

3. Question 3

3.1. Design code “single port ram”

```
1 module single_port_Ram #(
2     parameter MEM_WIDTH = 16,          /* Data in/out and memory word width */
3     parameter MEM_DEPTH = 1024,        /* Memory depth */
4     parameter ADD_SIZE = 10,           /* Address size based upon the memory depth */
5     parameter ADDR_PIPELINE = "FALSE", /* If "TRUE", the address should be pipelined before writing/reading the RAM, if
6                                           "FALSE" then the address input will be assigned directly to the RAM's address port */
7     parameter DOUT_PIPELINE = "TRUE",  /* If "TRUE" then the data out should be pipelined, if "FALSE" then the output will be
8                                           out of the RAM directly */
9     parameter PARITY_ENABLE = 1        /* If the parameter value is 1 then the parity should be calculated and assigned to
10                                           parity_out port, if the parameter is 0 then the parity_out port should be tied to 0 */
11 ) (
12     /*-----Inputs-----*/
13     input [MEM_WIDTH-1:0] din,
14     input [ADD_SIZE-1:0] addr,
15     input  addr_en,      /* enable signal for the flipflop that pipelines the address */
16     input  dout_en,      /* enable signal for the flipflop that pipelines the data out */
17     input  wr_en,        /* enable signal for writing in ram */
18     input  rd_en,        /* enable signal for reading from ram */
19     input  blk_select,   /* enable signal to the ram */
20     input  clk,          /* clock signal input */
21     input  rst,          /* active high synchronous reset */
22
23     /*-----Outputs-----*/
24     output [MEM_WIDTH-1:0] dout, /* data out of Ram */
25     output parity_out /* calculates the even parity on the dout bus */
26 );
27
28 reg [MEM_WIDTH-1:0] mem [MEM_DEPTH-1:0];
29
30 wire [ADD_SIZE-1:0] addr_final; /* the address I will use to access the location of reading or writing */
31 /* Address Pipeline stage */
32 generate
33     if (ADDR_PIPELINE == "TRUE")
34         peline_stage_mod #(.WIDTH(ADD_SIZE), /* parameter defines the width of input and output
35                                     .RSTTYPE("SYNC") /* parameter defines the type of rst, takes SYNC or ASYNC
36                                     ) ADDR_PIPELINE_reg (.in(addr), /* [Width-1:0] input
37                                                         .clk(clk), /* clock input
38                                                         .FF_en(addr_en), /* clock enable
39                                                         .rst(rst), /* reset signal
40                                                         .sel(1), /* selection of the mux
41                                                         .out(addr_final) /* output of the mux
42                                                         );
43     else if (ADDR_PIPELINE == "FALSE")
44         peline_stage_mod #(.WIDTH(ADD_SIZE), /* parameter defines the width of input and output
45                                     .RSTTYPE("SYNC") /* parameter defines the type of rst, takes SYNC or ASYNC
46                                     ) ADDR_PIPELINE_reg (.in(addr), /* [Width-1:0] input
47                                                         .clk(clk), /* clock input
48                                                         .FF_en(addr_en), /* clock enable
49                                                         .rst(rst), /* reset signal
50                                                         .sel(0), /* selection of the mux
51                                                         .out(addr_final) /* output of the mux
52                                                         );
53 endgenerate
54
55 reg [MEM_WIDTH-1:0] dout_from_mem; /* the direct data output of Ram */
56 always @(posedge clk) begin
57     if (rst)
58         dout_from_mem <= 0;
59     else begin
60         if (blk_select) begin
61             if (wr_en)
62                 mem[addr_final] <= din;
63             if (rd_en)
64                 dout_from_mem <= mem[addr_final];
65         end
66     end
67 end
68
69 /* Dout Pipeline stage */
70 generate
71     if (DOUT_PIPELINE == "TRUE")
72         peline_stage_mod #(.WIDTH(MEM_WIDTH), /* parameter defines the width of input and output
73                                     .RSTTYPE("SYNC") /* parameter defines the type of rst, takes SYNC or ASYNC
74                                     ) DOUT_PIPELINE_reg (.in(dout_from_mem), /* [Width-1:0] input
75                                                         .clk(clk), /* clock input
76                                                         .FF_en(dout_en), /* clock enable
77                                                         .rst(rst), /* reset signal
78                                                         .sel(1), /* selection of the mux
79                                                         .out(dout) /* output of the mux
80                                                         );
81     else
82         dout <= dout_from_mem;
83 endgenerate
84
85 parity_out <= 0;
```

```

82     else if(DOUT_PIPELINE=="FALSE")
83         peline_stage_mod #(.WIDTH(MEM_WIDTH), // parameter defines the width of input and output
84             .RSTTYPE("SYNC") // parameter defines the type of rst, takes SYNC or ASYNC
85         )Dout_PIPELINE_reg (.in(dout_from_mem), // [Width-1:0]input
86             .clk(clk), // clock input
87             .FF_en(dout_en), // clock enable
88             .rst(rst), // reset signal
89             .sel(0), // selection of the mux
90             .out(dout) // output of the mux
91         );
92     endgenerate
93
94     /* Calculate parity check if the PARITY_ENABLE = 1 */
95     generate
96         if(PARITY_ENABLE==1)
97             assign parity_out = ^dout;
98         else if(PARITY_ENABLE==0)
99             assign parity_out = 0;
100     endgenerate
101
102 endmodule //Ram

