Midterm Report

ECE 437 Processor Prototyping Laboratory

By: Jhen-Ruei Chen, Yue Yu

Section 3

TA: Abhishek Bhaumick

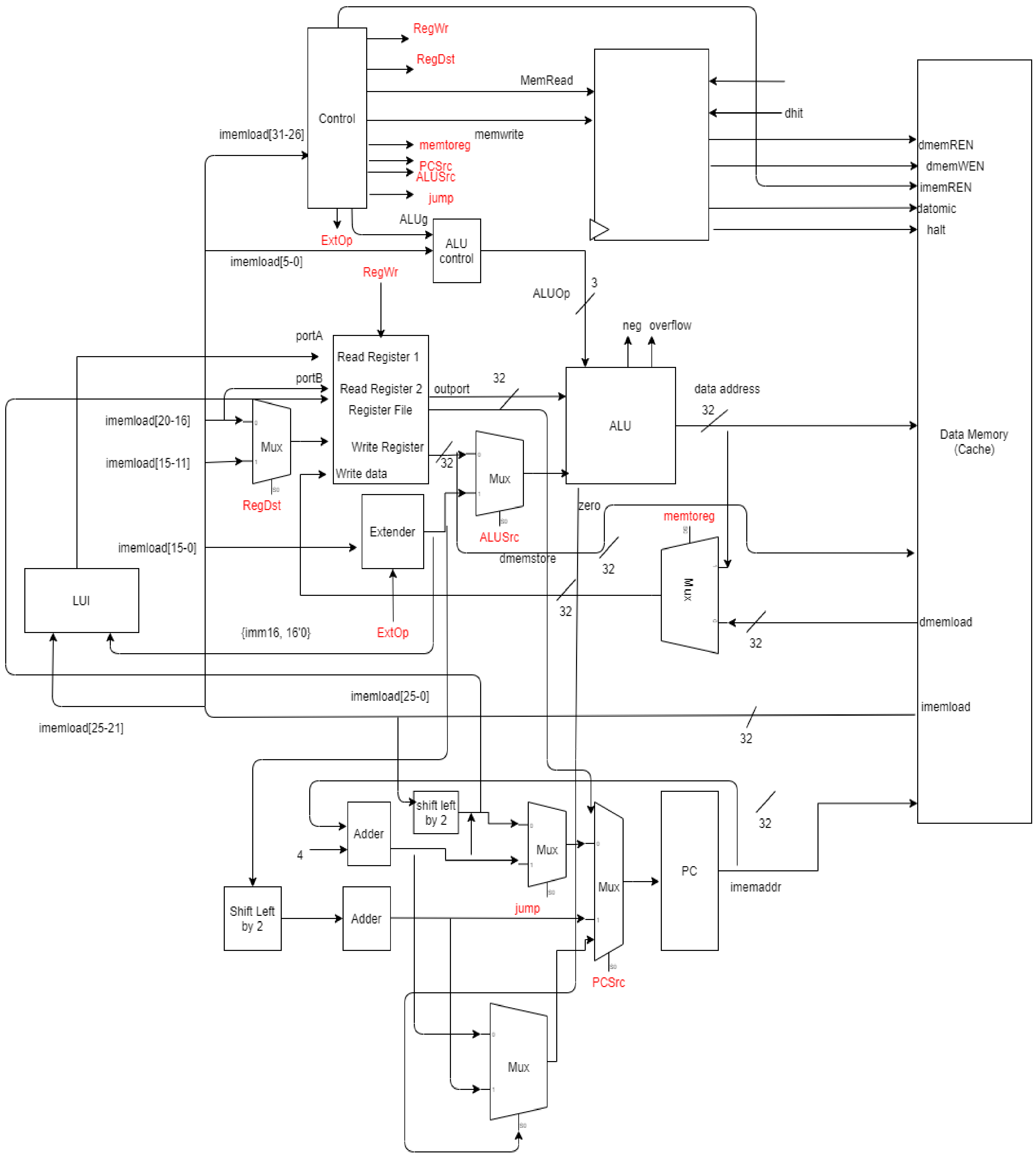
10/17/2021

Design Overview

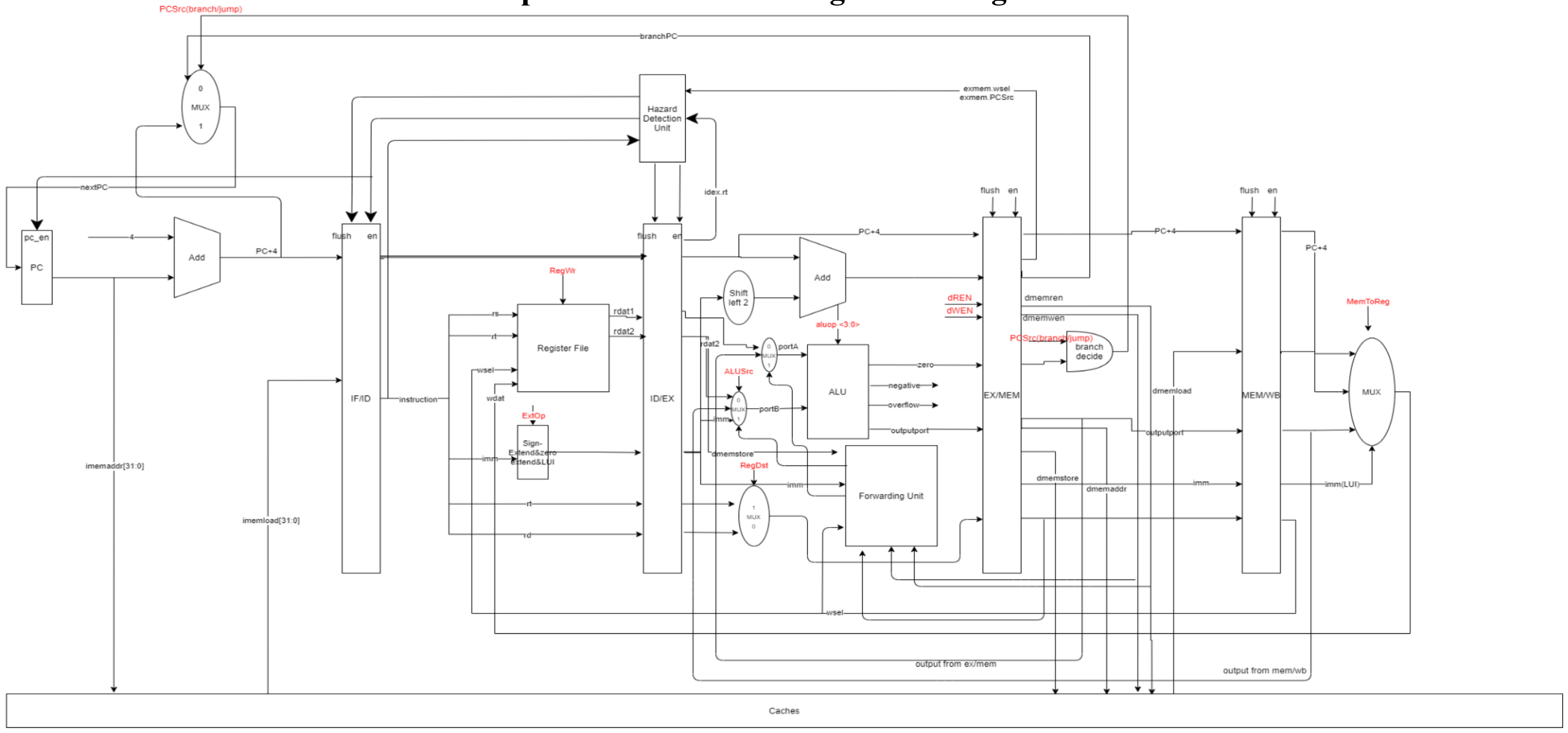
Throughout this report, we analyze and compare the performance between our single-cycle processor and pipelined processor. We will run both processors on MIPS instructions. Due to the increased throughput, the pipelined processor will run faster than the single-cycle processor. In this report, we will run both processors under the mergesort program. Mergesort program contains almost all the MIPS instruction that might be used. Except for this, one of the benefits that the mergesort program has is that it can create hazards that will only encounter in pipeline processors. Although the pipelined processor is faster than the single-cycle processor, the pipelined processor will create different hazards and it will require to implement forward unit and hazard unit so that we do not need to put any no operation in any asm files.

