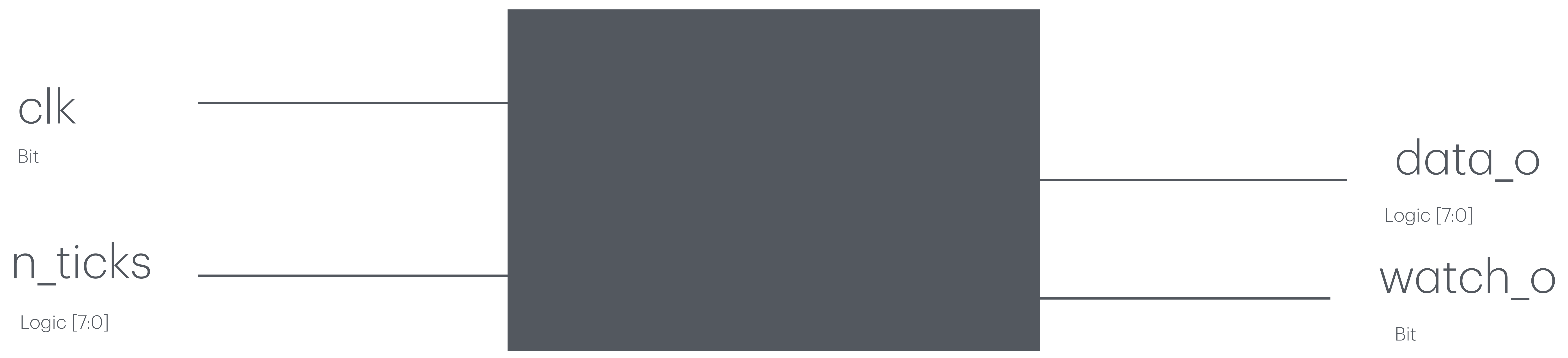
Sys-Verilog Questions Review

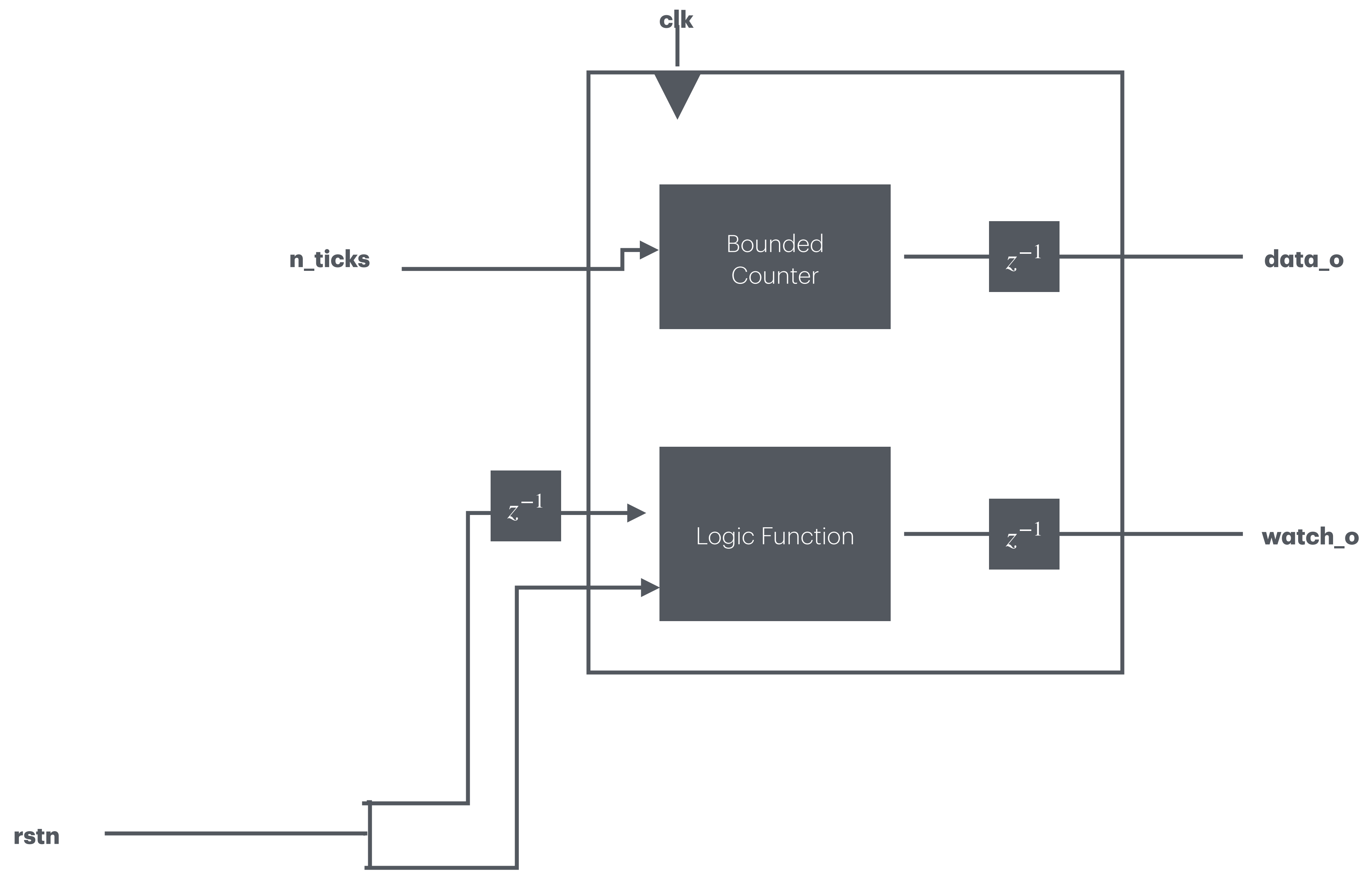
Some Solutions to questions from ChipIO-Dev

Hector “Hectron” Williams

Counter



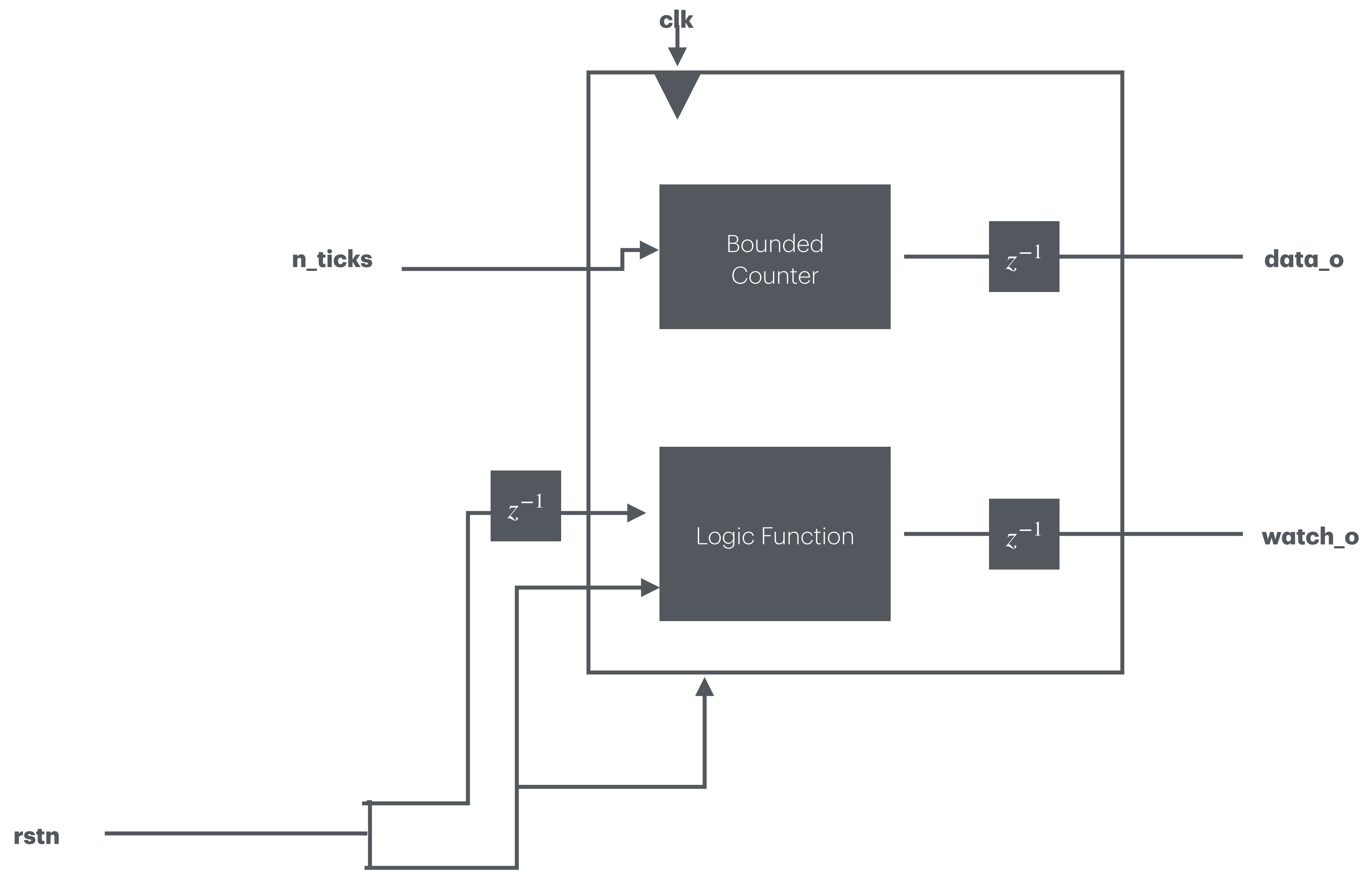
Counter



HDL Module

```
`timescale 1ns/1ps


// interface (top-level)

module counter (

 input logic clk,
 input logic rstn , // active low - reset
 input logic[7:0] n_ticks, // range 0 - 255
 output logic[7:0] data_o, // registed output, Tx/RX pins are not registered ( a submodule )
 output logic watch_o

);


// Constants


// Regs

 logic[7:0] count_reg; // not used

 logic rstn_z;

// dataflow

 always @ (posedge clk or negedge rstn)

  begin

   if (!rstn )

    begin
     data_o <= 0;
     count_reg <= 8'h00;
     watch_o <= 0;
    end

   else

    begin

     if (count_reg < n_ticks) begin

      count_reg <= count_reg + 1;

     end

     data_o <= count_reg;

     watch_o <= rstn ^ rstn_z;

    end

   rstn_z <= rstn;


  end



endmodule
```

