

Sys-Verilog Questions Review

Some Solutions to questions from ChipIO-Dev

Hector “Hectron” Williams

Counter

clk

Bit

n_ticks

Logic [7:0]



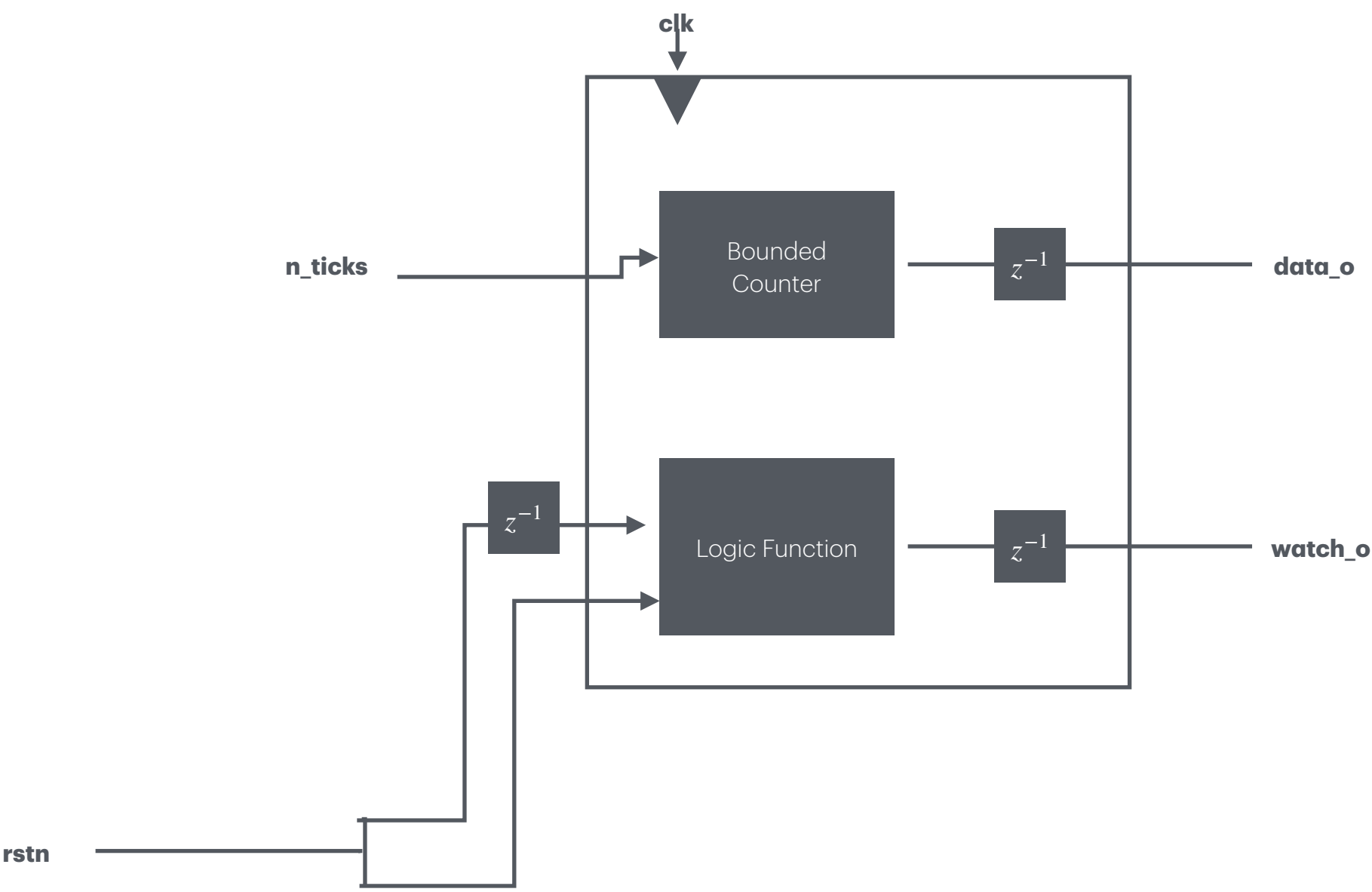
data_o

Logic [7:0]

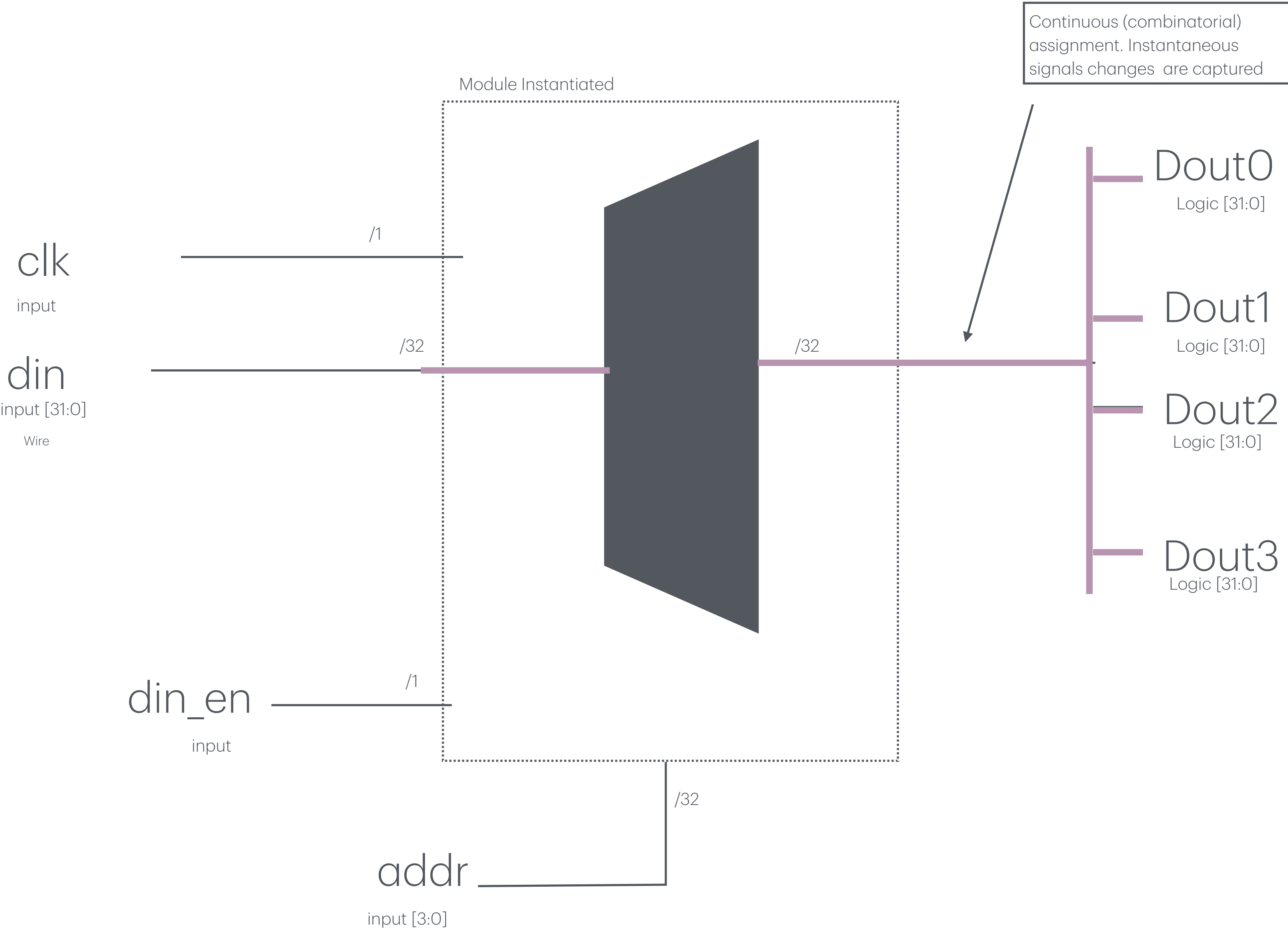
watch_o

Bit

Counter



Router



Connect (wire)



Log2

clk

Bit

X

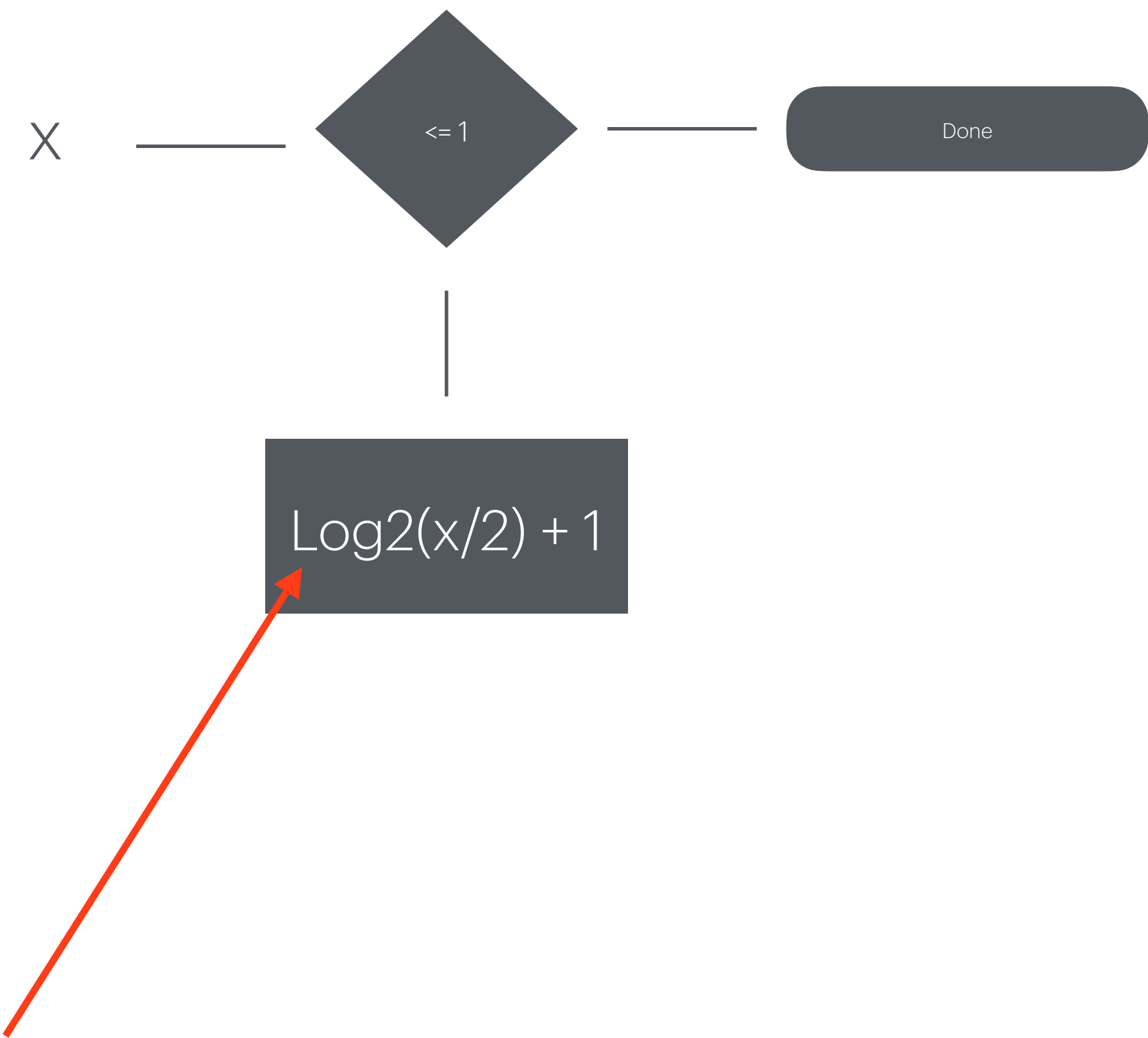
Logic [7:0]

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)

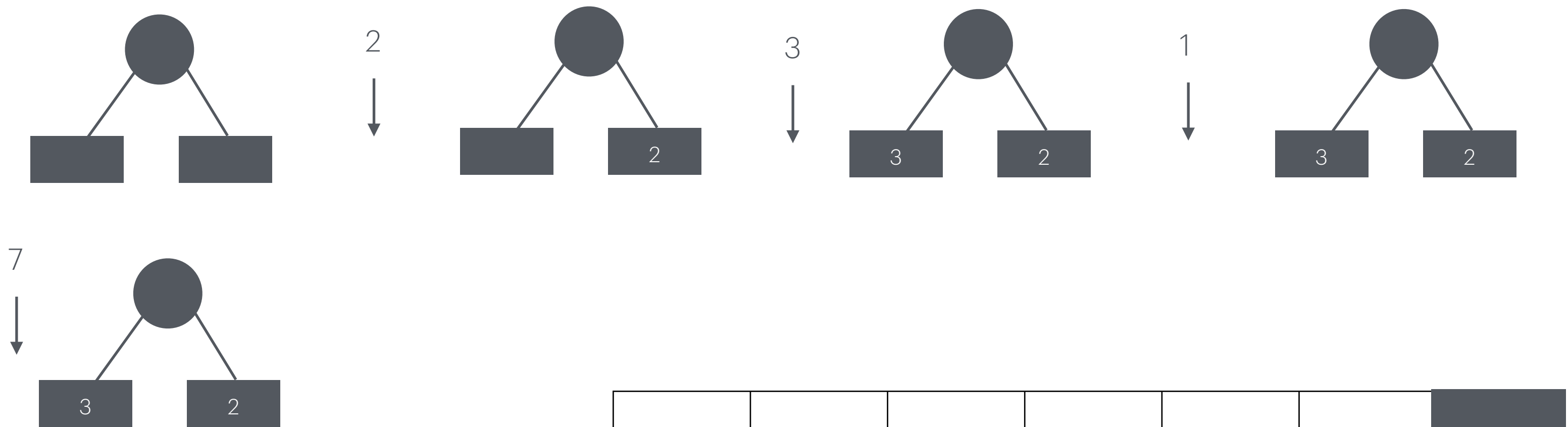
Log2 : Debug Results

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]