```

3.2. Pipeline_stage_module

```

1 module peline_stage_mod (in,clk,FF_en,rst,sel,out);
2     parameter WIDTH = 18 ; // Width of the input and the output */
3     /* type of the reset is sync or async. It takes SYNC or ASYNC */
4     parameter RSTTYPE = "SYNC"; // default reset type is synchronous */
5
6     input [WIDTH-1:0]in;
7     input clk,FF_en,sel,rst;
8     output [WIDTH-1:0]out;
9
10    reg [WIDTH-1:0]in_r;
11    /* generate block according to RSTTYPE parameter */
12    generate
13        if(RSTTYPE=="SYNC")begin
14            always @(posedge clk) begin
15                if(rst)
16                    in_r <= 0;
17                else begin
18                    if(FF_en)
19                        in_r <= in;
20                end
21            end
22        end
23        else if(RSTTYPE=="ASYNC")begin
24            always @(posedge clk or posedge rst) begin
25                if(rst)
26                    in_r <= 0;
27                else begin
28                    if(FF_en)
29                        in_r <= in;
30                end
31            end
32        end
33    endgenerate
34    /* mux design using assignment statement */
35    assign out =(sel)? in_r : in;
36 endmodule
37
38 /*
39 //-----module instantiation-----
40 peline_stage_mod #(.WIDTH(), // parameter defines the width of input and output
41     .RSTTYPE() // parameter defines the type of rst, takes SYNC or ASYNC
42 )module_name (.in(), // [Width-1:0]input
43     .clk(), // clock input
44     .FF_en(), // clock enable
45     .rst(), // reset signal
46     .sel(), // selection of the mux
47     .out() // output of the mux
48 );
49 */