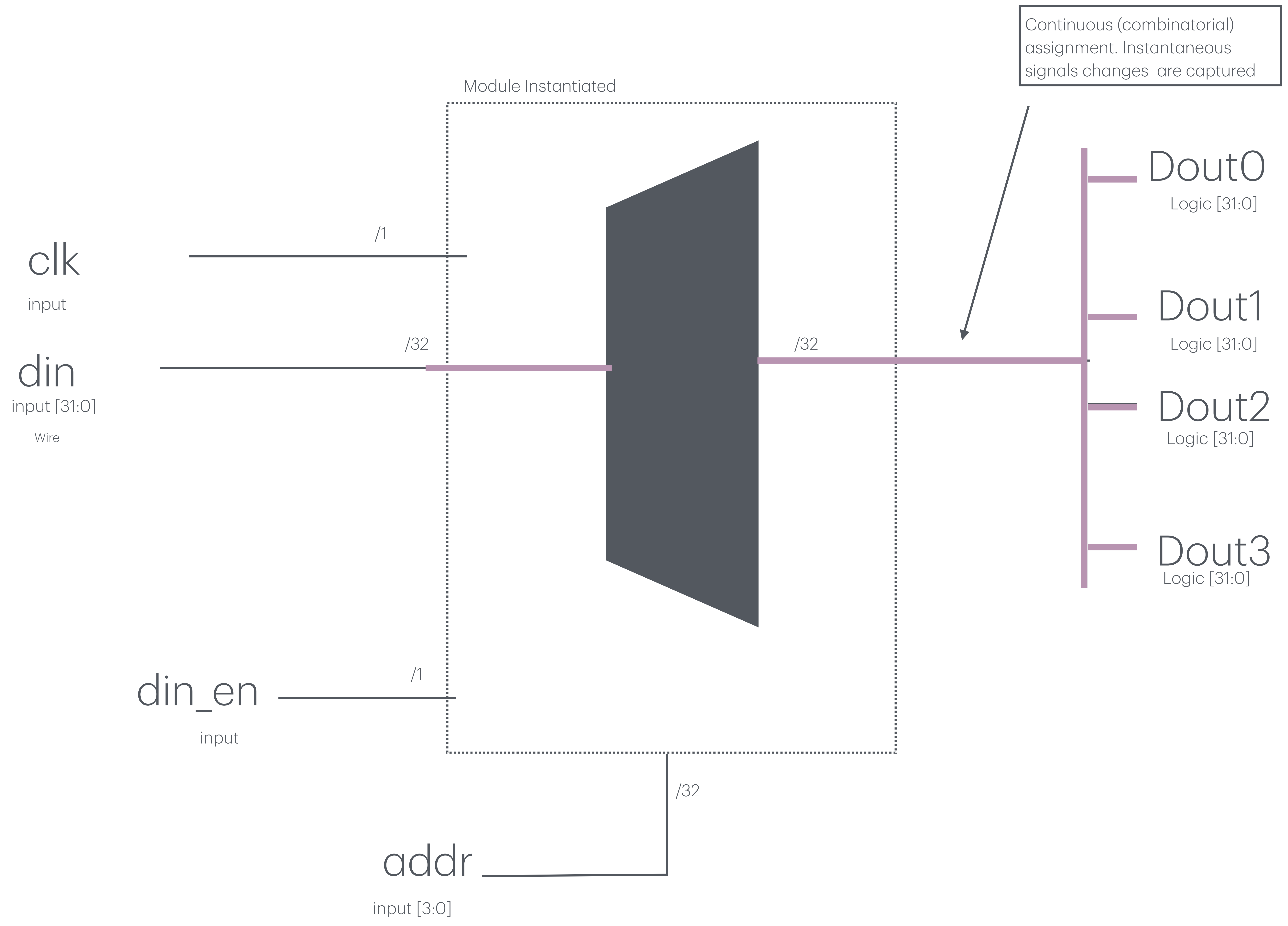
Counter



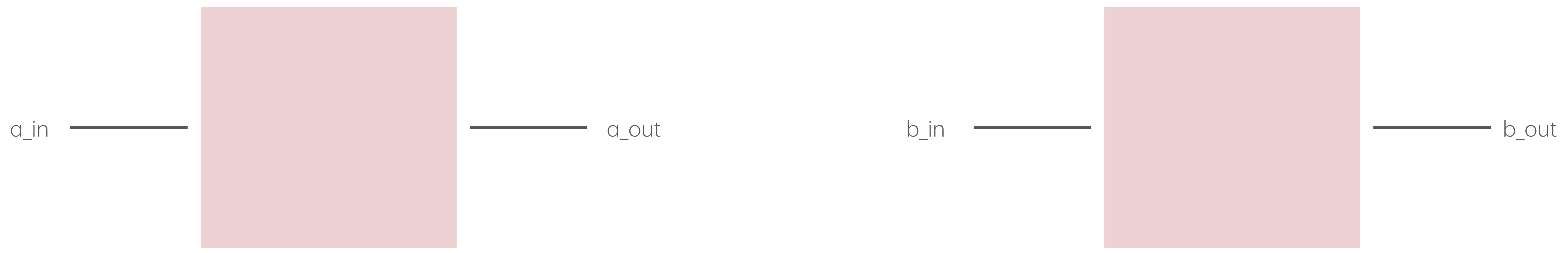
HDL Testbench

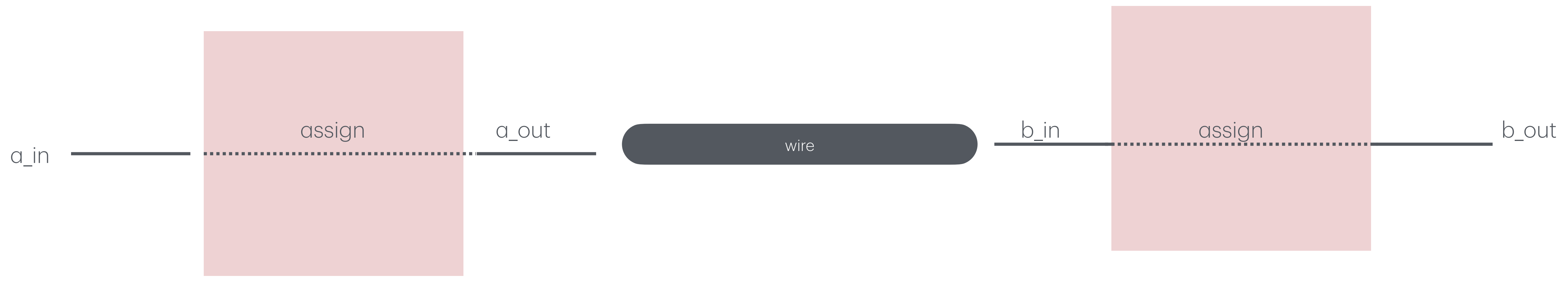
```
`timescale 1ns/1ps

module counter_tb;

 logic clk_tb;
 logic rstn_tb;
 logic [7:0] n_ticks_tb;
 logic [7:0] data_o_tb;
 logic watch_tb;

 parameter TICKS = 8'd10;
 parameter CLK_PERIOD = 2; // 2 ns
 parameter CLK_PERIOD_DIV_2 = CLK_PERIOD/2; // 1 ns


 // Stimuli

 // Reset
 initial begin
  rstn_tb = 0;
  #2 rstn_tb = 1;
 end

 //Clock

 initial begin
             clk_tb = 0;
  forever #(CLK_PERIOD_DIV_2) clk_tb = ~clk_tb;
 end

 // Routine
 initial begin

  n_ticks_tb = 8'd100;

  $display("Waiting for enable_signal to be high at time %0t", $time);

  wait (rstn_tb == 1);

  $display("Counter Enabled");

 end

 // Monitor for every positive edge of signal_a
 always @(posedge clk_tb ) begin

  if (watch_tb ) begin

   $display("posedge detected at time %0t", $time);

  end

 end

 // Instantiate DUT

 counter u0(
  .clk(clk_tb),
  .rstn(rstn_tb),
  .n_ticks(n_ticks_tb),
  .data_o(data_o_tb),
  .watch_o(watch_tb)
 );

endmodule
```

