Analog Alarm Clock
EE 214 (Digital Circuits Lab) Project

Praveen Agrawal (12D020030)

Abstract:

Putting up in the simplest possible words, this project basically is an analog alarm clock with four ticking hands (hour, minute, second and alarm). The clock is displayed on a VGA monitor and its complete functioning is controlled using an FPGA board. The clock could be set to show the desired time (the current time) and an alarm time can be set using the provided switches. When the clock reaches the alarm time the buzzer would start ringing and would continue for some pre-defined time after which it would automatically stop.

Introduction

In the beginning we had many things in our mind for this project, but finally we zeroed onto analog clock. As we all know clocks are an indispensable part of our lives so we thought why not make an analog alarm alarm clock and display it on the VGA monitor. To make our project visually attractive we added a bunch of themes to our clock which could be changed by a pushbutton. For the alarm we made use of a buzzer which would go on when the alarm hand and hour hand coincide. Rest of the technical design and logic could be found in the subsequent sections.

Project description:

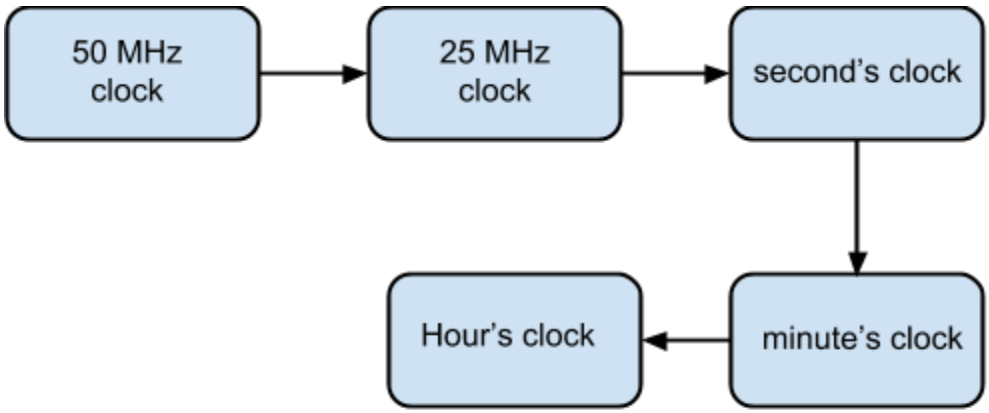
Project goal:

We were able to accomplish following goals in this project:

