Hadi Hesham (900221206), Farida Islam (900211819),
Ismail ElDahshan (900221719)

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

CSCE 2301: Digital Design 1

Digital Alarm Clock (Project Report)

Dr. Mohamed Shaalan

May 20th, 2024

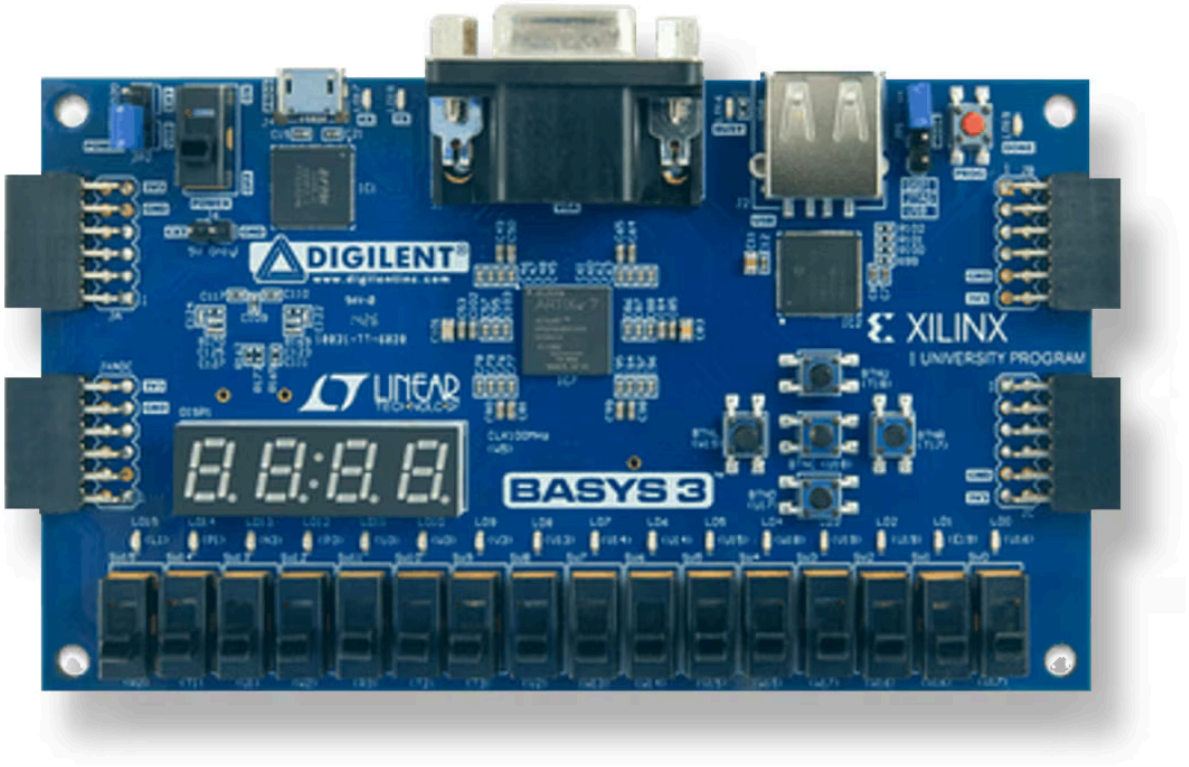
Table of contents:

Project Description:

This project involves the design and implementation of a simple digital alarm clock using an FPGA, specifically the BASYS3 board. The project aims to utilize various BASYS3 peripherals, including LEDs, a 7-segment display, and push buttons, to create a functional digital clock with alarm capabilities. It should meet the following specifications:

The digital alarm clock must utilize the following BASYS3 Peripherals:

- LEDs LD0, LD12, LD13, LD14 and LD15
- 7-Segment Display: Used to display hours (left two digits) and minutes (right two digits)
- Push Buttons: BTNC, BTNL, BTNR, BTNU, BTND



The digital alarm clock must have 2 modes:

- Clock/Alarm Mode (Default Mode) with the following attributes:
    - LD0 is OFF
    - The second decimal point from the left blinks at 1Hz
    - When the current time matches the set alarm time, LD0 blinks; pressing any button stops the blinking
- Adjust Mode with the following attributes:
    - Activated by pressing BTNC
    - LD0 is ON
    - The second decimal point from the left does not blink
    - Pressing BTNC exits Adjust mode and returns to Clock/Alarm mode
    - Parameter Selection:
    - Pressing BTNL or BTNR cycles through "time hour", "time minute", "alarm hour", and "alarm minute" (in sequence)
    - LD12, LD13, LD14, and LD15 indicate the selected parameter for adjustment
    - Pressing BTNU increments the selected parameter
    - Pressing BTND decrements the selected parameter

Conclusion

This project integrates hardware design principles with practical FPGA implementation, resulting in a functional digital alarm clock. By adhering to the outlined requirements, the project aims to develop a reliable and user-friendly digital clock with alarm functionality.

Code Overview:

The implemented code has 10 verilog modules. ButtonDetector, RisingEdgeDetector, Synchroniser, UpDown, alarm, clkdivider, counter, debouncer, hrsmin and display. The following is the explanation of each module.

- ButtonDetector: This module handles the input from push buttons to ensure reliable detection of button presses. It consists of three submodules: “debouncer” that eliminates the noise caused by mechanical bouncing of the button, providing a stable signal, “Synchroniser” that ensures the debounced signal is synchronized with the clock domain, avoiding metastability, and “RisingEdgeDetector” that detects the rising edge of the synchronized signal, indicating a button press.
- RisingEdgeDetector: This module detects the transition of an input signal from low to high (rising edge). It uses a finite state machine with three states: “a”, “b”, and “c”. The state transitions occur based on the input signal “x”, and the output “z” is high only during the transition. The state machine transitions between states based on the input signal. When a rising edge is detected, the output “z” goes high. On each positive clock edge, the state is updated based on the current state and input signal.
- Synchroniser: This module ensures that the asynchronous input signal “SIG” is properly synchronized with the clock signal “clk”. This is achieved using two flip-flops in series to prevent metastability.
- UpDown: This module implements an up/down counter with parameters for the bit width “n” and the modulus “mod”. It supports enable, load, increment, and decrement operations. Depending on the control signals, the counter can increment, decrement, or load a specific value and on reset, the counter is set to zero.
- Alarm: This module handles the alarm functionality, including setting and keeping track of alarm hours and minutes. Two instances of module “binaryCounter” are used for managing minutes and hours. The counters are updated based on the input signals for hours and minutes.
- clockDivider: This module divides the input clock frequency to generate a slower clock signal (“clk_out”). The division factor is determined by the parameter “n”. A counter increments on each clock cycle and toggles the output clock signal when it reaches the specified count. On reset, the counter is reset to zero.
- binaryCounter: This module is a generic counter that can count up or down based on the control signal “updown”. The counter increments or decrements based on it. It supports enable and reset operations. The counter has boundary conditions such as a maximum and a minimum value.
- Debouncer: This module eliminates noise from an input signal due to mechanical bouncing of switches. A three-stage shift register is used to stabilize the input signal and the output is high only when all three stages hold a high value.

- Hrsmin: This module manages the main clock functionality, including seconds, minutes, and hours. Three instances of “binaryCounter” are used for managing seconds, minutes, and hours. Enable signals are generated based on the current count of seconds and minutes to increment minutes and hours appropriately.
- Display: This module handles the seven-segment display logic and the state machine for different modes (clock, time set, alarm set). Two instances of “clockDivider” are used to generate different clock signals for display and state machine timing. Five instances of “ButtonDetector” handle input from push buttons for mode and value adjustments. A state machine manages transitions between displaying the clock, setting the time, and setting the alarm. The display logic updates the segments based on the current time or alarm values. The alarm is activated when the current time matches the alarm time, indicated by a buzzing sound and blinking LEDs.

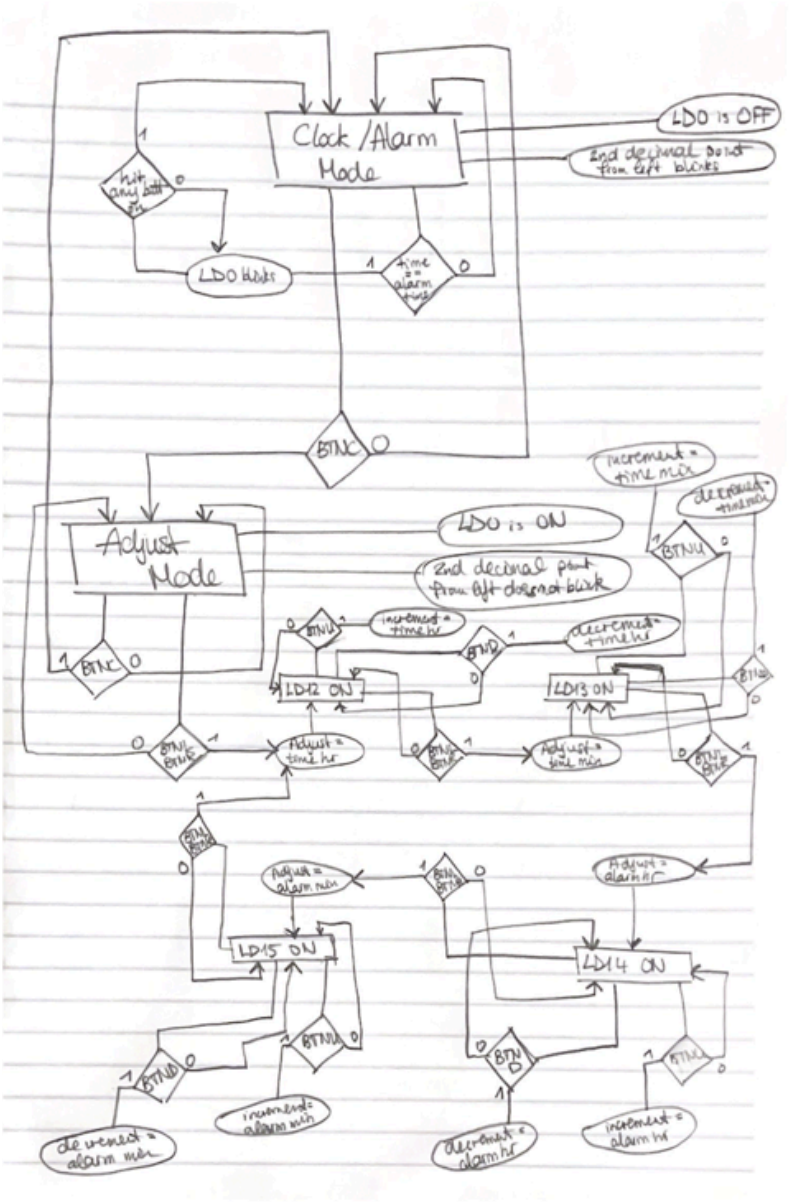
This code provides a comprehensive implementation of a digital alarm clock with functionalities for setting and displaying time, handling alarms, and debouncing button inputs.

Implementation Issues:

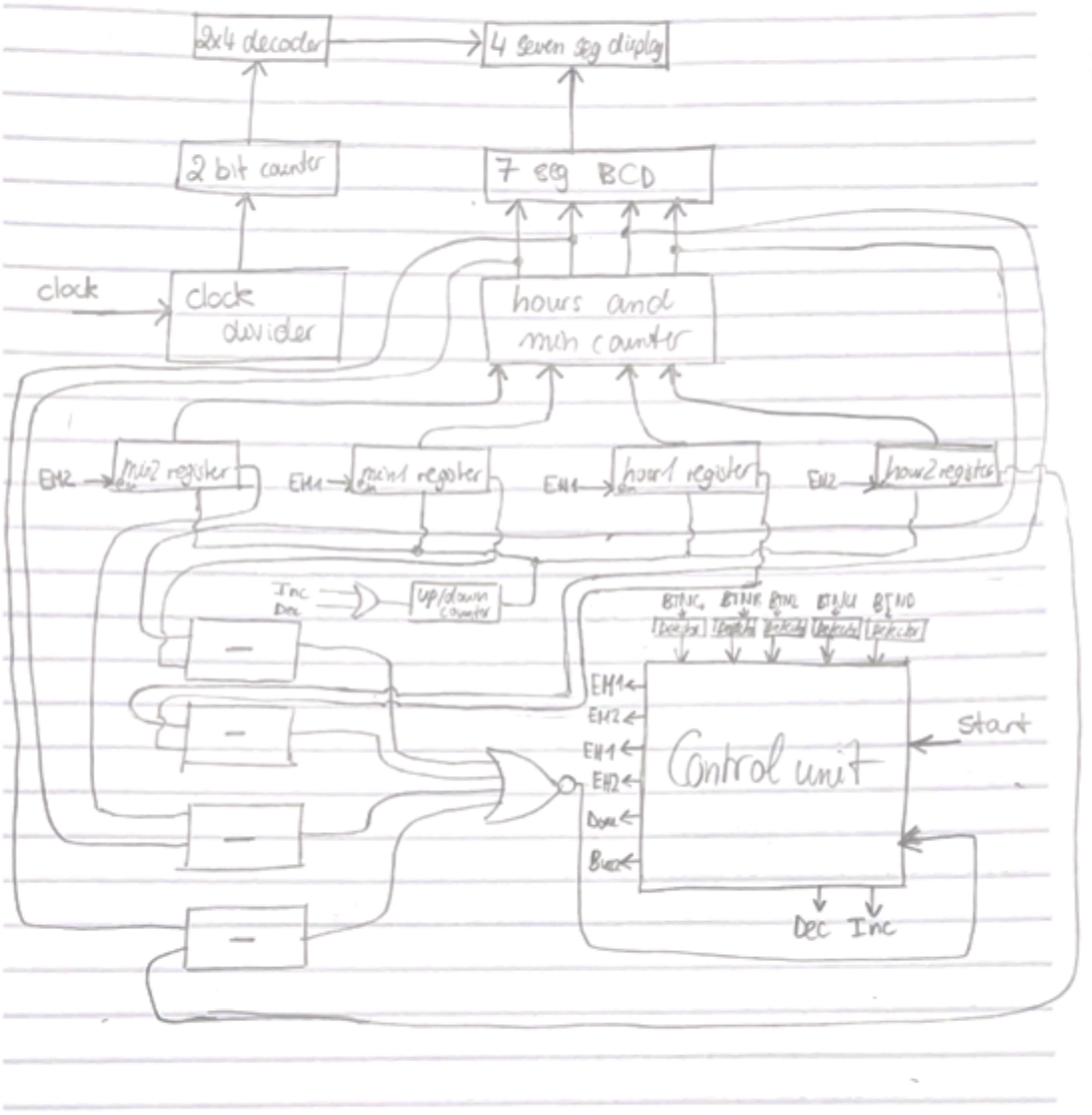
During the development of the digital alarm clock, we encountered several challenges that required troubleshooting and problem-solving. Initially, the clock failed to count, which was traced back to synchronization issues within the counter logic. Additionally, we observed that the clock and adjust modes were counting simultaneously, causing conflicts and erroneous time adjustments. This was resolved by refining the state transition logic to ensure mutual exclusivity between modes. The push buttons posed another challenge as they were unresponsive due to debounce issues caused by frequency mismatches, which were mitigated by passing the up/down module. Furthermore, we experienced difficulty in switching out of the alarm mode, necessitating a review and correction of the state machine transitions by writing all “alarm mode” conditions inside a state specifically for this mode, adding a conditional statement that will output the clock mode if a button has been pressed. Lastly, incrementing the hours or minutes was not functioning correctly, which we identified as an issue with the button press detection logic and subsequently fixed by fine-tuning the signal processing algorithms. Amidst all of that, our biggest challenge was facing random incrementing and random decrementing of time hr, time min, alarm hr and alarm min. The way we handled it was by reassessing our Mealy state machine and deciding that for every input, we’re going to state all possible outputs. Each of these obstacles provided valuable learning experiences and improved our understanding of FPGA-based design.

Diagrams:

ASM chart:

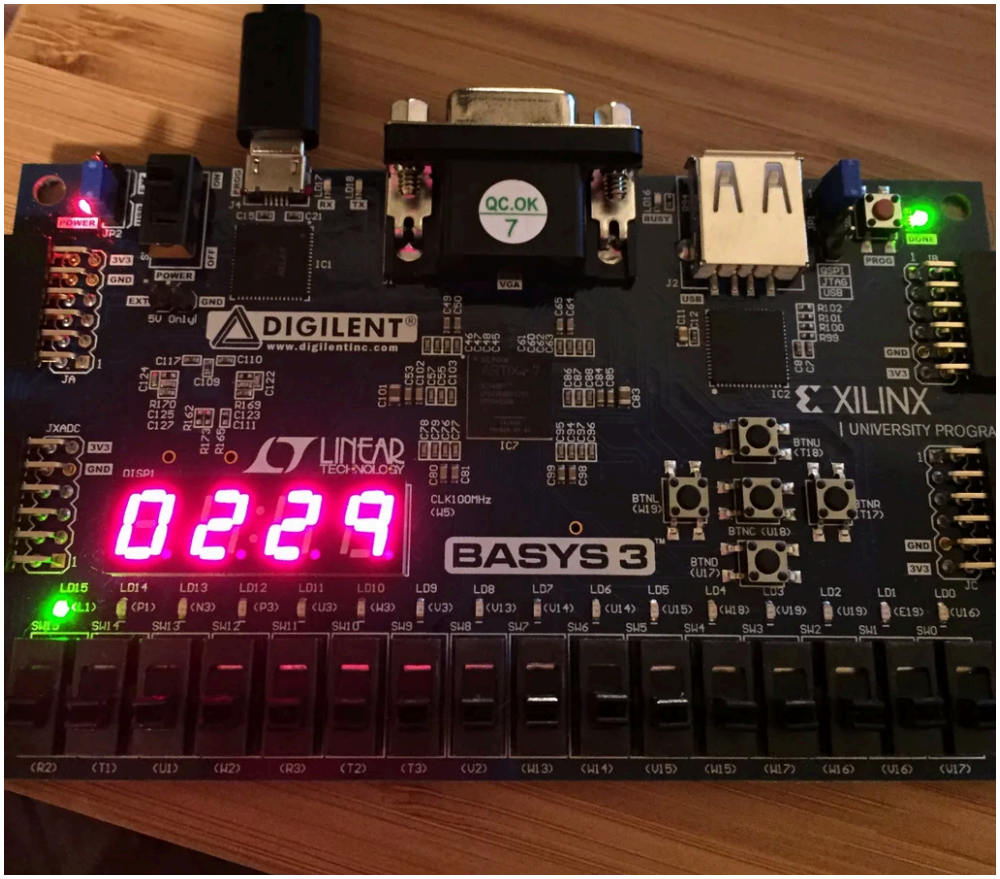


System Block Diagram:



Members' Contribution:

In this project, our team collaborated effectively to ensure the successful completion of our project, with each member contributing significantly to various aspects. Hadi focused on developing the code for the adjust mode and alarm mode, ensuring that users can easily set and manage alarms. Ismail took charge of the clock mode implementation and also created the Logisim files, which were crucial for simulating and testing our design. Farida was responsible for generating the diagrams that visually represent our system and compiling the comprehensive report that documents our work. Throughout the project, we consistently communicated and coordinated to make sure our individual contributions were integrated seamlessly. This collaborative effort ensured that our project not only met all requirements but also functioned reliably and as intended.



Project Retrospective:

Reflecting on our project, we are pleased to report that our teamwork was really good, marked by cooperation and mutual support. Throughout the project's duration, we encountered no conflicts or disputes with one another, which significantly contributed to our productive and positive working environment. Each team member was highly cooperative, readily offering help and sharing insights across all tasks, ensuring that everyone was supported and that the workload was evenly distributed. Despite the tight schedule, which added a layer of pressure, we managed to navigate through it effectively. This project was not easy, but through our combined efforts, effective communication, and shared commitment, we completed it successfully. We learned a lot from this experience, which has reinforced our belief in the power of collaboration and provided us with valuable lessons in teamwork and time management.