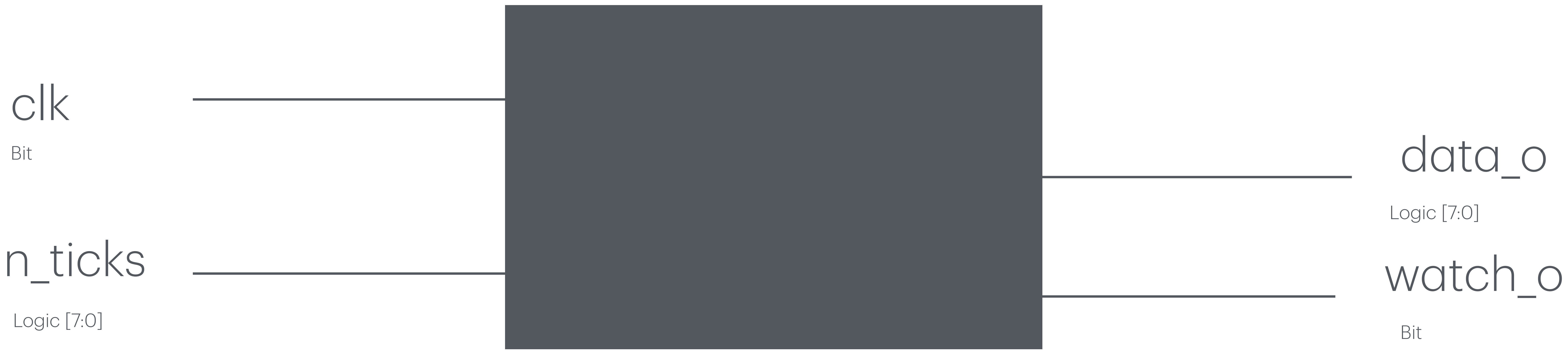
Sys-Verilog Questions Review

Some Solutions to questions from ChipIO-Dev

Hector “Hectron” Williams

Counter



clk

Bit

n_ticks

Logic [7:0]

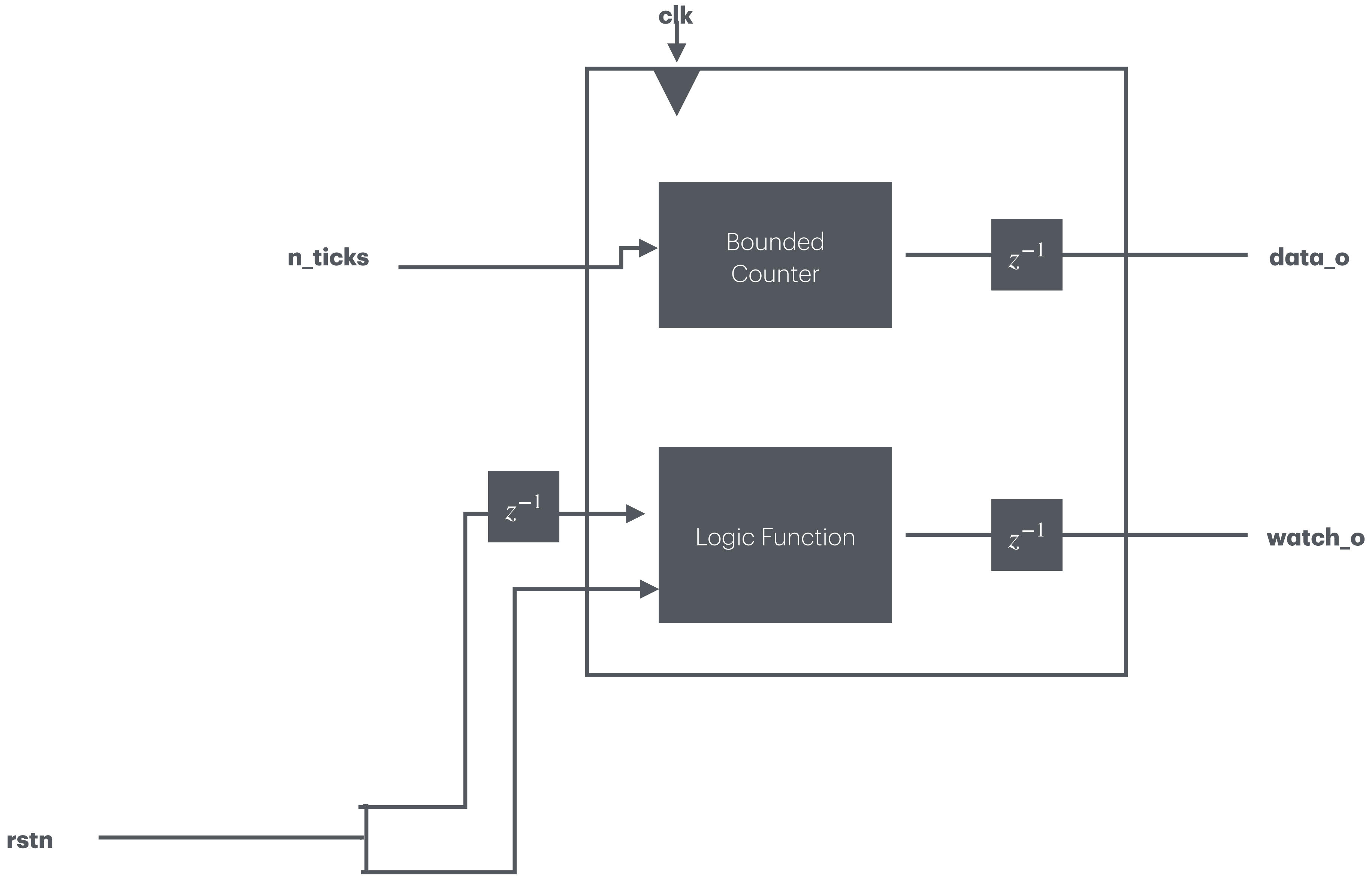
data_o

Logic [7:0]

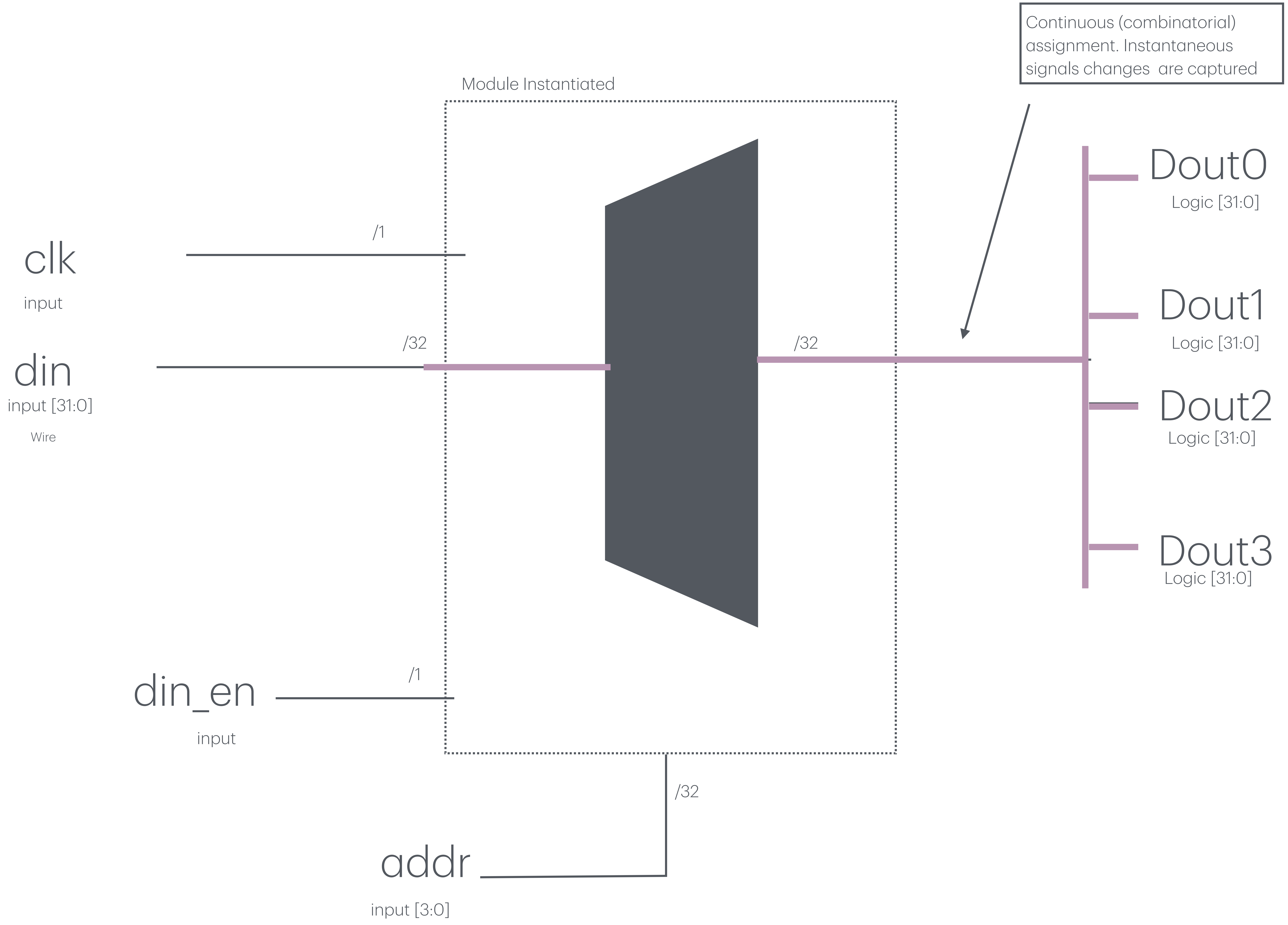
watch_o

Bit

Counter

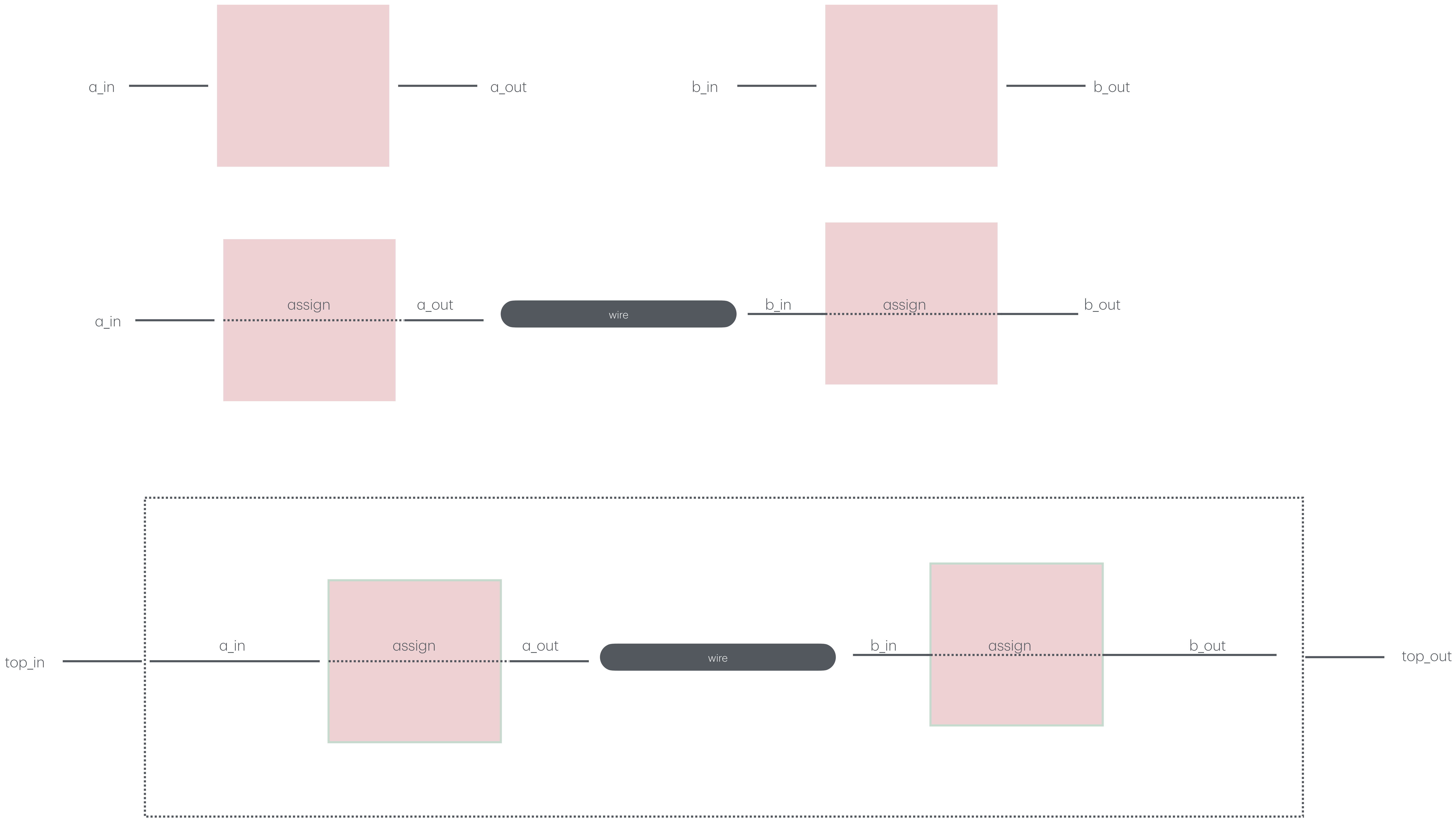

clk
n_ticks
Bounded Counter
z^{-1}
data_o
z^{-1}
Logic Function
z^{-1}
watch_o
rstn

Router

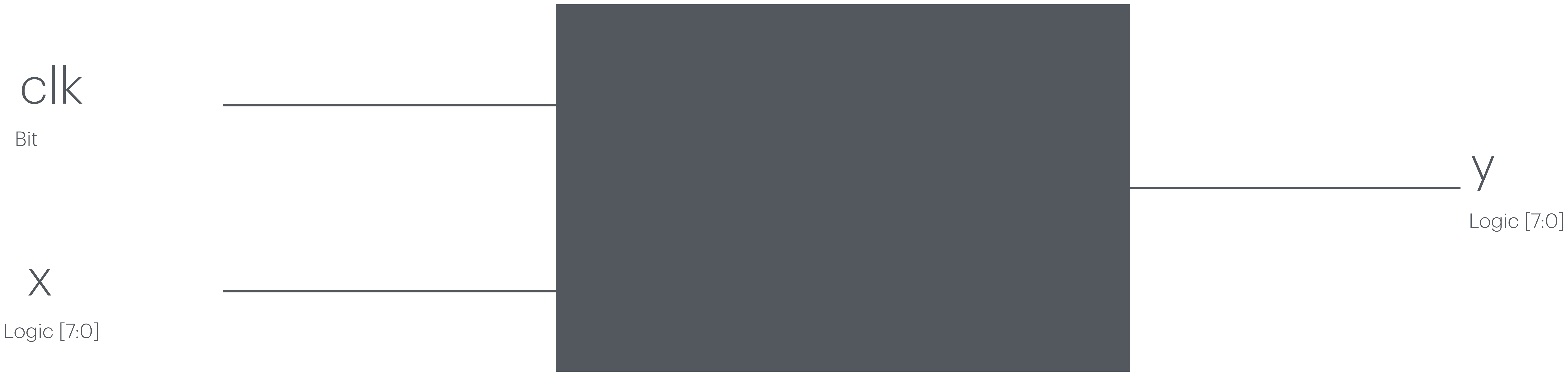


Continuous (combinatorial) assignment. Instantaneous signals changes are captured

Connect (wire)



Log2



clk

Bit

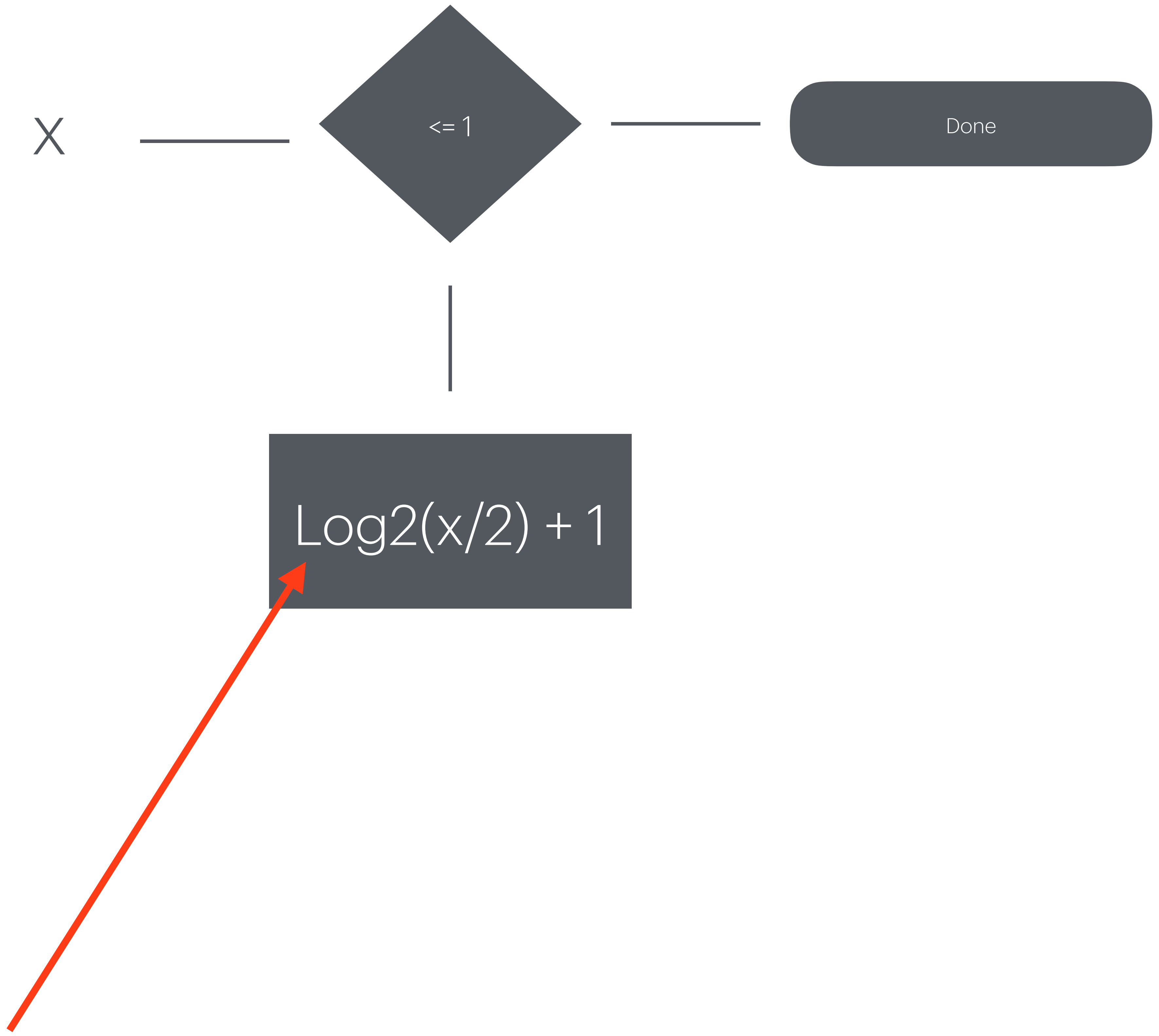
x

Logic [7:0]

y

Logic [7:0]

