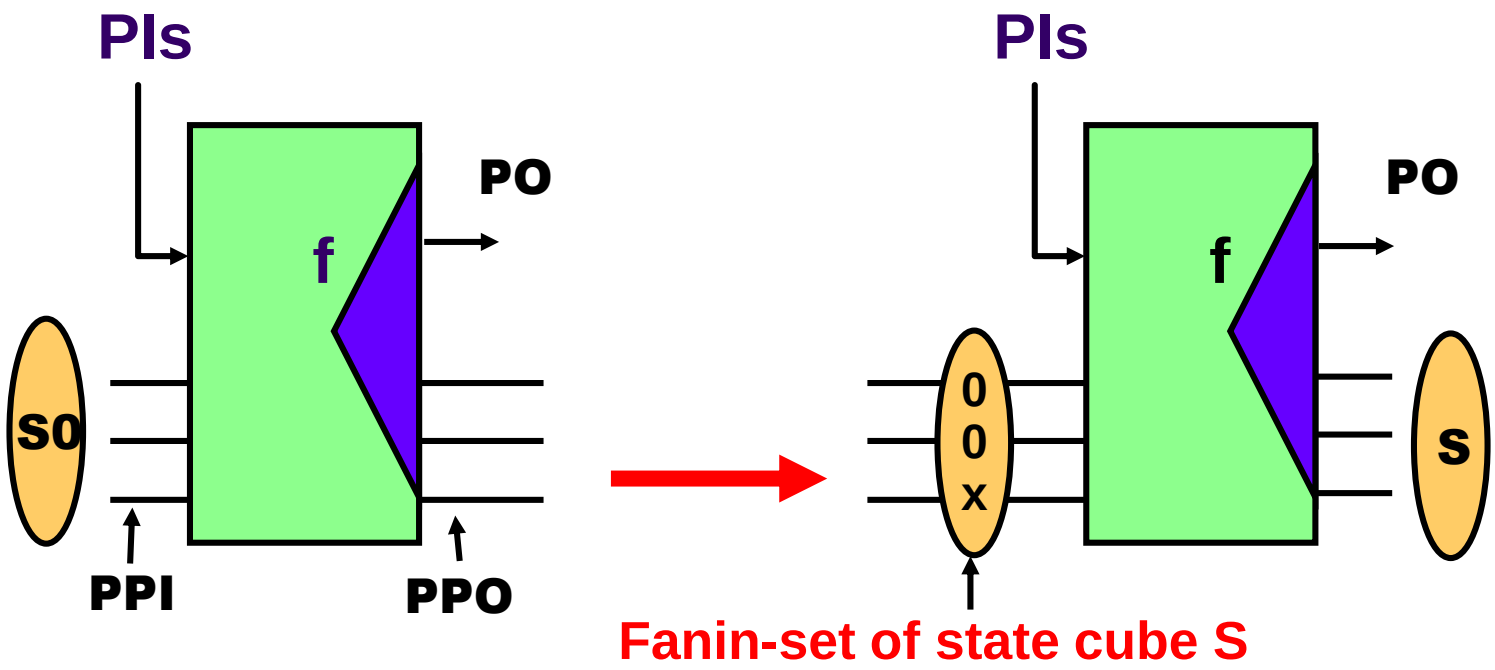
# STEED

**(Ghosh et al, IEEE TCAD, May 1991)**

- Generate ON/OFF sets for each PO and Next State line for fault-free circuit
- Generate a combinational test vector V=(I,S).
  - Assume that the PO and NS values are $O_g$ and $S_g$ for fault-free circuit and $O_f$ and $S_f$ for faulty circuit
- Find a transfer sequence from $S_0$ to S
  - Pick On-set (Off-set) of NS line if corresponding PS line has a 1 (0) in S
  - Intersect the picked On and Off sets (Fanin-set of S)
  - If any cube in Fanin-set covers $S_0$ stops. Otherwise select a cube in fanin-set and continue
- If $O_g \neq O_f$ stops. Otherwise, find a differentiating sequence for $S_g$ and $S_f$
  - Find a PO and an input pattern I that $(I,S_g) \in$ On-set and $(I, S_f) \in$ Off-set (or vice-versa)
  - If not possible, find a differentiating sequence of length greater than 1 using similar procedure

