ECE 2300 Lab 2 Report

Max McCarthy (msm296)
Raghav Kumar (rk524)
Robert Zhang (rdz26)

Introduction

Purpose:

As a natural extension of combinational logic, this lab explores the applications of sequential logic and memory by designing a stopwatch. Sequential logic takes the values generated by combinational logic and adds a time component. For example, outputs of combinational logic can be stored for later use. In the case of the stopwatch, memory can be used to store the numerical value of each digit so that they can be displayed, as is done in this lab with the help of a DE0-CV board, or for other applications.

Problem Summary:

The goal is to design a stopwatch that counts up to 10 minutes with a precision of centiseconds, then resets. Since this is sequential logic, the first step is to develop a way of storing data. A single bit T Flip-Flop serves as the foundational component of the stopwatch. Since one bit is not nearly enough information, these Flip-Flops can be combined into a 4-bit register. With four bits, there is enough information to assign each decimal digit of the stopwatch a unique value. Each register can be used to store the information of the five stopwatch digits: Minutes (1), Seconds (2), Centiseconds (2).

Tasks to Complete:

1. Create a T Flip-Flop module
2. Create a 4-bit T Flip-Flop Register module
3. Create a counter module
4. Upload to a test board

Tools Used:

1. Quartus II
2. DE0-CV Board

End Result:

Our implementation of the stopwatch worked as expected. It counts up to 10 minutes, then resets. Using the board, we were able to display the digits in real time and, by adjusting the clock frequency, were able to change the speed of the stopwatch. To get it to output the true time, the clock frequency must be set accordingly.

Circuit Description

The core of this circuit is the T Flip-Flop, which is a simple flip-flop that toggles at the positive edge of the clock signal when its toggle input is high. Four of these are connected to create a 4-bit T Flip-Flop register, which has a 4-bit input connecting to the toggle inputs, as well as a 4-bit output connecting to the outputs of the four individual flip-flop registers. Each 4-bit T Flip-Flop is connected to a counter module that contains combinational logic which causes higher-significance bits to toggle exclusively when all the lower-significance bits equal "1", thus creating a binary counter up to 15. Each of these counters are then connected to Boolean logic that causes each counter to toggle and reset at the appropriate time (e.g. for centiseconds, the counter is toggled every clock cycle, while for seconds, the counter is only toggled when centiseconds and deciseconds are both 9 and are about to toggle again). The value within each counter can then be displayed as a stopwatch. For all of the above modules, there is a reset function that allows the value inside to be reset to 0, either when needed to (e.g. when numbers carry over to the next digit), or when the circuit needs to be restarted.

Inputs/Outputs

Note: Multiple-bit input and output digits with greater significance have a higher bus index.

Stopwatch Inputs:

The inputs into the entire stopwatch circuit are as follows: CLK, which provides the clock signal for the entire circuit; RESET, which causes the entire stopwatch to reset, and causes all of the values of the various counters, etc. to reset to 0; and ENABLE, which only allows the clock to toggle when it is high, thus providing a way for the user to pause the clock.

Stopwatch Outputs:

The outputs from the entire stopwatch circuit are as follows: CENTISEC, which connects to the output of the centisecond counter; DECISEC, which connects to the output of the decisecond counter; SEC, which connects to the output of the counter for seconds; TENSEC, which is the output of the counter that handles the tens digit of the stopwatch; MIN, which is the output of the minute counter; and TIME, which collates all of the above outputs, and gives them out as a single 20-bit output.

Counter Inputs:

The inputs into the counters are as follows: CLK, which is the same as CLK in the overall stopwatch; CLR, a low active input which when activated causes the counter to reset back to 0000; ENP and ENT, both of which have to be high for the counter to toggle. This allows us to use one of the inputs (namely ENP in the lab) to be connected to the stopwatch's ENABLE input, while using the other one to signal when the counter should toggle, (e.g. with our stopwatch, they increment based on how the other counters are incrementing).

Counter Outputs:

The sole output from each counter module is Q, a 4-bit output that gives the binary representation of the number currently stored within the counter.

4-bit T Flip-Flop Register Inputs:

The inputs into the registers are as follows: CLK, which provides the clock signal to the register; RESET, which when activated causes all of the flip-flops to reset back to 0; and IN, a 4-bit input where each bit connects to the toggle input for a different flip-flop in the register.

4-bit T Flip-Flop Register Outputs:

The sole output from each register is OUT, a 4-bit output, where each bit connects to the output of a T Flip-Flop.

T Flip-Flop Inputs:

The inputs into each flip-flop are as follows: CLK, which provides a clock signal; RESET, which sets the flip-flop's output to 0; and T, which toggles the flip-flop.

T Flip-Flop Outputs:

The sole output is Q, which is the current state stored within the T Flip-Flop.

<u>**Circuit Functions**</u>

Stopwatch Function:

The primary function of the overall stopwatch module is to coordinate the various counters, in order to allow each one to increment at the right time to represent the various digits of the stopwatch (e.g. more significant digits increment only when all the less significant digits are "maxed out" at 9, or at 5 for ten-seconds). It also serves to send a reset signal to all of the counters, if its RESET input is on.

Counter Function:

The primary function of the counter modules is to coordinate the timing of the 4-bit T Flip-Flop register contained within each counter. In particular, it makes it so more significant bits of the register are toggled exclusively when all of the less significant bits are high, allowing the 4-bit output of the register to represent binary numbers incrementing upwards, with each clock tick. Another function of the counter is to restrict incrementing exclusively to when the inputs

ENP and ENT are on, allowing us to control when the counter increments (through the overall stopwatch module), and to pause the stopwatch if necessary.

4-bit T Flip-Flop Register Function:

This module serves to collate the inputs and outputs of four T Flip-Flops into a single module, while also provides a common clock signal to each of the flip-flops.

T Flip-Flop Function:

This module is a simple T Flip-Flop, where a high input signal causes the output Q to change at the positive clock edge.

Modules Used

For each portion of this lab assignment, we sought to break the circuit down into manageable modules (particularly if there were repeated portions of the circuit, for example with counters, and with the T Flip-Flops). As such, these modules each correspond with the components of the circuit described above, greatly facilitating assembling all of the components together at the end of the lab.

Path of Data Flow

At any moment, if RESET is high for the overall stopwatch module, then it passes through the CLR input of each counter (passing through a NOT gate in the process), which then flows to the RESET input of the register inside the counter (again passing through a NOT gate in the process), which in turn connects to each T Flip-Flop's RESET input, which causes them to reset to 0 on the next positive clock edge.

When ENABLE is on for the stopwatch, it causes all of the ENP inputs to be high for all of the counter modules. That allows for the counters to toggle when prompted to by the clock and the values of the other counters (this in particular connects to ENT). When ENP and ENT are both high, then the counter can count upwards. In the counter, AND gates are used to combine these two inputs with various flip-flop outputs in order to toggle each flip-flop only when required, by turning the appropriate index of the IN input of the 4-bit flip-flop register on, which makes the corresponding T input of the individual flip-flop high, toggling it.

Each T flip-flop's output flows into a register, which groups four of them into a bus and outputs them through its OUT output, which in turn is connected to each counter's Q output, which passes into the overall stopwatch module, which outputs the value given by Q for the digit of the stopwatch that the counter represents. All of these stopwatch outputs are collected into a single bus, and outputted as the TIME output.

Implementation & Testing

Counter Implementation:

The 4-bit tcounter acted as the basic unit of sequential logic for the overall stopwatch. To implement the tcounter, we used a module consisting of four 1-bit T Flip Flops. A T Flip-Flop toggles when the T input is a 1. It has an active high RESET input which sets the output Q to a 0 when 1. Adopting an incremental approach, we then made a treg4bit module consisting of four T Flip-Flops that shared the CLK and the RESET. This was also the first time that we declared an instance of a smaller component to build more complex modules. Having implemented the treg4bit module successfully, we went on to use that in the tcounter. However, the treg4bit module's T Flip-Flops toggled independently while the tcounter required them to trigger in a specific manner such that a particular bit only toggled when all the bits on its right became a 1. Along with this specific function, the tcounter also had additional constraints such as an active low Clear and the ENT, ENP inputs. The clear input set the output of the tcounter to a 0 and thus had to be inverted before being connected to the reset of the four T Flip-Flops. Also, the tcounter only counted when both the, ENT and the ENP bits, were a 1. Thus, after adding appropriate combinational logic for the bits to toggle appropriately and for the counter to start counting only when the ENT and ENP bits were a 1, we readied an appropriate counter for our stopwatch.

Overall Stopwatch Implementation:

The stopwatch we implemented in this lab displayed 5 digits with the smallest time being one centisecond and the largest being 9 minutes, 59 seconds, 99 centiseconds. Each digit ran a tcounter under the hood and since the digits went up to a maximum of 9; decimals from 10 to 15 never got used. We had to use a 4-bit tcounter because 8 would have been the maximum a 3-bit tcounter could display, which would have not met our stopwatch requirements. In due time we figured the conditions required for a each digit of the stopwatch to trigger. For example, the centisecond digit triggered at each clock cycle while the subsequent digits only incremented when the previous value got to a 9. The minutes digit however triggered when the tenseconds digit became a 5 and the rest became a 9. Similarly, all the digits also had specific reset conditions. The centisecond digit resetted when it reached a 9. The minutes digit resetted when it became a 9 in addition to the tenseconds digit being a 5 and the rest of the digits being a 9. Thus, we implemented the overall stopwatch by declaring 5 instances of the tcounter and implementing specific logic for the triggering and resetting of each digit. After a few minor bugs, we successfully ran our stopwatch on the DE0-CV Board.

4-bit T Flip-Flop Register (treg4bit.v) Implementation:

A 4-bit input where each bit connects to a flip-flop’s T input and a 4-bit output where each bit connects to a flip-flop’s Q output.

Supplemental Diagrams:

Overall stopwatch diagram:

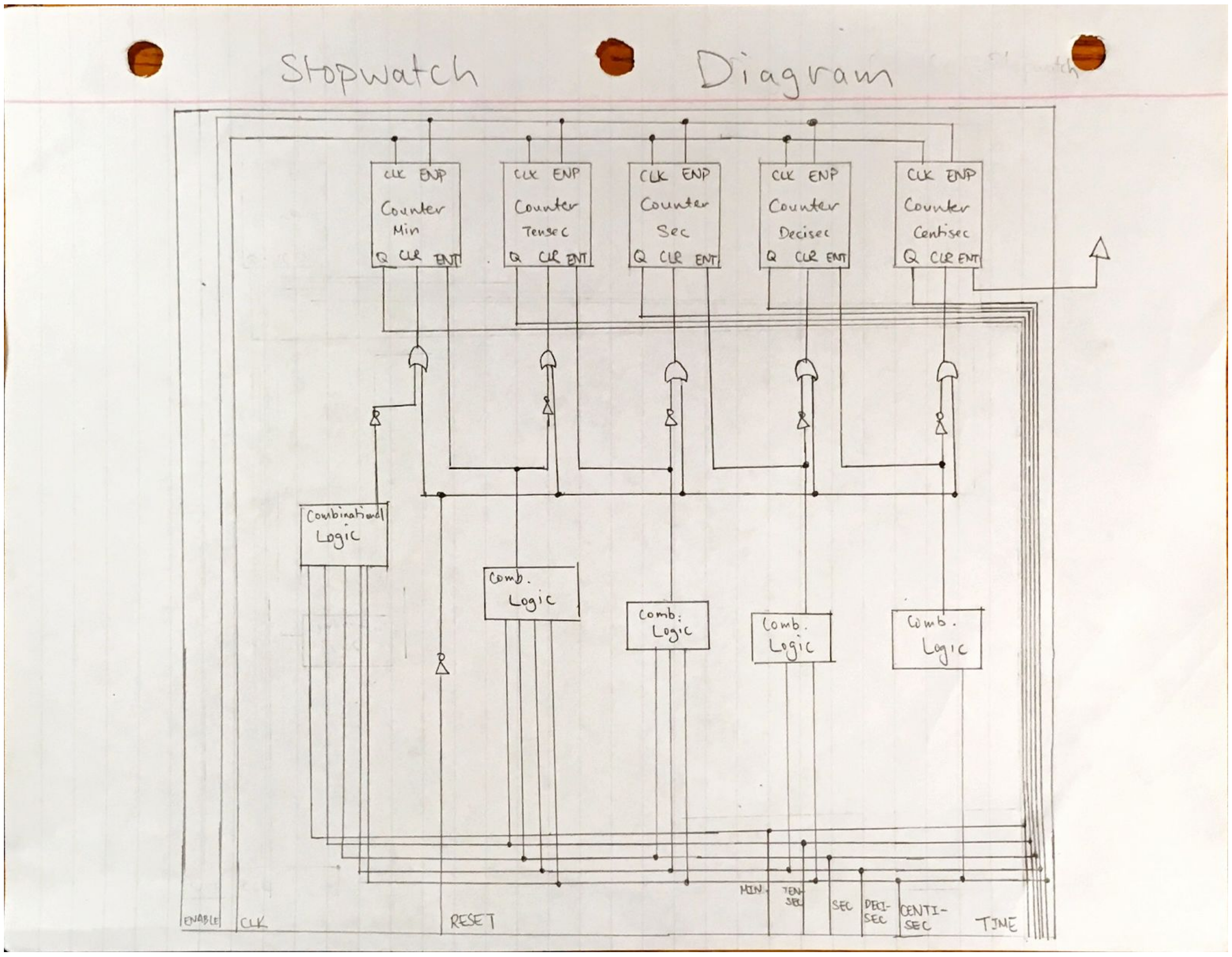
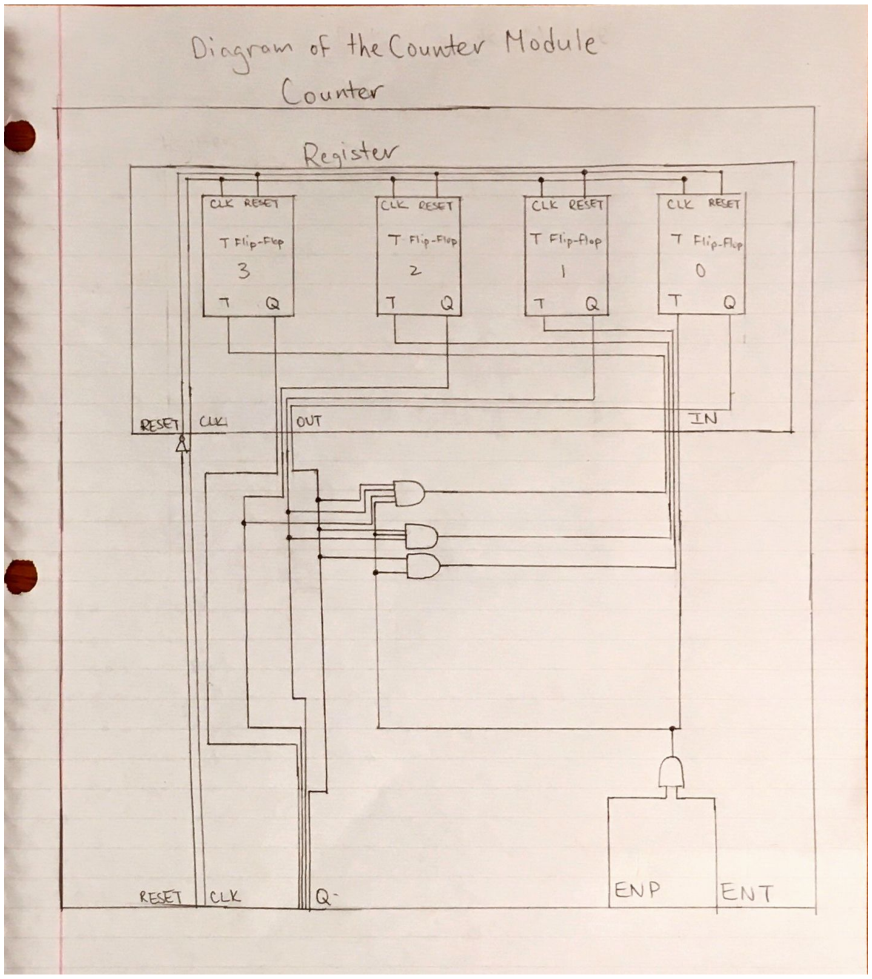


Diagram of a counter module:



Explain Test Cases of "lab2_test.v":

1. Test reset: Tests that reset clears the registers
2. Test incrementation: Compares register values to expected time values
3. Test hold: Tests that register values are held (pauses stopwatch)
4. Test hold after reset: Tests that register values stay zeroed after reset when held
5. Test incrementation after reset: Tests that counter increments properly after reset

Additional Comments:

It would not be possible to implement the counter without using sequential logic. This is because each counter has to store its value and hold it, until toggled by the stopwatch circuit; this is not possible in combinational logic, which is determined exclusively by the current state of the inputs.

Modifications

We implemented all the functions as said in lab file. We did not create any extra modules or functions.

Conclusion

Thus, as can be seen from the above sections, our group successfully implemented a timer using a counter in this lab. The lab was particularly helpful as it introduced us to Quartus and Verilog, taught us how basic combinational and sequential logic can be programmed and how that in turn can be used to make more complex designs. Having covered sequential logic in class, the lab helped us implement that logic in real life, first through simple T Flip-Flops and then through a counter and a stopwatch.

We undertook a building block approach in implementing our functions in this lab. First we created a basic T Flip-Flop, one of the fundamental unit of sequential logic. We then went on to create a 4-bit register using 4 of the T Flip-Flops we had created earlier. We learnt how smaller units, if synchronised correctly, could help us store larger data. However, even a 4-bit T Flip-Flop had limited capabilities. If we wanted to build a stopwatch, we would need something that could increment and reset to 0 without outside intervention and this led us to create the counter. Creating a 4-bit counter and then synchronising 5 such counters finally allowed us to build our stopwatch that ran successfully at the end of the lab.

Report Distribution

Max: Report Template and Formatting/Editing, Introduction, Test Cases

Raghav: Implementation & Testing, Modifications, Conclusion

Robert: Circuit Description, Inputs/Outputs, Circuit Functions,
Modules Used, Path of Data Flow