Practical Multidimensional Branch Prediction

The most efficient branch predictors exploit both global branch history and local history, but local history predictors introduce major design challenges. Drawing from recent work on multidimensional branch prediction, the authors introduce a practical, cost-effective mechanism for overcoming the challenges of managing local histories: the innermost-loop iteration counter.

André Seznec
INRIA

Joshua San Miguel
Jorge Albericio
University of Toronto

Improved branch-prediction accuracy directly translates in performance gains through a reduction of the total overall branch misprediction penalty. It also translates into a reduction of direct energy consumption through reducing the number of instructions on the wrong path. Therefore, replacing the branch predictor with a more accurate one is a simple and energy-efficient way to improve a superscalar processor's performance, because it can be done without reopening the overall execution core's design.

Since the introduction of two-level branch prediction,[1] 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 complex hardware management of speculative histories. Therefore, most effective hardware designs use only global history components and sometimes a loop predictor.[2,3]

Researchers recently introduced the wormhole (WH) branch predictor to exploit branch outcome correlation in multidimensional loops.[4,5] 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 (see Figure 1). Unfortunately, the practical implementation of the WH predictor is even more challenging than that of local history predictors.

In this article, which is based on our MICRO 2015 paper,[6] we show that branch output correlations that exist in multidimensional loops can be tracked by cost-effective predictor components: those based on the innermost-loop iteration (IMLI). The IMLI-based components can be added to a state-of-the-art global history predictor, and their speculative states can be easily managed. Our experiments show that, in association with a main global history predictor such as the tagged geometric history length predictor[7] (TAGE) or the geometric history length predictor[8] (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 comes at a much lower hardware cost and complexity: a smaller storage budget, fewer

Published by the IEEE Computer Society

tables, and simpler speculative management of the predictor states. Therefore, the IMLI-based components are much better candidates for real hardware implementation than local history predictors and even loop predictors.

Multidimensionality

Jorge Albericio and colleagues recognized that, in many cases, hard-to-predict branches are encapsulated in multidimensional loops.[4,5] They demonstrated that the outcome of a branch in the innermost loop is correlated with the outcomes of the same branch in the same iteration or neighbor iterations of the inner loop, but in the previous outer loop iteration. 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 (such as –1, 0, or 1).

Figure 1 illustrates this. 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 $(N-1, M)$.