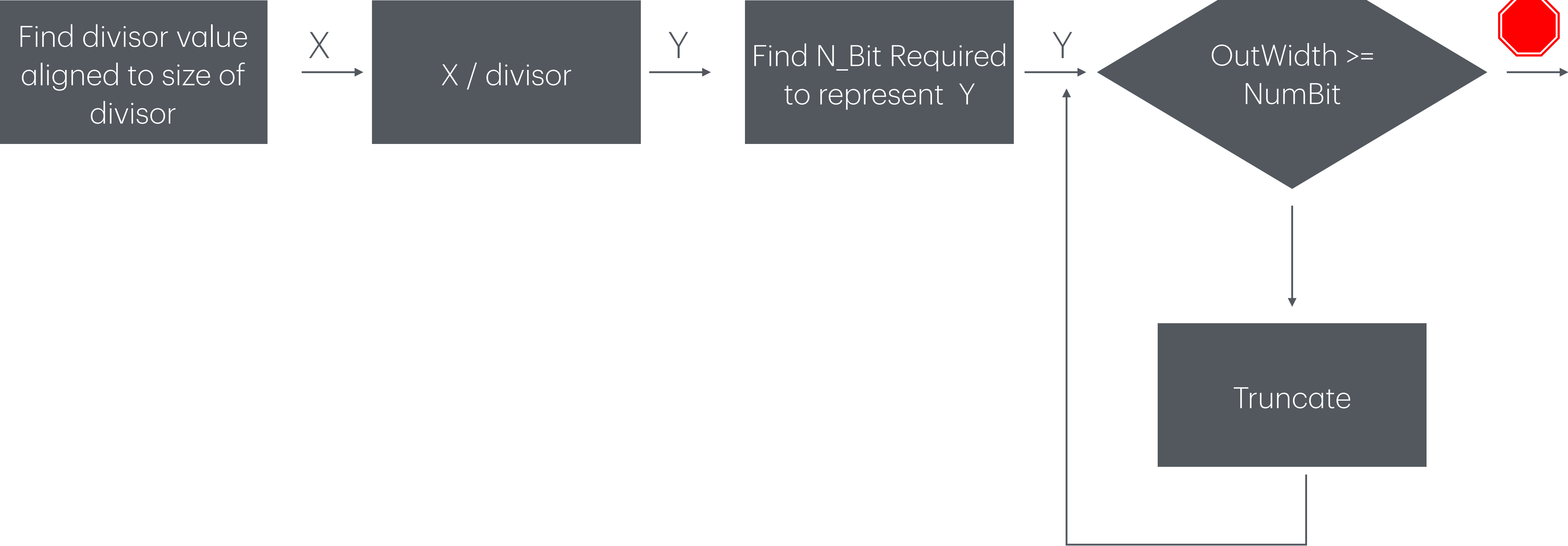
Second Largest



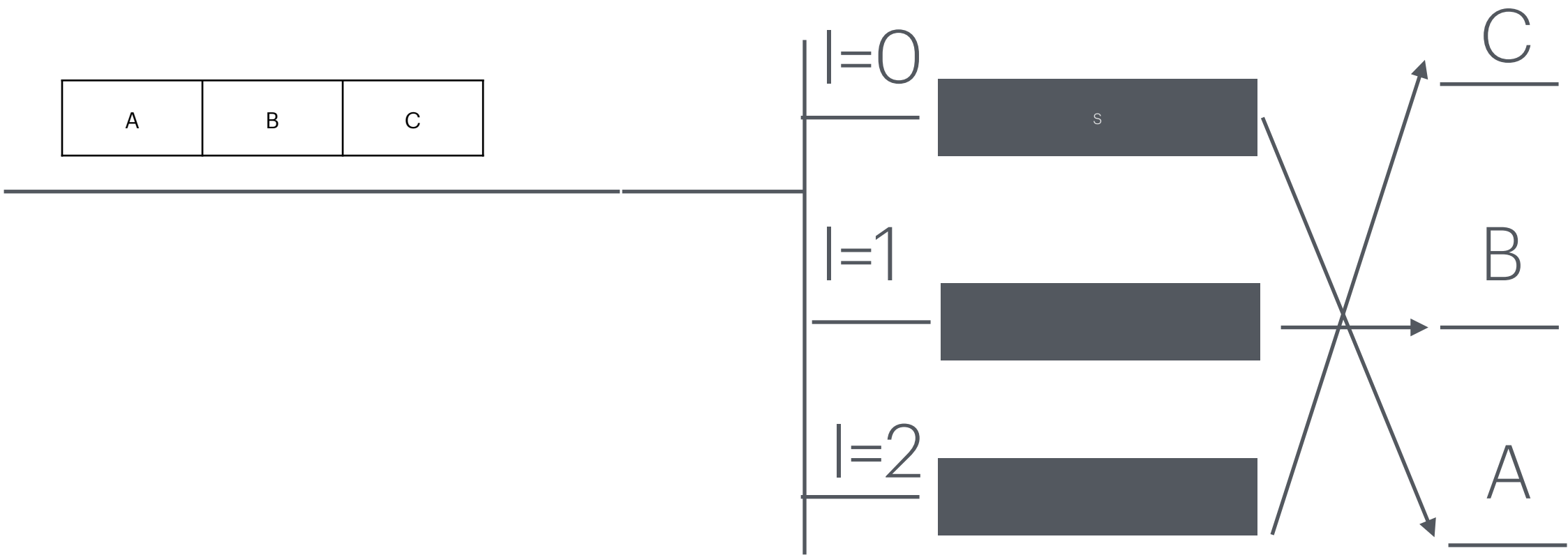
Count	0	1	2	3	4	
Data_In	D0	D1	D2	D3	D4	
2nd Largest	0	0	2	2	2	3

-, **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

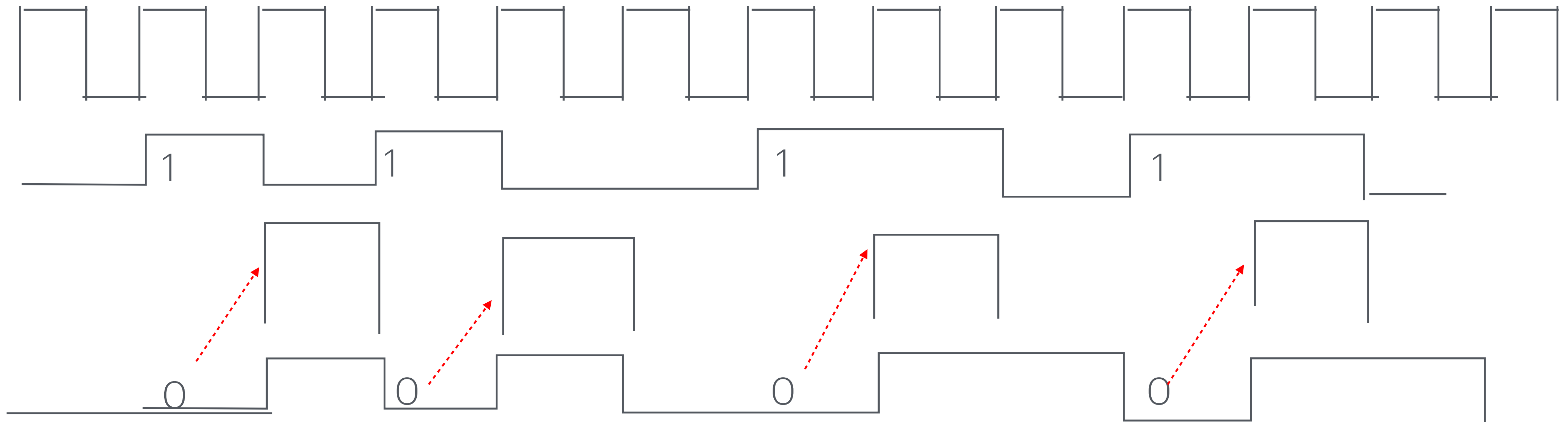
Gray code

Vertical Delay :)

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

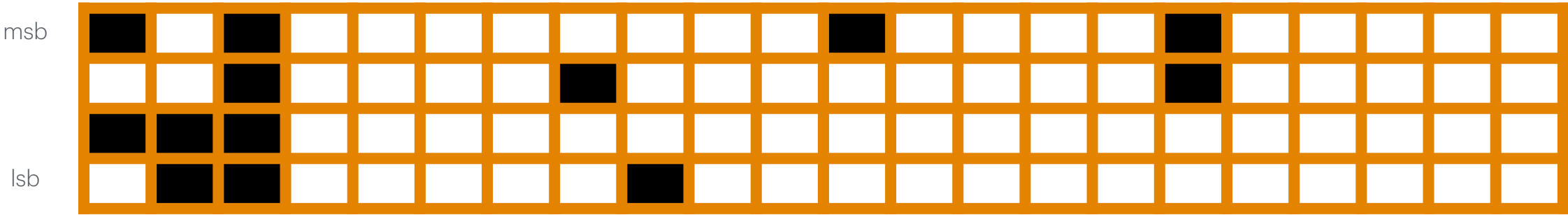
	0	0	0	0
	0	0	0	1
	0	0	1	1
	0	0	1	0
	0	1	1	0

