CMPE 260 Laboratory Exercise 5

Memory and Writeback Stage

Aden Crimmins
Performed: March 30, 2021
Submitted: April 13, 2021

Lab Section: 2
Instructor: Richard Cliver
TA: Corey Sheridan
Justin Soler
Jake Michalski

Lecture Section: 2
Professor: Mr. Cliver

By submitting this report, you attest that you neither have given nor have received any assistance (including writing, collecting data, plotting figures, tables or graphs, or using previous student reports as a reference), and you further acknowledge that giving or receiving such assistance will result in a failing grade for this course.

Your Signature: *Aden Crimmins*

Abstract

The purpose of this exercise was to translate the Memory and Writeback stages of the MIPS Datapath from their Verilog counterparts into VHDL. This stage focuses on the memory for longer term storage and loading, while redirecting specific signals into previous stages. Both the Data Memory and Writeback stages were tested using test benches to ensure proper functionality given different inputs. This lab was successful in designing the Memory and Writeback stages as well as enforcing the knowledge of Verilog.

Design Methodology

The design was created using the Xilinx Vivado Design Suite and implemented on the Basys3 FPGA. The Data Memory was translated to VHDL from Verilog. The Verilog implementation use a module to contain the entire data memory while the VHDL utilizes a entity followed by an architecture. In Verilog the inputs and outputs are listed in the initial module instantiation, then defined below as inputs and outputs, with parameters for the generic values. In VHDL each input and output is created within the entity assigning each to input and output at the same time. The memory is created in the Verilog as a reg that is a 2-D array of size defined by the RAM_ADDR_BITS holding data which has a size defined by the RAM_WIDTH. This is done similarly in the VHDL code after architecture but before the architecture begins. In the Verilog code the memory is written to using an always statement defined by the positive edge of the clock, afterwards write enable is checked and the data is written into memory at the provided address if the enable is high. In VHDL this is done in a process with the clock contained in the sensitivity list. It checks for a rising edge of the clock and then if the enable is high then it writes the data into memory. As switches and the seven segment display were not implemented the RD variable is always assigned to read the value at the specified address within the memory.

The Writeback stage was also translated from Verilog to VHDL with the same module to entity and architecture format, keeping each input and output as the same data type. In the Verilog there is an always statement that detects any time MemData, MemRd, or RegData are changed running the process if one changes. If MemRd is enabled then the result is the data from the memory, otherwise the result is the data from the register. This was converted to a process with the same three variables in the sensitivity list. The same logic operation based on the value of MemRd is performed to determine if the memory or register data is used. The Verilog code then uses assign statements to pass through the RegWrite and WriteReg values as RegWriteOut and WriteRegOut respectively.

Results and Analysis

The testbenches created for this lab tested a variety of cases of each of the input values for the functionality of the Data Memory and Writeback Stages using different values within each to ensure full functionality of each stage and their components. Figure 1 shows the behavioral waveform for the Data Memory stage.

