Table Size Reduction for Data Value Predictors by Exploiting Narrow Width Values

Toshinori Sato Itsujiro Arita

Department of Artificial Intelligence
Kyushu Institute of Technilogy
680-4 Kawazu, Iizuka, 820-8502 Japan
{tsato,arita}@ai.kyutech.ac.jp

Abstract

Recently, the practice of speculation in resolving data dependences has been studied as a means of extracting more instruction level parallelism (ILP). An outcome of an instruction is predicted by value predictors. The instruction and its dependent instructions can be executed simultaneously, thereby exploiting ILP aggressively. One of the serious hurdles for realizing data speculation is huge hardware budget of the predictors. In this paper, we propose a technique reducing the budget by exploiting narrow width values. The hardware budget of value predictors is reduced by up to 45.1%. Simulation results show that the technique, called 2-mode scheme, maintains processor performance with slight decrease of the value prediction accuracy.

Keywords: instruction level parallelism, data speculation, value prediction, narrow width operands, hardware implementation

1 Introduction

In order to improve performance, modern microprocessors rely on exploiting instruction level parallelism (ILP). However, ILP is limited by dependences between instructions. They are classified into three classes — control, name, and data dependences. There are many studies to reduce control and name dependences, but data dependences remains as a major bottleneck limiting ILP. Recently, the practice of speculation in resolving data dependences has been studied as a means of extracting more ILP. An outcome of an instruction is predicted by value predictors. The instruction and its dependent instructions can be executed simultaneously, thereby exploiting ILP aggressively.

However, in order to efficiently utilize value prediction without incurring misspeculation penalties, high prediction accuracy is required for value predictors. Some predictors such as hybrid [17] and context-based [14] predictors achieve significantly high prediction accuracies at considerable hardware cost. Recently, some studies [9, 12] are performed to reduce the hardware cost. They are based on classifying instructions according to their predictabilities and utilize value history table (VHT) efficiently. In this paper, we propose an alternative technique to reduce the hardware cost.

Brooks et al. [1] have found that across SPECint95 benchmark programs, over 50% of integer operands are 16 bit or less. That is, high-order bits of the VHT are merely utilized. Therefore, we propose to keep only low-order bits of data values in the VHT to reduce its hardware cost. Our expectation behind the proposal is as follows. If highly predictable values had narrow bitwidths and if it were very difficult to predict wide bitwidth values, limiting the bitwidths of values held in the VHT would not have serious impact on value prediction accuracy.

The organization of the rest of this paper is as follows. In Section 2, related works are surveyed. Section 3 describes our evaluation environment. In Section 4, relationship between data bitwidth and value prediction accuracy is investigated. Based on the investi-

gation, we propose 2-mode scheme in Section 5 for reducing the hardware cost by limiting VHT's data width with maintaining data value predictability. Finally, our conclusions are presented in Section 6.

2 Related Work

Data speculation [6,8] is a technique which executes instructions speculatively using predicted data values. Data dependences are speculatively resolved and thus ILP is increased. There are many studies proposing value prediction mechanisms, some of which achieve prediction accuracy as high as 80% [14, 17]. However, these predictors such as 2-level, hybrid, and context-based predictors require considerable hardware cost for realizing their high prediction accuracies.

Morancho et al. [9], Rychlik et al. [12], and Calder et al. [3] examine capacity constraints of value predictors. Morancho et al. [9] and Rychlik et al. [12] proposed to reduce the hardware cost by classifying instructions based on their value predictability. Instructions which are easily predicted use simpler predictors such as last-value and stride predictors, whose hardware cost is low. High-cost predictors such as the 2-level, the hybrid, and the context-based predictors are used only for hard to predict instructions. Calder et al. [3] filter instructions based on their value predictability. Only instructions which are useful for data speculation are held in a value predictor, reducing the number of entries of the predictor. On the other hand, Tullsen et al. [15] remove value prediction hardware completely with the aid of compiler management of values in registers.

In the research area of branch predictors, their high hardware cost is also a serious problem. Fagin [5] proposed to reduce bitwidth of tag array in branch target buffer (BTB). He investigated the tradeoff between the bitwidth and branch prediction accuracy, and found that it was possible to restrict bitwidth of the tag array to 2 bits without severe performance loss. Cascaded predictor proposed by Driesen et al. [4] classifies indirect jump instructions based on their predictability. Easily predictable instructions use low-cost BTB and hard to predict ones use high-cost 2-level adaptive predictor. They found that it is possible to reduce total hardware cost of predictor within 1/4–1/2 of the predictor which is totally based on the 2-level mechanism.

Wang et al. [16] proposed caching address tags (CAT) in order to reduce hardware cost of cache memories. The CAT focuses on reducing tag array cost by exploiting data locality. Since most of the address tags are same, it is possible to cache the address tags which are linked for their associated data using pointers.

3 Evaluation Environment

In this section, we describe the evaluation environment by explaining a processor model and benchmark programs.

3.1 Processor model

We use two types of simulators for this study. One is a functional simulator for counting value prediction accuracy and the other is a timing simulator for evaluating processor performance. We implemented the simulators using the SimpleScalar tool set (ver.3.0a) [2]. The SimpleScalar/PISA instruction set architecture (ISA) is based on MIPS ISA.

