



Branch Prediction Techniques Used in Pipeline Processors: A Review

Sweety¹ and Prachi Chaudhary²

Department of Electronics and Communication Engineering
Deenbandu Chhotu Ram University of Science and Technology, Murthal, Sonipat
{sweety¹nain28, dahiya.prachi²}@gmail.com

Abstract. The ability to predict the direction of branches is an important issue in modern computer architecture and advanced compilers. Today, all state-of-the-art microprocessors have branch prediction of static (software) and dynamic (hardware). This paper surveys the different techniques used for branch prediction. According to the survey, dynamic branch prediction provides better accuracy.

Keywords: Microprocessor, Branch prediction, Pipeline.

1 Introduction

High performance is an important issue in microprocessor and its importance is exponentially increasing over the years. To improve the performance, two alternative methods exist (a) To improve the hardware by providing faster circuit (b) To arrange the hardware, so that multi-operations can be performed [1].

On the basis of performance, pipelining is a process of arrangement of hardware elements of the CPU such that its overall performance is increased, Simultaneous execution of more than one instruction takes place in a pipelined processor. In an exceedingly parallel system, conditional instructions break the continuous flow of programs or decrease the performance of the pipelined processor, which causes delay. To decrease the delay prediction of branch direction is necessary. The disparity in the branches needs accurately branch prediction strategies [2]. So, branch prediction is a vital part of the present pipelined processor.

In this paper, Summary of various schemes of branch prediction (BP) is presented. The paper is well thought-out into four sections. In section 2, techniques of static and dynamic schemes are presented. Section 3 outlines the case study of branch prediction. Section 4 describes the comparison of some schemes of dynamic prediction while Section 5 followed by the conclusion.

2 Branch Prediction and its Techniques

Branch prediction (BP) is one of an earliest executions methods that still discover the importance of modern architecture [3]. Recently, branch prediction (BP) has prompted to the advancement of branch prediction techniques that accomplish better result and accuracy.

Basically branch prediction predicts two problems: 1) direction predicting, and 2) calculating the target address. Branch prediction schemes are of two types: static branch schemes and dynamic branch schemes.

A static branch scheme (software techniques) is very simple and easy. This scheme assembles the majority of the data/information prior to the execution of the program or during the compile time and it doesn't require any hardware whereas, a dynamic branch scheme (hardware techniques) is based on the hardware and it assembles the information during the run-time of the program. Dynamic schemes are more assorted as they keep track during run-time of the program execution [4].

2.1 Techniques for Static Branch Prediction

Static prediction techniques are very simple to analyze and require low cost [5] and less energy to execute instructions because it does not require any history table of instructions as well as hardware component [6]. In most of the system, compilers can provide good coverage for such types of branches.

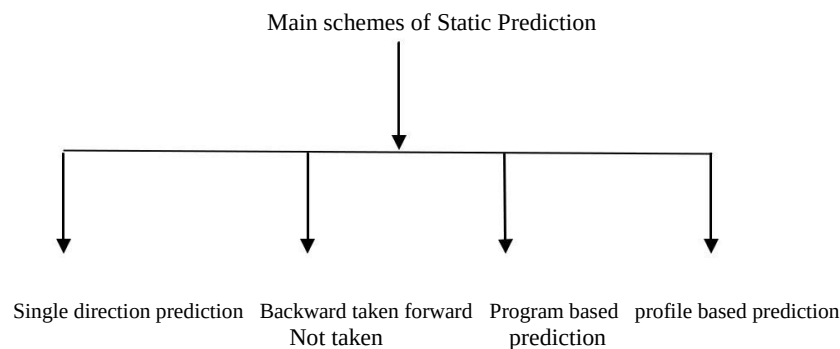


Fig. 1. Static Prediction Schemes

- **Single direction prediction:** Single direction is the easiest strategy in static prediction schemes. In this prediction, the directions of all branches will dependably go in a similar way, regardless of whether the branch prediction took or not. In this prediction, when the branch prediction is taken it gives a better result as compared to the not taken prediction [7].

- **Backward taken forward not taken (BTFT):** In branch taken forward not taken schemes, most of the loops are backward jump loops and which will be taken more often than forwarding loops. This will upgrade the execution of the prediction [8].
- **Program based prediction:** This prediction use structural based information of the program. A distinguished case of this prediction is portrayed by the Ball *et al.* [9] and Calder *et. al* [10]. A set of heuristics presents by Ball *et al.* [9] are based on the operands, opcode, and information of executed branches
- **Profile-based branch prediction:** This branch prediction utilizes information from the previous execution of the program with different inputs [9,10].

