Sys-Verilog Questions Review

Some Solutions to questions from ChipIO-Dev

Hector "Hectron" Williams

Counter

clk

Bit

n_ticks

Logic [7:0]
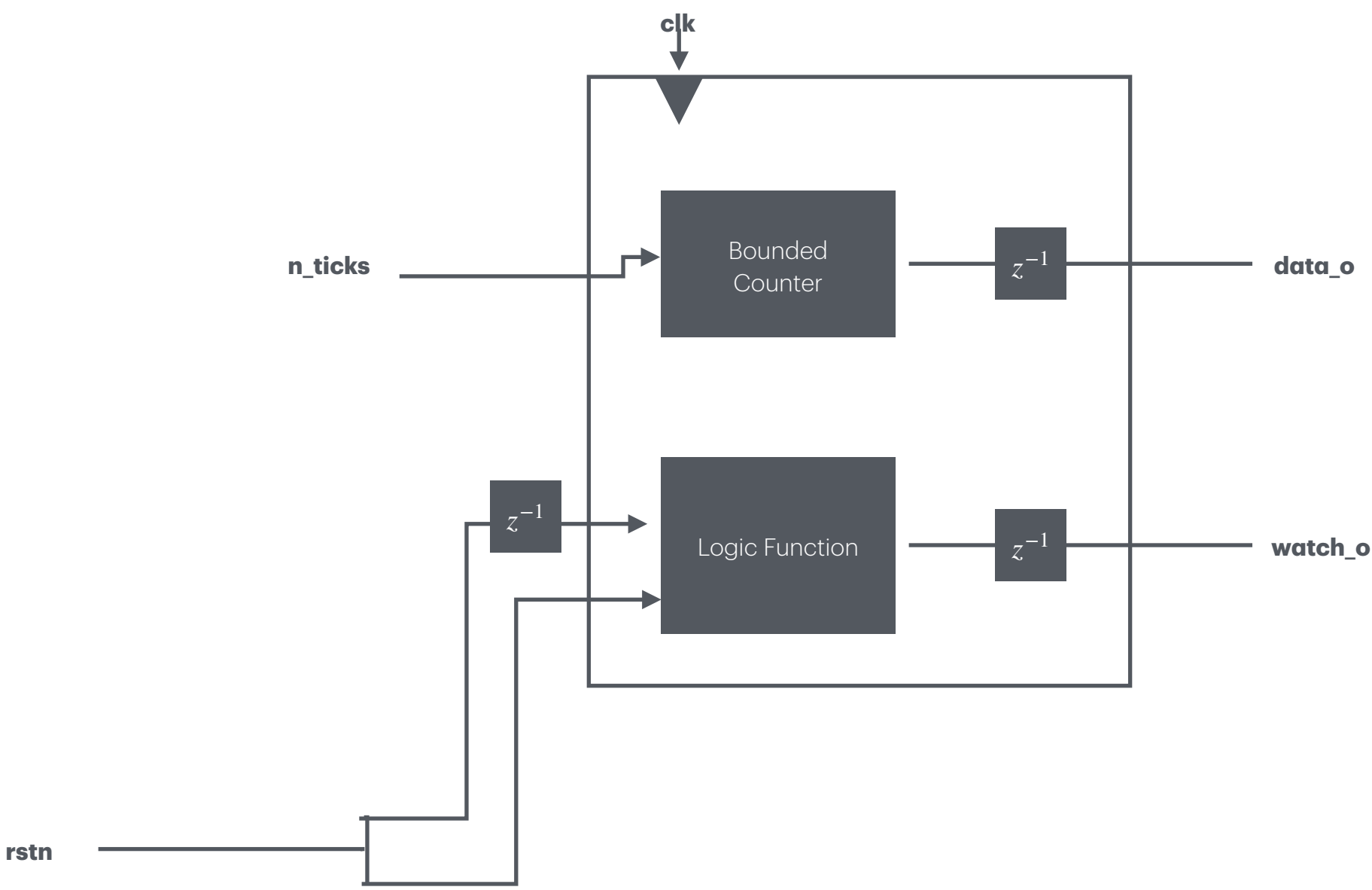
data_o

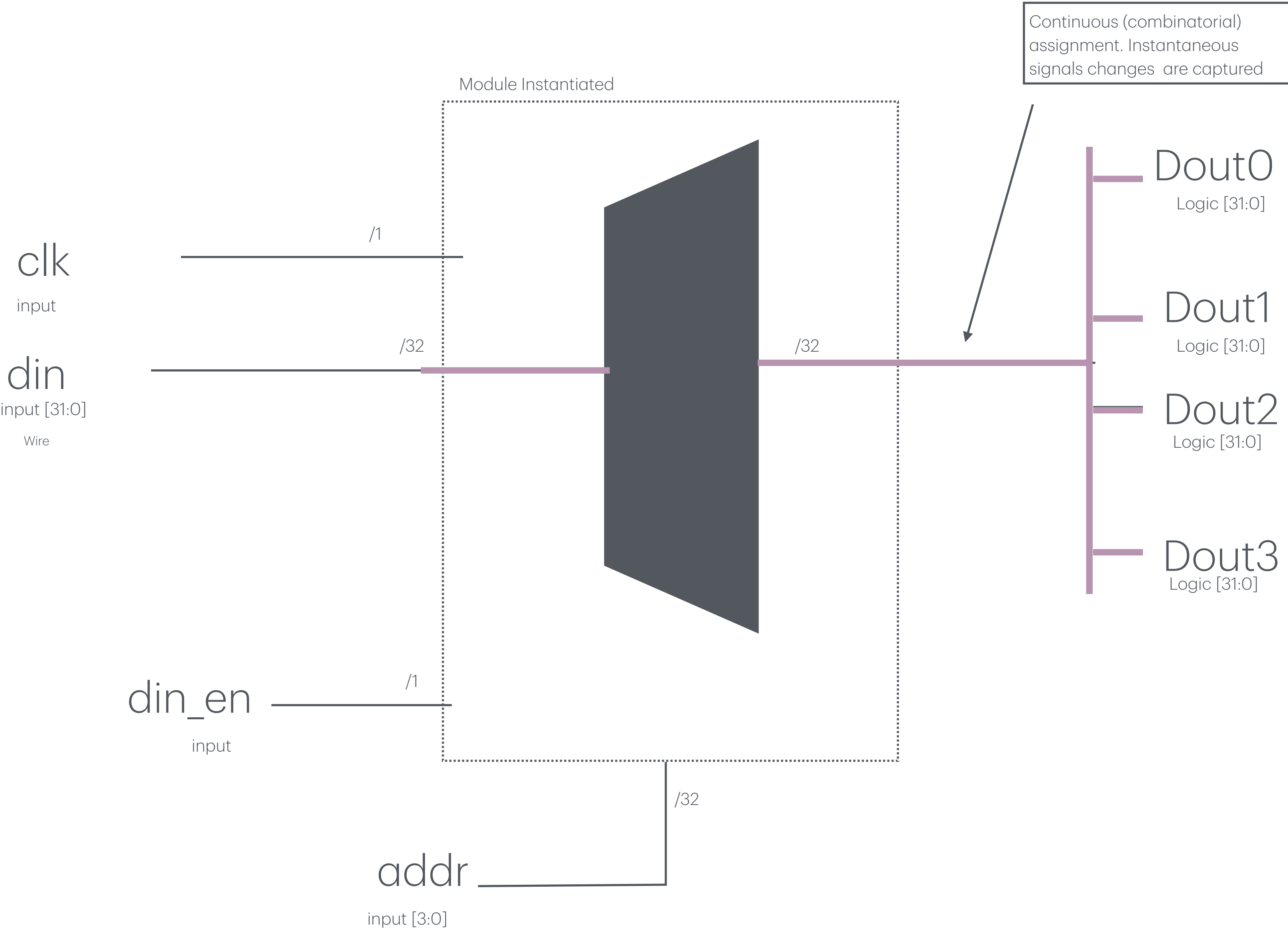Logic [7:0]

watch_o

Bit

Counter

Router

Module Instantiated

Continuous (combinatorial) assignment. Instantaneous signals changes are captured

clk
input

din
input [31:0]

Wire

din_en
input

addr
input [3:0]

/1

/32

/1

/32

/32

/32

Dout0
Logic [31:0]

Dout1
Logic [31:0]

Dout2
Logic [31:0]

Dout3
Logic [31:0]

Connect (wire)

a_in ———— [ ] ———— a_out   b_in ———— [ ] ———— b_out

a_in ———— [assign] a_out ———— (wire) ———— b_in [assign] ———— b_out

top_in ———— a_in [assign] a_out ———— (wire) ———— b_in [assign] b_out ———— top_out

Log2

clk

Bit

x

Logic [7:0]

y

Logic [7:0]

Log2

X —— ◇ <= 1 —— (Done)

Log2(x/2) + 1
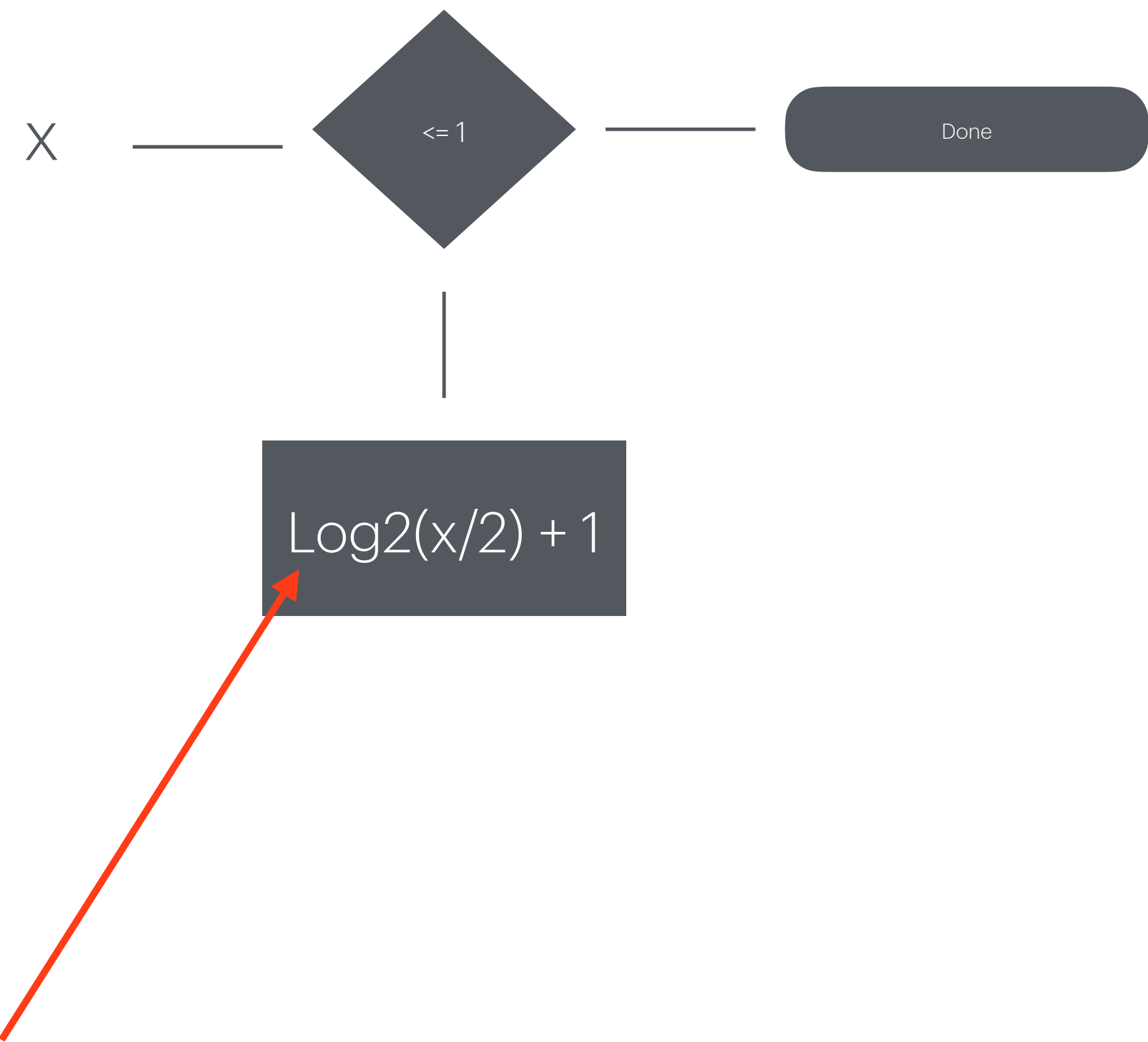
Recursion or calling the same hardware segment repeatedly

Log2(**5**)

Minimum number of bits represent 5?

