Bangladesh University of Engineering and Technology

Course No. : EEE 304

Course Title : Digital Electronics Laboratory

Project Title: Digital Clock

Name :Rafid U Murshed

Student ID : 1606186

Department : EEE

Section : C-2

Level : 3 Term : 2

Date of Submission : 22.12.2020

Members_of_Project_Group: 1606165,1606167,1606186

Abstract:

A digital clock was designed in this project using a few 74LS90 Decade Counters, basic logic gates and switches. The design is cost effective, simple and easy for maintenance.

A clock is an instrument for measuring time. In principle it requires no more than some physical process which will proceed at a known rate, and a way to gauge how long that process has been continuing.

Digital clocks display a numeric representation of time. Two numeric display formats are commonly used on digital clocks. They are: 24-hour notation with hours ranging 00 to 23 and 12-hour notation with AM/PM indicator. Most digital clocks use an LCD or LED display. Generally for the design of a digital clock, a microcontroller is used as the controller of the circuit and a Real Time Clock IC is used as a counter.

Our approach is to simulate the clock circuitry in Proteus software with all the necessary interfacing connection and program. After simulating in the Proteus, we move to hardware implementation of the Digital clock. We have also implemented a simple algorithm using Verilog code that can generate a timing diagram which can be used to display the clock's basic functionalities.

Introduction:

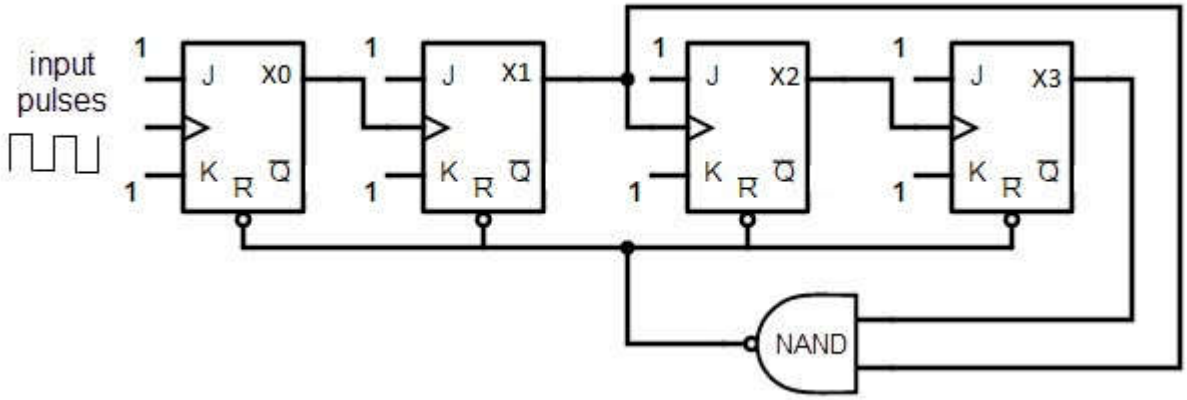
A digital clock is a type of clock that displays the time digitally (i.e. in numerals or other symbols), as opposed to an analogue clock, where the time is indicated by the positions of rotating hands. Digital clocks are often associated with electronic drives, but the "digital" description refers only to the display, not to the drive mechanism. The biggest digital clock is the Lichtzelt Pegel on the television tower Rheinturm Düsseldorf, Germany. Electronic clocks are widely used instead of Mechanical Clocks. The reason is Digital clocks are portable, reliable, accurate and maintenance free. Digital clocks are more common and independent of external sources. To implement a Digital Clock, we require some sort of counters in form of an integrated circuit or a micro-controller chip. In order to verify our design we also require some kind of software support.

Required Components:

In order to implement our design, we require the following components:

1. 74LS90 Decade Counter:

A binary coded decimal (BCD) is a serial digital counter that counts ten digits and it resets for every new clock input. As it can go through 10 unique combinations of output, it is also called a "Decade counter". A BCD counter can count 0000, 0001, 0010, 1000, 1001, 1010, 1011, 1110, 1111, 0000, and 0001 and so on.A 4 bit binary counter will act as decade counter by skipping any six outputs out of the 16 (24) outputs. There are some available ICs for decade counters which we can readily use in our circuit, like 74LS90. It is an asynchronous decade counter.



The above figure shows a decade counter constructed with JK flip flop. The J output and K outputs are connected to logic 1. The clock input of every flip flop is connected to the output of next flip flop, except the last one.

The output of the NAND gate is connected in parallel to the clear input 'CLR' to all the flip flops. This ripple counter can count up to 16 i.e. 24.

Decade Counter Operation:

When the Decade counter is at REST, the count is equal to 0000. This is first stage of the counter cycle. When we connect a clock signal input to the counter circuit, then the circuit will count the binary sequence. The first clock pulse can make the circuit count up to 9 (1001). The next clock pulse advances to count 10 (1010).

Then the ports X1 and X3 will be high. As we know that for high inputs, the NAND gate output will be low. The NAND gate output is connected to clear input, so it resets all the flip flop stages in the decade counter. This means the pulse after count 9 will again start the count from count 0.

