行政院國家科學委員會補助專題研究計畫成果報告

※※※※※※※※※※※※※※※※※※※※※※※※※
※　　　　　　　　　　　　　　　　　　　　　　　※
※　　　超純量架構分支預測單元設計　　　※
※　　　　　　　　　　　　　　　　　　　　　　　※
※※※※※※※※※※※※※※※※※※※※※※※※※

計畫類別：~**個別型計畫**　　□**整合型計畫**

計畫編號：NSC　89－2213－E－009－067－

執行期間：88 **年** 08 **月** 01 **日 至**　89 **年** 07 **月** 31 **日**

計畫主持人：陳昌居　副教授

共同主持人：

本成果報告包括以下應繳交之附件：

□**赴國外出差或研習心得報告一份**

□**赴大陸地區出差或研習心得報告一份**

□**出席國際學術會議心得報告及發表之論文各一份**

□**國際合作研究計畫國外研究報告書一份**

執行單位：國立交通大學資訊工程學系

中　華　民　國　89 **年**　10 **月**　30 **日**

行政院國家科學委員會專題研究計畫成果報告
超純量架構分支預測單元設計

The Design of Superscalar Branch Prediction Unit

計畫編號：NSC 89-2213-E-009-067
執行期限：88 年 8 月 01 日至 89 年 07 月 31 日

主持人：陳昌居　　國立交通大學資訊工程學系

中文摘要

由於現今的微處理器擁有較長的管線，並且具有每個時脈能夠發出多道指令的能力，所以分支預測的準確率就變的格外重要。到目前為止有各種不同的分支預測的機制已經被提出，其中有兩種至今仍被常常使用。第一種稱為「bimod branch predictor」，使用 2 位元的計數器來計錄分支預測的結果。由於他只使用簡單的 2 位元計數器，所以對一般偏向於跳或不跳的分支指令都能準確的預測。第二種是「2-level branch predictor」，它使用兩層式的架構記錄追蹤鄰近指令之間的相關性，如果某分支指令與臨近之間的分支指令存在著相關性，那麼 2-level branch predictor 能夠準確的作預測。可當我們使用不同的預測機制作實驗時，我們發現，沒有一種預測機制能同時滿足所有的 benchmark。

因為上述的原因，我們提出一種稱為「vote predictor」的分支預測機制，它能夠達到更準確的預測。在 vote predictor 中我們使用三種不同的預測機制來同時作預測，並且使用 vote circuit 來選擇最後的輸出。當一道分支指令抓取之後，此三個不同的預測機制會同時進行預測，如果有兩個以上的預測機制產生相同的輸出，那麼 vote circuit 就會把它當做最後的輸出。所以 vote circuit 只是做簡單多數決的動作而已。

關鍵詞：超純量、投票預測器、分支預測

1、Abstract

As modern microprocessors employ deeper pipelines and issue multiple instructions per cycle, they are becoming increasingly dependent on accurate branch prediction. Up to now, various branch prediction strategies have been proposed. There are two branch predictors are widely used today. The first is bimod predictor, using 2-bit saturation counters to record the history outcomes of every branch instruction. So bimod predictor is good to predict those branches which are bias taken or non-taken. The second is two-level adaptive branch predictor, which using two-level architecture to trace the correlation of nearby branch outcomes. So if one branch have correlation with nearby branches, the two-level branch predictor can make the correct predictor. These two branch predictors can gain benefits on their way individually. However, we find that there is no one branch predictor is good for all benchmarks.

With the factor above, we propose a branch prediction machine, called “vote predictor”, to make the more accurate prediction. In the vote predictor, we combine three different branch predictors to make prediction and use a vote circuit to select the final output. Every time when one branch is encountered, three branch predictors make prediction concurrently. If two or more branch predictors make the same output (taken or non-taken), the vote circuit will select it as the final output. So the vote circuit just work like the majority rule.

Keywords: Superscalar, Vote Predictor, Branch Prediction

2、Introduction

In the search for higher levels of

performance, recent machine designs have made use of increasing degrees of instruction level parallelism (ILP). For example, both superscalar and superpipelining techniques are becoming increasingly popular. In modern pipelined superscalar processor, multiple instructions can be fetched and processed concurrently. Out-of-order instruction issue is effective only when instructions can be supplied at a sufficient rate to keep the execution unit busy. But when a branch instruction occurs, there are two possibilities. If the branch is not taken, the next contiguous instruction will be fetched. Otherwise, if the branch is taken, the processor must know where the branch instruction redirects the instruction stream.

The branch delay of a processor is the delay from the time that the processor decodes a branch instruction to the time that it decodes the first target instruction [3]. The branch delay is one cycle in a typical RISC processor. The long branch delay of superscalar processor is relative to the cycle in which the branch is decoded, a direct result of the wider instruction fetch and decode path.

