Midterm Report

ECE 437: Computer Design and Prototyping

Karim Itani, Milind Shyam

Lab Section 3

March 4, 2018

Overview

This report compares the performance of singlecycle and pipeline processors. Singlecycle processors are easier to implement and are not as complex as pipeline processors. The entire process is completed in just one long clock cycle. Therefore, performance is compromised due to this. Pipeline processor on the other hand is much more complex and is harder to implement. But the performance of pipeline processors is much better than singlecycle processors. This is mainly due to the presence of 5 intermediate pipeline stages that increases throughput. So more tasks can be completed in a given time. But this also increases the chances of getting incorrect data due to various hazards. These hazards must be taken into account and prevented.

We use mergesort algorithm to compare the performance of the two processors as it has a lot of jumps, branches and other types of instructions that are good for comparing the two processors. It is also quite lengthy and complex due to its looping and is therefore good for comparing the two. The report compares the frequency of the two designs, the average instructions per clock cycle, the latencies, performance in MIPS and the FPGA resources utilised. After comparing the data for the two processors, we concluded that overall the pipeline processor is better than the singlecycle processor as it is more efficient.

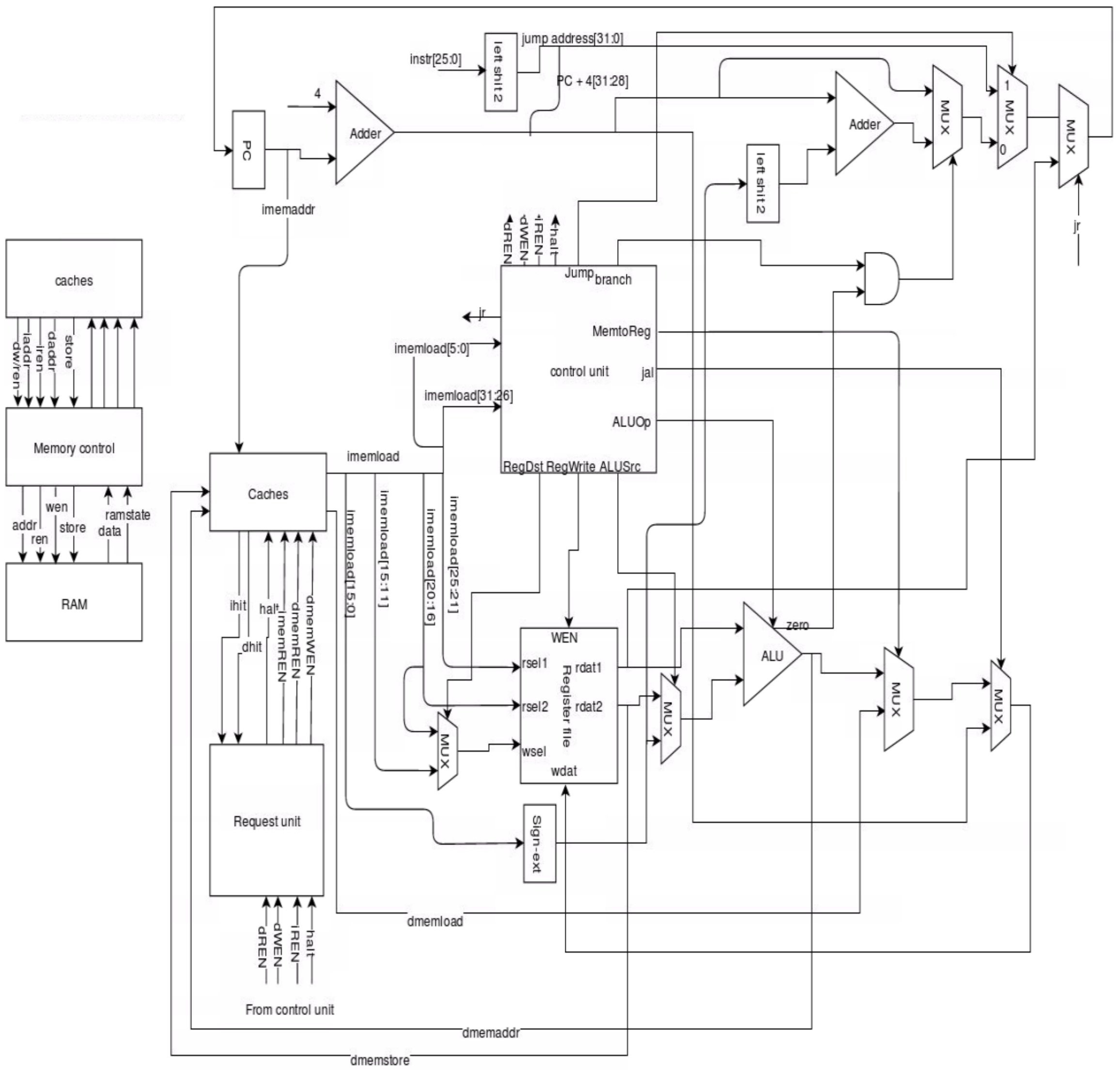

Figure 1: Single Cycle Schematic

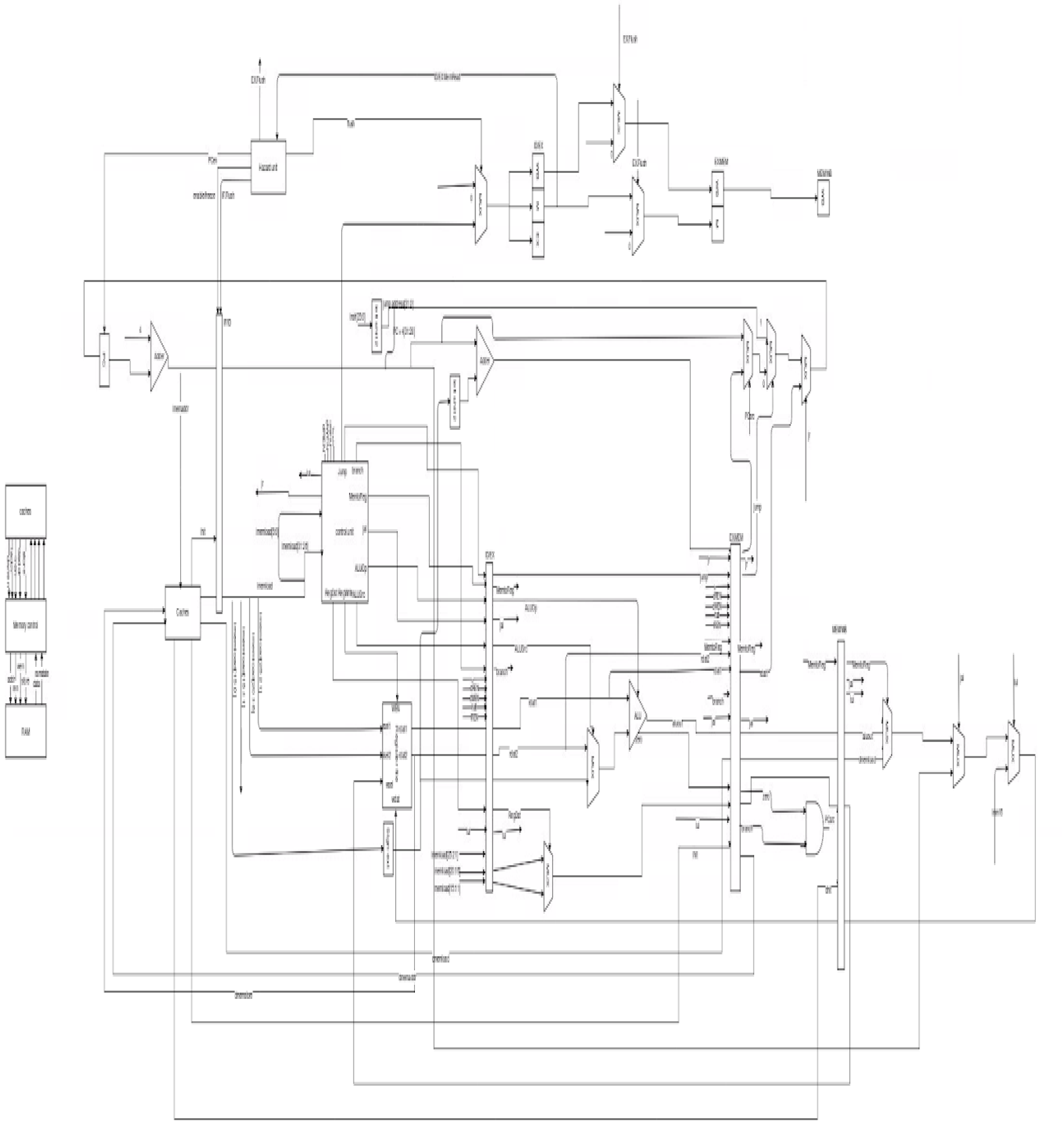

Figure 2: Pipeline Processor Schematic

Results

	Single Cycle	Pipeline	Units
Max Clock Frequency (logs)	35.8	43.09	MHz
IPC	0.126	0.325	instr/cycle
MIPS	6.883	15.12	Millions of instr /sec
Latency of Instructions	27.9323	116.036	ns
Length of Critical Path	25.23	22.34	ns
FPGA Resources	1279	1735	registers
Runtime	784582	356310	ns
Cycles	42857	16580	-
Instructions	5400	5400	-

