Digital Alarm Clock: Project Report

Group 2

Ahmed Bamadhaf 900205060 (section 1)

Adham Hassan 900212942 (section 2)

Maha Shakshuki 900225906 (section 2)

Youssef Aboushady 900223372 (section 1)

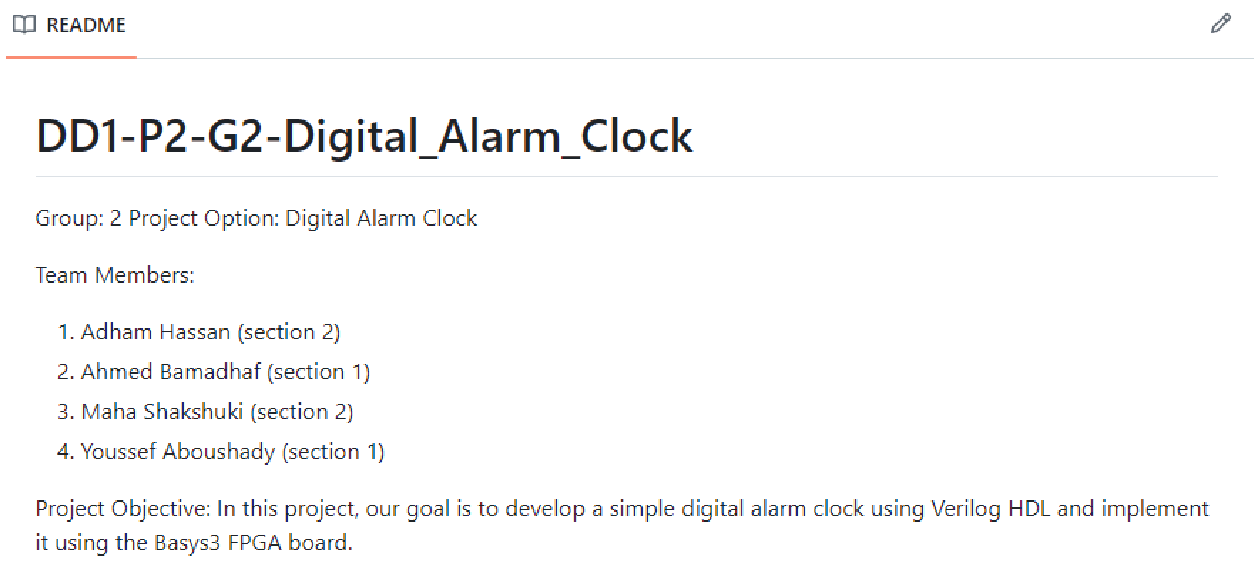
Department of Computer Science and Engineering, The American University in Cairo

CSCE 2301

Dr. Mohamed Shalan

May 18, 2024

Digital Alarm Clock: Project Report



DD1-P2-G2-Digital_Alarm_Clock

Group: 2 Project Option: Digital Alarm Clock

Team Members:

1. Adham Hassan (section 2)
2. Ahmed Bamadhaf (section 1)
3. Maha Shakshuki (section 2)
4. Youssef Aboushady (section 1)

Project Objective: In this project, our goal is to develop a simple digital alarm clock using Verilog HDL and implement it using the Basys3 FPGA board.

<u>Outlining the design,</u>

We have created in Verilog 12 modules.

- Module 1: clock_1
  - This module is responsible for combing clock division, counting, and control for a 4-digit 7-segment displayer. It has as inputs: clk for primary clock input, reset for signal to initialize, or reset for the system. For its outputs: segments for A 7-bit output controlling the segments of a 7-segment display (a-g), anod_active for 4-bit output controlling which digit is going to display. And decimal
- Module 2: Binary counter
  - The Binary Counter module increments a binary count on each clock pulse. It provides a straightforward mechanism for timekeeping or counting applications where a binary representation of the count is required. Inputs include `clk` for clock input, `reset` for resetting the count, and optionally `enable` for controlling counting activity. Output `count` presents the current binary count value.

