MULTIPLEXED STOPWATCH

GROUP 11:

SURESH : 121601013
SAURABH : 121601034
SUCHITH : 121601016

ABSTRACT:

The goal of this mini-project is to get insight view on how complex designs are realized in industry and acquire knowledge about the design process.

In this project, we aimed at implementing a STOPWATCH with the help of FPGA (ZYBO board) and HDL(Verilog).

CONTRIBUTION:

SURESH : REPORT + CODING + HARDWARE IMPLEMENTATION

SAURABH: REPORT + CODING + HARDWARE IMPLEMENTATION

SUCHITH: REPORT + CODING + HARDWARE IMPLEMENTATION

DESIGN SPECIFICATIONS:

The time format for our Stopwatch M:SS
(i.e Minutes:Seconds)

For displaying the time, we are using 3 seven segment displays.

As an interface to the user, the stopwatch includes:

- START/STOP push button : To stop or resume the time.
- RESET push button: To reset the counter to 0:00.
- LAP push button: To record the count at different time instants.

When powering up the device the display should show 0.00, where the digits before the decimal points are minutes and after are tens and units of a second.

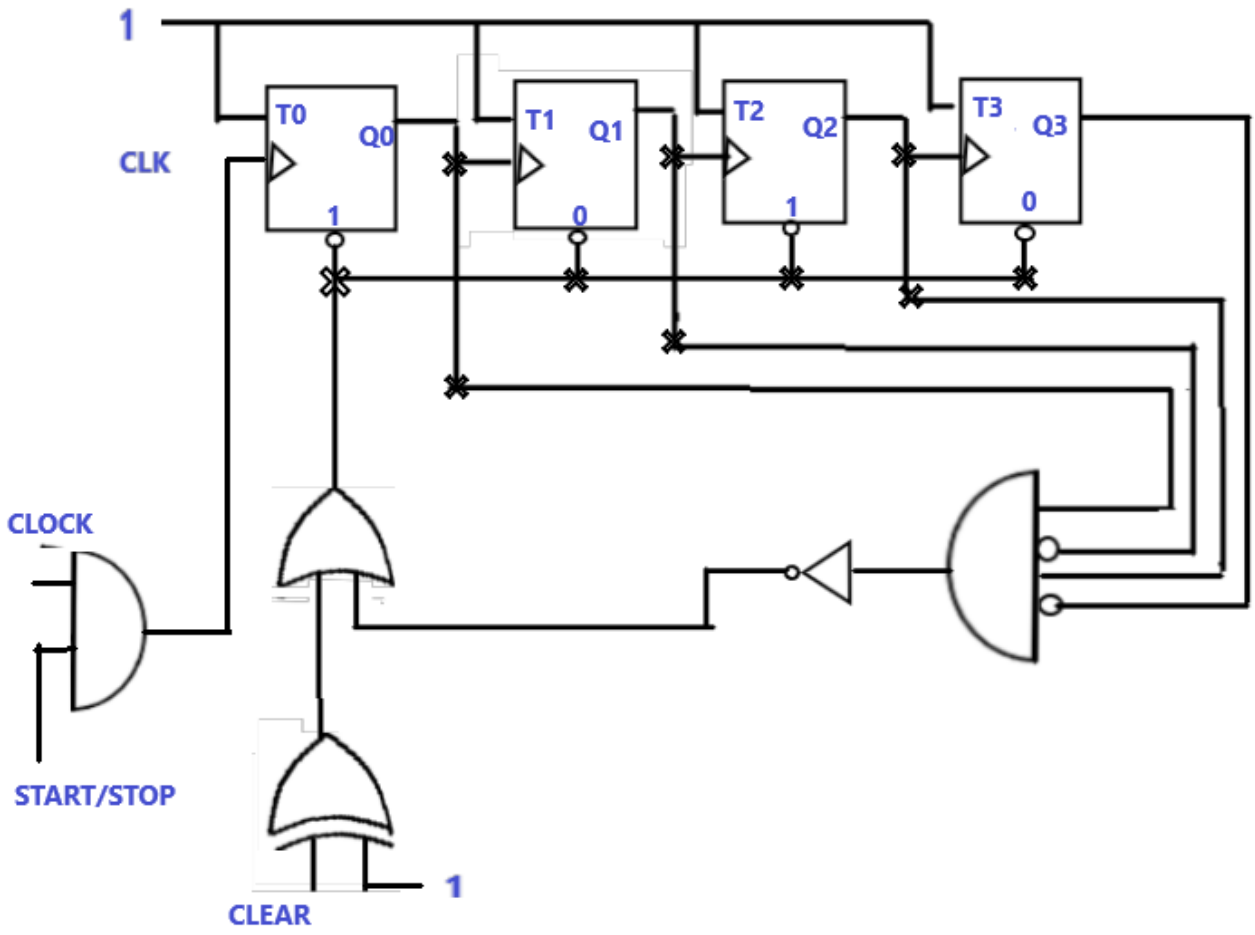
This means that the maximum count of our stop watch is 9.59 minutes. If the stop watch reaches this value it should wrap and continue counting from zero.

