CS 152A / EE M116L

Introductory Digital Lab Design

LAB #4

Stopwatch

May 20, 2014

Grade:

Tianheng Tu

Name1: Nathan Tung

SID1: 004-059-195

Name2: Mark Iskandar

SID2: 704-050-889

INTRODUCTION

The purpose of Lab 4 is to design and build a stopwatch using Xilinx schematics and the FPGA board. Given a clock input, our stopwatch must be able to count minutes and seconds from 00:00 to 59:59 before resetting. One button (PAUSE) allows us to pause the clock and another button (RESET) resets it to 00:00. Finally, a switch (ADJ) allows us to adjust either the minutes or seconds by speeding up the clock cycles of whichever one is selected by a second switch (SEL).

The most important parts of the stopwatch include a clock module that breaks down a single clock frequency into multiple different frequencies in order to increment the counters, make the display flash, and so on. In addition, a series of modulo 10 counters comprise the four digits of the stopwatch and feed logic into the FPGA board's seven-segment display.

DESIGN COMPONENTS

Overall

Our stopwatch design begins with the clock module, which accepts a master clock signal and divides it up into relevant clock signals of varying frequencies. These clock signals will decide when to increment the counters and how often the actual 7-segment display refreshes. The clock module sends a 1 Hz signal into the first of a series of four BCD counters, which are responsible for storing and counting the current stopwatch time from 00:00 to 59:59 and so on. Instead of increasing to 60:00, all counters reset to 00:00 and the entire process begins again. These BCD counter outputs are connected to the 7-segment display and outputted on the FPGA board. The RESET button needs to clear all counters, while the PAUSE button disables counters. ADJ similarly disables the minute or second counter that is not chosen by SEL and then speeds up the clock cycles going into the other counters. Finally, when ADJ is high, another counter incrementing at 4 Hz intercepts the 7-segment display outputs, causing the entire display to blink. Notice that all buttons and switches used are debounced to mitigate preventable errors.

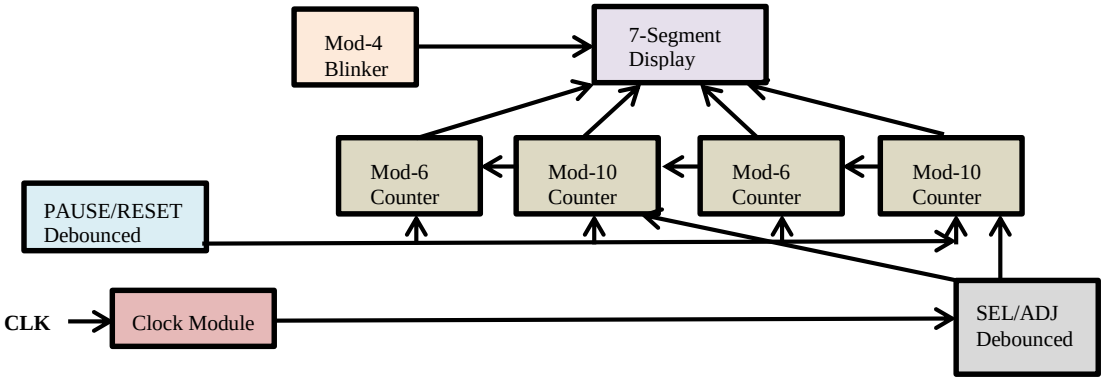

Figure 1: High-level schematic of the stopwatch

Clock Module

The clock module acts as a frequency divider and is integral in accurate timing of the stopwatch. Given the 100 MHz master clock signal from the FPGA, we need to produce clock signals with frequency 1 Hz (exactly once per second, for incrementing the stopwatch), 2 Hz (for adjusting the clock at a quick pace), 4 Hz (to make the 7-segment displays blink while in adjust mode), and

500 Hz (to multiplex changes to the 7-segment display, since all digits normally change at the same time). For example, to convert 100 MHz to 1 Hz, our clock module will need to divide by 100000000. This means our 1 Hz clock module produces a high clock cycle only once every 100000000 times the 100 MHz clock does.

To achieve this, we build a modulo-2 counter and a modulo-5 counter. The former is essentially a T (toggle) flip-flop, while the modulo-5 counter uses a modulo-5 BCD counter (that is, a BCD counter that outputs 1 on 0100 and clears on 0101) with two T flip-flops that work to increase the duty cycle to around 50%. With these two counters, we can also build a modulo-10 counter. Stringing these modulo counters up in series allows us to create a module that divides 100 MHz into 1 Hz, 2 Hz, 4 Hz, and 500 Hz.