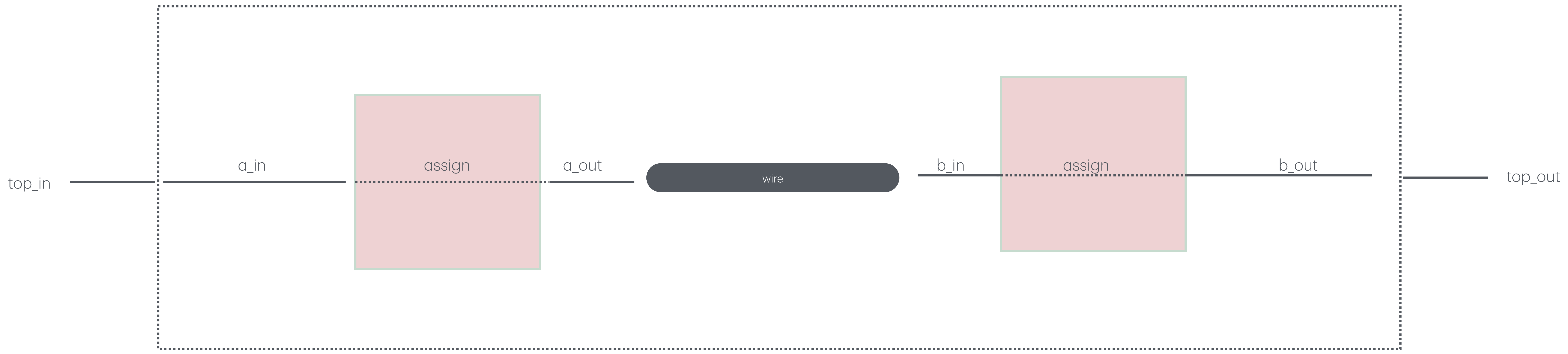
Router


Continuous (combinatorial) assignment. Instantaneous signals changes are captured
Module Instantiated
clk
input
/1
din
input [31:0]
Wire
/32
/32
Dout0
Logic [31:0]
Dout1
Logic [31:0]
Dout2
Logic [31:0]
Dout3
Logic [31:0]
din_en
input
/1
/32
addr
input [3:0]

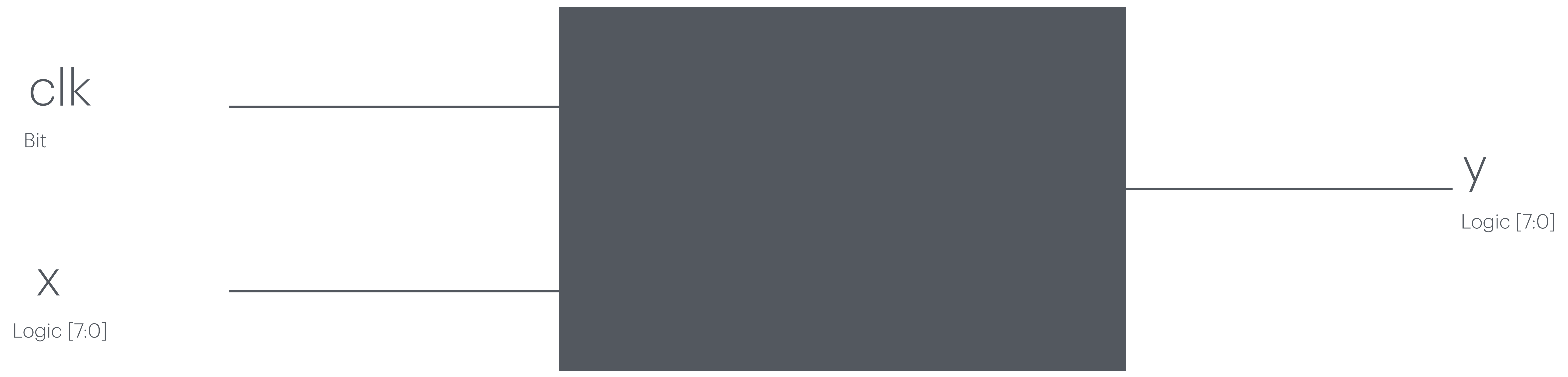
Connect (wire)



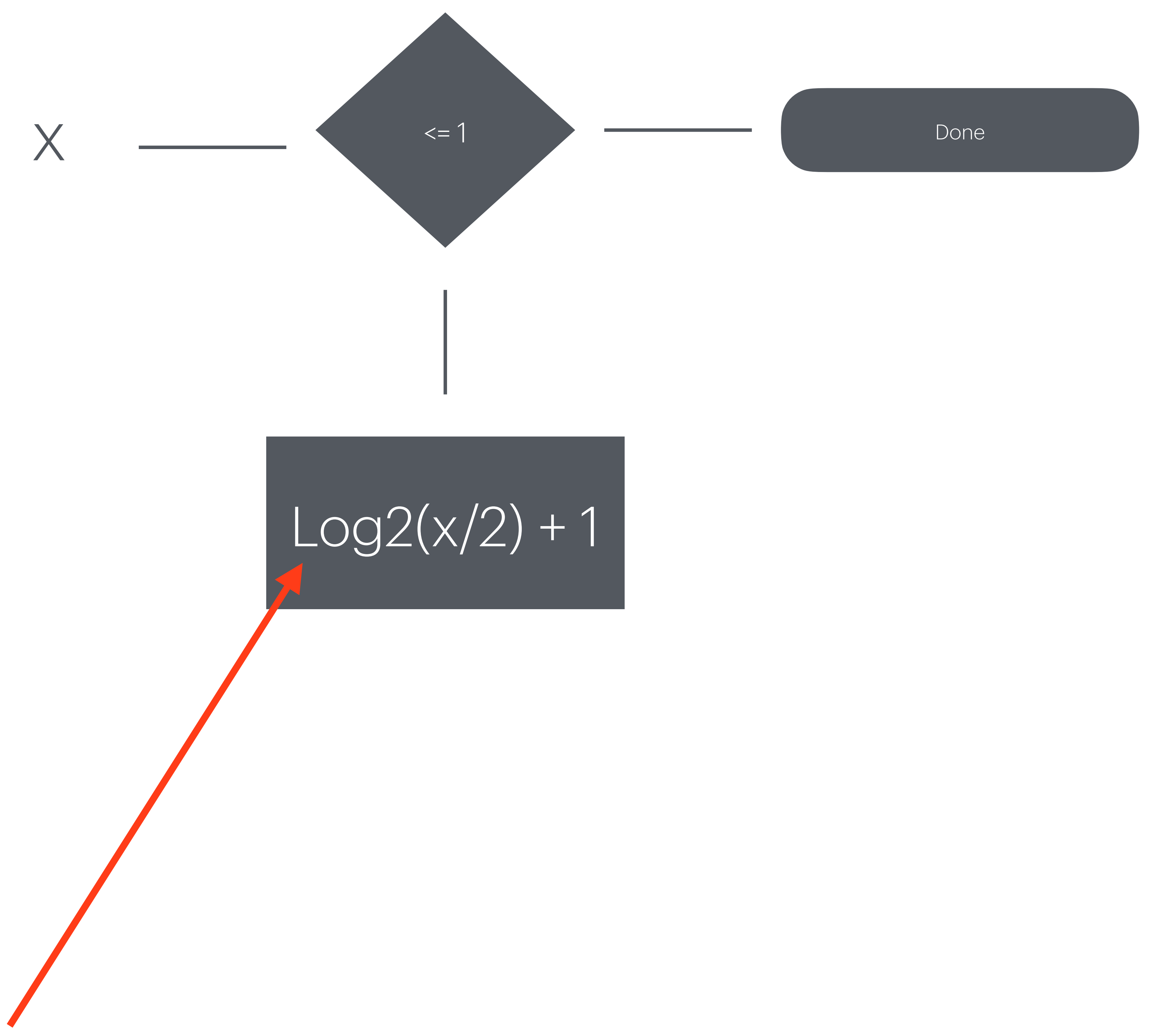




Log2

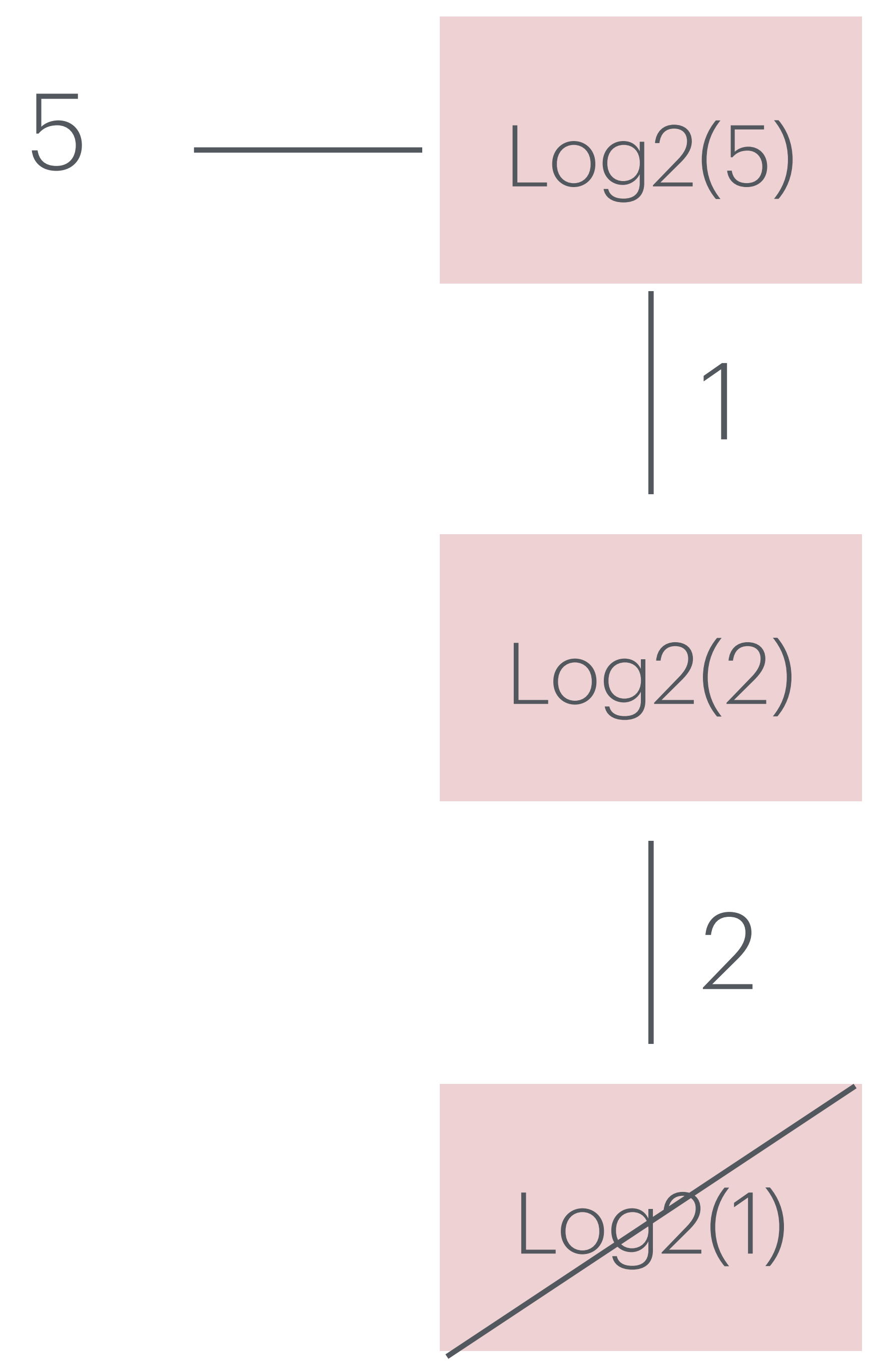


Log2



Recursion or calling the same hardware segment repeatedly

Log2(**5**)