- Module 3:Manual counter
    - The Manual Counter module allows manual adjustment of the count value using push buttons. It enables users to set the time or other parameters directly, offering flexibility besides the automatic counting method. Inputs include `clk` for timing, `reset` for resetting the count, `increment` for increasing the count, and `decrement` for decreasing it. Output `count` reflects the current adjusted value.
- Module 4: Pushbutton detector
    - This module detects the state changes of push buttons used for user inputs. It generates signals indicating button presses, facilitating user interaction with the clock system. It ensures accurate detection and interpretation of user commands, such as setting time or activating specific functions.
- Module 5: Debouncer :
    - We used the debouncers to clean up the signal from the button. When you press a button, it can bounce a bit and send multiple signals instead of just one. The debouncer makes sure we only get a single, clean signal when the button is pressed.
    - We used debouncers with every push button
- Module 6: Decrementer
    - The Decrementer module decrements a count value, used for the countdown for the alarm. It provides functionality to decrease the count manually or in response to specific triggers, supporting time management functionalities in the clock system. This module also has the incrementer
- Module 7: Rising Edge
    - The Rising Edge Detector module identifies the rising edge of input signals, converting continuous signal changes into single-pulse events. It detects and processes signal transitions from low to high, facilitating precise timing and synchronization within the clock system. This capability ensures accurate triggering of actions based on signal changes.
    -
- Module 8: Adjust Hours
    - This module manages adjustments to the hour displayed on the clock. It allows users to increment or decrement the hour setting manually, typically controlled via push buttons or other input methods. This functionality ensures accurate time setting and alignment with user preferences.
- Module 9 : Adjust minutes
    - Similar to Adjust Hours, this module handles adjustments to the minute value displayed on the clock. It enables users to increment or decrement the minutes

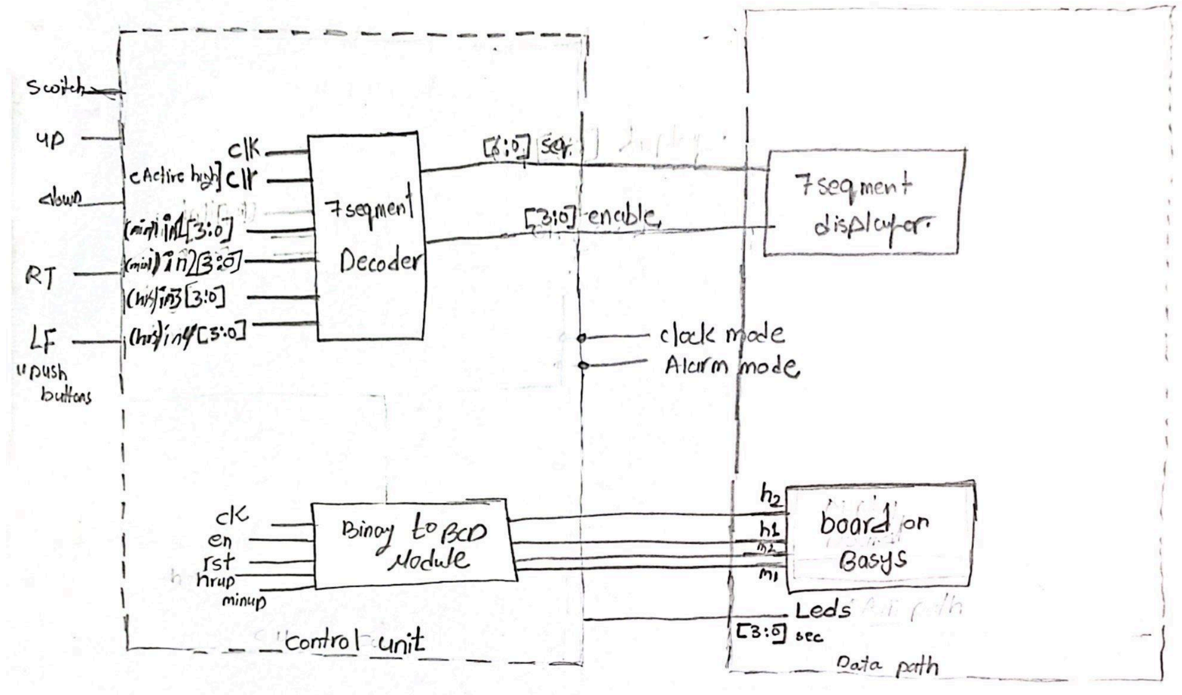
manually, providing precise control over time setting down to the minute level. User inputs trigger adjustments, ensuring accurate time representation.

