DIGITAL CLOCK REPORT

EE23BTECH11208(K.MANOHAR) EE233BTECH11215(P SRIKAR VARMA)

INTRODUCTION:

the project given is to implement a digital clock on vaman FPGA a board using verilog language.

Apparatus:
Vaman board,16 x 2 LCD Display, 10k Potentiometer,Breadboard,Jumper wires

for the Vaman Board refer to ‘https://github.com/gadepall/vaman/tree/master’

- the breadboard is used for making connections from LCD Display to potentiometer to Vaman board.

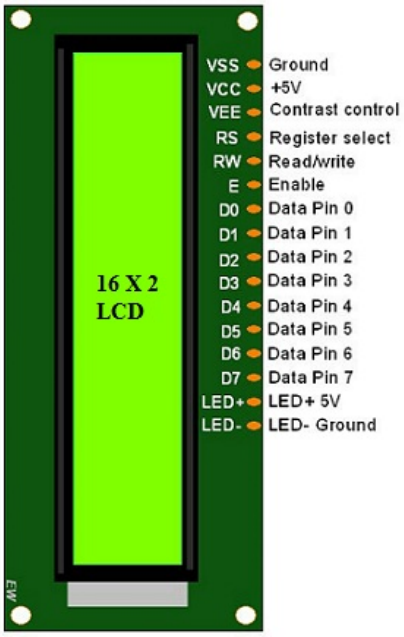
LCD Display(LCD 16×2) (JHD 162A):

- the LCD display has 16 pin and each of it has a specific functionality.

Pin1 (Ground/Source Pin): This is a GND pin of display, used to connect the GND terminal of the microcontroller unit or power source.
Pin2 (VCC/Source Pin): This is the voltage supply pin of the display, used to connect the supply pin of the power source.
Pin3 (V0/VEE/Control Pin): This pin regulates the difference of the display, used to connect a changeable POT that can supply 0 to 5V.
Pin4 (Register Select/Control Pin): This pin toggles among command or data register, used to connect a microcontroller unit pin and obtains either 0 or 1(0 = data mode, and 1 = command mode).
Pin5 (Read/Write/Control Pin): This pin toggles the display among the read or writes operation, and it is connected to a microcontroller unit pin to get either 0 or 1 (0 = Write Operation, and 1 = Read Operation).
Pin 6 (Enable/Control Pin): This pin should be held high to execute Read/Write process, and it is connected to the microcontroller unit & constantly held high.
Pins 7-14 (Data Pins): These pins are used to send data to the display. These pins are connected in two-wire modes like 4-wire mode and 8-wire mode. In 4-wire mode, only four pins are connected to the microcontroller unit like 0 to 3, whereas in 8-wire mode, 8-pins are connected to microcontroller unit like 0 to 7.
Pin15 (+ve pin of the LED): This pin is connected to +5V

