CSCE2301 - Digital Design 1

Digital Alarm

Names: Jana Elfeky, Kareem Elnaghy, Arwa Abdelkarim

Dr. Mohamed Shalan

1. Introduction

The Digital Alarm Clock project aims to implement a fully functional alarm clock on the BASYS3 FPGA board using Verilog HDL on Vivado software. The clock features time display, alarm functionality, and adjustable settings through the board's push buttons, LEDs and a buzzer. This project highlights fundamental digital design concepts, including counters and finite state machines (FSM).

Objectives

Time Display: Display the current time in hours and minutes on a 7-segment display.
Alarm Functionality: Allow setting of an alarm time, which activates a buzzer and a blinking LED when the set time is reached.
Adjustable Settings: Enable user adjustments for both the current time and the alarm time using push buttons.
Mode Operation: Switch between "clock/alarm" mode and "adjust" mode, with visual and audible indicators (LEDs and buzzer).

Components

BASYS3 FPGA Board: Primary Hardware Platform
7-Segment Display: Used for displaying hours and minutes.
LEDs (LD0, LD12, LD13, LD14, LD15): Indicators for different states and modes.
Push Buttons (BTNC, BTNU, BTND, BTNL, BTNR): Used for mode switching and adjusting time.

Operating Modes

The digital alarm clock operates in two distinct modes: “adjust” and “clock/alarm”. The default mode is “clock/alarm”, where the clock runs normally, displaying the time and monitoring for any matches with the set alarm time. In this mode, LD0 remains off, and the second decimal point from the left on the 7-segment display blinks at a frequency of 1Hz, indicating the clock is running. The transition between modes is user-driven, initiated by pressing BTNC to enter the “adjust” mode, where the user can modify the time or set the alarm using the push buttons BTNU, BTND, BTNL, and BTNR.

➢ ***Clock/Alarm Mode:***
- Default operating mode.
- LD0: Off.
- Decimal Point Blinking: The second decimal point from the left blinks at 1Hz to indicate the clock is running.
- Alarm Activation: When the current time matches the alarm time, LD0 blinks and the buzzer sounds until any button is pressed.

- ***Adjust Mode:***
  - Activated by pressing BTNC.
  - LD0: On.
  - Decimal Point: Does not blink.
  - Push buttons functionality:
    - BTNR: Used to switch between which parameter is being adjusted to the following parameter in the sequence:
      "alarm minute", "alarm hour", "time minute", "time hour", in a cyclic fashion.
    - BTNL: Used to switch between which parameter is being adjusted to the following parameter in the sequence:
      "time hour", "time minute", "alarm hour", "alarm minute" in a cyclic fashion.
    - BTNU: Used to increment the parameter being adjusted.
    - BTND: Used to decrement the parameter being adjusted.
  - LED Indicators:
    - LD12: Represents state in which the user sets the *time hour*.
    - LD13: Represents state in which the user sets the *time minute*.
    - LD14: Represents state in which the user sets the *alarm hour*.
    - LD15: Represents state in which the user sets the *alarm minute*.
  - Exit Adjust Mode: Press BTNC again to return to "clock/alarm" mode.

2. User Guide

- Hardware Setup
  - Power Up: Connect the BASYS3 board to a power source and turn it on.
  - Initial Display: The 7-segment display will show the time in HH:MM format, starting from 00:00 up to 23:59, and the clock will be in "clock/alarm" mode by default.

- Operating the Clock
  - Clock/Alarm Mode:
    - Default State: The clock operates normally, showing the current time. LD0 is off.
    - Blinking Decimal Point: The second decimal point from the left blinks at a 1Hz frequency.
    - Alarm Trigger: When the current time matches the alarm time, LD0 will start blinking at 1Hz and the buzzer will sound at 5Hz. Press any button to stop the blinking.

- Adjusting the Time and Alarm:
  - Press BTNC: Switch to "adjust" mode. LD0 turns on and the blinking decimal point stops.
  - Selecting Parameters:
    - Press BTNL: Cycle through the parameters in the following order:
    - Time Hour (LD12 on)
    - Time Minute (LD13 on)
    - Alarm Hour (LD14 on)
    - Alarm Minute (LD15 on)
    - Press BTNR: Cycle backward through the parameters in the same order.
  - Adjusting Parameters:
    - Increment: Press BTNU to increment the selected parameter.
    - Decrement: Press BTND to decrement the selected parameter.
  - Exiting Adjust Mode:
    - Press BTNC: Return to "clock/alarm" mode. LD0 turns off, and the decimal point resumes blinking.

3. Code Overview

The Verilog code implemented for this project is divided into 14 modules, namely:

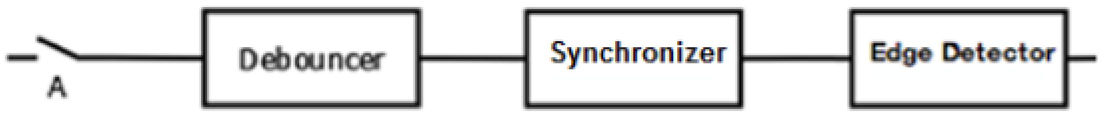
1. digitalClock: main module
2. minSecHourCount: used in digitalClock
3. alarmHourMinCounter: used in digitalClock
4. clockDivider: used in digitalClock
5. pushButtonDetector: used in digitalClock
6. modulo60: used in alarmHourMinCounter, minSecHourCount
7. modulo24: used in alarmHourMinCounter, minSecHourCount
8. counter_x_bit: used in clockDivider, modulo24, modulo60
9. debouncer: used in pushButtonDetector
10. synchronizer: used in pushButtonDetector
11. DFF: used in synchronizer
12. rising_edge: used in pushButtonDetector
13. SevenSegDecWithEn: used in digitalClock
14. Mux4x1: used in digitalClock

1. counter_x_bit

The module counter_x_bit simulates an up-down counter with the parameters x representing the number of bits needed to store the maximum count, and n representing the maximum count it can reach, i.e make it a modulo-n counter. The module also has an input called upDown, which makes the counter an up-counter when it's 1 and a down-counter when it's 0.

2. **pushButtonDetector**

A Pushbutton Detector is as shown below. The circuit has an input X which will correspond to the pushbutton, and one output Z. The circuit should detect whenever the button is pushed and accordingly generate only **one** clock pulse on the output Z. pushbutton detector module is used to interface a mechanical pushbutton with digital circuits, ensuring reliable and debounced input signals. Mechanical push buttons can generate noisy signals with multiple transitions when pressed or released, which can cause unintended multiple inputs (known as "bouncing"). The pushbutton detector module smooths out these transitions, providing a clean and stable signal to the digital system. This ensures accurate detection of button presses and prevents false triggering in the system.



"Department of Computer Science and Engineering. (2024, Spring). CSCE2301 lab manual."

The debouncer filters out the glitches associated with switch transitions as the pushbutton switches on the Basys3 board are mechanical devices. When pressed, the switch may bounce back and forth a few times before settling down. The bounces lead to glitches in the signal. A synchronizer receives an asynchronous input and a clock signal, ensuring the output is a stable logic level within a bounded time, even if the input changes during the clock's aperture time. It resolves potential metastability by passing the signal through two back-to-back flip-flops. The rising edge detector is a circuit that generates a short one-clock-cycle tick when the input signal changes from 0 to 1. It is usually used to indicate the onset of a slow time-varying input signal.

3. **clockDivider**

The clockDivider module divides the input clock frequency by toggling the output signal after every n clock cycles, where n is a parameter of the module. The counter, implemented using the counter_x_bit module described above, increments on each clock cycle and when it reaches the value n-1, it resets and toggles the output clock. The reset allows for an asynchronous reset of the output clock, setting the output clock to 0 regardless of the counter state. The desired output frequency is calculated using the equation: $f_{out} = \frac{f_{in}}{2n}$, keeping in mind that the default input frequency of the BASYS3 board is 100MHz.

4. **alarmHourMinCounter**

   The alarmHourMinCounter module displays time in hours and minutes, specifically in the case of setting the alarm time. It takes a clock (clk), reset signal (rst), an enable signal (en), a 2-bit control signal (enableMins_Hours), and an up/down control signal (upDown) as inputs. The module outputs a 16-bit bus (displayDigits) representing the set alarm time in a format suitable for the digital display. The module uses two counters: one (modulo60) for counting minutes and one (modulo24) for counting hours. The enable signals for these counters are determined by the 2-bit control signal input. When both bits of this control signal (enableMins_Hours) are 1, the minutes counter is enabled. When the first bit is 0 and the second bit is 1, the hours counter is enabled. The up/down counting for both minutes and hours is controlled by the upDown input, which determines whether we are incrementing or decrementing the alarm hours/minutes. The counted values for minutes and hours are then converted into a BCD (Binary-Coded Decimal) format and assigned to the output bus (displayDigits) for display purposes. The least-significant 8 bits of the bus represent the minutes (units and tens), and the most-significant 8 bits represent the hours (units and tens). This allows for easy interfacing with the digital clock display.

5. **minSecHourCount**

   The minSecHourCount counts seconds, minutes, and hours The module takes a clock (clk), reset (rst), enable (en), adjustAlarm, 2-bit control signal for enabling minutes and hours counting (enableMins_Hours), and a signal responsible for counting up/down (upDown) as an input. It outputs the current time, minutes and hours, in \`displayDigits\` (16-bit output each 4 bit representing a value) and \`secondDigits\` (8-bit output representing seconds). The \`secondsCounter\` is a modulo-60 counter that counts from 0 to 59, representing seconds. The \`enableMins\` signal is determined based on whether counting is enabled in adjust mode and if the seconds have reached 59 in clock mode. The \`minutesCounter\`, another modulo-60 counter, is enabled when not adjusting the alarm and when \`enableMins\` is active. The \`enableHours\` signal is set based on the current count of minutes and seconds, and the \`hoursCounter\` (a modulo-24 counter) is enabled similarly, provided the alarm is not being adjusted. The outputs \`displayDigits\` and \`secondDigits\` format the counts into decimal digits suitable for display. \`displayDigits\` contains the tens and units digits of the minutes and hours, while \`secondDigits\` contains the tens and units digits of the seconds. This module ensures that the counts for seconds, minutes, and hours are correctly incremented and displayed, while also providing mechanisms to adjust the time or alarm settings.

6. **SevenSegDecWithEn**

   The SevenSegDecWithEn module is a digital decoder designed to drive a seven-segment display, used to display a time in HH.MM format (H for hours and M for minutes). It takes a 2-bit enable input (en) to select which of the four anodes (digit positions) will be active in the multiplexed display, and a 4-bit input (in) to determine which digit (0-9) to display. The module outputs a 7-bit signal (segments) that controls the segments (a-g) of the display to form the correct number, and a 4-bit signal (anode_active) that activates the appropriate anode. The segments output is determined by the value of the input, with each digit from 0 to 9 corresponding to a specific combination of segments being turned on or off, which allows the module to display the digit on a multi-digit seven-segment display.

7. **digitalClock**

   The digitalClock module is the program's main module. The digitalClock module is a comprehensive digital clock with alarm functionalities. It takes various inputs including clock (clk), reset (rst), directional buttons (left, right, up, down), center button (center), and an alarm start signal (startAlarm). The outputs include 7-segment display segments (segments), anode activation signals (anode_active), decimal point control (decimalPt), status LEDs control (LD0, LD12, LD13, LD14, LD15), and a buzzer control (buzzer). The clock utilizes three frequency-divided clock signals (clk_200, clk_1, clk_buzzer_3) generated by clock dividers for various parts of the circuit. The push buttons are handled using five pushButtonDetector modules to provide stable input signals. The main functionality is handled by two counters: minSecHourCount for the main clock and alarmHourMinCounter for the alarm clock. These counters track and output the current time and alarm time, respectively. The finite state machine (FSM) controls the operation modes of the clock, described by the states:

   - clock:
     - Default state, displaying the time.
     - Transitions to:
       - timeHours state on pressing the center button.
       - triggerAlarm state if the current time matches the alarm time and the alarm is enabled.

   - timeHours:
     - Adjusts the hours of the time.

    - Transitions to:
        - alarmMins on pressing the left button.
        - timeMins on pressing the right button.
        - clock on pressing the center button.
        - Remains in timeHours and adjusts the hour value up or down on pressing the up or down buttons.
- timeMins:
    - Adjusts the minutes of the time.
    - Transitions to:
        - timeHours on pressing the left button.
        - alarmHours on pressing the right button.
        - clock on pressing the center button.
        - Remains in timeMins and adjusts the minute value up or down on pressing the up or down buttons.
- alarmHours:
    - Adjusts the hours of the alarm time.
    - Transitions to:
        - timeMins on pressing the left button.
        - alarmMins on pressing the right button.
        - clock on pressing the center button.
        - Remains in alarmHours and adjusts the alarm hour value up or down on pressing the up or down buttons.
- alarmMins:
    - Adjusts the minutes of the alarm time.
    - Transitions to:
        - alarmHours on pressing the left button.
        - timeHours on pressing the right button.
        - clock on pressing the center button.
        - Remains in alarmMins and adjusts the alarm minute value up or down on pressing the up or down buttons.
- triggerAlarm:
    - Activates the alarm (buzzer and flashing LED).
    - Transitions back to clock on pressing any button.

The SevenSegDecWithEn module handles the 7-segment display output, showing either the current time or the alarm time depending on the mode. The anode activation and decimal point signals ensure correct digit multiplexing and display. Finally, the buzzer and LED indicators provide visual and auditory feedback when the alarm is triggered, controlled by the FSM to indicate different operational states of the clock.

4. Implementation Bugs and Validation Activities

<u>Random Counter Increments When Switching States</u>

Bug: Switching states caused random counters to increment.
Solution: The issue was due to the enable signals being assigned incorrect values. Correcting the assignment of enable signals to ensure that each state transition correctly controls the respective counters resolved this issue.

<u>Clock Counter Changes During Alarm Adjustment</u>

Bug: Incrementing or decrementing in the alarm state (modifying the alarm counter) also affected the clock counter.
Solution: Introduced an additional flag to differentiate between adjusting the alarm and the normal clock. This flag helps track whether the system is in the state for adjusting alarm minutes or alarm hours, ensuring that the clock counter remains unaffected during alarm adjustments.

<u>Unable to Exit Alarm State Immediately</u>

Bug: Once in the alarm state, pressing any button failed to exit the alarm state immediately, requiring one minute to pass before exiting.
Solution: Adjusted the condition for entering the alarm state. Instead of relying on a flag, the condition was set to trigger the alarm state only when the clock time matches the alarm time stored and the seconds counter is at 0. This ensures that upon exiting the alarm state, the system does not re-enter the alarm state immediately since the seconds counter will no longer be at zero.

<u>Incorrect Display Switching Between Alarm and Clock Time</u>

Bug: The display did not correctly switch between showing the alarm time and the clock time at the appropriate moments, when transitioning between the different adjust states.
Solution: The problem was due to incorrect multiplexor output assignments, where the outputs were being assigned to the same register, causing data to be overwritten. The solution involved designing the system with two 4x1 multiplexers whose outputs are fed into a 2x1 multiplexor. This final multiplexor then determines the correct data to display, ensuring proper switching between alarm time and clock time displays.

<u>Validation Activities</u>

Validation activities for our digital alarm clock project included functional simulation using Logisim Evolution, incremental testing of individual components, and comprehensive real-time testing on the **BASYS3 FPGA** board to ensure accurate performance. Boundary testing verified correct time transitions and alarm functionality. We validated the user interface by testing the push buttons to ensure they correctly controlled the mode switches and parameter adjustments. This involved multiple cycles of pressing the buttons in different sequences to confirm the reliability and responsiveness of the interface. Debugging sessions helped identify and fix potential issues. To further ensure the accuracy of the clock, we compared the FPGA clock against a reference digital clock over a prolonged period. These steps ensured the clock met all specified requirements and operated reliably.

5. Contributions

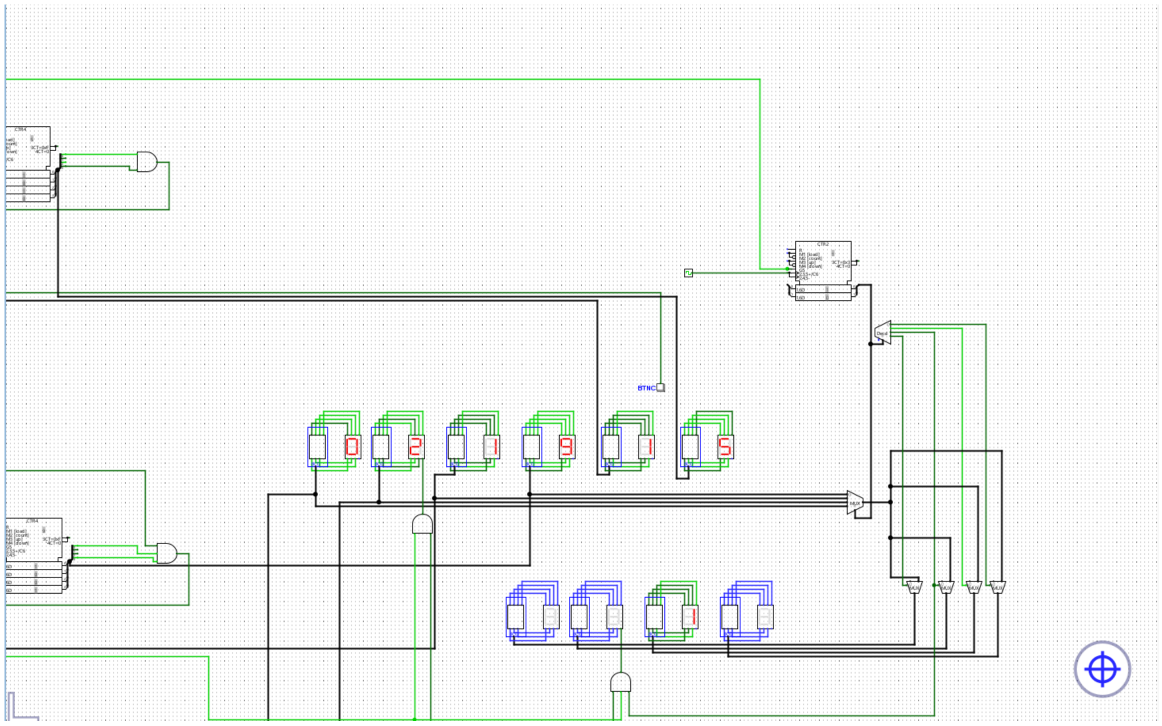
Arwa: Contributed to the Logisim simulation of a multiplexed HH:MM clock display. Designed the SevenSegDecWithEn module for efficient conversion and display of time on a 7-segment display. Worked on minSecHourCount and alarmHourMinCounter modules. Additionally, managed the assignment of LEDs to indicate various clock states and functionalities, including time setting modes and alarm states. Implemented a blinking decimal point synchronized with the clock for visual feedback.

Jana: Focused on alarm-related functionalities, modeling state transitions with an ASM chart. Implemented alarm adjustment states, clock states, and alarm states. Integrated a blinking LED to indicate alarm activation and a buzzer for audible alerts when triggered. Led the debugging team into resolving issues related to the alarm.

Kareem: Designed the datapath and control unit block diagrams. Modeled the main FSM states, handling the incrementing and decrementing for adjusting the alarm and clock and the transitions between states. Developed the respective constraint file to program the FPGA board. Implemented the up/down counter module and contributed to all-around debugging.

6 Appendix

Logisim Simulation



Github Repo Link

Click here for the project source code.