NC State University

Department of Electrical and Computer Engineering

ECE 463/563: Fall 2021 (Rotenberg)

Project #3: Dynamic Instruction Scheduling

by

SRI HARSHA TEJ PATNALA

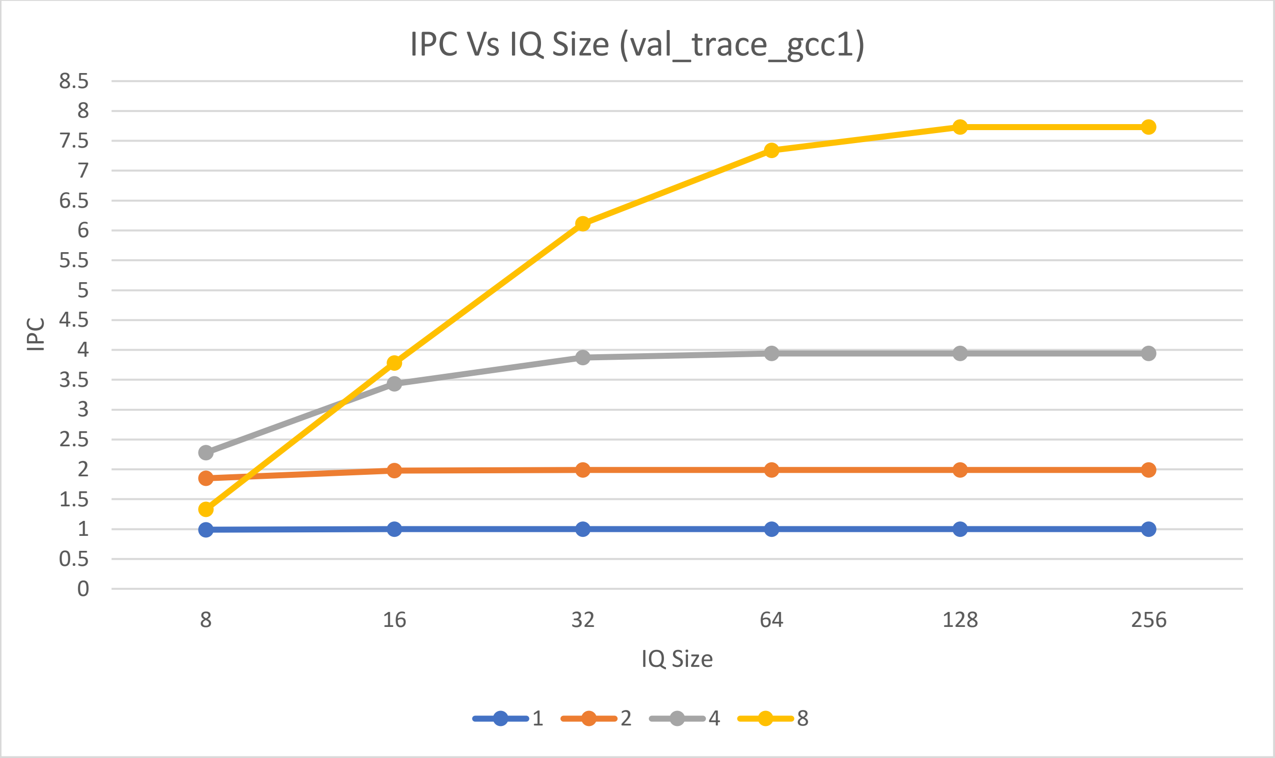
NCSU Honor Pledge: "I have neither given nor received unauthorized aid on this project."

