Oven Toaster Timer

List of Authors (Eric Hoskins, Manar Dano, Ka Chai, Umair Tahir)

Electrical and Computer Engineering Department
School of Engineering and Computer Science
Oakland University, Rochester, MI
e-mail: erichoskins@oakland.edu , manardano@oakland.edu , kchai2@oakland.edu, utahir@oakland.edu

***Abstract*—*This report is about the change in oven toaster timer when the temperature is set up in different degrees. The idea of this project seems interesting since using Nexxys board as an oven timer. The idea of the project is to show that the time start differently depends on different temperatures are set at the beginning. Also, there is a LED tells you whether the oven is overheating. LED shows that the oven automatically turn on/off itself depends on When the oven is overheating, the LED will go off. When the LED is not overheating, LED stays on. The timer will automatically reset to where it starts when the time hit 0. On the Nexxys board, temperature is control using switches 15, 14 and 13 and timer is using switches 11, 12. The enable switches is 1.*

I. Introduction

We wanted to make a circuit that will count down the timer of the display.

The circuit could be used in an oven stopwatch that was our motivation for our final project. The implications of this project were using the Nexys board to be the brains of the oven timer. The topics that we learnt in class are finite state machines, seven segment display, clock divider, multiplexer 7 to 1 to cover our final project. We learnt multiple seven segment display, counters and integers on our own. The applications of our project are controlling the time for the oven.

The main purpose of the final project is to test out the actual timer of the oven using the Nexys board. Our idea for the project is to show LED light whether turning on/off if oven overheating. We will add a switch for the on/off button.

II. Methodology

The design for our oven display consisted of several logical blocks strung together in a top file. To begin this design, first, we started with the actual display. In class, we had never attempted to drive multiple digits on the display. This was our first challenge to overcome. To accomplish this, we first thought of adding an 8-1 multiplexer. The output of the multiplexer had the values of each of the eight digits. This then was fed into the seven-segment decoder. To control which digit was on we connected the select line of the multiplexer to the seven-segment decoder. This allowed the seven-segment decoder to output the information to display the digit and the correct position. These select lines needed to count from 0-7 at a constant rate. To do this we built a clock divider that divided the input signal 3 times. It took the form of a F.S.M. with 8 states. Each clock tick triggered a transition to the next state.

We made a file that assembled these parts and input the digits 1-8 to positions 1-8 on the multiplexer. The result was illegible garbage. The clock signal was still too fast. This caused the LEDs to not have time to light up and turn off in time. To remedy this situation, we created a counter. This counter simply had an integer X that was logic high every thousand clock ticks. This solution worked beautifully. We were now able to display multiple digits on the NEXYS board at the same time. The next step was to create a look up table for the temperature display. This lookup table had 1 input and 4 outputs. The input was a 3-bit number based on the state of three switches. The output was four 4-bit numbers. We realized that since the temperature never exceeded 1000 degrees, that the first output would always be 0.

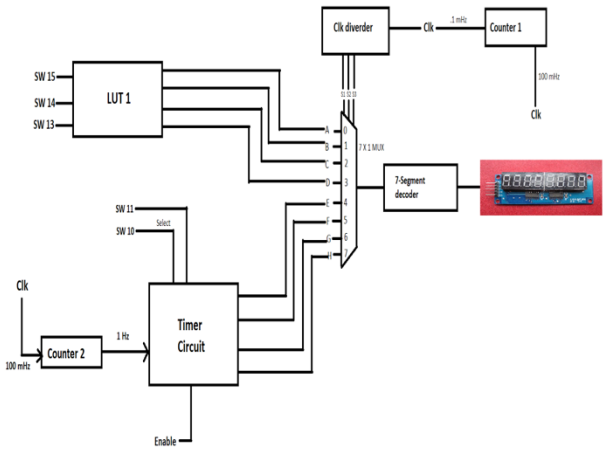
We kept it to make the system more robust. This choice made it so that we could modify the look up table, at a later date, to display temperatures up to 9999. This larger range makes the project able to be used in many different applications. We now had a circuit that would display 8 different temperatures.

