Analysing value substitution and confidence estimation for value prediction

Luis Piñuel, Rafael A. Moreno and Francisco Tirado
Departamento de Arquitectura de Computadores y Automática
Universidad Complutense de Madrid.
28040 – Madrid. SPAIN.
Fax: +34 91 394 46 87
lpinuel@dacya.ucm.es, rmoreno@dacya.ucm.es, ptirado@dacya.ucm.es

Abstract

Value Prediction is one of the newest techniques used to break down ILP limits. Despite being under continuous study during the last few years, a few aspects related to this emerging technique remain unanalysed in depth. Exhaustively investigated in the context of control speculation, confidence estimation has usually played a secondary role on value prediction and speculation. Closely linked to confidence estimation, value substitution also represents a relegated subject of research.

This paper is focussed on analysing, in an isolated way, the respective impact on predictor performance of both confidence estimation and value substitution mechanisms. By using detailed pipeline-level simulations, we prove that improvements in these mechanisms are as important as reducing the predictor aliasing or even improving the prediction model.

1. Introduction

As program dependencies become a bottleneck in superscalar processors, the speculation techniques are gaining importance. Among these techniques, value prediction has been one of the most investigated during the last few years. However, several aspects related to this emerging technique remain unanalysed in depth.

The value prediction technique, like branch prediction, allows temporal violation of the program constraints without affecting its semantics. Based on the previous history of program execution, the hardware predicts at run-time the outcome of an instruction, which is used by the consumer instructions when the real data is not yet ready. It is obvious that the more history the predictor is able to capture the greater the number of correct predictions it can produce. Nevertheless, the use of huge predictor tables is not realistic for the coming generations of processors, especially if we take into account that few table accesses per cycle are needed. It is therefore understandable that one of the early goals of value prediction research was to reduce the predictor table size [Gabb97]. Usually, less attention has been given to analysing other aspects related to value prediction, like for example confidence estimation. Through the presence of high misprediction penalties, confidence estimation has been extensively studied for branch prediction [Grun98], [Jaco96]. However only very recent works on value prediction [Calder99], [Beker99], [Burt99], lend some weight to its analysis. Value misprediction penalties are not negligible, especially in the case of squash recovery. Therefore, a deeper analysis of confidence estimation is required to improve the performance gain of the value speculation (or even to avoid performance degradation in some cases).

This paper examines the impact of confidence estimation on predictor behaviour and on the overall processor performance. So far, the replacement mechanism of value predictors has been

usually mixed with the confidence one: only low confidence values are replaced. This fact complicates the study of confidence estimation, hence we have isolated both mechanisms to better analyse their respective effect on value prediction. First, functional simulations of SPECint95 are used to evaluate, independently of the pipeline structure, several confidence estimation and replacement mechanisms. Lastly, our pipeline-level simulations show their respective performance impact and reveal the potential performance gain of improving confidence estimation and value substitution mechanisms respectively.

The rest of the paper is organised as follows. Section 2 summarises the previous work on data value prediction. Section 3 describes our machine model. Section 4 introduces the experimental framework. Section 5 examines the impact of confidence estimation and replacement mechanism on the predictor's behaviour. Section 6 shows their impact on processor performance and analyses their potential performance gain. Finally, section 7 presents the conclusions and future work.

2. Previous Work

Related work has been conducted in two different directions: providing new refined prediction models and improving implementation of existing prediction models.

2.1 Value Prediction Models

Early work on value prediction was centred on demonstrating the predictability of instructions [Lipa96] whereas later work was focussed on introducing new prediction models that increase the number of correct predictions of the previous ones. Most of the predictors proposed in the literature can be classified into one of the following types. *Last-value predictors* (LVP), which make a prediction based on the last outcome of the same static instruction, and can correctly predict constant sequences of data. [Lipa96], [Gabb97], [Wang97]. *Stride predictors*

(SP), which make a prediction based on the last outcome plus a constant stride, and can correctly predict arithmetic sequences of data (even constant sequences, whose stride is 0), [Gabb97], [Nakr99], [Wang97]. *Context based predictors* (CBP), which learn the values that follow a particular context and make a prediction based on the last values generated by the same instruction. They can correctly predict repetitive sequences of data [Saze97a], [Saze97b], [Wang97], [Nakr99]. *Hybrid predictors* (HP), which combine some of the previous predictors and include a selection mechanism, either hardware [Wang97], [Rych98], [Pinu99a], or software [Gabb97], [Gabb98]. Looking at the different proposals we can draw the following conclusion: the more complex the prediction model, the higher the number of correct predictions but also the more it is difficult to implement in a realistic way.

2.2 Implementation of value predictors

Implementation of value predictors entails dealing with three main problems: predictor table size (cost), confidence estimation and value replacement.

2.2.1 Reducing predictor table size

As was mentioned earlier, reducing the predictor table size is one of the main causes for concern in value prediction research. The use of limited predictor tables produces a decrease in prediction performance due to a significant instruction *aliasing*. One of the key issues for reducing table size without sacrificing performance is to attenuate the *aliasing* through the use of *tag* or through *associative tables*. Several different implementations of value predictors make use of tagged tables [Wang97], [Gabb97], nevertheless the results are not very promising. The tag field involves a substantial increase in the table entry size and the behaviour amelioration is not really outstanding, see [Pinu99b]. Some other implementation proposals employ associative tables [Rych98], [Calder99], which bring down the number of table entries at the expense of

hardware complexity. It should be noted that several predictor accesses per cycle are required. Consequently, the use of 4-way associative tables seems inappropriate. The most attractive way to face the problem of the predictor aliasing is *instruction filtering*. This method originally proposed by Calder *et al.* in [Calder99], tries to reduce the pressure on the prediction tables by predicting only those instructions that impact on performance.

2.2.2 Confidence Estimation

Confidence estimation is a technique for determining the quality of a particular prediction. Usually employed in the context of branch prediction [Jaco96], [Grun98], it is essential for the value prediction due to both the relatively low prediction rate of the predictors (compared to branch predictors) and the time penalty produced by miss-speculations. The confidence estimators (CE) discussed in the literature of value prediction can be classified into one of the following types. *Saturating Counter Estimator* (SCE): [Lipa96], [Saze97b], this method assigns an individual saturated counter for each entry in the predictor table. The counter is incremented/decremented when correct/incorrect predictions occur. Depending on the value of the counter and on a fixed *confidence threshold*, the predictor can decide whether the prediction is accepted or rejected. *Miss-distance Counter Estimator* (MCE): it is a kind of resetting counter, which is reset when incorrect predictions occur. A prediction is considered confident if the miss-distance exceeds a certain threshold. The MCE is based on the previous work of Jacobsen *et al.* [Jacob96] about confidence estimation for branch prediction, and it is also used by Bekerman *et al.* [Beker99] in the context of load-address prediction. *History Counter Estimator* (HCE): [Calder99], [Burt99], this method assigns an N-bit history register to each entry in the prediction table, keeping track of the accuracy of the last N value predictions. In the proposal of Calder *et al.*, the history register is used to index a global table of SCEs, which are responsible for

providing the confidence value. In contrast with this, Burtscher *et al.* employ the history register to index a global table, pre-programmed according to profiling information, where each entry stores a single bit that determines if the history pattern should trigger a prediction. *Static Estimator* (SE): [Gabb97] this method is based on compiler profiling and instruction annotation. For each static instruction, the compiler collects the percentage of correct predictions according to previous runs of the program. Then it uses a fixed *threshold* to determine the confident predictions and places directives in the *opcode* of the instructions, providing information about their *predictability*.