In order to resolve such problems, many branch prediction schemes had been proposed in these years [2,4,5,6,7]. This paper focuses on predicting branch directions.

Hardware branch prediction strategies have been studied extensively. The most well known technique referred to as bimod branch prediction uses 2-bit counters to record the last few history outcomes of every branch instruction. One method considers the history of each branch independently and takes advantage of repetitive patterns. Since the histories are independent, it is referred to as local branch prediction. Another technique uses the combined history of all recent branches in making a prediction. This technique is referred to as global branch prediction. Both local and global branch prediction strategies have different distinct advantages. The local branch prediction works well for branches with simple repetitive patterns. The global branch prediction works particularly well when the direction taken by sequentially executed branches is highly correlated. In order to get the more accurate branch prediction, the combining branch predictor was proposed [7]. This scheme allows the distinct advantages of branch predictors to be combined and is shown to provide more accurate predictions than any one predictor alone.

In this paper, we will present a new branch predictor called as “vote predictor”. This machine combines three different well-known branch predictors and a vote circuit to make predictions. We use “SimpleScalar Tool Set” to simulate “vote predictor” [1]. The experimental results shows that the vote predictor can obtain more accurate direction prediction than all his component predictors.

3、Vote Predictor Structure

The hardware configuration of the vote predictor is shown in Figure 1. We combine three different predictors into our vote predictor (such as predictor 1, predictor 2 and predictor 3). These three predictors can be any well-known predictors, such as gshare, bimod… etc. The another important machine in our vote predictor is “vote circuit”. When each predictor produces their own prediction result, the vote circuit will be responsible for arbitrating the final prediction.

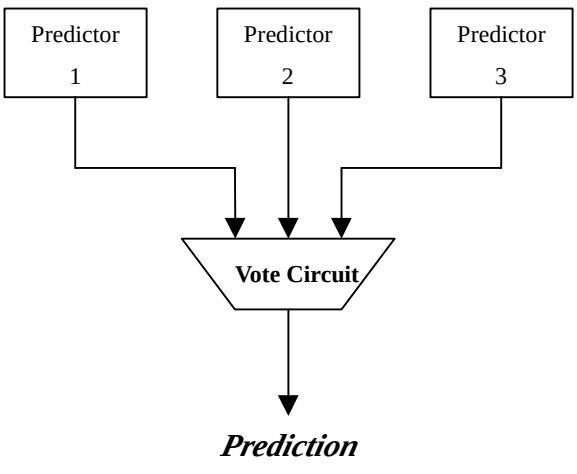

Figure 1. Logical organization of the vote predictor

The arbitrational rules are as follows:

- **Predict “Taken”**: if two or more predictors predict that the branch

instruction will jump to target address, then the vote circuit will output "Taken" signal.

- **Predict "Non-Taken**": if two or more predictors predict that the branch instruction will jump to fall-through address, then the vote circuit will output " Non-Taken" signal.

From the above, we can know that the vote circuit just works like the majority rule.

4、Simulation Result

The out-of-order issue, superscalar processor simulator in the SimpleScalar tool set employs a 16-entry register update unit (RUU) along with 4 integer ALUs and 4 floating point ALUs. It can decode 4 instructions at one time, and can issue and execute up to 4 instructions simultaneously if there are no data dependency among these instructions to be executed.

In our experiment, we will use four well-known predictors and combine three of them arbitrarily. These four predictors are bimod predictor, gshare predictor, PAg predictor, and the path-based branch predictor. Here the PAg predictor has some different from the original PAg predictor proposed by Y. N. Patt [8]. For the original PAg predictor, it indexes the pattern history table only by concatenating the branch history register (BHR) in the branch history table (BHT) and few address bits. But in order to utilize the pattern history table (PHT) more efficiently, we will XOR the BHR and branch address together to index the PHT.

Since we use four different predictors, it will produce four different combinations. We identify the four different predictors as the following:

- **Vote1 Predictor:** it combines a bimod, a gshare and a path-based predictor into one predictor.
- **Vote2 predictor:** it combines a bimod, a PAg, and a path-based predictor into one predictor.
- **Vote3 predictor:** it combines a bimod, a PAg, and a gshare into one predictor.
- **Vote4 predictor:** it combines a PAg, a gshare, and a path-based predictor into one predictor.

We will compare various vote predictors under 4k-entry PHT. The results of comparison between our four vote predictors are shown in Figure 2. Also, Table 1 lists the hardware costs of various vote predictor models.