Wormhole Predictor

To track these particular cases, Albericio and colleagues proposed the 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 (seven in the proposed design optimized for Championship Branch Prediction 4, or CBP4). For a branch B encapsulated in a regular loop IL (that is, 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 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

```
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 B1, B2, B3, and B4 are enclosed in a nested loop and their outcomes are correlated with previous iterations of the outer loop (N loop).

counters in each entry. A few bits (the gray squares in Figure 2) retrieved from the local history (as just described) are used to index this prediction array.

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 significant. When such correlation exists and is not captured by the main predictor, the accuracy benefit can be high. 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 only a few entries.[4]

Wormhole Limitations

The WH predictor exposes the 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 captures the behavior of only those 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 cannot track any branch if M_{max} varies in the example illustrated in Figure 1. Second, the WH predictor captures correlations only for branches that are executed on each iteration of the loop. The WH predictor does not address branches in nested conditional statements (for instance, branch B4). Lastly, WH uses long local histories. The speculative management of these long local histories is a major design challenge. Our proposed IMLI-

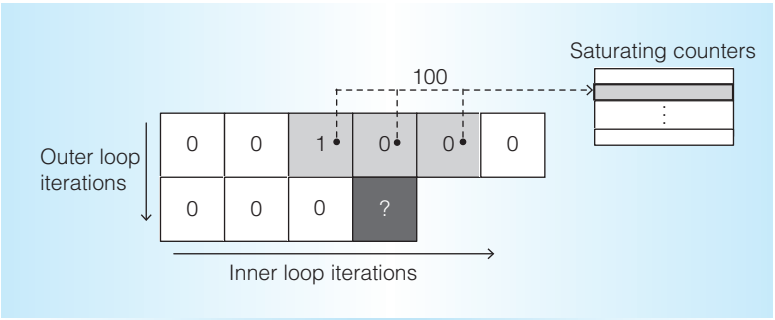

Figure 2. Example of wormhole (WH) prediction. History bits (100) from the previous outer loop iteration are used to index into a table of saturating counters.

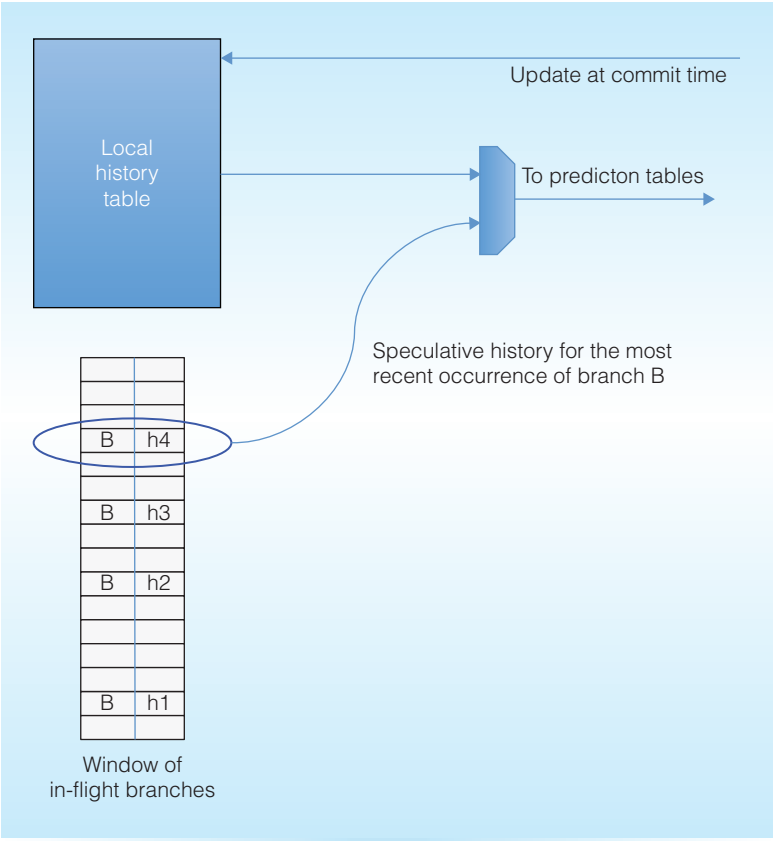

Figure 3. Retrieving the speculative local history for branch B. The window of in-flight branches is searched for the outcomes of branches that have yet to commit.

based predictor components address these shortcomings.

Speculative Local History

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 (for example, stale branch histories and predictor tables entries).

On 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.[9,10] On the other hand, using incorrect histories leads to reading wrong entries in the predictor tables and is likely to result in many branch mispredictions.[11] 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.

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 shown in Figure 3. The local history table is updated only at commit time. At the prediction time of branch B, the local history table is read and the window of all speculatively in-flight branches is checked 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, 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.

Innermost-Loop Iteration

Two IMLI-based components offer alternative approaches to predicting the class of hard-to-predict branches encapsulated in 2D loops.[4,5] These components can be incorporated into any of the two families of state-of-the-art branch predictors: the TAGE

