Stopwatch

Nicholas Rushaj, Tolin Faraj, Samantha Peters
Electrical and Computer Engineering Department
School of Engineering and Computer Science
Oakland University, Rochester, MI
e-mails: nicholasrushaj@oakland.edu, tolinfaraj@oakland.edu, samanthapeters@oakland.edu

Abstract— This project involves the design and implementation of a digital stopwatch that is meant to measure time in minutes, seconds, and hundredths of a second. Functions include start, pause, lap, and reset. The entirety of this project will be based on concepts learned throughout the course. The hardware testing confirmed the accuracy of the circuit diagrams and the proper functionality of the stopwatch. The design demonstrated the integration of timing circuits, inputs, and outputs. Main recommendations include adding a debouncer component to circuits to ignore fluctuations that may occur when pressing a button. Another recommendation is to use a finite state machine that can track how many laps you have stored.

I. INTRODUCTION

The goal of this project is to design a digital stopwatch using Vivado software, implementing it onto a Nexys FPGA Board. This report will consist of all the features and components that will be used to create the stopwatch itself. These features include a start, pause, lap, and reset. The components include the clock divider, decoders, edge detectors, counters, registers, and multiplexers.

The motivation for this project is to deepen our understanding of digital timing circuits and how to implement them into real world circuits. By developing a stopwatch, we can gain hands-on experience with all the concepts that were taught in the actual course. A lot of concepts that were from the course were applied to this project. This included component designs, VHDL codes, and how to apply all into the board for its actual use.

The applications for this project range from a standard stopwatch to much more advanced systems which can be further designed.

II. METHODOLOGY

We will start the methodology by explaining the functionality of each individual component in detail and continue explaining how all these components come together in the top circuit. It will not be mentioned again, so it should be noted that every component that can store values takes as an input the same active low CPU reset of the Nexys board.

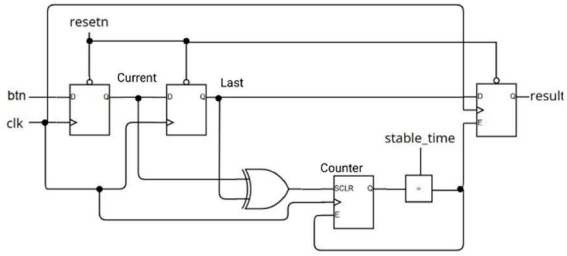

Figure 1: Debouncer

A. Each Component in Detail

We will start with the debouncer. The purpose of the debouncer is to ignore the fluctuations and bouncing of the input buttons. The structure of the circuit is shown in Figure 1. The circuit works by storing the last and current values of the button in D flip flops. There is a counter in the middle of the circuit. The sclr bit of the counter is connected through an xor gate to the current and last values. This means that the counter will continue to count up as long as the value of the button is stable. Once the output of the counter reaches a certain value, which will be a parameter of the circuit, the value of the output is allowed to be set to the value of the button. This is done using an equality detector circuit connected to the enable bit of a D flip flop whose D bit is the last value and whose Q bit is the result of the debouncer. We have chosen a value for the parameter that requires the button to be stable for 10 milliseconds to be valid. This creates a delay, but it is unnoticeable to the human eye. It also requires the button to be pressed for longer than 10 milliseconds to register, but the author of this section has not been able to press a button quickly enough for the

circuit not to notice. The idea of this debouncer circuit was inspired by [1].

The clock divider takes an input clock signal with a certain period and outputs a clock signal with a period that is an even multiple of the input period. We decided to use a process statement to design this circuit, rather than using a structural approach. An integer value is incremented on the rising edge of the input clock and resets to 0 when it reaches a certain value which is a parameter of the circuit. At the same time as the integer value resets, the value of the output clock inverts itself. The ratio of the periods of the input and output clocks is equal to the parameter value plus one multiplied by 2.

