

Sys-Verilog Questions Review

Some Solutions to questions from ChipIO-Dev

Hector “Hectron” Williams

Counter

clk

Bit

n_ticks

Logic [7:0]



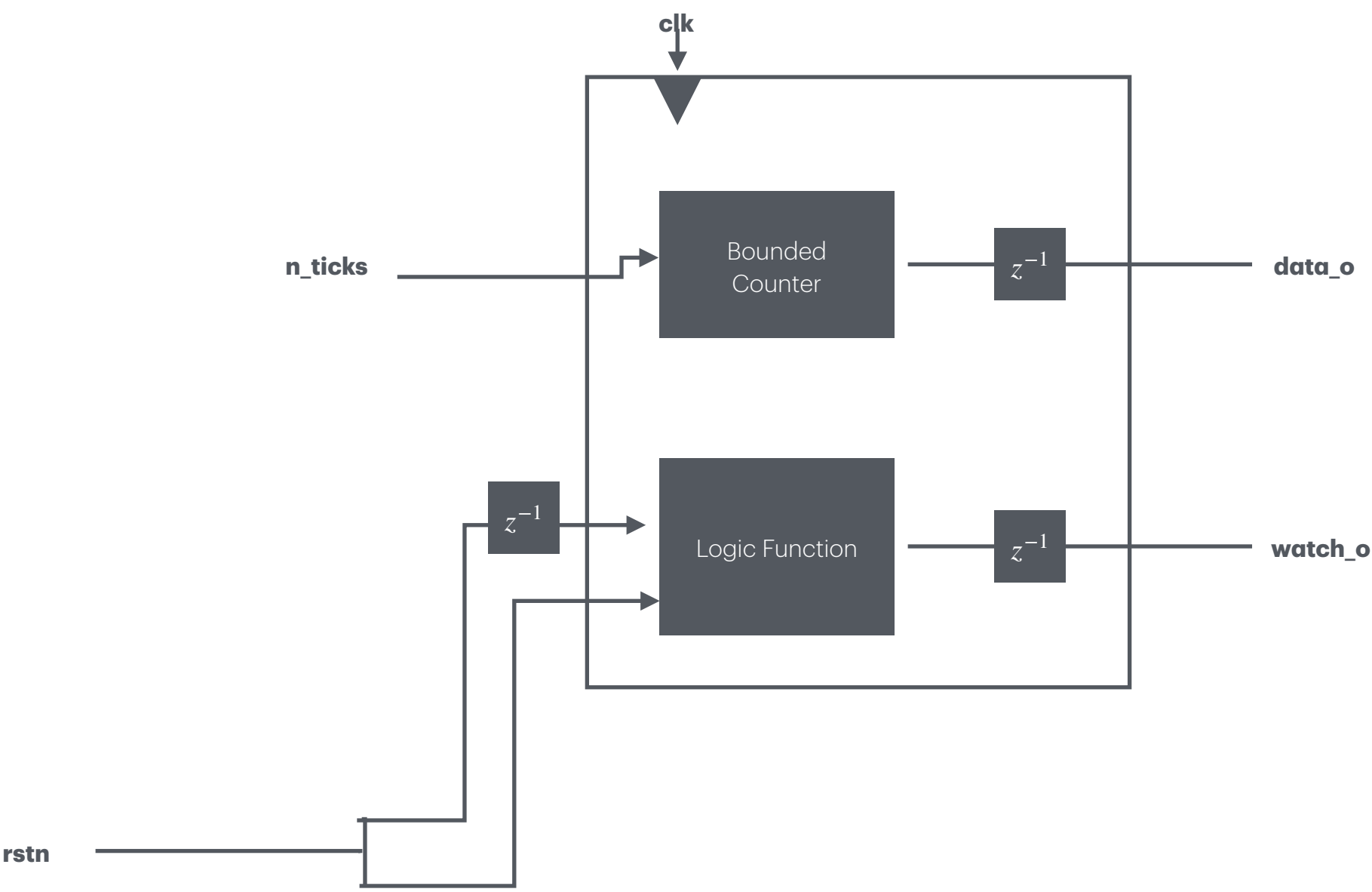
data_o

Logic [7:0]

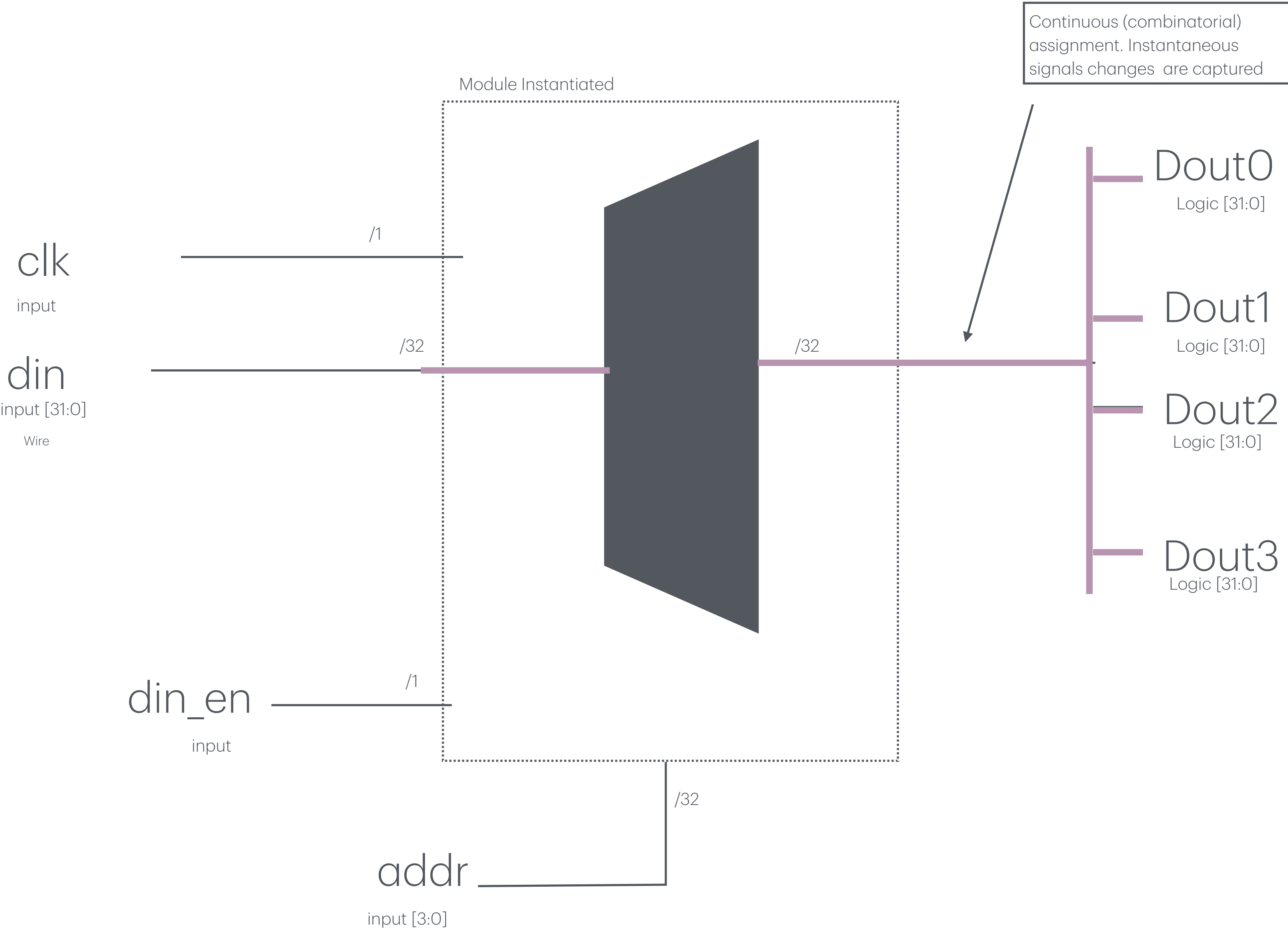
watch_o

Bit

Counter



Router



Connect (wire)