In the initial state the start button starts the stop watch and the current value is displayed on the display whereas the stop button is not supposed to have any function.While the stop watch is running a press of the stop button pauses the stop watch. A press of reset button resets the counter to 0. Pressing the Lap push button records and displays the time at that instant on another set of 3 seven segment displays.

Multiplexing of the seven segment displays was done so as to reduce the number of connections on breadboard.

INTRODUCTION:

The block diagram is given below:



Implementation and Results:

Logic :

1.MULTIPLEXING of seven segment displays:
The seven segment LED circuit uses seven different and individual LED's to display a hexadecimal symbol. It has 7 wires to control the individual LED's one wire to control the decimal point and one enable wire.

To reduce the number of wires a multiplexing circuit is used to control the display. Using the multiplexing circuit the number of wires required to light up all 3 displays are reduced from 21 to 10 (7 data bits and 3 enable bits).

The multiplexing circuit can take 3 inputs and have only one output. But the inputs should be displayed on the output fast enough to fool the viewer into thinking all outputs are enabled individually and simultaneously. This is achieved by having an enable signal that changes so fast that it appears that all displays are on simultaneously. The refreshing rate of the enable signal should be around 1000Hz to achieve this desired effect.

CODE FOR MULTIPLEXING:

For the purpose of multiplexing, we used a 1000 Hz clock.

```
always@(posedge clock)

begin

   if(counter==32'd62500)
   begin
      clk=~clk;
      counter=0;
   end


   else
   begin
      counter=counter+1;
   end

end
```

To multiplex the six seven segment displays, we used six control units whose values depended on the flag variables.

```
assign c1=(~flag[0]&~flag[1])|(reset_flag);

assign c2=flag[0]|(reset_flag);

assign c3=flag[1]|(reset_flag);

assign c4=(~flag1[0]&~flag1[1])&(~lap_reset)&lap_checker;
assign c5=flag1[0]&(~lap_reset)&lap_checker;
assign c6=flag1[1]&(~lap_reset)&lap_checker;
```

2.START/ STOP AND RESET OPTIONS:

In the initial state when the stopwatch starts the current value is displayed on the seven segment displays.
While the stop watch is running a press of the stop button pauses the stopwatch and pressing the same button again resumes the stopwatch.

Pressing reset button will bring stop watch to 0 from any state.

```
 if(reset_flag==1)
begin
flag=0;
q1=0;
q2=0;
q3=0;
x1=0;
x2=0;
x3=0;
end
```

We implemented the logic on the positive edge of press of push button(start/stop or reset or lap).

always@(posedge (start_stop|reset|lap))

The occurence of posedge was stored in flag variables which were used later to construct conditional statements.

```
 always@(posedge (start_stop|reset|lap))
begin

    if(start_stop==1)
    begin
    push_flag=~push_flag;
    reset_flag=0;
    if(lap_reset==1)
    begin
    lap_checker=0;
    end
    end


    if(reset==1)
    begin
    push_flag=1;
    reset_flag=1;
    lap_reset=1;
    end


    if(lap==1)
    begin
    lap_flag=lap_flag+1;
    lap_reset=0;
    lap_checker=1;
    end


end
```

3.LAP OPTION:

The Lap option is used to record time at any instant and display it on another set of three multiplexed seven segment display.

When the lap push button was pressed, we passed the recorded time to the three seven segment displays, whose control inputs were manipulated using flag variables which detected the occurence of the posedge of the push button.Thus basically the seven segment displays were selected by control inputs.

```
assign c4=(~flag1[0]&~flag1[1])&(~lap_reset)&lap_checker;
assign c5=flag1[0]&(~lap_reset)&lap_checker;
assign c6=flag1[1]&(~lap_reset)&lap_checker;
```