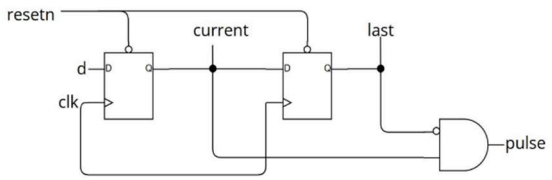

Figure 2: Synchronous Edge Detector

The synchronous edge detector is a simple circuit whose purpose is to output a one-clock-period-width pulse when it detects the value of the input going from low to high. This is accomplished, as can be seen in Figure 2, by storing the current and last values of the input in D flip flops. The output pulse then is equal to, in Boolean algebra, `current and not last`.

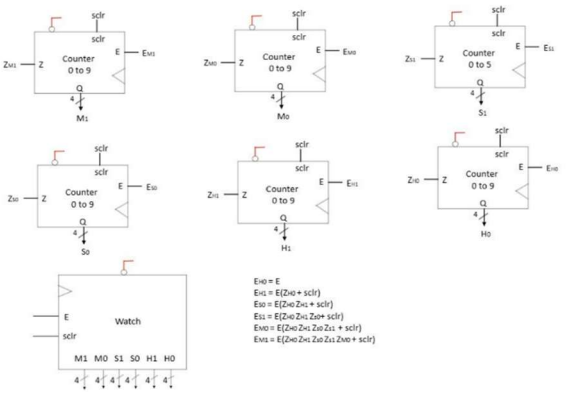

Figure 3: Watch

The watch is the heart of this project. The watch outputs 6 4-bit binary coded decimal (BCD) signals, the first two representing minutes, the middle two representing seconds, and the last two representing hundredths of a second. It takes as inputs a 100Hz clock, an enable bit, and a sclr bit. If the enable bit is on, on the rising edge of the clock, the time output by the watch increments by one hundredth of a second. This is accomplished, as can be seen in Figure 3, with 6 4-bit modulo counters. The maximum value of each clock is nine ("1001" in BCD), except for the tens digit of the seconds, whose maximum value is five ("0101"). The enable bit of each counter is high only when the maximum value of each counter before it has been reached. This way, for example, in a time of 00:15.23 the ones digit of the hundredths is the only value that will increment, but with a time of 01:59.99, all values but the tens digit of the minutes will increment. Just as has been taught in this class for registers, enable takes precedence over sclr.

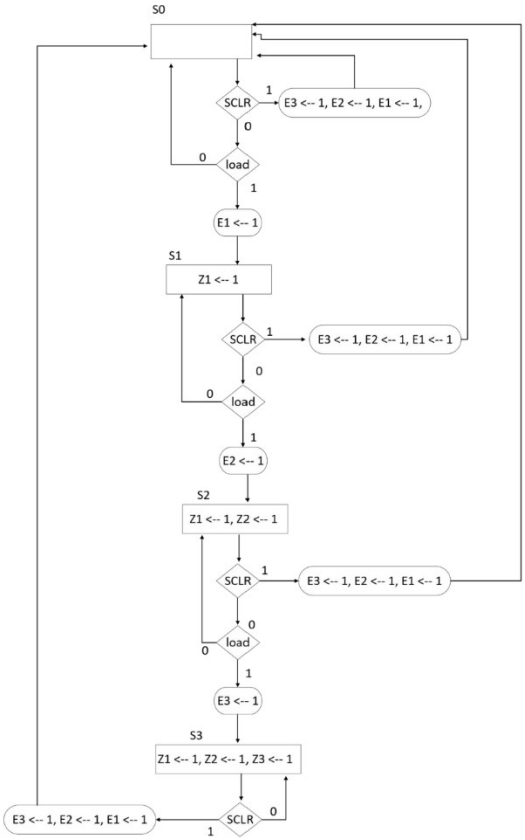

Figure 4: ASM Diagram

There is a finite state machine (FSM) used to control how the laps are stored. There are two inputs: load and slcr—and 6 outputs: E3, E2, E1, z3, z2, z1. There are four states, S0, S1, S2, S3, each corresponding to how many laps have been stored (more on this later). The transitions and outputs of the FSM are shown in the ASM diagram in Figure 4.