Log2

clk

Bit

X

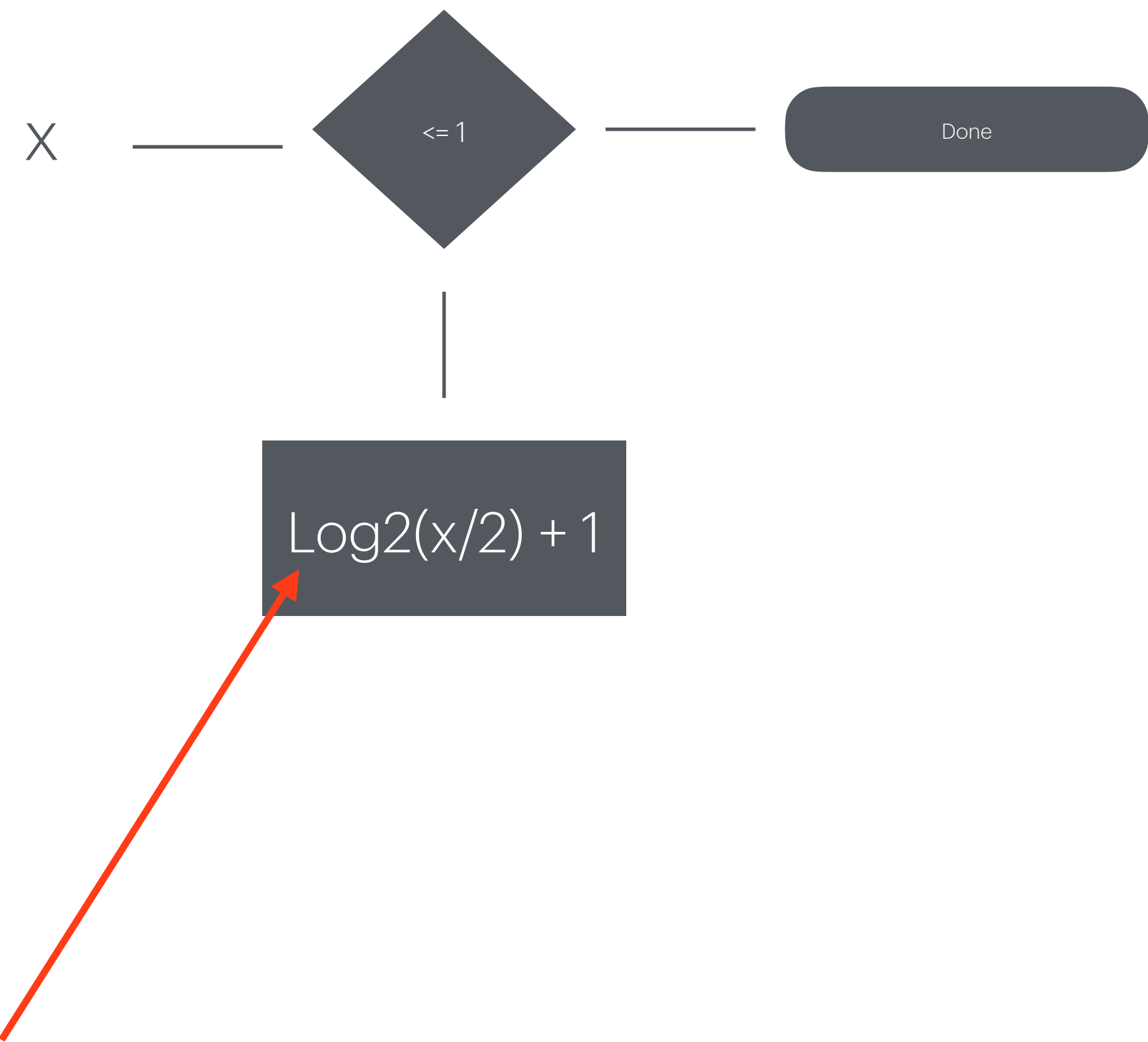
Logic [7:0]

y

Logic [7:0]



Log2



Recursion or calling the same hardware segment repeatedly

Log2(**5**)

5

Log2(5)

1

Log2(2)

2

Log2(1)

Minimum number of bits represent 5?

Min = 2

5 > 2^2 (increment)

Log2(**4**)

5

Log2(4)

1

Log2(2)

2

Log2(1)

Minimum number of bits represent 5?

Min = 2

4 \nless 2^2 (perfect)

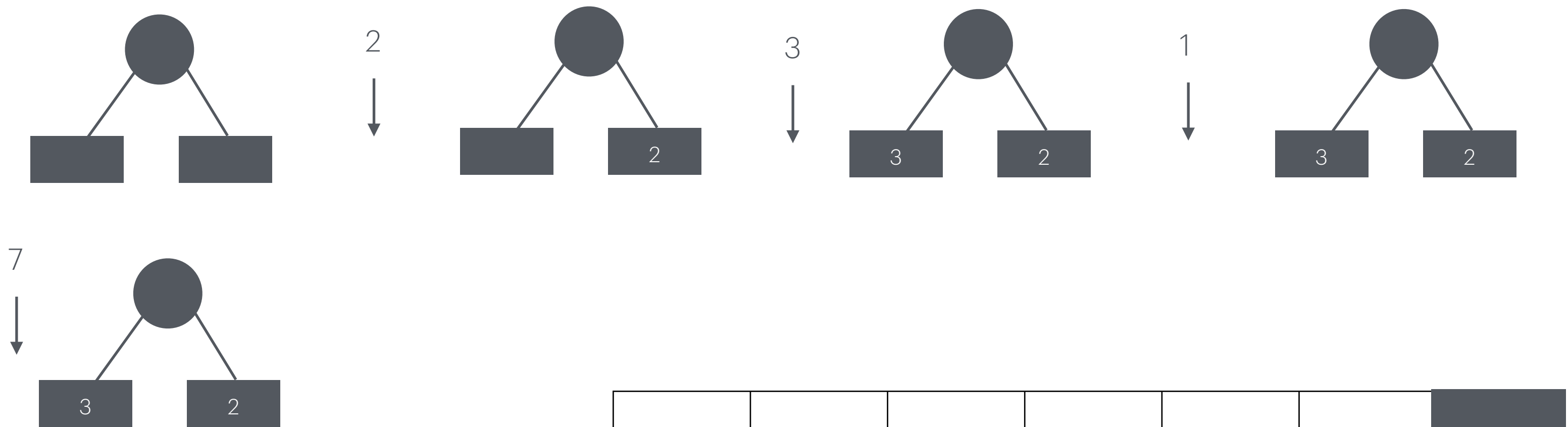
Log2 : Debug Results

VECTOR SENT [1] - [36]
VECTOR SENT [2] - [129]
VECTOR SENT [3] - [9]
VECTOR SENT [4] - [99]
VECTOR SENT [5] - [13]
VECTOR SENT [6] - [141]



VECTOR RCVD [6]
VECTOR RCVD [8]
VECTOR RCVD [4]
VECTOR RCVD [7]
VECTOR RCVD [4]
VECTOR RCVD [8]

