Kamran Mapar PID: A17132327

Sitthiphol Yuwanaboon

ECE 25

Section

Wednesday 7p - 10p, WLH2211

**Lab 2: Bit Adder Design**

**Introduction:**

The purpose of this lab is to understand and design a one-bit adder using Verliog’s code and the BASYS board to program it. Using our previous lab’s code and the pre-lab’s code and logic, we can create a working one-bit adder on the BASYS, using its switches to verify our truth table.

**Procedure:**

* To create the full adder on Vivado, use a top module in Verilog to separate logic and implementations
* Create a new source and instantiate the module defined in the pre-lab
* Call the function from your top module with the registers and wires ordered in the same way
* Write the Verilog code for the full adder design
* Write a simulation to test the full adder design to confirm its functionality
* Download the design onto the BASYS3 board using 3 input switches for A, B, and the carry-in and two LED outputs to represent the carry and sum
* Demonstrate the functionality of the design and take screenshots

**Circuit designs:**

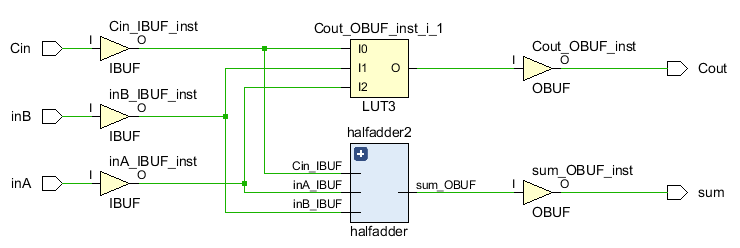
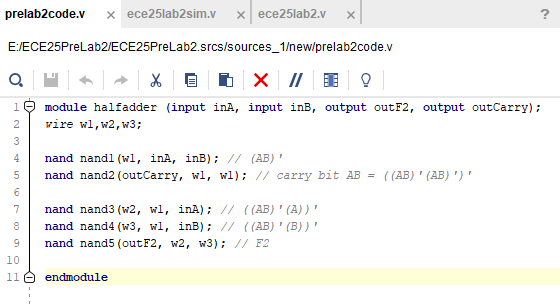
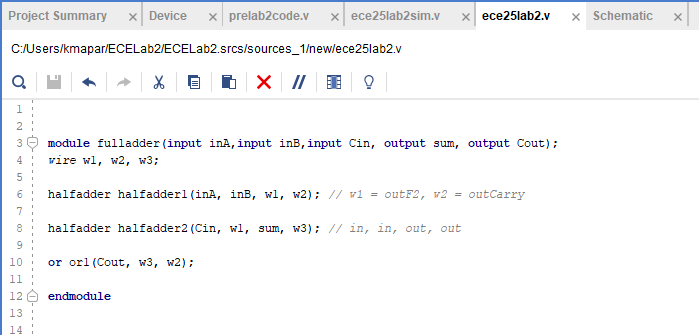
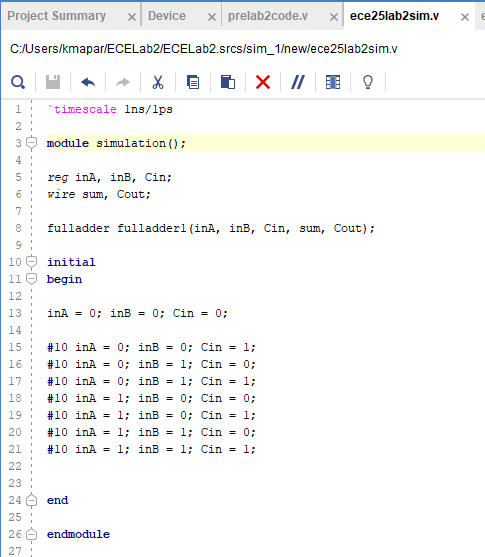


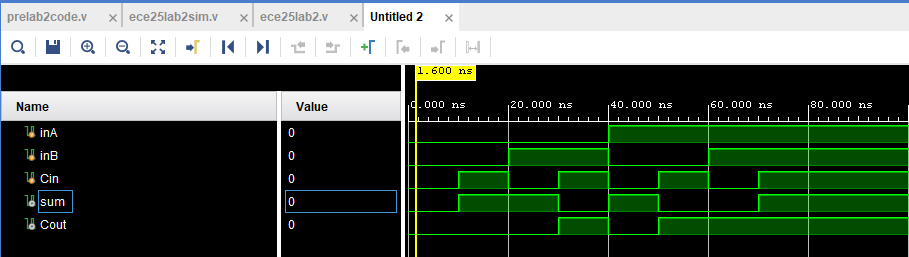
Figure 1: Schematic of full adder showing carry-in, A, B, carry-out, and sum

