SENSE-REVERSING PROFILE BASED CONFIDENCE ESTIMATOR FOR LOAD VALUE PREDICTION

Kannan Dhanasekaran, Muruganandam Panchalingam and A.P.Shanthi
Department of Computer Science and Engineering
Anna University, Chennai – 600 025, India.
p_murugu@rediffmail.com

Abstract. *This work presents a sense-reversing profile based confidence estimator (SRP CE) for load value prediction. This sense-reversing profile along with a prediction bit serves as an efficient confidence estimator for the Stride2Delta predictor, which outperforms the existing estimators. The drawback of the existing profile based confidence estimators is that they tend to saturate for some history patterns. The basic idea behind the proposed SRP CE is to overcome the saturation of history bits. This is done by making the prediction of the load value evenly spread on all the history patterns. This spread enables the decision logic of the confidence estimator to be implemented with a simpler hardware. The results show that this proposed scheme improves performance significantly.*

1 Introduction

The data dependences occurring between the instructions are more prevalent than the control dependences in a program. An obvious way to overcome these data dependences is to perform data speculation. Data speculation can be implemented by either *storage address prediction* or *data value prediction*. Recent studies have shown that the value prediction techniques have higher potential for performance improvement [1, 2]. *Stride value predictor* a type of *Computation based predictor* is shown to give performance when compared with other types of predictors [3,4,5]. The Stride predictor predicts values to be the sum of the last value of the load and some stride. A notable variant of stride value predictor is the "*Stride2delta predictor*". In this, two strides are maintained and the prediction value is the sum of the last value and one of the two strides. The main use of the extra stride is to reduce the learning at the change of the repeating sequence to be predicted. The learning of the stride2delta predictor is dependent on the existence of a dynamic "*Confidence Estimator (CE)"*, which is necessary for the prediction algorithm to decide whether to predict a value or not [5]. The Confidence Estimator consists of extra hardware along with the predictor. Hence for efficient implementation of the predictor, the role of the confidence estimator is important.

The rest of this paper is organized as follows. Section 2 discusses about the related work. Section 3 explains the proposed SRP CE. Section 4 details the simulations that were carried out and analyses the results obtained, while section 5 concludes the paper.

2 Related Work

Two commonly used dynamic confidence estimators are the *Bimodal CE* and the *SAg CE* [3]. In the case of the bimodal CE, a saturating up/down counter is used to count the correct and incorrect predictions for a particular load. If the counter value is above a particular threshold, then the value for that particular instance of load instruction is predicted. The main drawback of this Bimodal CE is that it cannot adapt quickly to alternating patterns of predictable and unpredictable loads. To overcome this limitation of the bimodal CE, a new CE has been proposed in [3] by the name SAg CE.

In the case of the SAg CE, the outcome of the prediction is stored in a bit-pattern (called a *history*), where the n^{th} bit represents the outcome of the n^{th} last prediction. Here "1" is shifted into the history pattern, when the prediction is correct and "0" is shifted into the history pattern, when the prediction is incorrect. In addition to the history patterns, saturating up/down counters are used to record the number of correct or incorrect predictions appropriately for the obtained history pattern. Only if the counter value is above a particular threshold, the predictor predicts the value. It has been found by research that four-bit history patterns yield good prediction accuracies [3]. Taking the case of four-bit history patterns, sixteen possible history patterns are available. A saturating up/down counter has been used for each of the history pattern and hence, sixteen counters are needed [3]. But, it has been found that these history patterns saturate in some patterns, tending to reduce the percentage of correct predictions, though the accuracy of the prediction is high. This is because of the fact that the prediction based on the history is not evenly spread on all the history patterns. In addition to this limitation, the counters incur an added complexity to the design logic and hardware cost.

In this work, a new Sense Reversing Profile (SRP) based CE is proposed. It overcomes the saturation problem encountered in the SAg CE and reduces the hardware cost and complex design logic of SAg CE [3] as well.

3 Design of the proposed *SRP CE*

The proposed SRP based confidence estimator uses the profile information to predict the load instructions. The profiling technique that is followed here enables to overcome the saturation of history pattern that

occurs at the extreme values. The architecture of the proposed SRP CE is shown in figure 1.

