Lab 2 Combinational Circuit-Adder Name: Jacob Hillebrand

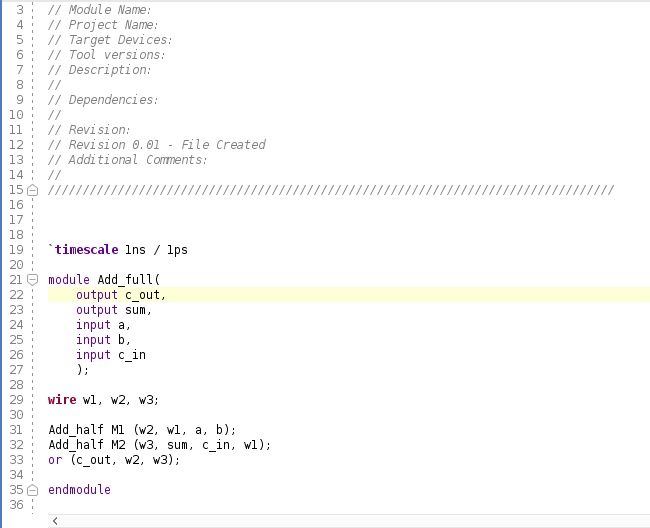
September 14, 2018

The primary purpose of this lab was to begin to explore the functionality of our FPGAs and using Verilog. To do this, we were assigned the task of creating a 16-bit adder based on half and full-adder designs nested upon one another. When properly organized, these smaller adder designs could work in conjunction to provide the functionality of a 16-bit adder.

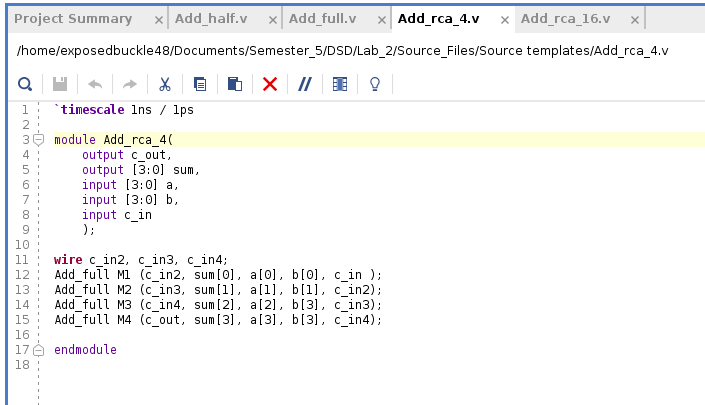
At the base of the structure was the half-adder, given by the file Add\_half.v (shown below)



The module takes in 2 bits of input, and processes them with an XOR and an AND gate to create the output of half of a full adder. When combined with another half adder, the 2 bits can be properly added together, thus completing the full adder. This was implemented with the the file called Add\_full.v (shown below)

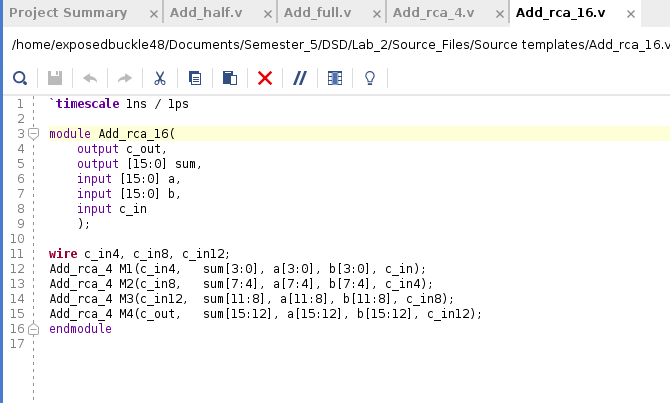


This module took in the output and carry bits from the two half-adders (as shown by the Add\_half M1 and Add\_half M2 functions above) and combined these with an or gate to give a full adder design. Once we had successfully created a full-adder, we could then begin stacking them to create the 4-bit-adder, as governed by the file Add\_rca\_4.v (shown below)

