Please attach this cover sheet to your completed homework

ECE 324 Homework5: FIFO simulation

Name: Alex Blake

Exercise	Course outcome	Grade
Homework5	2.a, 7.b	/30

2.a. Define engineering problems from specified needs for digital systems including implementation on FPGAs using HDL programming.
7.b. Employ appropriate learning strategies such as communicating with an expert, using external resources, experimentation, simulation, etc.

Introduction

Students are given a handful of modules all tied together into a single fifo module which acts as a FIFO buffer in hardware. Using the fifo module, write some assert statements that won't be synthesized but will be used during simulation for edge cases such as overflow as well as write a testbench to test all the features of the fifo module.

Procedure

After acquiring the files and setting up a project folder, analyze the code given. The fifo_ctrl module is a control module to change the pointers and to make sure no errors like overflow or underflow occur. Some assert statements are already included, but do not cover all edge cases, so those need to be added. Give an error message if both full and empty occur at the same time and give a warning if write is enabled while fifo is full (overflow).

Now create a testbench to verify the module works correctly. Make sure that DATA_WIDTH is set to 16 and ADDR_WIDTH is set so it has 8 possible combinations: $2^3 = 8, \rightarrow 3$ bits. Verify underflow and overflow case along with normal operation in a simulation. This concludes the assignment.

Results

The assert statements were simple to implement but was a learning experience; if assert statements are used without else statements, if they aren't true, they give errors constantly which is very annoying for debugging going through the console output. This is fixed by asserting the NOT of the assertion and using the else for the output. This is how the pre-existing assert statements were formatted, which made it clear that that is the way to do it.

Figure 1 shows the console output from running the code. Notice the underflow and overflow; the testbench was written to overflow and underflow once each to verify the asserts work. There also is the first error at time=0 where empty and full are 1. This is the only time it happens show it should be fine and not an issue if synthesized.

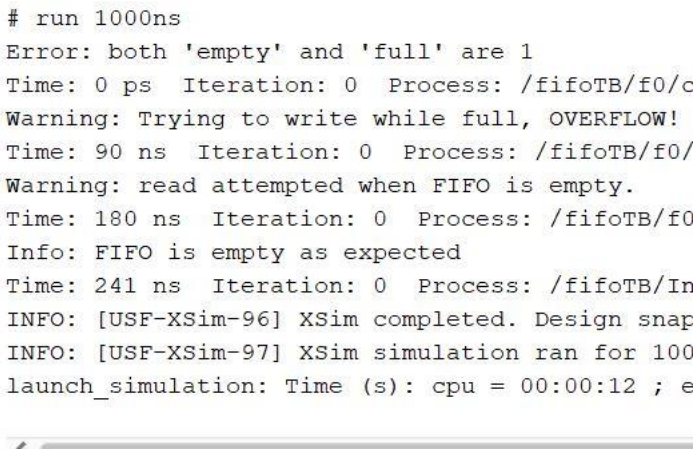

Figure 1: Console Output from Testbench Simulation

Figure 2 shows the waveform output (included in .zip if unreadable). After resetting in the beginning, wr=1 & rd=0. This stays true for 9 cycles (8 cycles of adding and 1 overflow at the end). This produced the overflow error at 90ns in Figure 1. Then the testbench does the same thing for reading instead of writing (wr=0, rd=1). This gave the underflow at 180ns (Figure 1). After that, both wr and rd are set high to read and write at the same time for 2 cycles. This is done after 3 cycles of writing to make sure it wasn't at an edge case of empty, so it does not start until 210ns. The module is then reset again at the end.

Figure 2: Waveform Output from Testbench Simulation

Figures 1 and 2 show that the code works exactly as expected. Both errors were given and the only error that was unexpected was the one at 0ns, but that is just from the beginning of the simulation and is fixed with the reset at the very beginning of the simulation and does not appear anywhere else. This completes the assignment.

Conclusion

This assignment was great practice in analyzing a pre-existing SystemVerilog module and how to add assert statements to test edge cases. The lab went well and did not have any issues. It also gave a great example of how to implement a FIFO buffer and how it could be used in future assignments along with the other previously used tools to test.