The Inner Most Loop Iteration counter: a new dimension in branch history[*]

André Seznec[*] Joshua San Miguel[†] Jorge Albericio[†]
[*] IRISA/INRIA [†]University of Toronto

Abstract

The most efficient branch predictors proposed in academic literature exploit both global branch history and local branch history. However, local history branch predictor components introduce major design challenges, particularly for the management of speculative histories. Therefore, most effective hardware designs use only global history components and very limited forms of local histories such as a loop predictor.

The wormhole (WH) branch predictor was recently introduced to exploit branch outcome correlation in multidimensional loops. For some branches encapsulated in a multidimensional loop, their outcomes are correlated with those of the same branch in neighbor iterations, but in the previous outer loop iteration. Unfortunately, the practical implementation of the WH predictor is even more challenging than the implementation of local history predictors.

In this paper, we introduce practical predictor components to exploit this branch outcome correlation in multidimensional loops: the IMLI-based predictor components. The iteration index of the inner most loop in an application can be efficiently monitored at instruction fetch time using the Inner Most Loop Iteration (IMLI) counter. The outcomes of some branches are strongly correlated with the value of this IMLI counter. A single PC+IMLI counter indexed table, the IMLI-SIC table, added to a neural component of any recent predictor (TAGE-based or perceptron-inspired) captures this correlation. Moreover, using the IMLI counter, one can efficiently manage the very long local histories of branches that are targeted by the WH predictor. A second IMLI-based component, IMLI-OH, allows for tracking the same set of hard-to-predict branches as WH.

Managing the speculative states of the IMLI-based predictor components is quite simple. Our experiments show that augmenting a state-of-the-art global history predictor with IMLI components outperforms previous state-of-the-art academic predictors leveraging local and global history at much lower hardware complexity (i.e., smaller storage budget, smaller number of tables and simpler management of speculative states).

1. INTRODUCTION

Improved branch prediction accuracy directly translates to performance gain by reducing overall branch misprediction penalty. It also directly translates to energy savings by reducing the number of instructions executed on the wrong path. Therefore replacing the branch predictor with a more accurate one is the simplest, energy-effective way to improve the performance of a superscalar processor, especially considering it can be done without reopening the design of the overall execution core.

Current state-of-the-art predictors are derived from two families of predictors: the neural-inspired predictors [1, 2, 3, 4, 5, 6, 7, 8] and the TAGE-based predictors [9, 10, 11]. Both rely on exploiting the two forms of branch outcome histories that were initially recognized by Yeh and Patt [12]: global and local branch histories. The TAGE predictor [9] has been shown to exploit the global path history (i.e., the history of all branches, conditional or not) very efficiently. But, in many applications, there are a few static branches whose behavior is more correlated to the local history than to the global history. Therefore, to capture this correlation, state-of-the-art predictors presented in the literature [10, 11, 6, 7] dedicate a small portion of their storage budget to local history components in addition to a large global history component.

Evers et al. [13] demonstrate that in many cases, the outcome of a branch is correlated with the outcome of a single past branch or the outcomes of a few past branches. Global history or local history predictors do not isolate this correlation but rely on brute force to capture it. However, they fail to accurately predict branches when the number of paths from the correlator branches to the predicted branch is too large. Only a few proposals rely on identifying the correlator branches rather than the path from the correlator to the predicted branch. Among these are the loop exit predictor [14, 15] — implemented in recent Intel processors — and the recently proposed wormhole predictor [16, 17]. Albericio et al. [16, 17] demonstrate that in some cases, the outcome of a branch encapsulated in the inner most loop of a multidimensional loop is correlated with the outcomes of the same branch in neighbor iterations of the inner loop but within the previous outer loop iteration. This is illustrated in Figure 1. They propose the wormhole (WH) predictor to capture (part of) this correlation. While only addressing a few branches, the WH predictor significantly reduces the misprediction rate of a few applications within a limited storage budget cost. Therefore, TAGE-SC-L+WH [16] can be considered as the state-of-the-art branch predictor in academic

[*]To appear in Proceedings of the 48th ACM-IEEE International Symposium on Microarchitecture, Dec 2015

literature.

However, managing the speculative states of branch predictors in superscalar processors is often ignored in academic studies. Management of speculative global history can be implemented through simple checkpointing, but management of speculative local history requires searching the window of in-flight branches on each fetch cycle. Therefore, real hardware branch predictors generally rely on global history components [18], sometimes backed by a very limited local history component (e.g. a loop exit predictor [15, 14] in recent Intel processors). As managing the speculative states of the WH predictor is even more complex than that of speculative local history, the WH predictor is difficult to implement in a real hardware processor.

In this paper, we show that branch output correlations that exist in multidimensional loops can be tracked by cost-effective predictor components: the IMLI-based predictor components. The IMLI-based components can be added to a state-of-the-art global history predictor, **and** their speculative states can be easily managed.

For a given dynamic branch, the **I**nner **M**ost **L**oop **I**teration counter is the iteration number of the loop encapsulating the branch. We show that for the inner most loop, the IMLI counter can be simply monitored at instruction fetch time and therefore can be used for branch prediction. We present two IMLI-based predictor components that can be included in any neural-inspired predictor: the IMLI-SIC (**S**ame **I**teration **C**orrelation) and IMLI-OH (**O**uter **H**istory) components. The IMLI-SIC prediction table is indexed with the IMLI counter and the PC. IMLI-OH captures the same correlation as the WH predictor. IMLI-OH features a prediction table and a specific IMLI Outer History table. This history is indexed with the PC and the IMLI counter. It allows for efficiently tracking very long local histories for the same branches addressed by the WH predictor. A major advantage of the IMLI-based components over the WH predictor is that their speculative states can be managed via checkpointing of a few tens of bits.

Our experiments show that in association with a main global history predictor such as TAGE or GEHL, the two IMLI-based components achieve accuracy benefits in the same range as the ones achieved with local history and loop predictor components. This benefit is obtained at much lower hardware cost and complexity: smaller storage budget, smaller number of tables and much simpler speculative management of the predictor states. Moreover, the IMLI-based components capture part of the branch outcome correlation that was captured by the local history components and the loop predictor. Therefore, the IMLI-based components are much better candidates for real hardware implementation than local history predictors and even loop predictors.

The remainder of this paper is organized as follows. Section 2 presents the related work, the issues associated with the speculative management of branch history and the motivations for the IMLI-based components. Our experimental framework is introduced in Section 3. Section 4 presents the IMLI-based predictor components and a performance evaluation of these components on top of state-of-the-art global history branch predictors. Section 5 argues the case for implementing IMLI-based components rather than local history components in branch predictors. Finally, Section 6 concludes this study.

2. RELATED WORK AND MOTIVATIONS

2.1 General Context

Since the introduction of bimodal branch prediction [19], branch prediction has been extensively explored for three decades. A major step was the introduction of two-level branch prediction [12, 20] promoting the use of the history of branch outcomes for predicting branches. Both global branch history and local branch history were recognized as possible sources of branch outcome correlation. Using the global control flow path history instead of the global branch history was suggested in [21]. Hybrid prediction combining local history prediction tables and global branch/path history prediction tables was proposed in [22]. Using multiple prediction tables indexed with different history lengths was suggested in several studies [23, 18].

