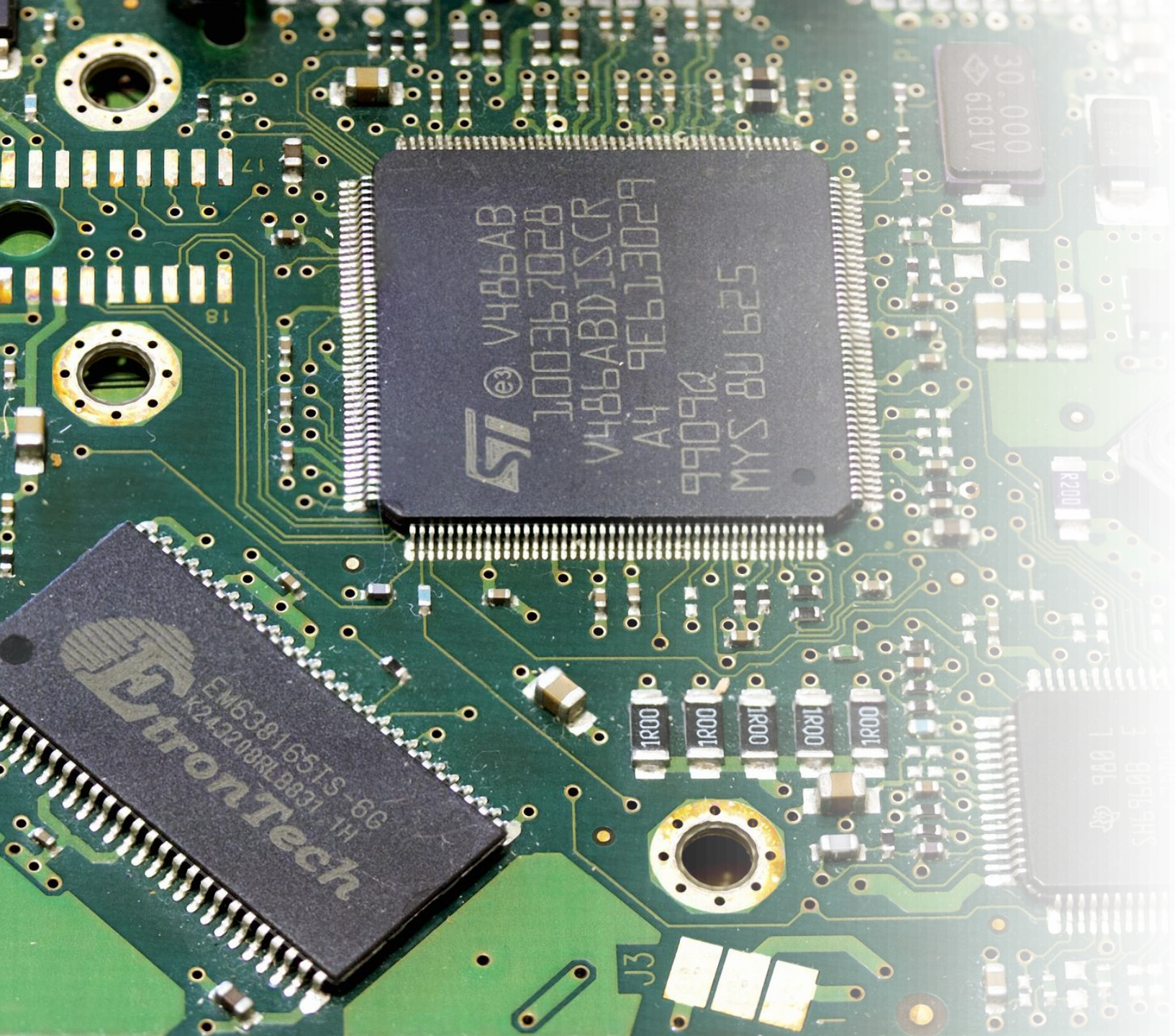


Digital Logic and System Design

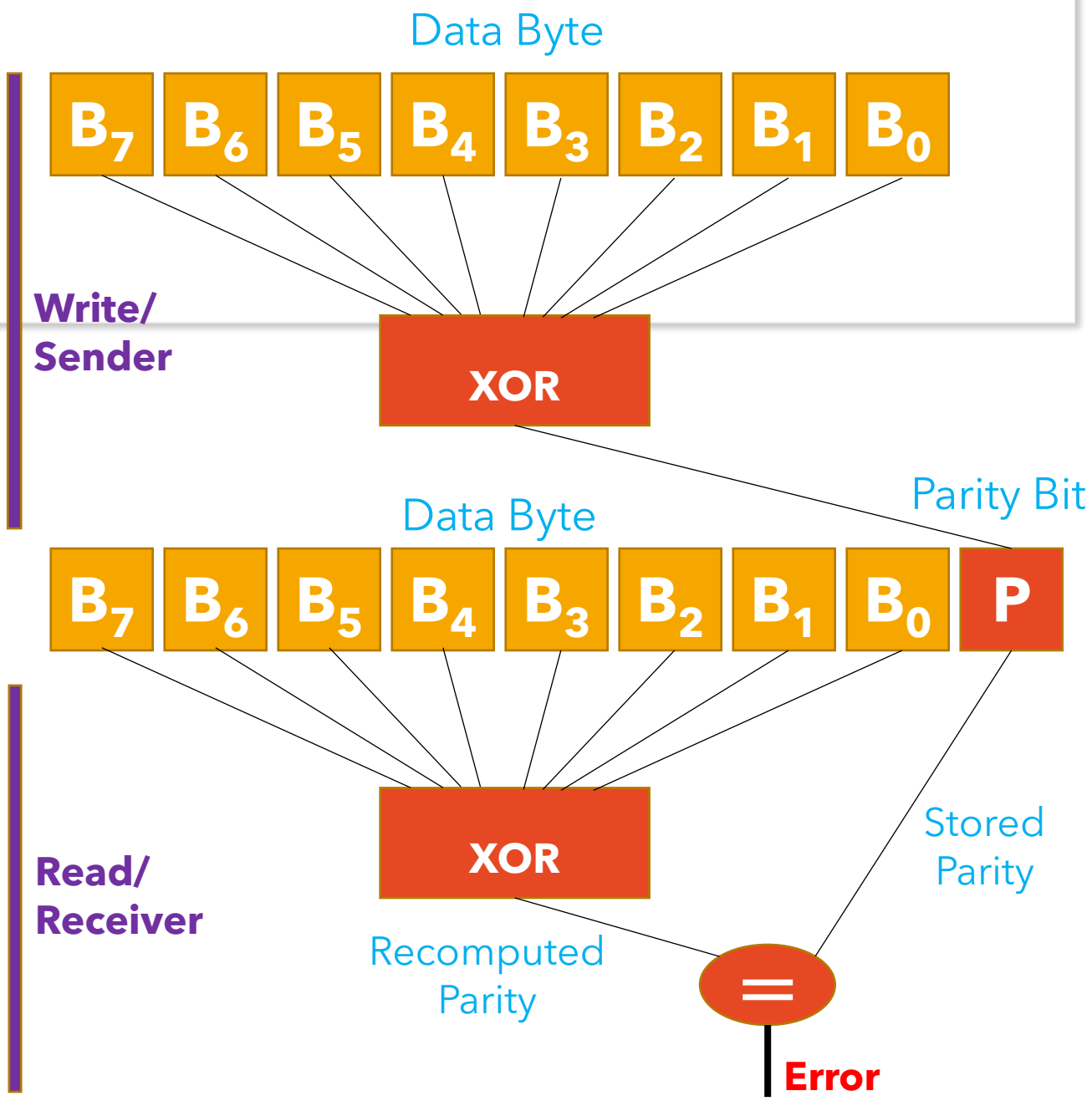
10. Error Detection & Correction
Test & Validation

