The Bimode++ Branch Predictor

Kenji Kise, Takahiro Katagiri, Hiroki Honda, and Toshitsugu Yuba
Graduate School of Information Systems,
The University of Electro-Communications

1 Introduction

The deeper pipelines and broader issue width increase the number of on-the-fly instructions in processors. Therefore a mispredicted branch can result in substantial amounts of wasted work. In order to reduce the performance loss, accurate branch predictor is required. Smith[3] discussed the branch prediction strategies in 1981. Since them, many branch prediction schemes have been investigated in order to attain high performance.

We propose the Bimode++ branch predictor as an enhanced version of the Bimode branch predictor. Throughout execution, from the start to the end of a program, some branch instructions have the same result at all times. These branches are defined as extremely biased branches. The Bimode++ branch predictor is unique in predicting the output of an extremely biased branch with a simple hardware structure.

2 The bimode++ branch predictor

2.1 Bimode branch predictor

Our predictor is proposed as an enhanced version of the bimode predictor introduced by Lee and Mudge[2].

The organization of the bimode predictor is shown in figure 1. The bimode predictor is an attempt to replace destructive aliasing with neutral aliasing with three PHTs: choice PHT, taken PHT and untaken PHT. The taken PHT and the untaken PHT are referred to direction PHTs. The direction PHTs are indexed by the branch address xored with the global branch history(BHR). The choice PHT selects the one of direction PHTs to be used, and the two-bit saturating counter of the selected PHT makes a prediction.

2.2 Transient state of two-bit saturating counter

The two-bit saturating counter is a component used in branch predictors. State transition of a two-bit saturating counter is shown in figure 2. The four state of 3, 2, 1 and 0 indicate strongly taken, likely taken, likely untaken and strongly untaken, respectively.

The states close to the threshold (likely taken and likely untaken in a two-bit saturating counter) are defined as the transient state because the prediction in their state may change in a short period. We found that the prediction accuracy in the transient state is somewhat lower than the accuracy in the non-transient state.

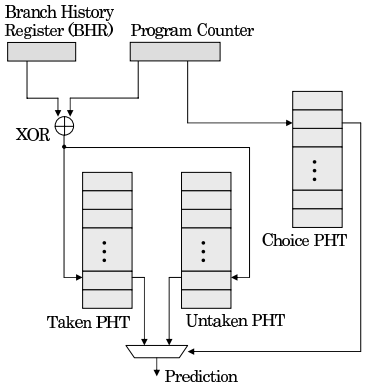

Figure 1: Bimode branch predictor.

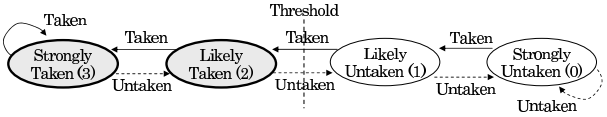

Figure 2: Two-bit saturating up-down counter.

We propose a mechanism that uses a fusion function when the branch in the transient state is predicted. The inputs of the fusion function is the outputs of choice PHT, taken PHT and untaken PHT. One promising candidate of a fusion function is a majority vote. The branch predictor with the mechanism (Bimode-Transient predictor) is shown in figure 3.

2.3 Indexing of the choice PHT

In the bimode predictor, the index of the choice PHT is generated using the program counter. We found that the use of the "Exclusive OR" function of the global branch history and the program counter to generate the index into choice PHT improves the prediction accuracy.

Since a two-bit saturating counter in the choice PHT catches the behavior of a static branch, the use of only a few BHR bits for the choice PHT index is sufficient to improve the accuracy.

2.4 Extremely biased branches

In execution of a program from start to finish, some branch instructions have the same result at all times. The branch instruction to detect a program error is one

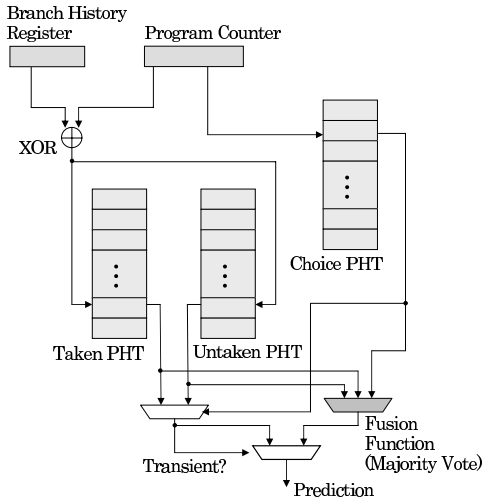

Figure 3: Bimode-Transient branch predictor.

