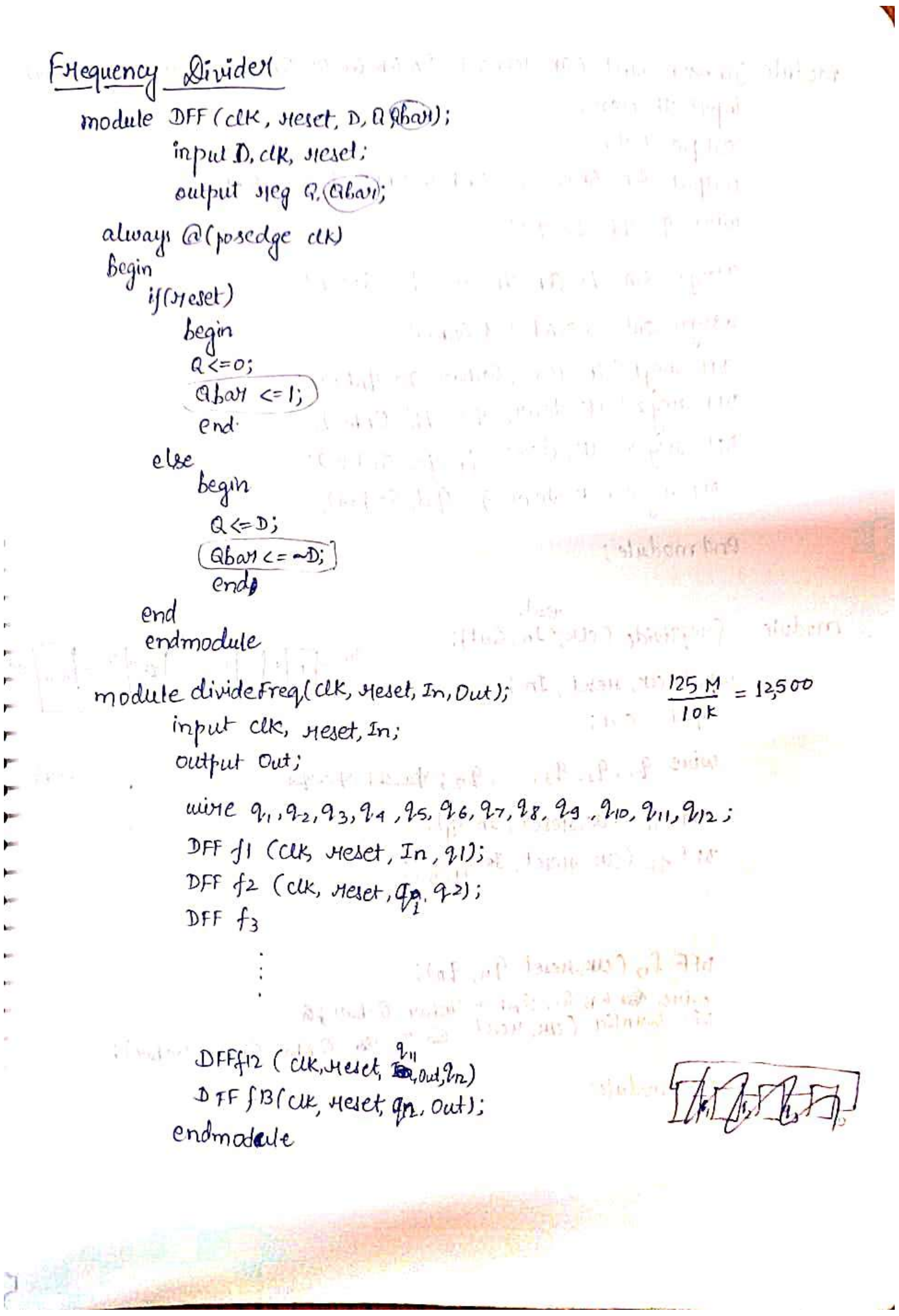
Digital Electronics Verilog Lab 6

Submitted By:
Saurabh Kumar
SC ID: SC22B146
Course: DIGITAL ELECTRONICS AND VLSI DESIGN LAB (AV232)

