Wormhole Branch Prediction using Multi-dimensional Histories

Jorge Albericio, Joshua San Miguel, Natalie Enright Jerger, and Andreas Moshovos

Department of Electrical and Computer Engineering, University of Toronto
{jorge, sanmigu2, enright, moshovos}@eecg.toronto.edu

Abstract

Analyzing the competition traces, we observe two important facts: 1) branches that require large local histories are typically enclosed within nested loops, and 2) many of these branches exhibit correlations with previous iterations of not only the inner loops but also the outer loops. In this work, we present the wormhole branch predictor, a mechanism that is able to predict and exploit these types of patterns by using multi-dimensional local histories.

Our proposal is integrated on top of a base ISL-TAGE predictor. The designs submitted to the 4KB, 32KB and unlimited tracks of the 4th Branch Prediction Championship achieve 3.995, 2.498, and 2.014 MPKI, respectively.

1 Introduction

State-of-the-art branch predictors commonly rely on the use of long global histories. Though powerful and accurate, these predictors are not able to deal with branches that require large local histories. Unfortunately, storing large local histories is impractical in most cases.

In the competition traces, we observe that some of the most difficult branches to predict exhibit consistent behavior across iterations of not only the inner loops but also the outer loops. This behavior leads us to rethink the way we traditionally manage local histories. Thus, we propose representing histories as multi-dimensional matrices as opposed to linear vectors. In this work, we present the wormhole branch predictor, which is able to capture patterns in the multi-dimensional local history space.

The wormhole predictor is integrated along a base ISL-TAGE predictor [3], which consists of the TAGE predictor itself[1], a loop predictor and a statistical corrector. Our complete prediction framework is composed of four different predictors (Figure 1): 1) a loop predictor, 2) a statistical corrector, 3) a TAGE predictor that includes a bimodal predictor and many tagged components, and 4) a wormhole predictor. The first three aforementioned components are directly inherited from the original ISL-TAGE. Their parameters have been adjusted depending on the competition track.

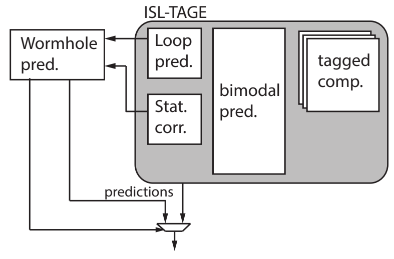


Figure 1: Design overview.

The rest of the paper is organized as follows. Section 2 briefly explains the components of the ISL-TAGE predictor. Section 3 motivates the use of multi-dimensional histories. Section 4 explains the wormhole predictor. Section 5 presents the mispredictions per kilo-instruction (MPKI) for the competition traces as well as the storage budgets of the different configurations. Finally, Section 6 concludes the paper.

2 ISL-TAGE Base Predictor

2.1 TAGE Predictor

The TAGE predictor is composed of a bimodal predictor and many tagged tables, each of which are indexed by varying global history lengths. These lengths form a geometric series. Our submission to the 4KB track considers the u field of each TAGE entry to be 3 bits. The *reset* of such field is triggered by the same mechanism as in the original ISL-TAGE, but instead of being a true reset, the value of u is decremented by one. Also, when the outcome predicted by the alternative component is different than that by the chosen component, the field u of the former is decremented by one.

2.2 Loop Predictor

The loop predictor aims to identify loops that have a constant number of iterations. An entry in the loop predictor is activated when a loop has appeared 7 times with the same number of iterations. Each entry requires 47 bits of storage.

In our predictor, the loop predictor is used to 1) determine whether a branch is inside a loop or not, and 2) determine the total number of iterations in the loop. This information is used to form a multi-dimensional representation of the local history (Section 4).

The storage required by the loop predictor for the different tracks of the competition are shown in table 2.

2.3 Statistical Corrector

The TAGE predictor is not able to predict branches that are not correlated to the global history path. However, some of these branches are statistically biased towards a certain direction. The statistical corrector aims to identify these kinds of branches.

The statistical corrector monitors the outputs of the TAGE predictor and dynamically decides whether or not to invert its prediction. To do this, the statistical corrector estimates the accuracy of the TAGE predictor for the current branch using a set of saturating counter tables. These tables are indexed with the TAGE prediction as well as a fragment of the global history. The lengths of the fragments are the same as those used in the tagged components of TAGE. The details about the decision and the update processes can be found in [3].