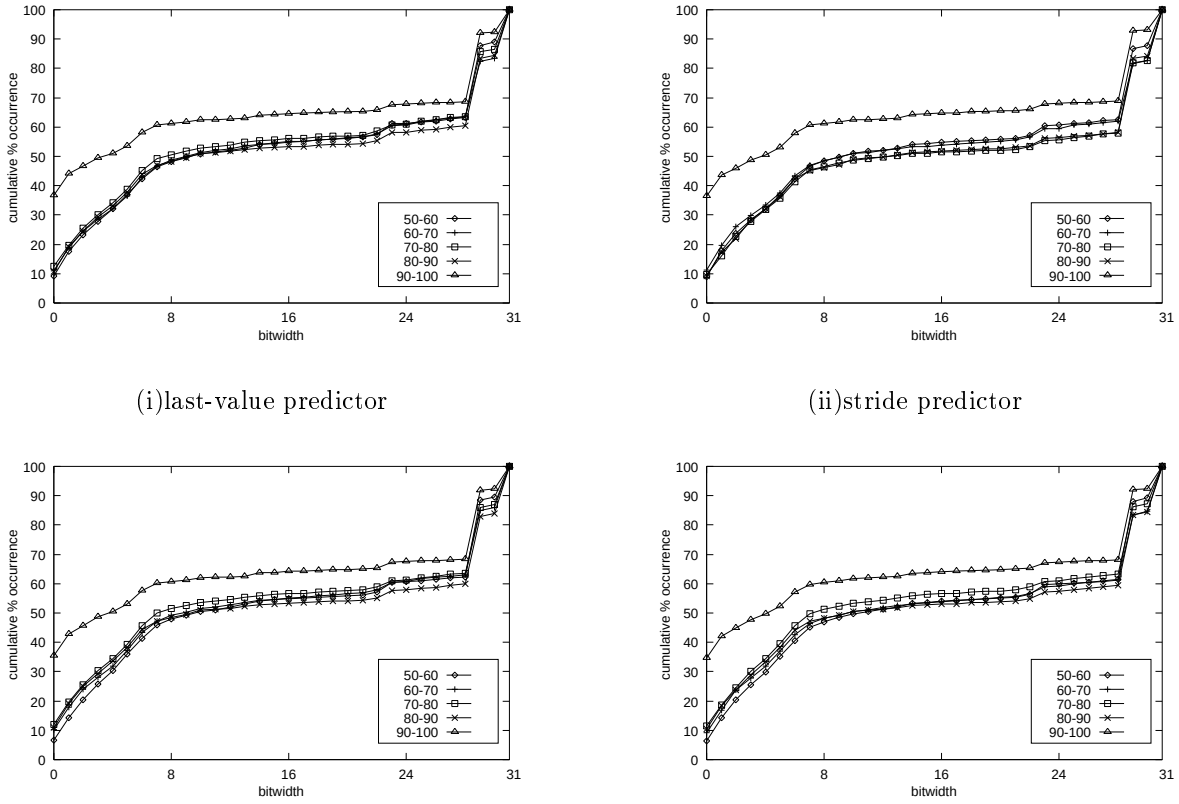
In this paper, we investigate four value prediction mechanisms — last-value predictor [8], stride predictor [6, 17], 2-level adaptive predictor [17], and hybrid predictor consisting the stride and the 2-level predictors [17]. Following Wang et al.'s study [17], we use their state transition for the stride predictor, and history lengths of the 2-level and the hybrid predictors are six and the thresholds initiating prediction of the predictors are three and six respectively.

The timing simulator models a realistic 8-way out-of-order execution superscalar processor based on register update unit which has 128 entries. Each functional unit can execute any operations. The latency for execution is 1 cycle except in the case of multiplication (4 cycles) and division (12 cycles). A 4-port, non-blocking, 128KB, 32B block, 2-way set-associative L1 data cache is used for data supply. It has a load latency of 1 cycle after the data address is calculated and a miss latency of 6 cycles. It has a backup of an 8MB, 64B block, direct-mapped L2 cache which has a miss latency of 18 cycles for the first word plus 2 cycles for each additional word. No memory operation can execute that follows a store whose data address is unknown. A 128KB, 32B block, 2-way set-associative L1 instruction cache is used for instruction supply and also has the backup of the L2 cache which is shared with the L1 data cache. For control prediction, a 1K-entry 4-way set associative BTB, a 4K-entry gshare-type 2-level adaptive branch predictor, and an 8-entry return address stack are used. The branch predictor is updated at instruction commit stage.

When a misspeculation occurs, it is necessary to revert processor state to a safe point where the speculation is initiated. We implement an instruction reissue mechanism which selectively flushes and reissues misspeculated instructions in the timing simulator.