COL215, I Semester 2023-2024
Venue: LHC 111
'E' Slot: Tue, Wed, Fri 10:00-11:00

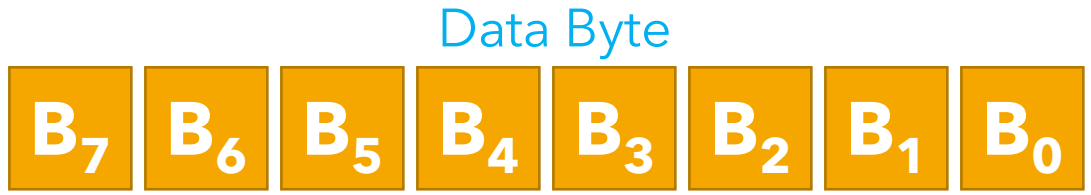
Instructor: Preeti Ranjan Panda
panda@cse.iitd.ac.in
www.cse.iitd.ac.in/~panda/
Dept. of Computer Science & Engg., IIT Delhi

Error Detection

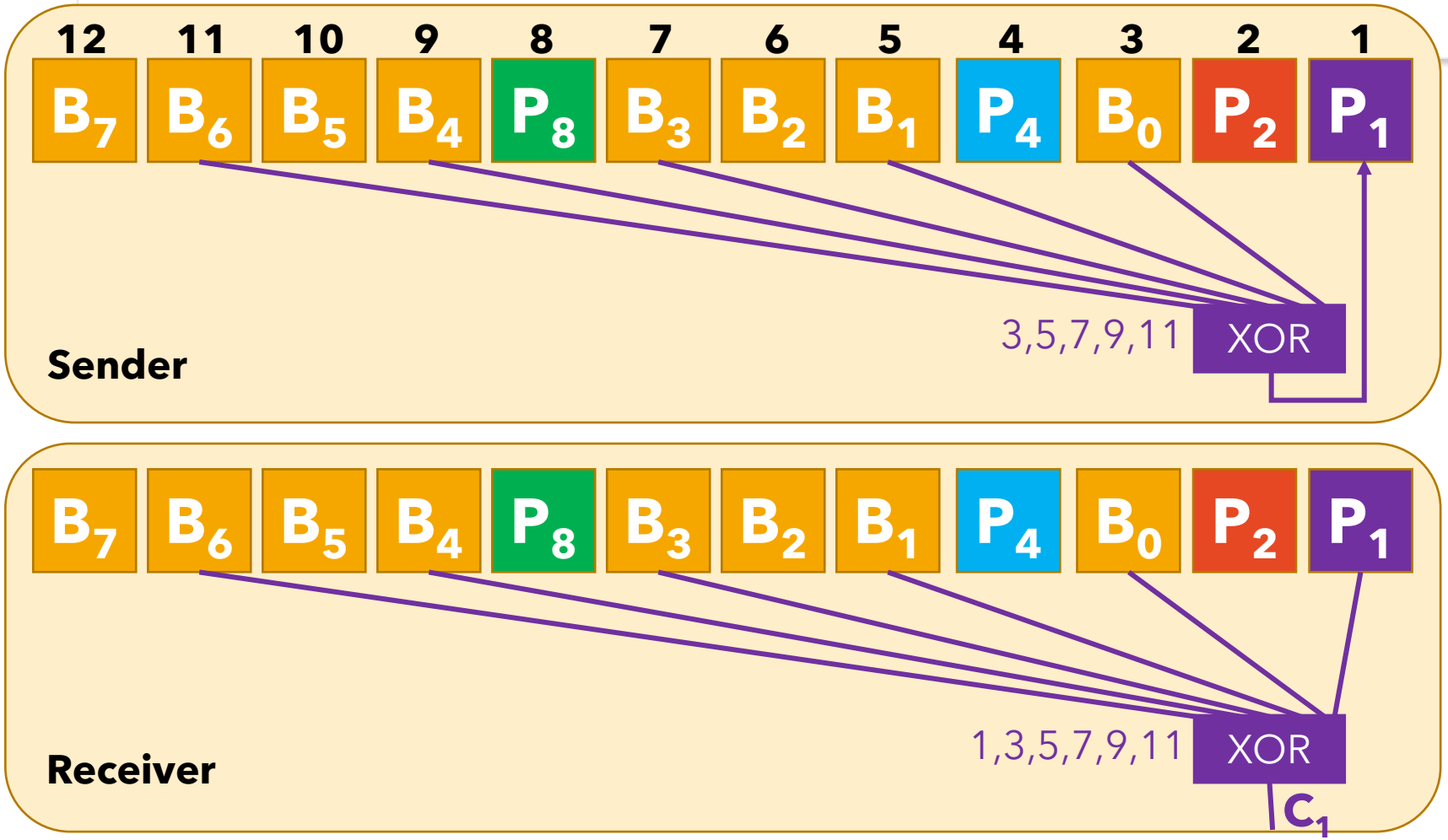
- Errors might occur in storage and transmission
- Error Detection: **Parity Bit**
- Parity: Additional bit stored along with data
- Parity **re-computed** by receiver, **compared** with stored parity
- If different, **error**



