Diagnosis for Interconnect Faults in Memory-based Reconfigurable Logic Device

Xihong Zhou, Senling Wang, Yoshinobu Higami, Hiroshi Takahashi

Department of Computer Science,

Ehime University

Nov. 25, 2021

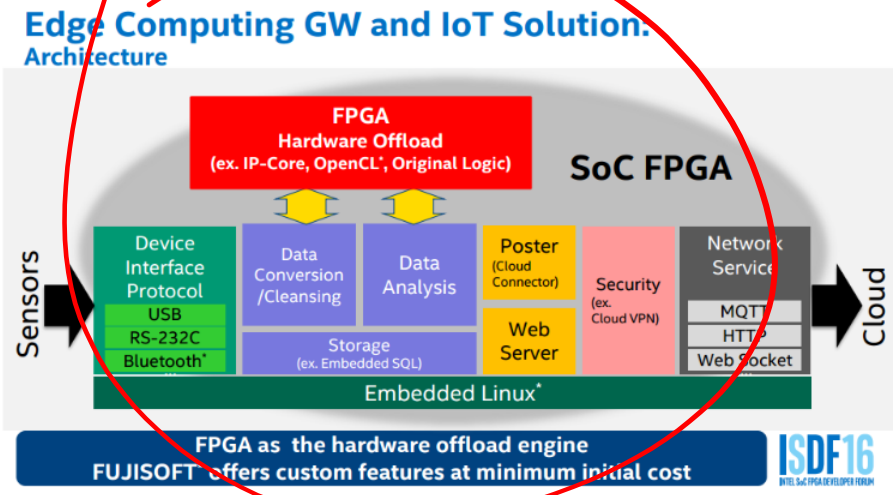
Outline

- Background
  - The application of Reconfigurable devices
  - FPGA and its challenge
  - MRLD and its chance
- Purpose
- Architecture and Working principle of MRLD
- Interconnect fault models in MRLD
  - Stuck-at faults
- Diagnosing the interconnect faults of MRLD
- Experimental results
- Conclusions

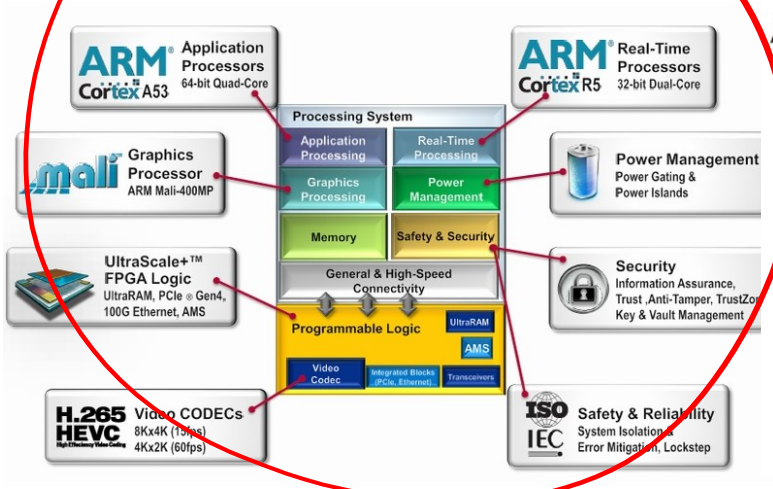
Background ~ The application of Reconfigurable devices ~

- **Reconfigurable devices** (e.g.: FPGAs) are gaining increased attentions for IoT, Automotive and AI field
  - ★ Flexibility and scalability
  - ★ High performance (parallel computing)
  - ★ Better time to market
  - ★ Low design cost (shortening of development cycle)

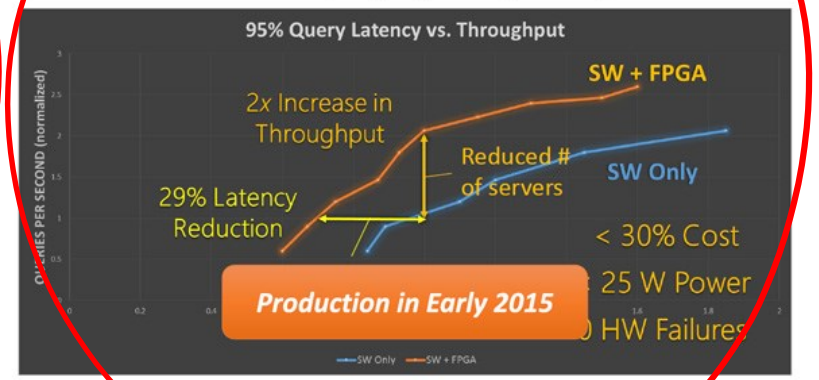
IoT Edge Computing



Xilinx's Automotive Solution: Zynq UltraScale＋MPSoC



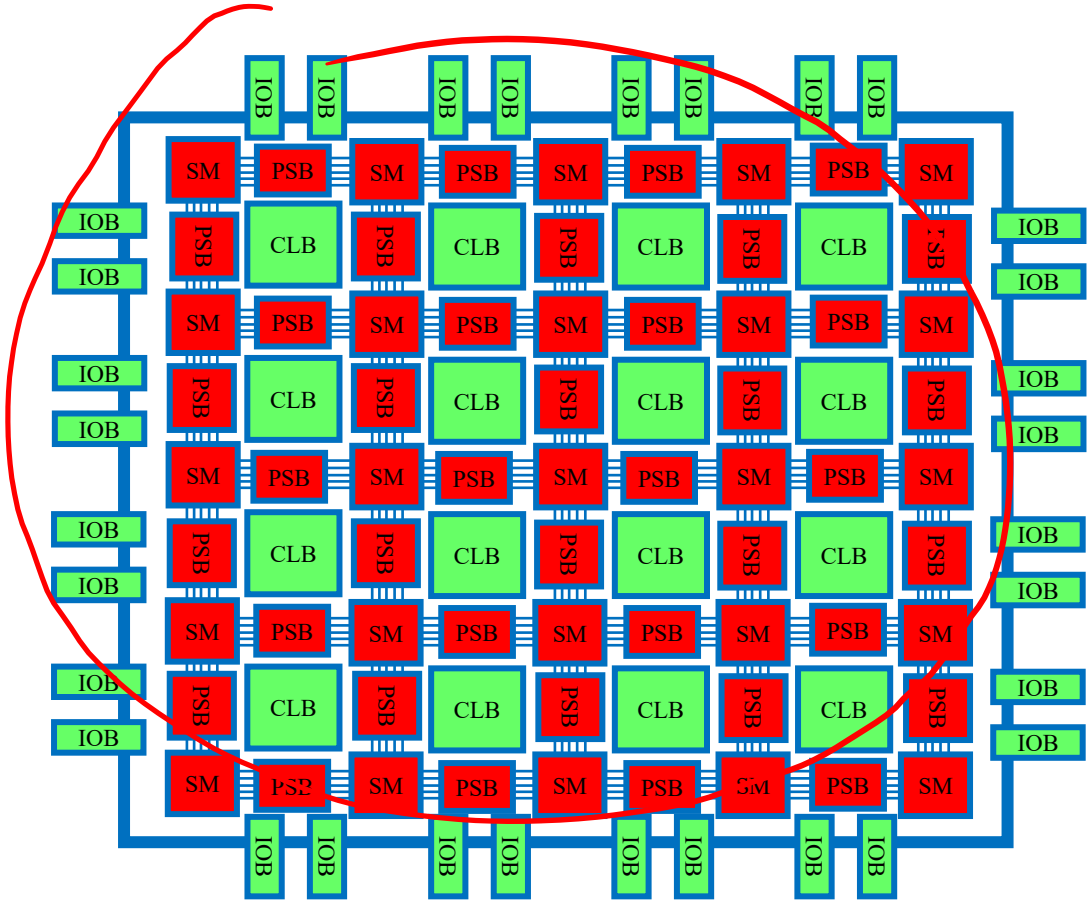
Bing Intelligent Search Engine FPGA accelerator



Background ~ FPGA and its challenge ~

- Three types of configurable elements
  - Input/output blocks (IOBs)
  - Configurable logic blocks (CLBs)
  - Programmable interconnect resources (SM: switch matrix, PSB: programmable switch blocks)

- Large amount of interconnect resource
  - ➢ Large area
  - ➢ Large delay
  - ➢ High power
  - ➢ Significant production cost



Overhead compared to ASIC

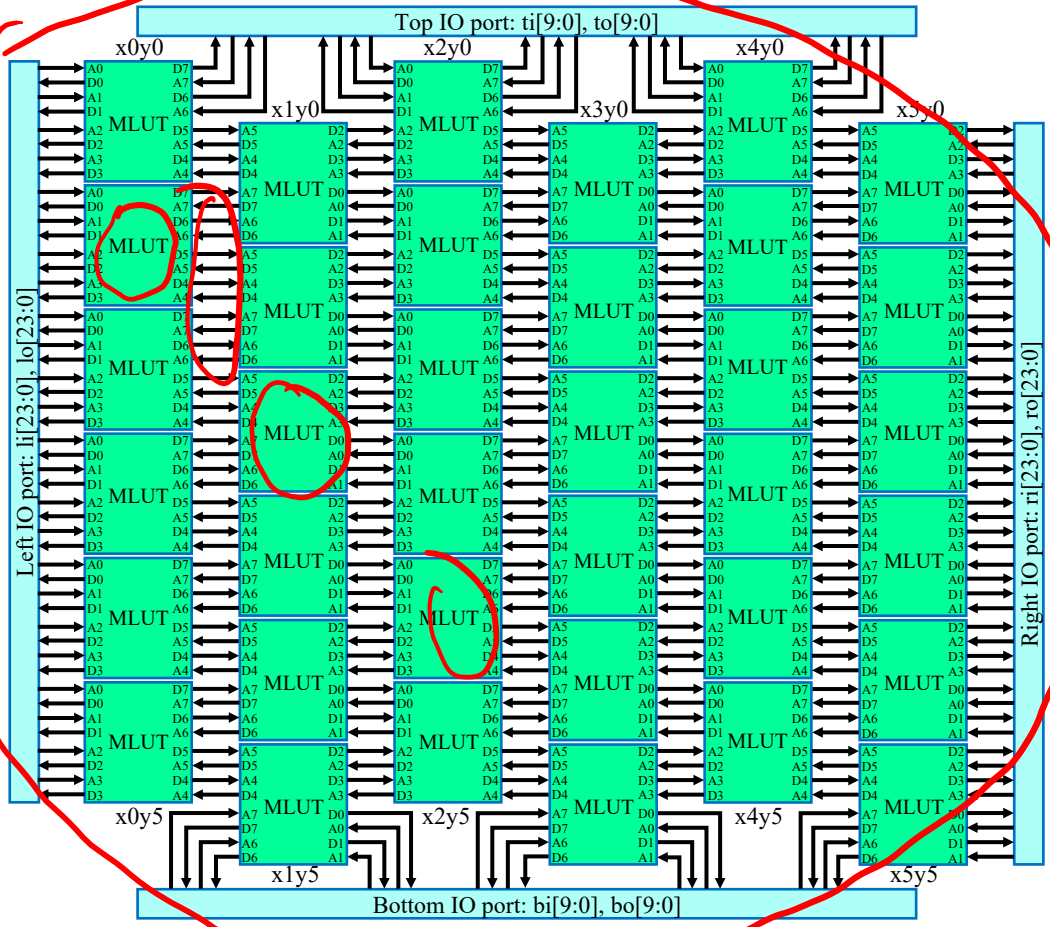
	Area	Delay	Power
ASIC	1	1	1
FPGA	20~35 or more	3~4	~10
Programmable interconnect resources	90%	40~80%	60~70%

