EC413 Alex Bleda
Final Project Barron Roth

MULTICYCLE CPU

Overview

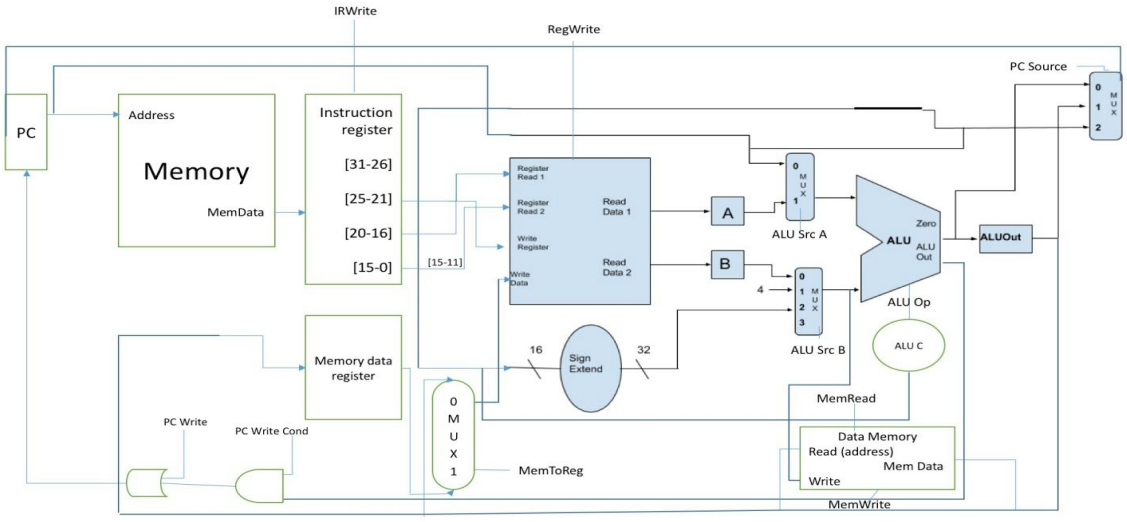
This multicycle central processing unit project was all about learning how to implement a pseudo MIPS processor. It was about learning how modern processors are implemented and essentially how a computer really works by learning its organization and architecture. This project was challenging but very rewarding and gives a great introduction to the basic organization of processors.

The main modules or parts of our CPU are the datapath and the controller. The datapath is the actual hardware the data moves around “data path”, and the controller is the module which controls which data is moving where at any given time. In building the overall multicycle CPU, we combine the controller with the datapath module, which contains two inputs (clock + reset) and zero outputs (it is a closed system). The controller itself will also have the clock and reset inputs, along with an OpCode input resulting from the output of the datapath module. The datapath will have clock, reset, and control signal inputs; the latter being the output of the controller module.

In the following pages we first include the datapath schematic which includes all the hardware elements and the controller control lines. The control lines are the named lines coming out of certain datapath elements, some examples are IRWrite or RegWrite. After our final datapath schematic, we include both the very initial state diagram we drew and the final. We decided to include both to give a sense on how we evolved from our initial idea once we started implementing our design. Keeping account of all the parameters and the format of each instruction became much more important than we initially thought.