```

3.3. Testbench

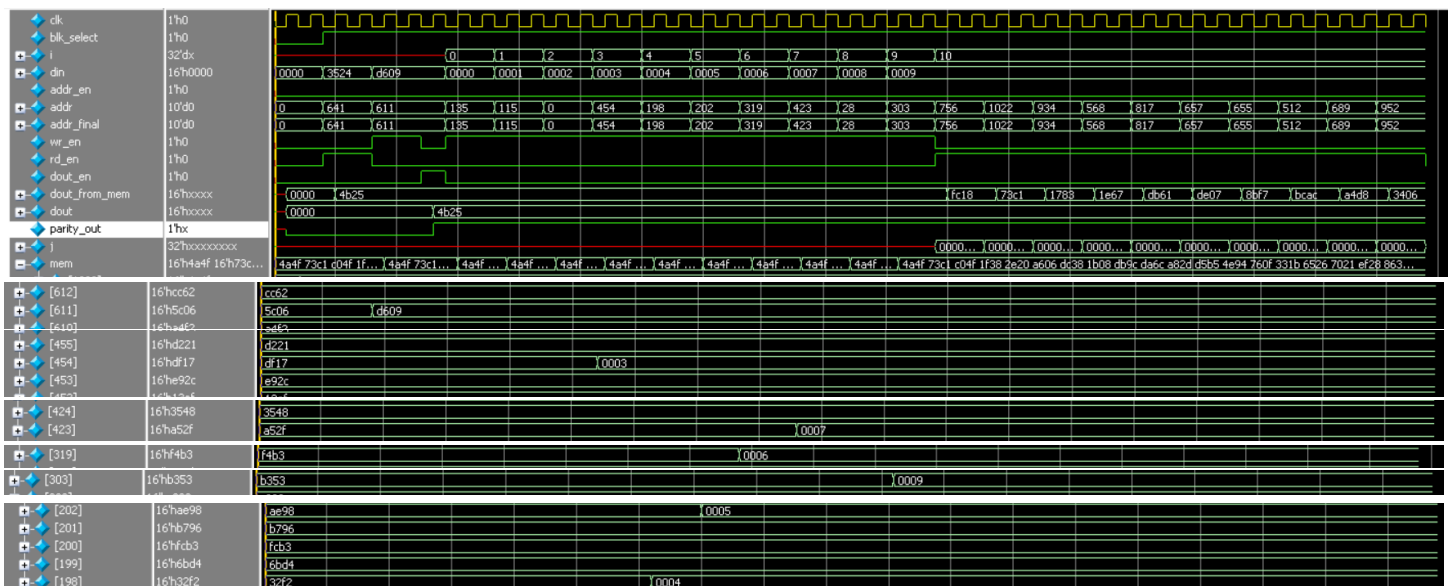
```
1 module single_port_Ram_tb #(
2     parameter MEM_WIDTH = 16,
3     parameter MEM_DEPTH = 1024,
4     parameter ADDR_SIZE = 10,
5     parameter ADDR_PIPELINE = "FALSE",
6     parameter DOUT_PIPELINE = "TRUE",
7     parameter PARITY_ENABLE = 1
8 ) ();
9 /*-----Inputs-----*/
10 reg [MEM_WIDTH-1:0]din;
11 reg [ADDR_SIZE-1:0]addr;
12 reg addr_en;
13 reg dout_en;
14 reg wr_en;
15 reg rd_en;
16 reg blk_select;
17 reg clk;
18 reg rst;
19 /*-----Outputs-----*/
20 wire [MEM_WIDTH-1:0]dout;
21 wire parity_out;
22 /*---Instantiation of DUT and Ref---*/
23 single_port_Ram DUT (.*) ;
24 /*-----Clock generation-----*/
25 initial begin
26     clk=0;
27     forever begin
28         #20; clk=~clk;
29     end
30 end
31 integer i;
32 integer j;
33 /*-----Stimulus process-----*/
34 initial begin
35     $display("---START SIMULATION---");
36     $readmemh("mem.dat",DUT.mem);
37     /* initialize the inputs */
38     rst = 0;
39     addr_en = 0;
40     dout_en = 0;
41     wr_en = 0;
42     rd_en = 0;
43     blk_select = 0;
44     din = 0;
```

```
43     blk_select = 0;
44     din = 0;
45     addr = 0;
46     /* enable reset signal */
47     rst = 1;
48     repeat(2) @(negedge clk);
49     /* disable reset signal */
50     rst = 0;
51     /* enable blk_select */
52     blk_select=1;
53     /* enable read and disable write */
54     rd_en = 1;    wr_en = 0;
55     din = $random;
56     addr = $random;
57     repeat(2) @(negedge clk);
58     /* disable read and enable write */
59     rd_en = 0;    wr_en = 1;
60     din = $random;
61     addr = $random;
62     repeat(2) @(negedge clk);
63     wr_en = 0;
64     /* Dout pipeline (if enabled) */
65     dout_en = 1;
66     @(negedge clk);
67     dout_en = 0;
68     /* Test Case 6: Write and Read multiple addresses */
69     blk_select = 1;
70     wr_en = 1;
71     for ( i = 0; i < 10; i = i + 1) begin
72         din = i;
73         addr = $urandom_range(0,512);
74         repeat(2) @(negedge clk);
75     end
76     wr_en = 0;
77     rd_en = 1;
78     for ( j = 0; j < 10; j = j + 1) begin
79         addr = $urandom_range(512,1024);
80         repeat(2) @(negedge clk);
81     end
82     rd_en = 0;
83
84     $display("---END SIMULATION---");
85     $stop;
86 end
87
88 endmodule
```