The next step was to design a timer circuit. We decided to go with times from 30-90 seconds. We settled on a two-bit input. This left us with the times 30, 45, 60, and 90. Because 90 was the largest value, we designed a F.S.M with 90 states. In retrospect, this may have not been the most efficient way to accomplish this. However, it worked very well. The two-bit number from the switches was input into the timer and used in a with-select block. We initialized a variable that was a different state based on the input from the switches. If the input was 00 for example, the variable would be S30. This allowed us to utilize one F.S.M to encompass four separate counts. The transition process counted down from the value of the variable to 1. At that point it would return to the value of the variable. The reset command for this F.S.M. was also based on the variable. It would return you to the correct state so if the timer was set for 60 seconds and the user pressed reset, the timer would return to 60 seconds.

The last step in this was to hook up a counter to slow down the clock signal. The internal clock signal on the board runs at 100Mhz. This is 100 million ticks per second. To cancel this out, we made another counter that exported a 1 every time a variable was incremented up to 100 million. This created a signal that oscillated at 1 hz. This signal was used to drive the transition process of the timer. The timer then exported the two numbers to the multiplexer. We now had the entire display created. It could display a temperature and drive a timer. We felt that this wasn't quite enough, so we started to think about integrating a heating element and a temperature sensor. The temperature sensor we had exported very complicated data, so we decided to simulate a temperature sensor with a simple binary input. A 9-volt battery connected to a voltage divider and a switch exported a logical high at 3.3 Volts at the users input. A logical high represented that the temperature was not hot enough and a logical low meant that the temperature was too hot. This signal was connected to the board on one of the Pmod connectors. This signal was initialized as a variable in the top file. This signal was ended with the enable of the timer and drove an LED. This LED represented the ovens heating element. The LED would only turn on when the timer was on and the oven wasn't hot enough. If the oven was too hot or the timer was turned off, then the oven would shut down. This LED was also taken off board. The Pmod has a 3.3Volt out and a GND. What we did was took the 3.3 Volts to a resistor, to a led, to a transistor, and then finally back to the ground. A wire was connected from the Pmod to the base of the transistor. The ended signal was then port mapped to this new wire. This allowed us to drive an LED based off signal from the board and from the environment. Overall, we used the simple blocks we learned how to create in class to design a fully functioning system.

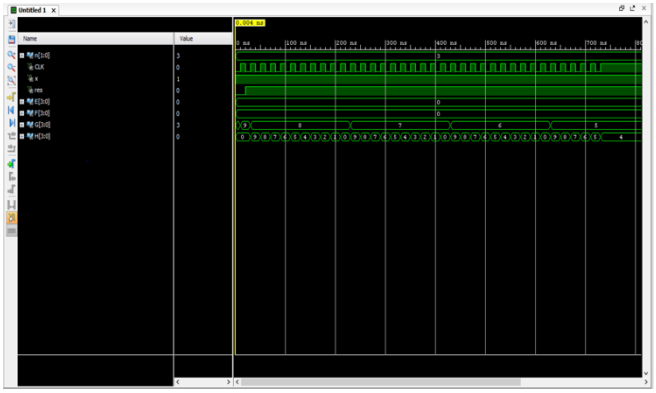
III. Experimental Setup

We used the testbench to verify the functioning our project. We used Vivado 2016.2(VHDL) for the software and Nexys board for the hardware. The specific configuration tools that we used is how we codes in each file separately and we put everything in top-file. The expected results were when we input the values, the timer should either start from 30 to 90 depending on the values of n coming in. The picture below is showing how we setup up before we run the experiment.



IV. RESULTS

The picture below show the output for the timing diagram after we putting all the code together. The result came out as what we expected. We also tested the Nexsys board to make sure we received the correct timer when we switching different temperature.



CONCLUSIONS

We were to test out the oven timer using the look up table 1, clock divider, multiplexer, seven segment, decoder, timer, and topfile. We used some of the codes from previous lab for the project. The main codes were mainly from the topfile and the timer. The topfile and the timer are the decision maker to facilitate and transformation. The main issues our group remain to solve is temperature sensor. We could not find out what the issues were from our codes and we decided to skip that part. If we have more time, we could have done better for our project.

REFERENCES

[1] "Signal Assignments in VHDL: with/Select, When/Else and Case." Edited by Philippe Faes,*Sigasi*, 4 July 2011, insights.sigasi.com/tech/signal-assignments-vhdl-withselect-whenelse-and-case.html.