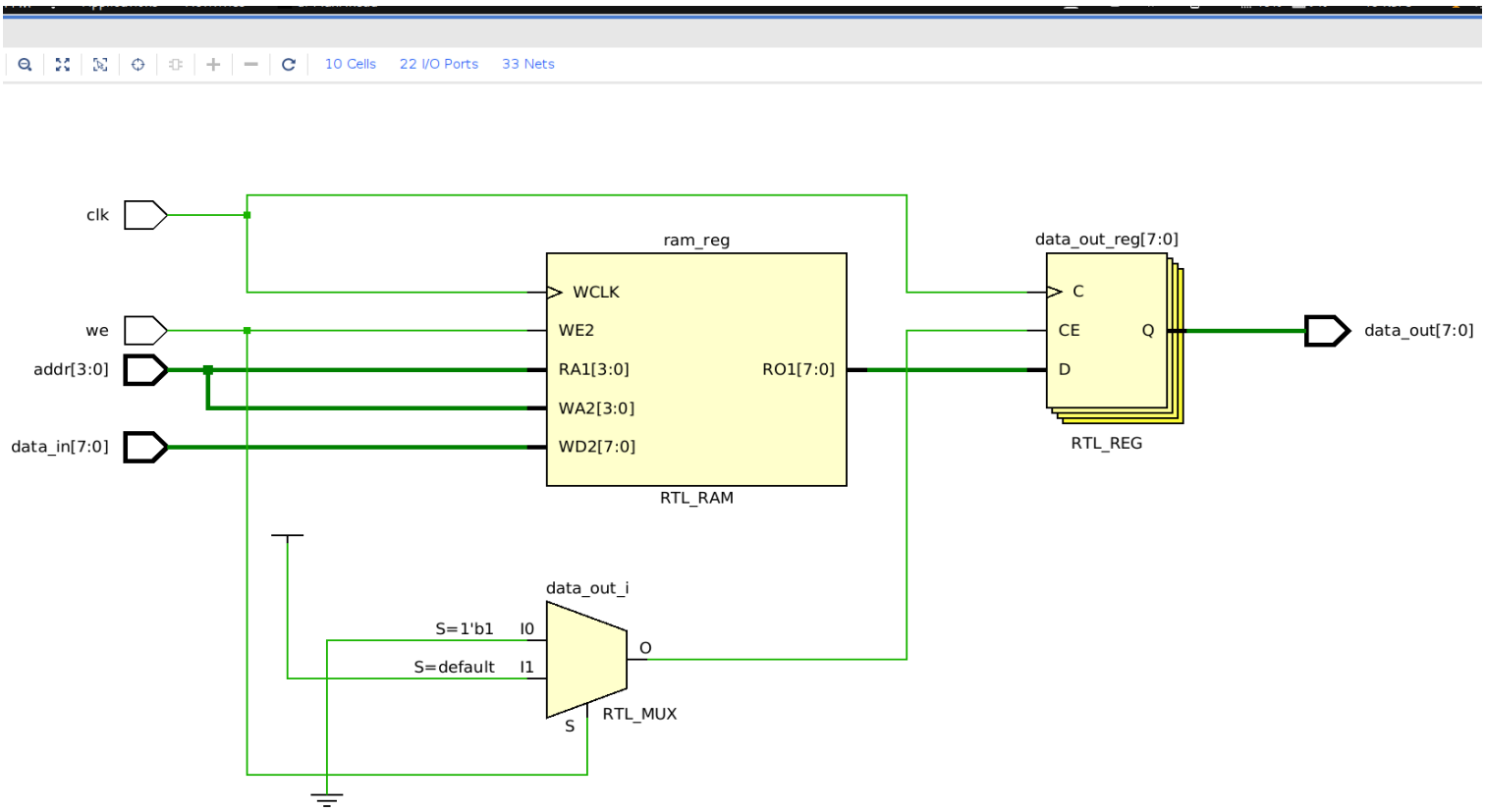
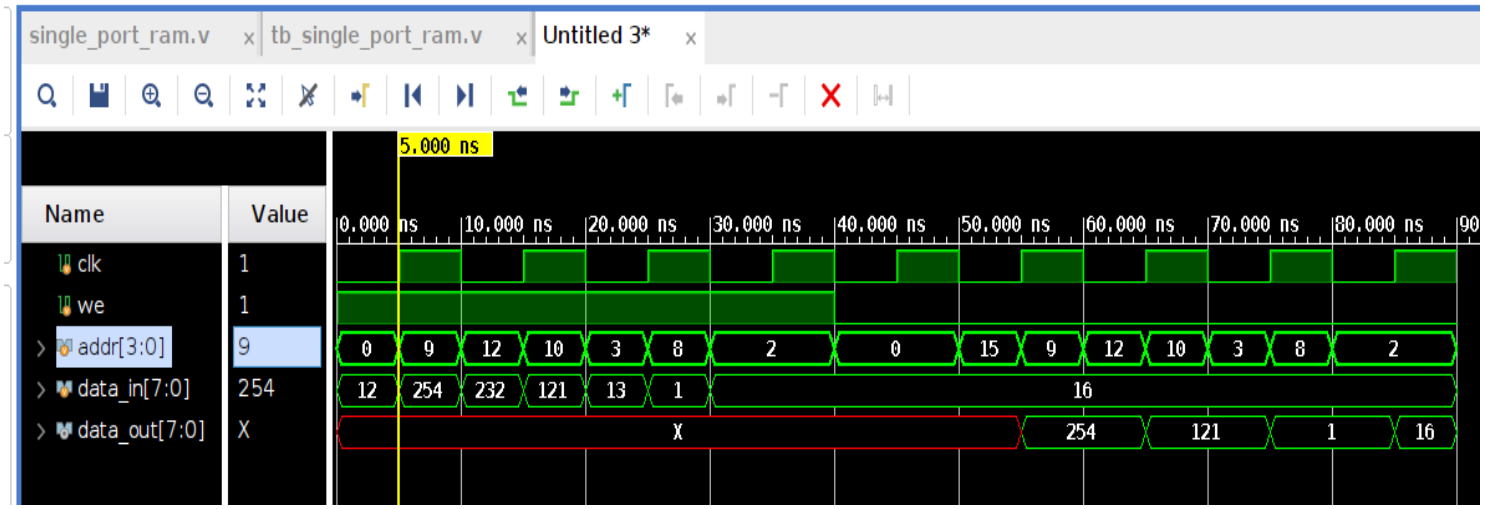