5 —— Log2(5)

| 1

Log2(2)

| 2

Log2(1)

Min = 2

$$5 > 2^2 \; (\text{increment})$$ ___ ___ ___

Log2(**4**)

5 —— Log2(4)

| 1

Log2(2)

| 2

Log2(1)

Min = 2

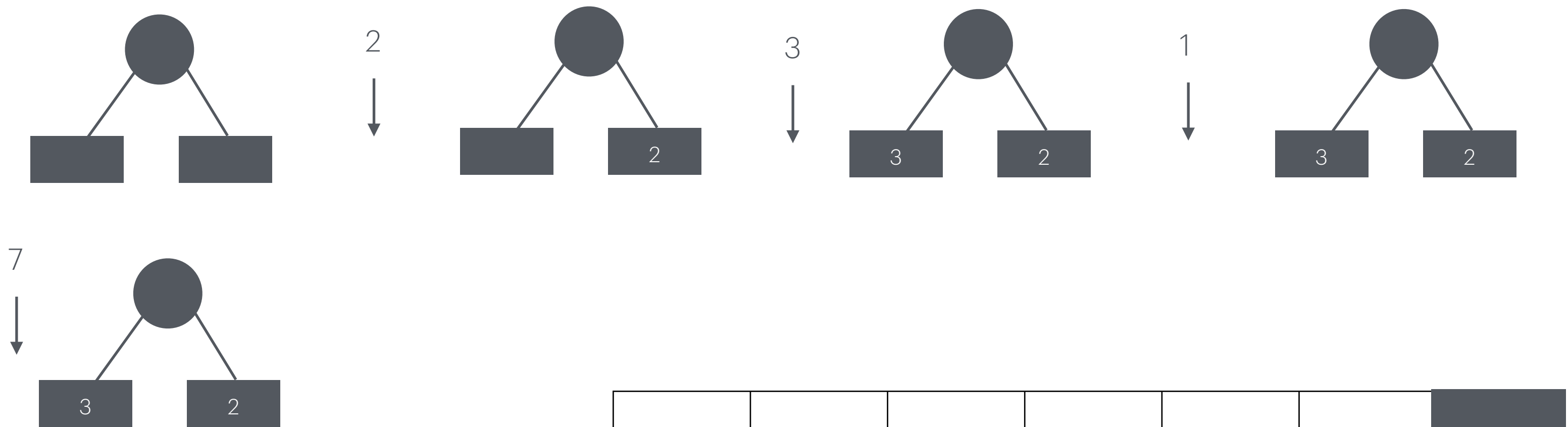$4 \not< 2^2$ (perfect)   ___ ___

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]

Log2
Module

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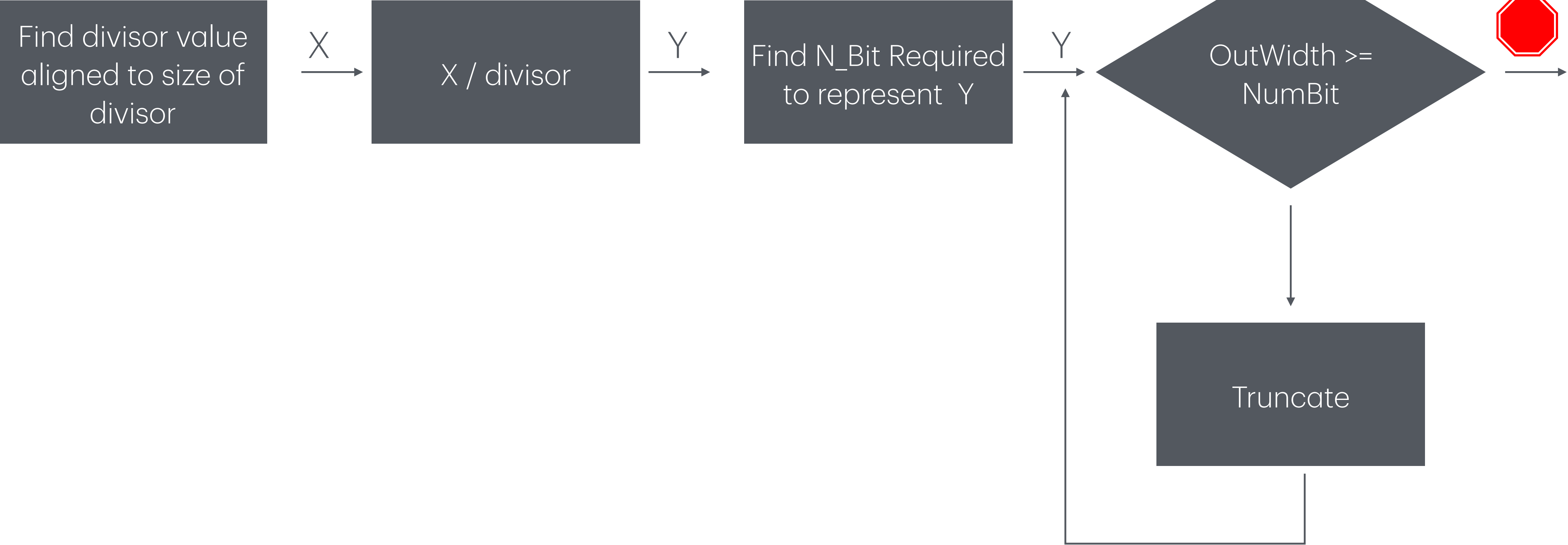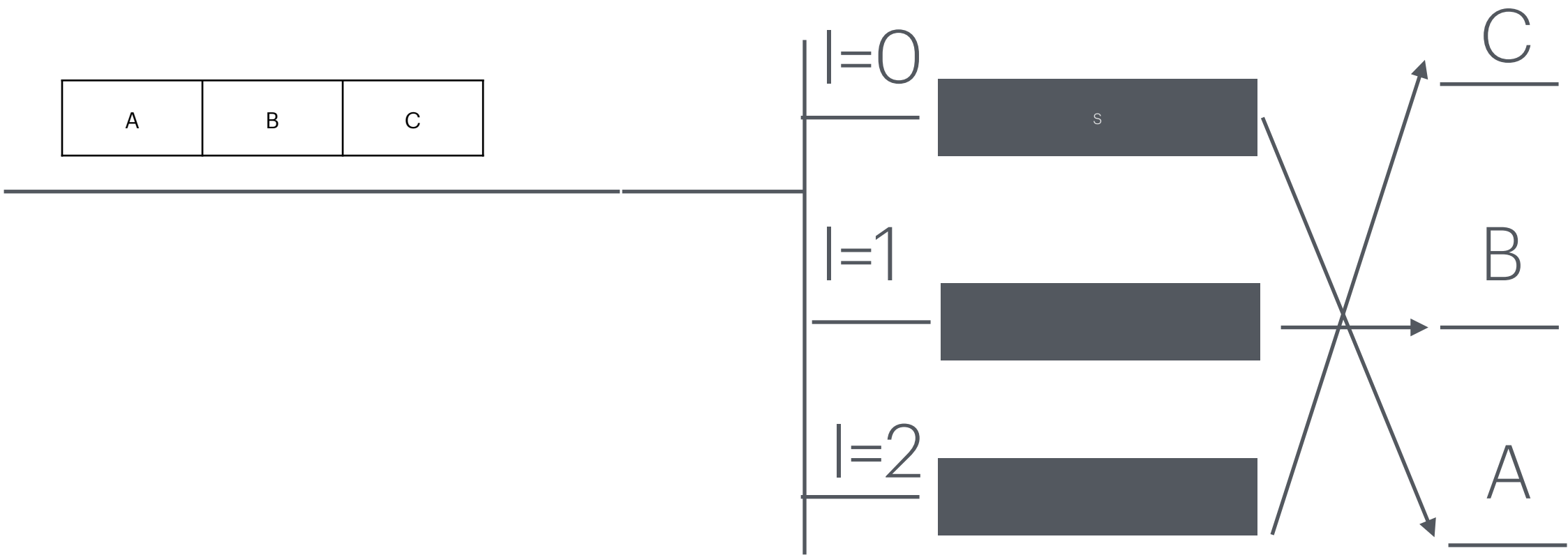VECTOR RCVD [4]
VECTOR RCVD [8]

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

```
Find divisor value        X        X / divisor        Y        Find N_Bit Required        Y        OutWidth >=
aligned to size of                                              to represent Y                      NumBit
divisor

                                                                                                        │
                                                                                                        ▼

                                                                                                    Truncate
```

Reverse Bits

| A | B | C |

I=0
I=1
I=2

C
B
A

Generate Logic Blocks

Gray code

RST

CLK

Counter

i

id

$2^{id} = (counter + 1)$ → FSM

Register
Size = 2^i

out[l]

i+1

id

$2^{id+1} = active\_bits$ → FSM

Register
Size = 2^i

out[l+1]

out

Gray code

Vertical Delay :)

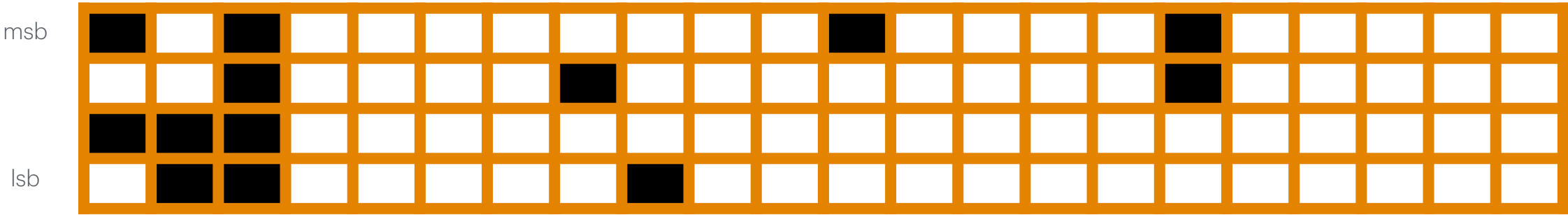| $2^3 = 8cycle$	$2^2 = 4cycle$	$2^1 = 2cycle$	$2^0 = 1cycle$
0	0	0	0
0	0	0	1
0	0	1	1
0	0	1	0
0	1	1	0