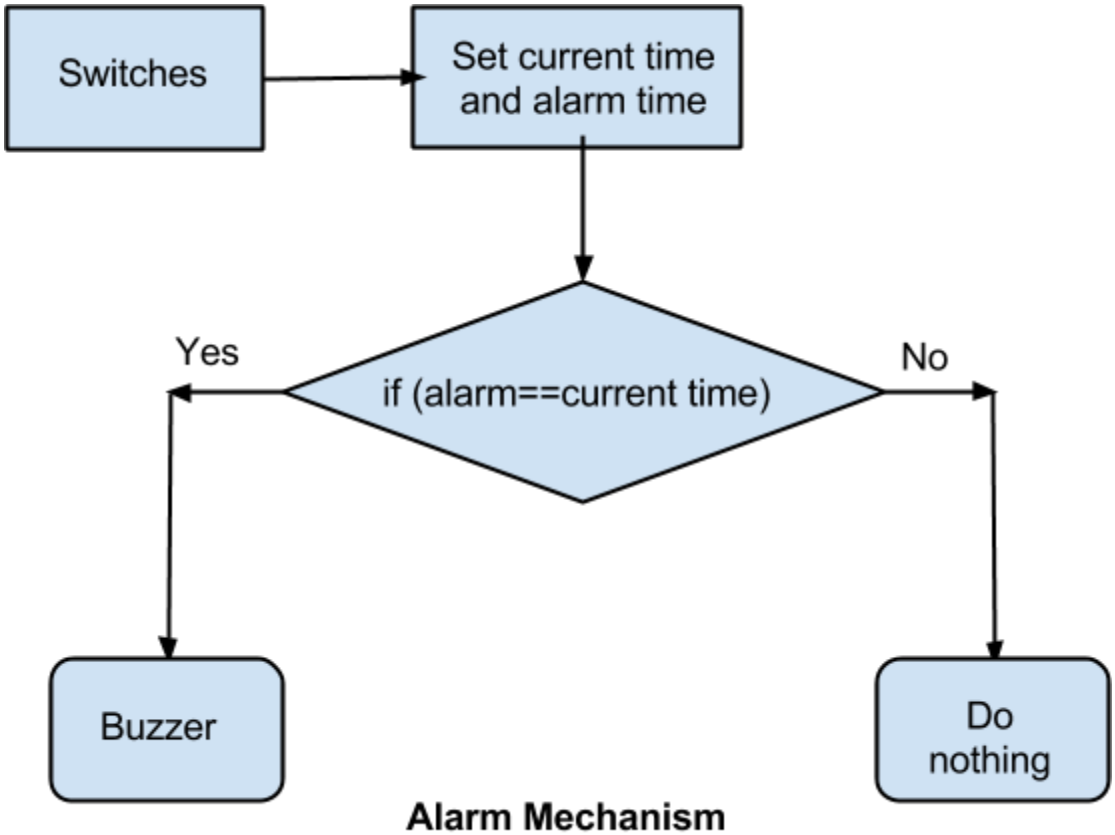
- successfully designed an analog clock using FPGA board which could display real time on VGA monitor
- made a separate hand for alarm and once the alarm-hand coincides with the hour hand alarm would start ringing for some pre-defined time by the user before it stops
- there are separate buttons to manipulate minute's hand (to set the current time) and alarm hand (to set the alarm)
- incorporated various themes for the analog clock which could be changed by pressing a push button

Technical Design

System-level overview



Time keeping mechanism of the clock



Alarm Mechanism

Software architecture

The different blocks of the program are explained below:

- Sychronising the vertical and horizontal sync signals for VGA interfacing.

```
always @(posedge dclk or negedge clrb)
begin
if(clrb==0)//reset condition
begin
   hc<=0;
      vc<=0;
end
else
begin    //counting till the end of line
if (hc < hpixels - 1)
   hc<=hc+1;
else
begin //When we hit the end of the line, reset hc<=0;     //the
horizontal counter
if(vc < vlines-1)
   vc<=vc+1;
else
   vc<=0;
end     end  end
```

  Screen resolution of 640x480 was chosen (a common choice) with 25MHz pixel clock (obtained by dividing by 2 the 50MHz clock available in the FPGA) and the following parameters:
  hpixels = 800;// horizontal pixels per line
  vlines = 525; // vertical lines per frame
  hpulse = 96; // hsync pulse length
  vpulse = 2;// vsync pulse length
  hbp = 144; // end of horizontal back porch
  hfp = 784;// beginning of horizontal front porch
  vbp = 35; // end of vertical back porch
  vfp = 515; // beginning of vertical front porch

- Printing a circle on the VGA screen: This part itself initially stood as a challenge for us as we had no clue about how to acheive it. Initially we came across an algorithm called ‘cordic’ algorithm which could provide us with the trigonometric values of different parameters and hence could be used to print the circle using its parametrized co-ordinates (a*cos (m), b*sin (m)). But it was becoming difficult to implement that algorithm and so was not used. The algorithm which then we used worked by identifying the points (pixels on VGA) which lied inside the area of the circle. An example is shown

below of how it is used in the program is shown below:

```
if(((x-x0)*(x-x0))+((y-y0)*(y-y0))<=(rad*rad))
begin
   {r,g,b}={1,0,0};
end
else
begin
   {r,g,b}={0,1,0};
end
```