10 Cells
22 I/O Ports
33 Nets
clk
we
addr[3:0]
data_in[7:0]
ram_reg
WCLK
WE2
RA1[3:0]
WA2[3:0]
WD2[7:0]
RO1[7:0]
RTL_RAM
data_out_reg[7:0]
C
CE
D
Q
RTL_REG
data_out[7:0]
data_out_i
S=1'b1
I0
S=default
I1
O
S
RTL_MUX


single_port_ram.v
tb_single_port_ram.v
Untitled 3*
5.000 ns
Name
Value
clk
1
we
1
addr[3:0]
9
data_in[7:0]
254
data_out[7:0]
X
0.000 ns
10.000 ns
20.000 ns
30.000 ns
40.000 ns
50.000 ns
60.000 ns
70.000 ns
80.000 ns
0 9 12 10 3 8 2 0 15 9 12 10 3 8 2
12 254 232 121 13 1 16
X 254 121 1 16

single_port_ram.v

/home/itzzinfinity/Cozy Drive/100daysofRTL/day_086/project_1/project_1.srcs/sources_1/new/single_port_ram.v

```
1  `timescale 1ns / 1ps
2  //////////////////////////////////////////////////////////////////////////////////
3  // Engineer: Anjan Prasad
4  // Create Date: 12/16/2024 10:19:35 AM
5  // Module Name: single_port_ram
6  //////////////////////////////////////////////////////////////////////////////////
7
8  module single_port_ram #(
9      parameter DATA_WIDTH = 8,
10     parameter ADDR_WIDTH = 4
11 )(
12     input clk,
13     input we,                              // Write enable
14     input [ADDR_WIDTH-1:0] addr,
15     input [DATA_WIDTH-1:0] data_in,
16     output reg [DATA_WIDTH-1:0] data_out
17 );
18
19     reg [DATA_WIDTH-1:0] ram [2**ADDR_WIDTH-1:0];
20
21     always @(posedge clk) begin
22         if (we) begin
23             ram[addr] <= data_in;
24         end
25         else begin
26             data_out <= ram[addr];
27         end
28     end
29
30 endmodule
31
```

tb_single_port_ram.v

/home/itzzinfinity/Cozy Drive/100daysofRTL/day_086/project_1/project_1.srcs/sim_1/new/tb_single_port_ram.v

```
1   `timescale 1ns / 1ps
2   //////////////////////////////////////////////////////////////////////////////////
3   // Engineer: Anjan Prasad
4   // Create Date: 12/16/2024 10:22:08 AM
5   // Module Name: tb_single_port_ram
6   //////////////////////////////////////////////////////////////////////////////////
7
8
9   module tb_single_port_ram;
10      parameter DATA_WIDTH = 8;
11      parameter ADDR_WIDTH = 4;
12
13      reg clk,we;
14      reg [ADDR_WIDTH-1:0] addr;
15      reg [DATA_WIDTH-1:0] data_in;
16      wire [DATA_WIDTH-1:0] data_out;
17
18      single_port_ram #(DATA_WIDTH, ADDR_WIDTH)
19      DUT (.clk(clk),
20          .we(we),
21          .addr(addr),
22          .data_in(data_in),
23          .data_out(data_out));
24
25      initial begin
26        clk = 0;
27        forever #5 clk = ~clk;
28      end
29
30      initial begin
31       // Write operations
32            we = 1; addr = 16'd32; data_in = 8'd12;
33        #5 we = 1; addr = 16'd25; data_in = 8'd254;
34        #5 we = 1; addr = 16'd12; data_in = 8'd232;
35        #5 we = 1; addr = 16'd10; data_in = 8'd121;
36        #5 we = 1; addr = 16'd19; data_in = 8'd13;
37        #5 we = 1; addr = 16'd24; data_in = 8'd1;
38        #5 we = 1; addr = 16'd34; data_in = 8'd16;
39       // read operations
40        #10 we = 0; addr = 16'd32;
41        #10 we = 0; addr = 16'd31;
42        #5  we = 0; addr = 16'd25;
43        #5  we = 0; addr = 16'd12;
44        #5  we = 0; addr = 16'd10;
45        #5  we = 0; addr = 16'd19;
46        #5  we = 0; addr = 16'd24;
47        #5  we = 0; addr = 16'd34;
48        #10 $finish;
49      end
50  endmodule
```