NC State University

Department of Electrical and Computer Engineering

ECE 463/521: Spring 2015

Machine Problem #3: Dynamic Instruction Scheduling

by

PRATHEEK BRAMHASAMUDRA MALLIKARJUNA

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

Student's electronic signature: Pratheek Bramhasamudra Mallikarjuna
(sign by typing your name)

Course number: 521

1. INTRODUCTION

Superscalar processors are more common today. As the VLSI technology got advanced more and more, number of transistor which can be fit in a dye area increased rapidly. Because of which new and new technologies where introduced to speed up the processor. One of such technology is pipelining. Next came the superscalar processors which can take in multiple instructions and process then in same clock cycle. One of the algorithms which most processor makers follow now a days to avoid many data dependence stalls is Tomasulo’s algorithm, which allows the processor to execute in out of order manner.

This project deals with simulating Tomasulo’s algorithm in a N was superscalar processor with S out of order scheduling buffers. For this, a class named SuperScalar is designed to operate like a processor, which fetch, dispatch, issue, execute, write back and retire. Perfect memory access time and, no beach instructions are assumed.

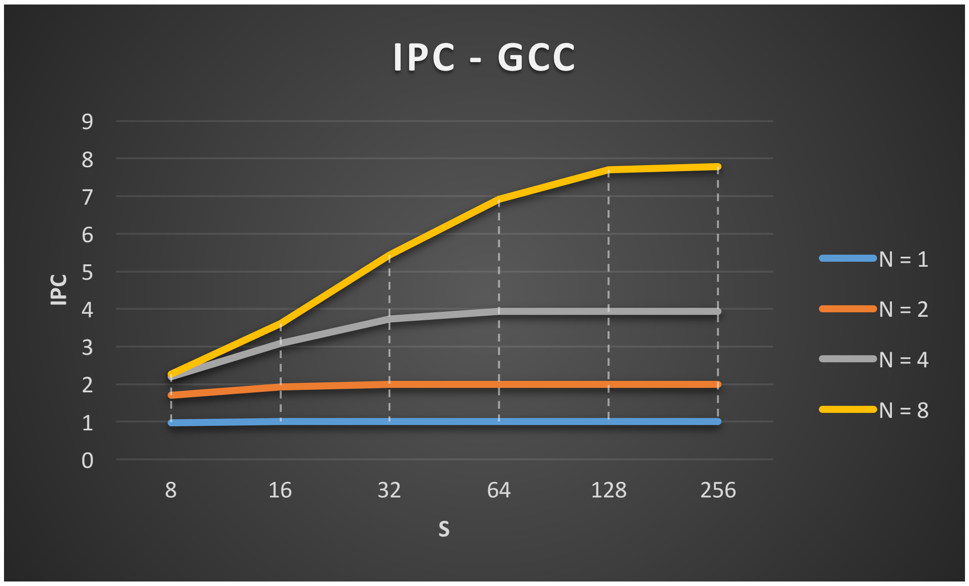
The project analyses the effect of S and N on Instructions per cycle (IPC). Different trace files are chosen so see the effect of different kinds of data dependencies as well.

2. RESULTS

This section shows the effect on IPC by N and S over different trace files.