Alternative reconfigurable devices with low cost, low power and small delay is required

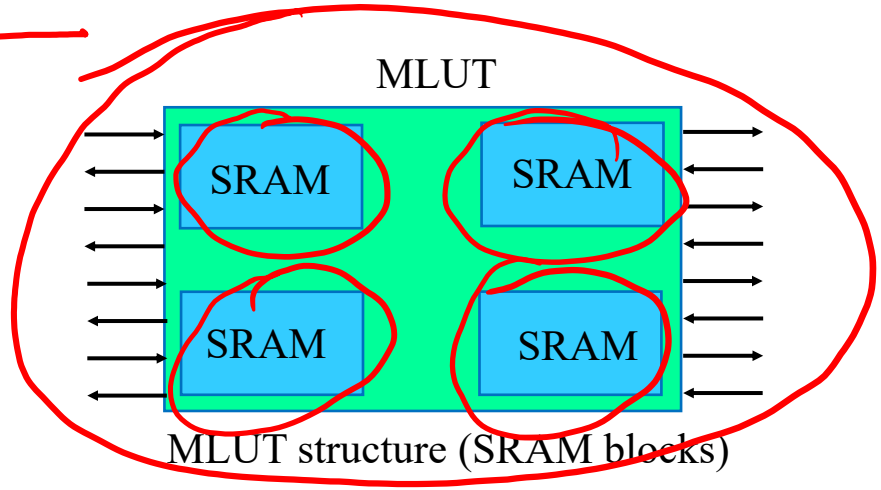
Background ~ MRLD and its chance~

- **M**emory-based **R**econfigurable **L**ogic **D**evice.
  - MLUTs (Multiple Look-Up-Table) array
  - MLUT configured with multiple SRAM blocks
  - Alternate interconnect of Address and Data line of MLUTs
  - Support Memory mode and Logic reconfiguration mode

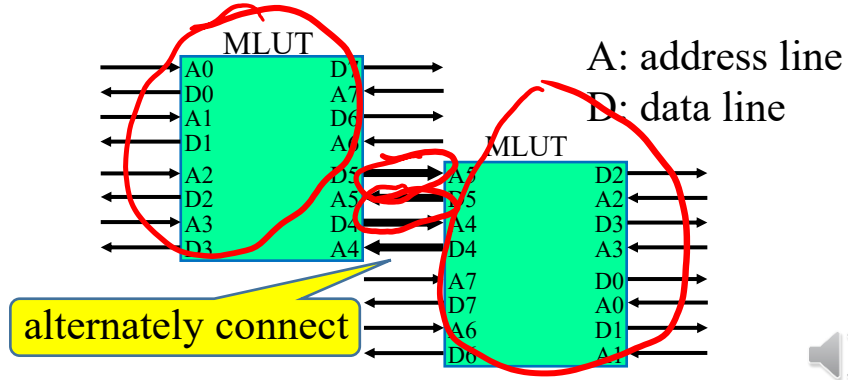
- Logic and wiring are directly configured in the MLUTs
  - Lower area overhead
  - Smaller delay (logic wiring)
  - Low power
  - Low production cost



The architecture of MRLD



MLUT structure (SRAM blocks)

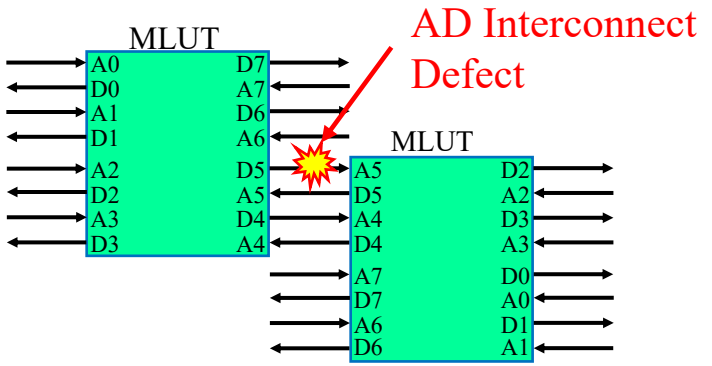


Interconnect Structure of MRLD

Motivation & Purpose & Objective

- To improve the yield and reliability of MRLD.
  - ➢ Detecting and locating the defects on AD interconnects

(AD: Address line & Data line)



- Purpose

Develop the diagnosis approaches for identifying the location of AD interconnect defects

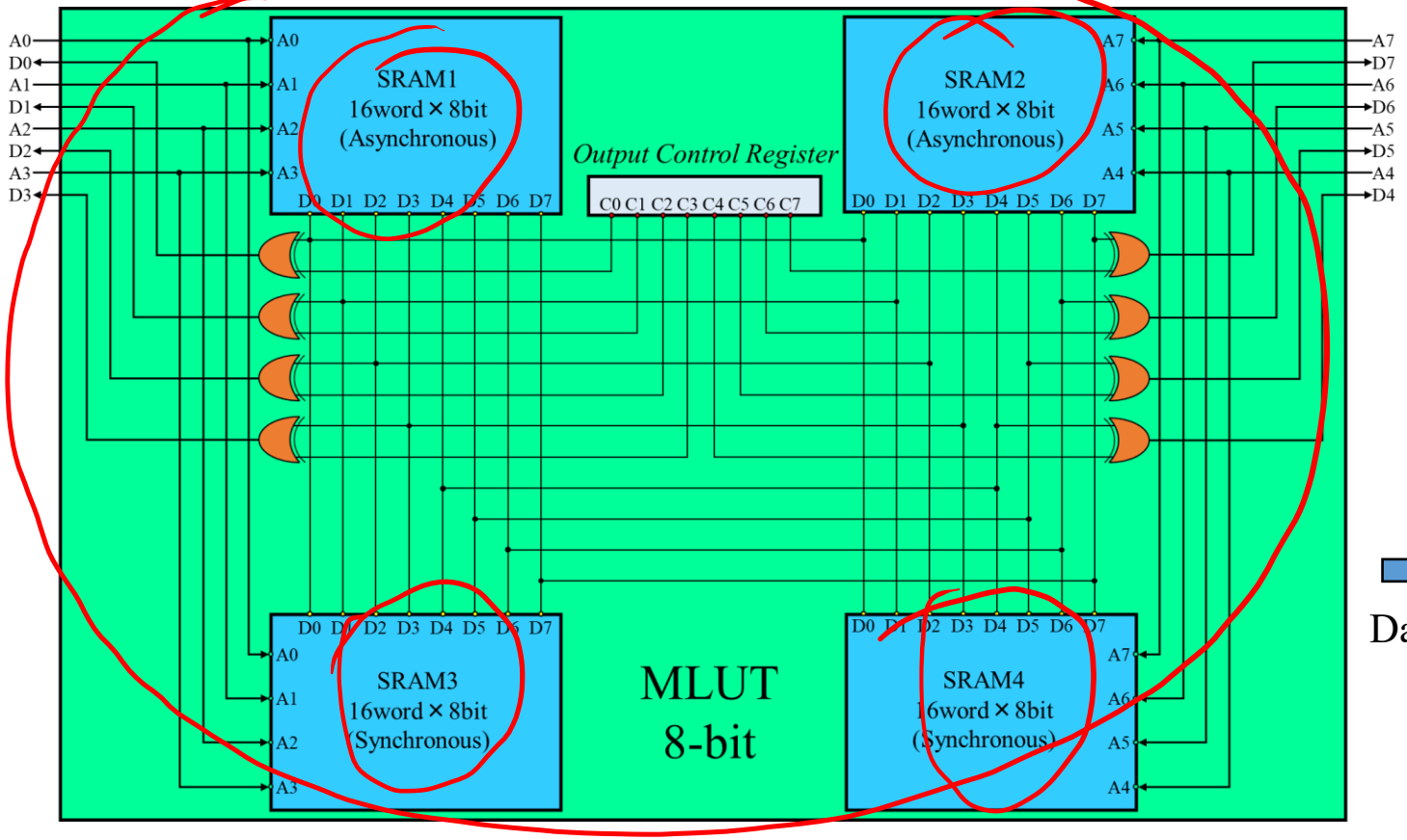
- Objective

1. **Propose the diagnosis strategy for interconnect defects of MRLD**
2. **Propose the diagnostic generation method for Stuck-at faults**

∗ Detection approaches have been proposed in our previous research in ATS2017

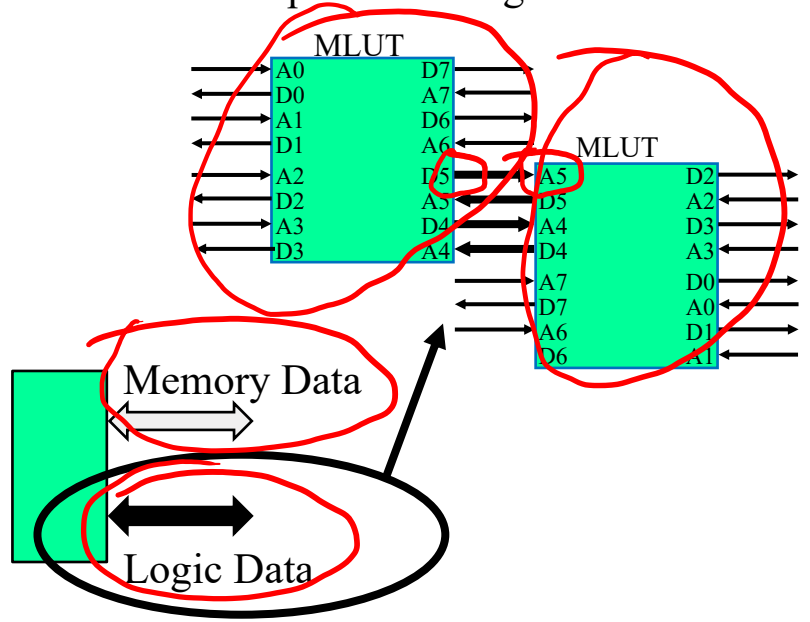
Architecture & Working principle of MRLD

- MLUT consists of four SRAM blocks (two asynchronous and two synchronous SRAMs)
- Each SRAM works as look-up tables (LUTs) to support logic reconfiguration by writing the corresponding truth tables into the SRAM
- Each MLUT can work at either Memory mode or Logic reconfiguration mode
- The data outputs of a MLUT are connected with the address inputs of other MLUTs



