ECE451Lab4

Huanjia Liu (ID:831993149)

Oct 30, 2020

Contents

1 Objective

During this lab, we are asked to design a ROM circuit in the Candence to store some message and write a Verilog program to achieve displaying frame on 8*8 LED matrix. And here are the general objective we need to realize.

1 Use hierarchical design to create complex circuits

2 Understand the design and function of a ROM and Decoder

3 Blinkenlights!

2 Introduction

2.1 Candence Part

In Candence part, we are asked to Use a 7xN ROM to write an English word or phrase at least 5 ASCII characters long to the output. The input is a char select number, in range: $[\log_2 N, 0]$. And the output is the ASCII format characters which is stored in ROM.

2.2 Verilog Part

In Verilog part, we are ased to use a ROM to display a sequence of frames to an 8x8 LED matrix. We must use at least 4 different frames. The frame rate should be 0.25 Hz at the slowest. (4 seconds per frame). We may make the frame rate as fast as necessary, so long as it is apparent that you are using multiple frames. As a guideline, 8-15 Hz should be sufficient for a simple animation. There are three input: clock50MHZ, run, reset. And two output 8 bits row data and 8 bits column data. Finally the output pins will connect to LED matrix to realize displaying.

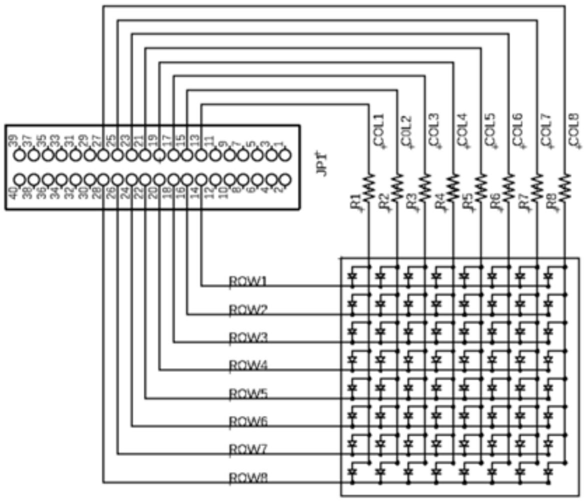

Figure 1: LED matrix circuit

3 Procedure and analysis

3.1 Candence Part

In order to implement ROM, decoder is require, here is my circuit of 3:8 decoder.

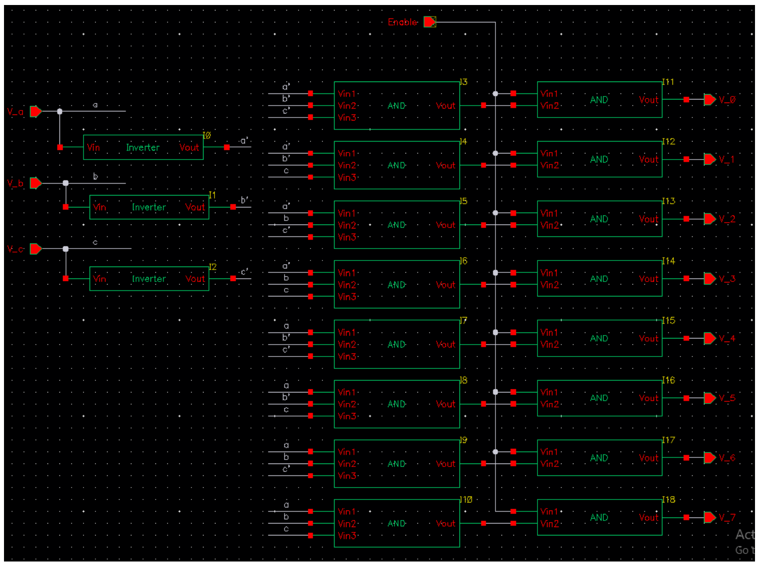

Figure 2: LED matrix circuit

Also here is my wave of decoder from to confirm my circuit works well.