predictor family[7] and the neural-inspired predictor family.[8,12–14] Figure 4 illustrates the addition of IMLI components to the statistical corrector in TAGE-GSC.[15] Both components exploit the IMLI counter, a simple mechanism that tracks the number of the current iteration in the innermost loop. The first component, IMLI-SIC (Same Iteration Correlation), captures a completely different correlation than the WH predictor. The second component, IMLI-OH (Outer History), essentially captures the same correlation as the WH predictor. Throughout this section, we will use the following notation when discussing branches in multidimensional loops: B is a branch in inner loop IL encapsulated in outer loop OL, and Out$[N][M]$ is the outcome of branch B in iteration M of IL and iteration N of OL.

IMLI Counter

In most cases, a loop body ends at a backward conditional branch. Therefore, for simplicity's sake, we consider that any backward conditional branch is a loop exit branch. We also consider that a loop is an innermost loop if its body does not contain any backward branch.

We define the IMLI counter, IMLIcount, as the number of times that the last-encountered backward conditional branch has been consecutively taken. A simple heuristic lets us track IMLIcount at fetch time for the innermost loop for any backward conditional branch:

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

In practice, the IMLIcount will be 1 or 0 on the first iteration, depending on the multidimensional loop's construction. We can then use the IMLI counter to produce the index of the two IMLI-based predictor components.

IMLI-SIC

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

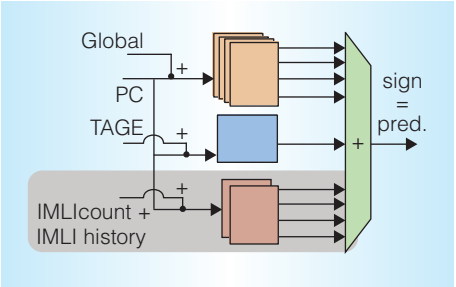

Figure 4. The statistical corrector predictor for TAGE-GSC (tagged geometric history length predictor) with innermost-loop iteration (IMLI) based components. The final prediction is selected according to the individual confidences of each predictor.

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

IMLI-OH

Our experiments showed that the IMLI-SIC component does not capture all correlations that the WH predictor does. Specifically, when predicting Out$[N][M]$ for a branch B, the outcomes Out$[N-1][M-1]$ and Out$[N-1][M]$ from the previous outer iteration also must 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 $M_{\max+1}$ and $M_{\max}$ of the local history, respectively, where $M_{\max}$ is the number of iterations of the inner loop as predicted by the loop predictor.

The IMLI-OH predictor component, shown in Figure 5, is an alternative solution to track Out$[N-1][M-1]$ and Out$[N-1][M]$

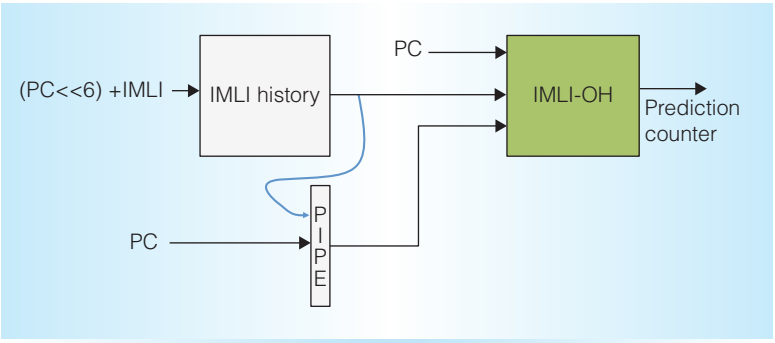

Figure 5. The IMLI-OH (Outer History) component. The component comprises a predictor table, history table, and PIPE (previous inner iteration in previous external iteration) vector.

for the inner branches in 2D loops using the IMLI counter. It comprises the IMLI-OH predictor table, which is incorporated in the SC part of the TAGE-GSC predictor, and the IMLI history table and PIPE (previous inner iteration in previous external iteration) vector, two structures to store and retrieve the history of the previous outer loop iteration.

