California State Polytechnic University, Pomona

Lab 4: 7-Segment Display Controlled by Counter

Group F:

Danny Manzo, Bronco ID: 014451190

Maher Mateen, Bronco ID: 013891189

ECE 3101L – Signals and Systems Laboratory

Professor Mohamed Aly

5 July 2022

1. **How a 7-Segment Display Works:**

A 7-Segment display contains seven different led inputs that can be controlled individually. To get a specific number as an output, you input a 7 bit binary number to the corresponding pins of the 7-segment display. Each of the segments of the 7-segment display have a corresponding letter assigned to them, such as segment a, segment b, segment c, all the way to segment g. When inputting the 7-bit binary number, it is important to note the order of the bits, whether is goes abcdefg, where segment a is the MSB and segment g is the LSB, or it goes gfedcba, where segment g is the MSB and segment a is the LSB. In this lab, we are using a Common-Anode (CA) 7-Segment Display, which normally means that to turn on each segment, we must drive the anode to be logic high, or 1, and the cathode to be logic low, or 0. However, since this board uses a transistor to drive into the common anode, the anode enable signals are inverted and thus we drive the anode to be logic low, or 0, which we show in the code.

1. **How We Coded It:**

The first thing we did was to create 2 parameters, incrRate and refreshRate, to control how often we increment the digit by 1 and how fast we change from one 7-segment display to the next. Next, we created our counter, which clears when the clear button is pressed, and increments when the enable switch is ON every 1/2 second through a comparison with the incrRate, which is set to 1/2 the clock frequency, or 1/2 of 100 MHz. This comparison was made by having a count register that increments by 1 every positive edge of the clock, and checking whether that count was equal to incrRate. We then cleared the count after the count was equal to the incrRate. We made a variety of checks to see if the ‘ones’ digit, the ‘tens’ digit, or if both are equal to 9 or not equal to 9, and what to do in each case. For example, when both are equal to 9, we reset back to 0, and when the ones digit is not equal to 9, we increment it by 1. We then made a 8x1 Multiplexer to select which digit is displayed, either the one’s digit or the ten’s digit. Afterwards, we made the decoder/truth table for the 7-segment display through a case statement with the binary equivalent of segments a through g being outputted for digits 0-9 and letters A-F. Afterwards, we made an always block to select which 7-segment displays we want on through the use of the select register. We coded it such that only the leftmost and rightmost 7-segment displays turn on when the select is equal to 7 or 0 respectively. Lastly, we updated the select register by incrementing it by 1 whenever the register refreshCount is equal to the refreshRate parameter, which is equivalent to every 1/16 ms.

1. **Vivado Data Collection:**

In Vivado, after running the synthesis and report utilization, we found that there was 43 look-up tables (LUTs), 55 slice registers, 27 slices, 43 LUTs as logic, 25 LUT Flip Flop Pairs, 19 bonded input/output bits (IOB), and 1 BUFGCTRL. The total power of the full adder with the demultiplexer was reported to be 0.085 W.

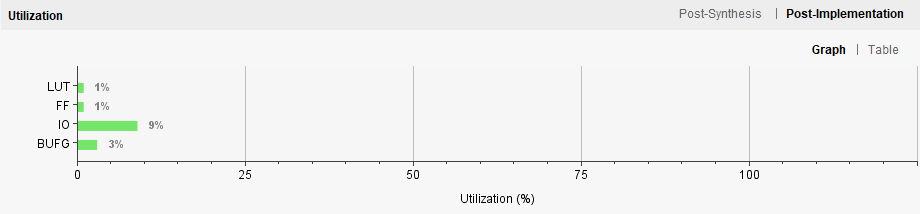


Figure 1: Resource Utilization Graph

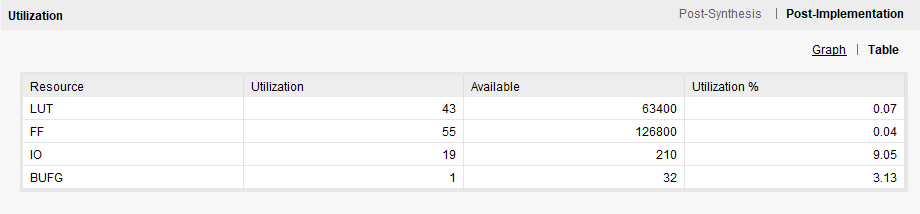


Figure 2: Resource Utilization Table 1

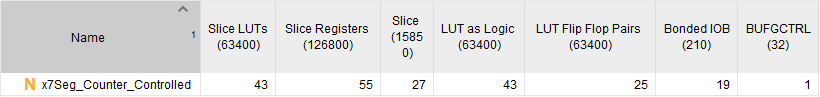


Figure 3: Resource Utilization Table 2

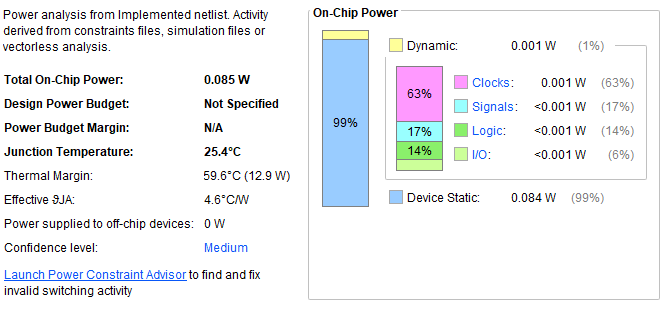


Figure 4: Power Usage