3.2 Benchmark programs

The SPEC95 CINT benchmark suite is used for this study. The input files are modified so that evaluation time is practical. Table 1 lists the benchmarks and the



(i)last-value predictor

(ii)stride predictor

(iii)2-level predictor

(iv)hybrid predictor

Figure 1: Bitwidth for instructions according to their prediction accuracies

Table 1: Benchmark programs

program	input set	#inst.s	%cand
099.go	9 9	133M	77.69
124.m88ksim	dcrand.big	120M	68.52
126.gcc	genrecg.i	117M	65.96
129.compress	14000 e 2231	48M	66.25
130.li	queens 7	202M	58.76
132.ijpeg	specmun.ppm	54M	79.22
134.perl	primes.in	10M	62.04
147.vortex	persons.250	101M	59.94

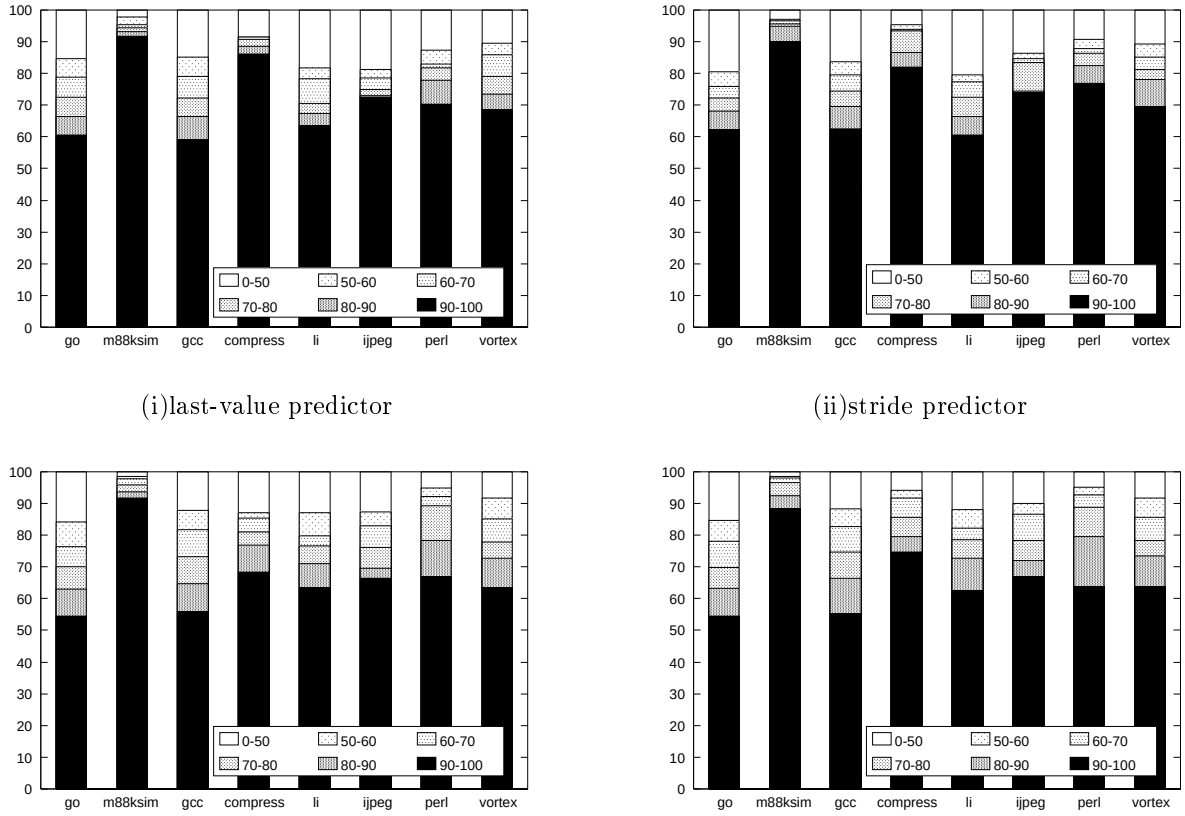
input sets. We use the object files provided by University of Wisconsin Madison [2], except `132.ijpeg` which is compiled by GNU GCC (version 2.6.3) with the optimization option, `-O3`. Each program is executed to completion. Table 1 also shows the number of instructions executed and the percentage of candidate instructions predicted by the value predictors. The candidate instructions are register-writing ones, and do not include branch and store instructions. We count only committed instructions.

4 Simulation Results

In this section, we investigate relationship between data bitwidths and their predictability. In order to evaluate value prediction accuracies, the functional simulator with infinite VHT is used.

4.1 Relationship between data width and prediction accuracy

In this study, we show that a wide rage of easily predictable instructions have small data sizes. Figure 1 illustrates this by showing the cumulative percentage of instructions in whole SPECint95 whose predicted values are less than or equal to the specific bitwidth according to their prediction accuracies. This is evaluated for the cases where instructions whose prediction accuracies are 50–60%(circle), 60–70%(bar), 70–80%(rectangle), 80–90%(cross), and 90–100%(triangle), respectively. It is found that eas-