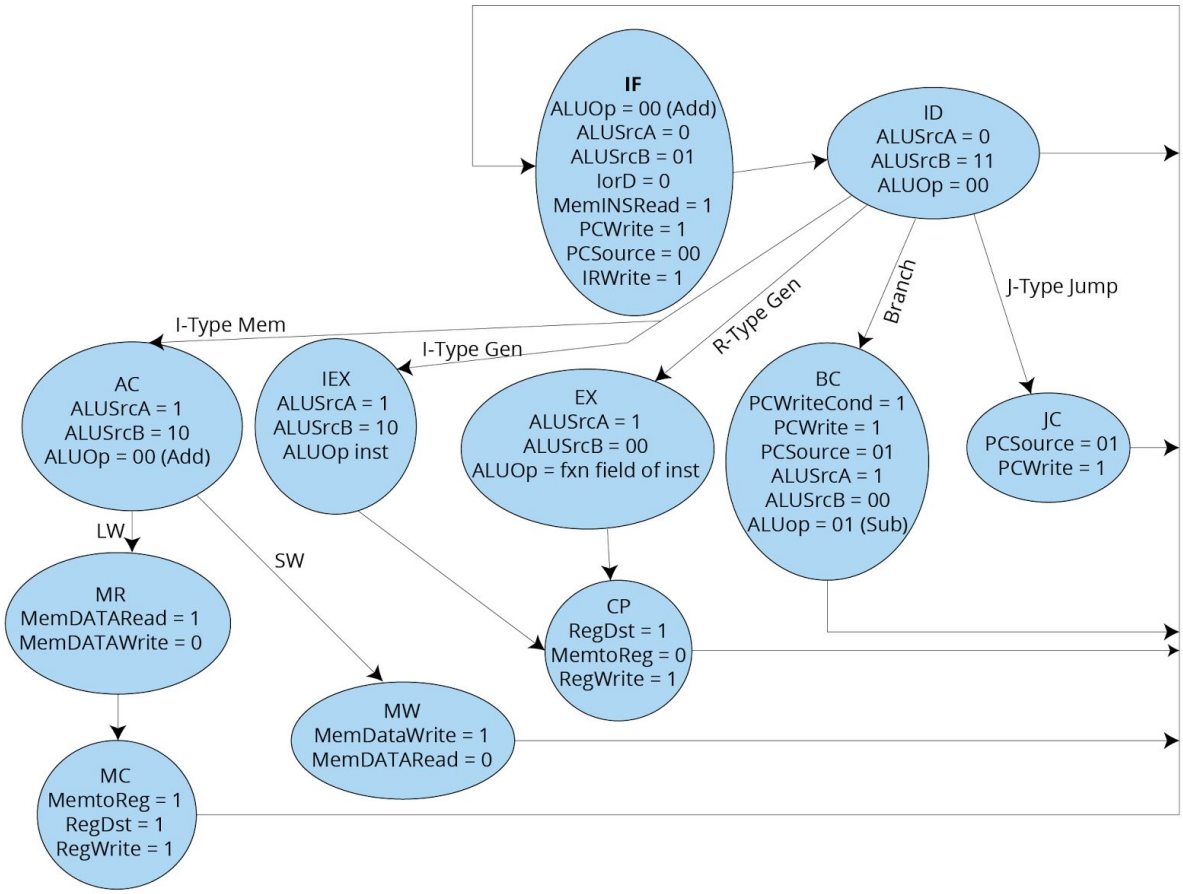
Finally, we include a description of some of the implementation and testing of the CPU with the test waveforms from our verilog test bench and a brief conclusion at the very end. All verilog modules including the test bench will be attached as well as copy of the test waveform pictures.

Datapath Schematic:

