Digital Stopwatch

List of Authors (Mohammed Shatit, Matthew Adams, Austin Nguyen)
Electrical and Computer Engineering Department
School of Engineering and Computer Science
Oakland University, Rochester, MI
e-mails: matthewadams@oakland.edu, austinnguyen@oakland.edu, mshatit@oakland.edu

***Abstract*— Stopwatch is a digital system of counting to account for time took for an event. It is widely used in sports and laboratories. The stopwatch works by pressing a button to start and stop or simply turning the count on and off. The count can then be re-started or continued. The saved time of the interval can be very precise and accurate depending on the design of the stopwatch. Some digital stopwatches can save more than an interval of time. The digital stopwatch here is implemented using VHDL in Vivado and coded on a Nexys Artix-7 FPGA. This design of the stopwatch includes 6 7-segment displays and a couple of buttons and switches to control it. The design has been tested and simulated to account for all scenarios and it mainly came down to controlling the displays through a finite state machine and connecting different types of counters together to show 6 digits going from 00:00.00 to 59:59.99 and repeating.**

I. Introduction

This report will cover the design methodology for a digital stopwatch along with a description of the experimental setup used to verify the functioning of the project and the results of this verification testing. A functional digital stopwatch has become a staple of modern life. Stopwatch capabilities are a feature that is simply expected in every wristwatch and smartphone in today's market, making the implementation of this technology a necessary skill. This project will utilize a variety of skills learned throughout this course, including the use of synchronous and asynchronous circuit elements, 7-segment display serialization, clock signal modulation, and memory storage and recall.

II. Methodology

A. *Counting Execution*

The goal of this portion of the system is to produce six signals, each of which will accurately reflect a digit of the time elapsed since the start switch was activated. This goal was achieved by directly adapting the stopwatch example provided in the VHDL Unit 7 slides on the course website [1] to function for 6 digits with the two most significant digits representing minutes, the next two most significant representing seconds, and the two least significant representing tenths of seconds and hundredths of seconds respectively. The primary difference between our design and the lecture example referenced above is the addition of another modulo-6 counter and BCD counter to the left of the existing modulo-6 counter. These modulo-n counters were made by generic mapping the file "my_genpulse_sclr" from the course website to suit our needs and the BCD counters were created from "mybcd_udcount" from the same website [1].

Below is an image of the lecture stopwatch design referenced previously, followed by the modified design. Both designs consist of three input signals (clock, pause, and resetn), and in both designs all three inputs serve the same purpose. The clock signal serves two purposes: to synchronize all of the counters as well as to provide a "base unit" for the counting functionality. The resetn signal resets all the counters when it goes "low", and the pause signal begins or resumes counting when it goes "low" and pauses when it goes "high".

The counting is implemented first by feeding the clock signal into a modulo-1,000,000 counter. Since the clock ticks every 10ns, the result is a "high" z output from the modulo-1,000,000 counter every 0.01 seconds. This z signal (Z0) is then fed directly into the first BCD counter of the array. This BCD counter will have a Q output (Q1) and a Z output (Z1). This will result in Q incrementing every 0.01 second ("0001", "0010", "0011"…) until reaching a maximum value of "1001" at which point it starts over. The signal Z1 will go "high" every time that Q1 reaches "1001". Signal Z1 is then fed into an AND gate along with signal Z0 and the output of this AND gate (E1) is fed to the "enable" input of the next BCD counter. This process is continued for the entire array of counters, the AND gate feeding each counter's enable fed by the Z output of the previous two counters.