2.2.3 Value Replacement

There are two main causes of replacement in the predictor table: *instruction aliasing* and *value sequence changes*. To minimise the unnecessary replacements in the table and better capture the instruction's value sequence, the use of confidence information for replacement is common practice. In this way, only low (or very low) confidence values are replaced. Although some authors make use of extra counters to provide hysteresis to the replacement mechanism, e.g. [Calder99], usually the same saturating counter employed for confidence is also used for replacement, but with a replacement threshold lower than the confidence one, e.g. [Lipa97].

3. Architecture Model

Most of the previous work, e.g. [Calder99], has analysed the impact of a few confidence mechanisms on a particular value predictor. Only Burtscher *et al.* [Burt99] have presented a comparative analysis for different predictors, but restricted to load value prediction and using different confidence estimators only for the last value predictor. The aim of the present work is to exhaustively analyse value predictors, confidence estimators and value substitution policies

and then identify their respective contribution to value prediction. This section presents the different alternatives under study as well as the processor model employed in our study.

3.1 Data Predictors

The different value predictors under analysis are particular implementations of last value, stride and context-based[1].

LVP: It is implemented by means of a direct mapped table. The table is indexed using the least significant bits of the instruction PC. Each table entry stores the following information: last-value produced by the instruction and confidence estimation (e.g. saturating counter).

SP: Like the previous predictor, it is implemented using a direct mapped table. In this case, in addition to last value and confidence bits, each entry stores the stride between the two last outputs of the instruction.

CBP: It is derived from the work of Sazeides *et al.* [Saze97b] and it uses a 2-level table. The first level table, called the Value History Table (VHT) is direct mapped and it is indexed using the least significant bits of the instruction PC. This table stores an order-3 context, i.e. 3 last outcomes produced by the instruction. The second level table, called the Value Prediction Table (VPT) is indexed by a hash function, which uses context information from the VHT. The VPT is responsible for storing the value prediction and the confidence estimation for each context. The hash function shift-xor-fold, [Pinu99b], differs from the original one proposed by Sazeides, and slightly reduces the aliasing in the VPT.

---

[1] The hybrid predictor is not consider, because it relies on the other prediction models for both, producing the value and assigning confidence.

3.2 Confidence Estimation

In addition to the SCE, we have also employed a MCE, and a particular implementation of a HCE. Our HCE differs slightly from the HCE proposed by Calder *et al.* [Calder99]. Instead of sharing a global table of SCEs between instructions, the N-bit history counter is used for selecting one of the SCEs included in the same entry table, i.e. SCEs are local for the prediction. Using the same length of history pattern the cost of our HCE is higher than Calder's, although in general a local SCE table needs a significantly smaller history register, and thus the costs of both approaches are similar. Note that in the case of the CBP, the history register (HR) is stored in the VHT whereas the SCEs are stored in the VPT. To analyse the potential of confidence estimation we have also simulated a perfect confidence estimator (PCE): predictions are done only when they are correct.

3.3 Replacement Policy

Two replacement mechanism are analysed, a *Saturating Counter Replacement* (SCR), and an *Oracle Replacement* (OR). The behaviour of the SCR is similar to that of the SCE; a value is not replaced if the counter value is above the threshold. The OR replaces a value only if the next prediction will be correct, otherwise it leaves the entry untouched.

3.4 Processor Pipeline

We must note at this point that only single precision instructions that write into general-purpose registers are considered as predictable. Therefore, control instructions and double precision instructions are not related to value speculation.

A detailed description of all the hardware mechanisms involved in the value speculation technique is beyond the scope of the present work. We just want to briefly introduce the

architecture employed in the timing simulations, which is explained in more detail in the Technical Report [More98].

Our baseline architecture, shown in figure 1, is derived from the architecture used by the SimpleScalar Out-of-Order simulator [Burg97]. This architecture is based on the Register Update Unit (RUU) [Sohi90], which is a scheme that unifies the instruction window, the rename logic, and the reorder buffer under the same structure.

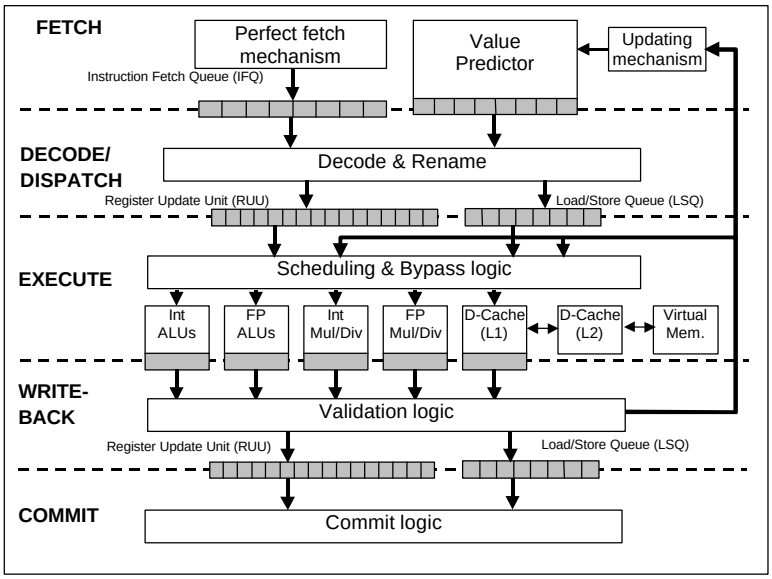


Figure 1: Architecture Block Diagram.

3.4.1 Predictor Lookup

The value predictor is accessed in parallel with the instruction fetch using the addresses of the instructions fetched in each cycle, and it provides the predicted output values (if available) of these instructions. Note that a *perfect fetch mechanism* is assumed, in order to feed the processor with a constant flow of instructions and avoid problems of instruction cache misses, non-contiguous instruction alignment, etc. [Rote96]. Recently, several mechanisms have been proposed that provide near perfect fetch [Yeh93], [Rote97].

3.4.2 Scheduling policy

The *scheduling policy* firstly issues the instructions with actual operands, and thus instructions with predicted or speculative operands are issued later. Within each group, an *oldest-instruction-first* policy is used. Using this policy, speculative instructions are not issued while there are enough non-speculative instructions ready to execute, even if these non-speculative instructions are newer than the speculative ones.

3.4.3 Validation and misprediction recovery

The process of validation/invalidation of speculative instructions is performed during write-back. This process is performed in parallel, i.e. all the instructions within a dependence chain can be validated/invalidated in a single cycle. This operation is achieved by transmitting a validation/invalidation signal in a chained way, namely, the signal has to travel through all the speculatively executed instructions belonging to the same dependence chain. The instructions whose operands have been validated can commit in the next stage. On the other hand, those instructions whose operands have been invalidated must be re-executed. In view of the fact that it is not possible to check the validity and re-schedule the invalidated instructions in the same cycle, it is obvious that these instructions cannot be re-executed in the next cycle. Consequently, they are delayed one cycle in relation to normal execution.

4. Experimental Framework

This section describes the framework employed in our research to obtain the experimental results. The simulators used in this work are derived from the SimpleScalar 3.0 tool set [Burg97], a suite of functional and timing simulation tools. The instruction set architecture employed is the PISA instruction set, which is based on the MIPS ISA.

