DS Lab Project

Soham P More - EE23BTECH11223
Prajwal M - EE23BTECH11217

May 5, 2024

Course instructor: Prof. Siva Vanjari

भारतीय प्रौद्योगिकी संस्थान हैदराबाद
Indian Institute of Technology Hyderabad

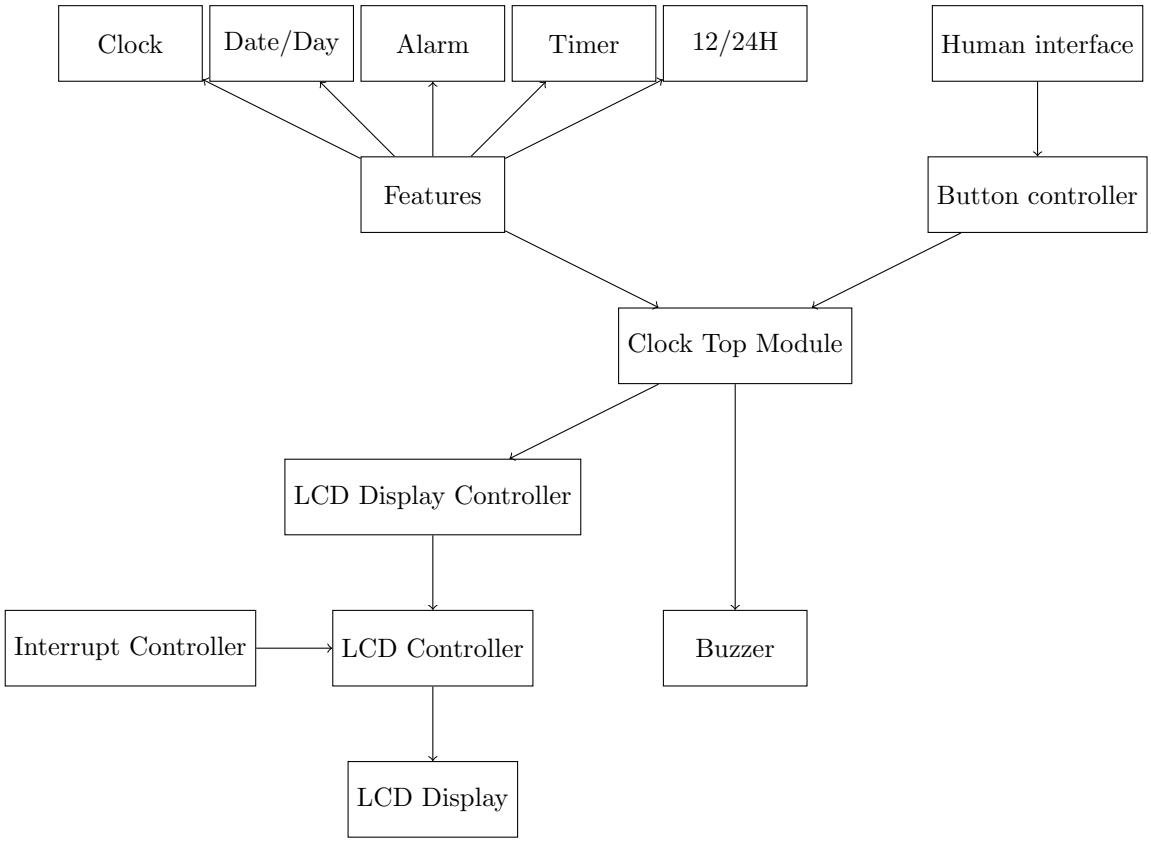
Contents

1 Problem statement

Functionalities of the Clock:

1. This PS is open-ended, but the bare minimum requirement expected from you is to display the time on an LCD display.
2. The clock should contain 2 modes: 24-hour format (HH—MM—SS) and 12-hour format (An external LED should display whether the time is AM or PM.
3. The clock should also have a timer feature, and when the timer drops to zero, a buzzer should ring.
4. The clock should also have a timer feature, and when the timer drops to zero, a buzzer should ring.
5. You can get as creative as you want and add as many additional features as you want. For example, your clock may display the date and day, provide time zone support, etc.