**Code:**

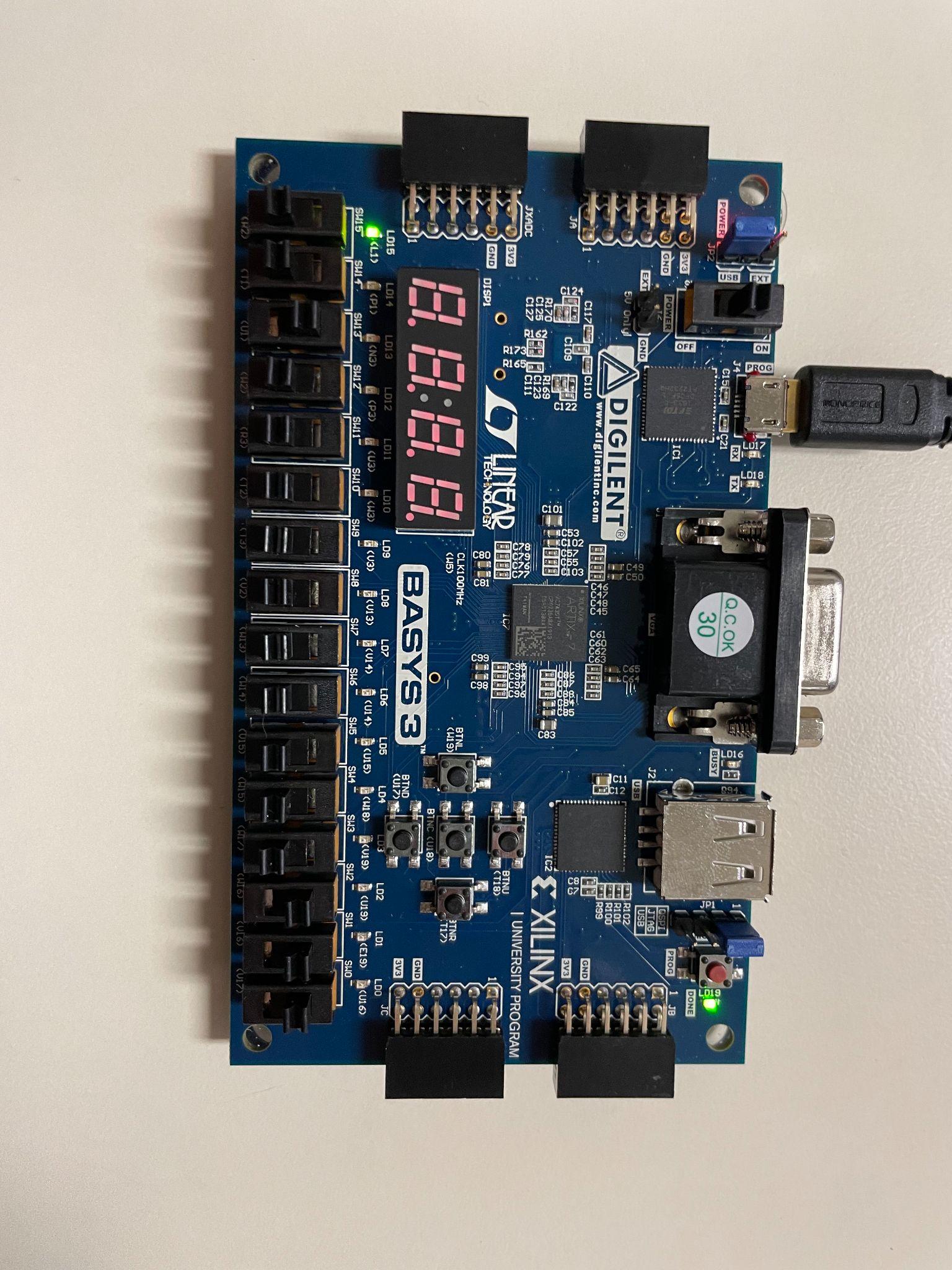




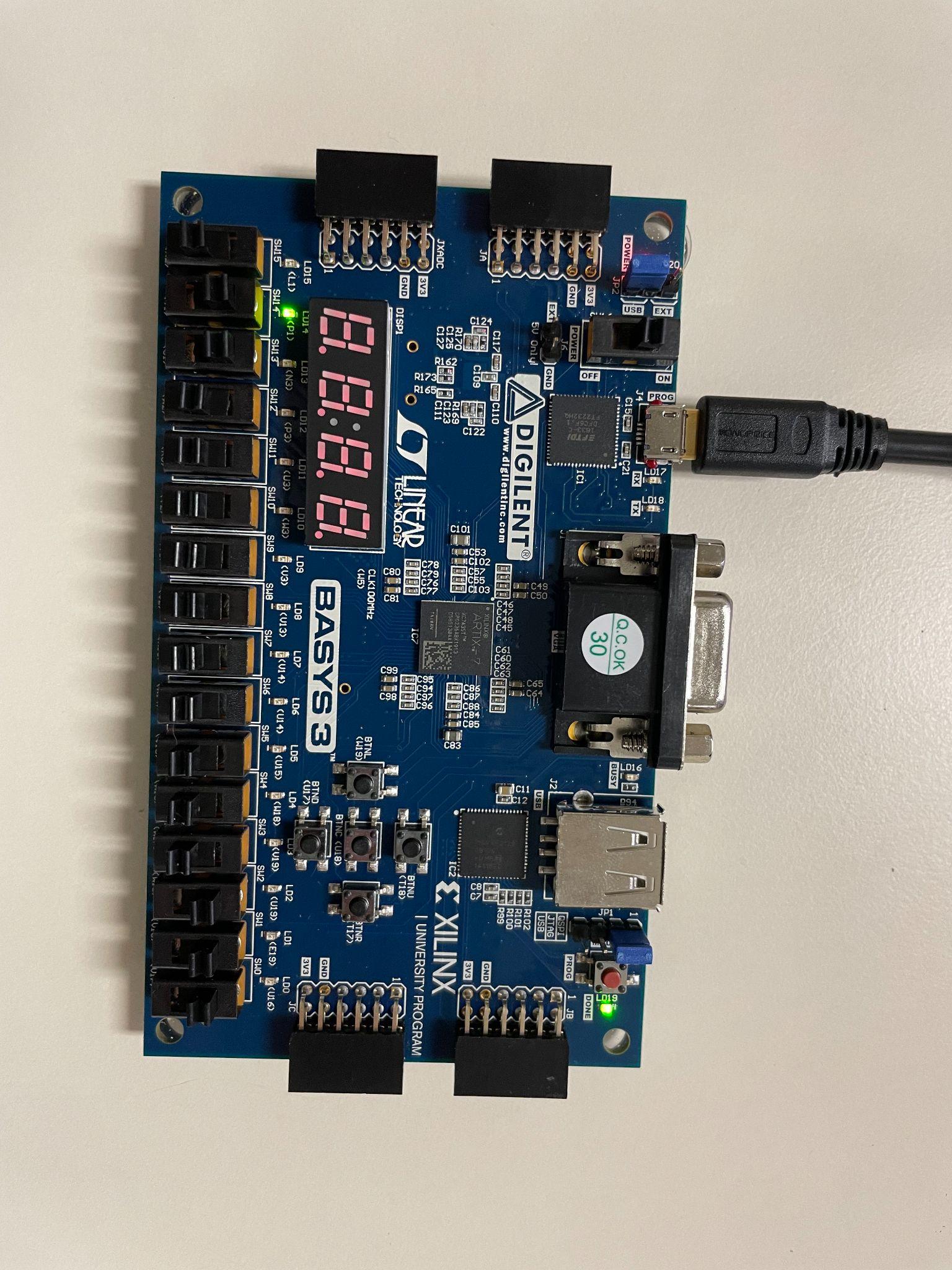
**Data:**



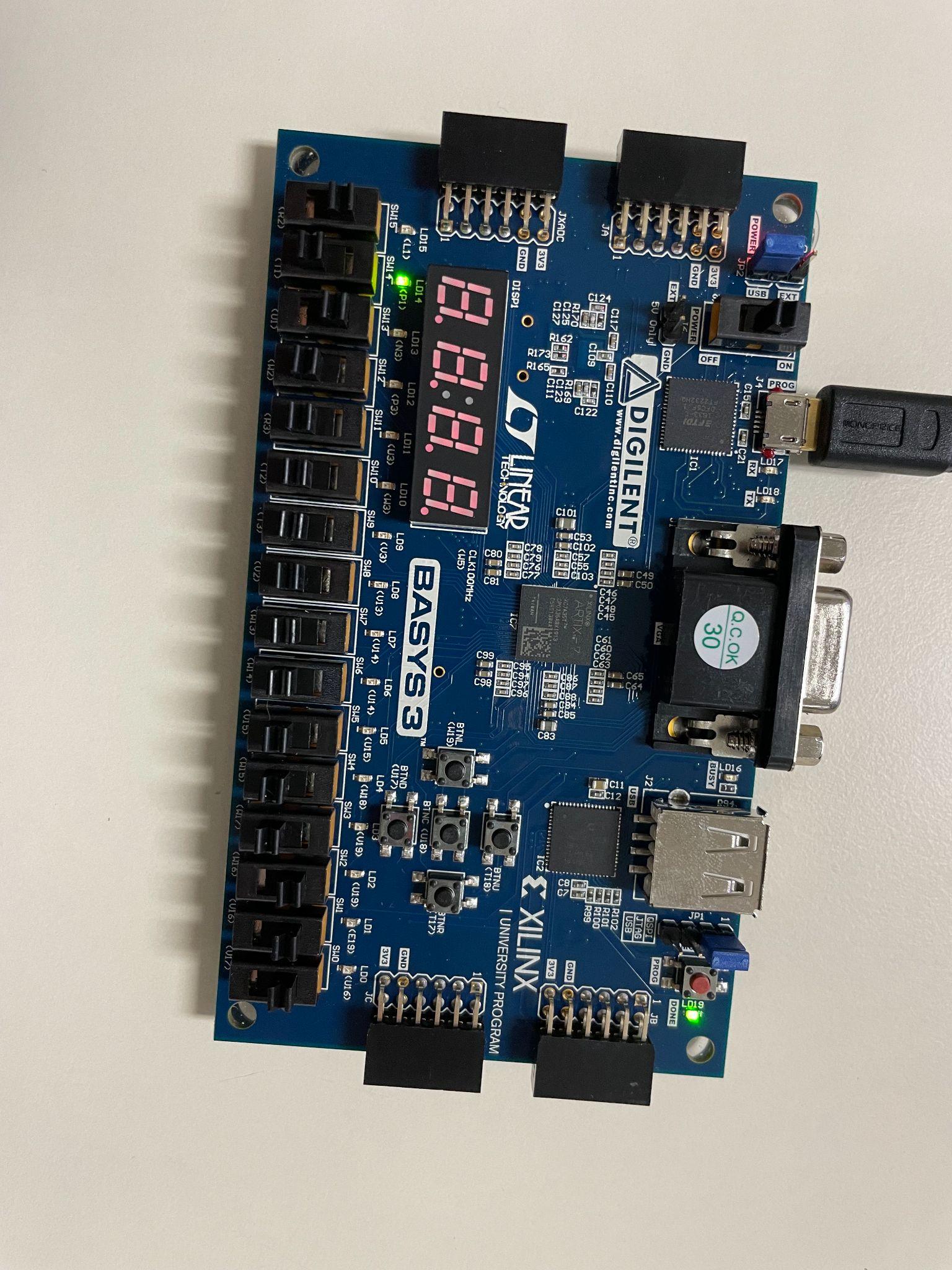
**Figure 2: Full adder output**



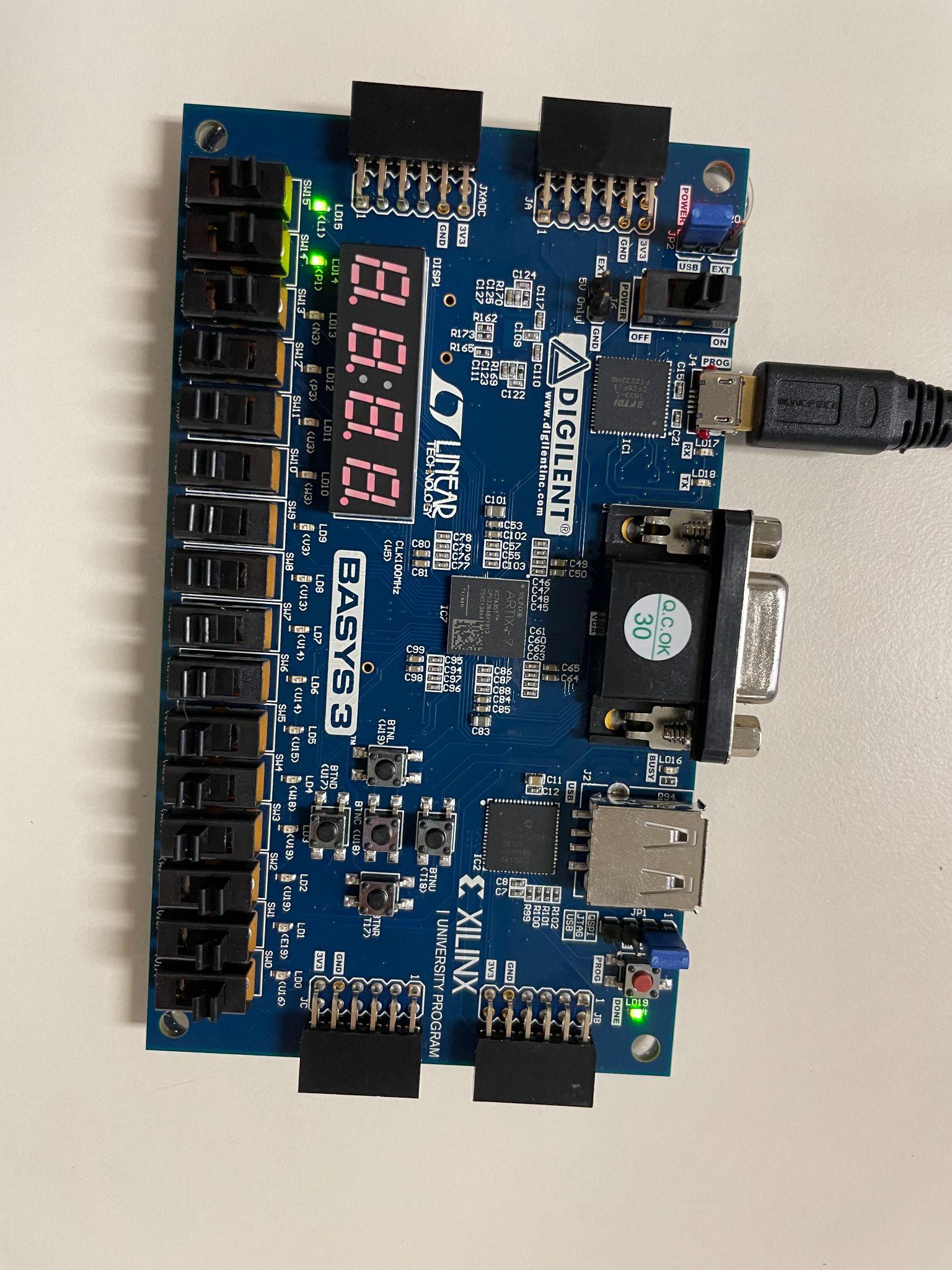
**Figure 3: Carry-in input on, sum LED on**



**Figure 4: B and carry-in input on, carry-out LED on**



**Figure 5: A and carry-in input on, carry-out LED on**



**Figure 6: A, B, and carry-in input on. Carry-out and sum LED on**

**Analysis:**

To create a full-bit-adder, using the pre-lab code which consisted of a half adder design, we can code a full adder design by using a top module in Verilog. To instantiate the module of the half adder we wrote in the pre-lab, we create a new source in Vivado and typed code to call or instantiate the module we wrote for the