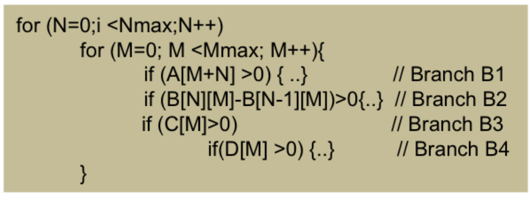
Jimenez and Lin introduced the perceptron predictor that exploits a very large number of different predictor entries [1]. Unlike the previous generation predictors, the perceptron predictor relies on accessing a large number of prediction counters (e.g. 8-bit counters) and using a tree of adders to compute the prediction. The perceptron predictor has been instrumental in the introduction of the large family of neural predictors, including the piecewise linear predictor [3], the hashed perceptron [4], the GEHL predictor [8], the SNAP predictor [5] and the FTL and FTL++ predictors [6, 7]. Apart from the complex management of speculative local history, neural predictors can smoothly combine local history components and global history components in the same predictor as illustrated by the perceptron predictor [24] and the FTL predictor [6].

However, the TAGE predictor [9] that only uses global branch history generally outperforms neural global history predictors at equivalent storage budgets. TAGE relies on prediction by partial matching [25] and the use of geometric history lengths for branch history [8]. To capture the behavior of branches that exhibit a non-perfect correlation with the global history, the TAGE predictor can be augmented with a neural-inspired component: the statistical corrector [26]. TAGE cannot accommodate global history and local history at the same time. Since a few branches are more accurately predicted using local history, it was proposed to incorporate both global and local history in the statistical corrector in TAGE-SC-L [11].

2.2 Tracking Precise Correlation

For two decades, state-of-the-art predictors have relied on exploiting the correlation of branch outcomes with global branch/path history and local history. These two forms of correlations were already recognized in the initial study by Yeh and Patt [12]. However, it is well known that in most cases, the outcome of a branch is not dependent on the complete global or local branch history but only on the outcomes of a very few branches in the past control flow [13] . We refer to these branches as the *correlator branches*.

In the general case, identifying the exact correlator branch in the past control flow is almost impossible at fetch time.



```
for (N=0;i <Nmax;N++)
        for (M=0; M <Mmax; M++){
                if (A[M+N] >0) { ..}              // Branch B1
                if (B[N][M]-B[N-1][M])>0{..}  // Branch B2
                if (C[M]>0)                       // Branch B3
                        if(D[M] >0) {..}          // Branch B4
        }
```

Figure 1: Branches whose outcomes are correlated with previous iterations of the outer loop

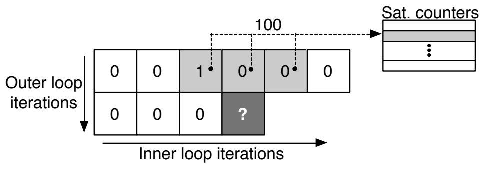


Figure 2: Example of WH prediction.

Nevertheless, predictors relying on local or global branch histories are able to capture this correlation when the number of paths (either local or global) from the correlator branch(es) to the current branch is relatively small, independent of the history length. When the number of these paths becomes large, current predictors fail to accurately predict the branch.

To our knowledge, there has only been two propositions of branch prediction mechanisms that aim at identifying the correlator branches. They both try to track specific cases: the loop predictor [14, 15] predicts the exit of the regular loops, and the recently introduced wormhole predictor [16, 17] predicts branches encapsulated in multidimensional loops. Both predictors are used as side predictors in collaboration with a main predictor.

2.2.1 The Loop Predictor

For loops with a constant number of iterations, the loop predictor identifies the last iteration of a loop by counting the number of consecutive loop iterations [14, 15].

Loop predictors have been implemented in recent Intel processors.

2.2.2 The Wormhole Predictor

Albericio et al. [16, 17] recognize that in many cases the hard-to-predict branches are encapsulated in multidimensional loops. Sometimes such a branch can be predicted using the outcomes of the branch itself in neighbor iterations of the inner loop, but in the previous iteration of the outer loop. Say B is a branch in the inner loop IL encapsulated in outer loop OL. If Out[N][M] is the outcome of B in iteration M of IL and in iteration N of OL, then Out[N][M] is correlated with Out[N-1][M+D], where D is a small number (e.g -1, 0 or +1).

This is illustrated in Figure 1. We assume that arrays A, B, C and D are not modified by the (not represented) internal code. The outcome of branch B1 in iteration (N,M) is equal to its outcome in iteration (N-1,M+1). The outcome of branch B2 is weakly correlated with its outcome in iteration (N-1,M). The outcome of branch B3 is equal to its outcome in iteration (N-1,M). If executed, the outcome of branch B4 is equal to its outcome in iteration (N-1,M)

To (partially) track these particular cases, Albericio et al. propose the wormhole (WH) predictor Similar to the loop predictor, WH is intended to be used as a side predictor. WH is a tagged structure with only a few entries (7 in the proposed design optimized for CBP4). For a branch B encapsulated in a regular loop IL (i.e., a loop predicted by the loop predictor with a constant number of iterations Ni), an entry is allocated in the WH predictor upon a misprediction. WH then records the local history of branch B. When B is fetched in iteration M of IL and in iteration N of OL, then Out[N-1][M+D] is recovered as bit Ni-D from its associated local history. Figure 2 illustrates the prediction process. WH embeds a small array of prediction counters in each entry. A few bits (grey squares in Figure 2) retrieved from the local history (as just described) are used to index this prediction array.

Since for most branches, the correlation tracked by WH does not exist, the WH prediction subsumes the main prediction only in the case of high confidence.

WH is the first predictor in the literature to track the outcome correlation of a branch encapsulated in a loop nest with occurrences of the same branch in neighboring inner loop iterations, but in the previous outer loop iteration. The number of dynamic instances of these branches can be very significant. When such correlation exists and is not captured by the main predictor, the accuracy benefit can be high as will be illustrated in Section 3.3. When associated with a state-of-the-art global history predictor, on average WH achieves accuracy improvement on the same range as local history components with a very limited number of entries [16].

WH limitations

The WH predictor exposes that there is an opportunity to exploit a new form of correlation in branch history. However, the original WH predictor has some limitations that could impair its practical implementation.

First, WH only captures the behavior of branches encapsulated in loops with a constant number of iterations. It uses the loop predictor to recognize the loop and extract the number of iterations of the loop. For instance, WH is not able to track any branch if Mmax varies in the example illustrated on Figure 1. Second, the WH predictor captures correlations only for branches that are executed on each iteration of the loop. Branches in nested conditional statements (i.e., branch B4) are not addressed by the WH predictor.

Lastly, WH uses very long local histories. The speculative management of these very long local histories is a major design challenge as detailed in the next section.

The IMLI-based predictor components proposed in this paper address these shortcomings.

2.3 A Major Challenge: The Management of Speculative Local History

In order to compute the branch prediction, the predictor states are read at prediction time; they are updated later at commit time. On a wide superscalar core, this read-to-update delay varies from a few tens to several hundreds of cycles. In the meantime, several branch instructions, sometimes tens of branches, would have already been predicted using possibly irrelevant information (i.e., stale branch histories and predictor tables entries).

