CS203 : DIGITAL LOGIC DESIGN

PROJECT REPORT

GROUP MEMBERS :-

Name:	Entry No.:
Aditi Das	2020CSB1064
Jugal Chapatwala	2020CSB1082
Shruti Sikri	2020CSB1128

IMPLEMENTING A SIMPLE ALARM CLOCK ON FPGA BOARD (BASYS 3) WITH PIEZO BUZZER

The Verilog code for implementing a 24-hour Alarm Clock on FPGA is presented in this project. The code is fully synthesizable for FPGA implementation with a Piezo buzzer for the Alarm sound. The alarm clock outputs a real-time clock with a 24-hour format and also provides an alarm feature. Users can set the clock and alarm time through switches manually. The LEDs glow according to the current second and one LED for when the Alarm sets off. The current time is displayed along with the feature of setting the alarm. When the current time coincides with the set-alarm time, the alarm signal goes high and the LED 15 on the FPGA board glows along with the Piezo buzzer.

1. MODULE HIERARCHY:-

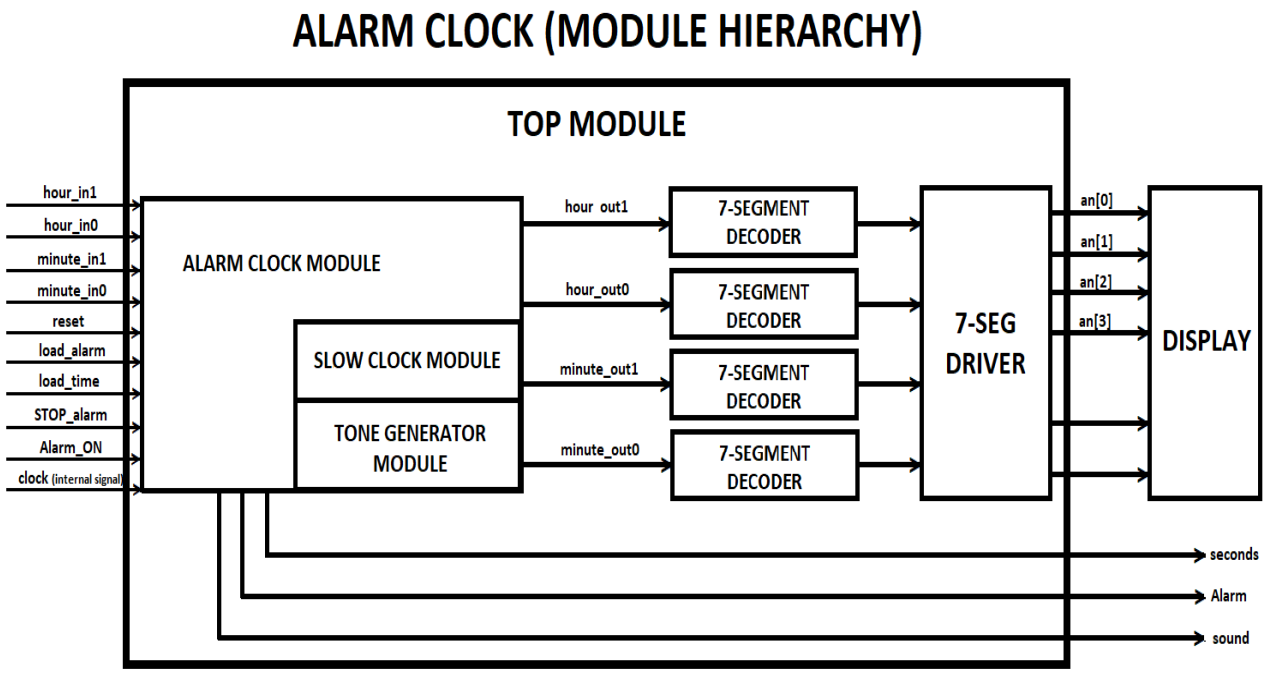


Fig. 1 Overall Module Hierarchy

2. DESCRIPTION OF MODULES AND SUBMODULES:-

a. alarm_clock (in alarm_module.v):

This module is the core module of our project. It is responsible for all the clock and alarm functionalities in our alarm clock. It gauges passage of time, maintains an internal clock, outputs the current time, stores the alarm time, outputs whether or not the current clock time is equal to alarm time and also connects to the buzzer system to generate sound when alarm goes off.

b. slowClock (in slow_clock.v):

This module is used to set an internal clock of 1 second that can be used to track the passage of time in our system. The original clock needs to be slowed down i.e. the 100MHz clock of the FPGA has been converted to a 1Hz signal.

c. display_decoder (in 7segment_decoder.v):

This module takes in a 4-bit number and outputs the corresponding 7 segment code (For active LOW specification). It is used to convert the alarm_module outputs to 7-segment codes so that the number can be displayed on the 7-segment display of the FPGA.

d. driver_module (in 7seg_driver.v):

The driver module is needed because in the in-built 7-segment display of the FPGA board the corresponding segments of each display (left, center-left, center-right and right) have a common cathode. However the anodes are different. So we can choose which display is to remain active. The driver module rapidly flips through the displays and the number that needs to be displayed in order to trick the human eye into believing that each display is showing a different number.

e. clock (in top_module.v):

The purpose of this clock module is to instantiate and set up connections for all the sub-modules. It also provides an interface of variables that are used in the "constraint.xcd" file to set-up connections with buttons, switches, leds and displays on the FPGA board.

f. tone_generator (in tone_generator.v):

This module takes the tone as input and generates a sound signal that can be directly fed to the buzzer to produce a sound.

g. constraint.xdc :

The XDC constraint file assigns the physical IO locations on the FPGA to the switches, LEDs and push buttons located on the board.

