ECE 4100 Project 2 Report

Out of order execution in a superscalar pipelined processor with speculative execution

Ruoyang Xu

Last Edited: March 29, 2019

Introduction

This is the report for ECE 4100 Advanced Computer Architecture Project 2: Processor Simulator. The processor will conduct its out-of-order execution by Tomasulo Algorithm, perform speculative execution using a GShare branch predictor and speed up memory access using a direct mapped L1 data cache, with a block size fixed to be 6 ($B = 6$). The processor is configured by the following 7 parameters:

- **F** : Dispatch rate.
- **K** : Number of k0 function units (ALU).
- **L** : Number of k1 function units (Pipelined MUL).
- **M** : Number of k2 function units (Load/Store Unit).
- **R** : defines the total number of schedule unit as $R \times (K + L + M)$.
- **G** : 2^G the number of of entries in GShare branch predictor.
- **C** : 2^C the size of data cache in byets.

Absolute best performing results are document at in table 1. Final results are summarized at the end of document in table 2.

Processor Experiment

This project is tasked to find the optimal configuration for the processor under a budget while minimizing the number of schedule unit and functional unit. **Optimality** is defined as the *Instruction retired per cycle*(IPC). The higher the IPC, the more optimal the configuration is.

The processor is limited by a memory budget, where the total storage bits **cannot exceed 64KiB**. This includes the L1 cache and branch predictor as well as the global history register, which is G bits long. Such budget posed limitation onto the range of G and C value.

In order to have a meaningful size of branch predictor and cache, the minimum value of G is set to 8 and that of C is 10. Number of storage bits is calculated using

$$2^{(C-6)} \times (2 + (64 - C) + 2^6 * 8) + 2^G \times (2 + 64) + G$$

and is converted to KiB by dividing (8×1024). The results yield that C would be between **10 and 15**, while G would have range between **8 and 12** except for $C = 15$ while maximum of G would be **11**.

Since a search for minimal configuration is required after obtaining the absolute optimal configuration, a thorough search is required, therefore the experiment would be conducted on all configurations for $F \in [1, 4], R \in [1, 4], \{K, L, M\} \in [1, 3]$.

General Guidelien in Finding the Best Configuration

Eliminating Data Dependencies The IPC is influenced by branch prediction accuracy and cache access miss rate. IPC can be further reduced by looking further in the future instructions and fire those that does not have dependencies. A hypothesis is made that *in general*, a larger fetch rate and R value should bring down the IPC. However having F smaller than R would be less effective since F would not be able to fill the reservation station in some cases. Figure 1 is a demonstration of the trend and shows that during the search for absolute best result, we should aim for a high F value associated with a high R value.

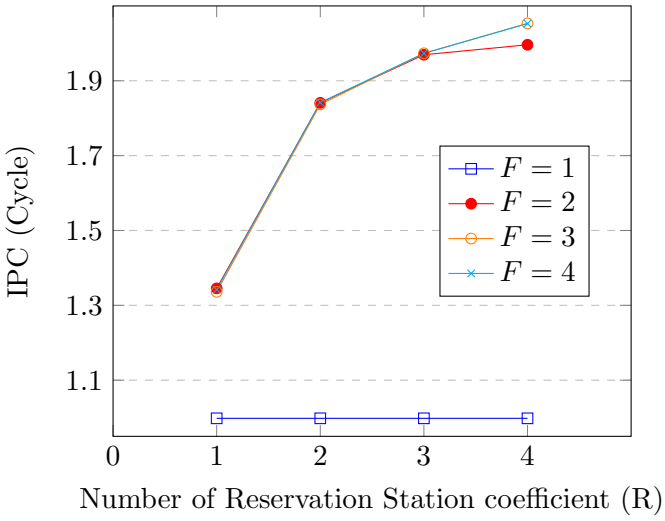

Figure 1: VC Miss Rate vs VC Size

L1 cache miss rate A crucial factor to the cycle time. Since the L1 data cache is a direct mapped cache, increasing the cache size should opens up more spot in the cache and decrease the conflict misses. Figure 3 is a demo showing the cache miss rate. All other configurations are kept the same.

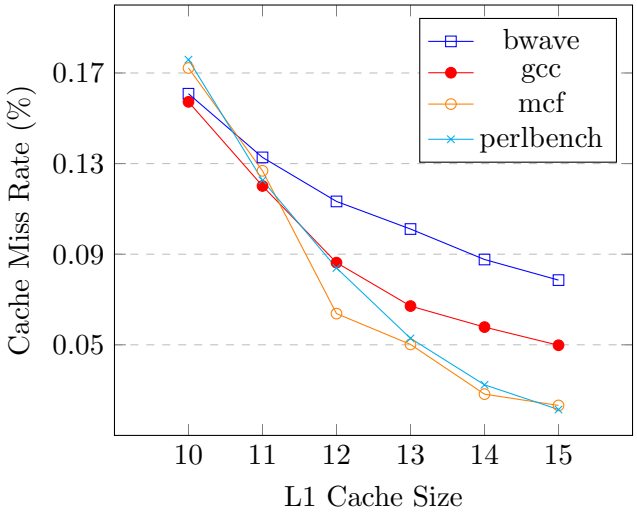

Figure 2: Cache Miss rate vs Cache Size on different traces

Branch Prediction One of the most crucial speculative part of the processor. Increasing the branch predictor table size should help branch predictor performance as it would be less likely to share branch

counters. However due to the specific implementation requirement that the counter is not updated using the GHR at the time of its prediction, the actual behavior cannot be entirely predicted. A sample experiment is conducted in the following. The figure suggest that except for gcc trace, a change in G could have little effect on the branch prediction accuracy.