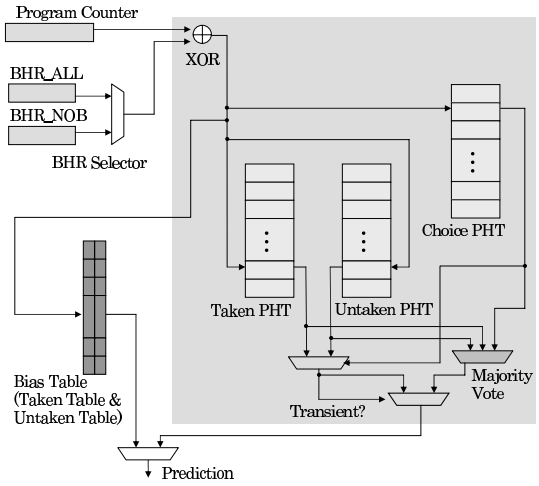

Figure 5: Bimode++ branch predictor.

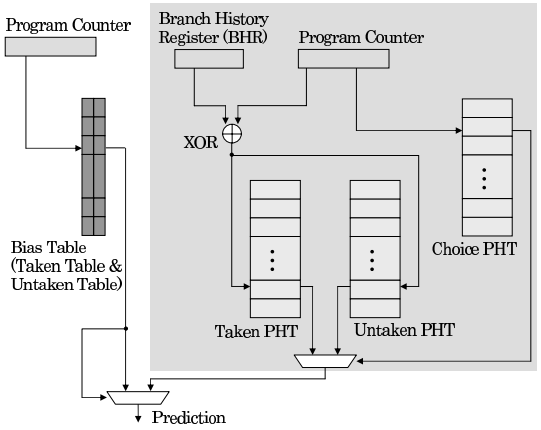

Figure 4: Bimode-Plus branch predictor.

of such example. In most cases, no errors occur and the branch result is always untaken.

We define branches that has the same result from start to the prediction as extremely biased branches. Here, we propose the Bimode-Plus predictor[1], which uses the property of extremely biased branches. In figure 4, the block diagram of the Bimode-Plus predictor is shown.

The shaded area is the bimode predictor. We use the taken and untaken tables (each entry is a one-bit flag) to record whether the branch is an extremely biased branch. The taken and untaken tables are referred to the bias table.

First, all entries of the bias table are initialized to zero. Then, when the branch outcome is known to be taken, the entry of the untaken table is updated with the value of one. When the branch outcome is known to be untaken, the entry of the taken table is updated with the value of one.

If the untaken table entry of the branch has the value of zero, then the branch may be an extremely biased untaken branch. If so, the predictor makes the prediction of untaken. If the taken table entry of the branch has the value of zero, then the branch may be an extremely biased taken branch and the predictor makes the prediction of taken ; otherwise, the outcome of the bimode predictor is used as a prediction.

2.4.1 Bimode-Plus with two BHRs

Determining the updating policy of the branch history register is difficult.

Here, we propose to use two BHRs updated using different policies and select an appropriate BHR from among them. One BHR, denoted as BHR_ALL, is updated with the results of all conditional branch instructions. This policy is similar to that used in the traditional bimode predictor. The other BHR, denoted BHR_NOB, is updated using the results of non-biased conditional branches. The BHR selector controls the selection of either BHR_ALL or BHR_NOB. These registers and the selector are shown at the upper-left of figure 5.

The appropriate algorithm and implementation of the BHR selector is another difficult issue. By the preliminary evaluation, we found that BHR_ALL is suitable for a benchmark having many extremely biased branches (such as server benchmarks). In order to detect the number of executed extremely biased branches, we use the saturating counter denoted as **bias_mod_cnt**, which is incremented when the content of the bias table entry is modified. If the counter has the maximum value and saturation, the BHR_ALL is selected.