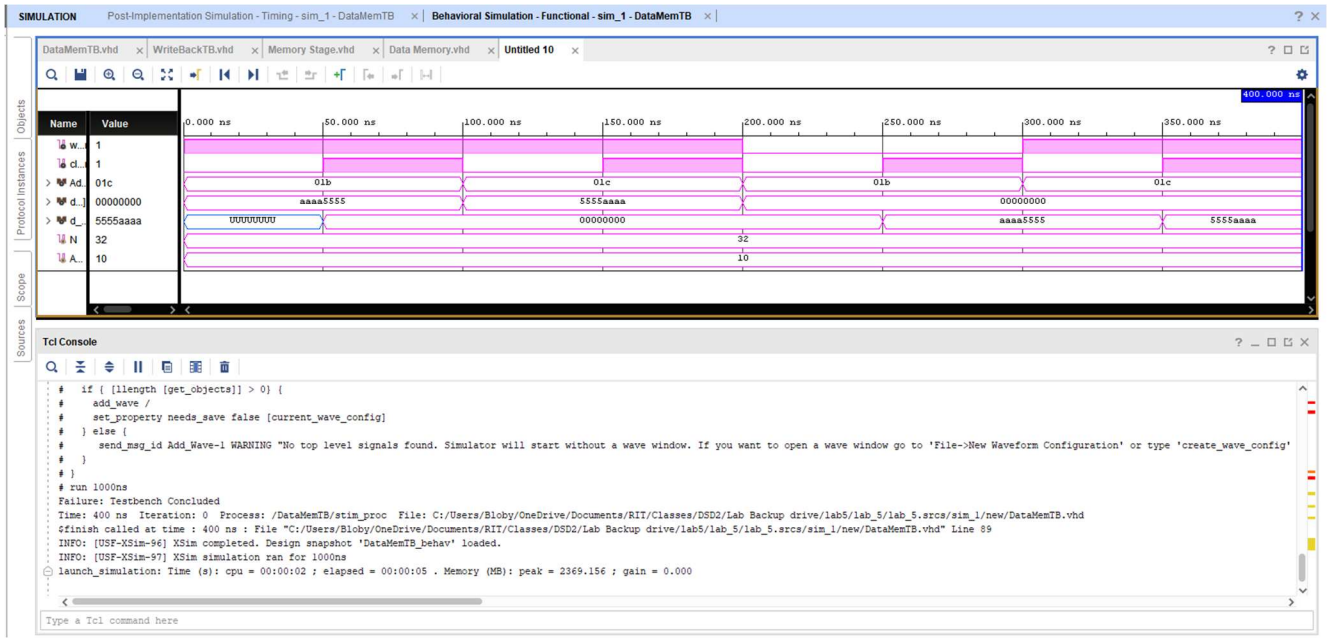

Figure 1: Data Memory Behavioral Simulation

The test bench starts by writing a value of x"AAAA5555" into the memory at x"1B" followed by writing x"5555AAAA" into memory at x"1C". Then the test bench turns off the write and sets the address back to x"1B" showing that the output from reading at that location now has a value of x"AAAA5555", then doing the same for the address of x"1C" showing a value of x"5555AAAA" in the memory. Each of the operations tested by the test bench matched the output specified by the testbench without any errors being thrown.

Figure 2 shows the Post-Synthesis Timing simulation of the Data Memory setup using the same inputs.

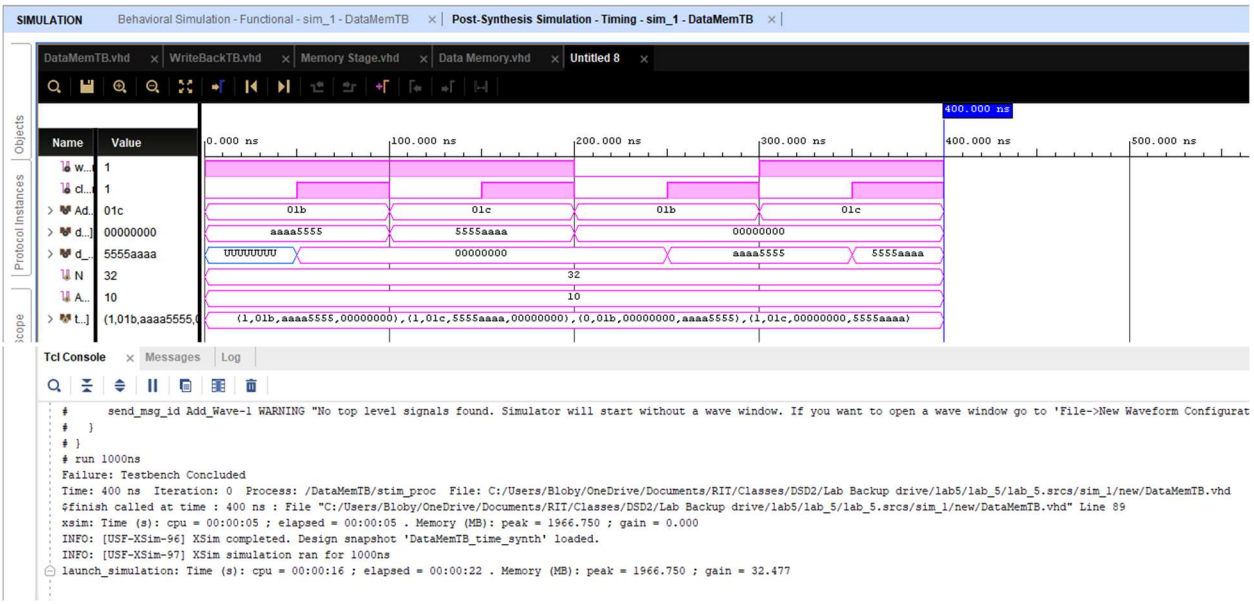

Figure 2: Data Memory Synthesis Simulation

The Post-Synthesis timing simulation tests the gate level model of the code without accounting for any delays. This waveform shows the same results as the behavioral simulation which means that there arent any major design flaws that are visible within the synthesis simulation.
The next test was the Post-Implementation Timing Simulation shown in figure 3 which tests the same set of inputs as the behavioral simulation.