2.2 Techniques for dynamic branch prediction

A modern processor relies on dynamic branch prediction strategies, as this prediction accumulated data during run-time or execution of an instruction. This prediction requires hardware to save some past data of branches and utilizes that data for the prediction of the branches [3,11].

Basically, this prediction based on two interacting method:

- Branch outcome prediction
- Target prediction

Branch outcome prediction: It alludes to foresee the branch direction whether it should be taken / not taken.

Branch target prediction: This prediction predicts the target when any branch prediction is taken.

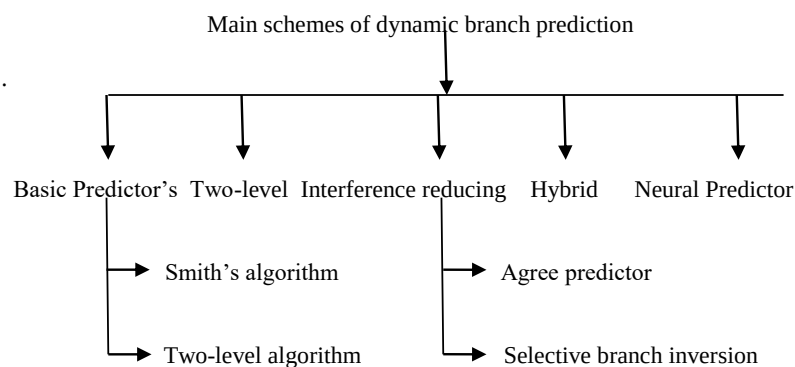


Fig.2: Dynamic prediction schemes

Advanced dynamic prediction algorithms used today are derived from one or more of these algorithms and various branch mechanisms are considered for improvement.

2.2.1 Basic predictor's: Basic predictors include earlier predictors that were used in dynamic branch prediction such as Smith algorithm and two-level algorithm.

- **Smith algorithm:** This algorithm is the simplest and most primitive proposed algorithms used as a dynamic branch prediction. According to Smith [12], single level predictor consists of a branch history table that keeps a record for each branch whether the previous branches were taken or not. It comprises a table of 2^m counters, where every counter tracks the past directions.
- **Two level algorithm:** Yeh *et. al.* proposed the two-level algorithm[13]. This approach gives a better improvement in prediction accuracy as compared to Smith algorithms because it requires two level of storage. a) History registers and b) pattern table. History register (HR) maintains the information of last m branches.

2.2.2 Two –level prediction table: The two-level predictor defines two separate tables to analyze the prediction of the branch instructions such as local and pattern history table. The Local table contains the address of the lowest bits of every branch instruction and records the history of m most current execution of the branch. Pattern history table has four entries of each branch and suitable entries are chosen by the saturating counter [14].

2.2.3 Interference reducing predictors: Interference refers as aliasing. Aliasing happens when a pair of address and history maps with a similar entry in pattern table or we can say that when any two random branches are utilizing the same prediction [15]. Aliasing has been classified as destructive and constructive/harmless. Destructive aliasing is considerably more common than constructive aliasing.

- **Agree Predictor:** Agree predictor reduces the interference problem in destructive aliasing within the pattern history table (PHT). It makes a preliminary prediction for each branch. In this technique, a predictor makes a prediction and predicts the agree/disagree statement with the help of base prediction. The main focus of this predictor is to decrease the negative aliasing or change the negative aliasing to a positive or neutral aliasing [4,15].
- **Selective branch inversion (confidence estimation):** Selective branch inversion enhances the execution of branches by inverting the branch prediction when the result is to be low-confidence. In the selective branch inversion, one of the conventional branch predictor (G-share, bi-mode) is operated parallelly with a confidence estimator. Now each of the predictor value will be compared with their threshold value to predict the taken / not taken condition and in last, if the value of confidence counter (CC) gets greater than or equal to the threshold value then branch should be labeled as “high confidence” or vice-versa [16].