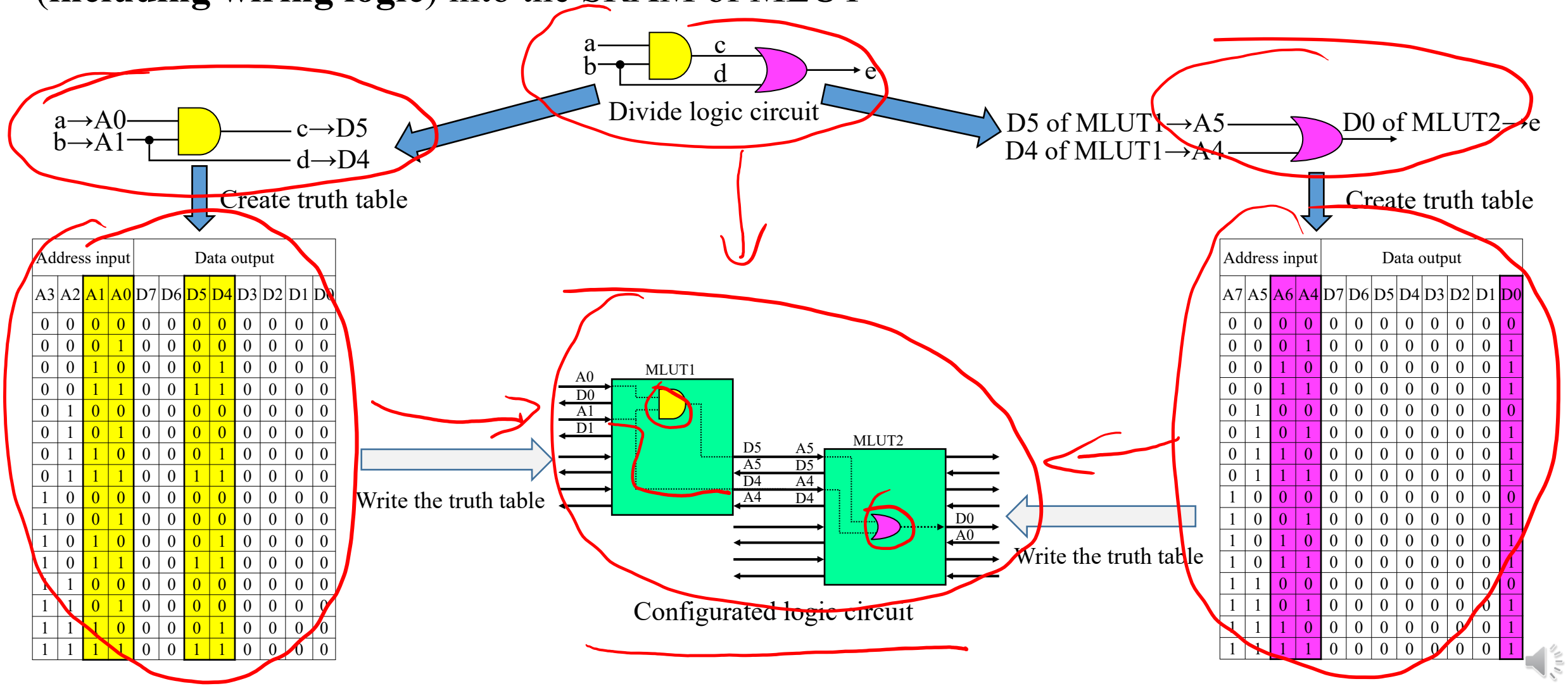
Data of MLUT

Logic output of a MLUT connects with the Address input of its neighbor MLUTs



Working principle of MRLD ~ an example ~

- Configure the logic circuit by writing the **truth table** of the logic circuit (**including wiring logic**) into the SRAM of MLUT

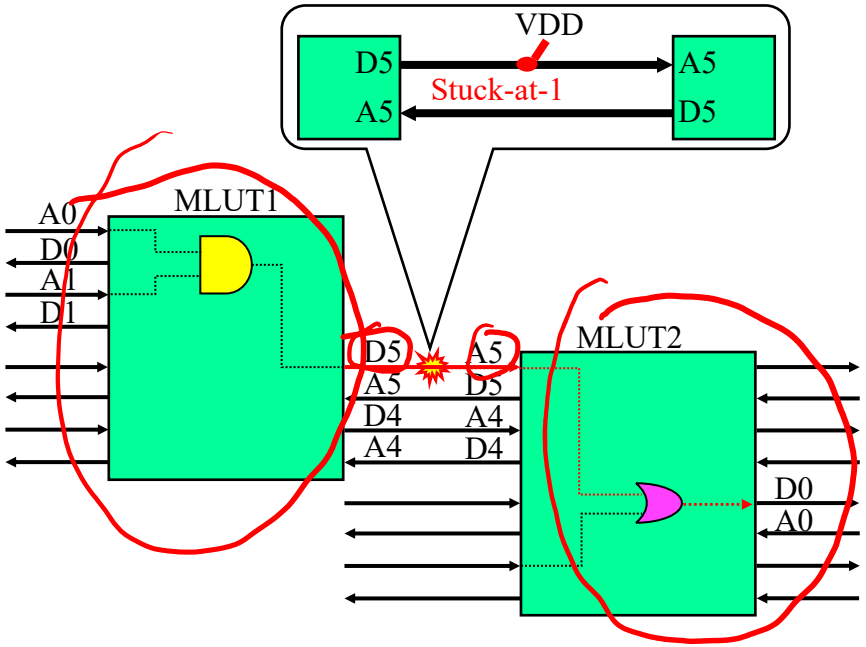


Address input				Data output							
A3	A2	A1	A0	D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	1	0	0	0	0
0	0	1	1	0	0	1	1	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0	0
0	1	0	1	0	0	0	0	0	0	0	0
0	1	1	0	0	0	0	1	0	0	0	0
0	1	1	1	0	0	1	1	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	1	0	0	0	0
1	0	1	1	0	0	1	1	0	0	0	0
1	1	0	0	0	0	0	0	0	0	0	0
1	1	0	1	0	0	0	0	0	0	0	0
1	1	1	0	0	0	0	1	0	0	0	0
1	1	1	1	0	0	1	1	0	0	0	0

Address input				Data output							
A7	A5	A6	A4	D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	0	0	1
0	0	1	1	0	0	0	0	0	0	0	1
0	1	0	0	0	0	0	0	0	0	0	0
0	1	0	1	0	0	0	0	0	0	0	1
0	1	1	0	0	0	0	0	0	0	0	1
0	1	1	1	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0	0	1
1	0	1	0	0	0	0	0	0	0	0	1
1	0	1	1	0	0	0	0	0	0	0	1
1	1	0	0	0	0	0	0	0	0	0	0
1	1	0	1	0	0	0	0	0	0	0	1
1	1	1	0	0	0	0	0	0	0	0	1
1	1	1	1	0	0	0	0	0	0	0	1

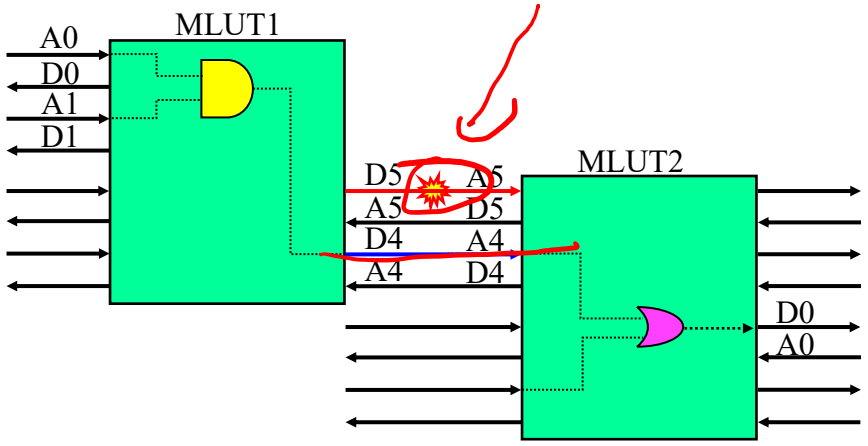
Interconnect fault models in MRLD ~Stuck-at~

- A short between the ground (supply) and AD interconnect (address line or data line)



MLUT1_D5	MLUT2_A5	behavior
0	**0/1**	MLUT2_D0/1
0	**0/1**	MLUT2_D0/1
1	1	1
1	1	1

Logic behavior of Stuck-at-1 Fault

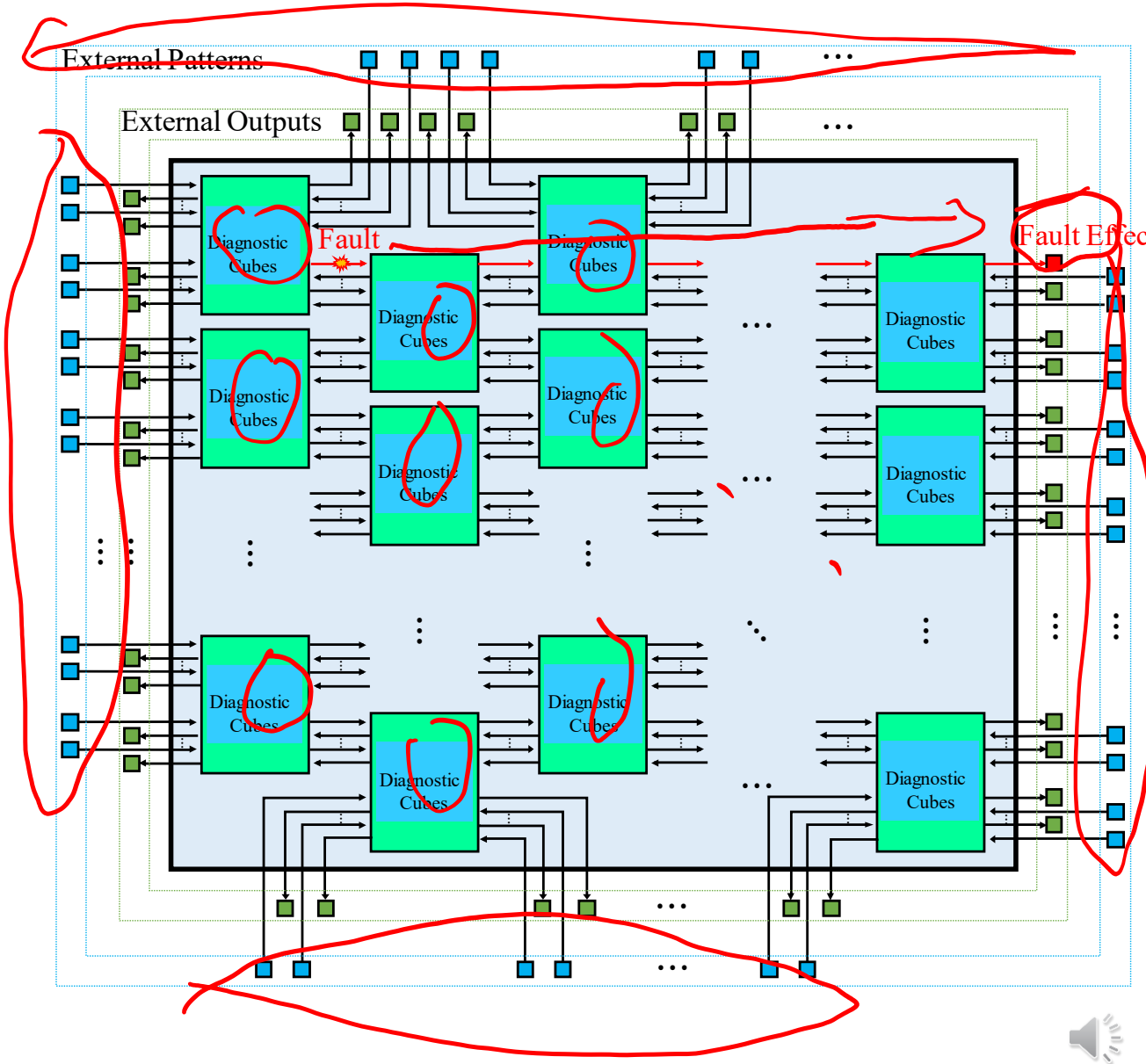


Diagnosing the location of the fault is helpful to avoid configuring the logic to pass through the faulty AD interconnects