Figure 3: Results of synthesis with mergesort

Calculations:

The FPGA resources and critical path were found in the log file

The runtime, cycles, and instructions were found in the synthesized file.

The maximum clock frequency was found by looking at the system.log file.

The IPC was calculated by dividing the number of instructions by the number of cycles in one successful execution of the program.

The MIPS was calculated using the following formula:

(Instructions/Cycles) * (Cycles/(Runtime * 10^6)).

The latency is the inverse of the max frequency from the log file times 10^9.

Conclusions

In conclusion, the pipeline is way better when it comes to speed . There is an improvement in every category of the table above except for FPGA resources. This could be achieved by having a register for each signal going into the different stages of the processor, including two additional modules which are hazard and forward units to deal with different kind of hazard. The number of registers grew by almost 50% compared to the single cycle but the execution time dropped by more than 50%. First, the IPC as well as the maximum frequency increased which increases speed. Second, the critical path got reduced by at least 2 ns.

The idea behind design is to increase throughput. Breaking up the design will slow down the time it takes for each instruction to be executed.

Finally, our pipeline achieved what we wanted and opened doors for other questions and rooms for improvements in terms of performance. In the upcoming weeks we will be looking to add some branch predictions to better predict branches and caches to even further speed up the program. while looking for potential hazards.

Team Contributions

Karim Itani– Made diagram for pipelined design, wrote hazard unit and testbench, worked on datapath and control unit and worked on midterm report

Milind Shyam – Planned and coded part of the pipeline design, wrote forwarding unit and testbench, worked on pipeline registers and worked on midterm report