# Mid Term Report

ECE437

Abdullah Khan, Neil Gupta

Nick Pfister

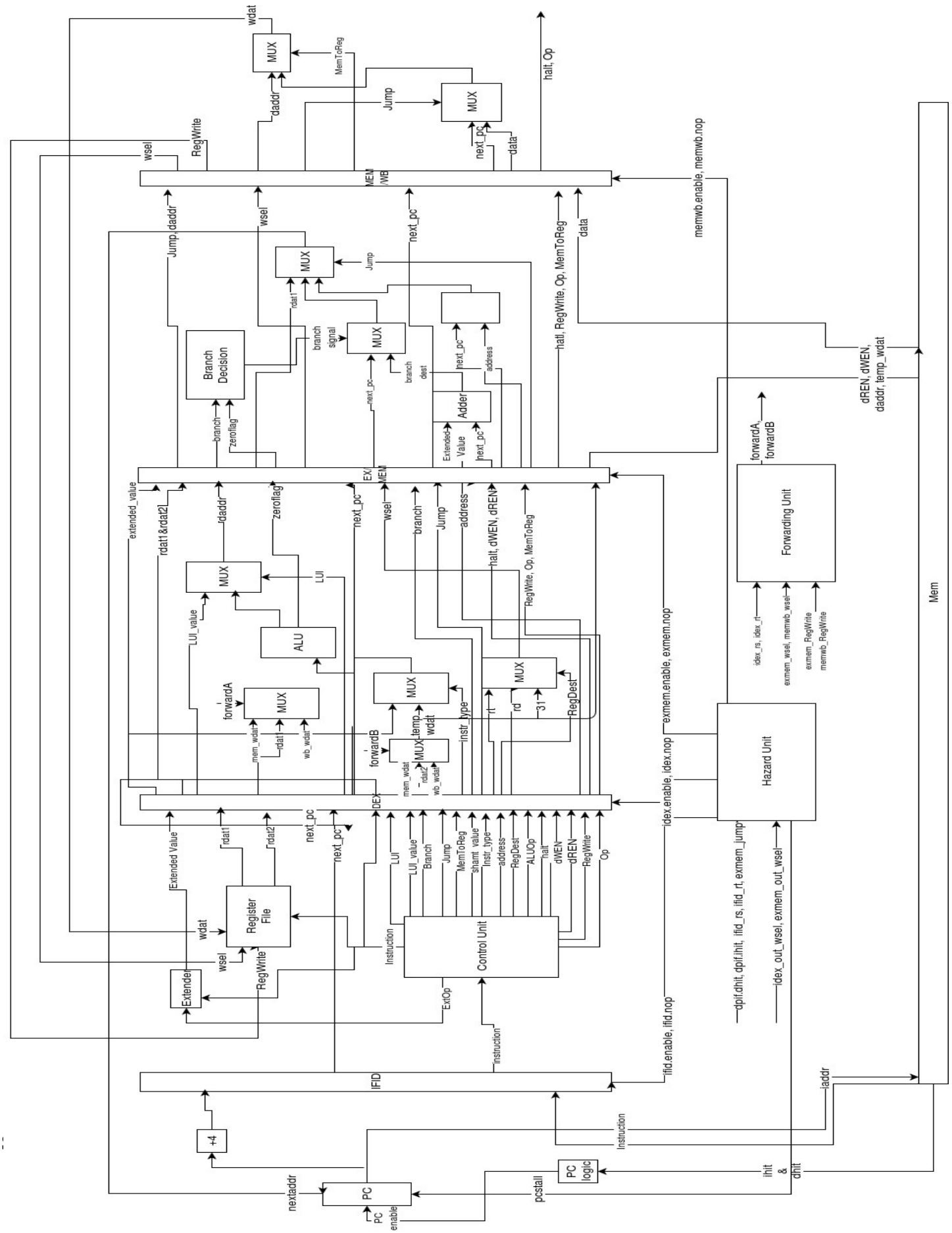
03/03/2017

**Overview**

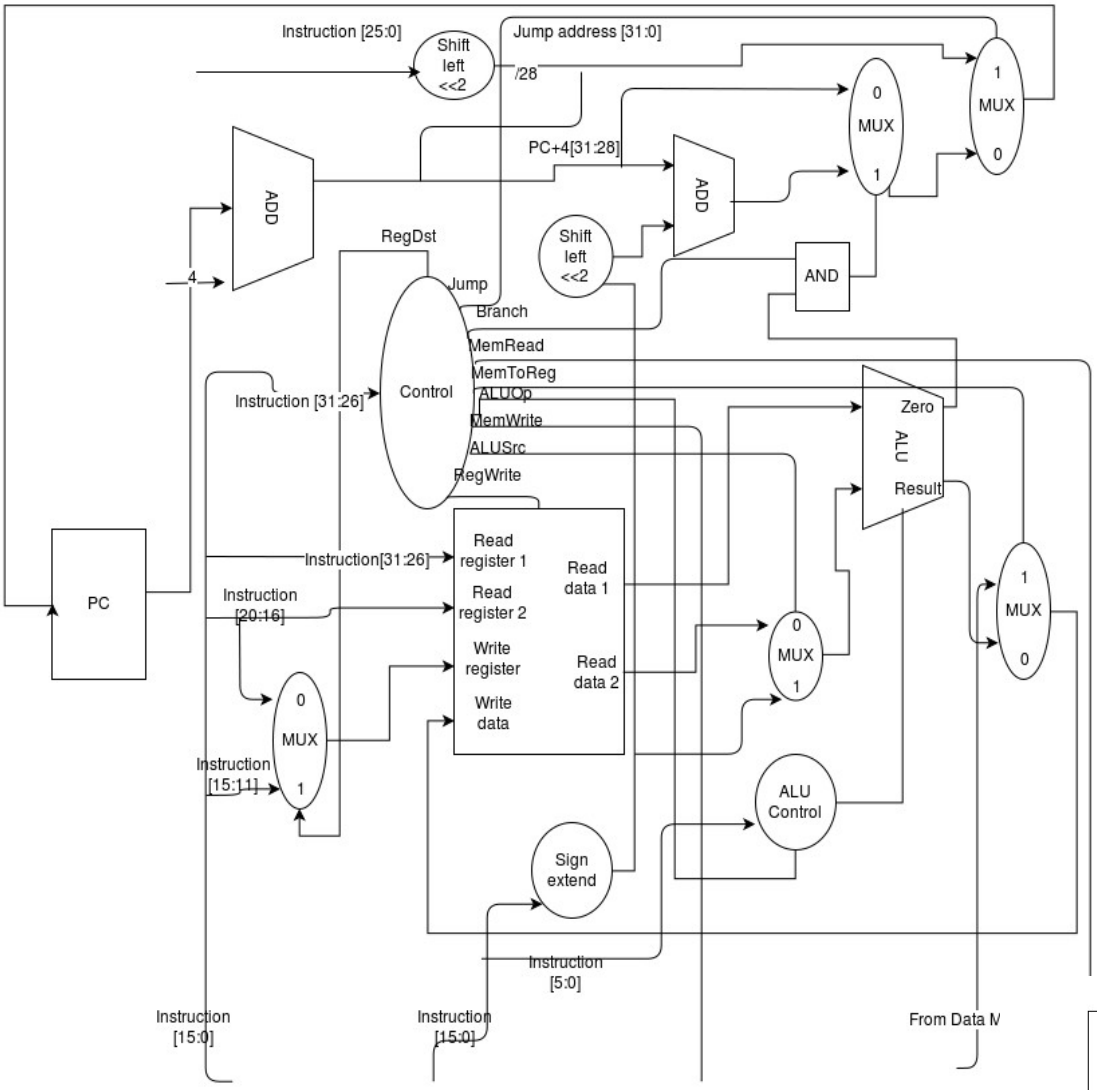
Over the course of this class we have designed, developed and implemented a Single cycle design and a Pipeline design. Some advantages of the Single cycle design are that it is simple to design and since there is less combinational logic it is easier to implement however compared to Pipeline design it lacks in efficiency and performance. This is because in Single cycle every instruction regardless of how complex it is will require the same amount of time, due to this the processors clock is dependent on how slow the longest instruction takes. In comparison, with a pipeline processor you have a higher throughput since you can queue instructions and the critical path is lower since it is broken up into several stages. Although, the the latency of each instruction suffers slightly the added performance makes up for it since overall the program executes faster. The disadvantage of the pipelined processor is that its much more complicated to design and requires far more combinational logic to run smoothly without hazards and requires more FPGA resources.

