Clock Timer

Designed in VIVADO using VHDL and Implemented on Nexys 4 DDR FPGA Board

Khanh Nguyen, Etjen Zeka, Abbigail Yaldo, Danish Ibadat
Electrical and Computer Engineering Department
School of Engineering and Computer Science
Oakland University, Rochester, MI

E-mails: kknguyen@oakland.edu, etjenzeka@oakland.edu, abigailyaldo@oakland.edu, dibadat@oakland.edu

Abstract - Purpose of this project is to make students to work as a group to design an electric powered digital stop-watch with a special function to save four different time instants. This specific electric-powered digital stopwatch is going to have start, stop, write, read and a reset button. This project was built in Vivado using VHDL and implemented on a Nexys 4 DDR board. However, the major purpose of this project is to make a stopwatch timer for everyday use. The components of this final project contains most fundamental VHDL codes.

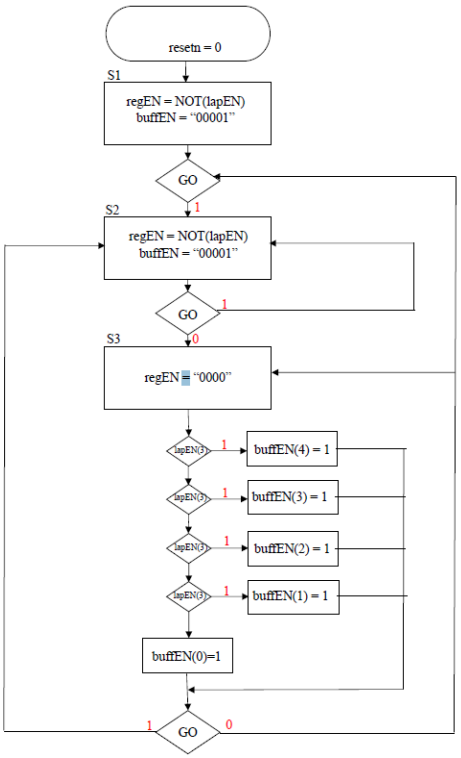
I. Introduction

The purpose of this project is to create a digital stopwatch using VIVADO to program the VHDL and Nexys 4 DDR board to implement the project. This electric-powered digital stopwatch can count up time and has multiple special functions. This electric powered digital stopwatch can reset back to zero by pressing a "reset" button, which its first special function. The next special function of this stopwatch is to save four different times instant, when the user wants to look at the saved time instant later, would be able to. In addition, this electric-powered digital stopwatch can display seven segments and up to eight digits. As a result, the time range can display from hours to hundredths of a second. The VHDL code consists of one single finite state machine, buffers, seven and gates and a single or gate, eleven different counters varies in functions, four registers registers four different time instants, multiplexers to decide which inputs going to watch destinations, three different types of decoders to decodes different inputs and outputs, last but not least two four digit seven segment display to display the running time and saved time instant. The timing of the individual seven segment displays required learning how to have them cascade such that the next nomination of time would only appear once the previous ones have elapsed. This project can later be adapted to use in an actual digital stopwatch.

II. Methodology

Coming up with the idea to create a stopwatch is not a simple task. Adding the idea of a lap feature makes the task that much more difficult. As a group, we had to come together and dissect various components and how they behaved. To achieve the desired result of this project our method consisted of studying previous labs and then taking that knowledge and integrating it into our idea of a stopwatch in order to successfully create this stopwatch. To start we had to understand what a stopwatch does in its simplest nature, which is to count. To achieve this task we had to incorporate a 32 bit counter composed of BCD counters in order to keep track of decimal numbers in binary. These counters are composed of (0-5) and (0-9), this is to keep track of time and then later be displayed onto the seven segment-displays. Once the task of counting was complete, the question of storing data came up. In order to keep track of data we needed a registry, but not just any registry this registry was to be made up of Dflip-flops. These Dflip-flops were to be connected to a FSM machine, which was to keep track of various states in order to store the data only when select switches are on or off. Once we had a method of counting and storing data we used an OR gate in order to have a controlled data path connecting to a MUX in order to select what data is to be presented. Now that we have all the data we asked ourselves how do we display this? This was achieved by integrating a combination of decoders, and counters to take data and make it suitable to fit the seven segments in binary and at the same time incorporating counters to control what is being shown at given times.

Finite State Machine



The FSMs main function is to display the output. There are three different states in this Design including initial state, count state, and a pause State. The first state is enabled when the resetn signal is low. Within this state, the buffer enables are set so that only the buffer for the count is enabled, so as to only display the count. The register enables are also set to the not of lapEn, which makes lapEn an active-low enable for the registers, recording the lap when a switch is toggled up. This state then moves on the next clock cycle to either one of the two remaining states, where the main operations take place. The count state is activated when the go input, in essence connected to stop_go is 1, in which the stopwatch counts up with increments of hundredths of seconds. This state also holds the same values as the initial state did. The third state, pause, is activated when the go input is 0. In this third state, all the register enables are set to 0, and the buffer enables are determined by lapEn. Finally the FSM checks the laps from 4 down to 1, making the data in priority, only showing the latest count when all the lap switches are at 0.

32 Bit Counter

For this 32-bit counter, we have implemented a combination (0-9) BCD counters and (0-5) BCD counters. The reason for the combination of the counters is due to working with time. It takes 60 seconds to keep count of 1 minute; this works the same with minutes to hours. For the project, we used a Nexys 4DDR board, which has eight seven-segment displays. For every seven-segment display, we used a version of our BCD counters for a total of eight BCD counters. Each BCD counter is linked with an (and) gate to keep the clock in sync and responsible for keeping the count for every individual seven segment-displays. The role of a BCD counter is to either keep track of the decimal numbers (0-9) or (0-5) in binary. Then these values are bussed over to the MUX to keep the clock running while the stop_go switch is high or over to the registry's to keep track of the various laps that will be saved. The 32 Bit counter is driven by our (ms) counter. This counter is responsible for taking the board's natural 100Mhz clock frequency, which comes to 10 ns, and converting it over to (ms).

Display Department

Thirty-two to four Multiplexer, seven segment-decoder, three to one decoder, three to eight decoder, two blocks of four seven segment displays, one millisecond and modulo eight counter.

Combination of eight four bits inputs that will make up thirty two bits input bus. This thirty-two bits bus going to thirty-two to four multiplexer, its function is to select a particular digit to display at a certain time. In addition, selector line contains three bits, going from zero to seven. At this point, only the anode is enabled, that would not make two four seven segments display to function properly. To make it work properly, cathode should be connected, so three to eight decoder should be connected to the cathode of the two four seven segment displays. Decoders can enable the necessary digit, and it only can send a single digit whose position is indicated by the selected line. Within the two four seven segment displays, there are decimal points to separate the hours from minutes and seconds from hundredths of seconds. In order to enable those decimal points, three to one decoder has to connect has to send a "1" on the DP input when position two or six is selected, to enable the decimal point.

According to the instruction manual of the Nexys 4 DDR board, two four seven segment display's refresh rate limit has to be between one to sixteen milliseconds. As a result, each digit's refresh

rate has to be to one millisecond. Since there are eights digits in total, so it would take eight milliseconds for the whole thing to take a single loop. Eights milliseconds is below ten milliseconds clock speed of the project. For one millisecond counter, enable activated and z output connect to Modulo eight counter, make sure selected line go at a given speed. The Q output of the Modular eight is then select line and going to cause the display to light up.

Lap Function/ Register

The lap function was the tweak in the design to offer the user another function on top of the general stop, go, and reset ones that are to be included. The data of the laps was stored in a register to keep track of a certain count. This is information was displayed on to the onto the seven segment display with a write function of a switch or button. There were four D Flip-Flops used as well as a buffer enable and register enable within to properly implement the lap function. Each D Flip-flop output was inputted into a not gate which also had a register enable input. All of them, as well as the counter were inputted into an OR gate which ensured that only one would be able to output at a time. This output then led to the MUX to transfer to the seven segment display.The display function above is to be implemented to keep track of the 1/100sec,sec,min,hours. This will allow the use of all the seven segment displays, which is eight in total.

III. Experimental Setup

To properly execute the project, it was necessary to use the Nexys Artix-7 board as the output of the results. To configure the board, Vivado VHDL coding was used and source files were created. The source files created were the seven segment, adder, counter, BCD counter, MUX, BCD counter modulo 6, topfile, testbench. To allow for correct output of the testbench, a xdc file that sets constraints is required. Each type of board has a different xdc file, hence it was necessary to pay very close attention when selecting the file for this project. After sources were set up, it became possible to continue to put the correct inputs and outputs making sure they contain buses that were needed. The Top file contained all of the sources within the main one and was required to have the correct signals and port mapping. Sources themselves need signals when needed along with mapping. Finally the test bench needed to be altered to its correct components. It was initially thought that the modification would work but didn't turn out successful.

IV. Results

Initially, there were issues between the top file and test bench so no results were outputted to the Nexys board. Once this issue was resolved, the program functioned as it was initially intended to. Near the end of the project, there were a complication of the code not running correctly for the finite state machine portion of the code. Through multiple hours of debugging, the errors were found and replaced with the right correction. The completed project outputted the current running time once the go input (stop_go switch) was enabled. The board was designed to include hours, minutes, seconds and milliseconds which were shown onto the 7 segment display counting up from hundredths of seconds. These results were the main goal that had been hoped to achieve when implementing the code onto the board. The results are unexplainable when the stop_go switch is on active low. If more than one of the registers switches in on active high at the same time, the number outputted is not a correct reflection of the data. Otherwise, all results are completely explainable and were as expected. At the very end of the code, LapEN zero and LapEN four would not be able to save the time instant due to wrong coding in register portion, that portion also got correct Lastly, we faced complications within the 32 bit counter that essentially not displaying correct output which we realized it was a simple issue within the MUX which was resolved

V. Conclusions

It is not as simple to create a digital stopwatch as one may initially assume. Using the Nexys board to output results was a more efficient option than using an arduino board and coding. As the process of creation continued, more than just educational knowledge was obtained. Furthermore, how to work well within a team and how to efficiently delegate and complete tasks was learned

as well. There were some issues regarding the connection of the testbench to the top file, but through the use of knowledge obtained throughout the semester it was possible to fix those problems. To differentiate the stopwatch, a lap functions were added to be able to save up to four different times while the timer was being run. The creation of the clock timer assisted in tying together everything that was learned throughout the duration of the course as well as through each one of the labs. All prior issues for the timer were resolved and there are no current issues that require fixing. There is always room for improvement and in the case of this timer, a possible improvement may be to add a microsecond display into the timer.

VI. References

http://www.secs.oakland.edu/~llamocca/Courses/F17_ECE2700/FinalProject/Group3_Stopwatchwlap_presentation.pdf

http://www.secs.oakland.edu/~llamocca/Courses/F17_ECE2700/FinalProject/Group3_Stopwatchwlap.pdf

http://www.secs.oakland.edu/~llamocca/VHDLforFPGAs.html

https://reference.digilentinc.com/reference/programmable-logic/nexys-4-ddr/reference-manual