(i)last-value predictor

(ii)stride predictor

(iii)2-level predictor

(iv)hybrid predictor

Figure 2: Percentage of instructions according to their prediction accuracies

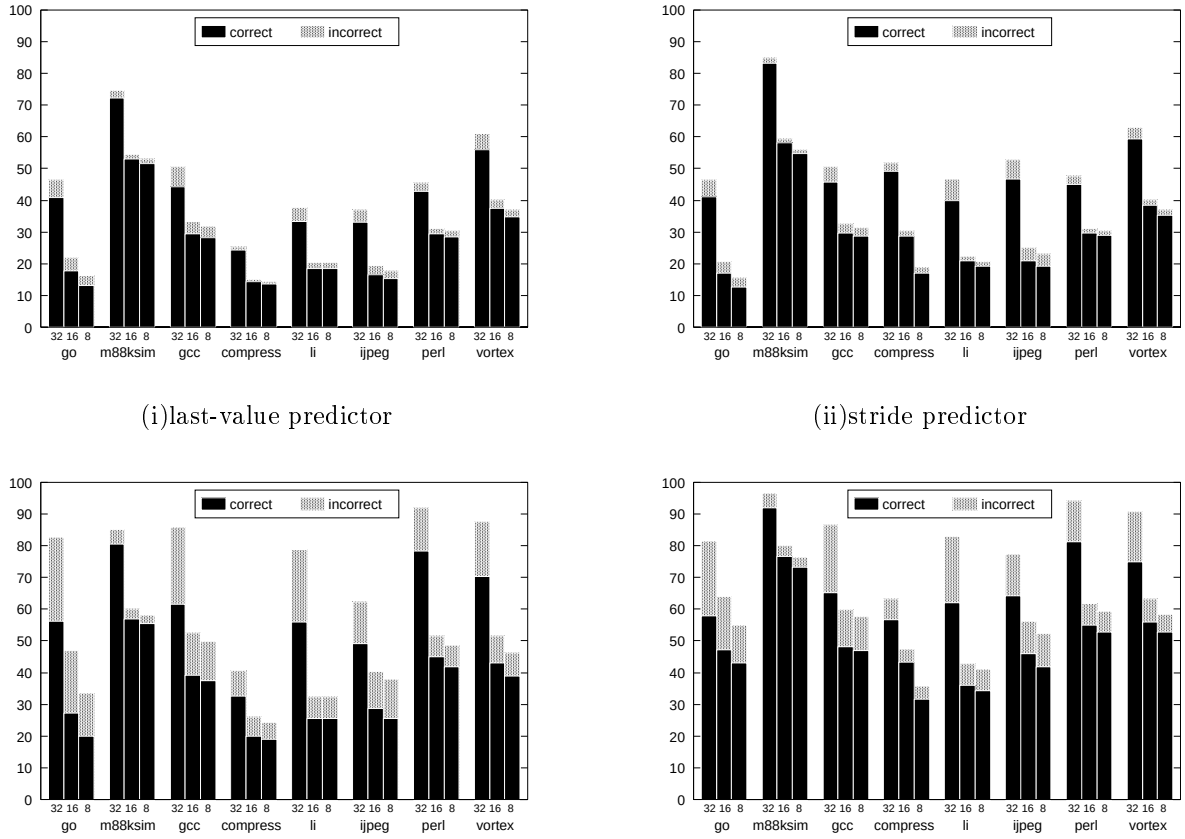
ily predictable instructions whose prediction accuracies are over 90% have smaller size values than other cases. This confirms our expectation explained in the previous section that highly predictable values have narrow bitwidths.

Figure 2 shows percentages of dynamic instructions which belong to specified prediction accuracies — 50–60%, 60–70%, 70–80%, 80–90%, and 90–100%, respectively. For example, in the case of `099.go`, 60.5% of dynamic instructions are predicted with the accuracy of over 90% on the last-value predictor. It is generally seen for every program on each predictor that instructions predicted with the accuracy of over 90% occupy more than 50% of total predicted instructions.

From the simulation results shown in Figures 1 and 2, we can find the followings. Easily predictable instructions which are predicted with the accuracy of over 90% have relatively narrow data. In addition, such instructions occupy over half of total predicted instructions. Thus, it is expected that the predictability of such instructions may suffer little impact when value predictors restrict bitwidths of their predicted data. It should be noted that some instructions affect severe damage from the restriction of data width, however, they are originally hard to predict. In addition, the percentage of hard to predict instructions are considerably smaller than that of easily predictable ones. From these observations, we expect that limiting bitwidths of predicted data may not degrade value prediction accuracy.

4.2 Effect of restricting data width on predictability

Based on the simulation results in the previous section, we investigate how restricting bitwidth affects value predictability. Since integers are expressed in 2's complement form, a narrow value predicted is sign-extended before compared with its actual value. Figure 3 shows value prediction rates for 32bit-, 16bit-, and 8bit-width value predictors. Each value predictor has infinite VHT. We define the prediction rate as the number of correctly predicted instructions over that of all instructions which are candidates for prediction.



(i)last-value predictor

(ii)stride predictor

(iii)2-level predictor

(iv)hybrid predictor

Figure 3: %Prediction rate when VHT's bitwidth is limited

For each group of three bars, the first one is for a 32bit-width VHT for each predictor. The remaining bars are for 16bit- and 8bit-width ones respectively. Each bar is divided into the two parts. The lower part indicates the prediction rate, and the upper part indicates the percentage of mispredictions. Since some instructions are not predicted due to low confidence, the sum of two parts does not always equal to 100%. Different from our expectation, the prediction rate is significantly affected by restricting predicted data width. In order to maintain the predictability, we have to consider any technique to predict large values. It is also found that performance of 8bit-width predictors is comparable to that of 16bit-width ones.

