Design Project 3: Dynamic Scheduling

Connor Smith

Department of Electrical and Computer Engineering

Brigham Young University, Provo, UT 84602

connor.smith.256@gmail.com

November 25, 2015

1 Introduction

For decades, processors have used pipelining to exploit instruction-level parallelism. However, dependences among instructions as well as memory latency result in pipeline stalls and reduce throughput. Out-of-order processors seek to minimize the number of idle cycles by reordering the execution of instructions, a process known as dynamic scheduling. As long as this reordering is guaranteed to produce the same results as in-order execution, dynamic scheduling can minimize or hide stalls, increasing overall performance.

Many architectural parameters contribute to the organization of a processor. For a typical processor, these include the cache configuration, branch predictor, and number of functional units. For out-of-order processors, these also include the instruction queue size, the decode bandwidth, the RUU issue width and size, instruction commit bandwidth, and the load/store queue size. Each of these (and others) must be carefully chosen to maximize the desired characteristics. Since the design space is so large, it is necessary to use a methodology that is capable of considerably narrowing the space to determine the best configuration for a chosen market.

2 Design Methodology

2.1 Base Configuration

The design choices in this project reflect the target market of mid- to high-performance desktop computing. As a result, the primary characteristic of concern is overall performance. Since the cycle time is set to 600 MHz and does not vary from one design to the next, instructions per cycle (IPC) is a fair metric to use when comparing the performance of different configurations. Constraints were also placed on the average

and maximum power consumption of the processor in an attempt to limit designs to those applicable to a typical desktop computer that does not require any special cooling.

The starting configuration is mostly based off of the default values in sim-outorder, with some modifications. The L1 and L2 cache configurations are based off those used in Intel's Haswell architecture and used in Design Project 1, and these remain constant across all designs explored in this project, since the focus of this project is on dynamic scheduling. The memory latency was fixed at 100ns, which corresponds to 60 cycles for a processory running at 600 MHz. Also, a perfect branch predictor was chosen for all of the designs. It should be noted that this limits the accuracy of the results; the reported IPC will be higher than in reality, and the reported power will likely be slightly lower. However, this choice eliminates a source of complexity, and the ranking of which designs are better should remain roughly the same. In addition, it is unlikely that the architecture would be utilized to its full potential when using an imperfect predictor, so the use of a perfect branch predictor should illustrate the full difference in performance between designs. Thus the end goal is less focused on presenting completely accurate metrics for a given design and more on determining which designs are better relative to others. The parameters related to out-of-order execution in the starting configuration were set to be the defaults of sim-outorder. Table 1 shows a summary of the base configuration.

Table 1: Baseline configuration

L1 Capacity	32KB
L1 Associativity	8
L2 Capacity	128KB
L2 Associativity	8
Block Size	64
Memory latency	60
Branch prediction	perfect
IFQ size	4
Decode width	4
Issue width	4
Commit width	4
RUU size	16
LSQ size	8
I ALU	4
I MULT	1
FP ALU	4
FP MULT	1

2.2 Parameter Variation

Efficient parameter variation is necessary to narrow the design space. For each configuration that is the best so far, a single next parameter is chosen and varied. If the new configuration is better overall, then

the parameter is further varied in the same direction until it is no longer beneficial. At this point, a new parameter is chosen and the process repeats until no further optimizations can be made. It is important to note that parameter values for a local maxima are not necessarily the same as for global maxima. In other words, the "best" value for a given parameter may change after altering a different parameter. Because of this, parameters were revisited at the end of the analysis to ensure that the chosen design really is the best so far.

The most important criterion for whether a given configuration change is beneficial is raw performance, measured in IPC. For many changes, however, the IPC increases slightly but at the cost of significantly more power dissipation. This tradeoff is evaluated and a determination is made whether it is worth it. Since the target market for these designs is desktop computing, a soft power limit was set at 150W. Both average and maximum power per cycle are used in these considerations. Average power is important for overall energy consumption, while maximum power is an important consideration due to heat dissipation issues. SimpleScalar offers multiple different power models, and the non-ideal conditional clocking (CC3) model was used for this project.

There are a few more metrics that were used to determine which parameter to change next. These are the fraction of time that the instruction queue, register update unit, and load/store queues were full. While these statistics are not directly useful for determining which configuration is best, they do provide a guide for whether greater performance would result from changing the parameter that corresponds to the full rate for a given unit.

3 Configurations and Results

3.1 Preliminary Configurations

The results for the baseline configuration are shown in Figures 1 and 2. As can be seen, the IPC varies wildly from one benchmark to the next, ranging from 0.162 to 2.72, with a mean of 1.330. The SPEC benchmarks are quite diverse, so in order for a configuration to be better, it must yield an increase in the average IPC (and hopefully across all benchmarks).