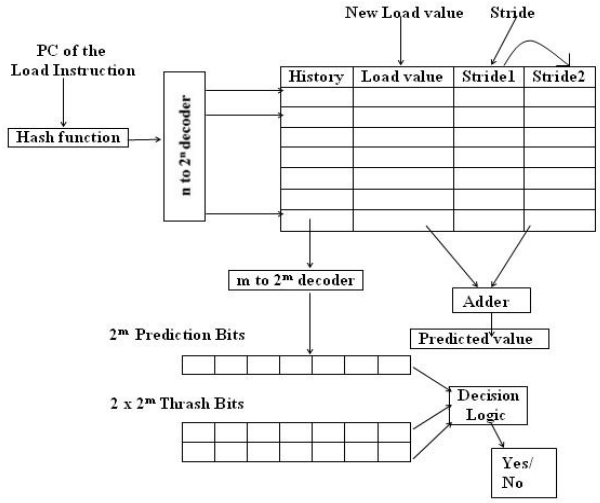

Fig.1 SRP CE hardware setup and control logic along with stride2delta predictor

The architecture of the proposed SRP CE has two modules, the *profiling module* and the *decision logic*. The profiling module manipulates the history bits in accordance with the correctness of the predictions. The history bits for a load instruction are selected using the program counter (PC) of that instruction. The PC indexes into the stride predictor table, which has separate fields for history, load value, stride1 and stride2 as shown in figure.1.

The decision logic comprises of the prediction bits and the thrash bits that decide whether to predict the load value or not. These modules are discussed in detail in the following sections.

3.1 Profiling module

The profiling module of the SRP CE gives the profile information for the load instruction during the *lookup phase*. In the lookup phase, the predictor table is referred to calculate the predicted load value. The proposed SRP CE technique is implemented for the stride2delta predictor.

Stride2Delta predictor uses stride values to predict the next load value. This predictor uses two stride values for each load instruction. The first stride is updated after each time the load instruction reaches the update stage of its execution. The second stride is updated only if the first stride and the current stride values are equal, and this second stride value is used to predict the next load value. The stride is calculated as the difference between the current load value and the previous value. Once the stride predictor enters the steady state in which the two stride values are equal, it starts predicting. The profiling methodology used by the proposed CE is elaborated below.

Profiling Methodology

A *four-bit* profile information is maintained for each load instruction. The information whether the previous instance of the load instruction was predicted correctly or not is kept in the profile. Initially, the profile pattern is set to *0000.* For each correct prediction a 1 is shifted in, for example 0111 becomes 1011 and for every misprediction a 0 is shifted into the profile, for example 0111 becomes 0011. This shifting of a *1* or a *0* is done during *update phase*. This type of profiling is continued until the upper limit of *1111* is reached. Once this *upper saturation point* is reached, the sense for a correct prediction and a misprediction is reversed. That is, hereafter a "*0"* is shifted in for a correct prediction and a "*1*" is shifted in for a misprediction from now on. This continues till *0000* is reached, which is the *lower saturation point*. Eventually at this point the sense is reversed and the process is repeated. This technique gives an equal spread for prediction among the sixteen possible history patterns and enhances the functionality of the existing profile based methods. This history pattern along with the decision logic resolves the outcome of the prediction.

3.2 Decision Logic

The main advantage that enriches this kind of profiling is that the decision logic used to decide whether to predict a load value or not is based on just a single bit that corresponds to that history pattern. For a profile of n history bits, a total of 2^n bits are used in the prediction logic. So for a four-bit profile, sixteen-bit (2^4) prediction logic is used. The profile pattern is used to index into the prediction logic. That is, *0000* is mapped onto the 0^{th} bit of the profile pattern, *0001* is mapped onto the 1^{st} bit, etc., and *1111* is mapped onto the 15^{th} *bit* of the profile pattern.

Depending on this prediction bit, the prediction is carried out. During the lookup phase, a *0* in this bit position indicates that a prediction cannot be done *(no prediction)* and a *1* in this bit position indicates that a prediction can be done *(prediction)*. Initially, all the sixteen prediction bits are set to *1* because the decision logic is consulted only after the stride2delta predictor reaches a steady state [3]. If a bit corresponding to a history pattern is *1 (prediction)* and it results in a *misprediction* then that bit is *reset to 0*. Similarly, if a bit corresponding to a history pattern is *0 (no prediction),* and if found to be predicted correctly by the stride predictor after the actual load had finished, then that bit is *set to 1*. In the case of a misprediction, the *re-fetch* mechanism is used [3] to correct the pipeline, because it is executing instructions with wrong inputs due to the misprediction.

Advantage of prediction bits

The setting and resetting of the prediction bit gives a *hardware design* that is easy to implement eliminating the complexities of the existing confidence estimators that use counters and some threshold values.

Thrash Bits

The prediction bits try to learn the sequence of values that a load instruction retrieves. During this learning phase mis-predictions are possible and this is considerably low because the *CE* learns the sequence of the load values within a few iterations. Once the learning is finished, there are no further mispredictions.

