Saturating Counter Design for Meta Predictor in Hybrid Branch Prediction

YOUNG JUNG AHN, DAE YON HWANG, YONG SUK LEE, JIN-YOUNG CHOI
AND GYUNGHO LEE
The Dept. of Computer Science & Engineering
Korea University
Anam-dong 5-ga-1, Sungbuk-gu, Seoul, 136-701
THE REPUBLIC OF KOREA
yjahn@formal.korea.ac.kr, dyhwang@formal.korea.ac.kr, duchi@korea.ac.kr,
choi@formal.korea.ac.kr, ghlee@korea.ac.kr

Abstract: High-performance computer systems have made use of super-pipelining, dynamic scheduling and multi-issue superscalar processor technologies. In these systems, branch prediction accuracy has a significant impact on the performance because the penalty for misprediction increases as pipelines deepen and the number of instructions issued per cycle increases. To have a better prediction accuracy, branch predictors utilized in high performance systems are a hybrid type. Hybrid branch prediction employs multiple branch predictors and selects one particular branch predictor per the program context of a given branch instruction instance for prediction. For choosing a particular branch predictor is a job of “meta-predictor”. This paper considers type and size of saturating counter design specifically for the meta predictor. State transitions different from a usual saturating counter may choose a particular predictor for prediction better and result a higher prediction accuracy.

Key-Words: saturating counter, prediction accuracy, hybrid branch predictor, meta predictor.

1 Introduction

There have been many recent studies and increasing efforts to improve the performance of computer system. Exploiting Instruction Level Parallelism (ILP) has been a major means of achieving high-performance computer systems [1]. Deep pipelines, various superscalar methods and many dynamic scheduling algorithms have been utilized for exploiting ILP. In such high performance systems, branch prediction to predict the outcome of a conditional branch has become an increasingly important component in determining overall performance. Without the branch prediction, processor would have to wait until a branch is resolved before the next instruction can enter the fetch stage in the pipeline. The branch predictor attempts to avoid this delay by trying to guess whether the conditional jump is most likely to be “taken” (true branch) or “not-taken” (false branch). The branch that is guessed to be more likely is then fetched and speculatively executed. If it is later detected that the prediction was wrong, then the speculatively executed or partially executed instructions are discarded: Processor starts over with the correct program control flow. Dynamic branch prediction records the history of branch instructions, which means the directions taken by the past instances of a branch instruction, and predicts the direction of a branch instance based on the history. Dynamic branch prediction schemes generally fall into two types. One is self-history-based branch prediction, and the other is correlation-based branch prediction. The self-history-based branch prediction predicts the direction of the current branch instruction with only using the history of past instances of the instruction, i.e. self-history. This scheme may achieve a high prediction accuracy when it’s using for the program with lots of loops. The correlation-based branch prediction predicts the direction of a branch using the history of branch instructions in addition to its own history. This can provide better accuracy when a branch direction depends on control flow paths reaching to a particular branch instance.

To take advantage of the both types of branch predictors, one may selects one of the predictors by employing two predictors, one with self-history based prediction and the other with correlation based prediction. Such a hybrid prediction has been a choice for high performance computer systems.

A hybrid branch predictor needs a “choice” predictor, also known as a meta predictor to choose one of the branch predictors employed to reflect the current
program context of a branch better. The meta predictor predicts which one of the branch predictors employed by utilizing a saturating counter as in a self-history based branch prediction. A saturating counter that is used in a branch predictor for predicting a branch direction decides the direction of branch. While the saturating counter used in the meta predictor decides a branch predictor which is more suitable for predicting the current branch instruction instance per the program context.

A specific saturating counter design adopted for the meta predictor obviously affects branch prediction accuracy, though there have been little research on the design of saturating counter for the meta predictor in a hybrid branch prediction. This paper considers saturating counter design specifically for the meta predictor used for a hybrid branch predictor. Different state transitions from a usual saturating counter may choose a particular branch predictor better and result a higher branch prediction accuracy. The saturating counter design may take a different form for the purpose of its use.

This paper is organized as follows: related works are presented in section 2; we introduce a typical branch prediction technique that could be used mainly by existing systems. In section 3, we introduce the saturating counter of meta predictor of the combining predictor for improving branch prediction accuracy and we analyze our experiment. Finally, we discuss the results of this study and our future work.

2 Related works

Alpha EV6 and alpha EV8 are microprocessors designed for achieving high-performance. Both two microprocessors use hybrid branch predictor for improving branch accuracy. The following explains alpha EV6 and alpha EV8.

The alpha EV6 implements a sophisticated tournament branch prediction scheme that dynamically chooses between local and global history to predict the direction of a given branch [2]. Attribute of local history is that pattern behavior sometimes correlates with the execution of a single branch at a unique PC location. And Attribute of global history is that pattern behavior sometimes correlates with the execution of all previous branches. When branch result is an alternating taken/not-taken sequence, the local prediction is very useful. As the branch executes multiple times, it will saturate the prediction counters corresponding to these local history value and make the prediction correct. When the result of a branch can be inferred from the direction of precious branches, the global prediction is very useful. The global history predictor can learn this pattern with repeated invocations of the two branches. When a branch instruction retires, the alpha EV6 updates the chooser. The chooser consists of two-bit saturating counters. If the results of the local and global predictor differ, the alpha EV6 updates the selected meta prediction entry to support the correct predictor.