On the one hand, it is well known that the delayed update of prediction tables has limited prediction accuracy impact for state-of-the-art branch predictors [27, 10]. On the other hand, using incorrect histories leads to reading wrong entries in the predictor tables and is very likely to result in many branch mispredictions [28]. Therefore, accurately managing speculative branch histories is of prime importance. Below, we contrast the simple management of speculative global history with that of speculative local history.

2.3.1 *Managing speculative global branch history*

For any branch predictor using very long global histories such as TAGE, GEHL or FTL, the management of the speculative global branch history can be implemented using a single circular buffer with two pointers: a speculative head pointer and a commit head pointer. When a branch is predicted, the predicted direction is appended to the head of the buffer and the speculative head pointer is incremented. At commit time, the commit head pointer is updated.

High-end processors repair mispredictions just after execution without waiting for commit time. To resume instruction fetch with the correct speculative head pointer, these processors rely on checkpointing [29]. The speculative head pointer is stored in the checkpoint and retrieved in the event of a misprediction. Then branch prediction and instruction fetch can resume smoothly on the correct path with the correct speculative global history head pointer.

In practice, the width of the global history head pointer to be checkpointed is small (e.g., 11 bits for the 256 Kbits TAGE-SC-L predictor [11]).

2.3.2 *Managing speculative local branch history and speculative loop iteration number*

Managing speculative local history is much more complex than managing speculative global history. On a processor with a large instruction window, distinct static branches can have speculative occurrences in-flight at the same time. In practice, speculative history can be handled as illustrated in Figure 3. The local history table is only updated at commit time. At prediction time of branch B, the local history table is read and the window of all speculatively in-flight branches is checked looking for occurrences of branch B (or more precisely of branches with the same index in the local history table). If any in-flight occurrence of branch B is detected, then the (speculative) local history associated with the most recent of these in-flight occurrences is used.

This necessitates an associative search in the window of in-flight branches. Local history must be stored with each in-flight branch in this window. On a misprediction of branch B, the branches fetched after B are flushed from the instruction window.

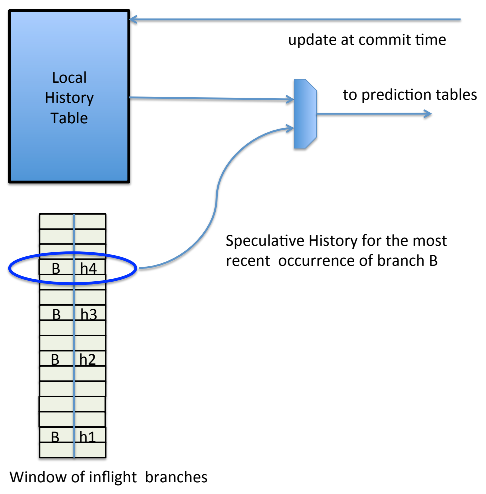

update at commit time
Local History Table
to prediction tables
Speculative History for the most recent occurrence of branch B
B h4
B h3
B h2
B h1
Window of inflight branches

Figure 3: Retrieving the speculative local history for branch B

Managing the loop iteration number

When a loop predictor is used, the current speculative loop iteration number can be managed just as described for speculative local history.

If a small loop predictor (e.g., 4 entries) is used, then an alternative is to systematically checkpoint the overall loop predictor (or at least the current loop iteration number from each entry).

Speculative states for WH predictor

Managing the speculative states of the WH predictor is essentially the same as managing the speculative local history embedded in the WH predictor entries. Therefore, it necessitates storing this long local history in the window of in-flight branches. A complex associative search in this window has to be executed on each fetch cycle.

2.3.3 *Are local history components worth the complexity?*

The prediction accuracy benefit that can be obtained from local history components is relatively small. For instance, deactivating the local history components and the loop predictor in the 256 Kbits TAGE-SC-L predictor increases the misprediction rate by only 4.8 % on CBP4 traces and by 6.5 % for CBP3 traces. A 16-entry loop predictor reclaims about one-third of these extra mispredictions.

To the best of our knowledge [30], no recent x86 processor from Intel nor AMD implements any local history components apart from the loop predictor on Intel processors.

3. EXPERIMENTAL FRAMEWORK

Throughout this paper, trace-based simulations of the branch predictors are used in order to motivate and validate the proposed designs. Misprediction rates measured as Mispredictions Per Kilo Instructions (MPKI) will be used as a metric of accuracy.

Trace-based branch prediction simulations are assuming immediate updates of the prediction tables and branch histories. On real hardware, branch histories are speculatively updated ensuring that the same prediction tables entries are read at fetch time and updated at commit time. The prediction tables are updated at commit time; thus in a few cases, a prediction table entry is read at prediction time before a previous branch in the control flow commits and updates it. However, for the state-of-the-art global history predictors considered in this paper, the delayed update of predictor tables has very limited impact on accuracy [10, 27]. Moreover, this impact can be mitigated [10].

3.1 Application Traces

To allow reproducibility of the experiments presented in this paper, all the simulations are performed using the two sets of traces that were distributed for the two last branch prediction championships in 2011 (CBP3) and 2014 (CBP4). Each set of traces features 40 traces. Traces from CBP3 were transformed in order to be compatible with simulations through the CBP4 framework.

These 80 traces cover various domains of applications including SPEC integer and floating-point applications, servers, client and multimedia applications.

3.2 Branch Predictors

The IMLI predictor components presented in this paper improve branch accuracy when combined with any of the two families of state-of-the-art branch predictors: the TAGE predictor family [9] and the neural-inspired predictor family [1, 24, 6, 8]. We consider one global history predictor from each of the two families as base references. As a representative of TAGE-based global history predictors, we use the TAGE-GSC predictor (i.e. the global history components of TAGE-SC-L [11], the winner of CBP4). As a representative of neural-inspired global history predictors, we use a GEHL predictor [8].

3.2.1 TAGE-GSC

The TAGE-SC-L predictor presented at CBP4 consists of three different functional components: a TAGE predictor [9], a statistical corrector (SC [10]) predictor featuring both local history and global history components, and a loop predictor [15, 14]. Our reference TAGE-GSC predictor[1] (Figure 4) is the exact same predictor presented at CBP4 except the loop predictor and the local history components in the SC are deactivated.

The TAGE predictor is a very effective global branch history predictor and provides the main prediction. The TAGE prediction is then used by the global history statistical corrector (GSC) predictor, whose role consists of confirming (general case) or reverting the prediction. In practice, GSC (illustrated in Figure 5) is a neural predictor featuring several tables indexed with global history (or a variation of the global history). Two predictor tables are indexed through

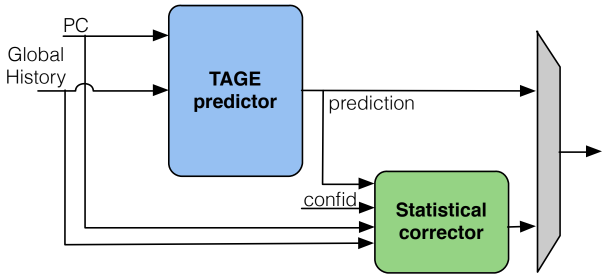


Figure 4: The TAGE-GSC predictor: a TAGE predictor backed with a Global history Statistical Corrector predictor

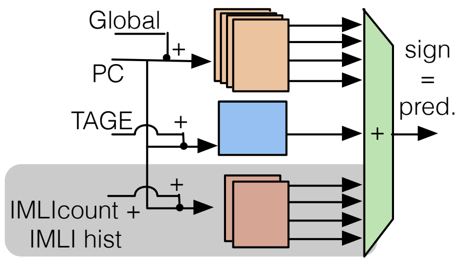


Figure 5: The Statistical Corrector predictor for TAGE-GSC with IMLI-based components

hashes of the PC concatenated with the TAGE prediction. In most cases, these later tables strongly agree with the TAGE prediction and dominate the GSC prediction. However, the GSC reverts (corrects) the prediction when it appears that in similar circumstances (PC, branch histories, etc.), TAGE has statistically mispredicted. Since TAGE is generally quite accurate, predictions are rarely reverted.