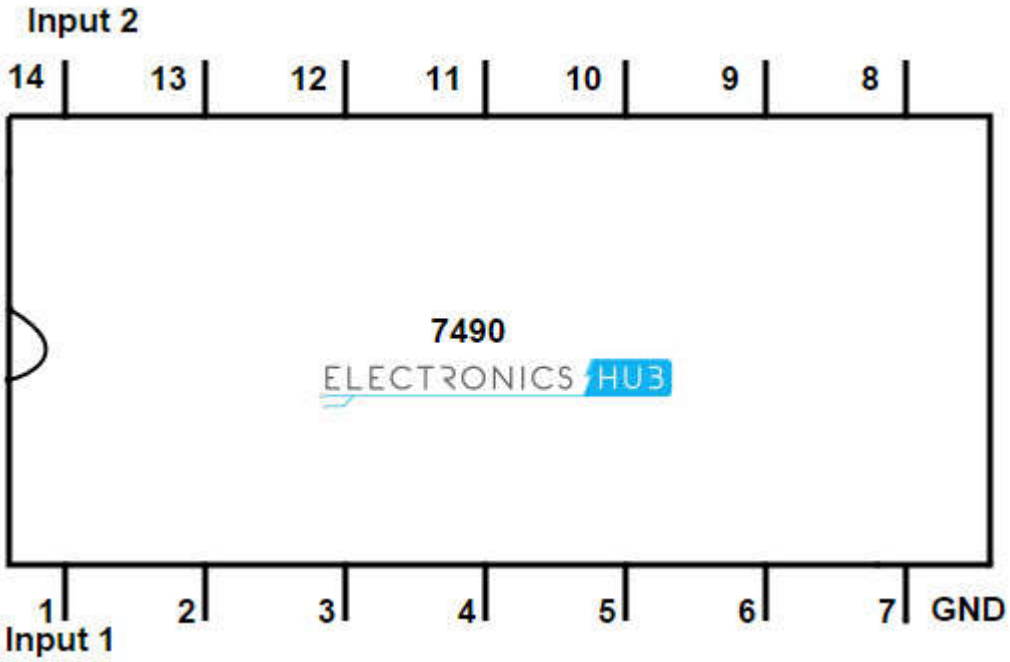
Truth Table of Decade Counter:

Input Pulses	D	C	B	A
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
0	0	0	0	0 (resets)

The above table describes the counting operation of the Decade counter. It represents the count of circuits for decimal count of input pulses. The NAND gate output is zero when the count reaches 10 (1010).

The count is decoded by the inputs of NAND gate X1 and X3. After count 10, the logic gate NAND will trigger its output from 1 to 0, and it resets all flip flops.

74LS90 decade counter IC description:



Pin No	Function	Name
1	Clock input 2	Input2
2	Reset1	R1
3	Reset2	R2
4	Not connected	NC
5	Supply voltage; 5V (4.75V – 5.25V)	Vcc
6	Reset3	R3
7	Reset4	R4
8	Output 3, BCD Output bit 2	Q_C
9	Output 2, BCD Output bit 1	Q_B
10	Ground (0V)	Ground
11	Output 4, BCD Output bit 3	Q_D
12	Output 1, BCD Output bit 0	Q_A
13	Not connected	NC
14	Clock input 1	Input1

Explanation:

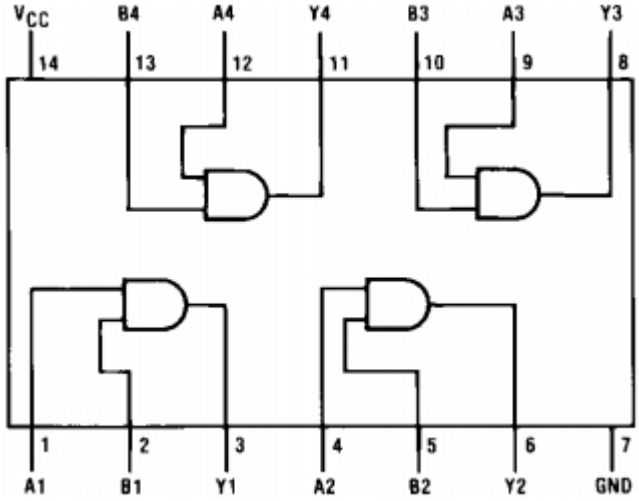
It is a simple counter which can count from 0 – 9. As it is a 4 bit binary decade counter, it has 4 output ports QA, QB, QC and QD. When the count reaches 10, the binary output is reset to 0 (0000), every time and another pulse starts at pin number 9. The Mod of the IC 7490 is set by changing the RESET pins R1, R2, R3, R4.If any one of R1 & R2 is at high or R3 & R4 are at ground, the counter will reset all the outputs QA, QB, QC and QD to 0. If the pins R3 & R4 are high, then the count on QA, QB, QC and QD is 1001.

As we studied earlier, we can increase the counting capability of a Decade number by connecting more ICs in series; we can count 99 with two 7490 ICs connected in series. This 7490 IC has inbuilt Divide by 2 and Divide by 5 counters in it.It can also be used to divide by 10 counters by connecting clock input 2 and QA and connecting all rest pins to ground and giving pulse input to 1. It is used to divide by 6 counters by supplying pulse at input 1 and grounding reset pins R3 and R4 and connecting QA with input 2.

7490 IC can work like bi –quinary counter, which is used to store decimal digits in the form of 4 bit binary numbers.

2. Basic Logic Gates:

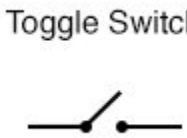
In order to implement our digital clock circuit we need some basic logic gates to pass appropriate inputs to the 74LS90. Here we have used 5 “AND” logics and 1 “OR” logic. This can be implemented by any kind of generic IC's like the 74LS08. 7408 IC is a QUAD 2-Input AND GATES and contains four independent gates each of which performs the logic AND function. A pin diagram is shown below:



3. Switches:

An electrical switch is any device used to interrupt the flow of electrons in a circuit. Switches are essentially binary devices: they are either completely on ("closed") or completely off ("open"). There are many different types of switches but in this project we mainly used some toggle switches and a couple of push-button switches. These are briefly explained below:

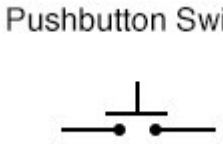
Toggle Switches:



Toggle switches are actuated by a lever angled in one of two or more positions. The common light switch used in household wiring is an example of a toggle switch.

