Digital Alarm Clock.

Maichel Salieb, Mario Ghaly, Youssif Abuzeid

Department of Computer Science and Engineering,

The American University in Cairo

Fall 2022

Abstract: Our project is a Digital alarm clock. The main goal of our project is to implement this digital alarm clock using Verilog HDL. We built this project on Basys 3 FPGA Board. The one we used in our project was Verilog HDL. We started our implmentation using a Circuit diagram and Finite state machines to have a full picture for the project. Then we began implementing the project in Verilog HDL. It made implementation much easier and effecient. This report will cover the methodology of the project, finite state machines, main modules, recommendations, and contribution of each team member.

Keywords: Digital Alarm Clock, Finite state machines, Mealy , Moore, Push Buttons, Buzzer, seven segment display .

1 Introduction

Logic Design has been an integral field in the study of computer science. It formed the basis of the development we witness today. One major aspect in logic design implementation technologies of logic functions. In the past, people used to implement logic functions using hardware directly. However, one major disadvantage of implementation using hardware directly is that it was both time and money consuming. Also, it was full of mistakes and hard to trace. One solution was to increase the level of abstraction to minimize problems. The greatest abstraction we computer scientists achieved is using a hardware description language to overcome these disadvantages. The most famous hardware description language is Verilog. However, in the process of designing many other techniques should be used such as Finite state machines. In this project, we integrated all of these techniques to implement the Digital Alarm Clock.

2 Problem Definition

Our project is to implement a Digital Alarm clock. This digital alarm clock should be implemented on a FPGA Board. It should have two modes. The first mode is the clock mode. In this mode, the clock should work normally and when the alarm time matches the clock time, a buzzer should generate a ringtone and LED0 in the FPGA Board should blink. To indicate the clock mode, LED0 in the FPGA should be off. Also, and the second decimal point in the 7 segment display should be blinking. The second mode is the adjust mode. In this mode, the user can adjust one of four options: Clock Hours, Clock Minutes, Alarm Hours, Alarm Minutes. The user can navigate between these variables using the left and the right push buttons of the FPGA. If the user wants to increment the selected variable, he should press the upper Push Button, and if he wants to decrement, he should press the lower push button. To indicate the presence of the adjust mode, LED0 should be On and the second decimal point should stop blinking. The default mode is the clock mode and the user can navigate between the two modes using the central push button of the FPGA board.

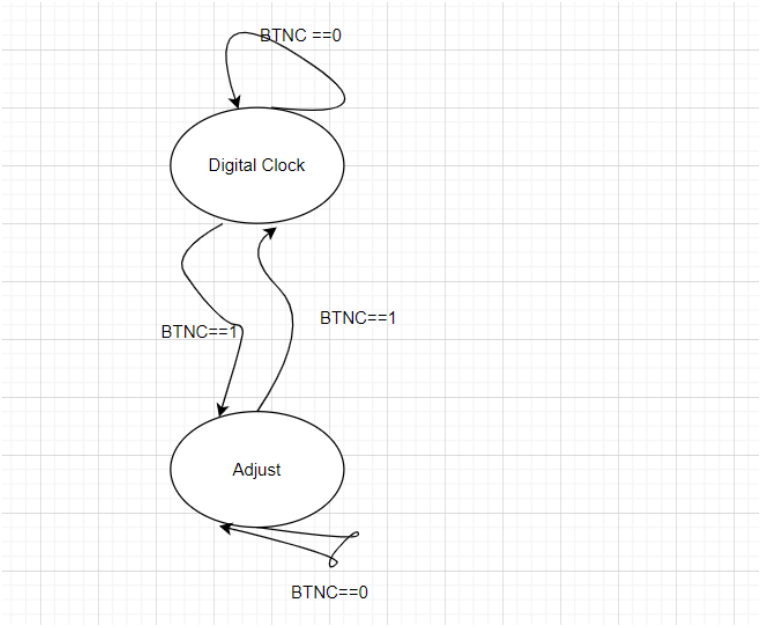
3 Methodology

The methodology of our project is composed of two main parts, which are: the Finite state machines, and the main modules in the project..