For example, the learning phase for a sequence of length 6 (1, 2, 3, 4, 5, and 6) is shown in table 1: Column 1 is the last value, column 2 is the history information, column 3 is the predicted value and column 4 is Prediction bit. The comments given alongside the figure denote the particular action taken for that entry.

Last Value	History Bits	Pred Value	Pred Bit	
3	0000	4	1	Iteration 1: steady state reached at value 3.
4	1000	5	1	
5	1100	6	1	
6	1110	7	0	Misprediction: So Pred_bit is reset to 0.
2	0111	3	1	Iteration 2: steady state reached at value 2.
3	1011	4	1	
4	1101	5	1	
5	1110	-	1	Pred_bit =0 but stride correct, so Pred_bit = 1.
6	1111	7	0	Misprediction: So Pred_bit is reset to 0
2	0111	3	1	Iteration 3: Learning complete. No further mispredictions.

Table 1. Learning phase of the SRP CE

But there is a possibility that the sequence may lead to a situation where the same history pattern has to predict the value during one iteration and should not predict in the next iteration. This situation is termed as *"thrashing"*. In the case of thrashing, a correct prediction may be prevented and an incorrect prediction may be allowed to proceed, thereby reducing the overall performance of the CE greatly. To overcome thrashing, two thrash bits are added to each of the prediction bits. For a profile of n bits, totally $2*2^n$ thrash bits are used. The thrash bits are initialized to 00. If a misprediction occurs for a particular history pattern, then the prediction bit is reset to 0 and a 1 is shifted into the thrash bits. So the thrash bits become 10. Now if another misprediction occurs at the same history pattern, then again the prediction bit is reset and another 1 is shifted into the thrash bits. Now the thrash bits are both set and show 11. Once this stage is reached, it indicates that a thrashing is taking place at that history pattern. So from now on, the thrash bits guide every prediction for that history pattern. Since both the thrash bits are set, the prediction logic decides whether to predict the load value or not, by "*flipping*" the prediction bit. This implies that if the thrash bits are 11 and the prediction bit is 0, then it is flipped to 1 and prediction is done. Similarly, if the prediction bit is 1, then it is flipped to 0 and no prediction is done. This mechanism to overcome thrashing is illustrated in table 2 for a sequence of length 9 (1, 2, 3, 4, 5, 6, 7, 8 and 9). Column 1 is the last value, column 2 is the history information, column 3 is the predicted value, column 4 is prediction bit and column 5 is thrash bits.

The two-thrash bits essentially eliminate two-way thrashing in which the prediction alternates between successive iterations. Three-way thrashing and other higher n-way thrashing will occur only for sequences that are very long and rare, and are not considered. However, increasing the number of thrash bits can eliminate this n-way thrashing. This implementation uses only two thrash bits.

Last Value	History bit	Pred Value	Pred Bit	Thrash Bits	
3	0000	4	1	00	Iteration 1: Stride predictor enters steady state when value is 3.
4	1000	5	1	00	
5	1100	6	1	00	
6	1110	7	1	00	
7	1111	8	1	00	Sense is reversed.
8	0111	9	1	00	
9	0011	10	0	10	Misprediction: Reset Pred_bit and shift a 1 into thrash_bits.
2	1001	3	1	00	
3	0100	4	1	00	
4	0010	5	1	00	
5	0001	6	1	00	
6	0000	7	1	00	Sense is reversed.
7	1000	8	1	00	
8	1100	9	1	00	
9	1110	10	0	10	Misprediction: Reset Pred_bit to 0 and shift a 1 into thrash_bits.
2	0111	3	1	00	
3	1011	4	1	00	
4	1101	5	1	00	
5	1110	-	1	10	No prediction: Set Pred_bit to 1.
6	1111	7	1	00	Sense is also reversed.
7	0111	8	1	00	
8	0011	-	1	10	No prediction: Set Pred_bit to 1.
9	0001	10	0	10	Misprediction: Reset Pred_bit to 0 and shift
2	1000	3	1	00	a 1 into thrash_bits.
3	0100	4	1	00	
4	0010	5	1	00	
5	0001	-	1	10	No Prediction: Set Pred_bit to 1.
6	0000	7	1	00	Sense is reversed.
7	1000	8	1	00	
8	1100	9	1	00	
9	1110	10	0	11	Misprediction: Reset Pred_bit to 0 and shift
2	0111	3	1	00	a 1 into thrash_bits. (Thrashing)
3	1011	4	1	00	
4	1101	5	1	00	
5	1110	6	1	11	Flip Pred_bit and prediction done.
6	1111	7	1	00	Sense is reversed.
7	0111	8	1	00	
8	0011	9	1	00	
9	0001	10	0	11	Misprediction: Reset Pred_bit to 0 and shift
2	1000	3	1	00	a 1 into thrash_bits. (Thrashing) Learning ends. No further mispredictions.