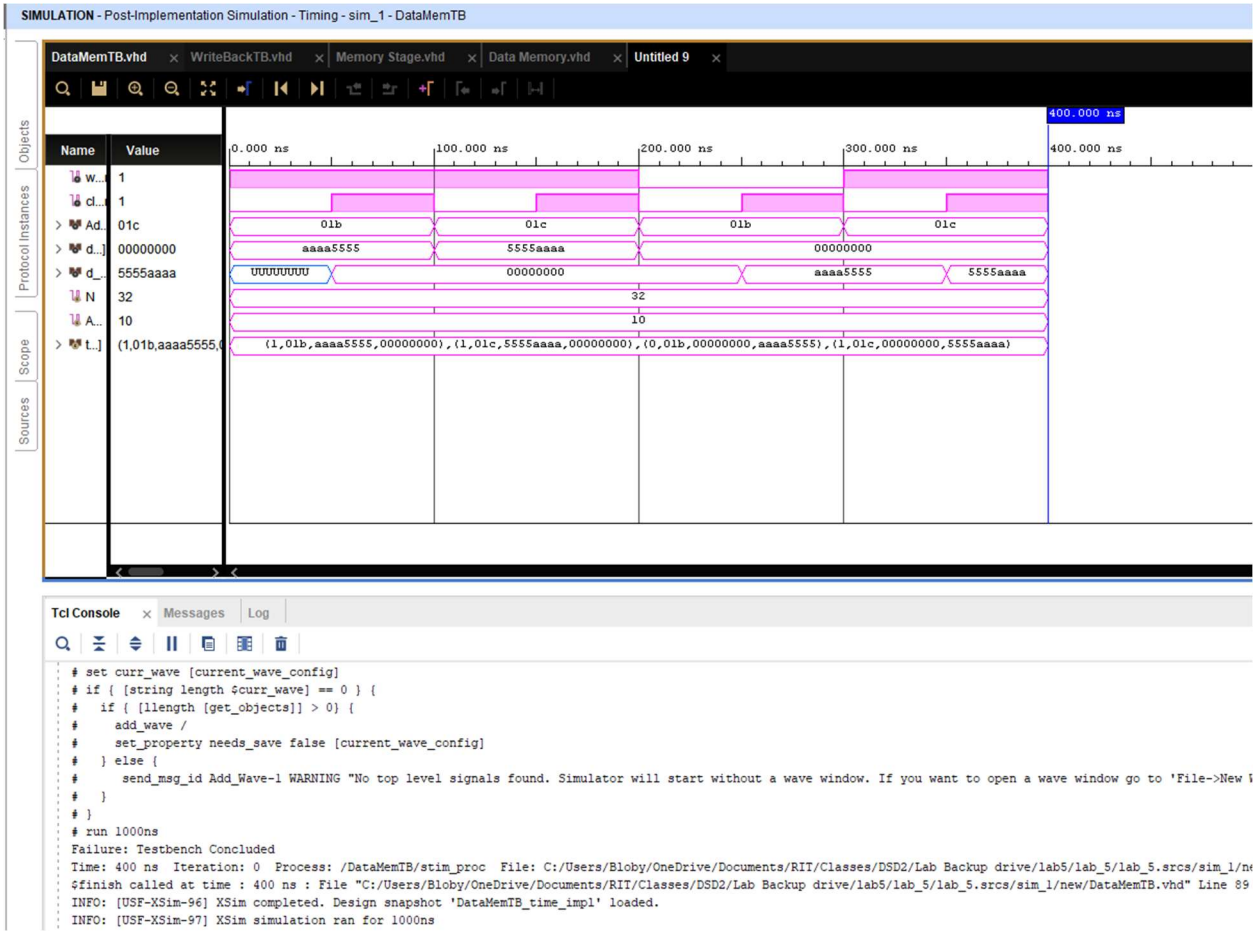

Figure 3: Data Memory Implementation Simulation

The Post-Implementation timing simulation tested the implemented model as it would behave on the actual hardware. This includes the delays resulting from each gate in the model which can be seen whenever there is a change in the inputs as there is delay between that change and the resulting output. This delay is relatively small as there is relatively few gates that the logic has to move through due to the simple nature of the Data Memory stage.
The Writeback Stage has its own testbench to confirm its' functionality and the behavioral simulation is shown in Figure 4.

