EXPERIMENT 2

Group's members:

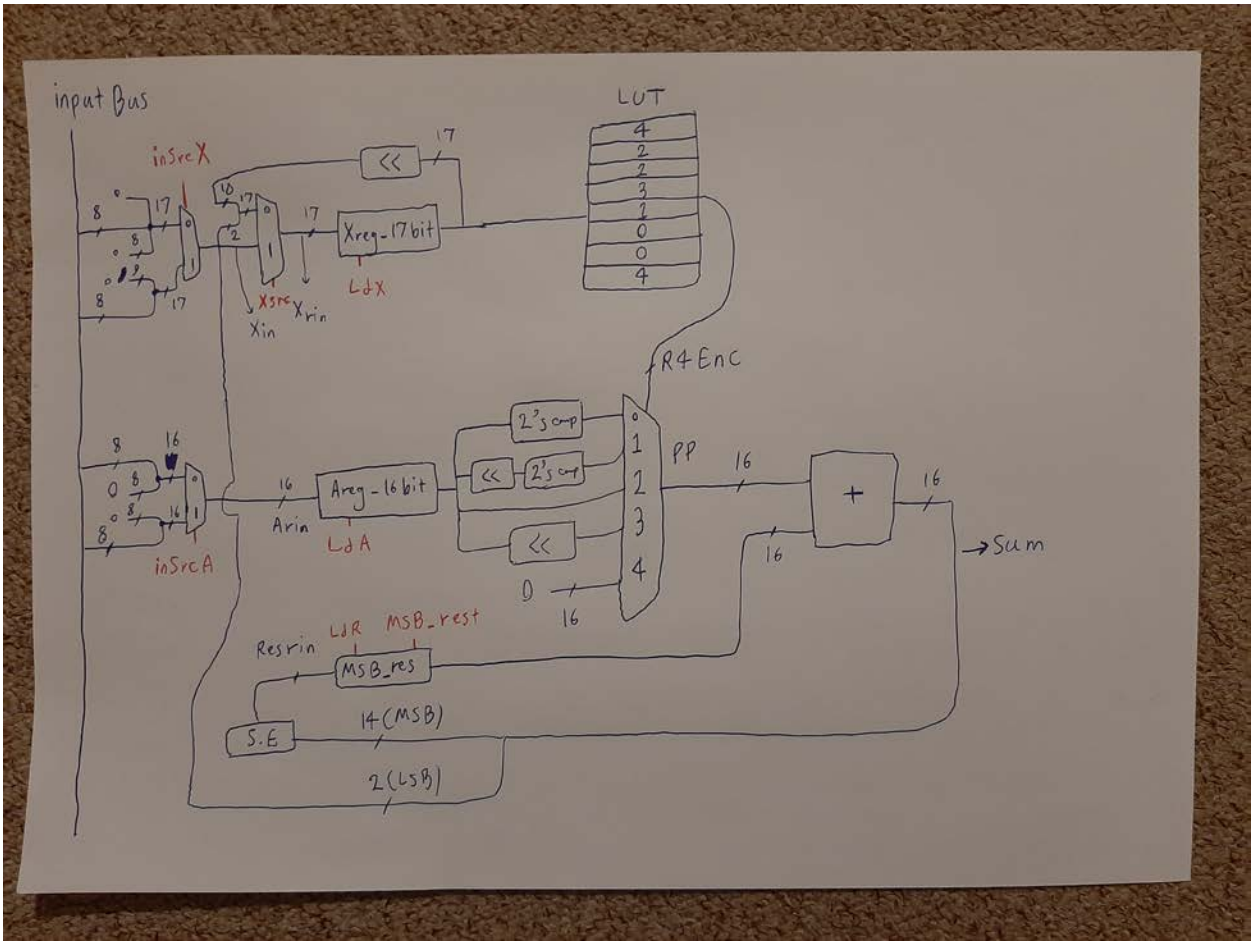
Mohammad Taghizadeh 810198373

Arian Niakan 810198536

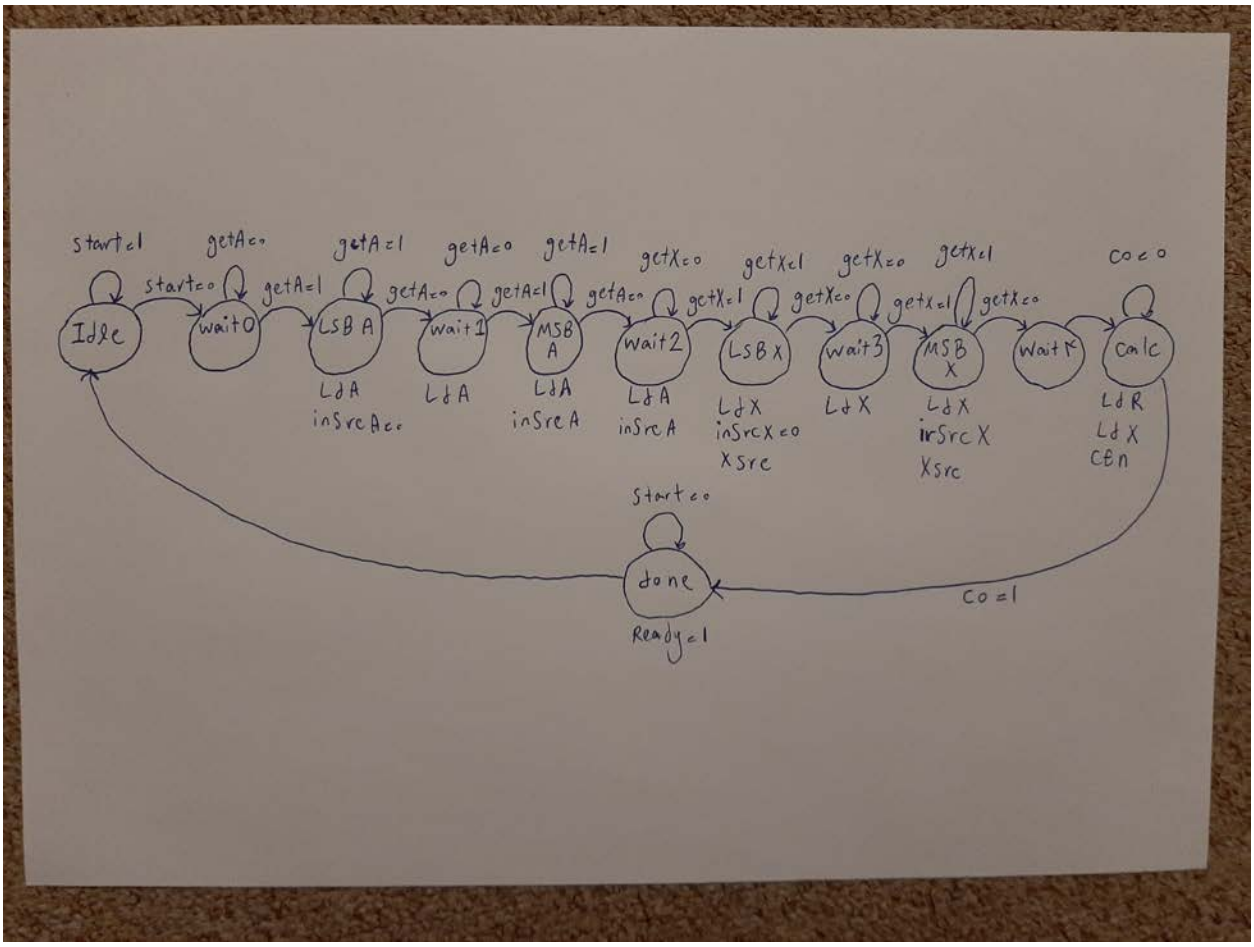
Contents

RTL Design and Simulation

1. Data Path

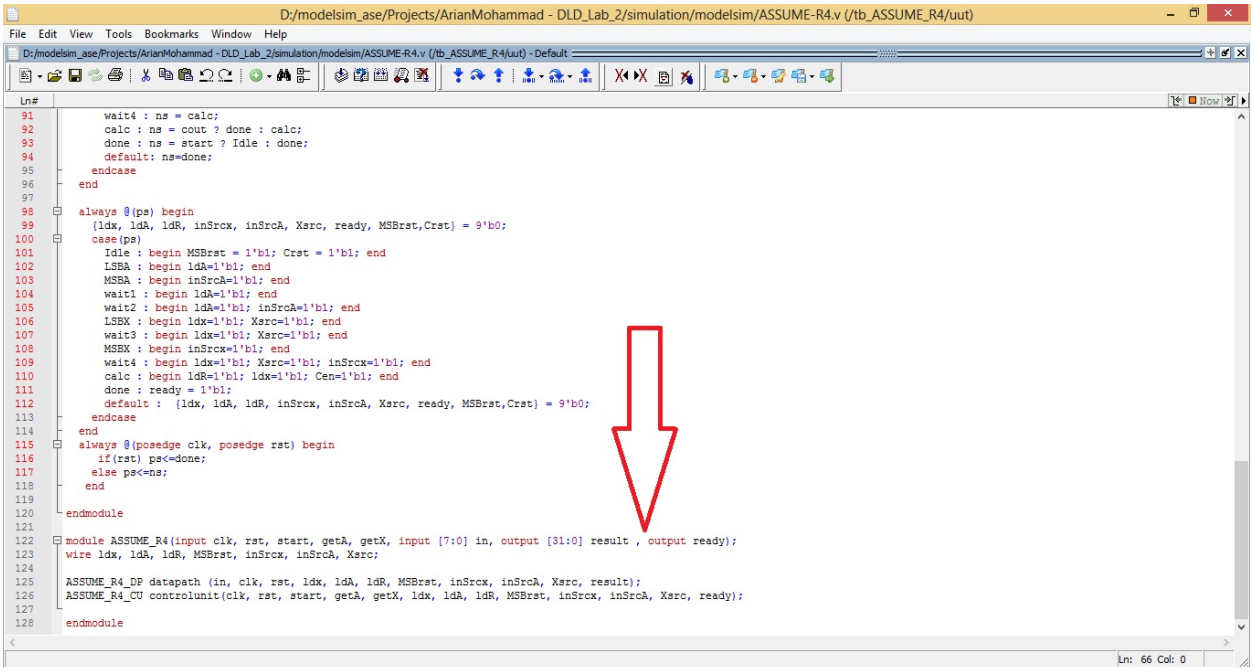


2. Controller state diagram

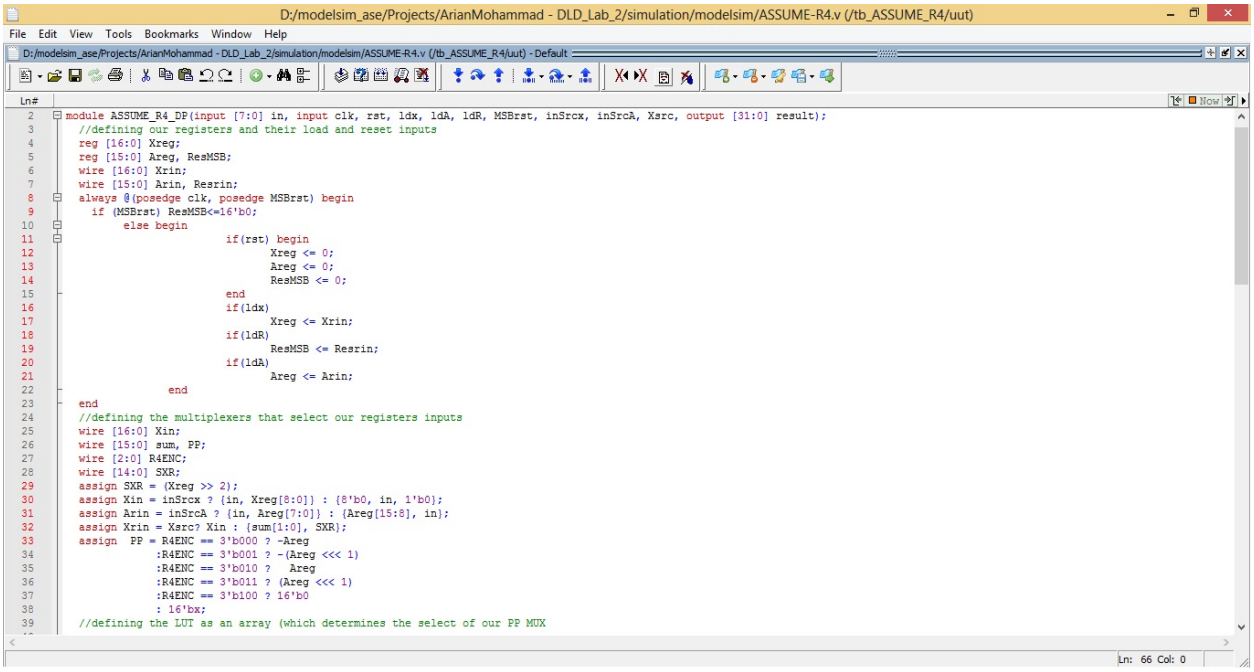


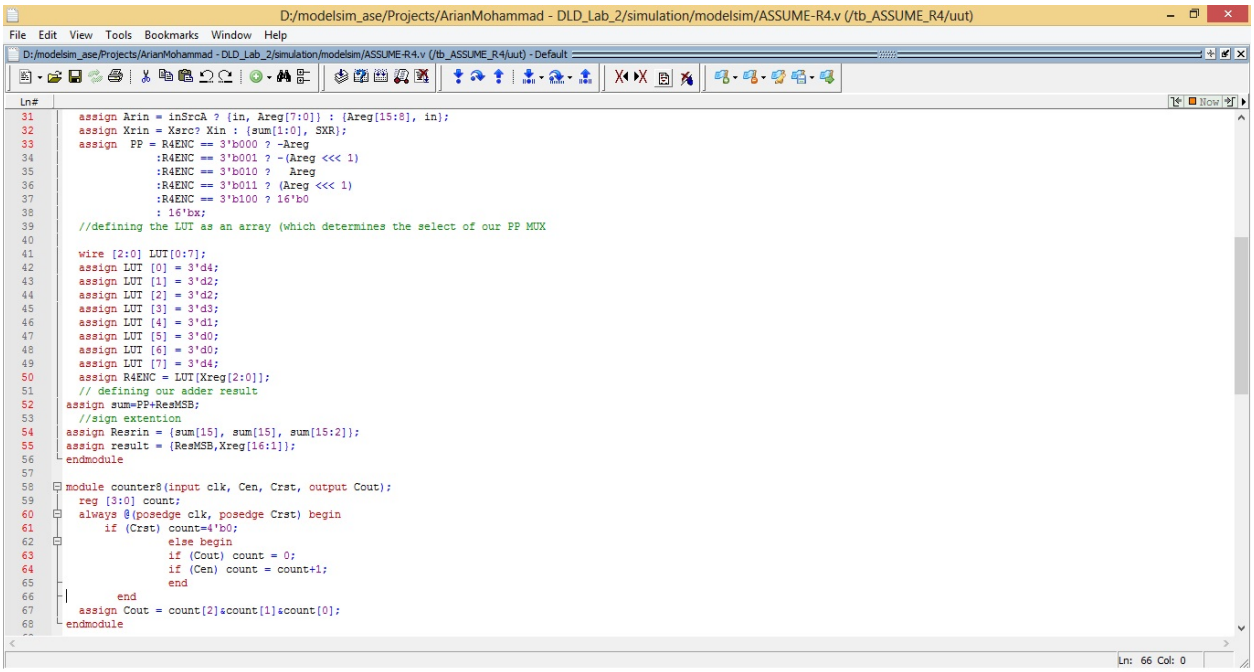
3. Verilog Code of Design

Top Level

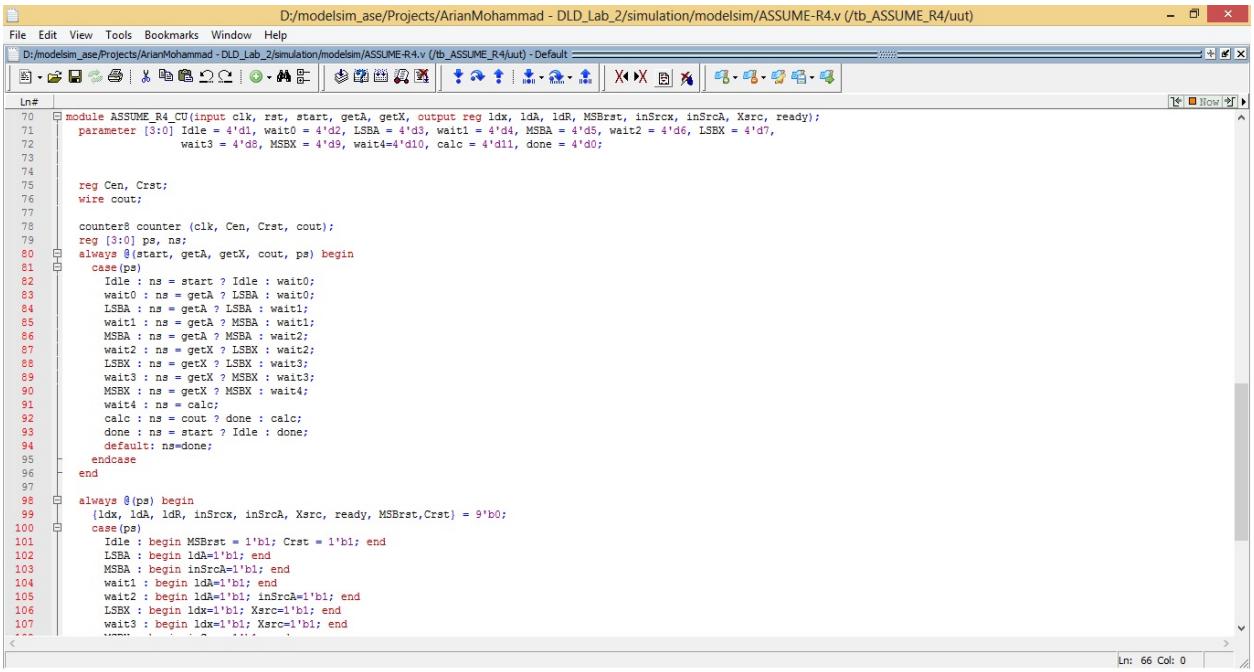


Data Path


D:/modelsim_ase/Projects/ArianMohammad - DLD_Lab_2/simulation/modelsim/ASSUME-R4.v (/tb_ASSUME_R4/uut)
File Edit View Tools Bookmarks Window Help
D:/modelsim_ase/Projects/ArianMohammad - DLD_Lab_2/simulation/modelsim/ASSUME-R4.v (/tb_ASSUME_R4/uut) - Default
Ln#
2 module ASSUME_R4_DP(input [7:0] in, input clk, rst, ldx, ldA, ldR, MSBrst, inSrcx, inSrcA, Xsrc, output [31:0] result);