3.4. Memory data file

[Mem.dat](#)

3.5. QuestaSim Waveform



3.6. Do File

```
1  #create Work Folder
2  vlib work
3
4  #Compile files with names
5  vlog Single_Port_syn_Ram.v Single_Port_syn_Ram_tb.v
6
7  #simulate The TB file with module Name
8  vsim -voptargs=+acc work.single_port_Ram_tb
9
10 #add the variables and internal signals with specific order to notice them easily
11
12 add wave -position insertpoint \
13 sim:/single_port_Ram_tb/DUT/MEM_WIDTH \
14 sim:/single_port_Ram_tb/DUT/MEM_DEPTH \
15 sim:/single_port_Ram_tb/DUT/ADD_SIZE \
16 sim:/single_port_Ram_tb/DUT/ADDR_PIPELINE \
17 sim:/single_port_Ram_tb/DUT/DOUT_PIPELINE \
18 sim:/single_port_Ram_tb/DUT/PARITY_ENABLE \
19 sim:/single_port_Ram_tb/DUT/rst \
20 sim:/single_port_Ram_tb/DUT/clk \
21 sim:/single_port_Ram_tb/DUT/blk_select \
22 sim:/single_port_Ram_tb/i \
23 sim:/single_port_Ram_tb/DUT/din \
24 sim:/single_port_Ram_tb/DUT/addr \
25 sim:/single_port_Ram_tb/DUT/addr_en \
26 \sim:/single_port_Ram_tb/DUT/addr_final \
27 sim:/single_port_Ram_tb/DUT/wr_en \
28 sim:/single_port_Ram_tb/DUT/rd_en \
29 sim:/single_port_Ram_tb/j \
30 sim:/single_port_Ram_tb/DUT/dout_from_mem \
31 sim:/single_port_Ram_tb/DUT/dout_en \
32 sim:/single_port_Ram_tb/DUT/dout \
33 sim:/single_port_Ram_tb/DUT/parity_out \
34 sim:/single_port_Ram_tb/DUT/mem
35
36 run -all
37
38 wave zoom full
```