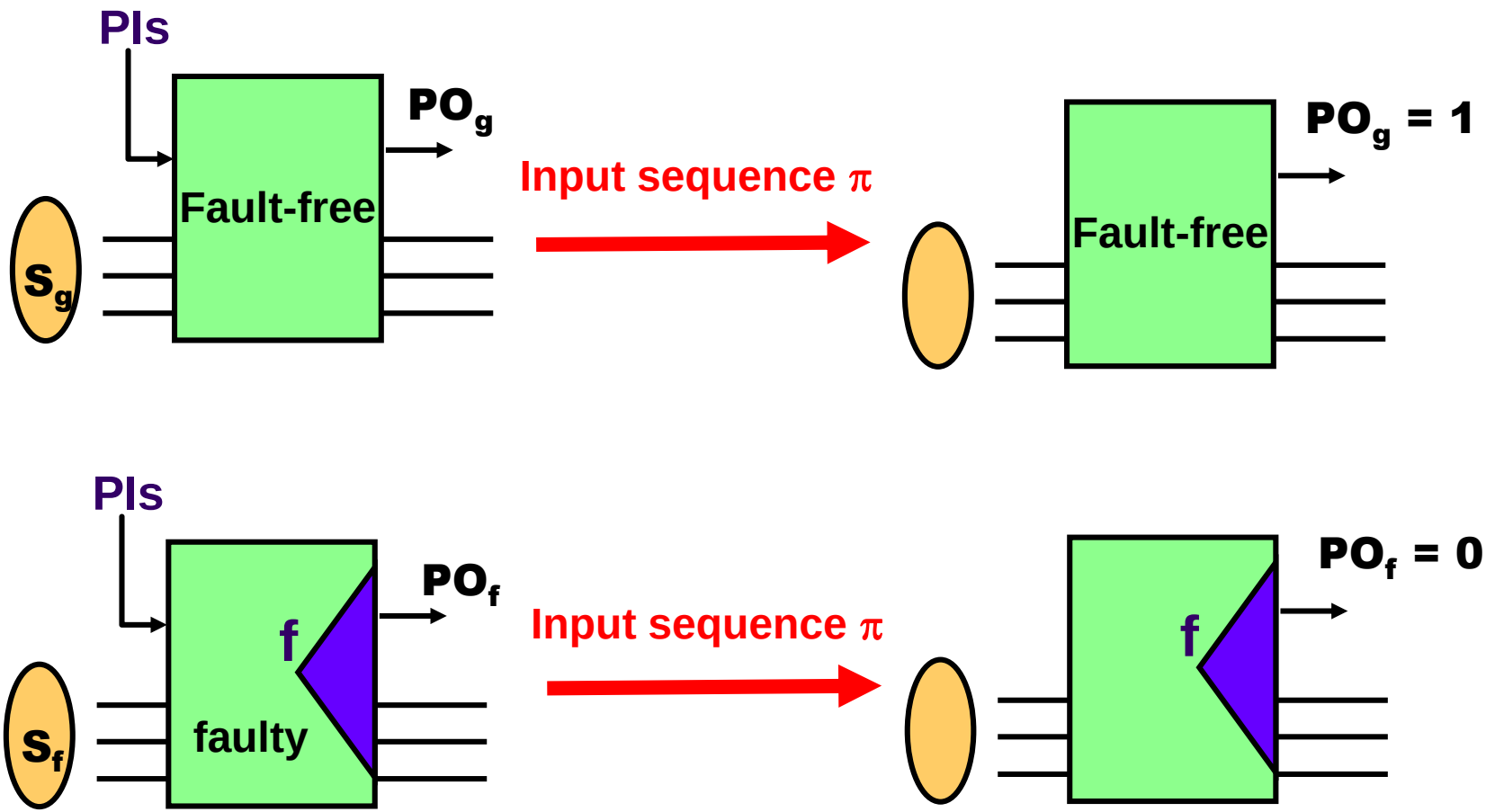
# Transfer Sequence

An input sequence causes a state sequence (S0 →…→ S)

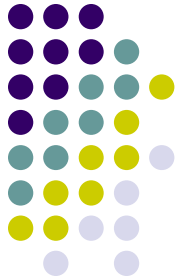


# Differentiating Sequence

An input sequence that can differentiate a state pair ($S_g$,$S_f$)
i.e., final $PO_{fault\text{-}free} \neq PO_{faulty}$

