A Wild Processor Appeared

ECE 437

Cole Reinhold & Samodya Abeysiriwardane

TA: Adam Hendrickson

Due: 10/17/14

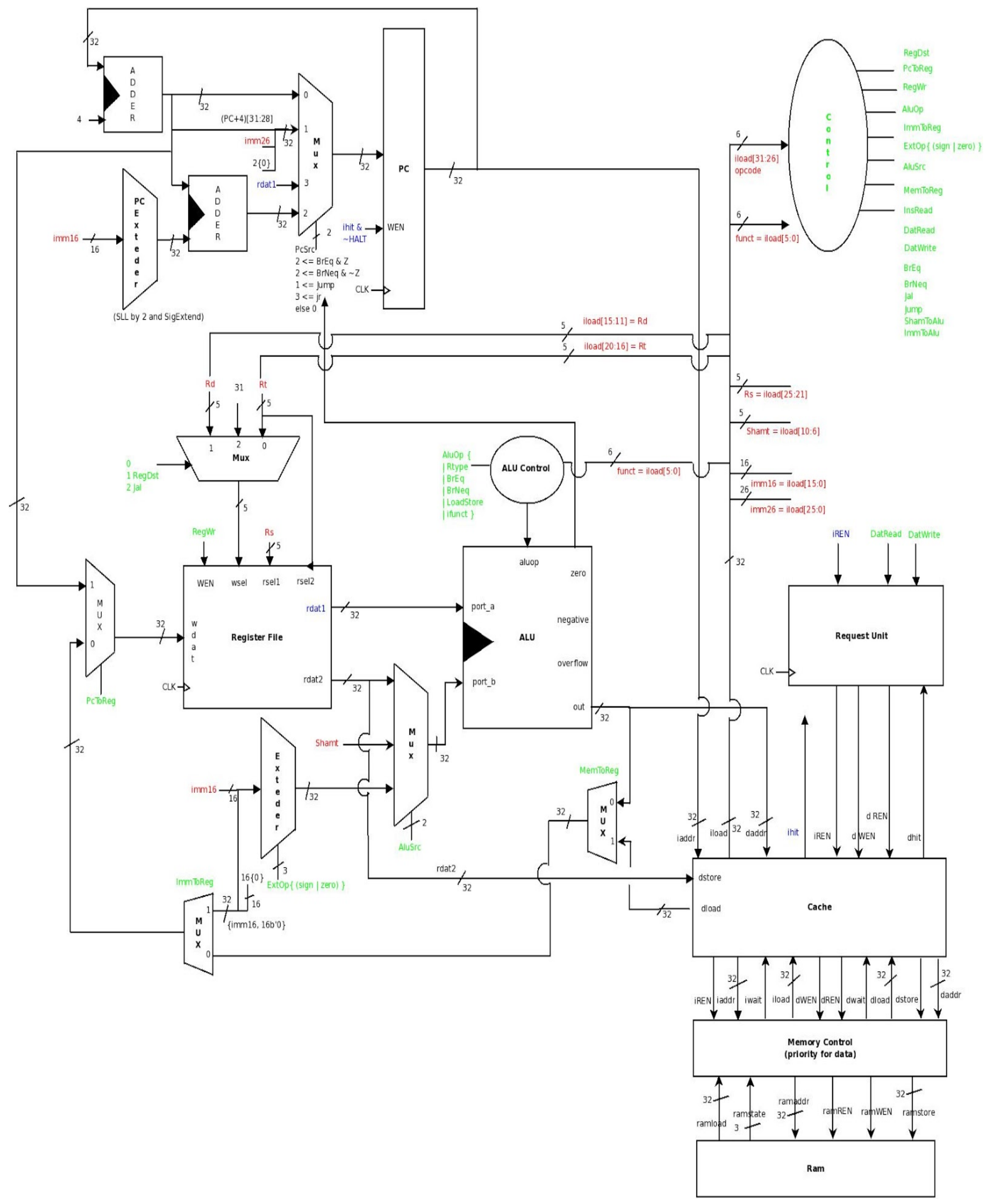
Overview

This report focuses on the comparison of two processor designs, a single cycle and a pipeline processor. For reference, block diagrams of the two designs can be found in the processor design section. In the results section, performance metrics are discussed after running a mergesort program on both processors. The single cycle design benefits include a smaller latency value and CPI. This is due to the single cycle being able to complete most instructions in one cycle. Although the pipeline design has a faster clock, it takes five cycles instead one one to push a single instruction through the pipeline, resulting in higher latency and CPI. Pipeline benefits from a faster CPU clock, higher