3 //defining our registers and their load and reset inputs
4 reg [16:0] Xreg;
5 reg [15:0] Areg, ResMSB;
6 wire [16:0] Xrin;
7 wire [15:0] Arin, Resrin;
8 always @(posedge clk, posedge MSBrst) begin
9 if (MSBrst) ResMSB<=16'b0;
10 else begin
11 if(rst) begin
12 Xreg <= 0;
13 Areg <= 0;
14 ResMSB <= 0;
15 end
16 if(ldx)
17 Xreg <= Xrin;
18 if(ldR)
19 ResMSB <= Resrin;
20 if(ldA)
21 Areg <= Arin;
22 end
23 end
24 //defining the multiplexers that select our registers inputs
25 wire [16:0] Xin;
26 wire [15:0] sum, PP;
27 wire [2:0] R4ENC;
28 wire [14:0] SXR;
29 assign SXR = (Xreg >> 2);
30 assign Xin = inSrcx ? {in, Xreg[8:0]} : {8'b0, in, 1'b0};
31 assign Arin = inSrcA ? {in, Areg[7:0]} : {Areg[15:8], in};
32 assign Xrin = Xsrc? Xin : {sum[1:0], SXR};
33 assign PP = R4ENC == 3'b000 ? -Areg
34 :R4ENC == 3'b001 ? -(Areg <<< 1)
35 :R4ENC == 3'b010 ? Areg
36 :R4ENC == 3'b011 ? (Areg <<< 1)
37 :R4ENC == 3'b100 ? 16'b0
38 : 16'bx;
39 //defining the LUT as an array (which determines the select of our PP MUX
Ln: 66 Col: 0

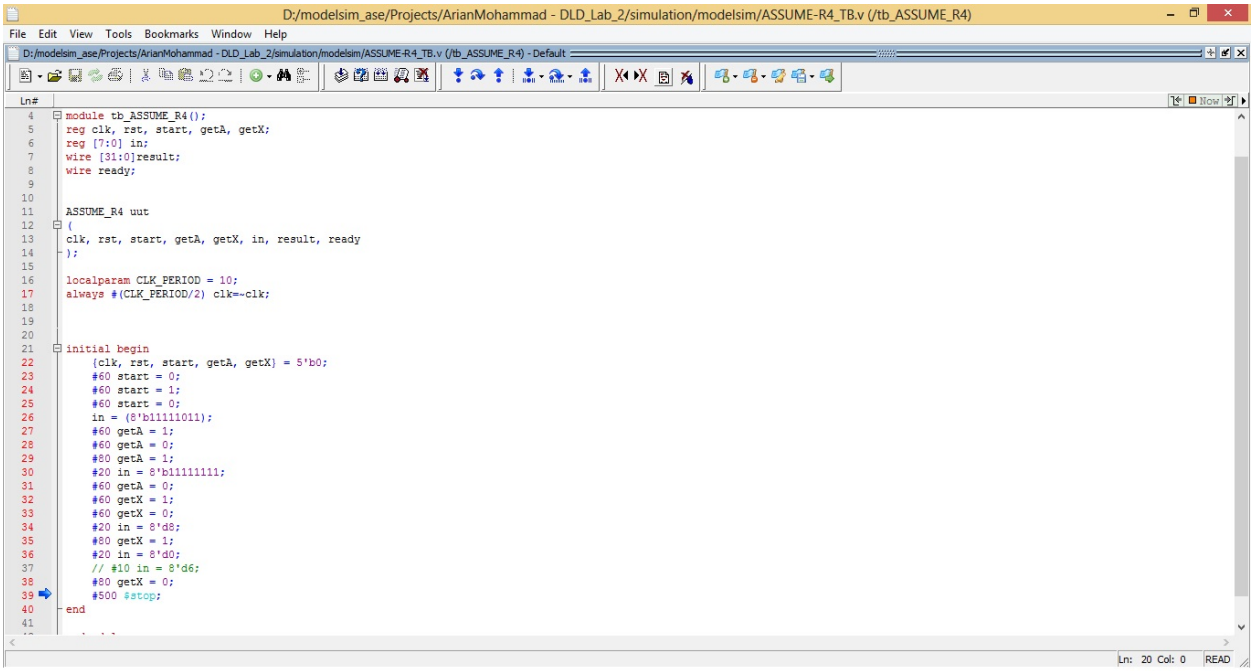

D:/modelsim_ase/Projects/ArianMohammad - DLD_Lab_2/simulation/modelsim/ASSUME-R4.v (/tb_ASSUME_R4/uut)
File Edit View Tools Bookmarks Window Help
D:/modelsim_ase/Projects/ArianMohammad - DLD_Lab_2/simulation/modelsim/ASSUME-R4.v (/tb_ASSUME_R4/uut) - Default
Ln#
31 assign Arin = inSrcA ? {in, Areg[7:0]} : {Areg[15:8], in};
32 assign Xrin = Xsrc? Xin : {sum[1:0], SXR};
33 assign PP = R4ENC == 3'b000 ? -Areg
34 :R4ENC == 3'b001 ? -(Areg <<< 1)
35 :R4ENC == 3'b010 ? Areg
36 :R4ENC == 3'b011 ? (Areg <<< 1)
37 :R4ENC == 3'b100 ? 16'b0
38 : 16'bx;
39 //defining the LUT as an array (which determines the select of our PP MUX
40
41 wire [2:0] LUT[0:7];
42 assign LUT [0] = 3'd4;
43 assign LUT [1] = 3'd2;
44 assign LUT [2] = 3'd2;
45 assign LUT [3] = 3'd3;
46 assign LUT [4] = 3'd1;
47 assign LUT [5] = 3'd0;
48 assign LUT [6] = 3'd0;
49 assign LUT [7] = 3'd4;
50 assign R4ENC = LUT[Xreg[2:0]];
51 // defining our adder result
52 assign sum=PP+ResMSB;
53 //sign extention
54 assign Resrin = {sum[15], sum[15], sum[15:2]};
55 assign result = {ResMSB,Xreg[16:1]};
56 endmodule
57
58 module counter8(input clk, Cen, Crst, output Cout);
59 reg [3:0] count;
60 always @(posedge clk, posedge Crst) begin
61 if (Crst) count=4'b0;
62 else begin
63 if (Cout) count = 0;
64 if (Cen) count = count+1;
65 end
66 end
67 assign Cout = count[2]&count[1]&count[0];
68 endmodule
Ln: 66 Col: 0