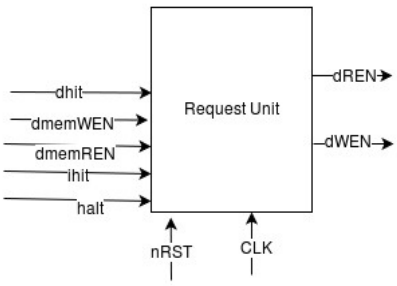
Moving on, to benchmark the performance of the two processors we have selected to run mergesort.asm as it will cover most metrics a processor should meet. There are a total of 5399 instructions and it tests plenty of I type, J type and R type instructions. Furthermore, mergesort.asm introduces many hazards, some which have to be stalled and some which require forwarding, so it properly tests the functionality for the hazard and forwarding unit in the pipelined processor while being complex enough for the single cycle design.

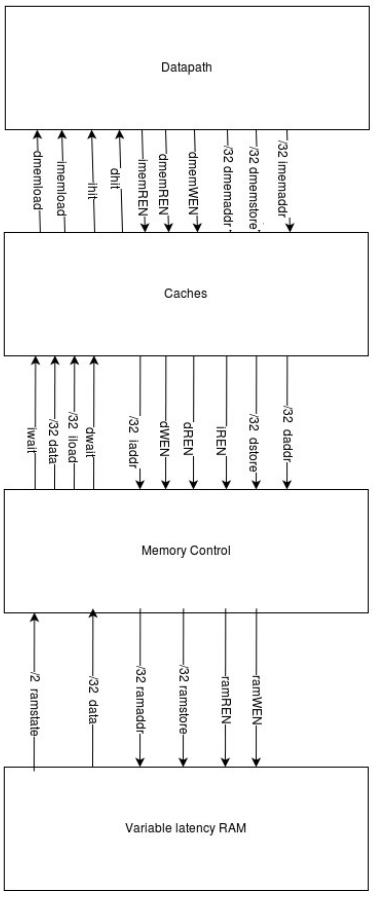
**Processor Design - Pipeline**



## Processor Design – Single Cycle



**Results**

| | Single Cycle | Pipeline |
|---|---|---|
| CLK Frequency | 29.55 MHz | 54.95 MHz |
| Length of critical path | 33.84 ns | 18.19 ns |
| Latency | 33.84 ns | 90.95 ns |
| Execution time | 13791 cycles * 33.84 ns | 17587 cycles * 33.84 ns |
| Maximum Achieved Frequency | 1/10 GHz | 1/2 GHz |
| Breaking parameter period | 10 ns | 2 ns |
| Total Combinational functions | 3,107 | 3,224 |
| Total Registers | 1279 | 1718 |
| Average Instructions/clock cycle | 5399 instructions / 33.84 ns = 159.54 x 10^9 instructions/cycle | 5399 instructions / 18.19 ns = 296.81 x 10^9 instructions/cycle |

**Conclusion**

In conclusion, analyzing the results we can see that the pipelined processor has a higher clock frequency this is because the shorter critical path allows us to run the processor at higher frequency as compared to Single Cycle. The shorter critical path is achieved in pipeline due to the design being split up in 5 parts as discussed in the overview. The latency increases for each instruction in the pipelined processor since it takes longer for each instruction to go through the design.

Moving on, the total execution time for our single cycle design was faster than the pipeline but this might be due to the added NOP's and stalls by the hazard unit since in theory the pipeline should be faster. Finally, the pipeline processor requires more FPGA resources since there is more logic with the pipes, hazard unit and forwarding unit that needs to be implemented.

**Contributions**

**Abdullah Khan**

- Prepared Block Diagram
- Designed and wrote pipelines
- Wrote datapath
- Debugged datapath
- Wrote code in forwarding unit
- Contributed to logic in writing code hazard unit
- Debugged forwarding unit
- Debugged hazard unit
- Worked on report

**Neil Gupta**

- Contributed to design and tinkering of final block diagram used
- Contributed to debugging pipeline code written
- Wrote hazard unit
- Contributed to logic and writing code in forwarding unit
- Contributed to logic in writing code hazard unit
- Debugged hazard unit
- Debugged forwarding unit
- Worked on the report