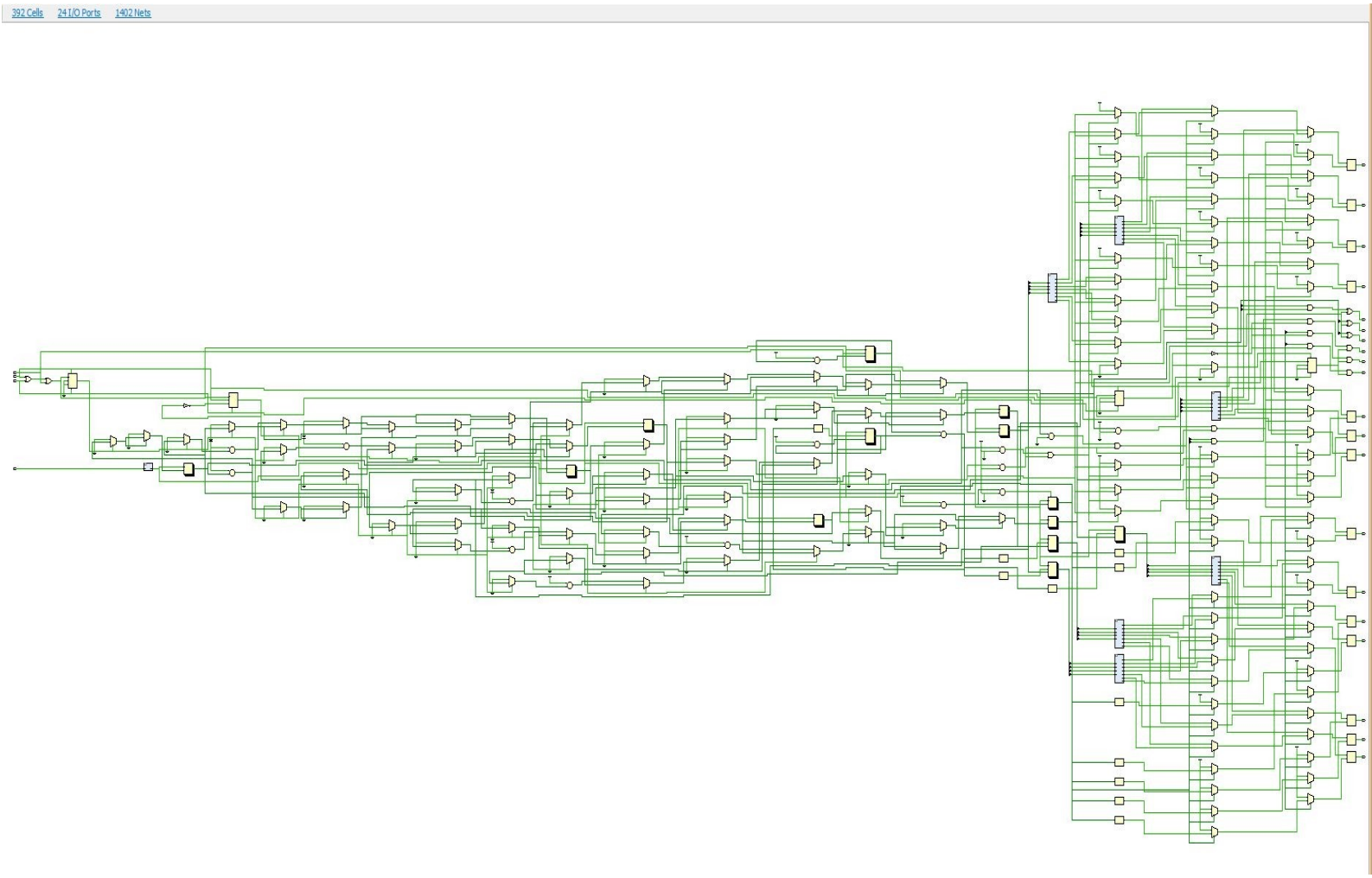
For displaying the time on the seve segments, the following code was used:

```
if(flag1==2'd0)

begin
ax=yx1;
bx=yx2;
cx=yx3;
dx=yx4;
ex=yx5;
fx=yx6;
gx=yx7;
end

else if(flag1==2'd1)
begin
ax=zx1;
bx=zx2;
cx=zx3;
```