The IMLI history table stores the branches' outcomes; 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 lets us recover Out$[N-1][M]$ when predicting Out$[N][M]$. However, when predicting the next iteration (that is, Out$[N][M+1]$), Out$[N-1][M]$ would already have been overwritten with Out$[N][M]$. Therefore, we use the PIPE vector to intermediately store Out$[N-1][M]$. This vector contains only 16 bits, corresponding to the 16 distinct branch addresses that the 1,024-entry IMLI outer history table can 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 earlier. We found that a 256-entry IMLI-OH predictor table was sufficient to cover all the practical cases in our set of 80 traces.

Speculative Management of IMLI

After the fetch of a given instruction block, we derive the new speculative IMLI counter from the previous speculative IMLI counter, along with the presence or absence of any forward branches in the instruction fetch block and their predicted directions. Checkpointing the speculative IMLI counter lets us resume branch prediction and instruction fetch with the correct IMLI counter after a branch misprediction.

For IMLI-OH, the IMLI PIPE vector is a small structure (16 bits in our study). It can be checkpointed for each instruction fetch block. 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 innermost loop IL and 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, which do not exhibit IMLI counter correlations, using the incorrect outer history has a limited impact.

Methodology

Throughout this article, we use trace-based simulations of the branch predictors to motivate and validate the proposed designs. We will use misprediction rates measured as mispredictions per kilo instructions (MPKI) as a metric of accuracy.

Trace-based branch prediction simulations assume 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 article, the delayed update of predictor tables has a limited impact on accuracy,[9,10] and this impact can be mitigated.[10]

Application Traces

To allow reproducibility of the experiments presented in this article, we performed all the simulations using the two sets of traces that

were distributed for two recent Branch Prediction Championships, in 2011 (CBP3) and 2014 (CBP4). Each set of traces features 40 traces. Traces from CBP3 were transformed to be compatible with simulations through the CBP4 framework. These 80 traces cover various application domains, including SPEC integer and floating-point applications, servers, and client and multimedia applications.

Branch Predictors

The IMLI predictor components presented here improve branch accuracy when combined with either of the two state-of-the-art branch predictor families: the TAGE predictor family[7] and the neural-inspired predictor family.[8,12–14] We consider one global history predictor from each family as base references: from the TAGE predictor family, we use the TAGE-GSC predictor (that is, the global history components of TAGE-SC-L,[15] the winner of CBP4), and from the neural predictor family, we use a GEHL predictor.[8]

Evaluation

Figures 6 and 7 illustrate the accuracy benefit obtained from augmenting TAGE-GSC with the two IMLI-based components on the whole set of 80 benchmarks and on the 15 most improved benchmarks, respectively. In both figures, the benefit of IMLI-SIC alone is illustrated by the lowest bar.