4.1 Benchmarks

To perform our experimental study, we have collected results for the integer SPEC95 benchmarks. The programs were compiled with the *gcc* compiler included in the tool set, using the optimisation level -O3. Table 3 shows the input data set for each benchmark and the total number of instructions. For each benchmark, we have performed two kinds of simulations: functional and pipeline-level. Due to time constraints, we have only simulated 100 million instructions, but a preliminary analysis has shown that the behaviour of the programs is sufficiently representative when using these limited simulations.

Benchmark	*Input Set*	# Instructions
compress95	30000 e 2231	95 M
cc1	SPEC95 Ref Input (gcc.i)	203 M
go	9 9	132 M
ijpeg	SPEC95 Test Input (specmun.ppm)	553 M
m88ksim	SPEC95 Train Input	120 M
perl	SPEC95 Train Input	40 M
li	SPEC95 Train Input (scrabbl.in)	183 M
vortex	SPEC95 Train Input	2520 M

Table 1: Benchmark statistics.

4.2 Baseline Architecture

Table 2 summarises the main architectural parameters used in our simulations. As the goal of this work is to show the impact of the confidence mechanism on processor performance, we have chosen a perfect branch predictor to avoid the interaction between both prediction mechanisms. However, we have observed that, although realistic branch prediction reduces the achievable IPC, the relative differences among confidence mechanisms remain almost constant.

Fetch, Decode, Issue, Commit	8	Memory Ports	4
Register Update Unit	64	L1 I Cache Size	64KB
Load Store Queue	16	L1 D Cache Size	64KB
Integer ALU	8	L1 Latency	1
Integer Multiplier	4	L2 Cache Size	4MB
Floating Point ALU	4	L2 Latency	6
Floating Point Multiplier	2	Memory Latency	18
Branch predictor	perfect		

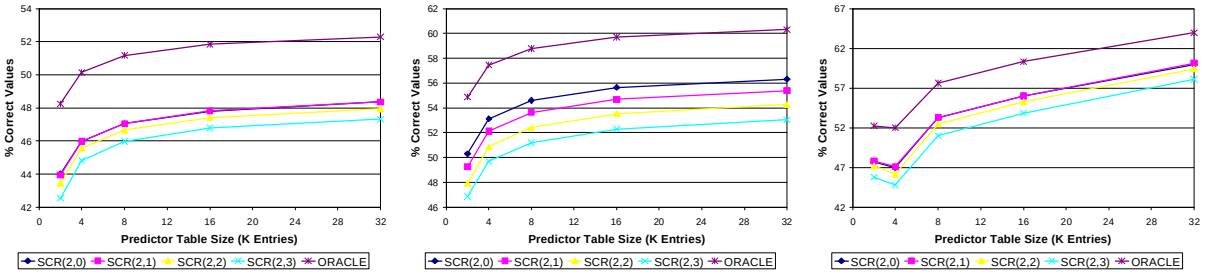
Table 2: Architectural Parameters.

5. Impact on Predictor Behaviour

In this section, we first study the contribution to *predictor efficacy* of several predictor elements, in particular the replacement policy. Then, we evaluate how these elements, along with the confidence estimator, influence *predictor performance*.

5.1 Predictor Efficacy

We understand *predictor efficacy* as the capacity to produce correct values, independently of whether they are confident or not. The three main predictor components that have a significant influence on efficacy are the following: predictor model, replacement mechanism and table size. Note that confidence estimation does not influence efficacy. Figure 2 shows the predictor efficacy for the different predictors under study using various SCR configurations, an oracle replacement and several table sizes. The notation *SCR(n,t)* represents an *n*-bit SCR with a replacement threshold *t*, $0 \leq t \leq 2^{n-1}$. In order to reduce the number of possible combinations, we have used a fixed size (2048) for the context-based VHT.



a) Last Value b) Stride c) Context-Based

Figure 2: Predictor Efficacy.

One of the most significant remarks that we should make about the previous figure is that for different replacement policies, variations in efficacy (distance between curves) are not influenced by the table size. This fact is particularly relevant because, one could think *a priori* that the smaller the table (the more *aliasing*) the higher the differences between replacement mechanisms, although this is not the case. Furthermore, we can also observe that for last value and stride predictors when using 32K tables, where the *aliasing* is negligible, differences remain significant. In view of these facts, we can conclude that differences in efficacy between replacement mechanisms depend mainly on the ability to identify changes in the instruction value sequence. For example, if we consider the value sequence “aaaXaaaXaaa....”, where “X” can be any value different to “a”, it easy to understand that a LVP with a SCR(2,0) predicts 75% of correct values, whereas a LVP with a SCR(2,3) only obtains an efficacy of 50%.

We should also remark that, as we expected, a notable improvement in efficacy could be obtained by increasing the table size of the predictors, i.e. significantly augmenting the HW. On the contrary, the achievable improvement by adding hysterisis to the replacement mechanism (decreasing the threshold of the SCRs), although smaller, it is obtained without any HW increase. To understand more clearly the contribution of replacement policy and predictor size to predictor efficacy, the relative improvements (in average) are summarised in table 3.

	SCR(2,0) *vs* SCR(2,3)	OR *vs* SCR(2,0)	32K *vs* 2K
LVP	2.5 %	8.8 %	9.9 %
SP	6.7 %	7.9 %	13.1 %
CBP	4.1 %	8.6 %	20.7 %

Table 3: % Improvement.

Looking at the table above, we can clearly see that SP and CBP are more sensitive to variations in the SCR configuration and predictor table size than LVP. These results also confirm that, although the SCRs are performing well compared to the OR, considerable improvements are still possible (≈8.4%) by using more efficacious replacement policies. In particular, for the LVP, improving the replacement mechanism could be as beneficial as increasing the table size.

Regarding the efficacy results, we also want to point out the anomalous behaviour of the CBP for VPTs below 4K. The reason in fact lies in the aliasing. When aliasing is very high (below 4K table size) the CBP is more sensitive to variations in the behaviour of the hash function, which is slightly different depending on the table size.

Despite not being totally reliable, we also show a comparative analysis of the predictor models based on the efficacy measure (only for illustrative purposes). Figure 3 shows the predictor efficacy of LVP, SP and CBP using the oracle replacement. It is evident that the most efficacious predictor is the SP below 16K table size, and the CBP above this size.

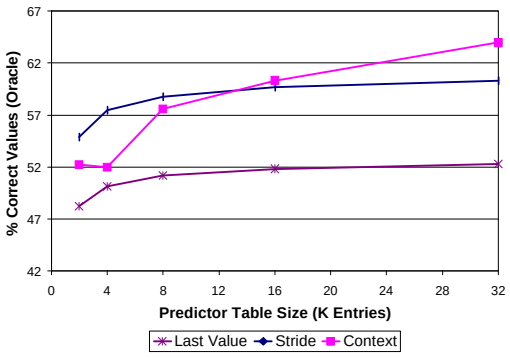


Figure 3: Oracle Replacement Efficacy.

Although predictor efficacy is commonly used to compare different value prediction models, this approach could lead to erroneous conclusions, as we show in section 6. In fact, the most efficacious predictor does not necessarily produce the highest impact on processor performance. Therefore, the efficacy measure should be essentially employed to compare different configurations of the same predictor model (replacement mechanism, table size, etc).

Finally, we can conclude that the best configuration among those studied is the SCR(2,0), which obtains around 92 % of the OR efficacy for all the prediction models. Consequently, in the remaining results, in order to reduce the number of combinations, the SCR(2,0) is used as a default replacement policy unless indicated otherwise.