Table 2. Mechanism to overcome thrashing

Since the prediction and thrash bits are global, the setting of thrash bits by one sequence once will result in wrong predictions for another sequence. This can be avoided by placing the prediction and thrash bits for each entry in the CE, but it will increase the CE size. But it has been proved in [3], that the number of long sequences in any program is very less. So thrashing is also less, since only long sequences cause thrashing. Hence, the prediction and the thrash bits can be kept global, without considerable performance degradation.

3.3. Algorithm

The algorithm for implementing this confidence estimator is given below. It has two phases: *lookup* and *update.*

Initially, the true_sense is set to 1.

```
Lookup Phase
Start:
  If thrash_bits = = 11 then  /* Thrashing case */
    If Pred_bit = =1 then
      Pred_bit = 0
      No Prediction done
    Else                      /* Pred_bit = = 0 */
      Pred_bit = 1
      Prediction done
  Else                        /* No thrashing */
    If Pred_bit = 1
      Prediction done
    Else                      /*Pred_bit = = 0 */
      No Prediction done
End.
Update phase
Start:
  If Prediction correct then
    Set Pred_Bit to 1
    If ( history bits = = 1111 )
      true_sense = 0
    If ( history bits == 0000 )
      true_sense = 1
    If ( true_sense ) then
      Shift a 1 into the history bits
    Else
      Shift a 0 into the history bits
Else                          /* Prediction wrong */
  Reset Pred_bit to 0
  Shift a 1 into thrash_bits
  If ( true_sense ) then
    Shift a 0 into the history bits
  Else
    Shift a 1 into the history bits
End.
```

The performance of the proposed SRP CE and the results are discussed below.

4 Simulation and results

The simulation for the SRP CE has been done with the Simplescalar toolset [6]. The results have been obtained for the SPEC2000 benchmark suite [7]. The baseline architecture that is assumed for this simulation has the following specifications:

- four way superscalar processor,
- 128-entry instruction window,
- 32-entry load store buffer,
- 4 integer and 2 floating point units,
- a 64 KB two-way set associative L1 instruction-cache,
- a 64 KB two-way set associative L1 data-cache,
- a 4 MB unified direct mapped L2 cache,
- a 4096-entry branch target buffer and
- a 2048-line hybrid gshare-bimodal branch predictor.

On execution, an instruction count of *1 billion* was used to warm up the caches by fast forwarding option provided with Simplescalar v3.0 and the statistics was obtained for another instruction count of *1 billion.* The results obtained for the simulation of the SPEC 2000 benchmarks are shown in following figures.

Figure 2 shows the prediction percentages, which includes the number of correct predictions done, number of incorrect predictions and the number of predictions not done. The correct prediction percentage is in the average of 48%, which is better when compared with results obtained in [3], which gives an average of 40%. Even though the correct prediction percentage for *gzip* is very low, SRP CE incurs a very low incorrect prediction rate of 0.6413% for that benchmark too.

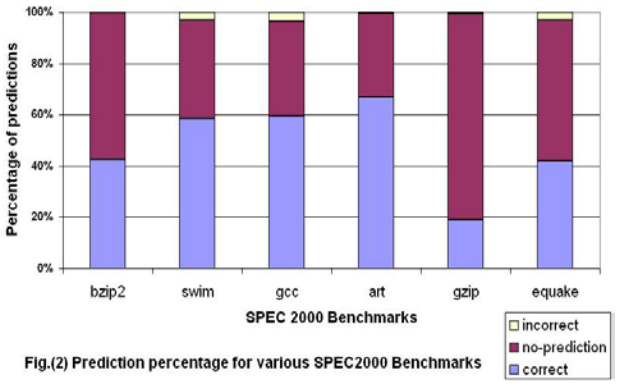

Fig.(2) Prediction percentage for various SPEC2000 Benchmarks

The *"accuracy of the prediction"* is defined as the probability that the attempted prediction is correct [3].

Accuracy of the prediction can be defined as:

$$\textbf{Accuracy} = \frac{\textbf{No. of Correct predictions}}{\textbf{Total no. of predictions made}}$$

Figure 3 shows the percentage of accuracy of predictions for selected benchmarks of SPEC2000 suite. The accuracy achieved is high ranging from 93% - 99.99%, which is quite high when compared with existing accuracies for stride2delta value predictor CE's [3, 5]. In particular the *bzip2* and *art* benchmarks show accuracy as high as 99%, with the percentage of incorrect predictions less than 0.2%. The prediction accuracy is low with the value of 93% for *gcc* benchmark. The average value of accuracy for these benchmarks is 96.5%.