In our work, we use the statistical corrector to select branches that TAGE frequently mispredicts, making them candidates for our wormhole predictor. The details of this selection process are explained in Section 4.

3 Multi-Dimensional Histories

Traditional branch predictors search for patterns and correlations in the local history bits, where each bit represents the outcome of a previous instance of a branch (which is typically enclosed in a loop). Unfortunately, local history is often represented as a one-dimensional bit vector, where predictions are based solely on the most recent outcomes. Though this works for many branches, it does not capture nested loop behavior. From studying the championship traces, we observe that many problematic branches occur within nested loops and that correlations exist not only across inner loops but also across outer loops.

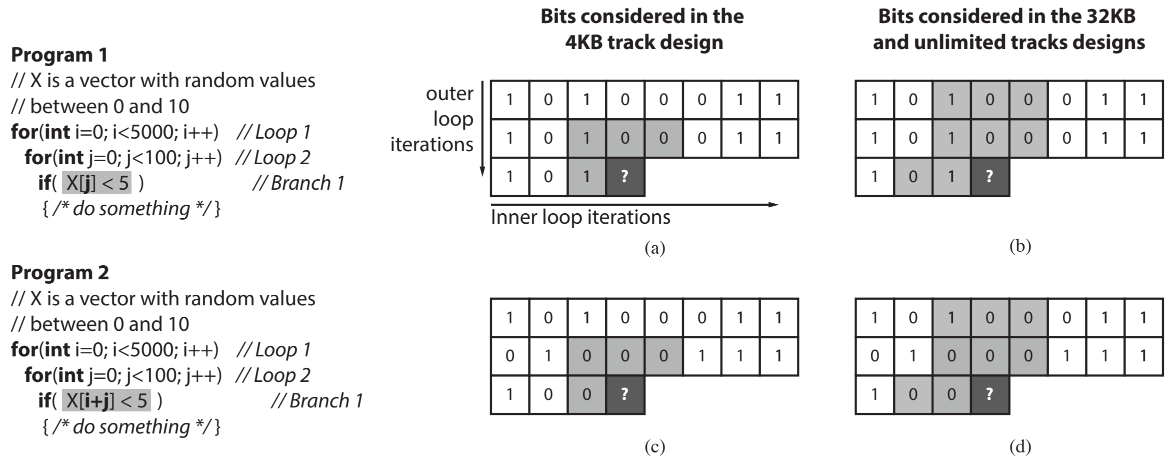
Consider the example code of Program 1 in Figure 2. If we assume that array X is initialized with random values, then predicting Branch 1 is very difficult using only the most recent local history bits, since its outcome on each iteration of the inner loop is independent of the other iterations. However, as shown in Figure 2a, Branch 1 is highly predictable in terms of the outer loop iterations (columns). Program 2 is similar, except the local history bits form a diagonal pattern across the two-dimensional local history space. This suggests that more accuracy can be gained by representing the local history as a multi-dimensional matrix as opposed to a one-dimensional vector. As a result, predictions can be based on not only the most recent outcomes in the inner loop, but also the most recent outcomes in the outer loop. Our wormhole predictor exploits this notion of multi-dimensional branch histories to improve prediction accuracy.

4 Wormhole Predictor

We propose a new mechanism that identifies the dimensionality of branches in nested loops. Our wormhole predictor effectively treats local history bit vectors as multi-dimensional bit matrices. This enables us to make predictions based on previous iterations of both the inner and outer loops. THe wormhole predictor can be divided into three stages: 1) Identify branches that are frequently mispredicted by the base TAGE predictor. 2) Detect the dimensionality of the current loop nest. 3) Learn patterns in the multi-dimensional local history space.