Error Correction with Hamming Codes



Error Correction with Hamming Codes

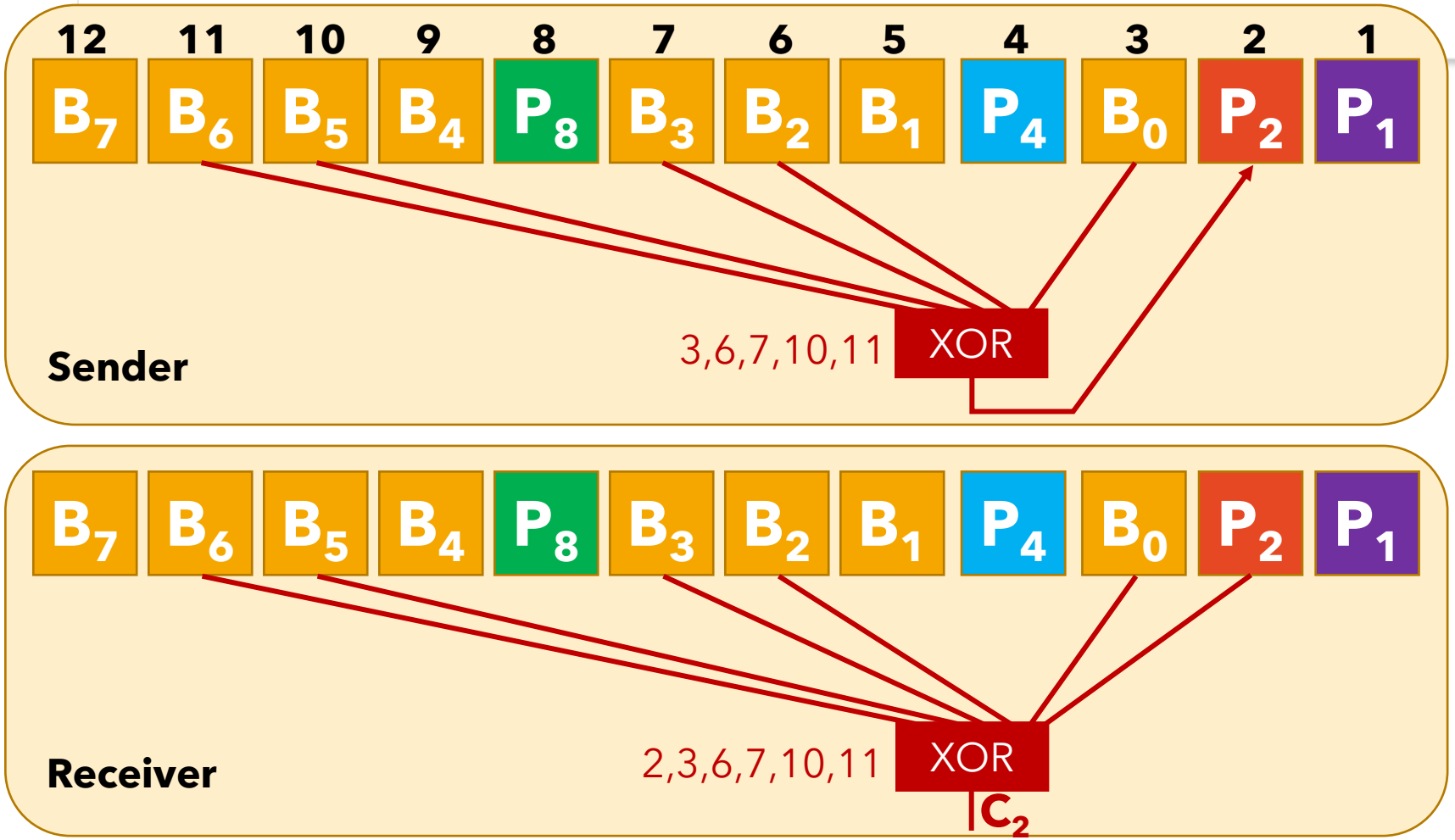


$C_1 = ?$

0 if bits 1,3,5,7,9,11 **OK**
1 if **error** on any of bits 1,3,5,7,9,11

Including Parity Bit
Error type: single bit flip

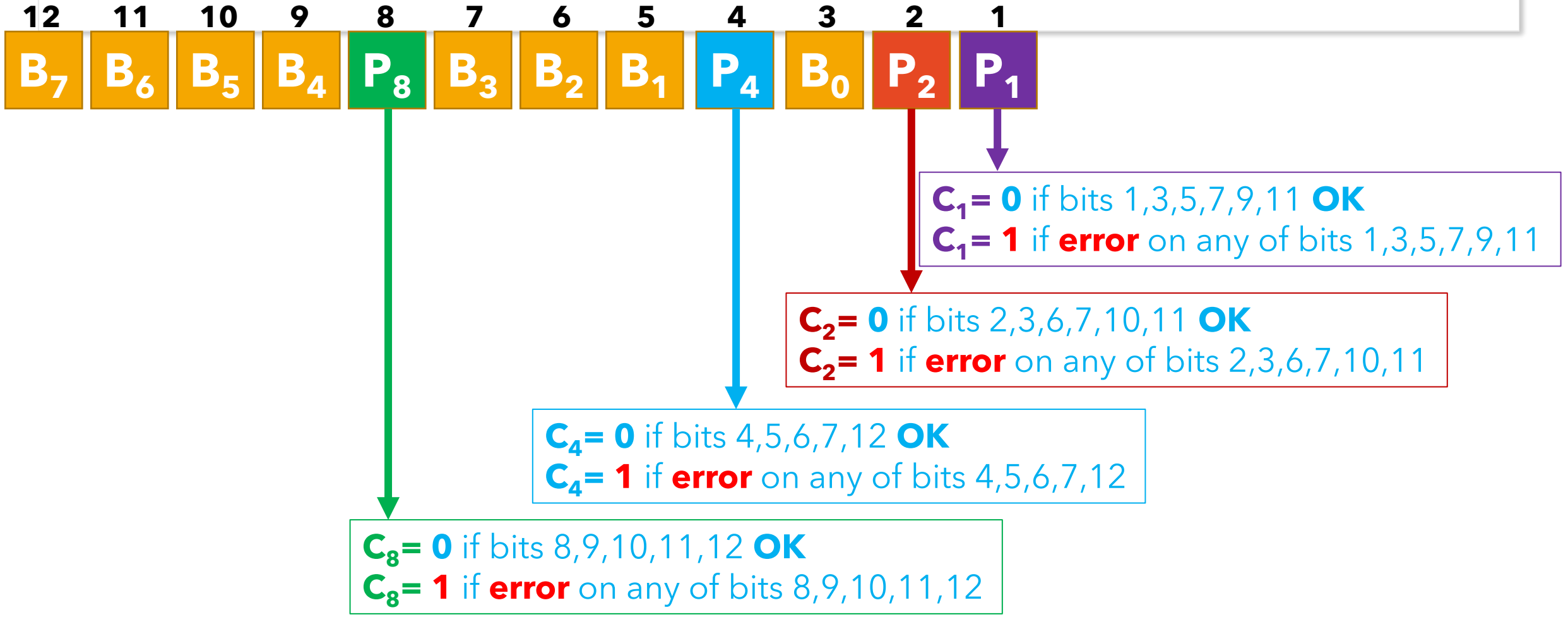
Error Correction with Hamming Codes



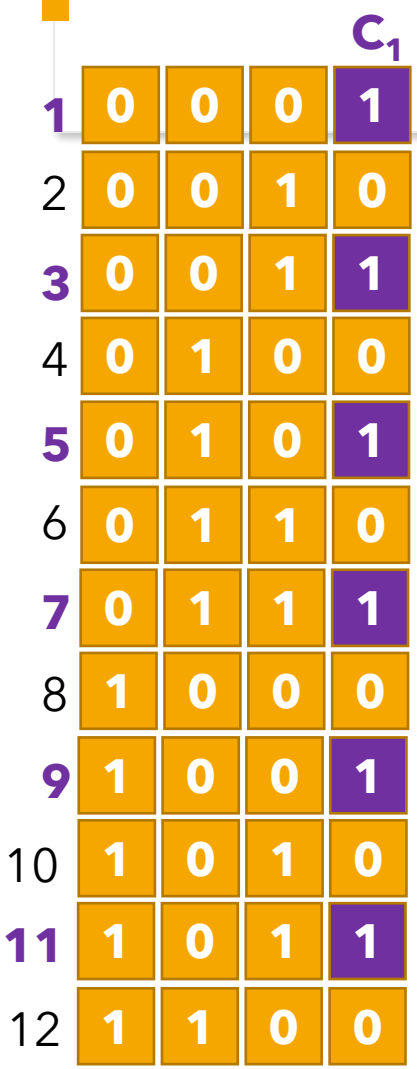
C_2= ?