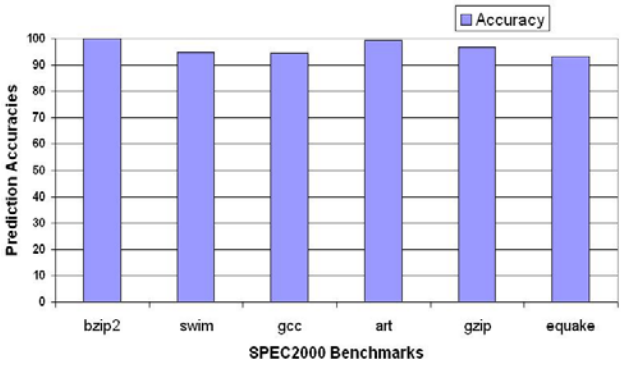

Fig (3) Prediction Accuracies for various SPEC2000 Benchmarks

Another performance measurement that is used is the speedup over the baseline architecture.

The speedup over the baseline architecture is given as

$$\text{Speedup over baseline architecture} = \frac{\text{New IPC - Old IPC}}{\text{Old IPC}} * 100\%$$

Figure 4 shows the speedup over the baseline architecture for selected benchmarks of the SPEC2000 benchmark suite. The speedup values are between 2 and 9. On an average the speedup over the baseline architecture is 4 for the selected five benchmarks of SPEC 2000 suite.

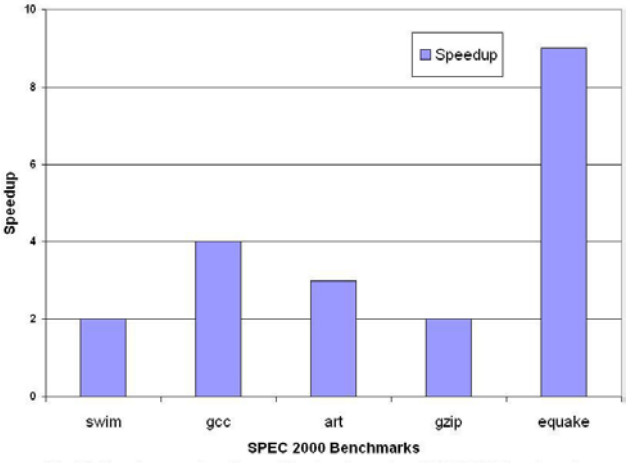

Fig (4). Speedup over baseline architecture for various SPEC 2000 benchmarks

Thus the SRP CE shows very good accuracy, prediction percentages and speedups over the baseline architecture, to give an improvement in performance. Its main impact is that it has considerably increased the correct prediction rate.

5 Conclusion

Analyzing why this scheme works, the fact that allowing the profile pattern to stabilize itself is more convincing than to fix that the pattern can go only till the upper or lower saturation point. Hence, the proposed SRP CE gives very good results than the existing profile based confidence estimators. The SRP CE provides better accuracies and prediction percentages with reduced hardware and simpler decision logic, since only manipulation with bits is involved. This confidence estimator can be extended to the other computation based prediction techniques like the *last 'n' value predictor* and the *register value predictor* and should yield similar results.

REFERENCES

[1] Y.Sazeides and J.E.Smith, "The Predictability of Data Value", *proceedings of the 30th Annual ACM/IEEE International Symposium on Microarchitecture*, pages 248-278, December 1997.

[2] M.H.Lipasti, C.B.Wilkerson and J.P.Shen, "Value Locality and Load Value Prediction", *proceedings of the 7th International Conference on Architectural support for Programming Languages and Operating Systems*, pages 138-147, October 1996.

[3] Martin Burtscher, "Improving Context-Based Load Value Prediction", *Ph.D. Thesis, Department of Computer Science, University of Colorado*, 2000.

[4] K.Wang and M.Franklin,"Highly Accurate Data Value Prediction using Hybrid Predictors", *proceedings of the 30th Annual ACM/IEEE International Symposium on Microarchitecture*, pages 281-290, December 1997.

[5] F.Gabbay and A.Mendelson, "Speculative Execution based on Value Prediction", *EE Department TR #1080, Technison – Israel Institute of Technology*, November 1996.

[6] Todd Austin and Doug Burger, "Simplescalar Simulatorv3.0", *www.simplescalar.org*

[7] SPEC2000 binaries,"*www.spec.org"*