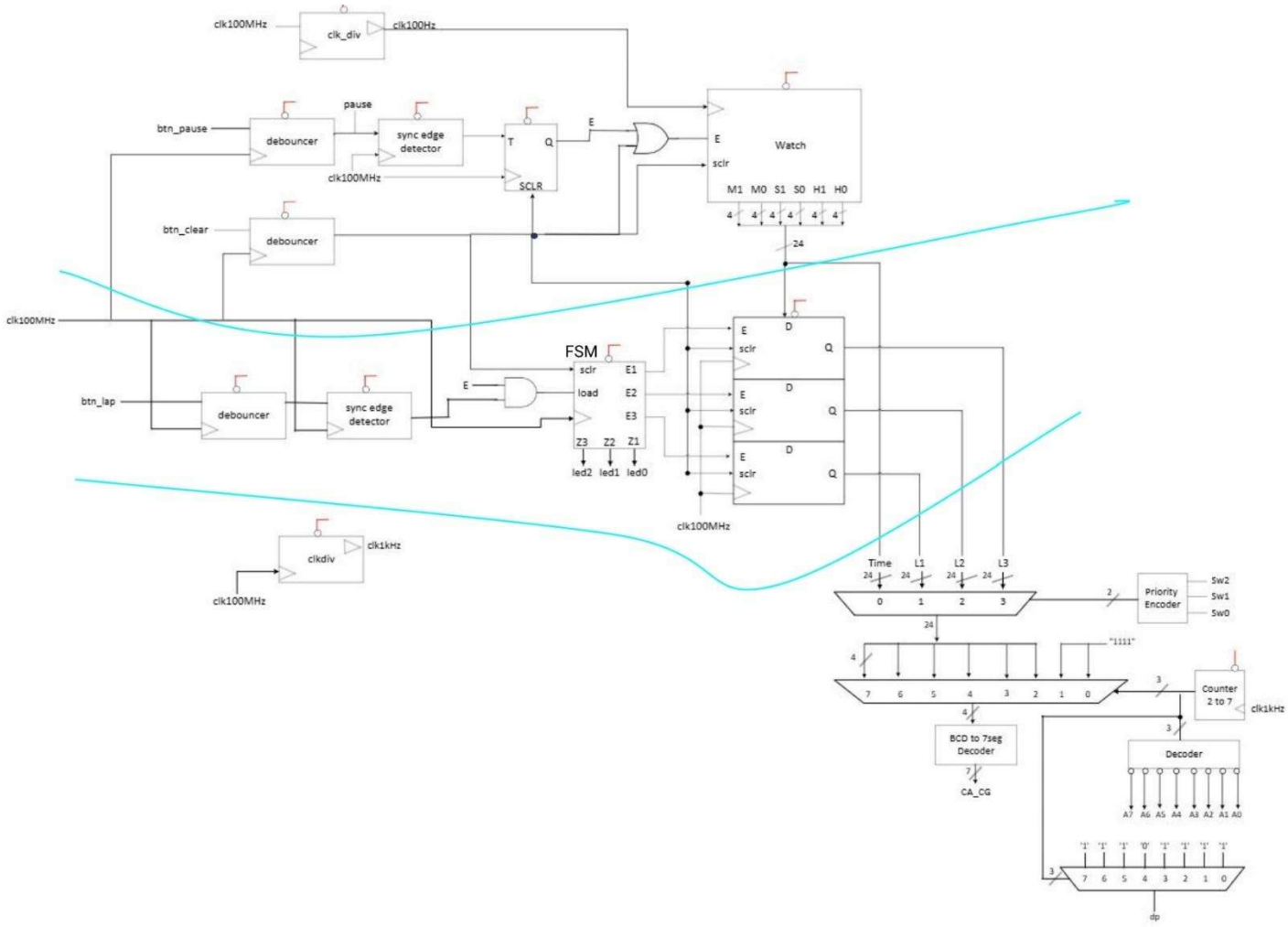

Figure 5: Top Circuit, Divided into Three Parts

There is a priority encoder in the circuit, but it is not a regular priority encoder. An input of “000” outputs “00”. “001” outputs "01”. “01X” outputs “10”. “1XX” outputs “11”. There is no z bit. The purpose of this component will be explained in a later section.