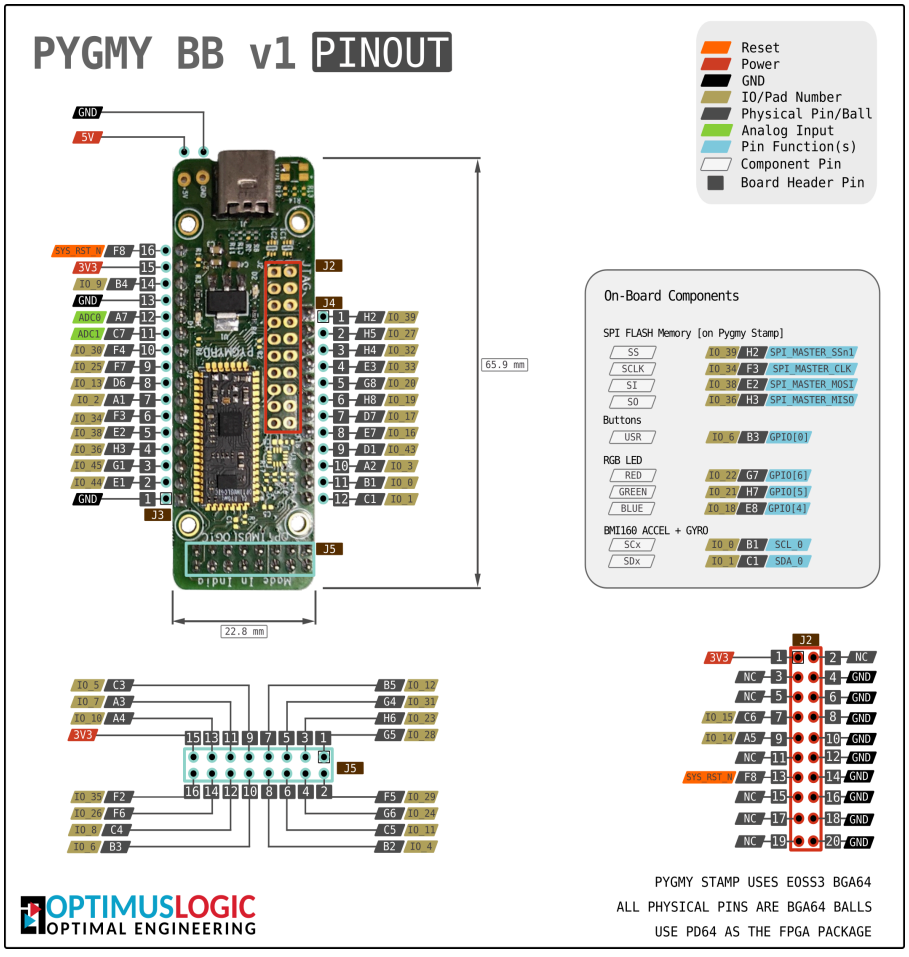
Pin 16 (-ve pin of the LED): This pin is connected to GND.

LCD



Display

Vaman Board Pins



PD64		
IO Locatio	Alias	IO Type
B1	IO_0	BIDIR
C1	IO_1	BIDIR
A1	IO_2	BIDIR
A2	IO_3	BIDIR
B2	IO_4	BIDIR
C3	IO_5	BIDIR
B3	IO_6	BIDIR
A3	IO_7	BIDIR/CLOCK
C4	IO_8	BIDIR/CLOCK
B4	IO_9	BIDIR
A4	IO_10	BIDIR
C5	IO_11	BIDIR
B5	IO_12	BIDIR
D6	IO_13	BIDIR
A5	IO_14	BIDIR
C6	IO_15	BIDIR
E7	IO_16	BIDIR
D7	IO_17	BIDIR
E8	IO_18	BIDIR
H8	IO_19	BIDIR
G8	IO_20	BIDIR
H7	IO_21	BIDIR
G7	IO_22	BIDIR/CLOCK
H6	IO_23	BIDIR/CLOCK
G6	IO_24	BIDIR/CLOCK
F7	IO_25	BIDIR
F6	IO_26	BIDIR
H5	IO_27	BIDIR
G5	IO_28	BIDIR
F5	IO_29	BIDIR
F4	IO_30	BIDIR
G4	IO_31	BIDIR
H4	IO_32	SDIOMUX
E3	IO_33	SDIOMUX
F3	IO_34	SDIOMUX
F2	IO_35	SDIOMUX
H3	IO_36	SDIOMUX
G2	IO_37	SDIOMUX
E2	IO_38	SDIOMUX
H2	IO_39	SDIOMUX
D2	IO_40	SDIOMUX
F1	IO_41	SDIOMUX
H1	IO_42	SDIOMUX
D1	IO_43	SDIOMUX
E1	IO_44	SDIOMUX
G1	IO_45	SDIOMUX

