ECE 437L Midterm Report

ECE 437L

Tessie McInerney
Pranjit Kalita

TA: Chuan Yean Tan

March 4, 2016

1 Overview

In this midterm report, we will be comparing our single cycle design, implemented in labs 3-4, and our pipelined design, implemented in labs 5-7. For this analysis, we compared the clock frequencies, instruction latency, performance in MIPS, and the FPGA resources utilized by each design. The data on these comparisons come from running mergesort.asm on both processors. We selected this program due to its abundance of hazards, which will give a good depiction of how our forwarding and branch prediction logic affect the data.

Both single cycle and pipeline processors have their share of advantages and disadvantages. The single cycle processor, while it executes one instruction per clock cycle, has a very long execution time. However, because it only executes one instruction per clock cycle, it is not subject to any hazards. This makes the design much simpler than the pipelined design. That being said, the pipelined processor increases the throughput of the overall program as well as increases the clock speed. However, there is a higher possibility that the pipelined design can incur an increase in CPI due to the insertion of bubbles or stalls when dealing with hazards. Because of those hazards, the pipelined design may not always have a higher performance in MIPS, as one would initially assume. With the introduction of forwarding and branch prediction, however, you can reduce the CPI and increase the performance.

As you will see in our data below, the values for all comparisons increase with our pipeline design. While the frequency and CPI were expected to have been higher than the single cycle design, an increase in performance would not typically be expected. However, with the implementation of forwarding and branch prediction, it is probable that performance would increase.