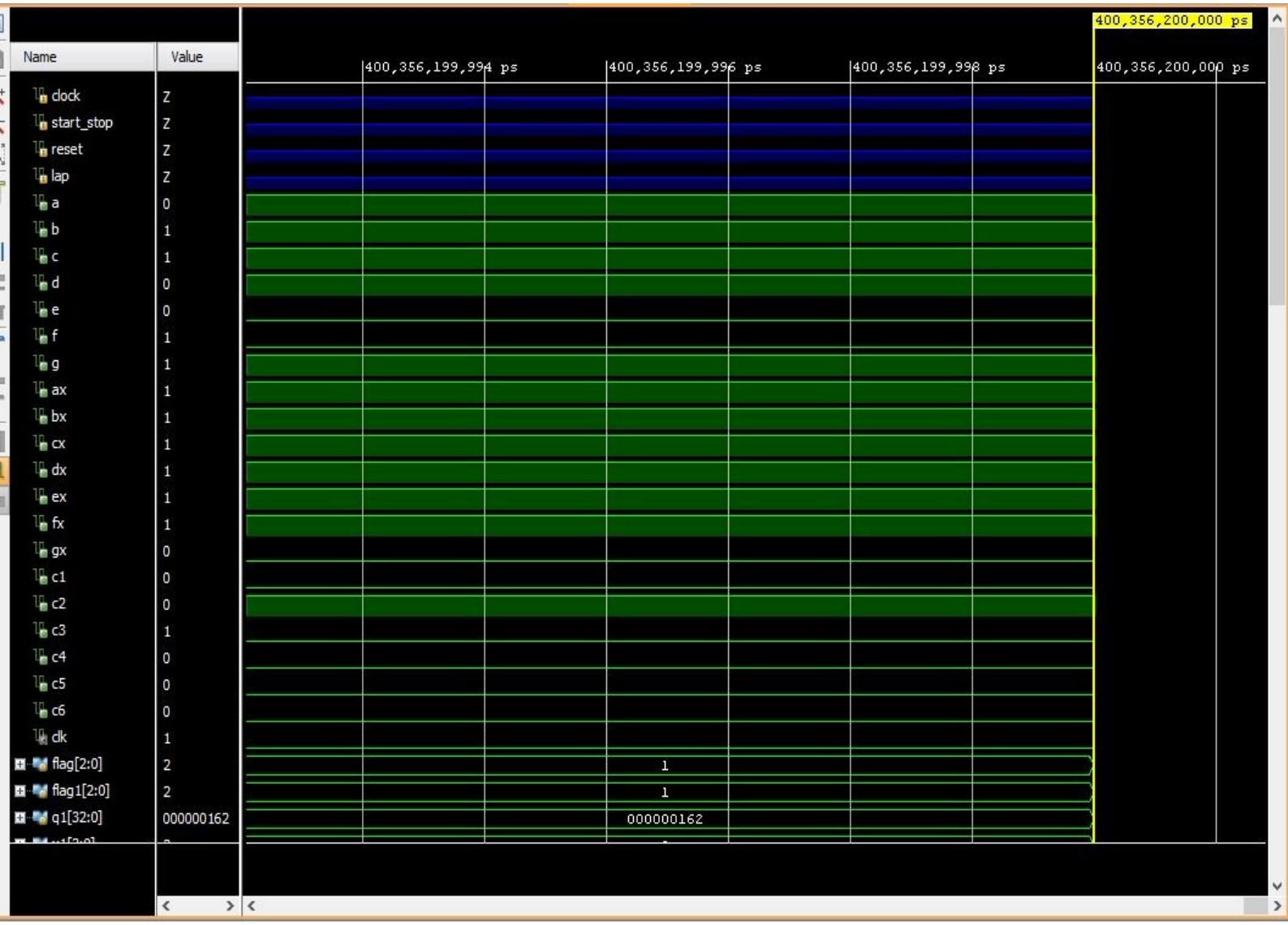
```
dx=zx4;
ex=zx5;
fx=zx6;
 gx=zx7;
end

else if(flag1==2'd2)
begin
ax=wx1;
bx=wx2;
cx=wx3;
dx=wx4;
ex=wx5;
fx=wx6;
gx=wx7;
end
```

IMPLEMENTATION:

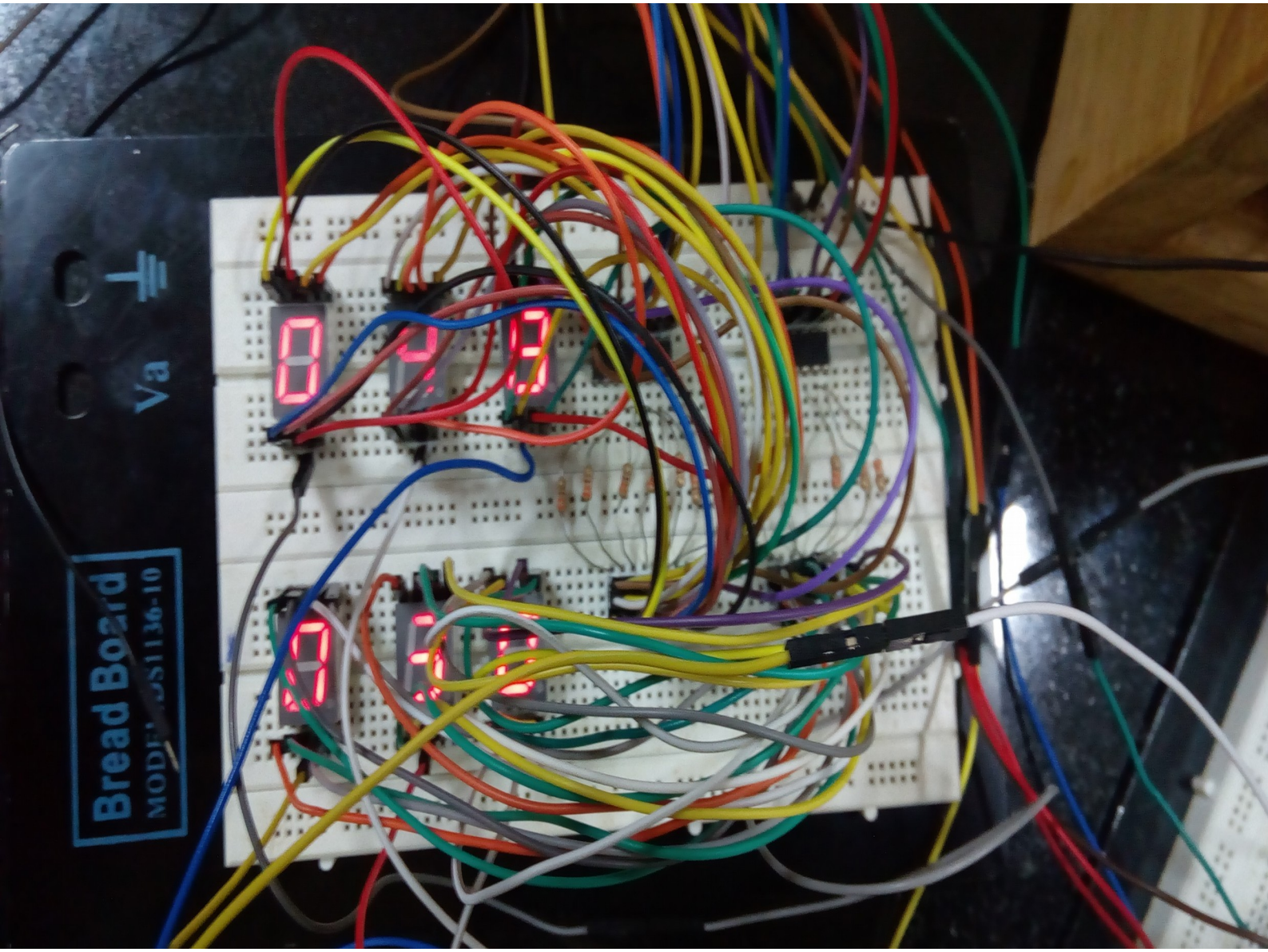
RTL SCHEMATIC:



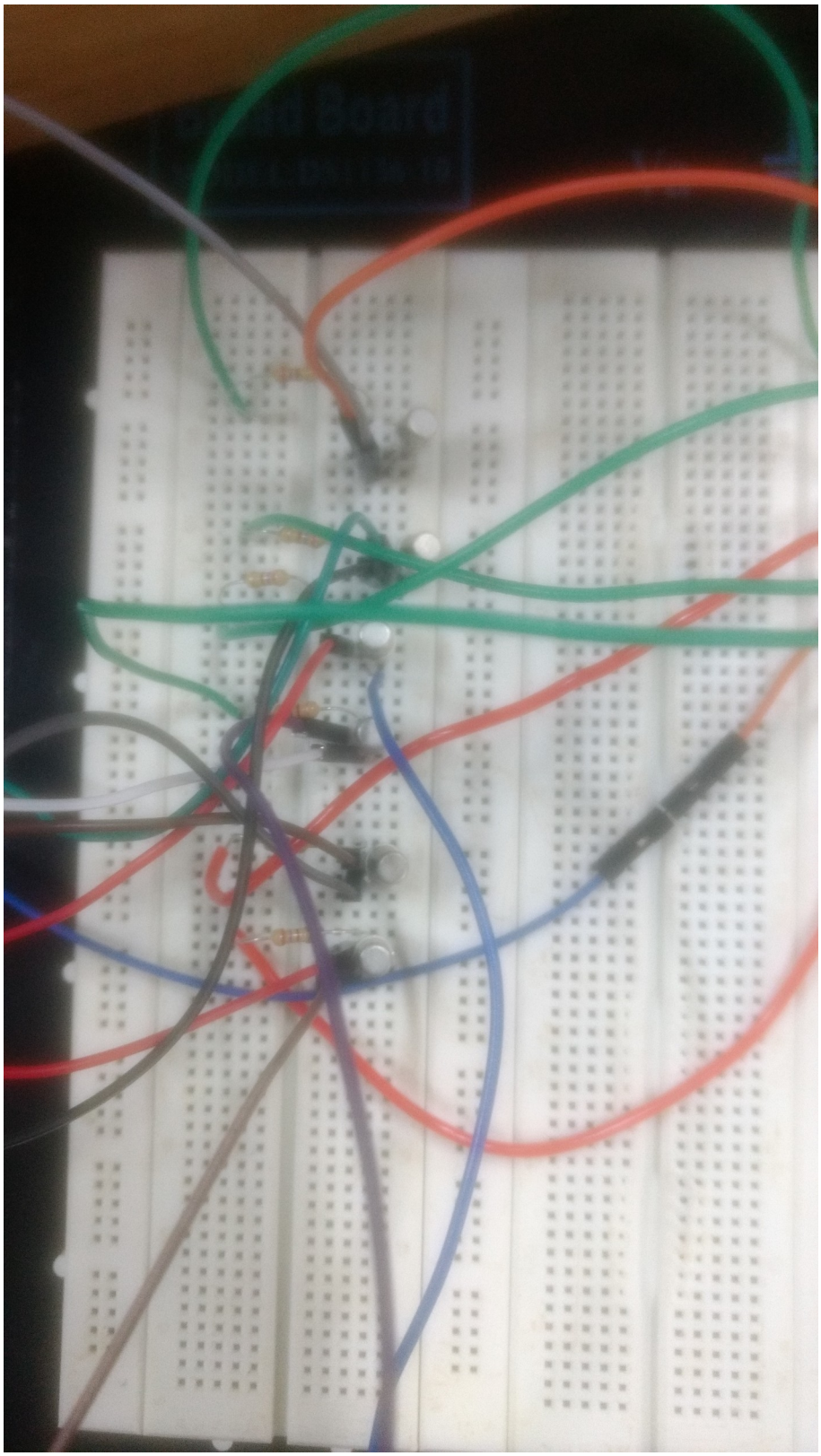
STIMULATION:



HARDWARE IMPLEMENTATION:



Another challenge that we overcame, was giving power to the seven segment display through the external power supply, instead of deriving power from the zybo. For this we used transistor(BC107) as a switch.



CONCLUSION:

We successfully implemented the stopwatch using HDL and FPGA.

We got an insight view on how complex designs are realized in industry and we also acquired knowledge about the design process.

Mainly through this project, we learnt that even simple day to day applications require designing and implementing some complex circuits.

We also learnt many aspects about verilog coding like modifying the constraint file and also some interesting techniques like multiplexing and using transistors.

APPENDIX

VERILOG CODE:

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
//
// Company: IIT PALAKKAD
// Engineer: ANNA
//
// Create Date: 11/01/2017 02:29:27 PM
// Design Name:
// Module Name: miniprojectanna2
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////
//

module k_map(A,B,C,D,a,b,c,d,e,f,g);        // kmap equations for seven segment
input A,B,C,D;
output a,b,c,d,e,f,g;
wire a,b,c,d,e,f,g;

assign a=C|A|(~B&~D)|(B&D);
assign b=~B | C&D | (~C&~D);
assign c=~C | B | D;
assign d=A | (~B&~D) | (~B&C) | (C&~D) | (B&~C&D);
assign e= (~B&~D) | (C&~D);
assign f= A | (~C&~D) | (B&~C) | (B&~D) ;
assign g=A | (~B&C) | (C&~D) | (B&~C);
endmodule




module clock_conversion(clock,clk);  // this module converts 125MHz to 1Hz

    input clock;
    output reg clk;

    reg [32:0]counter;

    initial
    begin
```

```
        counter=0;
        clk=0;
    end

    always@(posedge clock)
    begin

        if(counter==32'd62500)
        begin
            clk=~clk;
            counter=0;
        end


        else
        begin
            counter=counter+1;
        end


    end


endmodule


// main module

module miniprojectanna2(
    input clock,
    input start_stop,
    input reset,
    input lap,
    output reg a,
    output reg b,
    output reg c,
    output reg d,
    output reg e,
    output reg f,
    output reg g,
    output reg ax,
    output reg bx,
    output reg cx,
    output reg dx,
    output reg ex,
    output reg fx,
    output reg gx,
    output c1,
    output c2,
    output c3,
    output c4,
    output c5,
    output c6
    );
    wire clk;
    reg [2:0] flag;
    reg [2:0]flag1;

    reg [32:0] q1;
    reg [3:0] x1;
    reg [32:0] q2;
    reg [3:0] x2;
    reg[32:0] q3;
    reg [3:0] x3;
```