5 2-mode Prediction Scheme

In the previous section, we found that narrow bitwidth VHTs degrades value predictability severely. In this section, we propose 2-mode prediction scheme to predict large values which can not be predicted by restricting predicted value width. The basic concept behind the 2-mode scheme is that the number of instructions which can not be predicted by the restriction may be small.

5.1 Data width distribution

Table 2 shows bitwidth distribution of values produced by each instruction. It is the difference of the largest value and the smallest value for a static instruction. As can be easily seen, the distribution is very small. Bitwidths of correctly predicted values for 80% of instructions do not change. For every predictor, bitwidths of correctly predicted values for 90% of instructions changes less than 5 bits. Therefore, an instruction keeps on generating narrow data once it produces a narrow value. In other words, instructions generating large values are limited, and thus predictors for predicting large values have to consider such limited instructions.

5.2 Prediction mechanism

From the observations in Section 5.1, it is expected that large and small data can be predicted by different

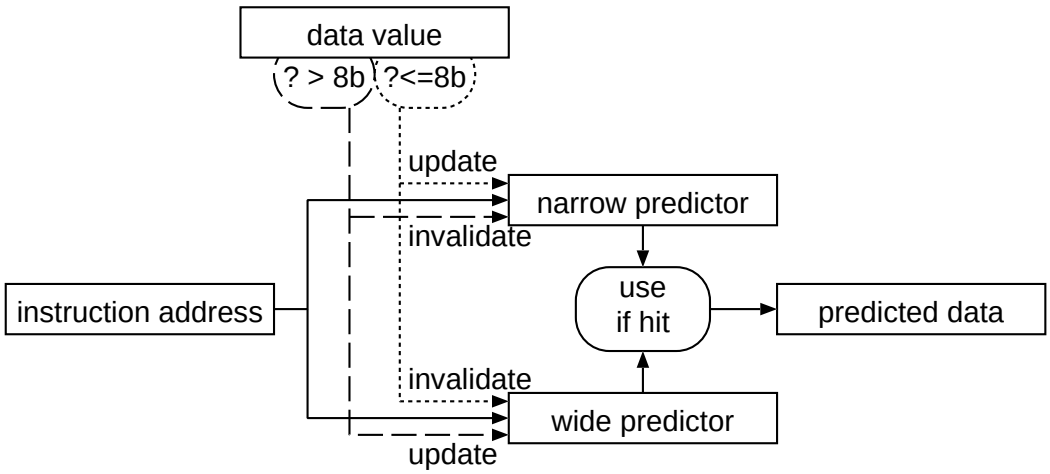


Figure 4: 2-mode value prediction mechanism

Table 2: %Distribution of bit lengths changed

predictor	bit length					
	0	1	2	3	4	>4
last-value	83.92	4.51	2.50	1.69	1.47	5.91
stride	83.40	4.20	2.77	1.81	1.55	6.27
2-level	80.37	5.24	2.96	1.97	1.79	7.67
hybrid	79.13	5.32	2.99	2.05	1.93	8.58

predictors each of which is specified by predicted data size. Based on this expectation, we propose 2-mode prediction scheme which combines two data value predictors — one is a predictor for narrow width data and the other is a predictor for wide width ones.

Figure 4 explains the 2-mode value prediction mechanism. It consists of narrow and wide predictors. The narrow predictor is responsible for predicting small size values (in the case of Figure 4, the threshold size is 8.). On the other hand, the wide predictor takes part in predicting large size values. The narrow and the wide predictors use an identical prediction scheme — one of the last-value, stride, 2-level, and hybrid schemes. It is possible to apply different prediction scheme for each predictor, but this consideration is beyond the scope of this paper and is future study. When updating value history for an instruction, only one predictor is selected based on the value size. When the size is less than a threshold, the narrow predictor is updated. Otherwise, the wide one is updated. The remaining predictor which is not selected invalidates the entry for the instruction if it is registered. That is, there is at most one value history for every instruction. When predicting a value for an instruction, both predictors are referred. As described above, at most one predictor must provide a predicted value. Therefore, any special mechanism for selecting one from predicted values is not necessary for the 2-mode scheme.

