Name: Mohammad Elkholy

ID: 90020159

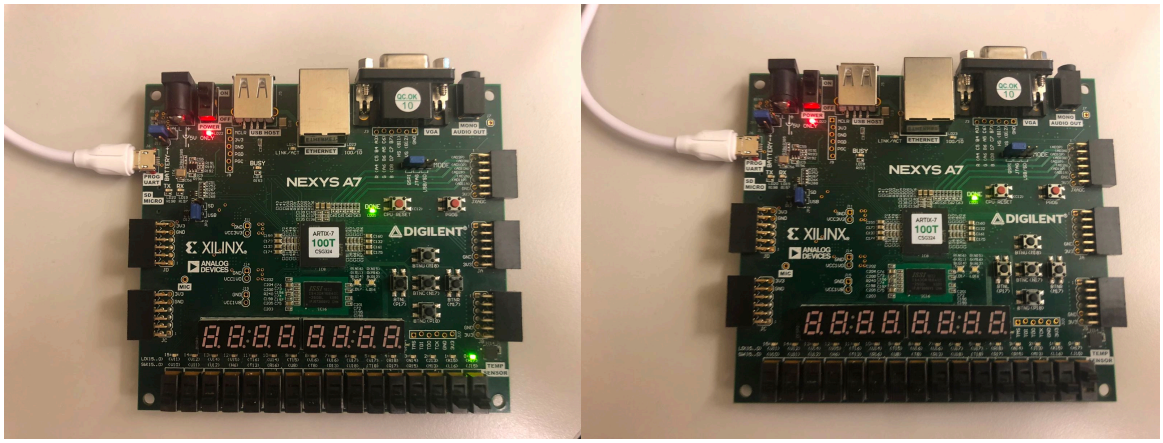
Team name: TwoBits

Team members: Mohammad Elkholy & Ahmed Bahssain

Lab # 1 Report

Summary:

1. Expriment # 1: We implemented a simple inverter that takes the logic from a switch on the Nexys 7 A7 board and inverts it. We know the output by mapping the output of the Verilog



   module to an led. The led lights up when the switch is off and dims when the switch is on.

2. Experiment # 2: We implemented a driver for one of the 4-digit 7-segment displays on the board. Since all the digits share the same cathode pins, we must activate each digit alone, then activate the one next to it. We do this in a rotational pattern. We reduce flickering by reducing the 100 MHz internal clock of the board to ~95 Hz. This is done by using dividing the internal clock by 2^{20} using a 20 bit counter. We can then activate each digit for a period of 2.6 ms by using the 2 MSB of the 20 bit counter. We change the active digit as soon as the 2 MSB change. The input for the module consists of the following:

   a. CLK100MHZ: internal clock of the board

   b. SW: 13 bit number that we can change by toggling each bit using 13 of the switches on the board

   c. AN: 4 bit number that chooses which anode to activate

   d. SEG: 7 bit number that chooses which segments on the digit to activate

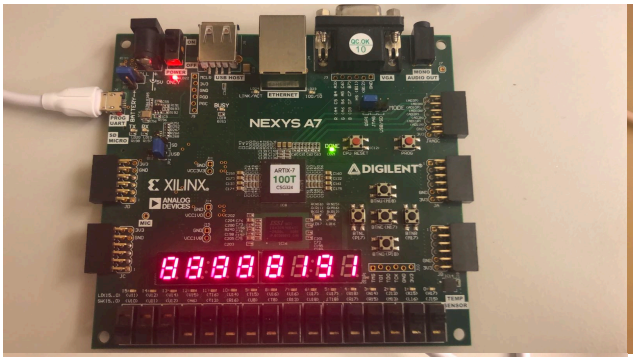
   The variable used are as follows:

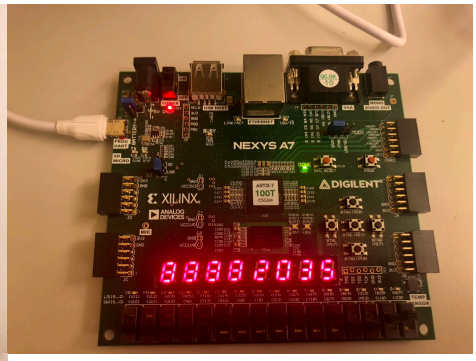
   a. LED_BCD: 4 bit number that represents the digit between 0 and 9 to be displayed.

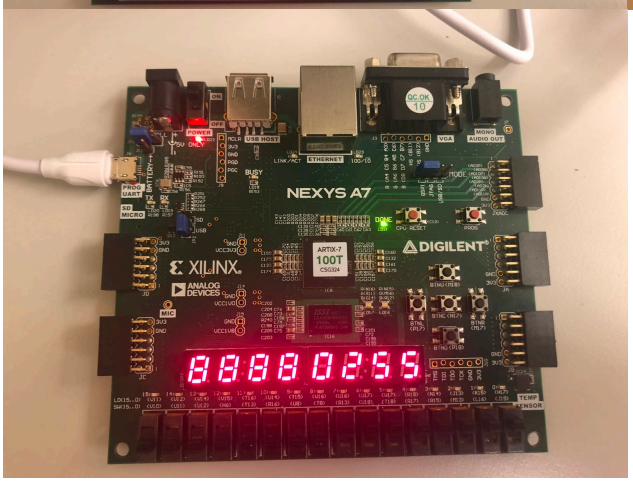
b. Refresh_counter: 20 bit number that is used to divide the clock and change anodes activated.

c. LED_activating_counter: 2 MSBs of refresh_counter.

The digits from the 13 bit number were extracted using the division and modulus operators in this experiment.