Other circuits designed in this project include D flip flops, T flip flops, registers, counters, a BCD-to-7seg decoder, muxes, and adder circuits, but these are considered fundamental enough to not need explanations.

B. Top Circuit

The top circuit, as can be seen in Figure 5, can be divided into three main parts: the watch control, the lap control, and the 7-segment module control. It should be noted that every synchronous circuit, unless otherwise mentioned, is connected to the 100 MHz clock.

- Watch Control

The main purpose of the watch control is to produce the time of the stopwatch is BCD. There are three important inputs here: the 100MHz clock, a pause button, and a clear button. The 100MHz clock is connected to a clock divider to create an output clock of 100Hz (T = 0.01 sec). This 100Hz clock is connected directly to the watch circuit. Both buttons are connected to debouncers. The output of the pause debouncer is connected to a synchronous edge detector, and the output of the synchronous edge detector is connected to a T flip flop. The output of the T flip flop will from here on simply be called E. The purpose of the synchronous edge detector and T flip flop is so that pressing the pause button once will have the effect of simply pausing or unpausing the watch, no matter how long the button is held down. The signal E essentially tells us whether or not the watch is paused. The output of the clear

debouncer, which will from here on simply be called clear, is connected directly to the slcr bit of the watch circuit. The clear signal is connected to the sclr bit of the T flip which outputs E. This is because we want resetting the stopwatch to also pause it. This creates a problem, since the watch will only clear when the enable bit is high. This is why the enable bit of the watch is not connected directly to E, but equal to, in Boolean algebra, `E or clear`. The 6 4-bit outputs of the watch are concatenated into one 24-bit signal which will from here on be referred to as watch_time.

- Lap Control

The main purpose of the lap control is to allow the user to store laps. The important inputs to the lap control are as follows: the lap button, the clear signal, and the E signal. There are three 24-bit registers which store the vectors L1, L2, and L3, which are the laps. The D vector of each register is connected to watch_time and the sclr bit of each register is connected to clear. The enable bits of each register are connected to E1, E2, and E3. These values are controlled by the FSM. The lap button is connected to a debouncer whose output is connected to a synchronous edge detector. The purpose of this is so that pressing the lap button will produce a single one-clock-period width pulse that will allow only one of the lap registers to be set. If we call the output of the synchronous edge detector lap_pulse, then we can define a signal lap_pulse to be equal to `lap_pulse and E`. The purpose of this is so that the user doesn't have the ability to store laps when the stopwatch is paused (this was a decision based on the way a typical cellphone's stopwatch works). lap_load will then be connected to the load bit of the FSM and clear will be connected to the sclr bit of the FSM the z1, z2, and z3 bits of the FSM are connected to led0, led1, and led3 on the Nexys board. The FSM assures that the following criteria are met: at any time, if the stopwatch is reset, all the lap registers are reset. If you have a register left for storing laps, pressing the lap button stores the current time in the next available lap register, unless the stopwatch is paused. If there are no lap registers, the lap button does nothing. The LEDs on the Nexys board help the user to know how many laps have already been stored. That is, no LEDs on means no stored laps, one LED on means one stored lap, etc.

- 7-Segment Module Control

The purpose of the 7-segment module control is to allow the user to choose whether he or she wants to see the current time or one of the laps and then to show what the user chooses on the 7-segment module on the Nexys board. The principal inputs to this part of the circuit are sw2, sw1, and sw0 from the Nexys board, watch_time, L1, L2, L3, and the 100MHz clock. The principal outputs are the 7-bit vector CA_CG, the 8-bit vector A, and the bit dp. The switches are connected to the priority encoder explained above. The output of the decoder decides whether the time will appear on the 7-segment module or one of the laps. No switches being on means the current time is visible. Sw2 being on (regardless of whether sw1 and sw0 are on) means lap 3 is visible, etc. The purpose of doing this is instead of choosing the lap with just two switches interpreted in binary is because the average user probably doesn't know binary. The 2-bit output of the decoder is then connected to the sel port of a 4-to-1 24-bit bus mux. watch_time is connected to the 0 port, L1 to the 1 port, L2 to the 2 port, and L3 to the 3 port. The output of this mux will simply be called time_viz.