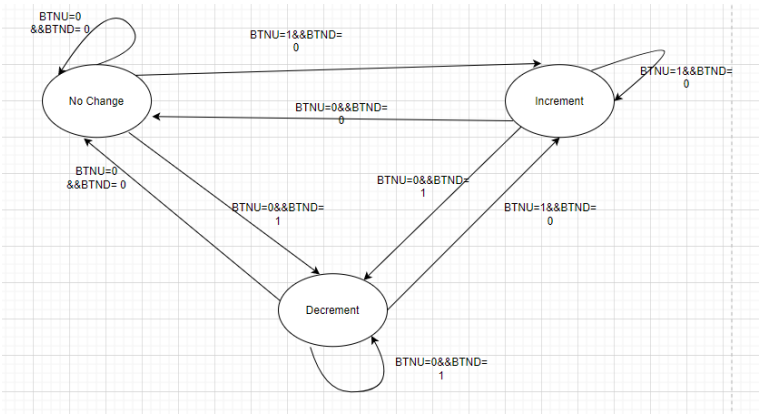
Finite state machines: Finite state machines in one of the most important tools that help in the implementation process. It helps in understand and simulating the

behavior of the circuits. Also, it helps in specifying the main requirements of the circuit and its interaction to any of the inputs at any point of execution. Finite state machines are composed of finite number of states. Each state describes a certain point in the program. The sequence of states is determined by the inputs to the circuit It has two main types. The first type is the Mealy finite state machine. In this type the output of the circuit is mainly determined by the inputs to the program and the current state. The other type is the Moore FSM. In this finite state machine, the output is dependent only on the state. In our project we, depended mainly on the Moore FSM. Below is a list of all the finite state machines we used in the project.