Most toggle switches will come to rest in any of their lever positions, while others have an internal spring mechanism returning the lever to a certain normal position, allowing for what is called "momentary" operation.

Pushbutton Switches:

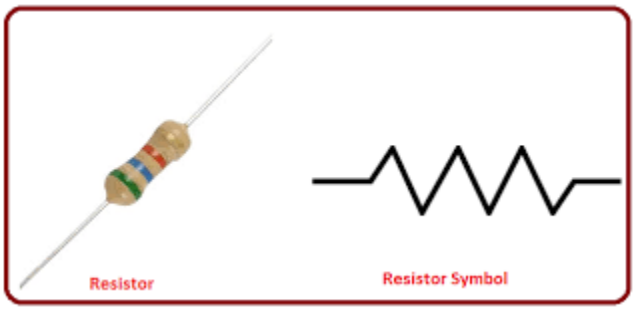


Pushbutton switches are two-position devices actuated with a button that is pressed and released. Most pushbutton switches have an internal spring mechanism returning the button to its "out," or "unpressed," position, for momentary operation.

Some pushbutton switches will latch alternately on or off with every push of the button. Other pushbutton switches will stay in their "in," or "pressed," position until the button is pulled back out.This last type of pushbutton switches usually have a mushroom-shaped button for easy push-pull action.

4. Diodes & Resistors:

All electrical circuits require some kind of current control mechanism. Here, we have used some resistors and diodes to perform this task.



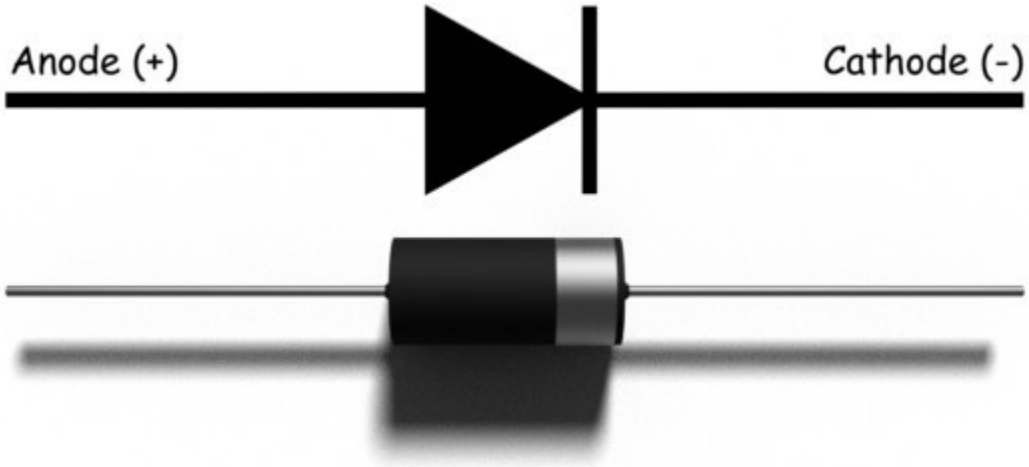


Figure: Diode

5. Seven Segment LED Display:

A seven-segment display is a form of electronic display device for displaying decimal numerals that is an alternative to the more complex dot matrix displays.

Seven-segment displays are widely used in digital clocks, electronic meters, basic calculators, and other electronic devices that display numerical information.

The 7-segment display, also written as “seven segment display”, consists of seven LEDs (hence its name) arranged in a rectangular fashion as shown. Each of the seven LEDs is called a segment because when illuminated the segment forms part of a numerical digit (both Decimal and Hex) to be displayed. An additional 8th LED is sometimes used within the same package thus allowing the indication of a decimal point, (DP) when two or more 7-segment displays are connected together to display numbers greater than ten.

Each one of the seven LEDs in the display is given a positional segment with one of its connection pins being brought straight out of the rectangular plastic package. These individually LED pins are labelled from a through to g representing each individual LED. The other LED pins are connected together and wired to form a common pin.

So by forward biasing the appropriate pins of the LED segments in a particular order, some segments will be light and others will be dark allowing the desired character pattern of the number to be generated on the display. This then allows us to display each of the ten decimal digits 0 through to 9 on the same 7-segment display.

Decimal Digit	Individual Segments Illuminated						
	a	b	c	d	e	f	g
0	×	×	×	×	×	×	
1		×	×				
2	×	×		×	×		×
3	×	×	×	×			×
4		×	×			×	×
5	×		×	×		×	×
6	×		×	×	×	×	×
7	×	×	×				
8	×	×	×	×	×	×	×
9	×	×	×			×	×

Figure: Truth Table for Seven Segment Display

Design Using a Hardware Description Language:

In order to design our project, we have used verilog which is a widely used hardware description language. **Verilog** is a Hardware Description Language; a textual format for describing electronic circuits and systems. Applied to electronic design, **Verilog** is intended to be used for verification through simulation, for timing analysis, for test analysis (testability analysis and fault grading) and for logic synthesis. The code we have used for producing different simulation results like the timing diagram is given below along with additional comments to enhance readability .