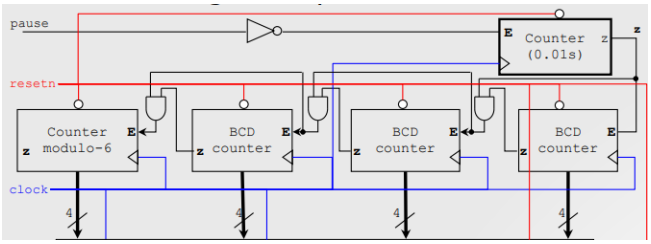


Fig 1. Counting Implementation from Lecture

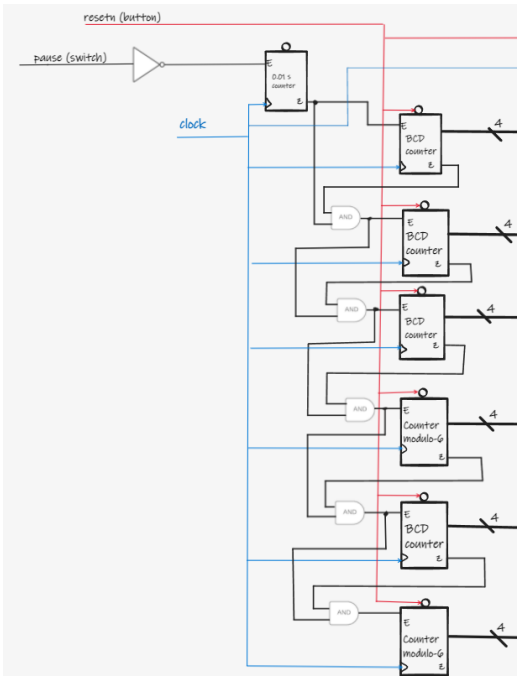


Fig 2. Extended Counting Implementation

B. Lap Memory Storage and Recall

The goal of the Lap Memory Storage and Recall section of this system is to be able to record and store the values of time at a specific point without interrupting the stopwatch Counting Execution and be able to display the recorded lap time on the seven segment displays also without interrupting the stopwatch. To do this, 6 4-bit registers are implemented into the design. The 4-bit registers will receive its input signal from the output of the BCD counters and the Modulo-6 counters from the Counting Execution portion of the system. The 'Enable' of these registers will receive its signal from a button on the FPGA board, this will act as the "Lap button." With this set up, whenever the 'Enable' receives an active high signal (button is pushed) the output from the BCD/modulo counters at that point in time is stored into the registers. This completes the Lap Memory Storage function.

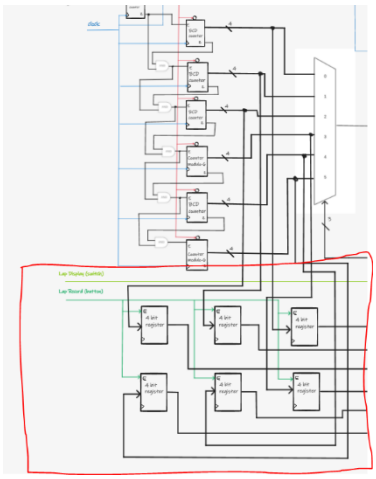


Fig 3. Lap Memory Storage Registers

Now that the Lap time is stored, the user of the stopwatch needs to be able to display the recorded time. This is done through the implementation of a 6-to-1 Multiplexor and a 2-to-1 Multiplexor. The 6-to-1 Multiplexor is used to separate and organize the 6 stored values and place them into their proper place value as a unit of time (the display portion of this report will explain this further). Next, a 2-to-1 Multiplexor is used to decide between displaying the current stopwatch count or the stored Lap time. This Multiplexor will receive 3 inputs: one from the 6-to-1 MUX connected to the Counting Execution portion, one from the 6-to-1 MUX connected to the Lap Memory Storage portion and one from a switch on the FPGA board. The switch from the FPGA board will decide whether the 2-to-1 MUX sends an output of the current time or the lap time. Therefore, this switch will be capable of choosing which signal is displayed. The multiplexors were made by modifying "my_busmux4to1" and the registers were implemented using "my_rege" from the course website [1].

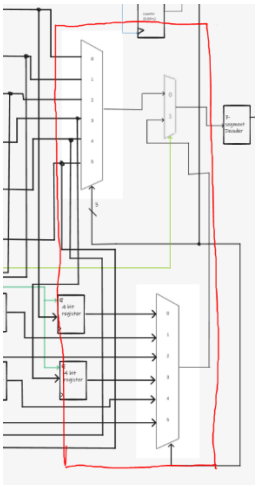


Fig 4. MUXs

C. Display

To display the 6 digits of the count of the stopwatch, 6 of the 8 7-segement displays were used of the Nexys board, however, it is not as simple as connecting the output of the counters to the displays. These 7-segment displays are all fed through the same signal, meaning a one counter output signal 'Q' will be displayed on all the displays at once. This is not what this design should accomplish. A solution to this is using a multiplexer, 6-to-1 specifically, to control which counter's output signal to display on the correct 7-segment display by turning all 7-segment displays off except for the right place for the digit of this counter. To control this operation, a finite state machine is used to alternate between the signals of the counters and the correct display for each one. For 6 displays, the state machine will have 6 states each stepping into the next state if E is "high", which is the Z output signal from a 1 millisecond counter to have a 1 millisecond show time for each display. It will also have a resetn to go back to state 1 in the case of restarting. The output of the state machine will be a 3-bit, S, that will go in as the select signal of the multiplexer and the input for a 3-to-8 decoder to convert these 3-bit into an 8-bit signal with "low" bits except for the right place of the displays that will be display the counter digit. This operation is called 7-segment serialization. Below is the Algorithmic State Machine shown. The decoder, modulo-n counter, and state machine where created by modifying provided files from the course website [1].