0 if bits 2,3,6,7,10,11 **OK**
1 if **error** on any of bits 2,3,6,7,10,11

Error Correction with Hamming Codes



Error Correction with Hamming Codes

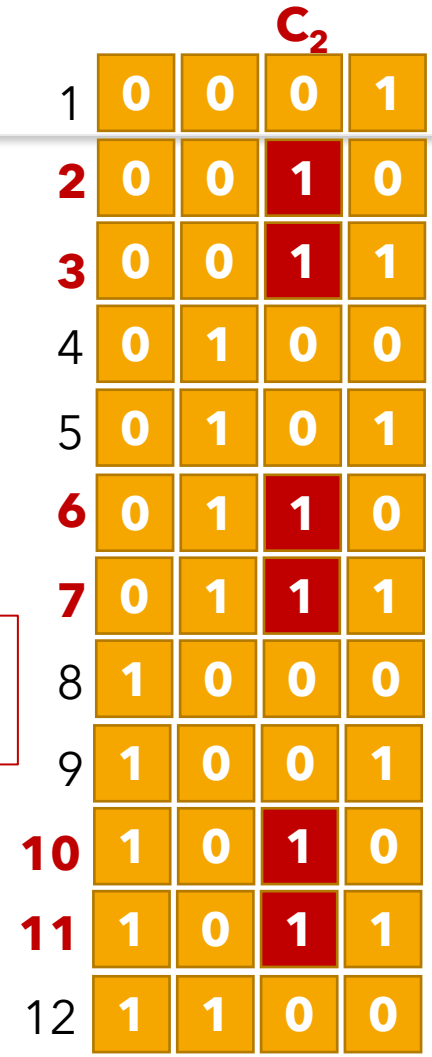


C_1= **0** if bits 1,3,5,7,9,11 **OK**
C_1= **1** if **error** on any of bits 1,3,5,7,9,11

C_1 gives Bit 1 of the erroneous bit position

C_2= **0** if bits 2,3,6,7,10,11 **OK**
C_2= **1** if **error** on any of bits 2,3,6,7,10,11

C_2 gives Bit 2 of the erroneous bit position



Error Correction with Hamming Codes

	C_4			
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0

C_4= 0 if bits 4,5,6,7,12 **OK**
C_4= 1 if **error** on any of bits 4,5,6,7,12

C_4 gives Bit 3 of the erroneous bit position

C_8= 0 if bits 8,9,10,11,12 **OK**
C_8= 1 if **error** on any of bits 8,9,10,11,12

C_8 gives Bit 4 of the erroneous bit position

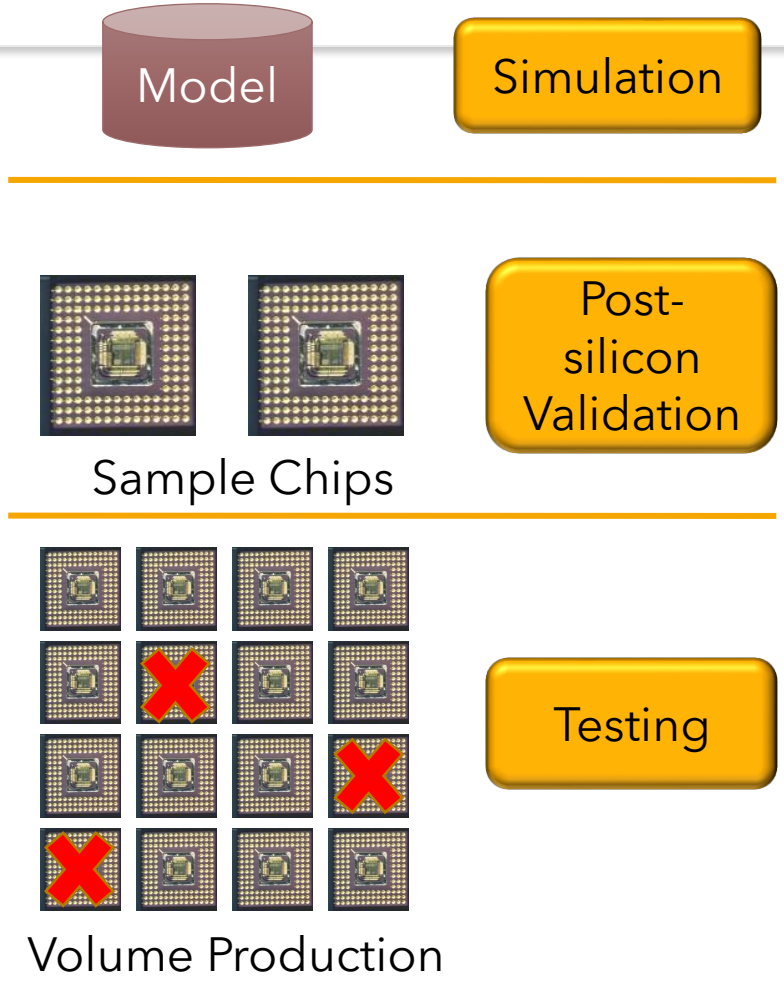
$C_8C_4C_2C_1$ together give the position of the erroneous bit!
$C_8C_4C_2C_1$ = 0 means no error

Now we can perform Error Correction (flip the erroneous bit)

	C_8			
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0

Chip Verification Phases

- Simulation
  - On software model of chip
  - Different abstraction levels
  - Slow: small sub-system tests
- Post-silicon Validation
  - Validating sample chips in lab
  - Logical and Electrical Problems
  - Fast: real usage scenarios
- Testing
  - Manufacturing failures
  - Fast, but not much time per chip
  - Good/Bad decisions only

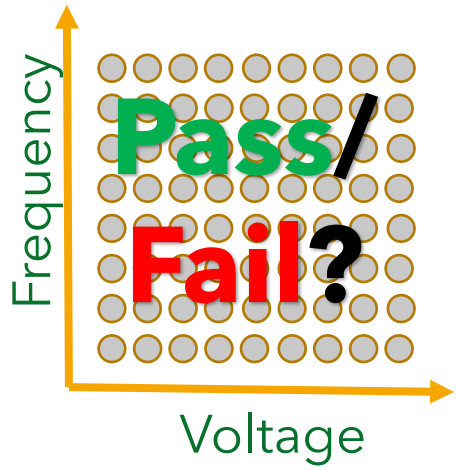


What if Bug is Found?