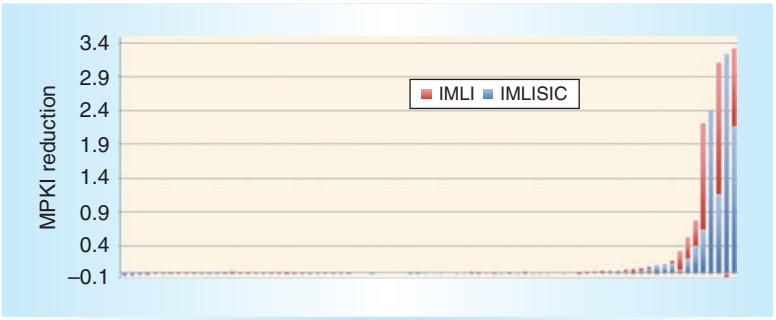

Figure 6. IMLI-induced reduction of mispredictions per kilo instructions (MPKI) on 80 benchmarks, using the TAGE-GSC predictor. The lower bar shows the MPKI reduction of IMLI-SIC alone.

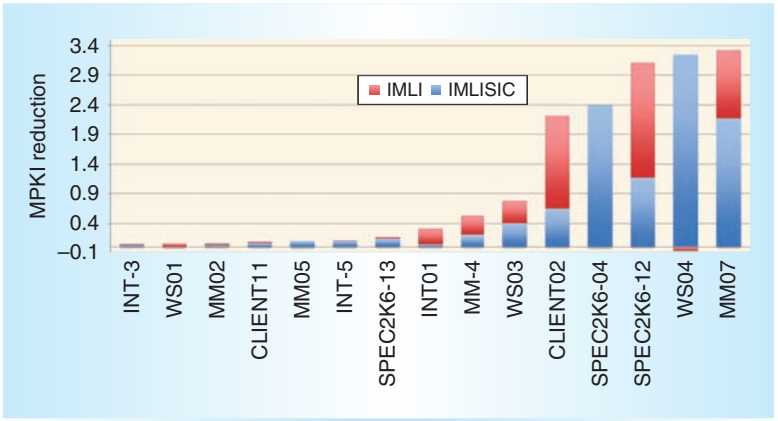

Figure 7. IMLI-induced MPKI reduction on the 15 most improved benchmarks, using the TAGE-GSC predictor. The lower bar shows the MPKI reduction of IMLI-SIC alone.

IMLI-SIC

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 two other benchmarks (MM4 and WS03) are marginally improved, whereas the other benchmarks remain mostly unchanged.

The impact of adding the IMLI-SIC table to GEHL is similar, reducing the misprediction rate from 2.864 to 2.752 MPKI for CBP4 traces and from 4.243 to 4.053 MPKI for CBP3 traces. The same benchmarks as for TAGE-GSC are improved by IMLI-SIC (see Figures 8 and 9).

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

The IMLI-SIC table lets us predict the number of iterations of the inner loop whenever the inner loop has a constant iteration

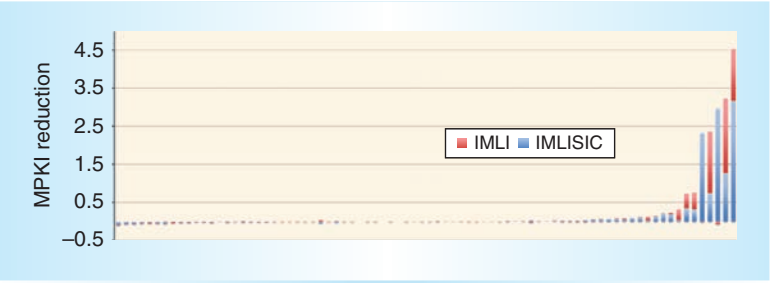

Figure 8. IMLI-induced MPKI reduction on 80 benchmarks, using the geometric history length (GEHL) predictor. The lower bar shows the MPKI reduction of IMLI-SIC alone.

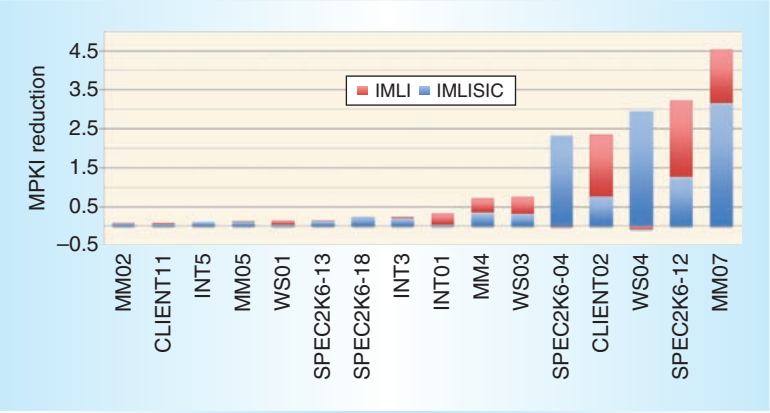

Figure 9. IMLI-induced MPKI reduction on the 15 most benefitting benchmarks, using the GEHL predictor. The lower bar shows the MPKI reduction of IMLI-SIC alone.

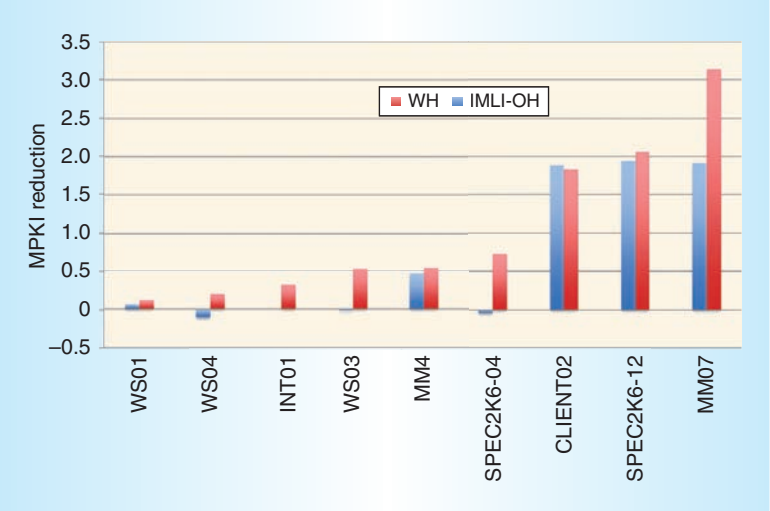

Figure 10. IMLI-OH versus WH prediction accuracy on top of the GEHL predictor for only the most improved benchmarks.

number. As a result, activating the loop predictor when IMLI-SIC is enabled has limited impact. For instance, with TAGE-GSC, the loop predictor's benefit is reduced from 0.034 to 0.013 MPKI on CBP4 and from 0.094 to 0.010 MPKI on CBP3.

IMLI-OH

First, we compare the benefits of IMLI-OH and WH when added to the base predictors. This is shown in Figure 10 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 (for example, SPECK6-04 and WS03) that are also enhanced by IMLI-SIC.

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

IMLI Overall

The total benefit of IMLI-SIC and IMLI-OH is illustrated as the full bar in Figures 6 and 7 for TAGE-GSC and in Figures 8 and 9 for GEHL. These benefits were obtained on just a few benchmarks but are significant for these benchmarks. The benefits of IMLI-OH and IMLI-SIC are not always cumulative, as SPECK6-04 shows.

For TAGE-GSC, the misprediction rate improved by 6.8 percent (from 2.473 to 2.313 MPKI) on CBP4 traces and by 6.1 percent (from 3.902 to 3.649 MPKI) on CBP3 traces. For the GEHL predictor, the misprediction rate improved by 6.0 percent (from 2.864 to 2.694 MPKI) on CBP4 traces and 6.5 percent (from 3.902 to 3.649 MPKI) on CBP3 traces. This misprediction reduction is most prominent for seven benchmarks: SPEC2K6-04, SPEC2K6-12, and MM-4 for CBP4, and CLIENT02, MM07, WS04, and WS03 from CBP3 (see Figures 7 and 9). Most of the other benchmarks neither benefit nor suffer from the IMLI components shown in Figures 6 and 8.

We can simply add these two predictor components as extra tables in the statistical

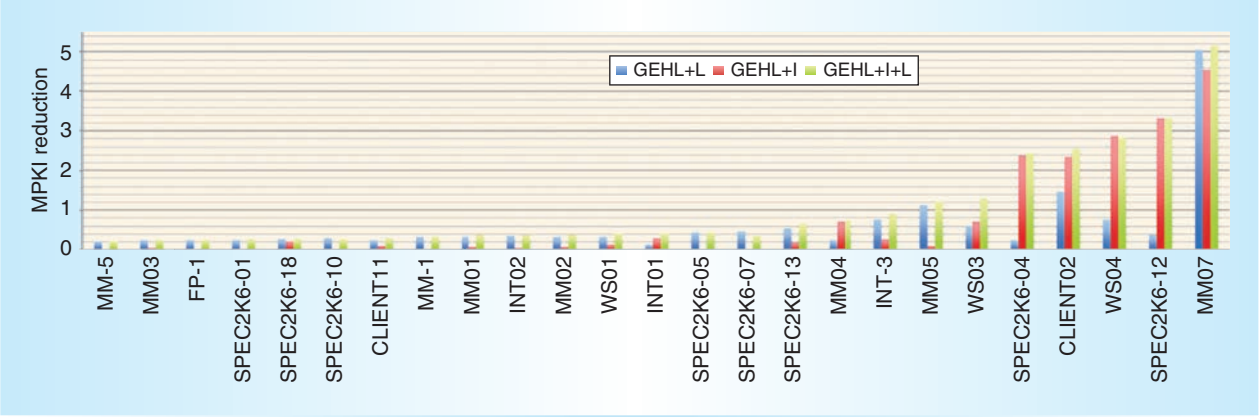