```
reg push_flag;
reg reset_flag;
reg [32:0] lap_flag;
reg [32:0]toggle;
reg lap_checker=0;
reg lap_reset=0;

reg [3:0]x4;
reg [3:0]x5;
reg [3:0]x6;

wire y1,y2,y3,y4,y5,y6,y7;
wire z1,z2,z3,z4,z5,z6,z7;
wire w1,w2,w3,w4,w5,w6,w7;

wire yx1,yx2,yx3,yx4,yx5,yx6,yx7;
wire zx1,zx2,zx3,zx4,zx5,zx6,zx7;
wire wx1,wx2,wx3,wx4,wx5,wx6,wx7;

clock_conversion inst1 (clock,clk);     // converting the clock


  // initializing the varialbes
initial
begin
    flag=0;
    flag1=0;
    q1=0;
    q2=0;
    q3=0;
    x1=0;
    x2=0;
    x3=0;
    x4=0;
    x5=0;
    x6=0;
    push_flag=0;
    reset_flag=0;
    lap_flag=0;
    toggle=0;
end


  // this will check whetherthe push button is pressed or not
always@(posedge (start_stop|reset|lap))
begin

    if(start_stop==1)         // if start_stop is pressed
    begin
    push_flag=~push_flag;
    reset_flag=0;
    if(lap_reset==1)
    begin
    lap_checker=0;
    end
    end


    if(reset==1)                  // if reset button is pressed
    begin
    push_flag=1;
    reset_flag=1;
```

```
    lap_reset=1;
    end


    if(lap==1)               // if lap is pressed
    begin
    lap_flag=lap_flag+1;
    lap_reset=0;
    lap_checker=1;
    end
end

always@(posedge clk)
begin
    flag=flag+1;

if(lap_flag-toggle==1)
begin
x4=x1;       //x1
x5=x2;    //x2
x6=x3;        // x3
toggle=toggle+1;
end

    if(reset_flag==1)        // if reset is pressed then all are to zero
    begin
    flag=0;
    q1=0;
    q2=0;
    q3=0;
    x1=0;
    x2=0;
    x3=0;
    end


    if(flag==2'd3)
    begin
    flag=0;
    end

    flag1=flag1+1;

    if(flag1==2'd3)
    begin
    flag1=0;
    end


    if(flag==0)
    begin
    // display of ones position of second
        if(q1==32'd500)
        begin

            q1=0;

            if(x1==32'd9)
            begin
                x1=0;
            end

            else if(push_flag==0)
            begin
```

```
                x1=x1+1;
            end


        end  //end q1=500

        else if(push_flag==0)
        begin
            q1=q1+1;
        end

    end                     // end of flag=0


    else if (flag==1)
    begin
              // for displaying the tenth position of second
        if(q2==32'd5000)
        begin

            q2=0;

            if(x2==32'd5)
            begin
            x2=0;
            end

            else if(push_flag==0)
            begin
            x2=x2+1;
            end

       end        // end of q=5000

            else if(push_flag==0)
            begin
                q2=q2+1;
            end
    end      // end of flag=1
    else if (flag==2'd2)
    begin                                   // for displaying the minute
                if(q3==32'd30000)
                begin

                    q3=0;

                    if(x3==32'd9)
                    begin
                    x3=0;
                    end

                    else if(push_flag==0)
                    begin
                    x3=x3+1;
                    end

                end           // end of q=300000

                    else if(push_flag==0)
                    begin
                        q3=q3+1;
                    end
   end           // end of flag=2
end              // end of posedge.
```

```
   // calling the k-map module
    k_map inst2 (x1[3],x1[2],x1[1],x1[0],y1,y2,y3,y4,y5,y6,y7);
    k_map inst3 (x2[3],x2[2],x2[1],x2[0],z1,z2,z3,z4,z5,z6,z7);
    k_map inst4 (x3[3],x3[2],x3[1],x3[0],w1,w2,w3,w4,w5,w6,w7);

    k_map inst5 (x4[3],x4[2],x4[1],x4[0],yx1,yx2,yx3,yx4,yx5,yx6,yx7);
    k_map inst6 (x5[3],x5[2],x5[1],x5[0],zx1,zx2,zx3,zx4,zx5,zx6,zx7);
    k_map inst7 (x6[3],x6[2],x6[1],x6[0],wx1,wx2,wx3,wx4,wx5,wx6,wx7);