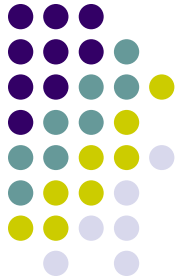


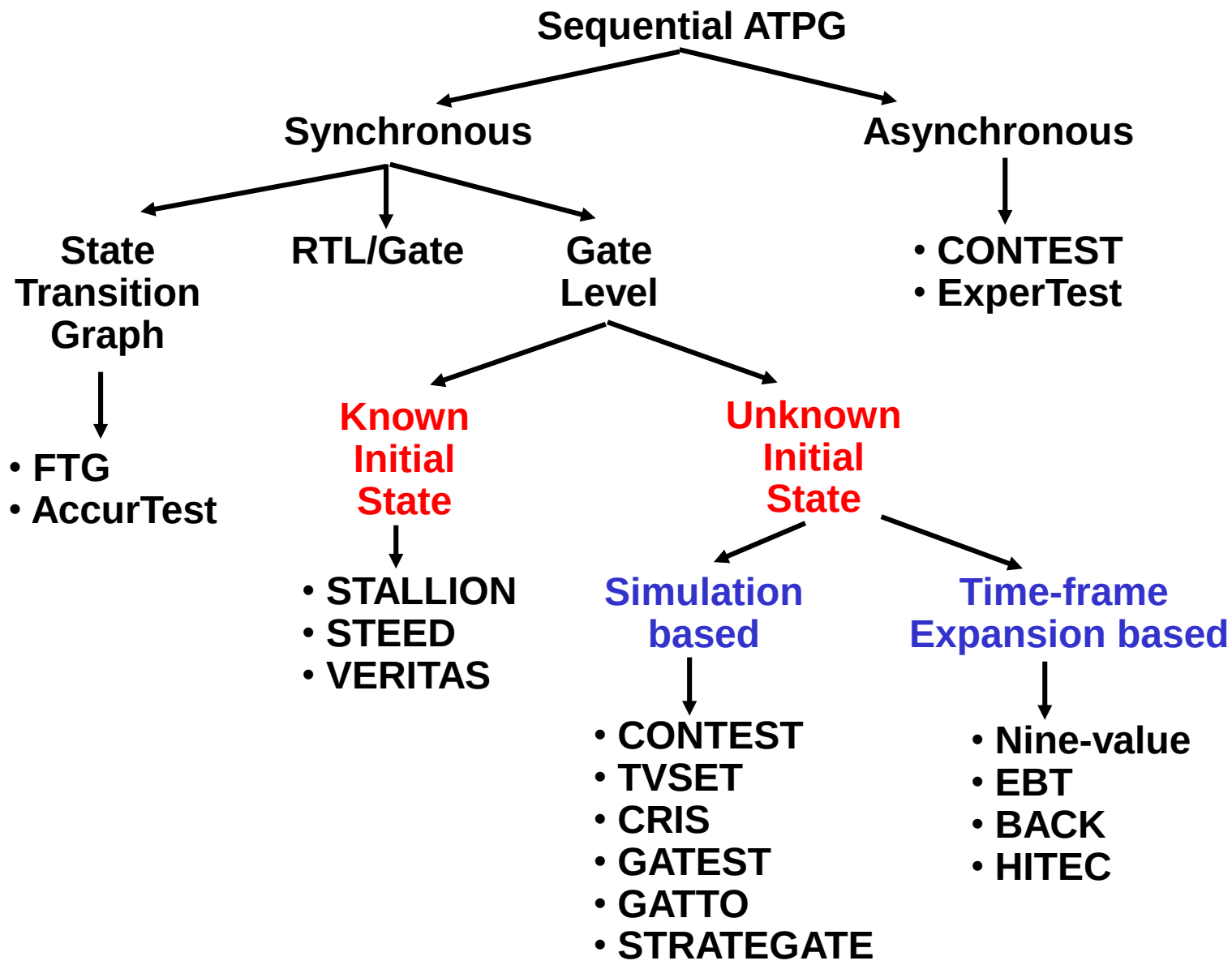
# STEED



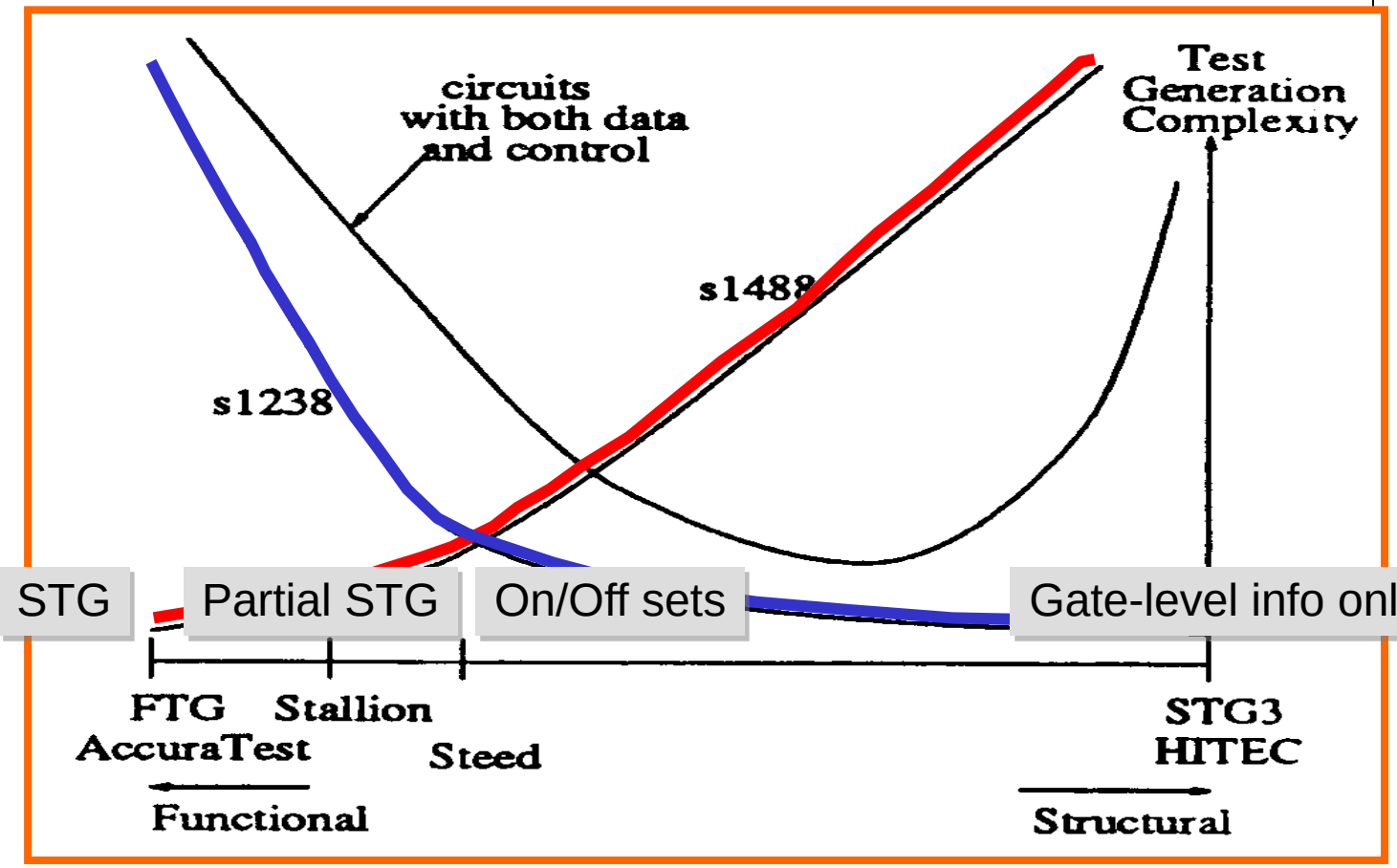
- Advantages
  - ON/OFF-set is a more compact representation than STG
  - Good performance for circuits that have relatively small ON/OFF-sets
- Disadvantages
  - Fault-free transfer and propagation sequences may not be valid
    - Approximation only
  - Generating, storing and intersecting the ON/OFF sets may be very expensive for certain functions, e.g., parity trees