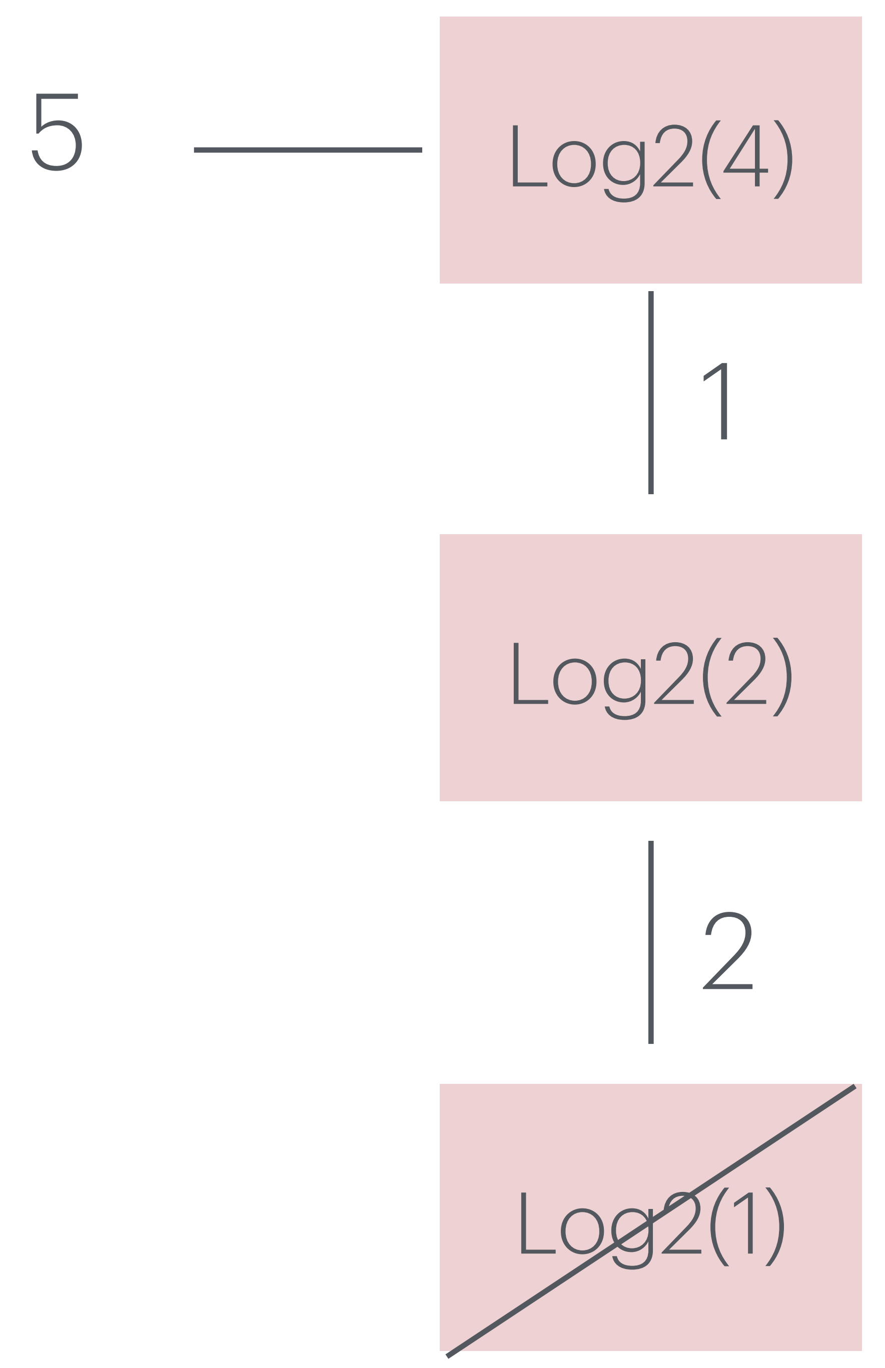
Minimum number of bits represent 5?

Min = 2

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

Log2(**4**)

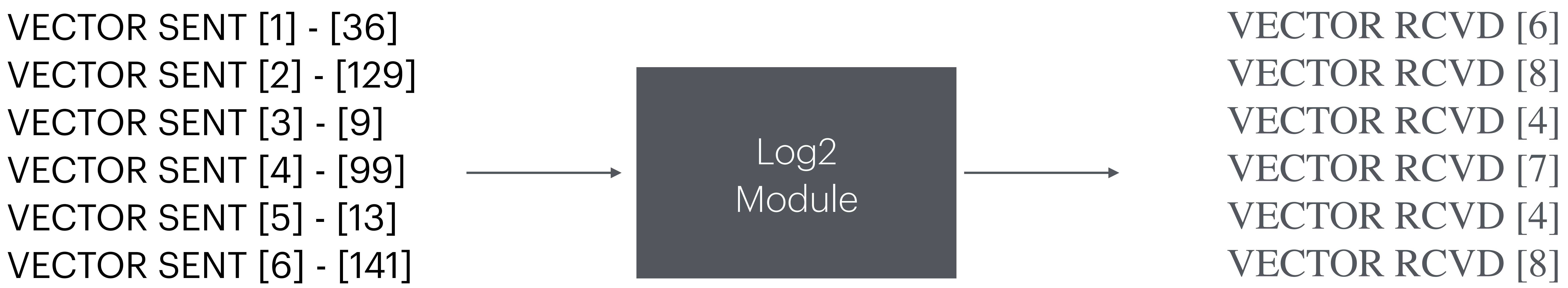
Minimum number of bits represent 5?



