A Digital Stopwatch

Designed in VIVADO & Implemented on NEXYS 4 DDR FPGA Board

Matthew Guirguis, Andy Lalaj, Jean-Pierre Ortiz, Derek Ramos
Electrical and Computer Engineering Department
School of Engineering and Computer Science
Oakland University, Rochester, MI
mgguirguis@oakland.edu, anduellalaj@oakland.edu, jortiz@oakland.edu, dkramos@oakland.edu

Abstract: A digital stopwatch was built in Vivado using VHDL and implemented on a Nexys 4 DDR board. The stopwatch had a pause, go, lap, and reset function. The primary goal of the project was to develop a better understanding of digital systems. The choice of designing a simple stopwatch system was made so that each member of the group could participate as a stopwatch contains the most fundamental digital logic components.

I. INTRODUCTION

The goal of this project is to construct a digital stopwatch using a combination of VHDL and the Nexys 4 DDR board. This stopwatch in particular will be able to count up based on the time elapsed; in addition, it is being designed such that it can save four of those times or ‘laps’ for future use. This allows the stopwatch mimic the functions of a mechanical watch, with the time being displayed on the seven segment display of the board. The time will range from hours to hundredths of a second as displayed on the seven-segment display. The code itself will use a combination of registers, state machines, buffers and logic gates to control the circuit. 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 then be adapted for use in an actual digital stopwatch.

METHODOLOGY

The two major components of this digital system design were the finite state machine (FSM) and the datapath circuit. A finite state machine is a mathematical model of computation that can be in exactly one of a finite number of states at any given time.[1]

The datapath circuit is the collection of functional units that performed the data processing. Seven AND, four NOT, and one OR gate was used in conjunction with three busses. The first bus was designed to hold the 32- bit output of the counters. The second and third bus were connected to the output of the FSM controlled the buffer enable and the register enable respectively.

Switches are the main control of the stopwatch. The first four switches on the board (SW[0] - SW[3]), going to signal lapEn, control the lap function of the stopwatch, this function allows a user to record up to four different times while the stopwatch is running, allowing a user to write and rewrite as he/she chooses, to be displayed after the stopwatch has been paused. The last switch (SW[15]), going to signal stop_go, controls the stop and go function of the stopwatch. The cpu_reset button, going to signal resetn, is coded to reset the value of the stopwatch to zero. The seven segment display displays the time elapsed down to a hundredth of a second. The maximum time possible is slightly over 100 hours. The major components of the system are as follows:

A. *Counting*

Every counter used in this project has a total of three inputs and two outputs. The first input, resetn, is an active-low reset which sets the count to zero when resetn is low. The next input, clock, is the clock signal from the board, where the counter will only function on a rising edge of the clock. The last input, E, is an enable that either allows the counter to function and continue counting if it is driven high, or pauses the counter when it is driven low. The two outputs are Q and z, where Q is the current count of the counter, and z is only high when the counter has reached its maximum count.

This stopwatch uses eleven counters to function, nine of them are shown in Figure 1. The other two counters will be discussed later in the report. The stopwatch increments every .01 s, or every 10 ms, however the Nexys 4 DDR board has a native clock speed of 100 MHz, which translates to 10 ns per cycle. In order to get a clock pulse for 10 ms, the first counter was used to produce an output high every 10^6clock ticks. Now, the clock has been translated from a speed of 10 ns to 10 ms, which is what is required for the stopwatch to function. The enable for this counter is the stop_go signal, which means that if the signal is low, the count will stop, only counting if the signal is high. The other eight counters in Figure 1 each correspond to a digit of the counter, with six 0-9 and two 0-5 BCD counters. The z output of the 10 ms counter then gets routed to two locations; the enable of the hundredths of seconds counter and an and gate linking the z output from the hundredths counter and its enable. The output of the and gate is then sent to the enable of the next counter, and this process is continued for the other seven counters. Thus, the counters will all be in sync, each only counting once the one before has reached its maximum count.

A BCD (binary coded decimal) Counter is a serial digital counter that counts ten digits and then resets for every new clock input.[2] This was done using a four bit counter that skips the values from ten to 15. This is known as a modulo 9 counter, as it retains the modulo function by resetting the value after the counter reaches nine.   A modulo 5 BCD counter is also used to display the maximum value for seconds and minutes, as the largest unit for minutes and seconds are 59.

The 4-bit Q outputs of all eight BCD counters then get placed into a 32-bit bus which will transport the count to the next portion of the system.