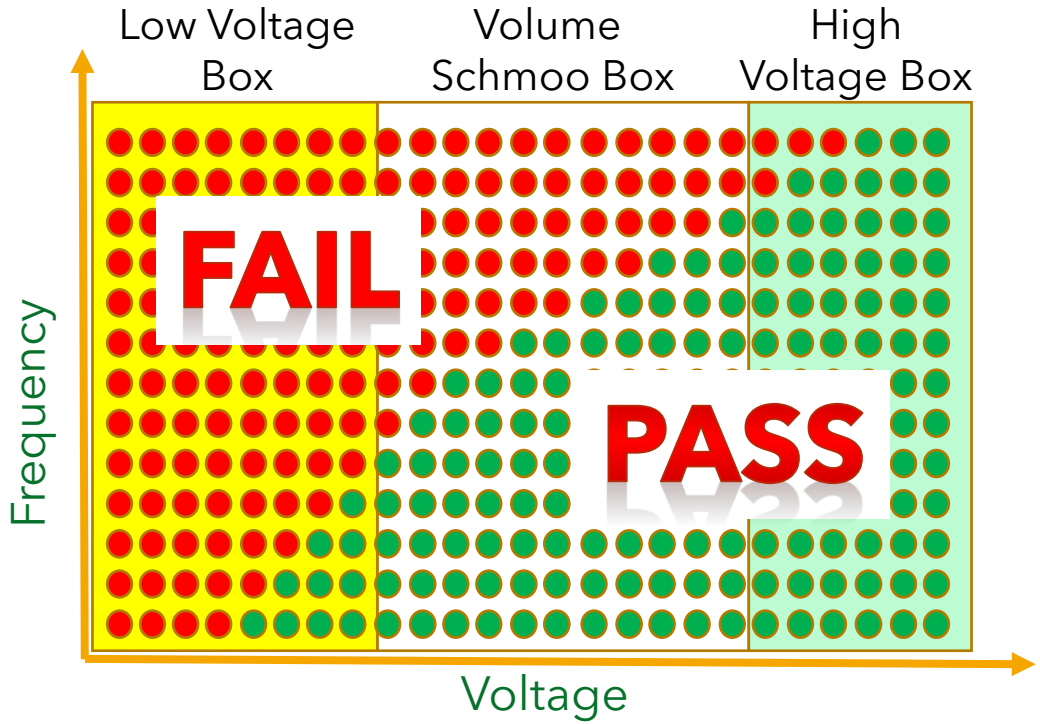
- If SEVERE (doesn't allow other validation to proceed)
  - RE-SPIN (Expensive!)
- If workaround exists (in software, etc.)
  - Proceed with other validation
- New revision only if certain threshold of bugs is crossed

Early Sanity Checks

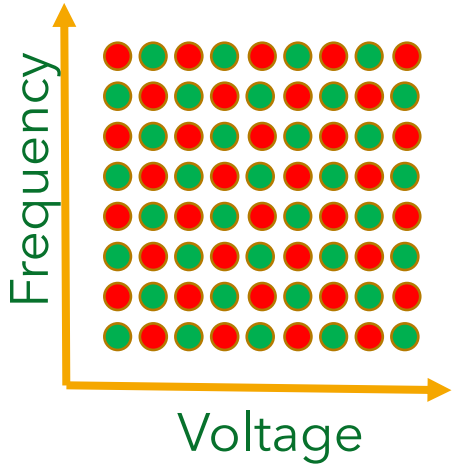
- Heartbeat (Iddq) Testing: is the chip alive?
- Schmoo Plot



Pass/Fail response under varying conditions



Normal Operation



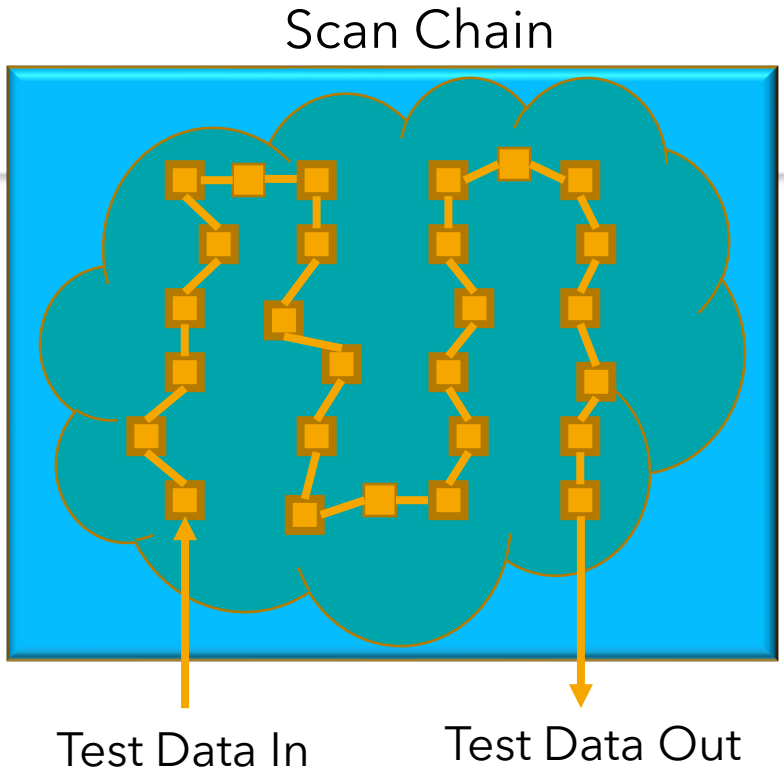
"Brick Wall" INITIALISATION PROBLEM

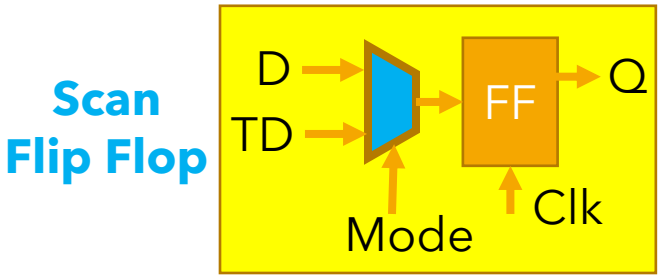
Analysing and Fixing Failures

- Direct observation (**Expensive**)
  - Laser Voltage Probing (LVP)
    - observe reflected laser from transistor's diffusion layer
    - tells whether voltage transition occurred or not
    - signal waveform can be generated
- Fixing (**Expensive**)
  - Laser Assisted Device Alteration (LADA)
    - characteristics of individual transistors can be altered
  - Focused Ion Beam (FIB) Edit
    - Create shorts/opens (destructive)
    - Important debug tool – permits validation to continue in presence of serious bugs

How to test a chip?

- Examine internal memory (too large!)
- Collect all important memory elements (**flip flops**)
- Hook them together into a long chain
- Replace normal Flip Flop by **Scan Flip Flop**
  - Design-for-Test feature
  - Note: we are changing the original design!