PU64		
IO Locatio	Alias	IO type
4	IO_0	BIDIR
5	IO_1	BIDIR
6	IO_2	BIDIR
2	IO_3	BIDIR
3	IO_4	BIDIR
64	IO_5	BIDIR
62	IO_6	BIDIR
63	IO_7	BIDIR/CLOCK
61	IO_8	BIDIR/CLOCK
60	IO_9	BIDIR
59	IO_10	BIDIR
57	IO_11	BIDIR
56	IO_12	BIDIR
55	IO_13	BIDIR
54	IO_14	BIDIR
53	IO_15	BIDIR
40	IO_16	BIDIR
42	IO_17	BIDIR
38	IO_18	BIDIR
36	IO_19	BIDIR
37	IO_20	BIDIR
39	IO_21	BIDIR
34	IO_22	BIDIR/CLOCK
33	IO_23	BIDIR/CLOCK
32	IO_24	BIDIR/CLOCK
31	IO_25	BIDIR
30	IO_26	BIDIR
28	IO_27	BIDIR
27	IO_28	BIDIR
26	IO_29	BIDIR
25	IO_30	BIDIR
23	IO_31	BIDIR
22	IO_32	SDIOMUX
21	IO_33	SDIOMUX
20	IO_34	SDIOMUX
18	IO_35	SDIOMUX
17	IO_36	SDIOMUX
15	IO_37	SDIOMUX
16	IO_38	SDIOMUX
11	IO_39	SDIOMUX
13	IO_40	SDIOMUX
14	IO_41	SDIOMUX
10	IO_42	SDIOMUX
7	IO_43	SDIOMUX
8	IO_44	SDIOMUX
9	IO_45	SDIOMUX

WR42		
IO Locatio	Alias	IO Type
A7	IO_0	BIDIR
B7	IO_1	BIDIR
C7	IO_3	BIDIR
A6	IO_6	BIDIR
B6	IO_8	BIDIR/CLOCK
A5	IO_9	BIDIR
B5	IO_10	BIDIR
A4	IO_14	BIDIR
B4	IO_15	BIDIR
E1	IO_16	BIDIR
D1	IO_17	BIDIR
C1	IO_19	BIDIR
F2	IO_20	BIDIR
E2	IO_23	BIDIR/CLOCK
D2	IO_24	BIDIR/CLOCK
D3	IO_25	BIDIR
F3	IO_28	BIDIR
E3	IO_29	BIDIR
F4	IO_30	BIDIR
E4	IO_31	BIDIR
D5	IO_34	SDIOMUX
F5	IO_36	SDIOMUX
E6	IO_38	SDIOMUX
F6	IO_39	SDIOMUX
D7	IO_43	SDIOMUX
E7	IO_44	SDIOMUX
F7	IO_45	SDIOMUX

PU64 Pins Data

PCF file:

```
set_io data(7) IO_30
set_io data(6) IO_29
set_io data(5) IO_28
set_io data(4) IO_27
set_io en IO_26
set_io rs IO_25
```

