Enhanced Local Branch Predictor Design: Detailed Methodology, Storage Overhead, and Trade-offs Analysis

Shengyi Wei, Ying Li
Univerisity of California, Los Angeles
shengyiwei@ucla.edu, ying.li@ucla.edu

***Abstract*—In this paper, we propose a novel branch prediction mechanism that employs both a bimodal table and a pattern history table (PHT), combined via a Branch History Register (BHR) to enhance prediction accuracy for conditional branches. The design aims to optimize the trade-off between prediction accuracy and storage overhead, focusing on the efficient use of branch history and pattern-based information for prediction. This configuration achieves a misprediction rate of 5.020 misses per thousand instructions (MPKI) on the CBP-2 training traces.**

***Index Terms*—Branch Predictor, Branch History Register, Pattern History Table, Bimodal Table**

I. Introduction

Branch prediction is an essential mechanism within the architecture of modern superscalar processors, aimed at mitigating control hazards associated with branch instructions. Inaccurate predictions can lead to significant performance penalties due to pipeline stalls and the necessity to flush instructions, thus degrading system efficiency and increasing execution times [1], [2].

In this study, we have developed a branch predictor that combines a Branch History Register Table and a Pattern Table to improve the accuracy of predicting the direction of conditional branches. Within our design framework, the `my_predictor` class, a bimodal predictor is integrated with a local history-based predictor. This combination leverages both global and local branch history patterns to enhance prediction accuracy.

Our design incorporates a storage-efficient architecture, utilizing unlimited storage, achieves a miss prediction rate of 5.020 misses per thousand instructions (MPKI) on the CBP-2 training traces. This performance metric underscores the efficacy of our approach in a realistic simulation environment.

II. Design Methodology

A. Predictor Components

1) Bimodal Predictor: This component utilizes a direct-mapped table, BimodalTable, indexed by the branch instruction's address bits. Each table entry consists of a saturating counter that predicts the likelihood of branch execution (taken or not taken) based on historical outcomes.
2) Local History Predictor: This component utilizes a Branch History Register (BHR), this predictor captures the execution outcomes of the most recent branches. The Pattern Table, indexed by combinations of branch addresses and BHR values, stores saturating counters that provide predictions based on observed patterns in branch behavior.

B. Predictor Integration

The predictor evaluates the outputs from both the bimodal and local predictors. The decision logic prioritizes the local history prediction when its confidence level (as determined by the saturating counter value) is high. In cases of low confidence, the bimodal predictor's outcome is used. This integration allows the predictor to dynamically adapt to varying branch behaviors, enhancing overall prediction reliability.

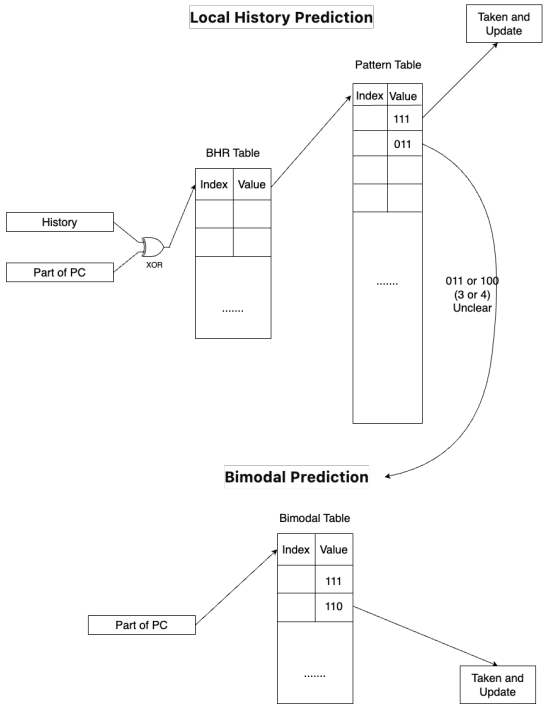

Fig. 1. Architecture of Branch Predictor

III. Storage Overhead Analysis

The design's storage requirements are as follows:

- Bimodal Table: 2^{15} entries, each 3 bits wide, amounting to 96 KB.
- Pattern History Table (PHT): 2^{22} entries, each 3 bits, totaling approximately 12 MB.
- Branch History Table (BHR): 22 entries, each 15 bits, totaling 330 bits.

These numbers reflect a significant consideration of storage versus performance, where larger tables typically offer better accuracy but at increased silicon and power costs.

IV. Trade-offs Consideration

A. Accuracy v.s. Storage

The design confronts the classic trade-off between accuracy and storage. The extensive use of a large PHT aims to maximize accuracy but incurs significant memory overhead. Conversely, the smaller BHR and bimodal table help limit the storage requirements while providing a baseline accuracy that is competitive.

B. Computational Complexity

The predictor's performance is also influenced by its computational complexity, primarily dictated by the hash functions used for index-

ing and the logic to manage multiple prediction mechanisms. The implementation ensures that these computations are efficient enough to maintain high clock speeds in processor pipelines.

C. Adaptability

The dual mechanism allows the predictor to be highly adaptable to various application behaviors, dynamically adjusting its predictions based on recent execution histories and long-term patterns, thus effectively managing the inherent variability in program control flows.

V. Conclusion

The `my_predictor` branch predictor represents a sophisticated approach designed to address the nuanced demands of contemporary CPU architectures. By balancing the trade-offs between accuracy, storage overhead, and computational complexity, this design provides a robust solution aimed at reducing branch misprediction rates and thereby enhancing overall processor performance. Future work will focus on further optimizing these trade-offs through algorithmic advancements and hardware implementation techniques.

References

[1] James E Smith. A study of branch prediction strategies. In *25 years of the international symposia on Computer architecture (selected papers)*, pages 202–215, 1998.

[2] Tse-Yu Yeh and Yale N Patt. Alternative implementations of two-level adaptive branch prediction. *ACM SIGARCH Computer Architecture News*, 20(2):124–134, 1992.