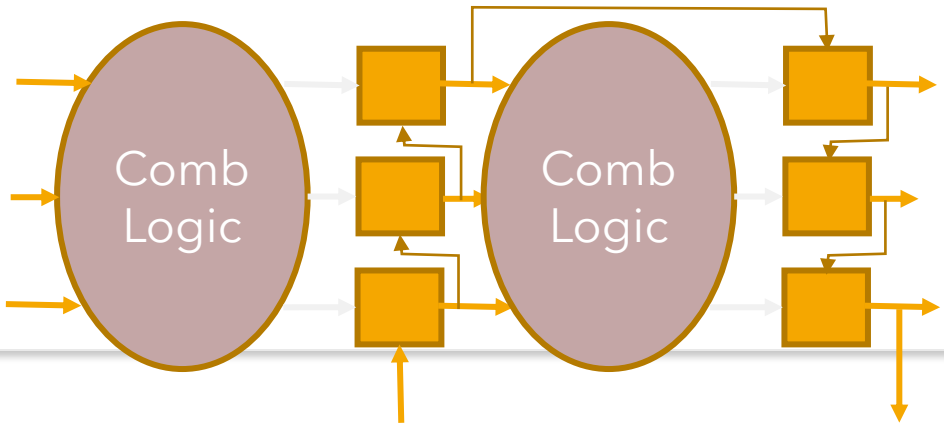




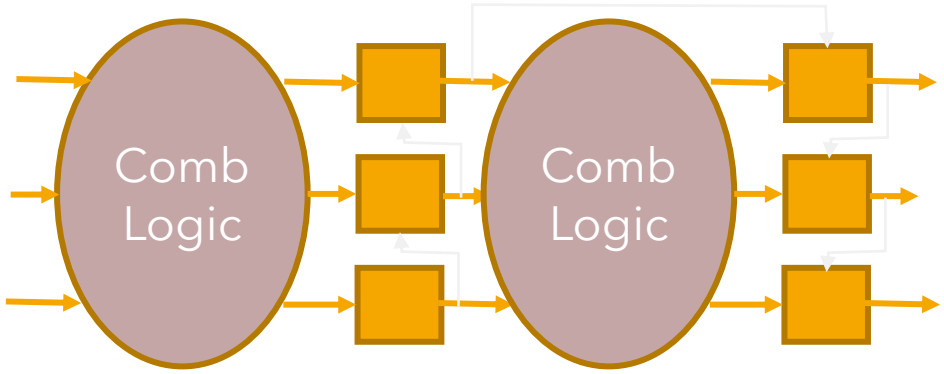
Scan Chain

- NORMAL mode:
  - Disable chain
  - Execute
- TEST mode:
  - Enable chain
  - Send data IN/OUT

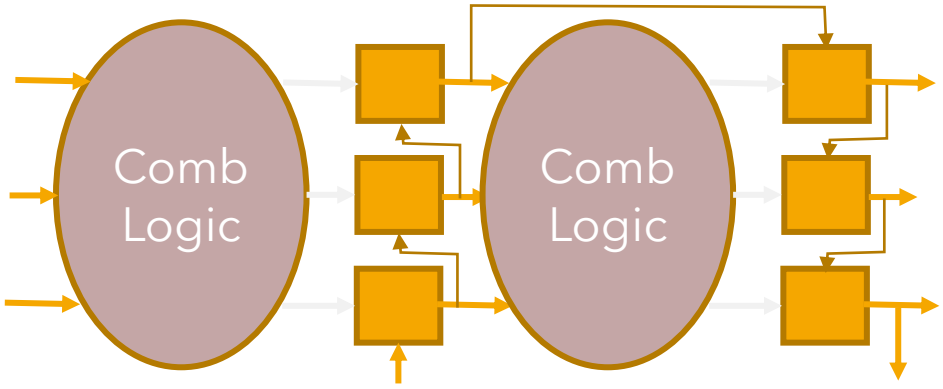
TEST: Scan In (initialise)