In addition, the BHR selector uses a flag denoted as bhr_policy. This flag is determined after some training

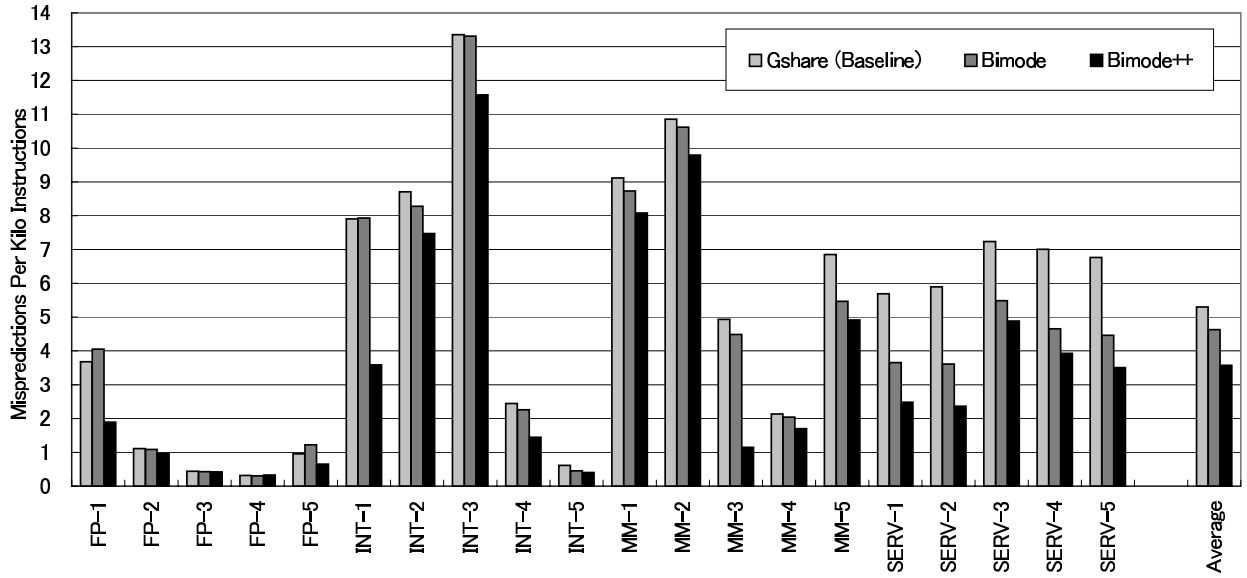

Figure 6: Performance results (Mispredictions Per Kilo-instructions). The data is obtained with the common evaluation framework version 3 for the championship branch prediction.

time with two saturating counters. The **cond_cnt** is incremented when a conditional branch is executed. The **biased_cnt** is incremented when an extremely biased branch is executed.

2.5 The bimode++ branch predictor

A block diagram of the bimode++ predictor is shown in figure 5. All tables are indexed with the program counter and the BHR. The bimode++ predictor is a combination of techniques discussed in sections 2.2, 2.3 and 2.5.

3 Prediction accuracy

The prediction accuracy is evaluated using the common evaluation framework version 3 for the championship branch prediction sponsored by Intel MRL and IEEE TC-uARCH. The framework contains 20 benchmark traces classified into 4 categories. The categories are SPECfp(FP), SPECint(INT), Multimedia(MM) and Server(SERV). The predictor parameters are optimized within the 8KB + 256 bit hardware budget.

In figure 6, the mispredictions per kilo-instructions (MPKI) of gshare (baseline), bimode, and bimode++ are summarized. On average, the baseline MPKI is 5.30. The MPKI of bimode predictor is 4.62, and bimode++ predictor achieves the MPKI of 3.57. The bimode++ predictor can reduce the mispredict rate by 32.5% to the gshare predictor [1].

[1] Although the bimode++ predictor was the 9th place in the championship branch contest, it has the advantage that structure is simple and realistic.

4 Summary

Accurate branch prediction has come to play an important role for modern microprocessors. In order to improve its prediction accuracy, many branch predictors have been investigated in the past few decades.

We proposed the bimode++ branch predictor as an enhanced version of the bimode branch predictor. Throughout execution, from the start to the end of a program, some branch instructions have the same result at all times. These branches are defined as extremely biased branches. The bimode++ branch predictor is unique in predicting the output of an extremely biased branch with a simple hardware structure.

Our experimental results with benchmarks from SpecFP, SpecINT, multi-media and server area showed that the bimode++ predictor can reduce the mispredict rate by 32.5% to the gshare predictor

References

[1] Kenji Kise, Takahiro Katagiri, Hiroki Honda, and Toshitsugu Yuba. The Bimode-Plus Branch Predictor. In *IEICE Technical Report CPSY-2003-10*, pages 25–30, 2003.

[2] Chih-Chieh Lee, I-Cheng K. Chen, and Trevor N. Mudge. The bi-mode branch predictor. In *Proceedings of the 30th Annual International Symposium on Microarchitecture*, pages 4–13, 1997.

[3] James E. Smith. A study of branch prediction strategies. In *Conference proceedings of the eighth annual symposium on Computer Architecture*, pages 135–148, 1981.