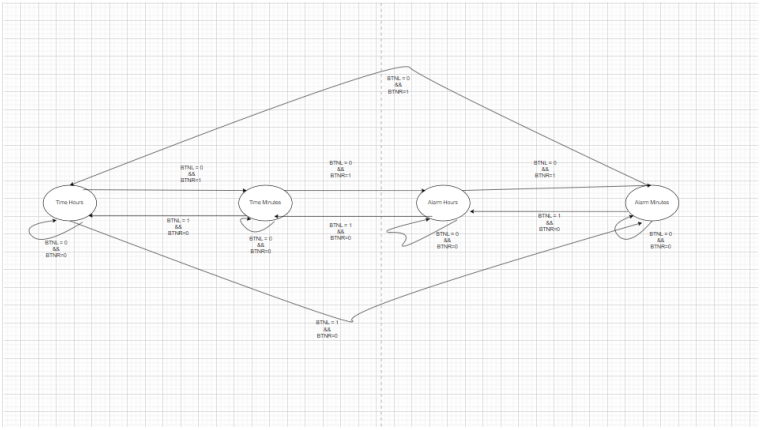
1. **The Mode FSM**



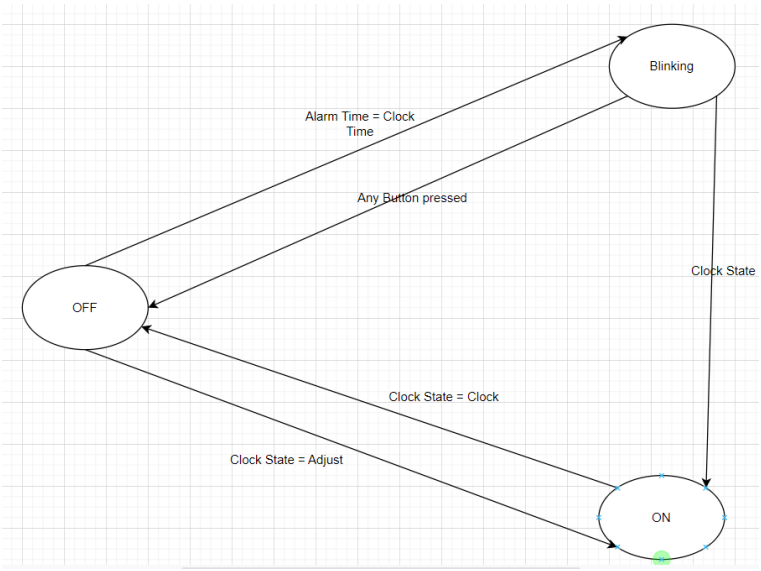
2. **Increment / Decrement FSM**



3. **The Adjusted Vriable FSM**



4. **LED0 FSM**


Blinking
Alarm Time = Clock Time
Any Button pressed
Clock State
OFF
Clock State = Clock
Clock State = Adjust
ON

4 Modules in the Project:

First of all, we divided the modules of the project into main types. The first type is the main modules of the project. The second type if the helper modules in the project.

Main Modules:

- Clock Module.
  - Description: This module is responsible for the clock. If is the circuit in the clock mode, the module will work as a normal clock. First, it will divide the clock of the FPGA to match the frequency of the normal clock. Also, it has 6 counter which are: 2 counters for the seconds, one for each bit, 2 counters for the minutes, 2 counters for the hours. In the adjust mode, if the user selects either of the minutes or the hours of the clock, the counters of either of them will count up or down based on the selection of the user to allow him to adjust the clock.
  - Inputs
    1. Clock: Responsible for getting the clock of the FPGA.
    2. Reset: Responsible for resetting the clock back to zero.
    3. UpDown: Responsible for detecting whether the clock will count up or down in the adjust mode. In the clock Mode, this input will always be zero or up.
    4. En: Responsible for detecting whether the clock will count normally, if it is in the clock mode, or it will not count normally if it is in the adjust mode.
    5. EnMin: This input detects that the user is in the adjust mode and wants to adjust the minutes of the clock.
    6. EnHours: This input detects that the user is in the adjust mode and wants to adjust the hours of the clock.
  - Outputs
    1. OutSeconds0: The first digit of the seconds.
    2. OutSeconds1: The second digit of the seconds.
    3. OutMinutes0: The first digit of the minutes.'
    4. OutMinutes1: The second digit of the minutes.
    5. OutHours0: The first digit of the hours.
    6. OutHours1: The second digit of the hours.
- Alarm Module.

- Description: This module is responsible for the alarm. The module consist of four counters, 2 for the minutes and two for the hours. If is the circuit in the clock mode, the counters will stop working to keep the values of alarm constant. In the adjust mode, if the user selects either of the minutes or the hours of the alarm, the counters of either of them will count up or down based on the selection of the user to allow him to adjust the alarm.
- Inputs
  1. Clock: Responsible for getting the clock of the FPGA.
  2. Reset: Responsible for resetting the alarm back to zero.
  3. UpDown: Responsible for detecting whether the alarm will count up or down in the adjust mode.
  4. EnMin: This input detects that the user is in the adjust mode and wants to adjust the minutes of the alarm.
  5. EnHours: This input detects that the user is in the adjust mode and wants to adjust the hours of the alarm.
- Outputs
  1. Minutes0: The first digit of the minutes.'
  2. Minutes1: The second digit of the minutes.
  3. Hours0: The first digit of the hours.
  4. Hours1: The second digit of the hours.

- Main Module.
  - Description: This module is responsible for integrating the different parts of the project together. It contains the implementation of the different finite state machines. It contains instantiations of the many modules such as the alarm, the clock, the push button detectors, seven segment displays, etc. It controls the mode and the working of the alarm and the clock modules based on the inputs to the push buttons. Also, it does the scanning of the seven segment display and selects what is displayed on the seven segments display. Also, it controls LED0 and the second decimal point.
  - Inputs
    1. Clock: Responsible for getting the clock of the FPGA.
    2. Reset: Responsible for resetting the clock and the alarm back to zero.
    3. Central Button: Responsible for detecting whether the central button (BTNC) is pressed or not.
    4. RL: This input is a vector which stores the values of the right and the left push buttons.