# Sequential ATPG: Taxonomy



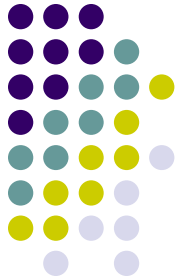


# Circuit Contents And ATPG Algorithms




- Algorithms using functional information (FTG, etc) runs faster on pure control circuits (small # of F.F.; complex NS/PO functions, e.g., s1488).
- STG3 and HITEC runs faster on s1238 (and most larger benchmarks).
- Hybrid ATPG is required

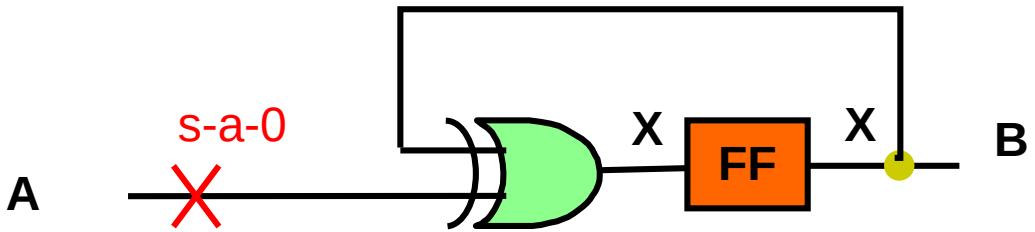
# General Issues on Sequential ATPG (I)