verilog code:

```
module co7(output reg rs, output reg en, output reg [7:4] data);
reg [7:0] one_sec,min9,hour9;
reg [7:0] ten_sec,min5,hour2;module co7(output reg rs, output reg en, output
reg [7:4] data);
reg [7:0] one_sec,min9,hour9;
reg [7:0] ten_sec,min5,hour2;
reg [7:0] ten_sec_,min5_,hour2_,one_sec_,min9_,hour9_;
// up counter
integer i=1;
integer count=0;
reg [3:0] nibs [1:51];
//reg [3:0] delay;//delay should go up till 10
reg [26:0] delay;
integer x=0;
initial begin
delay = 8'd48;

end
wire clk;
qlal4s3b_cell_macro u_qlal4s3b_cell_macro (
.Sys_Clk0 (clk),
);

always@(posedge clk) begin
  if(delay<12000000)
   delay<=delay+1;

else
begin
   delay<=27'b0000;

   if (x==0)begin
```

```
one_sec = 8'd0;
min9 = 8'd0;
hour9 = 8'd0;
ten_sec = 8'd0;
min5 = 8'd0;
hour2 = 8'd2;
ten_sec_=8'd48;
min5_=8'd48;
hour2_=8'd48;
one_sec_=8'd48;
min9_=8'd48;
hour9_=8'd48;
x=0;

end
else if (x==0)
begin
if (ten_sec==8'd5 && one_sec==8'd9 && min5==8'd5 && min9==8'd9 &&
hour9==8'd3 && hour2==8'd2) begin
   ten_sec<=8'd0;
   one_sec<=8'd0;
   min9<=8'd0;
   min5<=8'd0;
   hour9<=8'd0;
   hour2<=8'd0;
  end
  else if (ten_sec==8'd5 && one_sec==8'd9 && min5==8'd5 && min9==8'd9
&& hour9==8'd9) begin
   ten_sec<=8'd0;
   one_sec<=8'd0;
   min9<=8'd0;
   min5<=8'd0;
   hour9<=8'd0;
   hour2<= hour2+8'd1;
  end
  else if (ten_sec==8'd5 && one_sec==8'd9 && min5==8'd5 && min9==8'd9)
begin
   ten_sec<=8'd0;
   one_sec<=8'd0;
   min9<=8'd0;
   min5<=8'd0;
   hour9<=hour9+8'd1;
   end
```

```
  else if (ten_sec==8'd5 && one_sec==8'd9 && min9==8'd9 ) begin
   ten_sec<=8'd0;
   one_sec<=8'd0;
   min9<=8'd0;
   min5 <= min5 + 8'd1;
  end
  else if (ten_sec==8'd5 && one_sec==8'd9) begin
   ten_sec<=8'd0;
   one_sec<=8'd0;
   min9 <= min9 + 8'd1;
  end
 else if(one_sec==8'd9) begin
  one_sec<=8'd0;
  ten_sec<=ten_sec+3'd1;
  end

else if(one_sec<8'd9) begin

one_sec <= one_sec + 8'd1;

 end
end
end
end

parameter u=8'd0;


always @(posedge clk) begin
ten_sec_= ten_sec + 8'd48;
one_sec_ = one_sec + 8'd48;
min5_ = min5 + 8'd48;
hour2_ = hour2 + 8'd48;
min9_ = min9 + 8'd48;
hour9_ = hour9 + 8'd48 ;

u = hour2 * 10 + hour9 ;

if (u>=8'd12 && u < 8'd24) begin
   nibs[34]=4'h5;
   nibs[35]=4'h0;
   nibs[36]=4'h4;
```

```
   nibs[37]=4'hD;
    end
  else if (u<8'd12 && u>=8'd0) begin
   nibs[34]=4'h4;
   nibs[35]=4'h1;
   nibs[36]=4'h4;
   nibs[37]=4'hD;
end

nibs[1]=4'h3;
nibs[2]=4'h3;
nibs[3]=4'h3;
nibs[4]=4'h2;
nibs[5]=4'h2;
nibs[6]=4'h8;
nibs[7]=4'h0;
nibs[8]=4'hC;
nibs[9]=4'h0;
nibs[10]=4'h6;
nibs[11]=4'h0;
nibs[12]=4'h1;
nibs[13]=4'h8;
nibs[14]=4'h0;
nibs[15]=1'b1;
nibs[16]=hour2_[7:4];//hours
nibs[17]=hour2_[3:0];
nibs[18]=hour9_[7:4];
nibs[19]=hour9_[3:0];
nibs[20]=4'h3;
nibs[21]=4'hA;
nibs[22]=min5_[7:4];//mins
nibs[23]=min5_[3:0];
nibs[24]=min9_[7:4];
nibs[25]=min9_[3:0];
nibs[26]=4'h3;
nibs[27]=4'hA;
nibs[28]=ten_sec_[7:4];//sec
nibs[29]=ten_sec_[3:0];
nibs[30]=one_sec_[7:4];
nibs[31]=one_sec_[3:0];
nibs[32]=4'h2;
nibs[33]=4'h0;
nibs[38]=4'hC;
```

```
nibs[39]=4'h0;
nibs[40]=4'h2;
nibs[41]=4'h0;
nibs[42]=4'h2;
nibs[43]=4'h0;
nibs[44]=4'h2;
nibs[45]=4'h0;
nibs[46]=4'h2;
nibs[47]=4'h0;
nibs[48]=4'h2;
nibs[49]=4'h0;
nibs[50]=4'h2;
nibs[51]=4'h0;


if(i<15)