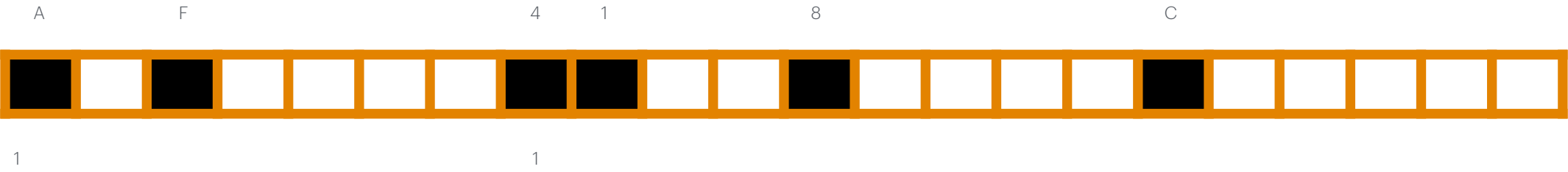Egde Detector
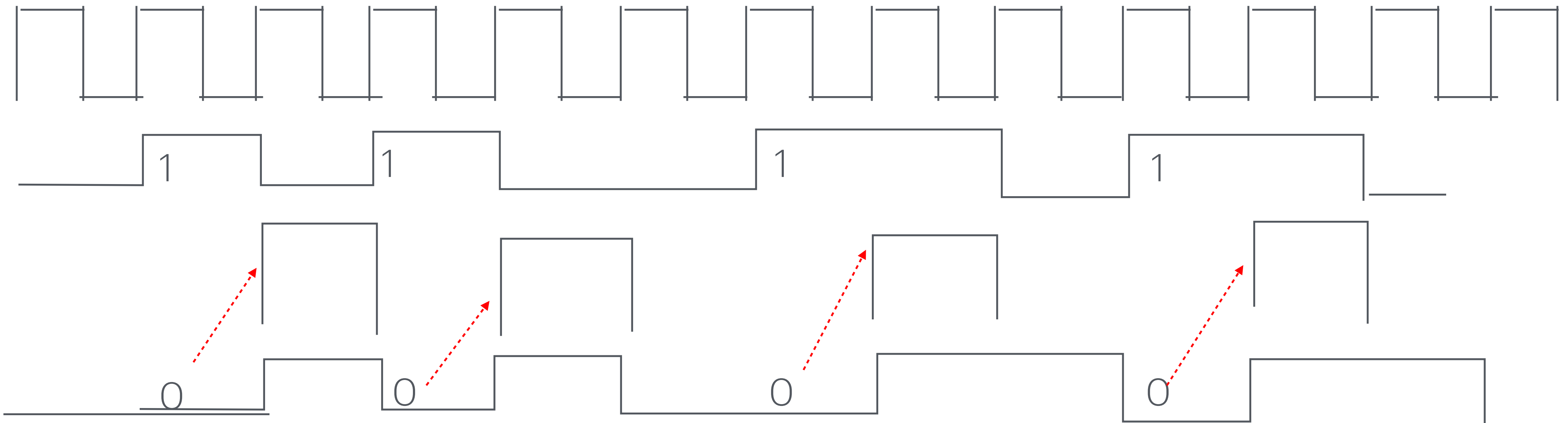
Parralel In -Serial Out

DATA

Bit Sequence

Enable

D_IN

RST

Output
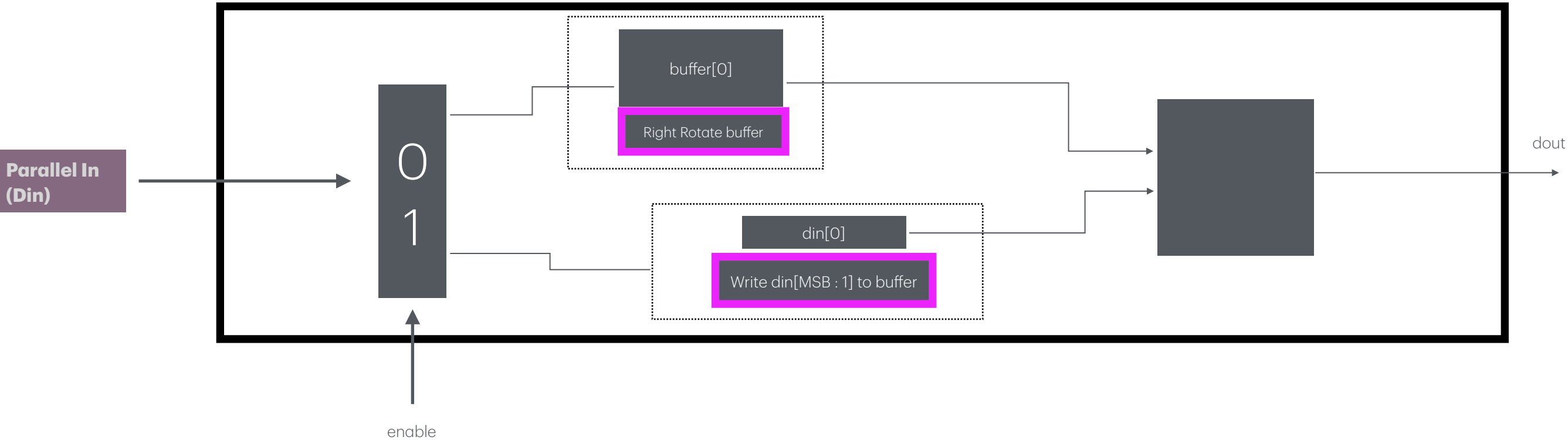
Serial to Parallel
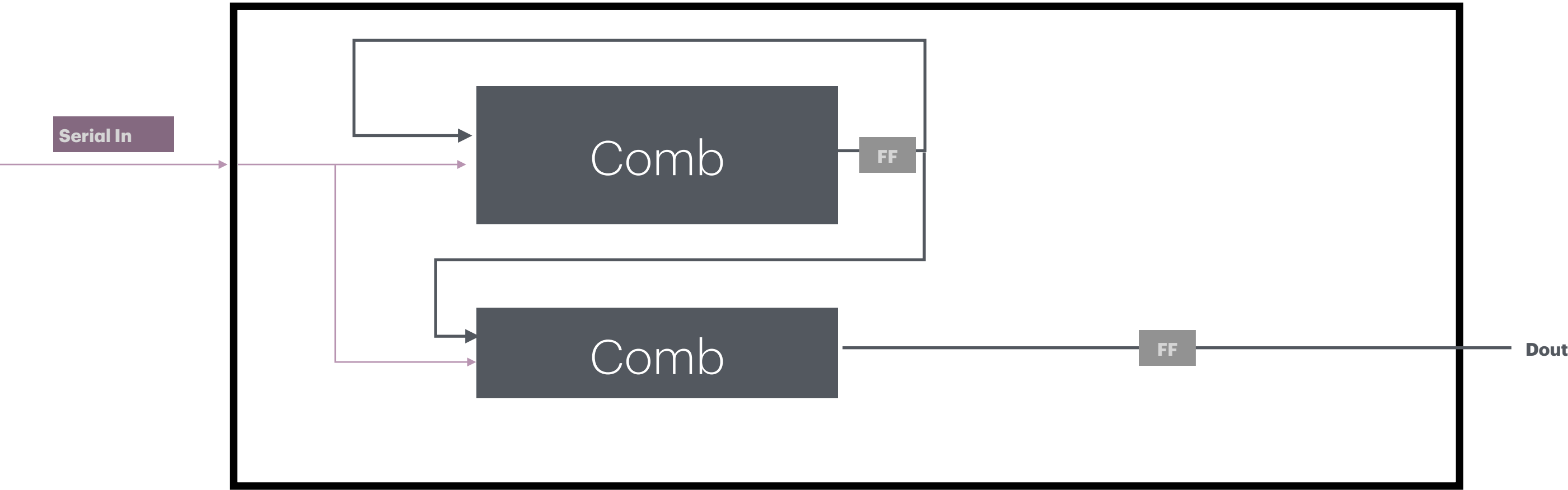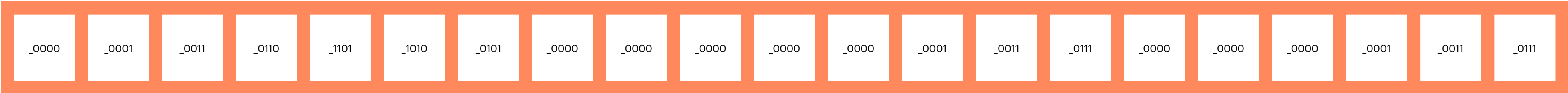
Serial to Parallel (Simulation Concept)

Holds RST Active Low

| UNDEF | UNDEF | RST[0] | RST[1] | RST[2] | RST[3] | RST[4] | RST[5] |

0 1 2 3 4 5

0 1 2 3 4 5

| DIN[0] | DIN[1] | DIN[2] | DIN[3] | DIN[4] | DIN[5] |

| OUT[0] | OUT[1] | OUT[2] | OUT[3] | OUT[4] | OUT[5] |
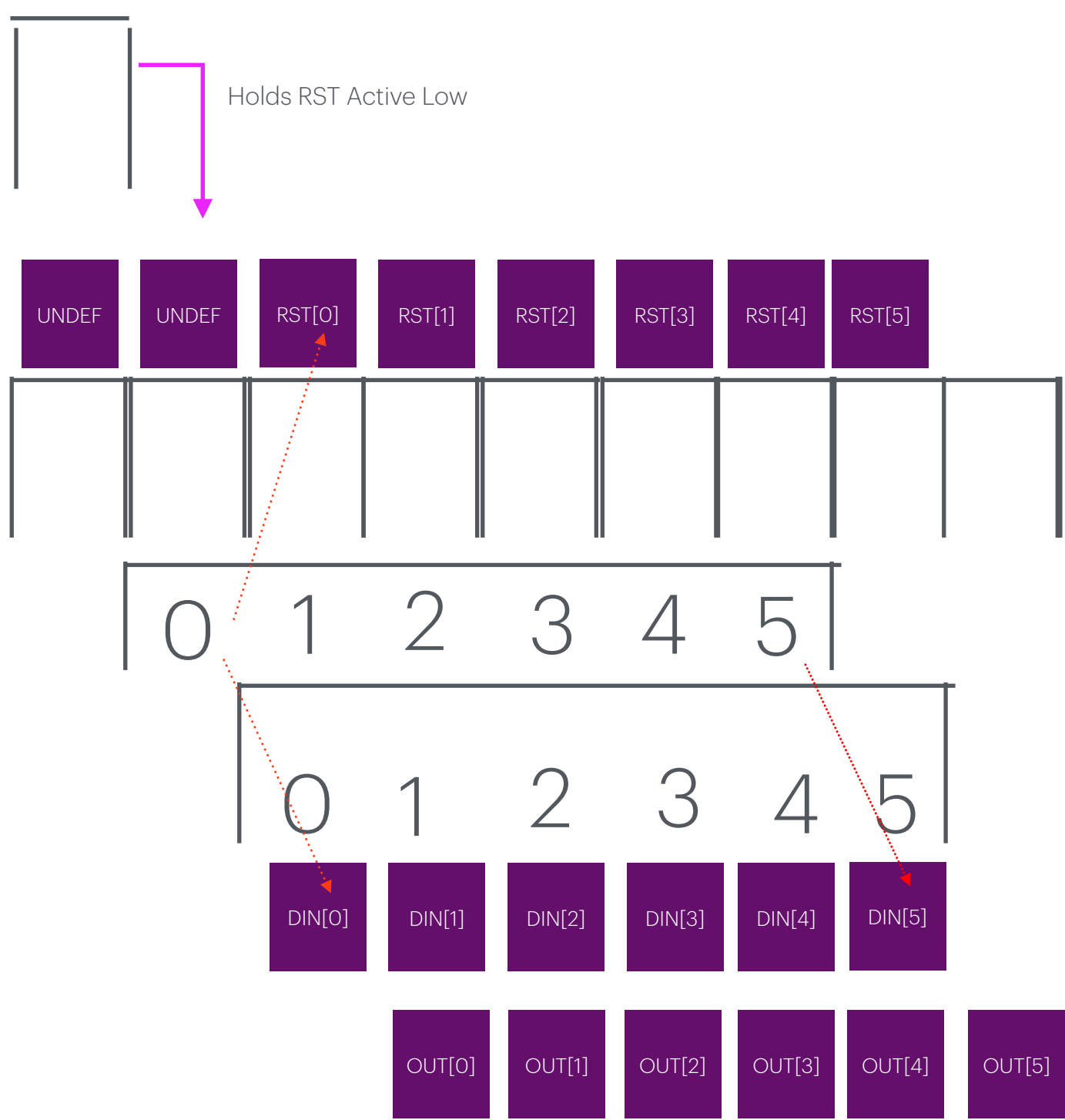
Fibonacci
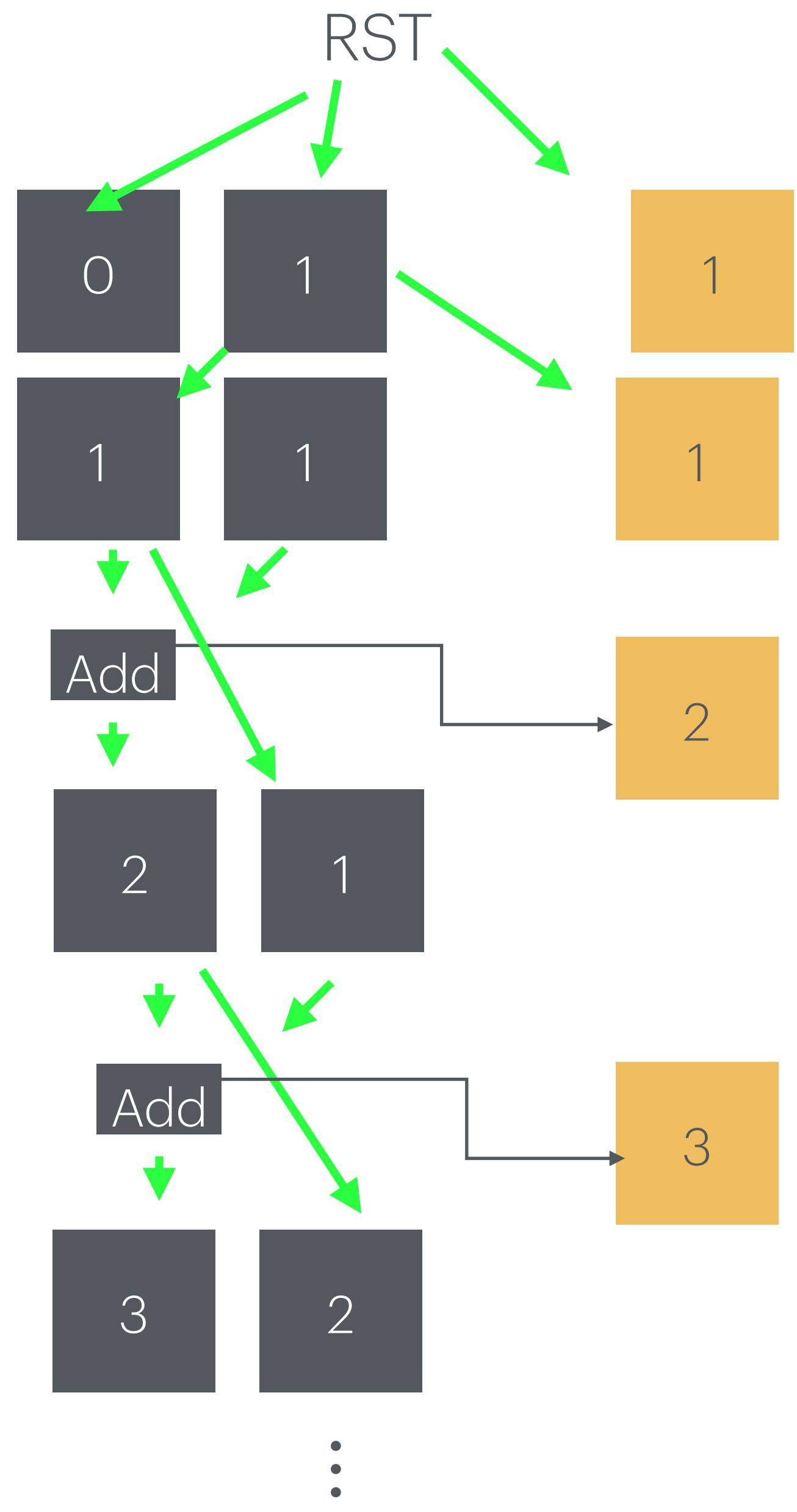
Count Ones