NORMAL Execution

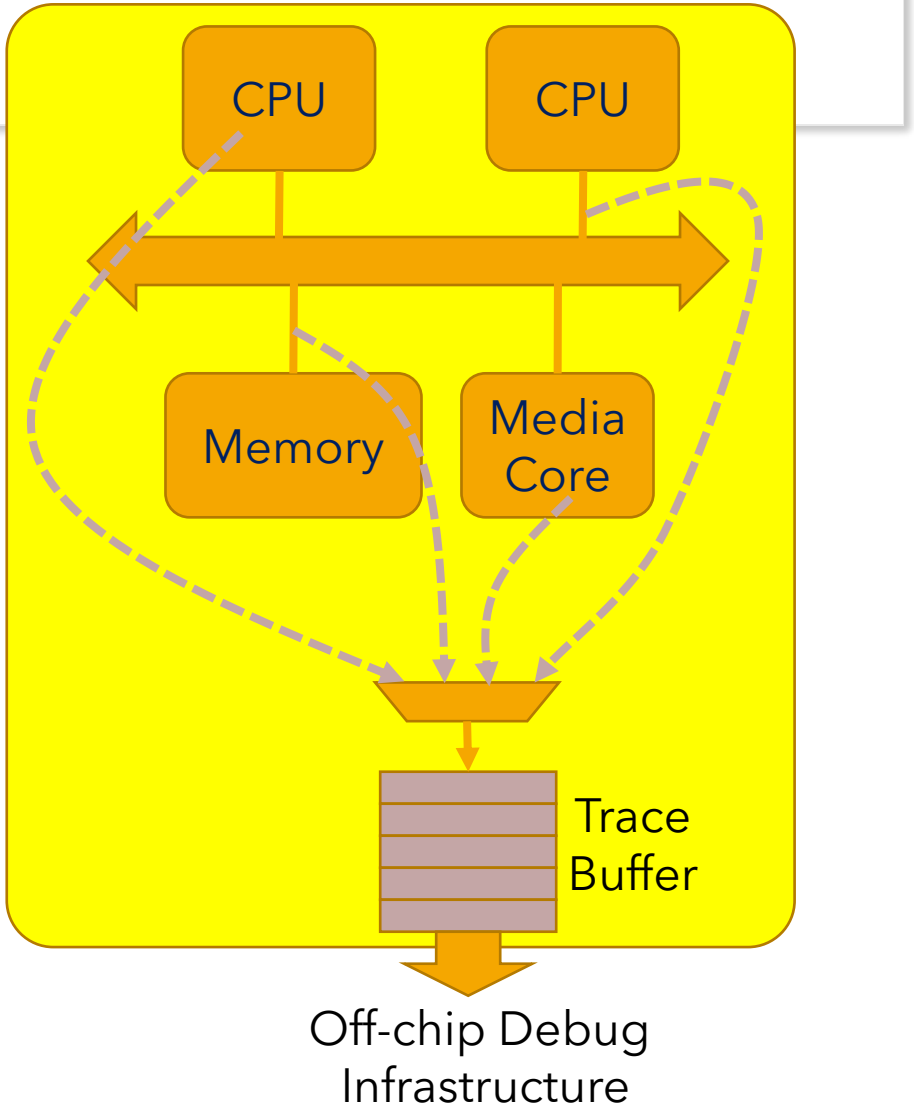


TEST: Scan Out



Generalising the Scan Chain: Trace Buffers

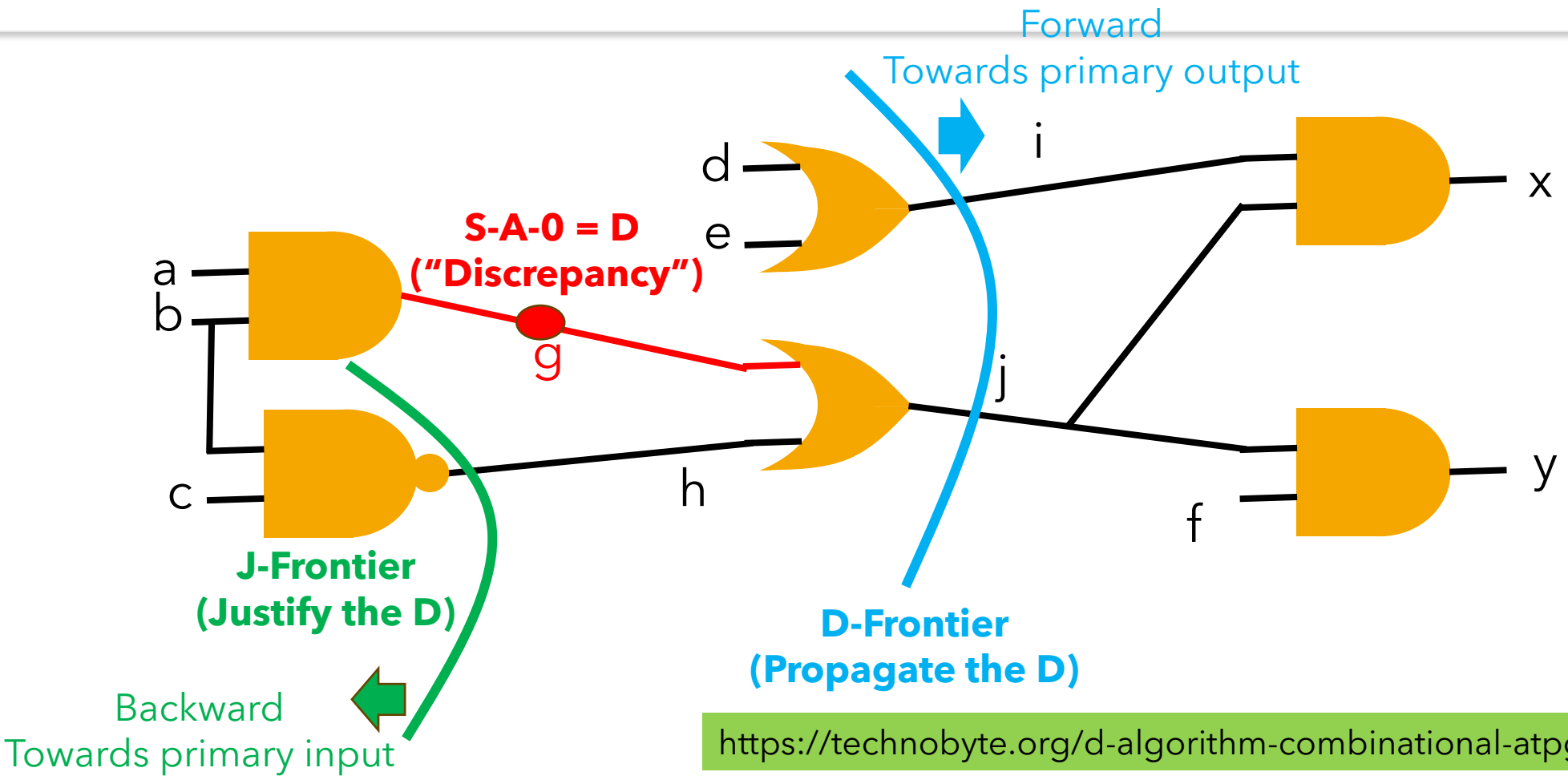
- Scan Chain OK in principle, but slow
  - Overwhelming data volume
- Need dedicated storage: Trace Buffers
  - FIFO structures
  - Store critical signal values



Classical Test/Debug Problems

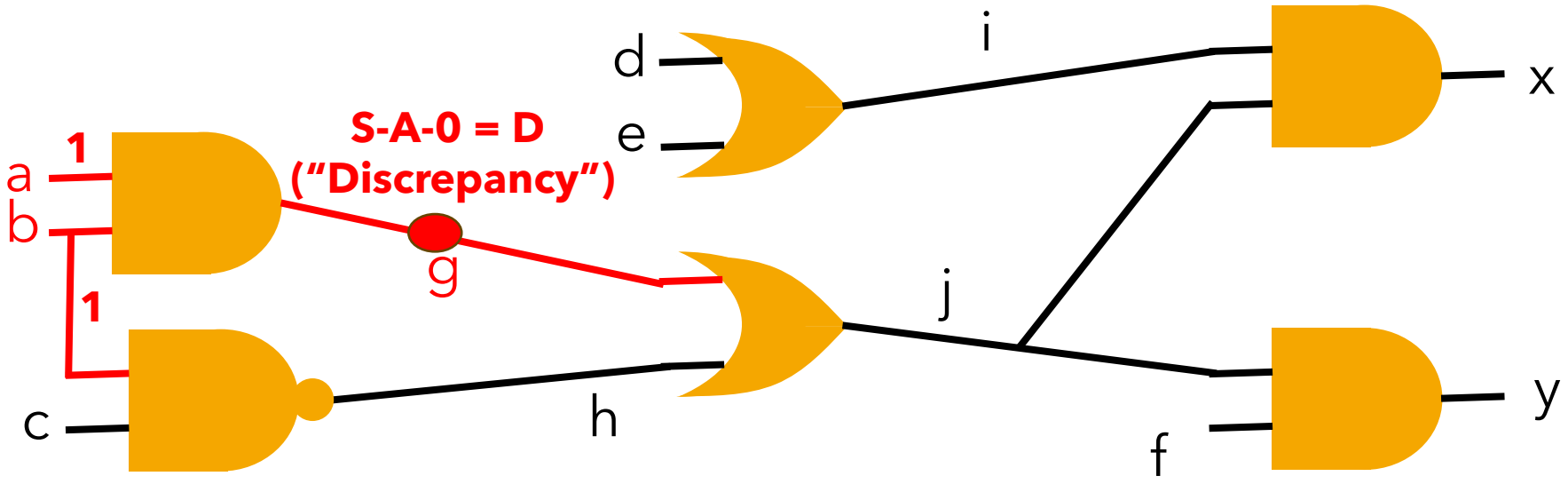
- Automatic test pattern generation (ATPG)
  - Minimise time spent in testing
- Fault modelling and test generation
  - Modelling (“stuck-at-0”) of manufacturing faults (short circuit)
  - Determine test pattern to detect fault
- Signal Tracing: Which signals to trace?