 begin
    rs<=1'b0;
    data=nibs[i];
    en<=1'b1;
    if(count == 800)
    begin
       en<=1'b0;
       count<=0;
       i<=i+1;
    end
    else
       count<=count+1;
end
    if(i==15)
    begin
       if(count==60000)
       begin
         count<=8'd0;
         i<=i+1;
       end
         else
         count<=count+1;
    end

    if((i>15 && i<38)||(i>39 && i<=51))
    begin
     rs<=1'b1;
```

```
    data=nibs[i];
    en<=1'b1;
    if(count==800)
    begin
      en<=1'b0;
      count<=8'd0;
      i<=i+1;
    end
    else
      count<=count+1;
    end
if(i>=38 && i<=39)
   begin
      rs<=1'b0;
      data = nibs[i];
      en<=1'b1;
      if(count==800)
      begin
        en<=1'b0;
        count<=8'd0;
        i<=i+1;
      end
      else
        count<=count+1;
      end
      if(i>51)
         i<=13;
end
endmodule



// code for 12 hour clock

/*
else if(button==0)//12hours begin
delay<=27'b0000;


if (ten_sec==8'd5 && one_sec==8'd9 && min5==8'd5 && min9==8'd9 &&
hour9>=8'd1 && hour2>=8'd1) begin//resetting clock to 00:00:00 after
23:59:59
  ten_sec<=8'd0;
```

```
   one_sec<=8'd0;
   min9<=8'd0;
   min5<=8'd0;
   hour9<=8'd0;
   hour2.<=8'd0;
  end
  else if (ten_sec==8'd5 && one_sec==8'd9 && min5==8'd5 && min9==8'd9
&& hour9==8'd9) begin //resetting minutes and sec
   ten_sec<=8'd0;
   one_sec<=8'd0;
   min9<=8'd0;
   min5<=8'd0;
   hour9.<=8'd0;
   hour2.<= hour2.+8'd1;
  end
  else if (ten_sec==8'd5 && one_sec==8'd9 && min5==8'd5 && min9==8'd9)
begin
   ten_sec<=8'd0;
   one_sec<=8'd0;
   min9<=8'd0;
   min5<=8'd0;
   hour9.<=hour9.+8'd1;
   end

  else if (ten_sec==8'd5 && one_sec==8'd9 && min9==8'd9 ) begin
   ten_sec<=8'd0;
   one_sec<=8'd0;
   min9<=8'd0;
   min5 <= min5 + 8'd1;
  end
  else if (ten_sec==8'd5 && one_sec==8'd9) begin
   ten_sec<=8'd0;
   one_sec<=8'd0;
   min9 <= min9 + 8'd1;
  end
 else if(one_sec==8'd9) begin
  one_sec<=8'd0;
  ten_sec<=ten_sec+3'd1;
  end

else if(one_sec<8'd9) begin

one_sec <= one_sec + 8'd1;
```

```
 end
end


if (hour2 <=8'd1 && hour9 < 8'd2) begin
part1=4'h4;
part2=4'h1;
end
else if (hour2 >= 8'd1 && hour9 >= 8'd2 )begin
part1=4'h7;
part2=4'h0;
end



end

end  //end for always */



/*
module alarmHex(clk, rst, en_in, time_in, time_set_in, set_time, ring,
end_ring);
  input clk, rst, set_time, end_ring, en_in;
  output reg ring;
  input [10:0] time_set_in, time_in;
  reg [10:0] time_alarm;

  reg en;

  //set alarm time
  always@(posedge clk or posedge rst)
    begin
      if(rst)
      begin
        time_alarm <= 11'd0;
      end
      else
      begin
       time_alarm <= (set_time) ? time_set_in : time_alarm;
      end
```

```
    end

  //handle to ringing of the alarm
  always@(posedge clk or posedge rst)
    begin
    if(rst)
        begin
       ring <= 1'b0;
        end
   else
        begin
        if(en)
         begin
              //while ringing: stop if  end ring pressed
              //otherwise start ringing when time is equal to snooze
              ring <= (ring) ? (~end_ring) : (time_alarm == time_in);
            end
        end
    end

  //keep ringing shut after end_ring if high, but not disable for next day
  always@(posedge clk)
    begin
