Performance Analysis

Noah Beer, Page Bennett

Our benchmarking suite consists of four benchmarks. Each benchmark is an assembly program that is heavy on one of the following sets of instructions: add/nand, lw/sw, beq, jalr. We ran each test on both our single-cycle simulator and our multi-cycle simulator and recorded the number of cycles it took to complete. The results are recorded below.

Benchmark	**Single-Cycle cycles**	**Multi-Cycle cycles**
add/nand	5000	12011
lw/sw	70	285
beq	205	539
jalr	44	94

We Used the following equation to calculate the relative execution time between the two designs. Because the clock time for the Single-Cycle design is 5 times longer than the Multi-Cycle design, we set t_{clk} of the single cycle design to 5, and t_{clk} of the Multi-cycle design to 1:

$$exec = IC * CPI * t_{clk} = Cycles * t_{clk}$$

Next, to calculate speedup, we used the following equation:

$$n = \frac{exec(single - cycle)}{exec(multi - cycle)}$$

The table below shows the results:

Benchmark	**Single-Cycle execution time**	**Multi-Cycle cycle execution time**	**Speedup**
add/nand	25000	12011	2.08
lw/sw	350	285	1.23
beq	1025	539	1.90
jalr	220	94	2.34

These results reflect exactly what we expected. Jalr heavy programs receive the largest speedup. This is because jalr only takes up one cycle in the multi-cycle design, benefitting from the change the most. Add/nand received the second highest speedup, and takes the second lowest number of cycles with two. Beq came in third owing to taking three cycles. Our lw/sw heavy program had the least speedup, as lw and sw both take a full 5 cycles in the multi-cycle design, which is a speedup of only one. The reason the program's speedup was still greater than one is because there were still instructions other than lw and sw in the program that benefitted from the move to a multi-cycle design.