# 2FAR: A 2bcgskew Predictor Fused by an Alloyed Redundant History Skewed Perceptron Branch Predictor

Veerle Desmet Hans Vandierendonck Koen De Bosschere VEERLE.DESMET@ELIS.UGENT.BE HANS.VANDIERENDONCK@ELIS.UGENT.BE KOEN.DEBOSSCHERE@ELIS.UGENT.BE

Ghent University – UGent Parallel Information Systems (PARIS), member HiPEAC Department of Electronics and Information Systems (ELIS) Sint-Pietersnieuwstraat 41, 9000 Gent, Belgium

## Abstract

This paper describes the 2bcgskew branch predictor fused by an alloyed redundant history skewed perceptron predictor, which is our design submitted to the 1st JILP Championship Branch Prediction (CBP) competition. The presented predictor intelligently combines multiple predictions (fusion) in order 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 history, local history and address information, these predictions are used in the fusion predictor, which is an alloyed redundant history skewed perceptron predictor (RHSP). The new predictor design outperforms gshare by 40% on the CBP traces. This improvement also manifests itself for the SPEC INT 2000 benchmarks.

### 1. Introduction

Past research has demonstrated that monolithic branch predictors do not profit much from very large hardware budgets and are easily outperformed by hybrid branch predictors [1, 2]. 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, 2], the majority vote [3] and fusion [4]. Fusion appears to be the most powerful combination scheme as it can learn when it should follow one of the components or if it should follow the majority vote. Additionally, fusion adapts this decision to each individual branch and branch history.

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 into a fusion table [4]. 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 [5]. The perceptron predictor computes the prediction as a weighted sum of its inputs (branch address, history bits, component outcomes). Through experimentation we determined that the perceptron predictor is indeed a more accurate fusion predictor.

After experimenting with various combinations of component predictors, we found the combination of a bimodal predictor, a gshare component, and the 2bcgskew predictor [3] gives the best results. Note that the 2bcgskew predictor itself is a hybrid predictor since it uses a meta-predictor to select between a bimodal predictor and the egskew (enhanced gskew) predictor [6]. 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 do not need to be duplicated. Moreover, this tweak makes it much easier to fit a fusion-based predictor into a fixed bit budget. Besides using the component outcomes (the bias bits of the predictors), the hysteresis bits were also fed 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.

This paper is organized as follows. After detailing the predictor organization in section 2, we present our baseline predictor in section 3. Performance results for a 2FAR predictor are discussed in section 4. This includes a comparison to other branch predictors, a sensitivity analysis to the parameter set, a demonstration of the scalability to other budgets, and results for the frequently reported SPEC INT 2000 traces. Section 5 concludes the paper.

#### 2. Predictor organization of 2FAR

The 2bcgskew predictor fused by an alloyed redundant history skewed perceptron predictor, which we call 2FAR, contains two large pieces as illustrated in Figure 1. On the left we show the 2bcgskew predictor with its four components, which is connected to the fusion predictor shown on the right.

The **2bcgskew** part contains a bimodal branch predictor BIM, two gshare-like predictors G0 and G1 for handling long and very long global histories 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.

A budget of  $2^N$  bits for the four 2bcgskew tables is spent as engineered by Seznec *et al.* [3, 7]:

- $2^{N-1}$  bits are shared by the G0 and G1 prediction tables,
- $2^{N-2}$  bits are shared by the BIM and META prediction tables, and finally
- $2^{N-2}$  bits are shared by the four hysteresis tables.

Further, each of the component predictors uses different lengths from the global branch history register (GBHR) to accommodate both programs that require long histories and programs that benefit from shorter histories. The history length equals (N-12), 4\*(N-12)and 8\*(N-12) bits for the META, G0 and G1 tables respectively. These lengths turn out to be slightly better than the original suggestion of using (N-11) bits. Although Seznec [7] uses history information to index the BIM component too, we found that in our setting it is