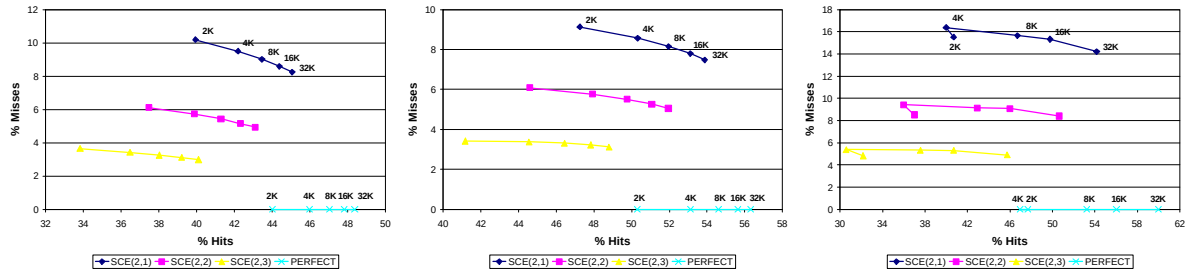
5.2 Predictor Performance

Predictor performance could be considered a trade-off between the number of predictions and their accuracy. Therefore, predictor performance depends not only on its capacity to produce correct values, but also on its capacity to identify them as such. The main goal of this section is to evaluate the contribution to performance of the different predictor elements. Hence, in addition to the predictor components seen in the previous section, we must consider in particular the confidence estimation mechanism. As the behaviour of this mechanism is also influenced by the size of the table and the predictor model employed, it is not suitable to analyse its contribution in an isolated way. Instead, we first analyse the predictor performance impact of each CE type using several predictor models, table sizes and CE configurations. Then we compare CE types for a given predictor model and table size.

5.2.1 Saturating Counter Estimator

Figure 4 shows the percentage of misses as a function of the percentage of hits, using different SCE configurations, table sizes and predictor models. Note that, in these figures, the

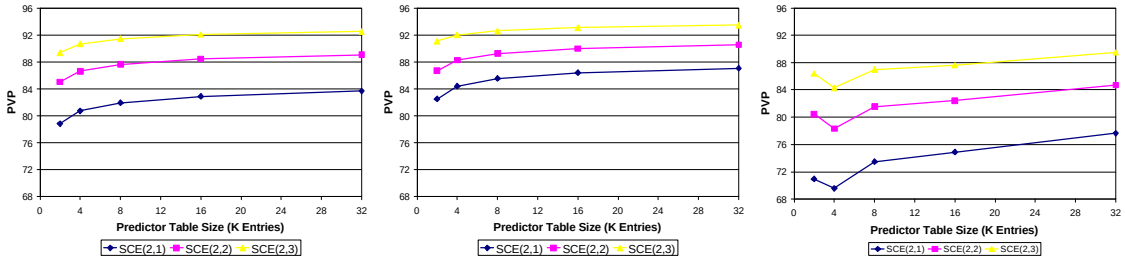
percentage of no-predictions is only implicitly represented (%no-predictions = 100 - %hits - %misses). As for the SCR, the notation *SCE(n,t)* represents an *n*-bit SCE with a confidence threshold *t*. When representing misses as a function of hits, it is evident that the best predictor configuration is the one that produces a result closest to the bottom-right corner of the figure. However, no matter how good the confidence estimator is, it is impossible to reach this corner, because the predictor performance is limited by the predictor efficacy. In order to highlight this fact, we also show in figure 4 the results obtained by a perfect confidence estimator.



a) Last Value b) Stride c) Context-Based

Figure 4: Percentage Misses vs. Percentage Hits for different SCE configurations.

To evaluate predictor performance, we must not only weigh up the number of hits and misses, but also the relationship between them, i.e. prediction accuracy. As a measure of prediction accuracy, we propose using the PVP metric (Predictive Value of a Positive test) introduced by Grundwald *et al.* in the context of control speculation [Grun98]. The PVP metric is defined as the ratio of correct predictions to total predictions. Figure 5 shows the PVP values obtained for different SCE configurations when using the LVP, SP and CBP.



a) Last Value b) Stride c) Context-Based

Figure 5: PVP value for different SCE configurations.

As we expected, the results show that aggressive saturating counter estimators (lower thresholds) produce a higher number of predictions at the expense of a loss in accuracy, whereas conservative ones (higher thresholds) improve accuracy at the expense of a loss in prediction. Furthermore, differences between these SCE configurations are higher when the table size is decreased. It is obvious that the higher the aliasing, the higher the number of mispredictions caused by aggressive configurations, and the lower the number of hits obtained by conservative ones. On average, we can observe that the reduction in misses produced by raising the threshold of SCEs is more significant than the increase in hits produced by lowering it. In fact, this is the reason why the accuracy is higher for conservative SCEs. The previous remark is more evident in tables 4 and 5, which summarise the relative improvements on hits, misses and PVP.

	HITS			**MISSES**	
	SCE(2,1) *vs* SCE(2,3)	PERFECT *vs* SCE(2,1)	32K *vs* 2K	SCE(2,3) *vs* SCE(2,1)	32K *vs* 2K
LVP	14.73 %	8.51 %	13.36 %	-63.65 %	-18.81 %
SP	12.22 %	5.28 %	14.04 %	-59.81 %	-14.57 %
CBP	24.47 %	14.38 %	27.17 %	-66.56 %	-2.78 %

Table 4: Improvement on hits and misses using SCE.

	PVP		
	SCE(2,3) *vs* SCE(2,1)	PERFECT *vs* SCE(2,3)	32K *vs* 2K
LVP	10.52 %	9.65 %	4.79 %
SP	17.88 %	8.17 %	4.19 %
CBP	15.73 %	15.03 %	6.10 %

Table 5: Improvement on accuracy using SCE.

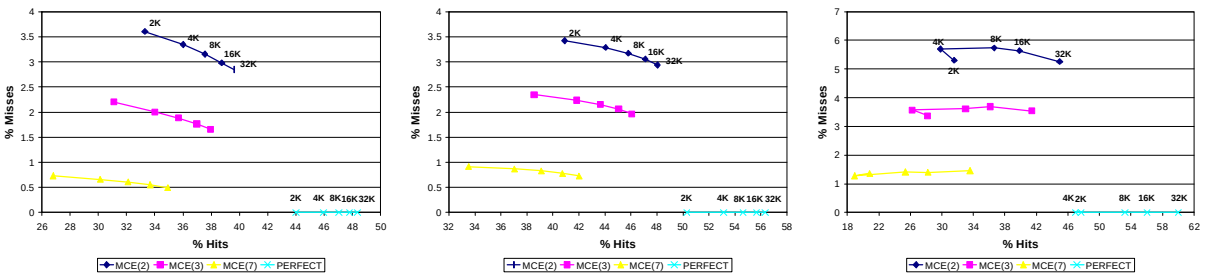
Increasing the size of the table betters the performance of the predictor (moves the curves to the bottom-right corner of figure 4). Both prediction rate and accuracy are improved. However, the prediction rate increase is considerably higher (especially for the CBP) because it is a consequence of the efficacy increase, whereas the accuracy improvement is due to the reduction in the SCE aliasing.

Another interesting observation is that differences are greater between SCE configurations than between table size configurations, especially in terms of misses (on average, 63% vs. 12%) and accuracy (on average 14.7% vs. 5%). This fact indicates that, for a given model, the predictor performance is mainly due to SCE configuration, and table size plays only a secondary role.

Finally, comparing the PVP results for different predictor models, we can conclude that the SCEs perform better for the LVP and SP, than for the CBP. The relatively low prediction accuracy of the CBP suggests that more advanced confidence estimators are needed for this particular prediction model; 15% improvement is still possible (on average). Regarding the results of the CBP, we must again remark on the anomalous behaviour for a table below 4K.