3.7. Elaboration

3.7.1. Messages Tab

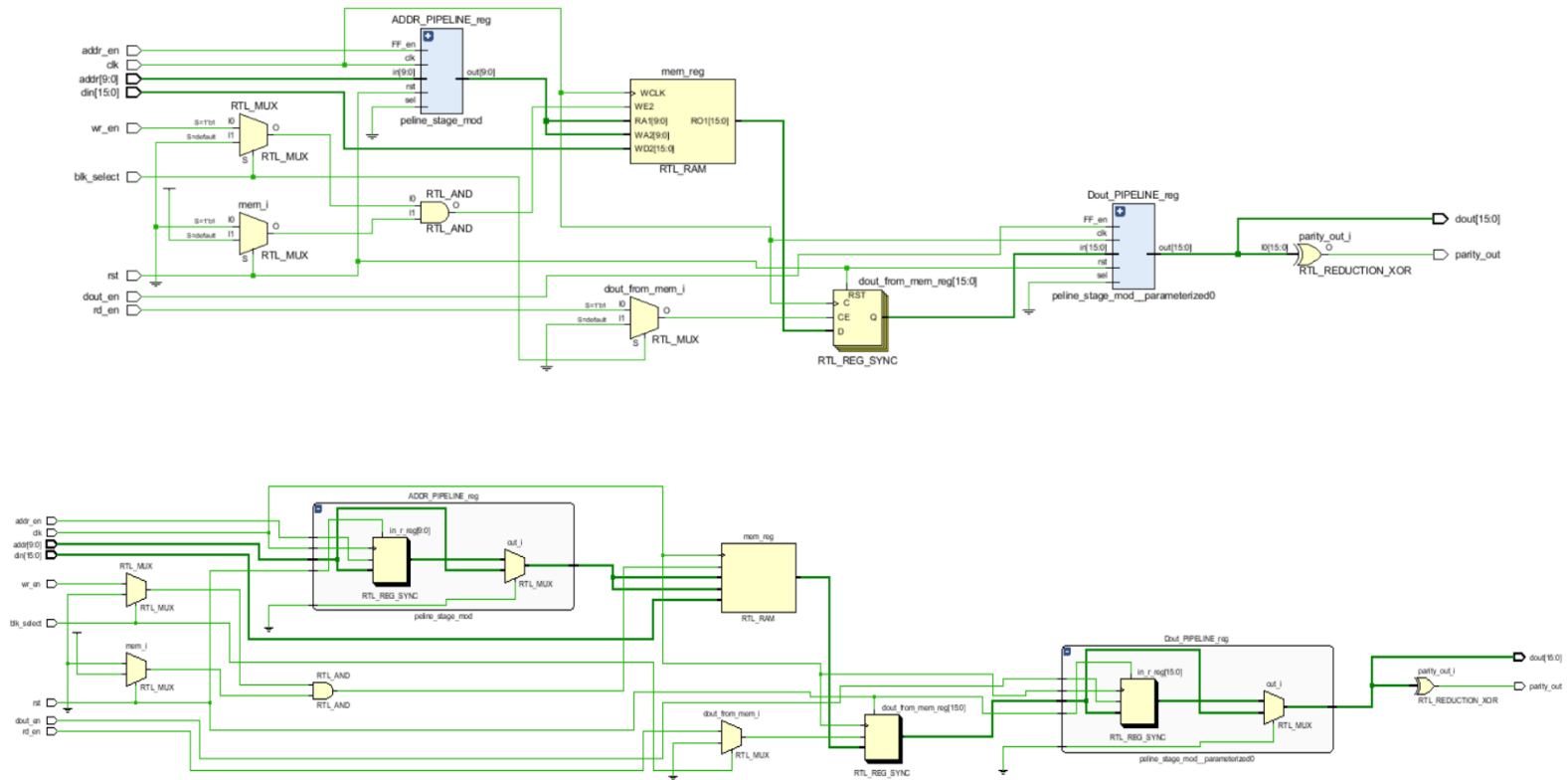
Vivado Commands (4 infos)

- General Messages (4 infos)

Elaborated Design (2 infos)

- General Messages (2 infos)
- [Project 1-570] Preparing netlist for logic optimization
- [Opt 31-138] Pushed 0 inverter(s) to 0 load pin(s).