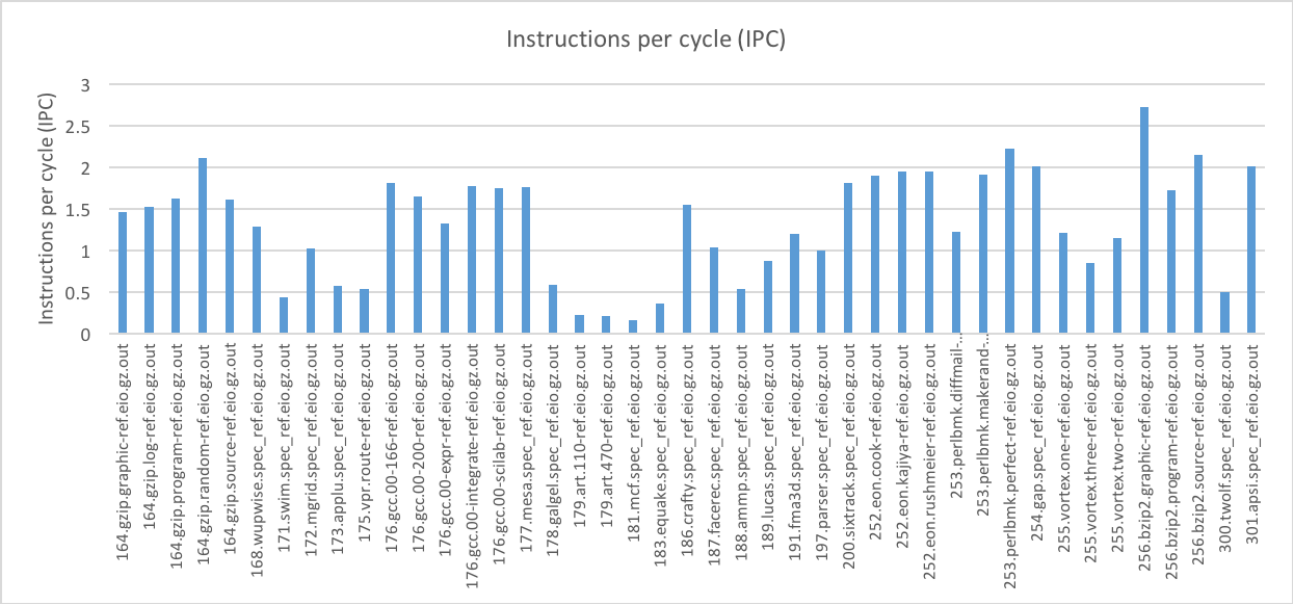

Figure 1: Instructions per cycle by benchmark

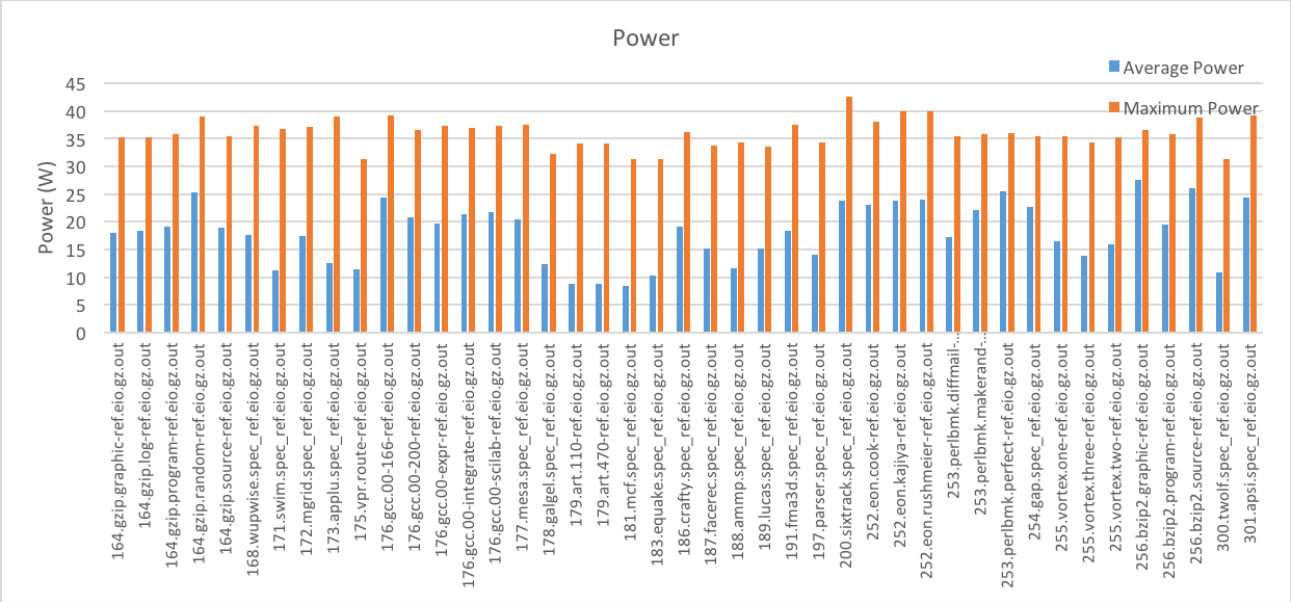

Figure 2: Average and maximum power by benchmark

Table 2 shows the average results for several different preliminary configurations, including the baseline. These configurations were analyzed to understand how much effect some of the parameters have on IPC and power. The "Double" configuration is the result of doubling each of the values, which yielded significant performance increases and kept power reasonable. "Max FUs" is the result of increasing the number of all the functional units to 8, which is the maximum value supported by SimpleScalar. Interestingly, this had a negligible effect on IPC but resulted in a spike in power, which illustrates that there are little to no structural

hazards with the functional units in the baseline configuration. The "Min FUs" configuration corresponds to having only a single functional unit of each type. This caused a dramatic drop in performance, and was used to compute a soft minimum for IPC.

Table 2: Preliminary Configuration Results

	Baseline	Double	Max FUs	Min FUs
IPC	1.330	1.896	1.367	0.788
Average Power	18.078	31.540	22.185	12.290
Peak Power	36.070	75.266	43.278	25.416

3.2 Register Update Unit

The first parameter that was changed was the register update unit (RUU), which is a combination of reservations stations and a reorder buffer (ROB). Based off the results of the baseline configuration, the RUU was full 50% of the time. Increasing the bandwidth of the other units will not help much if the RUU is full, so the size was increased to 32 and 64. This decreased the full rate to roughly 1%. The results for these configurations are shown in Table 3. IPC sees a modest improvement of 1.05x, while average and peak power increase by roughly 25% each. There are clearly diminishing returns when increasing IPC.

Table 3: Increasing RUU Size

RUU Size	16	32	64
IPC	1.330	1.386	1.389
Average Power	18.078	20.384	22.561
Peak Power	36.070	40.135	44.555

3.3 Out-of-Order Bandwidth