For a more complete description of the predictor configuration, please refer to [11].

Our reference TAGE-GSC predictor achieves 2.473 MPKI on CBP4 traces and 3.902 MPKI on CBP3 traces respectively. It features 228 Kbits of storage.

3.2.2 GEHL

The GEHL predictor [8] is one of the most effective neural global history predictors. Other representatives of this predictor family are SNAP [5] and the hashed perceptron [4]. GEHL is illustrated in Figure 6.

The predictor considered in this paper features a total of 17 tables with 2 K entries of 6-bit counters, indexed with global branch history. The maximum global history length is 600. This predictor features a total budget of 204 Kbits. It achieves 2.864 MPKI on CBP4 traces and 4.243 MPKI on CBP3 traces.

No specific optimizations were performed to achieve optimal misprediction rates on this base design.

3.3 Wormhole Prediction on top of TAGE-GSC and GEHL

We simulate WH as a side predictor in conjunction with TAGE-GSC or GEHL to evaluate its potential at exploiting

[1]We coined TAGE-GSC for TAGE + Global history Statistical Corrector in order to conform with the denomination TAGE-LSC introduced in [10] for TAGE + Local history Statistical Corrector.

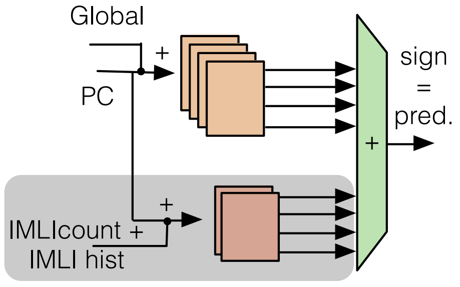


Figure 6: GEHL augmented with IMLI-based components

branch outcome correlations in multidimensional loops. The WH predictor necessitates the loop predictor to determine the number of iterations in the inner loop, but since we aim to isolate the potential of WH, the loop predictor outcome was not used for prediction but only for determining this number of iterations.

For an extra storage budget of 1413 bytes, TAGE-GSC+WH achieves 2.415 MPKI on CBP4 traces (-2.4 %) and 3.823 MPKI on CBP3 traces (-2.2 %). GEHL+WH achieves 2.802 MPKI (-2.2 %) and 4.141 MPKI (-2.5 %) on CBP4 and CBP3 traces respectively.

This accuracy benefit from the WH predictor comes from only 4 benchmarks out of a total of 80: SPEC2K6-12 and MM-4 from CBP4, and CLIENT02 and MM07 from CBP3. However, on these 4 benchmarks, the benefit is substantial: more than 1.5 MPKI for SPECK6-12, CLIENT02 and MM07 (which are hard-to-predict benchmarks with more than 11, 15 and 20 MPKI respectively on both predictors), and more than 0.3 MPKI reduction for MM-4 (with initial misprediction rates around 1 MPKI; see Figure 13).

4. IMLI PREDICTOR COMPONENTS

In this section, we propose the IMLI-based components, which are alternative approaches to predicting the class of hard-to-predict branches encapsulated in two-dimensional loops identified by Albericio et al. [16, 17].

The two IMLI-based prediction components can be added to the statistical corrector in TAGE-GSC or in the GEHL predictor (or any neural-inspired branch predictor) as illustrated for TAGE-GSC in Figure 5 and for GEHL in Figure 6. Both components exploit the Inner Most Loop Iteration (IMLI) counter, a simple mechanism that tracks the number of the current iteration in the inner most loop. The first component, IMLI-SIC (**S**ame **I**teration **C**orrelation), captures a completely different correlation than the WH predictor. The second component, IMLI-OH (**O**uter **H**istory), essentially captures the same correlation as the WH predictor.

Since the branches that we intend to address are encapsulated in multidimensional loops, throughout this section, we will use the same notation as in Section 2.2.2.

- B is a branch in inner loop IL encapsulated in outer loop OL.
- Out[N][M] is the outcome of branch B in iteration M of IL and in iteration N of OL.

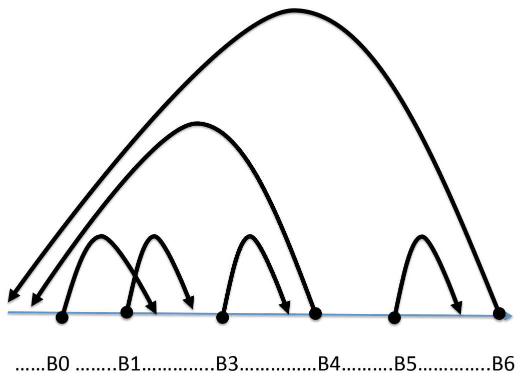

......B0 ....B1..............B3...............B4..........B5..............B6

Figure 7: Backward branches are generally loop exit branches: B4 is the inner loop exit branch, B6 is the outer loop exit branch

4.1 The Inner Most Loop Iteration Counter

In most cases, a loop body ends by a backward conditional branch (Figure 7). Therefore, for the sake of simplicity, we consider that any backward conditional branch is a loop exit branch. We will also consider that a loop is an inner most loop if its body does not contain any backward branch (e.g., branch B4 in Figure 7 is a loop exit branch of an inner most loop).

We define the Inner Most Loop Iteration counter, IMLIcount, as the number of times that the last encountered backward conditional branch has been consecutively taken. A simple heuristic allows us to track IMLIcount at fetch time for the inner most loop for any backward conditional branch:

```
if (backward){if (taken) IMLIcount++;
else IMLIcount=0;}
```

In practice, IMLIcount will be 1 or 0 on the first iteration depending on the construction of the multidimensional loop.

The IMLI counter can be used to produce the index of the two IMLI-based predictor components presented below.

4.2 The IMLI-SIC Component

In some applications, a few hard-to-predict branches encapsulated in loops repeat or nearly repeat their behavior for the same iteration in the inner most loop (i.e., Out[N][M]≡Out[N-1][M]) in most cases. For instance, this occurs when the same expression dependent on the inner most iteration number is tested in the inner loop body. In the example in Figure 1, branches B3 and B4 represent this case.

To capture this behavior, we add a single table to the statistical corrector of TAGE-GSC and to GEHL. We will refer to this table as the IMLI-SIC (Same Iteration Correlation) table. IMLI-SIC is indexed with a hash of the IMLI counter and the PC. With a 512-entries table, we capture most of the potential benefit on this class of branches on our benchmark set. However, the benefit can be further increased by inserting the IMLI counter in the indices of two tables in the global history component of the SC.

