Midterm Report

ECE 43700

Vikram Manja, Leo Welter

Mengchi Zhang

14 October, 2016

Overview

To compare the performance of two processors, one should compare a few key metrics. Included in this report are some of the more important ones, such as CPU clock frequency, peak clock speed, total cycles per program, instructions per cycle, instruction latency, and MIPS. To extract these measurements from our design, we ran a test program. The program in question was a simple algorithm for sorting, “merge sort”, which contained mostly branch instructions, combined with a large proportion of stores, loads, and arithmetic instructions. This was a good representation of a realistic application for a CPU, as numerical data manipulation is a highly practical solution to many real-world problems. Particularly, this algorithm tested many hazard detection logic elements in our pipelined design, and put it through a rigorous examination to determine the worst case of its operation.

In general, a pipelined CPU will have a greater clock frequency than that of a single cycle design with similar implementation. Additionally, designing a pipelined CPU requires both the introduction and solutions of new hazards, and forwarding of dependent data to new, future instructions from instructions already under processing. This introduces a modest degree of overhead on both the designer and the hardware, as the control modules required to resolve pipeline hazards create slowing, if necessary, bubbles, halts, and flushing of instruction data. That said, the pipelined design will certainly increase overall processor performance. This is accomplished by, in essence, utilizing all hardware resources simultaneously, rather than sequentially processing one instruction. In our findings, we found that the benefits of the pipelined processor design far outweighed its disadvantages, as with almost all metrics provided, it outclassed its predecessor.