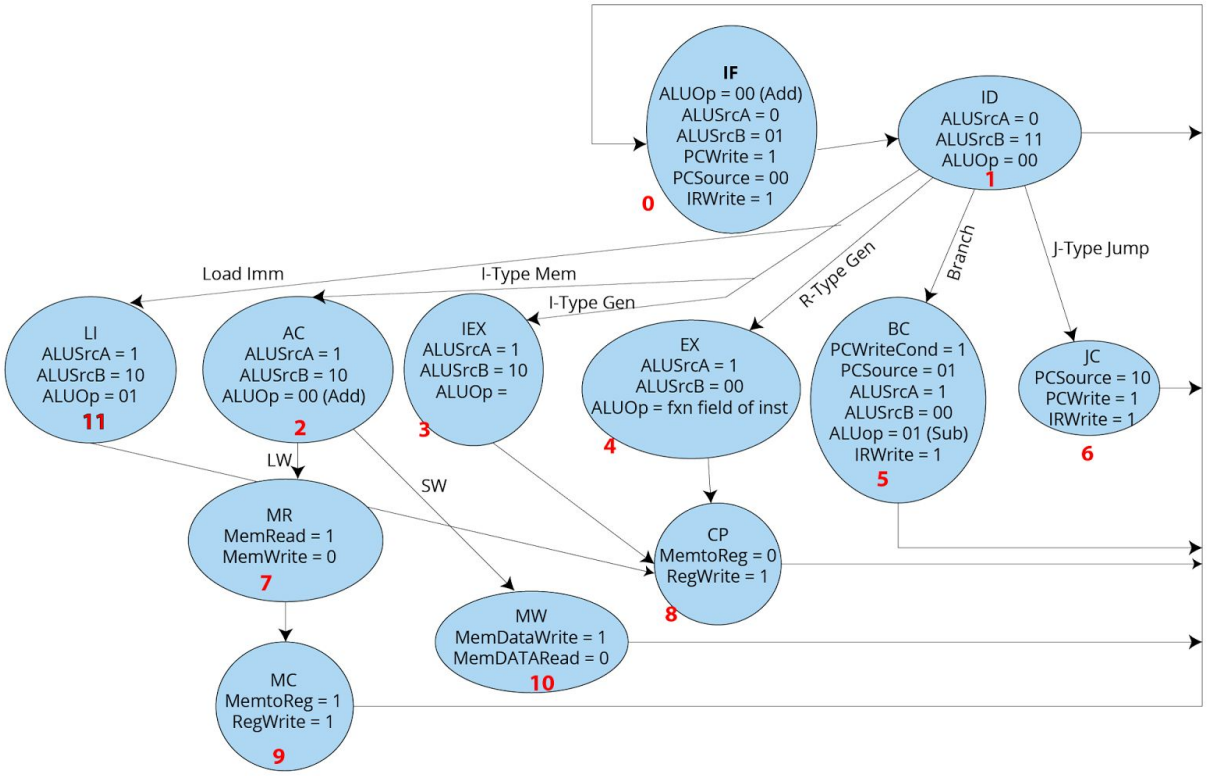


Final Datapath Schematic

State Diagrams



Initial State Diagram