Diagnosis Strategy ~ for the interconnect faults of MRLD ~

- **Diagnostic Test Generation**
  - **Diagnostic Cubes**:
    → Data in the SRAMs for Creating fault propagation path on MLUTs
  - **External Patterns**:
    → patterns applied to the external input ports of MRLD for fault excitation

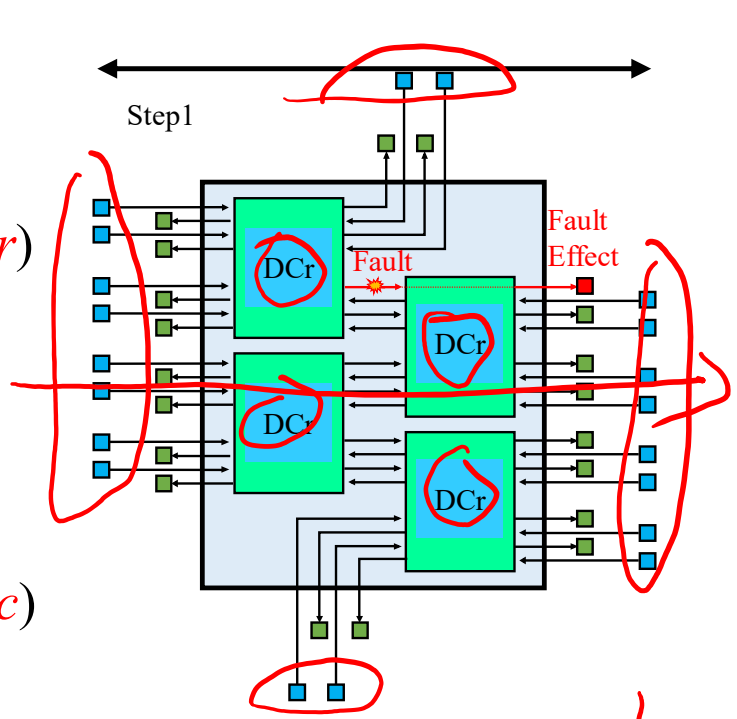
- **Basic principle of diagnosis**
  - **Configuring Fault Propa. Path on Row&Column**:
    → Diagnostic Cubes Reconfiguration
  - **Applying External Patterns**:
    → to external inputs of MRLD
  - **Observing External Outputs**:
    → fault effects can be propagated and observed at the external outputs of MRLD.

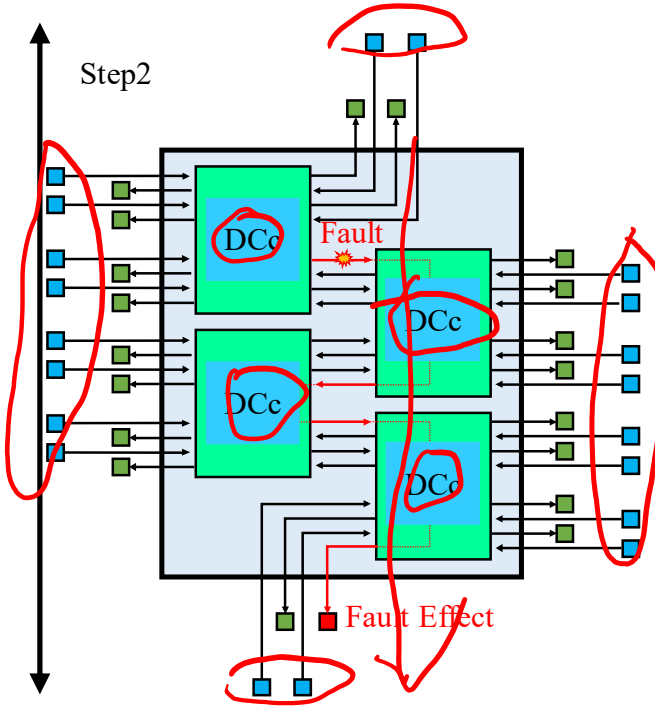


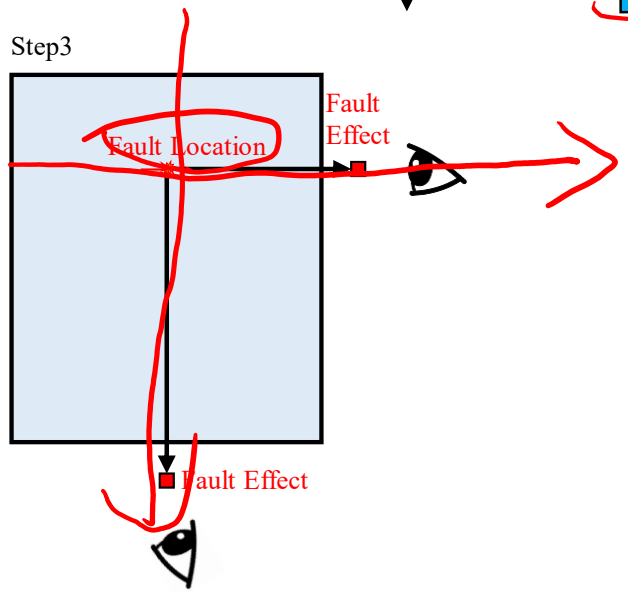
Diagnosis Flow ~ for the interconnect faults of MRLD ~

- **Diagnosis Flow**
  - **Step1: Row-direction diagnosis**
    a) Configuring *Diagnostic Cubes* (*DCr*)
    b) Applying *External Pattern*
    c) Obtaining *Fault Path* (*FPr*)
  - **Step2: Col-direction diagnosis**
    a) Configuring *Diagnostic Cubes* (*DCc*)
    b) Applying *External Pattern*
    c) Obtaining *Fault Path* (*FPc*)
  - **Step3: Determining fault location**

    ⟶ Find out the Fault location (F_{loc}) through computing the intersection of FPr and FPc: $\boldsymbol{F_{loc}}$ = FPr ∩ FPc







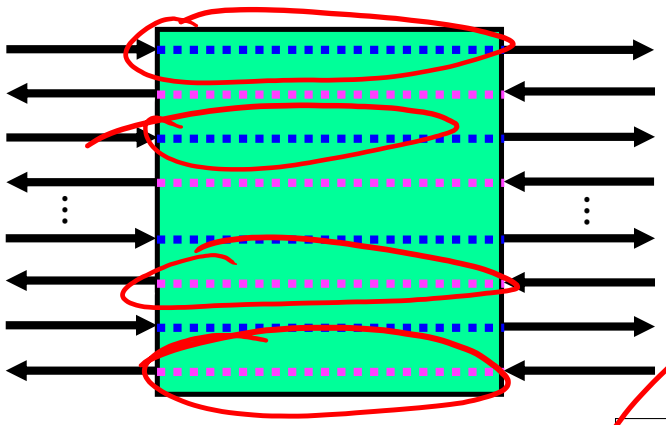
Diagnostic Cubes & External Patterns ~ Row-direction ~

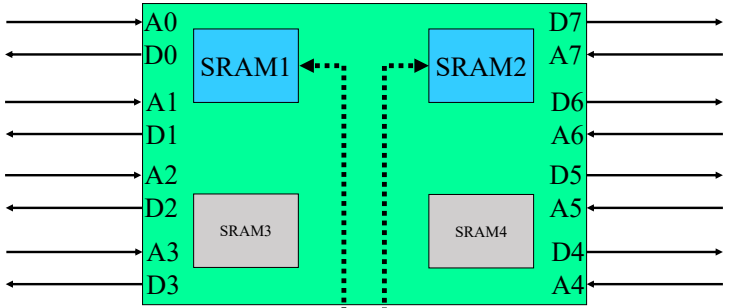
- **Diagnostic Cubes for row-direction (DCr):**

Diagnostic Cube 1: For the SRAMs share the low-order address inputs (A[m/2-1:0]) of MLUT, set contents of the address lines A[m/2-1:0] to D[m-1:m/2]=A[0:m/2-1], D[m/2-1:0]=all-zero.

Diagnostic Cube 2: For the SRAMs share the high-order address inputs (A[m-1:m/2]) of MLUT, set contents of the address lines A[m-1:m/2] to D[m-1:m/2]=all-zero, D[m/2-1:0]=A[m/2:m-1]

- **External Patterns:** *All-zero* for stuck-at-1 fault, *All-one* for stuck-at-0 fault



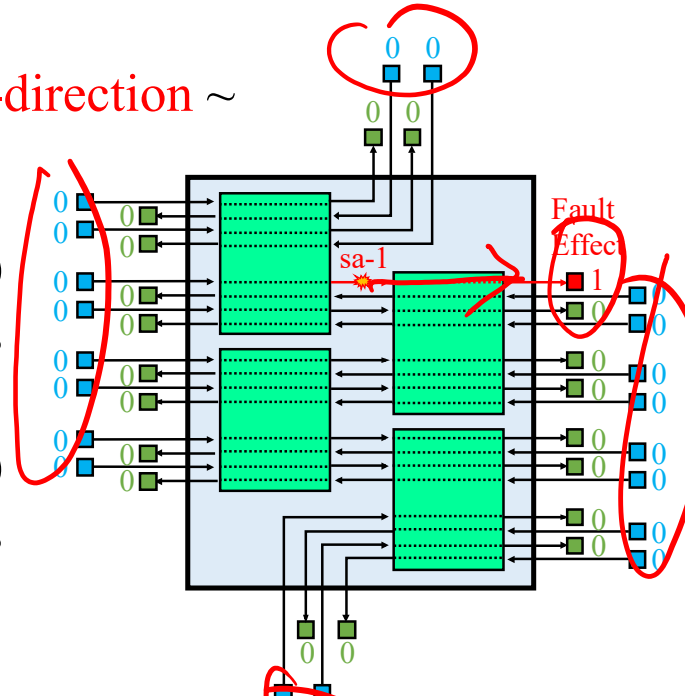


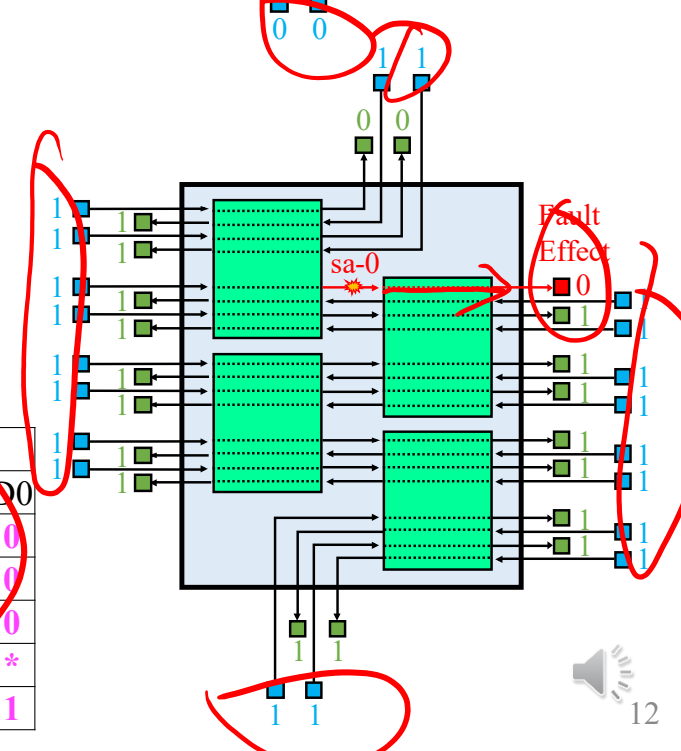
Diagnostic Cube 1

Address				Data							
A3	A2	A1	A0	D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	0	0	0	0	0	0	0
0	0	1	0	0	1	0	0	0	0	0	0
*	*	*	*	*	*	*	*	0	0	0	0
1	1	1	1	1	1	1	1	0	0	0	0