MIPS, and lower processing time. The pipeline is able to process the same number of instructions in less time due to the overlapping.

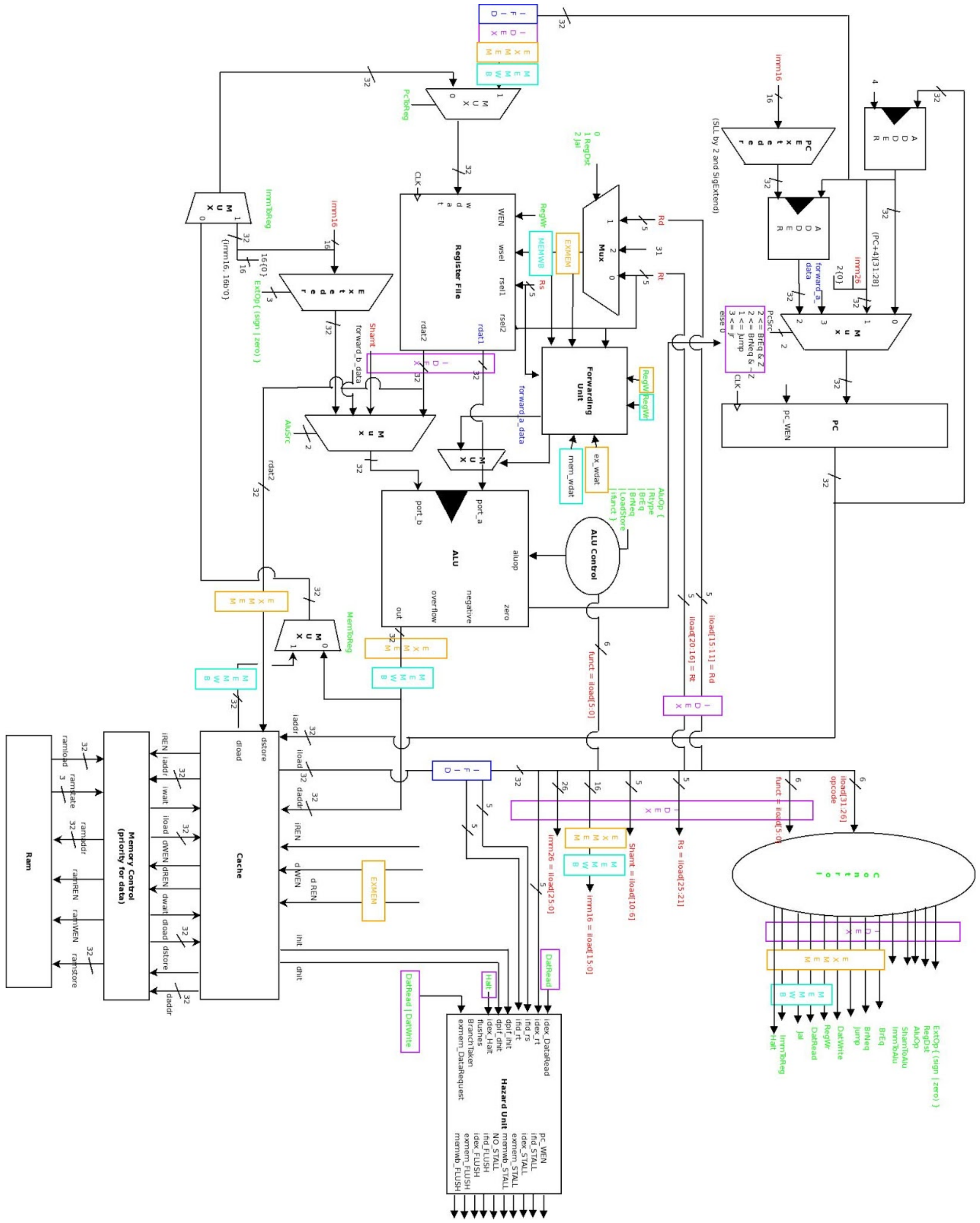
We thought that mergesort was a great program to test the functionality of our designs. It is made up of 17 different instructions for a total of 5394 instructions. This large scale test will take care of many corner cases and stress the design. Single cycle did have an advantage over pipeline while using this program however. Single cycle is able to complete about 75% of the instructions without a stall while pipeline will be able to complete about 50% of instructions without causing a stall or flush. After comparing the results and performance data from our tests, we have concluded that the pipeline design outperforms the single cycle and is our design of choice.