Controller




```
module ASSUME_R4_CU(input clk, rst, start, getA, getX, output reg ldx, ldA, ldR, MSBrst, inSrcx, inSrcA, Xsrc, ready);
  parameter [3:0] Idle = 4'd1, wait0 = 4'd2, LSBA = 4'd3, wait1 = 4'd4, MSBA = 4'd5, wait2 = 4'd6, LSBX = 4'd7,
                  wait3 = 4'd8, MSBX = 4'd9, wait4=4'd10, calc = 4'd11, done = 4'd0;


  reg Cen, Crst;
  wire cout;

  counter8 counter (clk, Cen, Crst, cout);
  reg [3:0] ps, ns;
  always @(start, getA, getX, cout, ps) begin
    case(ps)
      Idle : ns = start ? Idle : wait0;
      wait0 : ns = getA ? LSBA : wait0;
      LSBA : ns = getA ? LSBA : wait1;
      wait1 : ns = getA ? MSBA : wait1;
      MSBA : ns = getA ? MSBA : wait2;
      wait2 : ns = getX ? LSBX : wait2;
      LSBX : ns = getX ? LSBX : wait3;
      wait3 : ns = getX ? MSBX : wait3;
      MSBX : ns = getX ? MSBX : wait4;
      wait4 : ns = calc;
      calc : ns = cout ? done : calc;
      done : ns = start ? Idle : done;
      default: ns=done;
    endcase
  end

  always @(ps) begin
    {ldx, ldA, ldR, inSrcx, inSrcA, Xsrc, ready, MSBrst,Crst} = 9'b0;
    case(ps)
      Idle : begin MSBrst = 1'b1; Crst = 1'b1; end
      LSBA : begin ldA=1'b1; end
      MSBA : begin inSrcA=1'b1; end
      wait1 : begin ldA=1'b1; end
      wait2 : begin ldA=1'b1; inSrcA=1'b1; end
      LSBX : begin ldx=1'b1; Xsrc=1'b1; end
      wait3 : begin ldx=1'b1; Xsrc=1'b1; end
```

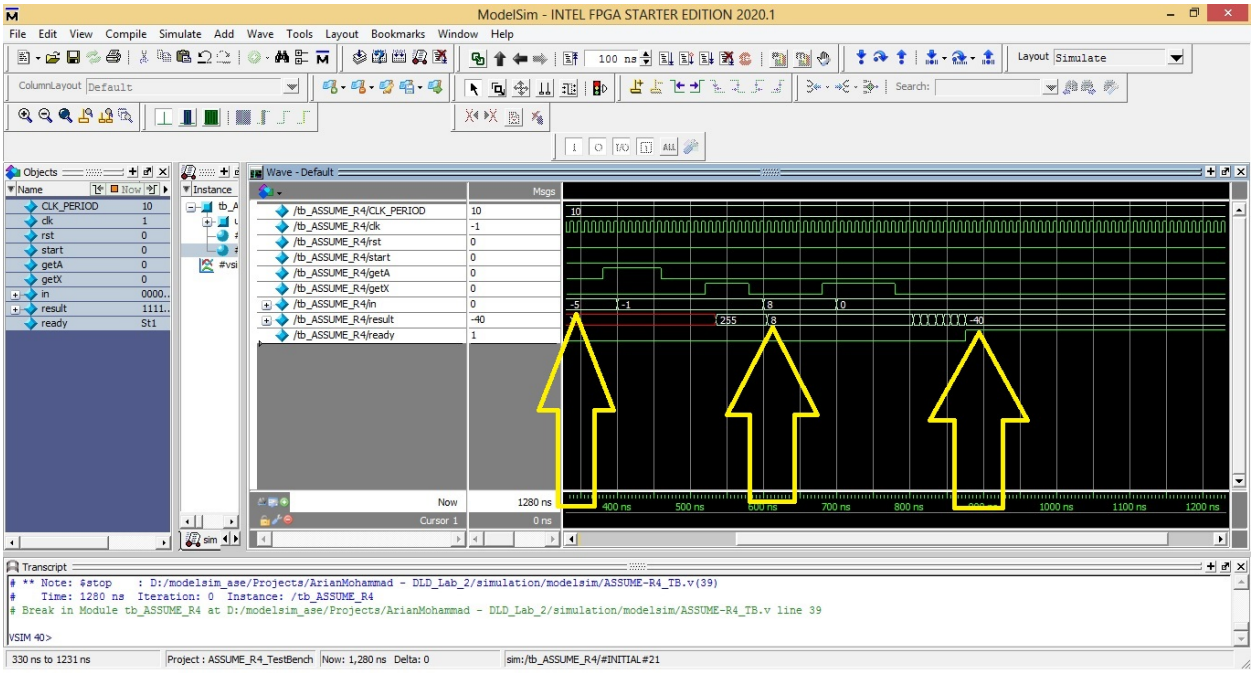
Verilog Code of Design's Test Bench



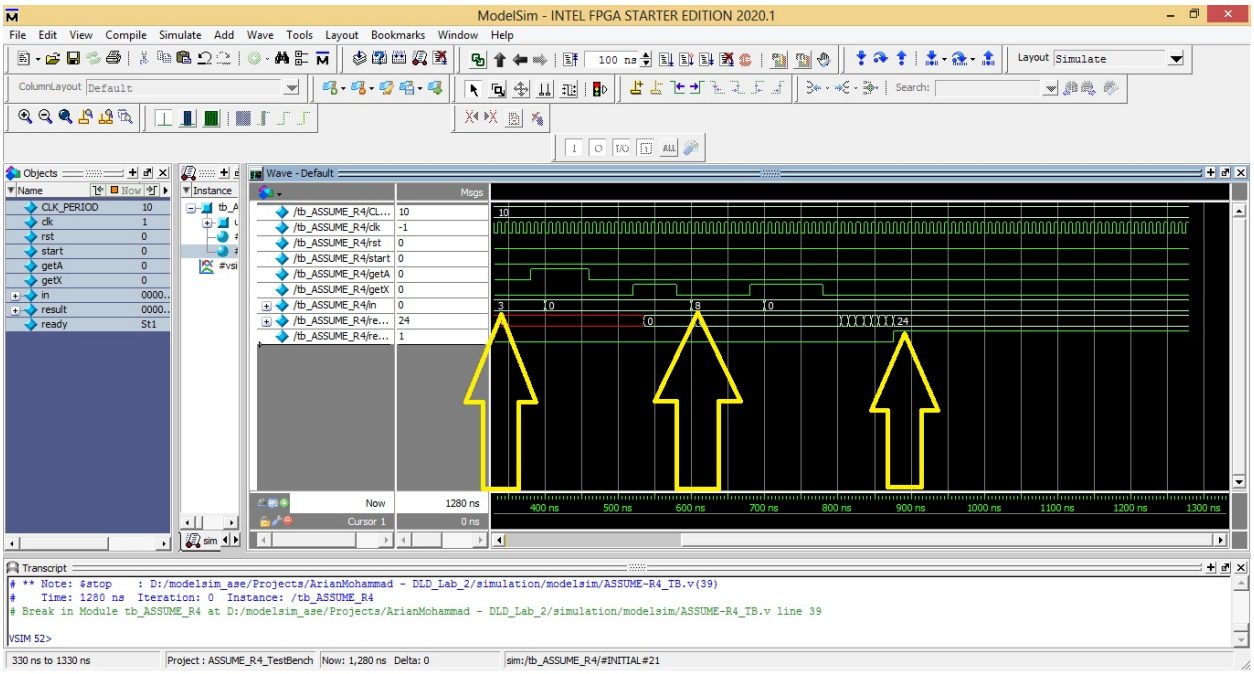

```
module tb_ASSUME_R4();
reg clk, rst, start, getA, getX;
reg [7:0] in;
wire [31:0]result;
wire ready;


ASSUME_R4 uut
(
clk, rst, start, getA, getX, in, result, ready
);

localparam CLK_PERIOD = 10;
always #(CLK_PERIOD/2) clk=~clk;


initial begin
    {clk, rst, start, getA, getX} = 5'b0;
    #60 start = 0;
    #60 start = 1;
    #60 start = 0;
    in = (8'b11111011);
    #60 getA = 1;
    #60 getA = 0;
    #80 getA = 1;
    #20 in = 8'b11111111;
    #60 getA = 0;
    #60 getX = 1;
    #60 getX = 0;
    #20 in = 8'd8;
    #80 getX = 1;
    #20 in = 8'd0;
    // #10 in = 8'd6;
    #80 getX = 0;
    #500 $stop;
end
```