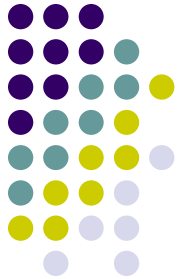
- May not be able to handle highly sequential circuits
  - Consider the stuck-at-0 fault at the MSB of a 20-bit counter....
- Timing may not be accurately modeled in ATPG process
  - The sequence generated may cause races and hazards
  - The sequence must be verified by a simulator
- Tools must guarantee that there is no hazard on asynchronous preset and clear
- Tools must guarantee that test sequence will not cause bus contention
- Multiple-clock/ multiple-phase designs
  - Include clock signals into ATPG process
  - Remodeling of flip-flops
- Efficient Identification of undetectable faults

# General Issues on Sequential ATPG (II)

- Potentially detected faults
  - 1/X, 0/X at primary outputs
- Multiple-observation test
  - Need to observe multiple cycles of outputs
  - For the example circuits
    - Fault-free: A=1→B=(0→1→0…), A=0→ B=0 or 1
    - A difficult circuit for 3-state simulator
    - A multiple-observation test: A=1 for two cycles
    - Fault-free B=0→1 or 1→0
    - Faulty B=1→1 or 0→0
  - An expensive test: need to observe some unknown outputs

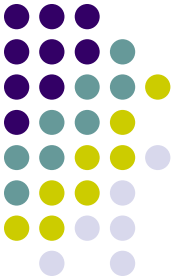


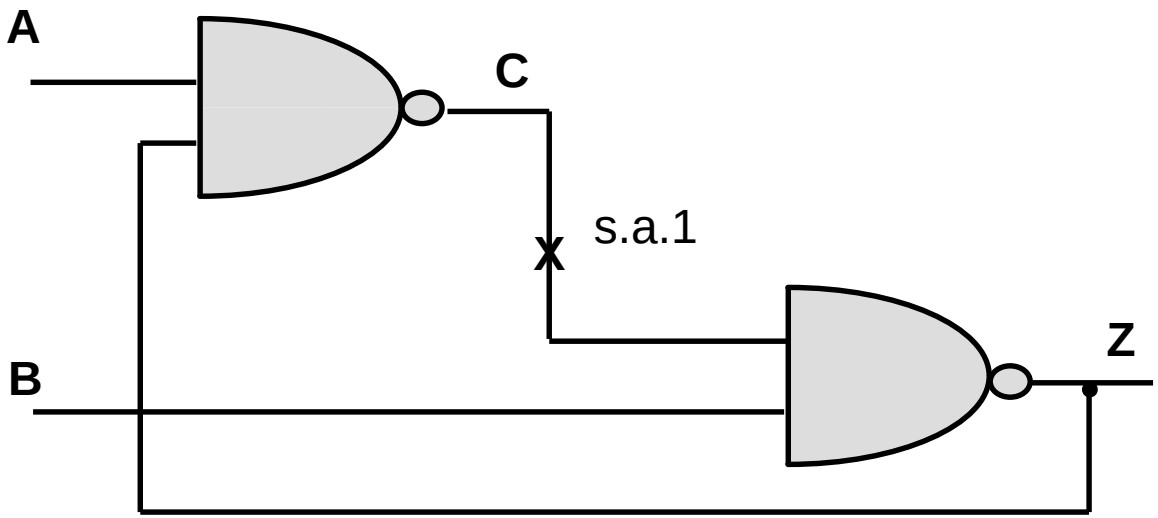
# Test Generation for an Asynchronous Circuit

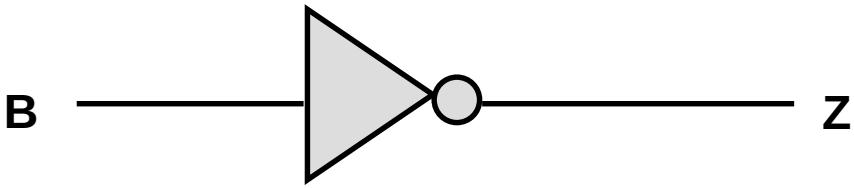


- Test generation for Circuits with combinational loops
  - Apply time-frame expansion techniques with additional constraints
  - Signal values on combinational loops have to be stable for a given PI, i.e., PS/NS have to have the same values under the current inputs.