Egde Detector



Parralel In -Serial Out

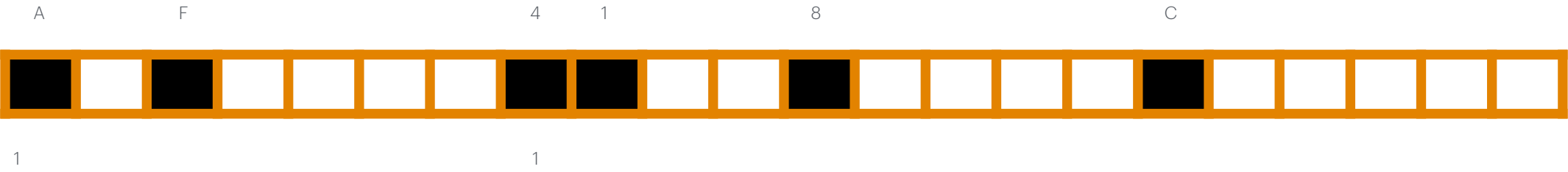
DATA



Bit Sequence



Enable



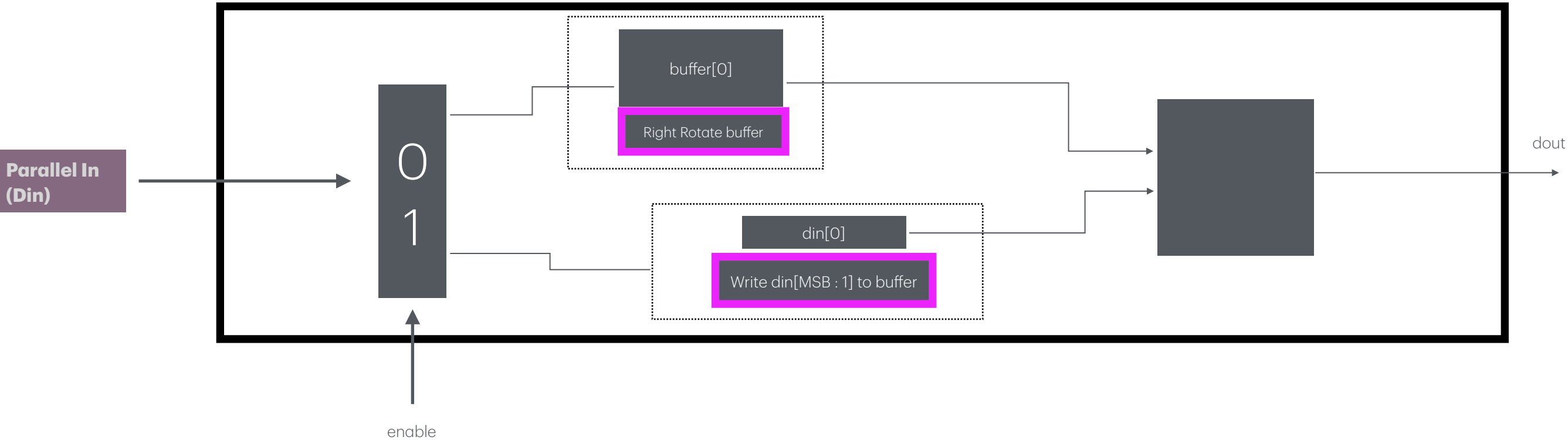
D_IN



RST



Output



Serial to Parallel

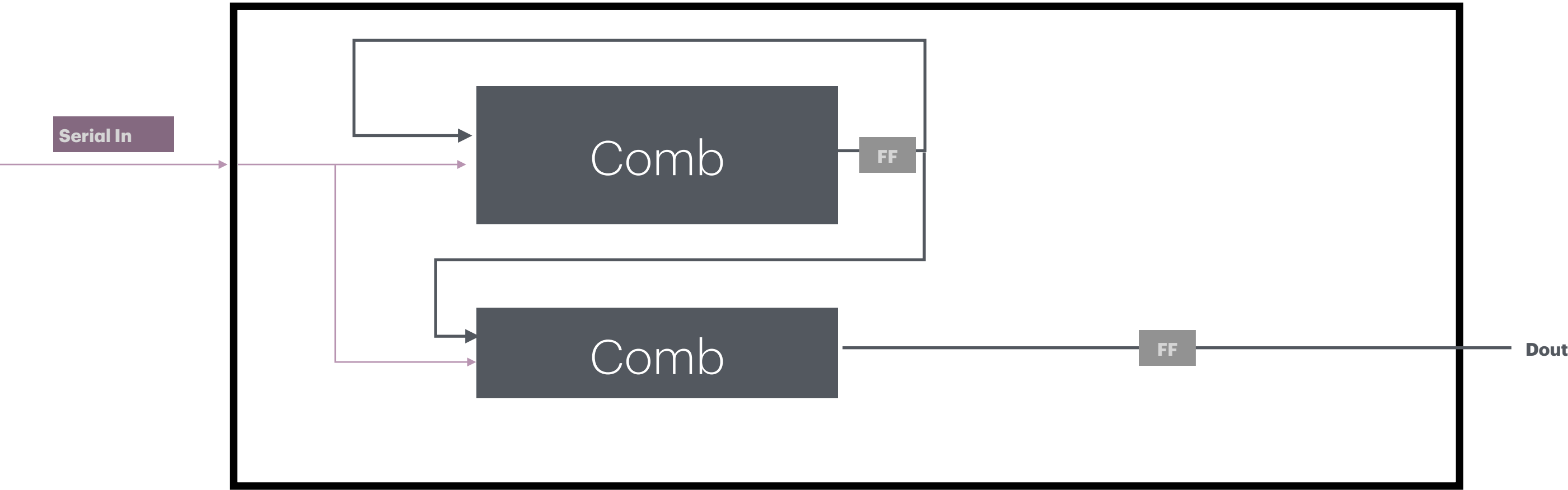
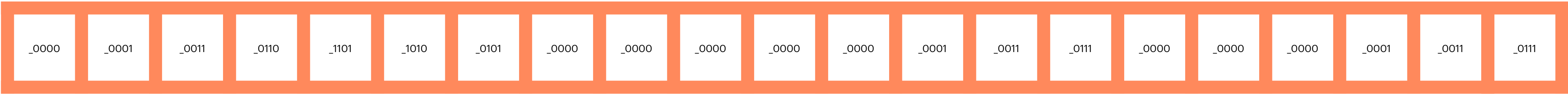
Din



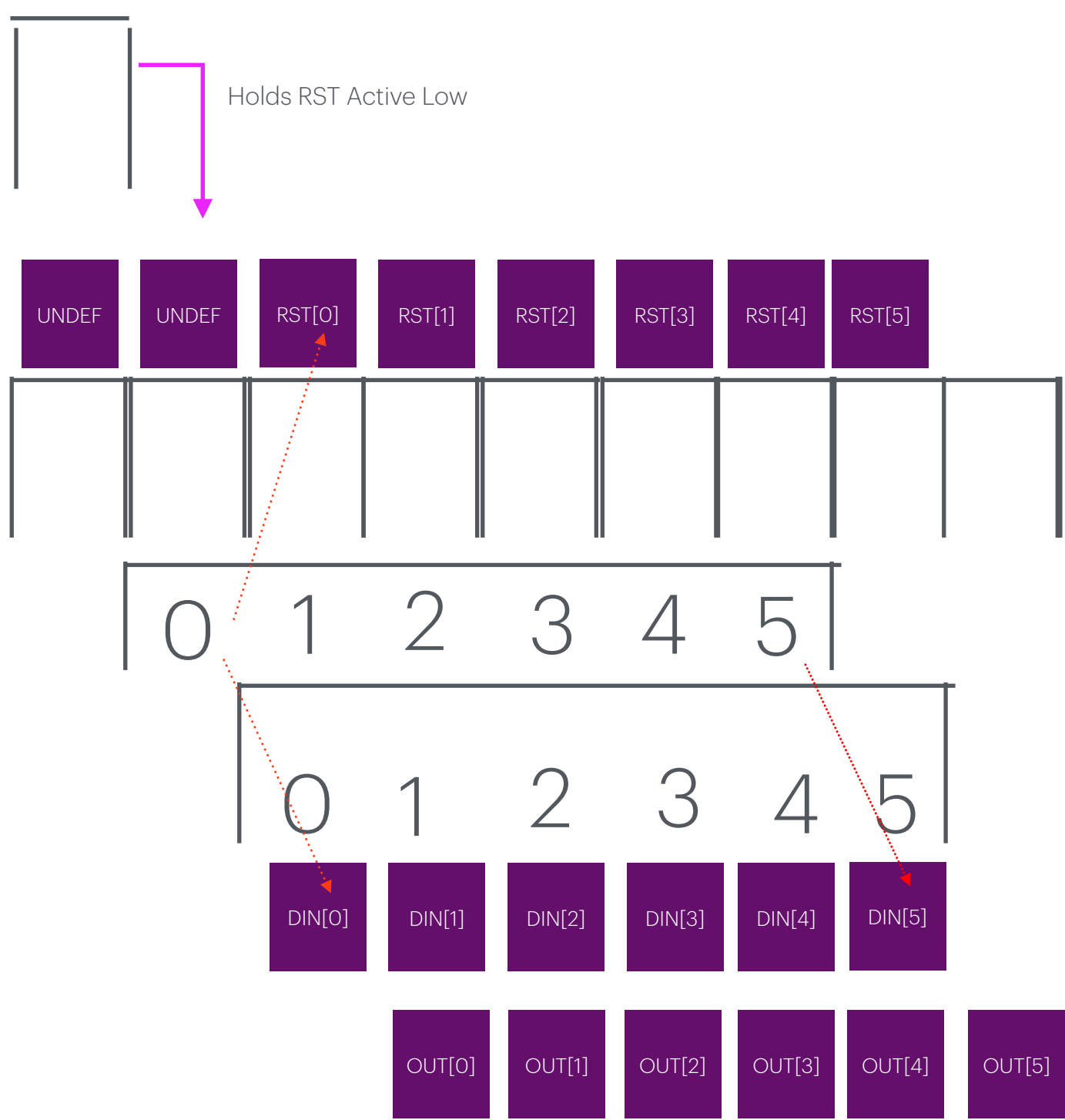
Reset



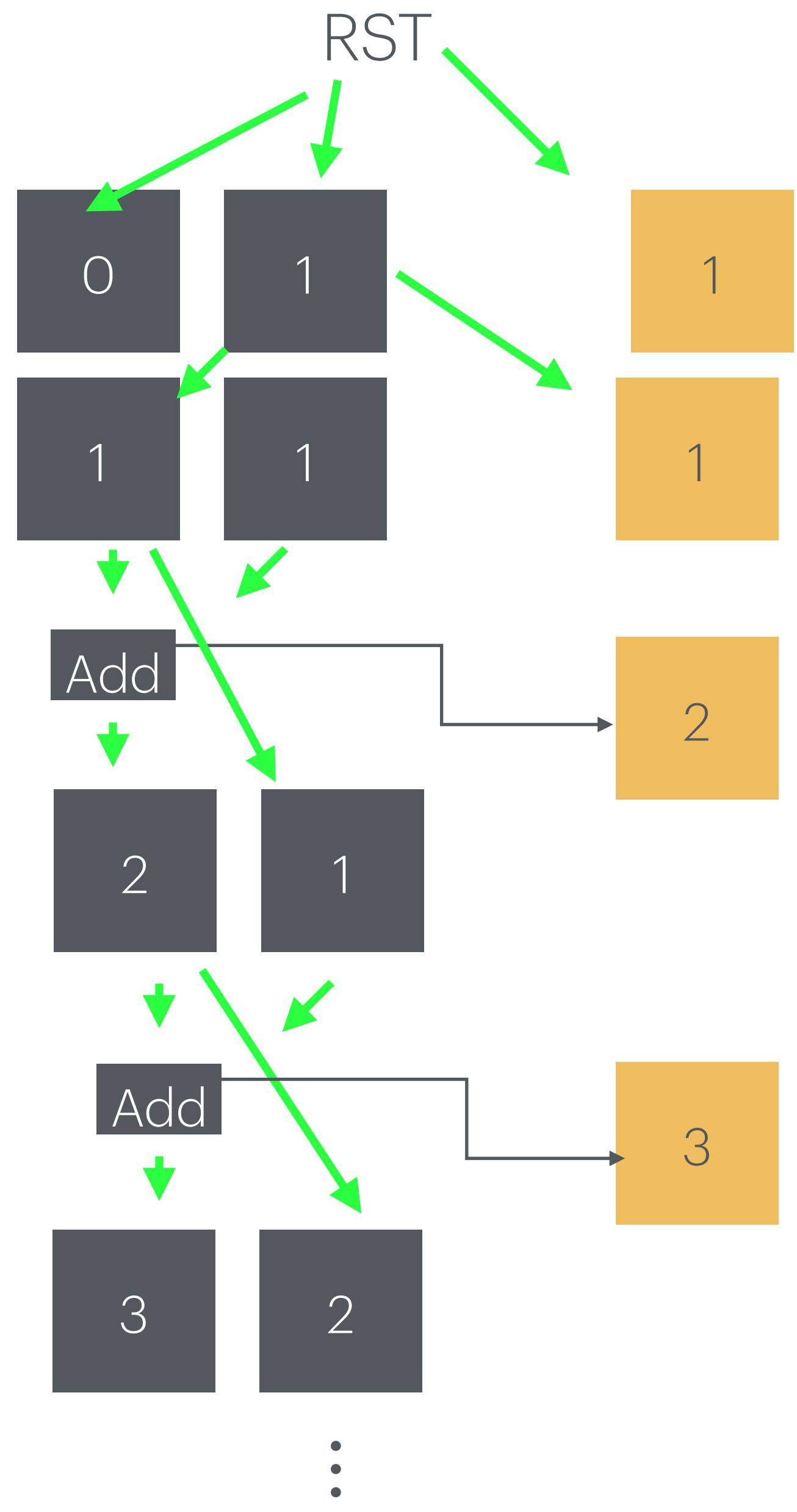
Data



Serial to Parallel (Simulation Concept)