- Module 10: Adjust mode
    - The Adjust Mode module governs the operational mode of the clock system, switching between functionalities such as time setting, alarm setting, or states. It controls the behavior and settings of the clock. We can also increase and decrease the time and modify the alarm through the push buttons.
    -
- Module 10: Synchronizer
    - We used a synchronizer to make sure signals are safely passed between different clock domains. If parts of the circuit run on different clocks, the synchronizer helps to avoid errors by ensuring the signal is stable and aligned with the local clock.
    - We used Synchronizer with every push button

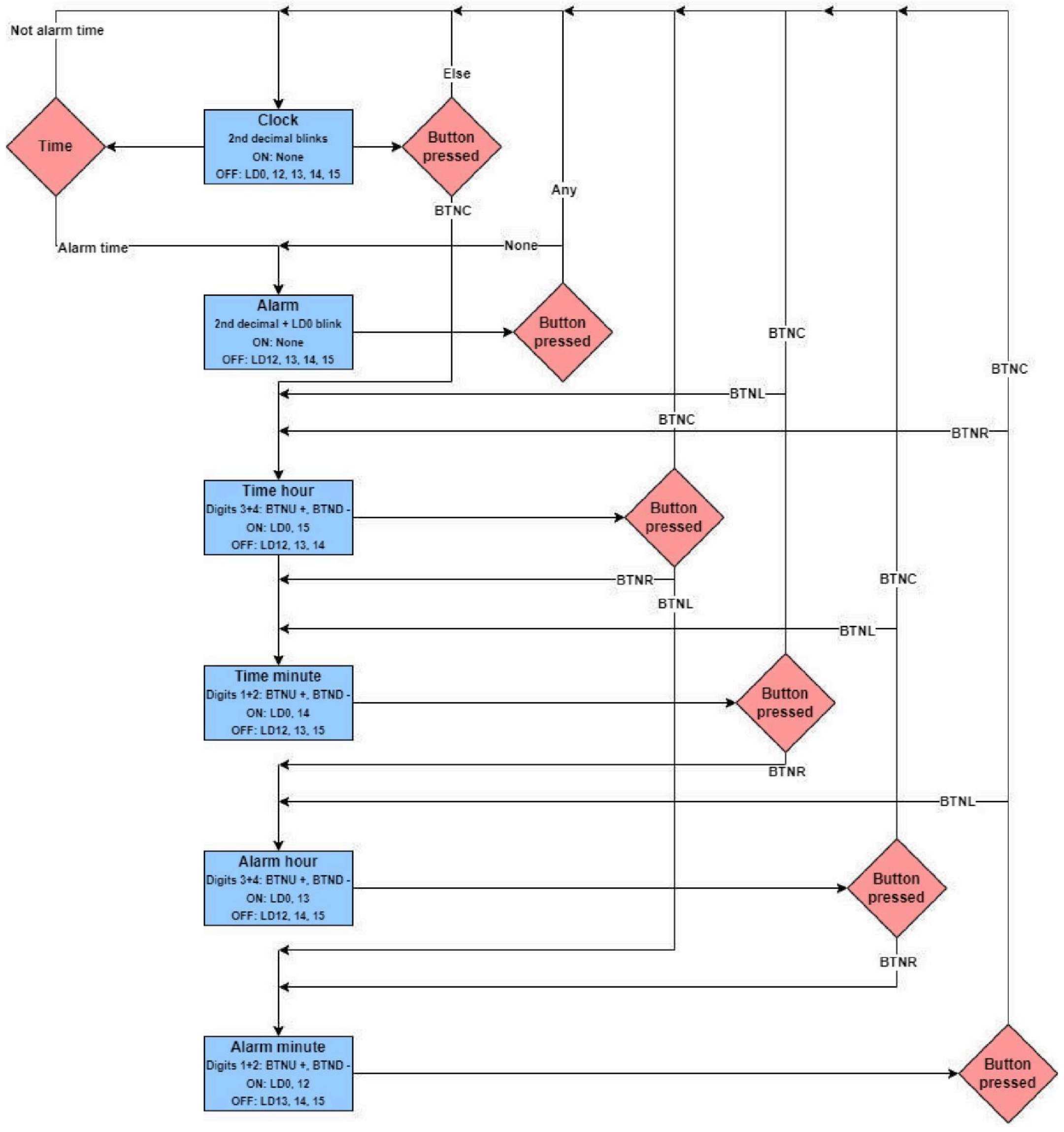
<u>Validation activities</u>

#DP and CU :

The aim behind DP and CU is to show the different components that the clock has such as push buttons, 7-segment displayers, LEDs, and the two modes(Clock/Alarm, Adjust) in order to show their functional units. By starting with this diagram, we were able to brainstorm and test how the code was going to work and how to design our functions in the code.