PRE-LAB TASK

Question: Write a verilog code for down conversion of the in-built clock of 125 MHz to 10 kHz (nearest possible).
Sol:



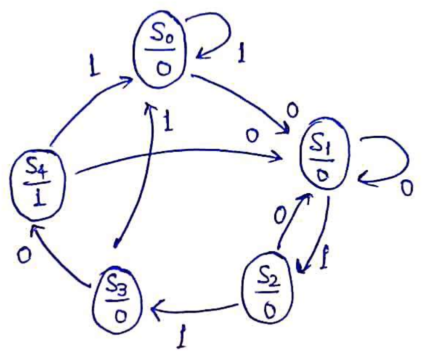
Frequency Divider

```
module DFF(clk, reset, D, Q, Qbar);
        input D, clk, reset;
        output reg Q, Qbar;
  always @(posedge clk)
  begin
     if(reset)
        begin
         Q<=0;
         Qbar <=1;
         end
     else
        begin
         Q<=D;
         Qbar<=~D;
         end
    end
    endmodule

 module divideFreq(clk, reset, In, Out);          125 M / 10k = 12,500
        input clk, reset, In;
        output Out;
         wire q1,q2,q3,q4,q5,q6,q7,q8,q9,q10,q11,q12;
         DFF f1 (clk, reset, In, q1);
         DFF f2 (clk, reset, q1, q2);
         DFF f3
               :
               :

          DFFf12 ( clk, reset, q11, q12)
          DFF f13(clk, reset, q12, Out);
        endmodule
```

LAB SHEET

Question No. 1. Integrate VIO and ILA IP cores in problem 1 of the previous lab sheet and visualize input-outputs virtually.
Sol: Moore Machine:

Moore Machine: 0110

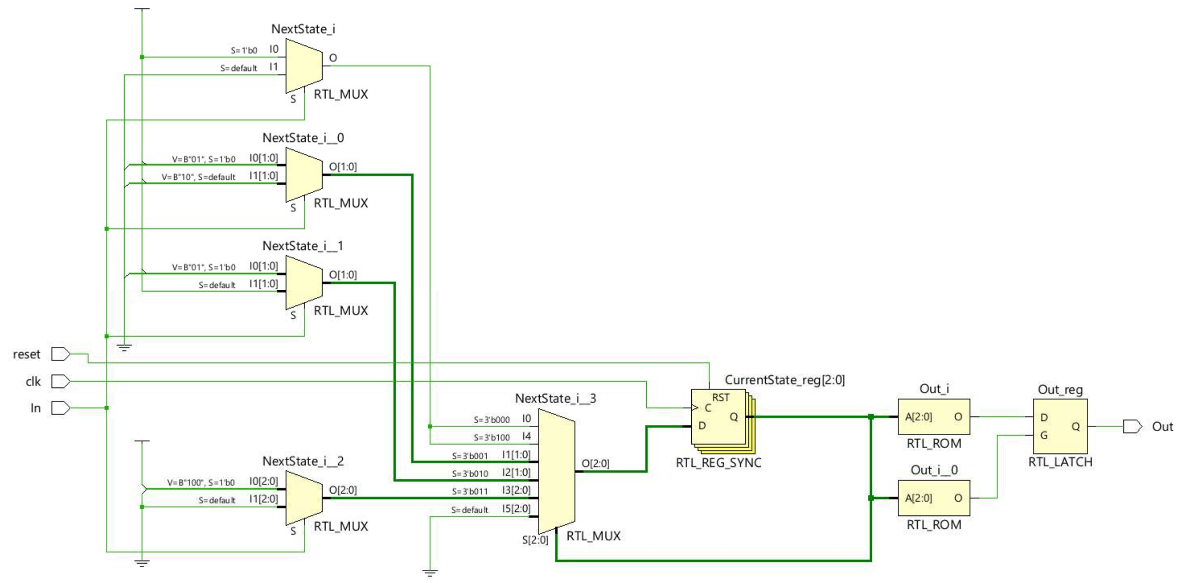


Code:

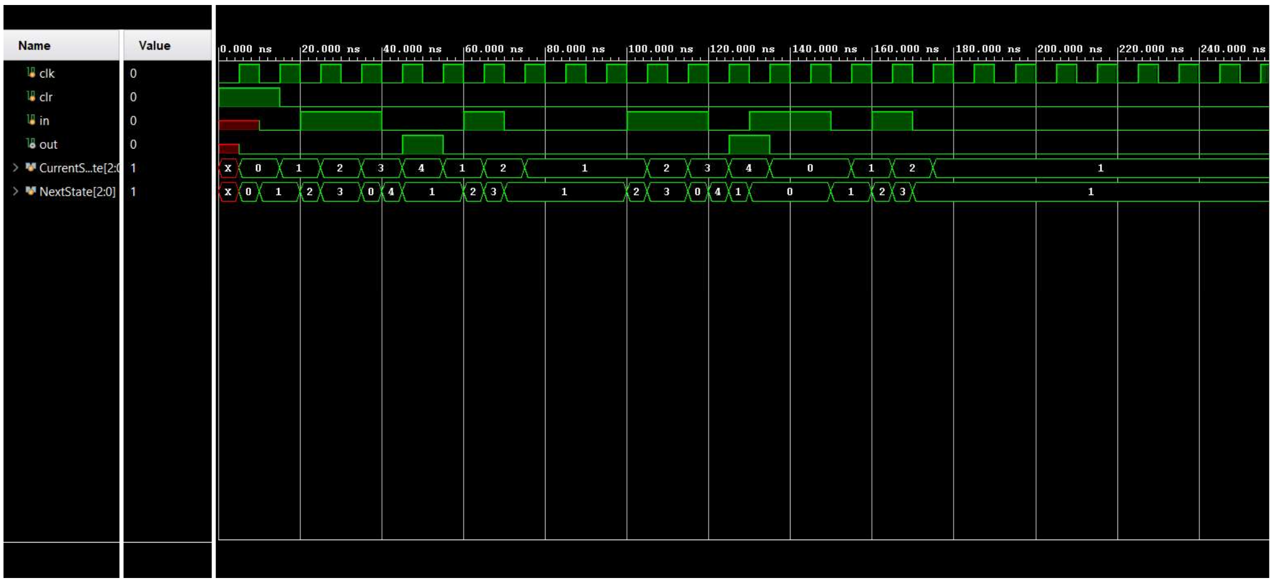
```
module detect_0110_Moore(clk,reset,In,Out);
   input clk, reset, In;
   output reg Out;

parameter S0 = 3'b000, S1 = 3'b001, S2 = 3'b010, S3 = 3'b011, S4 = 3'b100;
reg[2:0] CurrentState,NextState;

always@(posedge clk)
begin
if(reset)
   CurrentState = S0;
else
   CurrentState = NextState;
end

always@(CurrentState, In)
begin
case(CurrentState)
S0:begin
   if(!In) NextState = S1;
   else NextState = S0;
   end
S1:begin
   if(!In) NextState = S1;
   else NextState = S2;
   end
S2:begin
   if(!In) NextState = S1;
   else NextState = S3;
   end
S3:begin
   if(!In) NextState = S4;
   else NextState = S0;
   end
S4:begin
   if(!In) NextState = S1;
   else NextState = S0;
   end
default:NextState = S0;
endcase
end

always@(CurrentState)
begin
case(CurrentState)
```

```
S0: Out = 0;
S1: Out = 0;
S2: Out = 0;
S3: Out = 0;
S4: Out = 1;
endcase
end

endmodule
```

Schematic:

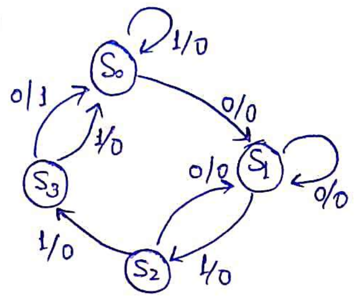


Simulation:



Mealy Machine:

Mealy Machine : 0110

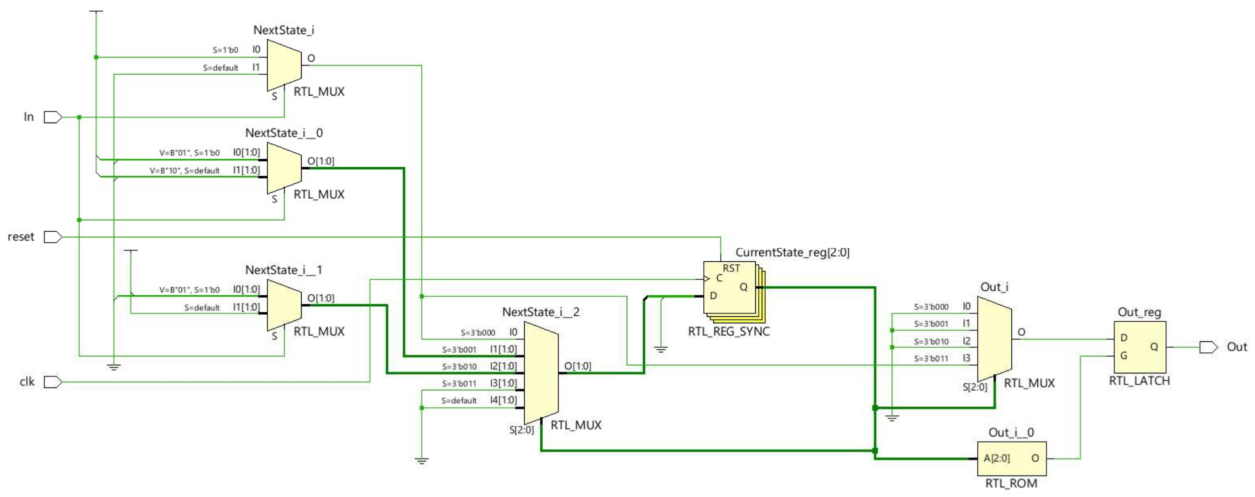


Code:

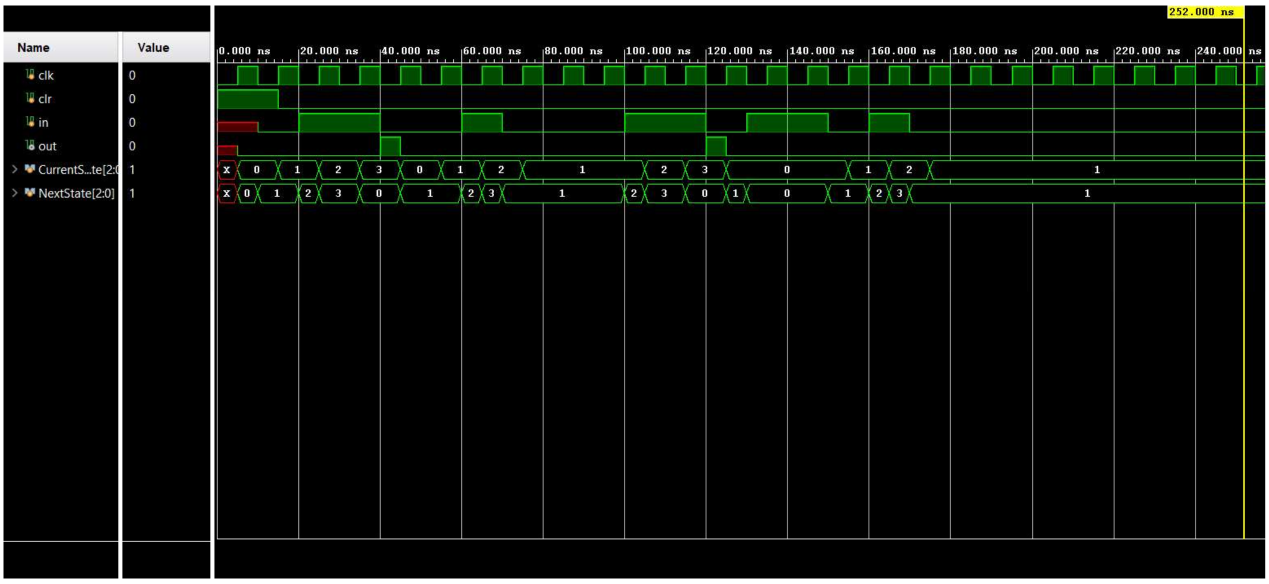
```
module detect_0110_Mealy(clk,reset,In,Out);
   input clk, reset, In;
   output reg Out;

parameter S0 = 3'b000, S1 = 3'b001, S2 = 3'b010, S3 = 3'b011;
reg[2:0] CurrentState,NextState;

always@(posedge clk)
begin
if(reset)
   CurrentState = S0;
else
   CurrentState = NextState;
end

always@(CurrentState, In)
begin
case(CurrentState)
S0:begin
   if(!In)
      begin
      NextState = S1;
      Out = 0;
      end
   else
      begin
      NextState = S0;
      Out = 0;
      end
   end
S1:begin
   if(!In)
      begin
      NextState = S1;
      Out = 0;
      end
   else
      begin
      NextState = S2;
      Out = 0;
      end
   end
S2:begin
   if(!In)
```

```
        begin
        NextState = S1;
        Out = 0;
        end
    else
        begin
        NextState = S3;
        Out = 0;
        end
    end
S3:begin
    if(!In)
        begin
        NextState = S0;
        Out = 1;
        end
    else
        begin
        NextState = S0;
        Out = 0;
        end
    end
default:NextState = S0;
endcase
end

Endmodule

endmodule
```