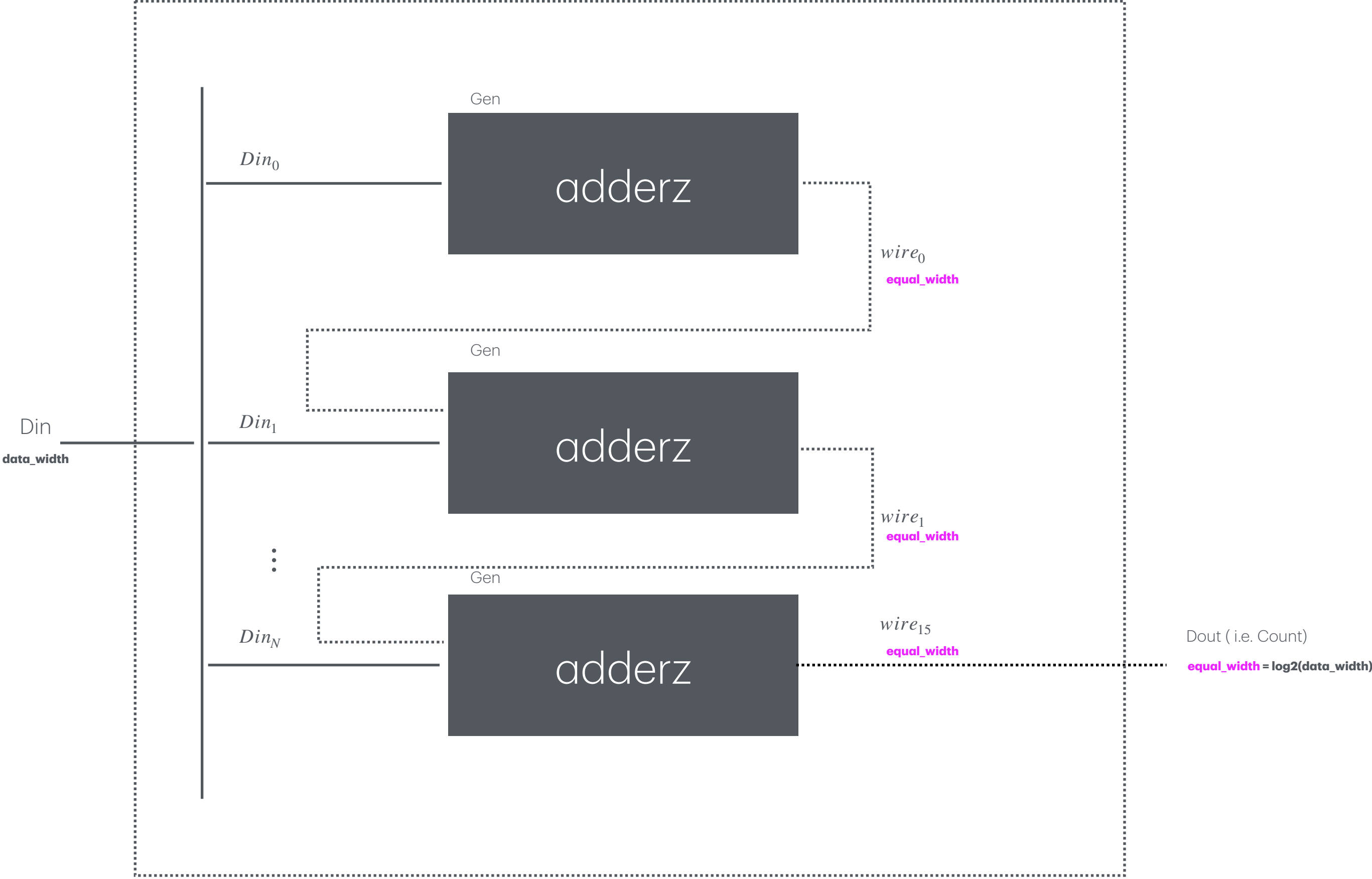


Fibonacci



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
```

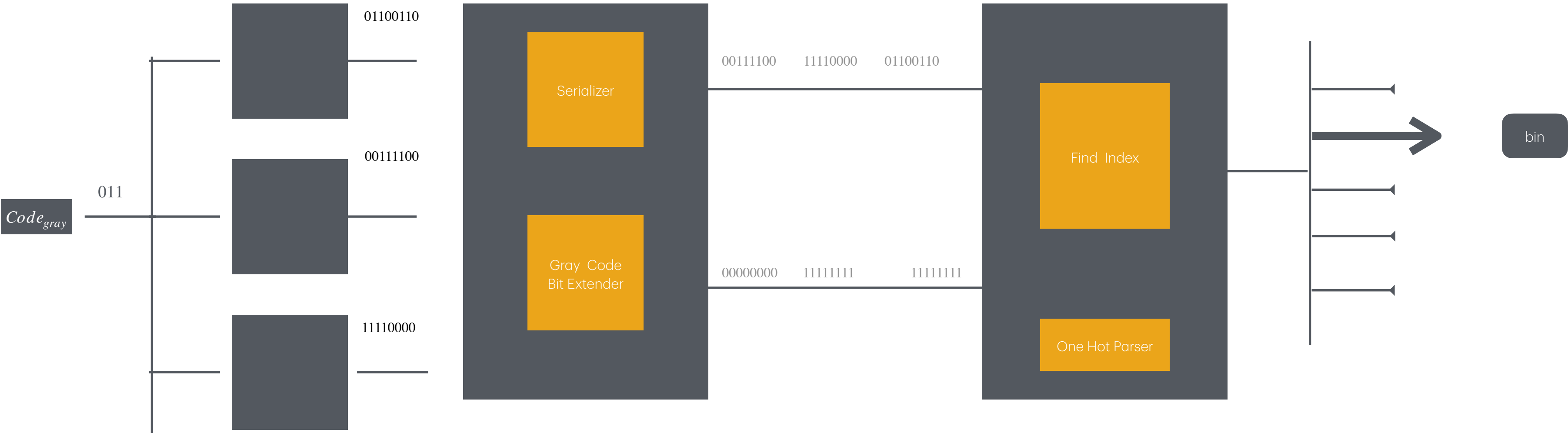
```
index - 1   input -    5   n_ones -  2
```

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

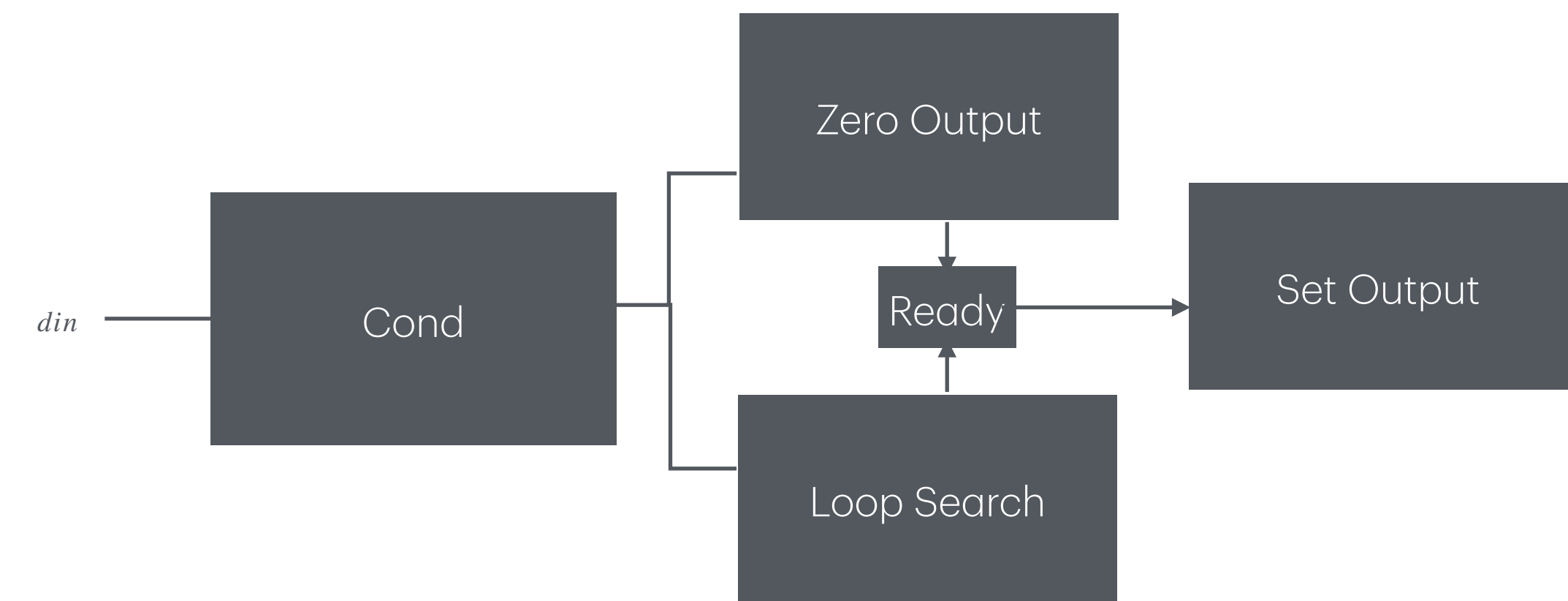
```
testbench.sv:44: $finish called at 9 (1s)
```

```
Done
```

Gray Code to Binary (Width = 3)



Trailing Ones



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

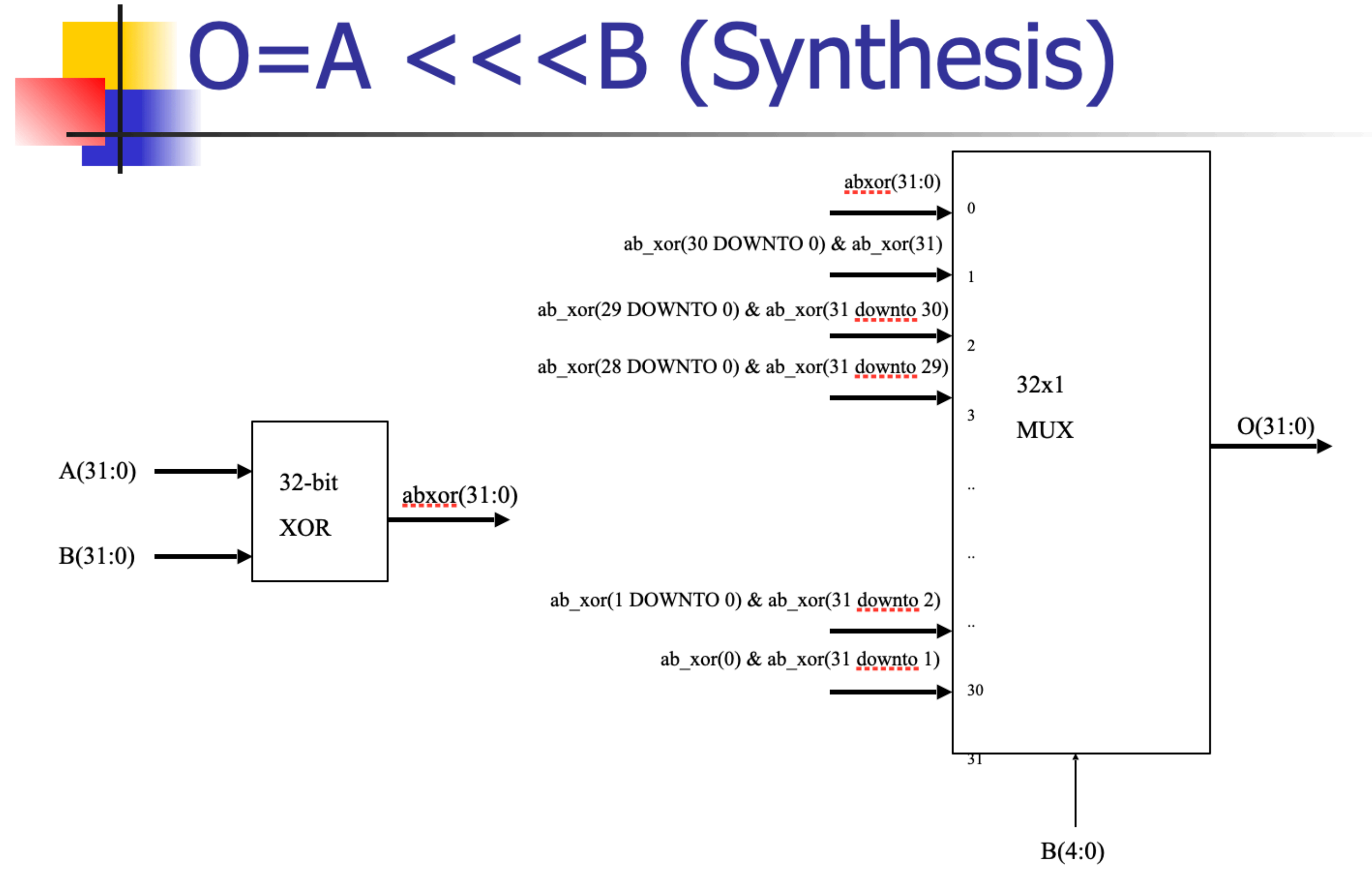
prev_button Event