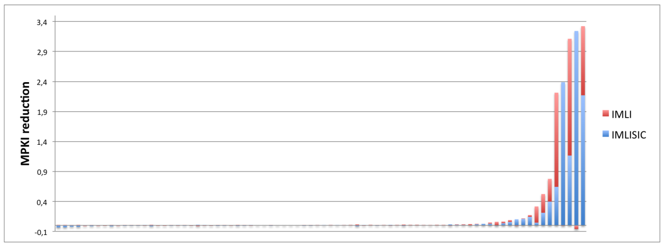


Figure 8: IMLI-induced MPKI reduction on the 80 benchmarks; TAGE-GSC predictor

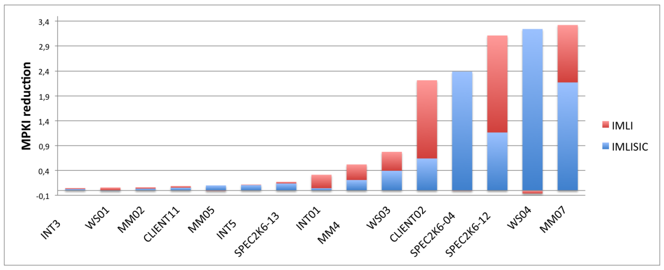


Figure 9: IMLI-induced MPKI reduction on the 15 most benefitting benchmarks; TAGE-GSC predictor

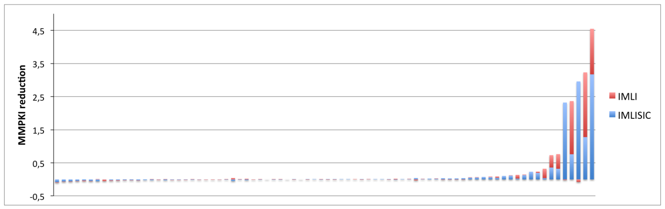


Figure 10: IMLI-induced MPKI reduction on the 80 benchmarks; GEHL predictor

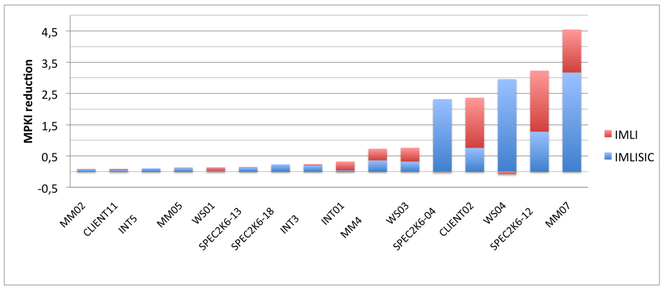


Figure 11: IMLI-induced MPKI reduction on the 15 most benefitting benchmarks; GEHL predictor

4.2.1 *Managing the speculative IMLI counter*

After the fetch of a given instruction block, the new speculative IMLI counter is derived from the previous speculative IMLI counter as well as the presence/absence of any forward branches in the instruction fetch block and their predicted directions. Checkpointing the speculative IMLI counter allows for resuming branch prediction and instruction fetch with the correct IMLI counter after a branch misprediction.

4.2.2 *IMLI-SIC performance evaluation*

Figures 8 and 9 illustrate the accuracy benefit obtained from augmenting TAGE-GSC with the two IMLI-based components on the whole set of 80 benchmarks and the 15 most improved benchmarks respectively. The benefit of IMLI-SIC alone is illustrated in the lowest bar.

IMLI-SIC reduces the average misprediction rate from 2.473 to 2.373 MPKI for CBP4 and from 3.902 to 3.733 MPKI on CBP3 traces. This benefit is essentially obtained on a few benchmarks: two CBP4 benchmarks — SPEC2K6-04 (-2.37 MPKI) and SPEC2K6-12 (-1.16 MPKI) — and three CBP3 benchmarks — WS04 (-3.20 MPKI), MM07 (-2.17 MPKI and CLIENT02 (-0.64 MPKI). The accuracies of a few other benchmarks (MM4 and WS03) are marginally improved while the other benchmarks remain mostly unchanged.

The impact of adding the IMLI-SIC table to GEHL is very similar, reducing misprediction rate from 2.864 MPKI to 2.752 MPKI for CBP4 traces and from 4.243 MPKI to 4.053MPKI for CBP3 traces. The same benchmarks as for TAGE-GSC are improved by IMLI-SIC. This is illustrated in Figures 10 and 11,

Interestingly, SPEC2K6-04 and WS04 are benchmarks that were not improved by the WH predictor. In practice, as already pointed out, the structure of the WH predictor only captures correlations for branches that are encapsulated in regular loops with constant iteration numbers and are executed on each iteration of the inner loop. IMLI-SIC does not suffer from these restrictions. On the other hand, benchmarks that are improved by WH — SPEC2K6-12, CLIENT02, MM07 and MM4 — are not as significantly improved by IMLI-SIC as with WH.

The IMLI-SIC table allows for predicting the number of iterations of the inner loop whenever the inner loop has a constant iteration number. As a result, activating the loop predictor when IMLI-SIC is enabled has limited impact. For instance, with TAGE-GSC, the benefit of the loop predictor is reduced from 0.034 MPKI to 0.013 MPKI on CBP4 and from 0.094 MPKI to 0.010 MPKI on CBP3.

4.3 The IMLI-OH Component

As mentioned above, the IMLI-SIC component does not capture all correlations that are captured by the WH predictor. We experimented adding the WH component on top of the two base predictors augmented with IMLI-SIC. This experiment showed that the WH predictor captures extra correlations that are not captured by IMLI-SIC. 2.323 (resp. 2.700) MPKI is achieved on CBP4 traces and 3.675 MPKI (resp. 3.984 MPKI) on CBP3 traces for the TAGE-GSC predictor (resp. GEHL predictor). As expected, SPEC2K6-12, MM4, CLIENT02 and MM07 are the only benchmarks that benefit from the WH component.

The analysis of these benchmarks reveals that the WH component exploits the correlation between Out[N][M] and Out[N-1][M-1] for a few inner most loop branches on SPEC2K6-12, CLIENT02 and MM07. This correlation can not be captured by IMLI-SIC. On MM4, correlations of the form Out[N][M]≡1-Out[N-1][M] are missed by IMLI-SIC. In order to track these correlations when predicting Out[N][M] for a branch B, the outcomes Out[N-1][M-1] and

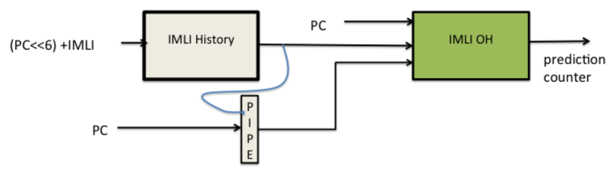


Figure 12: The IMLI-OH component

Out[N-1][M] from the previous outer iteration have to be memorized. In the WH predictor, these outcomes are memorized in the local history associated with branch B in the WH predictor entry. When predicting Out[N][M], these two outcomes are then retrieved as bits Mmax+1 and Mmax of the local history respectively, where Mmax is the number of iterations of the inner loop as predicted by the loop predictor.

4.3.1 *IMLI-OH architecture*

The IMLI-OH (**O**uter **H**istory) predictor component, illustrated in Figure 12, is an alternative solution to track Out[N-1][M-1] and Out[N-1][M] for the inner branches in two-dimensional loops using the IMLI counter. It consists of the IMLI-OH predictor table, which is incorporated in the SC part of the TAGE-GSC predictor or in the GEHL predictor. It also consists of two structures to store and retrieve the history of the previous outer loop iteration: the IMLI history table and the PIPE vector, described below.