2 Block diagram



3 Features

3.1 Clock

3.1.1 clocktime.v

Port	Direction	Width	Description
clk_1hz	input	1	Clock signal with 1 Hz frequency
rst	input	1	Reset signal
clk_mode	input	2	Clock mode selection
time_in	input	24	Input time signal
time_out	output	24	Output time signal

Table 1: Port Description for clocktime Module

The *clocktime* module processes time signals in hours, minutes, and seconds. It synchronizes with a 1 Hz clock and supports reset functionality. Depending on the clock mode, it either passes through the input time or increments the time signals, handling rollovers for seconds, minutes, and hours.

3.1.2 settime.v

Port Name	Direction	Width	Description
clk	Input	1	Clock signal
button1	Input	1	Button 1 input signal
button2	Input	1	Button 2 input signal
button3	Input	1	Button 3 input signal
set_mode	Input	2	Set mode selector signal
hour1	Output	4	First digit of the hour
hour2	Output	4	Second digit of the hour
min1	Output	4	First digit of the minute
min2	Output	4	Second digit of the minute
sec1	Output	4	First digit of the second
sec2	Output	4	Second digit of the second
dbg_led	Output	5	Debug LED output

Table 2: Port description table for the settime module

The *settime* module facilitates setting the time using buttons. It operates based on a finite state machine (FSM) where each button press cycles through setting hours, minutes, and seconds. A debug LED indicates the current setting stage, and the time digits are displayed in BCD format.

3.2 Date/Weekdays

3.2.1 date.v

Port Name	Direction	Width	Description
clk	Input	1	System clock signal
hour_in	Input	8	Input hour signal
date_in	Input	24	Input date signal
weekday_in	Input	3	Input weekday signal
date_mode	Input	2	Date mode input signal
date_out	Output	24	Output date signal
weekday_out	Output	3	Output weekday signal

Table 3: Port Description Table for date.v

date is a date and wekday counter. It allows external date setting and correctly handles leap years, month transitions, and year transitions. The output signals are updated accordingly based on the internal logic.

3.2.2 setdate.v

Port Name	Direction	Width	Description
clk	Input	1	Clock signal
button1	Input	1	Button 1 input signal
button2	Input	1	Button 2 input signal
button3	Input	1	Button 3 input signal
set_mode	Input	2	Set mode selector signal
day1	Output	4	First digit of the day
day2	Output	4	Second digit of the day
month1	Output	4	First digit of the month
month2	Output	4	Second digit of the month
year1	Output	4	First digit of the year
year2	Output	4	Second digit of the year
day	Output	3	Weekday output signal

Table 4: Port description table for the setdate module

setdate is a Verilog module for setting date and weekday using buttons with debounce and mode switching capabilities.

3.3 Alarm

3.3.1 alarm.v

The *alarm* module compares the input time with a stored alarm time. When the alarm mode is activated, and the input time matches the alarm time within

Port	Direction	Width	Description
clk	Input	1	Clock signal
rst	Input	1	Reset signal
alarm_mode	Input	2	Alarm mode selector signal
in_time	Input	16	Input time signal
ring	Output	1	Buzzer output signal

Table 5: Port description table for the alarm module

a one-minute window, the buzzer is triggered. The alarm is set using *settime* module.

3.4 12/24 Hour format

3.4.1 formattime.v

Port Name	Direction	Width	Description
clk_mode	Input	2	Clock mode selector
timer_mode	Input	2	Timer mode selector
setampm	Input	1	Time format toggle signal
clock_time	Input	24	Current clock time
alarm_time	Input	24	Alarm time
timer_time	Input	24	Timer time
bcd_time	Output	24	Converted time output
ampm	Output	1	AM/PM indicator

Table 6: Port Description for formattime Module

The *formattime* module converts clock or timer time to 12-hour format, considering AM/PM. It toggles between 12-hour and 24-hour format based on the input signal setampm. The output bcd_time provides the converted time, and ampm indicates whether it's AM or PM.

3.5 Timer

3.5.1 timer.v

The timer module operates as a countdown timer. It decrements time by one second each time the 1 Hz clock signal clk_1hz ticks, unless the timer is in set mode ($timer_mode = 2'b01$). In set mode, it updates the time registers based on the input time. The $time_out$ signal represents the current time, and the buzzer output goes high when the timer reaches 00:00:00.

