CMPE 260 Laboratory

Project 1 Pipelined MIPS Processor

Dhruv Rajpurohit
Performed: October 30, 2020
Submitted: November 13, 2020

Lab Section: 1
Instructor: Mr. Richard Cliver

TAs: Seth Gower
Michael Nichols

Lecture Section: 1
Professor: Mr. Richard 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: **Dhruv Rajpurohit**

Abstract

This project was to assemble 5 different stages of microprocessor to implement a complete MIPS processor architecture in VHDL. The implementation was performed in Vivado IDE. The 5 stages work as a pipeline where the first stage is stored in a register and the output of this stage is an input for the next stage. The main objective of the project was to get a clear understanding of how the MIPS Datapath function when all the stages are assembled as one component. The expectation from the implementation was to get the instruction from the memory and execute them per cycle. The project also expected the writeback stage to write the number of a Fibonacci series back to the data memory.

Design Methodology

To implement the MIPS data path all the stages were assembled one by one. First the fetch stage was executed to fetch the instruction from memory. Then the decode stage was implemented to use the output from the fetch stage as input and decode it machine language and to return the op code so that the next stage can execute it. The third stage is the execute stage where the execution of the instruction takes place. The instruction is executed using the opcode from the previous stage. The op code basically is used to identify the ALU in return the command is also identified by decode stage and this is what the execute stage uses to compute the result using the command. The input values are also identified by the decode stage and the execute stage takes it as its input. Once, the execution of the instruction is done the result is stored in memory. Memory is next stage where the result computed from the execute stage is stored for a long time. This stage exists because the register can only hold certain data for a limited time only that is why the memory stage is implemented. The memory knows what data has to be stored by the output from the execute stage. The next stage is writeback stage, this stage returns the signal sent by the register to previous stage if the stage requires the signal.

For the next part of the project Fibonacci series was implemented. It was implemented to check if all the stages of the MIPS processor are functioning as required. The Fibonacci numbers were calculated by the processor itself. The calculation was done in such a way that the first number was stored by memory stage and then the next number was calculated using the writeback stage sending the signal back to the execute stage to compute the next number in the series. This process continued until there were 10 Fibonacci numbers.

Result and Analysis

Once all the stages were assembled a testbench was created to return the seven segment display values and the ALU results from the MIPS processor. The MIPS processor was simulated in Vivado to check if the required values are being received and the processor is functions as required. Figure 1 shows the behavioral simulation of the MIPS processor.

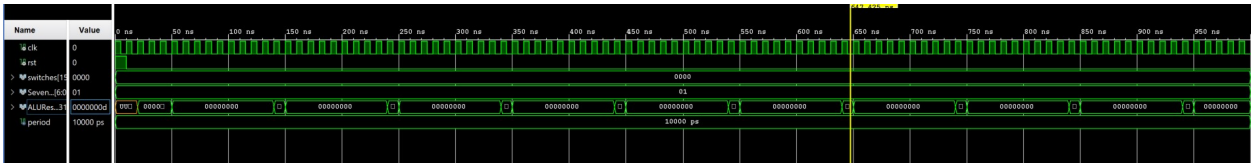

Figure 1 Behavioral Simulation of the MIPS Processor

In the simulation it can be seen that the input for the MIPS process are the switches, reset and clock cycle are the inputs and the outputs are the ALU results and the seven segment displays. For each clock cycle the output is being given in the ALU result wave and the seven-segment display. The ALU result is 32 bit long and the seven-segment display is 7 bits long. The MIPS process was implemented correctly because from the wave it can be seen that for each clock cycle there is an output. For a single switch value there are different ALU results and a constant

seven-segment display which indicates the MIPS processor is functioning correctly. Once the simulation in figure 1 was performed the next part of the project was simulated where the Fibonacci sequence was to be implemented. Figure 2 shows the simulation obtained by running the second part implementation.

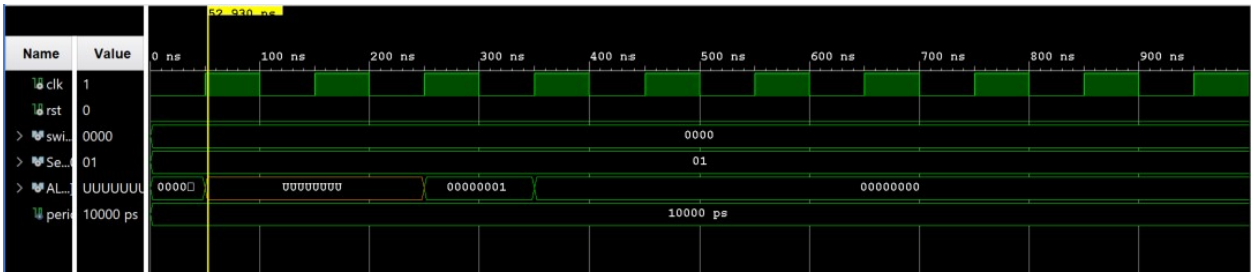

Figure 2 Post-Implementation Simulation of the memory stage

The objective of the second part was to make the MIPS processor compute the Fibonacci values. From the simulation it can be seen that for a longer clock cycle the value for the ALU result is being returned back that is why the clock cycle is taking longer.

Conclusion

Implementing the project consisted of many challenges and hurdles. The trickiest part was to assemble all the stages as one and make a testbench, but it was somehow tackled by closely looking at the functionality. The second part was also difficult to implement as it required a lot of new functions for most of the stages, but it was somehow worked out. The outcome from the project was that the understanding of MIPS processor got broadened and clearer. The results computed from the implementation also worked out well and returned what was expected from the results.