IOP Conf. Series: Journal of Physics: Conf. Series **1487** (2020) 012003 doi:10.1088/1742-6596/1487/1/012003

Design of Branch Predictor Based on Jaccard Coefficient

YICHENG Zhang

Nanjing Normal University, Nanjing, China

zhangyicheng_98@163.com

Abstract. Reasonable design of the branch predictor is one of the main factors that affect the performance of contemporary processors. In view of the fact that the high precision of existing single prediction algorithm is confined to the type of branch instructions, this paper proposes a composite branch prediction method with the combination of global predictor and local predictor. The branch selector based on Hamming distance and bitwise adder chooses the more rational prediction from local branch predictor and global branch predictor, leading to high-accuracy prediction about different types of branches. Results of functional simulation indicate that the composite branch predictor proposed in this paper can make highly precise prediction about various programs.

1. Introduction

There is an upper limit on the working frequency of modern processors, and the design of super pipeline, namely increasing pipeline stage appropriately while reducing the workload of each stage, is one of the most critical means to improve the performance of processors. However, super pipeline can result in pipeline stall with serve loss of power and performance in case of a branch misprediction. Enormous data show that there is one branch instruction in every 4 to 6 instructions in many programs related to integer applications. Therefore, the accuracy of branch prediction has become a vital factor that influences processor performance as the super pipeline is widely utilized in processor design.

Branch prediction can be divided into static branch prediction and dynamic branch prediction. Static branch prediction[1] is a mechanism that keeps making the same prediction of either taken or not taken, so its accuracy is limited to the type of branches and it cannot meet the requirements about high performance of processors. The basic implementation of dynamic branch prediction is one-level branch prediction[2], among which the 2-bit saturating counter is the most widely used. However, unsatisfactory performance occurs in case of certain patterns like an array of alternating taken and not taken, since branch history is not recorded and fully considered. Tse-Yu Yeh and Yale N. Patt proposed two-level adaptive branch prediction[3] to improve accuracy by making use of history. This method can be divided into 9 combinations depending on connection modes between branch history table and pattern history table, which implies mutual check between hardware cost and accuracy. For example, the GAg combination, whose hardware cost is the least, uses global prediction (branches share the sole branch history table) and the accuracy is lower than any other combinations, while the PAs utilizing local prediction (every branch has exclusive history table) is the inverse. G-share scheme[4] proposed by Scott Mc Farling, an improved method based on GAg, significantly increases the accuracy by merely adding a XOR unit between the BHSR (Branch HiStory Register) and the instruction address, which differentiate branch instructions and avoids the situation that different instructions use the same pattern. To alleviate interference resulting from aliasing in the pattern table,

Content from this work may be used under the terms of the Creative Commons Attribution 3.0 licence. Any further distribution of this work must maintain attribution to the author(s) and the title of the work, journal citation and DOI.
Published under licence by IOP Publishing Ltd

Lee proposed Bi-mode branch prediction method[5], of which pattern table divided into two direction pattern tables. According to the taken tendency of the branch instructions, choice pattern table determines the final prediction from two direction pattern tables, which improved prediction accuracy. However, methods mentioned above all use identical amount of history. For the branch instructions with simple patterns, prediction do not need too much history. The inappropriate amount of history may even lead to the decline of accuracy. To solve this problem, Seznec et al. proposed improved TAGE branch prediction strategy[6]-[8]that allocate the amount of history according to the complexity of branch patterns. However, this method requires higher hardware cost. When a single branch predictor cannot simultaneously meet the requirements of high accuracy, low power consumption and low hardware cost, the composite branch prediction gradually becomes a trend. In reference[9], Bi-mode predictor and neural network prediction technology are combined and higher prediction accuracy is realized on the basis of less learning time.

In view of the low accuracy of single branch predictor due to the restriction of specific branch pattern and program, and considering complementary relationship between global predictor and local predictor, a composite branch predictor implementation scheme is proposed in this paper, which can achieve high accuracy branch prediction.