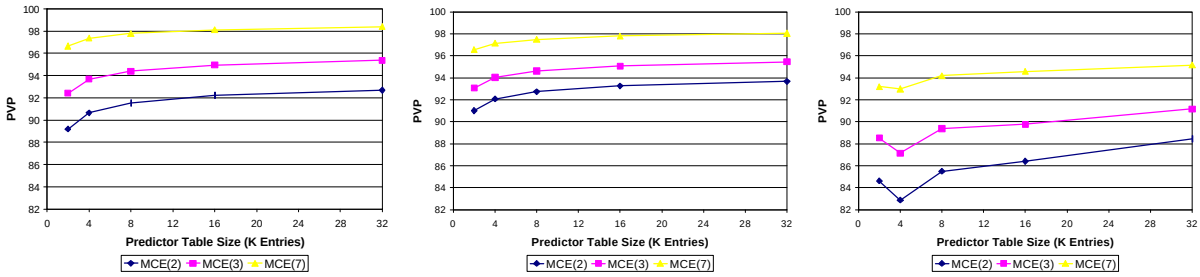
5.2.2 Miss-distance Counter Estimator

Figures 6 and 7 show the results for different MCE configurations when using LVP, SP and CBP. The notation *MCE(t)* represents an MCE with confidence threshold *t*.



a) Last Value b) Stride c) Context-Based

Figure 6: Percentage Misses vs. Percentage Hits for different MCE configurations.



a) Last Value b) Stride c) Context-Based

Figure 7: PVP value for different MCE configurations.

Like for SCEs, aggressive MCE configurations produce a higher number of predictions at the expense of a loss in accuracy and *vice versa*. In general, remarks made in the previous section on the effect of CE configuration and table size on predictor performance are also valid when using MCEs. Nevertheless, it must be noted that the prediction rate is more sensitive to both MCE configuration and table size, especially for the CBP where their relative improvement on hits is about 50% (see table 6). On the other hand, variations in accuracy are relatively low, as we can appreciate in table 7.

The results demonstrate that, although the overall number of predictions is substantially reduced with respect to the perfect confidence estimator (and also to the SCE), the prediction accuracy is very high when using MCEs, independently of the threshold (up to 98%). Therefore, the MCE represents a highly conservative type of confidence estimator. Its caution is responsible for the behaviour noted previously regarding the sensitivity to CE configuration and table size.

	HITS			**MISSES**	
	MCE(2) *vs* MCE(7)	PERFECT *vs* MCE(2)	32K *vs* 2K	MCE(7) *vs* MCE(2)	32K *vs* 2K
LVP	17.83 %	26.11 %	23.74 %	- 80.95%	- 25.98 %
SP	17.61 %	19.63 %	20.75 %	- 74.09 %	- 16.95 %
CBP	45.64 %	45.59 %	50.17 %	- 74.92 %	3.62 %

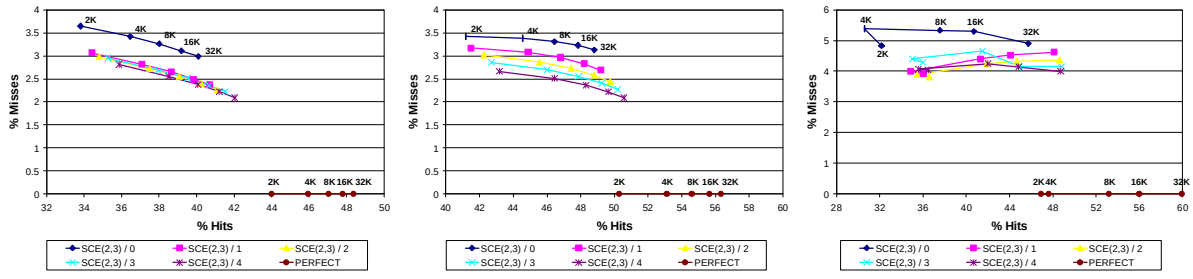
Table 6: Improvement on hits and misses using MCE.

	PVP		
	MCE(7) *vs* MCE(2)	PERFECT *vs* SCE(2,3)	32K *vs* 2K
LVP	7.03 %	2.33 %	2.73 %
SP	5.25 %	2.58 %	2.18 %
CBP	9.90 %	5.97 %	2.80 %

Table 7: Improvement on accuracy using MCE.

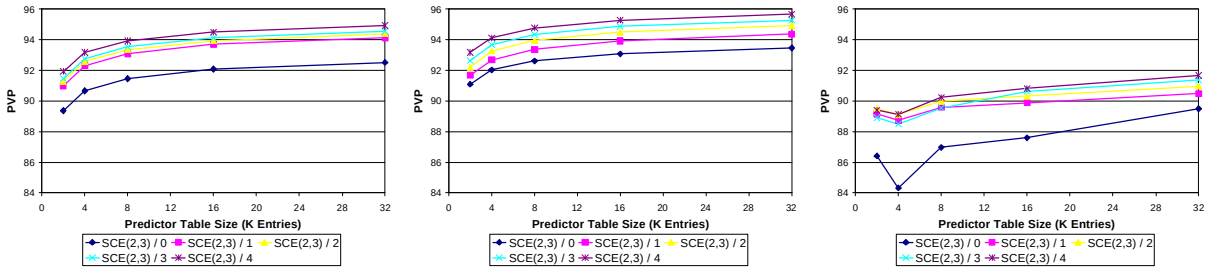
5.2.3 History Counter Estimator

In our HCE, SCEs (or MCEs) are ultimately responsible for classifying predicted values, hence it could be considered as a special implementation of SCEs (or MCEs) that produces an enhanced behaviour. Figures 8 and 9 respectively show the percentage of misses as a function of the percentage of hits, and the PVP values obtained for different HCE configurations when using LVP, SP and CBP. The notation *SCE(n,t) – m* represents an *m*-bits history-pattern HCE using *SCE(n,t)*, whereas *MCE(t) – m* represents an *m*-bits HCE using *MCE(t)*. Notice that the *SCE(n,t) – 0* is equivalent to *SCE(n,t)*. In the following figures we want to illustrate how the use of local history counters in the CE affects predictor performance. Therefore, in order to simplify the figures, we only show results using SCE(2,3).



a) Last Value b) Stride c) Context-Based

Figure 8: Percentage Misses vs. Percentage Hits for different HCE configurations.



a) Last Value b) Stride c) Context-Based

Figure 9: PVP value for different HCE configurations.

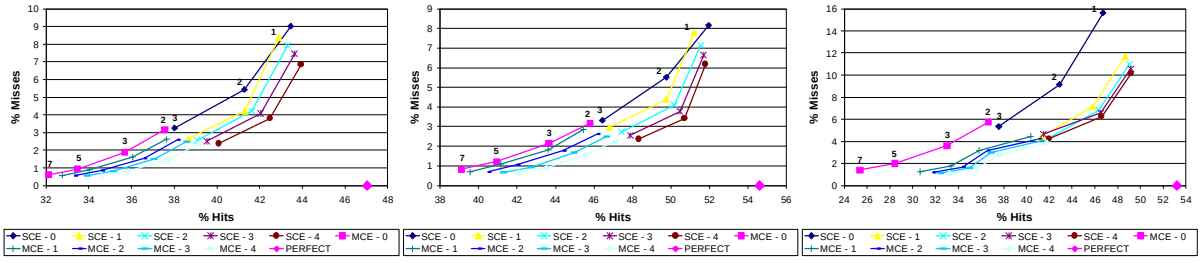
The impact on predictor performance of the HCEs is slightly different for the different predictor models under study. For the last value predictor, we can observe that using only a 1bit history register considerably reduces misses and thus substantially increases accuracy. On the other hand, only a relatively minor improvement on hits and accuracy are obtained by increasing the length of the HR above 1bit. For the stride predictor, in contrast, we can observe that the larger the HR the higher the predictor performance. It is important to note that for both LVP and SP, differences between HR lengths (distances between curves) are not influenced by table size. Therefore, we can conclude that, for these predictor models, the improvement produced by using HR is really due to a improvement in the confidence estimation and not to a reduction in the CE aliasing. For the context-based predictor, however, this assertion is not completely accurate. In fact, we can appreciate that, although predictor performance is always improved by using HCEs, for 1bit and 2bit HR the relative improvement is lower when increasing the table size, i.e. it is mainly due to the CE aliasing.