```
[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
```

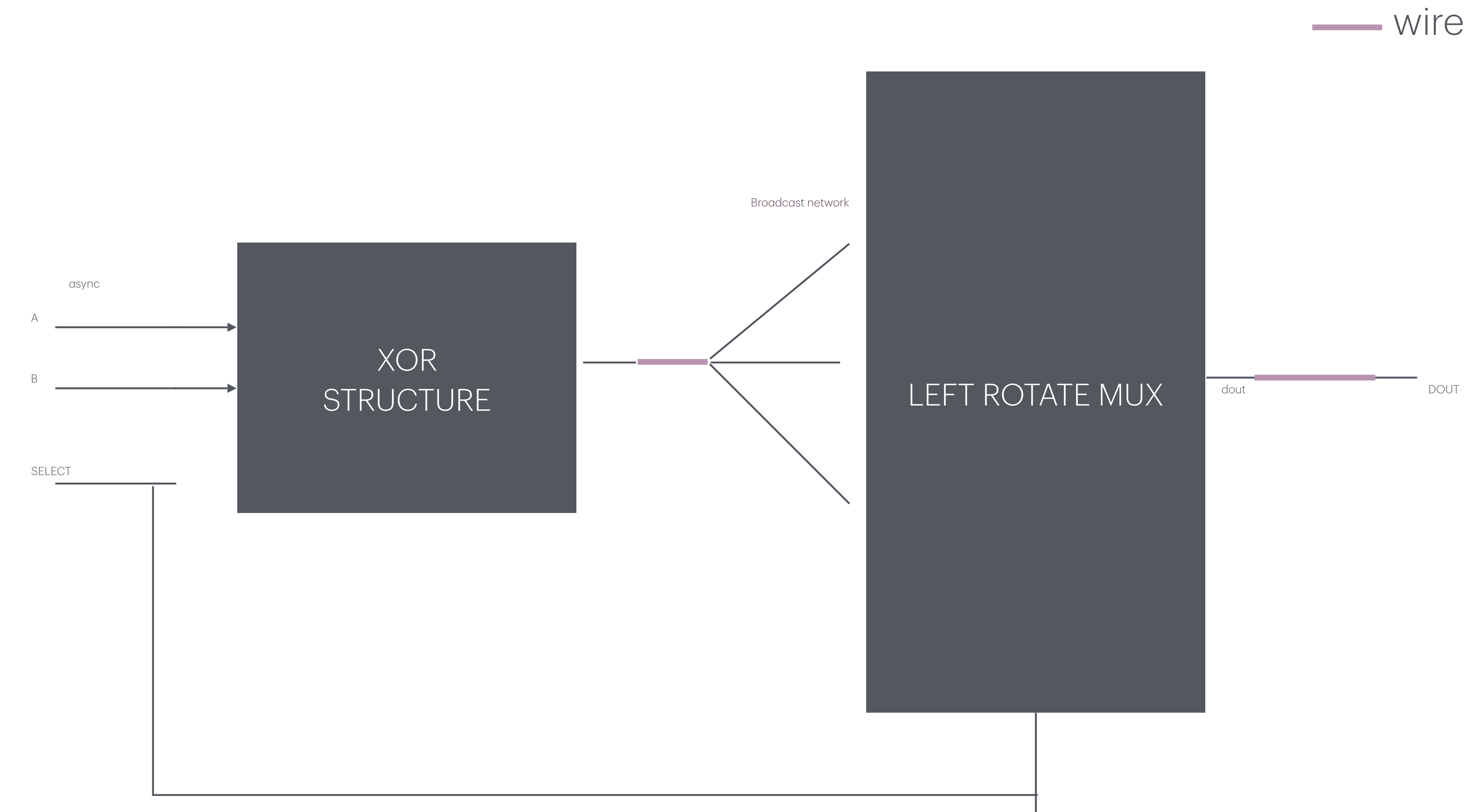
```
testbench.sv:136: $finish called at 114000 (1ps)
```

Done

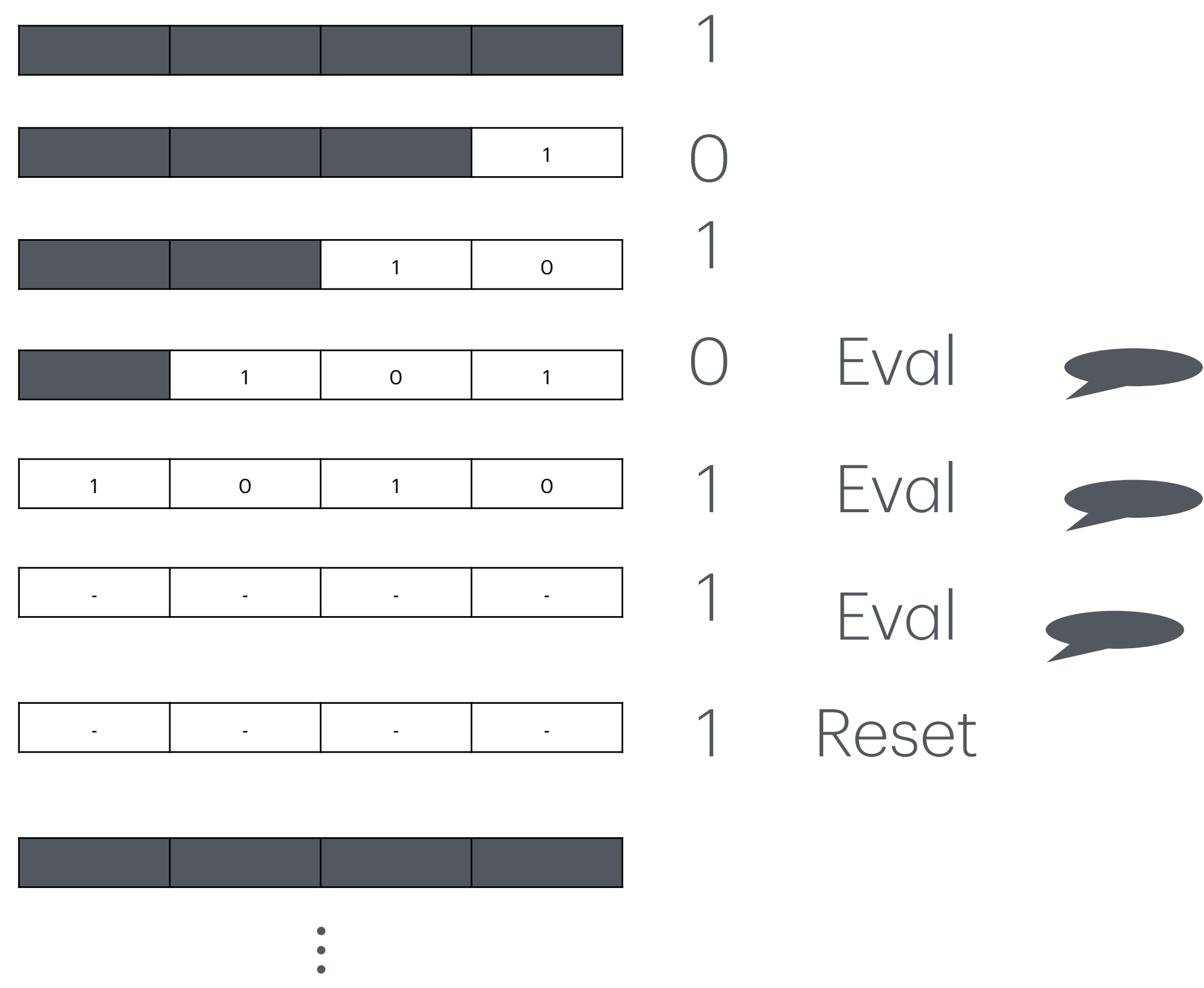


Circuit from OLD grad school slides :). Might as well build it in SV

Random Circuit 1

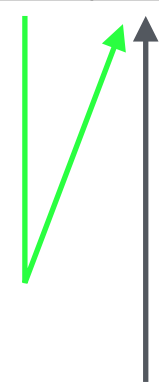


Sequence Detector



Sequence Detector

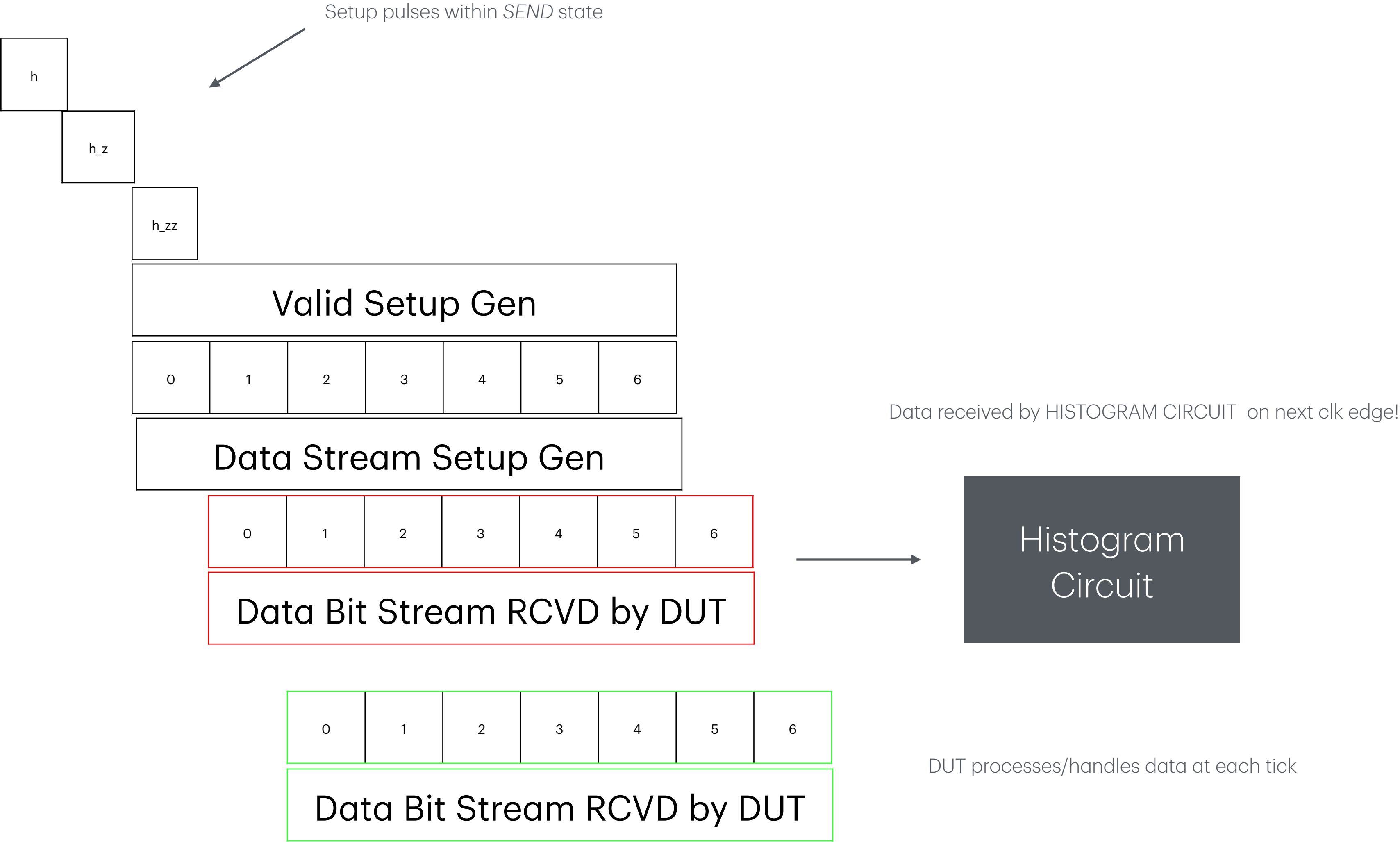
1	0	1	0	1	1	1	1	0	1	0	1	0	1	0	1	1	0	1	0	1	1	1	1	0	0	0	0	0	0	0		
NOP 0000	0001	0010	0101	1010	0101	1011	1111	1111	1110	1101	1010	0101	1010	0101	1010	0101	1011	0110	1101	1010	0101	1011	0111	1111	1110	1100	1000	0000	0000	0000	0000	0000



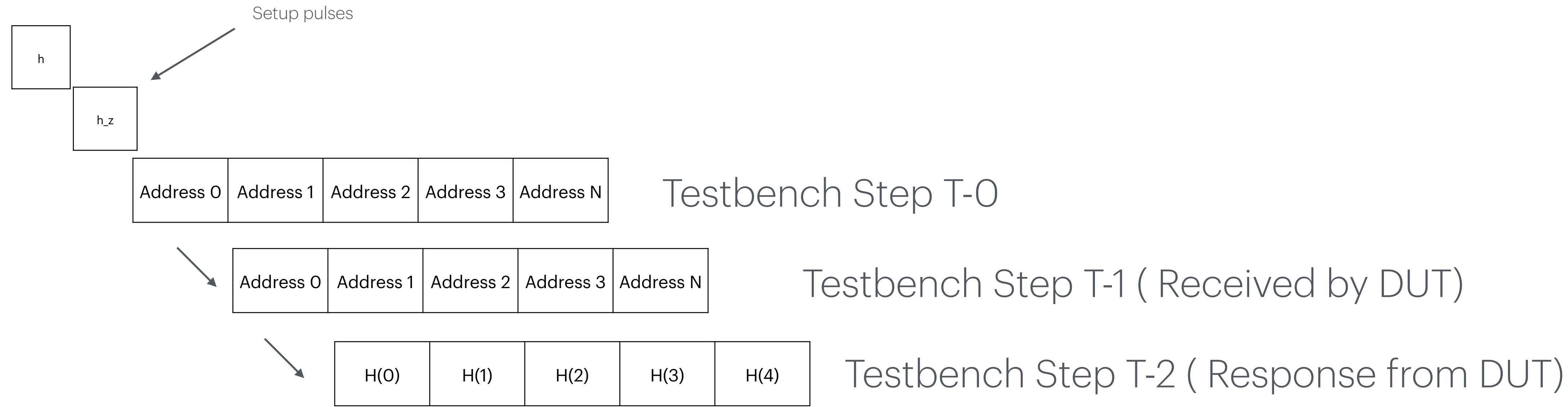
FIFO FULL

Evaluations of sequence are valid

Send Mechanism



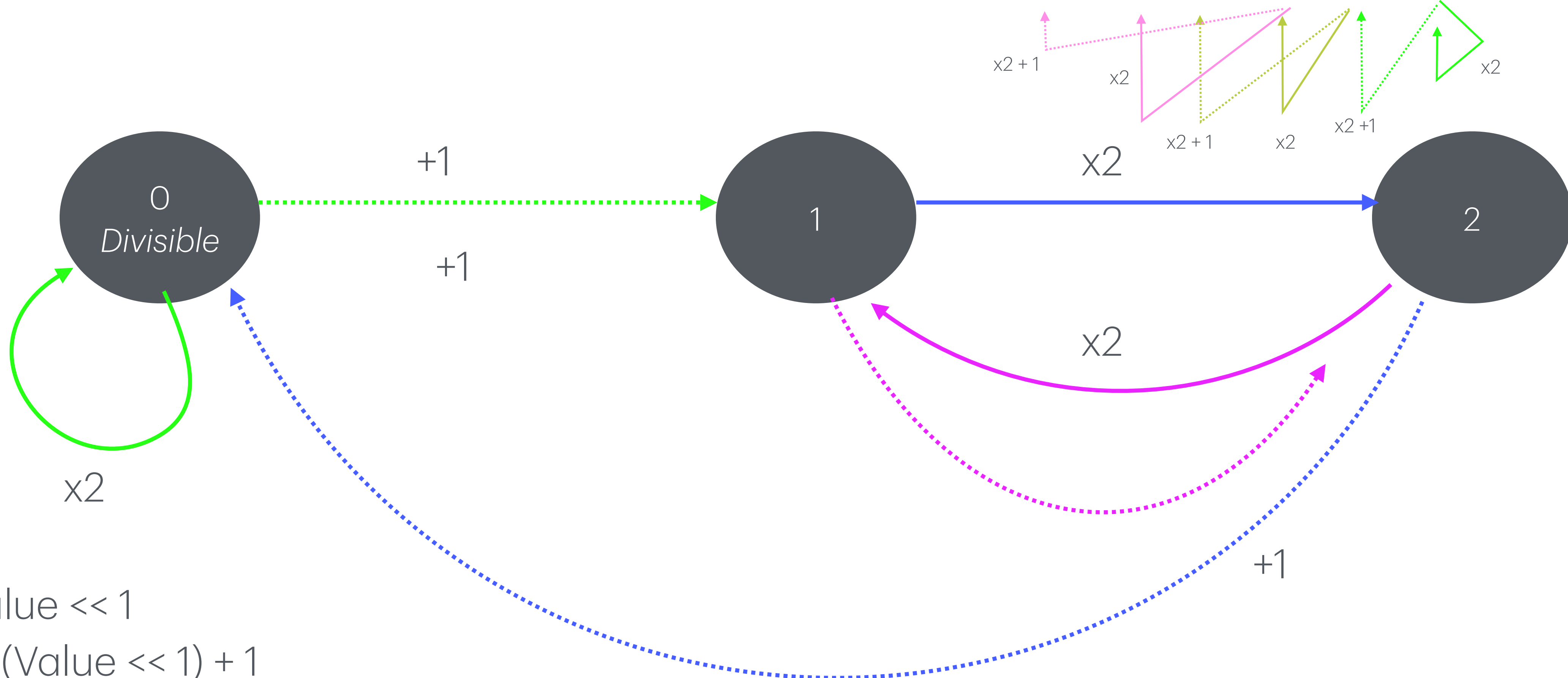
Receive Mechanism (Test Address)



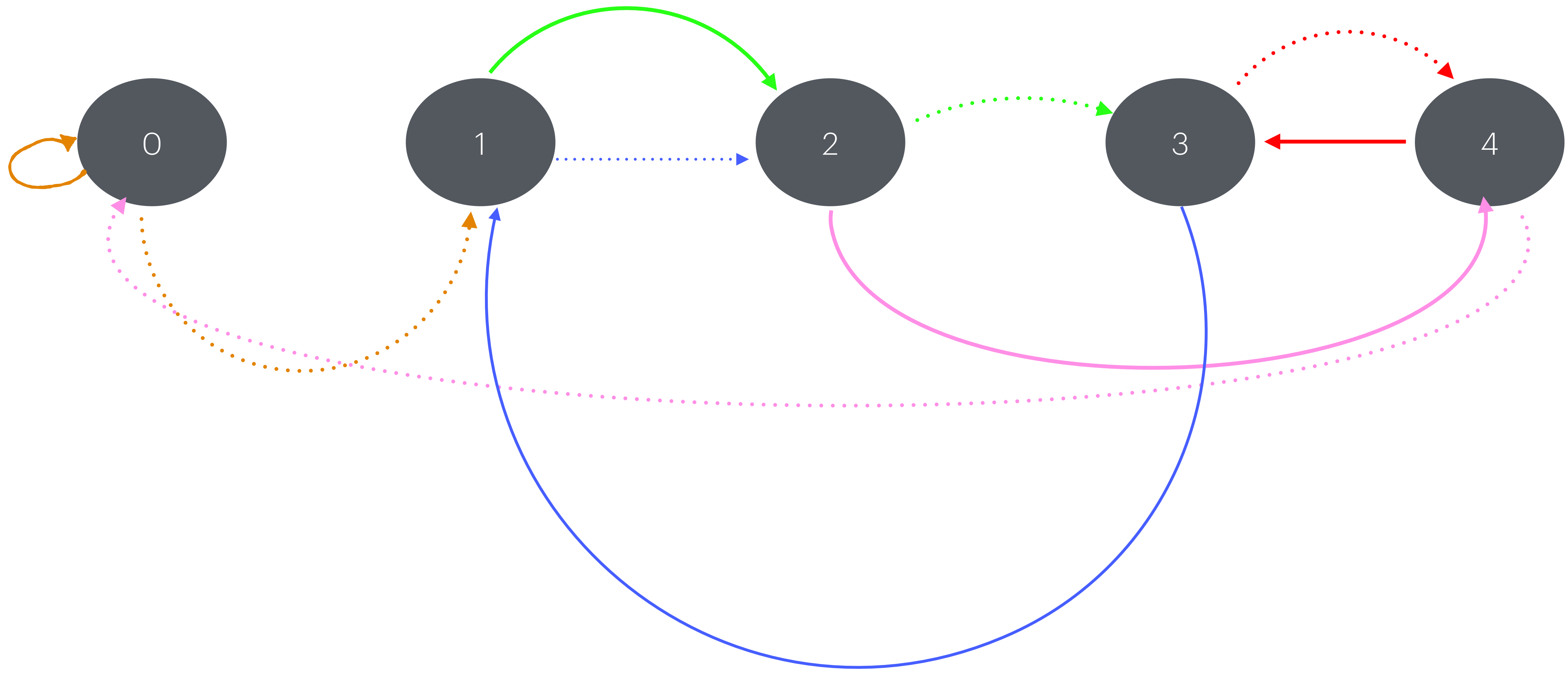
Divisible By Three

Index = 5			Index = 4			Index = 3			Index = 2			Index = 1			Index = 0		
2	1	0	2	1	0	2	1	0	2	1	0	2	1	0	2	1	0
17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

State Flow

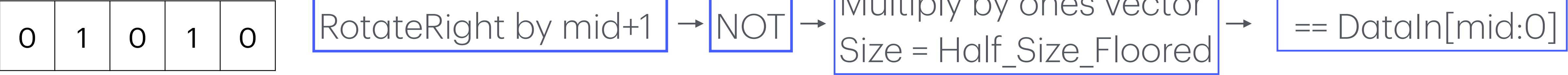


Divisible By Five



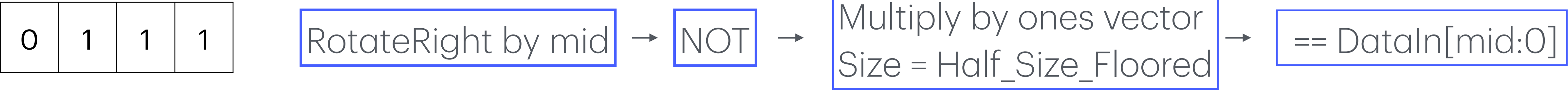
Palindrome

Odd Data Width

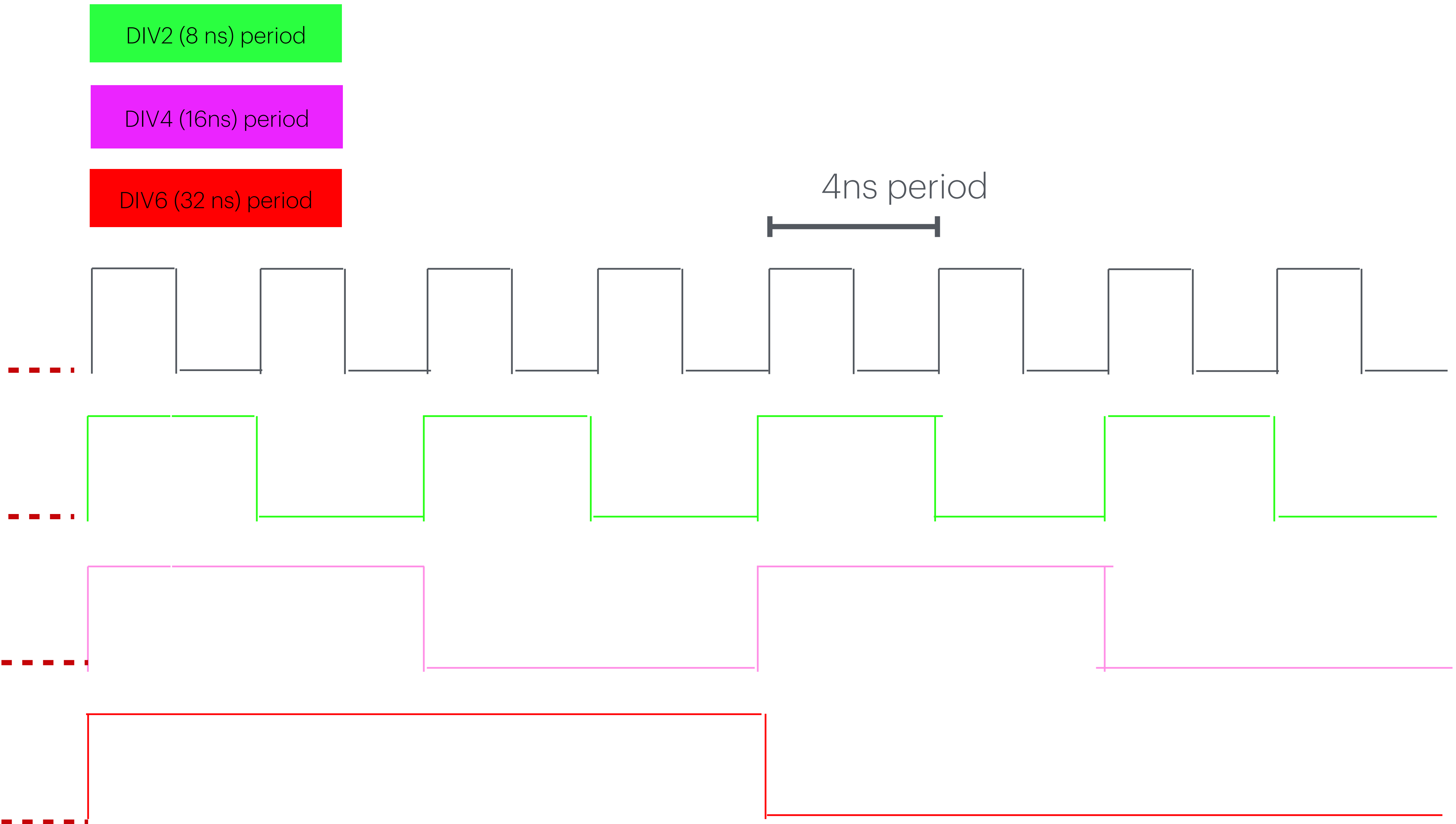


Half_Size_Floored = 2

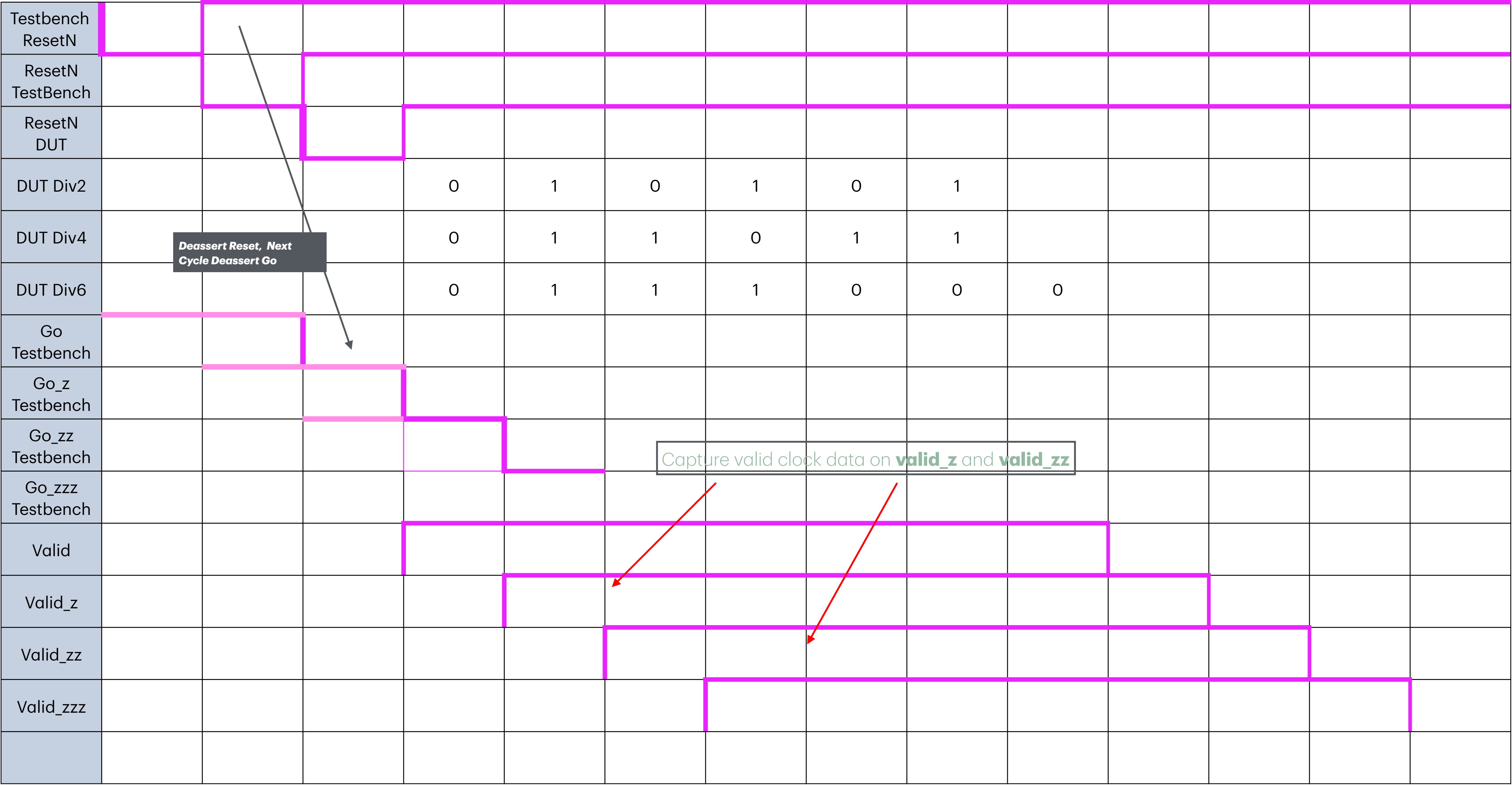
Even Data Width



Divide-By-Events



Divide-By-Events Timing



Testbench Log