The outcome of branches are stored in the IMLI history table. We found that a 1 Kbit table is sufficient. The outcome of a branch at address B is stored at address (B*64) + IMLIcount. This allows us to recover Out[N-1][M] when predicting Out[N][M]. However, when predicting the next iteration (i.e., Out[N][M+1]), Out[N-1][M] would have already been overwritten with Out[N][M]. Therefore, the PIPE (**P**revious **I**nner iteration in **P**revious **E**xternal iteration) vector is used to intermediately store Out[N-1][M]. This vector only contains 16 bits, corresponding to the 16 distinct branch addresses that the 1K-entry IMLI outer history table is able to track.

The IMLI-OH predictor table is indexed with the PC hashed with bits Out[N-1][M] and Out[N-1][M-1] retrieved as described above. Using a 256-entry IMLI-OH predictor table was found to be sufficient to cover all the practical cases in our set of 80 traces.

4.3.2 *Dealing with IMLI-OH speculative states*

The IMLI PIPE vector is a small structure (16 bits in our study). It can be checkpointed for each instruction fetch block.

At first glance, the speculative management of the IMLI outer history appears to be very complex, since there may be tens of speculative instances of the same branch progressing in the pipeline.

However, in practice, precise management of the IMLI outer history is not required. Analysis of simulations shows that the IMLI-OH component essentially captures correlation for branches that are encapsulated in loops with a large number of iterations. In practice, for these branches, when iteration M of IL in iteration N of OL is fetched, the occurrences around iteration M of the inner most loop IL and

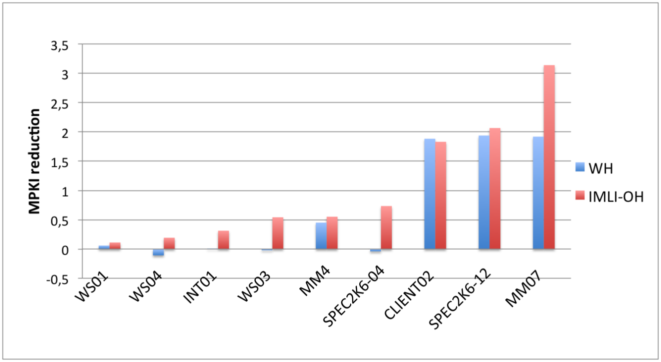


Figure 13: IMLI-OH vs WH prediction accuracy on top of the GEHL predictor

iteration N-1 of the outer loop OL have been committed for a long time. For these branches, the correct Outer History is used. For the other branches that do not exhibit IMLI counter correlations, using the incorrect Outer History has very limited impact. For these branches, IMLI-OH does not significantly contribute to the prediction but rather adds some noise to the prediction. For these branches, the leading components for the prediction compensate for this noise via weight reinforcement.

We checked that using delayed update on the IMLI history table has virtually no impact on prediction accuracy through the following experiment. We ran a simulation assuming that up to the next 63 conditional branches are fetched before the current branch updates the IMLI history table, thus accounting for the delayed update of the IMLI history table that would occur with a very large instruction window. The predictor experienced virtually no accuracy loss (0.002 MPKI).

4.3.3 *Performance evaluation of IMLI-OH*

First, we compare the benefits of IMLI-OH and WH when added to the base predictors. This is illustrated in Figure 13 for the GEHL predictor; results for TAGE-GSC are similar. As expected, the two predictors enhance the accuracy of the benchmarks that were enhanced by WH. IMLI-OH slightly enhances the accuracy of a few other benchmarks (e.g., SPECK6-04 and WS03) that are also enhanced by IMLI-SIC.

The total benefit of IMLI-SIC and IMLI-OH is illustrated as the full bar in Figures 8 and 9 for TAGE-GSC and in Figures 10 and 11 for GEHL. These benefits are obtained only on a few benchmarks but are significant for these benchmarks. Note that the benefits of IMLI-OH and IMLI-SIC are not always cumulative, as illustrated by SPECK6-04.

The overall benefits from IMLI-OH over the base predictors augmented with IMLI-SIC are proportionally smaller than the ones from IMLI-SIC alone: 2.0 % MPKI reduction on CBP4 traces (resp. 2.2 %) and 2.3 % (resp. 2.3 %) on CBP3 traces for TAGE-GSC (resp. GEHL).

4.4 Synthesis

The IMLI counter is a very simple mechanism for tracking the iteration number of the dynamic inner loop in a program. It is a graceful means of tracking the correlation between the outcome of a branch with the outcomes of the same branch in neighboring iterations in the inner most loop, but in the previous outer iteration. IMLI-OH and IMLI-SIC capture this correlation for some branches that are hard-to-predict with global history predictors.

These two predictor components can be simply added as extra tables in the statistical corrector predictor of TAGE-GSC or in the GEHL predictor. The overall storage budget for implementing the two IMLI-based components is low: a total of 708 bytes (i.e., 384 bytes for the IMLI-SIC table, 128 bytes for the IMLI outer history table, 192 bytes the IMLI OH predictor table, 4 bytes for the PIPE vector and the IMLI counter). Moreover, managing the speculative states of IMLI-SIC and IMLI-OH is as simple as that of speculative global history; it can be implemented by checkpointing only two small structures: the IMLI counter (10 bits) and the IMLI PIPE vector (16 bits).

Despite this low storage budget and hardware complexity, the IMLI-based components significantly reduce the misprediction rate for several benchmarks when added to TAGE-GSC and GEHL. For TAGE-GSC, the misprediction rate is improved by 6.8 % from 2.473 MPKI to 2.313 MPKI on CBP4 traces and by 6.1 % from 3.902 MPKI to 3.649 MPKI on CBP3 traces. For the GEHL predictor, the misprediction rate is improved by 6.0 % from 2.864 MPKI to 2.694 MPKI on CBP4 traces and 6.5 % from 3.902 MPKI to 3.649 MPKI on CBP3 traces. This misprediction reduction is most prominent for seven benchmarks: SPEC2K6-04, SPEC2K6-12 and MM-4 from CBP4 as well as CLIENT02, MM07, WS04 and WS03 from CBP3 (Figures 9 and 11). Most of the other benchmarks neither benefit nor suffer from the IMLI components as illustrated in Figures 8 and 10.

5. POTENTIAL BENEFIT OF LOCAL HISTORY

Up to now, we have considered IMLI-based components for branch predictors featuring only global history components. State-of-the-art academic branch predictors feature both local and global history components, but most real hardware processors only use global history predictors. In this section, we show that the potential accuracy benefit from using local history is further limited when using IMLI-based components.

The two base predictors, TAGE-GSC and GEHL, can be augmented with local history components. These local history components can be inserted in the SC predictor of TAGE-GSC and can be added as a local history GEHL predictor in GEHL, which yields FTL [6]. We consider augmenting both predictors with a local history component. For TAGE-GSC, we activate the local history components and the loop predictor in TAGE-SC-L [11]. For GEHL, we add 1) 4 tables of 2K 6-bit counters and a 256-entry table of 24-bit local history counters, and 2) a 32-entry loop predictor, thus yielding a FTL predictor [6].