      if(time_alarm == time_in)
        begin
          en <= (end_ring) ? 1'b0 : en;
        end
      else
        begin
          en <= en_in;
        end
    end
endmodule//alarm
*/
reg [7:0] ten_sec_,min5_,hour2_,one_sec_,min9_,hour9_;
// up counter
integer i=1;
integer count=0;
reg [3:0] nibs [1:51];
//reg [3:0] delay;//delay should go up till 10
reg [26:0] delay;
integer x=0;
initial begin
```

```
delay = 8'd48;

end
wire clk;
qlal4s3b_cell_macro u_qlal4s3b_cell_macro (
.Sys_Clk0 (clk),
);

always@(posedge clk) begin
  if(delay<12000000)
   delay<=delay+1;

else
begin
   delay<=27'b0000;

   if (x==0)begin
one_sec = 8'd0;
min9 = 8'd0;
hour9 = 8'd0;
ten_sec = 8'd0;
min5 = 8'd0;
hour2 = 8'd2;
ten_sec_=8'd48;
min5_=8'd48;
hour2_=8'd48;
one_sec_=8'd48;
min9_=8'd48;
hour9_=8'd48;
x=0;

end
else if (x==0)
begin
if (ten_sec==8'd5 && one_sec==8'd9 && min5==8'd5 && min9==8'd9 &&
hour9==8'd3 && hour2==8'd2) begin
   ten_sec<=8'd0;
   one_sec<=8'd0;
   min9<=8'd0;
   min5<=8'd0;
   hour9<=8'd0;
   hour2<=8'd0;
  end
```

```
  else if (ten_sec==8'd5 && one_sec==8'd9 && min5==8'd5 && min9==8'd9
&& hour9==8'd9) begin
   ten_sec<=8'd0;
   one_sec<=8'd0;
   min9<=8'd0;
   min5<=8'd0;
   hour9<=8'd0;
   hour2<= hour2+8'd1;
  end
  else if (ten_sec==8'd5 && one_sec==8'd9 && min5==8'd5 && min9==8'd9)
begin
   ten_sec<=8'd0;
   one_sec<=8'd0;
   min9<=8'd0;
   min5<=8'd0;
   hour9<=hour9+8'd1;
   end

  else if (ten_sec==8'd5 && one_sec==8'd9 && min9==8'd9 ) begin
   ten_sec<=8'd0;
   one_sec<=8'd0;
   min9<=8'd0;
   min5 <= min5 + 8'd1;
  end
  else if (ten_sec==8'd5 && one_sec==8'd9) begin
   ten_sec<=8'd0;
   one_sec<=8'd0;
   min9 <= min9 + 8'd1;
  end
 else if(one_sec==8'd9) begin
  one_sec<=8'd0;
  ten_sec<=ten_sec+3'd1;
  end

else if(one_sec<8'd9) begin

one_sec <= one_sec + 8'd1;

 end
end
end
end
```

```
parameter u=8'd0;

always @(posedge clk) begin
ten_sec_= ten_sec + 8'd48;
one_sec_ = one_sec + 8'd48;
min5_ = min5 + 8'd48;
hour2_ = hour2 + 8'd48;
min9_ = min9 + 8'd48;
hour9_ = hour9 + 8'd48 ;

u = hour2 * 10 + hour9 ;

if (u>=8'd12 && u < 8'd24) begin
   nibs[34]=4'h5;
   nibs[35]=4'h0;
   nibs[36]=4'h4;
   nibs[37]=4'hD;
    end
  else if (u<8'd12 && u>=8'd0) begin
   nibs[34]=4'h4;
   nibs[35]=4'h1;
   nibs[36]=4'h4;
   nibs[37]=4'hD;
end

nibs[1]=4'h3;
nibs[2]=4'h3;
nibs[3]=4'h3;
nibs[4]=4'h2;
nibs[5]=4'h2;
nibs[6]=4'h8;
nibs[7]=4'h0;
nibs[8]=4'hC;
nibs[9]=4'h0;
nibs[10]=4'h6;
nibs[11]=4'h0;
nibs[12]=4'h1;
nibs[13]=4'h8;
nibs[14]=4'h0;
nibs[15]=1'b1;
nibs[16]=hour2_[7:4];//hours
nibs[17]=hour2_[3:0];
nibs[18]=hour9_[7:4];
```

```
nibs[19]=hour9_[3:0];
nibs[20]=4'h3;
nibs[21]=4'hA;
nibs[22]=min5_[7:4];//mins
nibs[23]=min5_[3:0];
nibs[24]=min9_[7:4];
nibs[25]=min9_[3:0];
nibs[26]=4'h3;
nibs[27]=4'hA;
nibs[28]=ten_sec_[7:4];//sec
nibs[29]=ten_sec_[3:0];
nibs[30]=one_sec_[7:4];
nibs[31]=one_sec_[3:0];
nibs[32]=4'h2;
nibs[33]=4'h0;
nibs[38]=4'hC;
nibs[39]=4'h0;
nibs[40]=4'h2;
nibs[41]=4'h0;
nibs[42]=4'h2;
nibs[43]=4'h0;
nibs[44]=4'h2;
nibs[45]=4'h0;
nibs[46]=4'h2;
nibs[47]=4'h0;
nibs[48]=4'h2;
nibs[49]=4'h0;
nibs[50]=4'h2;
nibs[51]=4'h0;


if(i<15)
 begin
    rs<=1'b0;
    data=nibs[i];
    en<=1'b1;
    if(count == 800)
    begin