Architecture Similar to Linked List

Gen

Din_0

adderz

$wire_0$
equal_width

Gen

Din
data_width

Din_1

adderz

$wire_1$
equal_width

Gen

Din_N

adderz

$wire_{15}$
equal_width

Dout (i.e. Count)

equal_width = log2(data_width)
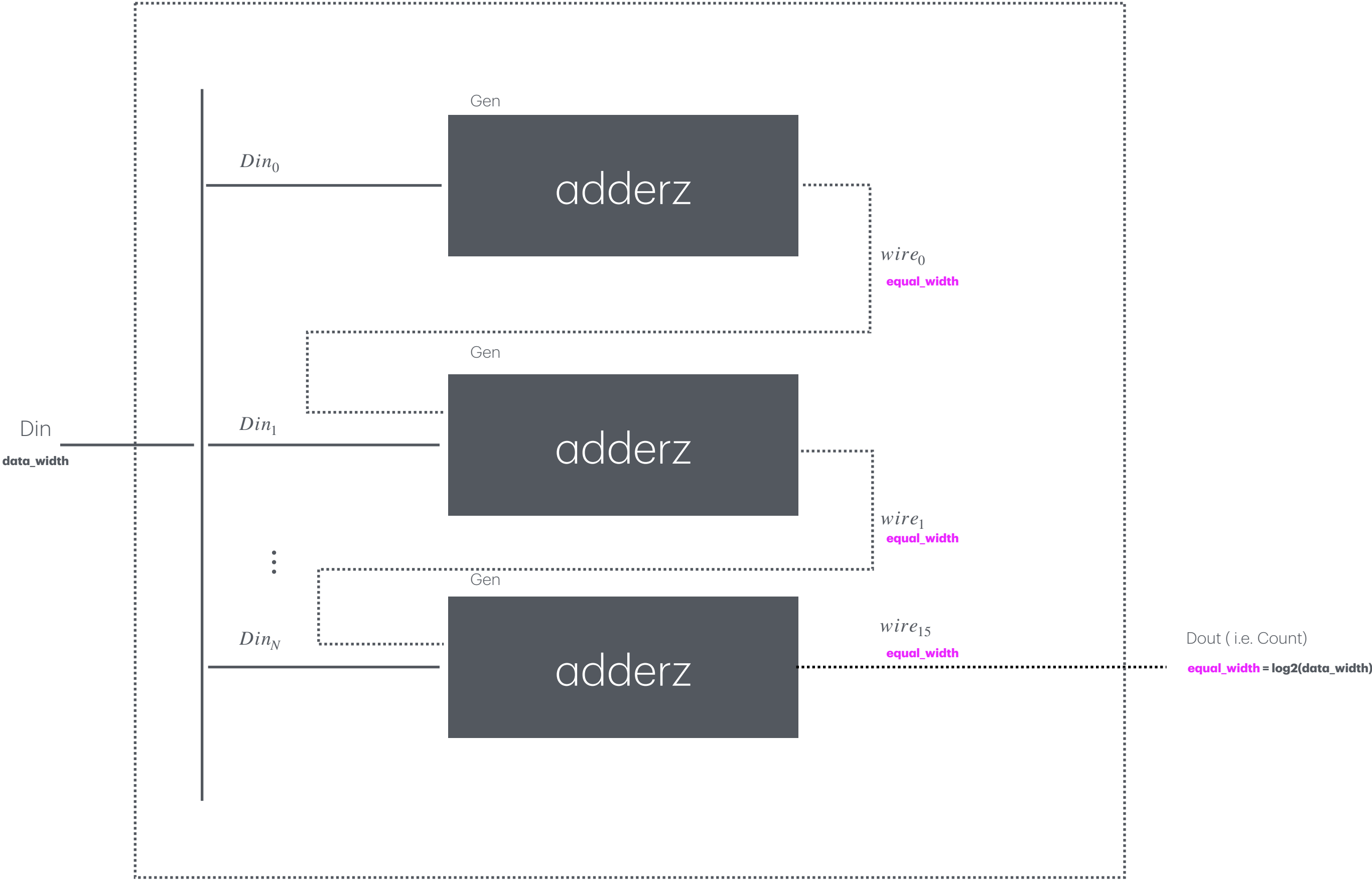
Count Ones

[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)

Code_gray

011

01100110

00111100

11110000

Serializer

Gray Code
Bit Extender

00111100   11110000   01100110

00000000   11111111   11111111

Find Index

One Hot Parser

bin

Trailing Ones

```
                    ┌──────────────┐
                    │              │
                    │  Zero Output │
                    │              │
                    └──────┬───────┘
                           │
      ┌──────────────┐     ▼        ┌──────────────┐
din ──│     Cond     ├──► Ready ───►│  Set Output  │
      └──────────────┘     ▲        └──────────────┘
                           │
                    ┌──────┴───────┐
                    │              │
                    │  Loop Search │
                    │              │
                    └──────────────┘
```
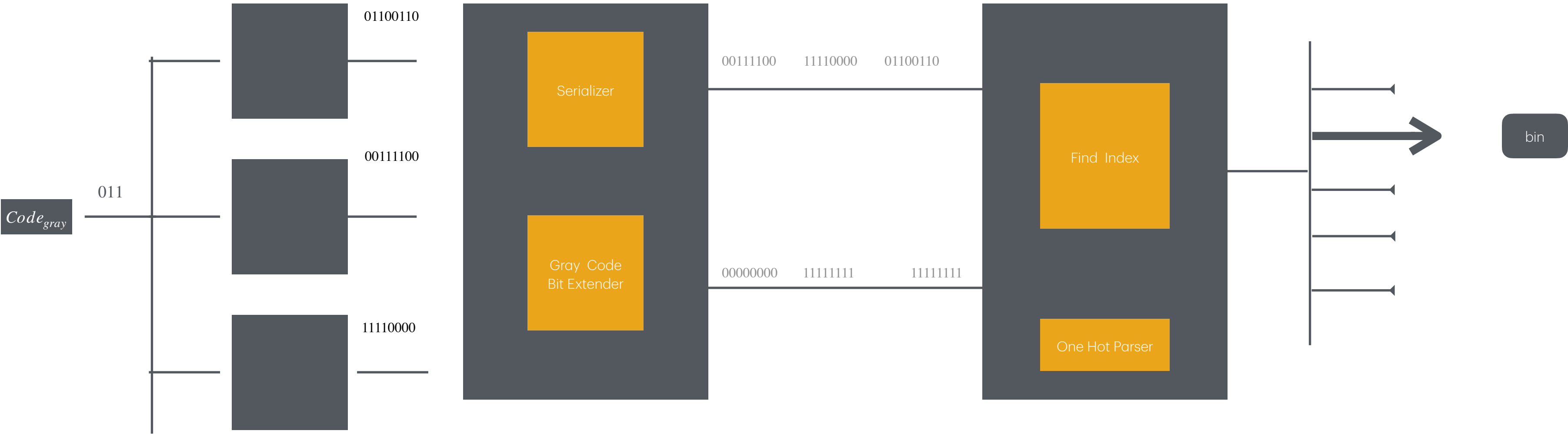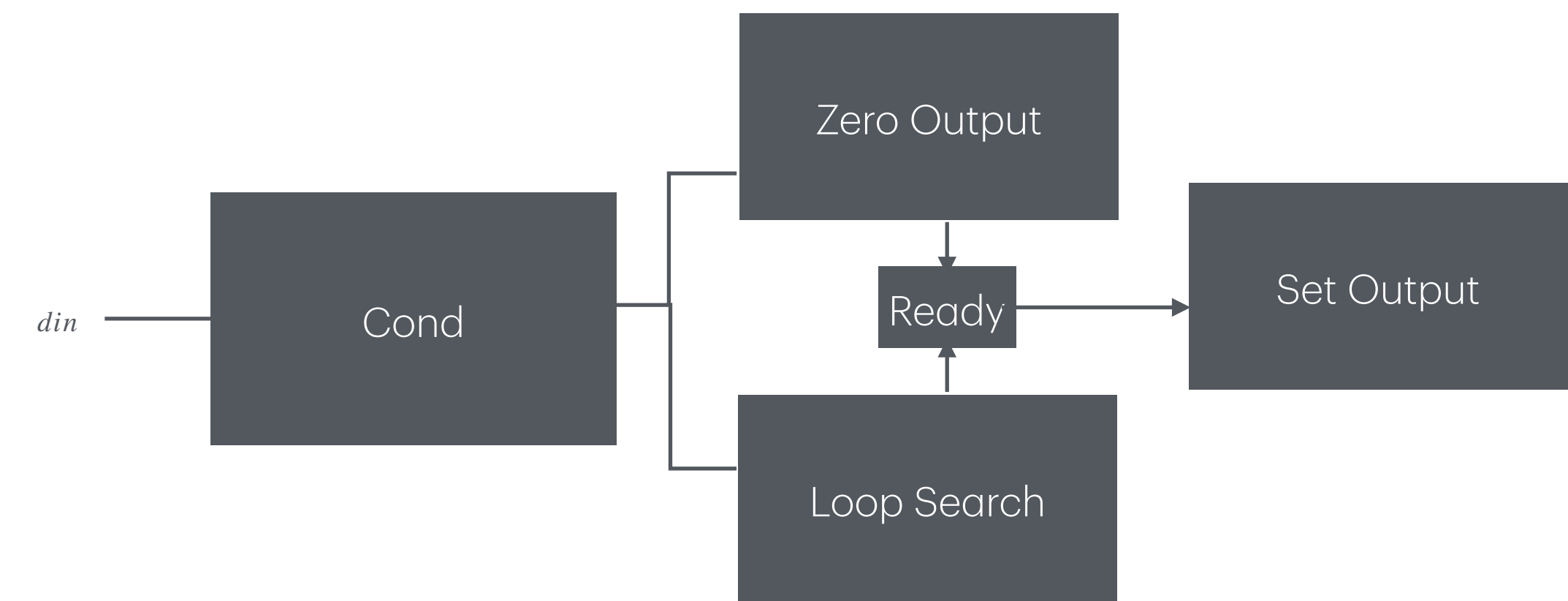
StopWatch Timer