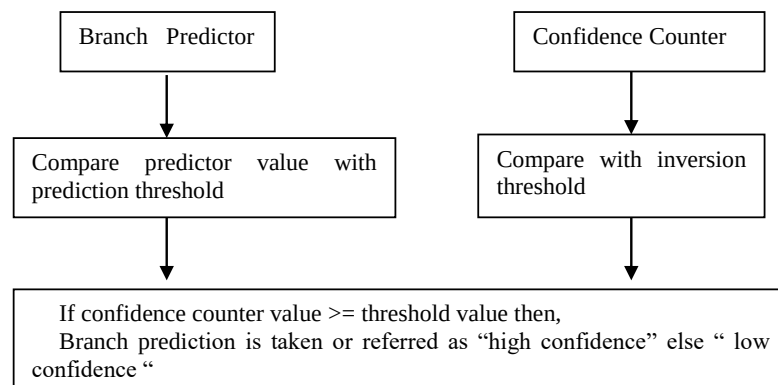


Fig.3: Selective branch inversion

2.2.4 Hybrid branch predictors: Mcfarling *et al.* [17] implement the idea of the hybrid. This predictor implements more than one prediction mechanism and final prediction is based on the predictors that made the best prediction in the past. There are some pros and cons of a hybrid predictor. Pros are a) different predictors give a better outcome for different branches b) by using more than one predictor simultaneously, time can be reduced. Cons are a) it needs selection counter b) longer access latency.

2.2.5 Neural predictor: The neural predictor has been well chosen to evaluate the performance of branch prediction in many research paper [18]. Neural predictor could make organized under different models, relying upon the number of layers, a number of neurons in each layer and neuron connection types. Advanced neural methods like Elman network, radial basis network and learning vector quantization (LVQ) ought to be analyzed and enforced to find the better accuracy and less miss-prediction rate in branch prediction.

2.2.6 Worm-hole predictor: Warm-hole (WH) predictor is recently introduced and this one is the first predictor that follows the relationship of some branches in the multi-dimensional loop [19]. These branches will be predicted by utilizing the outcomes of the branch itself in neighbor iterations of the inner loop. This predictor needs four stages for the prediction of branches: 1) Recognize conditional instructions that are miss-predicted by the predictor. 2) Identify the spatial property of the present loop and counts when the loop exists. 3) Branch history is recorded 4) In the last step, multidimensional patterns will be checked with-in the local history area.

3 Related work

This section describes the background works of branch prediction in pipelines. Branch prediction in pipelines has been broadly investigated and many prediction approaches have been proposed.

In static branch prediction, each instruction is expected to be taken or untaken during compile time. The earliest static branch predictors use single direction prediction.

Static predictors additionally use BTFN (branch taken, forward not taken) and profile information. The profile empirical approach runs a program with totally different input file sets and computes the frequency of every branch and optimizes the result of every branch with the next frequency [7, 8]. The static prediction approaches aim is to minimize the average misprediction rate, however recent WCET-aware static prediction [20] approaches gives a better outcome. This approach finds the longest path of a program, selects a branch on the longest path, and predicts in a way so that the longest path is reduced.

Dynamic prediction techniques predict a prediction for each branch at run-time to get higher prediction accuracy by considering its execution history. The single-level predictor uses a branch history table to trace the history of the branches and this will do by local dynamic branch predictor. For every branch, one-bit or two-bit saturating counter predicts, whether the branch prediction has been taken or not.

The two-level predictor uses the history information of local and global branch table to predict the branches. Basically, it saves the history of the last m occurrences of every branch and uses the saturating counter. This technique performs an artificial modification in the history register to enhance the accuracy of branch prediction and might be employed in completely different prediction approaches to enhance the prediction accuracy.

Hybrid branch predictor proposed for the higher prediction accuracy. This predictor combines several prediction schemes into one predictor and proposes a selection mechanism that uses the combination of branch predictors by using two-bit saturating counters that track the predictor which provides high accuracy for every branch. Hybrid predictors will develop the strengths of all the component of predictors, leading to higher prediction accuracy. However, the latency is that the major drawbacks for this hybrid predictors. WCET (worst-case execution time) approach is a new technique for a real-time embedded system that reducing the number of branch mispredictions and gives better accuracy as compared to previous techniques.

4 Comparison of Different Two-level Dynamic Prediction Schemes

Detailed information evaluation methodology and comparisons of some schemes are given in [16]. Below table 1 represents the performance of schemes.

