A Novel Branch Predictor Using Local History for Miss-Prediction Bias

Lin Meng¹, Katsuhiro Yamazaki¹, and Shigeru Oyanagi²
¹College of Science and Engineering, Ritsumeikan University, Kusatsu, Shiga, Japan
²College of Information Science and Engineering, Ritsumeikan University, Kusatsu, Shiga, Japan

Abstract—Increasing accuracy of branch prediction is important for enhancing the performance of current processors. This paper discusses an interesting behavior of a current branch predictor that a large rate of miss-predictions is occupied by a few branches. We propose a novel branch prediction mechanism by using local history for miss-prediction biased branches. This mechanism is attached to a conventional branch predictor, and utilizes local history of biased branch instructions without conflict aliasing. Experiments are done by attaching proposed mechanism to several predictors. The results show that the proposed mechanism reduces the miss-predictions about 10%, and increases performance about 2% comparing to the conventional predictors at SPECint2000.

Keywords: branch prediction; miss-prediction; miss-prediction bias;

1. Introduction

Current processors use deeper pipeline and wider instruction issue width for exploiting instruction level parallelism. However when a branch miss-prediction happens, it causes heavy miss-prediction penalty, such as wasting large number of cycles and power for miss-prediction recovery [1]. Hence, increasing accuracy of branch prediction is more important for improving the superscalar processor performance.

In 1981, the first branch predictor named Bimodal predictor [2] was proposed. Since then, many branch predictors have been proposed for increasing accuracy of branch predictions. Bimodal predictor used a 2 bits saturating counter to keep the behavior of branch. These 2 bits saturating counters configure the PHT (Pattern History Table) which is indexed by the lower branch instruction address bits. Gshare predictor[3] is widely used in current processors. Gshare predictor uses the exclusive OR of GBH (global branch history) and branch address to make the PHT index.

One of the main reasons of miss-predictions is conflict aliasing, which is caused by the different branches accessing the same PHT entry. Major proposed predictors use global history to utilize the correlation among recently executed branches. Despite the steady improvements that have been made, it is difficult to completely avoid conflict aliasing, hence many branches are still miss-predicted [10].

This paper analyzes the behavior of branch predictor, and focuses on the miss-predictions bias. Miss-prediction bias means that miss-predictions of a few branches occupy a large rate of all miss-predictions. We propose a novel branch prediction mechanism by utilizing miss-prediction bias. Proposed mechanism is attached to a conventional branch predictor, and solves the conflict aliasing by preparing different PHT entry for each miss-prediction biased branch and by utilizing local history of miss-prediction biased branches.

The rest of this paper is organized as follows. Section 2 reviews the current branch predictors. Section 3 shows the characteristics of miss-prediction bias by using conventional predictors. Section 4 explains our proposal which targets for miss-prediction biased branches. Section 5 describes the evaluation of our proposal by simulation on SimpleScalar Tool Set[15]. In this simulation, our mechanism is attached to Combining, Bimode, Bimode-Plus, Agree, Hybrid and TAGE predictors. Section 6 and 7 discuss conclusion and future research.

2. Related Work

Many current branch predictors can be explained by extending the base predictors to reduce miss-predictions. The widely used base predictors are Bimodal and Gshare predictors. Several predictors using this approach are shown as follows.

Combining[3]: Combining predictor consists of a Bimodal predictor which works well for local history and a Gshare predictor which works well for global history. A selector is constructed by 2 bits saturating counter to select the result from either Bimodal predictor or Gshare predictor.

Bimode[4]: Bimode predictor prepares two Gshare predictors, one for the branches biased toward Taken (Taken Gshare predictor), and the other for the branches biased toward NotTaken (NotTaken Gsahre predictor). Then, Bimode predictor uses ChoicePHT to choose the result from two Gshare predictors. Bimode predictor can reduce conflict aliasing by dividing biased branches into different PHTs.

