Middle East Technical University
Electrical & Electronics Engineering Department

EE446 Computer Architecture II - Experiment 3

Preliminary Work Report

Necati Teoman Bahar 2515583

April 6, 2025

Introduction

In this report, the development process of a Multicycle ARM Processor that is capable of executing a limited amount of ARM instructions will be inspected and documented. The processer design method will be upgrading and modifying the previously proposed **Single-Cycle Processor**. The datapath will remain mostly unchanged apart from small additions, while the controller design approach will change to an **FSM** to accommodate the multicycle operations. In the upcoming sections, each block will be explained in further detail. Finally the testbench results will be shared and analyzed.

Datapath Design

As the multicycle processor is the successive architecture of the single-cycle processor, most of the datapath connections remain same. One of the significant change is the merging of the instruction and data memory into a single "ID Memory". Another one is the removal of all of the adders in the datapath and adding registers throughout the system. The removal of the adders is one of the main advantages of the multicycle processor, as it uses the already present ALU for all the arithmetical operations in the system by dividing the instructions into cycles, which is achieved by the additional registers placed in the system. The finalized datapath RTL view can be seen in Figure 1. The design is done using gate-level approach which only utilizes wires and module instantiations similar to the previous experiment that was done.

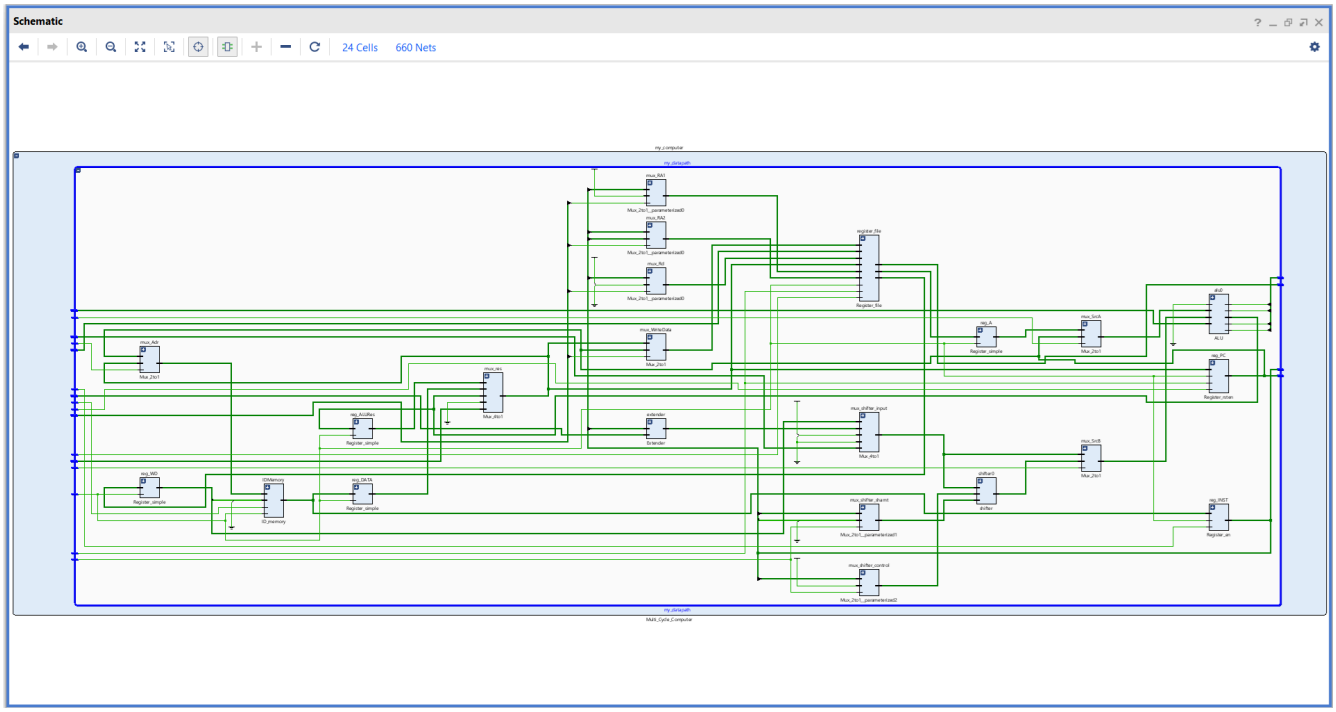

Figure 1: RTL view of the Finalized Datapath

MOV Instruction and Shifter

Similar to the single-cycle processor, a shifter with additional MUXs at the output and control signals are placed to the datapath. The MUXs are responsible for the rotation control signals when an instruction is need them, and the output MUX chooses between the nonshifted and shifted result as the finalized result.

Branch Instructions

Apart from the initial **Branch** instruction a constant binary is placed to the **"Write Destination"** port of the register file with a MUX. This MUX gives constant *"4'b1110"* or the R_d part of the instruction. Also the **"Write Data"** input is MUXed with the ALUResult and the "PCPlus4" to achieve proper BL instruction. For the **BX** instruction, the final datapath is sufficient. The proper control signal generation achieves the function.

Controller Design

As the multicycle processor is an **FSM**, a state machine is written at the beginning. The proposed state machine is derived from the EE446 Lecture Notes used in the lectures. To create proper signals, the instruction bits are decoded and case signals, such as **Data-Processing, Branch and Memory**, are created. The created signals are then used with condition check signals and states to create final control signals that enters the datapath.

The control signals for the shifter is created by finding the needs for each instruction then using state signals to enable and disable correct components within the datapath. The next state determination uses the decoded signals and the current signals like a normal FSM. For the control signals of the added components, criteria are selected and embedded within a case block.