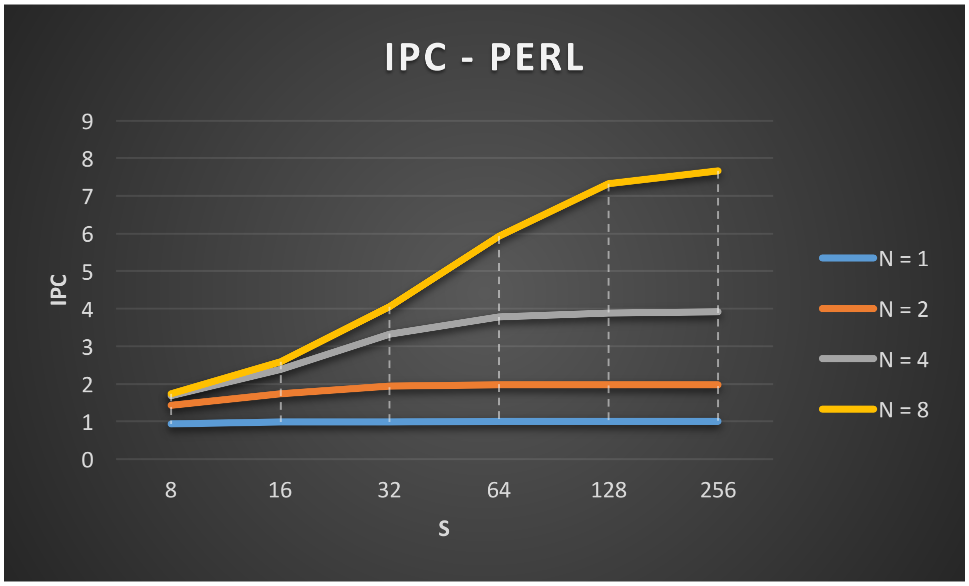
2.1 GCC Trace file

S	N = 1	N = 2	N = 4	N = 8
8	0.98	1.71	2.2	2.26
16	1	1.93	3.09	3.62
32	1	1.99	3.73	5.43
64	1	1.99	3.93	6.91
128	1	1.99	3.94	7.7
256	1	1.99	3.94	7.79



2.2 PERL trace file

S	N = 1	N = 2	N = 4	N = 8
8	0.93	1.43	1.69	1.74
16	0.98	1.74	2.39	2.59
32	0.99	1.95	3.32	4.05
64	1	1.98	3.79	5.93
128	1	1.98	3.89	7.33
256	1	1.98	3.91	7.66



3. DISCUSSIONS

The general trend we can see in both the graphs is IPC increases with increase in N (N – peak issue rate) and also increases with increase in S (number of Scheduling buffers). The following discussion will make the comparison clearer.

- **What happens to IPC as Scheduling Queue size (S) increases? (Does the answer depend on N?)**

We can see from both the graphs that IPC increases with increase in S. However the extent of the increase depends on N. For GCC trace file, the increase in IPC when S was increased from 8 to 16 was 0.98 to 1 for N = 1, while the increase in IPC was 2.26 to 3.62 for N = 8. The trend is similar in PERL trace file. Thus, we can see the greater effect of increasing S when N is large. This is because, with greater N, we will be needing more number of Scheduling buffers, and lower S will be the bottleneck of the pipeline thus hindering the performance.

One more trend we see is IPC getting saturated when S increases to higher amounts. This is because with increase in S, the bottleneck in the pipeline is removed thus avoiding structural hazards and the performance hits will be only due to data dependencies which cannon be handled by Tomasulo's algorithm. We can also see that, saturation hits quicker for smaller N values.

- **What happens to IPC as peak issue rate (N) increases? (Does the answer depend on S?)**

The trend in which both the graph follow is, increase in IPC with increase in N. However, the extent of this increase depends on S. When N changes from 1 to 8 for S = 8, the increase in IPC is 0.98 to 2.26 (GCC), 0.93 to 1.74 (PERL). However, when N changes from 1 to 8 for S = 256, the increase in IPC is 1 to 7.79 (GCC), 1 to 7.66 (PERL). From this, we can see that, N has greater effect on IPC when S is large compared to small S. This is because, with lower S, we will not be fully utilize ILP, since it will be the bottleneck of the pipeline thus affecting the performance. In general, with increase in N, we will need higher S to exploit ILP and gain good IPC.

Another thing we notice is IPC cannot be increased beyond N, i.e., the maximum value IPC can reach for a N – way super scalar processor is N. This is because, only N instruction can be processed by the N – way superscalar processor in one cycle thus limiting the number of overall instructions processed per cycle.

- **What is the relationship between S and N?**

For the better utilization of instruction level parallelism, S should be significantly higher compared to N (at least 8 to 10 times). This is because there will be data dependency among immediate instructions and with bigger S, we can utilize ILP to the fullest extent. If S is small or comparable to N, even though we might see some improvement in IPC, the improvement will not be significant.

- **Do some benchmarks show higher or lower IPC than other benchmarks, for the same microarchitecture configuration? Why might this be the case?**

  Yes, we can see that IPC has better values in case of GCC trace file compared to PERL trace file. This is because GCC might have lesser data dependency stalls (which cannon be avoided by Tomasulo's algorithm) compared to PERL because of which number of instruction executed per cycle will get a boost.