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

```
[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
```
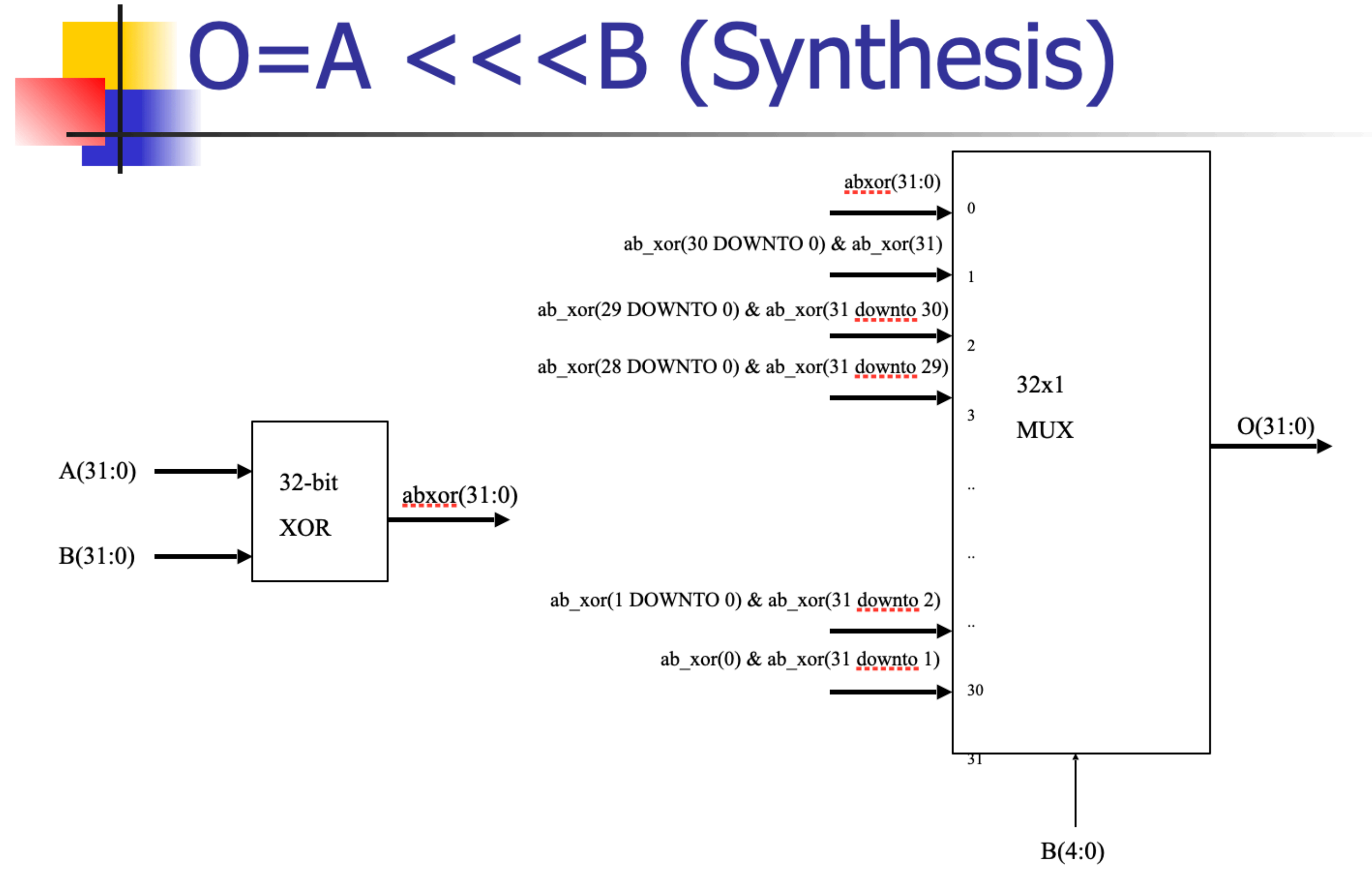
O=A <<<B (Synthesis)

abxor(31:0)

0

ab_xor(30 DOWNTO 0) & ab_xor(31)

1

ab_xor(29 DOWNTO 0) & ab_xor(31 downto 30)

2

ab_xor(28 DOWNTO 0) & ab_xor(31 downto 29)

3

32x1

MUX

..

A(31:0) → 32-bit XOR → abxor(31:0)

B(31:0)

..

ab_xor(1 DOWNTO 0) & ab_xor(31 downto 2)

..

ab_xor(0) & ab_xor(31 downto 1)

30

31

O(31:0)

B(4:0)

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

Random Circuit 1

wire

Broadcast network

async

A

B

XOR
STRUCTURE

LEFT ROTATE MUX

dout

DOUT

SELECT

Sequence Detector

				1

			1	0

		1	0	1

	1	0	1	0 Eval

1	0	1	0	1 Eval

.	.	.	.	1 Eval

.	.	.	.	1 Reset

⋮
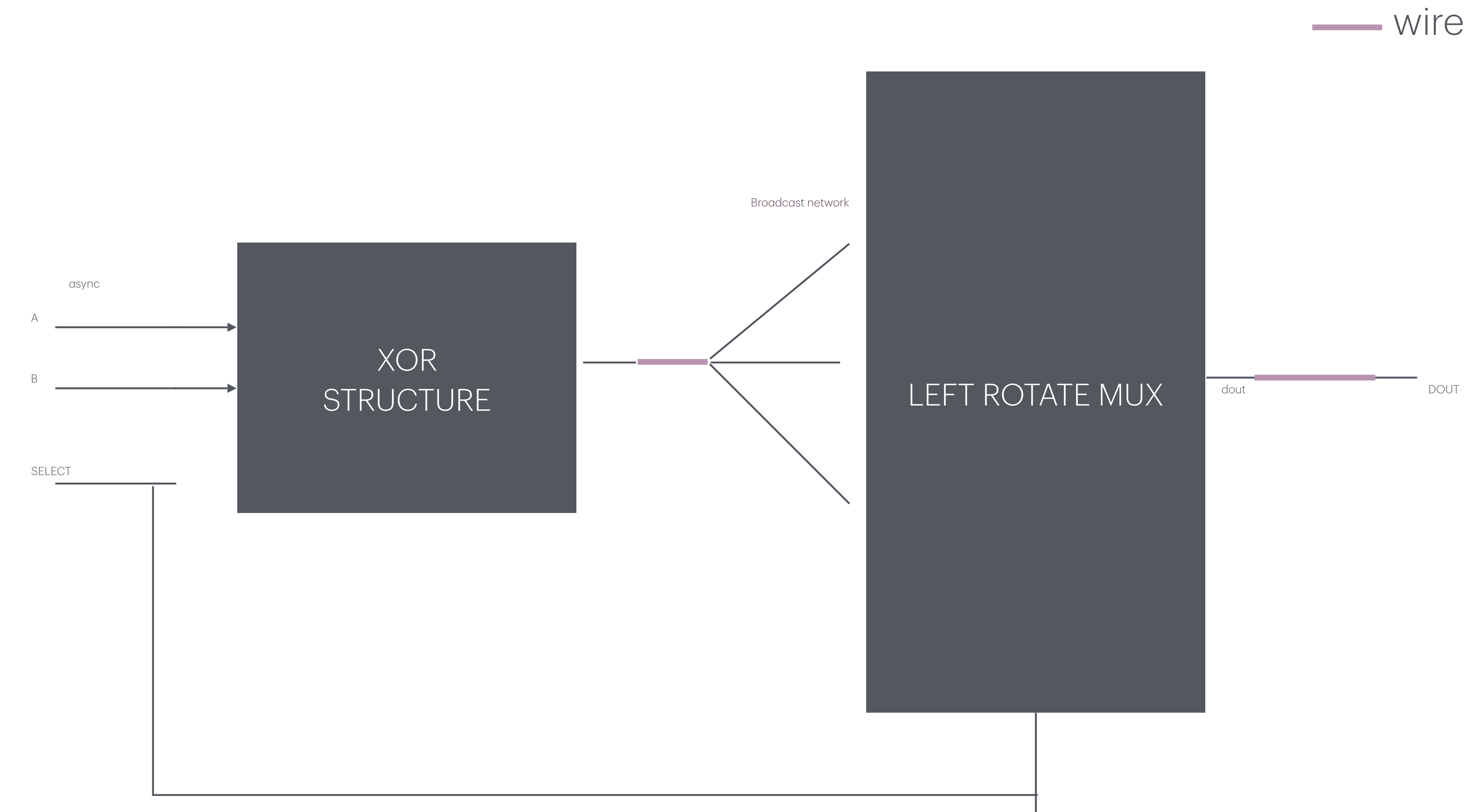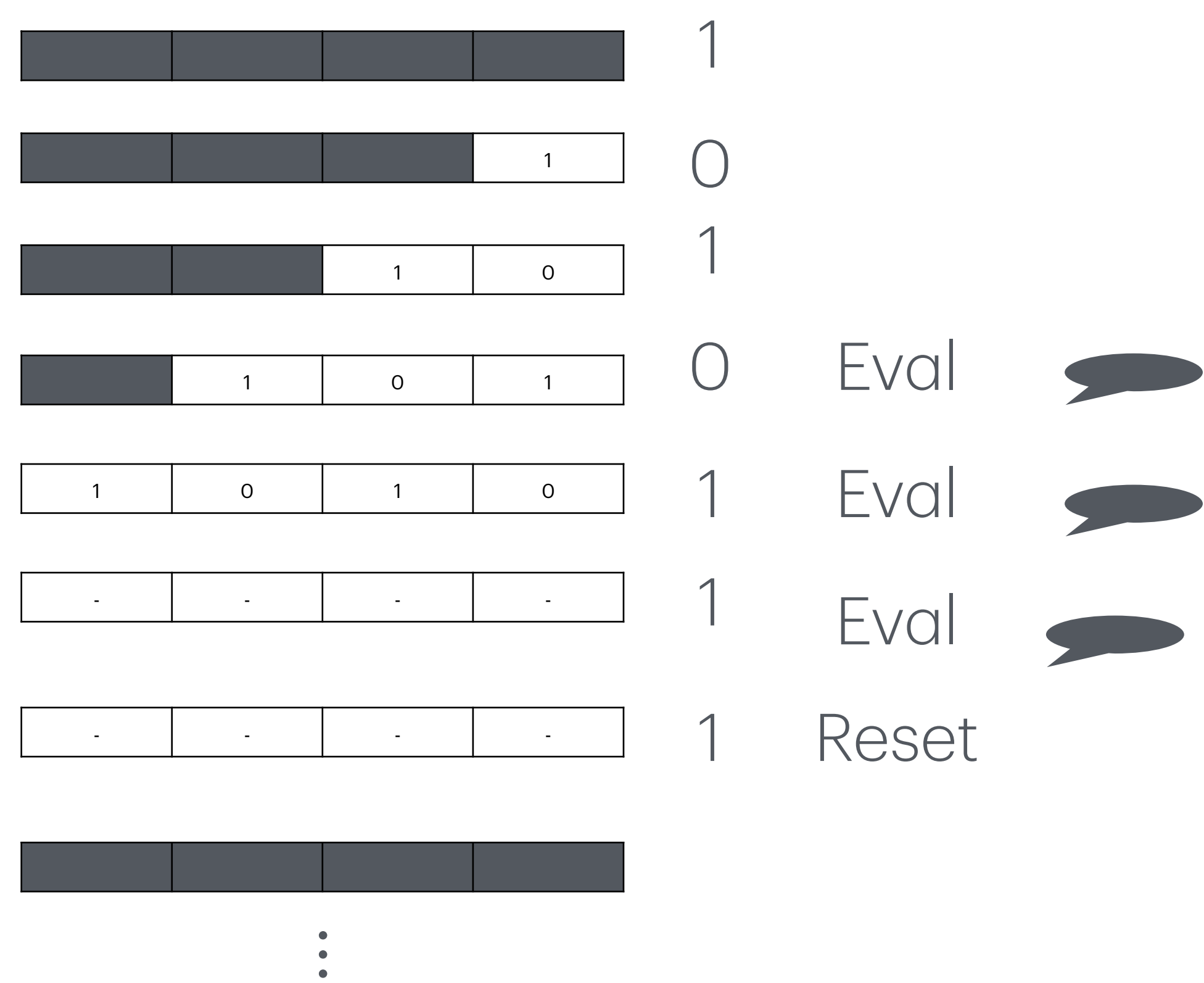
Sequence Detector