It is clear from the previous figures that the use of history counters improves the SCE behaviour in terms of hits, misses and prediction accuracy. However, it also implies a cost increase (e.g. 50% for a SCE(n,t)–3). In general, among HCE configurations, the best performance to cost ratio is obtained when using 1bit or 2bit history-patterns. We must remark

that similar conclusions could be drawn when using MCEs or when using a different threshold for the SCE, as we show in the next section for fixed table size.

5.2.4 Comparative analysis of CE

Previous sections were focused on analysing the effect on predictor performance of table size and CE configuration. We have shown that CE configuration plays a major role in predictor performance, whereas table size, despite being interrelated with CE, has a minor influence. In order to compare the different CE types, we now fix the size of the predictor table. Figure 10 shows the percentage of misses as a function of the percentage of hits when using LVP, SP and CBP, and an 8K table. The notation *SCE – n* represents a n-bit HCE that uses a *SCE(2,t)*, where t=1,2,3. In the same way, the notation *MCE – n* represents a n-bit HCE that uses a *MCE(t)*, where t=2,3,5,7.



a) Last Value b) Stride c) Context-Based

Figure 10: Percentage Misses vs. Percentage Hits for different CE (8K).

Looking at this figure, it is evident that, as previously mentioned, MCEs are more conservative than SCEs. Furthermore, both curves exhibit continuity, i.e. they are not overlapped. Hence, we can hierarchically classify CEs in function of their caution, from the most conservative one, MCE(7), to the most aggressive one, SCE(2,1). If we consider predictor performance as a trade-off between prediction rate and accuracy, the best CE configurations are SCE(2,3) and MCE(2). Another valid conclusion is that the use of a history register always ameliorates the behaviour of the CE, independently of its configuration (especially for SCEs).

Consequently, using HCE is always beneficial for performance, but at the expense of a slight cost increase (8,8% for 1bit HR; 23.5% for 2 bits).

Determining which CE is the best for a given predictor model is not a trivial problem. The relationship between predictor performance and processor performance depends essentially on processor architecture, and is hard to identify. The more evident way of facing the problem consists in selecting the CE according to the misprediction penalty. Depending on this penalty it could be more beneficial to employ either a conservative CE (higher accuracy) or an aggressive one (higher predictions rate). This way, for high misprediction penalty architectures an MCE is *a priori* preferable, whereas for low penalty architectures an SCE represents a more attractive choice. Nevertheless, there are many more factors that influence the relationship between predictor performance and processor performance (e.g. scheduling policy, instruction window size, functional unit characteristics, etc.) and thus the previous approach could be sometimes inaccurate.

Finally, we also consider it interesting to compare the different predictor models for a fixed HR configuration (2bits). Figure 11 simultaneously shows the results for LVP, SP and CBP.

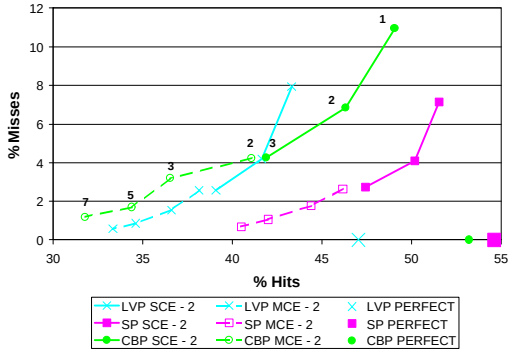


Figure 11: Percentage Misses vs. Percentage Hits for 2bits HCEs.

Remember that, for this kind of figure, the best predictor configuration is the one closest to the bottom-right corner. Therefore, with respect to predictor performance, it is obvious that the

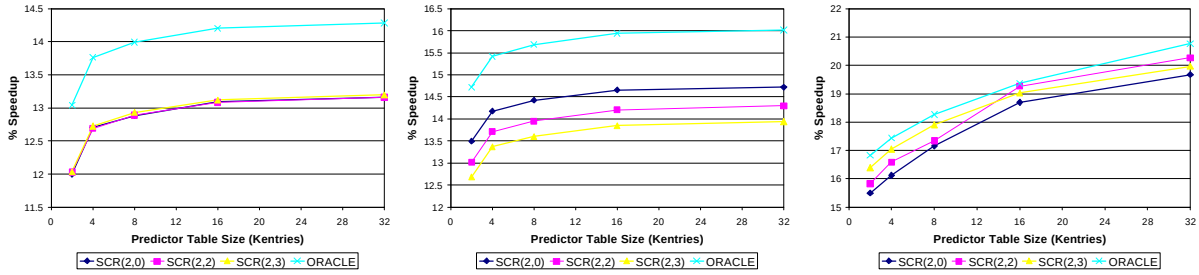
stride predictor is far superior to both LVP and CBP when using an 8K table. On the other hand, it must also be noted that the performance of the CBP is the worst only when using MCEs; this kind of CE seems too conservative for CBP. We should however remember that, like for the efficacy analysis, conclusions drawn from the comparison of predictor models must be carefully interpreted. In fact, the predictor with the highest predictor performance doesn't necessarily produce the highest impact on processor performance, as we show in next section.

6. Impact on Processor Performance

Predictor misses involve a re-execution of dependent instructions and hence entail a substantial time penalty. Consequently, miss-predictions could have a stronger effect on performance than correct predictions, as we have previously mentioned. Furthermore, not all the correct predictions contribute to the performance improvement (because they are not used, due to the dynamic scheduling and the forwarding) or contribute with different weighting according to the dependence chain, see [Calder99]. Therefore, in view of the fact that using functional simulations is not sufficient to evaluate the impact on the overall value speculation performance, we have also performed detailed pipeline-level simulations. In this section we try to illustrate how the effect on predictor efficacy and performance, produced by the different mechanisms studied in section 5, is translated to the overall processor performance.

6.1 Predictor efficacy

To evaluate the relationship between predictor efficacy and processor performance, it is essential to isolate them from the confidence estimation. Hence, we have employed a perfect confidence estimator. Figure 12 shows the performance obtained by the different replacement policies when using this ideal estimator.



a) Last Value b) Stride c) Context-Based

Figure 12: Speedup using PCE.