In addition, the design is able to handle all instructions within the desired cycle amounts. The RTL view of the finalized controller design can be seen in Figure 2.

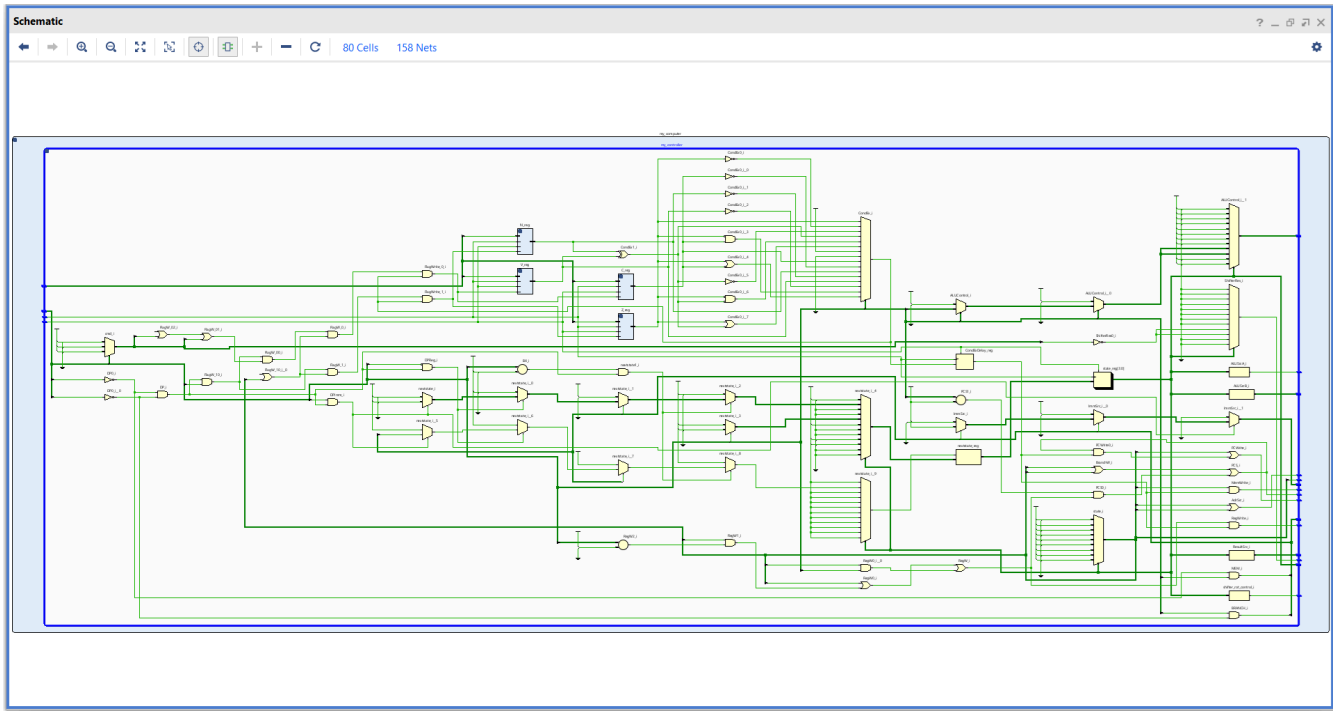

Figure 2: RTL view of the Controller

Testbench Results

For this assignments, we were given a testbench with debugging helpers. The debugging helpers are used during the finalization of the design and the resulting design is able to pass the testbench. The result of the testbench with a **Pass** flag can be seen in Figure 3 below.

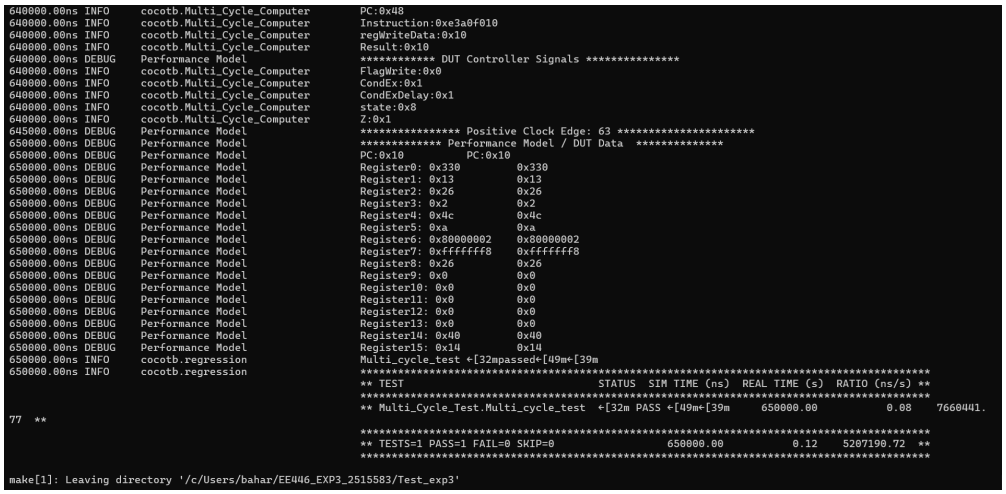

Figure 3: Testbench Results

Conclusion

In this design, the design process of Multicycle Processor is examined. A datapth design approach similar to the **Single Cycle Processor** we have tackled in the previous experiment was sufficient however the controller design has changed significantly to achieve a more complex functionality. The creation of new paths for specific instructions in a new architecture was rewarding. This tasks has allowed me gain insight to architectural designs that are more present in the daily life in more detail.