Bimode-Plus[7]: Some branches are strongly biased toward one direction (Taken or NotTaken) until the program finishes. Bimode-Plus predictor provides a Bias Table which keeps Taken bit and NotTaken bit to detect the branches which are strongly biased toward Taken or NotTaken. When the strongly biased branches are detected, the predictor uses...
Table 1: Processor configuration

| Pipeline       | 5 stages: 1 Fetch, 1 Decode, 1 Execute, 1 Memory Access, 1 Commit |
| Fetch,Decode   | 4 instructions |
| Issue          | Int: 4, fp: 2, mem: 2 |
| Window         | Dispatch queue: 256, Issue queue: 256 |
| BTB            | 2K-entry 4-way associative BTB, 32-entry RAS |
| Memory         | 64KB, 4-way associative, 1-cycle instruction and date caches, 2MB, 8-way associative, 10-cycle L2 |

Bias Table without using the result of Bimode predictor nor updating into Bimode Predictor. By this way, Bimode-Plus predictor reduces the conflict aliasing between the strongly biased branches and normal branches.

Agree[6]: Agree predictor keeps the Taken biased bit and NotTaken biased bit in BTB. PHT keeps the result whether the prediction is same to the biased bit. When the branch result is same to the biased bit, the entry of PHT is incremented, otherwise it is decremented. Exclusive OR of the biased bit and the PHT result is used for prediction.

Hybrid[5]: Hybrid predictor keeps several predictors (2bc, Gshare, GAS, AVG) to predict the branch. BTB keeps the result of each predictor, and is used at prediction to select the most accurate one among the several predictors.

TAGEL-TAGE[8], [9]: These predictors use PPM (prediction by partial matching) to search the patterns in the branch direction history[11]. This method has 4 PHTs which are accessed by the exclusive OR of different parts of GBHR and branch address. Every PHT entry has a tag (a part of branch address) to protect the conflict. Hence, this method uses the pattern of GBHR to improve the prediction accuracy.

3. Miss-Prediction Bias

We observe that miss-predictions of a few branches occupy a large rate of total miss-predictions on conventional predictors. This section shows the behavior of miss-prediction bias. Simulation is performed by using Bimode predictor on SimpleScalar Tool Set. Table 1 shows the processor configuration. Instruction set is PISA and benchmarks are bzip, gcc, gzip, mcf, parser, twolf, vpr and vortex from SPECint2000.

In experiments, top 8 and 16 miss-prediction branches are observed. We call these branches miss-prediction biased branches. Figure 1 shows the miss-prediction rate of top 8 and 16 miss-prediction biased branches to total miss-predictions. The experiments run 100M instructions ranged by 20M. The predictors size is set to 8KB, 16KB and 32KB.

In Figure 1, the horizontal axis means the executed instructions, and the vertical axis represents the rate of miss-predictions at the miss-prediction biased branches to total miss-predictions.

Clearly, Figure 1 shows that miss-predictions at top 8 miss-prediction biased branches occupy more than 70% of total miss-predictions, and those of top 16 miss-prediction biased branches occupy more than 80% of total miss-predictions.

4. Proposed Predictor Attached on a Conventional Branch Predictor

The behavior of miss-prediction biased branches has been demonstrated in the last section. It means that a small number of branches cause most of miss-predictions. This section proposes a novel predictor to utilize this behavior. One of the major reasons for miss-predictions of many predictors is the conflict aliasing that the different branches accessing the same PHT entry. Our approach to solve this problem is to allocate different PHT entry for each miss-prediction biased branch, and to utilize local history for each miss-prediction biased branch without conflict aliasing. The proposed mechanism can be attached to any kind of conventional branch predictors.

Figure 2 shows a block diagram of our proposal. In this proposal, MBD (Miss Bias Detector) and LHBP (Local History Branch Predictor) are attached on a base predictor. MBD is used to detect miss-prediction biased branches. LHBP is used to predict the miss-prediction biased branches by using local history.