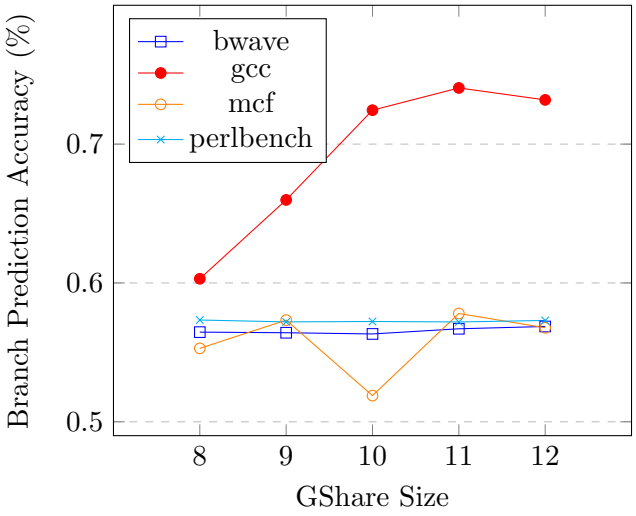

Figure 3: Branch Prediction Accuracy vs GShare size

Search for Optimal Configuration

The search for the optimal configuration would follow the general flow outlined in above.

mcf605_2M.trace

Following a general search flow, the absolute best performing bwaves603_2M.trace configuration is found to be

$$F = 4; k = 3; l = 3; m = 3; R = 4; G = 12; C = 14, IPC = 1.645092$$

In order to optimize the number of FU and reservation stations, 0.95 of this performance is 1.5628374. Leaving a total of 182 remaining configurations.

Two factors comes into play when optimizing the configurations: Number of Function units and the number of schedule units. Although Common Data Bus is not modeled in this assignment, the more the schedule units, the associative lookup path would have a higher delay, therefore the number of schedule units is considered to be more important in finding configuration. In the scope of this assignment, the minimization of schedule unit count would come first. In the event of a tie configuration, functional unit count would goes into consideration.

The optimal configuration with minimum number of schedule unit is:

$$F = 4; k = 3; l = 3; m = 3; R = 2; G = 12; C = 14, IPC = 1.574245$$

with only 18 schedule units.

bwaves603_2M.trace Optimum

The absolute best performing bwaves603_2M.trace configuration is found to be:

$$F = 4; k = 3; l = 3; m = 3; R = 4; G = 11; C = 15, IPC = 2.774749$$

95% of the performance would be 2.63601155, leaving 66 configurations. Minimizing schedule unit leaves 13 configurations that all have $k = 3, l = 3, m = 3, R = 3$. To further choose a less costly configuration, minimizing cache size leaves two configurations at approximately 25KiB:

$$k = 3; l = 3; m = 3; R = 3; G = 10; C = 14; IPC = \begin{cases} 2.641003 & F = 3 \\ 2.63931 & F = 4 \end{cases}$$

Given $F = 3$ out-performs $F = 4$ already, the optimal configuration is:

$$F = 3; k = 3; l = 3; m = 3; R = 3; G = 10; C = 14; IPC = 2.641003$$

perlbench600_2M.trace

The absolute best performing perlbench600_2M.trace configuration is found to be:

$$F = 3; k = 3; l = 3; m = 3; R = 4; G = 9; C = 15, IPC = 1.752853$$

95% of the performance would be 1.66521035, leaving 105 configurations. Minimizing schedule unit leaves 8 configurations that all have $k = 3, l = 2, m = 3, R = 3$. To further choose a less costly configuration, minimizing cache size leaves two configurations at approximately 33.92KiB:

$$k = 3; l = 2; m = 3; R = 3; G = 8; C = 15; IPC = \begin{cases} 1.68065 & F = 2 \\ 1.690074 & F = 3 \end{cases}$$

Although $F = 3$ yields a better result, it is merely improving by 0.56%, which is not worth the extra resources put into one more prefetch unit. Therefore the optimal configruation is chosen at

$$F = 2; k = 3; l = 2; m = 3; R = 3; G = 8; C = 15, IPC = 1.68065$$

gcc602_2M.trace Optimum

The absolute best performing perlbench600_2M.trace configuration is found to be:

$$F = 2; k = 3; l = 1; m = 3; R = 4; G = 10; C = 15, IPC = 1.724878$$

95% of the performance would be 1.6386341, leaving 169 configurations. Minimizing schedule unit leaves 7 configurations that all have $k = 3, l = 1, m = 3$. To further choose a less costly configuration, minimizing cache size leaves two configurations at approximately 33.92KiB:

$$k = 3; l = 1; m = 3; R = 3; G = 11; C = 14; IPC = \begin{cases} 1.663313 & F = 2 \\ 1.638992 & F = 3 \end{cases}$$

Consider $F = 2$ outperforms $F = 3$ also has a smaller dispatch rate, the optimal configuration is

$$F = 2; k = 3; l = 1; m = 3; R = 3; G = 11; C = 14; IPC = 1.663313$$

Summary

Trace	F	k	l	m	r	G	C	IPC
bwave	4	3	3	3	4	11	15	2.774749
gcc	2	3	1	3	4	10	15	1.724878
mcf	4	3	3	3	4	12	14	1.645092
perlbench	3	3	3	3	4	9	15	1.752853

Table 1: Absolute Best Performing Configuration

Trace	F	k	l	m	r	G	C	IPC
bwave	3	3	3	3	3	10	14	2.641003
gcc	2	3	1	3	3	11	14	1.663313
mcf	4	3	3	3	2	12	14	1.574245
perlbench	2	3	2	3	3	8	15	1.68065

Table 2: Optimal Configuration