```
TARGETS [11011] [11110]
Input 00001 Response 0
Input 00010 Response 0
Input 00101 Response 0
Input 01011 Response 0
Input 10111 Response 0
Input 01111 Response 0
Input 11111 Response 0
Input 11110 Response 1
Input 11101 Response 0
Input 11011 Response 0
testbench.sv:132: $finish called at 33 (1s)
Done
```

FizzBuzz

12	11	10	9	8	7	6	5	4	3	2	1	0	Tick
2	1	0	4 — 3		2	1	0	0 — 4		2	1	0 —	Fizz Mod
0 — 2		1	0 — 2		1	0 — 2		1	0 — 2		1	0 —	Buzz Mod

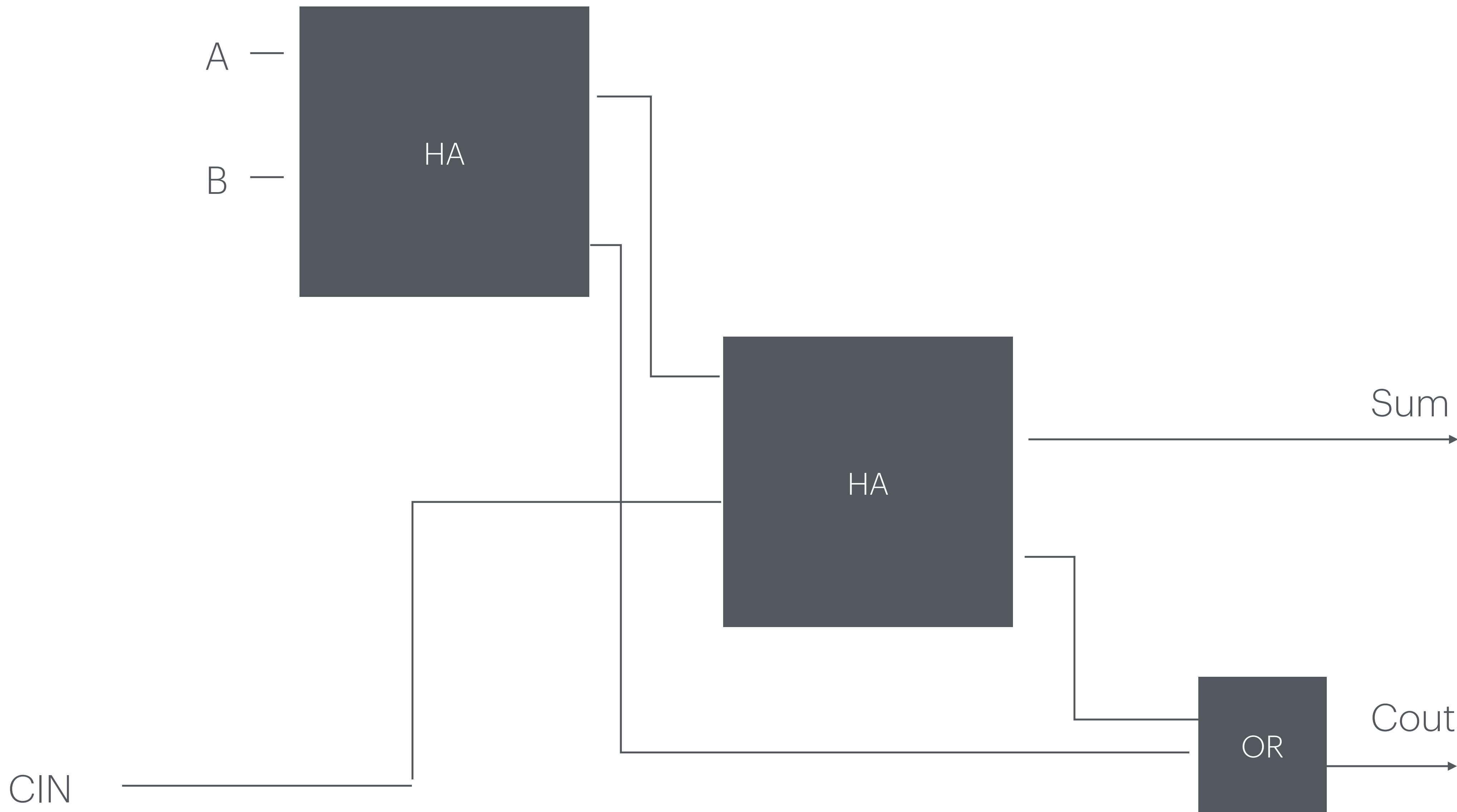
FizzBuzz