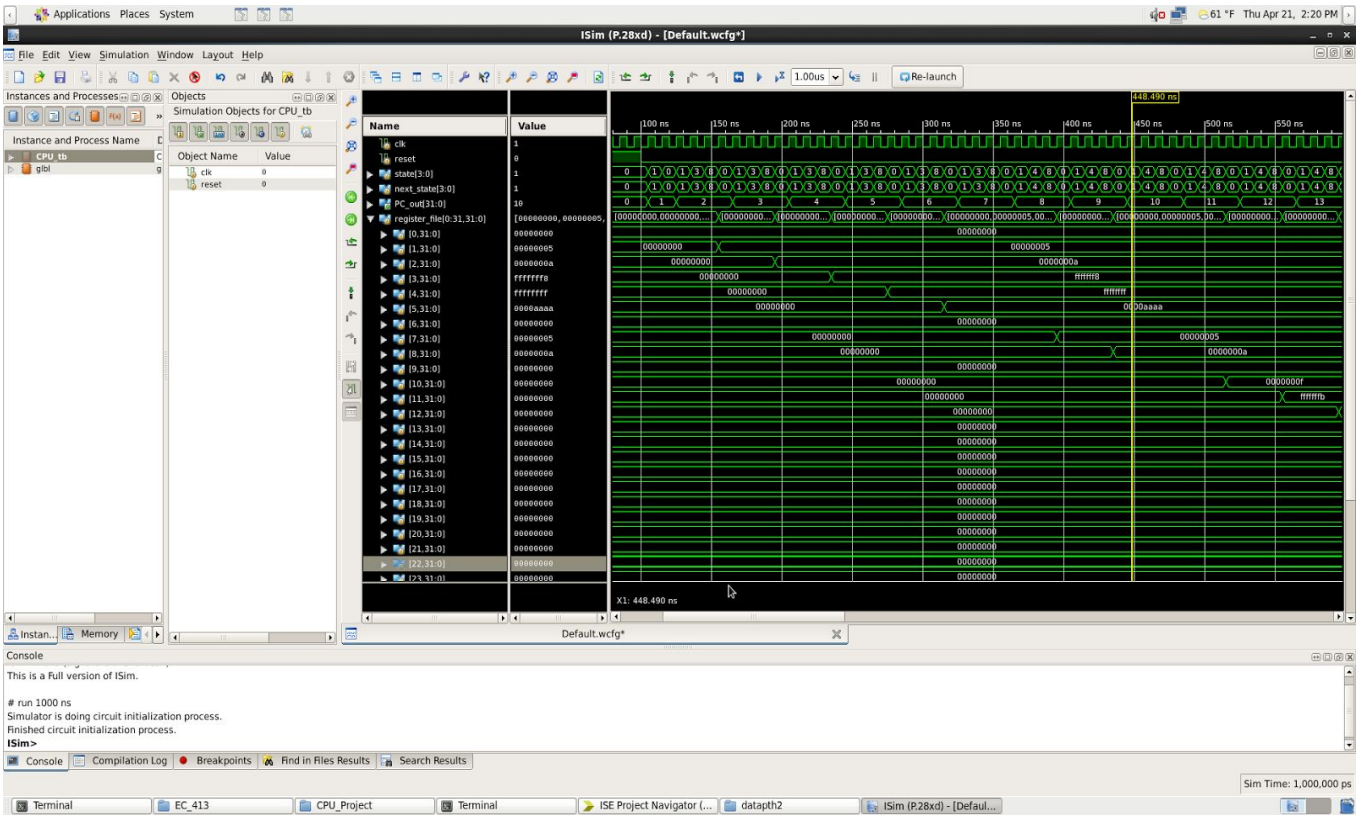
Final State Diagram

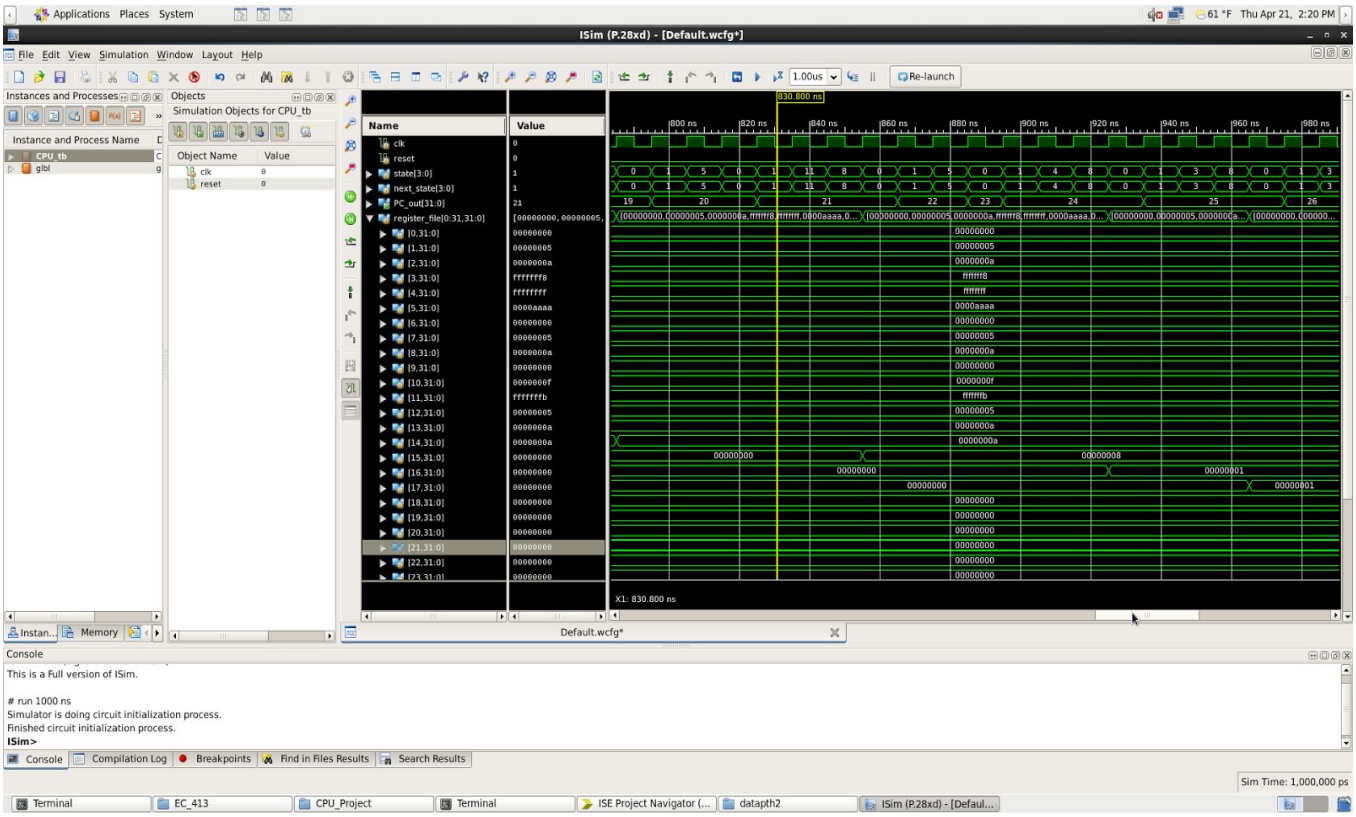
CPU Implementation and Testing:

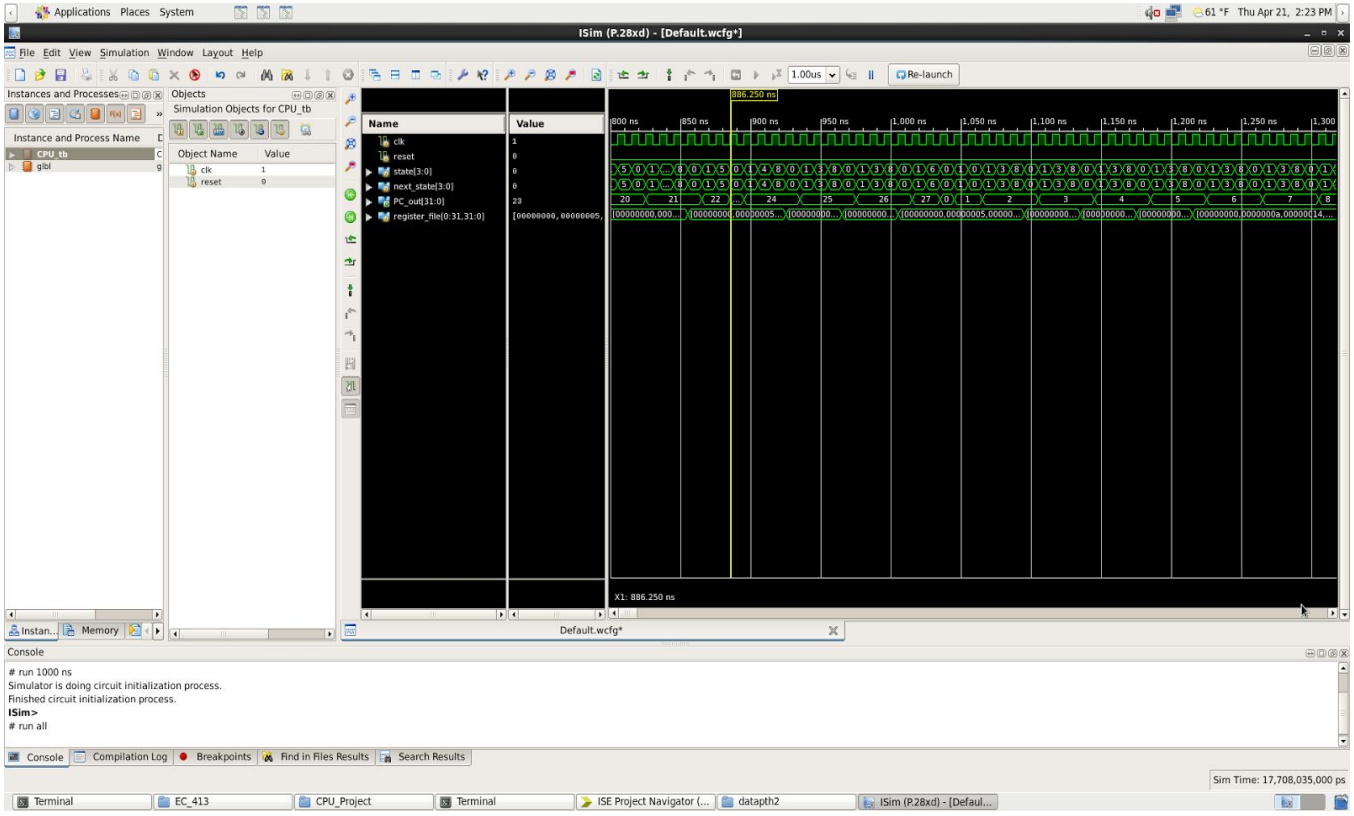
The differences between our initial schematic and diagram show how much we learned while implementing through the final functional CPU. Many tiny details slipped at the beginning that we later saw while spending hours looking at waveforms. Testing the controller by itself was relatively easy since we only had to check that the states and outputs were set correctly. The datapath testing was much more challenging since without the controller to help with the timing it was hard to keep track of how long each instruction was executed, making it safer to only test one instruction at a time. But the hardest test was putting both together and making the CPU execute the right instructions at the right time. It was when we finally put this together that we realized our smallest and biggest problems. Some of these problems include the destination register which differs in the Branch and Load Word Immediate instructions, or the realization that the three last bits of the opcode did not match the appropriate function for the ALU in the Load Immediate Instruction which caused us to add a new state to our diagram.