Simulation output of above Design

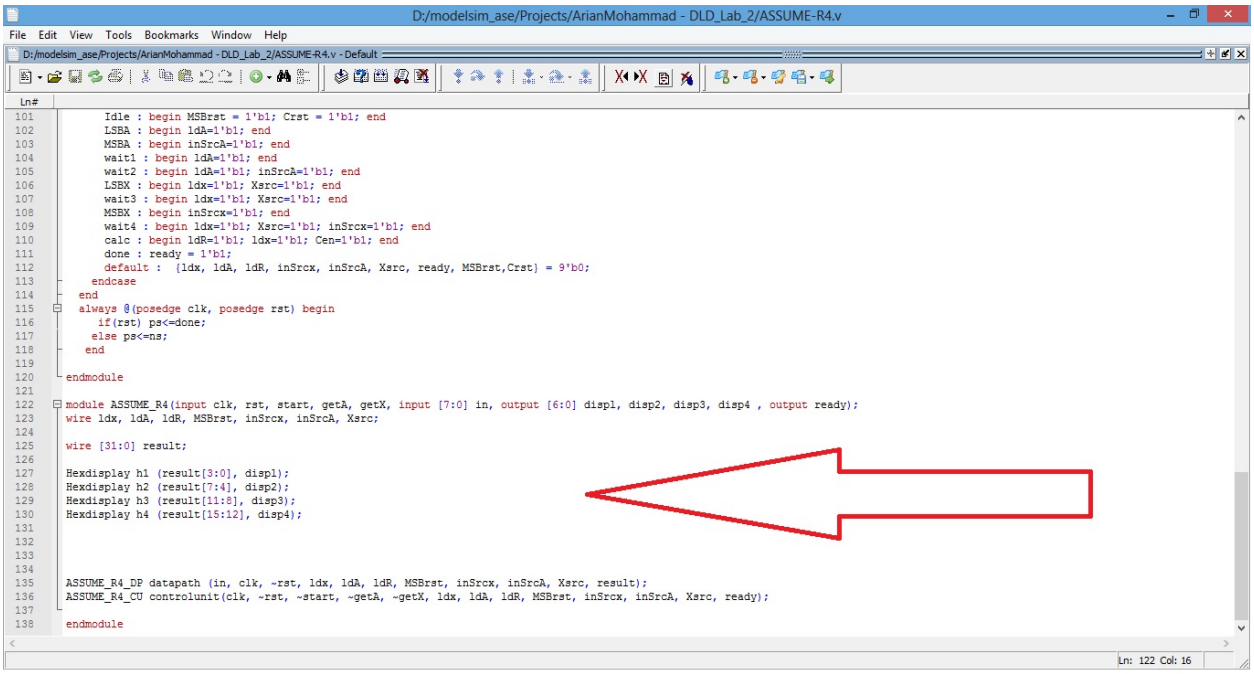
Signed number Multiplication



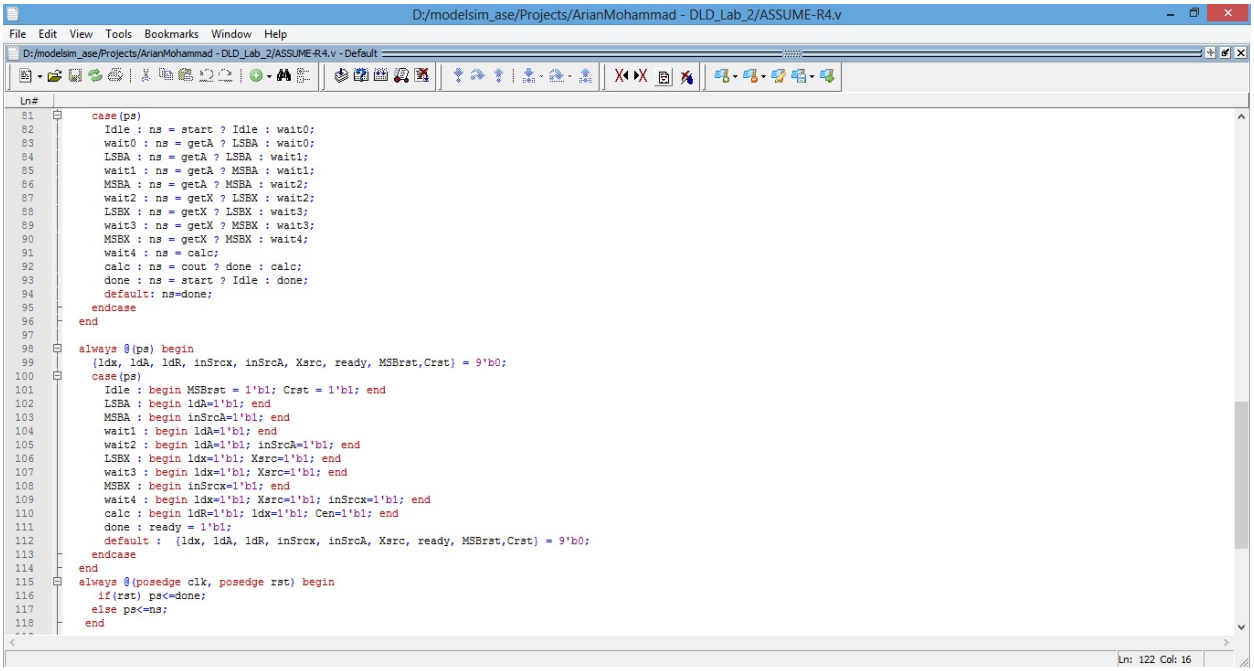
Unsigned number Multiplication