The prediction flows as follows. At first, miss-predictions of the base predictor are counted at the extended BTB (EBTB). When the count of miss-predictions exceeds the threshold, the branch is recognized as a miss-prediction biased branch, and its local history is stored into MBB (Miss Bias Buffer). Distinct LPHT entry is assigned for each miss-prediction biased branch to predict the branch direction based on the local history. At last, the Selector selects the result from either LHBP predictor or the base predictor as the final result.
4.1 Detection of Miss-Prediction Biased Branches by MBD

MBD is composed of EBTB and MBB. EBTB is used to detect miss-prediction biased branches. EBTB is an extended version of BTB by keeping a MCT (Miss Counter) on every entry. MCT is a saturating counter for keeping the miss-prediction count of the base predictor. EBTB also keeps a Tag and branch target address (T addr) as conventional BTB. MBB is composed of Addr, LH, U, and FR for keeping the information of miss-prediction biased branches. Addr keeps the address of miss-prediction biased branch, LH is a shift register for keeping the local history, U is a use bit for marking whether the MBB entry is used, and FR (Failure Rate) keeps the difference of miss-prediction count between LHBP and the base predictor.

Since the top 8 or 16 miss-prediction biased branches occupy most of total miss-predictions, MBB size is set to 8 or 16, and proposed predictor targets for top 8 or 16 miss-prediction biased branches.

The action of MBD includes registering the miss-prediction biased branches and updating the information of the miss-prediction biased branches. The action of MBD is explained as follows.

Registration of MBB: When the branch is committed, the branch address is associated in EBTB. If EBTB hits and miss-prediction occurs, EBTB’s MCT is incremented, else if EBTB misses, Tag is changed as conventional BTB and the MCT is set to 1. If MCT arrives at the threshold, the branch address is registered into MBB, and corresponding EBTB entry is reset.

At registering to MBB, the branch address is checked whether it is already existed in MBB. If the address is not existed in MBB, it is registered into the entry whose U bit is 0. Then, the corresponding U bit is set to 1. If all of MBB’s U bits are 1, the predictor uses LRU logic to search for the Least Recently Used entry, and registers the new branch address into the entry. The U bit is set to 1.

Update of MBB: When a branch is committed, the branch address is associated to MBB. If the branch address is found in MBB, the direction of the branch is shifted into LH. FR is used to denote whether the entry of LHBP is effective or not. FR is incremented when the predicted result of LHBP is incorrect and the predicted result of base predictor is correct. It is decremented when the predicted result of LHBP is correct and the predicted result of base predictor is incorrect. By this way, FR keeps the difference of miss-prediction count between LHBP and base predictor. When FR arrives at the threshold, it is recognized that the entry of LHBP is worse than base predictor. Then the LHBP entry is reset.

4.2 Prediction by LHBP

LHBP is a predictor targeted for the miss-prediction biased branches registered in MBB. Because the number of miss-prediction biased branches is small, LHBP uses local history of the miss-prediction biased branches for making PHT index. For avoiding the conflict among miss-prediction biased branches, distinct LPHT entry is prepared for each miss-prediction biased branch. Namely, LPHT entry is indexed by combining MBB entry address and its Local History. Hence, there is no conflict on the LPHT.

LPHT entry size is decided by the product of MBB entry size and local history length. Thus, when the local history length is n and the number of MBB entry is m, LPHT has \( m \times 2^n \) entries. Each entry of LPHT consists of NTCT and CF. NTCT is a 2 bits saturating counter for keeping the behavior of the branch. NTCT is incremented when the branch is Taken, and decremented when it is NotTaken. CF
5. Evaluation
5.1 Analysis of the Component Size

To evaluate our proposal, the optimum component size must be discussed.

Here we discuss MCT Length, MBB and LPHT entry size. MCT length is important for deciding the miss-prediction biased instruction. MBB entry size and LPHT entry size are important for increasing prediction accuracy. This section discusses the optimum component size by experiment. The experiment is performed on SimpleScalar Tool Set under the processor configuration given in Table 1. Benchmarks are bzip, gcc, gzip, mc, parser, twolf, vpr, vortex of SPECint2000 and drr, reed_dec, reed_enc, rtr, zip_enc of CommBench [16]. Instruction set is PISA.

MCT Length: MCT is a saturating counter for counting the miss of BTB entries to detect the miss-prediction biased branches. For determining the MCT size, experiment is performed on the organization of Bimode Predictor as a base predictor with 8 entries of MBB and 8K entries of LPHT by ranging MCT length from 4 bits to 8 bits.