We run simulations selectively activating the different components: Base, Base+I (I for IMLI), Base+L (L for local) and Base+I+L. The results for the 25 most affected benchmarks (out of 80) are illustrated in Figures 14 and 15. The average misprediction rates are reported in Tables 1 and 2.

	TAGE-GSC	+L	+ I	+I + L
size (Kbits)	228	256	234	261
CBP4	2.473	2.365	2.313	2.226
CBP3	3.902	3.670	3.649	3.555

Table 1: Average misprediction rate (MPKI) for TAGE-GSC-based predictors.

	GEHL	+L	+ I	+ I + L
size (Kbits)	204	256	209	261
CBP4	2.864	2.693	2.694	2.562
CBP3	4.243	3.924	3.958	3.827

Table 2: Average misprediction rate (MPKI) for GEHL-based predictors.

Overall, adding the local history predictor components and the loop predictor to the IMLI-augmented base predictors leads to lower accuracy gains than adding them to the base predictors. For TAGE-GSC, the benefit shrinks from 0.108 MPKI without IMLI to 0.087 MPKI for CBP4 traces and from 0.232 MPKI to only 0.094 MPKI for CBP3 traces. For GEHL, we observe a very similar trend with an accuracy benefit of 0.132 MPKI instead of 0.171 MPKI on CBP4 traces and 0.131 MPKI instead of 0.319 MPKI on CBP3 traces.

The IMLI components capture part of the correlations that are captured by the local history components and the loop predictor. Figures 14 and 15 show this phenomenon. When IMLI components are very effective (i.e., MM-4, SPECK2-04, SPECK6-12, CLIENT02, WS04 and MM07), the local history components are often somewhat effective, (e.g., MM07, WS04, WS03 and CLIENT02). However, their impact is only partially cumulative. On the other hand, Figures 14 and 15 also show that the benefit of local history components is more evenly distributed on the overall set of benchmarks than that of the IMLI-based components.

Summary

The accuracy benefits of using local history components and a loop predictor on top of a predictor implementing global history and IMLI-based components is limited. These reduced benefits further argue against the cost-effectiveness of local history predictor components when the predictor already features IMLI-based components.

Setting a New Branch Prediction Record

The TAGE-GSC-IMLI predictor presented in the previous sections outperforms the 256 Kbits TAGE-SC-L predictor — winner of CBP4 — despite using only 234 Kbits of storage.

Removing our self-imposed constraint of not using local history and merely adjusting the table sizes in the SC component of the TAGE-SC-L predictor, we were able to define a configuration of TAGE-SC-L enhanced with the two IMLI-based components. This configuration respects the 256 Kbits CBP4 constraint and achieves 2.228 MPKI, which is 5.8 % lower than the the record 2.365 MPKI of the original

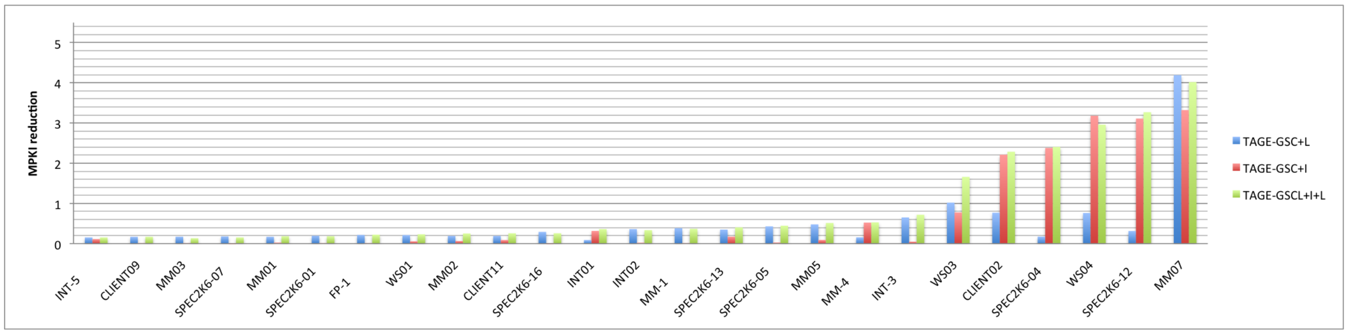


Figure 14: Benefits of local history components on TAGE;
the 25 most benefitting benchmarks

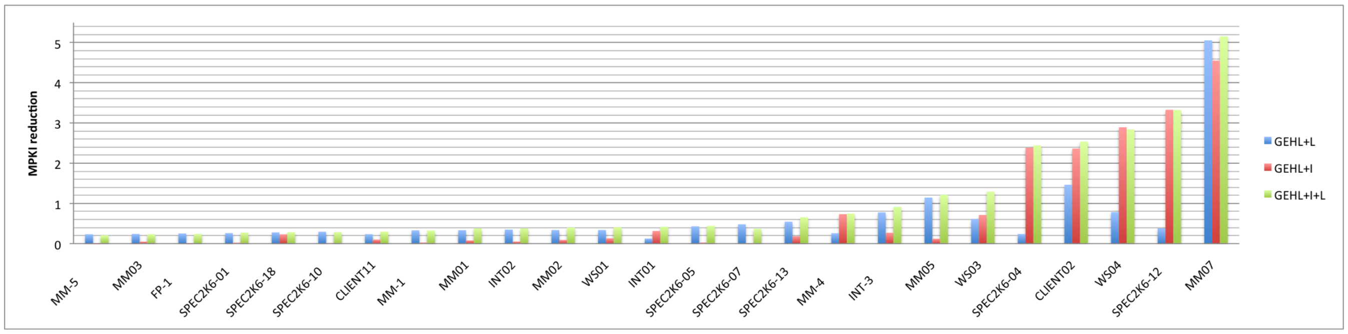


Figure 15: Benefits of local history components on GEHL;
the 25 most benefitting benchmarks

256 Kbits TAGE-SC-L predictor [11].

6. CONCLUSION

Since the introduction of two-level branch prediction [12], academic branch predictors have been relying on two forms of branch history: global branch or path history and local branch history. However, local history branch predictor components bring only limited accuracy benefit over global history predictors, yet they introduce very complex hardware management of speculative histories. Therefore, most effective hardware designs only use global history components and sometimes a loop predictor [15, 14].

Albericio et al. [16, 17] recently demonstrated that, in some cases, the outcome of a branch in the inner most loop is correlated with the outcome(s) of the same branch in the same iteration or neighbor iterations of the inner loop but in the previous outer loop iteration. They introduced the wormhole (WH) predictor to track this correlation. Unfortunately, the hardware implementation of WH is even more challenging than that of local history predictor components.

The IMLI-based predictor components build on the fundamental observation from Albericio et al. We show that the Inner Most Loop Iteration counter can be used to track two types of correlations. The IMLI counter can be simply monitored at instruction fetch time. IMLI-SIC tracks the (statistical) repetition of the same outcome for the same inner most loop iteration number for a given branch. IMLI-OH uses the correlation of the outcome of branch B with neighbor iterations in the inner loop but in the previous iteration of the outer loop. The two proposed IMLI-based predictor components can be incorporated into a neural-inspired global history predictor (e.g., GEHL) or a hybrid TAGE-neural global history predictor (e.g., TAGE-GSC). The speculative management of the predictor states of IMLI-SIC and IMLI-OH is as simple as that of the predictor states of global history components.

