Rodrigo Becerril Ferreyra
CECS 301 Section 03
Lab 1
Due 06 February 2020

1 Introduction

In this lab, we used the HDL Verilog to simulate strictly combinatorial logic. Combinatorial logic refers to logic whose output is only dependant on its inputs. The opposite type of logic is called sequential logic, which also depends on the output's previous state. We started by capturing the desired behavior in a truth table, reducing using K-Maps, and generating equations. These equations are used in the base- level full adder. After testing, we can then connect multiple full adders in order to make multiple-bit full adders, or vector full adders. In this lab, we made a four-bit adder and an eight-bit adder using the full adder.

2 Process

In this lab, the desired logic equations were given to us. These equations are well known and were derived in the class CECS 201. They can be found by documenting the desired action on a truth table, converting to sum-of- minterms, and reducing using boolean algebra. They are `y = a XOR b XOR ci` and `co = a*b + a*ci + b*ci`, where the sum of the three input bits `a`, `b`, and `ci` is the two output bits {`co,y`}.

These equations representing combinatorial logic were put into a module named `FA` for "Full Adder." This was then tested for correctness using a test fixture file provided by the instructor; its code simply chose three random values for the three input bits and checked whether or not they added to the two output bits. Once this was verified, the module was ready to be instantiated.

Instantiation of a module inside a different module is the process of using the first module's function in order to save text space or simplify a process. In this case, four `FA` modules were instantiated inside a different module named `FA4`. This allows two four-bit numbers to be added with a one-bit carry-in signal, instead of having every input be only one bit. Three extra wires were required in order to hold the carry-out bit of a less significant instantiation of the `FA` module and deposit it into the carry-in input of a more significant instantiation. Again, this was tested using test fixtures provided by the instructor, and it works the same way. Once it was verified that the `FA4` module was functioning properly, it was time to move on to the final step.

Lastly, it was required to instantiate two `FA4` modules into one large eight-bit adder, called `FA8`. The `FA8` module allows two eight-bit numbers to be added together. Because eight bits equals one byte, this allows the user to add two whole bytes of data: one byte can represent numbers from 0_{10} to 255_{10} unsigned, or -128_{10} to 127_{10} signed. This makes `FA8` a very powerful module. The way

this was achieved is by simply instantiating two `FA4` modules; this required an extra wire to hold the carry-out output of the first module and feed it into the carry- in input of the second module. Once this module was tested using the test fixture provided by the instructor, the lab was completed.

3 Conclusion

In this lab, I reviewed how to model combinatorial logic in Vivado, and how to instantiate modules to make other modules. I learned this last semester in CECS 201, but this lab was very helpful in reviewing this very important topic. This process of instantiation and building a working foundation in order to build higher-level modules is an important one that will most likely be critical in future labs.