Figure 11. Benefits of local history components on GEHL for the 25 most improved benchmarks with IMLI (I), Local History (L), and both (I+L). Results for the TAGE predictor (not shown) are similar to those for GEHL.

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 (that is, 384 bytes for the IMLI-SIC table, 128 bytes for the IMLI outer history table, 192 bytes for the predictor table, and 4 bytes total for the PIPE vector and the IMLI counter). Moreover, managing the speculative states of IMLI-SIC and IMLI-OH is as simple as managing the 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.

Impact on 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 use global history predictors alone. In this section, we show that the potential accuracy benefit from using local history is further limited when using IMLI-based components.

We can augment the two base predictors, TAGE-GSC and GEHL, 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 predictor in GEHL, which yields fused two-level (FTL) prediction.[14] We considered augmenting both predictors with a local history component. For TAGE-GSC, we activated the local history components and the loop predictor in TAGE-SC-L.[15] For GEHL, we added four tables of 2,048 6-bit counters, a 256-entry table of 24-bit local history counters, and a 32-entry loop predictor, thus yielding a FTL predictor.[14]

We ran simulations selectively, activating the different components: Base, Base + I (in which *I* stands for IMLI), Base + L (local), and Base + I + L. Figure 11 shows the results for the 25 most affected benchmarks (out of 80); Tables 1 and 2 report the average misprediction rates.

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 without IMLI, the benefit shrinks from 0.108 to 0.087 MPKI for CBP4 traces, and from 0.232 to only 0.094 MPKI for CBP3 traces. For GEHL without IMLI, we observe a similar trend, with an accuracy benefit of 0.132 MPKI (vs. 0.171 MPKI) on CBP4 traces and 0.131 MPKI (vs. 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

Table 1. Average misprediction rate for TAGE-GSC-based predictors.

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

Table 2. Average misprediction rate for GEHL-based predictors.

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

predictor. Figure 11 shows this phenomenon. When IMLI components are effective (on MM-4, SPECK2-04, SPECK6-12, CLIENT02, WS04, and MM07), the local history components often are somewhat effective as well (for example, on MM07, WS04, WS03, and CLIENT02). However, their impact is only partially cumulative. On the other hand, Figure 11 also shows that the benefit of local history components is more evenly distributed on the overall set of benchmarks than that of the IMLI-based components.

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

Acknowledgments

This work was partially supported by the European Research Council Advanced Grant DAL no. 267175. This work is also supported by a Queen Elizabeth II/Montrose Werry Scholarship in Science and Technology, Bell Graduate Scholarship, a Discovery grant, and a Strategic grant from the Natural Sciences and Engineering Research Council of Canada.

References

1. T.-Y. Yeh and Y.N. Patt, "Two-Level Adaptive Training Branch Prediction," *Proc. 24th Ann. Int'l Symp. Microarchitecture*, 1991, pp. 51–61.