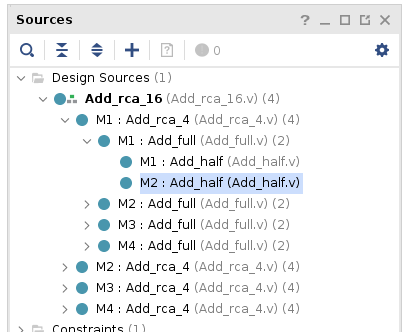


Similar to how the full-adder utilized two half-adders, the 4-bit adder utilized 4 full-adders (functions Add\_full M1, Add\_full M2, Add\_full M3, Add\_full M4), and simply wired them together and combined the outputs in the proper fashion to “ripple” the bits through the adder system.

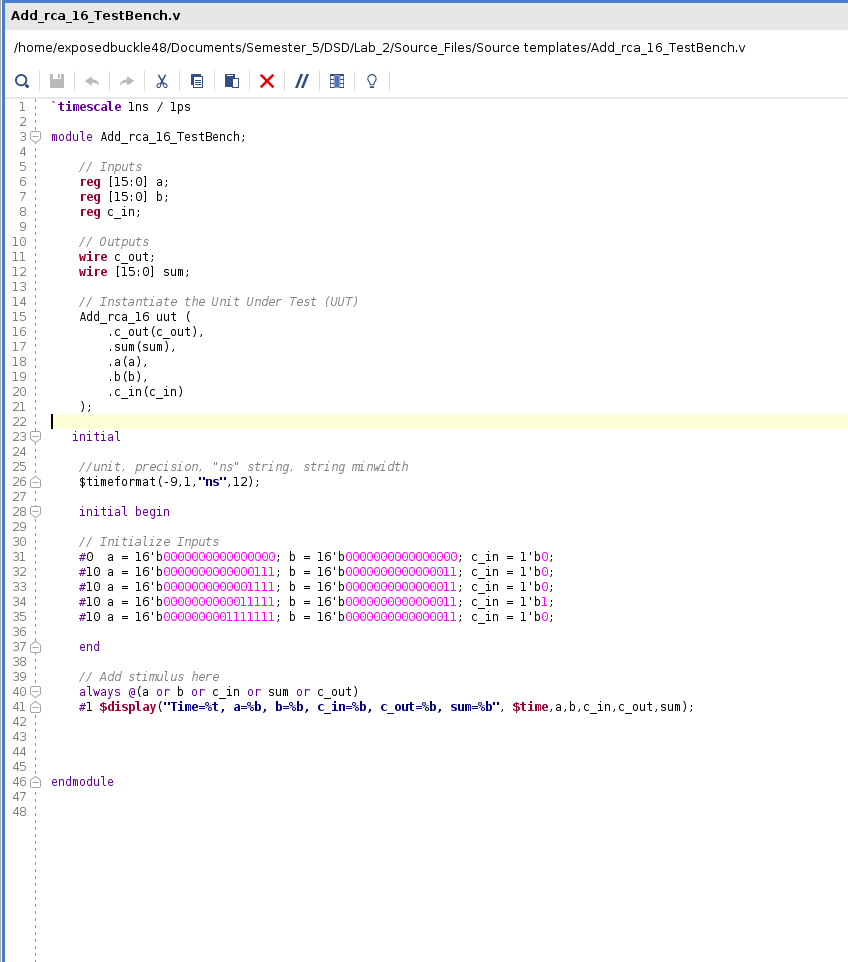
To finish off the hierarchy, four 4-bit adders were then combined to create the full 16-bit adder, as outlined in 16\_rca\_16.v (shown below)



