Digital Design
Project 2
Spring 2024

Mazen Zaki - Mustafa El Mahdi – Abdelrahman Elessawy – Mohanad Samy

Introduction

This report outlines the design and realization of a simple digital alarm clock using the BASYS3 FPGA board. This project was aimed at designing and implementing an embedded system that utilizes specific peripherals of the BASYS3 board and various operational modes, including requisite digital logic, for the driving of a 7-segment display. This design course involves creating a block diagram, developing the ASM chart, simulation in Logisim Evolution, and demonstrating the simulation using hardware. Development is also properly documented and structured within the GitHub version control.

The present project aims to design and implement a digital alarm clock where the design uses the BASYS3 FPGA board that has various peripherals like LEDs, push buttons, and a 7-segment display. Two designs are developed, one for each of the two operational modes: the "adjust" and "clock/alarm" design. A very practical design objective in designing an alarm clock is for the learning experience: learning how to design FPGA devices and use them in a system. Implementation requires the language to be Verilog. The initial simulation was done using Logisim Evolution.

Used Data Structures and Algorithms

Registers

Registers were used to store state information like time (hours and minutes) that is being displayed, alarm time, and mode selection and adjustment parameters control signals. Registers can hold a value that changes on every clock cycle and hence are very flexible and suitable for being signed off in this project.

Finite State Machine (FSM)

The FSM was an algorithmic structure that controlled the mode transitions between "clock/alarm" mode and "adjust" mode. The states and transitions of the FSM are such that the system responds accurately to the button input and takes care of the different modes of operation.

Clock/Alarm Mode:

State: Normal operating mode where the clock ticks and checks the alarm
Transition: Transitions to "adjust" mode upon the pressing of BTNC.
Adjust Mode:
State: It allows the user to set the time and alarm.
Transition: Returns to "clock/alarm" mode upon pressing BTNC.
Multiplexers
MUX is implemented to control selection to the 7-segment display and to identify which parameter -- hours or minutes -- is being adjusted based on BTNL and BTNR push buttons.

Counters
Counters are used to maintain the seconds, minutes, and hours of the clock. The counters are incremented on every rising edge of the clock and have roll over the number of

Seconds Counter: UP every second
Minutes Counter: UP every 60 Seconds
Hours Counter: UP every 60 Minutes
Debouncing Logic
Algorithms for de-bouncing the switches (BTNC, BTNL, BTNR, BTNU, BTND) are implemented in the code. On pressing a switch, there is only one input is it to the system so that the system does not receive a larger number of spurious triggering signals because of mechanical noise.

7-Segment Display Driver
The 7-segment display driver algorithm controls how time is displayed in terms of hours and minutes.

Input Mapping: Converts binary values of hours and minutes to 7-segment display codes.
Multiplexing: Rapidly switches between digits to give the appearance of a constant display.
LED Control Logic
The LED control logic from the current mode and whether the alarm is on controls the state of the LEDs.

LD0: Blinking if the alarm is on
LD12, LD13, LD14, LD15: Shows which parameter currently is adjusted, Time-Hour, Time-Minute, Alarm-Hour, or Alarm-Minute
Mode Control Algorithm
The Mode Control Algorithm controls the transitions between "clock/alarm" mode and "adjust" mode. It was monitoring the state of BTNC to go into the adjust mode and was reacting properly in the process of changing internal states.

Boolean Expression Evaluation
A simple boolean expression evaluation algorithm was used to determine the state of the alarm condition.

Alarm Check: Based on the current time checks for the alarm condition, if true - triggers the alarm.
Simulation Algorithm

In the main part of the project, the simulation algorithm that simulates the work of the digital alarm clock, i.e., must update time, check alarm condition, handle mode transitions, and update display and LEDs based on the current state and inputs.

Time Update: Basis on clock signal, increments time
Alarm Check: Based on the current time checks for the alarm condition, if true - triggers the alarm
Mode Handling: Based on inputs, provides a way of handling transitions between different modes on the clock
Input Handling: Processes the buttons inputs for the time and alarm setting
Display Update: Drives the 7-segment display based on the current time or adjust state

Testing
This implied that each of the components and the corresponding algorithm involved in the digital alarm clock project was individually tested prior to becoming integrated into the overall system to ensure correctness and desired functionality. The following sets of testing were performed:

Individual Component Testing

Clock Divider:

This tested through observation of the output clock signal to ensure that it generates a 1Hz pulse from the 100 MHz clock of the BASYS3.
Counters (Seconds, Minutes, Hours):

All the counters were tested to ensure that counting and rollover were done in the proper way. For instance, the operations of the seconds counter are examined to get a hold on the reopening to 00 when it gets to 59.
7-Segment Display Driver:

Drive a number sequence through the 7-segment display to ensure the correct display of digits and segments.
Debouncing Logic:

We tested each of the debouncing logics for the button by pressing the button hastily and observing the trigger system to confirm that no multiple triggers are registered for one press.
Mode Control Logic:

We tested the FSM for mode control where the mode is manually triggered, and the system is cycled in and out of the "adjust" and "clock/alarm" modes.