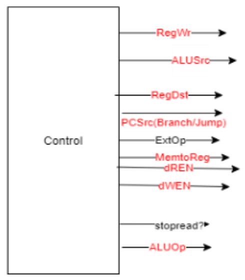
In order to compare the performance between both processors, we will check on the maximum clock frequency, the average instructions per clock cycle, the latency of one instruction, the performance of the designs in MIPS, and the FPGA resources. In conclusion, the pipelined processor has a better performance and better maximum clock frequency compare to the single-cycle processor. However, on the other hand, a single-cycle processor has a lower latency and fewer FPGA resources needed.

Single-Cycle Processor Design RTL Diagram



Pipelined Processor Design RTL Diagram





Results

Throughout this lab, we will run the mapped version by applying the command "synthesize -t -f 200 system" to enable timing and to set an effort of 200 MHz for Quartus. The program "mergesort.asm" will be used to collect performance data. Both processors will be running under latency = 0.

Formulas to Calculate different parameters

1. **Max Frequency:** From "system.log" and Fmax = minimum of CLK/2 and CPUCLK
2. **Number of instructions:** From running "sim -t"
3. **Number of cycles:** From running "system.sim"
4. **Latency:** a) Single-cycle: 1 / Fmax

   b) Pipelined: 5 / Fmax
5. **Performance:** clk period * number of cycles

Result Table:

	Single-Cycle Processor	**Pipelined Processor**	**Units**
Max Frequency	**36.71 (CLK)**	**53.37 (CPUCLK)**	**MHZ**
Average Instructions per clock cycle	**5404 instructions / 13801 cycles = 0.3916**	**5404 instructions / 17599 cycles = 0.3071**	**Instructions / Cycles**
Latency of one instruction	**27.24**	**93.69**	**Nanosecond (ns)**
Performance	**376.1**	**330**	**Microsecond (µs)**
FPGA resources	**Total Logic Elements: 3,133 / 114,480 (3%) Total Registers: 1280 / 117053 (1%)**	**Total Logic Elements: 3790 / 114480 (3%) Total Registers: 1820 / 117053 (2%)**	

Conclusion

According to the result we got from both processors, we can conclude that the overall performance of the pipelined processor is better than the single-cycle processor. Our Max Frequency for pipelined is greater than the single-cycled and this is expected because pipelined should have a faster execution speed compared to single-cycled. The runtime for pipelined is 46.1 microseconds which are about 14% less than the single-cycle processor.

However, there are also downsides to using the pipelined processor. Based on the data we had, pipelined processors will have a larger latency compared to the single-cycle processor with about 3.44 times larger. Besides the larger latency, the total FPGA resources required for pipelined is also larger compared to a single cycle with about 540 more registers to be used. This is also expected because in pipelined processor we not only need to hold the register values but also need to forward our output or detect hazards which requires two more new modules. As a result, the pipelined processor will need a larger latency and FPGA resources compared to single-cycle processors. Therefore, both single-cycle processors and pipelined processors satisfied the design expectations and criteria successfully.

Contribution

- Jhen-Ruei Chen: Single cycle design, Forward unit design, Unit test writing, Hazard unit testbench, Report data collecting, Report writing
- Yue Yu: Single cycle design, Hazard unit design, Forward unit testbench, Pipeline datapath design, Report Data Collecting, Report Writing