Verilog Code:

```
module DigitalClock(Clock,
                    switch,
                    setH,
                    setM,
                    stop_alarm,
                    alarm_minutes,
                    alarm_hours,
                    seconds,
                    minutes,
                    hours,
                    pm,
                    alarm,
                    led1,
                    led2,
                    led3,
                    led4,
                    led5,
                    led6);

    input Clock;  // The clock input with 1 hz frequency
    input switch; // To switch between 12 hour and 24 hour format
    input setH; // To set hours of the clock
    input setM; // To set minutes of the clock
    input [4:0] alarm_hours;  // To set hours of the alarm, should be between 0 to 23
    input [5:0] alarm_minutes; // To set minutes of the alarm
    input stop_alarm; // To stop the alarm

    output reg [4:0] hours;  // hours output of the clock
    output reg [5:0] minutes; // minutes output of the clock
    output reg [5:0] seconds; // seconds output of the clock
    output reg pm;  // Shows whether its am or pm
    output reg alarm; // This will be on when the alarm goes on

    output [6:0]led1;  // 7 segment display for units digit of the seconds
    output [6:0]led2;  // 7 segment display for tens digit of the seconds
    output [6:0]led3;  // 7 segment display for units digit of the minutes
    output [6:0]led4;  // 7 segment display for tens digit of the minutes
    output [6:0]led5;  // 7 segment display for units digit of the hours
    output [6:0]led6;  // 7 segment display for tens digit of the hours

```

```
41          reg [1:0] hours_tens;
42          reg [3:0] hours_ones;
43          reg [2:0] minutes_tens;
44          reg [3:0] minutes_ones;
45          reg [2:0] seconds_tens;
46          reg [3:0] seconds_ones;
47          reg [5:0] temp_hours;
48
49          always @(posedge(Clock))
50          begin
51              if(setH)
52                  temp_hours = temp_hours+1;  // Setting the hour of the clock
53              if(setM)
54                  minutes = minutes+1;  // Setting the minute of the clock
55
56              seconds = seconds + 1;
57              if(seconds == 60)
58              begin
59                  seconds = 0;
60                  minutes = minutes + 1;
61                      if(minutes == 60)
62                      begin
63                          minutes = 0;
64                          temp_hours = temp_hours + 1;
65                          if(temp_hours == 24) // For 24 hours format
66                                  temp_hours = 0;
67                      end
68              end
69
70              if(~switch)  // For 12 hours format
71              begin
72                  hours = (temp_hours%12);
73                  if(hours==0)
74                      hours = 12;
75                  pm = (temp_hours>11);
76              end
77              else if(switch) // For 24 hours format
78                      hours = temp_hours;
79          if((alarm_hours==temp_hours)&(alarm_minutes==minutes)&(stop_alarm==0)) // Setting the alarm
80              alarm = 1;
81          if((stop_alarm==1)) // To stop the alarm
82              alarm = 0;
83
84          hours_tens = (hours/10)%10;  // tens digit of the hour of the clock
85          hours_ones = hours%10;  // units digit of the hour of the clock
86          minutes_tens = (minutes/10)%10; // tens digit of the minute of the clock
87          minutes_ones = minutes%10;  // units digit of the minutes of the clock
88          seconds_tens = (seconds/10)%10; // tens digit of the second of the clock
89          seconds_ones = seconds%10;  // units digit of the second of the clock
90
91      end
92
93
94          segment7Led st0(seconds_ones,led1);  // 7 segment output of units digit of seconds
95          segment7Led st1({1'b0,seconds_tens},led2); // 7 segment output of tens digit of seconds
96          segment7Led st2(minutes_ones,led3); // 7 segment output of units digit of minutes
97          segment7Led st3({1'b0,minutes_tens},led4); // 7 segment output of tens digit of minutes
98          segment7Led st4(hours_ones,led5); // 7 segment output of units digit of hours
99          segment7Led st5({2'b00,hours_tens},led6); // 7 segment output of tens digit of hours
100
101 endmodule
```

```
104  module segment7Led(bcd,seg);  // To see the digits of hours, minutes and seconds in 7 segment LED
105
106       input [3:0] bcd;
107       output reg [6:0] seg;
108
109      always @(bcd)
110      begin
111          casex (bcd)
112              0 : seg = 7'b1111110;
113              1 : seg = 7'b0110000;
114              2 : seg = 7'b1101101;
115              3 : seg = 7'b1111001;
116              4 : seg = 7'b0110011;
117              5 : seg = 7'b1011011;
118              6 : seg = 7'b1011111;
119              7 : seg = 7'b1110000;
120              8 : seg = 7'b1111111;
121              9 : seg = 7'b1111011;
122              default : seg = 7'bx;
123          endcase
124      end
125
126  endmodule
```

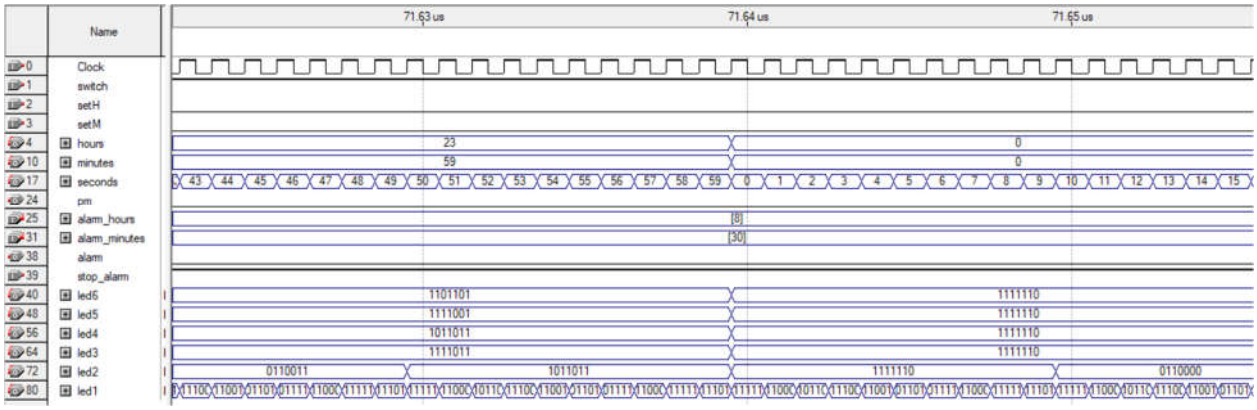
The code can primarily be broken down into 5 parts.