| 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 | 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 |
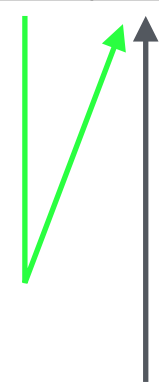
FIFO FULL

Evaluations of sequence are valid

Histo Amazon

Send Mechanism

Setup pulses within *SEND* state

h

h_z

h_zz

Valid Setup Gen						
0	1	2	3	4	5	6

Data Stream Setup Gen						
0	1	2	3	4	5	6

Data Bit Stream RCVD by DUT

0	1	2	3	4	5	6

Data Bit Stream RCVD by DUT

Data received by HISTOGRAM CIRCUIT on next clk edge!

Histogram Circuit

DUT processes/handles data at each tick

Histo Amazon
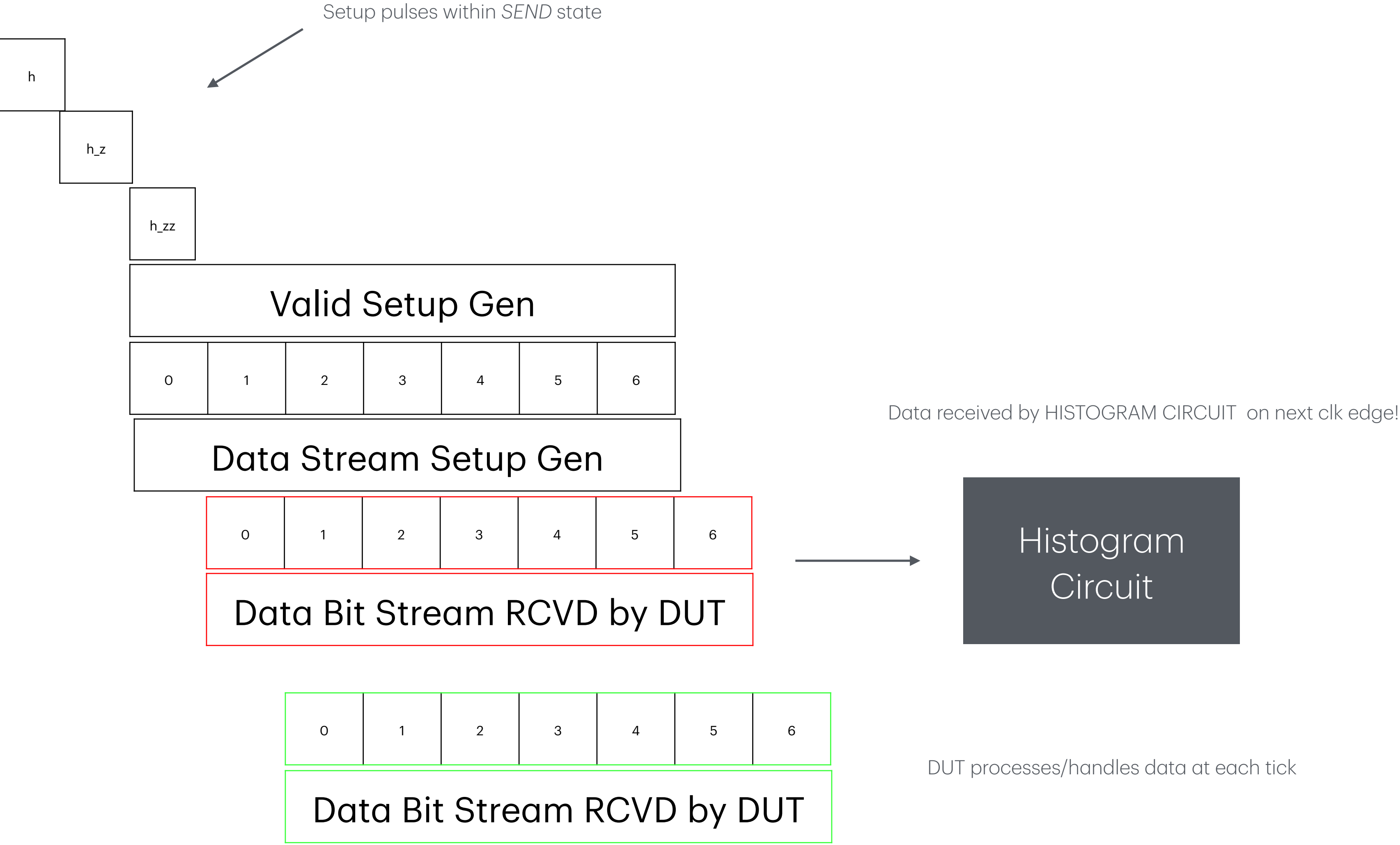
Receive Mechanism (Test Address)

Setup pulses

h

h_z

Address 0	Address 1	Address 2	Address 3	Address N

Testbench Step T-0

Address 0	Address 1	Address 2	Address 3	Address N

Testbench Step T-1 (Received by DUT)

H(0)	H(1)	H(2)	H(3)	H(4)

Testbench Step T-2 (Response from DUT)
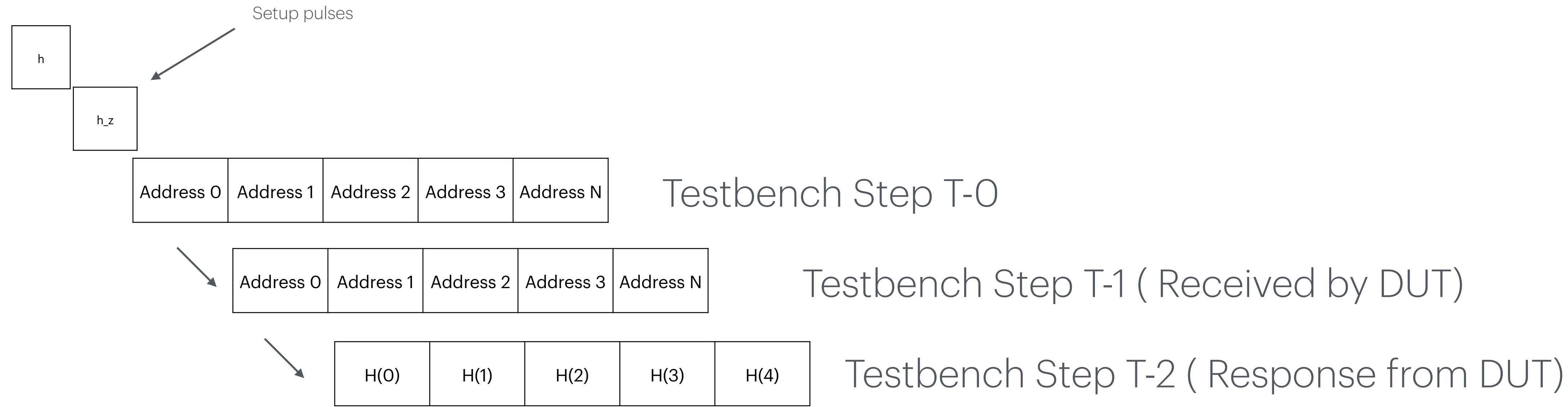
Divisible By Three