5. UD: This input is a vector which stores the values of the up and the down push buttons.

– Outputs

1. 7segment Enable:A vector which is responsible for the enables of the 4 seven segment displays.
2. 7segment display: A vector containing the output that should appear in the seven segment display.
3. LEDS: a vector containing the output for 4 LEDS that indicates the variable to be adjusted in the adjust mode.
4. CCC: The output to LED0 which detects whether this LED should be ON, blink or OFF.
5. Pin: The output to the second decimal point.

- **Helper Modules:** These modules are not the main modules in the project but they are essential to the project. These modules are: clock divider, positive edge detector, synchronize, debouncer, push button detector, binary counters.

5 Testing & User Ineraction.

The project is mainly a clock/ alarm with two modes, the display mode and the adjust mode.
The display mode is only specified for the clock, where the user can use it as a clock for him to see the time.
The adjust mode is applied in two scenarios, where you can adjust the clock or the alarm. The initial state that the program starts with is showing the digital clock, but when you click on the central push button, you move from the display mode to the adjust mode, and there is a led that gets high, indicating that you are in the adjust mode. While in the adjust mode, you have four options to choose which one you want to adjust. The four options are time hour, time minutes, alarm hours, and alarm minutes, and the initial state that you will be changing when you enter the adjust mode is the time hour, and you will know that because there is a led for each one of the four options that indicates what are you changing. Changing the value of each one of the four options is done by the up and down push buttons, where you change the value of hours from 0 to 23 and the value of minutes from 0 to 59. After that, you need to set the alarm for a value and return to the clock state again, where the digital clock is displayed until the clock becomes equal to the time you adjusted in the alarm. When this happens, the LED indicating that you are in the adjust mode starts to blink, indicating that the alarm time is equal to the clock time; accompanied by this, the buzzer starts to generate a tone. Clicking on any button will stop the

blinking of the led and also stops the buzzer. For resetting the clock/alarm, there is a switch that returns all the values to zeroes again.

6 Analysis and Problems in the program

The project matches all the requirements which are stated in the description. It switches between the two modes easily and smoothly. Also, the user is capable for adjusting the clock and the alarm. However, there is still a room for development. For instance, the project could be modified to add the capability of having multiple alarms at the same time. Also, the user can select the ring tone of the alarm.

7 Contributions of every team member:

We started by making a GitHub repository for saving our work. We did not work in parallel, so we did not need to work on different branches. Our contribution to the project was split as follows:

1. *Maichel*: He created the first draft of the module for the digital clock, then we all needed to work on the module since it had a lot of edge cases, especially in the hours' counters. Thus, we worked together and had to adjust the module of the counters multiple times, and added parameters to it. At this level, we made sure that the clock in the clock mode is working properly. Furthermore, Michael was responsible for writing the FSM of up-down push buttons.

2. *Youssif*: he started by creating the GitHub repository and created the modules for the clock dividers, counters, rising edge detector, debouncer, synchronizer, push button detectors, and BCD to 7-segment display. He used the past modules used in the lab. However, there were some mistakes that he needed to debug. Moreover, he wrote the FSM code for the clock mode and the blinking mode.

3. *Mario*: He created the first draft of the module for the digital clock, then we all needed to work on the module since it had a lot of edge cases, especially in the hours' counters. Thus, we worked together and had to adjust the module of the counters multiple times, and added parameters to it. At this level, we made sure that the clock in the clock mode is working properly. Furthermore, Michael was responsible for writing the FSM of up-down push buttons.

4. *All of us*: After making sure that everyone did his part right, and that everything is working efficiently on its own, we sat together to write the main module where we run the program itself. Moreover, we worked on debugging

the frequencies of the clock dividers and the FSM since they were erroneous in some cases.