Figure 3 shows the rate of detected miss-prediction biased branches to total miss-predictions by varying the MCT length. Vertical axis is the average rate of miss detections. Figure 4 shows the reduction rate of miss-predictions. Vertical axis is the average reduction rate of miss-predictions. By analyzing Figures 3 and 4, about 70% of miss-predictions can be detected and about 10% of miss-predictions can be reduced by our proposal. Besides, the miss-prediction reduction rate does not change so much by varying the MCT length. Hence, 4 bits MCT length is used in the following experiments in order to minimize hardware costs.

MBB and LPHT Entry Size: LPHT entry size is defined by the product of number of MBB entry and local history length. Generally, larger MBB size can store more miss-prediction biased branches, and bring better performance. Larger LPHT entry size can store longer local history, and bring more accurate prediction. In this experiment, MBB and LPHT entry size is set to (8,4K) with 9 bits LH, (8,8K) with 10 bits LH, (16,8K) with 9 bits LH, and (16,16K) with 10 bits LH. Bimode Predictor are used as base predictors. Figure 5 shows miss-prediction reduction rate by attaching our proposal to the base predictors. Vertical axis shows the average reduction rate.

By analyzing Figure 5, the difference of miss-
5.2 Miss-Prediction Reduction Rate and IPC Performance

We measure the performance of our proposal on several base predictors (Combining, Bimode, Bimode-Plus, Agree, Hybrid, and TAGE predictor).

Figures 6 and 7 show the miss-prediction reduction rate of our proposal to the 6 kinds of base predictors sized of 8KB and 32KB. Horizontal axis shows the benchmark and vertical axis shows the miss-prediction reduction rate.

The results show that our proposal can reduce miss-predictions in 6 kinds predictors on average. When the base predictor size is 8KB, our proposal can reduce more than 7% of miss-predictions on average to the base predictors. When the base predictor size is 32KB, our proposal can reduce more than 5% of miss-predictions on average.

Figures 8 and 9 show the IPC performance enhancement of our proposal to the 6 kinds of base predictors sized of 8KB and 32KB. Horizontal axis shows the benchmark and vertical axis shows the IPC performance enhancement. The results show that our proposal can improve IPC about 2% on average to the base predictors.

In these experiments, we find Combining, Bimode, and Bimode-Plus predictors work similarly, because they use gshare and bimodal predictor. We find that there are several cases to prevent reducing the miss-predictions on our proposal. One is that some benchmarks have already good prediction on base predictor like reed_dec. In this case, our proposal can not further improve the prediction. Another is that the benchmark has a lot of miss-prediction branches like gcc. Our proposal does not improve because MBB size is not big enough to catch the miss-prediction bias branches correctly. When the branch predictor has several kinds of branch predictors like Hybrid Predictor, our proposal can not bring a large reduction of the miss-prediction. This is because Hybrid predictor can adopt to the branch behavior by changing the branch predictor. Our proposal can not catch the branch behavior so early.
6. Discussion

6.1 Factors for Miss-Prediction

We think that major factors for miss-predictions are random behavior, conflict aliasing and undetectable branch patterns.

In order to improve the prediction accuracy, improvement of the latter two factors is important. Our proposed predictor aims to reduce conflict aliasing by detecting miss-prediction biased branches and preparing dedicated predictor for them by using local history. A number of predictors have been proposed to reduce conflict aliasing as explained in chapter 2. But, these predictors do not utilize local histories of miss-predicted branches. The key issue of our proposal is to utilize local history of miss-prediction biased branches, which makes it possible to detect patterns in the local history. The effectiveness of our proposal is proved by the experiment that the longer local history brings better performance, and prediction accuracy is improved by attaching proposed mechanism to conventional predictors.