       en<=1'b0;
       count<=0;
       i<=i+1;
    end
    else
```

```
      count<=count+1;
end
   if(i==15)
   begin
     if(count==60000)
     begin
       count<=8'd0;
       i<=i+1;
     end
       else
       count<=count+1;
   end

   if((i>15 && i<38)||(i>39 && i<=51))
   begin
    rs<=1'b1;
    data=nibs[i];
    en<=1'b1;
    if(count==800)
    begin
      en<=1'b0;
      count<=8'd0;
      i<=i+1;
    end
    else
      count<=count+1;
    end
if(i>=38 && i<=39)
   begin
     rs<=1'b0;
     data = nibs[i];
     en<=1'b1;
     if(count==800)
     begin
       en<=1'b0;
       count<=8'd0;
       i<=i+1;
     end
     else
       count<=count+1;
     end
     if(i>51)
        i<=13;
```

```
end
endmodule



// code for 12 hour clock

/*
else if(button==0)//12hours begin
delay<=27'b0000;


if (ten_sec==8'd5 && one_sec==8'd9 && min5==8'd5 && min9==8'd9 &&
hour9>=8'd1 && hour2>=8'd1) begin//resetting clock to 00:00:00 after
23:59:59
   ten_sec<=8'd0;
   one_sec<=8'd0;
   min9<=8'd0;
   min5<=8'd0;
   hour9<=8'd0;
   hour2.<=8'd0;
  end
  else if (ten_sec==8'd5 && one_sec==8'd9 && min5==8'd5 && min9==8'd9
&& hour9==8'd9) begin //resetting minutes and sec
   ten_sec<=8'd0;
   one_sec<=8'd0;
   min9<=8'd0;
   min5<=8'd0;
   hour9.<=8'd0;
   hour2.<= hour2.+8'd1;
  end
  else if (ten_sec==8'd5 && one_sec==8'd9 && min5==8'd5 && min9==8'd9)
begin
   ten_sec<=8'd0;
   one_sec<=8'd0;
   min9<=8'd0;
   min5<=8'd0;
   hour9.<=hour9.+8'd1;
   end

  else if (ten_sec==8'd5 && one_sec==8'd9 && min9==8'd9 ) begin
   ten_sec<=8'd0;
   one_sec<=8'd0;
```

```
   min9<=8'd0;
   min5 <= min5 + 8'd1;
  end
  else if (ten_sec==8'd5 && one_sec==8'd9) begin
   ten_sec<=8'd0;
   one_sec<=8'd0;
   min9 <= min9 + 8'd1;
  end
 else if(one_sec==8'd9) begin
  one_sec<=8'd0;
  ten_sec<=ten_sec+3'd1;
  end

else if(one_sec<8'd9) begin

one_sec <= one_sec + 8'd1;

 end
end


if (hour2 <=8'd1 && hour9 < 8'd2) begin
part1=4'h4;
part2=4'h1;
end
else if (hour2 >= 8'd1 && hour9 >= 8'd2 )begin
part1=4'h7;
part2=4'h0;
end



end

end  //end for always */




/*
module alarmHex(clk, rst, en_in, time_in, time_set_in, set_time, ring,
end_ring);
  input clk, rst, set_time, end_ring, en_in;
```

```
output reg ring;
input [10:0] time_set_in, time_in;
reg [10:0] time_alarm;

reg en;

//set alarm time
always@(posedge clk or posedge rst)
  begin
    if(rst)
    begin
      time_alarm <= 11'd0;
    end
    else
    begin
     time_alarm <= (set_time) ? time_set_in : time_alarm;
    end
  end

//handle to ringing of the alarm
always@(posedge clk or posedge rst)
  begin
  if(rst)
      begin
     ring <= 1'b0;
      end
 else
      begin
      if(en)
       begin
            //while ringing: stop if  end ring pressed
            //otherwise start ringing when time is equal to snooze
            ring <= (ring) ? (~end_ring) : (time_alarm == time_in);
          end
      end
  end

//keep ringing shut after end_ring if high, but not disable for next day
always@(posedge clk)
  begin
    if(time_alarm == time_in)
      begin
        en <= (end_ring) ? 1'b0 : en;
```

```
        end
      else
        begin
          en <= en_in;
        end
    end
endmodule//alarm
*/
```