2 Processor Design

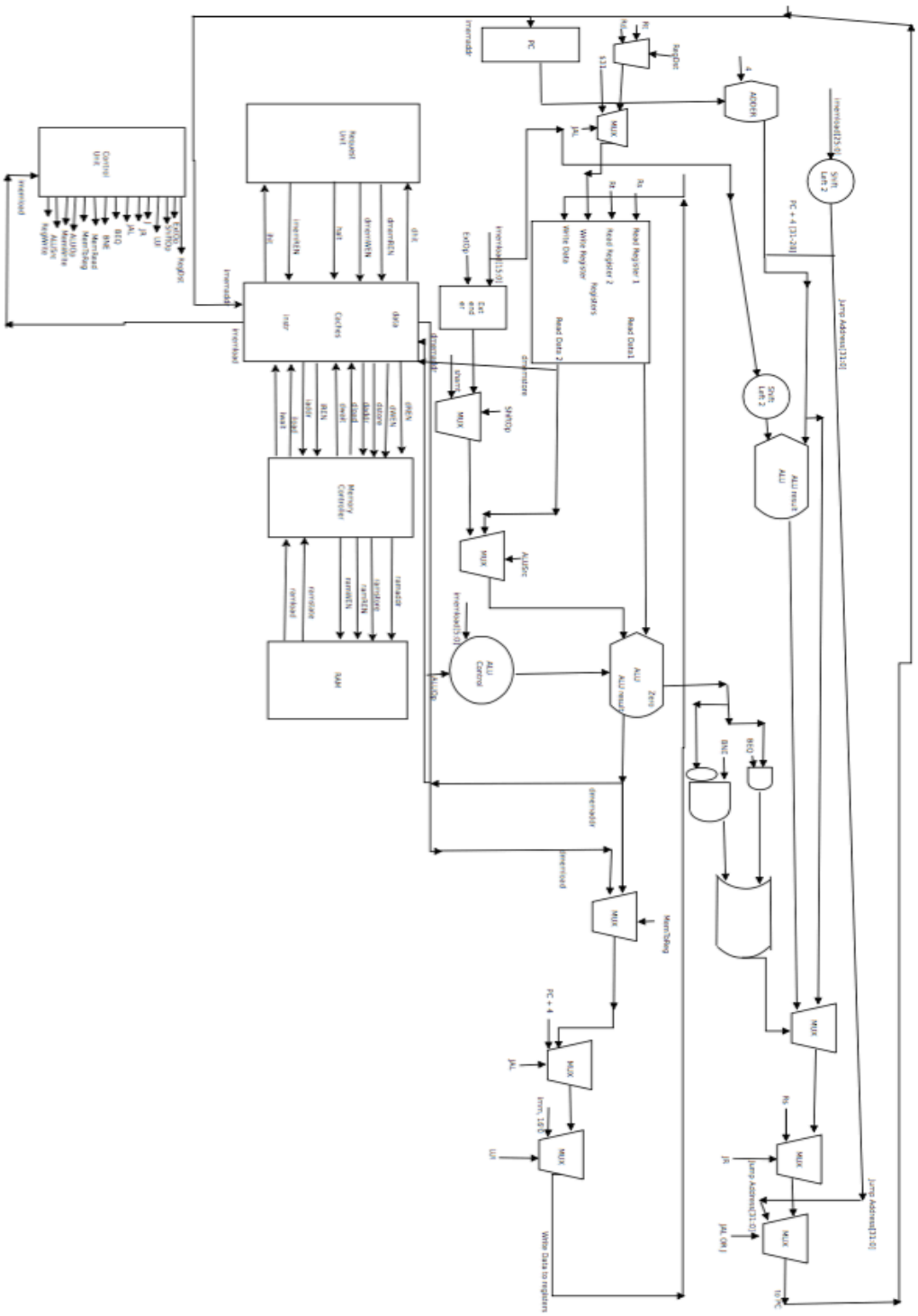

Figure 1 – Single Cycle Design

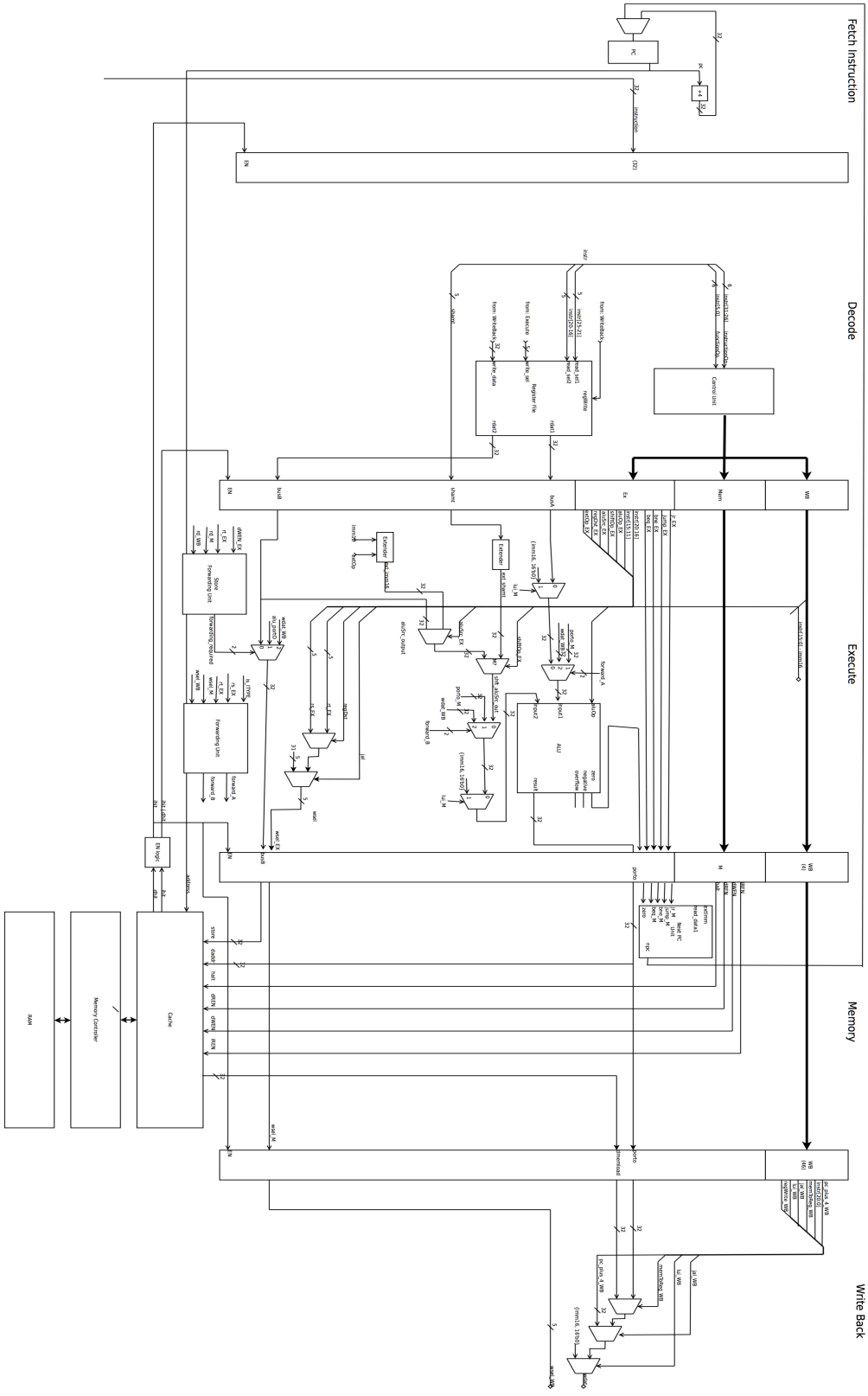

Figure 2 – Pipelined Design

3 Results

	Single Cycle	**Pipeline**
Frequency	39.04 MHz	64.56 MHz
Mergesort Instructions	5399	5399
Cycles	55167	75673
Average CPI (cycles/instruction)	10.22	14.01
Latency (5 * period) for pipeline	25.61 ns	77.45 ns
Performance (MIPS) (Freq / CPI * 10^6)	3.82	4.611
FPGA Resources	Total combinational: 2783 Logic registers: 1283	Total combinational: 3627 Logic registers: 1976

4 Conclusions

As expected with the pipelined implementation, the average CPI increased, primarily due to the numerous branch and jump conditions, as well as the prevalence of hazards, in the mergesort file. Therefore, the introduction of bubbles and stalls would account for this increase in CPI compared to its single cycle counterpart. In addition, we see an increase in clock speed, shown in the table above by the maximum frequency arrived at for the pipelined processor. The increase in clock speed was the primary reason for selecting the pipeline processor over the single cycle as an improved processor design.

While the clock speed did not increase by 5 times compared to the single cycle, as is suggested by the ideal case, we were still able to arrive at an increased performance. This could very well be attributed to our addition of branch prediction in the ID stage, which significantly reduced the number of bubbles and stalls in our design. Furthermore, the increase in latency could be attributed to the less-than-expected increase in clock speed, as well as the additional hardware-induced delays such as those due to latches and muxes. With all this being said, we can see the usefulness of using pipeline over single cycle through the increase in clock speed and MIPS performance rating.

5 Contributions

Pranjit's single cycle processor design was used as the comparison in this report, while both Pranjit and Tessie created the pipeline processor. Throughout the entirety of the design of the pipeline processor, both Pranjit and Tessie contributed equal amounts. Both contributed to the initial design of the pipeline design as well as the forwarding and branch prediction logic additions. Furthermore, both were present throughout the debugging process, and worked together to fix the issues that arose in the design.