Diagnostic Cube 2

Address				Data							
A7	A6	A5	A4	D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	1	0	0	0
0	0	1	0	0	0	0	0	0	1	0	0
*	*	*	*	0	0	0	0	*	*	*	*
1	1	1	1	0	0	0	0	1	1	1	1





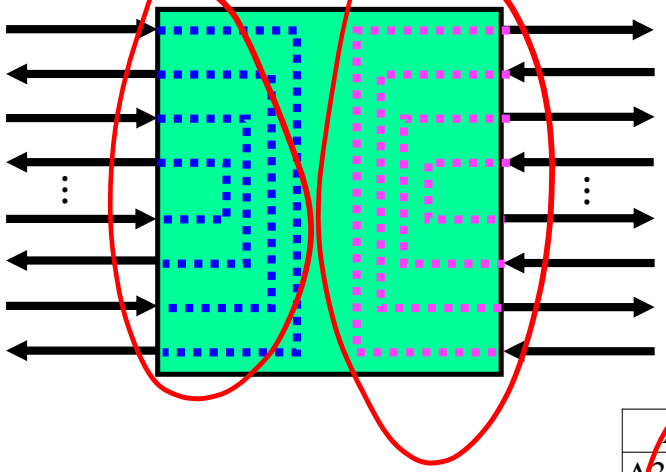
Diagnostic Cubes & External Patterns ~ Col-direction ~

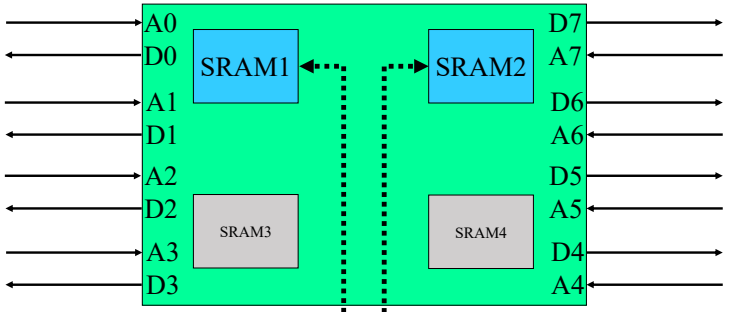
- **Diagnostic Cube for col-direction (DCc):**

Diagnostic Cube 1: For the SRAMs share the low-order address inputs (A[m/2-1:0]) of MLUT, set contents of the address lines A[m/2-1:0] to D[m-1:m/2]=all-zero, D[m/2-1:0]=A[0:m/2-1].

Diagnostic Cube 2: For the SRAMs share the high-order address inputs (A[m-1:m/2]) of MLUT, set contents of the address lines A[m-1:m/2] to D[m-1:m/2]=A[m/2:m-1], D[m/2-1:0]=all-zero.

- **External Patterns:** *All-zero* for stuck-at-1 fault, *All-one* for stuck-at-0 fault



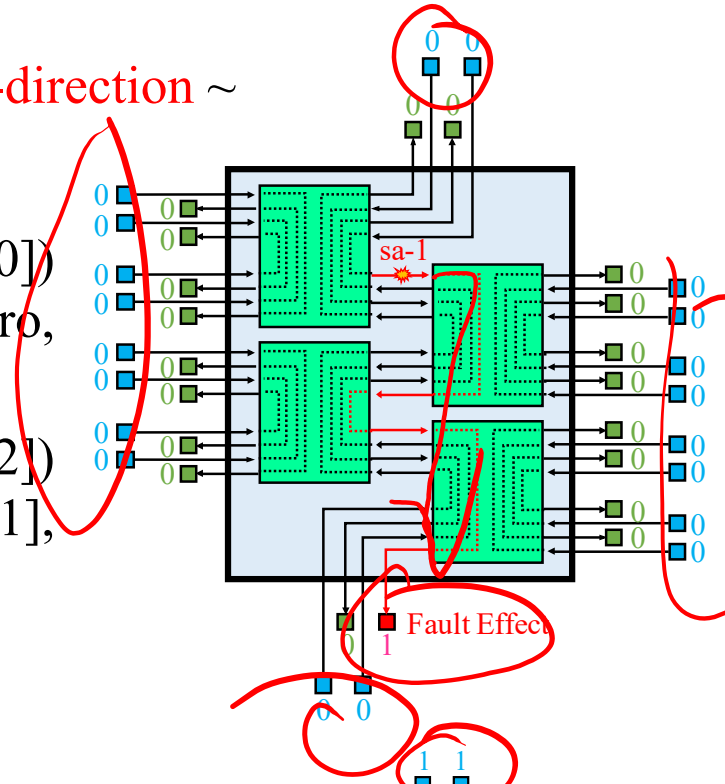


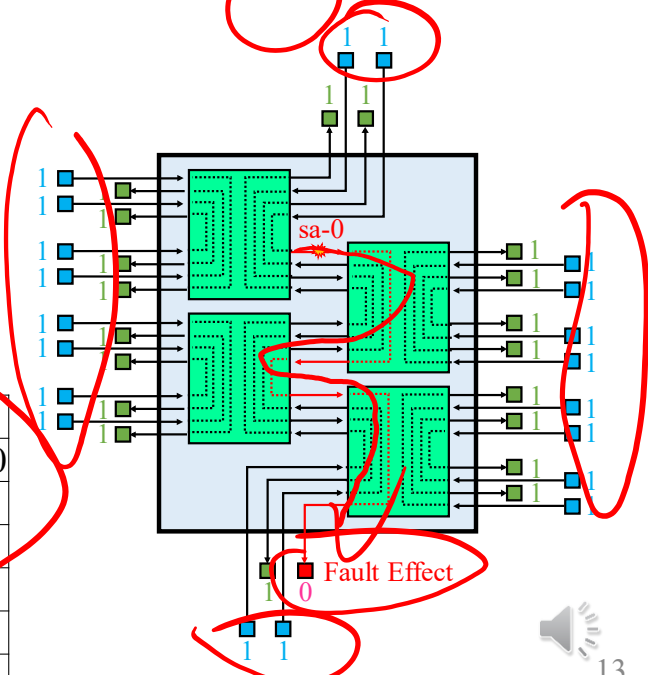
Diagnostic Cube 1

Address				Data							
A3	A2	A1	A0	D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	1	0	0	0
0	0	1	0	0	0	0	0	0	1	0	0
*	*	*	*	0	0	0	0	*	*	*	*
1	1	1	1	0	0	0	0	1	1	1	1

Diagnostic Cube 2

Address				Data							
A7	A6	A5	A4	D7	D6	D5	D4	D3	D2	D1	D0
0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	0	0	0	0	0	0	0
0	0	1	0	0	1	0	0	0	0	0	0
*	*	*	*	*	*	*	*	0	0	0	0
1	1	1	1	1	1	1	1	0	0	0	0



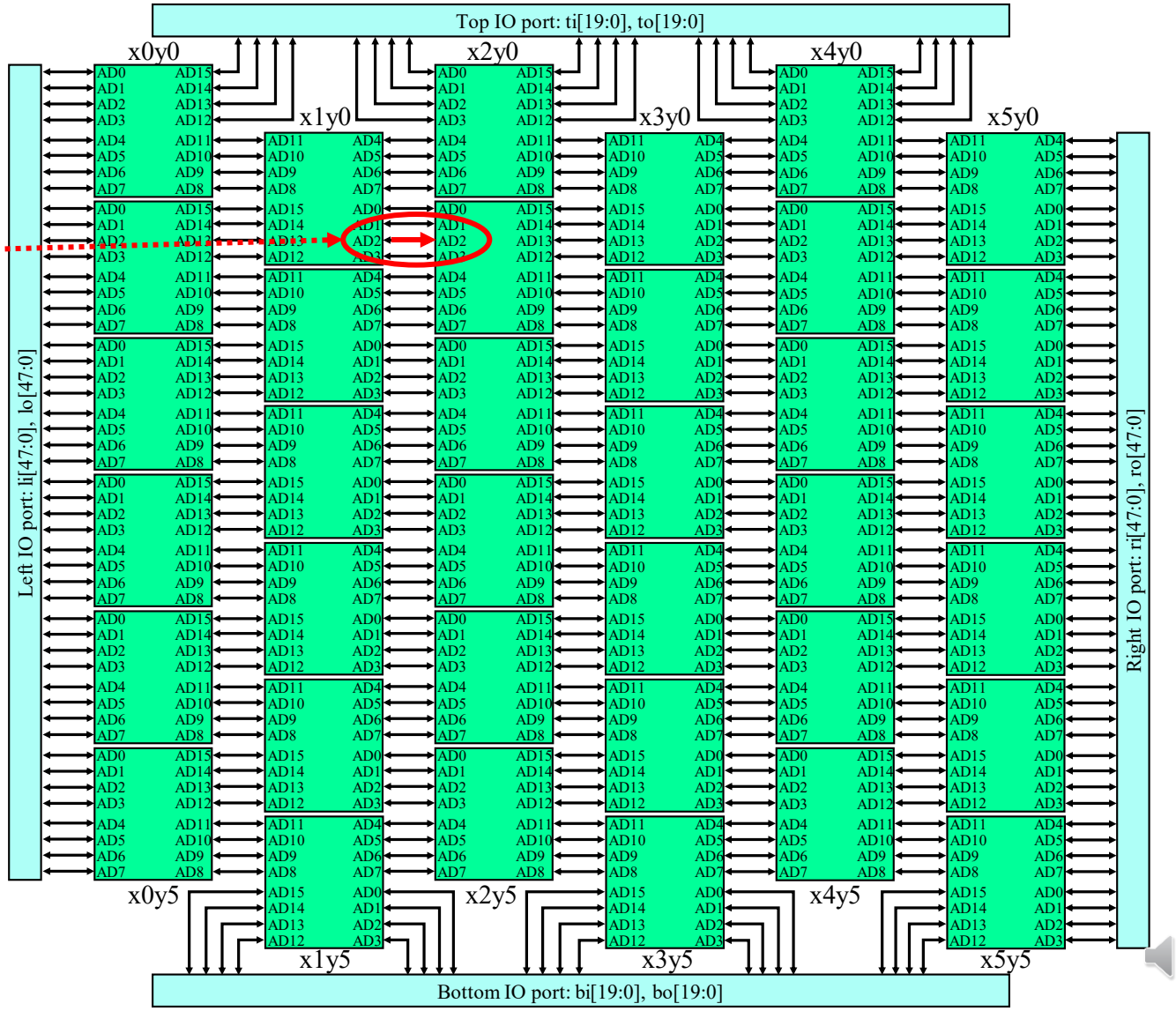


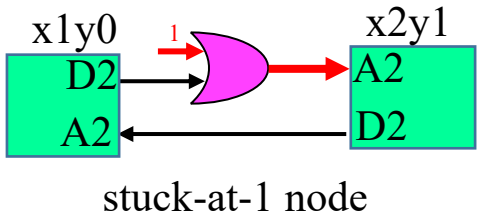
Simulation method

- MRLD design: 6 × 6 MLUT array
  - IO ports: left & right 48bits, top & bottom 20bits
  - MLUT: Four 256word × 16bit SRAMs
- Simulation method
  - Logic simulation by ModelSim
  - Fault node insertion (random)