Log2



X

<= 1

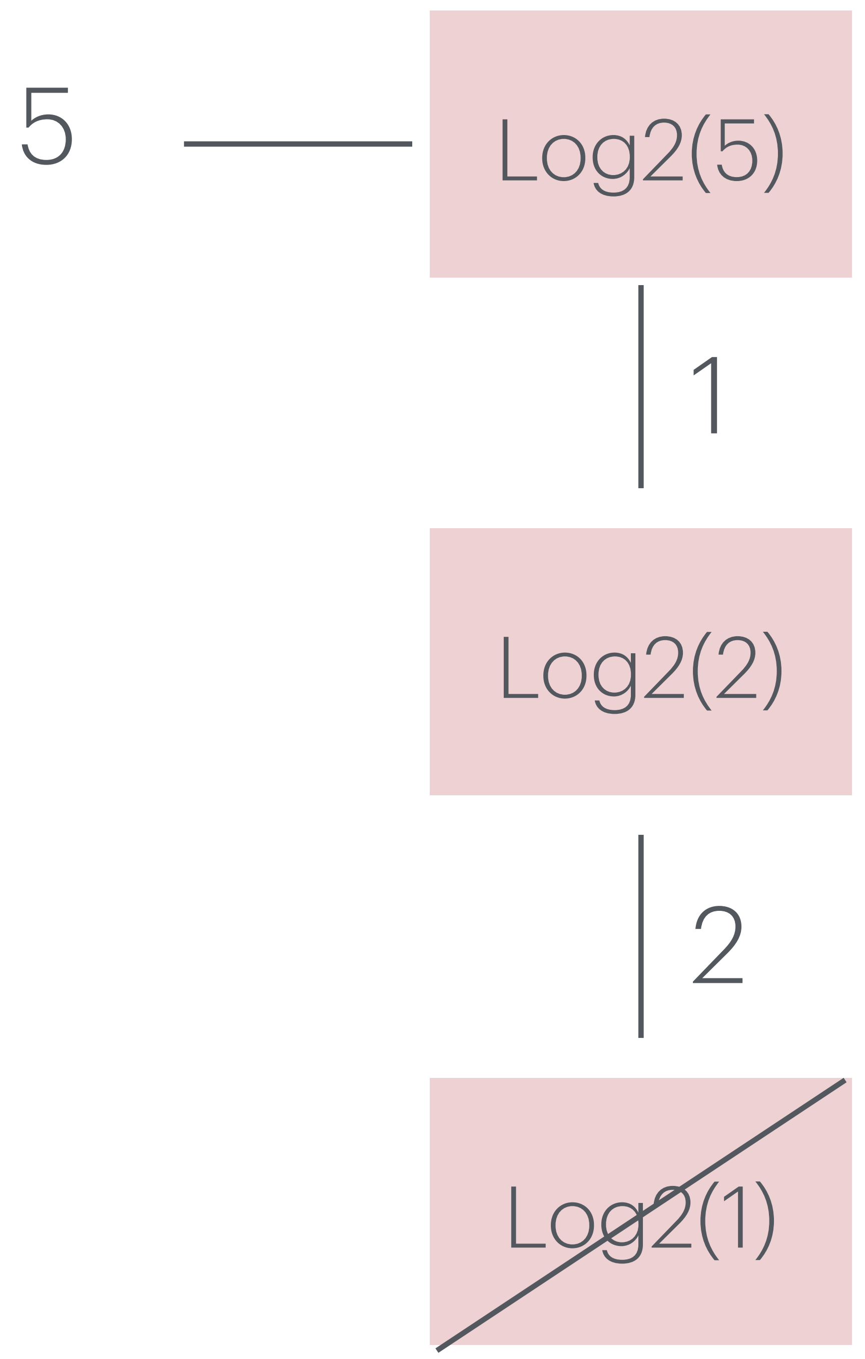
Done

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

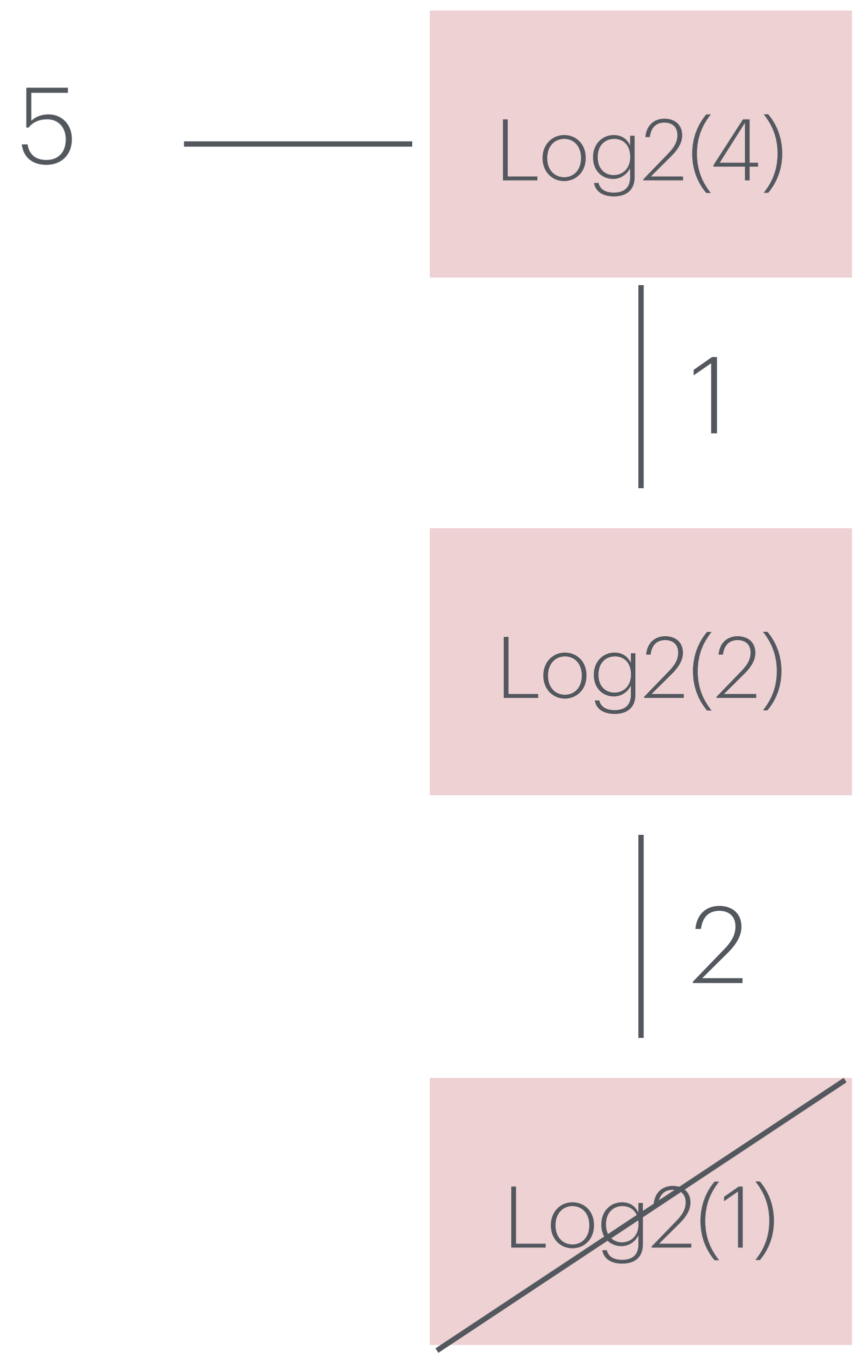
~~Log2(1)~~

Min = 2

$5 > 2^2$ (increment) ___ ___ ___

Log2(**4**)

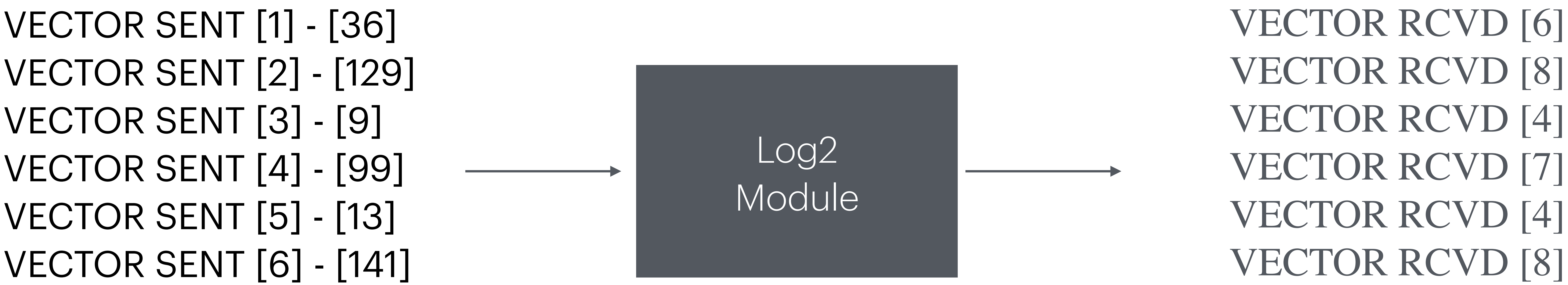
Minimum number of bits represent 5?



Min = 2

$4 \nless 2^2$ (perfect) ___ ___

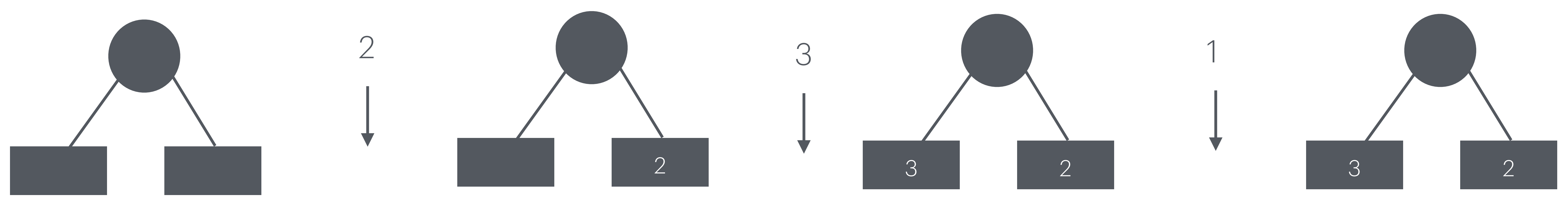
Log2 : Debug Results

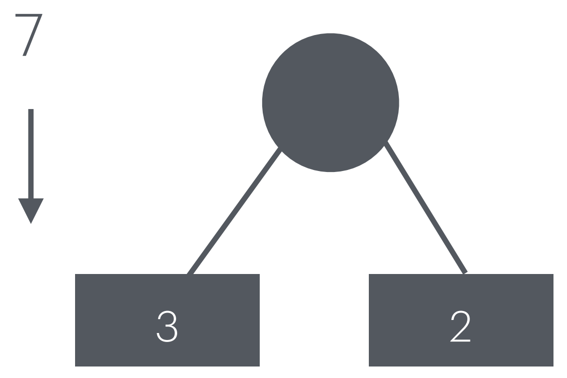


VECTOR SENT [1] - [36]
VECTOR SENT [2] - [129]
VECTOR SENT [3] - [9]
VECTOR SENT [4] - [99]
VECTOR SENT [5] - [13]
VECTOR SENT [6] - [141]

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

Second Largest

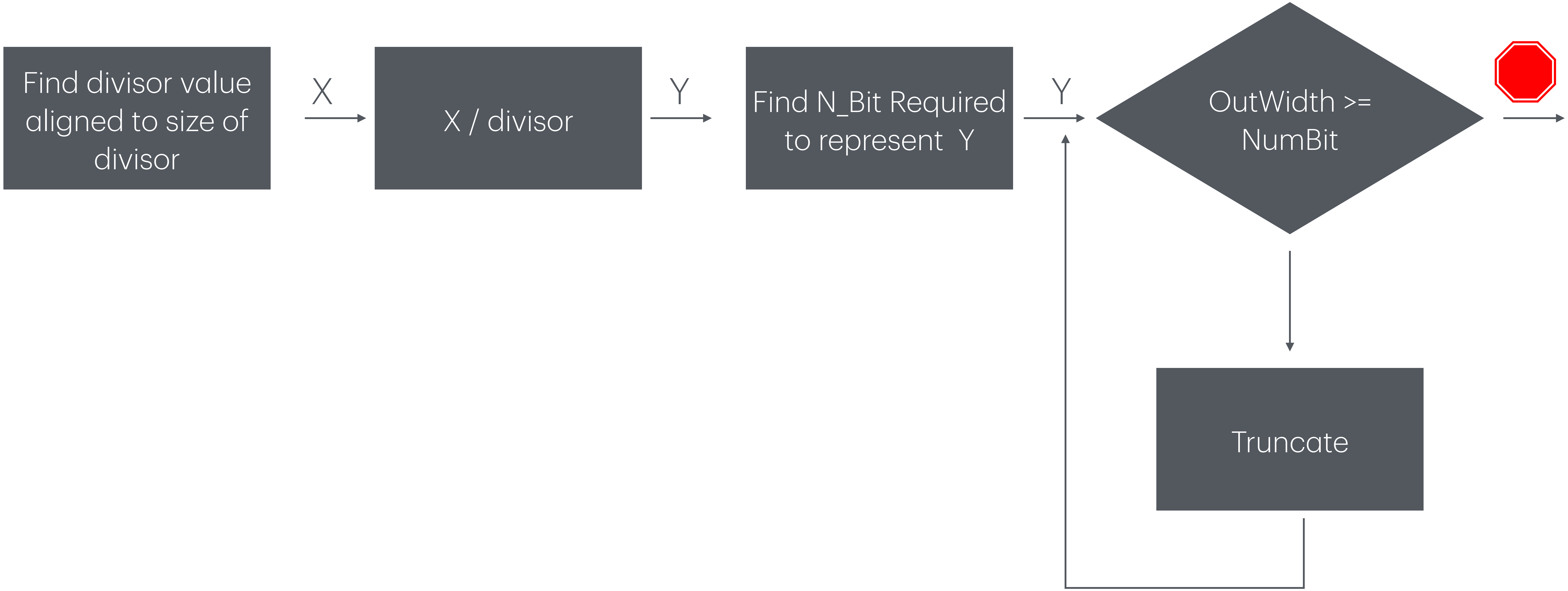




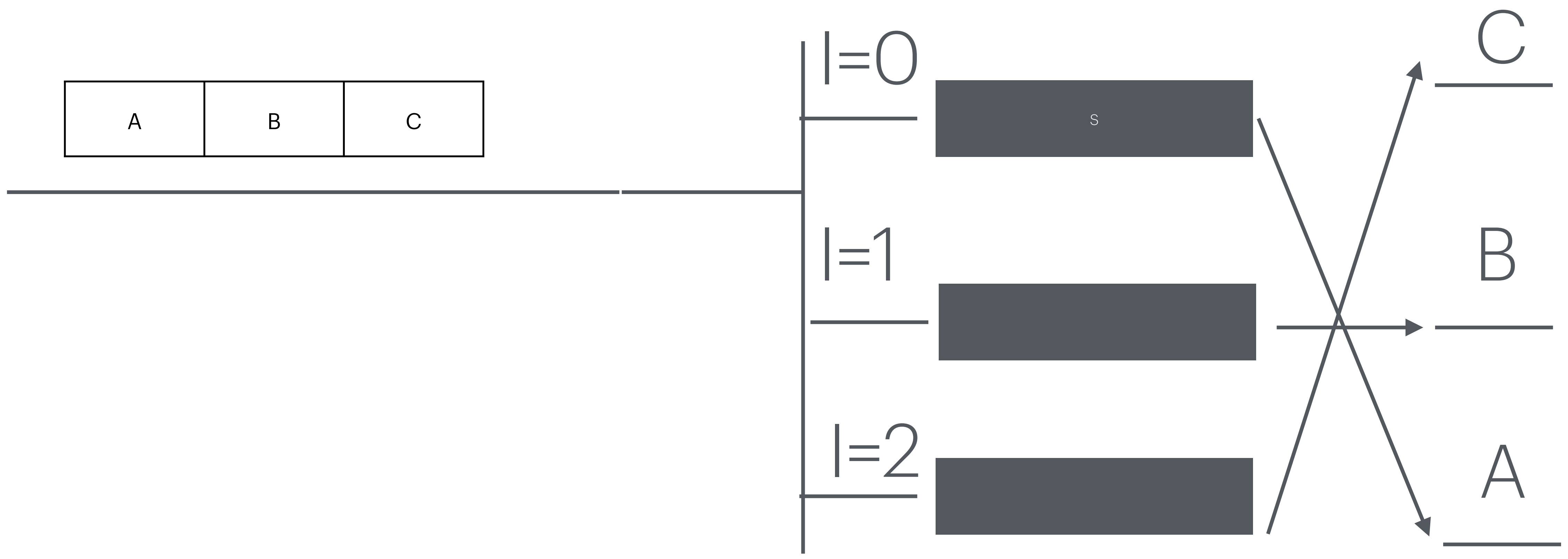
Count	0	1	2	3	4	
Data_In	D0	D1	D2	D3	D4	
2nd Largest	0	0	2	2	2	3