Min = 2

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

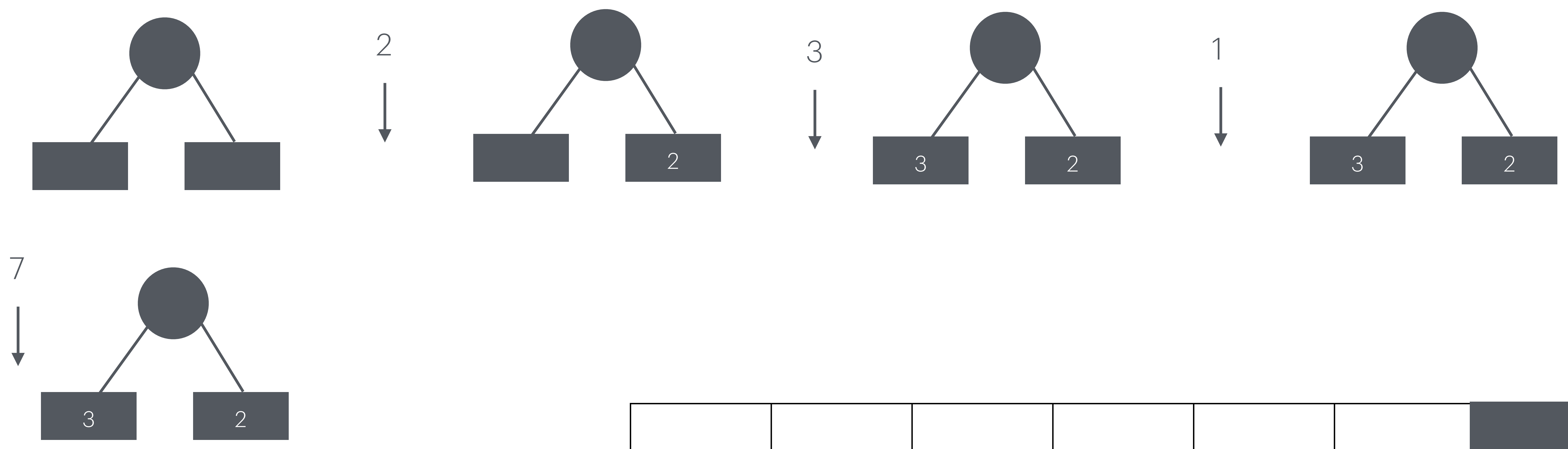
Log2 : Debug Results



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

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

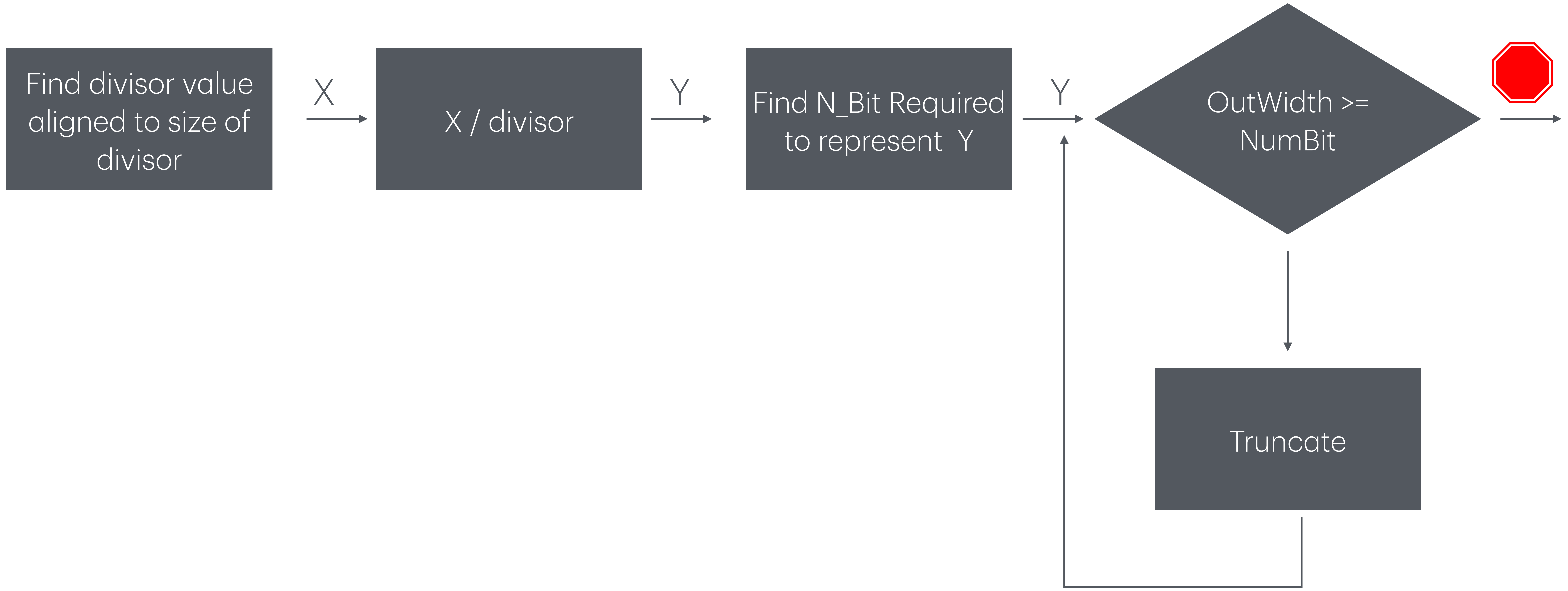
Second Largest



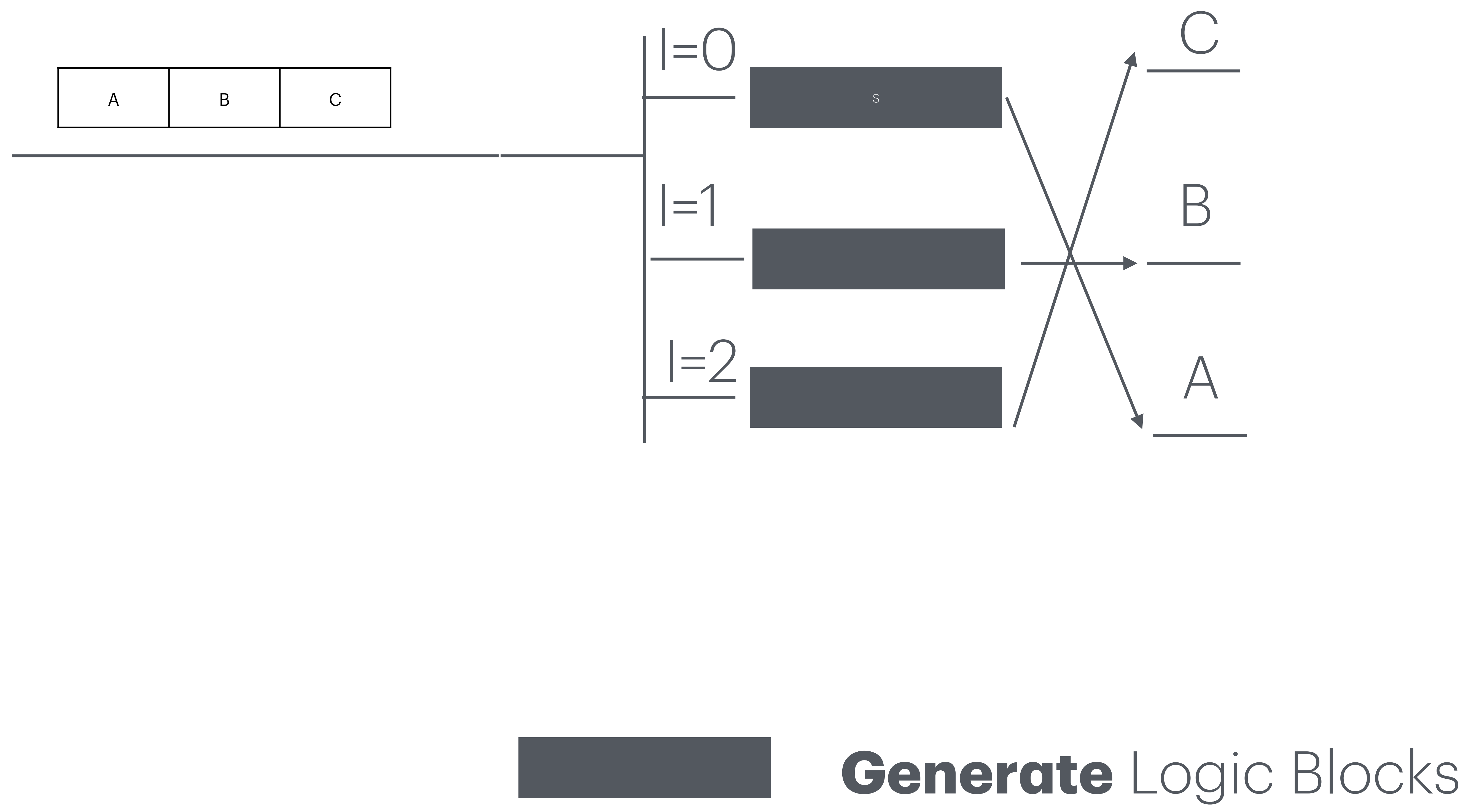
Count	0	1	2	3	4	
Data_In	D0	D1	D2	D3	D4	
2nd Largest	0	0	2	2	2	3