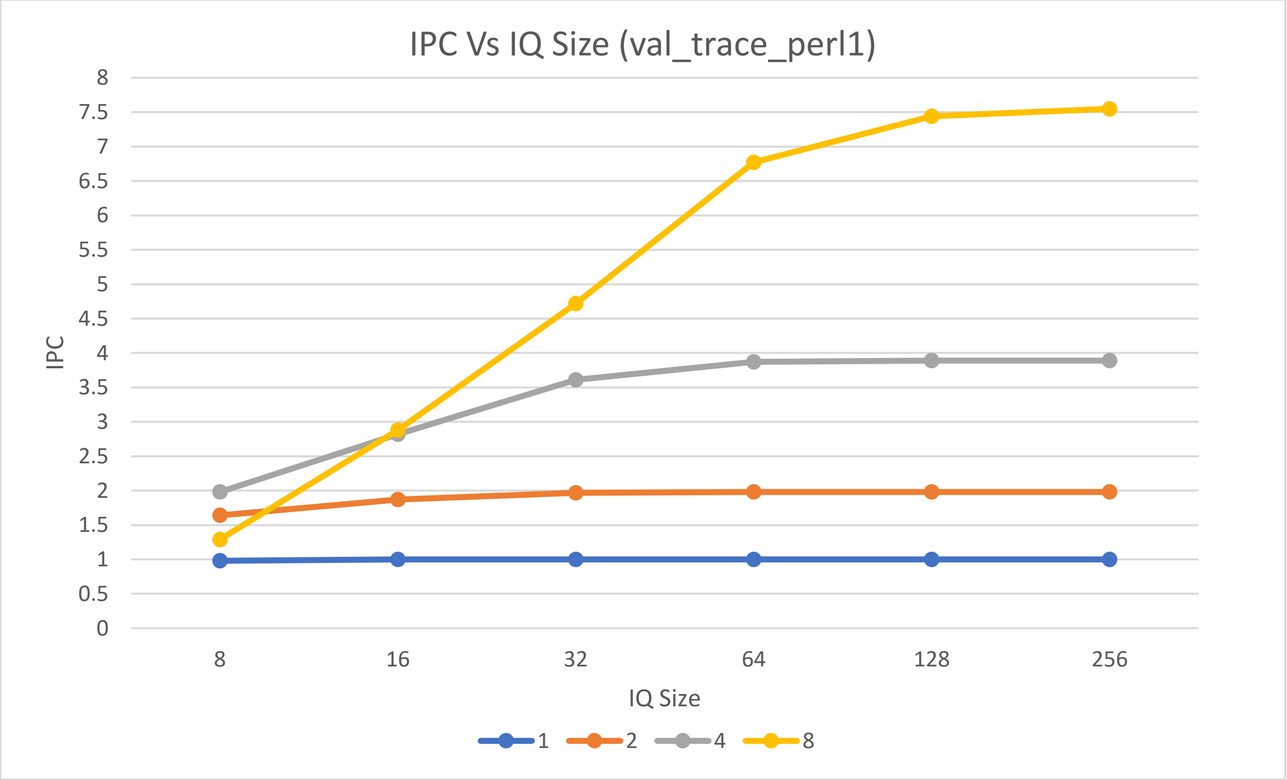
Student’s electronic signature: ___________Sri Harsha Patnala____________________
(Sign by typing your name)

Course number: _________563__________
(463 or 563?)

Effect of Issue Queue Size

1. Graphs for various benchmark traces





2. Graph Analysis

Benchmark 1 (val_trace_gcc1)

The graph above indicates that for a fixed reorder buffer size, increasing the issue queue size has improved the IPC for increasing superscalar width. For larger width, a larger issue queue size helps in achieving the theoretical IPC (which is WIDTH instructions/cycle).

For WIDTH=1, a small issue queue size would be sufficient as the graph proves this fact, as increasing the issue queue size has only saturated the IPC. This is true, because regardless of the IQ size, we still fetch only one instruction each cycle. So, a lot of it has to do with the benchmark trace that was used for simulating, which indicates that a small IQ size would yield the optimal performance.

However, taking only IPC into consideration would not help in identifying the optimal IQ size. So, we also need to consider the total number of cycles it has taken to find out the best IQ size. Based on the simulations, we find that for an IQ size of 16 or more, the number of cycles remains constant (10032). Since, in actual hardware a smaller size implies lesser area in general, we can consider that for WIDTH=1, IQ size of 16 would be optimal.

Similarly for larger widths also, observing the total number of cycles would help us to determine the optimal IQ size within 5% of the maximum IQ size used for simulation.

For WIDTH=2, the total number of cycles and IPC saturate for an IQ size of 32. So, a size of 32 would be optimal for this width.

For WIDTH=4, the IPC and total number of cycles and IPC starts to saturate from size 64 and larger. Hence 64 would be the optimal size of this width.

Similarly, for WIDTH-8, for an IQ size of 128 and larger, the IPC almost remains the same. So, 128 would be optimal for this width.

Benchmark 2 (val_trace_perl1)

Like the discussion for the previous benchmark trace, to determine an optimal IQ size, we need consider both the IPC and the total number of cycles taken to retire all the instructions.

For WIDTH=1, starting from an IQ size of 16 and larger, the IPC tends to settle down, and the total number of cycles also saturate and doesn’t improve drastically. By taking into consideration the total number of cycles, an IQ size of 16 would be optimal for this configuration.

For WIDTH=2, an IQ size of 16 seems to yield an optimal performance which is within 5% of the highest IPC for largest IQ size.