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

Rounded Division

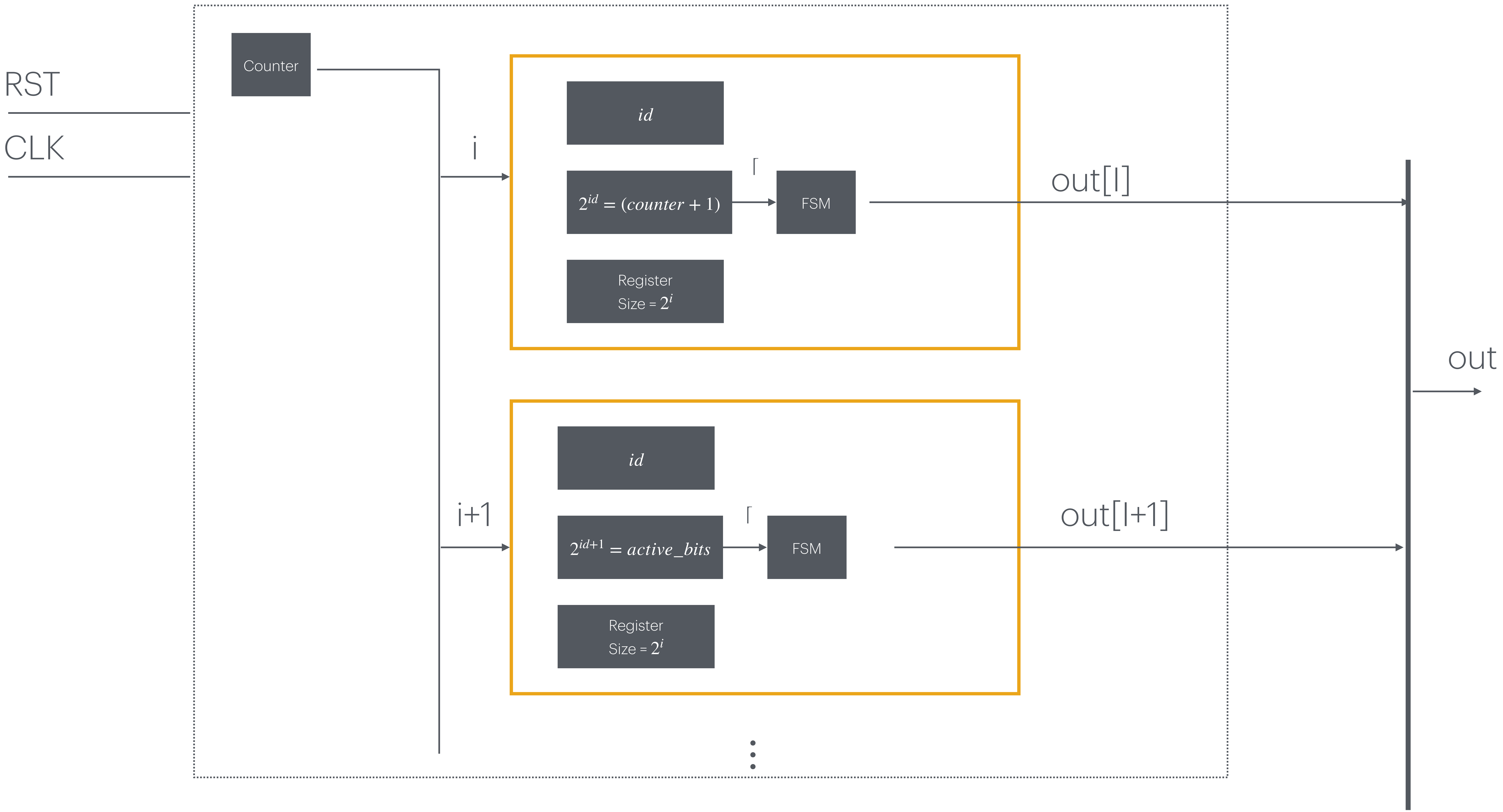

Find divisor value aligned to size of divisor
X
X / divisor
Y
Find N_Bit Required to represent Y
Y
OutWidth >= NumBit
Truncate

Reverse Bits



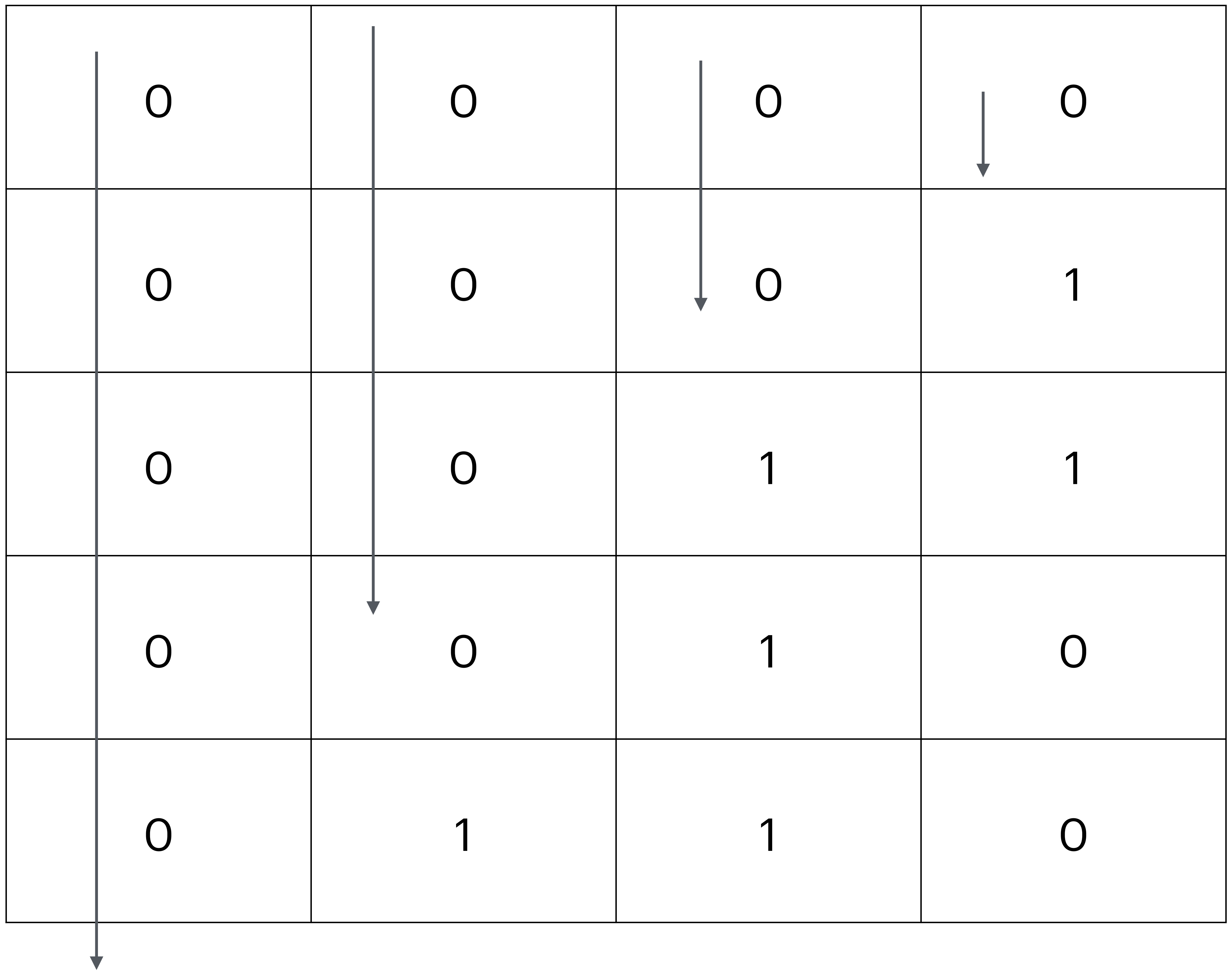
■ **Generate** Logic Blocks

Gray code


RST
CLK
Counter
i
i+1
id
$2^{id} = (counter + 1)$
⌈
FSM
Register
Size = 2^{i}
out[I]
id
$2^{id+1} = active\_bits$
⌈
FSM
Register
Size = 2^{i}
out[I+1]
out

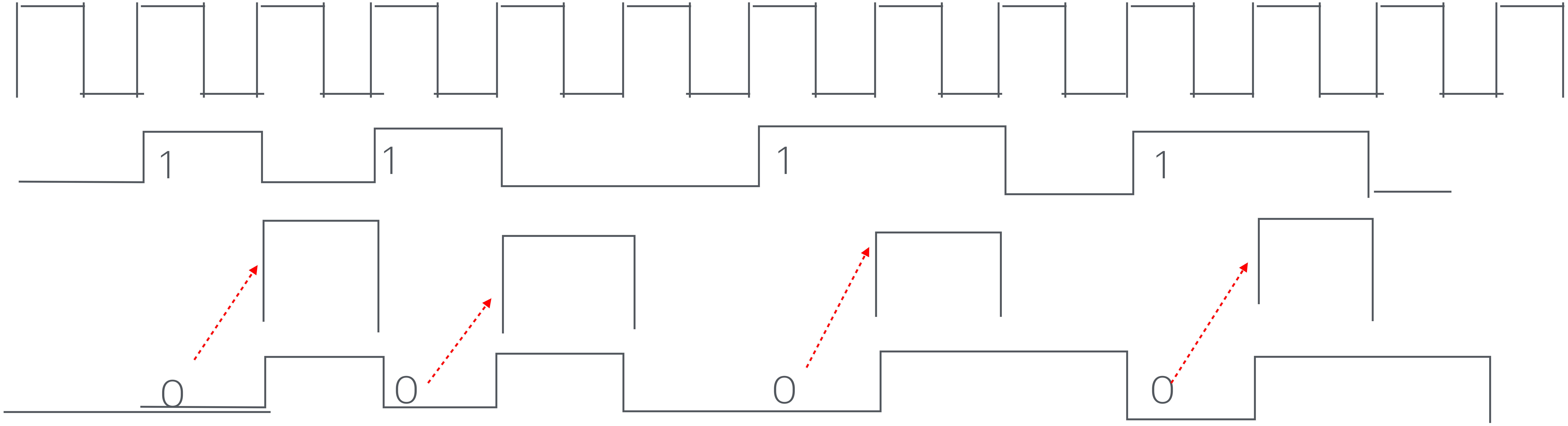
Gray code

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

msb	■	□	■	□	□	□	□	□	□	□	□	■	□	□	□	□	■	□	□	□	□	□
	□	□	■	□	□	□	□	■	□	□	□	□	□	□	□	□	■	□	□	□	□	□
	■	■	■	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□	□
lsb	□	■	■	□	□	□	□	□	■	□	□	□	□	□	□	□	□	□	□	□	□	□
	A		F					4	1			8					C					

Bit Sequence →

D_IN →

Enable

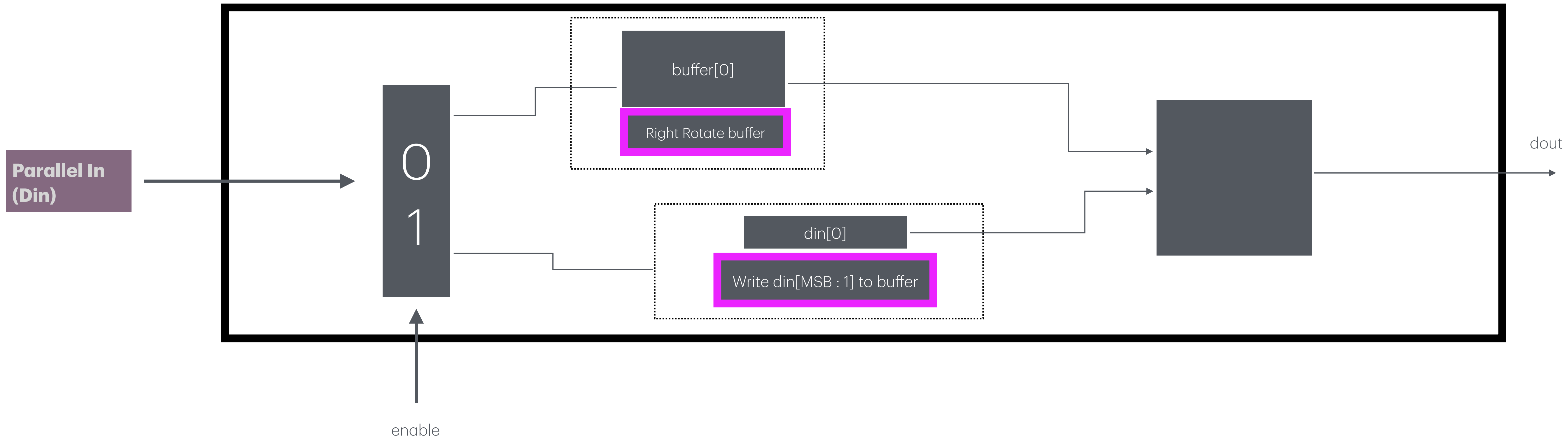
■	□	■	□	□	□	□	■	■	□	□	■	□	□	□	□	■	□	□	□	□	□
1							1														