Processor Design

Single-cycle Diagram



<u>Pipeline Diagram</u>



Results

Pipeline

Frequency Target (Mhz)	50	75	100
Max Frequency (Mhz)	62.77	75.27	80.4
Passed CPU CLK (Mhz)	50	41.66	41.666
CPI	1.441694475	1.44160178	1.44160178
MIPS Theoretical	43.53904456	52.21275463	55.7712963
MIPS Actual	34.68141195	28.89841049	28.90257253
Latency (ns)	100	120.0192031	120.00192
Time (ns)	155540	186636	186636
Total logic elements	3,388 / 114,480 (3 %)	3,434 / 114,480 (3 %)	3,421 / 114,480 (3 %)
Total registers	1620	1620	1620
Total pins	102 / 529 (19 %)	102 / 529 (19 %)	102 / 529 (19 %)

Single cycle

Frequency Target (Mhz)	50	75	100
Max Frequency (Mhz)	19.5	20.64	20.67
Passed CPU CLK (Mhz)	11.11	10.6383	10.4167
CPI	1.277437894	1.277345198	1.277345198
MIPS Theoretical	15.26492998	16.15851379	16.182
MIPS Actual	8.697096002	8.328445602	8.154960784
Latency (ns)	90.0090009	93.9999812	95.9996928
Time (ns)	606408	661536	661536
Total logic elements	2,826 / 114,480 (2 %)	2,841 / 114,480 (2 %)	2,866 / 114,480 (2 %)
Total registers	1280	1280	1280
Total pins	102 / 529 (19 %)	103 / 529 (19 %)	104 / 529 (19 %)

Our highest performance pipeline design came from synthesizing at a target frequency of 50 Mhz. We obtained a MIPS of 34 compared to 8.6 for single cycle, about a 4 times improvement. It seems that the Quartus synthesize tool that created the theoretical max frequency was overgenerous in its calculations as we were unable to obtain these max frequencies. We increased the clock frequency until the output did not match the expected output. This is how we found the max passed clock frequency.

Conclusion

Although the single cycle processor had a lower latency and lower CPI during tests, the pipeline design had a much higher performance. Our designs were synthesized at target frequencies of 50, 75, and 100 Mhz. As the frequency increased, so did the theoretical max frequency our design could run at. We were able to get a max theoretical frequency of 80.4 Mhz on our pipeline processor and 20.67 Mhz on the single cycle processor. Our actual max frequency for pipeline was 50 Mhz and single cycle was 11.11 Mhz. These results are displayed in the table above.

Execution time is used to measure the overall performance improvement of the two designs. Due to the number of instructions remaining constant, MIPS is also a valid measurement of performance. Overall we saw about a 4 times improvement in performance when running mergesort on a pipeline processor over the single cycle. The pipeline used more of the FPGA resources. This was expected as the pipeline design adds complexity.