    // these are control units
    assign c1=(~flag[0]&~flag[1])|(reset_flag);
    assign c2=flag[0]|(reset_flag);
    assign c3=flag[1]|(reset_flag);

    assign c4=(~flag1[0]&~flag1[1])&(~lap_reset)&lap_checker;
    assign c5=flag1[0]&(~lap_reset)&lap_checker;                        //
flag1[0]&(lap_flag)
    assign c6=flag1[1]&(~lap_reset)&lap_checker;


    always@(*)
    begin

    if(flag==1&&reset_flag!=1)
    begin
     a=y1;
     b=y2;
     c=y3;
     d=y4;
     e=y5;
     f=y6;
     g=y7;
    end                // end of flag==1

    else if(flag==0&&reset_flag!=1)
    begin
     a=z1;
     b=z2;
     c=z3;
     d=z4;
     e=z5;
     f=z6;
     g=z7;

    end                    // end of flag==0
    else if(flag==2'd2&&reset_flag!=1)
        begin
         a=w1;
         b=w2;
         c=w3;
         d=w4;
         e=w5;
         f=w6;
         g=w7;

        end                // end of flag==2

        else if(reset_flag==1)
                begin
                 a=1;
```

```
                b=1;
                c=1;
                d=1;
                e=1;
                f=1;
                g=0;

               end

    if(flag1==2'd0)
    begin
    ax=yx1;
    bx=yx2;
    cx=yx3;
    dx=yx4;
    ex=yx5;
    fx=yx6;
    gx=yx7;
    end

    else if(flag1==2'd1)
    begin
    ax=zx1;
    bx=zx2;
    cx=zx3;
    dx=zx4;
    ex=zx5;
    fx=zx6;
    gx=zx7;
    end

    else if(flag1==2'd2)
    begin
    ax=wx1;
    bx=wx2;
    cx=wx3;
    dx=wx4;
    ex=wx5;
    fx=wx6;
    gx=wx7;
    end



    end            // end of always(*)

endmodule
```

CONSTRAINT FILE

```
set_property -dict {PACKAGE_PIN V15 IOSTANDARD LVCMOS33 }
[get_ports {a}];
set_property -dict {PACKAGE_PIN W15 IOSTANDARD LVCMOS33 }
[get_ports {b}];
set_property -dict {PACKAGE_PIN T11 IOSTANDARD LVCMOS33 }
[get_ports {c}];
set_property -dict {PACKAGE_PIN T10 IOSTANDARD LVCMOS33 }
[get_ports {d}];
set_property -dict {PACKAGE_PIN W14 IOSTANDARD LVCMOS33 }
[get_ports {e}];
set_property -dict {PACKAGE_PIN Y14 IOSTANDARD LVCMOS33 }
[get_ports {f}];
set_property -dict {PACKAGE_PIN T12 IOSTANDARD LVCMOS33 }
[get_ports {g}];

set_property -dict {PACKAGE_PIN L16 IOSTANDARD LVCMOS33 }
[get_ports {clock}];


set_property -dict {PACKAGE_PIN T20 IOSTANDARD LVCMOS33 }
[get_ports {c1}];
set_property -dict {PACKAGE_PIN U20 IOSTANDARD LVCMOS33 }
[get_ports {c2}];
set_property -dict {PACKAGE_PIN V20 IOSTANDARD LVCMOS33 }
[get_ports {c3}];

set_property -dict {PACKAGE_PIN Y18 IOSTANDARD LVCMOS33 }
[get_ports {c4}];
set_property -dict {PACKAGE_PIN Y19 IOSTANDARD LVCMOS33 }
[get_ports {c5}];
```

```
set_property -dict {PACKAGE_PIN W18 IOSTANDARD LVCMOS33 }
[get_ports {c6}];


set_property -dict {PACKAGE_PIN R18 IOSTANDARD LVCMOS33 }
[get_ports {start_stop}];
set_property -dict {PACKAGE_PIN P16 IOSTANDARD LVCMOS33 }
[get_ports {reset}];
set_property -dict {PACKAGE_PIN V16 IOSTANDARD LVCMOS33 }
[get_ports {lap}];



set_property CLOCK_DEDICATED_ROUTE FALSE [get_nets
start_stop_IBUF] ;
set_property CLOCK_DEDICATED_ROUTE FALSE [get_nets reset_IBUF] ;
set_property CLOCK_DEDICATED_ROUTE FALSE [get_nets lap_IBUF] ;


set_property -dict {PACKAGE_PIN T14 IOSTANDARD LVCMOS33 }
[get_ports {ax}];
set_property -dict {PACKAGE_PIN T15 IOSTANDARD LVCMOS33 }
[get_ports {bx}];
set_property -dict {PACKAGE_PIN P14 IOSTANDARD LVCMOS33 }
[get_ports {cx}];
set_property -dict {PACKAGE_PIN R14 IOSTANDARD LVCMOS33 }
[get_ports {dx}];
set_property -dict {PACKAGE_PIN U14 IOSTANDARD LVCMOS33 }
[get_ports {ex}];
set_property -dict {PACKAGE_PIN U15 IOSTANDARD LVCMOS33 }
[get_ports {fx}];
set_property -dict {PACKAGE_PIN V17 IOSTANDARD LVCMOS33 }
[get_ports {gx}];
```