RST

■	□	□	□	□	□	□	■	□	□	□	□	□	□	□	□	□	□	□	□	□	□

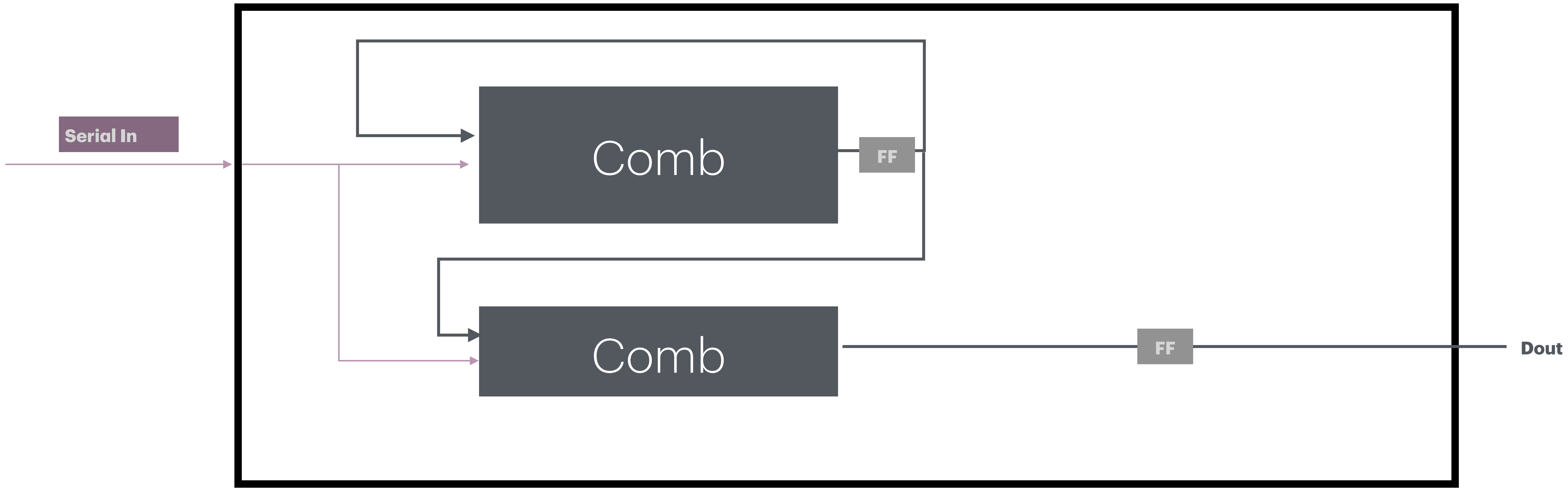
Output

□	□	□	■	■	■	■	□	□	■	□	□	□	□	□	■	□	□	■	■	□	□

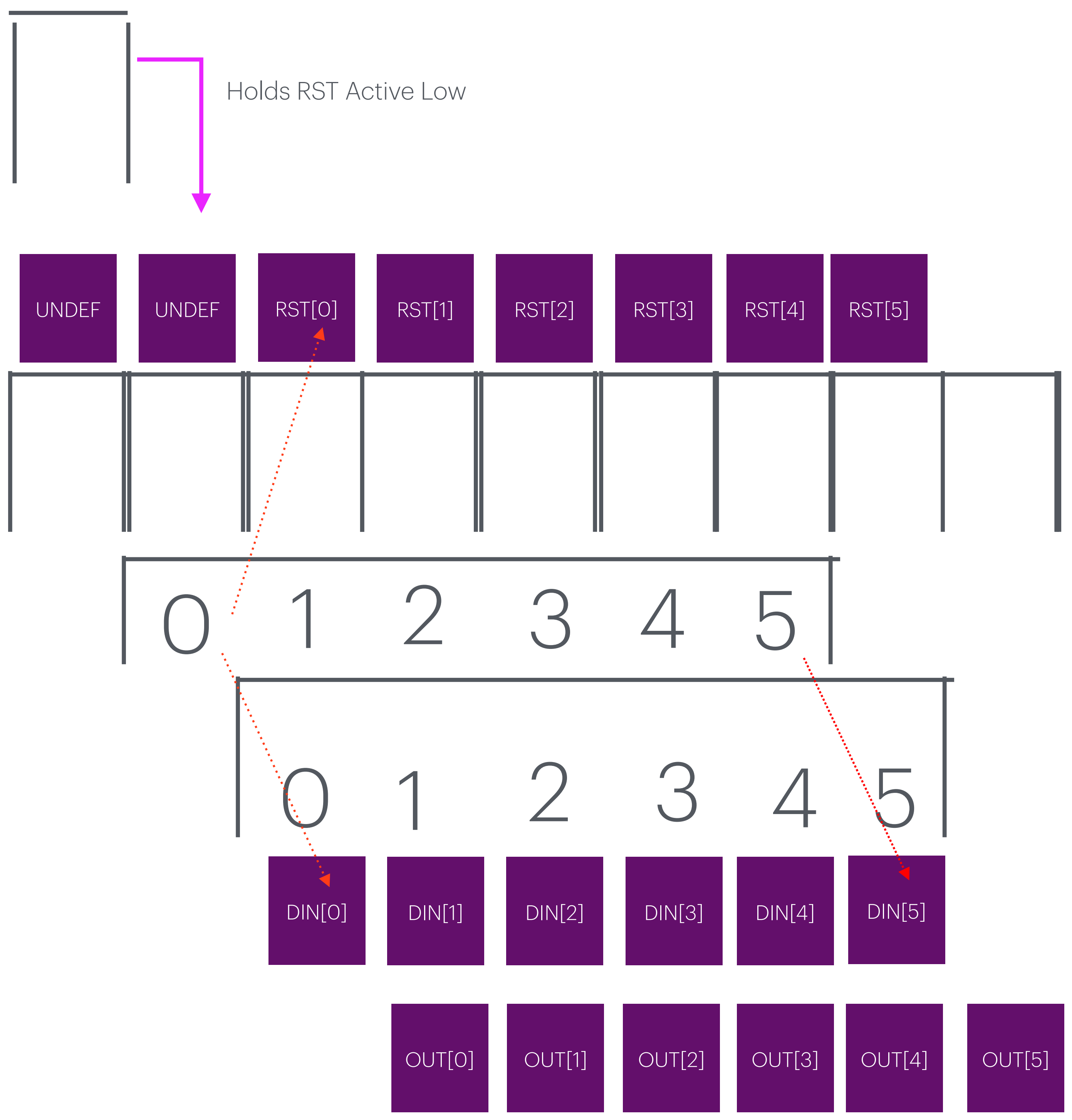


Serial to Parallel

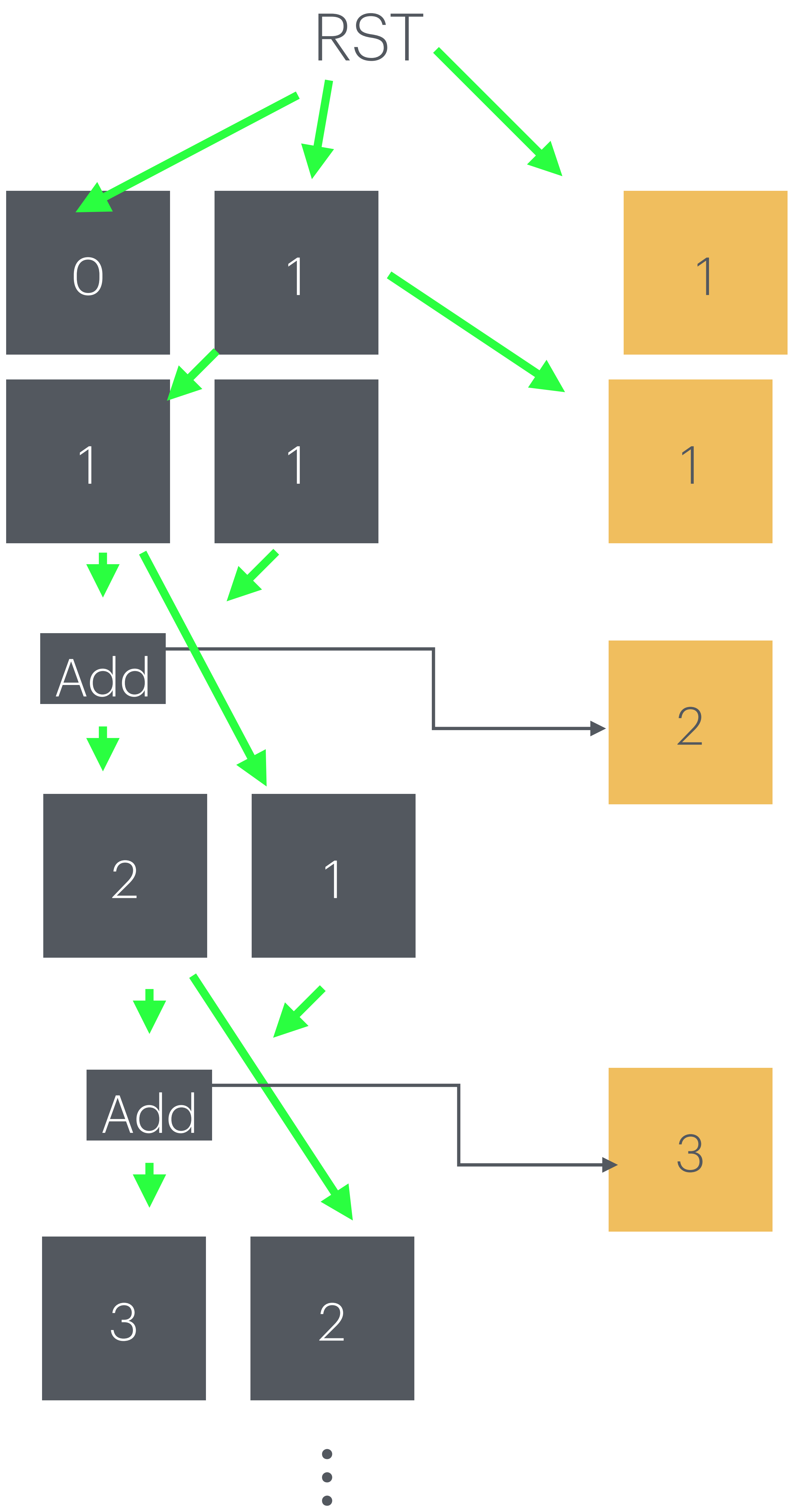
Din	■	■	□	■	□	■	■	□	□	■	□	■	■	■	□	□	□	■	■	■	
Reset	□	□	□	□	□	□	▣	□	□	▣	□	□	□	□	▣	□	□	□	□	□	
Data	_0000	_0001	_0011	_0110	_1101	_1010	_0101	_0000	_0000	_0000	_0000	_0000	_0001	_0011	_0111	_0000	_0000	_0000	_0001	_0011	_0111



Serial to Parallel (Simulation Concept)

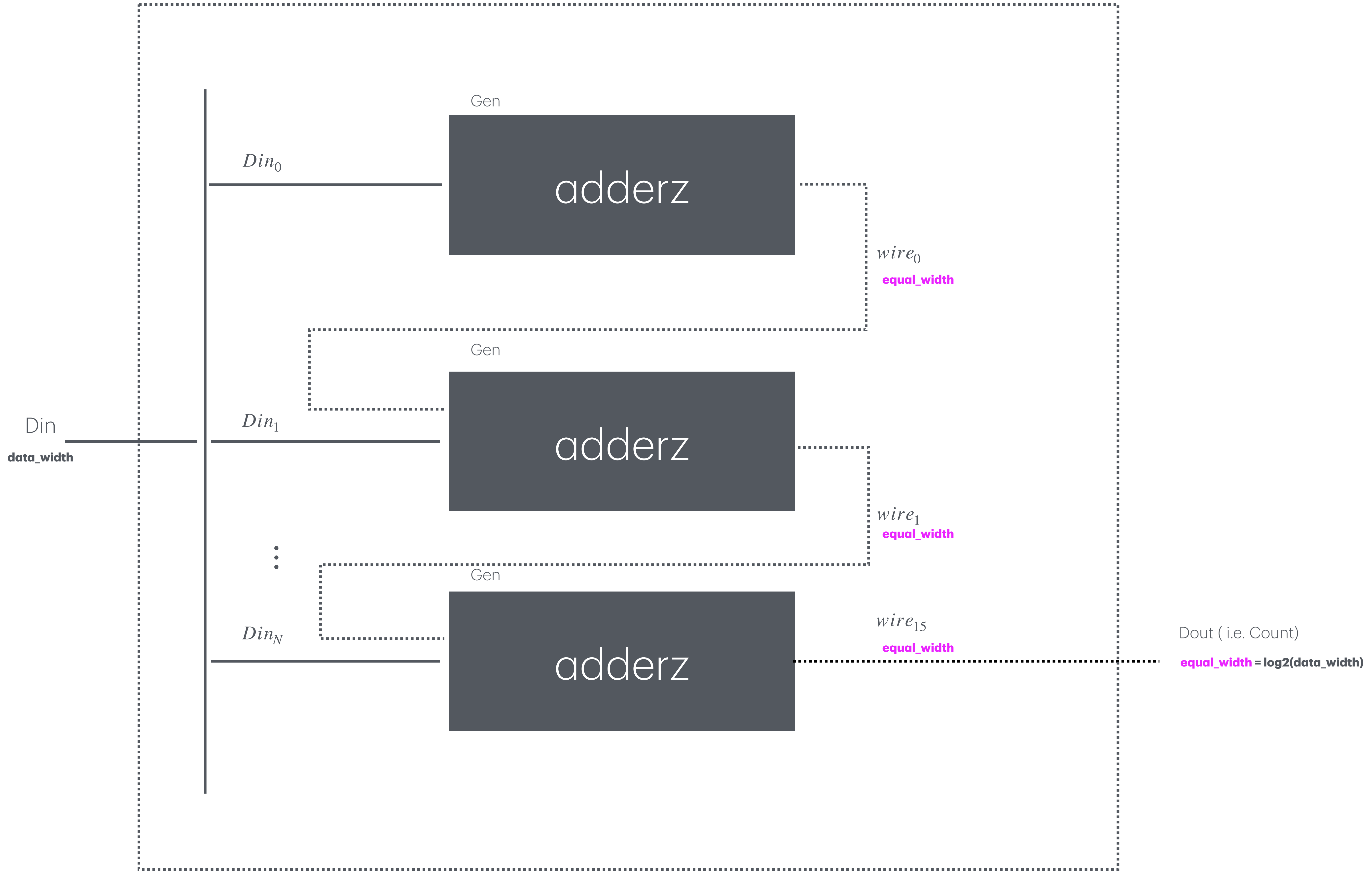


Fibonacci

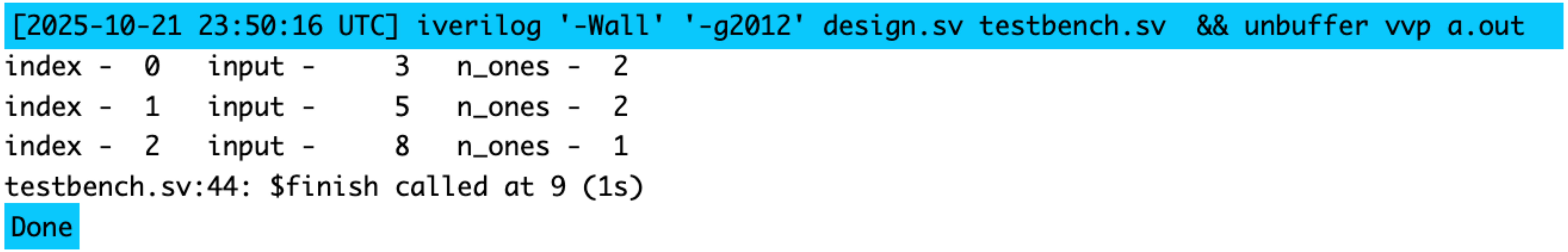


Count Ones

Architecture Similar to Linked List

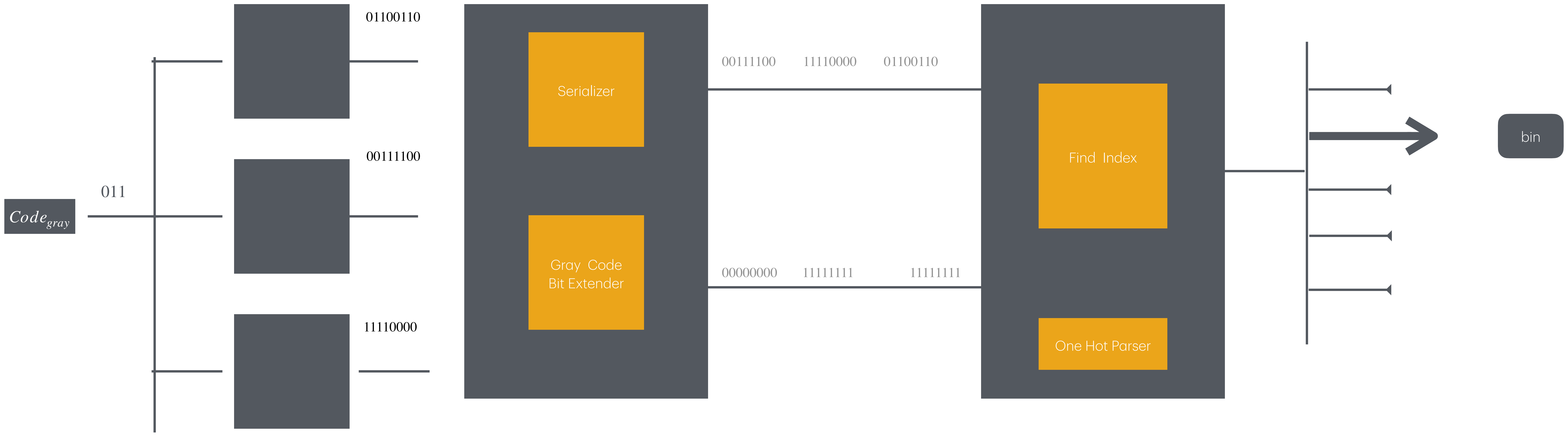


Count Ones

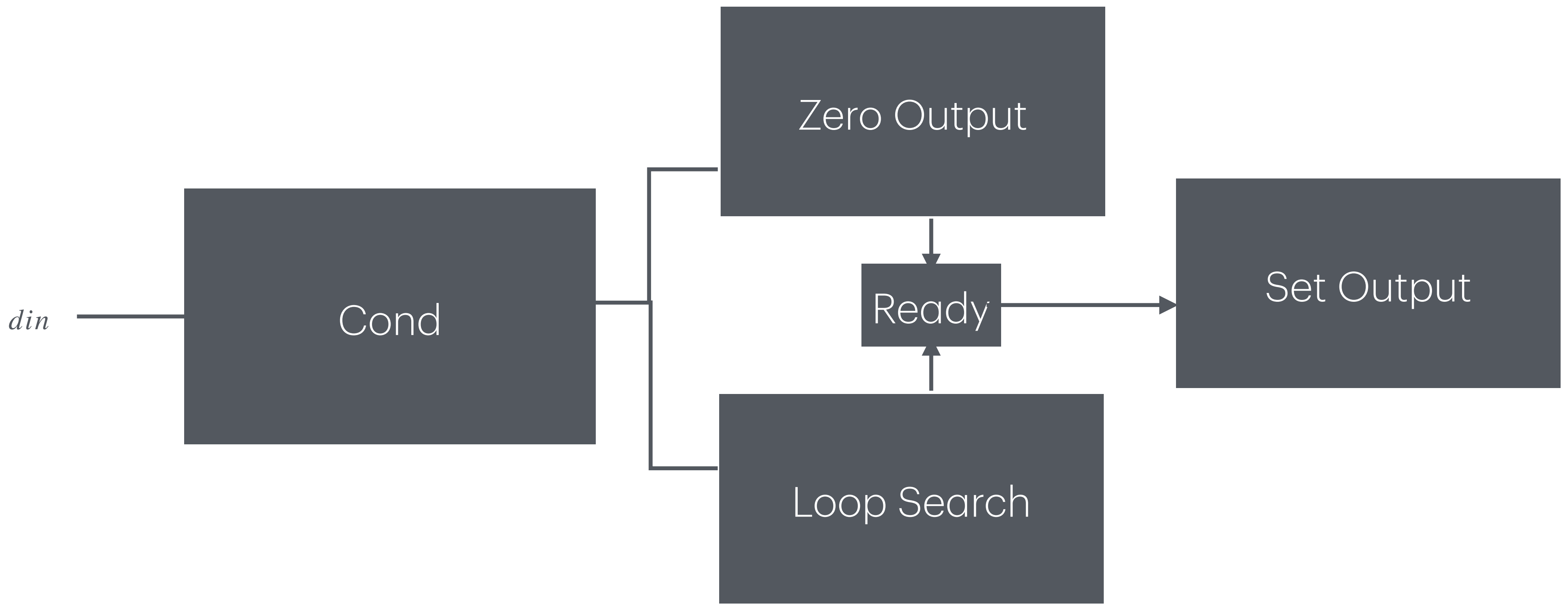


```
[2025-10-21 23:50:16 UTC] iverilog '-Wall' '-g2012' design.sv testbench.sv  && unbuffer vvp a.out
index -  0   input -     3   n_ones -  2
index -  1   input -     5   n_ones -  2
index -  2   input -     8   n_ones -  1
testbench.sv:44: $finish called at 9 (1s)
Done
```

Gray Code to Binary (Width = 3)