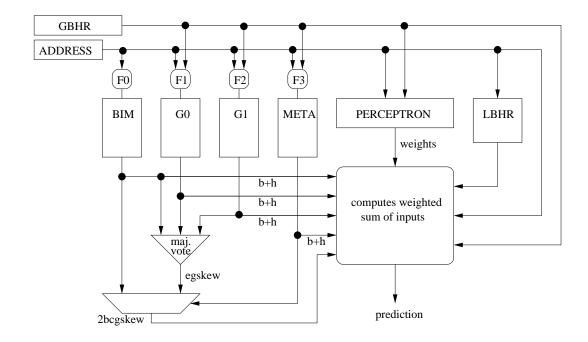


Figure 1: The 2FAR predictor: a 2bcgskew predictor fused by an alloyed redundant history skewed perceptron predictor.

more appropriate not to use history information. Indeed, making the component predictors more/ accurate does not necessarily make the fusion predictor more accurate.

The **fusion predictor** is a perceptron predictor, which is motivated in the introduction. We use the most accurate variant of the perceptron predictor documented in the literature, i.e. the redundant history skewed perceptron predictor (RHSP) [8]. This RHSP is a *single layer perceptron* that computes the perceptron output y as the dot product of an input vector and a weight vector. The input vector of the fusion predictor is composed of a large number of bits including global history bits, local history bits, bits representing internal state 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. Each of these input bits has a weight associated with 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  $y \ge 0$  and not taken otherwise.

The vector of weights is read from the perceptron table. This perceptron table is *skewed* [8], meaning that it is divided into 4 banks, each bank containing a subset of the weights, and each indexed using a different hash of the branch address and the global history. The branch address is hashed with itself (further termed as *PC hashing*) 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  $[a_8 \oplus a_0, a_7 \oplus a_1, a_6 \oplus a_2, a_5 \oplus a_3, a_4]$ . The 4 banks use 0,  $\lfloor (i+1)/2 \rfloor$ , *i* and 2*i* global history bits respectively, where *i* denotes the number of bits to index the perceptron table.

The reorganized address vector and the history are used in the different (skewed) hashing functions, each selecting a subset of the weights. The weights are 8-bit signed integers. Neighboring weights are stored in a ((A+B),(A-B)) format [8]. Saving bits was not a purpose of this alternative bit representation, but it has a small positive impact on the predictor performance.

The accuracy of a single layer perceptron is limited as it can only learn linearly separable functions. This is a real limitation as XOR (to test the 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 *redundancy* [8]. Redundant histories can be computed in many ways. We compute the *n*-th order redundant version of a history *h* as  $h \oplus (h \gg n)$ . For a history *h* with length *l*, there are at most l - nbits for its *n*-th order redundant history. 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  $[h_3 \oplus h_5, h_2 \oplus h_4, h_1 \oplus h_3, h_0 \oplus h_2]$  and has only l - n = 6 - 2 = 4 bits. We apply redundancy separately to global history, local history and the pseudo-tags as will be further discussed in the next section.

During **update**, the 2bcgskew predictor uses a partial update strategy [6]. All components are periodically forced to weakly correct predictions to avoid the problems associated with this strategy. The update of the perceptron weights is similar to Jiménez's technique [5], 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 the total length of the input vector.

# 3. Baseline 2FAR

While the previous section describes the 2FAR predictor in general, this section details the baseline 2FAR configuration of 64K bits. Note that this configuration differs from the one used for the CBP competition and gives slightly better performance (3.15 versus 3.17 mispredicted conditional branches per kilo instructions [MPkI]).

Roughly half of the 64K bits budget available for the competition is consumed by the 2bcgskew predictor, i.e.  $2^{15}$  bits for the shared component tables and 24 bits to store the global history. The fusion predictor has a perceptron table with 32 entries. The 4 banks of the perceptron are indexed with 0, 3, 5, and 10 global history bits. Each perceptron entry has 127 8-bit weights so the fusion predictor requires 31.7K bits. The corresponding input bits are documented in Table 1 together with their distribution between the four banks. The 6 bits of local history are taken from a 64-entry local history table which consumes 0.4K bits. Finally, 40 bits of global history are needed. The grand total is 65728 bits.