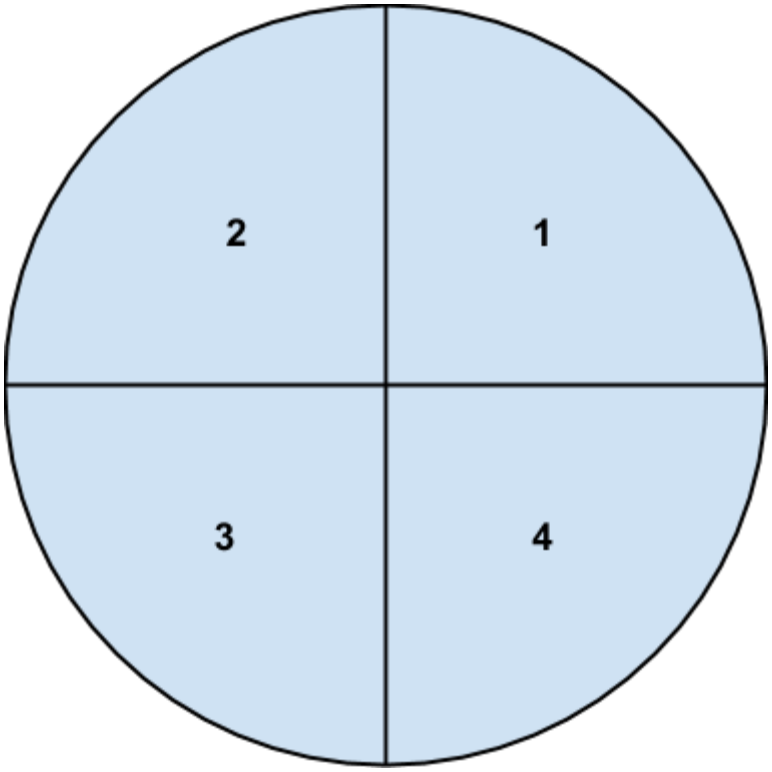
Here ‘x’ and ‘y’ represent the current pixel’s position and ‘r’, ‘g’ and ‘b’ are the signals sent to the red, blue and green pins of the VGA port. A ‘1’ corresponds to 3.3V at the FPGA ouput pin and 0.7V at the VGA port’s respective pin (pulled down using resistors). ‘rad’ is a fixed parameter for the radius of the circle to be printed. ‘x0’ and ‘y0’ are also fixed parameters to represent the center of the circle.

```
x<=hc-hbp; // hbp=144
y<=vc-vbp; //vbp=35
```

- Displaying the various hands of the clock, all simultaneously: For this part it was required to print lines of fixed lengths (length would indeed be different for different hands). For this part again the need for trigonometric values came to the picture as we required equation of a line which is best dealt in terms of slope. What was required to be done was, whenever the ‘x’ and ‘y’ variables served as a solution to the equation of the line (lets talk in terms of the second’s hand for time being) then the corresponding pixels must be coloured.

```
if((((m1*(y-y0))+(m2*(x-x0)))==0)) //eqn of line
begin
   /*
   Some other conditions to meet all the
required needs */
end
```

In the above codeblock ‘m1’ and ‘m2’ together form the slope of the second’s hand line (slope =m2/m1). Now, we need to change the values of m1 and m2 every second. So we defined m1 and m2 for all the 60 seconds. Now if we see the figure drawn below then the slope of the seconds hand would be same in the 1st and 3rd quadrants (would actually form two parts of the same line) and those in 2nd and 4th quadrants would be negative of the above values (Slope= tan (a), a=angle. And tan (a+180’)= -tan (a)).