Similarly, for WIDTH=4, IQ size of 32 looks to be the optimal size for this configuration.

Finally for WIDTH=8, an IQ size of 128 yields an IPC within 5% of the highest IPC (7.55) for the largest IQ size.

The below shows a summary of the analysis in a tabular form, for each benchmark trace that was used for analysis.

Optimal IQ Size per WIDTH		
	Benchmark 1 (*val_trace_gcc1*)	Benchmark 2 (*val_trace_perl1*)
WIDTH=1	16	16
WIDTH=2	32	16
WIDTH=4	64	32
WIDTH=8	128	128

3. Discussion

By looking at the summarized table above, we can say that generally for a larger WIDTH, a larger IQ size would help in achieving better performance and improve the IPC closer to the theoretical desired value. This trend is observed because every time we keep on fetching exactly WIDTH instructions. And we continue to pass on the instructions at the same pace, at every stage until we reach the dispatch stage. Only at this stage, we are limited by the number of available entries in the Issue queue.

So even if there are available entries in the IQ, we still must dispatch to the queue all the WIDTH instructions at once. This means that if the issue queue still has fewer entries than WIDTH, we can still add the instructions to the queue. But we don’t do so and wait until the queue can support all the WIDTH instructions. So eventually we have to stall at the dispatch stage and wait till the queue is large enough to add all the WIDTH instructions. So having a larger IQ size would help us resolving the stalled cycles at the dispatch stage. So, increasing the IQ size to support larger WIDTH, helps in reducing the number of stalled cycles, and further improves the IPC for the simulator.

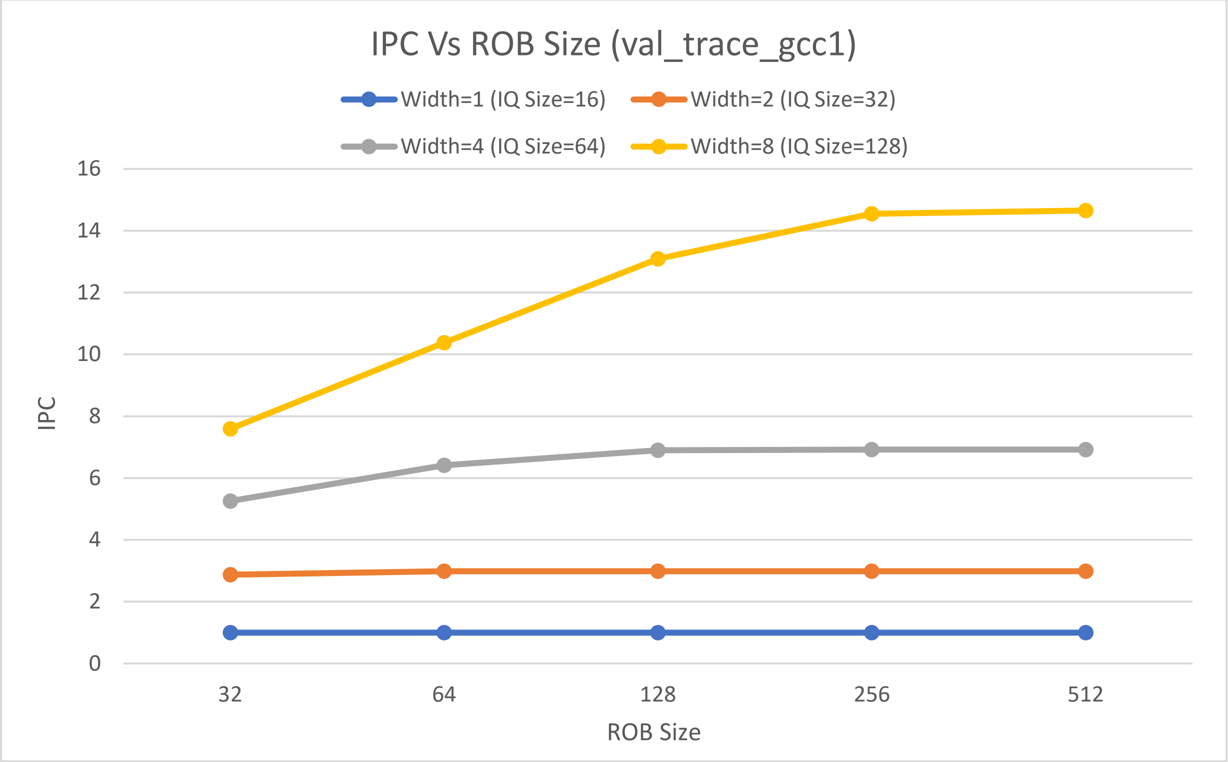
Also, the trend saturates at a certain IQ size and increasing the size further doesn’t help in improving any performance. This is because, we still fetch WIDTH instructions only, regardless of the IQ size. So, it is evident that at a certain size, WIDTH would not be the limiting factor, and increasing the IQ size further doesn’t better the performance significantly.