Processor Design

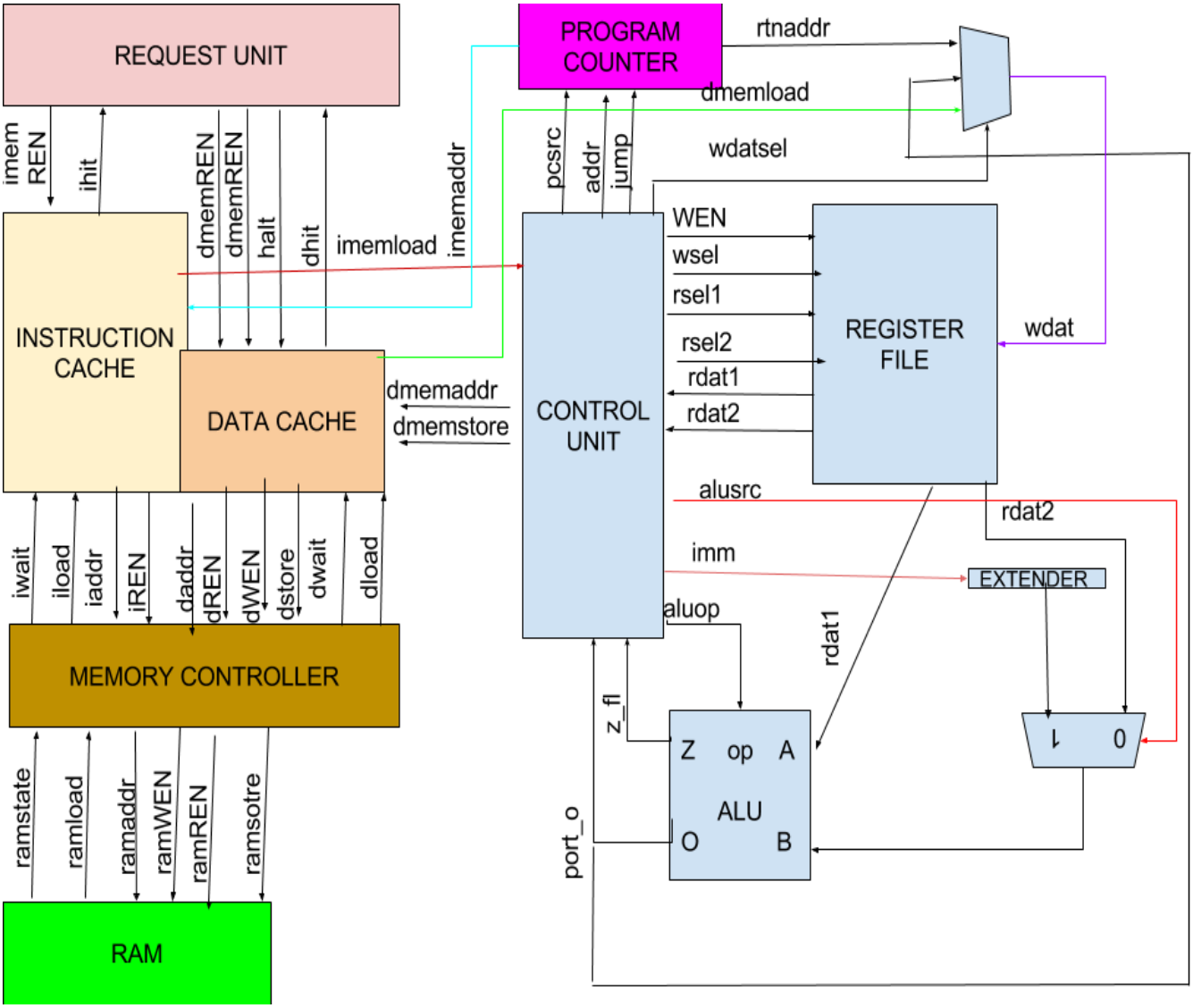

Fig 1.1 (Single Cycle Processor Design)