With a larger RUU, a good next choice of parameter to alter was the bandwidths of the out-of-order units: the fetch queue size, decode width, issue width, and commit width. These values were increased from 4 to 8. While they could be increased separately, it did not seem to make much sense to have a fetch queue size or commit width that were different, for example, so for the sake of simplicity each value was altered as a group. The results are shown in Table 4. As with increasing the RUU size, changing these parameters yielded a modest improvement of 1.05x to IPC. This time, however, average and peak power increased by 43% and 78%, respectively. These numbers are still well within the typical range for a desktop CPU, so the increase in IPC is worth it.

Table 4: Increasing Out-of-order Unit Size/Bandwidth

Size/BW	4	8
IPC	1.389	1.463
Average Power	22.561	26.045
Peak Power	44.555	60.148

3.4 Functional Units

Using the architecture at this point, the processor can reorder and feed multiple instructions per cycle to the functional units, but there are still only a few FUs available. To reduce this congestion, the count of each functional unit was doubled, bringing the number of ALUs to 8 and multiply units to 2 for both integer and floating-point. Table 5 shows the results for this change. Yet again, this change resulted in a 1.05x boost in IPC. Average power increased by another 40%, and peak power increased by 46%. At this point, the power ceiling for desktop computing isn't too far off from the peak value of 88W reached by this new configuration.

Table 5: Adding Functional Units

Number of FUs (ALU, MULT)	(4, 1)	(8, 2)
IPC	1.463	1.547
Average Power	26.045	36.479
Peak Power	60.148	87.705

3.5 Load/Store Queue

Thus far, not much has been done about latencies due to memory constraints. Indeed, at this point, the L/S queue is full 80% of the time. Increasing the size of the load/store queue can help hide these limitations. The size of the queue was increased from 8 to 16, and the results are shown in Table 6. This caused the largest increase in IPC so far (1.36x). The increases in average and peak power (22% and 40%, respectively) were also some of the smallest of the changes so far. Increasing the size of the L/S queue was clearly a big win.