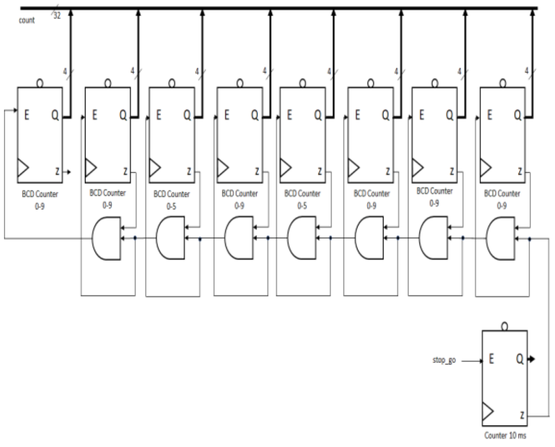


Figure 1:Counters

B. *Lap Function*

The lap function is made up of four 32-bit registers, 5 buffers and an or gate. A 32 bit register has four inputs. A data input, where the signal to be stored is sent, in this case a 32-bit signal. An active-low resetn input sets the output of the register to 0 if it is driven low. A clock signal dictating when the register outputs the data input, on every rising edge. Finally, the enable input only allows the register to output the data input when it is driven high, otherwise, the register maintains its previous value. The output ,Q, of the register is a 32-bit signal that is taken from the data input. A buffer either passes a signal through it or outputs a 0, depending on high or low of an enable input.

For the lap function to work properly, the 32-bit count bus from the eight counters is fed into the data input of all four registers. The enables for the registers come from a 4-bit bus, lapEn. When an enable is high, the register will be recording the current count. Once the enable is switched low, the register will hold the count value from that moment, effectively holding the lap time. The Q outputs of the registers, along with the count bus, get sent into 5 buffers. The buffers are controlled by the 5-bit bus, buffEn, which enables one buffer at a time. This means that at the or-gate, only one output will contain any ones, as the other four outputs will be all zeros, thus outputting the signal from the enabled buffer. While the stopwatch is counting, the buffer for the count is enabled, which sends the count output to the display portion of the system.

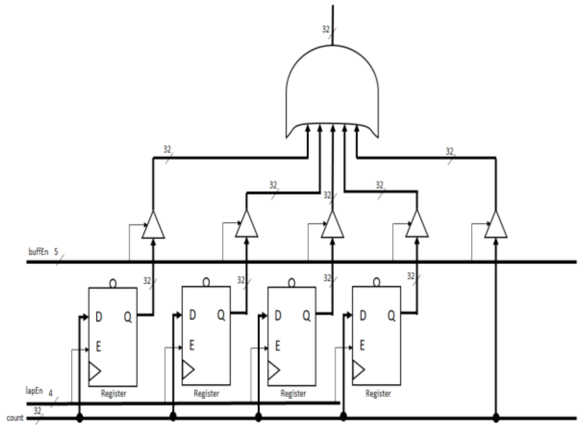
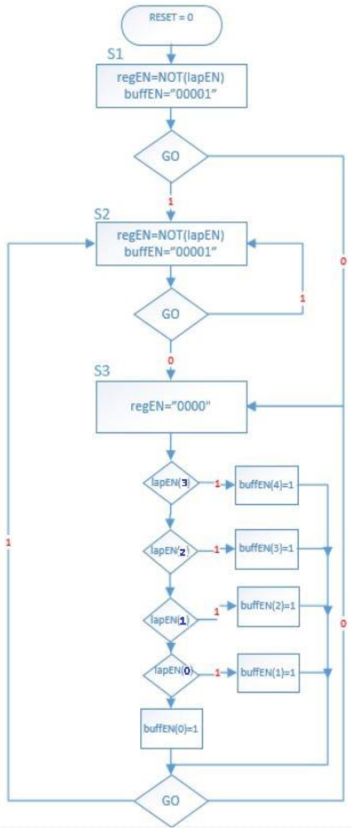


Figure 2: Lap Function

C. Finite State Machine



The FSM controls the output to be displayed. There are three different states in this design: an initial state, a count state, and a pause state.

The first state is triggered only 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 immediately 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 (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 state, all the register enables are set to 0, and the buffer enables are determined by lapEn. The FSM checks the laps from 4 down to 1, enabling the data in priority, only showing the latest count when all the lap switches are at 0.

D. Display Function

The stopwatch display is made up of two banks of four 7-Segment Displays. These displays are driven by a combination of two counters, a 32-to-4 MUX, a 7-Segment Decoder, a 3-to-8 Decoder and a 3-to-1 Decoder.

Since the display is made up of eight displays who's data lines are all connected, it is necessary to multiplex through all of the inputs. In order to accomplish this, the 32-bit bus, containing eight 4-bit inputs, is put through the MUX in order to select a particular digit to display at a time. This MUX will have a 3-bit select line, going from 0-7. At the same time, only the anode for the display that applies to the particular input may be enabled. Therefore, it is necessary to also tie the select line into a 3-to-8 Decoder that will enable the necessary digit. The decoder will send a 1 only to the digit whose position is indicated by the select line. In order to enable the decimal points to separate the hours from minutes and seconds from hundredths of seconds, it is necessary to also send the select line into a 3-to-1 Decoder. This decoder will only send a 1 on the DP input when position 2 or 6 is selected, to enable the decimal point.