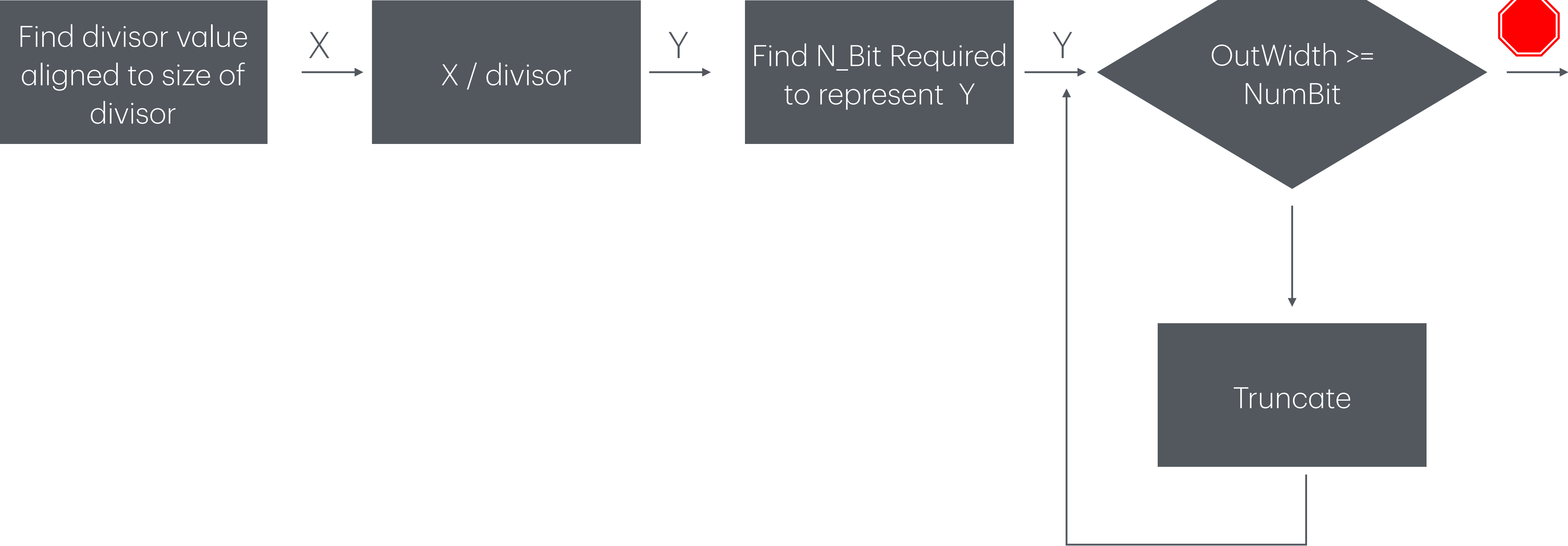
Second Largest



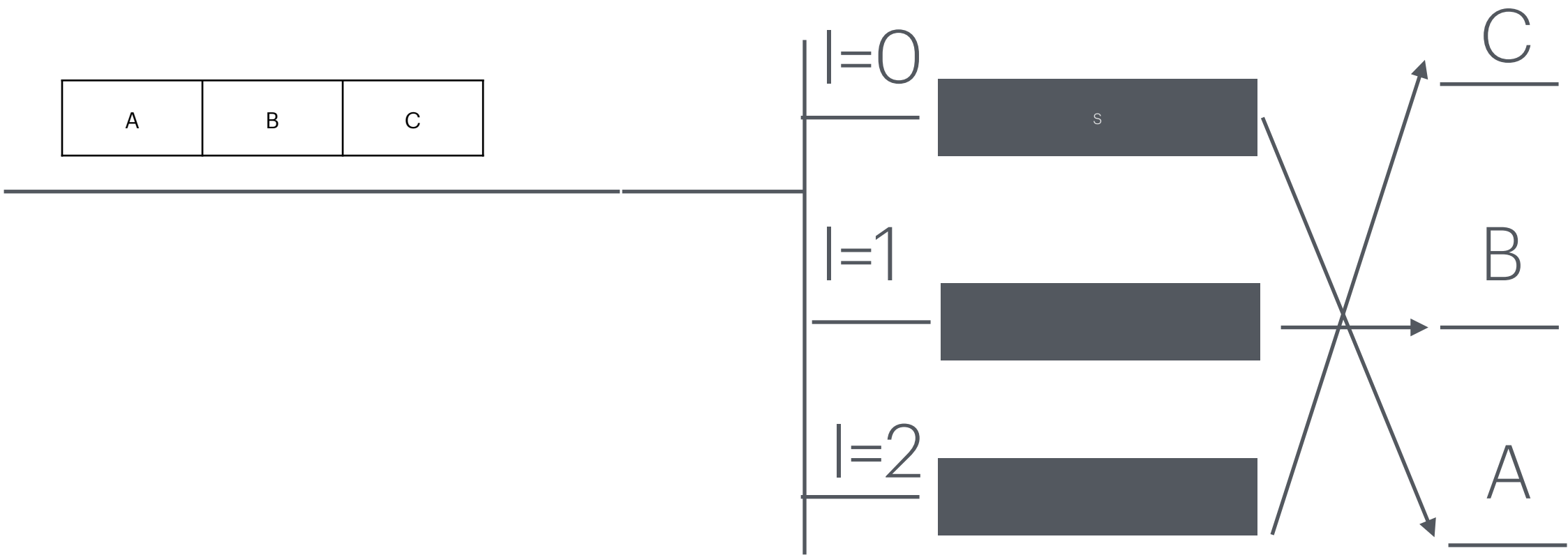
Count	0	1	2	3	4	
Data_In	D0	D1	D2	D3	D4	
2nd Largest	0	0	2	2	2	3

-, **2** 3, **2** 3, **2** 3, **2** 7, **3**

Rounded Division



Reverse Bits



Generate Logic Blocks

Gray code

RST

CLK

Counter

i

id

$$2^{id} = (counter + 1)$$

\lceil

FSM

out[l]

Register
Size = 2^i

$i+1$

id

$$2^{id+1} = active_bits$$

\lceil

FSM

out[l+1]

Register
Size = 2^i

⋮

out

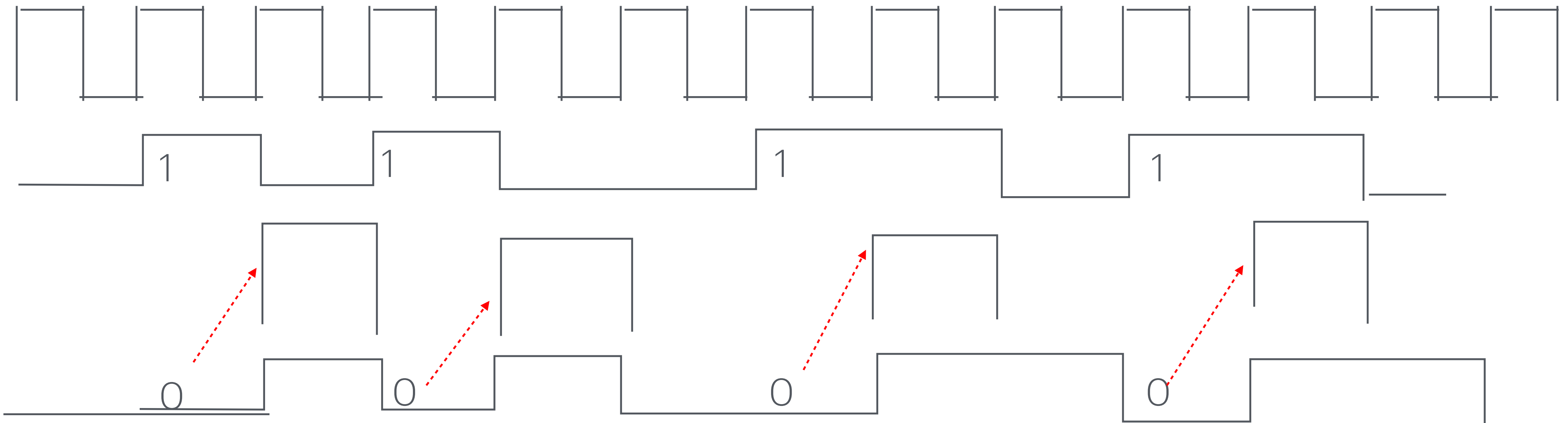
Gray code

Vertical Delay :)