explanation:

the inputs and outputs are assigned in the pcf file according to the top module.
There is an inbuit clock in Vaman Board
qlal4s3b_cell_macro u_qlal4s3b_cell_macro (.Sys_Clk0 (clk),)
in the first few lines we defined all parameters , variable and integers req for required to generate a clock.
{ten_sec,min5,hour2}represents ten_sec is the tenths place of second,min5 is tenths place of minutes,hour2 is the tenths place of hour.
{ one_sec,min9,hour9}these are the units place of second , minute and hour .
We then introduce an integer x = 0 , we used it to intitiate the units and tenth's values of sec , min ,hour . By creating an if statement in side the first always block.
And when they are intialised after that x value is made 1 so that the same 'if' block doesnt get excetued many .
When the value of x=1,the main code stars.
In the first if block we reset the time when it reaches 23:59:59.
in the first else if block we reset the time when unit place of hour reaches 9 and also add '+1' to tenths place of hour 'and' reset when min is 59 'and' reset when sec is 59.
in the second if else block min 'and ' sec are reset as they reach 59.
inthe third if else block the sec resets as it reaches 59 and the thenths pplace of min gets increment of +1 as units place reaches 9.
in the fourth if else block the sec get reset as it reaches 59.
in the fifth if else block the seconds get reset when the units digit reaches 9 and increments tenths place by +1.
In the next always block we try to display the clock with AM / PM.
We uised a parameter u to find wether the hours part is greater than or equal to 12 of less then 12 to show AM /PM.
We took help of Rajashekar sir to implement the LCD display and by using the hex value table we assigned the values of hours,min and sec to respective cells in a LCD display.
We then made an if statement depending on u wether to print AM,PM.
We then have some If else Statements which are used to assign the values of the hours , min ,sec to data [7: 4] .
at last we also included our 12 hour clock code , since we dont how to get input by giving an port into FPGA , we tried many combinations of ports and ran several bin file creations but was always not giving the expected LCD display.

This part of 12 hour clock was working when seperated from the 24 hour clock.
Another code for alarm was written but not implimented as we were facing issues due to LCD display and changing the modes in clock.

Flashing the code into Vaman FPGA:

the code used to generate a bin file is

“ ql_symbiflow -compile -d ql-eos-s3 -P PU64 -v clovk245.v -t co7 -p .pcf -dump binary ”

the code used to send this bin file into Vaman FPGA
“ python3 TinyFPGA-Programmer-Application/tinyfpga-programmer-gui.py --port /dev/ttyACM0 --appfpga co7.bin --mode fpga –reset
”

Conclusion:
In conclusion, the Vaman FPGA board's digital clock has been successfully implemented using Verilog, proving the usefulness of digital design principles in actual embedded systems.

The link for codes and video:

https://github.com/Manohark25/clock25 -------codes
https://drive.google.com/file/d/1FyeQV09TPwFVnOXhEq9D8zPGMOAZiwPi/view?usp=sharing ------video