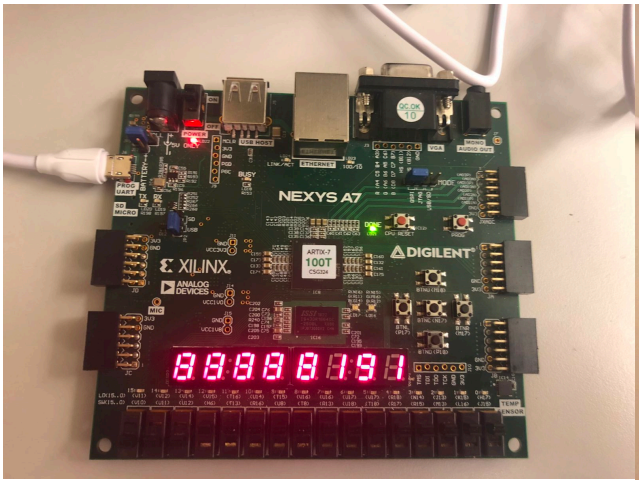


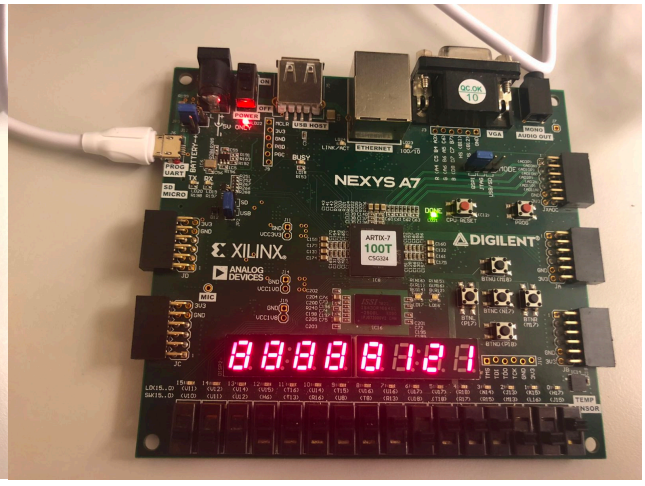


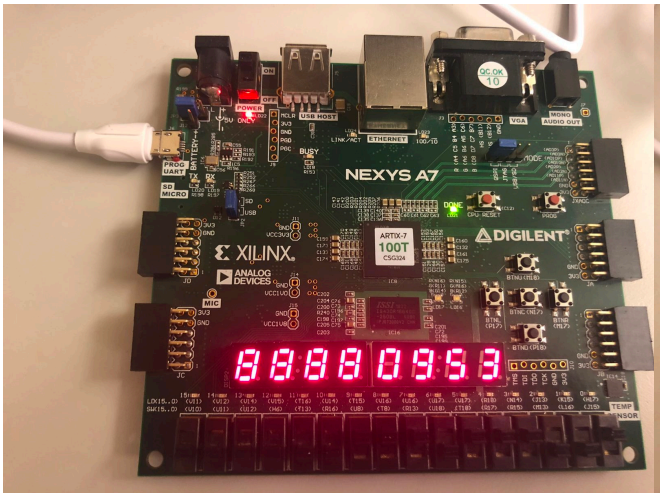


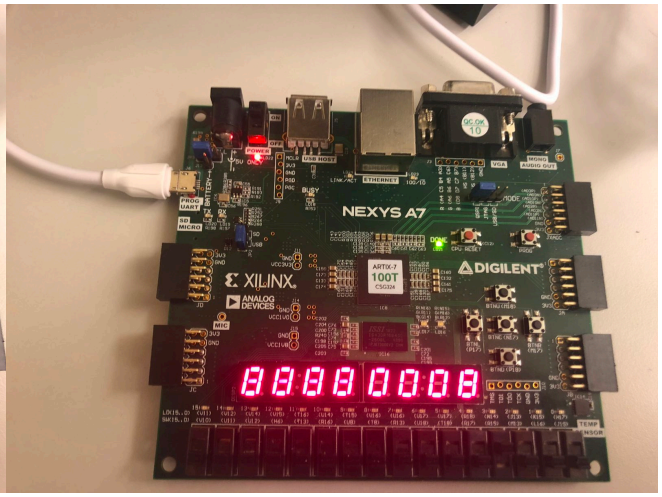
Results: Code worked as expected, we used the switches to toggle the number that should be displayed on the display and it changed accordingly.

3. Experiment # 3: In this experiment, we implemented the same thing as experiment 2 but we used a different method to extract the digits of the number. We used the shift and add 3









algorithm. We copied the code for the shift add 3 algorithm and changed the loop variable to 12 instead of 7 in order to be able to read the thousands digit. We then used the output from the shift add 3 module as inputs for LED_BCD. Code worked as expected.

Differences between Experiment #2 and #3:

- Utilization:
    - LUTs: Experiment 2 used a bit more than twice the number of LUTs that experiment 3 used (119 vs 52). This is due to the use of division and modulus operators in the verilog code for extracting the digits from the 13 bit number. Experiment 3 uses much less LUTs as it makes use of the shift and add algorithm, where we only use the left shift and the add operators which are much less costly hardware wise.
    - FF: The number of flip flops used was the same in both experiments (20).
    - I/O: The number of I/O ports used for both experiments was the same (25) as we did not change anything in the input and output of the top module.
    - Delay: The timing delay for experiment 2 was much larger than that of experiment 3. This is expected as shift and add operations take much less time than division and modulus operations, as well as the fact that experiment 3 uses much less blocks than experiment 2.

Note: Timing report, utilization report and pictures of schema are attached with this report.