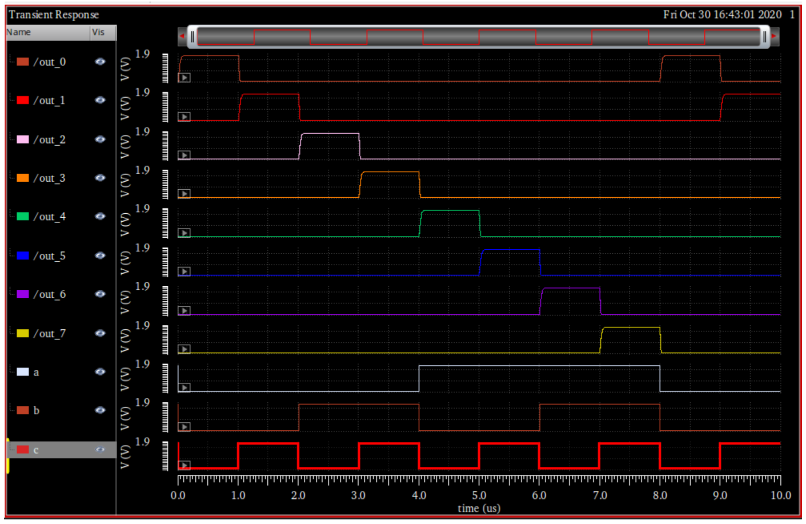

Figure 3: Waveform of 3:8 decoder

Then adding the decoder to ROM circuit, here is the circuit (the green block is 3-8 decoder) and the waveform after simulating. The words I store in ROM is "GO RAMS ".