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

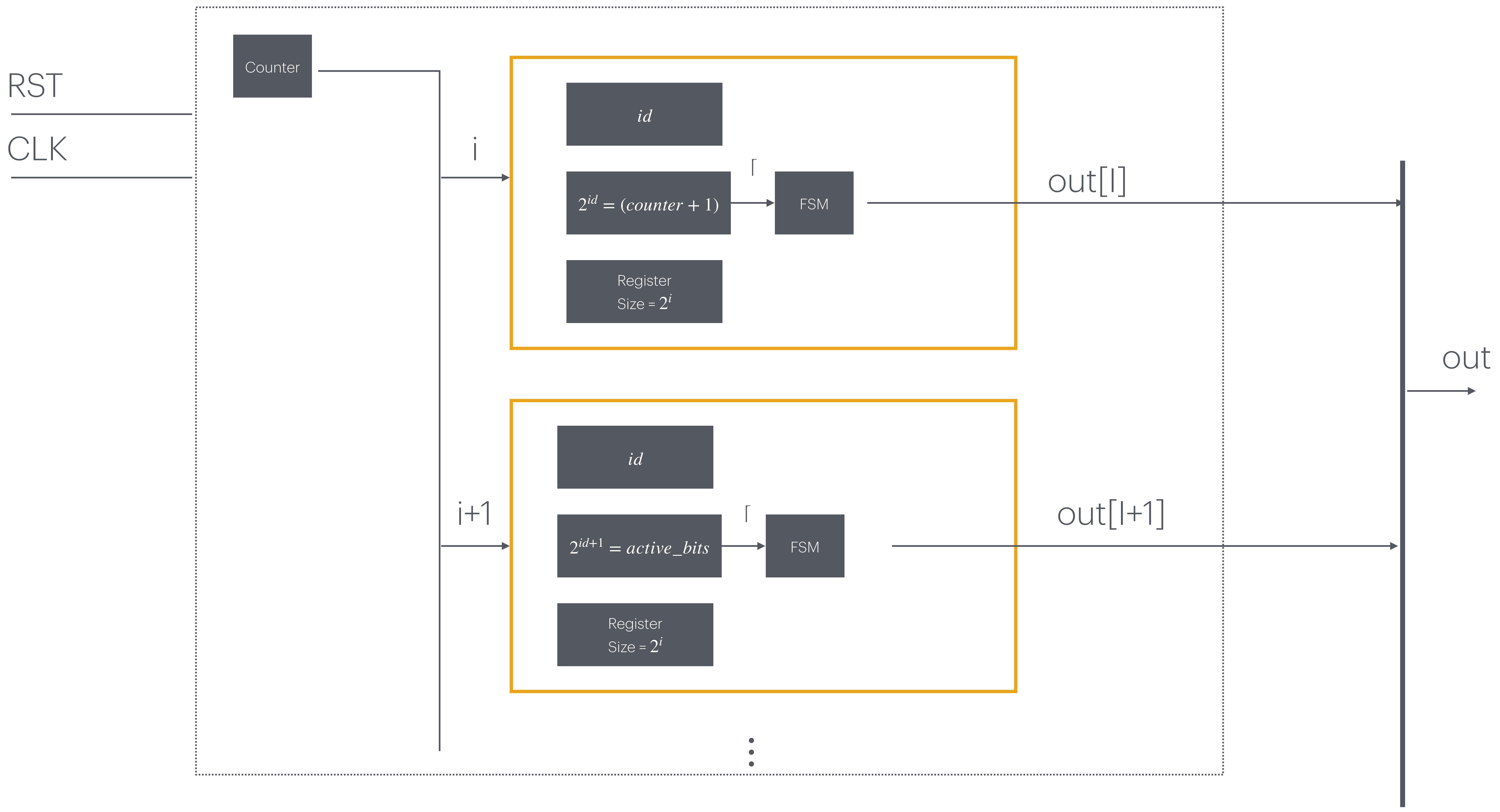
Rounded Division


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



Gray code

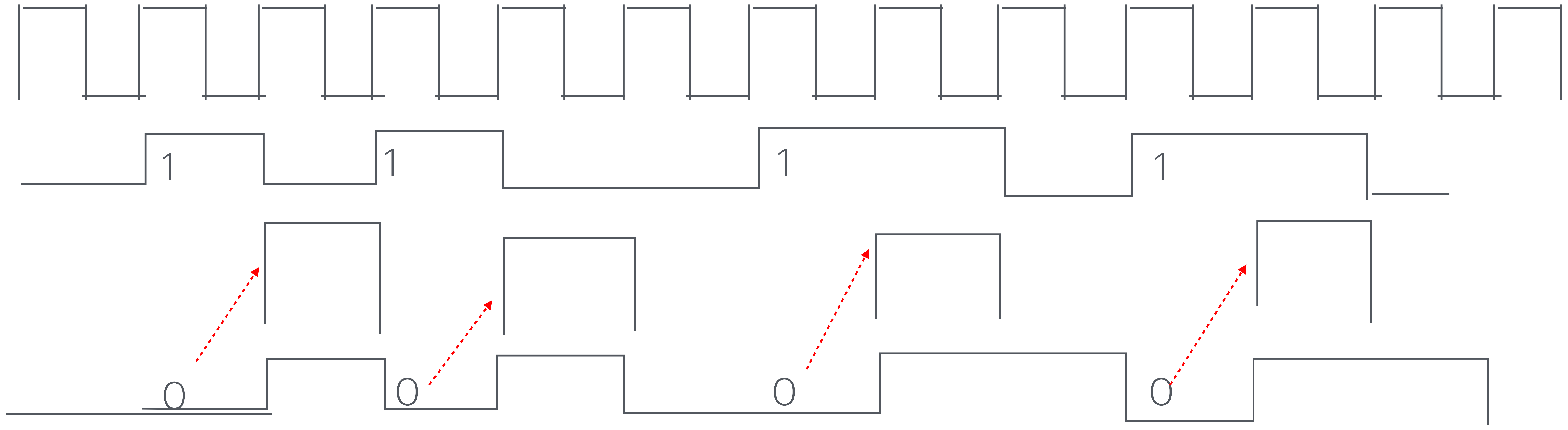

RST
CLK
Counter
i
i+1
id
$2^{id} = (counter + 1)$
FSM
Register
Size = 2^i
id
$2^{id+1} = active\_bits$
FSM
Register
Size = 2^i
out[I]
out[I+1]
out

Gray code

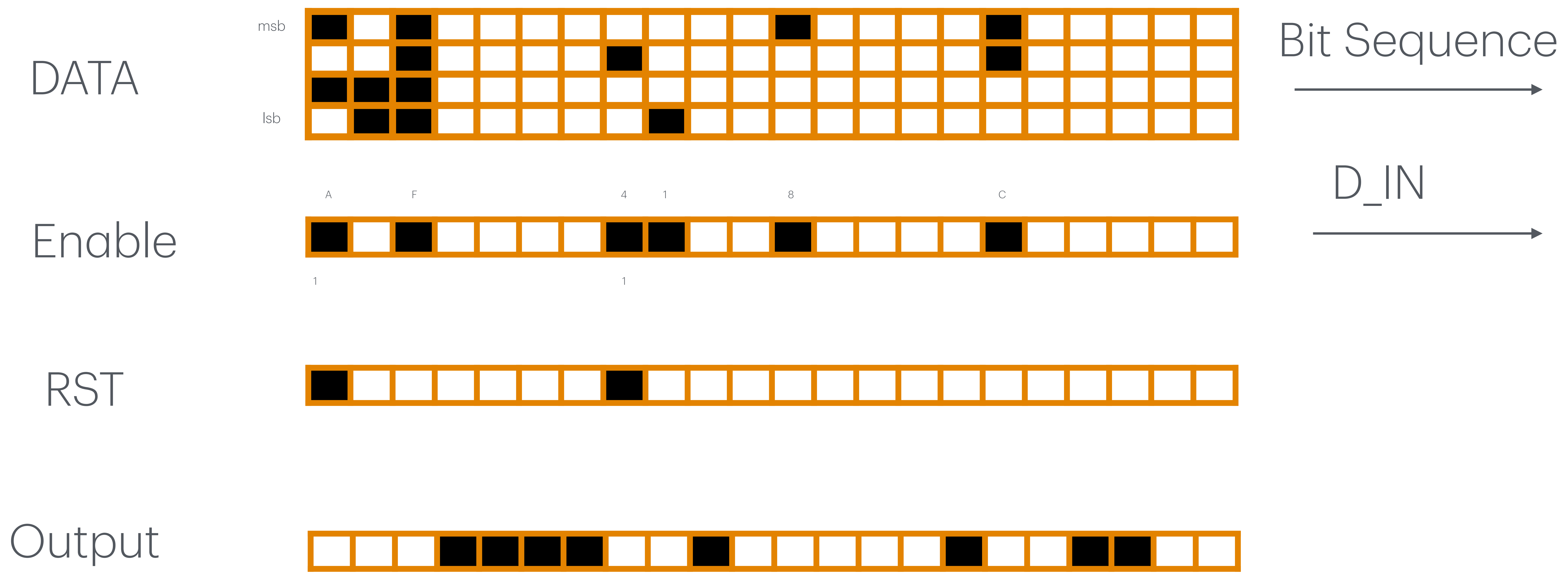
Vertical Delay :)

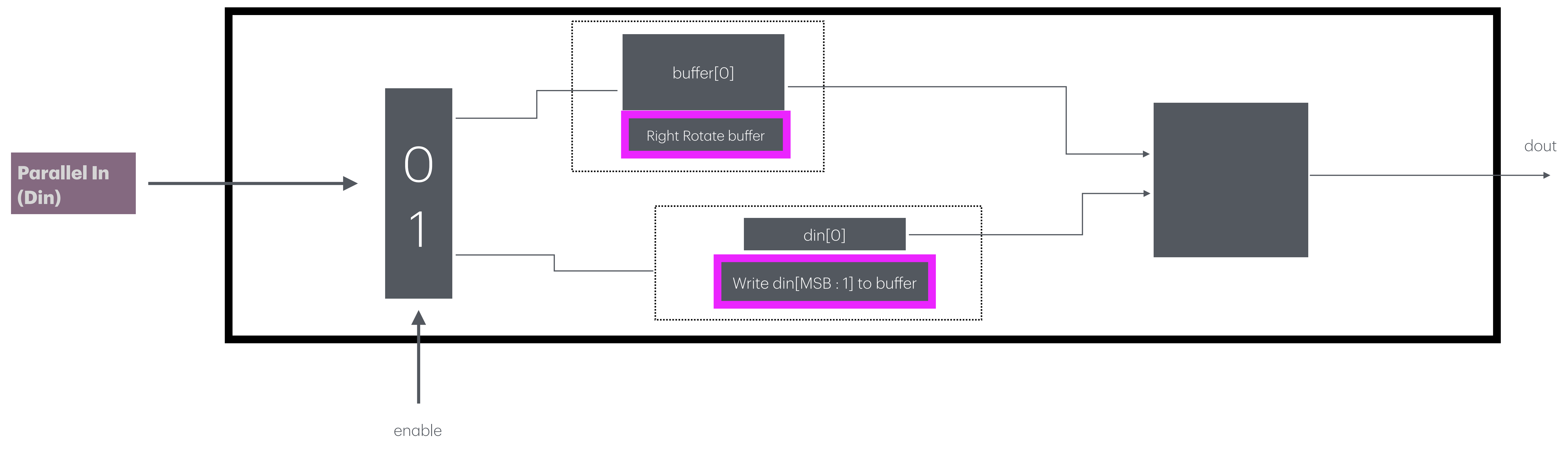
$2^3 = 8cycle$	$2^2 = 4cycle$	$2^1 = 2cycle$	$2^0 = 1cycle$
0	0	0	0
0	0	0	1
0	0	1	1
0	0	1	0
0	1	1	0