Branch predictor of 21464 Microprocessor Architecture Global branch history branch predictor tables lead to a phenomenon known as aliasing or interference, in which multiple branch information vectors share the same entry in the predictor table, causing the predictions for two or more branch substrings to intermingle[3, 4, 5]. "De-aliased" global history branch predictors have been recently introduced: the enhanced skewed branch predictor e-gskew, the agree predictor, the bimode predictor and the YAGS predictor [6, 7, 8, 9]. These predictors have been shown to achieve higher prediction accuracy at equivalent hardware complexity than larger "aliased" global history branch predictors such as gshare or GAs [2, 10]. However, hybrid predictors combining a global history predictor and a typical bimodal predictor only indexed with the PC may deliver higher prediction accuracy than a conventional single branch predictor [11, 2]. Therefore, "de-aliased" branch predictors should be included in hybrid predictors to build efficient branch predictors.

The EV8 branch predictor is derived from the hybrid skewed branch predictor 2Bc-gskew presented in [12]. 2Bc-gskew combines e-gskew and a bimodal branch predictor [6]. It consists in four identical predictor-table banks, i.e., the three banks from the e-gskew-including a bimodal bank-plus a meta predictor. 2Bc-gskew-pskew combines a bimodal component, a global history register component and a per-address history component [12].

Nair has been researching much about the saturating counter for predicting a branch direction decides the direction of branch [13]. This paper proposes one of various 2-bit saturating counter of meta predictor, which provide better prediction accuracy. This saturating counter is used in many branch predictors, but there is no research about the saturating counter that is used in the meta predictor. This saturating counter influenced in the performance of the hybrid branch predictor. Therefore, this paper treats the saturating counter that is used for the meta predictor to choose one of the branch predictors having the highest branch prediction accuracy.

3 Saturating Counter

Many researches are being done on saturating counters which are used for predicting branch direction. When a branch instruction’s direction is determined, there is a high probability that the branching direction of the instruction will be same. Nair proposed a saturating counter based on this idea. Assume that a saturating
counter uses N-bit to represent a state [13]. The branch
direction of branch instruction is represented as ‘taken’
or ‘not-taken.’ If the size of the saturating counter is N-
bit, there can be 0 to N-1 states. If the state is 0 to N/2-1,
branch predictor estimates the direction as ‘not-taken.’
The branch predictor will predict the branch direction
as ‘taken’ if the state of saturating counter is N/2 to N-1.
The actual branch direction of a branch instruction can
be checked after the instruction executed at the function
unit of the microprocessor. The state of saturating
counter is transited by the result of the branch
instruction. If the direction of branch instruction is
‘taken’, then the state transition is as following. The
saturating counter stays at N-1 if its state is N-1. In
other words, the saturating counter isn’t transited. If the
state of saturating counter is N/2-1, the saturating
counter transit to N-1. If the state of saturating counter
is neither of N/2-1 nor N-1, it would be just increased.
When the branch direction of the branch instruction is
‘not-taken’, the transition is following. If the state of the
saturating counter is 0, the counter stays 0. It means
saturating counter is not transited. If the state of
saturating counter is N/2, the counter is transited 0. The
state of saturating counter will be decreased if it is not
either 0 or N/2. Fig.1 shows that the relation of
transition when the Nair’s saturating counter is 2-bit.

This paper considers type and size of saturating counter
used in the meta predictor. A meta predictor in a hybrid
branch predictor should decides a branch predictor
which is more suitable for predicting the current branch
instruction instance per the program context. In other
words, the accuracy of hybrid branch predictor
increases only when the meta predictor chooses one of
the branch predictors having the highest branch
prediction accuracy.

Next is a dynamic method for meta predictor to
choose more accurate branch predictor. The two branch
predictors for predicting the direction in the hybrid
branch predictor will be named A and B. In both A, B,
there is a memory to store the weight of each
instruction. The weight is increased if the branch
predictor for predicting the direction hits, and is
decreased if the predictor mispredicts. The role of a
meta predictor is to choose the highest weighting
branch predictor. However, large amount of memory is
needed to use this method.

This paper suggests type of saturating counter used
in meta predictor as below. The saturating counter will
transit only when the predict direction of A and B are
different. If the predict direction of A and B are same,
the meta predictor’s decision is not changed, so the
saturating counter transit when the predict direction of
A is different with the predict direction of B. If the
direction of the branch predictor which is selected by
meta predictor is correct then represent hit, if it’s not
then represent mispredict. If a saturating counter’s size
is N-bit, the counter takes value from 0 to N-1. Meta
predictor chooses A when the state of saturating counter
is from 0 to N/2-1, and chooses B when the state is
from N/2 to N-1. After the branch predictor is executed
in the function unit of a microprocessor, the direction
predicted by A and B could be checked whether it is a
hit or a mispredict. If A is a hit, the saturating counter’s
state is increased, but if it is a mispredict, the state
would be decreased. Fig.2 illustrates the transition of a
2-bit saturating counter proposed.
The saturating counter showed above transit only when the predict direction of A and B is different. Because if A is hit, B is mispredict, and if A is mispredict, B is hit. Fig.3 shows a conversion of Fig.1’s saturating counter by using the meta predictor’s saturating counter.