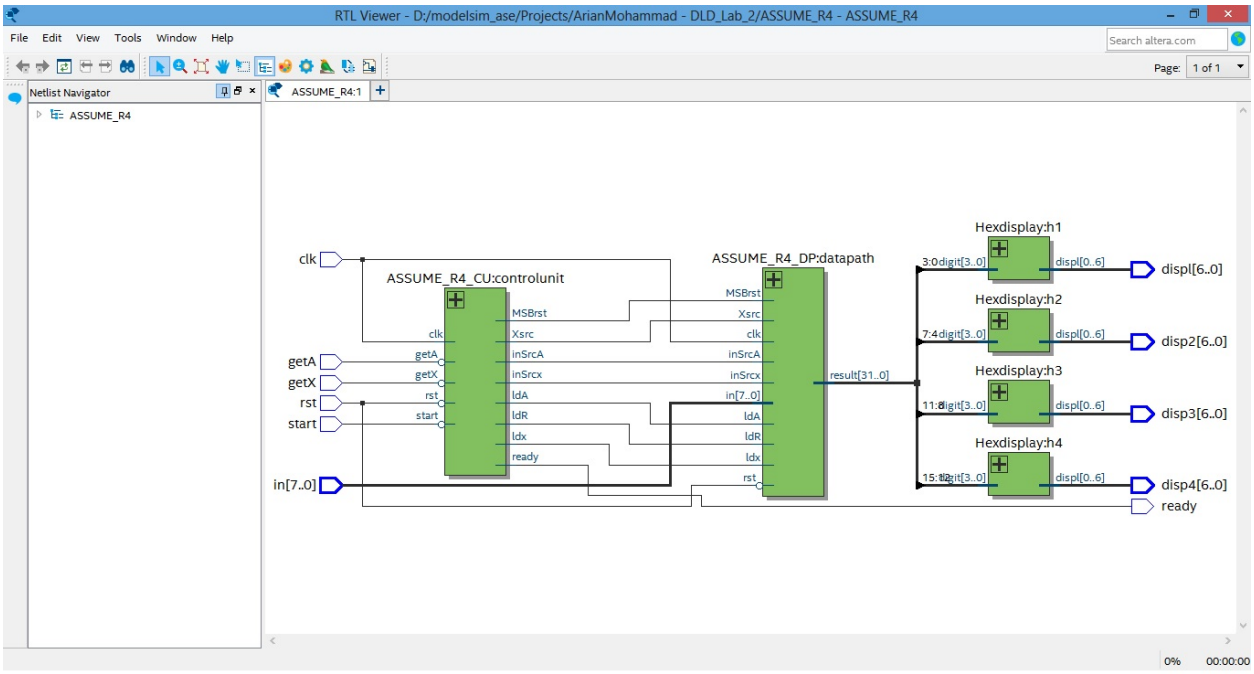
Modified Top Level of previous RTL Design



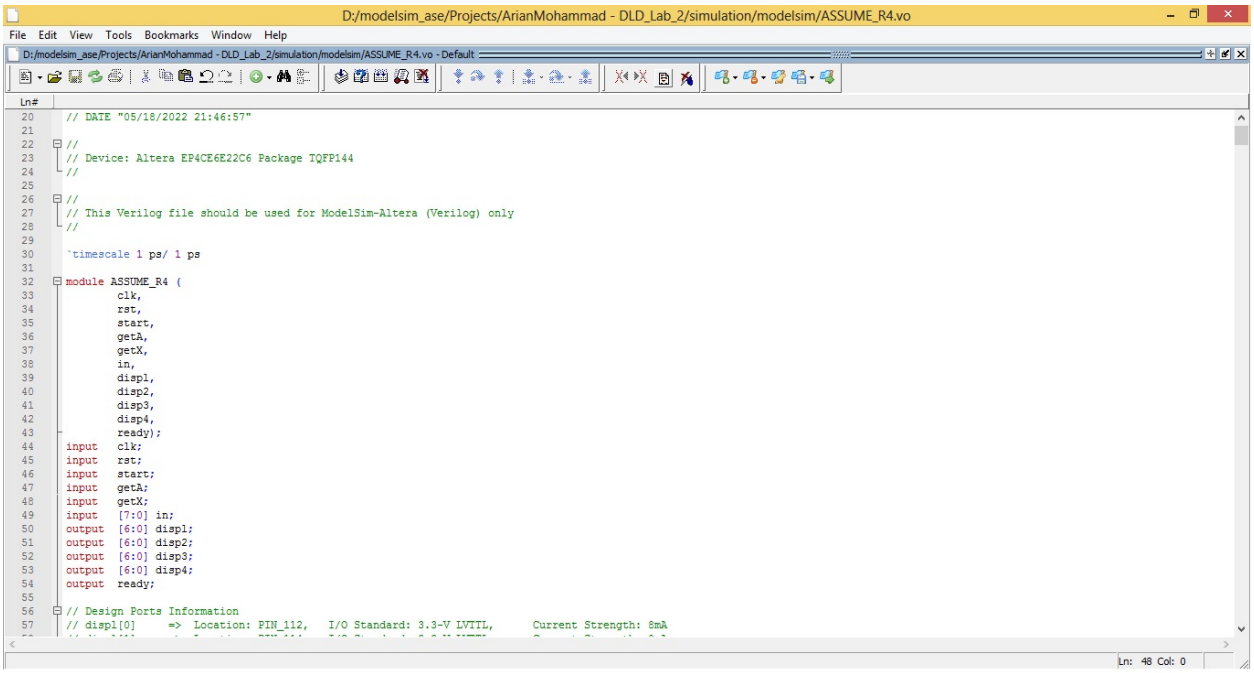
Modified Controller of previous RTL Design



