Alarm Clock

2700 Group Final Project

List of Authors (Calvin Yousif, Dave Pattison , Blake Baxter, Paolo Nikaj)

Electrical and Computer Engineering Department
School of Engineering and Computer Science
Oakland University, Rochester, MI
e-mails: calvinyousif@oakland.edu, dapattis@oakland.edu, pnikaj@oakland.edu, bjbaxte3@oakland.edu

ABSTRACT

We are creating an alarm clock using the Nexys A7 board and Xilinx Vivado software to create our files and compile them together. The purpose of the project is to create a 24-hour clock that can have both time and alarm set by the user. This project will combine design and analysis of combinational and sequential logic circuits with a constraint file that will connect the alarm clock to specified switches and buttons to result in a fully functional alarm clock.

I. Introduction

When the clock is in run mode current time is displayed on the 7-segment screen. A verbal message “RISE” is displayed and a buzzer goes off when alarm time is triggered. The center push button disables the alarm. A switch is used to select program mode for setting current time and the center push button is used to save current time. Once current time is set another switch is used to select program mode for setting alarm time. In the program mode minutes and hours are displayed and the Up/down buttons change the values and Left/right buttons change the position. The selected position flashes to give visual confirmation.The internal timer is desigin function is based on just the number of minutes in a daya and by doing this allows for simple math operation to adjust time. The Project covers the topics of registers, counters, case statements, finite state machines, boolean logic, port mapping, and design and analysis of combinational and sequential logic circuits.

II. Methodology

For this project it is recognized that there was a process for the clock, 7-segment display, button functionality and top loop. Each process is written out with a diagram to understand how the circuit will work based on bit value. For complete functionality we have decided to run a clock counting by seconds and minutes which will then display the minutes onto the 7-segment display. A switch will be used to set the clock into program mode which in turn will allow the user to use push buttons to input a number (in minutes) for each display then the user will use the center pushbutton to set the clock. A maximum of 1440 minutes is allowed to be entered by the user else the clock will reset to zero.

A. Programmer

The initial challenges were with the button process on how to handle the bit flow for updating individual time registers along with checking that a user doesn’t try to enter a value outside the 1440min (24hour) range. Addressing this issue a check bit with the vector value of 1440min is used to compare

a variable (TimeBuffer) that is incremented by the user. To simplify the incrementing of the TimeBuffer variable a counter using the position bit with specific values was used. The values of each position represent the specific amounts of time in min within a 24hour period such as flashpositionINT =1 incuramented 1min, flashpositionINT = 2 incuramented 10min, flashpositionINT =3 incremented 60min, flashpositionINT =4 incremented 600min. This design offered the user the most efficient way to achieve the desired set time or alarm time. From this point the next design question was how to control the position bit and value bit with the push buttons. Initial design was to use a standard rising clock edge and bitcoutner , but with that design it was found that a debouncing and clock divider circuit would be needed. In later research it was found that just the use of sequential if statements would achieve the same results without the use of the clock divider simplifying the push button design to its current state. Other solutions were considered for this problem such as using a N_bitcounter that could be used to handle each vector position individually but with this design came the challenge of controlling the 1's place in the hours position. It was found that because there are two different value ranges to consider and other obstacles it was found to be needlessly more complex than the first design solution. With the set_Alarm it was determined that the alarm value is a unique bit value within the range of the clock. From a simple comparison operator it would make it simple to define when the alarm goes off and on.

B. *ClockCount*

The clock uses a counter that will first increment by 1 second then increment minutes respectively until the 1440 minute mark hits, this equates to 24 hours in minutes. The minutes will then reset to zero and continue counting. The challenging part of this section is adding in additional signals that will allow the user to set the time and start it at that specific time. This took unexpectedly more effort in making sure the inputs were mapped correctly while converting integers to standard vectors, due to the lack of online resources. Further research led to a parallel load counter that was used with another counter to make a clock that counted normally with a trigger that sets the minute variable to be updated then started at the user inputted value. Three inputs, one output and a couple of components were used to design the clock counter. This will result in a single output variable clock, a counter drives the clock that can be updated by input from the programmed values. The counter driver and the input variable will both feed out using the output variable which will be displayed on the 7-segment display. Making the clock solely a minute variable allowed the code to be slightly simplified with a greater efficiency when connecting the variable through the top file. This representation also eliminated the amount of warnings given through the error messages on vivado of potentially creating open latches which would have led to greater issues once the different sections of the project were all brought together.

C. *DisplayDriver*

A key element in this project is the display. The display will be on at all times and will show the output of the clock as time passes by. There are three main modes of function for the display. While the clock is in regular display mode, it just shows the current time. Once the program is switched to set the time by using the first switch on the board, then

the display will adjust to give us feedback. When it is in set time mode, the display flashes the corresponding display position that is being adjusted to set the time. When the switch is turned off, then it goes back to displaying the set time. When the second switch is turned on, the display changes to allow the user to set the time using the buttons to navigate. Once the second switch is turned off, then the display will go back to displaying the current time. Finally, once the alarm time is reached, the display will then switch from displaying the time, to displaying "RISE" until the reset button is pressed. The method used to achieve this was done by using a Moore-type finite state machine (FSM), which uses 4 states to cycle through the 4 displays to keep them running and displaying at the same time. To keep the lights on and displaying brightly and properly, a component was used to generate a pulse at the right speed. Another component was created to make the displays flash on and off by using the clock and pulsing at a set speed for an on and an off signal. To switch the modes of the display, processes were made using "if" statements to account for what we want displayed and when to display it. A lot of testing and debugging went into this part. The values of the count on and off for the blinking feature was fine tuned to display a nice timing for the flashing of the display.

D. *Topfile*

The Topfile will route all variables to and from other subroutines. The components are port-mapped into the proper inputs and outputs. The time values sent to the display driver will come from either the Programmer or Clockcounter subroutine depending on operator conditions. The operator conditions account for changes in between the functions of the alarm clock. The Topfile will compare the alarm time to the current time to decide when to trigger the alarm. This is done by using "if-then-elseif" statements. This was done to account for all the possible cases to ensure that the alarm clock functions properly in each mode that it is in. It was difficult at first to create a memory bit to make sure that the alarm does not continue to go off after being disabled while the compared time values are still the same. To solve this problem the memory bit was added to the process that is used to control the different clock modes which allowed for the storage of the set time values depending on the position of the switches.

Experimental Setup

The use of Xilinx Vivado software is used to verify that circuit is syntax vhdl free and to simulate each component. The project tested on a Nexys A7 board (xc7a50tcsg324-1) allowing for physical verification of functioning processes.

Results

The results were a 24 hour clock that displays current time on the 7-segment with the option of setting an alarm and current time. The buttons and switches function as expected. But one thing that was unexpected was the colon on the display is not functional according to the datasheet. One interesting result was recognizing that the clock has both a mealy and moore finite state machine functions. When the clock is in run mode it functions as a moore machine (there is no input directly fed to the output logic) but when the clock is in program mode you are directly feeding the inputs to the output logic of the clock acting like a

mealy (FSM). The end result is an alarm clock that is well put together and thought out with the maximum and minimum values and the ease of use to prevent user error in any way.

Conclusions

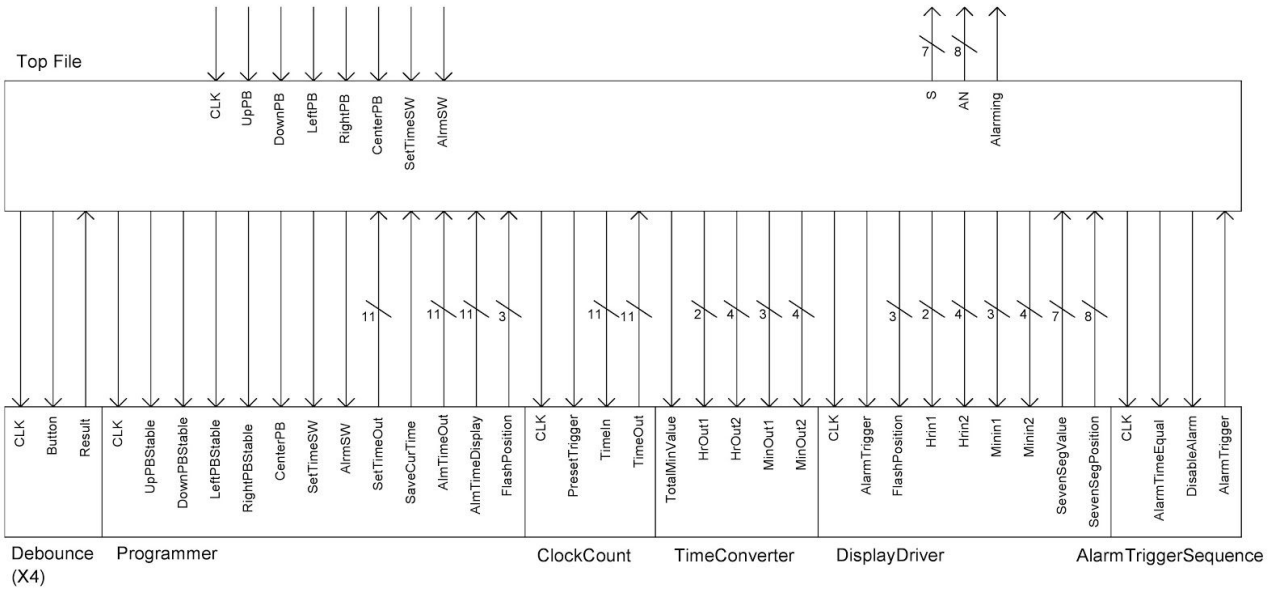
The key takeaway from working on this project is that there are multiple ways of coding, except there are challenges that you might not have considered and verify your logic is sound before coding. Furthermore, doing the work independently on each section of the project can make it very challenging to bring together when design ideas are not fully flushed out. All the bugs were sorted out and after many hours spent by the group, the project came to become a success. A lot was learned through this process, both in programming using VHDL and also in teamwork.

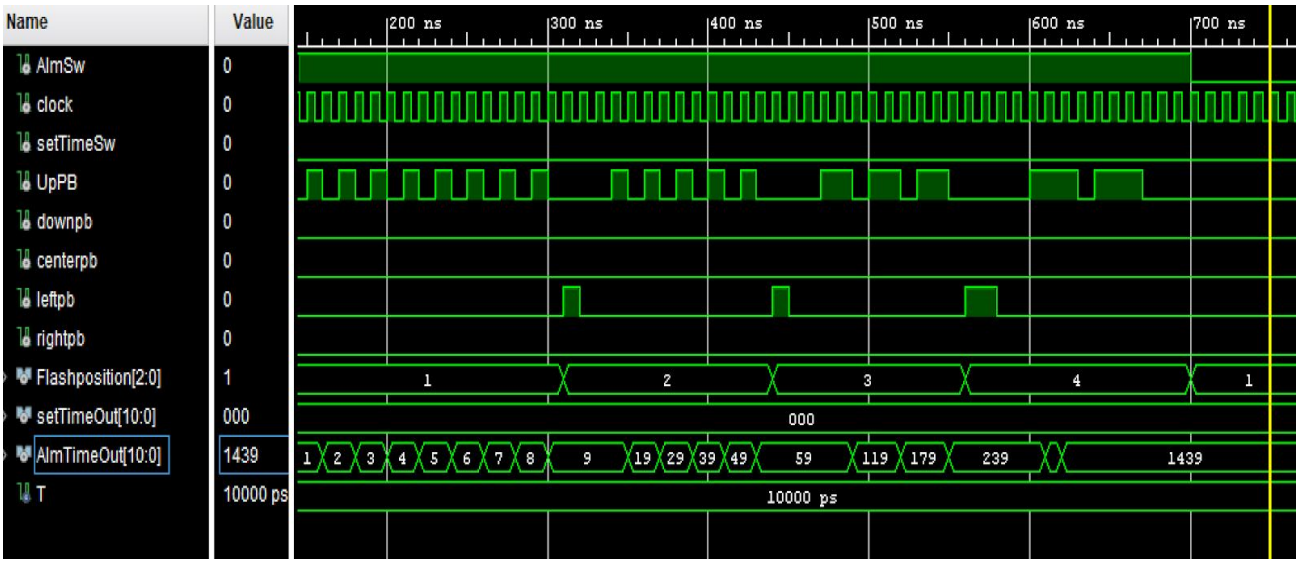
References

[1] Daniel Llamoca Obregon, serializer 7-segment display. (2013) Retrieved from http://www.secs.oakland.edu/~llamocca/VHDLforFPGAs.html

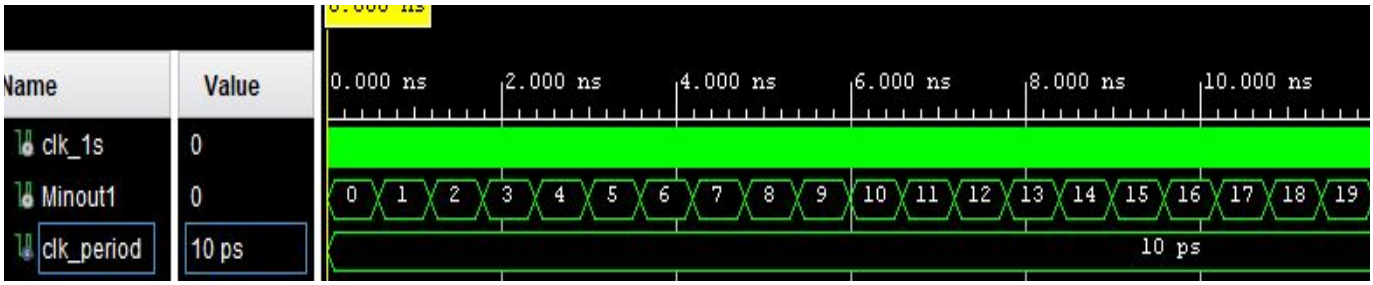
[2] Examples of VHDL Conversions. (n.d.). Retrieved April 17, 2020, from https://www.nandland.com/vhdl/tips/tip-convert-numeric-std-logic-vector-to-integer.html

Diagrams and Simulations

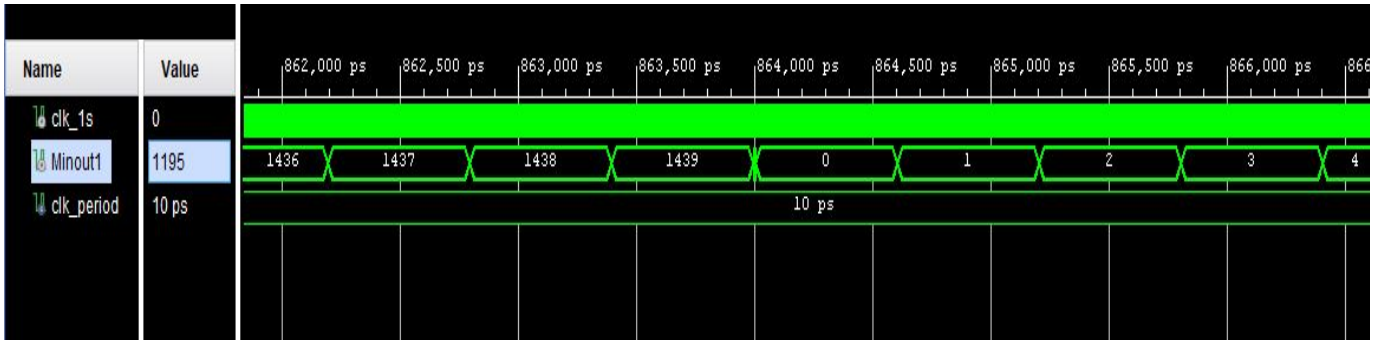




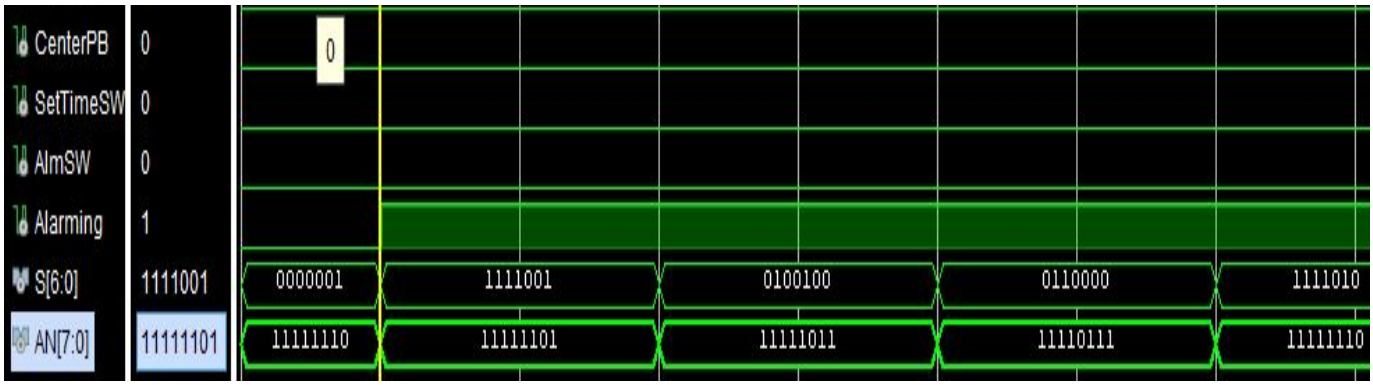
Pushbutton simulation: This shows how the program records each button press



Clock simulation 1: This is a representation of the counter incrementing the minout value by one



Clock simulation 2 : The once the counter reaches max value the minout resets to zero



Display simulation: This shows the S and AN outputs which display the number on the seven segment display and the position respectively. The AN shifts to keep all 4 of the displays illuminated.