Fig.3 shows a conversion of Fig.1’s saturating counter by using the meta predictor’s saturating counter.

The black bar of Fig.4 shows the result when using a saturating counter illustrated in Fig.3 as the meta predictor. The white bar is the result when using the proposed saturating counter. The graph shows suitability of the proposed saturating counter as the saturating counter of the meta predictor.

Table 1. Branch prediction accuracy

<table>
<thead>
<tr>
<th></th>
<th>2-bit</th>
<th>3-bit</th>
<th>4-bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>bzip</td>
<td>input.graphics</td>
<td>0.9635</td>
<td>0.9641</td>
</tr>
<tr>
<td></td>
<td>input.program</td>
<td>0.9492</td>
<td>0.9507</td>
</tr>
<tr>
<td></td>
<td>input.source</td>
<td>0.9457</td>
<td>0.9477</td>
</tr>
<tr>
<td>gap</td>
<td>ref</td>
<td>0.9683</td>
<td>0.9687</td>
</tr>
<tr>
<td></td>
<td>166</td>
<td>0.9532</td>
<td>0.9537</td>
</tr>
<tr>
<td></td>
<td>200</td>
<td>0.9317</td>
<td>0.9317</td>
</tr>
<tr>
<td></td>
<td>expr</td>
<td>0.9245</td>
<td>0.9249</td>
</tr>
<tr>
<td>gcc</td>
<td>integrate</td>
<td>0.9524</td>
<td>0.9529</td>
</tr>
<tr>
<td></td>
<td>scilab</td>
<td>0.9189</td>
<td>0.9193</td>
</tr>
<tr>
<td>gzip</td>
<td>input.graphics</td>
<td>0.9291</td>
<td>0.9303</td>
</tr>
<tr>
<td></td>
<td>input.program</td>
<td>0.9022</td>
<td>0.9037</td>
</tr>
<tr>
<td></td>
<td>input.source</td>
<td>0.9364</td>
<td>0.9373</td>
</tr>
<tr>
<td></td>
<td>input.random</td>
<td>0.9186</td>
<td>0.9204</td>
</tr>
<tr>
<td>mcf</td>
<td>inp</td>
<td>0.9935</td>
<td>0.9937</td>
</tr>
<tr>
<td>perfbench</td>
<td>b3.m4</td>
<td>1.0000</td>
<td>1.0000</td>
</tr>
<tr>
<td>twolf</td>
<td>ref</td>
<td>0.8864</td>
<td>0.8900</td>
</tr>
<tr>
<td>vpr</td>
<td>place_log</td>
<td>0.9039</td>
<td>0.9071</td>
</tr>
<tr>
<td></td>
<td>route_log</td>
<td>0.9657</td>
<td>0.9632</td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td>0.9413</td>
<td>0.9433</td>
</tr>
</tbody>
</table>

Table 1. Branch prediction accuracy

A hybrid branch predictor of Alpha EV6 and the benchmark program of SPEC CINT2000 was used for the experiment [14, 15]. The simulation was performed with 5 billion instructions per input data in the SimpleScalar 3.0b [16].

This paper suggests size of saturating counter used in meta predictor as below. Table 1 shows the experimental results of the Alpha EV6 predictor using saturating counters of 2-bit, 3-bit and 4-bit proposes saturating counters. When using a 3-bit saturating counter, the accuracy of branch prediction is 0.2% higher than that with using 2-bit saturating counter. Because it has increased that the state of the saturating counter that stored information of the branch predictor’s weight. Branch predictors can choose a more accurate branch predictor because they can compare with the information about of the branch predictor’s weight. However, 3-bit and 4-bit saturating counters have shown similar branch prediction accuracy results, which means that in the benchmark program that was used in this experiment, the saturating counter doesn’t need to be bigger than 3-bits.

4 Conclusion

Important factors to consider in designing a branch predictor are:

- The way to represent the history of branch decisions made;
- The way to store the history;
- Type of saturating counter to be employed;

In this paper we focus on the third factor. Saturating counter used for a meta predictor obviously affects branch prediction accuracy. We compare types and sizes of saturating counter used in meta predictor, and shows the which type and sized saturating counter has the best the accuracy by experimental results. Next is the transition relation of the saturating counter should transit only when the directions predicted by the branch predictors are different. When the chosen branch predictor is a hit, the saturating counter will increase one state. And when the branch predictor which was chosen by the meta predictor is a mispredict, the saturating counter will decrease one state. We conclude this paper with suggesting that a saturating counter for meta predictor should be 3-bit, if there is no constraint on memory size.

References


