Lab Report 1: Introduction to ModelSim with a 4-bit Full Adder

Angelique Taylor, Dylan Samson, Seth Kurtenbach

Objective

The objective of this lab was for the group to familiarize itself with the ModelSim VHDL simulation tool and some of the basic syntax of VHDL. Specifically, the group was provided with VHDL code implementing a 4-bit full adder and was instructed to debug the code and simulate it multiple times using ModelSim, thereby verifying the code's functionality.

Lab Work

Our first task was to verify that we had correctly implemented the given VHDL code and corresponding usage instructions. The instructions asked us to simulate the addition problem 7 + 5 with no carry in, using the encoded 4-bit full adder. The correct solution to this problem is 12 with a carry out of 0, or binary 1100, with 0 for the carry out bit. Below is a screen capture of our simulated code, showing the correct result, at the correct times. The sum is denoted by s, and the carry out bit is denoted by co.

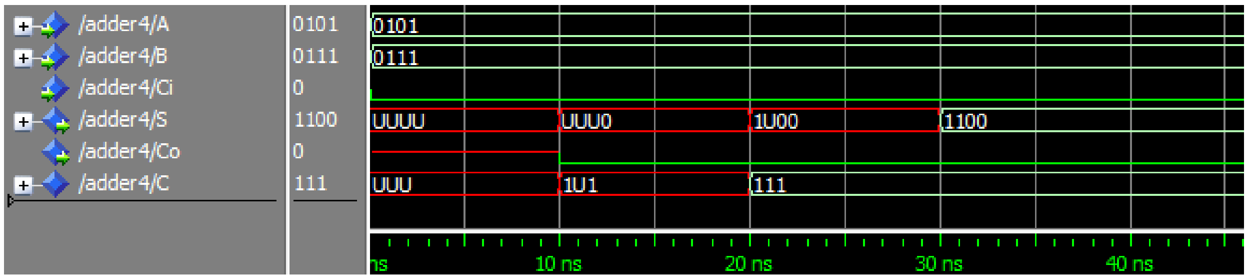

Figure 1: a <= 0101, b <= 0111, ci <= 0, summed in the full adder.

In order to ensure that the code functioned properly, we conducted a total of three additions, including the first. Our second addition tested the adder in the event of a carry in equal to 1. We set a = 8 (1000 binary), b = 6 (0110 binary), and ci = 1. The correct solution is 15 (1111 binary with carry out = 0), and indeed our implemented adder reached this result, with the proper delays. The result is shown in figure 2.

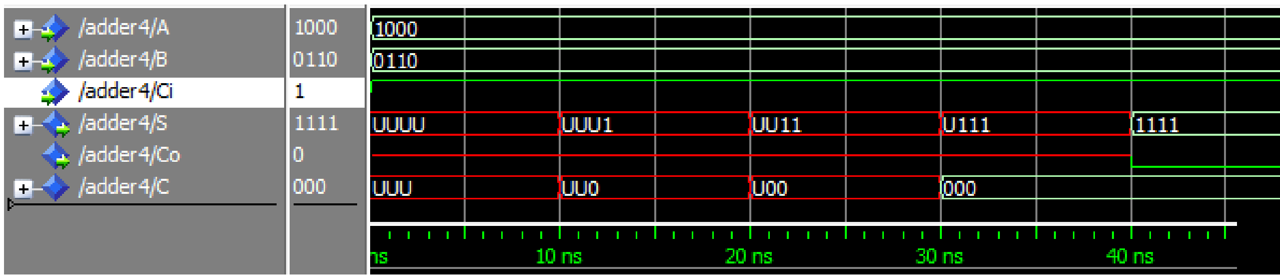

Figure 2: a <= 1000, b <= 0110, ci = 1, summed in the full adder.

Our final addition implementation sums 10 + 4 (1010 + 0100 binary) with a carry in of 1. The correct solution to this probem is also 15 (1111 binary), and indeed the adder achieves this result, with the correct delays, as shown in figure 3.

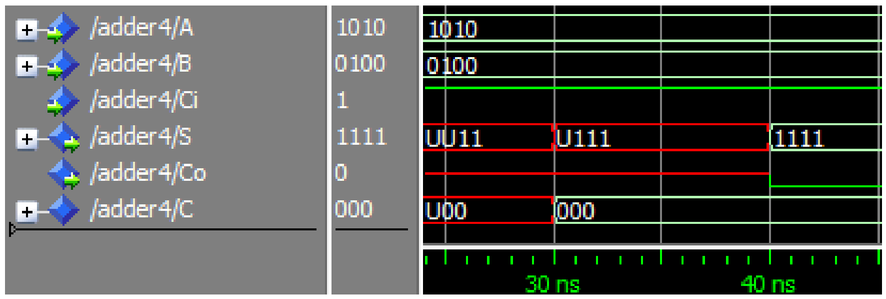

Figure 3: a <= 1010, b <= 0100, ci = 1, summed in the full adder.

In addition to printing and commenting on the results, we were also instructed to answer a question pertaining to the VHDL code itself. The *port map* statements in the VHDL code instantiate the full adder component described earlier in the code and assign specific elements from the input vectors, and the signals used in computation, to appropriate ports in the instantiate 4-bit full adder, comprising four instances of the full adder component. In order for the 4-bit full adder to work properly, the bits must be computed upon and passed around in the correct order, because binary numbers make use of positional notation, or place value. For example, in the first addition, we encode the *port map* of the first full adder instance with *(A(0), B(0), Ci, C(1), S(0));* because its role in the computation is to take the first bit of the first input vector, the first bit of the second input vector, and the carry-in bit, and add them all together. Its carry out bit is assigned to *C(1)*, which in turn gets transferred into the next full adder instance as the carry-in bit: *(A(1), B(1), $\underline{C(1)}$, C(2), S(1));*. The general definition of the full adder component describes five ports, and the architecture defines computations on the data from these ports. The instantiated full adders are mapped onto the general full adder component by the order of their input, first to first, second to second, etc, and the general full adder's defined computations operate accordingly.

Conclusion

The VHDL code provided by the lab instructions had a syntax error, a missing ";". The ModelSim compiler caught this error and indicated it to us, which we fixed by adding a ";" to the proper place at the end of the line. In addition to this, we initially had trouble interfacing our defined full adder with the provided code. Rather than adding the new file to the existing project, we created a new project by mistake. We solved this problem by adding the full adder component code to the existing project.