Schematic:



Simulation:



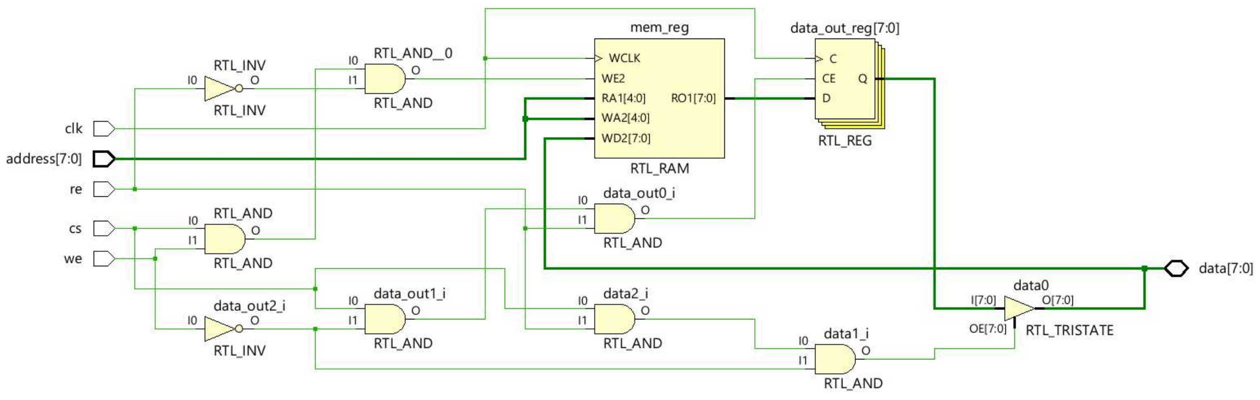
Question No. 2. Design RAM and ROM of size 5*5 to store 8-bits data. Use the design steps from design template shown in the presentation to include enable options for read and write operations, reset, etc. in the code.