Combinational ATPG: D Algorithm

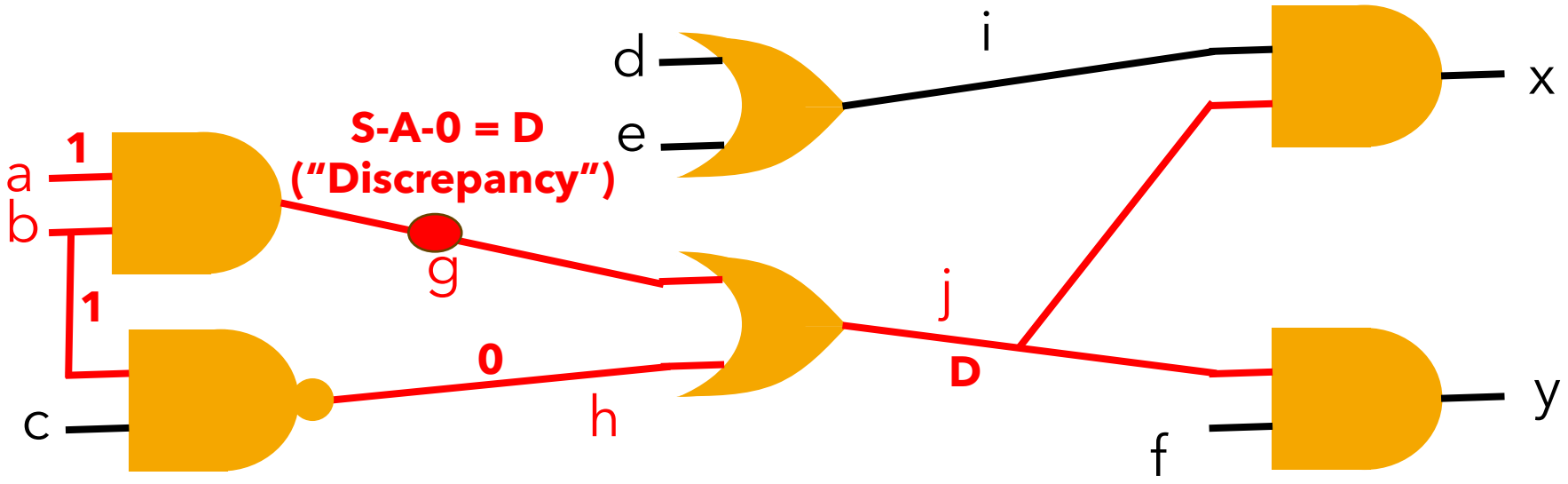


https://technobyte.org/d-algorithm-combinational-atpg-in-dft-vlsi/

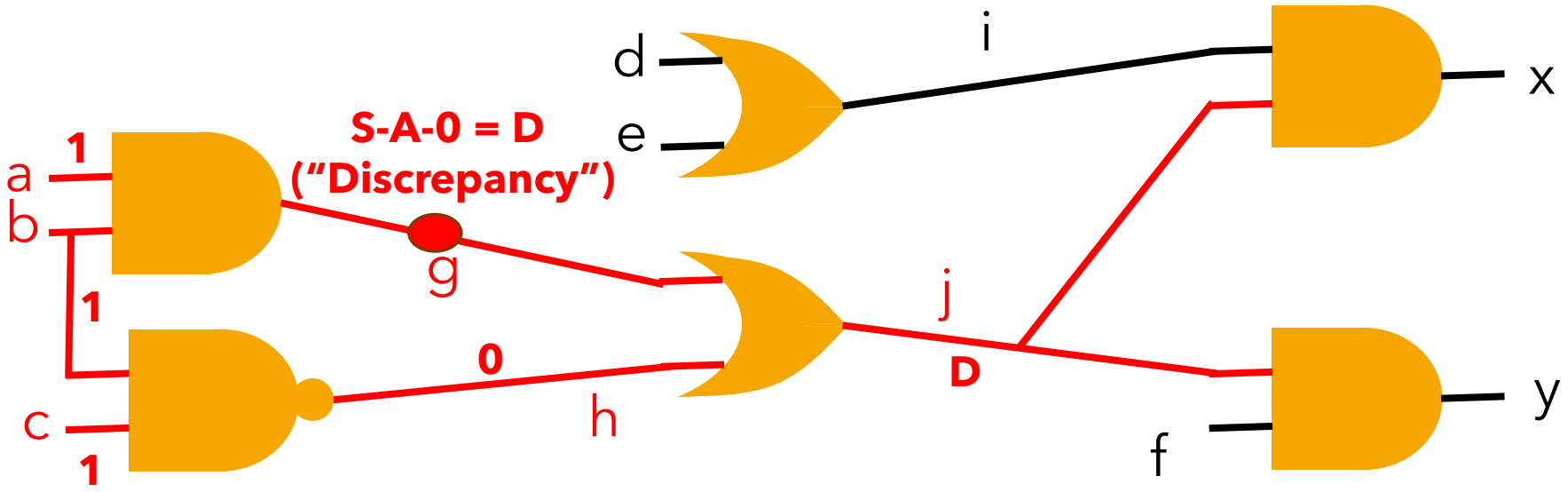
Combinational ATPG: D Algorithm



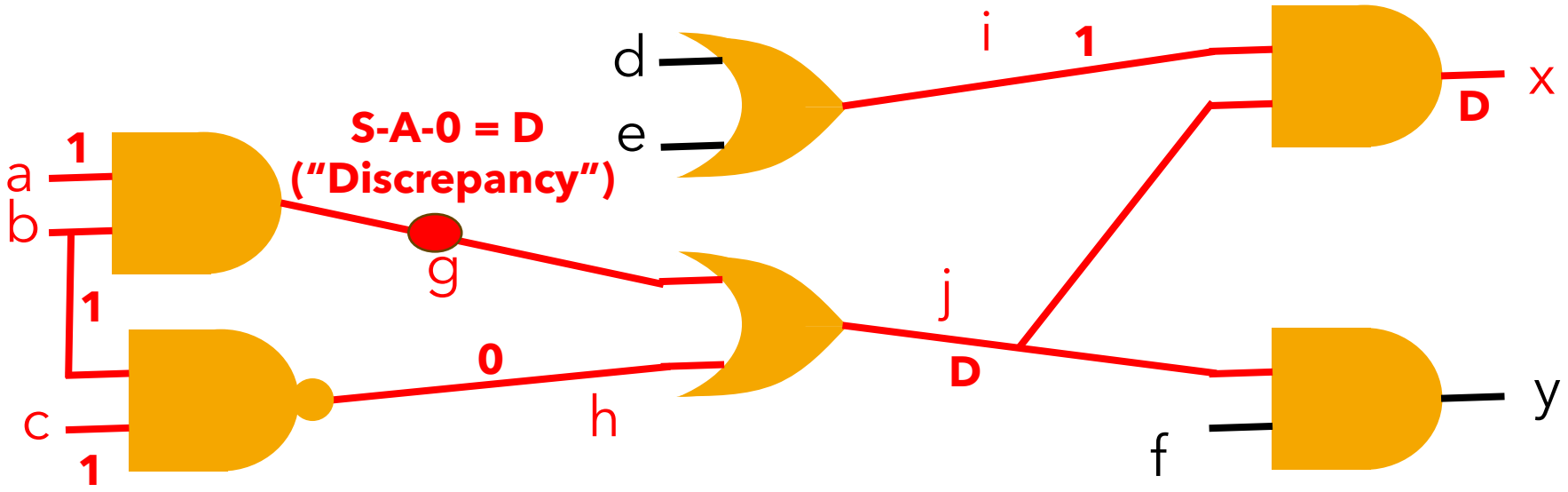
Combinational ATPG: D Algorithm



Combinational ATPG: D Algorithm



Combinational ATPG: D Algorithm



Combinational ATPG: D Algorithm