$2^3 = 8cycle$ $2^2 = 4cycle$ $2^1 = 2cycle$ $2^0 = 1cycle$

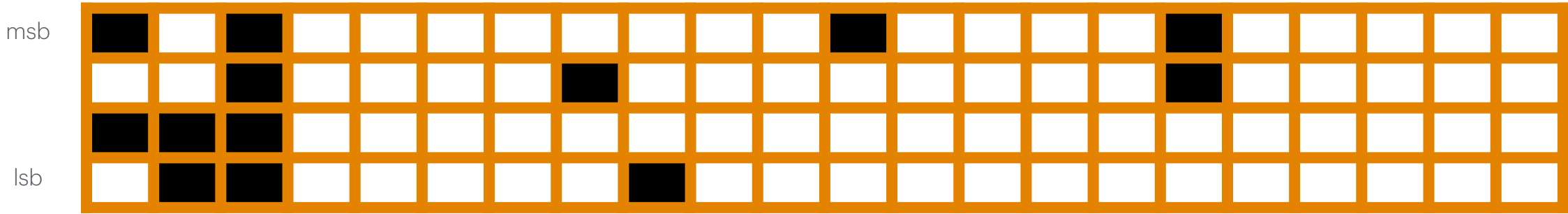
	0	0	0	0
	0	0	0	1
	0	0	1	1
	0	0	1	0
	0	1	1	0

Edge Detector



Parralel In -Serial Out

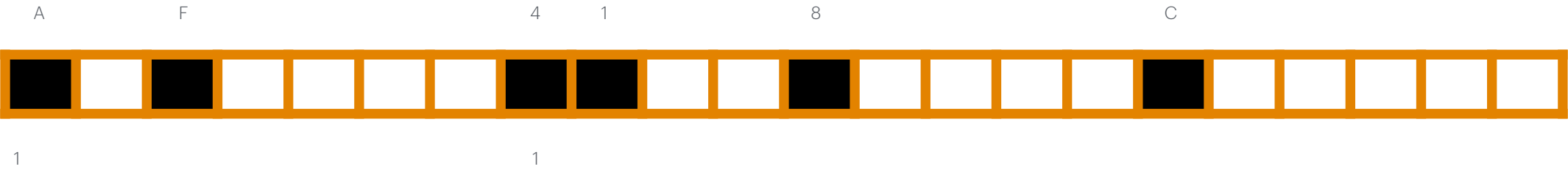
DATA



Bit Sequence



Enable



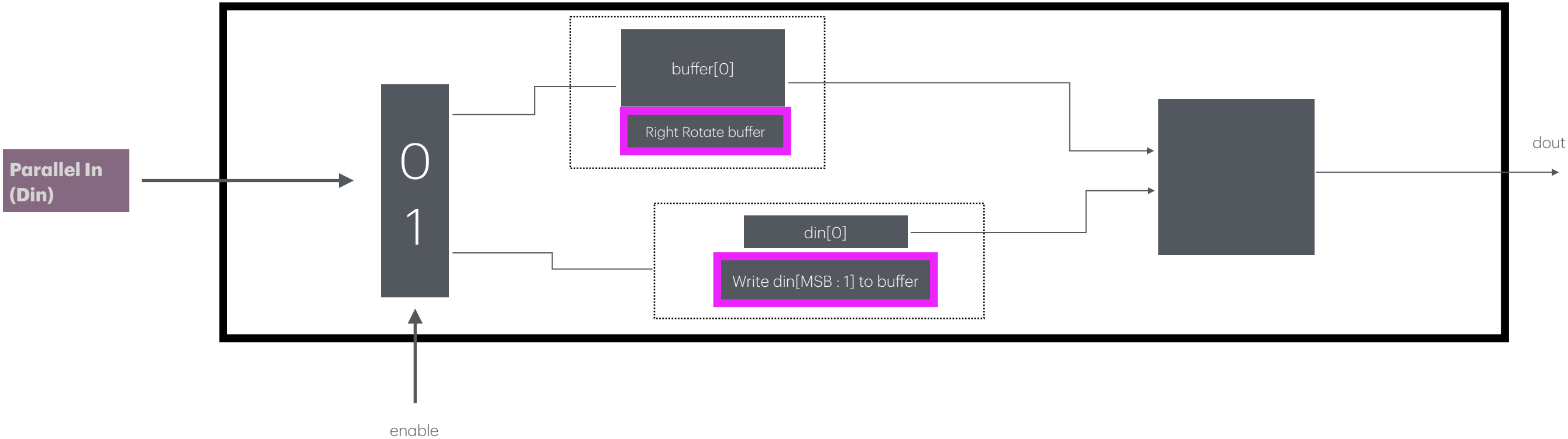
D_IN



RST



Output



Serial to Parallel

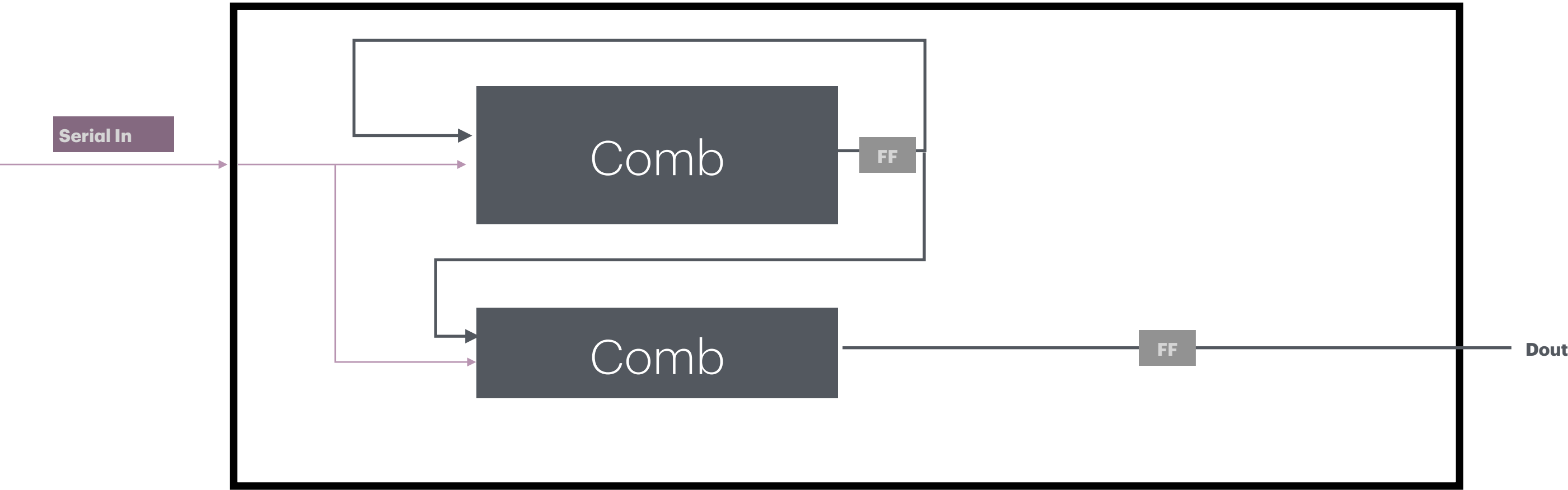
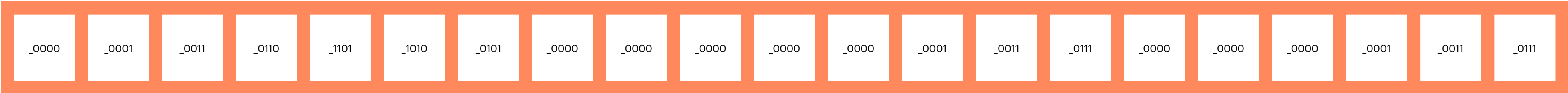
Din



