Performance Analysis and Comparison of Single-Cycle and Multi-Cycle Design

Daniel Bielejeski and Ricky Rodriguez

April 2nd, 2020

For this project we designed a multi-cycle processor with a clock speed of approximately 9 times faster than our previously designed single-cycle processor. We designed five test case programs to test and compare both designs.

The first was all add/nand commands. File: add_nand_heavy.asm. Each add or nand takes 2 cycles to complete and we included 96 instructions for a total of 191 cycles between the add and nands and the halt command.

The second was all beq commands. File: beq_heavy.asm. This utilized two branches used to create a loop. We included two load words and an add command to help the running of the loop. Ultimately, we ran the first beq 101 times, the second 100 times, the add 100 times, and the lw twice. We also had one halt command. This used a total of 1021 cycles and had 304 instructions.

The third was all jalr commangs. File: jlar_heavy.asm. This utilized 110 jalr commands and 1 halt command. This program had 111 cycles and 111

instructions.

The fourth was primarily lw and sw commands. File: lw_sw_heavy.asm. This utilized 113 instructions. 56 sw, 56 lw, and 1 halt command. This ran for 897 cycles and performed 113 instructions.

The fifth was a balanced program of all the commands. File: balanced.asm. This included many instructions of each type. It executed 82 instructions and performed 305 cycles.

We also executed all of these test programs on the single-cycle processor. The number of cycles was the same as the number of instructions in the single-cycle design. The number of instructions was the same for both the single-cycle and the multi-cycle processor.

To calculate the total execution time for both programs we used the following formula.

$$exec = cycles * t_{clk}$$

The following table shows the execution time for each of the programs on the single cycle processor. Because the single cycle clock time is 9 times slower than the multi-cycle clock, we assumed the multi-cycle is 1 nanosec-

ond per cycle and the single-cycle is 9 nanoseconds per cycle.

Program Name	Total Instructions	Total Cycles	exec
add_nand_heavy.asm	96	96	864 ns
beq_heavy.asm	304	304	2736 ns
jlar_heavy.asm	111	111	999 ns
lw_sw_heavy.asm	113	113	1017 ns
balanced.asm	82	82	738 ns

The following table shows the execution time for each of the programs on the multi-cycle processor. We assumed the clock time to be 1 nansecond per cycle.

Program Name	Total Instructions	Total Cycles	exec
add_nand_heavy.asm	96	191	191 ns
beq_heavy.asm	304	1021	1021 ns
jlar_heavy.asm	111	111	111 ns
lw_sw_heavy.asm	113	897	897 ns
balanced.asm	82	305	305 ns

To calculate the total speedup of the program we use the following equation:

$$\frac{perf(A)}{perf(B)} = \frac{exec(B)}{exec(A)} = n$$

For this we will assume the multi-cycle processor is A and the single-cycle processor is B. The following table shows the speedup for each program of

the programs.

Program Name	Single-Cycle Exec	Multi-Cycle Exec	Speedup
add_nand_heavy.asm	864	191	4.5
beq_heavy.asm	2736	1021	2.8
jlar_heavy.asm	999	111	9
lw_sw_heavy.asm	1017	897	1.1
balanced.asm	738	305	2.4

For a balanced program we achieve a speedup of approximately 2.4. This is in line with the given cycles per instruction for each instruction. Lw and sw take almost 8 cycles so it makes sense that we do not get much speedup on programs with primarily those. Similarly for jlar, we only take 1 cycle per instruction so it makes sense that we get a speedup of 9. Overall the multi-cycle design is faster in every case and much faster in most cases.