a) The lines 1-47 declared the inputs and outputs of our digital clock module.
b) The skeleton of the code which implements the timekeeping part of our code can be found in the lines 49-78
c) The alarm system has been implemented in the next few lines from 79 to 82.
d) The lines 84-100 are used to display the time.
e) The rest of code contains the module that implements the seven-segment display.

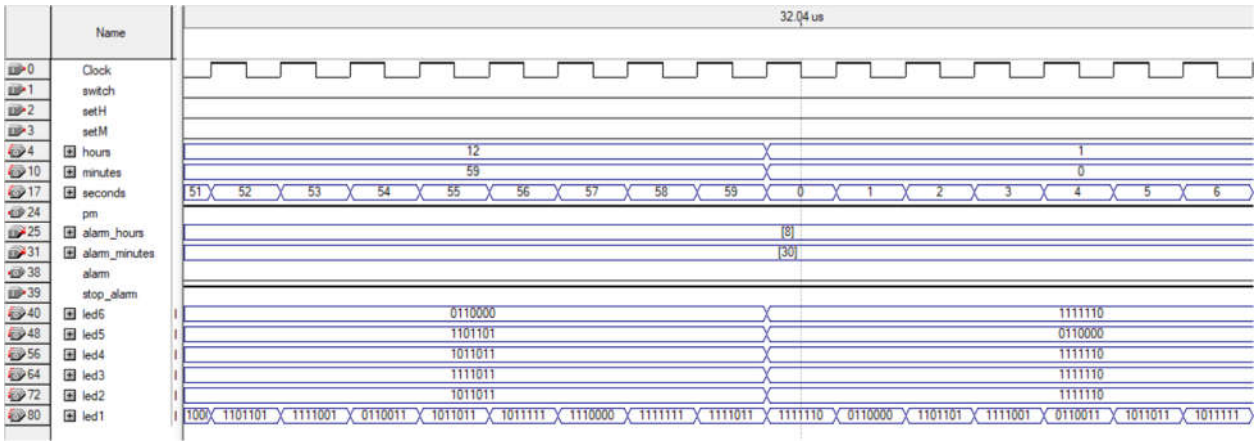
Timing Diagram:

The period of a practical clock should be 1 second. But for demonstration purposes, we took the period of the clock to be 1 ns and set the end time at 100 us to observe all the changes.

Main Clock:

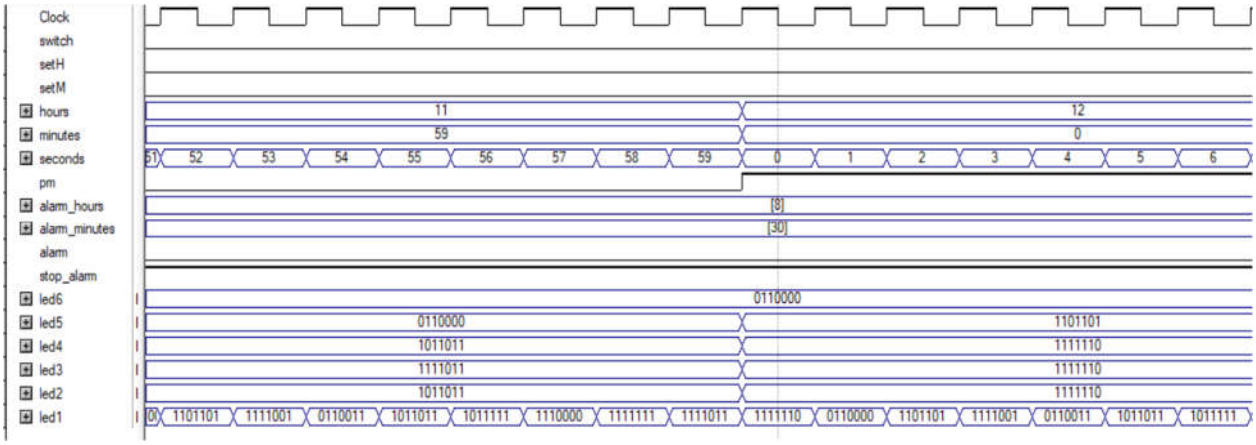


Here, **switch** is 1. So, the clock is counting in 24 hour format. After 23:59:59, the clock starts counting from 0:0:0 again.