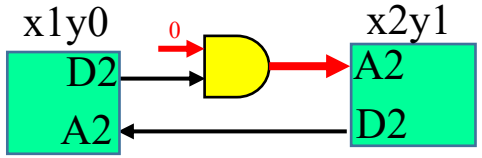
Port Definition:

li: left address input	*lo: left data output*
ri: right address input	*ro: right data output*
ti: top address input	*to: top data output*
bi: bottom address input	*bo: bottom data output*





stuck-at-1 node

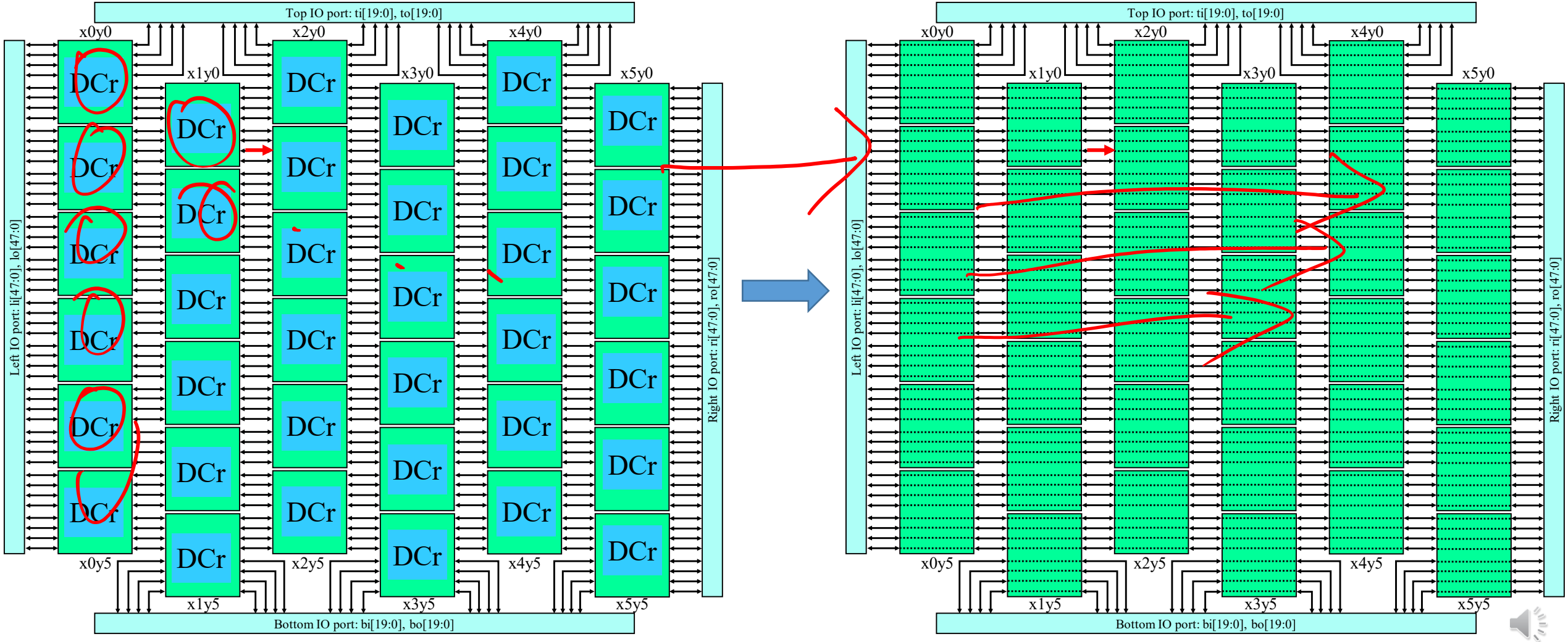


stuck-at-0 node

Simulation Flow

Step 1 : Reconfigure Diagnostic Cube for row-direction (DCr)