LED Control:

WE tested LEDs LD0, LD12, LD13, LD14, and LD15 to observe the mode and adjustment-based LED response.
System Level Testing
After the individual components are validated, the entire system was tested for functionality using the BASYS3 FPGA board. The following steps outlined the overall testing that was carried out:

Initial Setup:

Loaded the design onto the BASYS3 board and verified operation of the default "clock/alarm" mode. Verified the 2nd decimal point from left appears to blink at 1Hz.

Clock/Alarm Mode Tested:

Set current time and alarm time to the same value; verified LD0 blinked to represent that the condition for alarm was met.
Verified that pushing any button made LD0 stop blinking.
Adjust Mode Tested:

Entered "adjust" mode by pressing BTNC. Verified that LD0 is on and the decimal point stopped blinking.
Tested adjustment of every parameter: time hour, time minute, alarm hour, alarm minute, by pressing BTNL and BTNR in order to cycle parameters, and BTNU/BTND to increment/decrement values. Verified LD12, LD13, LD14, and LD15 reflected the selected parameter.
Mode Transition Tested:

Cycled the system several times between "adjust" and "clock/alarm" modes by pressing BTNC. Verified the mode of the system transitions and the system along with it.
Self Check and Output Verification
Finally, the self check and functional procedure of the output was done. This required:

Time Display Verification:

Time Display:

we manually set the time of day and observed that the 7-segment display would correctly reflect the time being sent to it.
Alarm Functionality:

We set one alarm time and verified if LD0 blinked for that particular time. Further, I had to stop the alarm by pressing a push button and resume its normal functioning.
Boundary Test:

Tested the system with boundary input; for example, from 23:59 to 00:00, it throws no error.
Simulation File Validation:

Developed simulation files for various test cases and performed manual calculation of the expected result. The simulation results were compared with the manual computations. For example, the results are given below with the following sample cases:
Test Case 1:

Time set as current time 12:30
Alarm time set to 12:30
Output: Time: 12:30, Alarm: 12:30, LD0: Blink, BTNC pressed: Turn LD0 OFF
Test Case 2:

Time set to 14:45. Display is verified.
Output: Time: 14:45, Display: 14:45, Mode: Adjust.
Test Case 3:

Increment time hours from 23 to 00.
Output: Time: 23:59, Increment Hours, Time: 00:00, Display: 00:00.

Challenges in the Digital Alarm Clock Project

It was therefore laden with several design and implementation challenges of the digital alarm clock using the BASYS3 FPGA board. Generally, problems became hardware limitations encountering design complexities and problems in debugging. The salient present challenges and how they were surmounted are explained below:

Clock Synchronisation

Challenge: Providing for clock dividers from the 100MHz oscillator at the BASYS3, ensuring exactitude in time, and not failing to provide a 1Hz clock signal.
Solution: A clock divider circuit was designed and simulated to produce a 1 Hz clock. This required careful design so as to eliminate timing errors or drift in the generated clock signal.
Debouncing Push Buttons

Problem: Mechanical push buttons have a propensity to bring up ripples of signals (bounces) for a single press, which can eventually lead to incorrect mode transitions and value adjustments.
Solution: Debouncing logic was implemented for each push button with the help of a counter and state machine included within to be able to filter spurious signals at the input. In that way, all inputs from the buttons were reliable and stable.
Multiplexing the 7-Segment Display

Challenge: How is it possible to drive multiple digits on the 7-segment display using multiplexing without inviting any flickering or display artifacts?
Solution: To show all digits on the display without flickering, a high-speed multiplexing algorithm was implemented to rapidly switch between the digits. This gave the impression of an unblinking display. Several timing adjustments were carefully made to allow just enough time for each digit to be displayed.
Alarm Matching Logic

Problem: The task of correctly matching the present time with the setting of the alarm and sounding the alarm without false alarms and missing a correct alarm.
Solution: Alarm matching logic compares all the current time registers with all the alarm time registers. Testing proves that when the times are equal, the alarm is triggered at the right time and each button pressing resets the alarm state properly.
Debugging and Validation

Problem: It is very hard in the integrated system to detect and fix bugs which were related to timing or which were causing unpredictable behaviour of the FSM.
Solution: The systematic debugging approach, one by one validating each component; a lot of use of testbenches during simulation. Hardware-related issues were detected and resolved by using the onboard debugging tools and signal probes.

External Resources Used

The only resources that were used are the course notes in addition to the labs done and what we learned from them.

Member contribution
All members of the group worked equally on the project, coding in the lab and drawing the diagram needed.

Conclusion

The project of the digital alarm clock was a dimension in its own, where successful design and implementation of the time-keeping device, fully operational on the BASYS3 FPGA board, has been accomplished. Quite an array of challenges have been encountered but surmounted in designing carried out right from the methodical steps to the rigorous testing and validation as per the specified requirements.

Some key accomplishments realized with this project were the accurate implementation of the Clock Divider, Debouncing Logic, Finite State Machine for mode control, 7-segment display driver, and timekeeping counters.

The project gave many insights into digital design using FPGAs, where proper modular development accompanied by extensive testing and efficient use of external resources should be properly applied. Enormous in making the project come true is the contribution from every team member, from designing components separately to integrating the final result into one piece and making proper tests.