The results are substantially different depending on the predictor model. For the LVP, there isn't any appreciable variation in speedup among SCR configurations. On the contrary, SP and CBP are more sensitive to replacement mechanisms. In particular, for the SP we can remark on a direct relationship between predictor efficacy and processor performance. Decreasing the threshold of the SCRs improves both the efficacy and the speedup in the same proportion. Consequently, for this predictor model, using a more efficacious mechanism is as beneficial as increasing the size of the predictor table (decreasing the aliasing). On the other hand, for the CBP, the results apparently belie the previous conclusions about the influence of the SCR configuration on predictor behaviour. Although, on average, decreasing the threshold of the SCRs is beneficial for the CBP efficacy (see figure 2), for the benchmarks, *compress95*, *li* and *m88ksim*, it is detrimental. Nevertheless, these are precisely the benchmarks where the CBP obtains the highest speedups. Consequently, for this particular predictor, the relationship between efficacy and speedup is only individually verified for each benchmark.

Regarding the oracle replacement, we can remark that its relative improvement is as significant for efficacy as for processor performance, except for the CBP where the speedup obtained by the SCRs is surprisingly close to that of the oracle. Therefore, for the LVP and SP notable improvements in performance are still possible by using more efficacious replacement

policies. Finally, we must note that, as it was expected, table size plays an important role in both efficacy and speedup, especially for the CBP.

It is obvious that using a realistic CE, differences will be lower. Nevertheless, the previous assertions remain valid. If we now compare the predictor models, however, the relationship between predictor efficacy and processor performance is only partially verified. Figure 13 shows the performance for the different predictors under study when using oracle replacement and perfect confidence estimation.

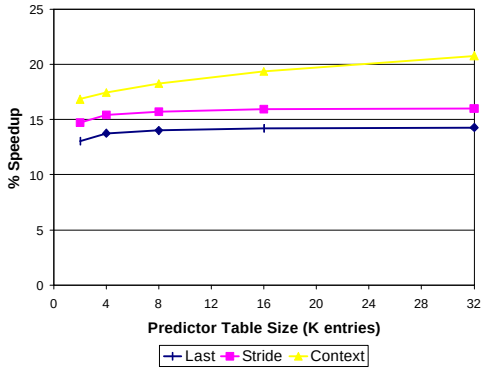
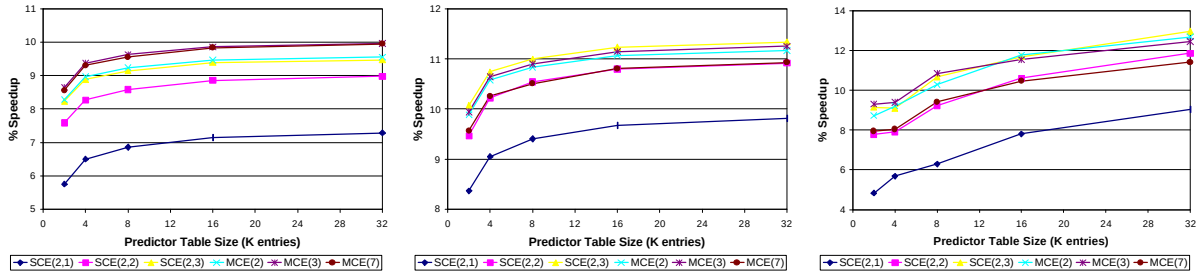


Figure 13: Speedup for different prediction models (LVP, SP and CBP).

Although the SP is the most efficacious predictor for tables under 16K (see figure 3), the CBP produces a significantly higher performance independently of the table size. Therefore, we can conclude that the efficacy measure should be essentially employed to compare different configurations of the same predictor model.

6.2 Predictor performance

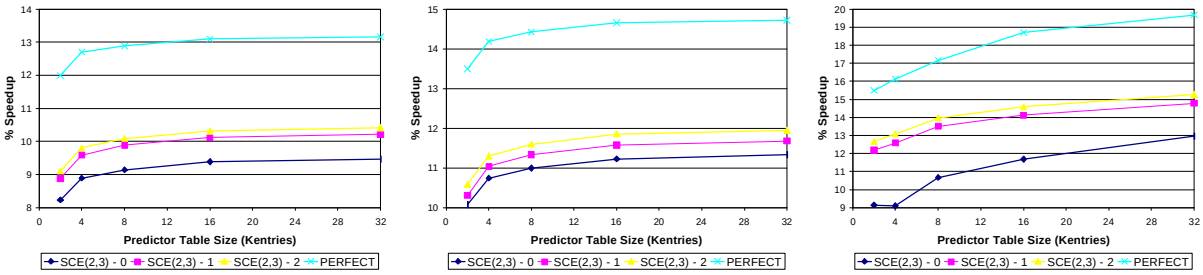
To analyse more precisely how variations in predictor performance affect the overall processor performance, it is more suitable to fix the replacement mechanism. Figure 14 shows the speedup obtained for different confidence estimator configurations (SCE and MCE) when using a SCR(2,0).



a) Last Value b) Stride c) Context-Based

Figure 14: Speedup for different SCE and MCE configurations.

As previously mentioned, predictor performance could be considered a trade-off between prediction rate and accuracy. Moderately cautious CEs – SCE(2,3), MCE(2) and MCE(3) – present a good balance between prediction rate and accuracy, and, as expected, they also produce the highest speedups. In order to reduce the number of possible combinations in the remaining analysis, we only consider the SCE(2,3), because similar results could be obtained by using a MCE(2) or MCE(3). Figure 15 shows the percentage of speedup for several HCE configurations. We observe that using history counters is as beneficial for predictor as for processor performance.



a) Last Value b) Stride c) Context-Based

Figure 15: Speedup for different HCE configurations (using SCEs).

Two main conclusions can be draw from the results presented above. Firstly, a higher performance gain could be obtained by improving the confidence estimation than by increasing the predictor size (reducing *aliasing*). Secondly, the use of a small history counter provides a significant increase in speedup, especially in the case of the CBP. For instance, a 4K-table CBP

with a *SCE(2,3) – 2* obtains as much speedup as a 32K-table CBP with a *SCE(2,3) – 0*. In view of the fact that its cost increase is relatively low (26% increase for the same table size), it is obvious that this configuration represents the most interesting alternative to implement a CBP confidence estimator from all the considered combinations.

Finally, we must remember, once more, that predictor performance and efficacy analyses are only valid for comparing predictor configurations (table size, replacement policy, confidence estimator, etc) but not for comparing predictor models. The problem lies in the differences between the predictable instructions sets. The predictors do not predict the same instructions and the instructions do not affect the performance in the same way. Due to the out-of-order issue, dynamic scheduling and forwarding, not all the predictions are useful. In fact, a moderate portion of data dependencies can be hiden by using these mechanisms. Furthermore, even is the predictions are useful, they contribute to the performance improvement with a different weighting according to the length of the broken dependence chain (see [Calder99]). Figure 16, which shows the speedup of different predictor models using a SCE(2,3) – 2, illustrates the previous statement. We observe that although the predictor performance is lower, the CBP obtains the highest speedup independently of the table size.

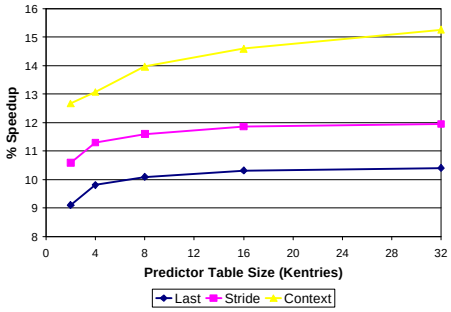


Figure 16: Speedup for a fixed HCE configurations, SCE(2,3) –2.