IMLI-based components significantly reduce the misprediction rate of several hard-to-predict benchmarks. On average, they contribute a reduction in misprediction rate that is in the same range as that of local history components, but at much lower hardware complexity: smaller storage budget, smaller number of predictor tables and much simpler management of speculative states. Moreover, IMLI-based components capture part of the correlation that is captured by local history components.

As a result, IMLI-based components could be preferred over local history components for effective hardware implementation in future generation processors.

While global history predictors are efficient at tracking the branch outcome correlations when the number of paths leading from the correlator branch to the branch to be predicted is not too large, there are still applications that suffer significant branch misprediction rates. In most cases, most of these mispredictions are encountered due to a small number of hard-to-predict branches. The approach proposed for the WH predictor [16, 17], that we extend and enhance with the IMLI-based components, isolates this correlation for the particular case of correlators within the previous iteration of the outer loop. Future developments in branch prediction research may identify other typical correlation situations and propose hardware mechanisms to exploit these correlation scenarios for other hard-to-predict branches.

Reproducibility of Simulations

The simulator used in this study can be downloaded from http://www.irisa.fr/alf/downloads/seznec/TAGE-GSC-IMLI.tar.

Acknowledgement

This work was partially supported by the European Research Council Advanced Grant DAL No 267175. This work is also supported by a Bell Graduate Scholarship, a Discovery grant, and a Strategic grant from the Natural Sciences and Engineering Research Council of Canada.

7. REFERENCES

[1] D. Jiménez and C. Lin, "Dynamic branch prediction with perceptrons," in *Proceedings of the Seventh International Symposium on High Performance Computer Architecture*, 2001.

[2] D. Jimenez, "Fast path-based neural branch prediction," in *Proceedings of the 36th Annual IEEE/ACM International Symposium on Microarchitecture*, dec 2003.

[3] D. Jiménez, "Piecewise linear branch prediction," in *Proceedings of the 32nd Annual International Symposium on Computer Architecture*, june 2005.

[4] D. Tarjan and K. Skadron, "Merging path and gshare indexing in perceptron branch prediction," *TACO*, vol. 2, no. 3, pp. 280–300, 2005.

[5] R. S. Amant, D. A. Jiménez, and D. Burger, "Low-power, high-performance analog neural branch prediction," in *MICRO*, pp. 447–458, 2008.

[6] Y. Ishii, "Fused two-level branch prediction with ahead calculation," *Journal of Instruction Level Parallelism (http://wwwjilp.org/vol9)*, May 2007.

[7] Y. Ishii, K. Kuroyanagi, T. Sawada, M. Inaba, and K. Hiraki, "Revisiting local history for improving fused two-level branch predictor," in *Proceedings of the 3rd Championship on Branch Prediction, http://www.jilp.org/jwac-2/*, 2011.

[8] A. Seznec, "Analysis of the O-GEHL branch predictor," in *Proceedings of the 32nd Annual International Symposium on Computer Architecture*, june 2005.

[9] A. Seznec and P. Michaud, "A case for (partially)-tagged geometric history length predictors," *Journal of Instruction Level Parallelism (http://www.jilp.org/vol8)*, April 2006.

[10] A. Seznec, "A new case for the tage branch predictor," in *Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO-44, (New York, NY, USA), pp. 117–127, ACM, 2011.

[11] A. Seznec, "Tage-sc-l branch predictors," in *Proceedings of the 4th Championship on Branch Prediction, http://www.jilp.org/cbp2014/*, 2014.

[12] T.-Y. Yeh and Y. Patt, "Two-level adaptive branch prediction," in *Proceedings of the 24th International Symposium on Microarchitecture*, Nov. 1991.

[13] M.Evers, S. Patel, R. Chappell, and Y. Patt, "An analysis of correlation and predictability: What makes two-level branch predictors work," in *Proceedings of the 25nd Annual International Symposium on Computer Architecture*, June 1998.

[14] D. Morris, M. Poplingher, T. Yeh, M. Corwin, and W. Chen, "Method and apparatus for predicting loop exit branches," June 27 2002. US Patent App. 09/169,866.

[15] T. Sherwood and B. Calder, "Loop termination prediction," in *High Performance Computing, Third International Symposium, ISHPC 2000, Tokyo, Japan, October 16-18, 2000. Proceedings*, pp. 73–87, 2000.

[16] J. Albericio, J. San Miguel, N. Enright Jerger, and A. Moshovos, "Wormhole: Wisely predicting multidimensional branches," in *Proceedings of the 47th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO-47, (Washington, DC, USA), pp. 509–520, IEEE Computer Society, 2014.

[17] J. Albericio, J. San Miguel, N. Enright Jerger, and A. Moshovos, "Wormhole branch prediction using multidimensional histories," in *Proceedings of the 4th Championship on Branch Prediction, http://www.jilp.org/cbp2014/*, 2014.

[18] A. Seznec, S. Felix, V. Krishnan, and Y. Sazeidès, "Design tradeoffs for the ev8 branch predictor," in *Proceedings of the 29th Annual International Symposium on Computer Architecture*, 2002.

[19] J. Smith, "A study of branch prediction strategies," in *Proceedings of the 8th Annual International Symposium on Computer Architecture*, 1981.

[20] S. Pan, K. So, and J. Rahmeh, "Improving the accuracy of dynamic branch prediction using branch correlation," in *Proceedings of the 5th International Conference on Architectural Support for Programming Languages and Operating Systems*, 1992.

[21] R. Nair, "Dynamic path-based branch correlation," in *Proceedings of the 28th Annual International Symposium on Microarchitecture*, 1995.

[22] S. McFarling, "Combining branch predictors," TN 36, DEC WRL, June 1993.

[23] P. Michaud, A. Seznec, and R. Uhlig, "Trading conflict and capacity aliasing in conditional branch predictors," in *Proceedings of the 24th Annual International Symposium on Computer Architecture (ISCA-97)*, June 1997.

[24] D. Jimenéz and C. Lin, "Neural methods for dynamic branch prediction," *ACM Transactions on Computer Systems*, vol. 20, Nov. 2002.

[25] P. Michaud, "A PPM-like, tag-based predictor," *Journal of Instruction Level Parallelism (http://www.jilp.org/vol7)*, April 2005.

[26] A. Seznec, "A 64 kbytes ISL-TAGE branch predictor," in *Proceedings of the 3rd Championship Branch Prediction*, June 2011.

[27] D. Jiménez, "Reconsidering complex branch predictors," in *Proceedings of the 9th International Symposium on High Performance Computer Architecture*, 2003.

[28] E. Hao, P.-Y. Chang, and Y. N. Patt, "The effect of speculatively updating branch history on branch prediction accuracy, revisited," in *Proceedings of the 27th Annual International Symposium on Microarchitecture*, (San Jose, California), 1994.

[29] W. W. Hwu and Y. N. Patt, "Checkpoint repair for out-of-order execution machines," in *Proceedings of the 14th Annual International Symposium on Computer Architecture*, ISCA '87, (New York, NY, USA), pp. 18–26, ACM, 1987.

[30] A. Fog, "The microarchitecture of intel, amd and via cpus, an optimization guide for assembly programmers and compiler makers," 2014.