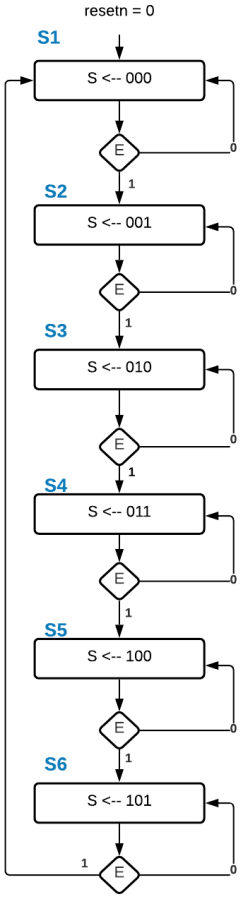


Fig 5. ASM

III. Experimental Setup

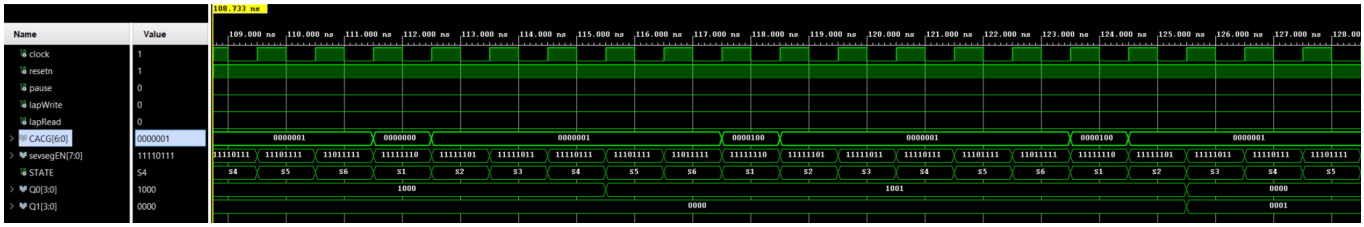
The functioning of most features (stop/start, lap write/read, resetn) for this project were verified directly by loading the project onto the board and testing the features. However, observing the behavior of the state machine as well as the correct execution counting a behavioral simulation is especially useful. There is, however, a problem with simulating this design directly; namely the timing of the circuit must be scaled down. The simulation below was generated using a clock with a period scaled down to 1ns from 10ns. Also, the modulo-1,000,000 counter was scaled down to a modulo-10 counter and the modulo-100,000 counter was scaled down to a modulo-1 counter.

IV. Results

In order to verify the counting functionality, we will take note of the simulation output of the Q outputs of the counters. For the sake of time, we will focus here on the two least significant digits. Please note, these simulation expectations