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

x2 + 1   x2   x2 + 1   x2   x2 + 1   x2

State Flow

x2 = Value << 1

x2 + 1 = (Value << 1) + 1
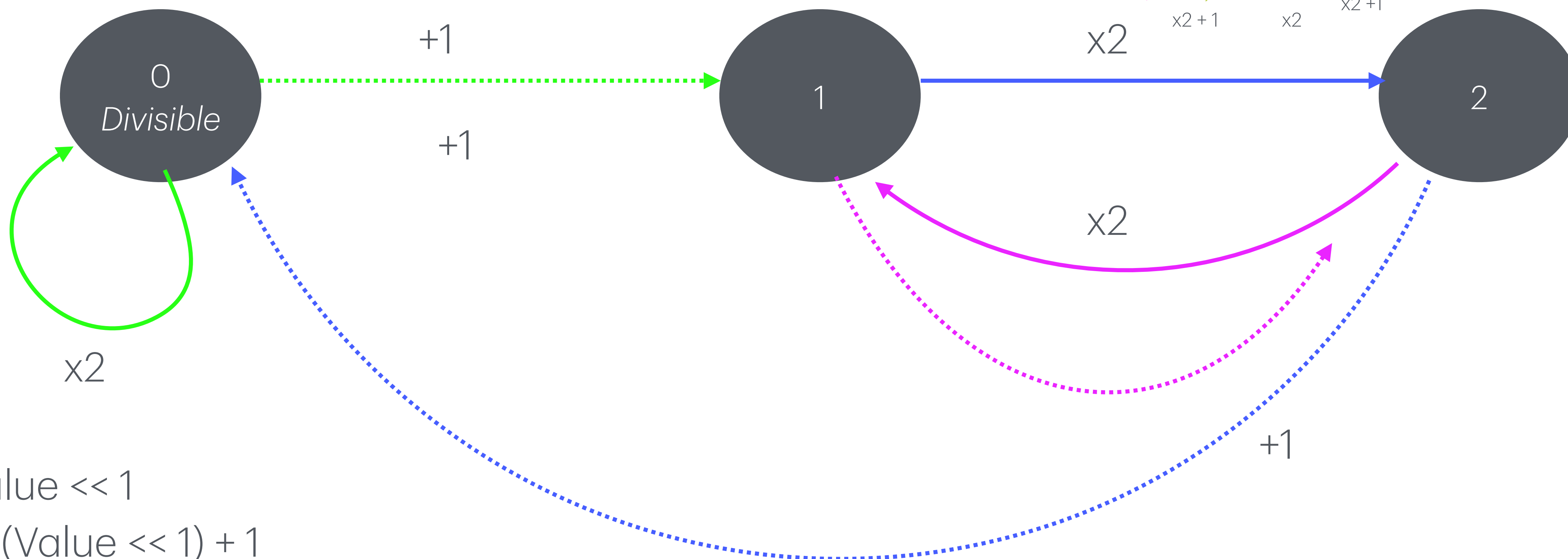
Divisible By Five
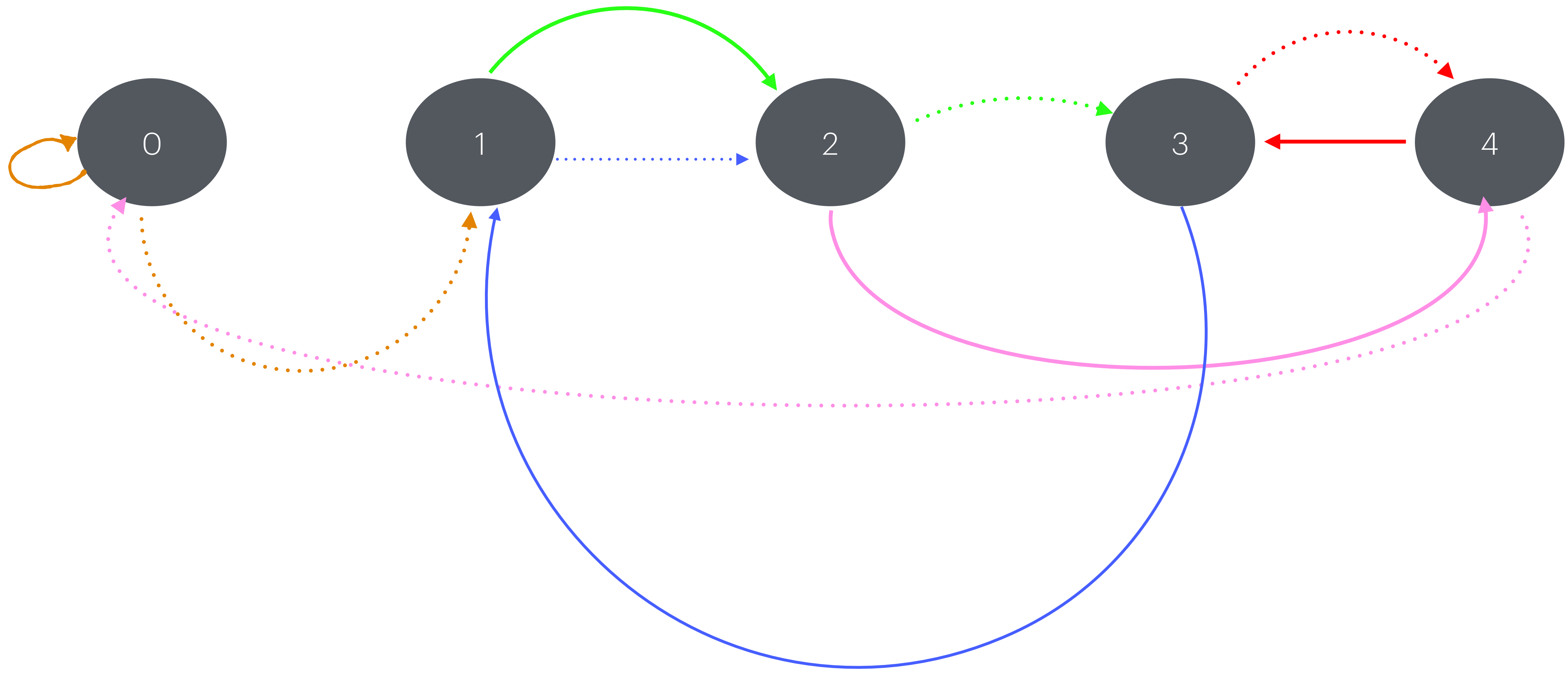
Palindrome

Odd Data Width

0	1	0	1	0

RotateRight by mid+1 → NOT → Multiply by ones vector Size = Half_Size_Floored → == DataIn[mid:0]

Half_Size_Floored = 2

Even Data Width

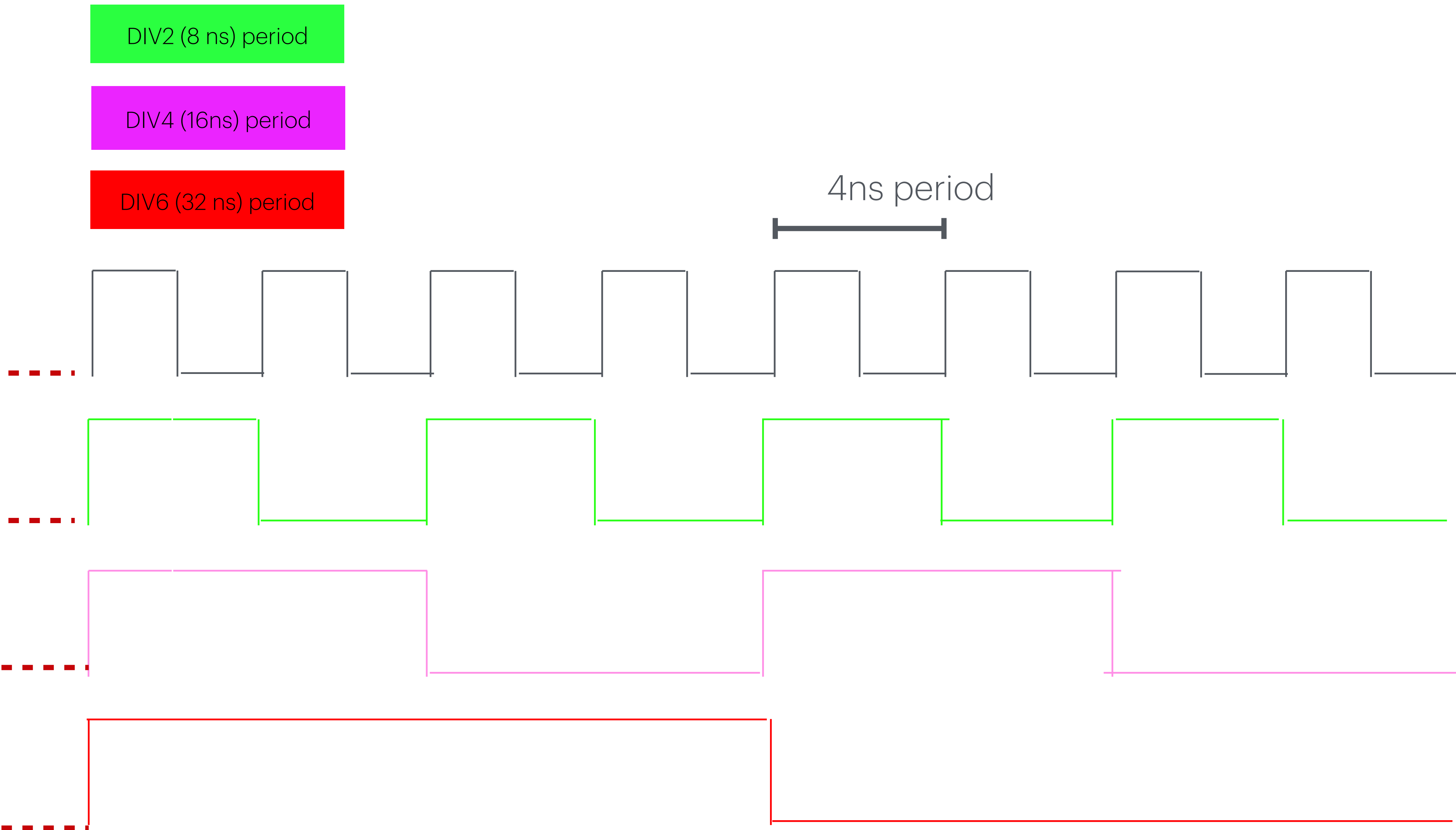| 0	1	1	1

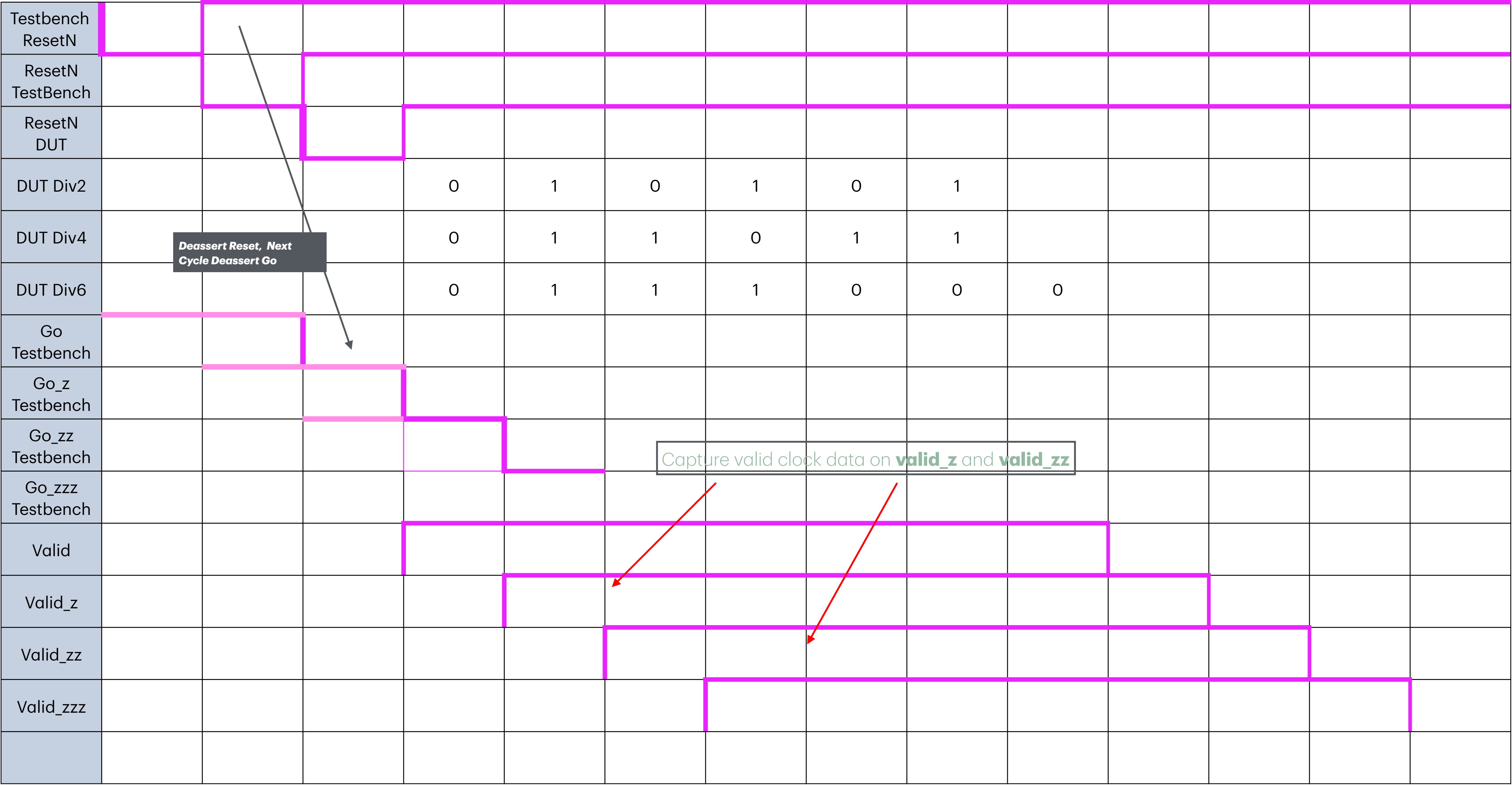RotateRight by mid → NOT → Multiply by ones vector Size = Half_Size_Floored → == DataIn[mid:0]