#### 4. Evaluation

We mainly evaluate our 2FAR predictor with the distributed CBP traces each of which contains approximately 30 million instructions and includes both user and system activity. These CBP traces belong to four categories—SPECfp (FP), SPECint (INT), Multimedia (MM), and Server (SERV)—with 5 traces per category. Any reported average is the arithmetic mean over these 20 CBP traces. The metric for comparing branch prediction tech-

| Information              | Redundancy | Length | Bank 1 | Bank 2 | Bank 3 | Bank 4 |
|--------------------------|------------|--------|--------|--------|--------|--------|
| Global history           | pure       | 40     | 10     | 10     | 10     | 10     |
| Global history           | 1          | 37     | 9      | 9      | 9      | 10     |
| 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 |            | 127    | 37     | 30     | 28     | 32     |

Table 1: Information bits supplied to the perceptron component of the baseline 2FAR at 64Kbits. The redundancy of the bits is shown as well as how the bits are distributed to the 4 banks.

niques is the number of mispredicted conditional branches per kilo instructions (MPkI), which is a better metric than misprediction rate because it also captures the density of conditional branches in a program [9]. Obviously, lower MPkI is better.

In this section, we first compare the 2FAR predictor against three well-studied branch predictor designs—gshare [2], 2bcgskew [7], and the redundant history skewed perceptron branch predictor [8]—at the bit budget available for the contest; i.e. (64K + 256) bits. After a sensitivity analysis to the many parameters in the 2FAR predictor, we extend the 2FAR design to explore both smaller and larger bit budgets. To conclude, we illustrate the performance on the SPEC INT 2000 traces because these are frequently used in today's branch predictor studies.

#### 4.1 Comparison to other branch predictors

Starting with an average 5.3 MPkI for the baseline gshare branch predictor provided with the framework, we gradually improve the performance by introducing more complexity into the branch predictor. In search of more accurate predictors, there are two important milestones worth mentioning. The first is a 64K bits 2bcgskew branch predictor for which we measure 3.87 MPkI. In this design, the 2bcgskew is exactly as engineered by Seznec [7] and not optimized to be part of a fusion predictor scheme; i.e. the history lengths are computed as N - 12. The second is a RHSP (more specifically, an alloyed skewed perceptron predictor without redundancy) that consumes the total bit budget. This predictor reaches 3.42 MPkI. Its input layer consists of the (mandatory) bias input, 35 global history bits, 8 pseudo-tag bits, and 4 local history bits, which are read from a 4096 entry local history table. The 8-bit weights are read from a 128-entry perceptron table. The table's 4 banks are accessed

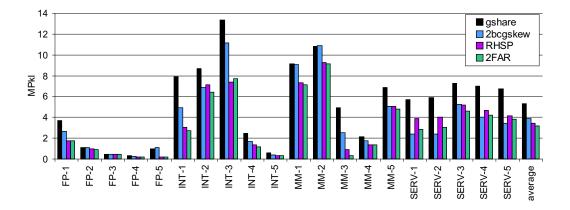


Figure 2: Performance comparison at 64Kbits between gshare, 2bcgskew, RHSP, and 2FAR.

using indices generated from (skewed) hash functions of the branch address and either 0, 4, 8, or 16 global history bits.

The 2bcgskew fused by an alloyed RHSP (i.e. 2FAR) on average outperforms both aforementioned prediction schemes achieving 3.15 MPkI. However, this does not hold for every individual trace. E.g. for INT-3 the RHSP would be a better choice, and most server traces can significantly benefit from a pure 2bcgskew design. In the next section, we focus on specific design options and their impact on the individual traces.

#### 4.2 Sensitivity analysis of parameters

Due to its fusion and alloyed nature, the 2FAR branch predictor is susceptible to adjustments of a lot of parameters, e.g. history length, table size, redundancy etc. In this section, we take the baseline described in section 3, and we quantify the influence of the various design aspects of the 2FAR predictor by varying one aspect at a time.