2. T. Sherwood and B. Calder, "Loop Termination Prediction," *Proc. 3rd Int'l Symp. High Performance Computing*, 2000, pp. 73–87.
3. D. Morris et al., *Method and Apparatus for Predicting Loop Exit Branches*, US patent 09/169,866, 2002.
4. J. Albericio et al., "Wormhole: Wisely Predicting Multidimensional Branches," *Proc. 47th Ann. IEEE/ACM Int'l Symp. Microarchitecture*, 2014, pp. 509–520.
5. J. Albericio et al., "Wormhole Branch Prediction Using Multidimensional Histories," *Proc. 4th Championship Branch Prediction*, 2014; www.jilp.org/cbp2014/paper/JorgeAlbericio.pdf.
6. A. Seznec, J. San Miguel, and J. Albericio, "The Inner Most Loop Iteration Counter: A New Dimension in Branch History," *Proc. 48th Ann. IEEE/ACM Int'l Symp. Microarchitecture*, 2015, pp. 347–357.
7. A. Seznec and P. Michaud, "A Case for (Partially) TAgged GEometric History Length Branch Prediction," *J. Instruction Level Parallelism*, 2006; www.jilp.org/vol8/v8paper1.pdf.

8. A. Seznec, "Analysis of the O-Geometric History Length Branch Predictor," *Proc. 32nd Int'l Symp. Computer Architecture*, 2005, pp. 394–405.
9. D.A. Jiménez, "Reconsidering Complex Branch Predictors," *Proc. 9th Int'l Symp. High-Performance Computer Architecture*, 2003, pp. 43–52.
10. A. Seznec, "A New Case for the TAGE Branch Predictor," *Proc. 44th Ann. IEEE/ACM Int'l Symp. Microarchitecture*, 2011, pp. 117–127.
11. E. Hao, P.-Y. Chang, and Y.N. Patt, "The Effect of Speculatively Updating Branch History on Branch Prediction Accuracy, Revisited," *Proc. 27th Ann. Int'l Symp. Microarchitecture*, 1994, pp. 228–232.
12. D.A. Jiménez and C. Lin, "Dynamic Branch Prediction with Perceptrons," *Proc. 7th Int'l Symp. High-Performance Computer Architecture*, 2001, pp. 197–206.
13. D.A. Jimenéz and C. Lin, "Neural Methods for Dynamic Branch Prediction," *ACM Trans. Computer Systems*, vol. 20, no. 4, 2002, pp. 369–397.
14. Y. Ishii, "Fused Two-Level Branch Prediction with Ahead Calculation," *J. Instruction Level Parallelism*, vol. 9, 2007, pp. 1–19.
15. A. Seznec, "TAGE-SC-L Branch Predictors," *Proc. 4th Championship on Branch Prediction*, 2014; www.jilp.org/cbp2014/paper/AndreSeznec.pdf.

André Seznec is a senior research director at INRIA. His research interests include speculative execution, pipeline design, and memory hierarchy and systems. Seznec received a PhD in computer sciences from the University of Rennes. He is an IEEE Fellow. Contact him at andre.seznec@inria.fr.

Joshua San Miguel is a PhD candidate in the Edward S. Rogers Sr. Department of Electrical and Computer Engineering at the University of Toronto. His research interests include branch prediction, approximate computing, and networks on chip. San Miguel received a BASc in engineering science from the University of Toronto. Contact him at joshua.sanmiguel@utoronto.ca.

Jorge Albericio is a postdoctoral fellow in the Edward S. Rogers Sr. Department of Electrical and Computer Engineering at the University of Toronto. His research interests include branch prediction, architectures for machine intelligence algorithms, memory hierarchy, and approximate computing. Albericio received a PhD in systems engineering and computing from the University of Zaragoza. Contact him at jorge@ece.utoronto.ca.