```
[2025-10-27 23:34:07 UTC] iverilog '-Wall' '-g2012' design.sv testbench.sv && unbuffer vvp a.out
```

```
Time 0  f1ss - 1. buzz - 1. f1zzbuzz 1
Time 1  f1ss - 0. buzz - 0. f1zzbuzz 0
Time 2  f1ss - 0. buzz - 0. f1zzbuzz 0
Time 3  f1ss - 1. buzz - 0. f1zzbuzz 0
Time 4  f1ss - 0. buzz - 0. f1zzbuzz 0
Time 5  f1ss - 0. buzz - 1. f1zzbuzz 0
Time 6  f1ss - 1. buzz - 0. f1zzbuzz 0
Time 7  f1ss - 0. buzz - 0. f1zzbuzz 0
Time 8  f1ss - 0. buzz - 0. f1zzbuzz 0
Time 9  f1ss - 1. buzz - 0. f1zzbuzz 0
Time 10 f1ss - 0. buzz - 1. f1zzbuzz 0
Time 11 f1ss - 0. buzz - 0. f1zzbuzz 0
Time 12 f1ss - 1. buzz - 0. f1zzbuzz 0
Time 13 f1ss - 0. buzz - 0. f1zzbuzz 0
Time 14 f1ss - 0. buzz - 0. f1zzbuzz 0
Time 15 f1ss - 1. buzz - 1. f1zzbuzz 1
Time 16 f1ss - 0. buzz - 0. f1zzbuzz 0
Time 17 f1ss - 0. buzz - 0. f1zzbuzz 0
Time 18 f1ss - 1. buzz - 0. f1zzbuzz 0
Time 19 f1ss - 0. buzz - 0. f1zzbuzz 0
Time 20 f1ss - 0. buzz - 1. f1zzbuzz 0
Time 21 f1ss - 1. buzz - 0. f1zzbuzz 0
Time 22 f1ss - 0. buzz - 0. f1zzbuzz 0
Time 23 f1ss - 0. buzz - 0. f1zzbuzz 0
Time 24 f1ss - 1. buzz - 0. f1zzbuzz 0
Time 25 f1ss - 0. buzz - 1. f1zzbuzz 0
Time 26 f1ss - 0. buzz - 0. f1zzbuzz 0
Time 27 f1ss - 1. buzz - 0. f1zzbuzz 0
Time 28 f1ss - 0. buzz - 0. f1zzbuzz 0
Time 29 f1ss - 0. buzz - 0. f1zzbuzz 0
testbench.sv:97: $finish called at 67 (1s)
```

Done

Full Adder



Full Adder

```
[2025-10-28 03:53:35 UTC] iverilog '-Wall' '-g2012' design.sv testbench.sv && unbuffer vvp a.out
```

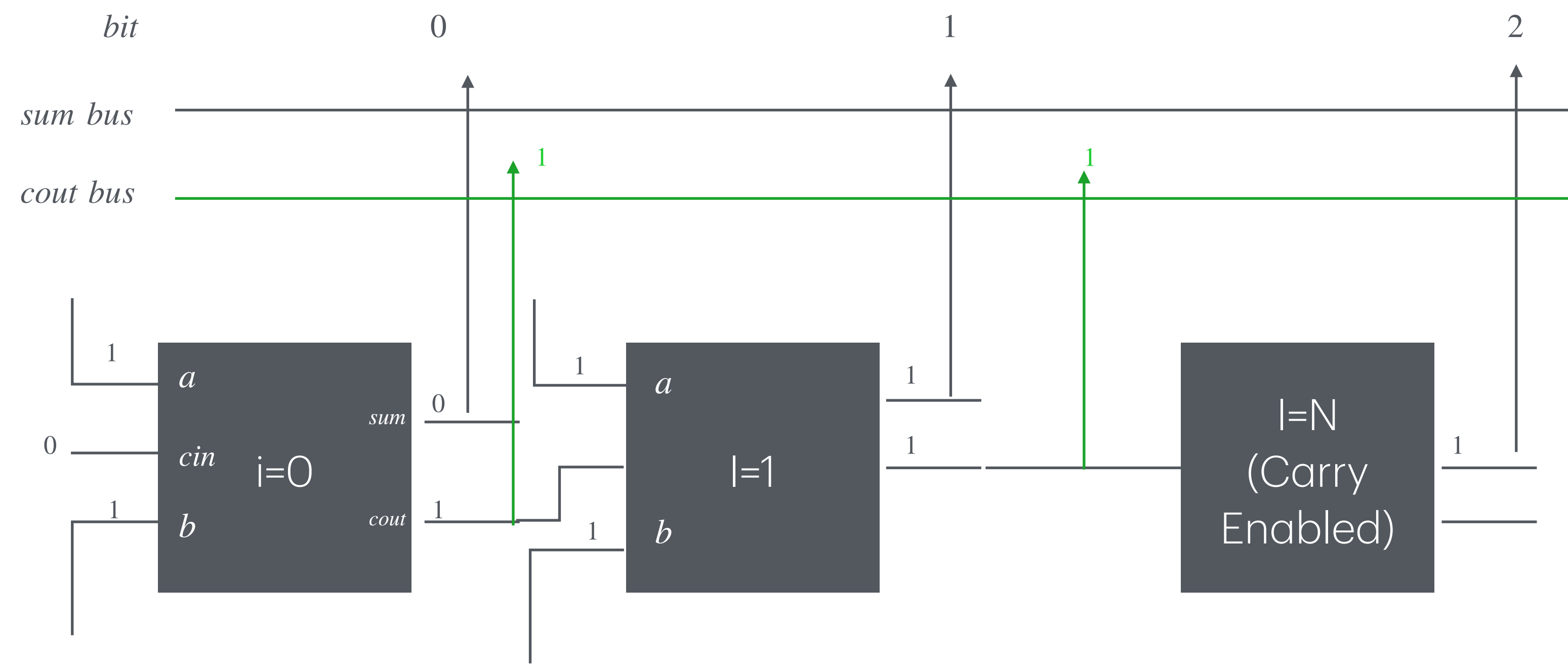
```
a - 0  b - 0. cin - 0 sum - 0  cout - 0
a - 0  b - 0. cin - 1 sum - 1  cout - 0
a - 0  b - 1. cin - 0 sum - 1  cout - 0
a - 0  b - 1. cin - 1 sum - 0  cout - 1
a - 1  b - 0. cin - 0 sum - 1  cout - 0
a - 1  b - 0. cin - 1 sum - 0  cout - 1
a - 1  b - 1. cin - 0 sum - 0  cout - 1
a - 1  b - 1. cin - 1 sum - 1  cout - 1
```

Done

Ripple Adder

A = 2b'11

B = 2b'11



1
1 1
1 1
—
0

1 1
1 1
1 1
—
1 0

1 1
1 1
1 1
—
1 1 0

Ripple Adder Testbench Logs

```
[2025-10-28 05:50:07 UTC] iverilog '-Wall' '-g2012' design.sv testbench.sv  && unbuffer vvp a.out
```

```
a -   3 b -   3  sum 000000110 cout 00000011
a -   7 b -  10  sum 000010001( 17) cout 00001110
a - 123 b - 189  sum 100111000(312) cout 11111111
```

Done

Time Step	0	1	2	3	4	5	6	7	8	9	10	11	
DIN	0	A	A	A									
ADDR		1	2	2	0	0	3	4	5	6	6	7	
WR		1		1									
RD		1	1		1		1	1	1	1	1	1	
RESETN													
DOUT						A	A	A	A	A	A	A	A
ERRR			1	1		1	0	1	1	1	1	1	1

Previous valid read
persists on dout
port

```
[2025-10-28 18:35:06 UTC] iverilog '-Wall' '-g2012' design.sv testbench.sv && unbuffer vvp a.out
```

```
Time -    0 dout -    0 error - 0
```

```
Time -    1 dout -    0 error - 1
```

```
Time -    2 dout -    0 error - 1
```

```
Time -    3 dout -    0 error - 0
```

```
Time -    4 dout -    0 error - 1
```

```
Time -    5 dout -    0 error - 0
```

```
Time -    6 dout -    0 error - 1
```

```
Time -    7 dout -    0 error - 1
```

```
Time -    8 dout -    0 error - 1
```

```
Time -    9 dout -    0 error - 1
```

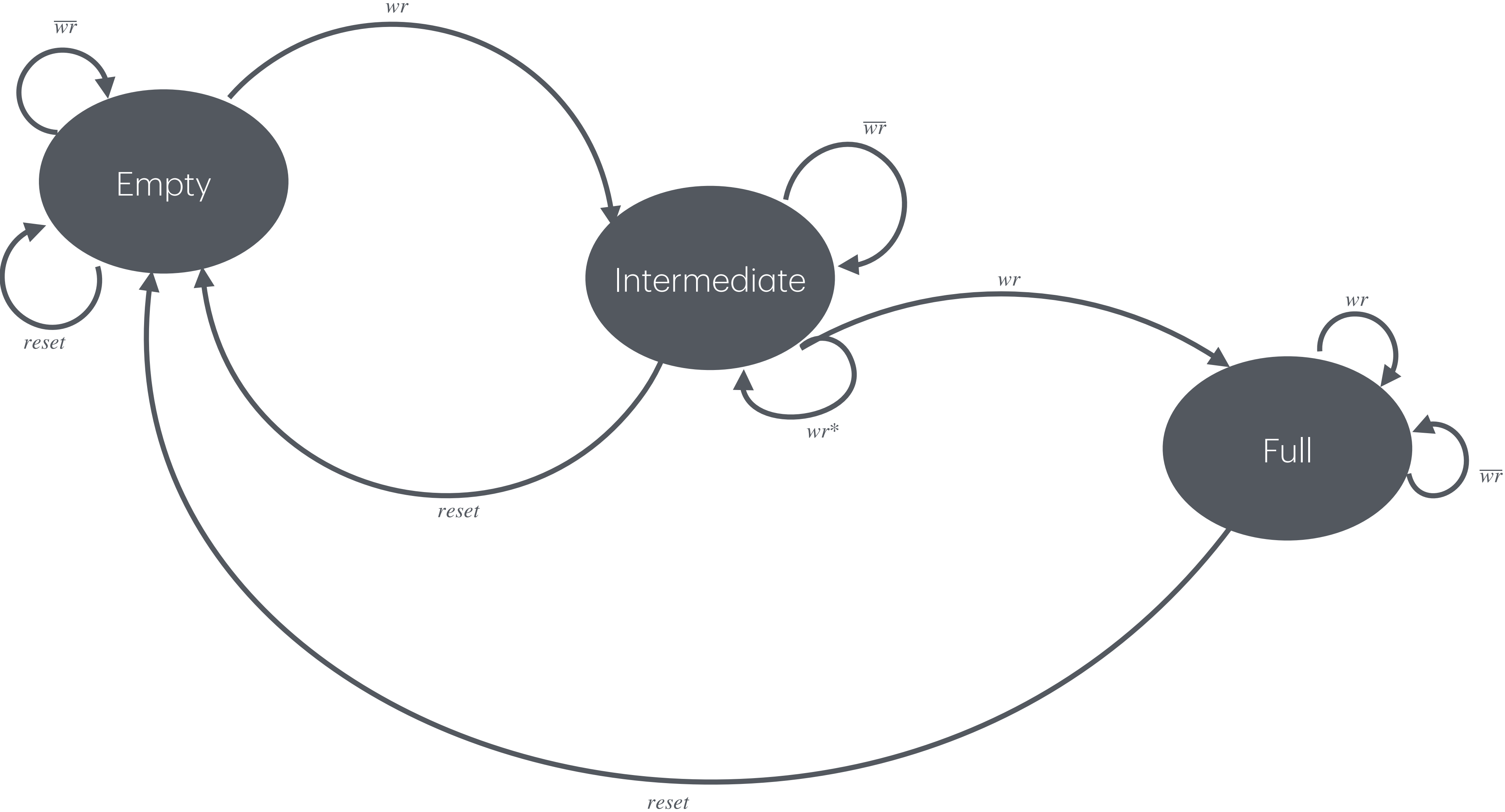
```
Time -   10 dout -    0 error - 1
```