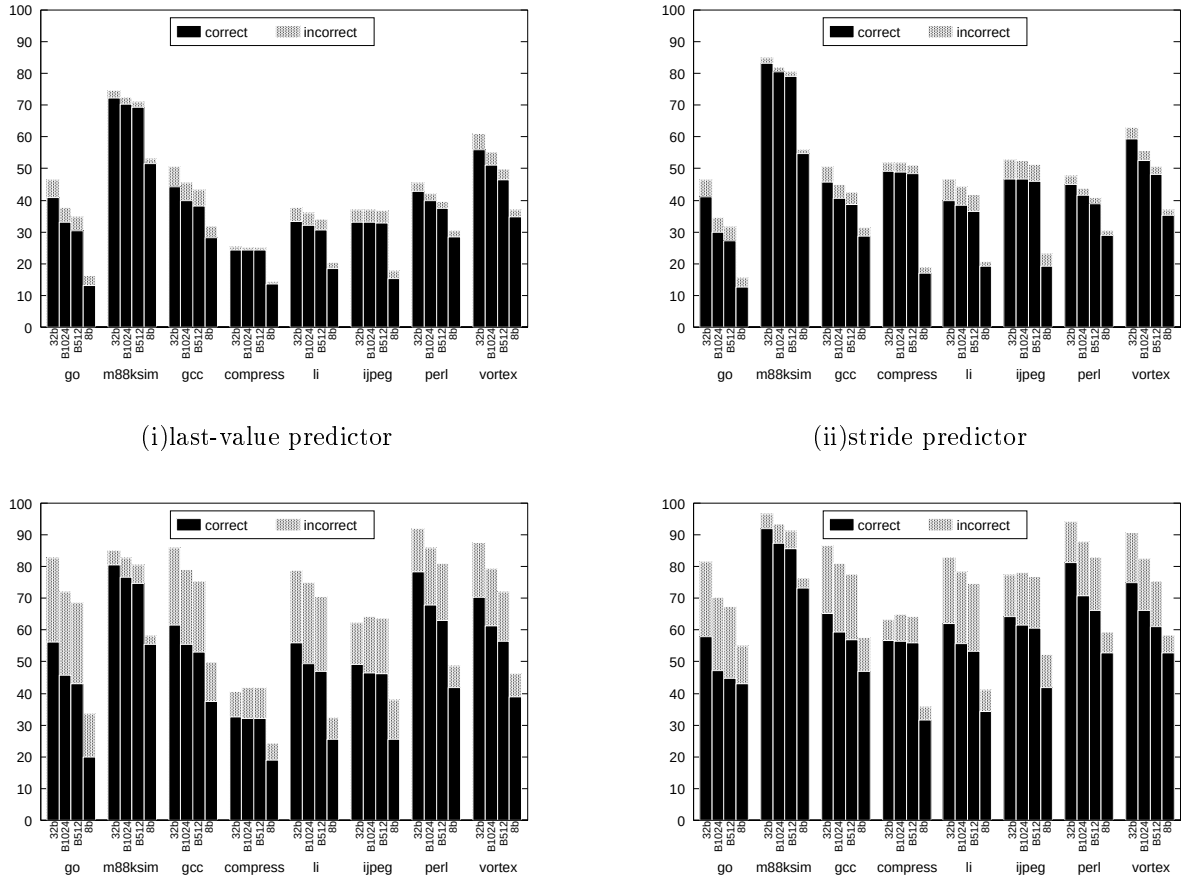
Using the wide predictor makes it possible to predict large values which can not be predicted by 16bit- and 8bit-wide predictors evaluated in the previous section. In addition, selecting one from the wide and narrow predictors prevents the 2-mode scheme by necessitating large hardware budget. This is because most of the values are small and predicted by the narrow predictor and thus the wide predictor whose hardware budget may be considerable should not have large VHT.

5.3 Evaluation

First we evaluate predictability and then processor performance.

When evaluating predictability, the narrow predictor has infinite VHT and the wide predictor has a direct-mapped VHT whose capacity is limited to between 512 and 1024. Table 3 shows the number of VHT entries utilized by the predictors evaluated in the previous sections. In comparison with the total entries utilized, 512 and 1024 entries are 0.7–24.5% and 1.4–49.1% respectively. It should be noted that the wide predictor uses a direct-mapped VHT whose entries may not be efficiently utilized while the narrow predictor has a full-associative VHT.

Figure 5 shows value prediction rates when the 2-



(i)last-value predictor

(ii)stride predictor

(iii)2-level predictor

(iv)hybrid predictor

Figure 5: %Prediction rate of 2-mode predictors

Table 3: # of VHT entries used

program	#entries
099.go	40,496
124.m88ksim	7,064
126.gcc	71,032
129.compress	2,087
130.li	3,768
132.ijpeg	1,3240
134.perl	7,498
147.vortex	35,152

mode scheme is utilized. Its component predictors are a 32bit-width wide predictor and an 8bit-width narrow predictor. We decided that since the prediction rates for the 16bit- and the 8bit-width narrow predictors are comparable. Figure 5 also shows prediction rates for the 32bit- and the 8bit-width predictors. For each group of four bars, the first one is for the 32bit-width predictor. Next two bars are for the predictors utilizing 2-mode scheme where the VHTs of the wide predictors have 1024 and 512 entries respectively. The remaining bar is for the 8bit-width predictor. The layout of Figure 5 is the same as for Figure 3. The lower part indicates the percentage of correct predictions, and the upper part indicates that of mispredictions. It is found that the predictability is improved for every configuration. Thus, it is confirmed that filtering small values by the narrow predictor is possible and the wide predictor only has to predict values passing the filter.