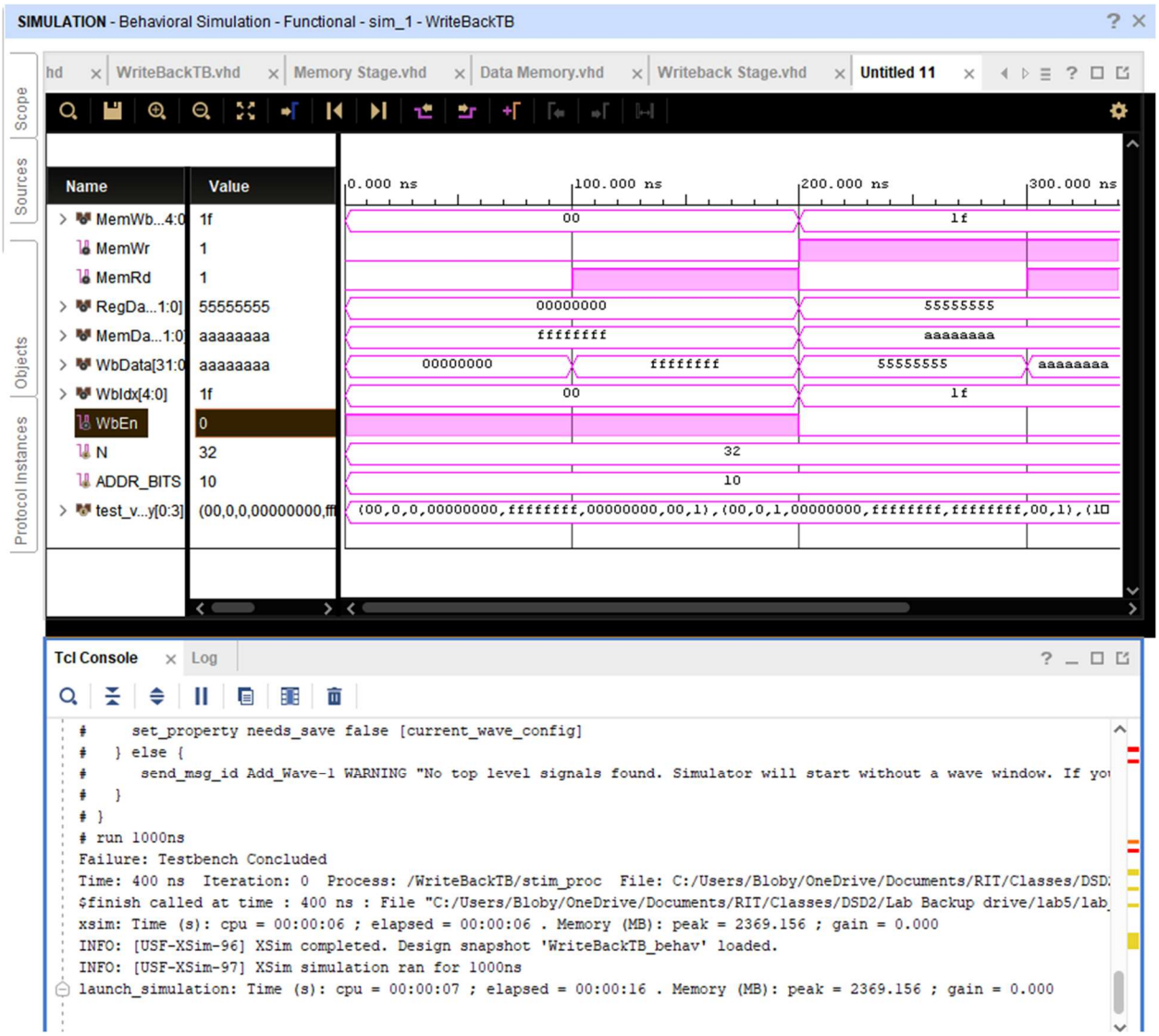

Figure 4: Writeback Stage Behavioral Simulation

The Writeback Stage was tested by making sure each of the passthrough bits was correct, while also testing to make sure that it correctly used the MemRd bit to choose between the register and memory data. It first tested the case where the MemWbIdx value was “00000”, then “11111” and asserted that the output WbIdx should be the same as these inputs without causing the flag. MemWr was tested to make sure it had the NOT operation performed before being passed through as the WbEn output by setting it first to 0 then 1 then checking if it was 1 after the first input and 0 after the second. These tests confirmed that the Multiplier functions as desired in this operation. To test the register and memory data function MemRd was set to 0 followed by one given the same set of inputs for MemData and RegData showing that when 0 it would select the Regdata value and when 1 it would select MemRd. This test was repeated once more with 2 different values and did not cause the flag on the assert statement to generate proving that the Writeback logic was implemented correctly.

The results obtained by these two stages show that both the Data Memory and the Writeback stages are functional on their own and are ready to be added to the overall MIPS design.

Conclusion

The Data Memory and Writeback Stages created were implemented successfully to complete the functionality to perform the correct operations with the given inputs, outputting the correct values for each set of inputs. Each of these stages were tested in the Xilinx architecture through the Behavioral, Post-synthesis and Post-Implementation simulations with the test benches designed for both stages. The simulations showed that both stages were designed correctly. There were minor issues when creating the test benches due to an error that was resolved by switching back from VHDL 2008 to normal VHDL and reformatting the code accordingly. This exercise was successful in both the design of the Data Memory and Writeback stages, as well as the principles necessary to create it including the conversion from Verilog to VHDL. It was also successful in providing more practice in the creation of functional test benches to test the created code.