Single Cycle vs Pipelined processor

ECE 437

Diego De La Fuente

Kyle Diekhoff

John Skubic

Jake Stevens

10/14/2016

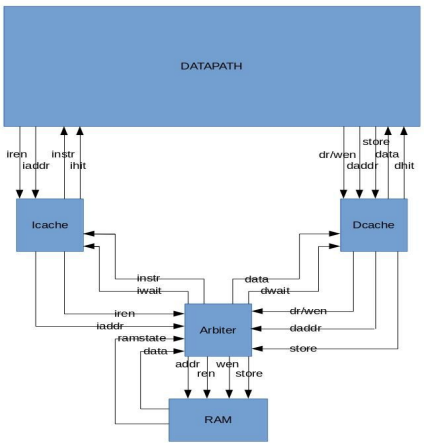
Overview

This report compares results of two processor designs that ran the same program. One of the designs was a single cycle processor, while the other was a 5 stage pipeline processor. We compare several metrics between the two in order to better understand the comparison of each processor's performance running the same set of instructions from the test program called *mergesort.asm.* The main results that give the best display of performance are the MIPS (millions of instructions per sec), which utilizes the Max Frequency that each processor can handle and instructions per cycle in order to display the speed that each processor can run instructions. Our FPGA value shows the number of registers that each processor had, the Instruction Latency shows the amount of time an instruction on average takes to run in the processor and Max Frequency TB was the highest frequency up to 500 MHZ that we were able to run the instruction set at and not crash.

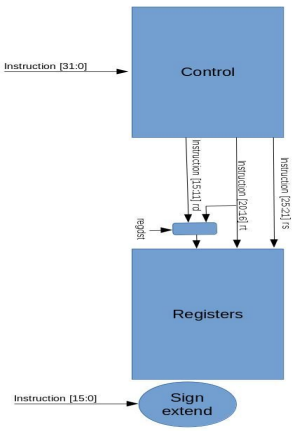
Mergesort.asm was the instruction set that we used for a variety of reasons. For one, this file extensively tested most of the MIPS instruction set a few times, and checked strange cases such as several jumps and branches that happen consecutively. The file also consisted of many instructions that were run overall (5399) to make sure were are extensively testing each processor. The varied amount of instructions also made sure to test the forwarding unit of the pipeline extensively in order to make sure most dependencies are accounted for as well. When we compared the two processors in the end, we saw that the pipeline processor was faster with a MIPS of 31.06 compared to that at 11.03 for the Single Cycle. More detail on this will be discussed in the conclusion.

Processor Design

Pipelined Processor Datapath



Top Level Single Cycle



Single Cycle Instruction Separation

Single Cycle Datapath

Results

	Single Cycle Processor	Pipeline Processor	Formula
Max Frequency	28.17 Mhz (CLK)	126.2 Mhz (CPUCLK*2)	
Max Frequency TB	25 Mhz	100 Mhz	1 / system_tb #PERIOD
Average Instr	0.391 Instr / Clock Cycle	0.242 Instr / Clock Cycle	# of Instr /

/ Clock Cycle			# of Clock Cycles
MIPS	11.03 Instr / s	30.50 Instr / s	Max Frequency * (Average Instr / Clock Cycle)
FPGA	3060	3458	
Latency Used	0	0	
Instruction Latency	0.04 stage * s	0.0396 stage * s	# of Stages * 1 / Max Frequency

Max Frequency was found within the system.log file generated by running the processors with synthesize -t -f 200 system.

Max Frequency TB was found by changing the period within the system_tb.sv file until the highest frequency was found.

# of Instr was found by running the ‘sim’ command.

# of Clock Cycles was found by running ‘make system.sim’

Total Pipeline CC amount: 22341 cycles

Total Single Cycle CC amount: 13791 cycles

Total amount of instructions: 5399 instructions

Conclusions

By comparing the test results obtained we can see that the pipelined processor is nearly 3 times as fast based upon the MIPS values that were calculated. This is due to a higher throughput

in the pipelined design in combination with a clock that is approximately 5 times faster allows for this improvement over the single cycle design.

The reason why the design did not speed up by a factor of 5 was due to the fact that the CPI for our pipelined design went up compared to the CPI of the single cycle design, however it was not close to the substantial speedup of the the clock cycle which allowed us to get us the roughly 3x speedup. This behavior is typical of a pipeline vs a single cycle processor and due to the Iron law, the Time/program with the pipeline processor with a far smaller sec/cycle and similar cpi to the single cycle will obviously be smaller, which is the case for us. The smaller sec/cycle then also contributes to a large max Frequency, which then gives us a larger MIPS result for the pipeline.

Contributions

Diego De La Fuente

Worked on writing the Pipeline processor used for this report. Includes writing source and interface files. Also Worked with Kyle to debug problems that we encountered throughout.

Files contributed to - datapath.sv, control_unit.sv, hazard_unit.sv, execute.sv, memory.sv, decode.sv, fetch.sv, write_back.sv, register_file.sv, pc.sv, forwarding_unit.sv, extender.sv, memory_control.sv, all interface files except forwarding_unit_if.vh,

Kyle Diekhoff

Created testbenches to be used for testing within the Pipelined Processor design. Designed the diagrams used within this document. Worked with Diego to debug various problems encountered within the Pipelined Processor.

Files contributed to - hazard_unit.sv, fetch.sv, execute.sv, memory.sv, decode.sv, forwarding_unit.sv, system_tb.sv, forwarding_unit_tb.sv, hazard_unit_tb.sv, forwarding_unit_if.vh, execute_if.vh,
fetch_if.vh, memory_if.vh, decode_if.vh.