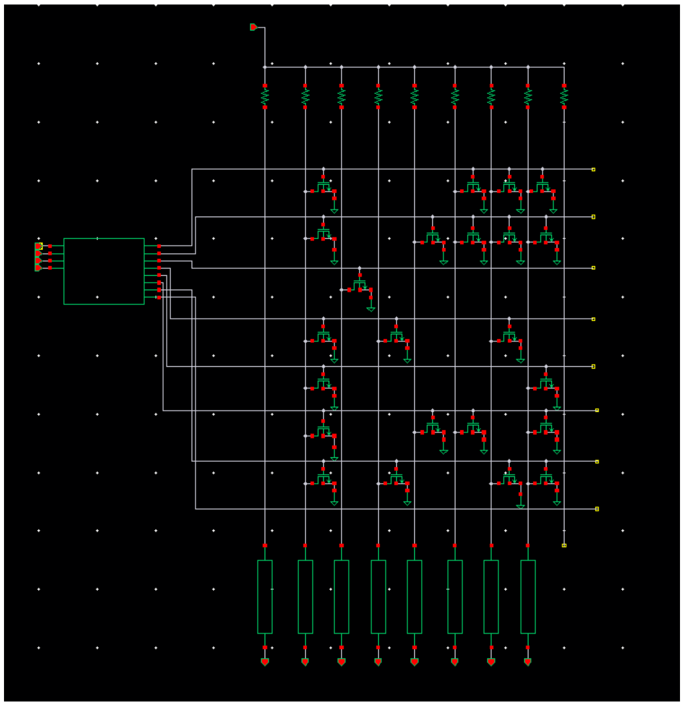

Figure 4: ROM circuit

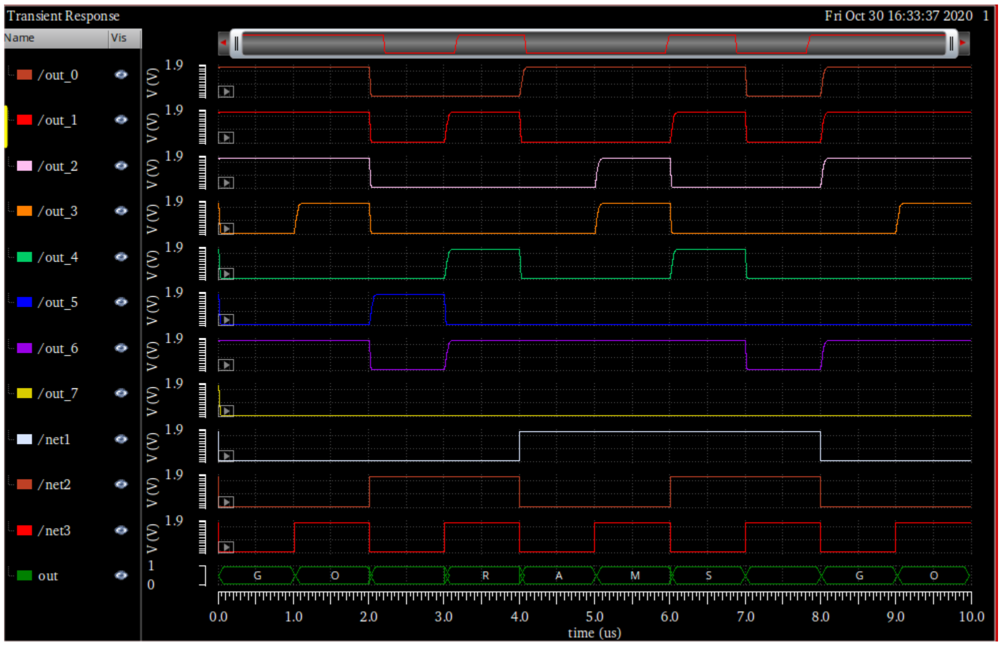

Figure 5: Final display

3.2 Verilog Part

In Verilog part, the most important part is clock divider module, here I use 50MHZ crystal oscillator clock generate from my FPGA DE1 board.

```
reg [12:0] row_counter;
reg  clk_counter;
always @(posedge clk)
begin
   if(row_counter ==0 )
   begin
      clk_counter <= ~clk_counter;
   end
   row_counter <= row_counter +1;
end
```

Variable **row_counter** can control the clock time length, and variable **clk_counter** is the new clock signal be created whose period is depend on **row_counter**

Then is the LED display part, **frame** control the picture which will be displayed on LED matrix, we are asked to display 4 pictures, thus, I use 2 bit to declare **frame**. Varibale **z** control the delay time between four frames, and variable **i** control the update rate LED matirx. By the way, I add run pin here: when run pin is on, **z** will keep count, however, if run pin is off, **z** will keep same which means variable **frame** which is controled by the **z** will keep same, so that the frame be displayed will keep same.

```
reg [3:0] i = 3'b000;

reg [12:0] z;

reg [1:0] frame = 2'b00;

always @(posedge clk_counter)

begin

  case(frame)

  2'b01:

  begin

    case(i)

    \\ROM part, I will explain it later.

    endcase;

  end


  2'b10:

  begin

    case(i)

    \\ROM part, I will explain it later.

    endcase;

  end


  2'b11:

  begin

    case(i)
```

```
      \\ROM part, I will explain it later.
      endcase;
   end

   2'b00:
   begin
      case(i)
      \\ROM part, I will explain it later.
      endcase;
    end
   endcase
   i = i+3'b001;
   if(run)
   begin
      z <= z + 1;
   end
   if(z == 0)
   begin
      frame <= frame+1;
   end
if(reset)
begin
   frame <= 2'b01;
end

end
```

The code below is LED matrix display, each case only display one column, but if the update rate is quick enough, all the columns will combine together because of visual residue. (hline control which column will be displayed, and vline control the picture data).

```
case(i)

  3'b000:

  begin

    hline = 8'b01111111;

    vline = 8'b10000101;


  end


  3'b001:

  begin

    hline = 8'b10111111;

    vline = 8'b10000101;


  end

  3'b010:

  begin

    hline = 8'b11011111;

    vline = 8'b10000101;

  end

  3'b011:

  begin

    hline = 8'b11101111;

    vline = 8'b10000101;


  end

  3'b100:

  begin

    hline = 8'b11110111;
    vline = 8'b10000101;
  end
  3'b101:
  begin
    hline = 8'b11111011;
    vline = 8'b10000101;
  end
  3'b110:
  begin
    hline = 8'b11111101;
    vline = 8'b10000101;

  end
  3'b111:
  begin
    hline = 8'b11111111;
    vline = 8'b00000000;

  end
endcase
```

I will attach complete code in appendix, reader can get it from there.

4 Questions

4.1 Discuss the concept and advantages of hierarchical design.

Hierarchy in circuit design refers to the creation of higher level cells based on lower lever cells. For example, if you create a cell that is composed of inverters and nand gates, you can instantiate (add) your previously designed inverter and nand cells rather than recreating them in the higher level cell.

4.2 How was hierarchical design utilized in this lab?

For example, in Candence part, we use the and, or, not gate which have already be designed in past lab, and that's is so convenient, that we don't need to consider the transistor level anymore, just use it. Also, the 3 to 8 decoder, we design it before we create our final circuit, so we can just use the block, and don't need to consider the circuit layout.

5 Conclusion

Now we can get the conclusion, I finish this lab successfully, and achieve all the objective. I have learned two important knowledge during this lab. First, there are some crystal oscillators in each board, we can use them to generate any delay time length we want. Second, do not connect the component to the voltage source without measuring the voltage level, since I burned my 8*8 LED matrix because connect to FPGA board without resistor.