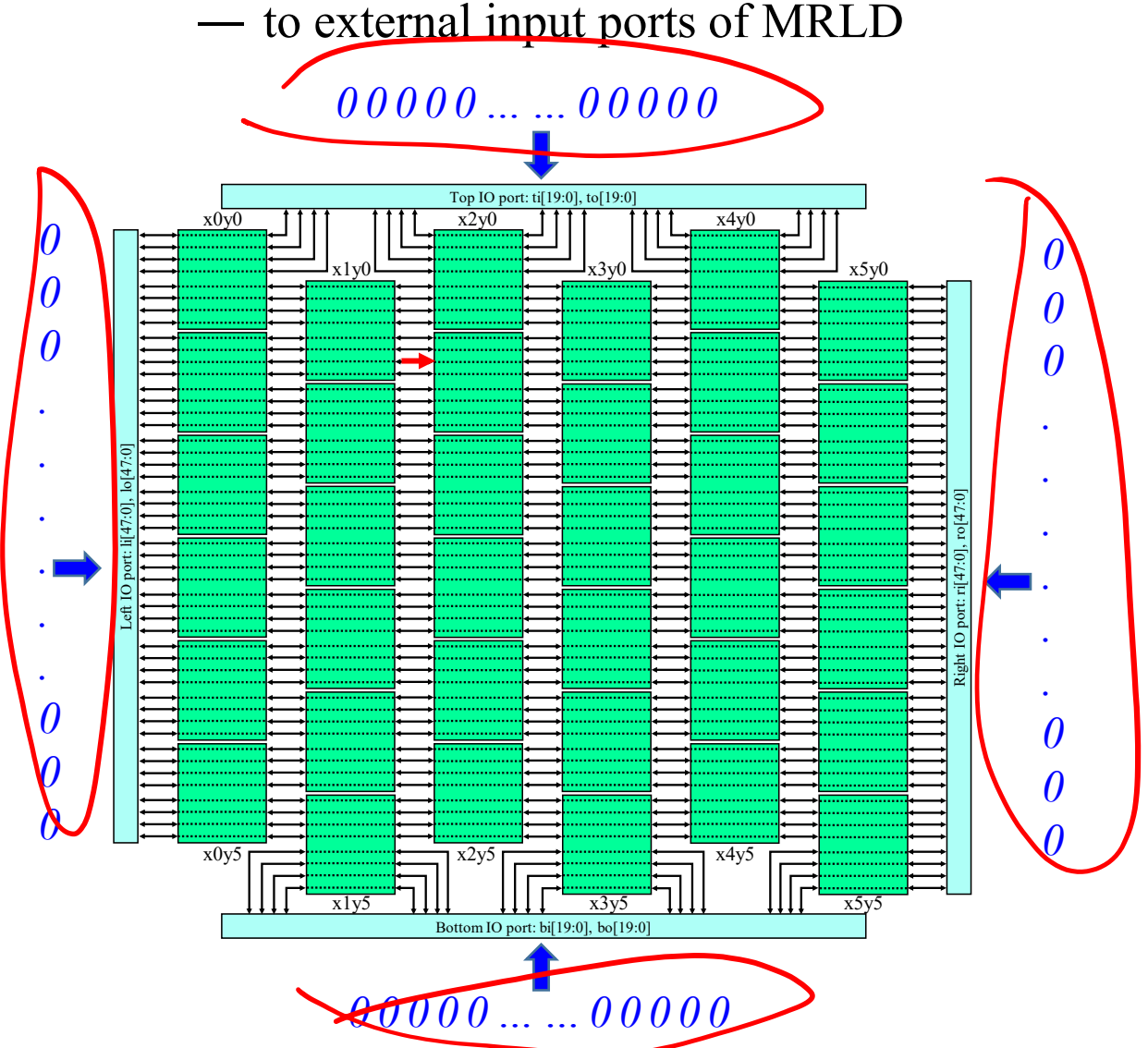
— into each MLUT



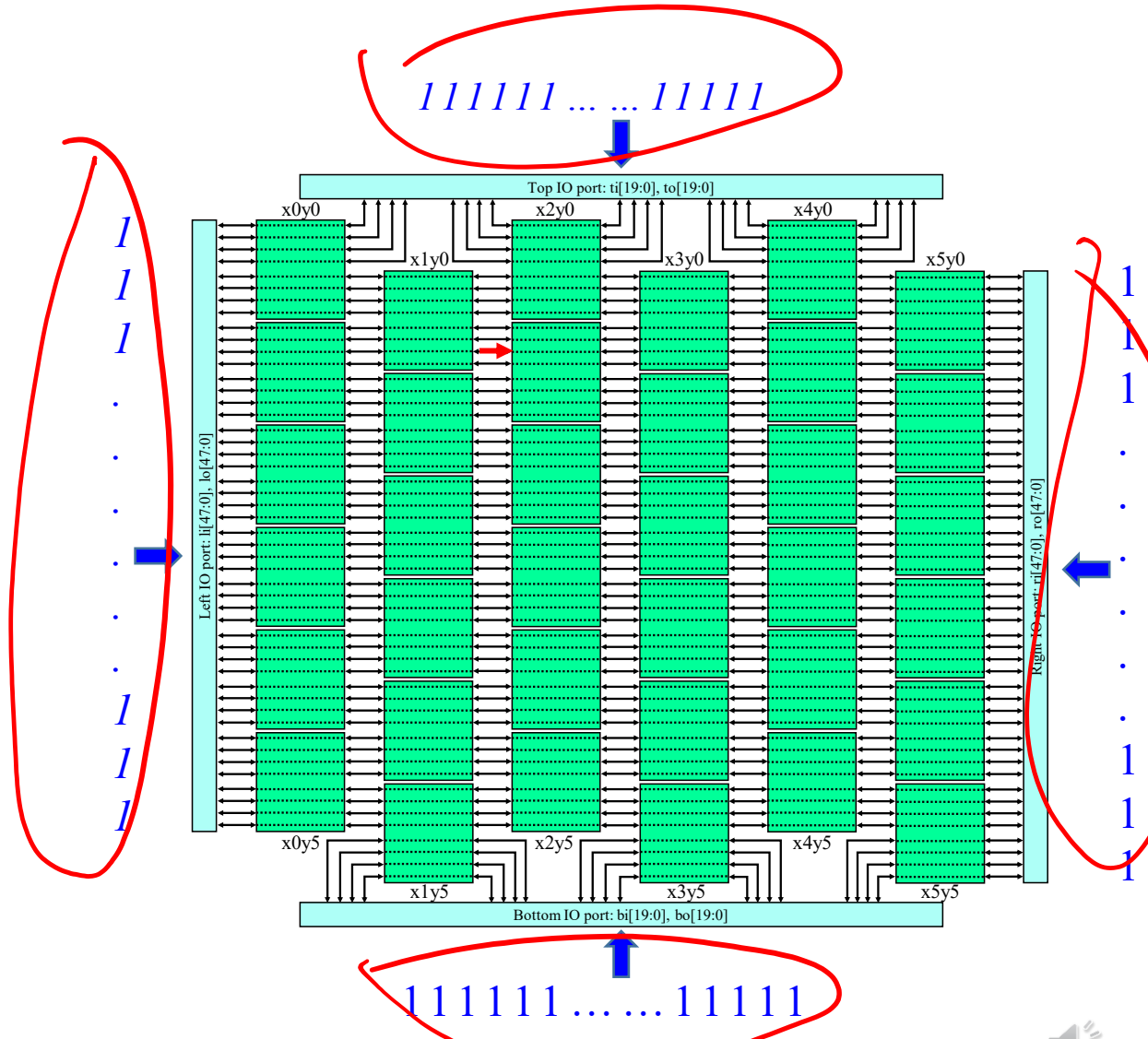
Simulation Flow

Step 2 : Apply External Patterns

— to external input ports of MRLD



External Patterns *all-0* for stuck-at-1

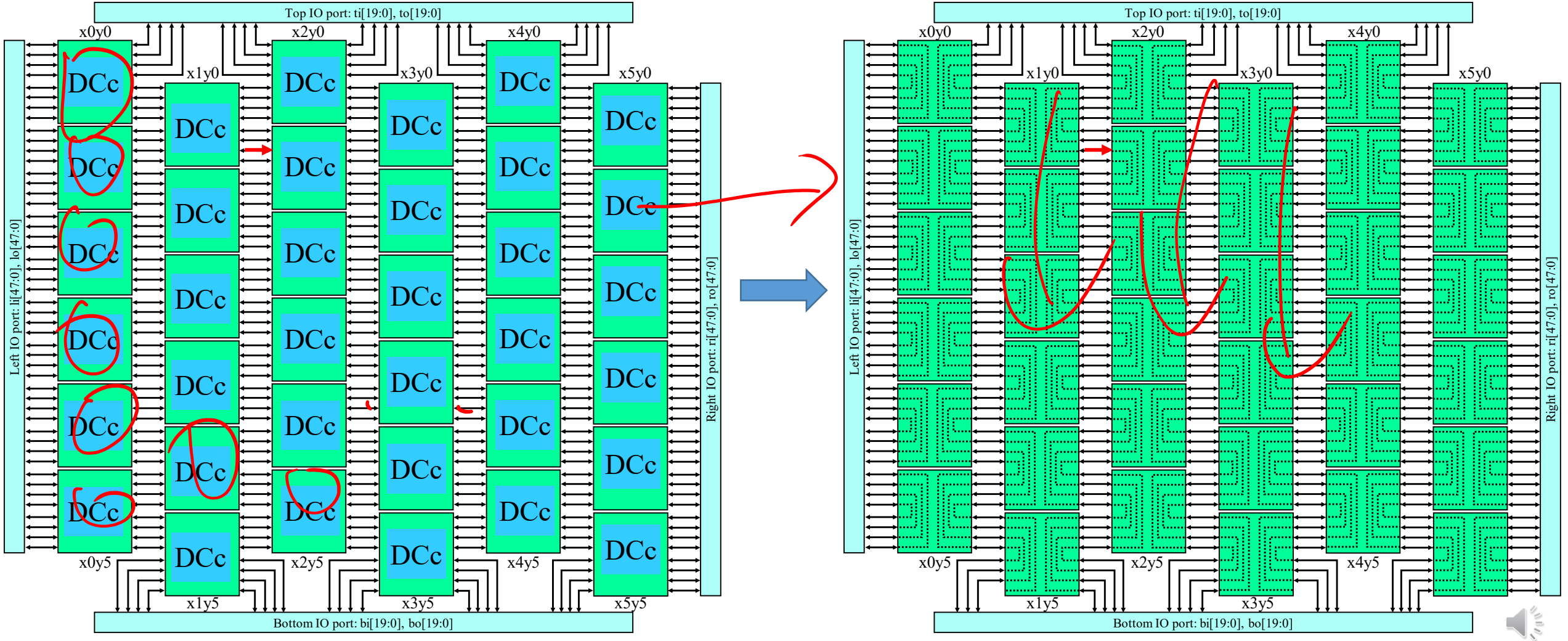


External Patterns *all-1* for stuck-at-0

Simulation Flow

Step 3 : Reconfigure Diagnostic Cube for col-direction (DCc)