Trailing Ones



StopWatch Timer

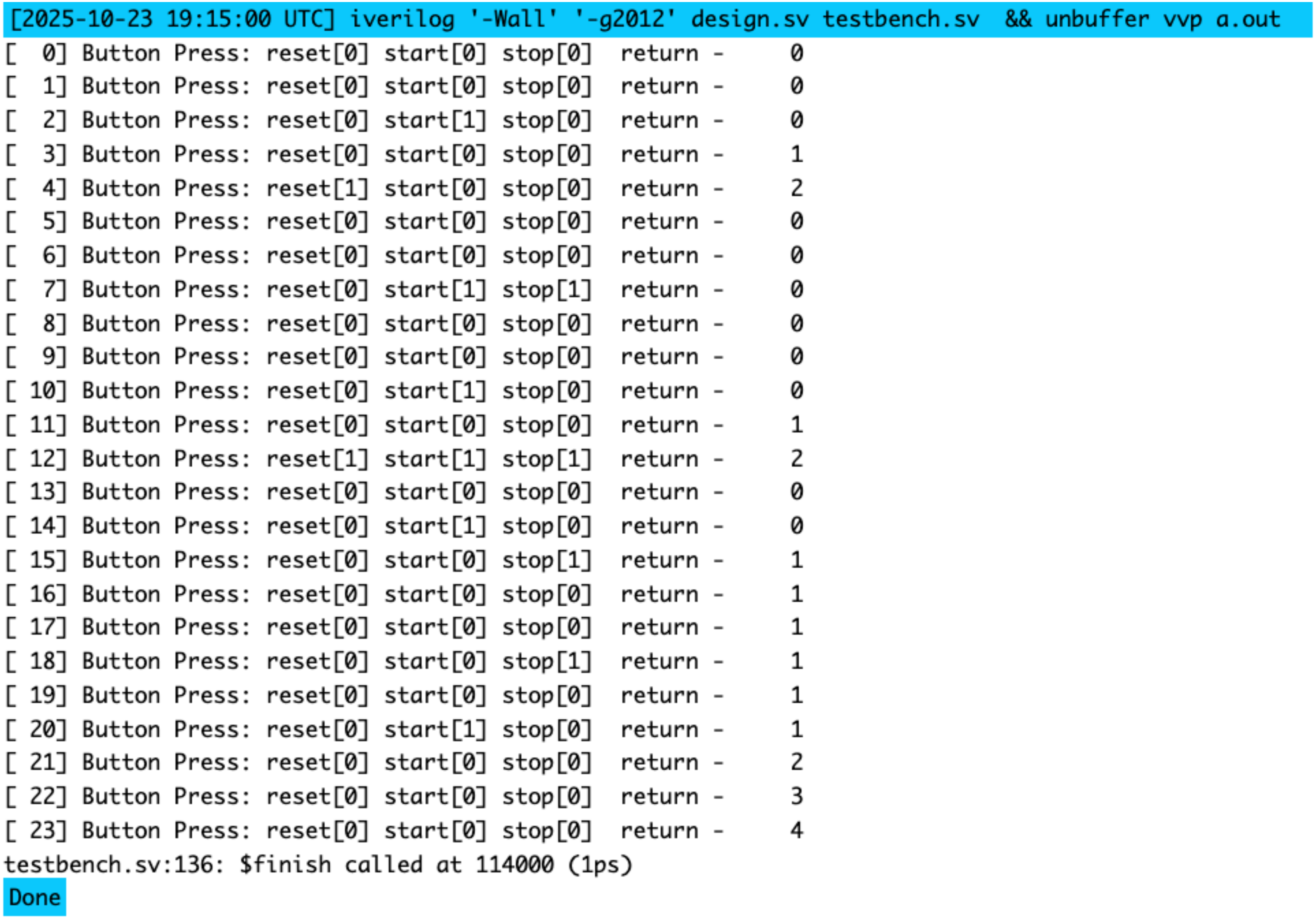
TICK	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	
START			🟩	┈	┈			🟩			🟩	┈	🟩		🟩	┈					🟩	┈	┈	┈
RESET					🟩								🟩											
STOP								🟩	┈	┈	┈		🟩			🟩			🟩					
COUNT	0	0	0	1	2	0	0	0	0	0	0	1	2	0	0	1	1	1	1	1	1	2	3	4