# Example of an Asynchronous Circuit

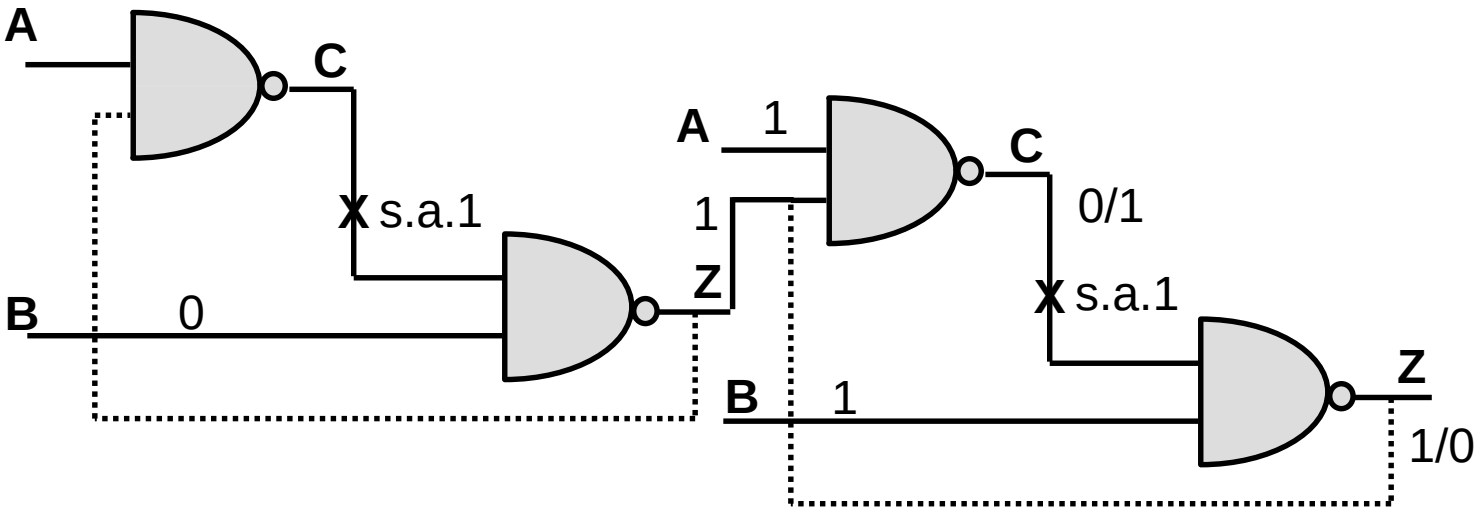





Faulty function

# Time-Frame Expansion for Asynchronous Test Generation




previous time frame

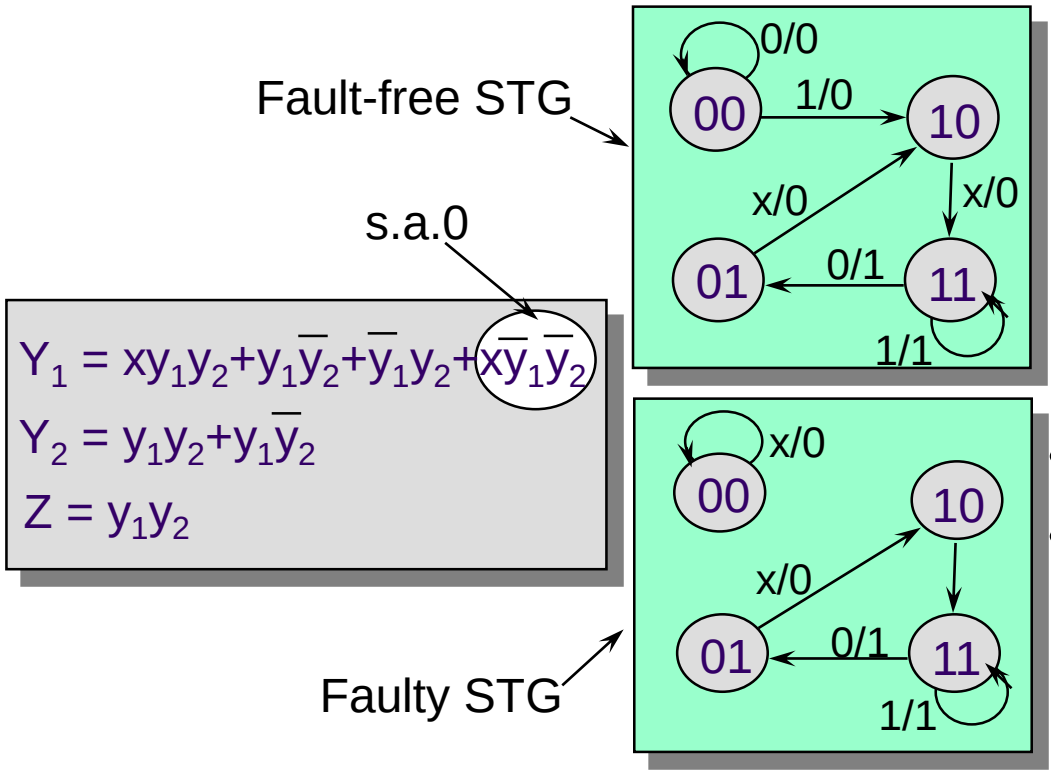
current time frame

Test needs to be verified by a fault simulator with proper timing model !!