Table 4 presents how frequently histories registered in the VHTs are invalidated due to moving between two VHTs. It should be noted that the frequency is identical among four prediction schemes — last-value, stride, 2-level, and hybrid, since this is evaluated using

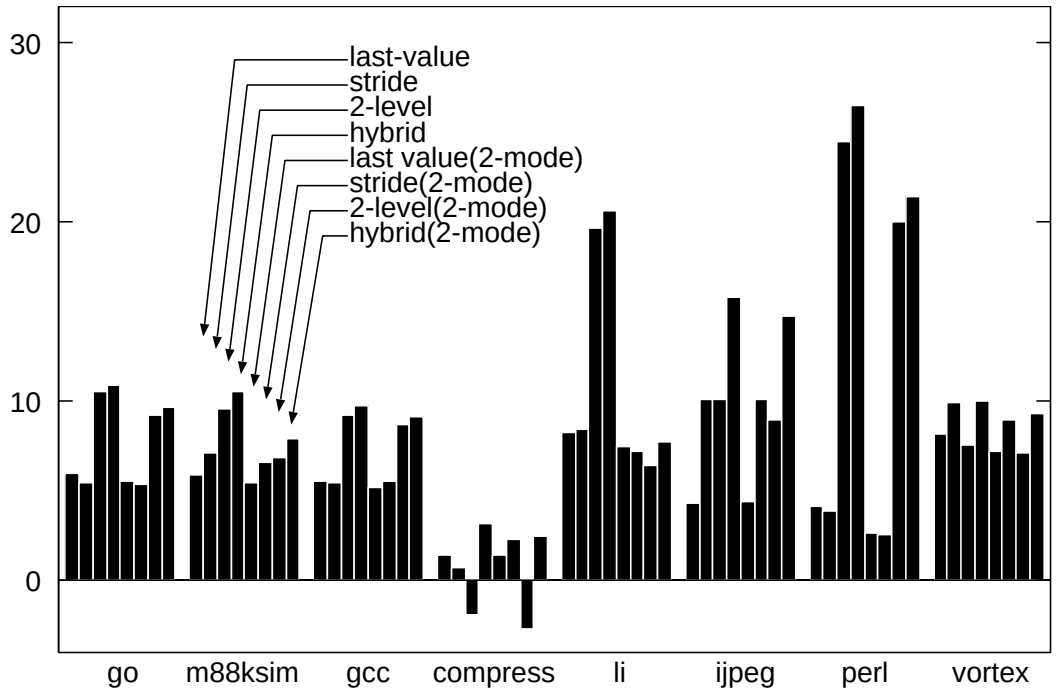


Figure 6: %Processor performance improvement

Table 4: %Frequency of invalidations (functional simulation)

program	%frequency	
	512	1024
099.go	1.74	1.81
124.m88ksim	1.16	1.22
126.gcc	3.20	3.36
129.compress	2.29	2.29
130.li	4.44	4.50
132.ijpeg	1.83	1.87
134.perl	2.81	3.01
147.vortex	3.40	3.85

the functional simulator. As we expected, the invalidations are rare so that it is possible to filter small values using the narrow predictor.

Referring back to Figure 5, we can also observe the followings. In the cases of 2-level and hybrid predictors, it is found that applying 2-mode scheme increases the percentages of mispredictions. This is because the 2-level and hybrid predictors rely on histories more than the last-value and stride ones. The invalidations of histories diminish predictability. In the cases of the 2-level and hybrid predictors, this effect is very severe even when the invalidations are rare as can be seen in Table 4.

Next, we evaluate processor performance when the 2-mode prediction scheme is utilized for hardware reduction. As we explained above, the timing simulator is used for performance evaluation. In order to confirm usefulness on real processors, we use finite VHTs in this case. The narrow and the wide predictors of which the 2-mode scheme consists have direct-mapped 4096- and 512-entry VHTs respectively. We select the number of the 4096 entries since we can find a lot of articles where proposed value predictors are assumed to have 4096-entry tables. Evaluating other sizes and balance between two tables is future study. Pattern history tables used in the 2-level and the hybrid predictors are shared by the narrow and the wide predictors. In this case, the 2-mode scheme can reduce hardware budget as follows. Data arrays are reduced by 62.5% for every prediction scheme. When total hardware is considered, it is reduced by 31.3%, 43.5%, 42.7%, and 45.1% for the last-value, the stride, the 2-level, and the hybrid predictors, respectively.

Figure 6 shows performance improvement rates when the 2-mode scheme is utilized. For measuring performance, we use the committed instructions per

cycle (IPC). Only useful instructions are considered for counting the IPC. We do not count nop instructions. We define the performance improvement rate as the increased IPC over the IPC of the baseline model. The previously proposed value predictors are also presented in Figure 6. For each group of eight bars, the bars from left to right indicate the performance improvement rates for the last-value, the stride, the 2-level, the hybrid, the 2-mode last-value, the 2-mode stride, the 2-mode 2-level, and the 2-mode hybrid predictors, respectively. It can be easily found that the rates for 2-mode predictors are equivalent with the ones for the original predictors except for `130.li`. This confirms that the 2-mode scheme reduces hardware budget considerably with maintaining the effect of data speculation.

Table 5 presents the frequency of invalidating histories when the 2-mode predictors are evaluated using the timing simulator. Different from Table 4, the timing simulator can count different frequency for every prediction scheme. Table 5 shows that the frequencies in `130.li` and `147.vortex` is similar. Thus, the frequency is not the reason for the decline of the performance improvement in the case of `130.li`. We are currently investigating why `130.li` results in significant difference from the other programs.