Table 6: Adding Functional Units

L/S Queue Size	8	16	32	64
IPC	1.547	1.886	2.073	2.100
Average Power	36.479	40.853	43.485	44.376
Peak Power	87.705	106.141	116.737	122.404

3.6 Returning to Past Parameters

At this point, there has been an overall IPC improvement of 1.58x, at the cost of a 2.45x increase in average power and 3.4x gain in peak power. With a peak power of over 120W, any further changes must have dramatic performance increases to justify increasing the power consumption any further. Since the number of ALUs was already at the maximum value that SimpleScalar supports, only the out-of-order width parameters and the RUU size were left to revisit. Table 7 shows the results of these changes. Increasing the RUU size does yield a performance increase, but at the cost of increasing the peak power consumption to a whopping 206W, which simply cannot be justified. In addition, increasing the issue/dispatch/commit widths again actually resulted in a net decrease in IPC while still increasing power consumption drastically.

Table 7: Revisiting Past Parameters

	Best So Far	Increasing RUU Size	Increasing width
IPC	2.100	2.373	2.065
Average Power	44.376	61.814	63.878
Peak Power	122.404	206.089	185.695

4 Conclusion

The final configuration is shown in Table 8, and a comparison of IPC for each benchmark is shown in Figure 3. There is a clear and significant improvement in performance; no benchmark performs worse under the new configuration, and while some see only marginal benefit, those with high starting IPC see a dramatic improvement.

Table 8: Baseline configuration

IFQ size	8
Decode width	8
Issue width	8
Commit width	8
RUU size	64
LSQ size	64
I ALU	8
I MULT	2
FP ALU	8
FP MULT	2

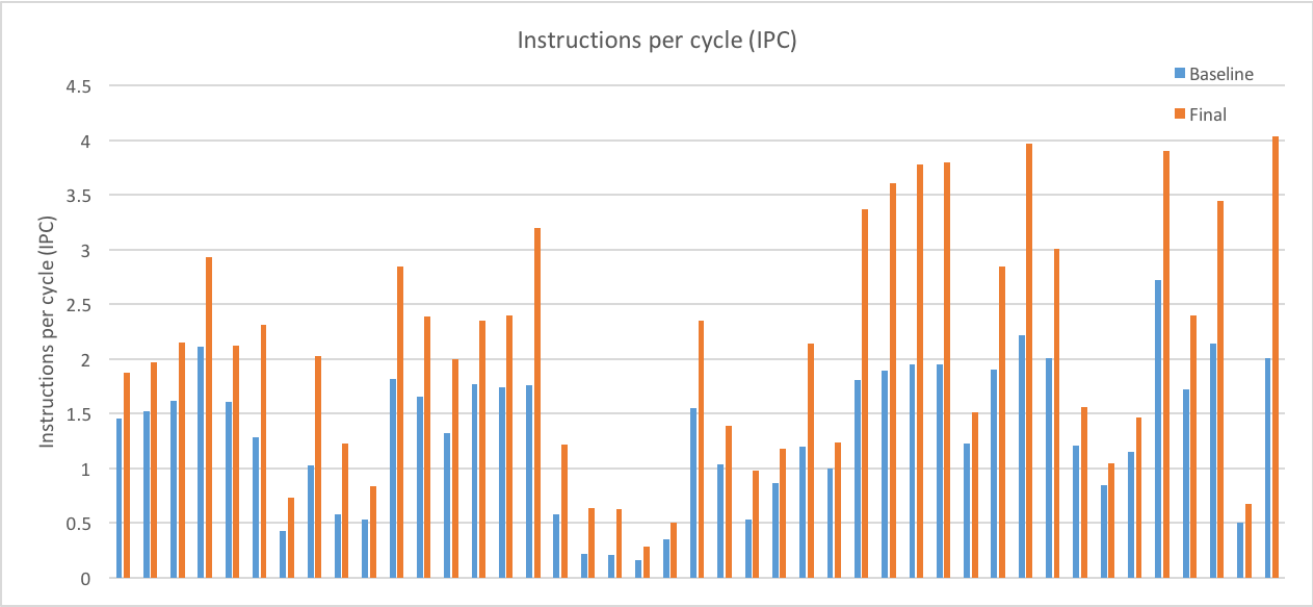

Figure 3: Baseline vs. Final configurations

Designing an out-of-order processor is clearly much more involved than designing a simple in-order one. Each of the different parameters can affect each other significantly, and even after narrowing the design space considerably, there are still a huge amount of configurations available. Only a few such designs could be tested within the scope of this project, but they should be representative. As with most designs in computer architecture, the configurations in this project were full of trade-offs. If the target market were chosen differently, the best design would be markedly different from the one selected here. If a lower IPC were acceptable, the power consumption could be drastically lower. Likewise, if power consumption wasn't as much of a concern, more performance could have been squeezed out of the designs. OVerall, the final design strikes a good balance between performance and power consumption, which is excellent for the chosen market.