Tara Moses

010630228

Mondays 4:10 – 5:55

Lab #1: Introduction to EDK

February 8, 2016

**Introduction**

The objective of this lab was to introduce students to the ISE development kit by writing VHDL code to implement 3 things: (1) a full adder with 16-bit inputs; (2) a 4-to-1 multiplexer with 16-bit inputs; and (3) a register with 16-bit inputs.

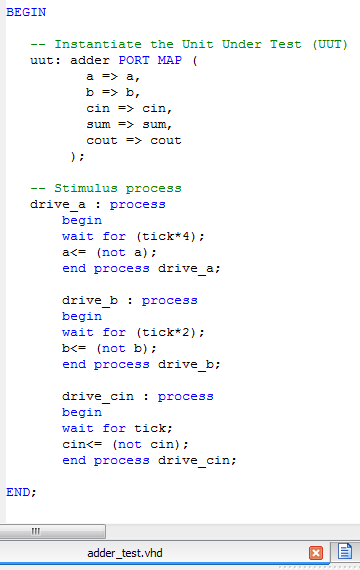
**Approach**

The full adder involved coding a 1-bit adder and then extending it to 16-bit inputs by using the ieee\_std\_unsigned.all library. The multiplexer involved taking in four 16-bit inputs and two one-bit selects. A with-X-select statement was used to choose which input to output. The two selects were and-ed into one two-bit signal as the X in the with-X-select statement. The 16-bit register took in one 16-bit input, a one-bit asynchronous clear, and a one-bit clock. The register outputted the 16-bit input if and only if the asynchronous clear was low.

**Experimentation**

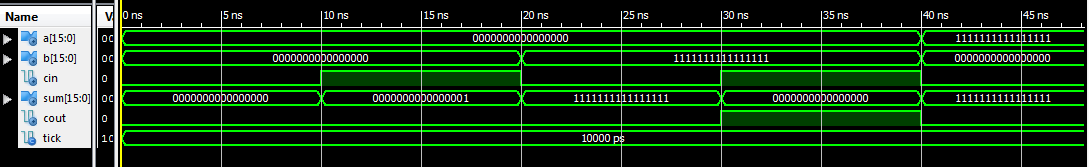
For the adder, the two inputs were added together using the ieee\_std\_unsigned.all library. The adder added two unsigned 16-bit numbers together. The multiplexer was built using the VHDL construct we learned in class called the with-X-select statement. It was used along with a two-bit signal representing the two selects for the multiplexer. The register was built using the same architecture as a D flip-flop, which was covered in Digital Design last semester. However, instead of it storing a one-bit input, this register stored and outputted a 16-bit input.

**Results**

The adder was tested using two 16-bit inputs, each changing from all 0s to all 1s. The carry-ins changed from 0 to 1 as well. The logic for the adder’s test-bench code is displayed in Figure 1 below.

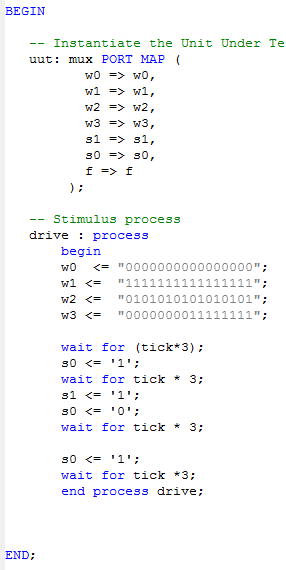
**Figure 1**: test-bench for full adder.

The simulation was run, and the waveform is shown in Figure 2 below. The simulation starts with both inputs equaling 0. Since the carry-in was 0, the output was 0. After the carry-in changed to 1, the output added “0000 0000 0000 0000”, “0000 0000 0000 0000”, and ‘1’ together and outputted “0000 0000 0000 0001”. A little bit later, “0000 0000 0000 0000” was added to “1111 1111 1111 1111” and ‘1’. As a result, the output was “0000 0000 0000 0000” since all of the inputs were unsigned. Overflow happened.



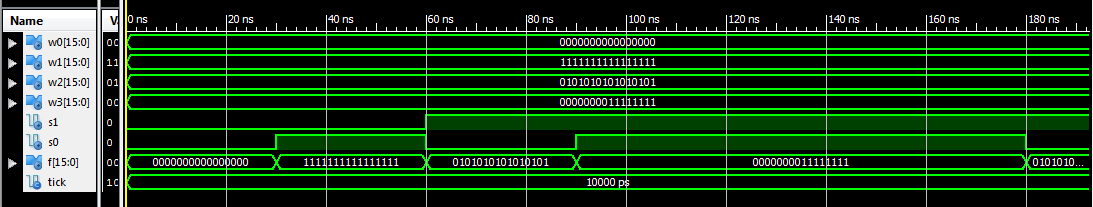
**Figure 2**: waveform diagram for full adder.

The second part of the lab was coding and testing a 4-to-1 multiplexer. The four inputs w0, w1, w2, and w3 for this mux are “0000 0000 0000 0000”, “1111 1111 1111 1111”, “0101 0101 0101 0101”, and “0000 0000 1111 1111”, respectively. The test-bench for this mux is shown in Figure 3 below.



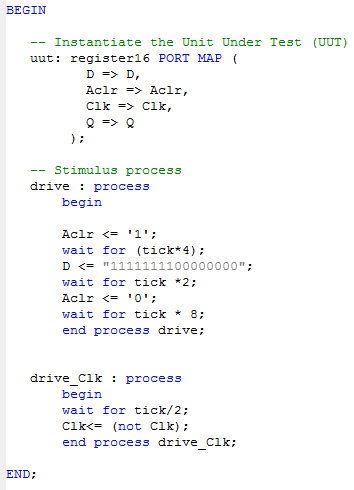
**Figure 3**: test-bench for 4-to-1 multiplexer.

The waveform simulation for the mux is shown in Figure 4 below. When s1 and s0 were both 0, w0 was outputted. When s1 was 0 and s0 was 1, w1 was outputted. When s1 was 1 and s0 was 0, w2 was outputted. For all other combinations of s1 and s0, w3 was outputted.



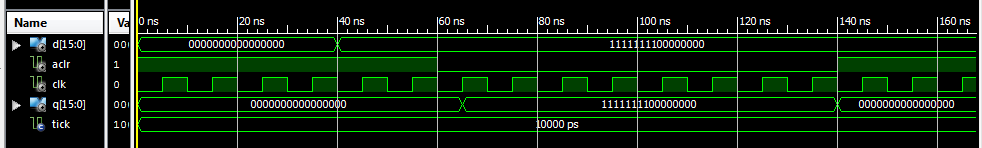
**Figure 4**: waveform diagram for 4-to-1 multiplexer.

The register was tested using one 16-bit input that started out as “0000 0000 0000 0000” but changed later on. The test-bench for the register is shown in Figure 5 below.



**Figure 5**: test-bench for 16-bit register.

The register’s simulation was run, and the waveform it generated is shown in Figure 6 below. The input D changed to “1111 1111 0000 0000” after the asynchronous clear went high to make sure that the asynchronous clear was working. The output when the asynchronous clear was high was always “0000 0000 0000 0000”, regardless of the value of D and Clk.



**Figure 6**: waveform diagram for 16-bit register.

**Conclusion**

All three of the VHDL projects worked as expected. The adder added what it was supposed to (including the expected overflow), the multiplexer selected what it was supposed to, and the register successfully outputted the value it was supposed to: either the stored data, or all 0s in the case of a high asynchronous clear. In this lab, I learned how to create a useful test-bench, something that I had not previously done in Digital Design.