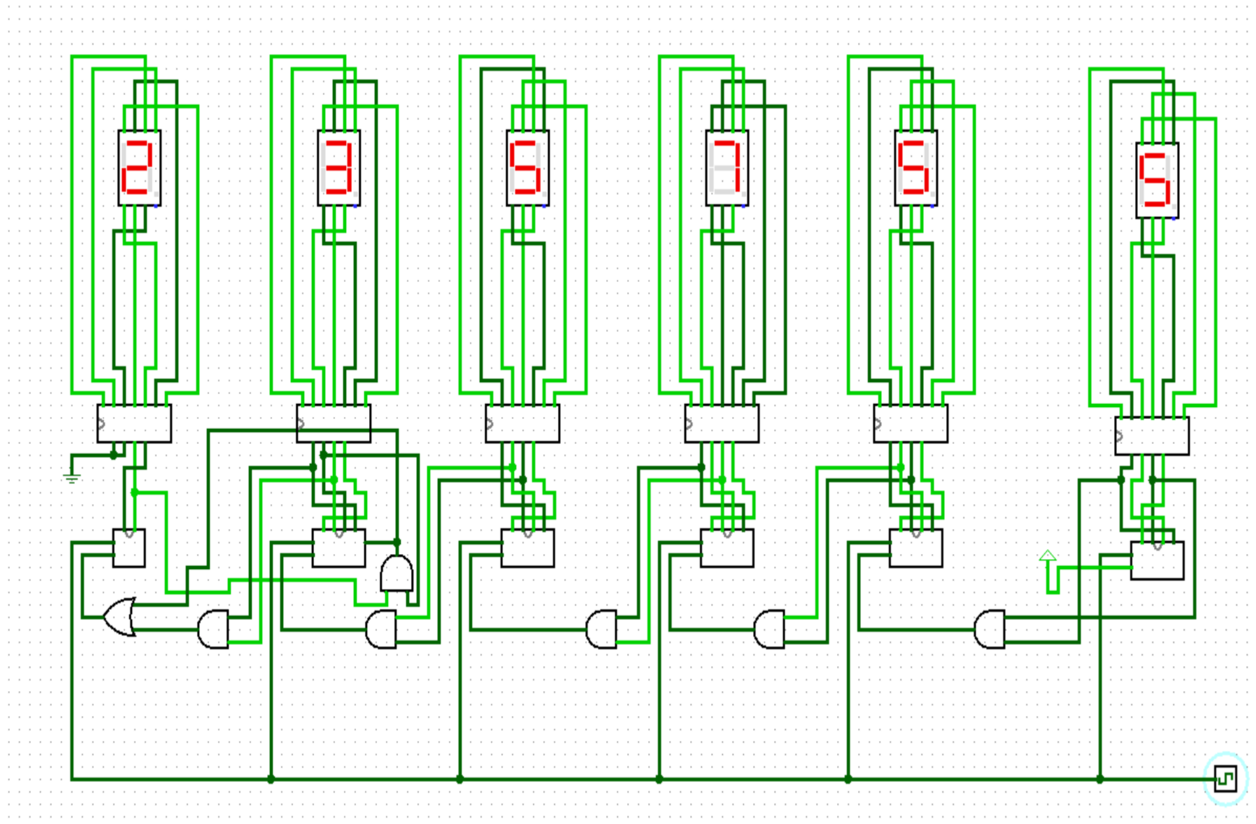
#FSM:
The finite state machine chart helped us imagine how the states are going to be displayed in the digital alarm clock and therefore, we were easily able to convert the chart into Verilog code to design them as Moore machine.



#Logisim:

We have designed the seven segments display with the time to show how the digital clock works. We designed it in a hierarchy shape so it appears as a digital clock, neat and tidy. We made a mistake creating the circuit without the multiplexer which does not align with how the FBGA, Basys 3 works. The way we have designed the clock in Logisom by making mini circuits and then combining them in the Main.