4.1 Identifying Problematic Branches

Since we are using ISL-TAGE as a base, naturally, our wormhole implementation specifically targets branch instructions that are problematic for the base predictor. To identify such branches, we leverage information in the statistical corrector. Recall that on every branch, the statistical corrector computes the accuracy (SC_Acc) of the current TAGE prediction. We say that a branch instruction is problematic if its accuracy (centered) is within a threshold: $abs(SC\_Acc \times 2) + 1) \leq SC\_THRES$. When a new problematic branch is encountered, an entry is allocated in the wormhole prediction table (WPT). WPT entries are ranked; if the WPT is full, the entry with the lowest ranking is selected for replacement. Whenever a branch is deemed problematic by the statistical corrector, its entry moves up one spot in the ranking. In this way, the WPT is able to identify branch instructions that are frequently mispredicted by the base predictor.



```
Program 1
// X is a vector with random values
// between 0 and 10
for(int i=0; i<5000; i++)   // Loop 1
  for(int j=0; j<100; j++)  // Loop 2
    if( X[j] < 5 )              // Branch 1
      { /* do something */ }
```

```
Program 2
// X is a vector with random values
// between 0 and 10
for(int i=0; i<5000; i++)   // Loop 1
  for(int j=0; j<100; j++)  // Loop 2
    if( X[i+j] < 5 )              // Branch 1
      { /* do something */ }
```

Bits considered in the 4KB track design
Bits considered in the 32KB and unlimited tracks designs
outer loop iterations
Inner loop iterations
(a)
(b)
(c)
(d)

Figure 2: Example programs.

4.2 Detecting Loop Dimensionality

To detect loop dimensionality, we leverage information in the loop predictor of ISL-TAGE. Whenever a branch hits in the loop predictor, a global register (LP_Iters) is updated. If the loop is currently in progress, then LP_Iters stores the total number of iterations in the loop. When the loop terminates, LP_Iters is reset to zero. Thus, at any point in time, LP_Iters stores the size of the current innermost loop. Our wormhole predictor then uses this information to represent local history bit vectors as multi-dimensional bit matrices. Our current implementation only utilizes two dimensions, but it is simple to incorporate more dimensions by adding registers to track the sizes of the outer loops.

4.3 Learning Multi-Dimensional Patterns

Table 1 lists the contents of each entry in the WPT. When making a prediction, we select specific bits that are spatially local to the current branch instance in the two-dimensional local history space, as shown in Figure 2. These bits represent the local branch outcomes from previous iterations of both the inner (horizontal) and outer (vertical) loops. To identify iterations of the outer loop, we use LP_Iters. For example, in Figure 2a, we select the bits at locations 0 (most recent), $LP\_Iters-2$, $LP\_Iters-1$ and LP_Iters in the local history vector. The selected local history bits are then used to index into a table of saturating counters, as shown in Figure 3. The wormhole prediction is prioritized above the base ISL-TAGE prediction. A prediction is only made if both the confidence is high. By exploring the local history in terms of multiple dimensions, our wormhole predictor is able to learn patterns across inner and outer loops.

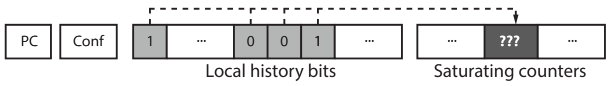


Figure 3: Wormhole predictor entry..

Track	4KB	32KB	Un.
Tag	18	18	18
Confidence	4	4	4
Sat. counters	4 (x 16)	5 (x 256)	5 (x 256)
Ranking	3	3	6
History vector	101	301	301
Vector length	7	9	9
Num. entries	5	7	64

Table 1: Wormhole prediction table configurations (all sizes in bits).

5 Evaluation and storage

5.1 Results

Figure 4 shows the mispredictions per kilo-instruction (MPKI) for the 40 traces distributed in the competition framework. These results are grouped in 5 different segments: the first group of 20 bars ($LONG$) represent the MPKI of traces from the SPEC CPU 2006 benchmarks suite. These traces contain 150 million instructions each. The next group of 20 bars are applications from 4 different domains: floating point (FP), integer (INT), multimedia (MM), and server ($SERV$). There are 5 traces from each of these domains, and each trace contains 30 million instructions. First four bars of each group of five, represent the MPKI of the original ISL-