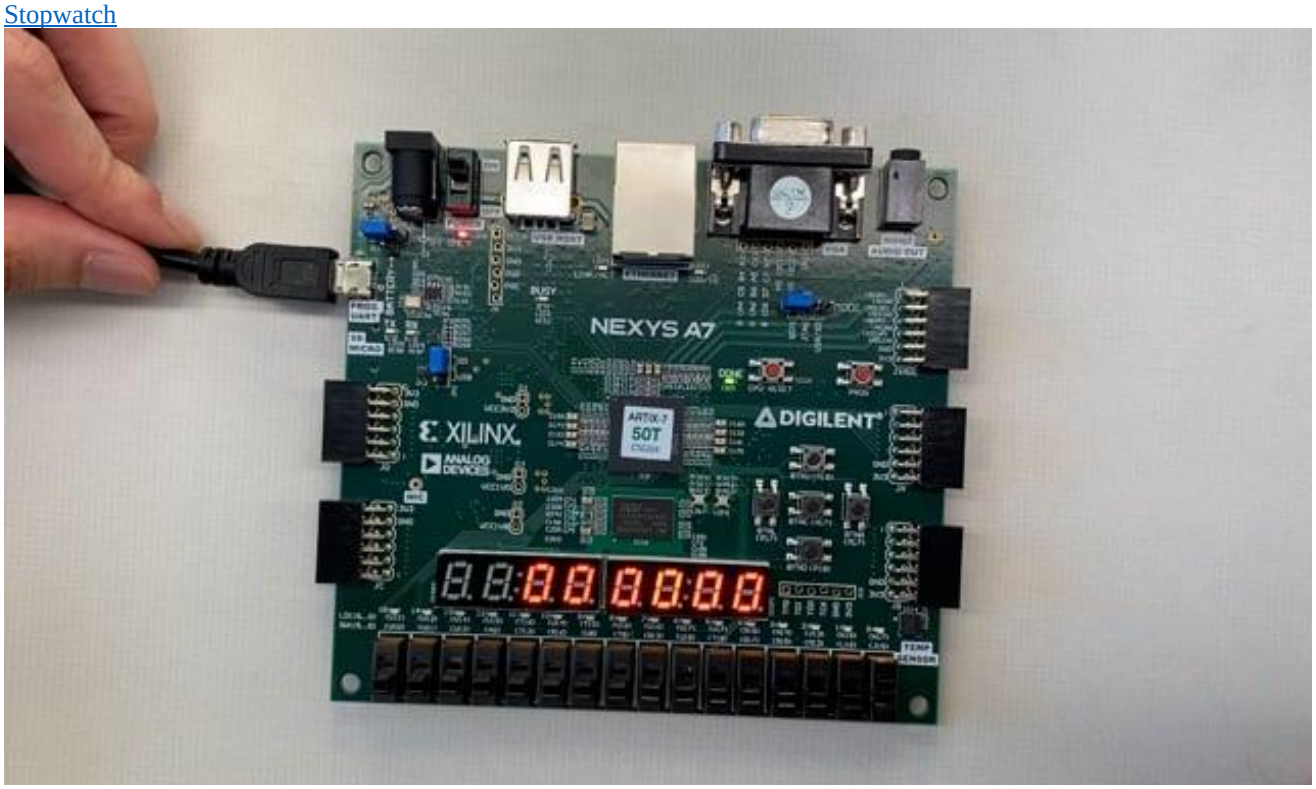
are utilizing the “scaled down” circuit mentioned in section III. The expected effect is that every 10ns the count of the first counter of the array will increment by 1 varying from “0000” to “1001” and repeat. Similarly, since the first counter restarts it’s count every 100ns, we expect the count of the second counter (Q1) to increment by 1 every 100ns, varying from “0000” to “1001” before starting over.

The expectation for the state behavior in this scaled down simulation is that the state should change every 1ns. Further, if the decoder is working correctly in conjunction with the state machine S1 should correspond to a “sevsegEN” value of “000001”, S2 should correspond to a “sevsegEN” value of “000010”. If this behavior is demonstrated in the simulation the seven- segment serialization is working properly. Comparing these expectations with the simulation results reveals that the design functions as intended. For the sake of formatting and clarity, the simulation results along with a video of the final project functioning properly are shown on the following page in order to present them in a clear aspect ratio.



Scaled-down Simulation Result (clock 1ns, Q0 increment every 10ns, Q1 increment every 100ns, state change every 1ns)

Below is a link to a video of the final fully functional result:
Stopwatch



Conclusions

The design of a digital stopwatch proved to be a deceptively challenging project with many great learning outcomes. Because there are very few inputs and outputs involved in a stopwatch, it seems at first glance to be a very simple and easy project to design. The circuit ended up with three main sections: counting, memory storage and recall, and display. Each of these sections needed to work together, necessitating the use of multiplexors controlled by an algorithmic state machine. One surprisingly difficult design problem was memory storage and recall. The first attempt we made was to capture the value coming out of the 6 to 1 counting multiplexer, however we then realized this would only record one digit value at any given time and therefore was not what we wanted to record. Potential improvements to the design would be the expansion of the lap functionality to record and recall multiple different lap times, as well as the inclusion of colons separating minutes from seconds and seconds from tenths of seconds. Another great outcome of this project was working with a group of designers to reach a specific goal. There were many times when the group was stuck on a problem, and it was very rewarding when one of the members arrived at a working solution.

References

[1] D. Llamocca, "VHDL Coding for FPGAs," http://dllamocca.org/VHDLforFPGAs.html