Sol: **RAM:**

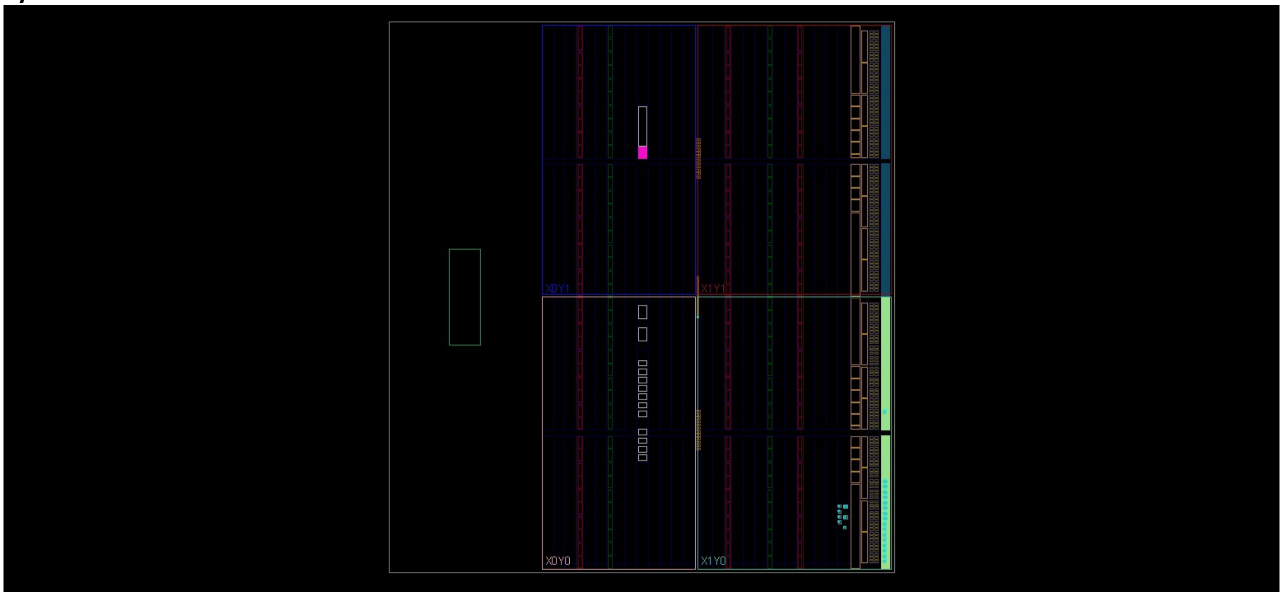
Code:

```
module ram(clk,address,data,cs,we,re);
  input clk,cs,we,re;
  input[7:0] address;
  inout[7:0] data;

  reg[7:0] data_out;
  reg[7:0] mem[0:25];
  assign data = (cs && re && !we) ? data_out:8'bz;

  //Memory Write Block when we=1; cs=1; re=0
  always@(posedge clk)
  begin:MEM_WRITE
    if(cs && we && !re) begin
      mem[address] = data;
    end
  end

  //Memory Read Block
  //Read Operation: When we=0; re=1; cs=1
  always@(posedge clk)
  begin:MEM_READ
    if(cs && !we && re) begin
      data_out = mem[address];
    end
  end
endmodule
```

Schematic:



Synthesis:



ROM:

Code:

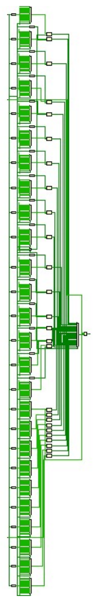
module rom(clock,add,dataOut,en);
  input clock,en;
  input[7:0] add;
  output reg [7:0] dataOut;
  reg[7:0] memory[0:25];

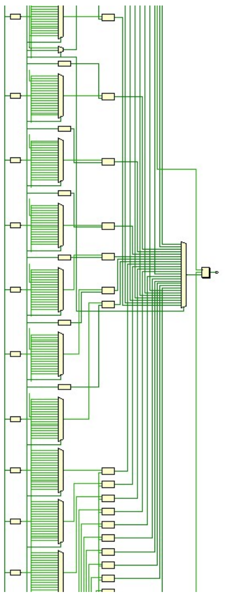
  initial
  begin
    dataOut = 25'b0;
  end

  always@(add)
  case(add)
    8'b00000000 : memory[add] = 25'b00000001;
    8'b00000001 : memory[add] = 25'b00000010;
    8'b00000010 : memory[add] = 25'b00000011;
    8'b00000011 : memory[add] = 25'b00000100;
    8'b00000100 : memory[add] = 25'b00000101;
    8'b00000101 : memory[add] = 25'b00000110;
    8'b00000110 : memory[add] = 25'b00000111;
    8'b00000111 : memory[add] = 25'b00001000;
    8'b00001000 : memory[add] = 25'b00001001;

