A 2bcgskew Predictor Fused by a Redundant History Skewed Perceptron Predictor

Veerle Desmet, Hans Vandierendonck, and Koen De Bosschere
Ghent University, member HiPEAC
Department of Electronics and Information Systems (ELIS)
Sint-Pietersnieuwstraat 41, 9000 Gent, Belgium
{vdesmet|hvdieren|kdb}@elis.UGent.be

Abstract

The presented predictor fuses multiple predictions to obtain a more accurate prediction. The various predictions are delivered by a 2bcgskew predictor and include the 2bcgskew prediction itself as well as the bias and hysteresis bits of its component predictors. Together with global and local history, these predictions are used in the fusion predictor, which is an alloyed redundant history skewed perceptron predictor (RHSP).

1 Introduction

Past research demonstrated that monolithic branch predictors do not profit much from very large hardware budgets and are easily outperformed by hybrid branch predictors [1, 4]. Hybrid branch predictors use the predictions of multiple smaller predictors (the component predictors) to compute another yet more accurate prediction. Several mechanisms to combine the component predictors are documented in the literature, e.g. the meta-predictor that predicts which component predictor is correct [1, 4], the majority vote [8] and fusion [3]. Fusion appears to be the most powerful combination scheme as it is able to track the patterns from the component predictions and to learn when it should follow one of the components, the majority, etc.

In the original design, the fusion predictor takes the outcomes of the component predictors and combines these with the branch address, global history, etc. to form an index in a fusion table [3]. The fusion table contains saturating counters that store the preferred branch direction for this index. These saturating counters need to be several bits wide (e.g. 4) indicating that the true branch direction varies frequently, but the saturating counter should not track it very closely. This indicates that saturating counters are not the best choice for a fusion predictor. A different approach to compute predictions is used in the perceptron predictor [2]. The perceptron predictor computes the prediction as a weighted sum of its inputs (branch address, history bits, component outcomes). We determined through experimentation that the perceptron predictor is indeed a more accurate fusion predictor.

After experimenting with various combinations of component predictors, we found that the combination of a bimodal predictor, a gshare component, and the 2bcgskew predictor [8] gives the best results. Note that the 2bcgskew predictor itself is a hybrid predictor. It uses a meta-predictor to select between a bimodal predictor and the egskew (enhanced gskew) predictor [5]. The egskew prediction is the majority vote between a bimodal predictor and two gshare predictors, each having different history lengths. Since the 2bcgskew predictor already contains the bimodal and gshare components that we desire in the fusion-based predictor, these components need not be duplicated. This tweak makes it much easier to fit a fusion-based predictor into a 64Kbits storage bud-

get. Besides using the component outcomes (the bias bits of the predictors), it is also interesting to feed the hysteresis bits into the fusion predictor. These bits give additional information as they indicate whether the component predictor is certain (strongly biased) or uncertain (not strongly biased) about its prediction. In conclusion, the fusion predictor benefits from the outcome of the 2bcgskew predictor as well as from the outcome and internal state of the 2bcgskew components.

2 Predictor organization

The presented predictor contains two large pieces as illustrated in Figure 1: the 2bcgskew predictor (on the left) and the fusion predictor (on the right).

We relied on the optimizations to the 2bcgskew predictor as engineered by Seznec *et al.* [6, 8]. It contains a bimodal branch predictor BIM, two gshare-like predictors G0 and G1 for handling long and very long global history, respectively, and a meta-predictor META. The meta-predictor selects either the BIM prediction or the egskew prediction, i.e. the majority vote between BIM, G0, and G1. We choose to spend about half the budget into the optimized 2bcgskew. The four 2bcgskew tables together thus contain $2^N = 2^{15}$ bits of storage, which are used as follows [6]: $2^{N-1} = 2^{14}$ bits are shared by G0 and G1 prediction tables, $2^{N-2} = 2^{13}$ bits are shared by BIM and META prediction tables, and finally $2^{N-2} = 2^{13}$ bits are shared by the four hysteresis tables. These components use different history lengths to accommodate both programs that require long histories and programs that benefit from shorter histories. The history length equals $(N-12) = 3$ global branch history bits for META, $4*(N-12) = 12$ bits for G0, and $8*(N-12) = 24$ bits for G1, respectively. These lengths turn out to be slightly better than the original suggestion of using $(N-11)$ as basis. Although Seznec [6] uses history information to index the BIM component too, we found that in our setting it is more appropriate not to use history information in the bimodal component. Indeed, making the component predictors more accurate does not necessarily make the fusion predictor more accurate.

The fusion predictor is a perceptron predictor, as motivated in the introduction. We use the most accurate variant of the perceptron predictor as documented in the literature, i.e. the redundant history skewed perceptron predictor (RHSP) [7]. It is a single layer perceptron that computes the perceptron output y as the dot product of the input vector and the weight vector. The input vector contains branch address, global and local history and the outcomes of the component predictors. Each of these input bits has a weight associated to it. The output y is computed as the sum of the weights, multiplied by $(-1)^b$, where b is the input bit. The perceptron predicts taken when $y \geq 0$ and not taken otherwise.

The input vector of the fusion predictor is composed of a large number of bits including global history bits, local history bits, outcome bits of the component predictors and pseudo-tags. Pseudo-tags are the least significant bits of the branch address that have not been used to select the weight vector in the perceptron table. Including the pseudo-tag as an input to the perceptron has a similar effect on performance as increasing the number of perceptrons in the table, yet takes less space.

The accuracy of a single layer perceptron is limited as it can only learn linearly separable functions. This is a real limitation as the exclusive or (to test equality of two history bits) is a frequently occurring yet non-linearly separable function. This limitation can be relieved by explicitly adding the XOR of the history bits. This technique is known as redundant history [7]. Redundant history can be computed in many ways. We compute the n-th order redundant version of a history h as $h \oplus (h \gg n)$. Thus, for a history h with length l, at most $l-n$ useful bits of the n-th order redundant history can be computed without storing additional bits. For instance, the 2-nd order redundant version of history $[h_5, h_4, h_3, h_2, h_1, h_0]$ is defined as $[h_3 \oplus h_5, h_2 \oplus h_4, h_1 \oplus h_3, h_0 \oplus h_2]$ with only $l - n = 4$ bits. We apply redundancy separately to global history, local history and the pseudo-tags, as documented in Table 1.

The vector of weights is read from the percep-

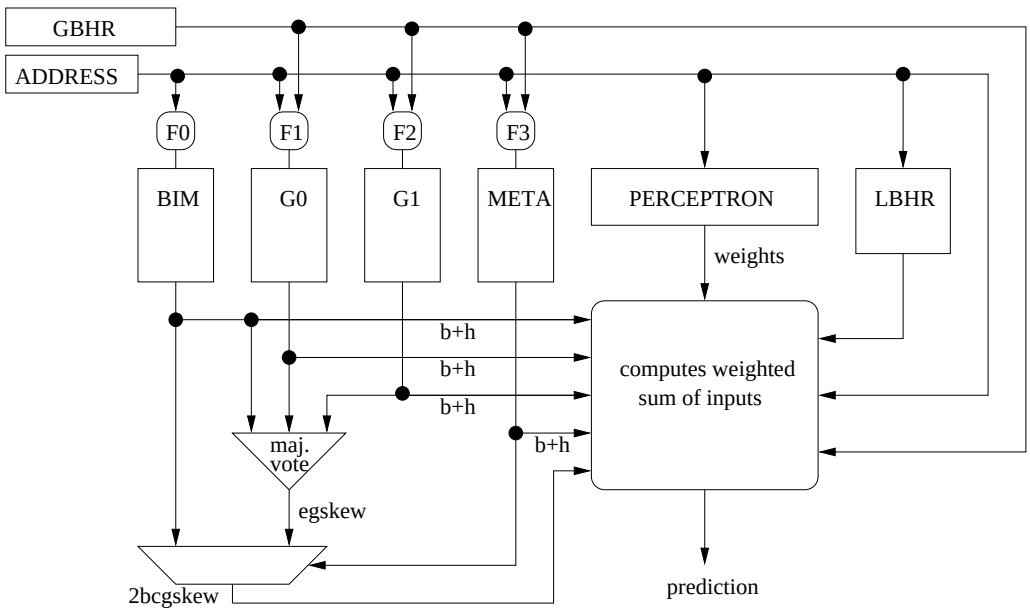

Figure 1. The 2bcgskew predictor fused by a redundant history skewed perceptron predictor.

tron table. The perceptron table is skewed [7] meaning that the perceptron table is divided into four banks, each indexed using a different hash of the branch address and the global history. The banks use 0, 3, 5 and 10 global history bits, respectively. The branch address is hashed by itself before hashing with the history bits. The address $[a_8, a_7, a_6, a_5, a_4, a_3, a_2, a_1, a_0]$ is hashed to $[a_8 \oplus a_0, a_7 \oplus a_1, a_6 \oplus a_2, a_5 \oplus a_3, a_4]$. This reorganized address vector is used in the different skewing functions, each selecting part of the weights as illustrated in Table 1. The weights are 8-bit signed integers. Neighboring weights are stored in a ((A+B),(A-B)) format [7]. Saving bits was not a purpose of this alternative bit representation, but it has a small positive impact on the predictor performance.

The 2bcgskew predictor is updated using a partial update strategy [5]. Artefacts in this strategy are avoided by periodically forcing all components to weak correct prediction. The update of the perceptron weights is similar to Jiménez's technique [2], i.e. a weight is incremented (by one) when the branch outcome agrees with the corresponding input bit, and is decremented (by one) when it disagrees. Also, the weights are only updated on a misprediction or when $|y| \leq \lfloor 1.93h + 14 \rfloor$, with $h = 116$ the total length of our input vector.

3 Budget

About half of the 64K bits budget is consumed by the 2bcgskew predictor, i.e., 32K bits for the prediction tables and 24 bits to store global history hashed with the branch address. Furthermore, 5 bits are needed for the random number generator. The fusion predictor stores the weights of 32 perceptrons. Each perceptron requires 116 8-bit weights, so the fusion predictor requires 29K bits. The local history table containing 512 entries of 6 history bits consumes 3K bits. Finally, 34 bits of global history are needed. This sums to a bit-budget of 65599 bits.

4 Conclusion

Even though the optimized 2bcgskew and the RHSP both deliver very accurate predictions, combining these predictors in a fusion-based prediction scheme outperforms their stand-alone versions. Our 2bcgskew predictor fused by a redundant history skewed perceptron predictor uses multiple predictions to obtain more accurate pre-

Table 1. Components and redundancy of the information bits supplied to a perceptron, and how these information bits are spread among the 4 banks.

Information	Redundancy	Length	Bank 1	Bank 2	Bank 3	Bank 4
Global history	pure	33	8	8	8	9
Global history	1	33	8	8	8	9
Local history	pure	6	1	2	1	2
Local history	1	5	1	1	1	2
Local history	2	4	1	1	1	1
Local history	3	3	0	1	1	1
Pseudo-tag	pure	8	2	2	2	2
Pseudo-tag	1	8	2	2	2	2
Pseudo-tag	2	6	1	2	1	2
Outcome history	pure	9	9	0	0	0
Bias	pure	1	1	0	0	0
Total information length		116	34	27	25	30

dictions. The various predictions are delivered by a 2bcgskew predictor and include the 2bcgskew prediction itself as well as the bias and hysteresis bits of its component predictors. Together with some local history, the global history, and address bits these predictions serve as input to our fusion predictor, an alloyed redundant history skewed perceptron predictor. The presented predictor implementation needs exactly (64K + 63) bits.

Acknowledgments

Veerle Desmet is supported by a grant from the Flemish Institute for the Promotion of the Scientific-Technological Research in the Industry (IWT). Hans Vandierendonck is a postdoctoral researcher of the Fund for Scientific Research-Flanders (FWO). This research was also funded by Ghent University, and the authors are indebted to Toon De Pessemier for his programming work.

References

[1] Marius Evers, Po-Yung Chang, and Yale N. Patt. Using hybrid branch predictors to improve branch prediction accuracy in the presence of context switches. In *Proceedings of the 23rd Annual International Symposium on Computer Architecture*, pages 3–11, 1996.

[2] Daniel A. Jiménez. Neural methods for dynamic branch prediction. *ACM Transactions on Computer Systems*, 20(4):369–397, November 2002.

[3] Gabriel H. Loh and Dana S. Henry. Predicting conditional branches with fusion-based hybrid predictors. In *Proceedings of the 11th International Conference on Parallel Architectures and Compilation Techniques*, pages 165–176, September 2002.

[4] Scott McFarling. Combining branch predictors. Technical Report TN-36, Digital Western Research Laboratory, June 1993.

[5] Pierre Michaud, André Seznec, and Richard Uhlig. Trading conflict and capacity aliasing in conditional branch predictors. In *Proceedings of the 24th Annual International Symposium on Computer Architecture*, pages 292–303, June 1997.

[6] André Seznec. An optimized 2bcgskew branch predictor. September 2003.

[7] André Seznec. Redundant history skewed perceptron predictors: Pushing limits on global history branch predictors. Technical Report PI-1554, Institut de Recherche en Informatique et Systèmes Aléatoires, May 2003.

[8] André Seznec, Stephen Felix, Venkata Krisnan, and Yiannakis Sazeides. Design tradeoffs for the alpha EV8 conditional branch predictor. In *Proceedings of the 29th Annual International Symposium on Computer Architecture*, pages 295–306, May 2002.