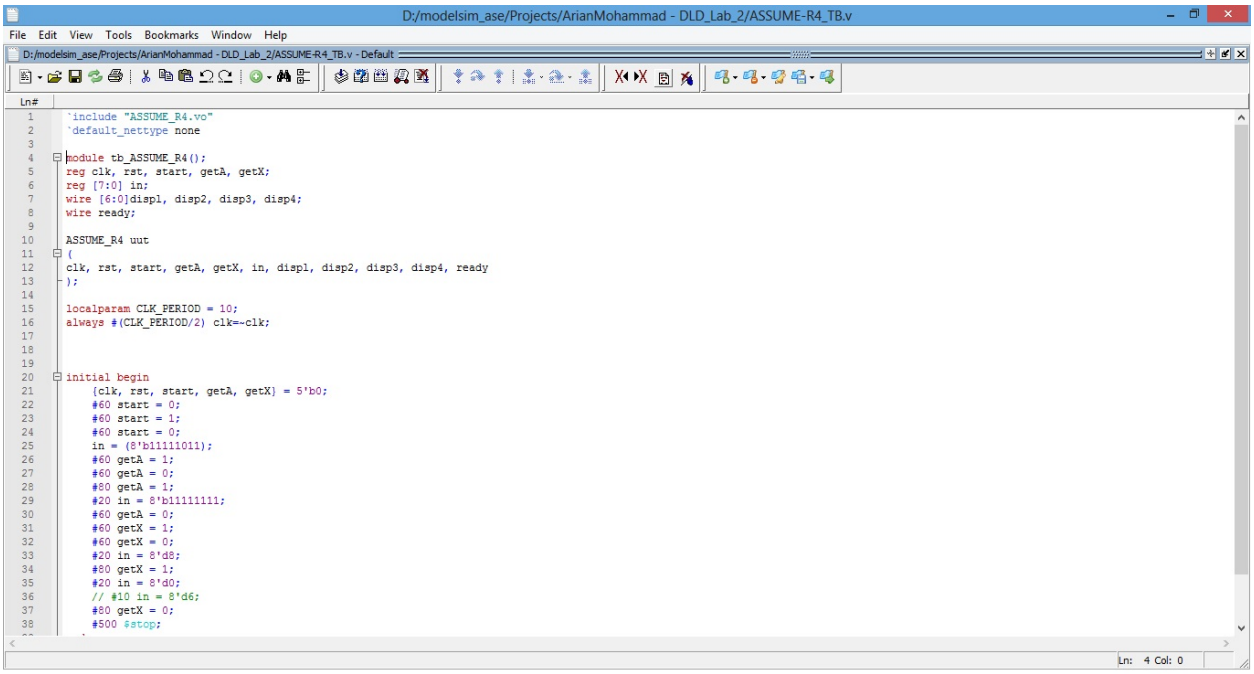
Quartus RTL viewer output



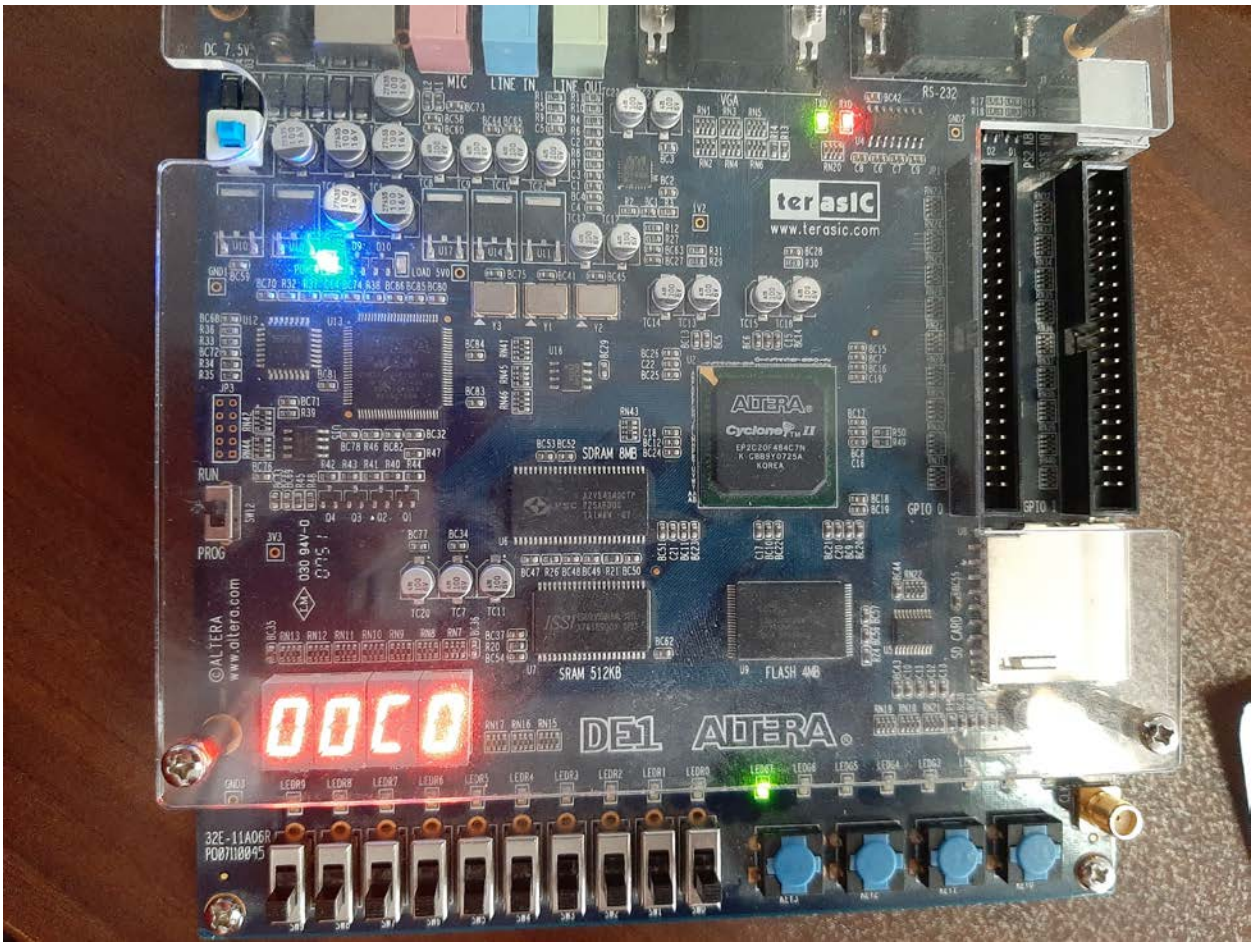
Quartus output Verilog code



Quartus Verilog output Test Bench



FPGA output



Inputs are 3 and 4.

As you can see in above image, product of inputs (3 and 4) is shown on FPGA's seven segments