Divide-By-Events

DIV2 (8 ns) period

DIV4 (16ns) period

DIV6 (32 ns) period

4ns period

Divide-By-Events Timing

Testbench ResetN														
ResetN TestBench														
ResetN DUT														
DUT Div2				0	1	0	1	0	1					
DUT Div4				0	1	1	0	1	1					
DUT Div6				0	1	1	1	0	0	0				
Go Testbench														
Go_z Testbench														
Go_zz Testbench														
Go_zzz Testbench														
Valid														
Valid_z														
Valid_zz														
Valid_zzz														

Deassert Reset, Next Cycle Deassert Go

Capture valid clock data on **valid_z** and **valid_zz**

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

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

Full Adder

A —

B —

HA

HA

Sum

CIN

OR
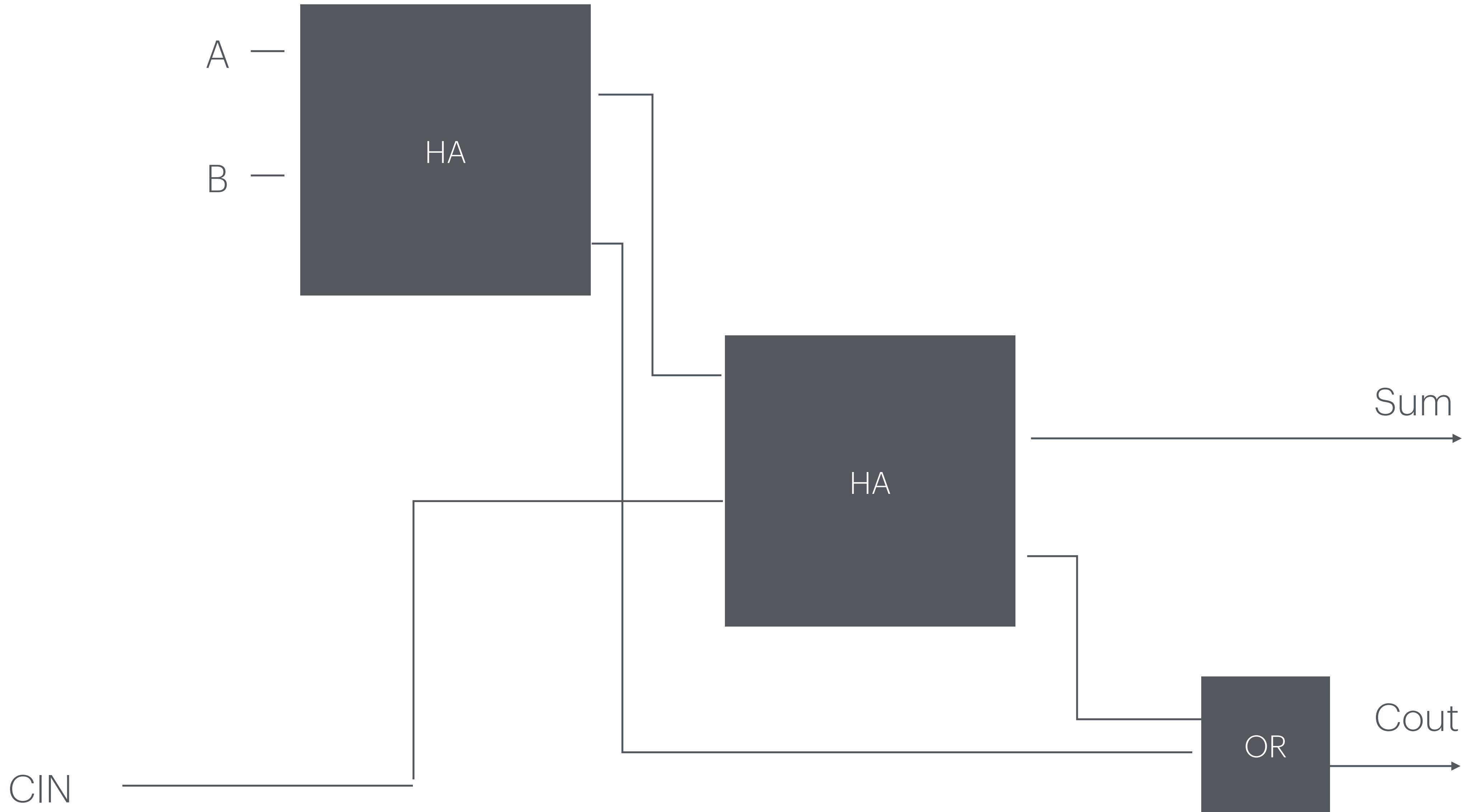
Cout

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
```
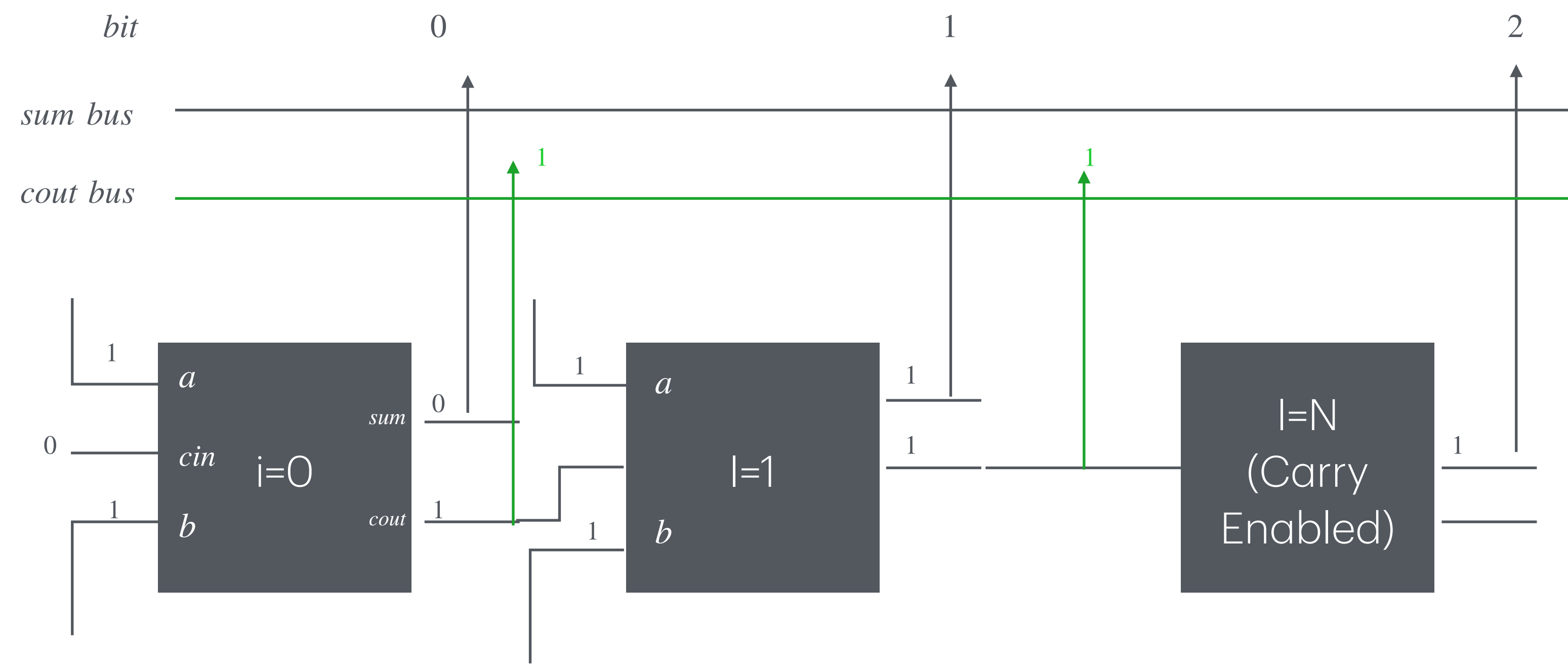
Ripple Adder

A = 2b'11   B= 2b'11

bit   0   1   2

sum bus

cout bus

1

1

1	*a*	
0	*cin*	i=O
1	*b*	
	sum	0
	cout	1

1	*a*	
		I=1
1	*b*	

1
1

I=N
(Carry
Enabled)

1

1

1
1   1
1   1
_____
O

1   1
    1   1
    1   1
_____
1   O

1   1
    1   1
    1   1
_____
1   1   O

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
```

FlipFlop

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

FlipFlop

```
[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
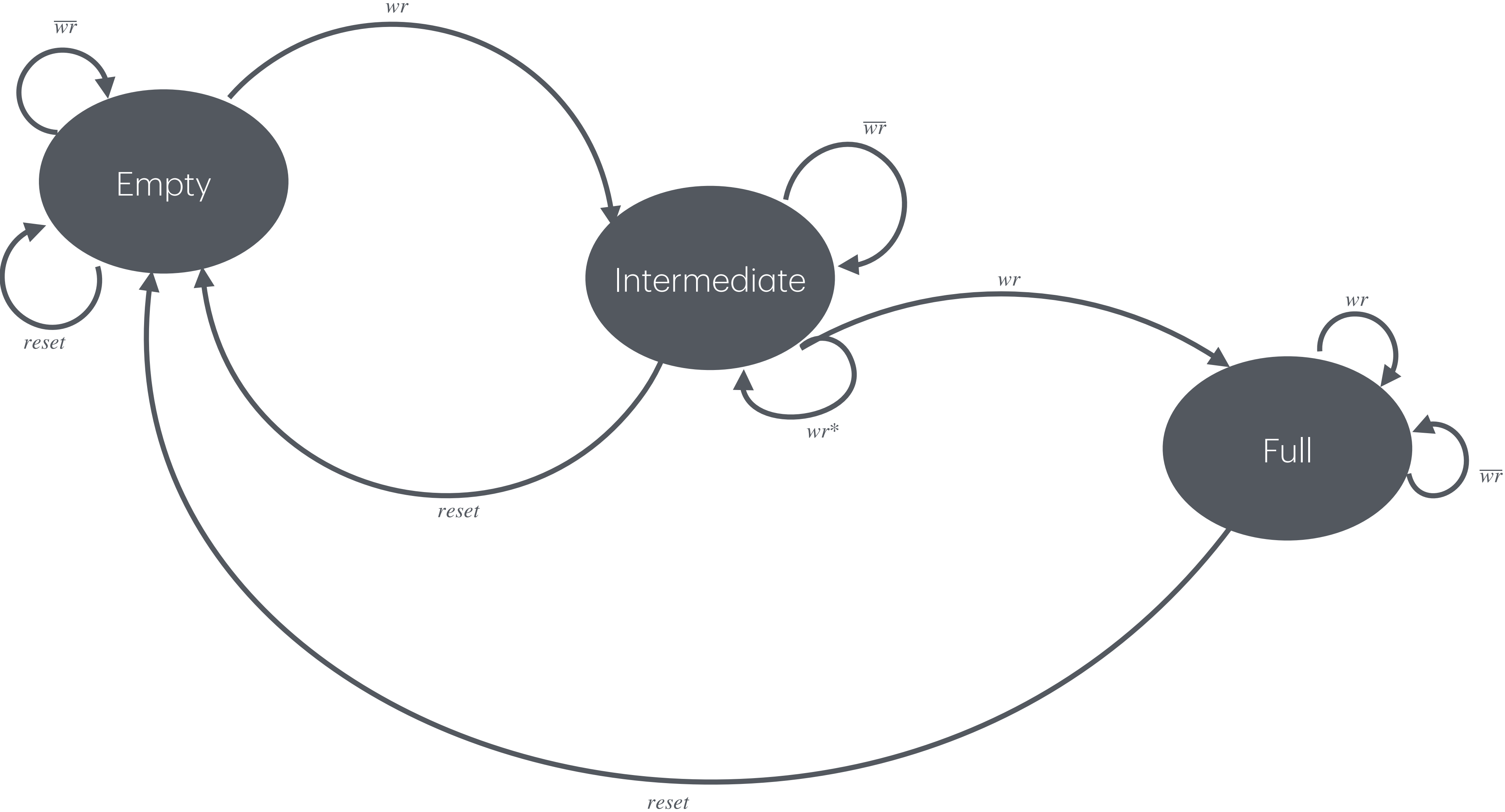
MULTI BIT FIFO

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