7. Conclusions and future work

In this paper, we have focused on analysing the effect of the value substitution and confidence estimation mechanisms on value predictor performance and on the overall processor performance. This study has been carried out by considering several different value predictors, replacement policies and confidence estimators. The main conclusions that can be drawn from this study are the following:

- Improving value substitution policy could help to better capture the sequences of values, even in the presence of a large table. Therefore, enhancements in value replacement mechanisms could be as important as reducing the *aliasing*, especially for the SP.
- Predictor performance could be considered a trade-off between prediction rate and accuracy. However, for high miss-prediction penalty processors, more emphasis should be put on accuracy.
- Confidence estimators could be hierarchically classified in function of their caution. This classification could help to select which confidence estimator is preferable for using in a particular processor model.
- Moderately cautious confidence estimators presents a good balance between prediction rate and accuracy, and also produce the highest speedups for low miss-prediction penalty architectures (e.g. re-execution recovery).
- Confidence estimation is the most important issue in improving value speculation, especially for the CBP model where conventional estimators do not work well enough. A higher potential for improvement could be obtained by using more sophisticated estimators than by reducing the aliasing.

- The potential performance benefit from improving the replacement policy is also significant. In particular, for LVP and SP, where traditional confidence estimators are performing reasonably well, progress in the replacement policy could have a definite importance.
- We propose a local history based confidence estimator that betters all the others estimators at a moderate cost increase. Among the studied confidence estimators the SCE(2,3) – 2, presents the best prediction performance to cost ratio.
- Predictor performance and efficacy analyses are useful for comparing predictor configurations (table size, replacement policy, confidence estimator, etc) but not for comparing predictor models.

We believe that there is considerable work to be done in value speculation, particularly with regard to predictor accuracy and cost. Value speculation will require cheaper predictors and more precise confidence estimators, and this is one of the main goals of our future research. Furthermore, reducing the hardware cost and access delay of the predictors will also be an important aim in our future work.

References

[Burg97] D. Burger and T.M. Austin. "The SimpleScalar Tool Set, Version 2.0". Technical Report CS#1342, University of Wisconsin-Madison, 1997.

[Beker99] M. Bekerman, S. Jourdan, R. Ronen, G. Kirshenboim, L. Rappoport, A. Yoaz and U. Weiser. "Correlated Load-Address Predictor", Proc. of the 26th Int. Symp. on Computer Architecture (ISCA-26), May 1999.

[Burt99] M. Burtscher and B. G. Zorn, "Prediction Outcome History-based Confidence Estimation for Load Value Prediction", Journal of Instruction Level Parallelism, May 1999.

[Calde99] B. Calder, G. Reinman and D. M. Tullsen. " Selective Value Prediction". Proc. of the 26th Int. Symp. on Computer Architecture (ISCA-26), May 1999.

[Gabb97] F. Gabbay and A. Mendelson, "Can Program Profiling Support Value Prediction?", Proc. of the 30th Int. Symp. on Microarchitecture (MICRO-30), pp. 270-280, Dec. 1997.

[Gabb98] F. Gabbay and A. Mendelson, “Improving Achievable ILP Trough Value Prediction and Program Profiling”, Microprocessores and Microsystems, Vol 22, n.3, Sept. 1998.

[Grun98] D. Grunwald, A. Klauser, S. Manne and A. Pleszkun, “Confidence Estimation for Speculation Control”, Proc. of the 25th Int. Symp. on Computer Architecture (ISCA-25), pp. 122-131, 1998.

[Jaco96] E. Jacobsen, E. Rotenberg, and J. E. Smith, “Assigning Confidence to Conditional Branch Predictions”, Proc. of the 29th Int. Symp. on Microarchitecture (MICRO-29), Dec. 1996.

[Lipa96] M.H. Lipasti and J.P. Shen, "Exceeding the Dataflow Limit via Value Prediction," Proc. of the 29th Int. Symp. on Microarchitecture (MICRO-29), pp. 226-237, Dec. 1996.

[Lipa97] M. H. Lipasti, “Value Locality and Speculative Execution”, Ph.D. thesis, Carnegie Mellon University, April 1997.

[McFa93] S. McFarling, “Combining Branch Predictors.” Technical Report TN-36, DEC., June 1993.

[More98] R. Moreno, “Using value prediction as a complexity-effective solution to improve performance”, Technical Report 5/98, Dep. Computer Architecture, Universidad Complutense of Madrid, 1998.

[Nakr99] T. Nakra, R. Gupta, M.L. Soffa, “Global Context-Based Value Prediction”, Proc. of the 5th Int. Symp. On High Performance Computer Architecture (HPCA-5), Jan. 1999

[Pinu99a] L. Piñuel, R.A. Moreno and F.Tirado, “Implementation of hybrid context-based value predictors using value sequence classification”. Proc. of the 5th International Euro-Par Conference, Aug. 1999.

[Pinu99b] L. Piñuel, R.A. Moreno and F.Tirado, “Effect of Saturating counters configuration on Data Value Speculation”. Technical Report 3/99, Dep. Computer Architecture, Universidad Complutense of Madrid, 1999.

[Rote96] E. Rotenberg, S. Bennett and J. E. Smith, “Trace Cache: A Low Latency Approach to High Bandwidth Instruction Fetching,” Proc. of the 29th Int. Symp. on Microarchitecture (MICRO-29), pp. 24-34, Dec. 1996.

[Rote97] E. Rotenberg, Q. Jacobson, Y. Sazeides, and J. Smith, “Trace Processors,” Proc. of the 30th Int. Symp. on Microarchitecture (MICRO-30), pp. 138-148, Dec. 1997.

[Rych98] B. Rychlik, J. Faisty, B. Krug, J.P. Shen, “Efficacy and Performance Impact of Value Prediction”, Proc. Parellel Architecture and Compilation Techniques (PACT-98), 1998.

[Saze97a] Y. Sazeides, J.E. Smith, "The Predictability of Data Values," Proc. of 30th Int. Symp. on Microarchitecture (MICRO-30), pp. 248-258, Dec. 1997.

[Saze97b] Y. Sazeides, J.E. Smith. “Implementations of Context Based Value Predictors”. Technical Report #ECE-TR-97-8, University of Wisconsin-Madison, 1997.

[Smit81] J.E. Smith, “A Study of Branch Prediction Strategies”, Proc. of the 8th Int. Symp. on Computer Architecture (ISCA-8), pp. 135-148, 1981

[Soda98] A. Sodani and G.S. Sohi, “Understanding the differences between value prediction and instruction reuse”, Proc. of 31th Int. Symp. on Microarchitecture (MICRO-31), pp. 205-215, Dec. 1998.

[Sohi90] G.S. Sohi, “Instruction Issue Logic for High-Performance, Interrumpible, Multiple Functional Unit, Pipelined Computers”, IEEE Trans. on Computer, 39(3), pp. 349-359, 1990

[Wang97] K. Wang and M. Franklin, "Highly Accurate Data Value Prediction using Hybrid Predictors," Proc. of 30th Int. Symp. on Microarchitecture (MICRO-30), pp. 281-290, Dec. 1997.

[Wall93] D.W. Wall. “Limits of Instruction-Level Parallelism” Technical Report WRL 93/6 Digital Western Research Laboratory, 1993.

[Yeh92] T-Y. Yeh, Y. Patt, “Alternative Implementations of Two-Level Adaptive Branch Prediction” Proc. of the 19th Int. Symp. on Computer Architecture (ISCA-19), pp. 124-134, 1992

[Yeh93] T-Y. Yeh, D. Marr, and Y. Patt, “Increasing the Instruction Fetch Rate via Multiple Branch Prediction and a Branch Address Cache”, Proc. of the 7th ACM Int. Conf. on Supercomputing, (ICS’93) pp. 67-76, July 1993.