3.7.2. Schematic



3.8. Synthesis

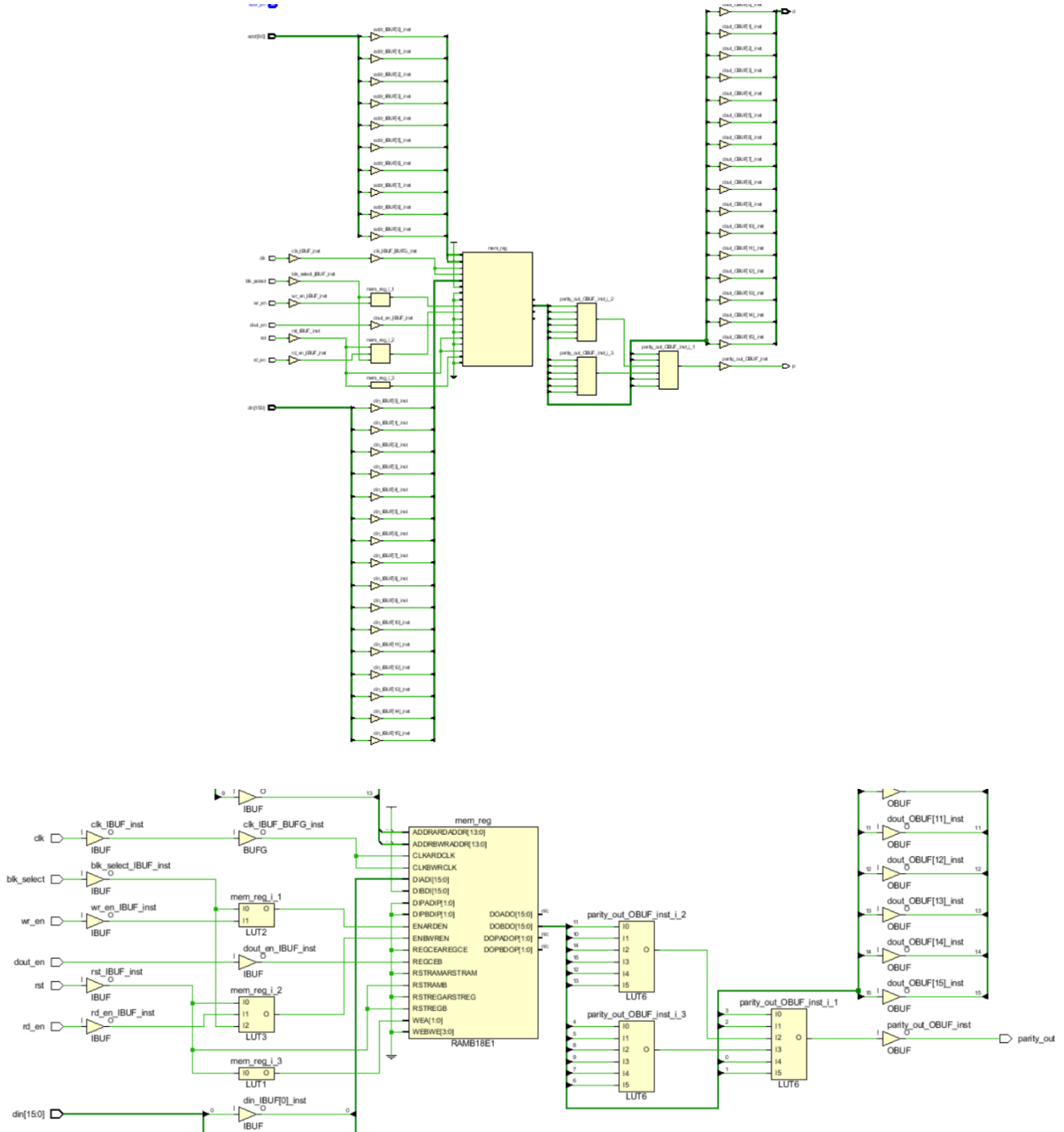
3.8.1. Messages Tab

Warning (11) Info (28) Status (20) Show All

Synthesis (11 warnings)

- [Synth 8-3332] Sequential element (ADDR_PIPELINE_reg/in_r_reg[9]) is unused and will be removed from module single_port_Ram. (9 more like this)
- [Constraints 18-5210] No constraint will be written out.

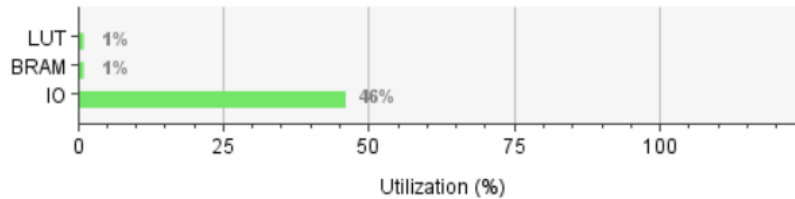
3.8.2. Schematic



3.9. Utilization report summary

Summary

Resource	Utilization	Available	Utilization %
LUT	6	20800	0.03
BRAM	0.50	50	1.00
IO	49	106	46.23