Testbench **monitors rdy** at this time stamp

StopWatch Timer

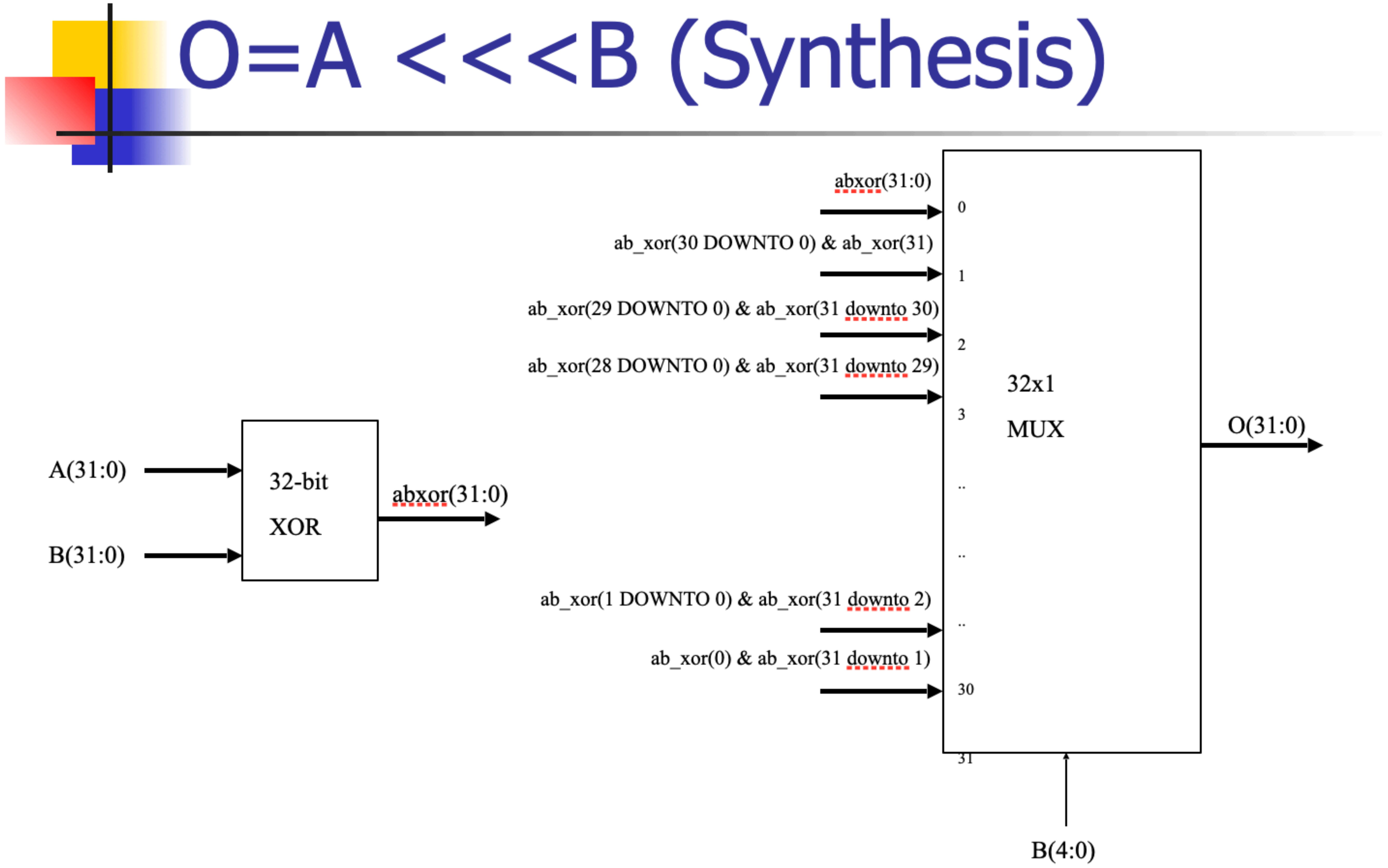
prev_button

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
```

Random Circuit 1

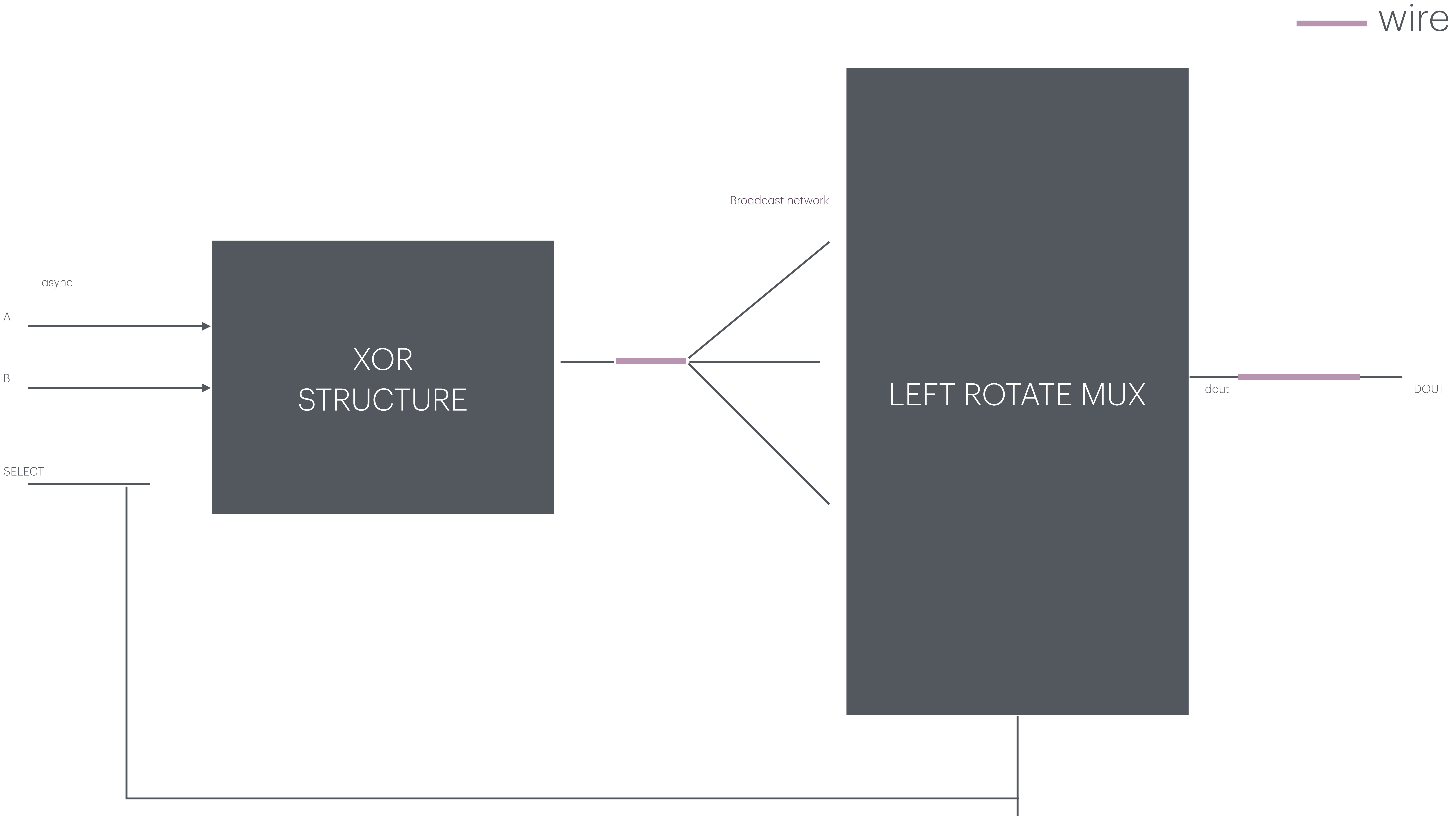


O=A <<<B (Synthesis)

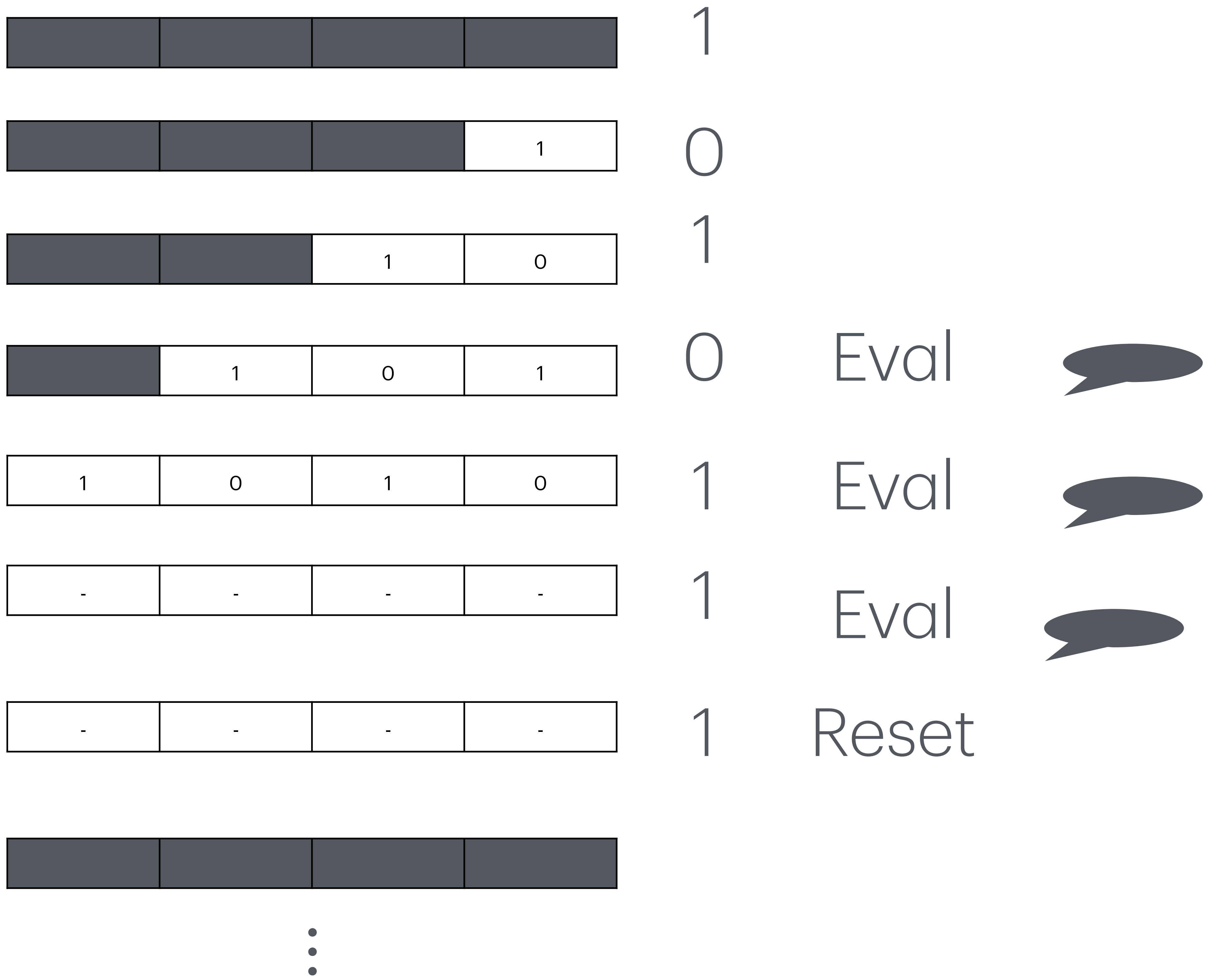
A(31:0)
B(31:0)
32-bit
XOR
abxor(31:0)
abxor(31:0)
ab_xor(30 DOWNTO 0) & ab_xor(31)
ab_xor(29 DOWNTO 0) & ab_xor(31 downto 30)
ab_xor(28 DOWNTO 0) & ab_xor(31 downto 29)
ab_xor(1 DOWNTO 0) & ab_xor(31 downto 2)
ab_xor(0) & ab_xor(31 downto 1)
0
1
2
3
..
..
..
30
31
32x1
MUX
O(31:0)
B(4:0)

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

Random Circuit 1



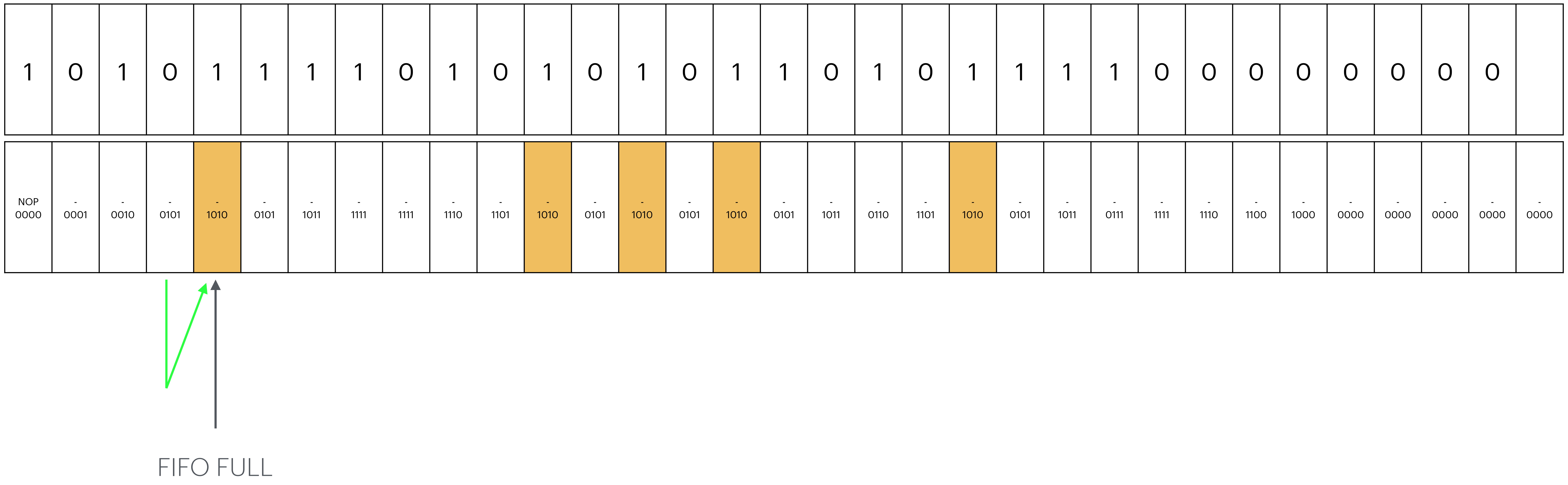
Sequence Detector



▨	▨	▨	▨	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	- 0000

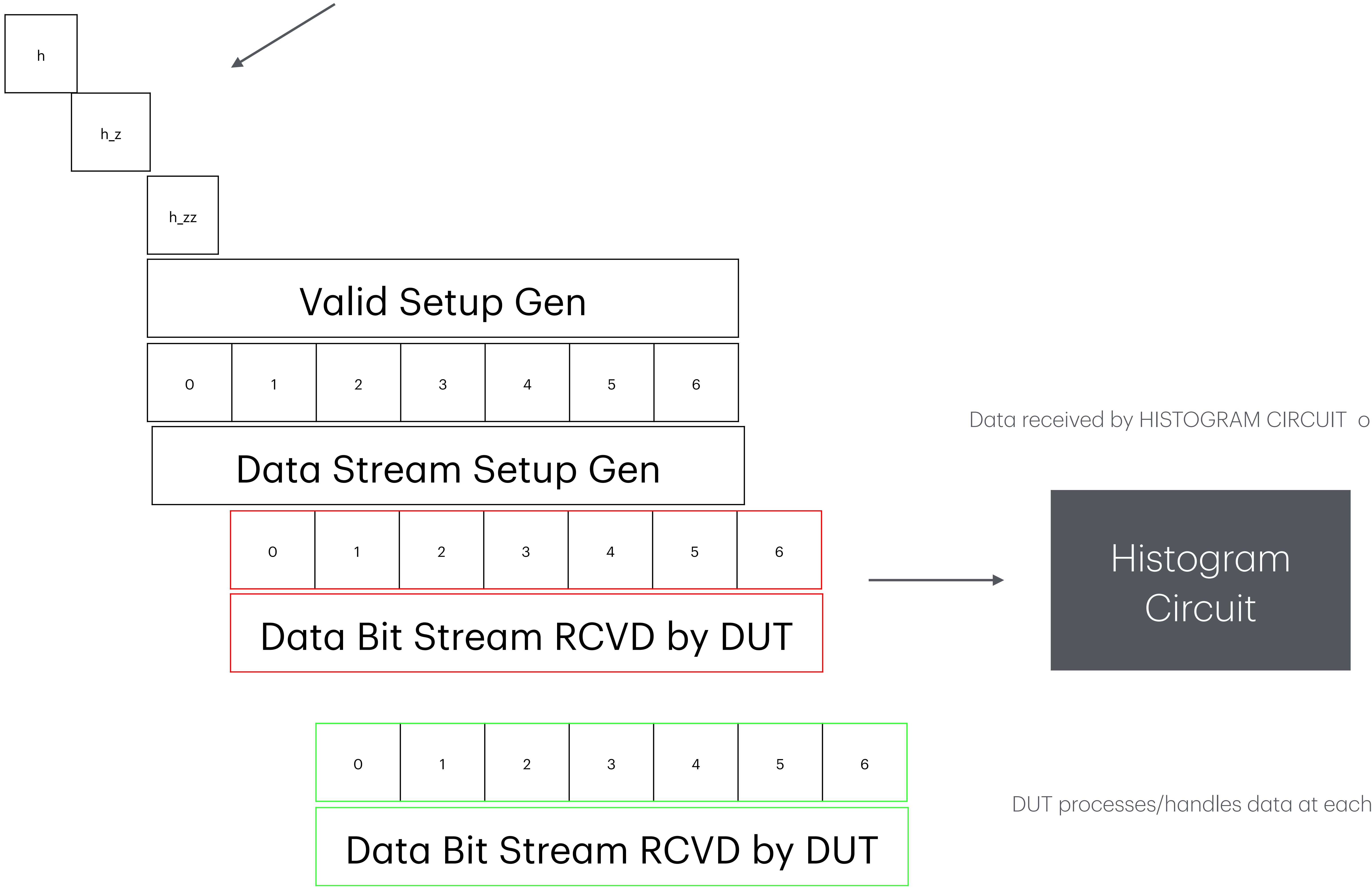
FIFO FULL

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

Data received by HISTOGRAM CIRCUIT on next clk edge!

Histogram Circuit

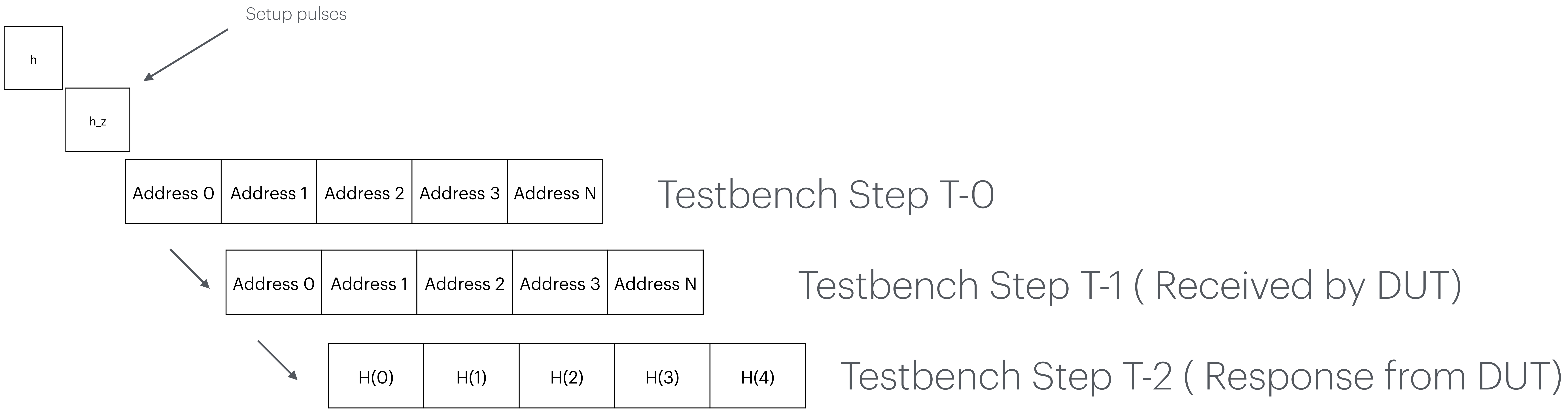
0	1	2	3	4	5	6

Data Bit Stream RCVD by DUT

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

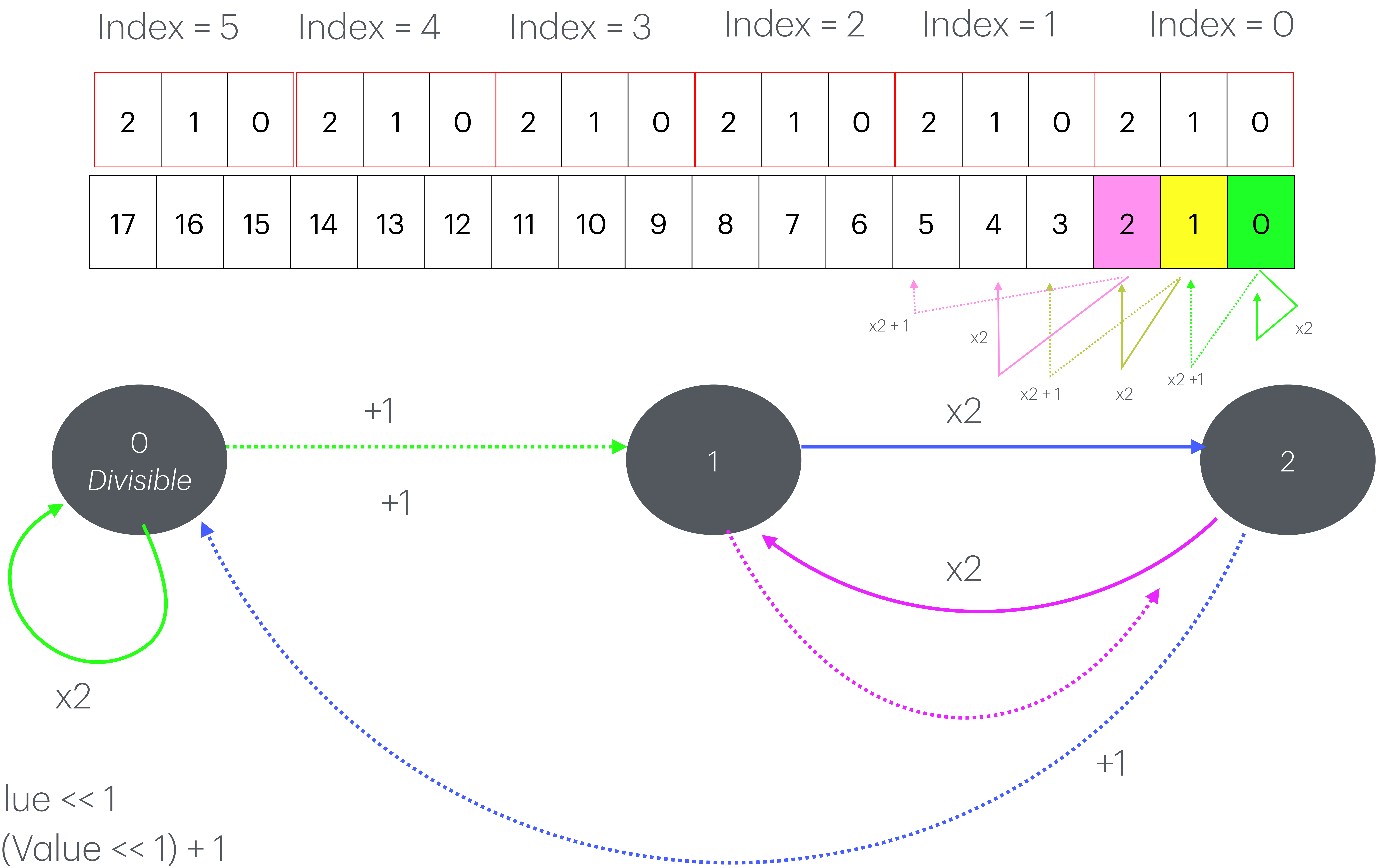
Address 0	Address 1	Address 2	Address 3	Address N

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



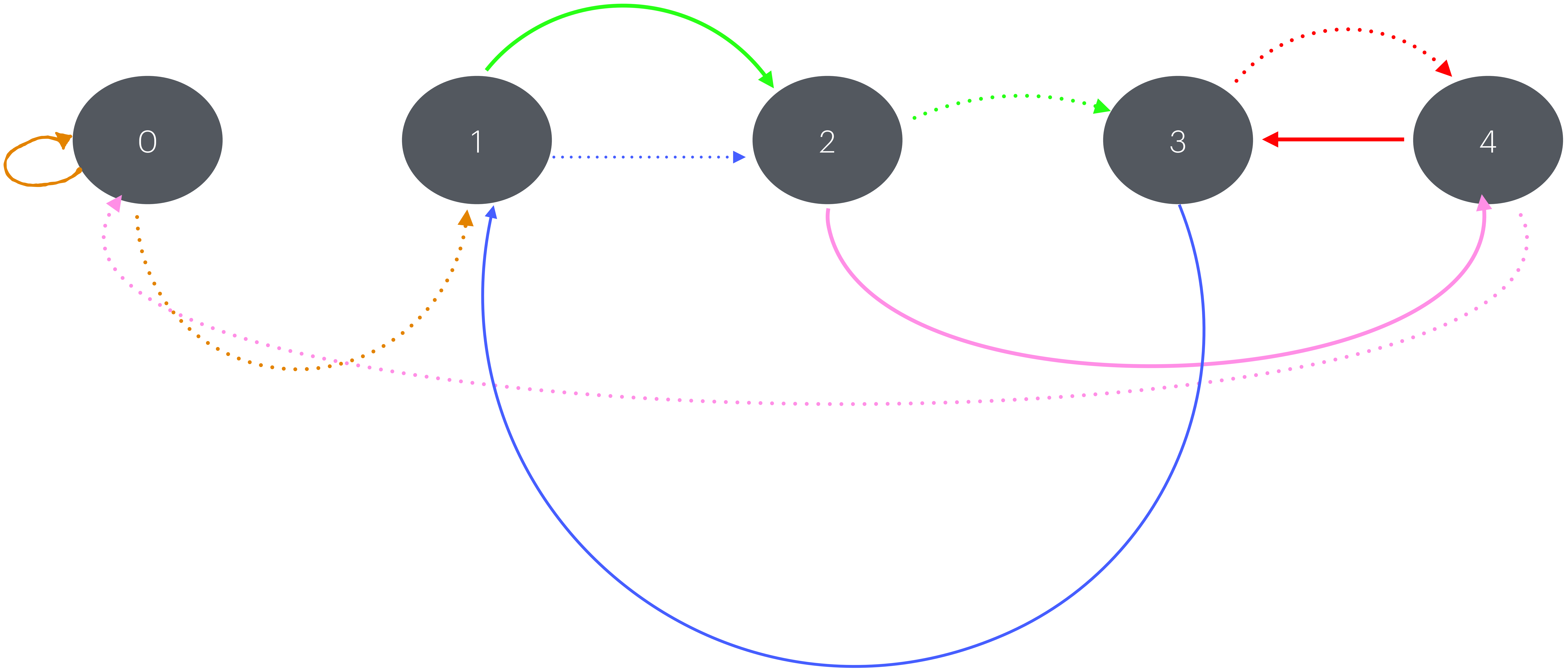
Index = 5			Index = 4			Index = 3			Index = 2			Index = 1			Index = 0		
2	1	0	2	1	0	2	1	0	2	1	0	2	1	0	2	1	0
17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

State Flow

x2 = Value << 1

x2 + 1= (Value << 1) + 1

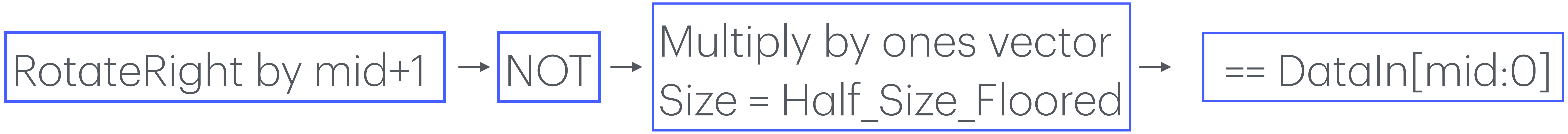
Divisible By Five



Palindrome

Odd Data Width

0	1	0	1	0

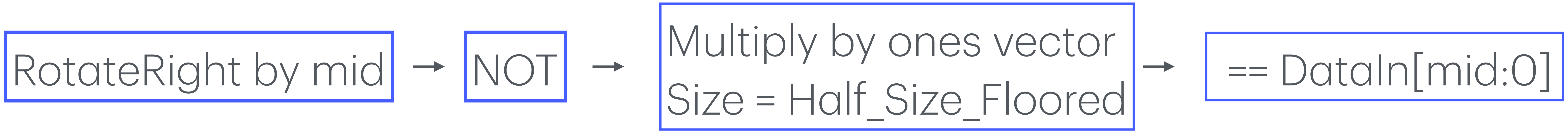


RotateRight by mid+1 → NOT → Multiply by ones vector Size = Half_Size_Floored → == DataIn[mid:0]

Half_Size_Floored = 2

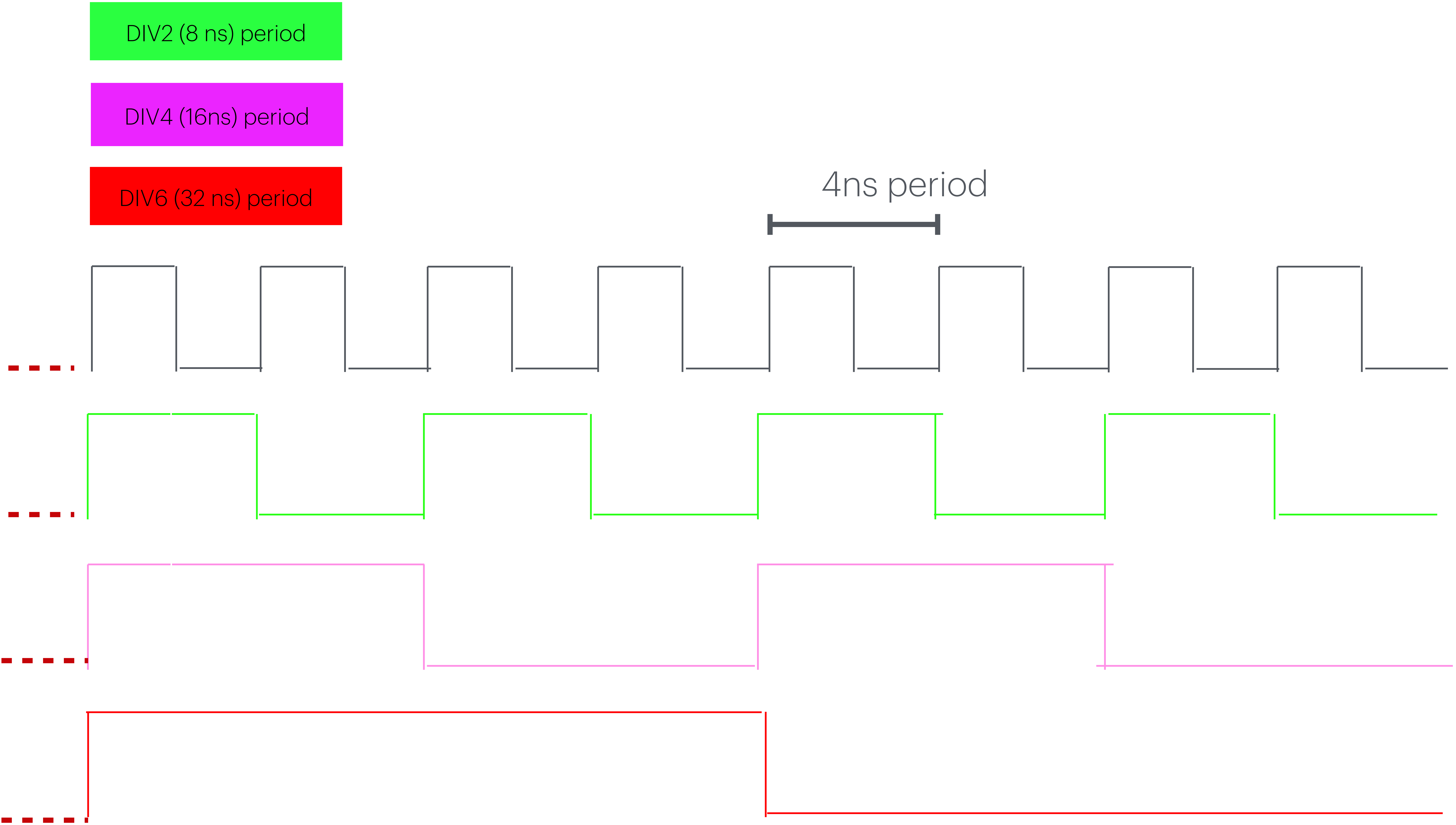
Even Data Width

0	1	1	1



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

Divide-By-Events

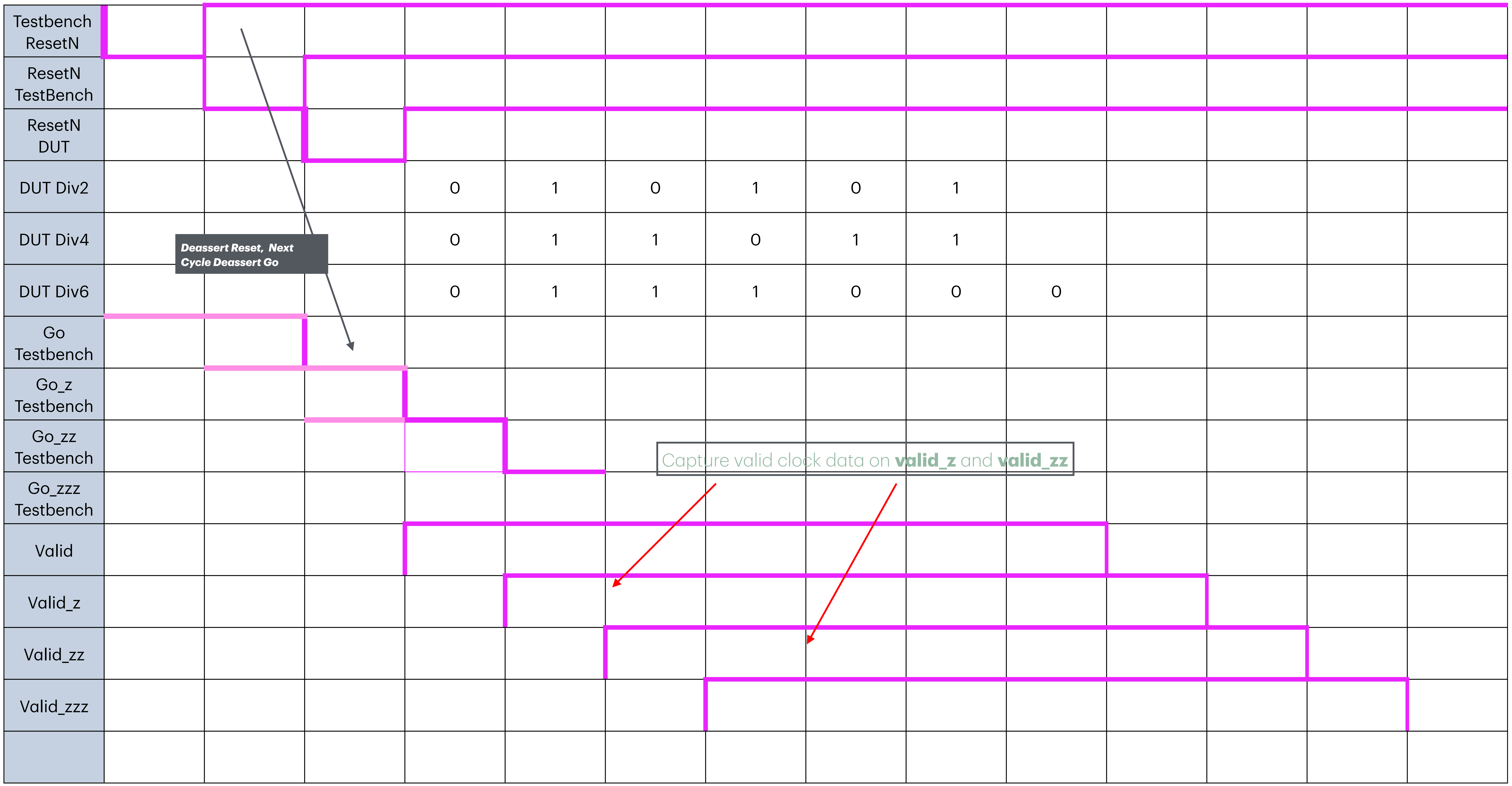


🟩 DIV2 (8 ns) period

🟪 DIV4 (16ns) period

🟥 DIV6 (32 ns) period

Divide-By-Events Timing

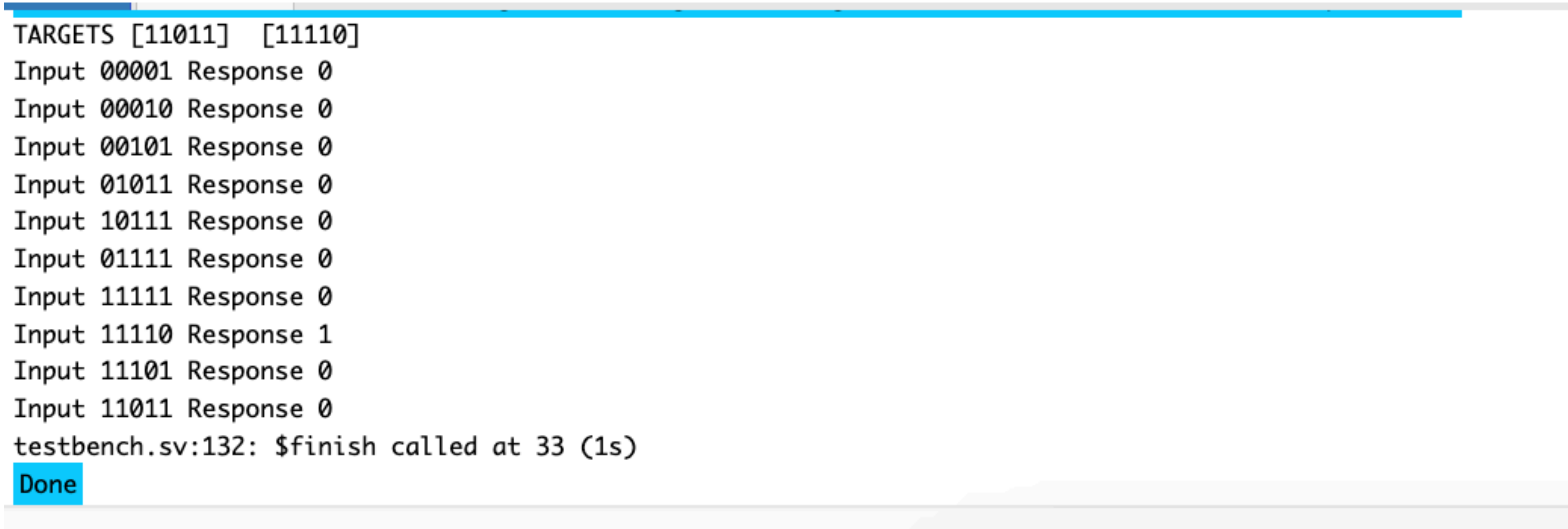


Testbench ResetN														
ResetN TestBench														
ResetN DUT														
DUT Div2				0	1	0	1	0	1					
DUT Div4				0	1	1	0	1	1					
DUT Div6				0	1	1	1	0	0	0				
Go Testbench														
Go_z Testbench														
Go_zz Testbench														
Go_zzz Testbench														
Valid			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

Deassert Reset, Next Cycle Deassert Go

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

Testbench Log

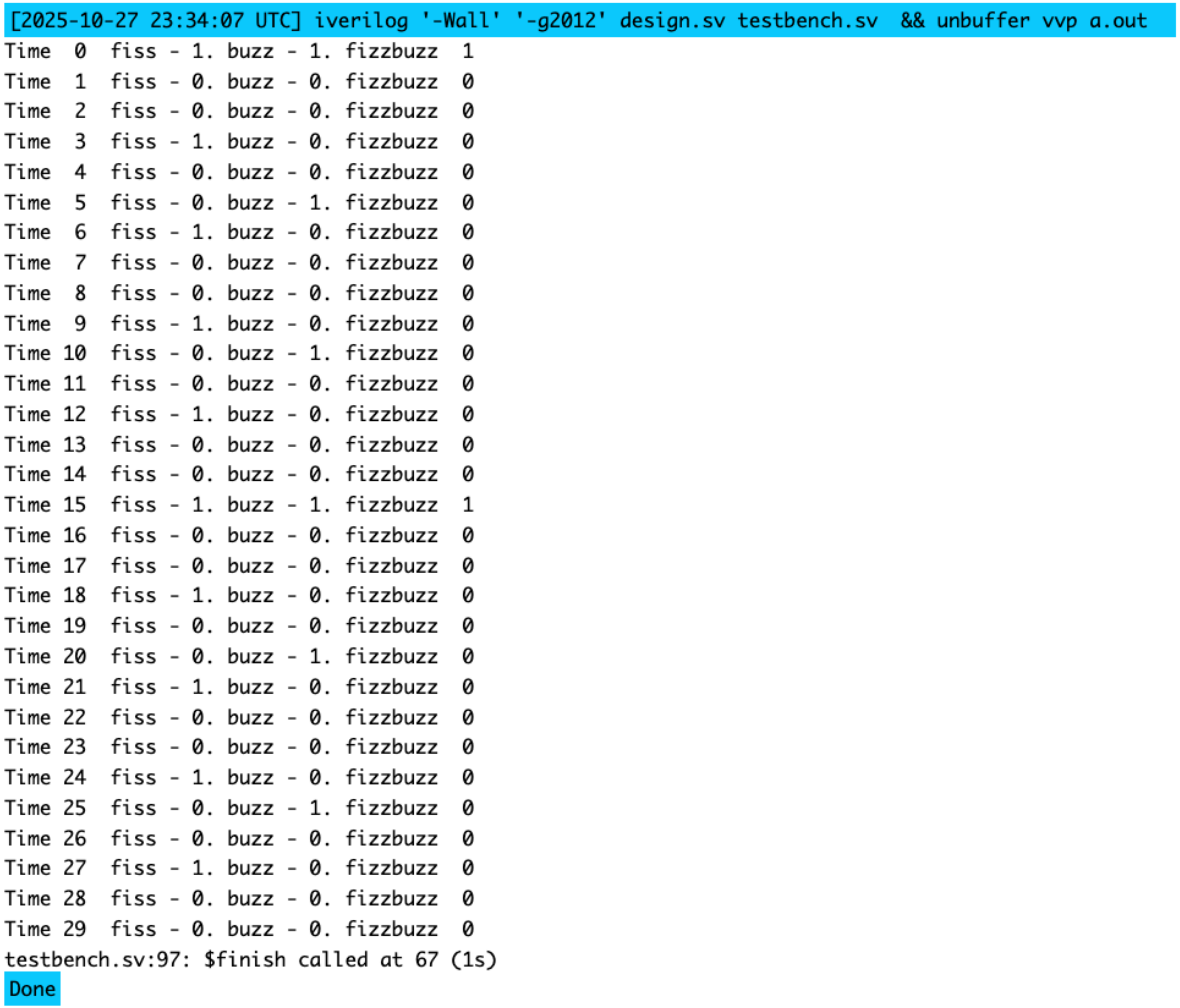