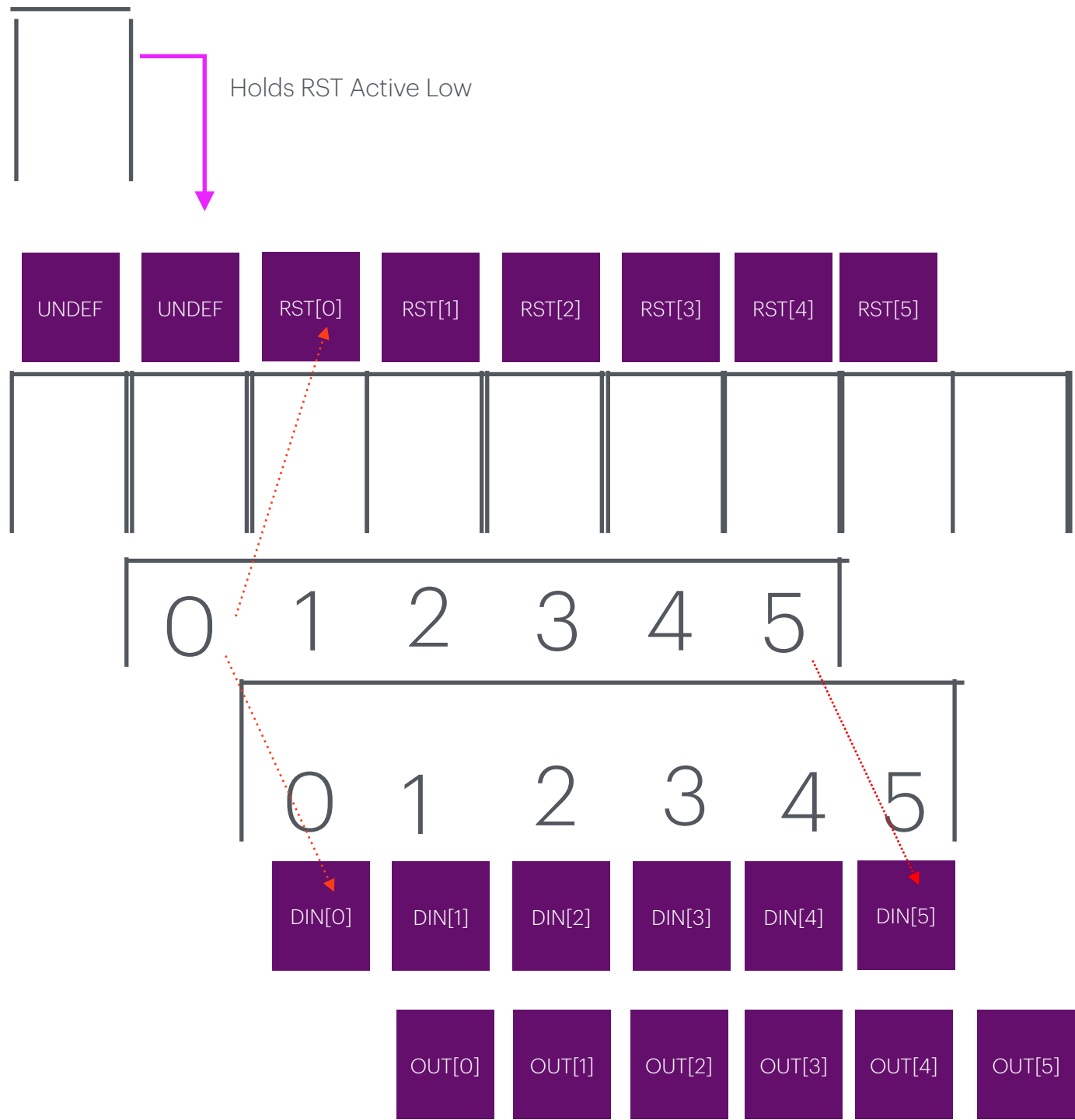
Reset



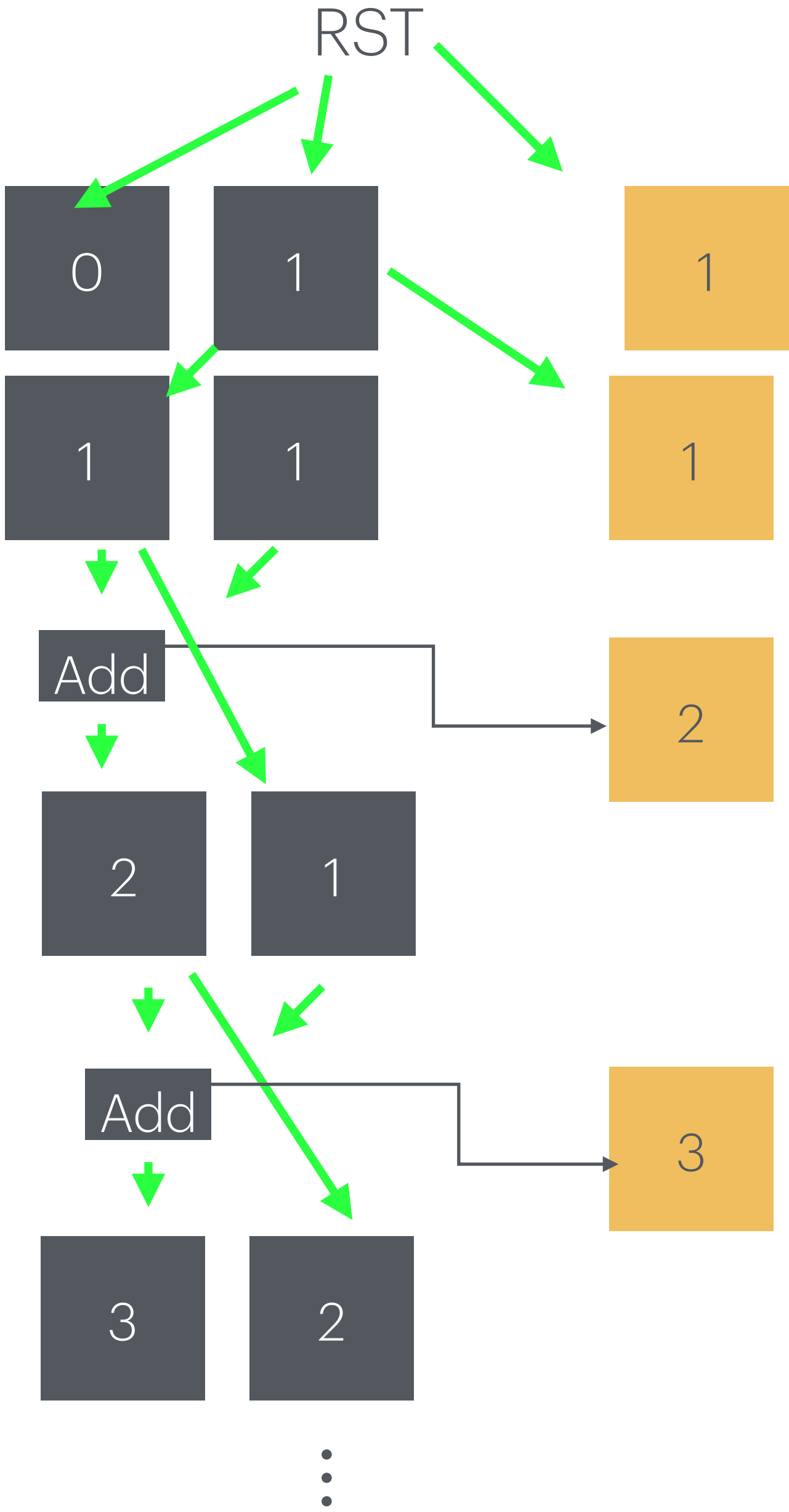
Data



Serial to Parallel (Simulation Concept)