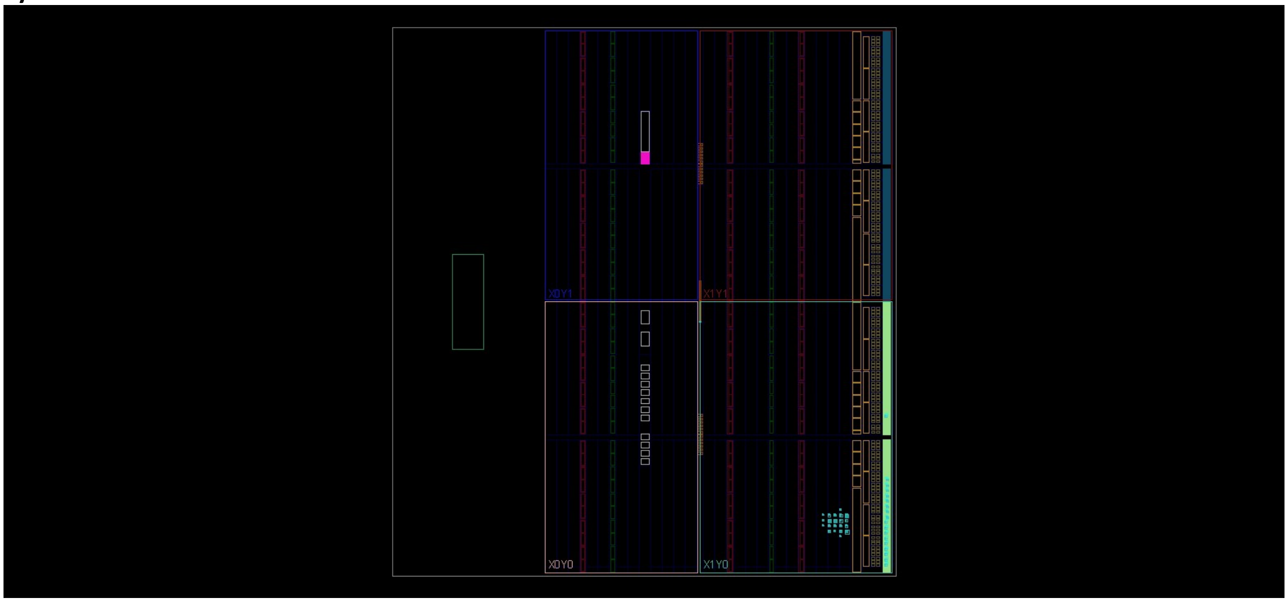
```
        8'b00001001 : memory[add] = 25'b00001010;
        8'b00001010 : memory[add] = 25'b00001011;
        8'b00001011 : memory[add] = 25'b00001100;
        8'b00001100 : memory[add] = 25'b00001101;
        8'b00001101 : memory[add] = 25'b00001110;
        8'b00001110 : memory[add] = 25'b00001111;
        8'b00001111 : memory[add] = 25'b00010000;

        default: memory[add] = 25'b00000000;

    endcase



    always@(posedge clock)
    begin
        if(en) begin
            dataOut <= memory[add];
        end
        else
            dataOut <= dataOut;
    end


endmodule
```

Schematic:





Synthesis:



ASSIGNMENT

Question: In problem 2 above, use a text file to feed the data using $readmemb or $readmemh commands.
Sol: Code:

```
module memory(inbus, outbus, add,rd_mem, wr_mem);
  input [5:0] add;
  input [7:0] inbus;
  input rd_mem, wr_mem;

  output [7:0] outbus;

  reg [7:0] mem [0:25];

// mem read data from file
initial begin
  $readmemb("./memfile.txt",mem,50,60);
end
//read part

assign outbus = rd_mem ? mem [add] : 8'bz;
always @ (wr_mem)
  if (wr_mem == 1) mem [add] <= inbus;
```

endmodule

Schematic: