ENGR 272 Lab 4: Counters

4.1 Background Information

Lab 3 developed a design to drive a single digit on a seven-segment display. Lab 4 will go further than that to use a clock to drive all 6 digits of the display. The board we are using has a bus to all the segments of all six digits, so they can all be lit at once. Two new logic symbols will be used in this section of the lab. The T flip-flop (tff) and the T flip-flop with enable (tffe). These flip-flops can be used to create an effective counter with the data stored in their latches.

Each new T flip-flop put in sequence toggles with half the frequency of the previous one. When four of these flip-flops are in sequence, it will take 2^4 cycles of the original clock to toggle through all the states. The four bits stored across the TFFs make a counter for one digit of HEX. The T flip-flop is a symbol in Quartus, you do not have to make it.

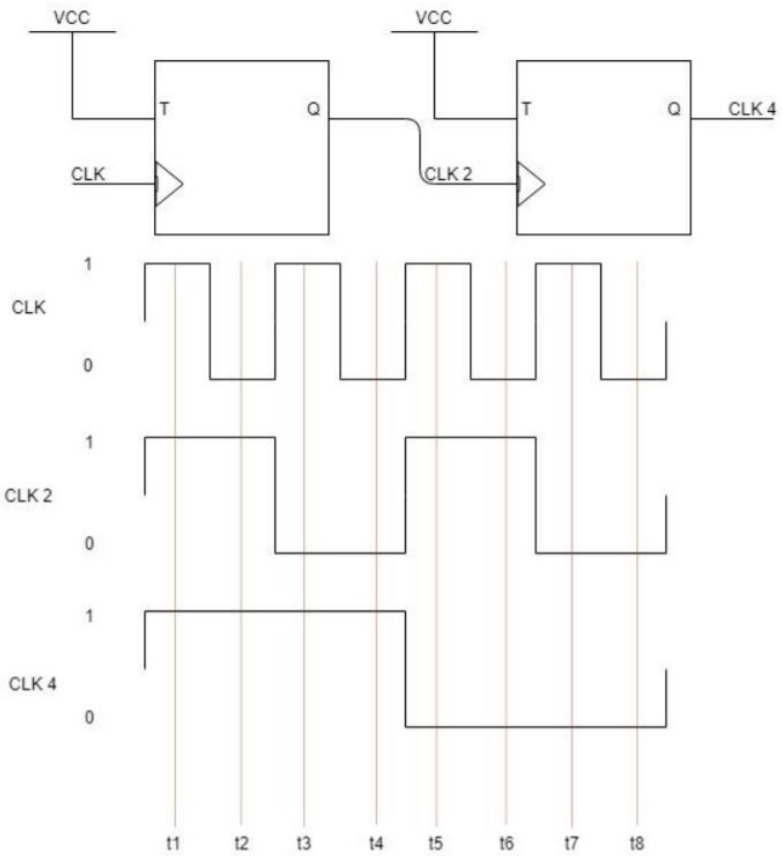

Figure 4.1: The T Flip-Flop(TFF) used to make a counter. When put in series, each TFFs cuts the clock speed in half.

4.2 Section Overview

Objective: Design and implement a system on the FPGA using TFFs that slow the clock as close to 1Hz as you can get and output a counter to each digit of the display.

1. Create a new project.
2. Create a software-level block diagram to describe the Sequential logic required for this section.
3. Create a .bdf file for the top level.
4. Create a .bdf file for any other modules in your block diagram.
5. Enter the Design.
6. Program and test hardware.

4.3 Learning Objectives

In this section, the following items will be covered:

1. Creating software-level block diagrams.
2. Sequential logic.
3. Clock Modulation.

4.4 Materials

1. Quartus Prime Lite Edition V. 18.0/18.1
2. DE10-Lite kit with MAX10 10M50DAF484C7G FPGA
3. USB to USB-B cable
4. The ECE 271 textbook, Digital Design and Computer Architecture by Drs. David and Sarah Harris

---

4.5 Procedure:

There are 6 steps to digital logic design

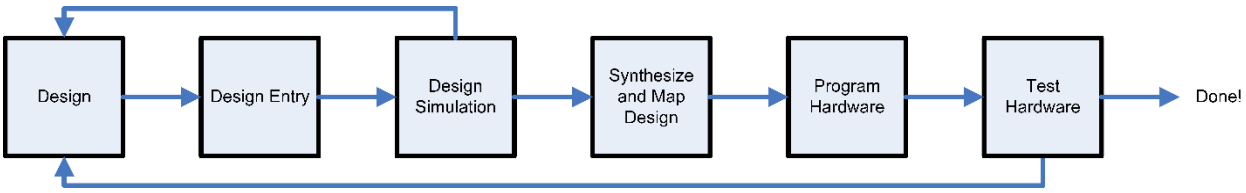

Figure 4.2: Use this process for designing section 4.

1. **Design:** The context of the design is established in this step. The context involves defining the inputs, desired outputs, and all the logic required in-between. In this step, all the minimizations and layout are planned for the design entry process. While this step is not always the longest, it should involve the most thought and effort. This typically requires a complete block diagram

showing all the logic blocks and the connections between them, often with written explanations of specific functions.

2. **Design Entry:** The actual drafting of the digital logic design occurs in this step, translating the design from block diagrams and descriptions into the software. This can be accomplished directly by writing HDL code, or graphically by drawing a schematic that a software tool can convert into HDL code.
3. **Design Simulation:** Before committing to hardware, this step tests the design in a controlled computer simulation. If the design does not function as specified in the "Design" step, it is revised.
4. **Synthesize and Map Design:** When the design simulates correctly, the HDL and schematic source files are synthesized into a design file that can be written to the FPGA. This includes assigning the inputs and outputs of the design to IO pins.
5. **Program Hardware:** After the design file is created, it is used to configure the FPGA. Quartus Prime sends a bit stream over the USB-B cable to configure the DE10-Lite FPGA.
6. **Test Hardware:** Verify hardware operation once the FPGA has been programmed. The FPGA should operate exactly as the design predicted, which was verified by the simulation. Synthesis problems, timing violations, or incorrect assumptions about the hardware can require the designer to return to the "Design" step.

4.6 Design

When the clock on a TFFE or TFF is rising, the T input is high, and the enable is high, the output signal will toggle its value.

CLOCK	RESET	T	Q
X	0	X	0
1	1	0	Q
1	1	1	~Q
0	1	X	Q

Table 4.1(T flip-flop)

By sending a clock signal into the clock port and connecting T to VCC (Constant logic high), the output becomes a clock with half the frequency of the input clock.

4.6.1 Understanding Flip-flops

Implement in Quartus Prime.

Use the T flip-flop symbol (tff). Connect one switch to the CLR Pins, and the PRN Pins to switches.

CLK should be input from a button or switch. CLK2 and CLK4 are the outputs, connect them to LEDs or the least significant bits of the 7 seg display decoder from lab 3.

Program the board and experiment with the functionality of the T Flip-flop. Does it follow the truth table in figure 4.1?

4.6.2 Make an HDL 7 seg display decoder from lab 3

Rather than importing the 7-segment display decoder from lab 3, we will simplify things by using an HDL 7-segment display decoder. We made a System Verilog version of the 7-segment display decoder In ENGR 271 homework 5 (sevenseg,sv). Copy this file into your Lab directory and load it into your project. To make a symbol that can be used in your schematics, use the same process that was used in Lab 2. To generate a symbol from the sevenseg.sv file, first select the file, then by selecting File→ Create/Update →Create Symbol file for current file. This will add a new symbol into the [Project] library of the Symbol menu in the schematic editor.

4.6.3 One Digit of Hex

Make a new module for a counter of one digit of hex. It should input a clock and a reset signal. It should output a 4-bit bus of data and a clock.

The output (Q) of the 4th T Flip-flop is both the 4th data bit and the output clock. Connect this to the 7 seg display decoder and use a button or switch as the clock to step through F-0.

4.6.4 Make the Counter Count-Up

At this stage, you will notice that the counter will count down rather than up. Use Figure 4.3 to help understand how to make the counter count-up. The difference between what you have now, and the Four-bit binary ripple counter shown below, is the inverted inputs to the clock of the T flip-flops. In other words, just add inverters to the clock inputs and the counter will count up.

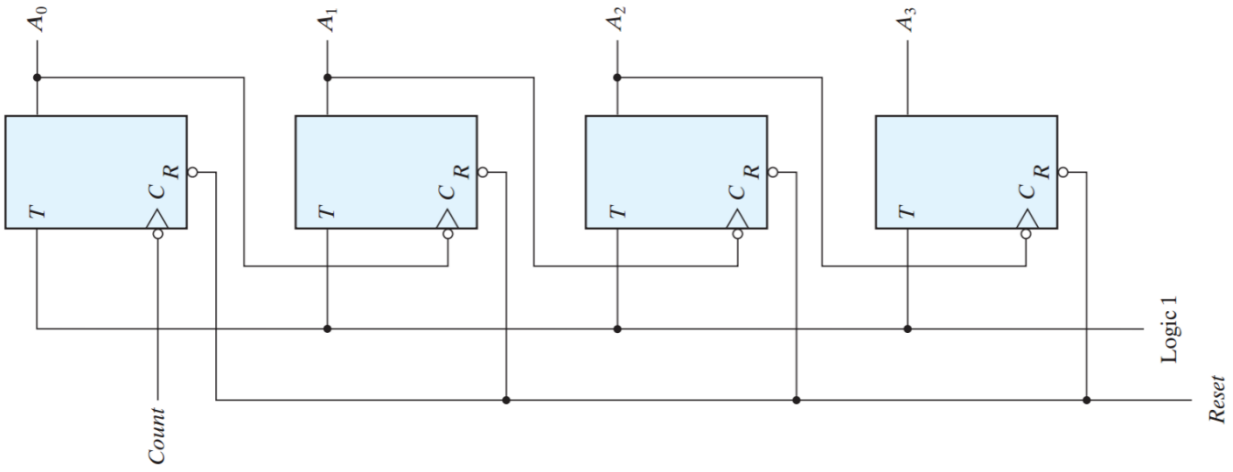

Figure 4.3: Four-bit binary ripple counter with T flip-flops

Connect up your four-bit binary ripple in Quartus similar to what is shown in Figure 4.4 shown below. Notice how the bus is connected. The wires to be connected to the bus are labeled from the least significant X[0], X[1], X[2], and X[3] as the most significant. On the bus wire, it is named X[3..0] indicating the order of the connection of the wires as 3 down to 0. If you label it as X[0..3] the wires would be flipped.

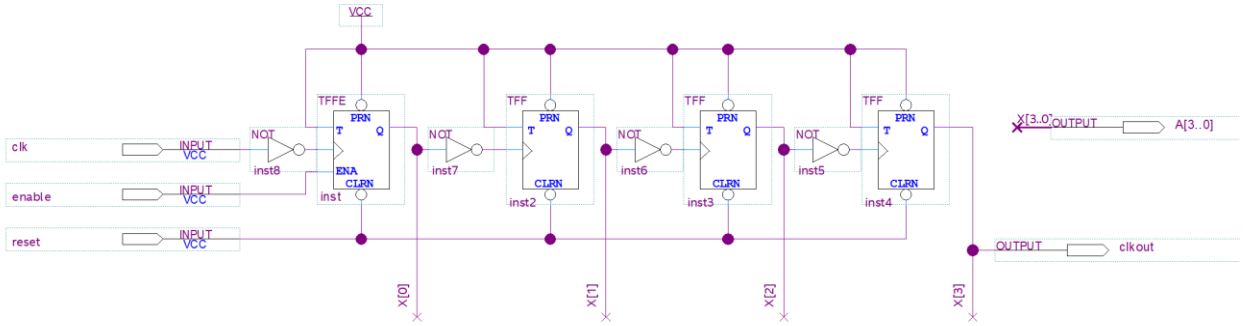

Figure 4.4: Four-bit binary ripple counter with T flip-flops implemented in Quartus

4.6.5 Driving all six digits

Now connect the 10MHz or 50MHz clock to the clk input of the first T flip-flop.

Your design must chain together at least 24 tffs (4 for each of the 6 digits of the display), and pipe each of their outputs into an instance of the decoder from lab 3. See Figure 4.4 for an example of how the 4-bit counters, connects to the seven segment decoder and each of the seven-segment displays. The first TFF used should be a TFFE, with one button connected to the enable, and the other button should be connected to the reset signal (clear) of every tff in the system. It is functionally equivalent to have the first T serve as the enable (and not use a tffe).

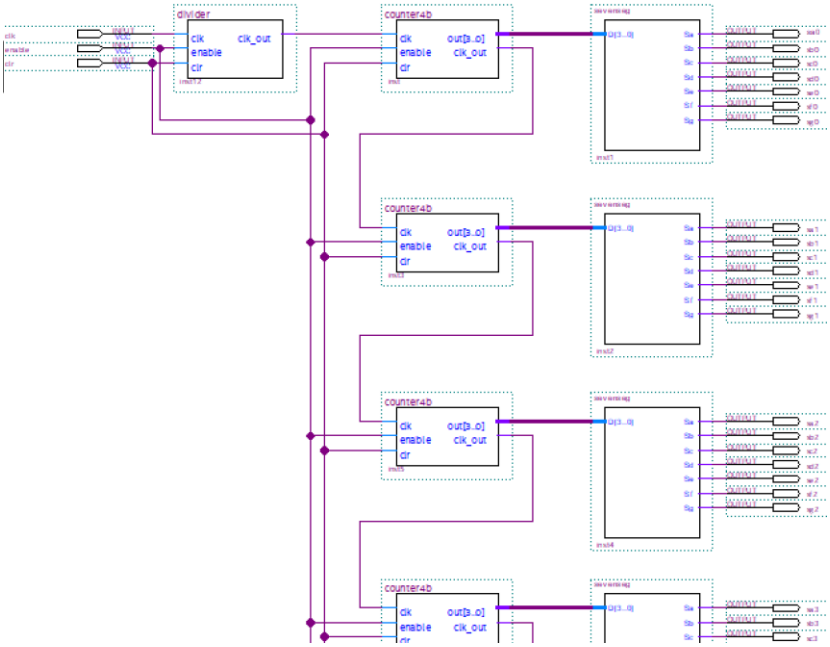

Figure 4.5 Example of how the 4-bit counters connect to the seven segment decoder and to each of the seven-segment displays.

The divider block you see in Figure 4.5 is additional T flip-flops connected like the 4-bit counter and are used to slow down the clock. The number of T flip-flops used depends on the clock signal chosen. (HINT: Make a Module to slow down the clock by 2^4 and output each connecting signal (this is the data of the counter). This module should have 2 input bits (Clock and T), and 4 output bits)

Now connect the 10MHz or 50MHz clock to the clk input of the first T flip-flop.

Do the math to figure out how many TFFs it takes to get as close to 1Hz as you can. You will need 3 additional TFFs to hold the 3 other bits of information so the most significant digit can still count to F.

The most significant digit should increment about once a second, with the least significant digits changing so fast you cannot see them. See Figure 4.6. If you disable the enable, you should be able to read each of the displays. See Figure 4.7.

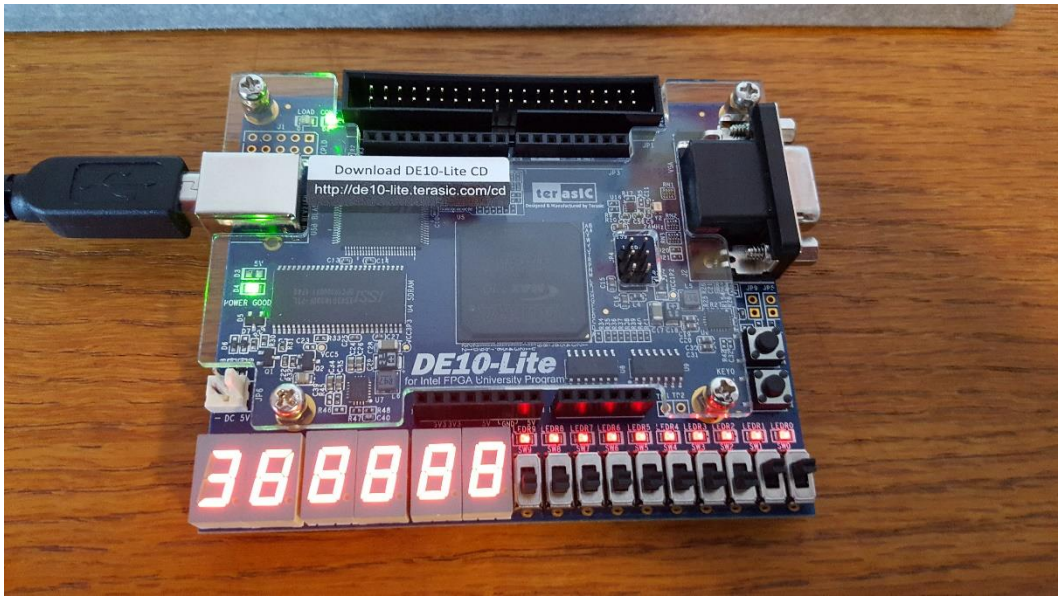

Figure 4.6 Picture of counter counting showing that the most significant digit incrementing about once a second, with the least significant digits changing so fast you cannot see them.

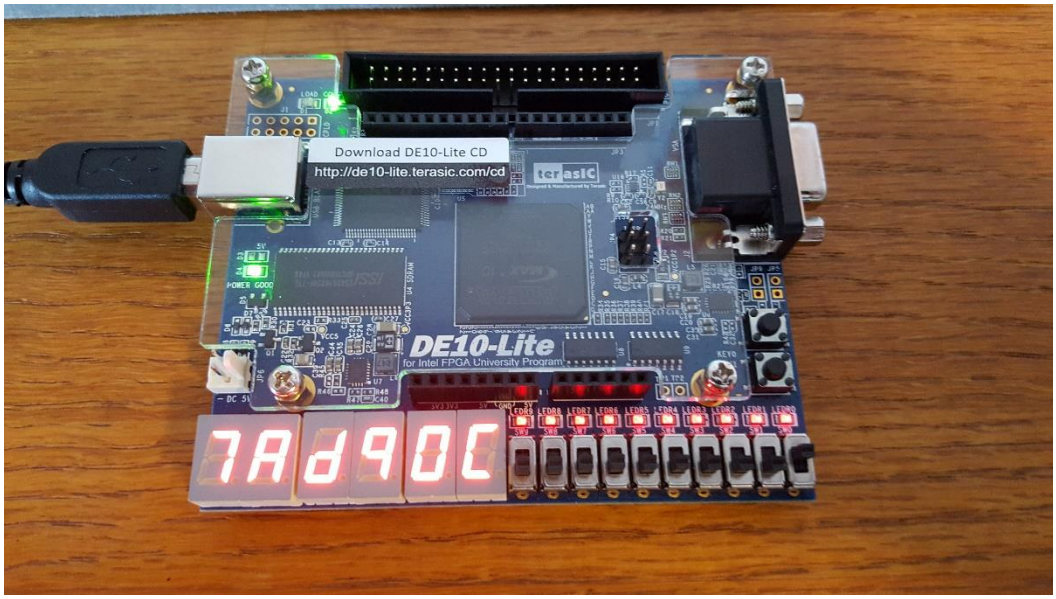

Figure 4.7 Picture of the counter with the enable turned off allowing you to see the values on the displays.

4.7 Design Entry

1. Draw up your modules in a .BDF files.
2. Create a symbol for the completed modules by clicking file -> create/update -> Create a symbol for the current file.
3. Use the Bus Tool (Right next to the node tool) to create busses, which automatically connect to busses of the same name and can contain multiple bits of information. These buses will be used to connect your modules together. Set a bus name by right-clicking it and selecting properties. Multiple bits can be assigned with the A HDL notation of Name[3..0]
4. Connect everything together in the top level.
5. Assign pins to the outputs.

4.8 Design Simulation

It is required that you use ModelSim to simulate your design using the same process as in Section 1.7.

4.9 Map Design

Synthesize your design and assign pins to your inputs and outputs using the same process as in Section 1.8.

4.10 Program Hardware

Program the MAX10 using the same process as in Section 1.9.

4.11 Test Hardware

Test to make sure digits display correctly.

It may be helpful to test one digit of the display at a time. To do this, have a button or switch input for the clock, and test the output from 4 T flip-flops.

4.12 Checkoff

1. A design in .BDF files made with TFFs.
2. Valid hardware output.

4.13 Study Questions

1. Why did the counter count down before the inverters were added to the design? Answer this with respect to the operation of the Flip-flop.
2. What would happen if the T Flip-flops were replaced with D flip-flops in this design?
3. Did you use the 10MHz or the 50MHz clock in your design, and why?

4.14 Report Rubric

The reports for the labs are to be formal reports and are to be completed using a word processing application. Please turn in the report to Moodle in a PDF or MS Word format (No Google Docs share links).

Your report should contain at a minimum the following items:

1. **Title**
2. **Date**
3. **Name**
4. **Objectives**
5. **Equipment/Parts/Materials**
6. **Procedure**
    a. **Design**
    b. **Design Entry**
        i. **Block Diagrams**
        ii. **All schematics you created in Quartus Prime (Screen Captures).**

iii. **A screen capture of what pin assignments you made in the Assignment Editor in Quartus.**

c. **Design Simulation**

i. **ModelSim simulation figures (Screen Captures).**

7. **Complete the study questions in section 4.13**
8. **Observations**
9. **Conclusion**
10. **References**
11. **Appendix (Source code of Verilog and System Verilog files used for your simulations).**

All images need to have a title and a figure number. For more detailed information about what goes into each of the different sections, see "ENGR 272 Guidelines for Technical Lab Reports" in Moodle.

For an example of what your reports should look like see "ENGR 272 Example Report" in Moodle.

References:

1. Oregon State University ECE 272 Section 4: Counters (2019) https://eecs.oregonstate.edu/tekbots/courses/ece272/section4

2. Harris, S. L. & Harris, D. H. (2016) Digital Design and Computer Architecture: ARM Edition 1st Edition, Waltham, MA: Elsevier.