2. The structure of the composite branch predictor

Branch prediction based on branch history has two strategies: global branch prediction and local branch prediction [10]. With regard to global branch prediction, history of all branch instructions is recorded together and then used to predict the direction of current branch instruction. Local branch predictor records the history of every branch instruction respectively, based on which prediction about relative branch instruction is made. These two branch predictors have their own advantages and disadvantages. For branch instruction pattern characterized by repetition and simplicity, the local branch predictor shows higher accuracy, while in case of branch instructions relevant to each other, global branch predictors have higher accuracy.

2.1. Local branch predictor

Gshare Branch predictor is used widely in international global Branch predictor, which is based on a branch prediction buff (BPB), an 8-bit branch history register (BHR) shared by all branch instructions and a branch pattern table (BPT) composed of 2^8 2-bit saturating counters. Its structure as shown in Figure 1. the 2-bit saturating counter is indexed by XOR of the BHR and the least 8 significant bit of instruction address, which avoids accuracy decline resulting from the situation that a counter make prediction for more than one pattern. This method is also characterized by low hardware cost.

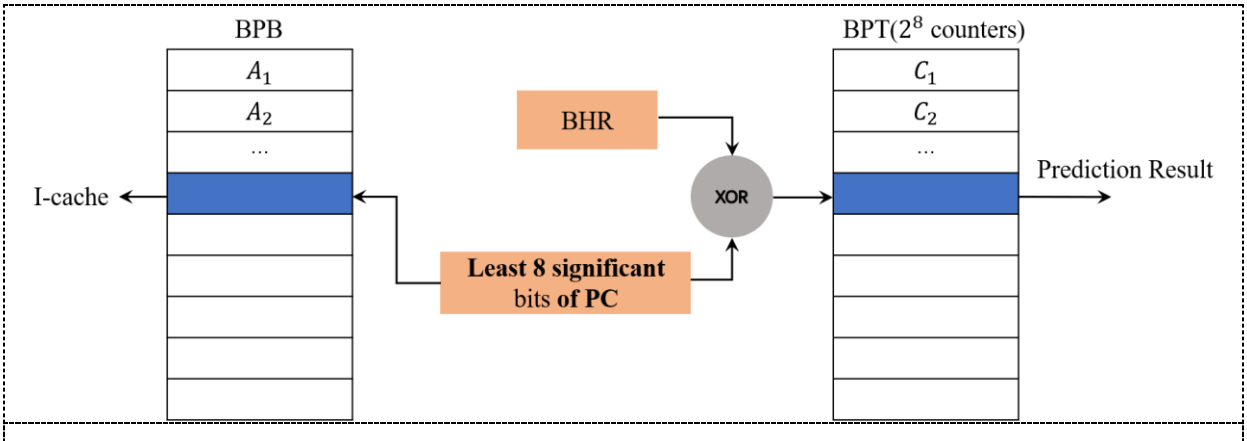

Figure 1. Gshare prediction algorithm

The basic branch prediction algorithm is as follows:

(1) Retrieving in the BPB by address the of currently fetched instruction. If new address is retrieved, i.e. the fetched instruction is a branch, the instruction corresponding to this address will be fetched.

(2) Retrieving counters by BHR in the BPT. Then if the counter make prediction of 'not taken', the instruction retrieved in BPB will be deleted in pipeline and the subsequent instruction, whose address is PC+4, will be fetched.

2.2. Local branch predictor

Local branch predictor can precisely predict independent branch instructions whose directions do not have direct relationship with other branch instructions. As shown in Figure 2, the local branch predictor consists of a branch history table (BHT) and a branch pattern table (PHT). The BHT is composed of branch history registers (BHR). Each register records the history of a branch instruction. The Branch pattern table is made up of 2-bit saturating counters that make prediction.