pre-lab. This way, we can simply call it over and over again without any need to rewrite code and end up with a lengthy program.

We can then call our module from the top module using Port Order Convention, using the same order of inputs and outputs in the original definition. So, in our code, wire 1 corresponds to F1 from the pre-lab, and wire 2 corresponds to the out-carry. This way, our function/module call can produce an output that is passed to our output variables of the top module. This can be seen in our code above.

To download and verify our Full Adder design, we replicated the steps from lab 1 to download the design onto the BASYS3 board. In this lab, we assigned SW15 as A input, SW14 as B input, and SW13 as the carry-in. The outputs were two LED’s; LED15 as the result, and LED14 as the carry-out. We used the same convention of assigning inputs to switches on the board from lab 1, as the computer recognizes them differently than we do.

With all inputs off, no LEDs were on. Turning on 1 input resulted in a sum, so LED15 would light up. If we flipped on two inputs so as to add two bits, we would see LED15, the carry-out LED light up as we expected. If we turned on a carry-in and an input A or B, we also saw the carry-out LED turn on. And finally, if we turned all inputs on to true, the sum and carry LED turned on. Our full adder design is correct.

**Conclusion:**

We can conclude that making a sum and carrying out LED to turn on is to make the A, B, and Carry in to turn on. If there is only one input, there will be either sum and carry out on the terminal. If A or B with carry-in switches turn on, a carryout light will turn on its terminal. If the carry-in switch turns on, there will be a sum light on. Since the half adder can not manipulate adding the carry in this process, we use the process in the lab to make a full adder function to calculate the multiple bit adder. This is the process of how to call a function of the hierarchy level of the function. So we condense the half adder function down to one function, then we use the half adder to make a full adder function to manipulate the data in this process. The full adder can manipulate more data bits, and it can add it to carry in this process. This is why we are adding two-bit expressions for the lab purpose and having the number of carrying and a number of results present in this process.