Our final multicycle CPU after going through all the errors is depicted below with the test waveforms from our CPU test bench. The test bench was simple and had no outputs since the CPU is a closed system. The test waveforms depict the changes in the register file which are essentially the main and fastest way to store and manipulate the data in a CPU. The following ones represent the instructions written in the Instruction Memory module attached in the folder. They include 26 instructions with branches and jumps so that not actually all of them end up being executed.

Test Waveforms:




ISim (P.28xd) - [Default.wcfg*]
File Edit View Simulation Window Layout Help
Re-launch
Instances and Processes
Objects
Simulation Objects for CPU_tb
Name
Value
clk
reset
state[3:0]
next_state[3:0]
PC_out[31:0]
register_file[0:31,31:0]
830.800 ns
X1: 830.800 ns
Default.wcfg*
Console
This is a Full version of ISim.
run 1000 ns
Simulator is doing circuit initialization process.
Finished circuit initialization process.
ISim>
Console
Compilation Log
Breakpoints
Find in Files Results
Search Results
Sim Time: 1,000,000 ps
Terminal
EC_413
CPU_Project
Terminal
ISE Project Navigator (...
datapth2
ISim (P.28xd) - [Defaul...


ISim (P.28xd) - [Default.wcfg*]
File Edit View Simulation Window Layout Help
Re-launch
Instances and Processes
Objects
Simulation Objects for CPU_tb
Name
Value
clk
reset
state[3:0]
next_state[3:0]
PC_out[31:0]
register_file[0:31,31:0]
886.250 ns
X1: 886.250 ns
Default.wcfg*
Console
run 1000 ns
Simulator is doing circuit initialization process.
Finished circuit initialization process.
ISim>
run all
Console
Compilation Log
Breakpoints
Find in Files Results
Search Results
Sim Time: 17,708,035,000 ps
Terminal
EC_413
CPU_Project
Terminal
ISE Project Navigator (...
datapth2
ISim (P.28xd) - [Defaul...

Conclusion:

After going through the whole project we realize how much work and persistence is required to build hardware. Debugging is a completely different thing, but looking at waveforms and signals forces you to really visualize and understand what is going on. Learning to combine datapath and a control in harmony was a challenging but rewarding task, and we believe that it serves as a great learning experience, not only for building more hardware, but also for future complex problems that require big picture thinking. We can only admire how a modern processors works now that we know how to build one a rudimentary one. We would recommend this project to anyone who really wants to learn how computers work. We believe that more people should in fact do it given the fact that the majority of the population in this world does not know what is underneath of perhaps the most powerful machine man has ever created.