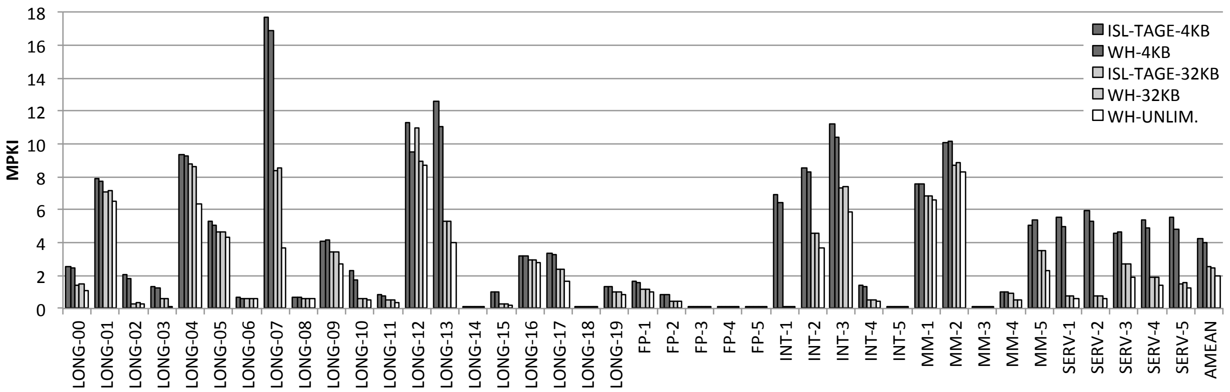


Figure 4: Misspredictions for the 40 traces.

TAGE and our design configured to target the 4KB and 32KB competition tracks. The last bar of each group represents the MPKI of our design to target the unlimited competition track.

In the first group (LONG), the average MPKI is dominated by 6 of the 20 traces, which suffer 5 MPKI or more. The MPKI decrease is constant when larger components are used, with the exception of workloads 8 and 14 (corresponding to traces LONG-07 and LONG-13). The MPKI of these two traces drastically increases (from 8.3 to 16.2, and from 5.2 to 9.5) when the tables of the predictors are shrunk from the 32KB to the 4KB budget. This is likely provoked by the high number of different branches present in that traces (4080 and 862, respectively). If a high number of branches are present along the whole trace, the small tables are not able to retain the cases learned because entries for new branches have to be allocated. SHORT-INT-2 and SHORT-INT-3, and all the traces from the SERV group also suffer drastic increases in their MPKI when the 4KB budget is considered. In the SERV case, all the traces contain between 10,000 and 16,000 different branches. Applications rich in floating-point operations present low MPKIs even with small tables.

Our design achieves 3.995, 2.498, and 2.014 for the 4KB, 32KB, and unlimited budgets, respectively.

5.2 Storage

We only show here the configurations of our design targeting the 4KB and 32KB budgets. Table 2 shows the storage used by the different components of our design. Notice that the ISL-TAGE extra space is lower than in the original proposal; this is because in this competition framework, there is no need to have separate storage for fetch and retire histories.

Our design employs 4,195 and 32,649 bytes when it targets the 4KB and 32KB budget tracks, respectively.

	4KB	32KB
Statistical corrector, size (bytes)	96	384
Loop predictor, size (bytes)	376	376
TAGE predictor, size (bytes)	3048	29952
ISL-TAGE extra, size (bytes)	524	524
Wormhole predictor, size (bits)	1065	11305
Total size (bytes)	4195	32649

Table 2: Storage of the different predictors.

6 Conclusions

In this work, we present the wormhole branch predictor, a mechanism that is able to predict branches that exhibit correlations with previous iterations of not only the inner loops but also the outer loops by using multi-dimensional local histories.

Our proposal is integrated on top of a base ISL-TAGE predictor. The designs submitted to the 4KB, 32KB and unlimited tracks of the 4th Branch Prediction Championship achieve 3.995, 2.498, and 2.014 MPKI, respectively.

References

[1] A. Seznec and P. Michaud. A case for (partially) tagged geometric history length branch prediction. *Journal of Instruction Level Parallelism*, 2006.

[2] André Seznec. A 64 kbytes isl-tage branch predictor. *JWAC-2 : Championship Branch Prediction*, 2011.

[3] André Seznec. A New Case for the TAGE Branch Predictor. In ACM, editor, *MICRO 2011 : The 44th Annual IEEE/ACM International Symposium on Microarchitecture, 2011*, December 2011.