```
TARGETS [11011]  [11110]
Input 00001 Response 0
Input 00010 Response 0
Input 00101 Response 0
Input 01011 Response 0
Input 10111 Response 0
Input 01111 Response 0
Input 11111 Response 0
Input 11110 Response 1
Input 11101 Response 0
Input 11011 Response 0
testbench.sv:132: $finish called at 33 (1s)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

FizzBuzz

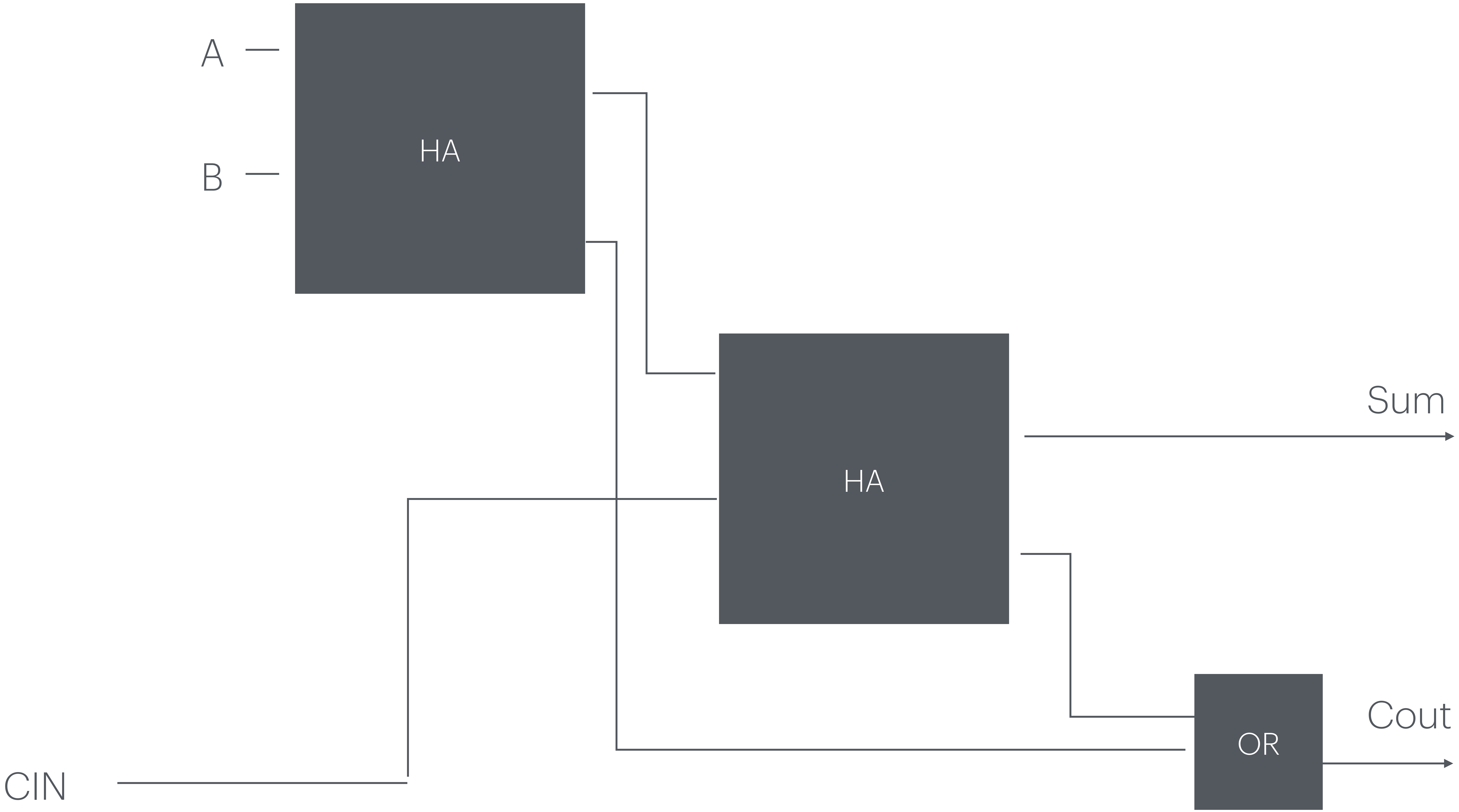
12	11	10	9	8	7	6	5	4	3	2	1	0	Tick
2	1	0	4	3	2	1	0	0	4	2	1	0	Fizz Mod
0	2	1	0	2	1	0	2	1	0	2	1	0	Buzz Mod