6 Appendix

```
module Lab4(run,reset,clk,vline,hline);


input wire clk;

input run;

input reset;

//input wire run;

//input wire reset;


output reg [7:0] vline;

output reg [7:0] hline;


reg [12:0] row_counter;

reg  clk_counter;
reg [3:0] i = 3’b000;

reg [12:0] z;

reg [1:0] frame = 2’b00;


always @(posedge clk)

begin

   if(row_counter ==0 )

   begin

      clk_counter <= ~clk_counter;

   end

   row_counter <= row_counter +1;

end




always @(posedge clk_counter)

begin




   case(frame)

   2’b01:

   begin

      case(i)

      3’b000:

      begin

         hline = 8’b01111111;

         vline = 8’b10000101;
```

```
end


3'b001:

begin

   hline = 8'b10111111;

   vline = 8'b10000101;


end

3'b010:

begin

   hline = 8'b11011111;

   vline = 8'b10000101;

end

3'b011:

begin

   hline = 8'b11101111;

   vline = 8'b10000101;


end

3'b100:

begin

   hline = 8'b11110111;

   vline = 8'b10000101;

end

3'b101:

begin

   hline = 8'b11111011;

   vline = 8'b10000101;
   end
   3'b110:
   begin
      hline = 8'b11111101;
      vline = 8'b10000101;

   end
   3'b111:
   begin
      hline = 8'b11111111;
      vline = 8'b00000000;

   end
   endcase
end

2'b10:
begin
   case(i)
   3'b000:
   begin
      hline = 8'b01111111;
      vline = 8'b11110101;

   end

   3'b001:
   begin
      hline = 8'b10111111;

      vline = 8'b10000101;


   end

   3'b010:

   begin

      hline = 8'b11011111;

      vline = 8'b10000101;

   end

   3'b011:

   begin

      hline = 8'b11101111;

      vline = 8'b11110101;


   end

   3'b100:

   begin

      hline = 8'b11110111;

      vline = 8'b00010101;

   end

   3'b101:

   begin

      hline = 8'b11111011;

      vline = 8'b00010101;


   end

   3'b110:

   begin

      hline = 8'b11111101;
```

```
      vline = 8'b11110101;


   end
   3'b111:
   begin
      hline = 8'b11111111;
      vline = 8'b00000000;


   end
   endcase
end


2'b11:
begin
   case(i)
   3'b000:
   begin
      hline = 8'b01111111;
      vline = 8'b11110101;


   end


   3'b001:
   begin
      hline = 8'b10111111;
      vline = 8'b10000101;


   end
   3'b010:
```

```
begin
   hline = 8'b11011111;
   vline = 8'b10000101;
end
3'b011:
begin
   hline = 8'b11101111;
   vline = 8'b11110101;

end
3'b100:
begin
   hline = 8'b11110111;
   vline = 8'b10000101;
end
3'b101:
begin
   hline = 8'b11111011;
   vline = 8'b10000101;

end
3'b110:
begin
   hline = 8'b11111101;
   vline = 8'b11110101;

end
3'b111:
begin
```

```
      hline = 8'b11111111;

      vline = 8'b10000101;


   end

   endcase

end


2'b00:

begin

   case(i)

   3'b000:

   begin

      hline = 8'b01111111;

      vline = 8'b10010101;


   end


   3'b001:

   begin

      hline = 8'b10111111;

      vline = 8'b10010101;


   end

   3'b010:

   begin

      hline = 8'b11011111;

      vline = 8'b10010101;

   end

   3'b011:
```

```
begin
   hline = 8'b11101111;
   vline = 8'b11110101;

end
3'b100:
begin
   hline = 8'b11110111;
   vline = 8'b10000101;
end
3'b101:
begin
   hline = 8'b11111011;
   vline = 8'b10000101;

end
3'b110:
begin
   hline = 8'b11111101;
   vline = 8'b10000101;

end
3'b111:
begin
   hline = 8'b11111111;
   vline = 8'b00000000;

end
endcase
```

```
   end



   endcase
   i = i+3'b001;
   if(run)
   begin
      z <= z + 1;
   end
   if(z == 0)
   begin
      frame <= frame+1;
   end
if(reset)
begin
   frame <= 2'b01;
end

end

endmodule
```