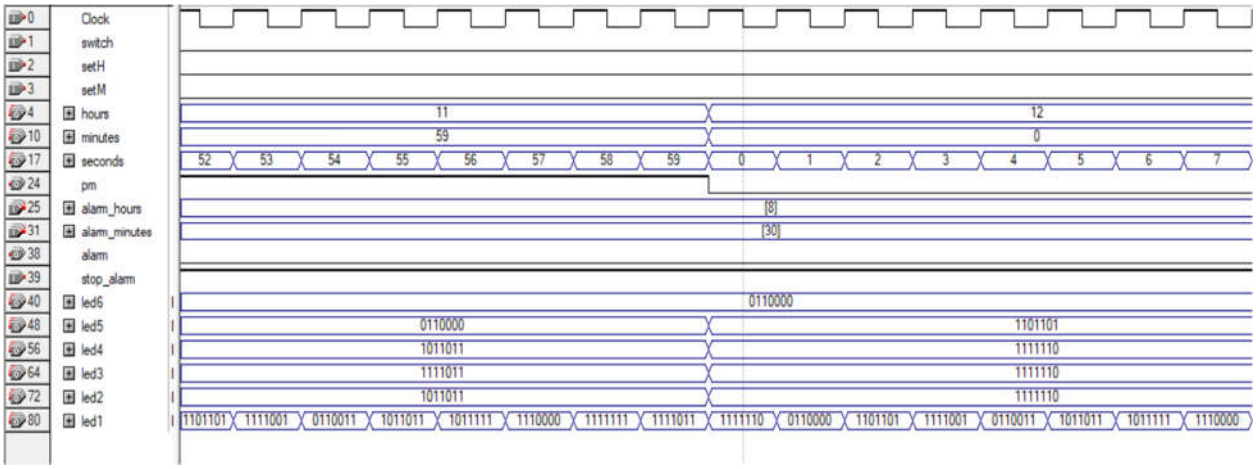


Here, **the switch** is 0. So, the clock is counting in 12 hour format. After 12:59:59, the clock starts counting from 1:0:0 again.

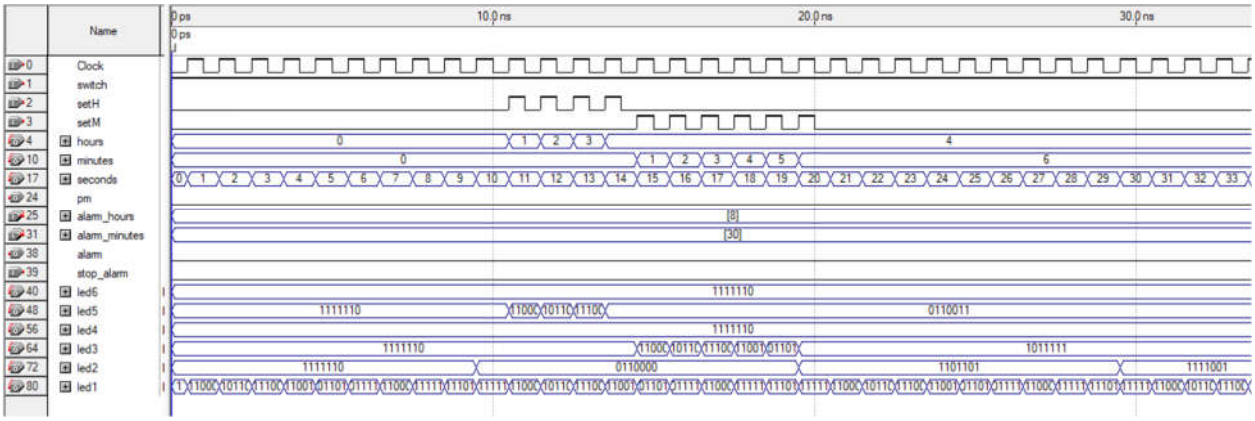
pm indicates whether it is am or pm. For the first 12 hours, **pm** is 0. So, it's am.



For the next 12 hours, **pm** is 1. So, it's pm. Then the cycle repeats.



Setting Hours and Minutes of a clock:

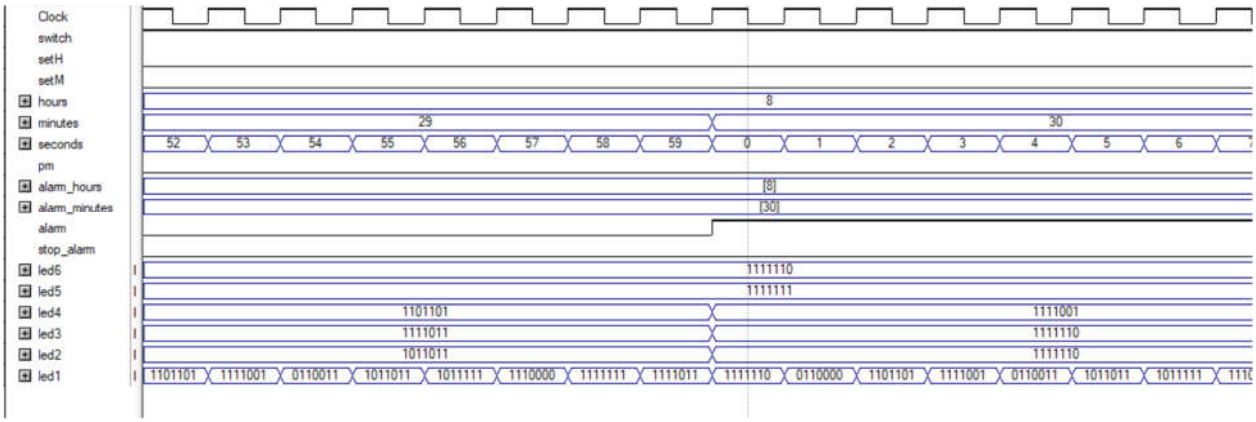


There are 4 pulses in setH. So, it adds 4 with the current value of **hours**. As the current value of **hours** is 0, the **hours** is set to 4.

