Class Report 1
ELC 4396 System on a Chip

Jordan Edwards
Baylor University

September 5, 2023

Introduction

For this assignment, I completed experiment 4.8.3 from the textbook. Using four of the seven-segment displays on the Digilent Nexys 4 DDR board, the assignment was to implement a short animation as given in the textbook. The image from the textbook is shown below in Figure 1. The animation should also be able to be paused or reversed.

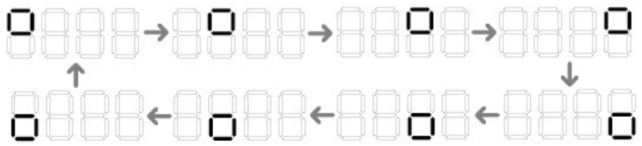

Figure 1: Desired board output from textbook

Implementation

I began by building a counter in order to slow down the incoming clock for use in the rest of the design. Because the board clock has a frequency of 100 MHZ, operating the entire circuit on this clock directly would cause the animation to play fast enough to simply appear as if all segments were illuminated constantly, with the center segments being slightly brighter than the others since there are the only segments used in both the upper and lower squares. I chose to implement the counter with the ability to count up and down. This would allow for simple reversal by reversing the direction of the counter. The clock enable pin on the counter also allows for simple pausing of the animation.

The other module which I designed was the main block, which I called "spinner." This module used the top three bits of the counter to decide which digit should be enabled. The top bit is also used to determine whether the top or bottom square should be illuminated. The code for this logic is shown below.

Because the size of the counter is parameterized with N, the speed of the animation can be doubled or halved by increasing or decreasing N respectively.

```
//pick which display is on
case(clk[N-1:N-3])
        0: digit = 1;
        1: digit = 2;
        2: digit = 4;
        3: digit = 8;
        4: digit = 8;
        5: digit = 4;
        6: digit = 2;
        7: digit = 1;
endcase

segments = clk[N-1]? top:bottom;
```

Finally, the values are written to the appropriate output pins on the FPGA to actually drive the displays. Because the signals are low-active, the one-hot digits array and the segments to be illuminated are all inverted before being assigned.

```
//write to 7seg
CA = ~segments[7];
CB = ~segments[6];
CC = ~segments[5];
CD = ~segments[4];
CE = ~segments[3];
CF = ~segments[2];
CG = ~segments[1];
DP = ~segments[0];
AN = ~digit;
```

In Figure 2 below, the overall design diagram shows this structure. The only inputs are the switches and the external clock. The only outputs are the segments and anodes for the seven-segment displays.

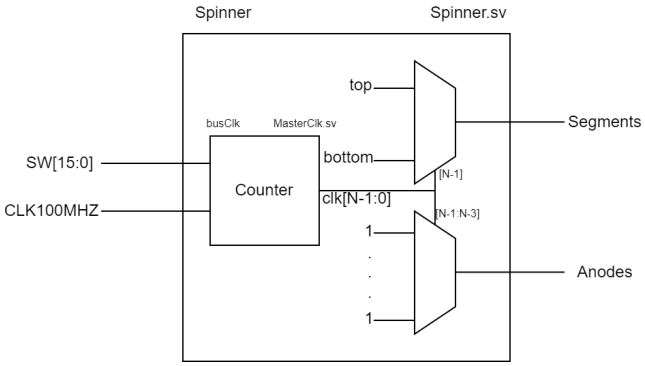

Figure 2: Design diagram

Results

The implementation works correctly. The switches on the board are able to reset, pause, and reverse the animation. The circle "moves" correctly around the board. In addition to looking at the physical results, I also set up a test bench to view the simulated results. The output of that simulation is shown below in Figure 3.

In the simulation, the reset pin is first asserted, putting the system in a known state. Around 20ns in, the enable pin is toggled, starting the animation. From this point until about 200ns, the bottom bus, which controls the anodes, clearly enables them in the correct order, one at a time. Around 225ns, the pause feature is shown by turning off the enable pin. From 250ns to the end of the simulation, the reverse functionality is shown. The CA, CB, CC, etc. pins in the center of the simulation also properly transition when the animation reaches the edges of the display.

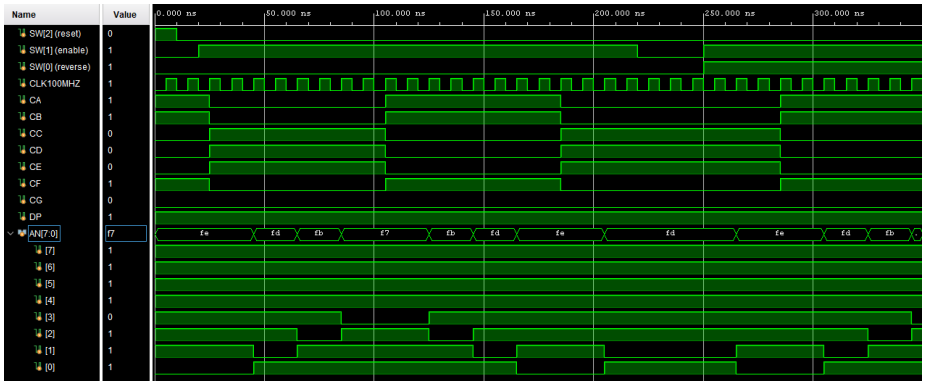

Figure 3: Simulation results

Analysis

The counter module will most likely be very reusable for further projects due to its generic and versatile nature. Unlike what was discussed in class, I chose not to implement a multiplexed output for the seven segment displays because only one needed to be on at any given time. This implementation allows for the lit segment to be brighter since it can be run at a 100% duty cycle. It worked out well that the two animation states changed at exactly halfway through the cycle. Due to this, I was able to use an already available signal, clk[N-1], to drive the decision as to which character to display.