However, to run all these components, it is necessary to drive the select lines. Xilinx recommends driving the 7-Segment displays so that they are refreshed every 1 to 16 ms [3]. This means that each digit should refresh at this rate, implying that all 8 digits need to be looped through once during this timeframe. Therefore, activating each digit for a time frame of 1 ms will mean a refresh speed of 8 ms, which is still less than the slowest clock speed. This ensures that each tick of the stopwatch will have at least one refresh for each digit. Therefore, hooking the z output of a counter, counting *10^5* clock cycles (.001s) with an activated enable, to a modulo-8 counter will drive the select lines at this speed. The Q output of the mod-8 counter is then the select line and will cause the display to work.

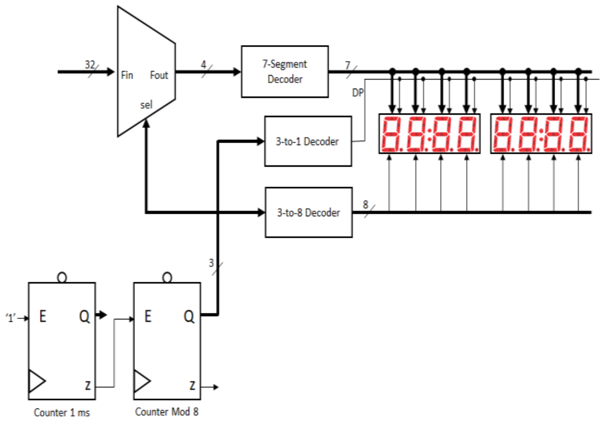


Figure 4: Display Function

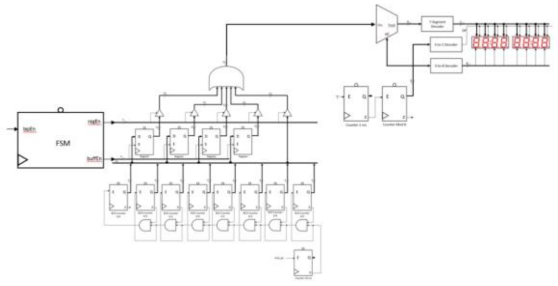


Figure 5: Circuit Design

II. EXPERIMENTAL SETUP

The hardware used in this project was an FPGA board, the Nexys 4 DDR, along with a computer in order to write to the FPGA board. The software used on this computer was Vivado 2017.2. This program allows a user to write VHDL code that can then be implemented onto an FPGA board. Also, Vivado allows for simulation, by the writing of a testbench, in which a user is able to write a script/process for stimulating the inputs of the digital system. Simulating what would happen on the actual FPGA board.

The testbench was designed in order to test the distinct functions of the digital system. This includes the multiplexing through the display by the select line, the 7-segment decoder, the cascaded counters, the registers and buffers. Although there is a plethora of things to test, it isn't entirely necessary to test all cases, just that the system functions properly.

The issue that was encountered in testing the project was simply the time required to implement the simulations. Therefore the testbench mostly revolves around the lap function, only counting up to .05 s, and recording 4 laps within that time. Then pausing and going through and displaying the recorded laps and count. At the same time that this happens, the anode enable output should be seen to multiplex through all 8 digits. The 7-segment decoder, along with the DP should also be seen to correspond to the specific outputs. The testbench used to accomplish this is seen in Figure 6 below.

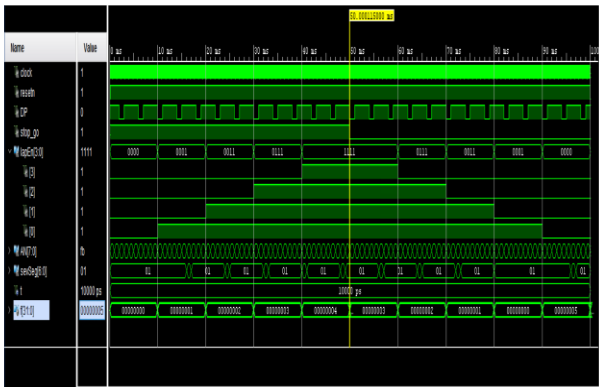


Figure 6: Count and Lap Function Test

The rest of the more rigorous and thorough testing was able to be accomplished by implementing the program onto the Nexys 4 DDR board. In this fashion it would be able to ascertain that the circuit works as it should.

III. RESULTS

The stopwatch functioned according to the design. The results were exactly what was expected. The only problem was the use of the decimal point instead of the colon. The code was designed to include a colon in the display but the proper leds in the seven segment display would not power. This was strange considering the fact that they are on the same power rail when the schematic was inspected.

IV. v

The project was done to develop a greater understanding of digital systems. Everyone that participated has a better understanding of digital systems now. To further this design a number of things can be done. Considering this is a basic counter with stop and go function the can be implemented in a considerable amount of projects a a form of time keeper.

REFERENCES

[1] Wright, David R. (2005). "Finite State Machines" (PDF). *CSC215 Class Notes*. David R. Wright website, N. Carolina State Univ. Retrieved July 14, 2012.

[2]https://www.electronicshub.org/decade-counterbcd-counter/

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

[4]http://www.secs.oakland.edu/~llamocca/VHDLforPGAs.html