Egde Detector



Parralel In -Serial Out





Parralel In -Serial Out

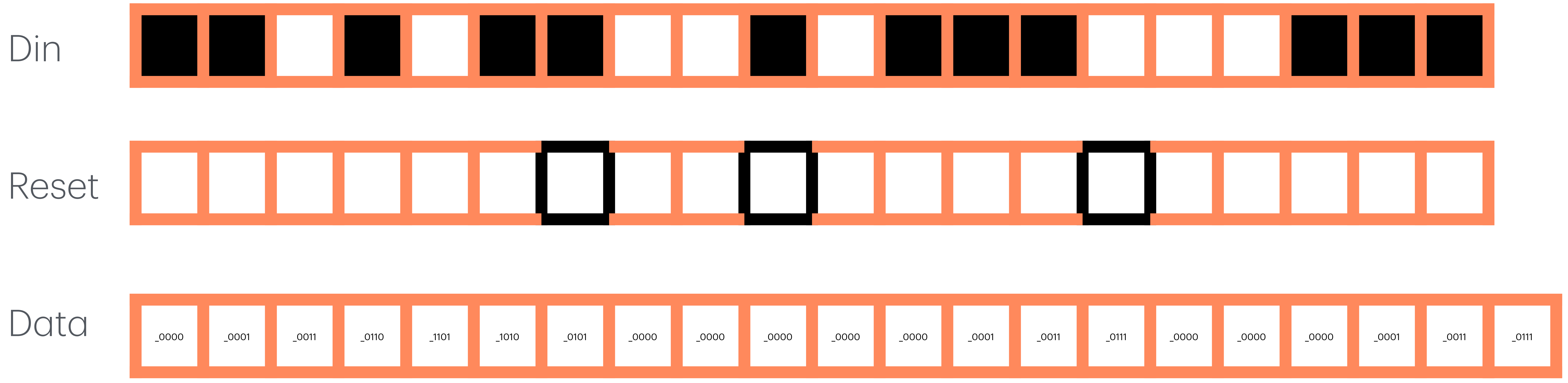
TB

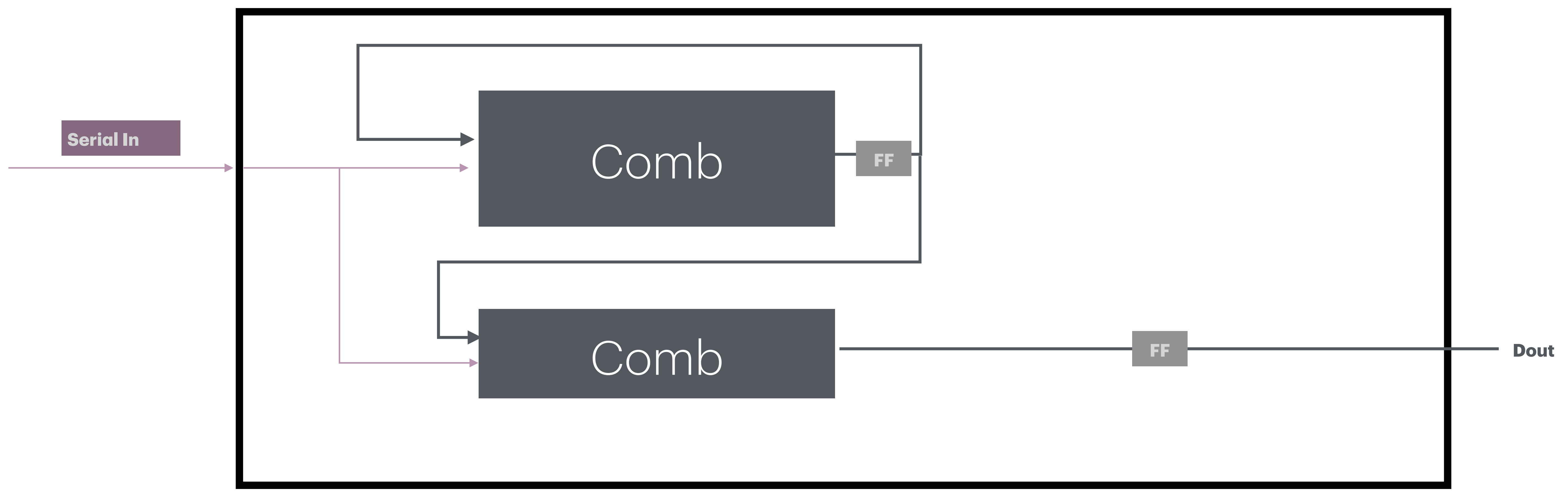
```
`timescale 1ns/1ps

module parallel_to_serial_tb;

 parameter DATA_WIDTH = 4;
 parameter CLK_PERIOD = 2;
 parameter CLK_PERIOD_DIV_2 = CLK_PERIOD / 2;
 parameter N_PARALLEL_SAMPLES = 22;
 parameter RST_VECTOR = {{1'b1},{6{1'b0}}, {1'b1}, {14{1'b0}}};
 parameter ENABLE_VECTOR = {
   {1{1'b1}},
   {1{1'b0}},
   {1{1'b1}},
   {4{1'b0}},
```

Module

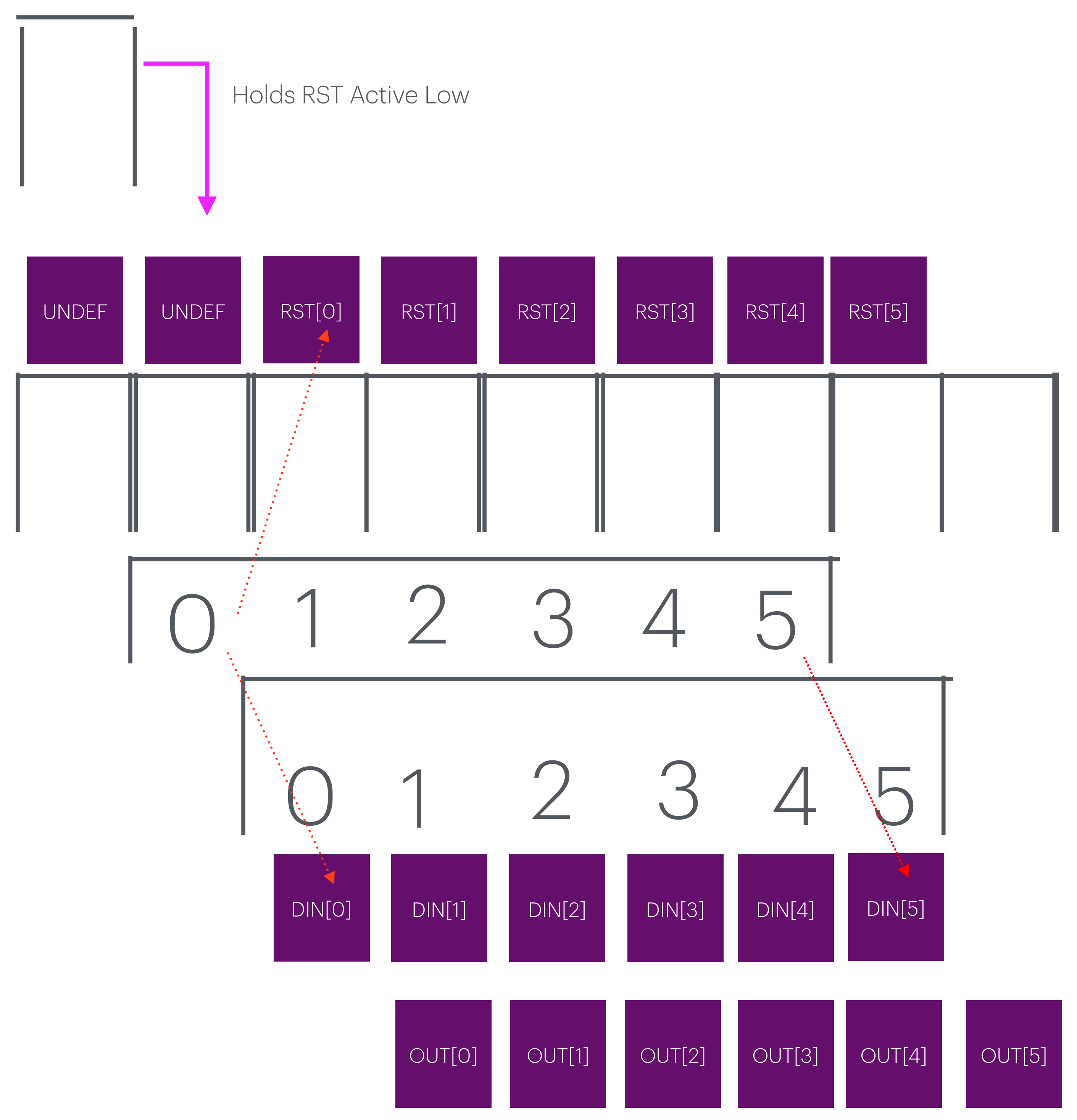
```
`timescale 1ns/1ps

module parallel_to_serial #(parameter DATA_WIDTH = 16) (
  input clk,
  input resetn,
  input [DATA_WIDTH-1:0] din,
  input din_en,
  output logic dout,
  input rdy
);

  logic [DATA_WIDTH - 1: 0] buffer, buffer2;
  logic resetn_z;
  logic rdy_z;
  typedef enum {IDLE, SHIFT} fsm_t;
```