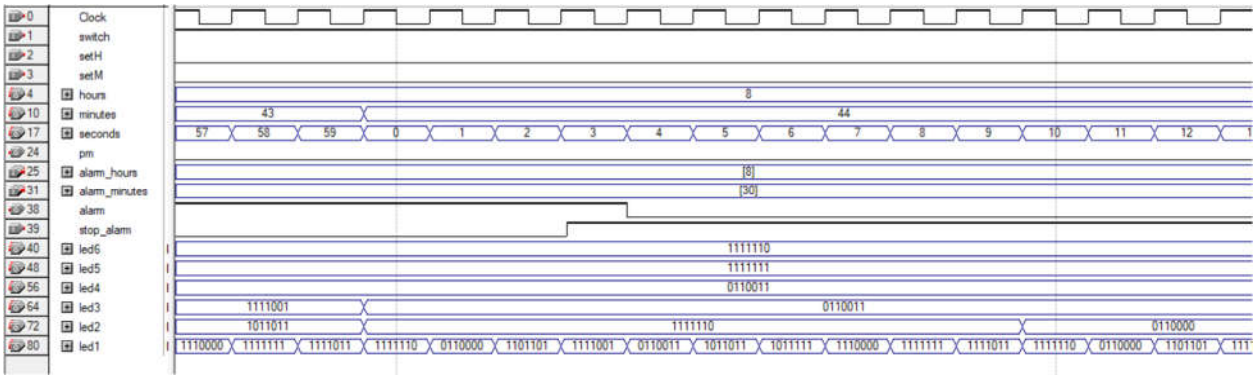
There are 6 pulses in setM. So, it adds 6 with the current value of **minutes**. As the current value of **minutes** is 0, the **minutes** is set to 6.

Alarm Feature:

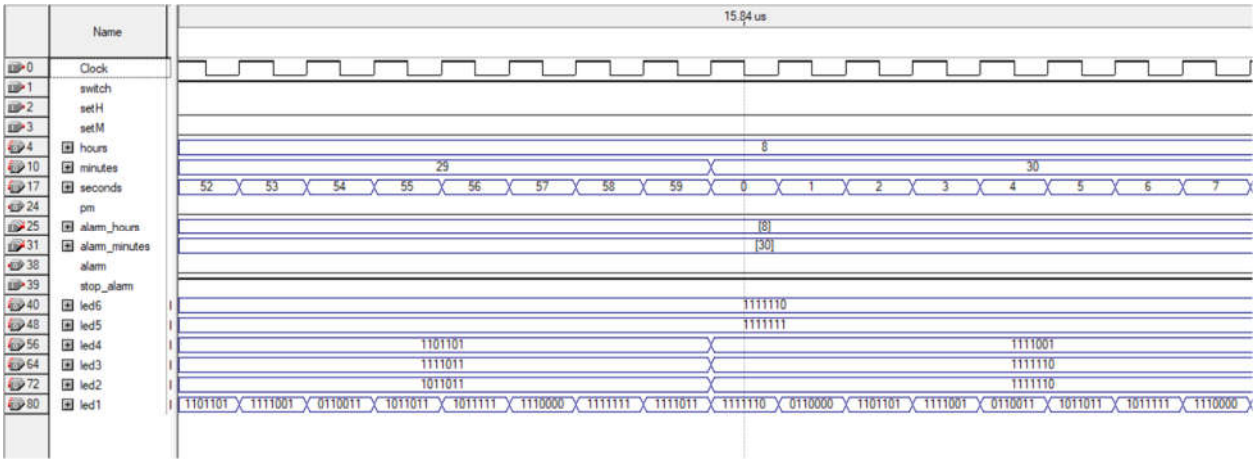
The **alarm** was set to 8:30. So, when it's 8:30:0, the **alarm** goes on.



After the **alarm** goes on, if the **stop_alarm** is 1, **the alarm** will stop.



If we don't want the alarm feature, we simply have to make the **stop_alarm** 1.