Table 5: %Frequency of invalidations (timing simulation)

program	%frequency			
	last	stride	2-level	hybrid
099.go	0.76	0.75	0.81	0.81
124.m88ksim	1.01	1.01	1.03	1.04
126.gcc	1.93	1.93	2.03	2.04
129.compress	1.58	1.58	1.51	1.60
130.li	3.07	3.06	3.11	3.14
132.ijpeg	1.40	1.46	1.42	1.57
134.perl	1.96	1.94	2.30	2.32
147.vortex	2.94	2.97	2.97	3.00

6 Conclusion

In this paper, we investigate reducing hardware budget of data value predictors. By exploiting narrow data width, the data array size of the value history tables can be 1/4–1/2 of original one. If only small bitwidth predictor is used, its predictability is seriously degraded. Solving this problem, we propose the 2-mode value prediction scheme. The 2-mode scheme combines the wide predictor for large values and the narrow predictor for small values. Combining the two predictors prevents the 2-mode scheme from enlarging VHTs with maintaining the predictability. From timing simulations, it is found that performance improvement rates of the previously proposed value predictors and the predictors based on the 2-mode scheme are equivalent.

In this paper, we apply the 2-mode scheme only for last-value, stride, 2-level, and hybrid predictors. Future study includes evaluating the 2-mode scheme on other predictors such as context-based [14] and global history-based [10] predictors. Moreover, tag array of the value history table has to be considered for reducing hardware budget. Recently, we have applied Fagin's scheme [5] on value predictors [13]. Future study investigates to combine Fagin's scheme with the 2-mode prediction scheme. On the other hand, the bitwidth restriction is efficient for multimedia applications. Thus, we will evaluate the 2-mode scheme using multimedia applications such as MediaBench suite [7]. Furthermore, we are currently extending the bitwidth restriction scheme for applying on SIMD instructions such as MMX [11] and for predicting vector values.

This paper is one result of our ongoing research in high performance microprocessor *COSMOS* at Kyushu Institute of Technology. More information is available at `http://www.mickey.ai.kyutech.ac.jp/~tsato/`.

Acknowledgments

The authors thank the anonymous reviewers whose comments and suggestions helped to improve the quality of this paper.

References

[1] Brooks,D., Martonosi,M.: Dynamically exploiting narrow width operands to improve processor power and performance, *5th International Symposium on High Performance Computer Architecture* (1999).

[2] Burger,D., Austin,T.M.: The SimpleScalar tool set, version 2.0, *ACM SIGARCH Computer Architecture News*, vol.25, no.3 (1997).

[3] Calder,B., Reinman,G., Tullsen,D.M.: Selective value prediction, *26th International Symposium on Computer Architecture* (1999).

[4] Driesen,K., Holzle,U.: The cascaded predictor: economic and adaptive branch target prediction, *31st International Symposium on Microarchitecture* (1998).

[5] Fagin,B.: Partial resolution in branch target buffers, *IEEE Transactions on Computers*, vol.46, no.10 (1997).

[6] Gabbay,F.: Speculative execution based on value prediction, *Technical Report #1080*, Department of Electrical Engineering, Technion (1996).

[7] Lee,C., Potkonjak,M., Mangione-Smith,W.H.: MediaBench: a tool for evaluating and synthesizing multimedia and communications systems, *30th International Symposium on Microarchitecture* (1997).

[8] Lipasti,M.H., Wilkerson,C.B., Shen,J.P.: Value locality and load value prediction, *International Conference on Architectural Support for Programming Languages and Operation Systems VII* (1996).

[9] Morancho,E., Llaberia,J.M., Olive,A.: Split last-address predictor, *International Conference on Parallel Architectures and Compilation Techniques* (1998).

[10] Nakra,T., Gupta,R., Sofa,M.L.: Global context-based value prediction, *5th International Conference on High Performance Computer Architecture* (1999).

[11] Peleg,A., Wilkie,S., Weiser,U.: Intel MMX for multimedia PCs, *Communications of the ACM*, vol.40, no.1 (1997).

[12] Rychlik,B., Faistl,J.W., Krug,B.P., Kurland, A.Y., Sung,J.J., Velev,M.N., Shen, J.P.: Efficient and accurate value prediction using dynamic classification, *Technical Report CMuART-98-01*, Department of Electrical Computer Engineering, Carnegie Mellon University (1998).

[13] Sato,T., Arita,I.: Reducing hardware budget of data value predictors using partial resolution, *Technical Report of IEICE*, CPSY-2000 (2000) (in Japanese).

[14] Sazeides,Y., Smith,J.E.: Implementations of context based value predictors, *Technical Report TR-ECE-97-8*, Department of Electrical Computer Engineering, University of Wisconsin-Madison (1997).

[15] Tullsen,D.M., Seng,J.S.: Strageless value prediction using prior register values, *26th International Symposium on Computer Architecture* (1999).

[16] Wang,H., Sun,T., Yang,Q.: CAT – caching address tags: a technique for reducing area cost of on-chip caches, *22nd International Symposium on Computer Architecture* (1995).

[17] Wang,K., Franklin,M.: Highly accurate data value prediction using hybrid predictors, *30th International Symposium on Microarchitecture* (1997).