- The circuits we have designed had several modules. Synchronous counter divided by 10 (modulo 10) : This is a typical synchronous 4-bit counter utilizing 4 JK flip-flops. Normally, this counter stops at the number 15.
- Synchronous counter divided by 6 (modulo 6) : This is a typical synchronous 4-bit counter utilizing 4 JK flip-flops. Normally, this counter stops at the number 5. Used for the second digit of the seconds and minutes.
- Synchronous counter divided by 2 (modulo 2) : 2-bit counter using JK flip flops and we want the counter to count only from 0 to 2.
- Synchronous counter divides input : This is basically the same as explained in counter_div_10, it is a synchronous counter div 10, but it can also be a divider for another number which is when the clock' hours become more than 20 hours, we have to rest the clock at 23 hours 59 minutes and 59 seconds to 0 hours 0 minutes and 0 seconds.



#Verilog part :

The project went through many tests to produce the final version of the digital alarm clock.
 In the early process, we created the digital clock code based on the lab guidelines, and we tested it to see if it worked probably in order to start with generating and writing the new requirements.
After testing the digital clock code, we started initializing the time hour and time minutes by code and displaying it on the 7-segment displayer, we countered many errors regarding displaying the hour/min timer and it turned out, that there were missing assigned variables and it was solved.
After making sure from the hour/min counter, we started working on the FSM code and started to run the cases based on the FSM chart that we created in phase one and make sure that it passed the states correctly as we used the Moore approach.
After making the states, we started creating 5 pushbutton detectors which are up, down, right, left, and center and each one is specified to control certain parts as the project guidelines required.
We began adjusting the code accordingly to work on the clock states which are 5 states (time minutes, time hours, alarm hour, alarm minute, alarm clock), and started testing it with the assigned pushbuttons.
We have created points to control the point that blinks while displaying the digits , we faced some problems regrading the blinking part and we tried to solve it.
For the adjust mode , we first tried to control the minute and hours , then we linked it to the led0 so we can know that we are in adjust mode to fix the hour mode and to control it from the push button (center) one.

Implementation issues

(Hardware)

- Regarding the FPGA , we faced various challenges :
    - First thing, when we implemented the time hour/minute code, and we tested it on the board, we had an error in which the wrong digits were displaying and we solved it by reassigning the variables in the code.
    - We did not have the buzzer to try the alarm.

(Software)

- Syntax errors occurred sometimes
- Generating bitstream failed due to errors in the constrain
- Building the constrain files was kind of an issue because we faced unknown errors and we needed to spend a lot of time finding the solutions.
- We have made a mistake by creating different modules in the project. That was not very efficient because the problem we faced later was how to pair up the different modules.
- We did not add the enables in the binary counters at the beginning, and then we had some guidance from the TA.
- The blinking point (second decimal) was not blinking and whenever we wanted it to blink, we faced the problem of either it blinks only one time, or the four decimal points blink. We tackled that problem by connecting it with the state where the Active anode is only on, on the second decimal..
- The push buttons had some issues. First, when we increase, both hours and minutes increased together. After trying to debug that problem, we faced a problem of only one digit increases in the fourth digit from the right.
- We faced a problem when we tried to decrease the number, as we could not have both digits in the minutes counter decreasing. It was only, also, the fourth digit.
- The switch that was responsible for resetting the clock was alright.

Contributions of Each Member

Ahmed Amer Bamadhaf

- I have designed the digital clock on Logisim making sure it works for all of the seconds, minutes, and hours. I have elaborated on how I created the circuit above.
- Contributed to the final code
- Participated in the report.

Maha Shakshuki

- I designed the block diagram for the Digital clock/ Alarm, to show exactly how the data path and control units work. I included two blocks, the first block is for the control unit which has the push buttons, and switch key, and the 7-segment decoder which has the digits and binary to bcd converter, and the data path contains the 7-segment displayer and the LEDs.
- Contributed to the final code
- Participated in the report.

Youssef Aboushady

- I was responsible for the Algorithmic State Machine (ASM) chart
- Contributed to the final code
- Participated in the report.

Adham Hassan

- Participated in the report
- Contributed to the final code

● All the work is here : https://github.com/YoussefAboushady/DD1-P2-G2-Digital_Alarm_Clock