Table 1. Performance of different 2-level schemes

Parameters	Branch prediction schemes		
	Global history	Per-set history	Per-address history
Cost implementation	more	highest	less
Size of history register	large	optimal	Needs entries
Cost-effectiveness	More	less	-----
remarks	If-else-then-else	-----	loops

4 Conclusion

This paper has covered the static and dynamic branch prediction strategies. It had been found that static schemes are less accurate as compared to the dynamic schemes. Static schemes are good for simple processors whereas dynamic schemes give high accuracy rate due to additional uses of hardware components.

References

- [1] I. East, *Computer Organization and Architecture*. 2004.
- [2] R. Thomas, M. Franklin, C. Wilkerson, and J. Stark, "Improving branch prediction by dynamic dataflow-based identification of correlated branches from a large global history," *Proc. 30th Annu. Int. Symp. Comput. Archit.*, vol. 31, no. 2, pp. 314–323, 2003.
- [3] K. Thangarajan, W. Mahmoud, E. Ososanya, and P. Balaji, "Survey of branch prediction schemes for pipelined processors," in *Proceedings of the Annual Southeastern Symposium on System Theory*, 2002, pp. 324–328.
- [4] V. R. Kulkarni, Ketan N and Mekala, "A Review of Branch Prediction Schemes and a Study of Branch Predictors in Modern Microprocessors." 2016.
- [5] M. Evers and T. Y. Yeh, "Understanding branches and designing branch predictors for high-performance microprocessors," *Proc. IEEE*, vol. 89, no. 11, pp. 1610–1620, 2001.
- [6] P. Chakraborty and S. Gautam, "Compilation Techniques to Generate Energy-Conserving Codes for Embedded Systems: A Review," *IUP J. Electr. Electron. Eng.*, vol. 10, no. 3, 2017.
- [7] A. M. González, "A survey of branch techniques in pipelined processors," *Microprocess. Microprogramming*, vol. 36, no. 5, pp. 243–257, 1993.
- [8] B. C. Gordiya and T. Shah, "Implementation of Backward Taken and Forward Not Taken Prediction Techniques in SimpleScalar," *Int. J. Adv. Res. Comput. Sci. Softw. Eng.*, vol. 3, no. 6, pp. 859–863, 2013.
- [9] J. R. Ball, Thomas and Larus, *Branch prediction for free*, no. 6. 1993.
- [10] B. Calder, D. Grunwald, M. Jones, J. Martin, M. Mozer, and B. Zorn, "Evidence-Based Static Branch Prediction Using Machine Learning," *ACM Trans. Program. Lang. Syst.*, vol. 19, no. 1, pp. 188–222, 1997.
- [11] L. Wang, Yu and Chen, "Dynamic Branch Prediction Using Machine Learning," 2005.
- [12] J. E. Smith, "A study of branch prediction strategies," in *Proceedings of the 8th annual symposium on Computer Architecture*, 1981, pp. 135–148.
- [13] T.-Y. Yeh and Y. N. Patt, "Alternative implementations of two-level adaptive branch prediction," *Proc. 19th Annu. Int. Symp. Comput. Archit. - ISCA '92*, vol. 20, no. 2, pp. 124–134, 1992.
- [14] T.-Y. Y. T.-Y. Yeh and Y. N. Patt, "A Comparison Of Dynamic Branch

- Predictors That Use Two Levels Of Branch History,” *Proc. 20th Annu. Int. Symp. Comput. Archit.*, vol. 21, no. 2, pp. 257–266, 1993.
- [15] J. Patil, Harish and Emer, “Combining static and dynamic branch prediction to reduce destructive aliasing,” in *High-Performance Computer Architecture. HPCA-6. Proceedings. Sixth International Symposium on*, 2000, pp. 251–262.
- [16] S. Manne, A. Klauser, and D. Grunwald, “Branch prediction using selective branch inversion,” in *International Conference on Parallel Architectures and Compilation Techniques (Cat. No.PR00425)*, 1999, pp. 48–56.
- [17] S. Mcfarling, “Combining Branch Predictors,” 1993.
- [18] V. M. Ribas, M. F. Figueiredo, and R. A. De Lara, “Simulating a simple neural network on branch prediction,” no. 2, pp. 153–160, 2003.
- [19] J. Albericio, “Wormhole branch prediction using multi-dimensional histories,” *JWAC-4 Championsh. Branch Predict.*, pp. 3–6, 2014.
- [20] F. Bodin and I. Puaut, “A WCET-oriented static branch prediction scheme for real time systems,” *Proc. - Euromicro Conf. Real-Time Syst.*, vol. 2005, no. 1, pp. 33–40, 2005.