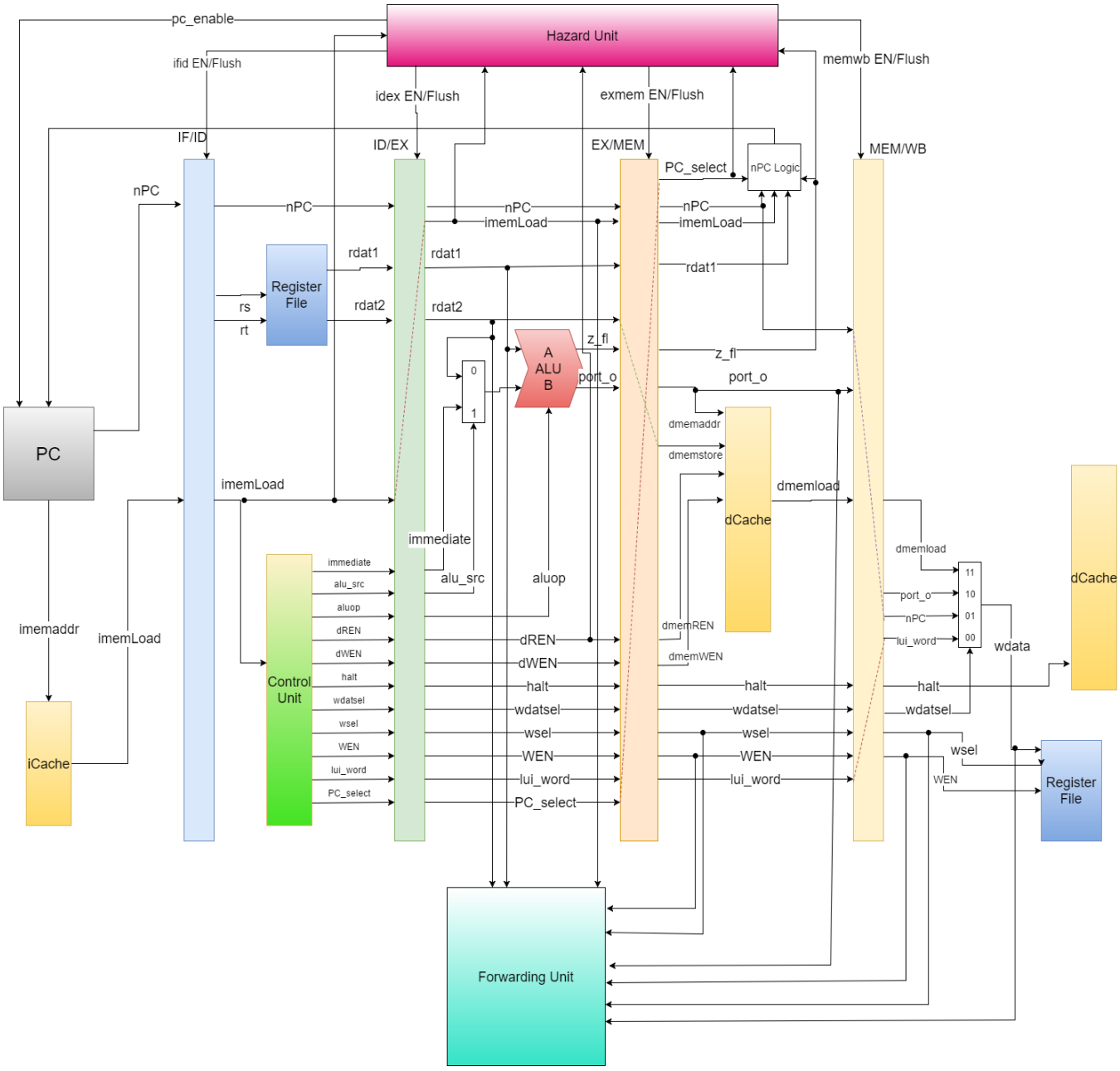

Fig 1.2 (Pipelined Processor Design)

Results

Table 1.3: Results of Test Program		
	Single Cycle	*Pipeline*
Max Frequency Possible(MHz)	56.72	133.54
CPUCLK (MHz)	33.84	66.77
CLK(RAM) (MHz)	56.72	150.92
Highest Achieved Freq. (MHz)	55.55	100
Cycles total	75108	101505
Instructions/Cycle	0.13	0.096
Latency (ns)	32.15	70.3
Critical Path (ns)	27.7	13.833
MIPS	4.18	6.83
Total Logic Elements	3%	3%
Total Combinational Functions	2%	3%
Dedicated Logic Registers	1%	2%
Total Registers	1284	1773

CPUCLK = <from system.log>

CLK(RAM) = <from system.log>

Highest Freq. = 1/(min TB period successful)

Cycles Total = <from TB>

Instructions / Cycle = <Instructions from 'sim -t' / Cycles Total>

Latency (SingleCycle) = Clock Period

*Latency (Pipelined) = Clock Period * 5 (number of stages)*

Critical Path = <from fitter report>

*MIPS = < Instructions/Cycle * Cycles/Second / 1E6 >*

FPGA Resources = <From fitter report>

Conclusions

When the results from the SC design and the PL design are compared, the PL design is better in almost every measurement related to speed and performance. However, the SC design wins out in a few key categories, including latency, instructions per cycle, and total cycles per program. This is due to the PL inherent design philosophy; split the instructions into five cycles each, then utilize every piece of hardware simultaneously. Each hardware resource handles the processing of one instruction, allowing for five instructions to be executed at any given time. The PL design fails to improve on the SC design in a few areas, including instructions per cycle, and the latency of an individual instruction. Issues of latency arise from the PL design's need for five cycles to process one instruction. Even if this produces a better design overall, the individual instructions suffer. This design philosophy of the PL also affects the processors individual instructions per cycle; as it requires five cycles to execute one instruction, the individual IPC suffers.

When compared to the SC performance metrics, however, one can easily see a nearly 50% increase and decrease in MIPS and time per program, respectively. The actual hardware resources required for the PL design increased slightly when compared to the SC design (from 1284 to 1773 total registers). This, when combined with the very large clock frequency, will probably cause a larger power dependency. Overall, the PL design achieved far greater performance when compared to the SC design. This is due to its clever, parallel use of hardware resources; in spite of the greater overhead required to patch the data dependence issues and various hazards this introduces. This utilization of parallelism allows a tighter clock cycle, allowing the machine to run at a much greater frequency overall.

Contributions

Individual contributions to the pipeline project scope were generally split into two parts: Vikram Manja was responsible for the overall design of major blocks, Leo Welter was responsible for debugging and rooting out errors in code. With regard to the implementation of the design, both members of the team joined to pair program on the same machine, allowing one member to easily catch logical flaws as soon as they were introduced. With this workflow, the members allowed each other to work separately on their independent duties (design decisions, testing) whilst simultaneously developing main modules via group work.

Table 1.4: Individual Workload Breakdown

Design Element	*Welter*	*Manja*
Design Decisions	25%	75%
Testing	40%	60%
Debugging	60%	40%
Implementation	50%	50%
Documentation	90%	10%