FizzBuzz

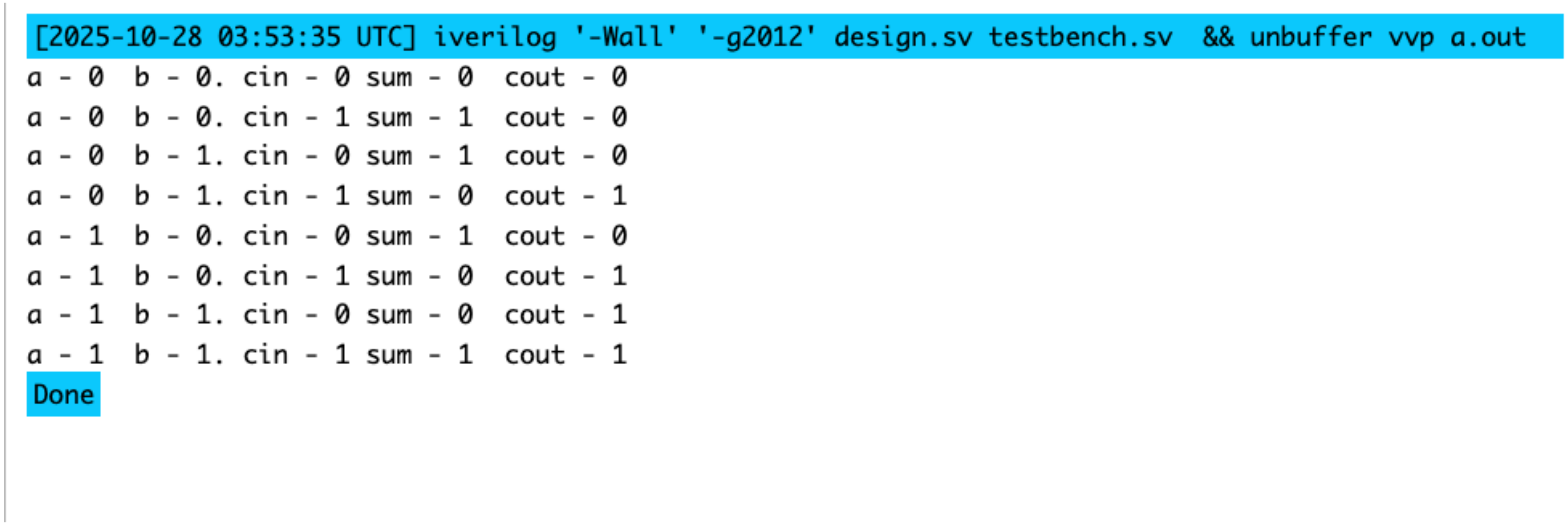


```
[2025-10-27 23:34:07 UTC] iverilog '-Wall' '-g2012' design.sv testbench.sv  && unbuffer vvp a.out
Time  0  fiss - 1. buzz - 1. fizzbuzz  1
Time  1  fiss - 0. buzz - 0. fizzbuzz  0
Time  2  fiss - 0. buzz - 0. fizzbuzz  0
Time  3  fiss - 1. buzz - 0. fizzbuzz  0
Time  4  fiss - 0. buzz - 0. fizzbuzz  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



Full Adder

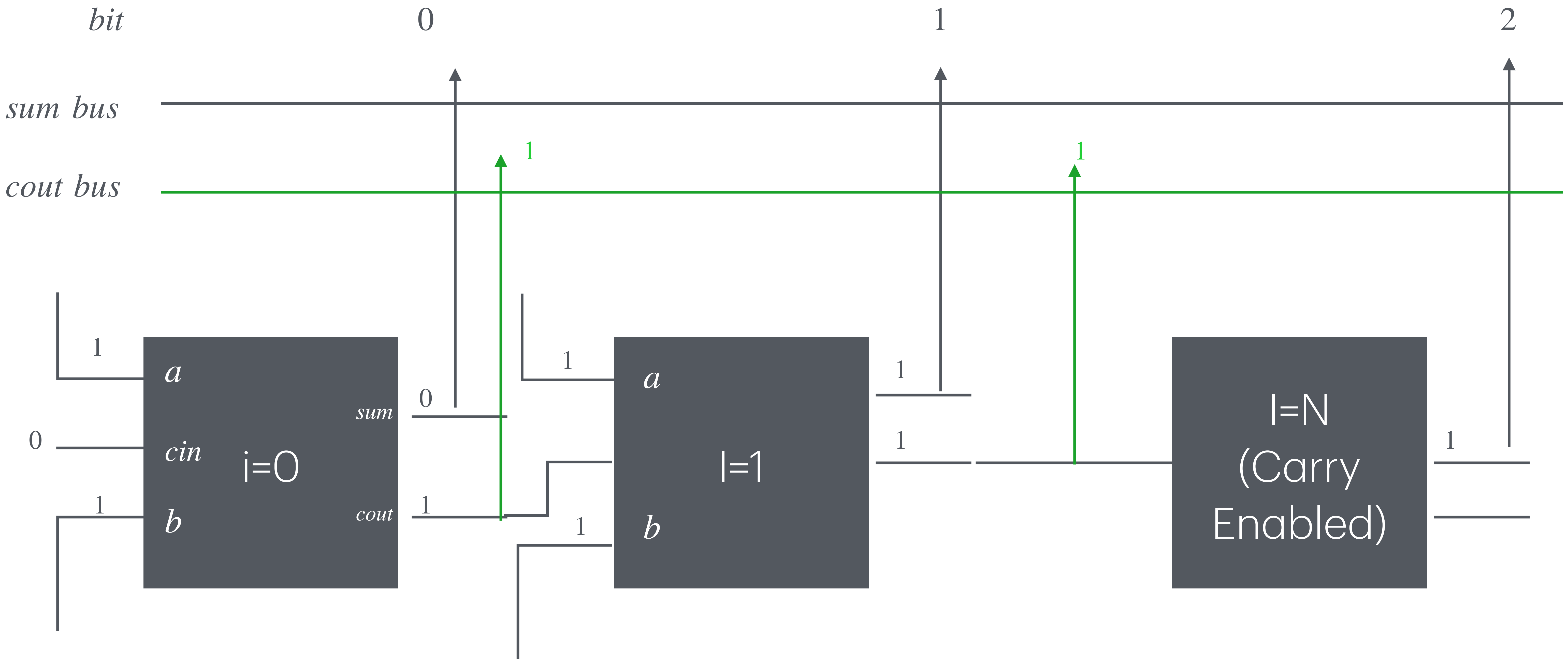


```
[2025-10-28 03:53:35 UTC] iverilog '-Wall' '-g2012' design.sv testbench.sv  && unbuffer vvp a.out
a - 0  b - 0. cin - 0 sum - 0  cout - 0
a - 0  b - 0. cin - 1 sum - 1  cout - 0
a - 0  b - 1. cin - 0 sum - 1  cout - 0
a - 0  b - 1. cin - 1 sum - 0  cout - 1
a - 1  b - 0. cin - 0 sum - 1  cout - 0
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

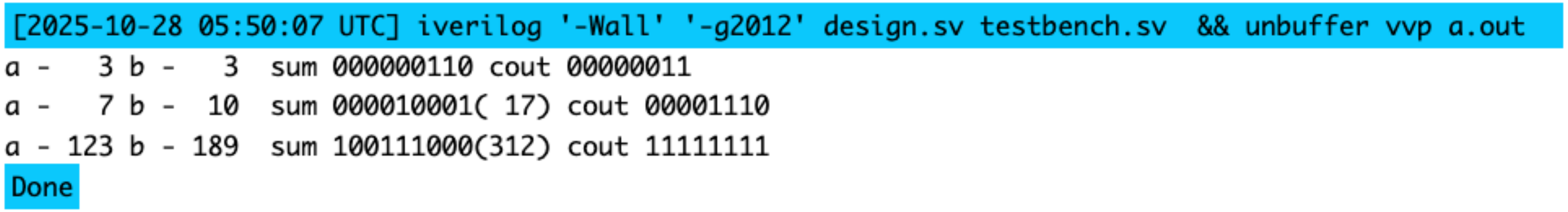


```
 1
 1  1
 1  1
______
    0
```

```
1  1
   1  1
   1  1
________
   1  0
```

```
1  1
   1  1
   1  1
________
1  1  0
```

Ripple Adder Testbench Logs



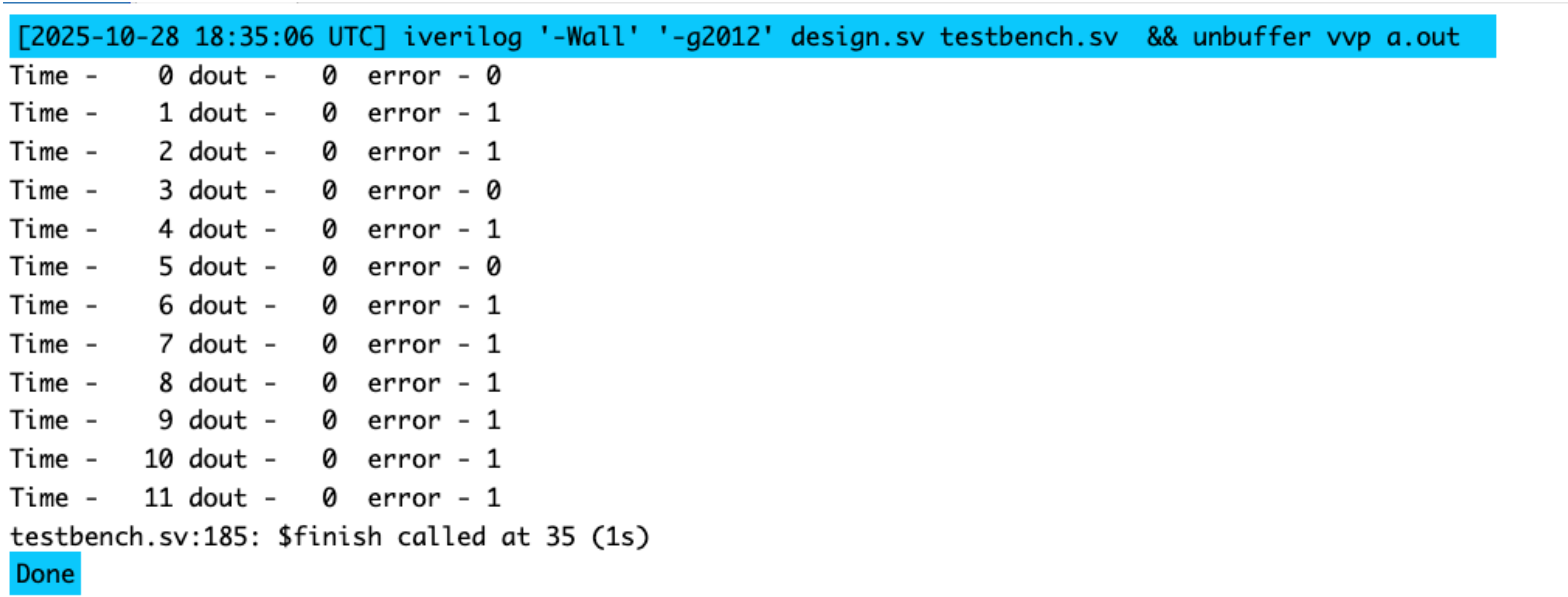
```
[2025-10-28 05:50:07 UTC] iverilog '-Wall' '-g2012' design.sv testbench.sv  && unbuffer vvp a.out
a -   3 b -   3  sum 000000110 cout 00000011
a -   7 b -  10  sum 000010001( 17) cout 00001110
a - 123 b - 189  sum 100111000(312) cout 11111111
Done
```

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

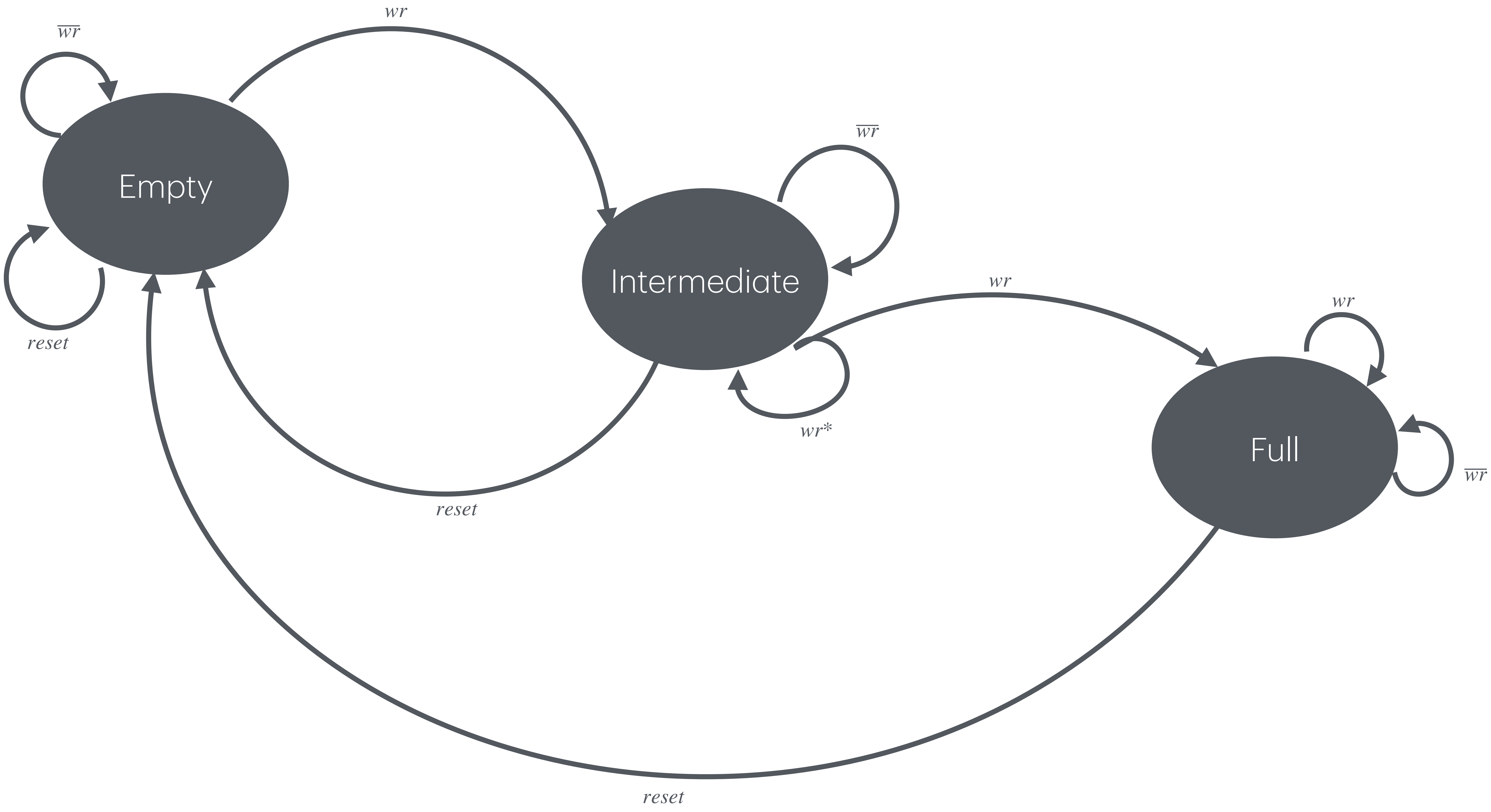
Previous valid read persists on dout port

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
```

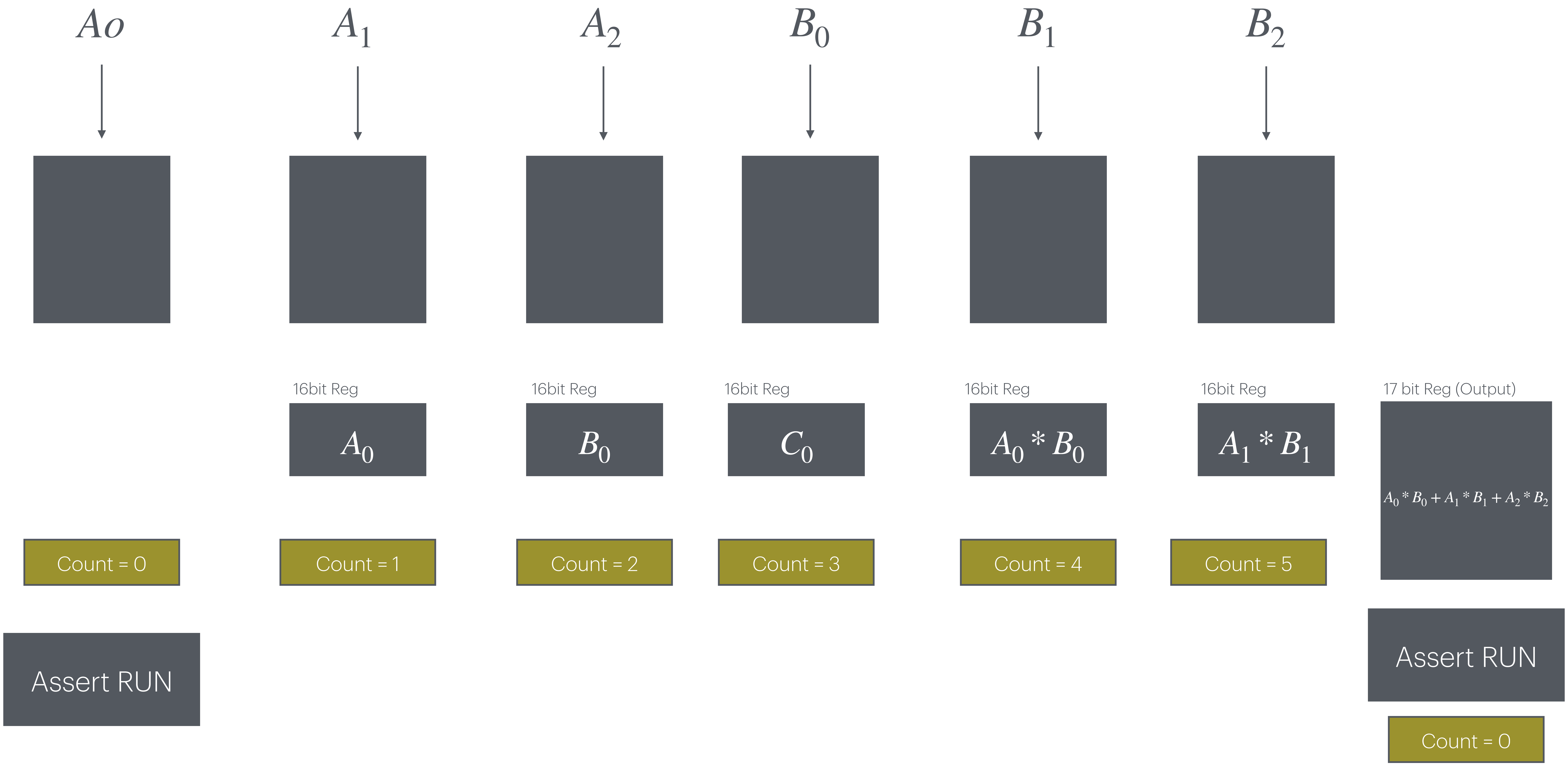
MULTI BIT FIFO



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

Dot Product



Ao A_1 A_2 B_0 B_1 B_2

16bit Reg A_0

16bit Reg B_0

16bit Reg C_0

16bit Reg $A_0 * B_0$

16bit Reg $A_1 * B_1$

17 bit Reg (Output) $A_0 * B_0 + A_1 * B_1 + A_2 * B_2$

Count = 0

Count = 1

Count = 2

Count = 3

Count = 4

Count = 5

Assert RUN

Assert RUN

Count = 0

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	0(255) + 1(1)
0000_0001	0(254) + 1(2)

●
●
●