(A B)=(X 0) and (1 1) will detect the fault at Z, but assign (A B)=(0 0) will cause a race in the fault-free circuit

# Potentially Detectable Faults

- Faults that are detectable for some initial power-up states & undetectable for other initial power-up states.



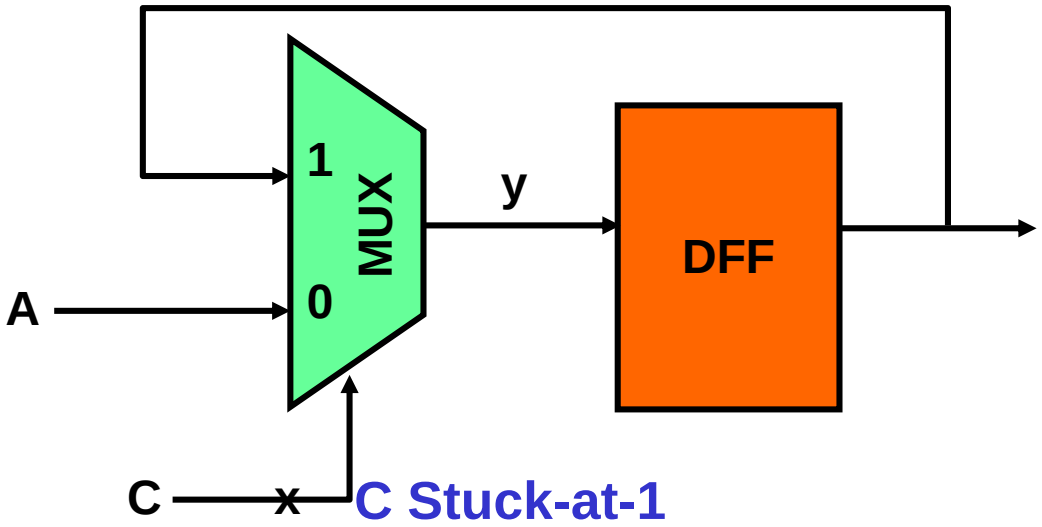

- {011} initializes the circuit to state 11
- The response at Z to {0110}: xxx1

- {011} cannot initialize the circuit
- The response at Z to {0110}:
  - If power-up state is 00: 0000
  - If power-up state is 01,10 or 11: xxx1

The fault is detectable if the power-up state of the faulty circuit is 00, otherwise, it is undetectable => It is potentially detectable.

# Detectability V.S. Redundancy

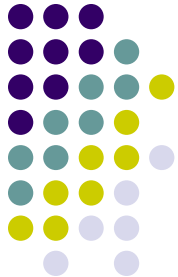
- **A fault is not detectable under some test strategy (e. g. single observation) may not be redundant.**



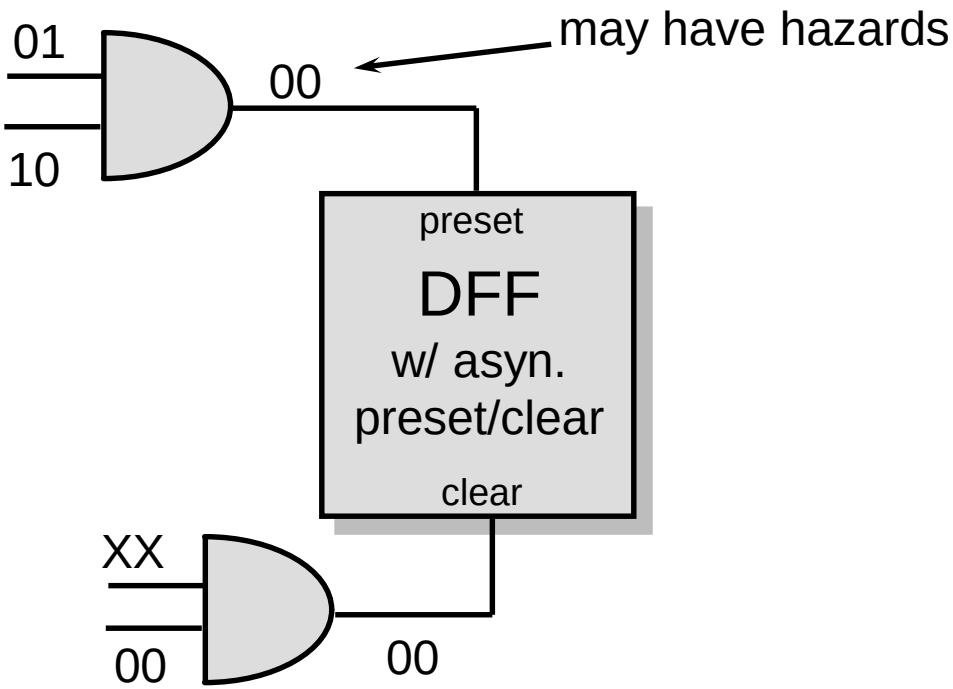

| Sequence | A | C | Initial state = X/0 | Initial state = X/1 |
|---|---|---|---|---|
| $V_1$ | 1 | 0 | 1/0 | 1/1 |
| $V_2$ | 0 | 0 | 0/0 | 0/1 |

(A C)=(1 0) (0 0) will test the fault under multiple observation tests.