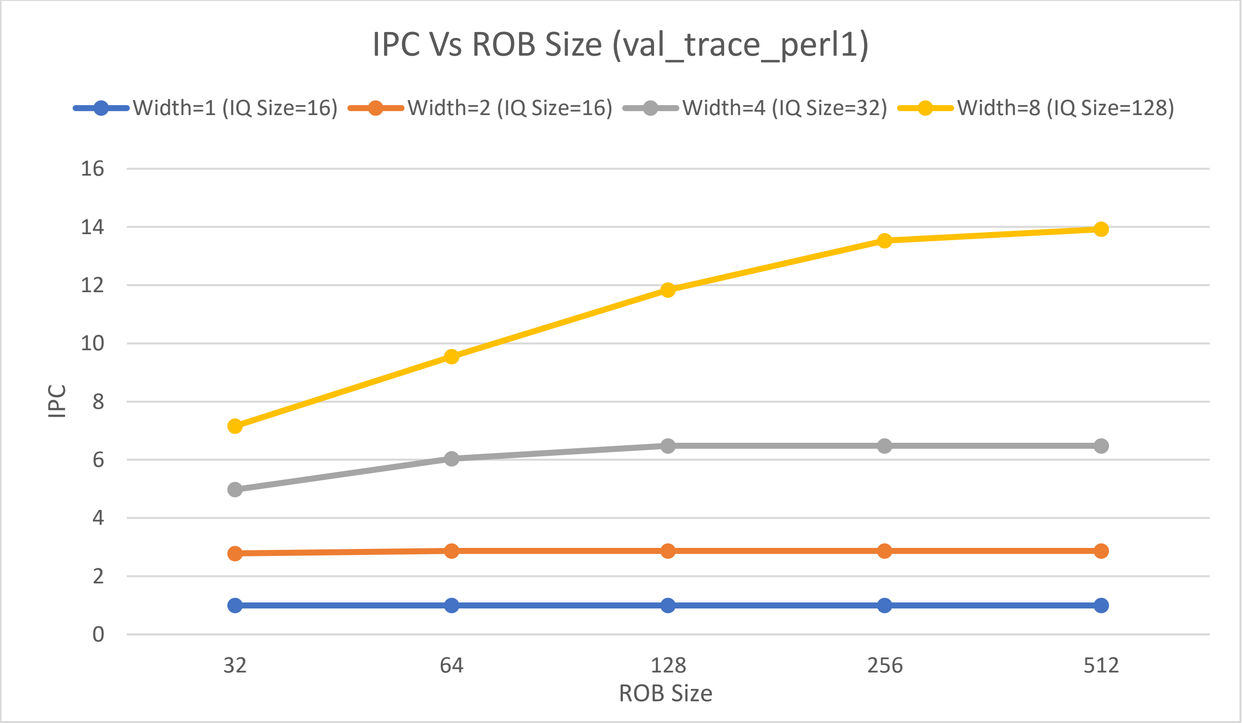
Observing the statistics collected for this analysis, it is found that a certain benchmark trace yields a better IPC in overall. This is true, because the total number of cycles an instruction take depends a lot on the previous instructions that might have introduced dependencies among source and destination registers. If all the preceding instructions somehow introduce dependency for the current instruction, then clearly these instructions might have to stall or wait in the Issue queue longer, and thus eventually increasing the total number of cycles to retire safely.

So, the benchmark trace might have instructions that introduced dependencies among several other instructions, and hence could affect the IPC and the total number of cycles to finish the simulation.

Effect of Reorder Buffer Size

1. Graphs for various benchmark traces


IPC Vs ROB Size (val_trace_gcc1)
Width=1 (IQ Size=16)
Width=2 (IQ Size=32)
Width=4 (IQ Size=64)
Width=8 (IQ Size=128)
IPC
ROB Size


IPC Vs ROB Size (val_trace_perl1)
Width=1 (IQ Size=16)
Width=2 (IQ Size=16)
Width=4 (IQ Size=32)
Width=8 (IQ Size=128)
IPC
ROB Size