6.2 Discussion on Hardware Costs

This subsection discusses hardware cost of our proposal by measuring the required memory size. EBTB uses the current BTB’s association port. Hence it does not need to add any association port. Every entry in EBTB has additional 4 bits MCT and 2K entries, so additional hardware cost is only 1KB memory in BTB. MBB has 8 entries, every entry has 32 bits Addr, 9 bits LH, 1 bit U and 7 bits FR. So, MBB is a small CAM of 392 bits total. LPHT has 4K entries, every entry has 2 bits NTCT and 2 bits CF. So LPHT is a 2KB memory total. Hence, added hardware is small enough in our proposal.

Here we use MPKI (miss-predictions per kilo instructions) as a measure. Table 2 shows the MPKI of 6 kinds of base predictors (Combining, Bimode, Bimode-Plus, Agree, TAGE and Hybrid). The MPKI is the average of SPECint-2000 and CommBench, where the base predictor size is 10.5KB, 17.75KB, 30KB, and 60.5KB in Hybrid Predictor and the base predictor size is 8KB, 16KB, 32KB, and 64KB in other predictors.

Table 2 shows that our proposal can reduce MPKI than base predictors. In SPECint2000, our proposal attached on 8KB sized base predictor (Combining, Bimode, Bimode-Plus and Agree predictor) can achieve the same MPKI as 64KB sized base predictor. Experiment results show the same tendency in CommBench. And attaching proposed mechanism to smaller Hybrid and TAGE predictor can achieve the same MPKI as the larger predictor too.
7. Conclusion and Future Work

Improving branch prediction accuracy is important for the modern processors which exploit instruction level parallelism by deeper pipeline and wider instruction issue width. This paper proposed a novel branch predictor for improving prediction accuracy by utilizing the behavior of miss-prediction biased branches. This predictor detects the miss-prediction biased branches and predicts the branch direction by using a local history based predictor attached on the base predictor.

This proposal is evaluated by experiment on the SimpleScalar Toolset. Our proposal is attached to Combining, Bimode, Bimode-Plus, Agree, Hybrid, and TAGE predictors. Miss-prediction reduction rate to the base predictors is evaluated. The results of experiment show that our proposal reduces the miss-predictions about 10%, and increases performance about 2% at SPECint2000. It also reduces the miss-predictions about 7%, and increases performance about 1% at Combincbench.

Although our proposal reduces average miss-predictions to the conventional predictors, miss-predictions increases in some benchmarks. It must be detected and improved in the future work.

8. Acknowledgment

This work is supported in part by Scinics Co.,Ltd.

References


### Table 2: Result of miss-predictions per kilo instructions (Average)

<table>
<thead>
<tr>
<th>Predictor</th>
<th>Specint2000</th>
<th>CommBench</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>8KB</td>
<td>16KB</td>
</tr>
<tr>
<td>Combining</td>
<td>5.53</td>
<td>5.30</td>
</tr>
<tr>
<td>Combining+proposal</td>
<td>4.76</td>
<td>4.60</td>
</tr>
<tr>
<td>Bimode</td>
<td>5.40</td>
<td>5.14</td>
</tr>
<tr>
<td>Bimode+proposal</td>
<td>4.76</td>
<td>4.55</td>
</tr>
<tr>
<td>Bimode-Plus</td>
<td>5.36</td>
<td>5.12</td>
</tr>
<tr>
<td>Bimode-Plus+proposal</td>
<td>4.71</td>
<td>4.50</td>
</tr>
<tr>
<td>Agree</td>
<td>6.71</td>
<td>6.49</td>
</tr>
<tr>
<td>Agree+proposal</td>
<td>5.94</td>
<td>5.79</td>
</tr>
<tr>
<td>TAGE</td>
<td>4.86</td>
<td>4.34</td>
</tr>
<tr>
<td>TAGE+proposal</td>
<td>4.41</td>
<td>4.15</td>
</tr>
<tr>
<td></td>
<td>10.5KB</td>
<td>17.75KB</td>
</tr>
<tr>
<td>Hybrid</td>
<td>7.50</td>
<td>6.23</td>
</tr>
<tr>
<td>Hybrid+proposal</td>
<td>6.92</td>
<td>5.76</td>
</tr>
</tbody>
</table>

- **Table 2:** Result of miss-predictions per kilo instructions (Average)