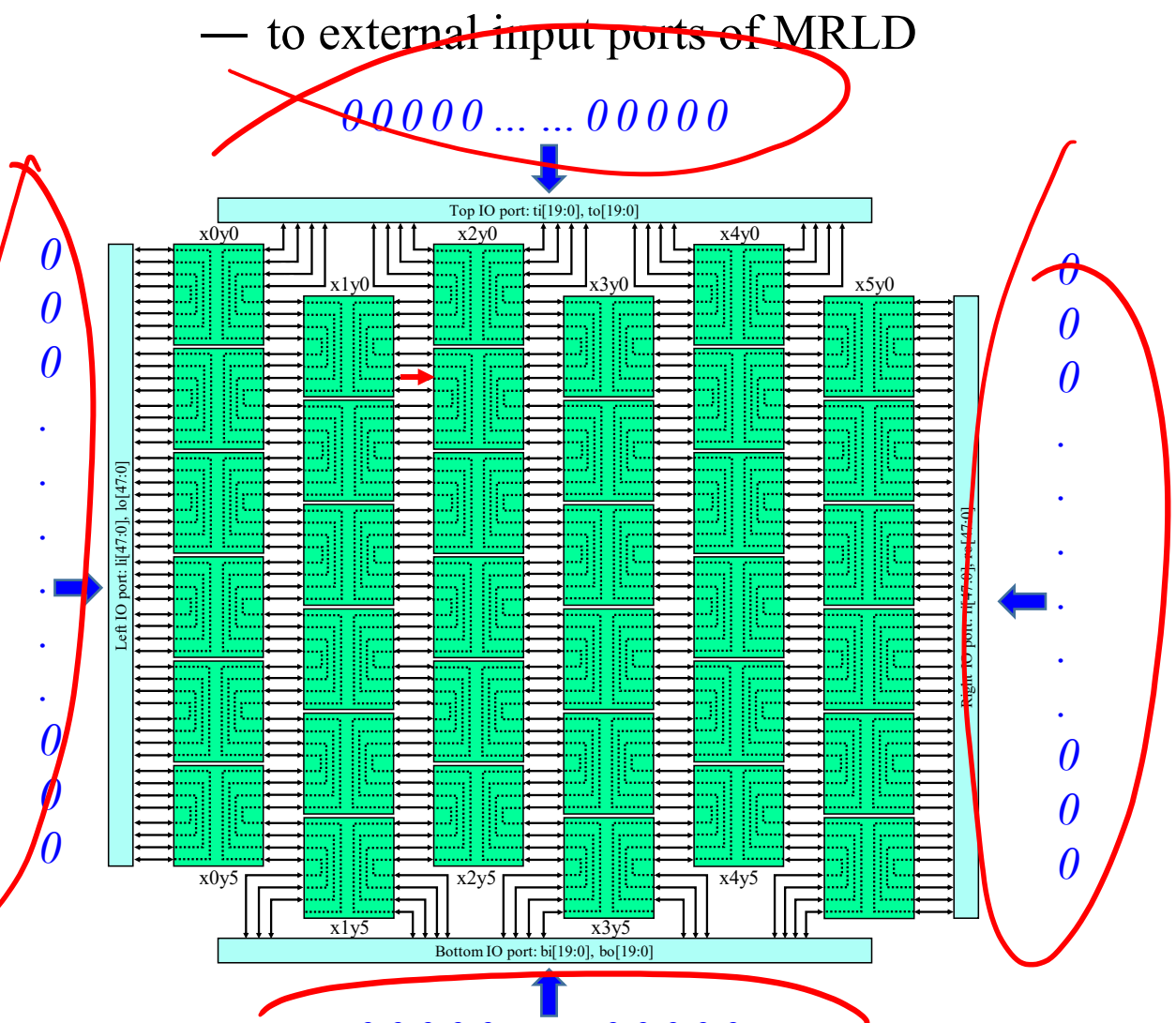
— into each MLUT



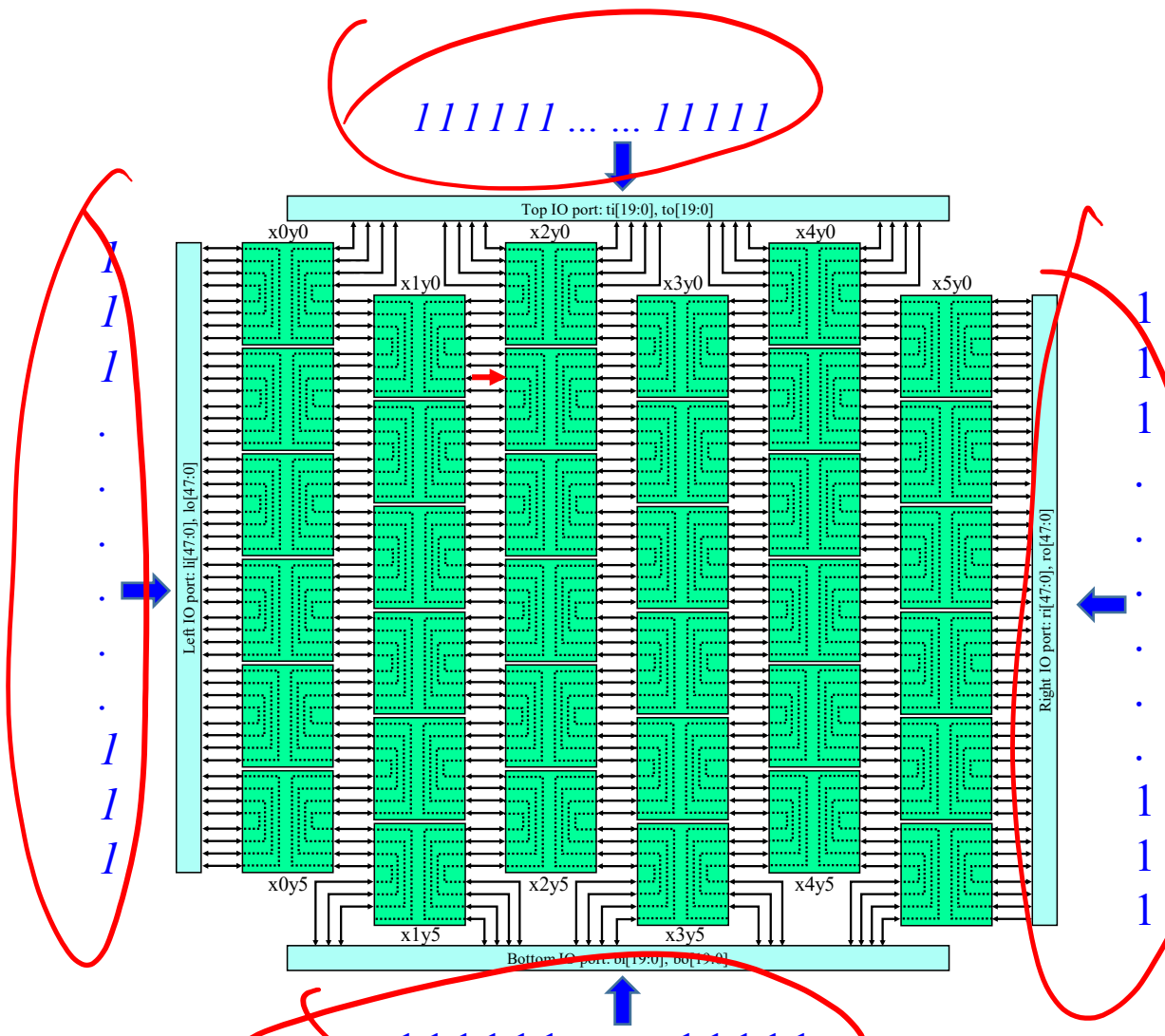
Simulation Flow

Step 4 : Apply External Patterns

— to external input ports of MRLD

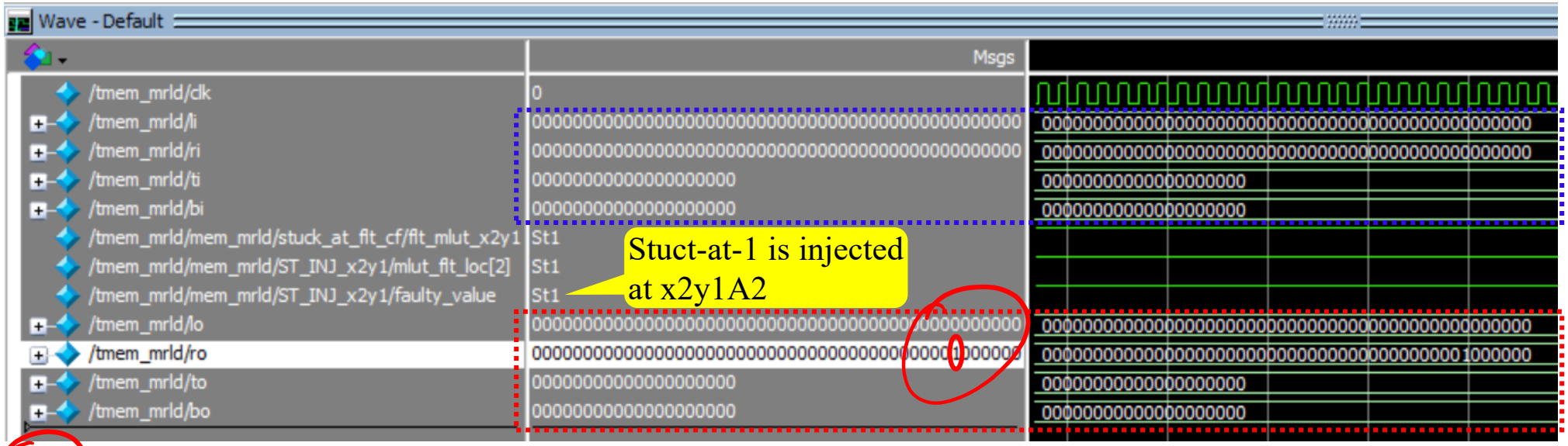


External Patterns *all-0* for stuck-at-1



External Patterns *all-1* for stuck-at-0

Simulation result ~stuck-at 1 fault diagnosis ~

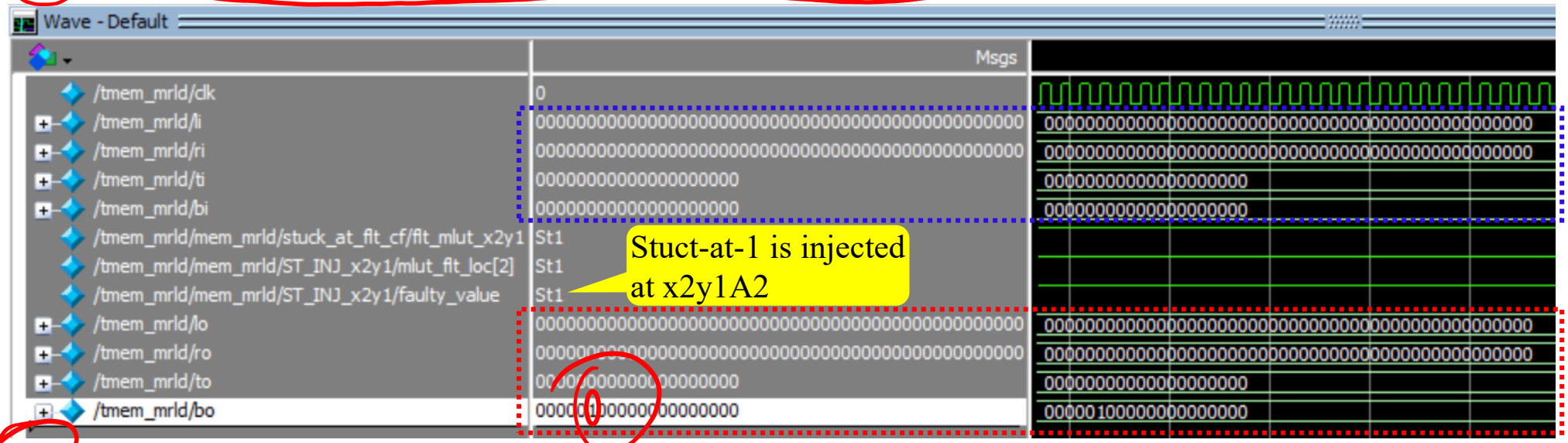


a. row-direction

External Patterns: *all-zero*

External Outputs: ro[6] = 1

FPr = {x0y1A2 → x1y0A13 → x2y1A2 → x3y0A13 → x4y1A2 → x5y0A13 → ro[6]}



b. col-direction

External Patterns: *all-zero*

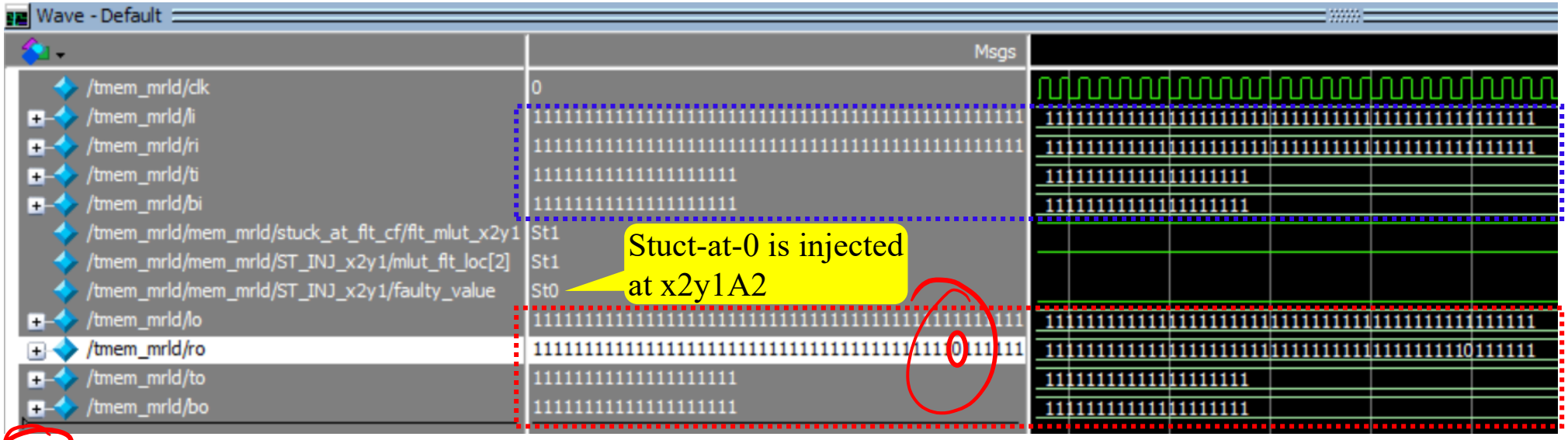
External Outputs: bo[14] = 1

FPc = {x2y0A2 → x1y0A5 → x2y1A2 → x1y1A5 → x2y2A2 → x1y2A5 → x2y3A2 → x1y3A5 → x2y4A2 → x1y4A5 → x2y5A2 → x1y5A5 → bo[14]}

F_{loc} = FPr ∩ FPc= x2y1A2

Simulation result ~stuck-at 0 fault diagnosis ~

a. row-direction

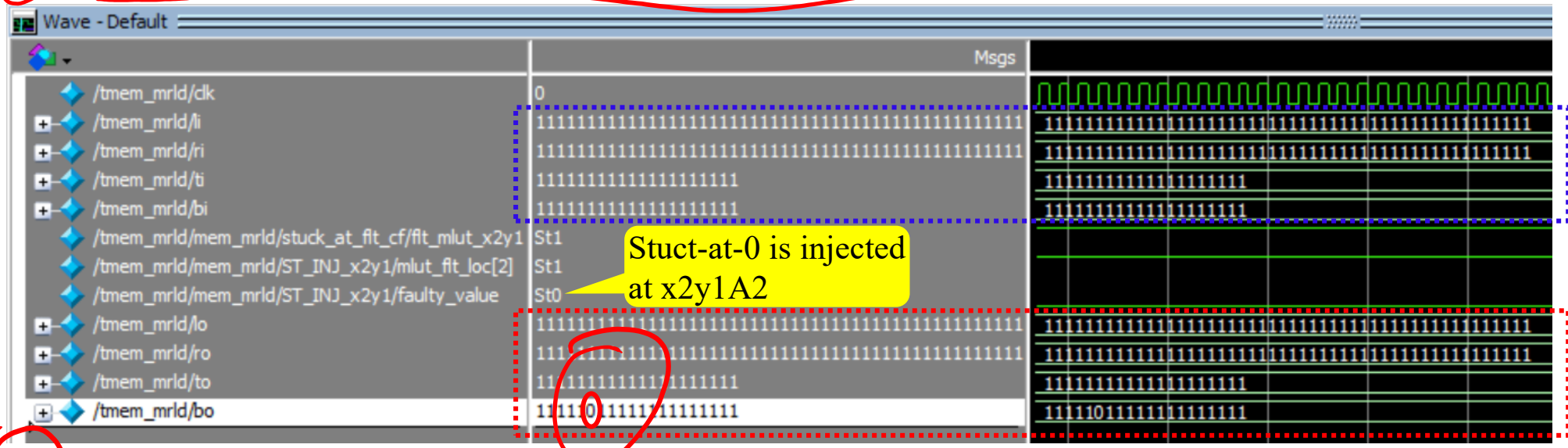


External Patterns: *all-one*

External Outputs: ro[6] = 0

FPr = {x0y1A2 → x1y0A13 → x2y1A2 → x3y0A13 → x4y1A2 → x5y0A13 → ro[6]}

b. col-direction



External Patterns: *all-one*

External Outputs: bo[14] = 0

FPc = {x2y0A2 → x1y0A5 → x2y1A2 → x1y1A5 → x2y2A2 → x1y2A5 → x2y3A2 → x1y3A5 → x2y4A2 → x1y4A5 → x2y5A2 → x1y5A5 → bo[14]}

F_{loc} = FPr ∩ FPc= x2y1A2

Conclusions

- MRLD should be a promising alternative reconfigurable device to FPGA with the benefits of low production cost, low power and small delay.
- We proposed the diagnosis strategy and the method for locating the stuck-at interconnect faults.
  - The method can diagnose the location of all stuck-at faults at any interconnects.

MRLD size	Total fault numbers	Locatable fault numbers	Reconfiguration (Row and Column)	External Pattern (sa-1 and sa-0)
X × Y MLUT(with M-bit) array	$\left((X+1)Y+\frac{X-1}{2}\right)$M	$\left((X+1)Y+\frac{X-1}{2}\right)$M	2 times	2 times

**X, Y: the number of rows and columns for MLUT array*
**M: the number of AD line pairs for an MLUT*

- Future work
  - Evaluate the effectivity of the proposed diagnosis method for multiple stuck-at faults .
  - Analyze the diagnostic generation method for locating others interconnect faults such as bridge faults and open fault in MRLD.

Thanks for listening