# Hazards on Asynchronous Preset and Clear

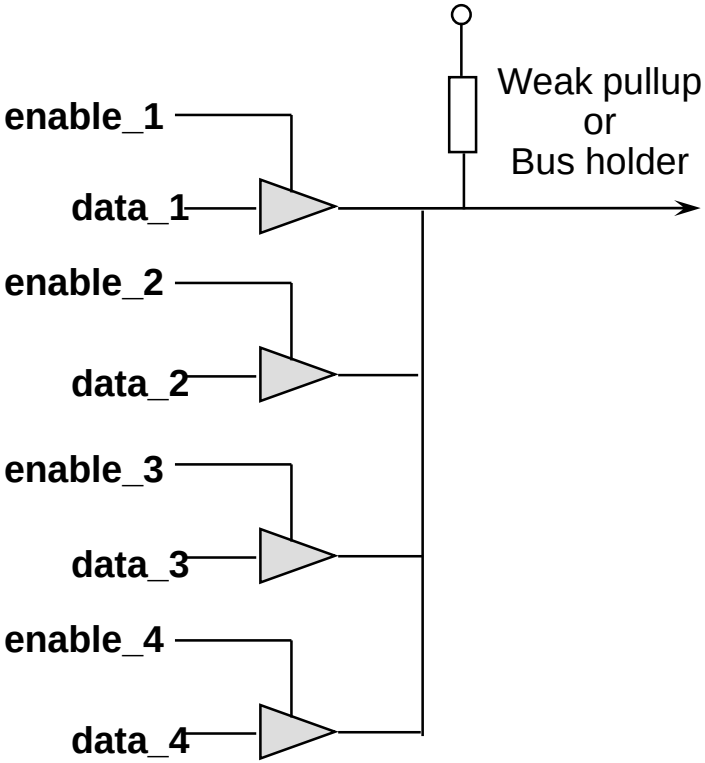


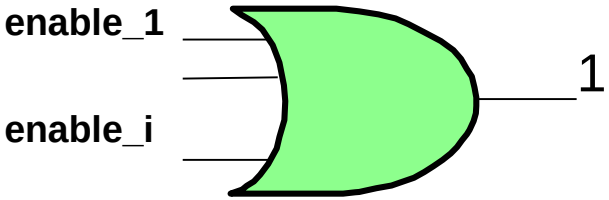
- **Require careful logic simulation of ATPG patterns to prevent unwanted results**



# Bus Contention Resolution

- **Add ATPG constraints (pseudo logic) to confirm**
  - At least one driver is enabled
  - Only drivers with the same data can be enabled




At least one driver is enabled

# References


[1] G. R. Putzolu and T. P. Roth, "A Heuristic Algorithm for the Testing of Asynchronous Circuits", IEEE Trans. Computers, pp. 639-647, June 1971.

[2] P. Muth, "A Nine-Valued Circuit Model for Test Generation", IEEE Trans. Computers, pp. 630-636, June 1976.

[3] V. D. Agrawal, K. T. Cheng, and P. Agrawal, "A Directed Search Method for Test Generation Using a Concurrent Simulator", IEEE Trans. CAD, pp. 131-138, Feb. 1989.

[4] K. T. Cheng and V. D. Agrawal, "Unified Methods for VLSI Simulation and Test Generation", Chapter 7, Kluwer Academic Publishers, 1989.

[5] H-K. T. Ma, et al, "Test Generation for Sequential Circuits", IEEE Trans. CAD, pp. 1081-1093, Oct. 1988.

[6] K.-T. Cheng, “Recent Advances in Sequential Test Generation”, Int’l Test Symposium, April 1992.

[7] K.-T. Cheng, “Gate-Level Test Generation for Sequential Circuits”, ACM Transactions on Design Automation of Electronic System, Vol 1, No 4, Oct. 1996.

[8] M. S. Hsiao, E. M. Rudnick, and J. H. Patel, 1996. “Automatic test generation using genetically-engineered distinguishing sequences.” VTS 1996.

[9] I. Hamzaoglut and J. H. Patel, “Deterministic test pattern generation techniques for sequential circuits”, ICCAD 2000.