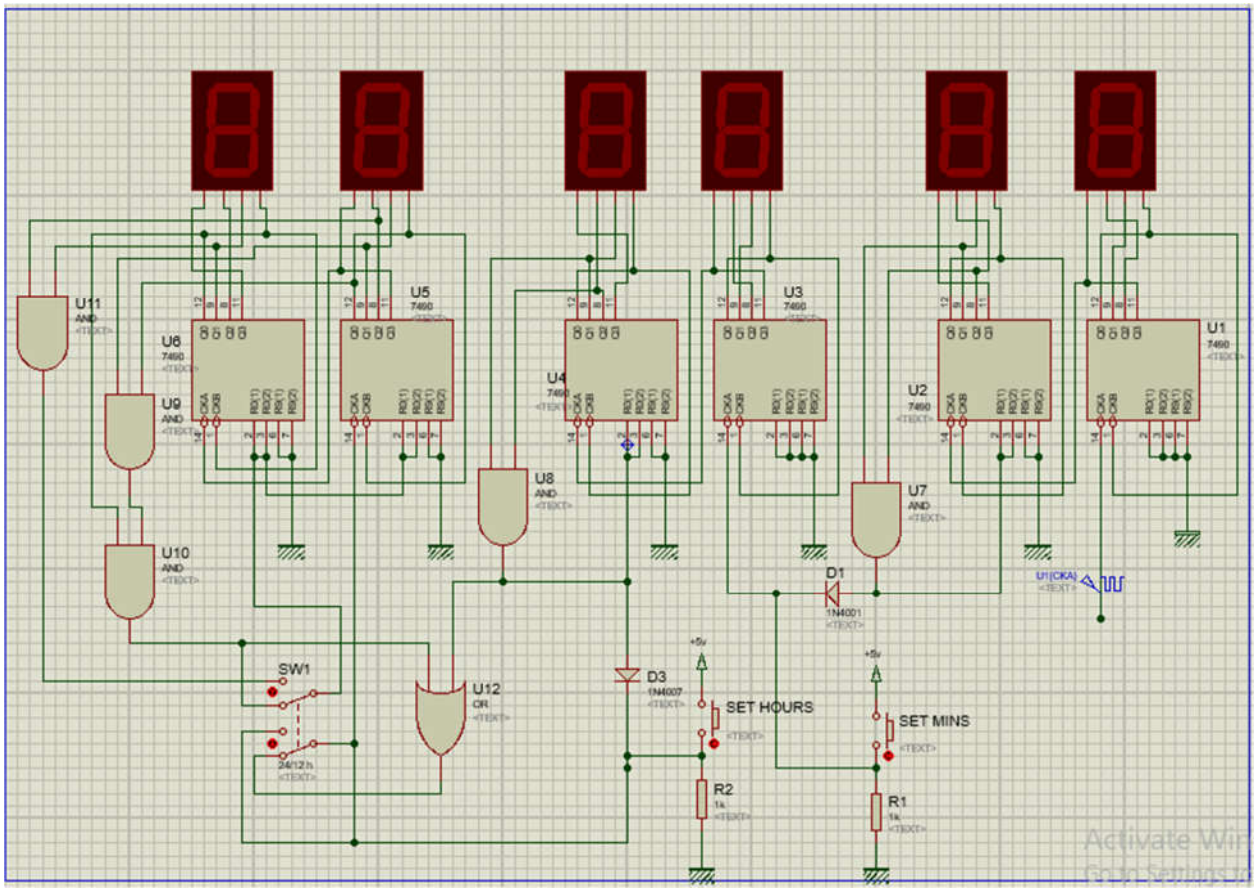


. From the diagram, we can see that as **the stop_alarm** was 1, the alarm didn't go on at 8:30:0.

The leds are just the output of a 7-segment led display.

Simulation Using Proteus:

The Proteus Design Suite is a proprietary software tool suite used primarily for electronic design automation. The software is used mainly by electronic design engineers and technicians to create schematics and electronic prints for manufacturing printed circuit boards.
In order to implement our design to build a fully functional digital clock we need to simulate our whole design in proteus to check whether all the features are working properly. A schematic for our simulation is shown below:



Here, we can clearly visualize all the required logic elements and their interconnections. This vivid portrayal and the simulation results helped us to implement our design. The push button switches of “SET HOURS” AND “SET MINS” can be used to manually set the hours and minutes. The toggle switches can be used to change the format of the clock between 12 hours and 24 hours.

Challenges:

Although the design of our project looks simple and the components used seem very basic, we had to face quite a few challenges during the implementation of the whole project. These are mentioned below:

1. Designing the algorithm and code structure:

   We used the Verilog hardware description language to design our project and this verilog HDL differs from the typical widely used languages like C++,python etc in more than one way. As a result we had to face a lot of syntaxial difficulties. The execution of the verilog interpreter is also quite different which gave rise to some code structuring problems. All in all, the coding part of the project was quite challenging.

2. Proteus Simulation:

   Due to complex interconnections between various components we found it difficult to connect all the circuit components in the logic circuit. Connecting the 7 segment led display to the counter output was also quite challenging.

3. Timing Diagram:

   Due to the inherent complex nature of the timing diagram, we had to face quite a few problems while simulating the timing diagram outputs. Selecting the proper frequency of the clock and showing the various features clearly was also very time consuming and cumbersome.

Advantages and features of the digital clock system:

The main advantage of the **digital clock system** is the **indication of an absolutely exact time** (with an error of run – 1s for 300000 years). The accuracy of signal timing is provided by the cesium master clock at the Paris Bureau of Weights and Measures.

Digital clock system provides:

- Coordinated indications of all clocks at the site with a global universal time (UTC/GMT);
- Synchronous indication of an exact zone time in the pointer and digital formats on all clocks;
- Automatic conversion of clocks during the transition to winter/summer time;
- Automatic setting of the clock to the exact time after restoration of power or liquidation of the accident on a line;
- Automatic restoration of the correct indications of the clock at failures or during power interruption for a period of up to 1 week;
- Synchronization of the computer network in accordance with the calendar date and exact time;
- Simplicity of usage that does not require special training of the engineering personnel.

Limitations of The Project:

For avoiding complexity, we did not simulate the alarm feature in the proteus simulation. Also, the stop alarm function of the code doesn't work exactly as it was intended. In Spite of these limitations, we were able to finish our project with quite satisfactory results.

Conclusion:

All in all, this project has been accomplished effectively and fulfills the overall project aims and goals. The new digital clock system is a ic-based system that allows easy reprogramming of the required outcomes compared to the hardware-based scheme that uses fixed timing systems. By studying integrated logic circuits, it is evident that it is much easier to design a digital clock than to use other hardware components. Some additional features can be added to the clock if required based upon practical usage. As well as programming and working with different simulation software, we have used various troubleshooting methods and learned a lot of new things.

Project Contribution:

1. Project Idea: Tanvir Haider Pantha
2. Basic Project Structure: All
3. Code Skeleton: Rafid U Murshed, Tanvir Haider Pantha
4. Code Modification, Feature Addition, Code Structuring: Rafid U Murshed
5. Enhancing Code Readability: Tanvir Haider Pantha, Rafid U Murshed
6. Code Simulation and Timing Diagram: Tanvir Haider Pantha
7. Proteus Simulation: Asad Fahim
8. Project Report and Presentation: Rafid U Murshed
9. Project Demonstration (Video): All

List of References:

1. Basic Electronics Tutorials and Revision
2. Digital Clock Using AT89C51
3. What does digital clock mean?
4. https://leater.com/en/services/digital-clock-system.html#:~:text=The%20main%20advantage%20of%20the,Bureau%20of%20Weights%20and%20Measures.