Processor Prototyping Lab

ECE 437

Midterm Report

Abhijay Achukola

Geetha Prasuna Yarramneni

Lab section 2

Zhaoyu Jin

13th October 2024

1 Executive Overview

In the first four weeks of this project, work was put forward into developing a RISC-V single-cycle processor. In this processor, every instruction is completed in one long clock cycle. While the single-cycle architecture is straightforward to implement, it suffers from performance inefficiencies due to its inability to overlap operations. To enhance performance, the processor was transitioned to a pipelined architecture, which splits instruction execution into multiple stages that can be processed in parallel. The decision was made to opt for a five stage pipeline, with the stages being instruction fetch, instruction decode, execute, memory access, and register write back. While this improved throughput by enabling parallel execution, it also introduced new challenges, such as data hazards (e.g., Read After Write) and control hazards (e.g., branches). Addressing these issues required the addition of hardware like a forwarding unit and a simple "always-not-taken" branch predictor.

In theory, an idealized five stage pipelined processor should speed up execution by a factor of five. However, this ignores issues such as the stages being unbalanced in how much work each one does, having to handle hazards with things such as data hazards and control hazards as mentioned previously, and the overhead of creating pipeline latches in between each stage. Due to all of these features slowing down the processor compared to an idealized processor, the speed-up that will be calculated when comparing the single cycle design and pipelined design will not be a factor of five. Regardless, the pipelined processor should still provide a noticeable improvement. Both designs were evaluted using the merge sort algorithm, chosen for its diverse instruction mix, including jumps, branches, and R-type operations. The pipelined design outperformed the single-cycle design on several metrics across all ram latency values tested: clock frequency, cycles per instruction, and overall execution time. However, latency for a single instruction did increase due to the increased complexity of the processor.

In conclusion, while the single-cycle processor was easier to design, the pipelined processor was more efficient in terms of both speed and throughput, making it a clear winner in performance-critical applications.