Serial to Parallel

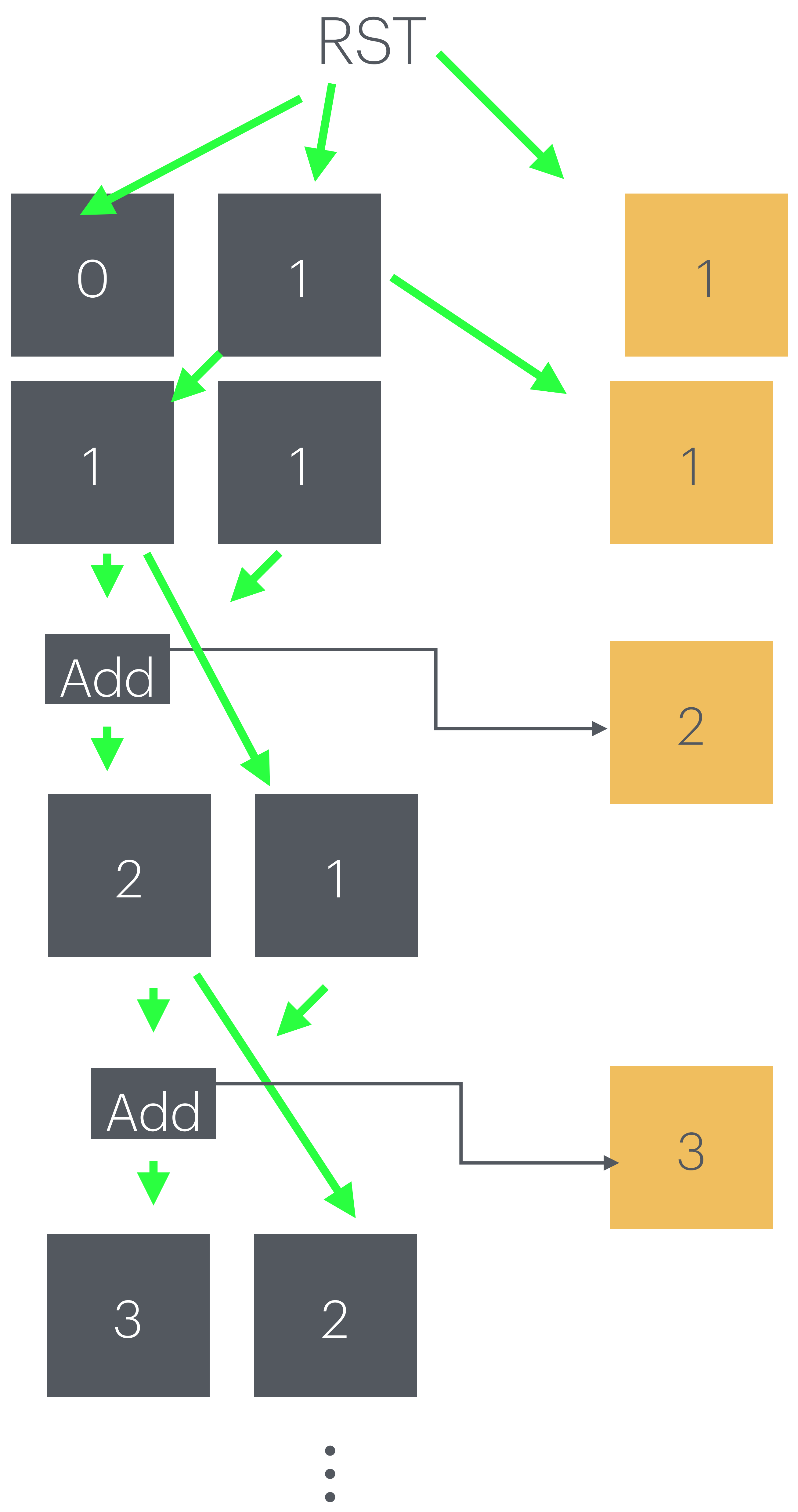




Serial to Parallel (Simulation Concept)

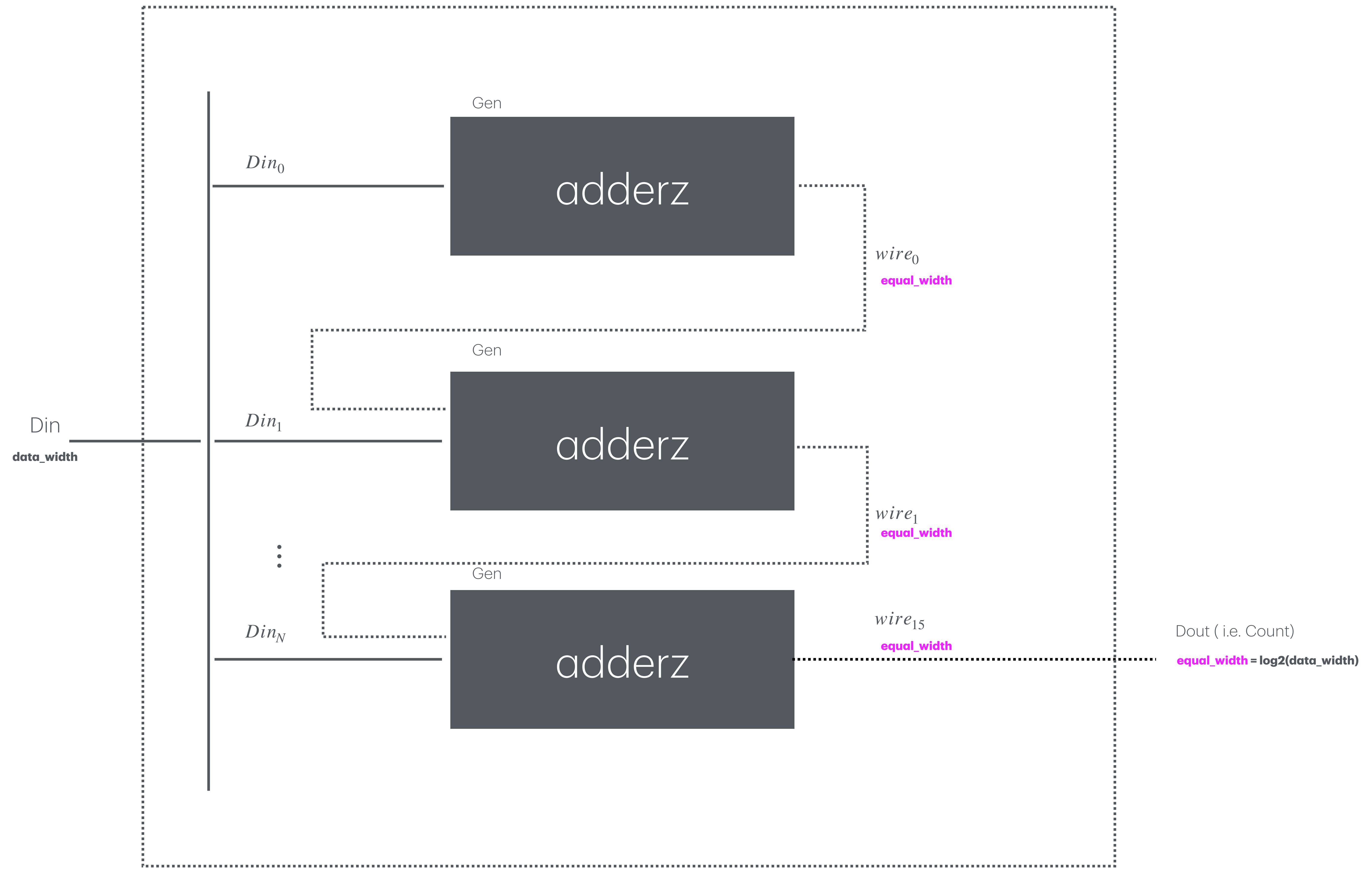


Fibonacci

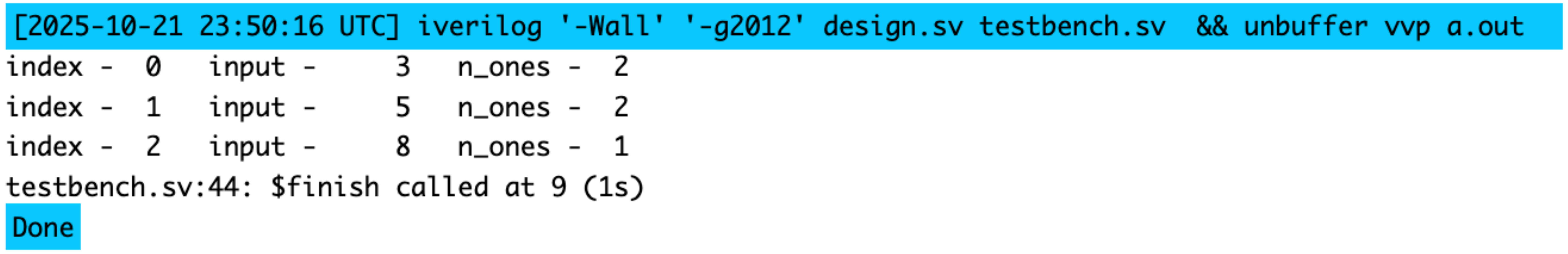


Count Ones

Architecture Similar to Linked List

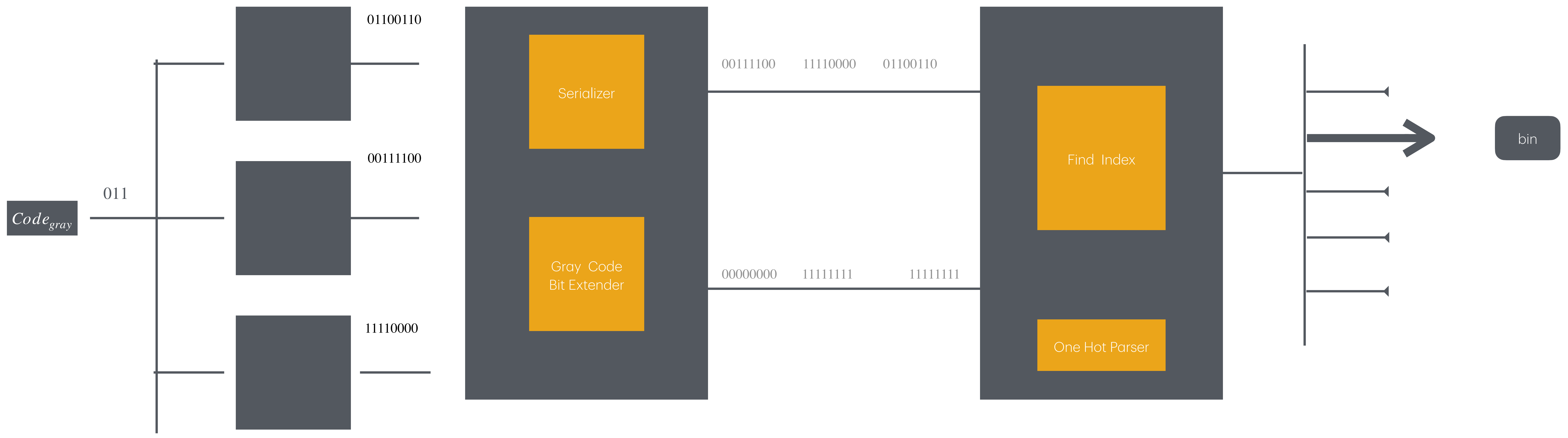


Count Ones



```
[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)


$Code_{gray}$
011
01100110
00111100
11110000
Serializer
Gray Code Bit Extender
00111100
11110000
01100110
00000000
11111111
11111111
Find Index
One Hot Parser
bin