Port Name	Direction	Width	Description
clk_1hz	input	1	1 Hz clock signal
timer_mode	input	2	Timer mode signal
time_in	input	24	Input time signal
time_out	output	24	Output time signal
buzzer	output	1	Buzzer output signal

Table 7: Port Description for the timer Module

4 Controllers

4.1 Button Controller

4.1.1 button_controller.v

Port	Direction	Width	Description
mclk	input	1	Main Clock signal
rst	input	1	Reset signal
pSetButton	input	1	Set Button
pAlarmButton	input	1	Set Alarm Button
pTimerToggle0	input	1	Set Timer Button
pTimerToggle1	input	1	Show Timer Button
pButton0	input	1	Button0 input
pButton1	input	1	Button1 input
pButton2	input	1	Button2 input
clk_mode	output	2	Clock Mode Signal
timer_mode	output	2	Timer Mode Signal
vButton	output	6	Virtual Button Output

Table 8: Port Description for `Button Controller` Module

This module handles button inputs. To prevent bouncing of buttons, it samples each button at 5 ms / Sample. It outputs the output of these buttons as vButtons, it also decreases the number of buttons needed by mapping a single button acting in different modes to different vButtons.

4.2 Interrupt Controller

4.2.1 interrupt_controller.v

This module manages timings for LCD display, Since a instruction needs 4.1-8.00 ms to execute vs 40-100 us for data, this module is used in lcd_controller module to efficiently use time. When raiseInterrupt goes HIGH, the module counts delay_ms milliseconds before raising the interrupt output HIGH.

Port	Direction	Width	Description
mclk	input	1	Main Clock signal
rst	input	1	Reset signal
raiseInterrupt	input	1	Raise Interrupt Signal
delay_ms	input	16	Amount of time to wait
interrupt	output	1	Interrupt Line
MFREQ_KHZ	parameter	_	No of clock cycles in 1 ms
REPEAT	parameter	1	Repeats the interrupt signal every delay_ms ms if set to 1

Table 9: Port Description for `Interrupt Controller` Module

4.3 LCD Controller

4.3.1 lcd_controller.v

Port	Direction	Width	Description
mclk	input	1	Main Clock signal
rst	input	1	Reset signal
LineA	input	128	Data to show on 1st line.
LineB	input	128	Data to show on 2nd line.
DB	output	8	Data output pins to LCD Display
E	output	1	LCD Display Enable Line
RS	output	1	LCD Display RS Line
RW	output	1	LCD Display RW Line
MFREQ_KHZ	parameter	_	No of clock cycles in 1 ms
InsWaitTime	parameter	_	Time to wait after sending an instruction
DataWaitTime	parameter	_	Time to wait after sending data

Table 10: Port Description for `LCD Controller` Module

This module initializes and sets up the LCD display, then sends the data from the lcd_display_controller module. To send data/instruction the module sets up RS, RW, DB(bus) lines then raises Enable line to HIGH for few ms, then pulls it down and waits either InsWaitTime/DataWaitTIme before sending the next data/instruction. It runs initializing code once, then sends LineA, LineB data repeatedly.

4.4 LCD Display Controller

4.4.1 lcd_display.v

This module chooses what to show on the display based on the mode the clock is in.

Port	Direction	Width	Description
mclk	Input	1	Main Clock signal
rst	Input	1	Reset signal
clk_mode	Input	2	Clock mode selection
timer_mode	Input	2	Timer mode selector
vButton	Input	6	Virtual Buttons Input
DB	Output	8	Data output pins to LCD Display
E	Output	1	LCD Display Enable Line
RS	Output	1	LCD Display RS Line
RW	Output	1	LCD Display RW Line
MFREQ_KHZ	Parameter	_	No of clock cycles in 1 ms
InsWaitTime	Parameter	_	Time to wait after sending an instruction
DataWaitTime	Parameter	_	Time to wait after sending data