3. FEATURES ON FPGA BOARD:

a. Switches 0-3 (connected to minute_in0):

Used to set the 4-bit input for the least significant minute digit of the clock (if load_time=1), or the least significant minute digit of the alarm (if load_alarm=1). Valid values are 0 to 9.

b. Switches 4-7 (connected to minute_in1):

Used to set the 4-bit input for the most significant minute digit of the clock (if load_time=1), or the most significant minute digit of the alarm (if load_alarm=1). Valid values are 0 to 5.

c. Switches 8-11 (connected to hour_in0):

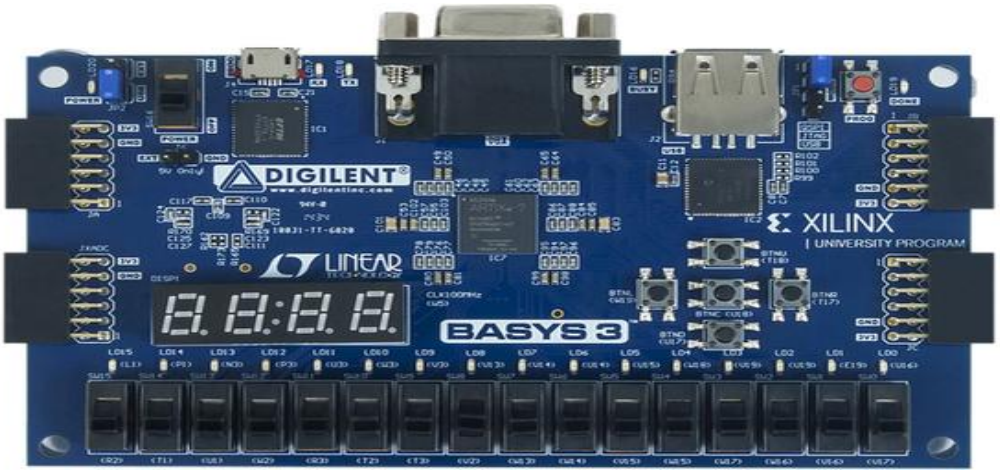
Used to set the 4-bit input for the least significant hour digit of the clock (if load_time = 1), or the least significant hour digit of the alarm (if load_alarm=1). Valid values are 0 to 9.

d. Switches 12-13 (connected to hour_in1) :

Used to set the 2-bit input for the most significant hour digit of the clock (if load_time=1), or the most significant hour digit of the alarm (if load_alarm=1). Valid values are 0 to 2.

e. Switch 15 (connected to Alarm_ON) :

If HIGH, the alarm is ON (and Alarm will go off if the alarm time equals the clock time). If low then the alarm function is OFF.



Alarm_ON **hour_in1 (2 bits)** **hour_in0 (4 bits)** **minute_in1 (4 bits)** **minute_in0 (4 bits)**

Fig. 2 Distribution of the respective input switches on the FPGA

f. LEDs 0-5 (connected to seconds):

Displays the seconds in binary notation. It displays seconds from 0 to 59, after which it resets to 0 and starts over. Hence, LEDs 0-5 glow according to the current second in binary.

g. LED 15 (connected to Alarm) :

Glows when the Alarm sets off along with the Buzzer tone.

h. Center Push Button (connected to reset) :

Resets the clock i.e. sets alarm time and clock time to 24:00 and 00:00 respectively.

i. Right Push Button (connected to load_alarm) :

Loads the alarm time input via switches onto the clock.

j. Left Push Button (connected to load_time) :

Loads the clock time input via switches onto the clock.

k. Up Push Button (connected to STOP_alarm) :

To bring the alarm signal to LOW which causes the buzzer to stop ringing.

l. Piezo Buzzer (connected to sound) :

Connected to the FPGA board through a breadboard and connecting wires. It goes off when the Alarm is high.

Fig. 3 The Piezo Buzzer used

4. RESULT:

We need to first set the hour and minute through the switches and then click on the Left push button to load time. The clock will then display the current time on the display of the FPGA. Every 60 seconds, one minute is incremented in the display and hence clock time moves ahead. We then set the hour and minute of the alarm time through the switches and click on the Right push button to set the alarm time. Now, when the current time coincides with the set-alarm time, the alarm signal goes high and the LED 15 on the FPGA board glows along with the Piezo buzzer setting off. We can then use the Center push button to reset the alarm time to 24:00 and clock time to 00:00. The Up push button can be used to stop the alarm. Following is the design flow of the implementation of Alarm Clock on the FPGA board using Vivado software:

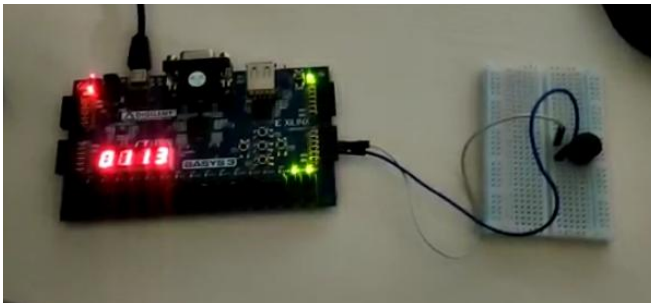


Fig. 4 24-hour Alarm Clock on FPGA with Piezo Buzzer