McFarling proposed the combining branch predictor. It combines multiple branch predictors into one, including bimod and gshare predictors, and then uses a meta-table to choose which result to be used. In order to compare our vote1 predictor with combining predictor fairly, we configure the combining branch predictor with one bimod, one gshare which has 8-bit global history register, and one meta-table, and the same size of PHT as the vote1 predictor except the size of meta-table. For the meta-table, we configure it with extra 2048-entry PHT. The results are shown in Figure 3. From the results, we can find that when the PHT entry is less than or equal to 4k, our vote predictor is not better than the combining predictor. However, when the PHT entry is larger than or equal to 8k, our vote predictor outperforms the combining predictor in all benchmark programs expect hydro2d. This is because that the PHT is shared with three predictors in vote1 predictor, but the PHT is shared with two predictors in combining predictor. As we can image that there will be many conflicts in fewer-entry PHT, this results in reducing prediction accuracy in our vote predictor more than in combining predictor.

5、Conclusions

In this paper, we experiment on eight benchmark programs with several well-known branch predictors to compare their prediction accuracy. According to the experimental results, we can find that different predictors work well for different benchmark programs and there is not one predictor that can work well for all benchmark programs. This is because of different predictors have different prediction characteristics. In order to solve such problems, we propose the vote predictor to

combine different advantages from predictor and enhance the prediction accuracy. From our experimental results, the vote predictor really outperforms all his component predictors.

In order to evaluate the impacts of prediction accuracy on different vote predictor models. We use four well-known branch predictors and combine three of them individually. The results show that those predictors that combine the PAg predictor have the better performances (vote2, vote3, and vote4). However, if we consider the cost/performance trade-off, the vote1 predictor model has the best cost-effectiveness.

We also compare our vote predictor with combining branch predictor. When the PHT entries are less than or equal to 4K, our vote predictor is not better than combining. However, when the PHT entries are bigger or equal to 8K, our vote predictor can outperform the combining branch predictor.

As the fabrication technique has been improved, the chip area is increased and more control component can be inserted in a chip. When the hardware cost is affordable, the complex and accurate predictor can be included in the CPU design. In this paper, we only use four simple predictors to construct the vote predictor and the average prediction accuracy is 95.5%. In the future maybe we can put other complex and accurate predictors into the vote predictor to obtain more prediction accuracy.

6、References

[1] D. Burger and T. M. Austin, “The SimpleScalar Tool Set Version 2.0,” Technical Report 1342, Computer Sciences Department, University of Wisconsin, Madison, WI, 1997.

[2] J. K. F. Lee and A. Smith, ”Branch Prediction Strategies and Branch Target Design,” Computer, vol. 17,no. 1, pp. 6-22, Jan. 1984.

[3] M. Johnson, Superscalar Microprocessor Design, Prentive Hall, Englewood Cliffs, NJ, 1991.

[4] P. -Y Chand, E. Hao, and Y. N. Patt, “Target Prediction for Indirect Jumps,” Proceedings of the 24th International Symposium on Computer Architecture, Denver, June 1997.

[5] Pierre Michaud, Andre Seznec, Richard Uhlig, “Trading Conflict and Capacity Aliasing in Conditional Branch Predictors,” 24th Intl. Symp. On Computer Architecture, pp. 292-303, June 1997.

[6] R. Nair, “Dynamic Path-Based Branch Correlation,” Proceedings of the 28th Annual ACM/IEEE International Symposium on Micorarchitecture, pages 15-23, 1995.

[7] S. McFarling, “Combining Branch Predictors,” Technical Report TN-36, Digital Western Research Laboratory, June 1993.

[8] T. - Y. Yeh and Y. N. Patt, ”A Comparison of Dynamic Branch Predictors that use Two Levels of Branch History,” Proceedings of the 20th Annual International Symposium on Computer Architecture, Pages 257-266, 1993.