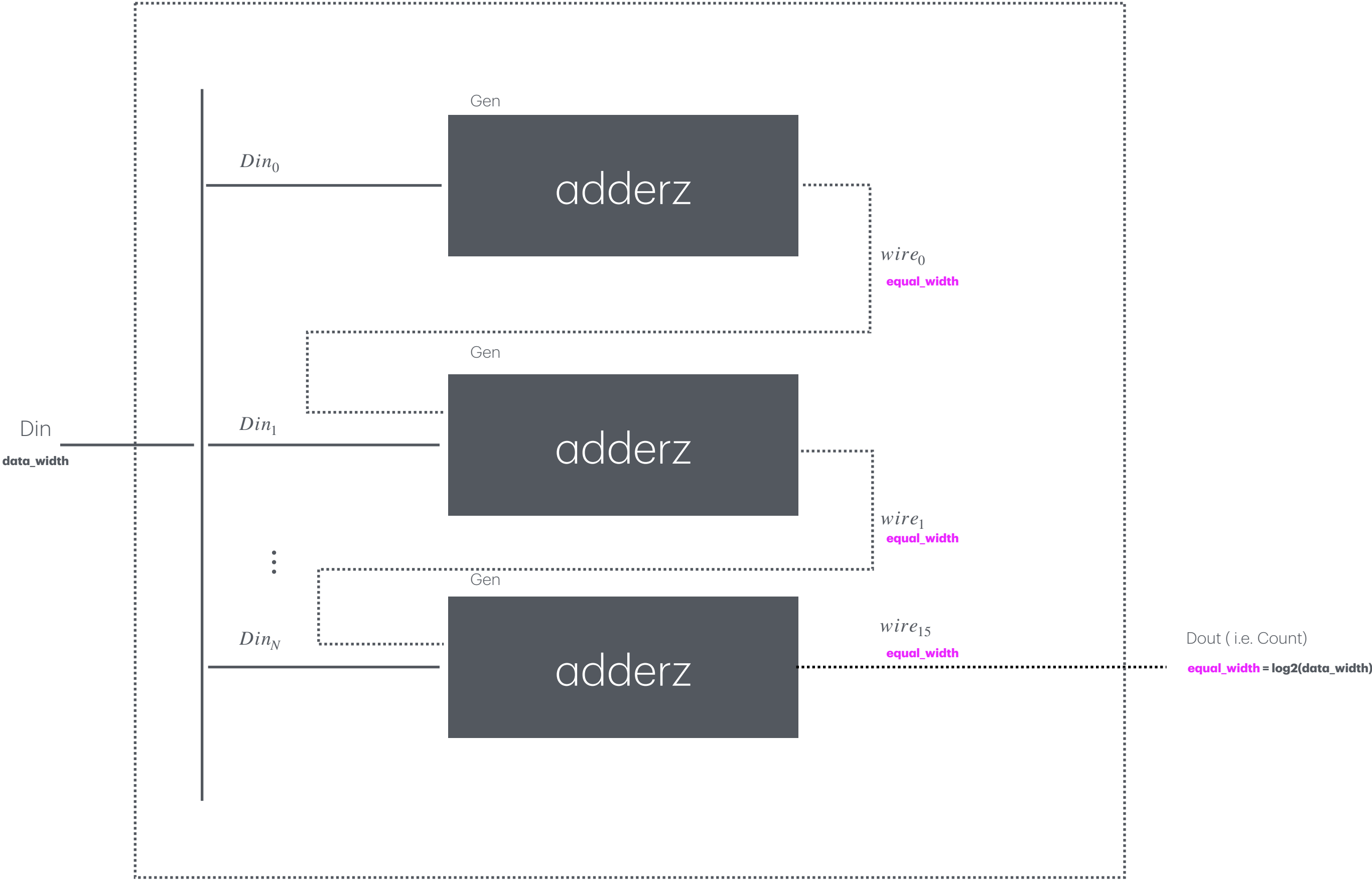


Fibonacci



Count Ones

Architecture Similar to Linked List



Count Ones

```
[2025-10-21 23:50:16 UTC] iverilog '-Wall' '-g2012' design.sv testbench.sv && unbuffer vvp a.out
```

```
index - 0   input -    3   n_ones -  2
```

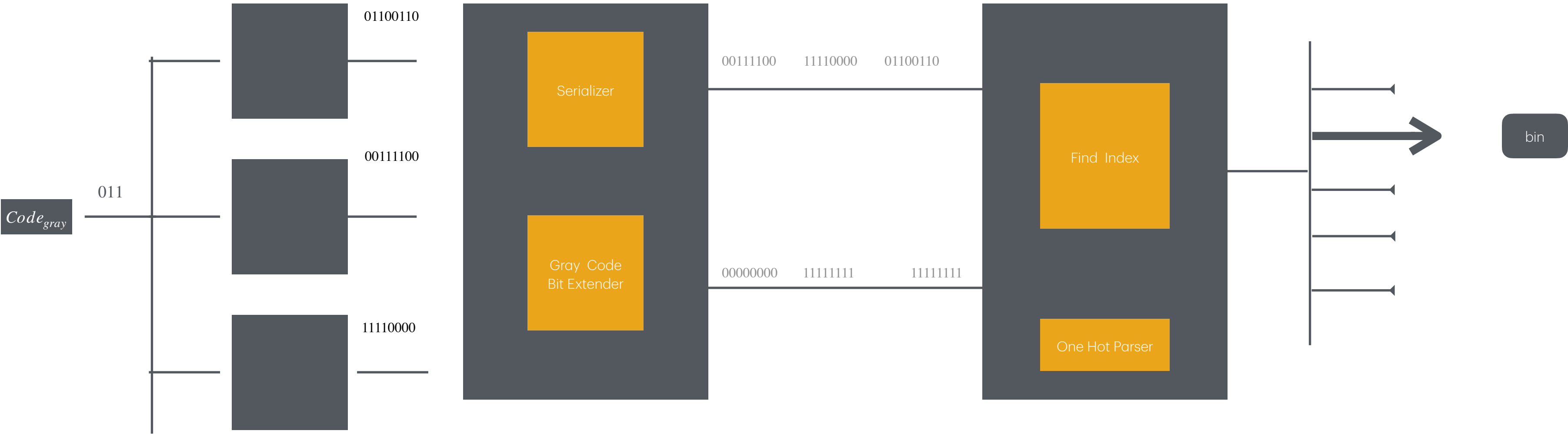
```
index - 1   input -    5   n_ones -  2
```

```
index - 2   input -    8   n_ones -  1
```

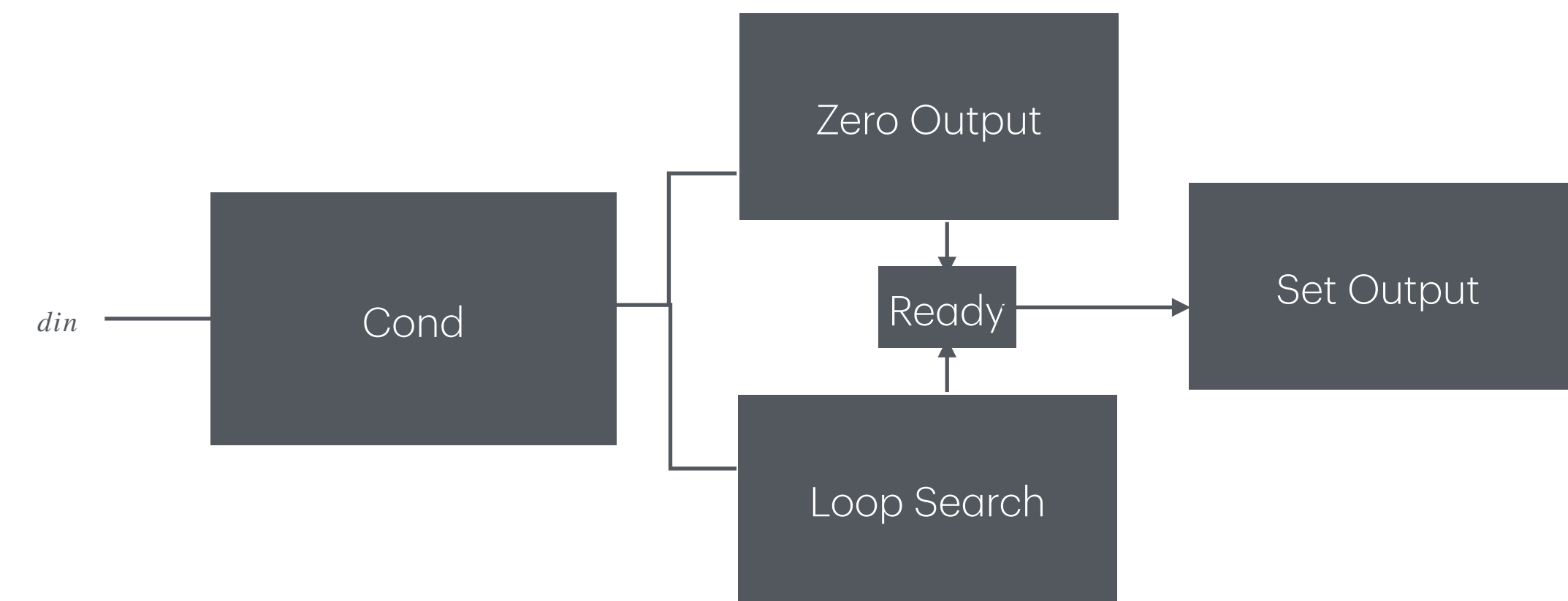
```
testbench.sv:44: $finish called at 9 (1s)
```

```
Done
```