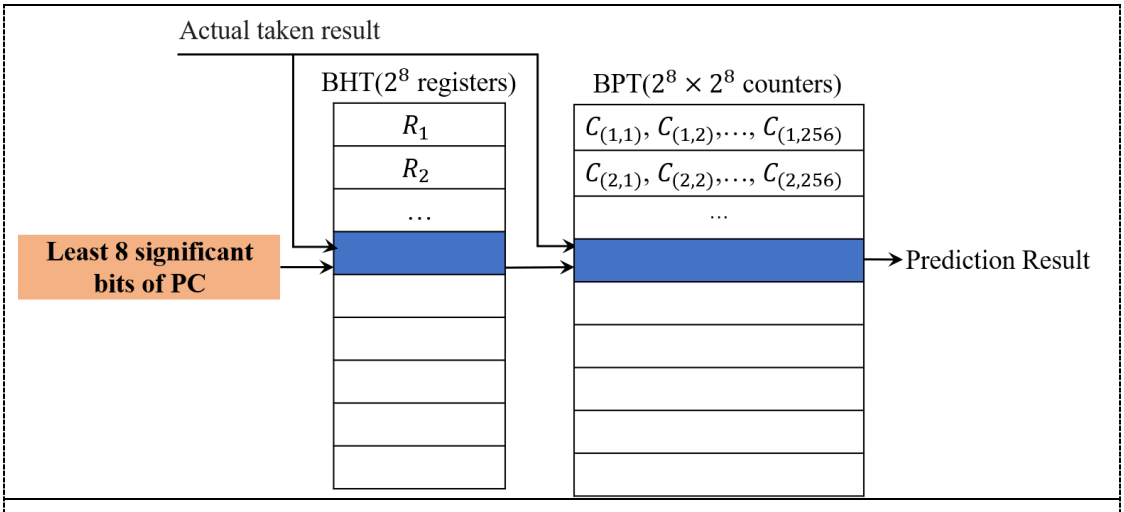

Figure 2. The structure of local branch predictor

The BHT is indexed by least 8 significant bits of instruction address so that history of every branch instruction is orderly recorded and easily accessible. Accordingly, number of BHRs is 2^8 in total. For loop statements with fixed patterns, accurate prediction can be achieved when the number of recorded historical information in bits are not less than the number of cycles. However, in order to curb hardware cost, this paper chooses 8-bit registers. The actual direction in bits of the branch instruction is written into the corresponding register to update the history after the register shift to remove the most previous bit and create a vacancy.

The PHT contains $2^8 \times 2^8$ 2-bit saturating counters in total, indexed by branch history registers. As shown in Figure 3, counters are essentially state machines, and the four states numbered from small to large indicate tendency of taken from weak to strong. When the most significant bit is 1, the prediction is taken, otherwise the prediction is not taken. In addition, the actual direction is used as a transition condition to update the state.

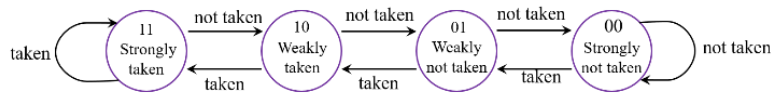

Figure 3. State Transition of 2-bit saturating adder

3. Branch predictor structure based on Jaccard coefficient

In order to improve the accuracy of branch prediction, this paper proposes a composite branch prediction that improves the accuracy and reduces the computational complexity of the global branch predictor by exploiting complementary relationship between global predictor and local predictor and branch selector based on Jaccard coefficient.

Composite branch selection algorithm is the key factor to determine the performance of the whole predictor, thus reasonable selection algorithm is of great importance. Weighted method of the 2-bit counter is frequently used. However, it is necessary to allocate a 2-bit counter for each branch predictor and its selection performance is far from satisfactory. As machine learning and neural network are applied extensively, scholars focus on how to select the prediction results based on neural network theory. The embedded processor has limited circuit area and the high precision of prediction requires the recording of historical prediction results over a long period of time. Therefore, Jaccard coefficient is utilized to solve the prediction result selection problem in composite branch predictor. Essentially, accuracy of previous prediction determines whose prediction can be adopted currently. That is, the Jaccard coefficient represents the similarity between the historical prediction records and the global prediction results. The larger the Jaccard coefficient is, the higher the similarity is, and the higher the prediction accuracy is.

Jaccard coefficient is used to measure the similarity between two sets. It is defined as:

$$J(A,B) = \frac{\sum_i \min(a_i, b_i)}{\sum_i \max(a_i, b_i)} \tag{1}$$

Where a_i and b_i are the i-th elements of samples A and B respectively, and samples A and B are two n-dimensional vectors. The Jaccard coefficient can also be used to measure the similarity between two vectors and uses the proportion of different elements in the two vectors to measure the similarity. Therefore, while ensuring the accuracy of branch prediction, the recording time of historical prediction information can be reduced to reduce the circuit area.