Table 11: Port Description for `LCD Display Controller` Module

5 Top module

```
module test_lcd_ctrl(
    input wire rst, output wire redled, output wire dbg_l2, output
        wire dbg_l3, output wire dbg_l4,
    input wire pSetButton, input wire pAlarmButton, input wire
        pTimerToggle0, input wire pTimerToggle1,
    input wire pButton0, input wire pButton1, input wire pButton2,
    output wire RS, output wire E, output wire RW,
    output wire LCD_DB0,
    output wire LCD_DB1,
    output wire LCD_DB2,
    output wire LCD_DB3,
    output wire LCD_DB4,
    output wire LCD_DB5,
    output wire LCD_DB6,
    output wire LCD_DB7,
    output wire buzzer
);

    reg reset = 0;
    wire clk;
    wire[5:0] vButton;
    wire[1:0] mode;
    wire[1:0] timer_mode;
    wire[4:0] tmpLED;
    reg led;

    qlal4s3b_cell_macro qlal4s3b_cell(.Sys_Clk0(clk));

    button_controller #(20000) bctrl(
        .mclk(clk), .rst(reset),
        .pSetButton(~pSetButton), .pAlarmButton(~pAlarmButton), .
            pTimerToggle0(~pTimerToggle0), .pTimerToggle1(~
            pTimerToggle1),
```

```
30          .pButton0(~pButton0), .pButton2(~pButton2), .pButton1(~
                pButton1),
31          .vButton(vButton), .clk_mode(mode), .timer_mode(timer_mode)
32      );
33
34      lcd_display_controller #(10000, 5, 2, 2) lcd_disp_ctrl(
35          .mclk(clk), .rst(reset),
36          .E(E), .RS(RS), .RW(RW),
37          .DB({LCD_DB7, LCD_DB6, LCD_DB5, LCD_DB4, LCD_DB3, LCD_DB2,
                LCD_DB1, LCD_DB0}),
38          .vButton(vButton), .clk_mode(mode), .timer_mode(timer_mode)
                ,
39          .dbg_led(tmpLED),
40          .buzzer(buzzer)
41      );
42
43      always @ (posedge clk) begin
44          if(vButton[0]) begin
45              led <= ~led;
46          end
47      end
48
49      assign redled = vButton[4];
50      assign dbg_l2 = vButton[5];
51      assign dbg_l3 = timer_mode[0];
52      assign dbg_l4 = timer_mode[1];
53
54  endmodule
```

Signal	PU64 pin alias	I/O pin
rst	62	6
redled	34	22
dbg_l2	30	26
dbg_l3	27	28
dbg_l4	25	30
LCD_DB7	6	2
LCD_DB6	2	3
LCD_DB5	64	5
LCD_DB4	63	7
LCD_DB3	60	9
LCD_DB2	57	11
LCD_DB1	55	13
LCD_DB0	40	16
E	38	18
RW	36	19
RS	39	21
buzzer	32	24
pSetButton	33	23
pAlarmButton	31	25
pTimerToggle0	23	31
pTimerToggle1	61	8
pButton0	28	27
pButton1	26	29
pButton2	3	4

Table 12: quickfeather.pcf

6 Working

7 Conclusion

We have succesfully built a clock using Vaman FPGA with features such as alarm, timer, 12-24 hour format and date.


Set Time
17:50:00
05-05-2024,Sun
05:40:55 PM
05-05-2024,Sun
05:40:53 PM

Figure 1:

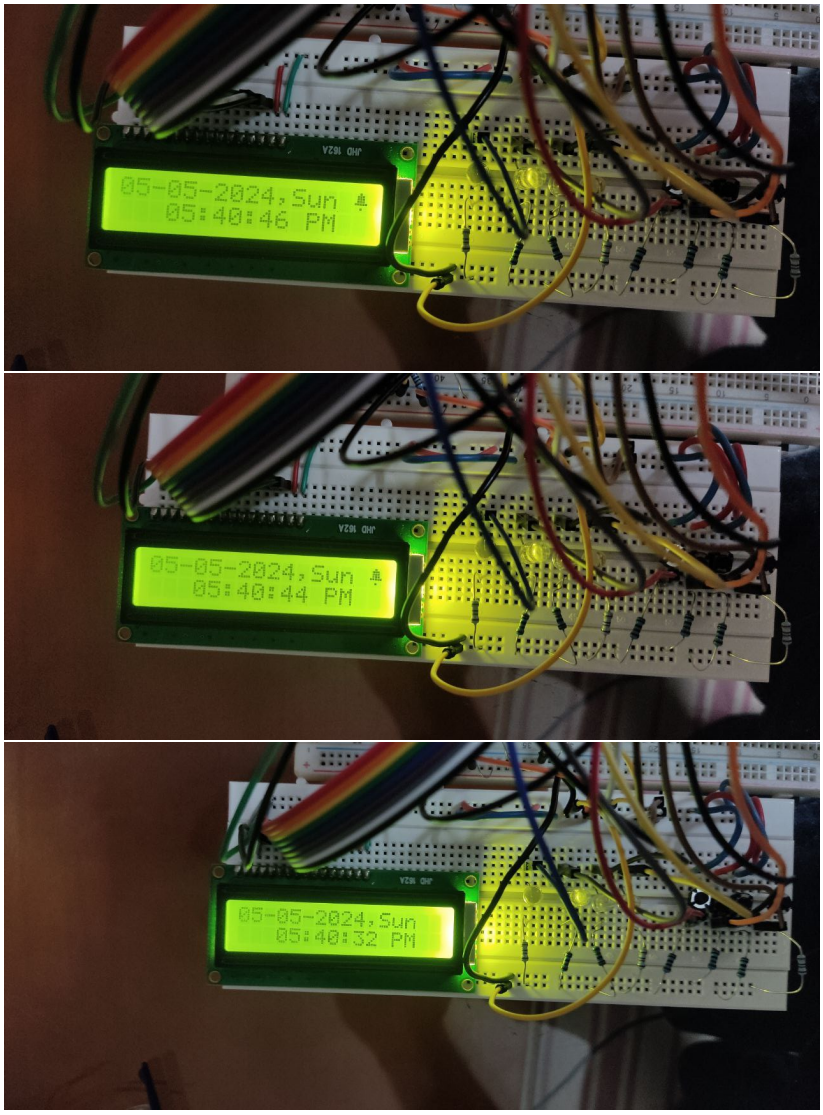

05-05-2024,Sun
05:40:46 PM
05-05-2024,Sun
05:40:44 PM
05-05-2024,Sun
05:40:32 PM

Figure 2:

Figure 3:

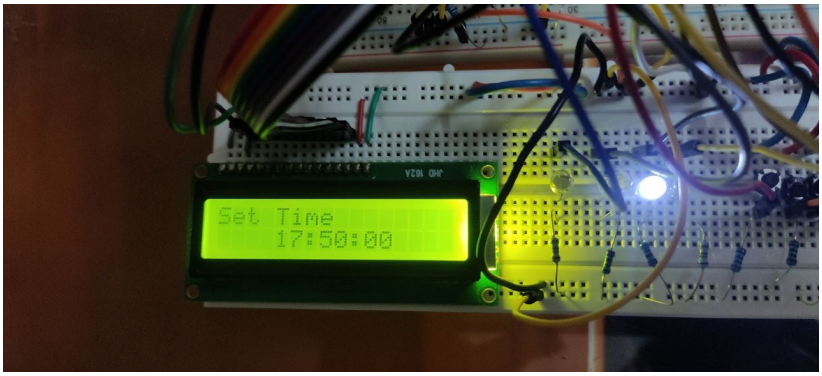


Figure 4: Set time

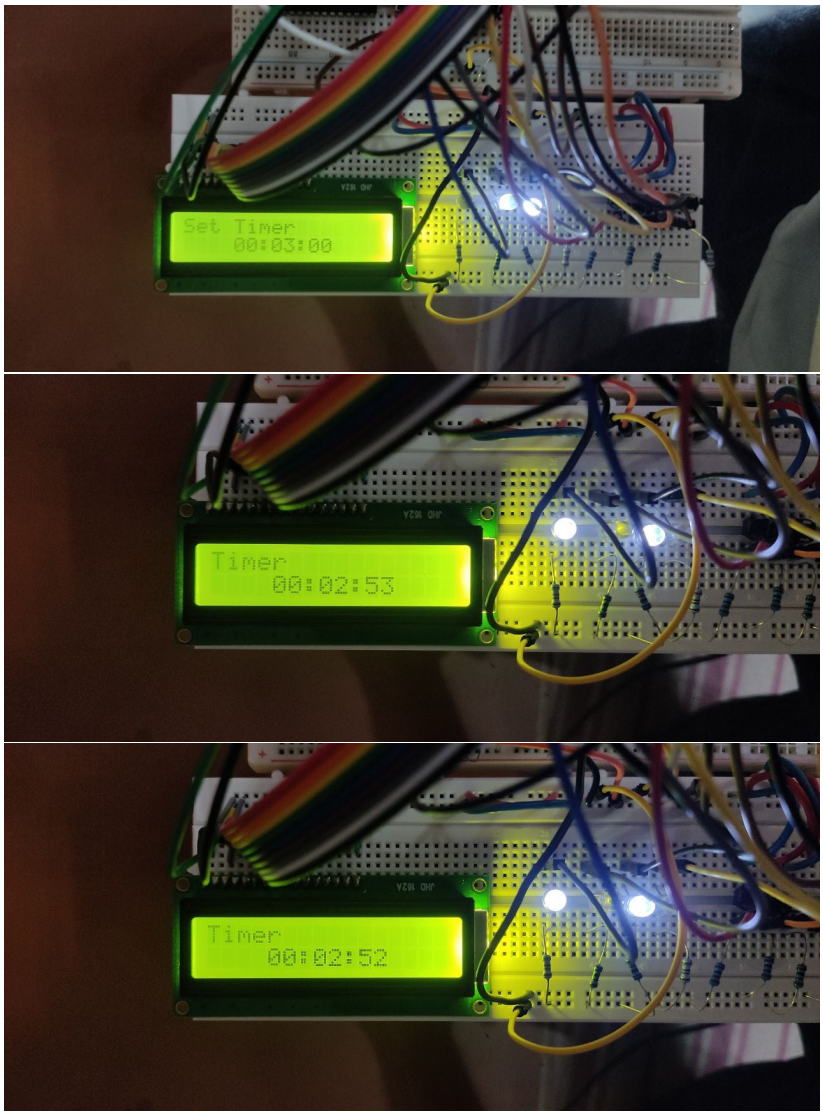


Figure 5: Settimer (above), Timer (below)

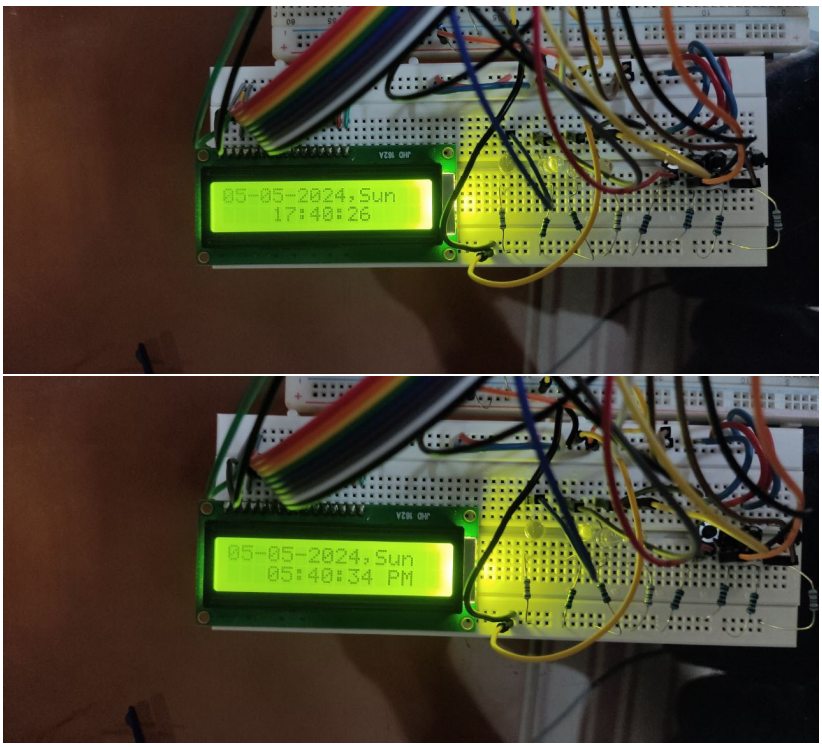


Figure 6: 24 Hour format (above), 12 Hour format (below)