Gray Code to Binary (Width = 3)



Trailing Ones



One-Hot Detector

Testbench

```
module one_hot_tb;

    // Parameters
    parameter DATA_WIDTH = 6;
    parameter N_ELEMENTS = 6;

    // Signals
    logic clk;

    logic go, go_z;

    logic [2:0] rdy;

    logic [DATA_WIDTH - 1: 0] rom[N_ELEMENTS];

    logic [DATA_WIDTH-1:0] din;
```

Module

```
module model
#(parameter

    DATA_WIDTH = 32

) (

    input  [DATA_WIDTH-1:0]
    din,

    output logic onehot

);

    always @ (din) begin

        if(din == 0)

            onehot = 0;

        else
```

StopWatch Timer

TICK	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	
START																								
RESET																								
STOP																								
COUNT	0	0	0	1	2	0	0	0	0	0	0	1	2	0	0	1	1	1	1	1	1	2	3	4



Testbench **monitors rdy** at this time stamp

StopWatch Timer

prev_button

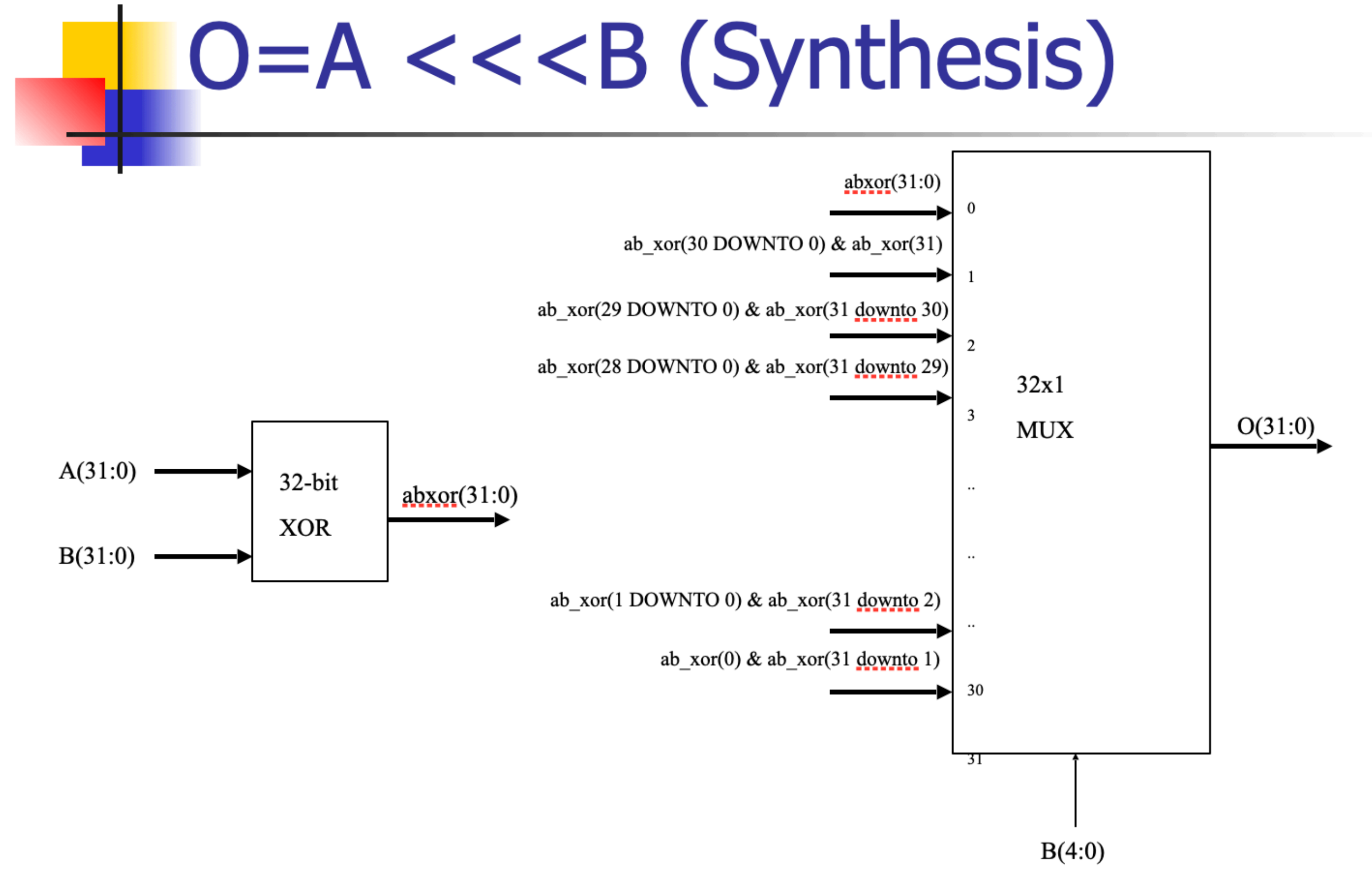
prev_button Event

```
[2025-10-23 19:15:00 UTC] iverilog '-Wall' '-g2012' design.sv testbench.sv && unbuffer vvp a.out
```

```
[ 0] Button Press: reset[0] start[0] stop[0] return - 0
[ 1] Button Press: reset[0] start[0] stop[0] return - 0
[ 2] Button Press: reset[0] start[1] stop[0] return - 0
[ 3] Button Press: reset[0] start[0] stop[0] return - 1
[ 4] Button Press: reset[1] start[0] stop[0] return - 2
[ 5] Button Press: reset[0] start[0] stop[0] return - 0
[ 6] Button Press: reset[0] start[0] stop[0] return - 0
[ 7] Button Press: reset[0] start[1] stop[1] return - 0
[ 8] Button Press: reset[0] start[0] stop[0] return - 0
[ 9] Button Press: reset[0] start[0] stop[0] return - 0
[10] Button Press: reset[0] start[1] stop[0] return - 0
[11] Button Press: reset[0] start[0] stop[0] return - 1
[12] Button Press: reset[1] start[1] stop[1] return - 2
[13] Button Press: reset[0] start[0] stop[0] return - 0
[14] Button Press: reset[0] start[1] stop[0] return - 0
[15] Button Press: reset[0] start[0] stop[1] return - 1
[16] Button Press: reset[0] start[0] stop[0] return - 1
[17] Button Press: reset[0] start[0] stop[0] return - 1
[18] Button Press: reset[0] start[0] stop[1] return - 1
[19] Button Press: reset[0] start[0] stop[0] return - 1
[20] Button Press: reset[0] start[1] stop[0] return - 1
[21] Button Press: reset[0] start[0] stop[0] return - 2
[22] Button Press: reset[0] start[0] stop[0] return - 3
[23] Button Press: reset[0] start[0] stop[0] return - 4
```