The way a typical 7-segment module works is that it is impossible to show different numbers on different parts of the module at the same time. But it is possible to turn all the digits off but one, show the number that corresponds to that digit, turn it off and then turn the next one and show the digit that corresponds to that digit, etc. Doing this quickly enough gives the human eye the illusion that all the digits are on at once. To do this, a clock with a period of 1 millisecond (frequency of 1000 Hz) is needed. The 100 MHz clock is put into a clock divider which outputs a 1KHz clock. The 1KHz clock is connected to the clock bit of a 3-bit counter. This counter counts repeatedly from 2 to 7. The output of the counter is connected to the select port of an 8-to-1 4-bit bus mux. The Nexys board has 8 digits, but only 6 are needed, which is why the clock only counts between 2 and 7. The signal time_viz is separated into 6 4-bit signals which input into the ports 7 down to 2 of the mux. The ports 1 and 0 are connected to a constant value of "1111". The output of this mux will be called p. The output of the 2-to-7 counter is connected to an active low decoder, whose output is connected to the vector A. The output of p is connected to an active low BCD-to-7seg decoder. The output of the 7seg decoder is connected to CA_CG, which tells the board which segments to turn on and

which to turn off. The result of this is that the value of time_viz is constantly displayed on the 7-segment module. The output of the 2-to-7 counter is also connected to the select port of another 8-to-1 mux. All the input ports of this mux are connected to a constant '1', except for the 4 port which is connected to a constant '0'. The output of this mux is dp, an active low bit which tells the Nexys board when to turn on the decimal point on. All this has the effect of constantly keeping the decimal point after the 4th digit (the ones digit of the seconds) on.

III. EXPERIMENTAL SETUP

We used the software Vivado to write vhdl code to implement our circuit. We also wrote a testbench in vhdl to simulate the circuit and test it. But a vhdl testbench in vivado was not enough to test our circuit for two reasons. First, to make sure our code would work correctly, we'd have to run a simulation that lasts multiple minutes, and Vivado simulates everything by the picosecond, so running the simulation properly would take a very long time. Second, to test the debouncer circuit, we'd have to see how it works on a real physical button. For this reason, we had to implement our circuit on a Nexys board to test it. The board we used was an Artix-7 XC7A100T-1CSG324C. The expectation was that everything would work properly.

IV. RESULTS

The results included a functional stopwatch displayed on the FPGA board. It ended up having 3 buttons, one to start and stop the time, one to clear everything, and one to set a lap (with a maximum allowance of 3 set laps). In addition to the buttons the stopwatch time was displayed on the 7-segment display and there were 3 switches to toggle between the current timer and the saved laps. All buttons worked as intended. As the code was developed, we ran into some challenges. For example, we had trouble designing the debouncer, which is why we had to get help from the internet.

From the actual design, we saw how FSMs could be more useful than regular digital circuits. From lecture, we were able to incorporate it into our design and make it work.

CONCLUSIONS

Our main takeaway was that the stopwatch was slightly more complex than we originally thought. Drafting and implementing a way to store the laps and be able to return to them was something that took us more time than we expected. The possible improvements we considered were adding more laps, designing it so trying to look at a lap before it has been stored will result in flashing lights, and adding a button to reset the laps but nothing else. All in all, we did learn a lot and were able to create a simple functioning stopwatch that was capable of basic time tracking.

REFRENCES

[1] Scott_1767, DigiKey Employee, https://forum.digikey.com/t/debounce-logic-circuit-vhdl/12573