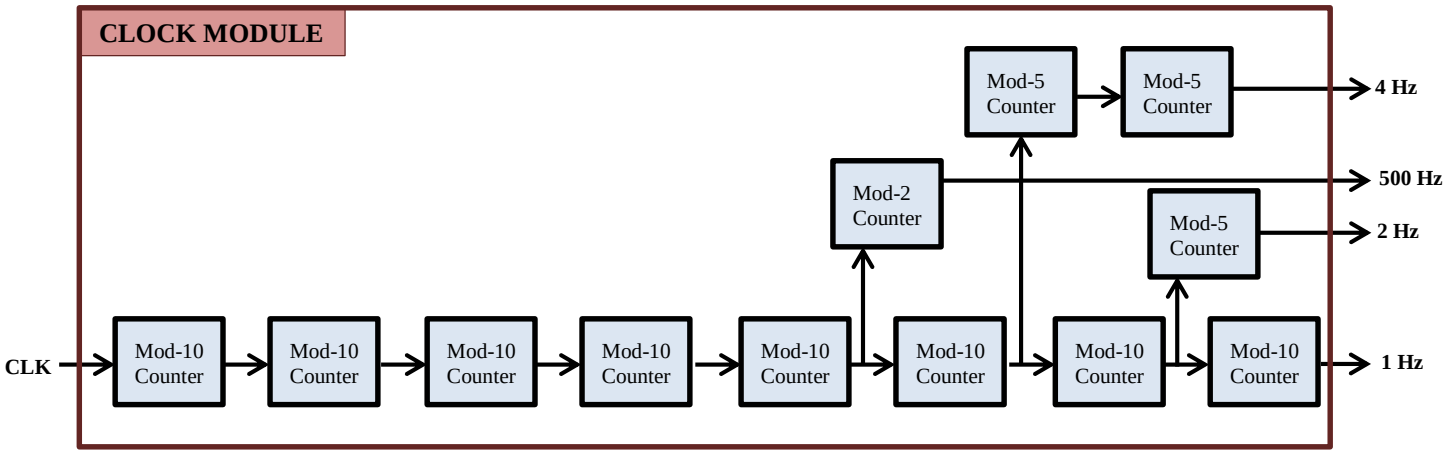

Figure 2: Clock Module for dividing frequency of an input clock signal

4-Bit Minutes/Seconds Counter

Since the BCD counter is modulo 10, it naturally counts from 0-9. We use two of these for each of the minute/second one's-digit. Similarly, we use a modified modulo-6 BCD counter (that is, a BCD counter that outputs high and also clears upon reaching 0110) for each of the minute/second ten's digit. In effect, when the least significant counter is about to reach 10, it instantly resets to 0 and instead sends a clock signal to the next least significant counter. Therefore, the 1 Hz clock cycle only needs to feed into the least significant counter, and the counters themselves act as (modulo-10 and modulo-6) frequency dividers by propagating the divided clock signal up to the most significant counter.

7-Segment Display

The 7-segment display module was taken from our work in Lab 1. By converting our VHDL code into a Xilinx schematic, we constructed a symbol and replicated it four times in our stopwatch – once for each of the clock digits. Each 4-bit output (Q3, Q2, Q1, and Q0) of each 4-bit counter is sent to this module, where the logic we created in Lab 1 successfully displays each of the numbers. Our 7-segment display essentially involves NOT, AND, and OR gates in order to represent the sum of products equations that capture logic behind each segment.

Selecting the actual digits to display is more difficult. The 7-segment modules send their output into seven 4-bit MUX gates, where the selector chooses which digit to display. But since the

FPGA board by default displays all digits at the same time, we create a counter (together with a decoder) that accepts the 500 Hz clock signal and loops through the selectors (AN3, AN2, AN1, AN0) for all four digits at that frequency. The result is a 7-segment display that refreshes so quickly that the human eye cannot tell.

Adjustment/Select

The ADJ and SEL switches work hand-in-hand. SEL tells us whether we want to adjust minutes or seconds the next time ADJ is flipped on. The logic for adjusting and selecting is not difficult, but involves many MUX gates. Notice that SEL is completely ignored when ADJ is flipped off. When ADJ is high, SEL is taken as a selector into MUX gates to determine whether to disable the one's digit clock of minutes or seconds. Similarly, a MUX gate passes in 2 Hz into the clock input of whichever counter is not disabled. This enables the stopwatch to increment twice as fast. Furthermore, when ADJ is high, a new BCD counter is enabled with 4 Hz clock signal and connected to the 7-segment display digit selectors; only when this counter increments does the FPGA show the 7-segments lighting up, effectively creating a blinker.

Pause/Reset

Both PAUSE and RESET are rather simple in design. For pausing, we wire a pushbutton into a T flip-flop, which holds and toggles our last state indefinitely, and disable the clock (set CE to 0) when we are not in adjust mode (when ADJ is 0). When ADJ is high, we disable pausing functionality. For resetting, we wire a pushbutton into an OR gate with whatever logic already exists for each counter to self-clear. This goes straight into all four of the BCD counters' CLR inputs. Therefore, each counter will reset when its own count value is too large or when the user hits RESET.

Debouncers

Both buttons and switches need to be debounced; otherwise, the unstable contact in our FPGA board can cause unpredictable behavior in our design. The buttons can be debounced by using the pre-designed DebounceSync symbol given to us. However, we created our own gated SR latch to debounce the switches. After removing the bounce in our buttons and switches, our outputs were much less likely to glitch.

CONCLUSION

In this lab report, we addressed the specifications for our stopwatch, as well as the design and low-level implementation of the stopwatch and its modules. Some of the challenges in Lab 4 involved deciding which components or symbols to use and figuring out how to use them correctly. Our Xilinx software was also rather buggy: our schematic had the correct logic, but our FPGA board would not work correctly sometimes unless we added a redundant output tag (one that was neither named nor used). Nevertheless, having the opportunity to design and produce a functional stopwatch was certainly an interesting experience.

Lab Contribution:

Nathan Tung: 50%

Mark Iskandar: 50%