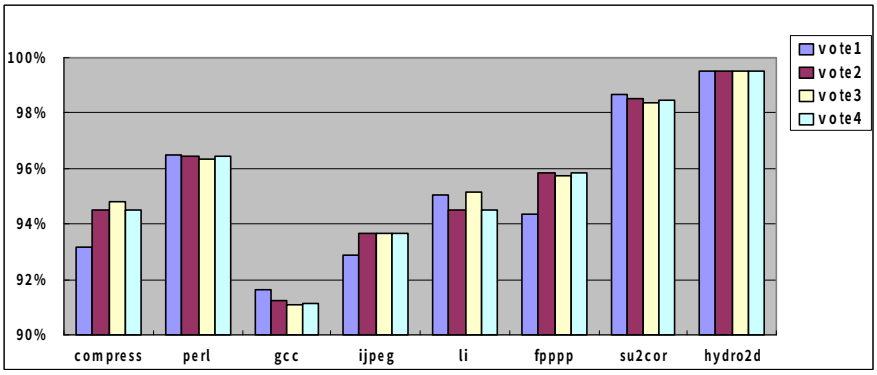

Figure 2. Comparison of various vote predictors models with 4K-entry PHT

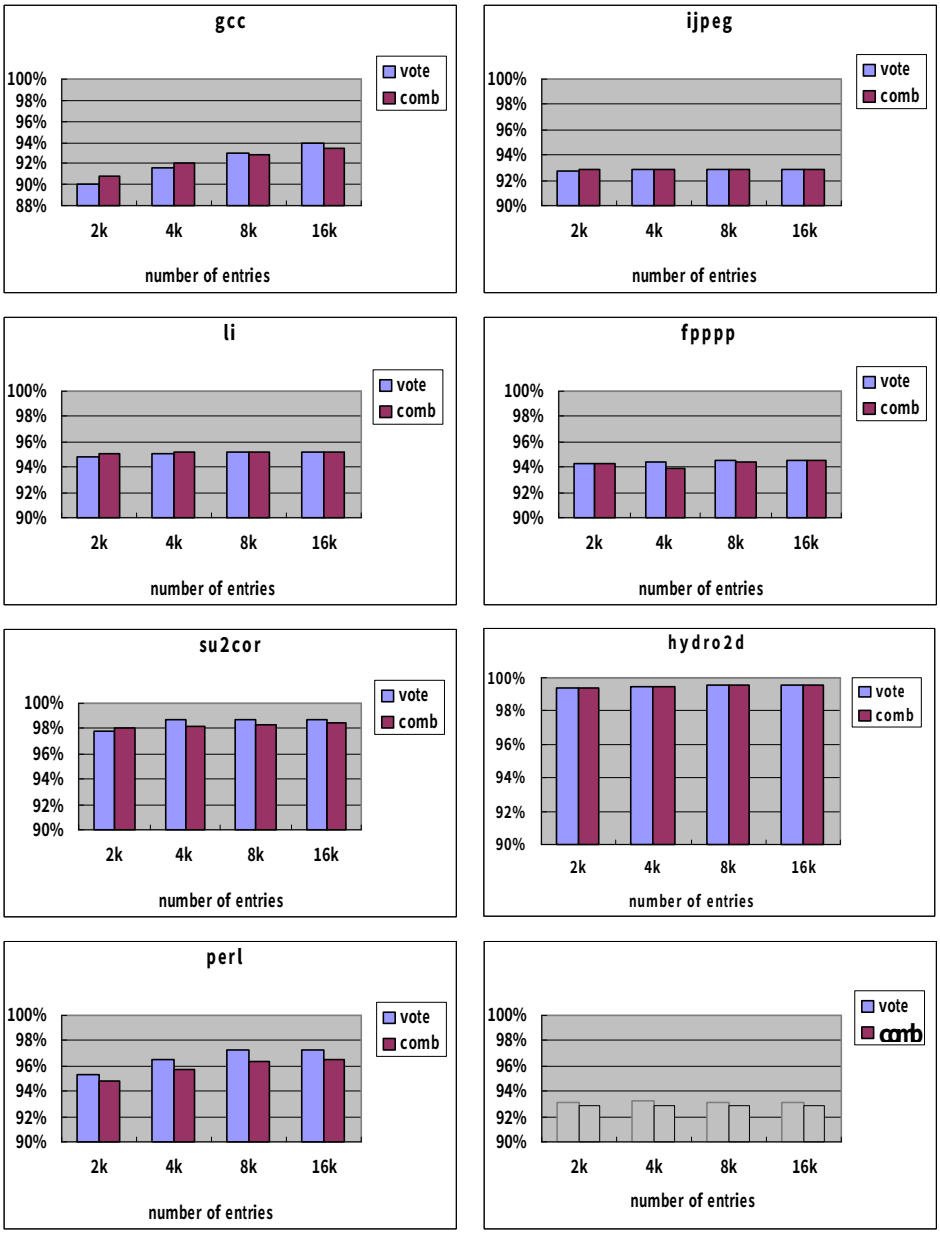

Figure 3. Branch direction hit rate of vote predictor vs. combining branch predictor

Table 1. Hardware costs of various vote predictor models

	gshare	bimod	path-base	PAg	PHT	Total
Vote1	8 bits	0 bits	9 bits	N/A	4096*2 bits	**8209 bits**
vote2	N/A	0 bits	9 bits	2048 * 8 bits	4096*2 bits	24585 bits
vote3	8 bits	0 bits	**N/A**	2048 * 8 bits	4096*2 bits	24584 bits
vote4	8 bits	**N/A**	9 bits	2048 * 8 bits	4096*2 bits	24593 bits