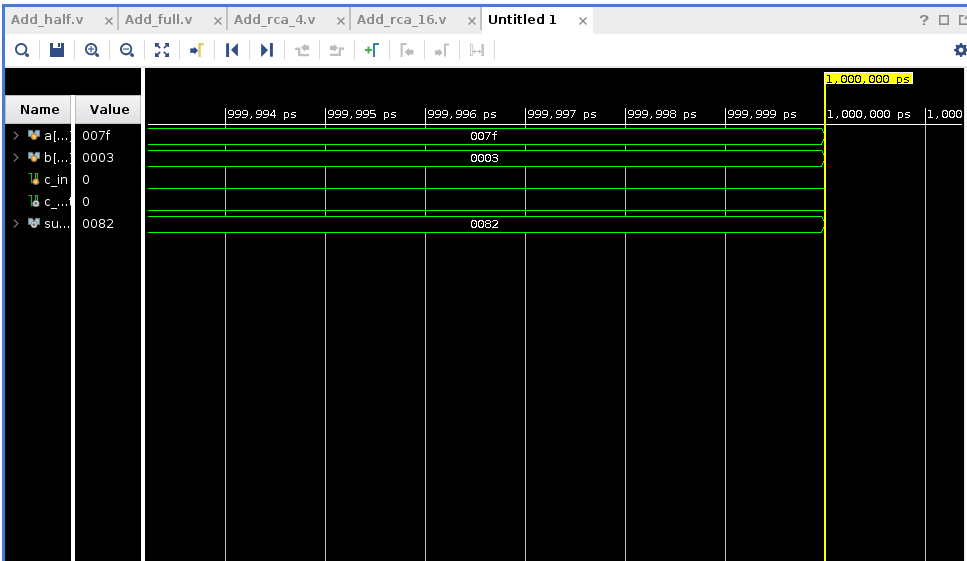
When complete, the adder file hierarchy looked like this



With the file hierarchy established, a testbench file was created, outlining the implementation of the Add\_rca\_16.v and dictating a few test conditions and console outputs to ensure the 16-bit adder was successfully created and tested (shown below)

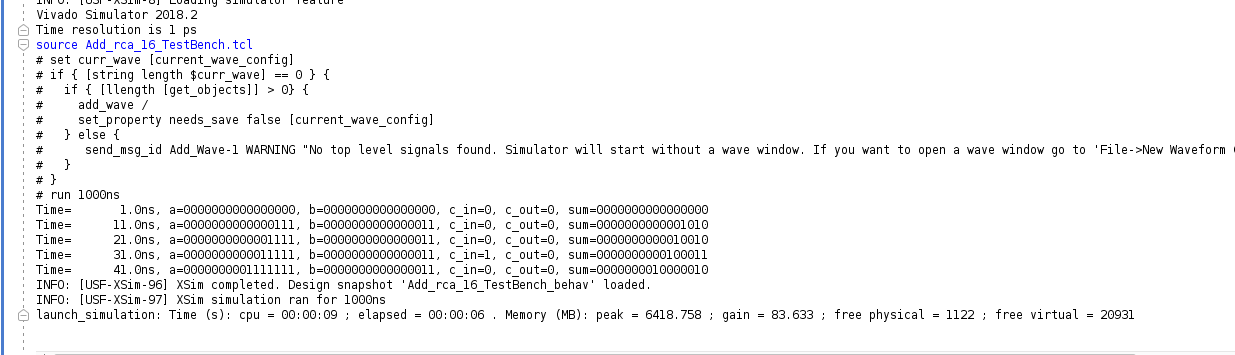


All that remained was to run the test simulation, which produced the following results



Simulation Graph

Console output



As shown in the console output above, the 16-bit adder was successfully implemented. In each test case, the given binary values for “a” and “b” were properly summed and produced in the output “sum.”

There were a few interesting things to note from this lab. Primarily, it is important to note that this design for an adder, known as a “ripple” structure, is very easy to implement, but can cause problems in larger adder structures. Understandably, there is a delay as the signal goes from adder to adder. As this lab involved only a 16-bit adder, the delay was negligible. However, had this been a larger circuit, the latency would have severely reduced the timeliness of the system, thus reducing overall efficacy. The second thing to note from this lab was the lack of implementation of the 16-bit adder. While the design passed the simulation with flying colors, it was not actually ever downloaded to the FPGA. This was due to the fact that this was a **16**-bit adder, coupled with the fact that the FPGA’s used have less than 16 test switches; the FPGA would not have been able to be test the adder. This was unfortunate, but fortuneately the simulation capability of Vivado still provided the experience of creating a functional 16-bit adder.