The assigning of the slopes was done as shown below:

```
always @(posedge clk)
begin
if ((second==0) || (second==30))
begin
   m1=0;
   m2=1;
end
if   ((second==1)   ||   (second==29)   ||   (second==31)   ||
(second==59)) //tan(84)=9.514
begin
   m1=1; //1000
   m2=9; //9514
end
.
.
if(second==14||(second==16)||(second==44)|| (second==46))
//tan(6)=0.1051
begin
   m1=10; //10000
   m2=1; //1051
```

```
end
if ((second==15) || (second==45)) //tan(0)=0
begin
   m1=1;
   m2=0;
end     end
```

Similarly, the slopes for alarm hand, hour hand and minute hand were declared which kept on getting updated with the change in time.

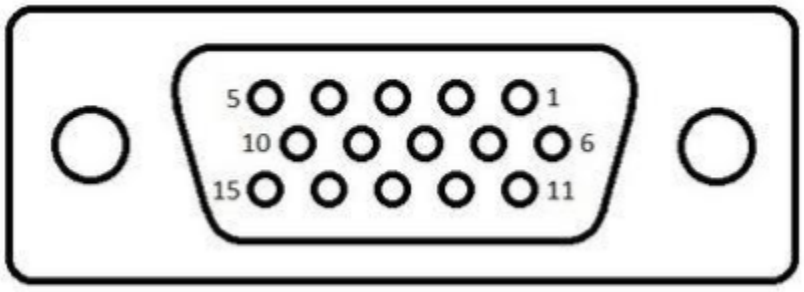
```
if((((m1*(y-y0))+(m2*(x-x0)))<=5)) //eqn of line
begin
if((second>0) && (second<=15))                                  begin

if((x>=x0)&&(y<=y0))//the part of line
{r,g,b}<=sc;      //in 1st quadrant                        end
else if((second>=30) && (second<45))                            begin
   if((x<=x0)&&(y>=y0))
            {r,g,b}<=sc;                                        end
end
```

- Setting the time and alarm: For this we needed different switches, so as to provide signals to increment and decrement the time and to change the position of the alarm hand. We used the Krypton board for this purpose as it had a good no. of switches and pushbuttons. The output signals from the Krypton board were used as inputs for the FPGA.

Hardware interfacing

1. **VGA interfacing:**

   VGA connections use a 15 pin connector called a DB15. Figure below shows the DB15 female receptacle.

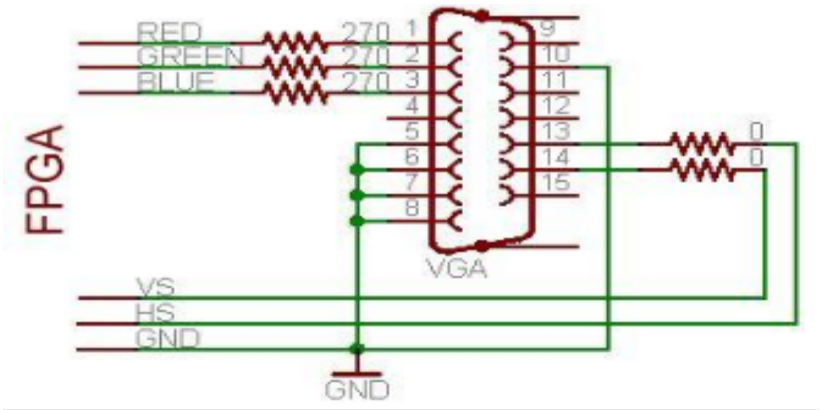


VGA Female Connector (DB15 Receptacle)

The table below lists the various pin connections of the DB15 connector:

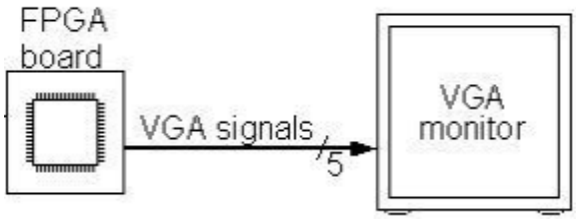
Pin	Signal	Description	Connection
1	R	analog red, 0-0.7V	DAC output
2	G	analog green, 0-0.7V	DAC output
3	B	analog blue, 0-0.7V	DAC output
4	EDID Interface	function varies depending on standard used	no connect
5	GND	general	GND
6	GND	for R	GND
7	GND	for G	GND
8	GND	for B	GND
9	no pin	or optional +5V	no connect
10	GND	for h_sync and v_sync	GND

We were provided with a female-female connector with 270 ohm series resistors connected to the rgb pins so as to pull down the 3.3V output of the FPGA to 0.7V for the DB15 connector's pins.