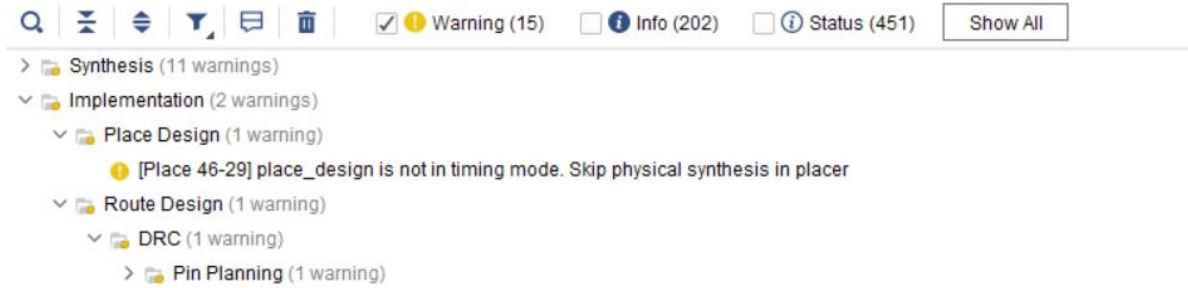


3.10. Snippet from Netlist file

```
1 // Copyright 1986-2018 Xilinx, Inc. All Rights Reserved.
2 //
3 // Tool Version: Vivado v.2018.2 (win64) Build 2258646 Thu Jun 14 20:03:12 MDT 2018
4 // Date : Fri Aug 2 08:00:45 2024
5 // Host : Youssif running 64-bit major Release (build 9200)
6 // Command : write_verilog {D:/CUFE/Diplomas/Digital
7 //           : DiplomaCoding/Assignments/Assignment_5/FPGA/project_single_port_ram/project_single_port_ram.v}
8 // Design : Single port Ram
9 // Purpose : This is a Verilog netlist of the current design or from a specific cell of the design. The output is an
10 //          IEEE 1364-2001 compliant Verilog HDL file that contains netlist information obtained from the input
11 //          design files.
12 // Device : xc7a35t1cpg236-1L
13 //
14 timescale 1 ps / 1 ps
15
16 (* ADDR_PIPELINE = "FALSE" *) (* ADD_SIZE = "10" *) (* DOUT_PIPELINE = "TRUE" *)
17 (* MEM_DEPTH = "1024" *) (* MEM_WIDTH = "16" *) (* PARITY_ENABLE = "1" *)
18 (* STRUCTURAL_NETLIST = "yes" *)
19 module single_port_Ram
20 (din,
21  addr,
22  addr_en,
23  dout_en,
24  wr_en,
25  rd_en,
26  blk_select,
27  clk,
28  rst,
29  dout,
30  parity_out);
31 input [15:0]din;
32 input [9:0]addr;
33 input addr_en;
34 input dout_en;
35 input wr_en;
36 input rd_en;
37 input blk_select;
38 input clk;
39 input rst;
40 output [15:0]dout;
41 output parity_out;
42
43 wire \const0;
44 wire \const1;
45 wire [9:0]addr;
46 wire [9:0]addr_IBUF;
47 wire blk_select;
48 wire blk_select_IBUF;
49 wire clk;
50 wire clk_IBUF;
51 wire clk_IBUF_IBUF;
52 wire [15:0]din;
53 wire [15:0]din_IBUF;
54 wire [15:0]dout;
55 wire [15:0]dout_OBUF;
56 wire dout_en;
57 wire dout_en_IBUF;
58 wire mem_reg_i_1_n_0;
59 wire mem_reg_i_2_n_0;
60 wire mem_reg_i_3_n_0;
61 wire parity_out;
62 wire parity_out_OBUF;
63 wire parity_out_OBUF_inst_i_2_n_0;
64 wire parity_out_OBUF_inst_i_3_n_0;
65 wire rd_en;
66 wire rd_en_IBUF;
67 wire rst;
68 wire rst_IBUF;
69 wire wr_en;
70 wire wr_en_IBUF;
71
```

3.11. Implementation

3.11.1. Messages Tab



3.11.2. Schematic