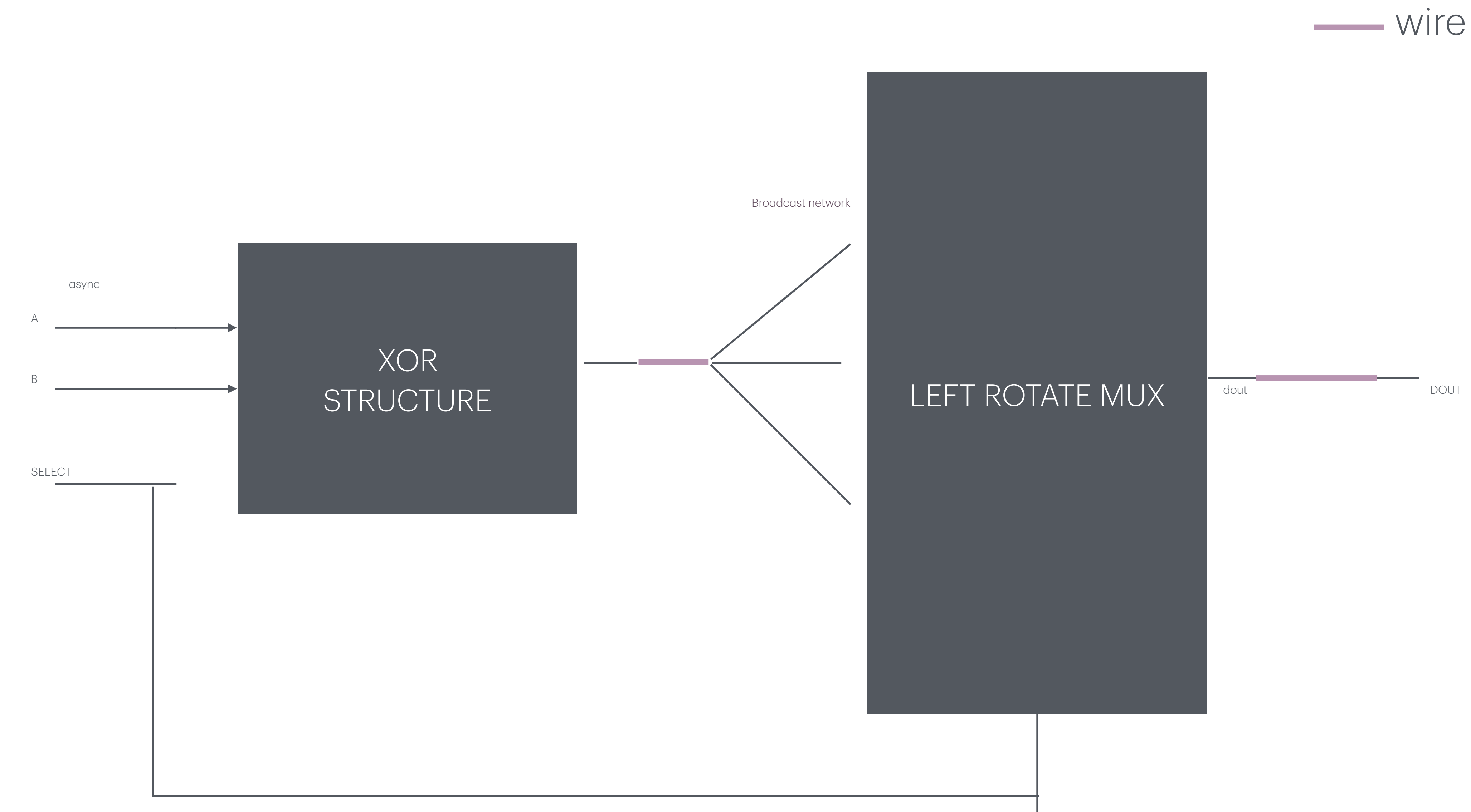
```
testbench.sv:136: $finish called at 114000 (1ps)
```

Done

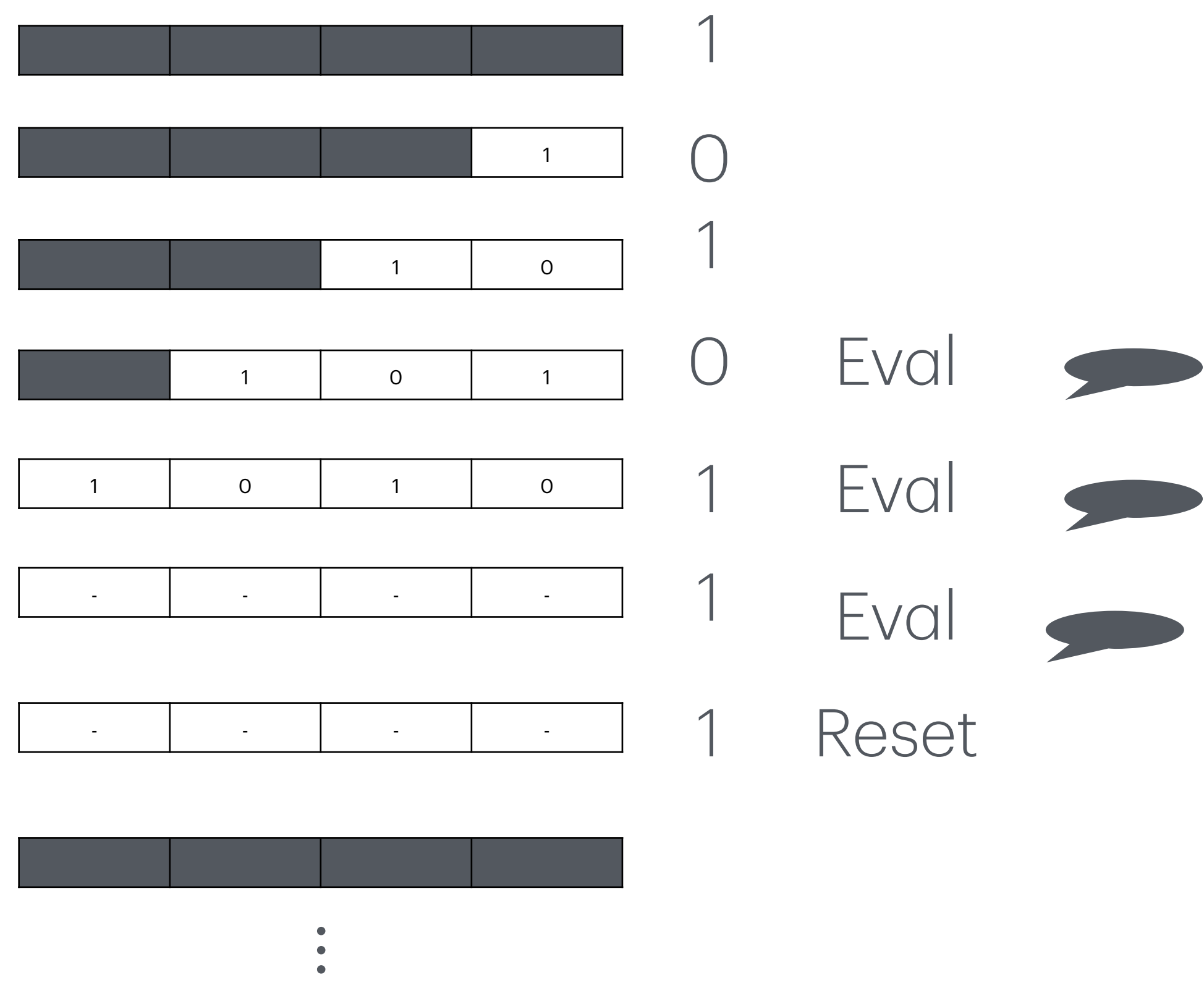


Circuit from OLD grad school slides :). Might as well build it in SV

Random Circuit 1

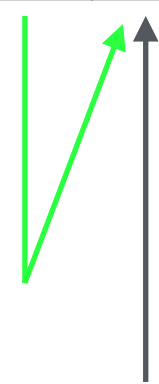


Sequence Detector



Sequence Detector

1	0	1	0	1	1	1	1	0	1	0	1	0	1	0	1	1	0	1	0	1	1	1	1	0	0	0	0	0	0	0		
NOP 0000	0001	0010	0101	1010	0101	1011	1111	1111	1110	1101	1010	0101	1010	0101	1010	0101	1011	0110	1101	1010	0101	1011	0111	1111	1110	1100	1000	0000	0000	0000	0000	0000



FIFO FULL

Evaluations of sequence are valid