2 Processor Design

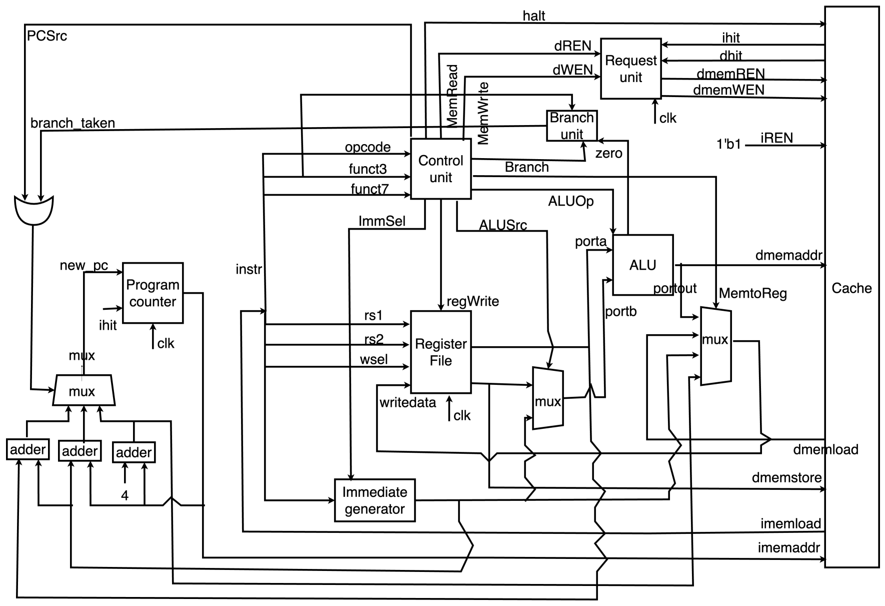

Figure 1: Datapath of Single Cycle Processor

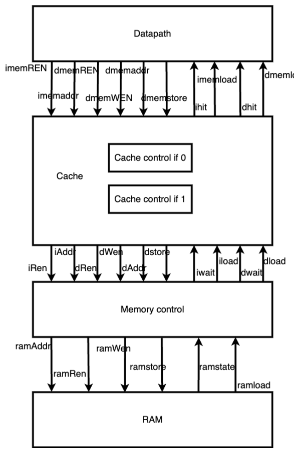

Figure 2: CPU Block diagram

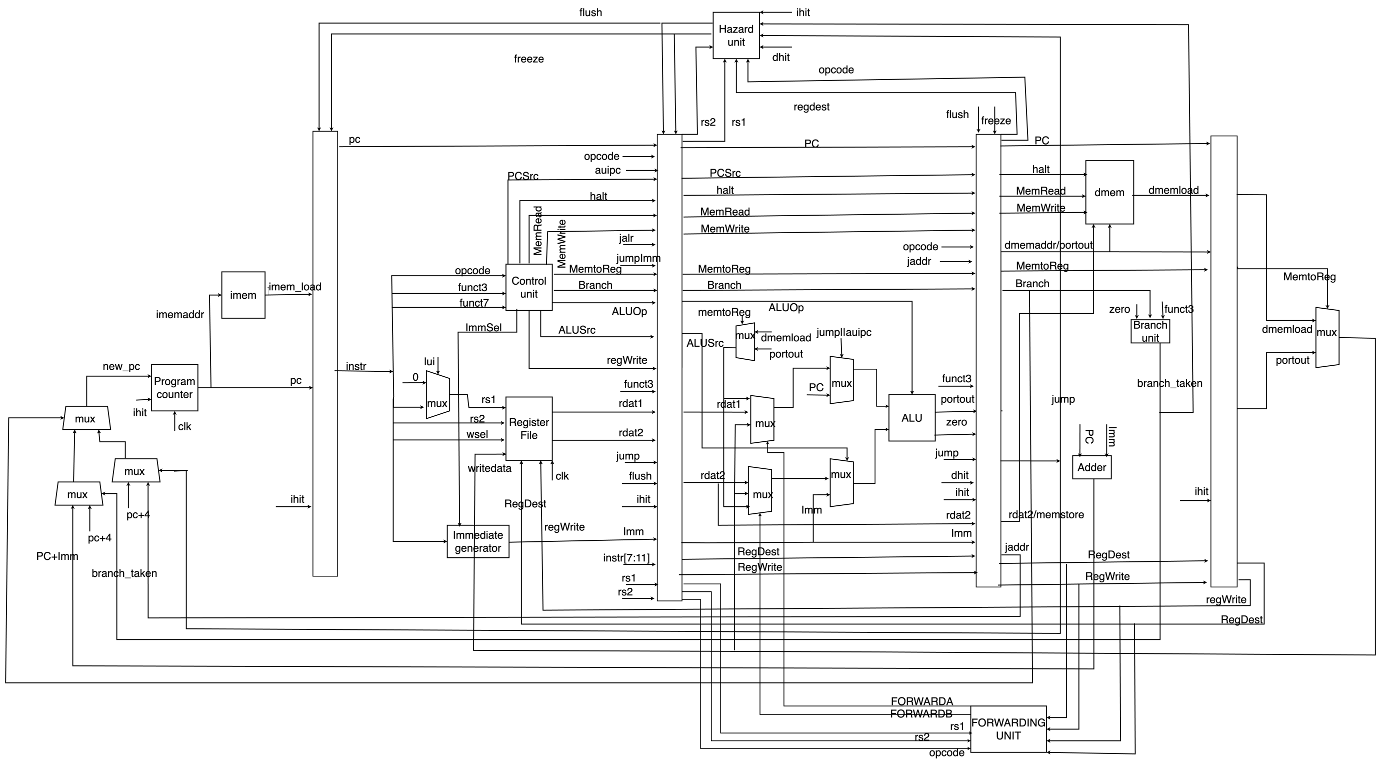

Figure 3: Datapath of Pipelined Processor

3 Results

Testing both processor designs on the merge sort algorithm at various RAM latencies produced the max frequency of the CPU clock (F_{max}) and the number of cycles taken for each scenario within a log file. These values are depicted in table 1 below. The number of cycles taken in the log file represents the number of cycles of the testbench, so the values were divided by two before being used for other calculations, as the testbench clock is twice as fast as the CPU clock. The number of CPU Cycles taken is the value depicted in the table below. The FPGA resources needed for each design was also shown in the table below. Taking into account that the instruction count of the merge sort algorithm the processor was tested on is 5409, we can calculate other important values to judge processor performance. To calculate CPI we used the following equation:

$$CPI = \frac{Total\ Number\ of\ CPU\ Cycles}{Total\ Number\ of\ Instructions}$$

Since the CPI, clock speed, and the number of instructions per program is known, the time per program can be calculated using the iron law:

$$Execution\ time\ per\ program = \frac{CPI * Number\ of\ Instructions\ per\ Program}{CPU\ Clock\ Speed}$$

From total execution time, the average latency of each instruction can be calculated using this simple formula:

$$Average\ Latency = Number\ of\ Stages * \frac{Exectuion\ time\ per\ program}{Number\ of\ Instructions\ per\ Program}$$

As mentioned before number of instructions per program will always be 5409 and the total number of CPU cycles and the CPU clock speed for each processor and RAM latency can be found in the table. The number of stages will be one for single cycle and five for pipelined, as there are five stages in the pipelined processor. The calculated values can be seen below.

Metric	**Single Cycle**	**Pipelined**
FPGA Resources Utilized		
Combinational Functions	2875	3297
Registers	1287	1768
RAM Latency = 0		
CPU Clock (MHz)	30.68	63.37
CPU Cycles Taken	6906.5	9480.5
CPI	1.28	1.75
Latency (ns)	41.62	138.29
Milliseconds/Program	0.2251	0.1496
Speedup	-	1.505
RAM Latency = 2		
CPU Clock (MHz)	29.10	62.97
CPU Cycles Taken	13814.5	18897.5
CPI	2.55	3.49
Latency (ns)	87.76	277.41
Milliseconds/Program	0.4747	0.3001
Speedup	-	1.582
RAM Latency = 6		
CPU Clock (MHz)	29.10	62.97
CPU Cycles Taken	27628.5	37729.5
CPI	5.11	6.98
Latency (ns)	175.53	553.86
Milliseconds/Program	0.9494	0.5992
Speedup	-	1.585
RAM Latency = 10		
CPU Clock (MHz)	29.10	62.97
CPU Cycles Taken	41442.5	56561.5
CPI	7.66	10.46
Latency (ns)	263.29	830.31
Milliseconds/Program	1.4241	0.8982
Speedup	-	1.585

Table 1: Comparison of Single Cycle and Pipelined Processors

4 Conclusion

Given the metrics calculated, the pipelined processor's performance was superior to the single cycle processor's performance. Across all RAM latency values, the pipelined processor was over 1.5 times faster. Clock speed doubled even though CPI went up from the single cycle processor to the pipelined processor. Latency per instruction increased significantly while transitioning to a pipelined model, as was expected. In exchange for the increased latency, the pipeline model delivered by increasing throughput with a faster execution speed. In terms of resources, both processors used similar amounts of registers and combinational functions, but the additional overhead that is involved in creating a pipelined processor made the pipelined processor take more FPGA resources.

In conclusion, the overlap in stages that is found within the pipelined processor provided a significant improvement in processor speed by massively increasing throughput in exchange for more FPGA resources and higher latency. The pipelined model allowed for a clock that is over twice as fast as the single cycle model which displays the massive increase in througput. Overall, this project demonstrated why single cycle processors are not produced in any major way in the modern day, as pipelined processors are superior for almost any application.

5 Contributions

The original single cycle processor design that was built upon for the pipelined design was created by Geetha Prasuna Yarramneni. Geetha created the hazard unit and forwarding unit test bench, while Abhijay Achukola made the forwarding unit and the hazard unit testbench. Abhijay created the pipelining latches and Geetha helped connect them to the datapath. Both Abhijay and Geetha helped debug the design till it was complete passing all tests.