```
Time -   11 dout -    0 error - 1
```

```
testbench.sv:185: $finish called at 35 (1s)
```

```
Done
```

MULTI BIT FIFO

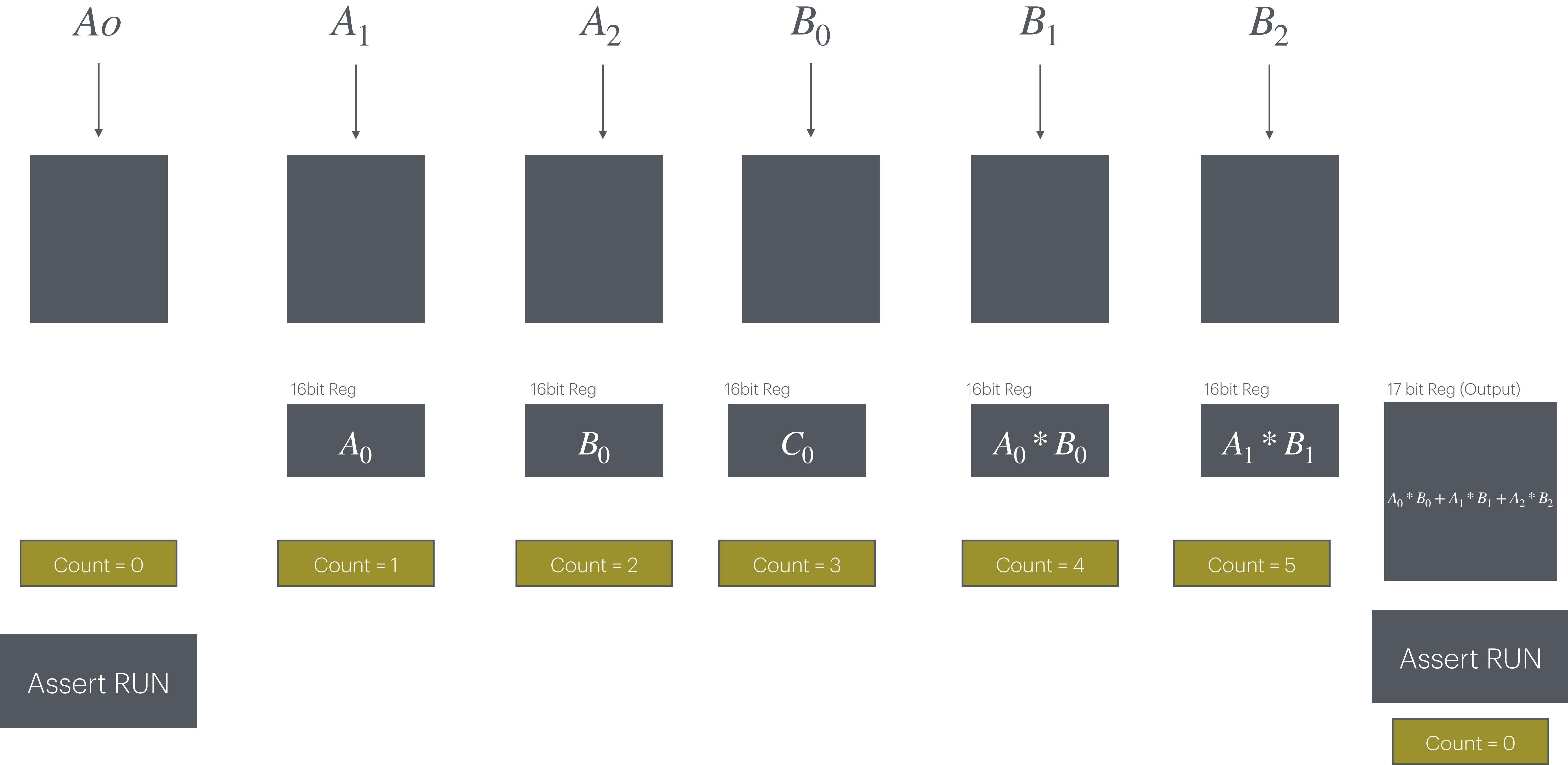


MULTI BIT FIFO

Time Step	0	1	2	3	4	5	6	
DIN	0	5	3	6	6	0	0	
WR	0	1	1	1	1	0	0	
DOUT	-	0	5	5	3	6	6	0
FULL	-	0	0	1	1	1	1	1
EMPTY	-	1	0	1	1	1	1	1

The diagram illustrates the operation of a Multi-Bit FIFO. Red arrows show the flow of data from the DIN (Data In) row to the DOUT (Data Out) row. The arrows indicate that data is read out from the FIFO at a later time step than it was received, demonstrating the first-in, first-out behavior. For example, the first '0' in DIN at time step 0 is read out at time step 1. The last '0' in DIN at time step 6 is read out at time step 7.

Dot Product



Dot Product

Rstn	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	
Din	16h0	16h0	16h0	16h0	16h0	16h1	16h2	16h3	16h4	16h5	16h6	16h7	16h8	16h9	16hA	16hB	16hC	16hd	
Dout	-	0	0	0	0	0	0	0	0	0	0	16h20	16h20	16h20	16h20	16h20	16h20	16h10A	16h10A
Run	-	1	1	1	1	1	0	0	0	0	0	1	0	0	0	0	0	1	0
Internal Counter	-	-	-	0	0	0	1	2	3	4	5	0	1	2	3	4	5	0	

Counter zeroed

Counters running

Binary To Thermometer

0000_0000

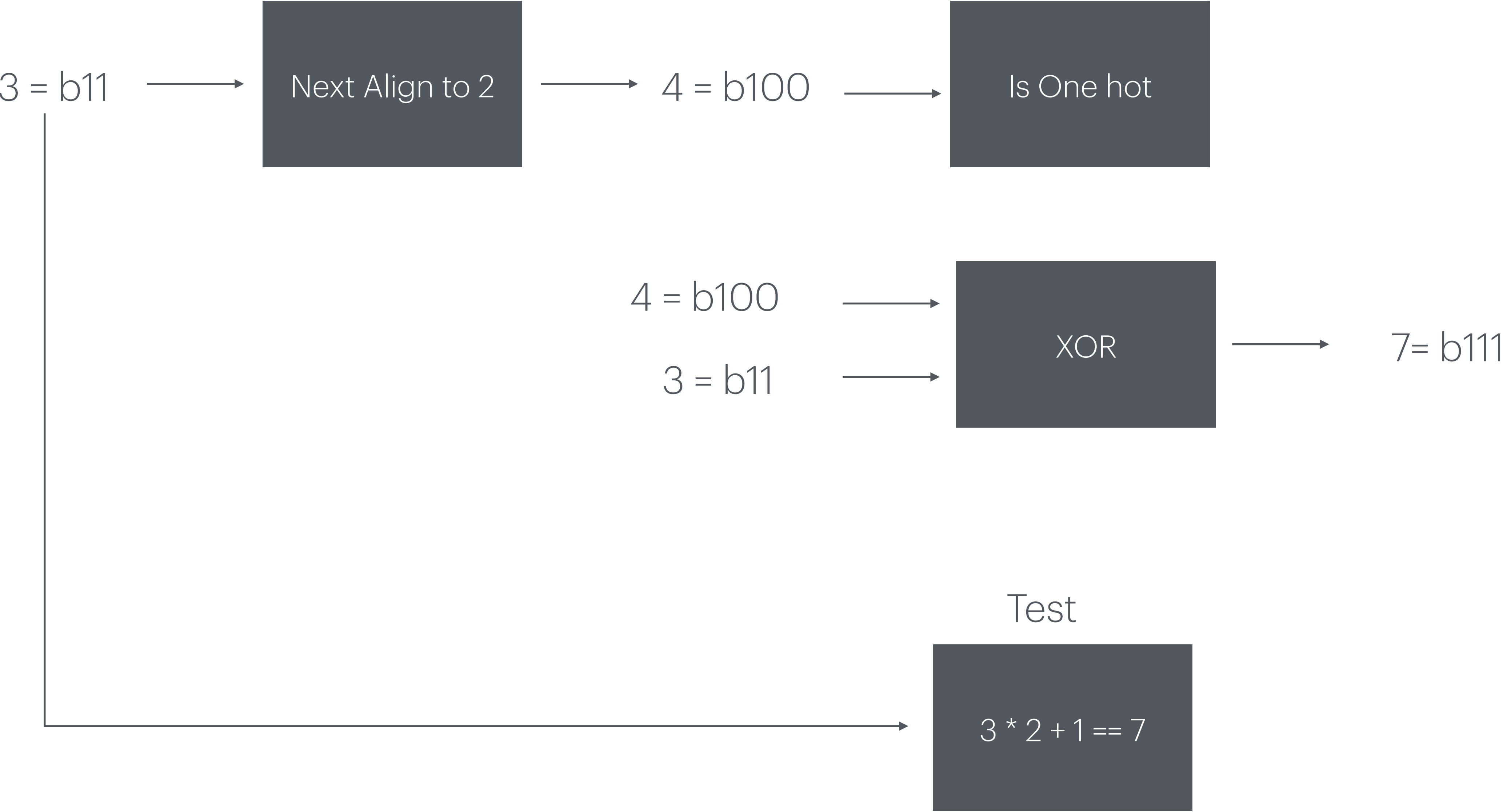
0000_0001

$$0(255) + 1(1)$$

$$0(254) + 1(2)$$

-
-
-

Thermometer Code Detector



Thermometer Code Detector

```
[2025-10-29 19:17:40 UTC] iverilog '-Wall' '-g2012' design.sv testbench.sv && unbuffer vvp a.out
```

```
code -      1  isThermometer - 1
code -      3  isThermometer - 1
code -      7  isThermometer - 1
code -     15  isThermometer - 1
code -     31  isThermometer - 1
code -     63  isThermometer - 1
code -    127  isThermometer - 1
code -    255  isThermometer - 1
code -    511  isThermometer - 1
code -   1023  isThermometer - 1
code -   2047  isThermometer - 1
code -   4095  isThermometer - 1
code -   8191  isThermometer - 1
code -  16383  isThermometer - 1
code -  32767  isThermometer - 1
code - 65535  isThermometer - 1
```

```
Done
```

2 Read 1 Write Register File

Index	0	1	2	3	4	5	6	7	-
resetrn	1	1	1	1	1	1	1	1	1
Din	0	30	100	20	0	40	29	0	10
Wad1	0	0	0	1	0	16	17	0	0
Rad1	0	0	1	0	1	0	0	16	0
Rad2	0	0	1	0	0	0	0	17	0
Wen1	0	0	0	1	0	1	1	0	0
Ren1	0	0	1	0	1	0	0	1	0
Ren2	0	0	1	0	0	0	0	1	0
Collision	0	0	0	1	0	0	0	0	0
Dout1	0	0	0	0	0	20	0	0	40
Dout2	0	0	0	0	0	0	0	0	29
Valid DUIT	1								
Valid Testbench Response	0	1							

2 Read 1 Write Register File

```
[2025-10-29 22:58:39 UTC] iverilog '-Wall' '-g2012' design.sv testbench.sv && unbuffer vvp a.out
```

```
dout1-    0 dout2-    0 collision-0
```

```
dout1-    0 dout2-    0 collision-0
```

```
dout1-    0 dout2-    0 collision-0
```

```
dout1-    0 dout2-    0 collision-0
```

```
dout1-   20 dout2-    0 collision-0
```

```
dout1-   20 dout2-    0 collision-0
```

```
dout1-   20 dout2-    0 collision-0
```

```
dout1-   40 dout2-   29 collision-0
```

```
testbench.sv:199: $finish called at 23 (1s)
```

```
Done
```