The historical prediction information record table can only be represented by 0 and 1, so the Jaccard coefficient can be expressed as:

$$J(A,B) = \frac{\sum_i a_i + b_i}{\sum_i a_i \oplus b_i} \tag{2}$$

Where $\oplus$ is the xor operator. In the above formula, the numerator represents the number of vectors A and B with elements of 1, and the denominator represents the vectors A and B with elements of 0 and 1, respectively. The higher the $J(A,B)$ is, the higher similarity between A and B is. d represents the difference between the prediction and the actual direction, namely accuracy. Branch selector based on Jaccard coefficient requires an 8-bit register for each predictor to record the history of prediction. As shown in figure 4, the algorithm is as follows:

(1) Based on global history, local branch predictor and global branch predictor respectively make 8 consecutive prediction, produced by which two 8-bit binary sequences are recorded into corresponding registers;

(2) Calculating Jaccard coefficient between latest 8 -bit global historical information and the sequence;

(3) Prediction made by predictor related with higher Jaccard coefficient will be adopted next time;

(4) If Jaccard coefficient of two predictor is equal, the predictor with bigger weight will be selected. Assuming that accuracy of two predictor is A_1 and A_2 respectively, which are from large amount of simulation. The weight of predictor j is:

$$W_j = \frac{A_j}{\sum_{i=1}^{2} A_i} \tag{3}$$

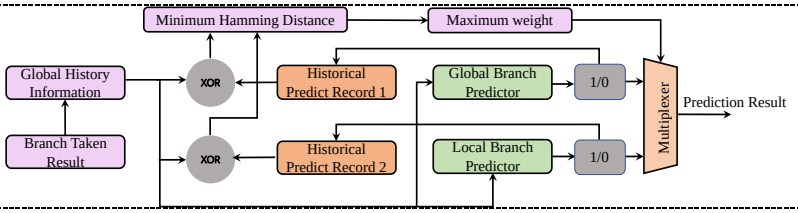

Figure 4. The structure of composite branch predictor based on hamming distance

4. Performance evaluation

In this paper, the prediction performance of the proposed branch predictor is analysed by using the SPEC CPU2000 test program on Gem 5 simulator. The branch miss rate model is used as the index to measure the prediction accuracy. SPEC CPU2000 is a test program developed by the standard performance evaluation organization for evaluating CPU performance. The branch prediction miss rate is defined as the proportion of the branch prediction failed instructions in the total branch instructions of the test program, so the miss rate can be expressed as:

$$r = \frac{c_m}{c_t} \tag{4}$$

Where c_m represents the instruction number of branch prediction failure, and c_t represents the total branch instruction number of test program.

The host platform used in this paper is the Intel i5 6400 x86-64 bit instruction set architecture, the operating system is Ubuntu 16.04, and the memory is 12GB. The platform running the SPEC CPU2000 test program is based on the ARM v8-a 32-bit instruction architecture and the Linux 2.6.9 operating system. Gem 5 emulators are open source system-level and processor emulators for running the Linux 2.6.9 operating system as well as the SPEC CPU2000 test program. To improve simulation efficiency, this paper adopts Gem 5 emulators in atomic mode to run the test assembly. Table 1 shows the simulation time of Gem 5 running the test program on the Linux platform.

Table 1. The simulation time of test program.

SPEC CPU2000 test program	Simulation time (min)
gzip	211
gcc	264
parse	170
vortex	225
bzip2	213
gap	206

The gzip, GCC, parse, vortex, hzip2 and gap programs in SPEC2000 were selected as the test benchmark program in this paper, and the above simulation hardware platform was used to compare the branch predictor miss rate of Gshare branch predictor, local predictor and proposed branch predictor. The simulation results are shown in Figure 5 below:

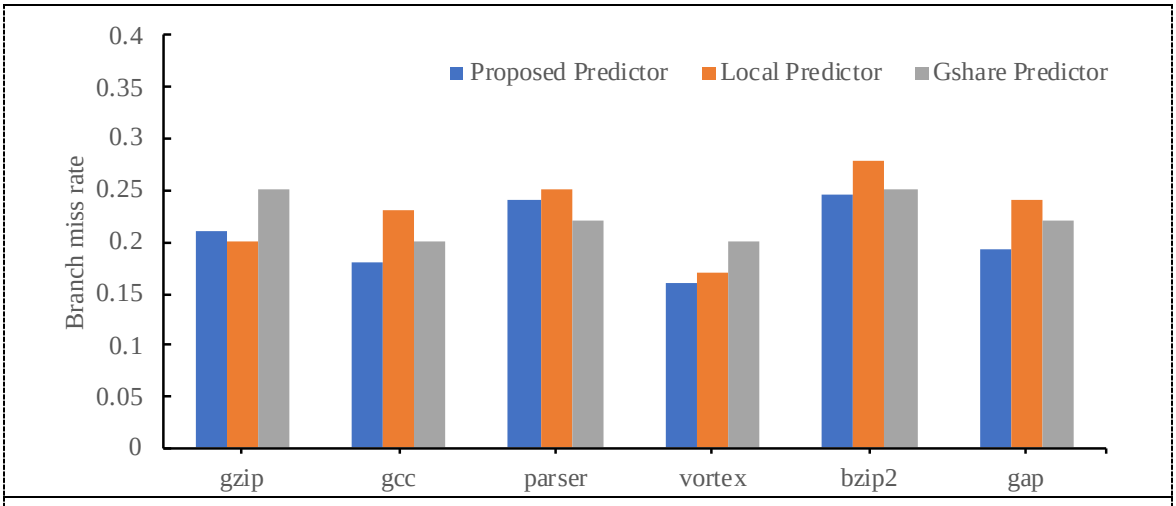

Figure 5. Performance simulation of predictors

As can be seen from Figure 5, the branch predictor proposed in this paper has a significantly better miss rate than the Gshare branch predictor and local predictor. In other words, the prediction performance of Gshare global branch predictor and local predictor can be well complemented. Local and Gshare global predictors have different prediction accuracy about different test programs. When making prediction about different test programs, the prediction is selected from the predictor with highest accuracy, so that the accuracy of whole predictor can be improved.

5. Conclusion

The branch instruction is an important part of programming, and it is also one of the key factors that affects the performance of modern processors. How to maximize the accuracy of branch prediction is one of the main challenges faced by modern processor design. Aiming at the defect of single predictor's prediction ability and the complementary relationship between the global branch predictor and the local branch predictor, a composite branch prediction algorithm based on these two predictors is proposed in this paper. While guaranteeing the predictive ability of the processor by using local branch predictor, global branch predictor based on Gshare and branch selection based on Hamming distance, the device has a simple structure and it is suitable for embedded processor applications with limited area, resources and power.

References

[1] Chang P Y. Branch Classification: A New Mechanism for Improving Branch Predictor Performance[C]// International Symposium on Microarchitecture. IEEE, 1994.
[2] Lee J K F. Branch prediction strategies and branch target buffer design[J]. IEEE Computer, 1984, 17
[3] Yeh T Y, Patt Y N. A Comparison of Dynamic Branch Predictors that use Two Levels of Branch History[J]. ACM SIGARCH Computer Architecture News, 1995.
[4] S. McFarling. Combining branch predictors [R]. Technical Report TN-36, Digital WesternResearchLaboratory,1993.
[5] Lee C C, Chen I C K, Mudge T N. The Bimode++ Branch Predictor[C]// Thirtieth IEEE/ACM International Symposium on Microarchitecture. IEEE, 1997.
[6] A. Seznec, P. Michaud. A case for (partially) TAgged GEometric history length branch prediction[J]. Journal of Instruction Level Parallelism,2006.
[7] A. Seznec. A 256 Kbits L-TAGE branch predictor [C]. // Proceedings of the 2nd Championship Branch Prediction,2007.

[8] A. Seznec. A 64 Kbytes ISL-TAGE branch predictor[J]. JWAC-2: Championship Branch Prediction,2011.
[9] Yang-Xu-Rui L, Shu-Ming C, Yong L I. Low-cost Composite Neural Network Branch Predictor[J]. Computer Engineering, 2011.
[10] MA Feng, FANG Xiao-Min, WANG Chun-Jun, HU Qiu-Hui. Design and Implementation of Compound Branch Predictor Circuit[J]. Computer Engineering, 2011, 37(13): 243-245.