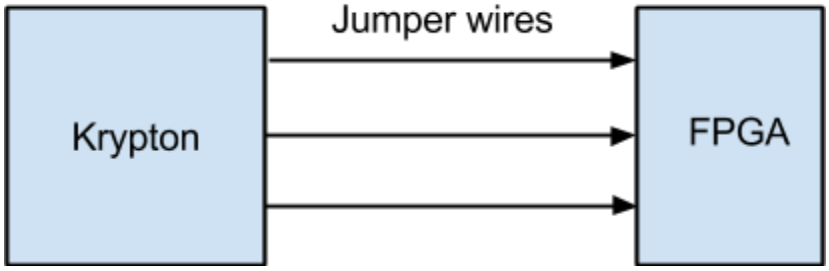
Connections of the female-female D15 connector

The overall block diagram of this VGA interfacing can be shown as below:



2. Switches for time setting:

The switches of the Krypton board were used and the output pins of the Krypton board were connected to the input pins of the FPGA using jumper wires.



Performance validation:

We tried many test cases, for eg. we set different times and waited for long enough to check whether our clock runs faithfully or not, we set the alarms at different time and checked that it rings at the correct time. We tried our clock on different VGA monitors and checked that it worked consistently on every monitor, we even took care of those cases wherein user accidentally or unknowingly presses more than button simultaneously by assigning priorities to various functions.

Project implementation

Work	Timeline
1. Study on VGA interfacing	5th March to 11th March
2. Devise the algorithm to print the clock on VGA	12th March to 18th March
3. Implementation of the devised algorithm	19th March to 25th March
4. Control of time and alarm through external switches	26th March to 1st April
5. Debugging and Finishing	2nd April to 7th April
6. Final demonstration	8th April

Difficulties faced:

The following difficulties were faced:

- VGA interfacing: We had written a simple code for printing colour bars on the VGA screen but that code wasn't working and the entire screen used to remain blank. After working on it for about 6-7 hours in total trying to debug it and find the error it was found that the values given for the front and back porch were not very correct for the required 640x480 resolution. Upon correcting those values the code executed perfectly.
- Printing the circle: The issue of this part has been explained above in the software architecture part.
- Printing the hands of the clock: This too has been dealt in the software architecture part.
- Interfacing the switches: Initially we were trying to interface a hex keypad but even after working on it for as long as 10 continuous hours we didn't find any success and so we decided to switch over to using the buttons and switches present in the Krypton board (FPGA doesn't have sufficient number of switches and push buttons).

List of components

- DE0 Nano FPGA Board
- Krypton CPLD Board
- VGA monitor
- VGA connector
- Buzzer
- Jumper wires

Results

We were able to accomplish following goals in this project:

- successfully designed an analog clock using FPGA board which could display real time on VGA monitor
- made a separate hand for alarm and once the alarm-hand coincides with the hour hand alarm would start ringing for some pre-defined time by the user before it stops
- there are separate buttons to manipulate minute's hand (to set the current time) and alarm hand (to set the alarm)
- incorporated various themes (background) for the analog clock which could be changed by pressing a push button

References:

The following links were referred to during the course of the project:

- http://eewiki.net/pages/viewpage.action?pageId=15925278
- https://drive.google.com/file/d/0B7beiFUTQkk8NkpUQXdta0Rpb2s/edit?usp=sharing
- http://www.fpga4fun.com/PongGame.html
- http://www.element14.com/community/thread/23394/l/draw-vga-color-bars-with-fpga-in-verilog