In our first approach, we neglect one aspect (e.g. redundancy) and reconfigure the 2FAR predictor in order to consume the complete 64Kbits in an optimal way. Figure 3 summarizes this analysis by illustrating absolute increase of MPkI (negative values are for reductions) over the baseline 2FAR predictor for various designs. We order the impact of the design features from high to low, as follows:

- **Removing skewing** from the fusion perceptron predictor has a major negative effect on MPkI, increasing it to 3.78, which is over 16% worse on average, and this trend holds for every individual trace. Skewing limits aliasing in the perceptron weight table, and since it does not require additional storage it is a very effective way to increase accuracy.
- If global history is discarded, the 2FAR design replaces the global history inputs with older and more accurate local history bits: 17 bits out of a 512-entry local history

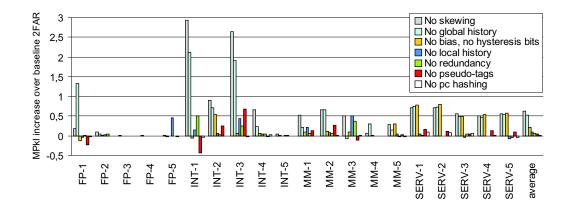


Figure 3: Sensitivity to design options.

table. Except for some minor reductions, all traces show significant increase in MPkI when no global history is available.

- No fusion—not shown in Figure 3—represents the pure RHSP configuration from section 4.1. As said before, only INT-3 significantly benefits from not fusing.
- Applying simple fusion—only the 2bcgskew outcome is connected to the RHSP, the **bias and hysteresis bits are not**—increases the MPkI by 6.5% up to 3.37. In order to consume the total bit budget, this design allocates 6 additional weights for global history and 2 more for redundant pseudo-tag information (in exchange for the weights associated with bias and hysteresis bits). All traces except FP-1 and INT-1 benefit from using internal state from the 2bcgskew components as input to the skewed perceptron predictor.
- Without local history the MPkI would increase by 3% to 3.25. In exchange for the freed bit budget, we allocate a much longer global history (up to 48 bits) and slightly increase the number of pseudo-tag inputs. Despite substantial losses for some traces like FP-5, INT-3, and MM-3, most traces are insensitive to the presence of local history information.
- When **redundant information is left out** of the history vector supplied to the perceptron, the simplest configuration change is to just use more original information. However, the optimal design in this case doubles the entries of the perceptron table in addition to slightly shortening the global history length. This illustrates the importance of reducing aliasing in the perceptron table over lengthening the global history.
- Absence of pseudo-tags shows minor influence on average. However, discarding pseudo-tags increases the MPkI of INT-1 by 15%. Including pseudo-tags 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.

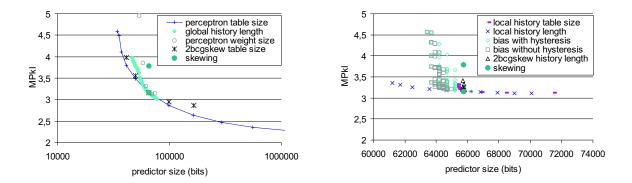


Figure 4: Tuning the different parameters of a 2FAR predictor.

• **PC hashing** or rotating the program counter before hashing it with itself has little impact, but it also has no cost.

In an orthogonal approach, we consider the impact that changing the various design parameters of our baseline 2FAR predictor has on the bit budget in order to explore how to use smaller and larger budgets. Figure 4 shows average MPkI versus predictor size. For clarity the results are broken down into two figures according to whether the impact on predictor size is large (left) or small (right).

In the left figure, we show:

- The size of the perceptron table, varied in number of entries, has a large influence on both MPkI and predictor size.
- The global history length has a major impact on the overall predictor size because it mainly determines the number of weights that have to be stored in the perceptron table. Since MPkI continued to decrease up to a length of 64 bits, which is the maximum length we simulated, very long global histories do contain relevant information and should be considered in future research.
- For the **perceptron weight size** an optimum is obtained at 8 bits. Smaller weights are not accurate enough, whereas larger weights do not further decrease the MPkI.
- The **2bcgskew proportion**. Here we found that half the budget needs to be spent on the 2bcgskew predictor to provide accurate component predictions for fusion.

The right figure (which uses a different scale on the x-axis) illustrates that:

- The optimal local history table size is 64 entries, which is used in the baseline configuration.
- Local history length has a minor influence and only effects a few benchmarks (INT-1, MM-3, and MM-5).
- Nine information bits flow from the components of the 2bcgskew to the alloyed RHSP: the bias and hysteresis bits of each of the four 2bcgskew components and the final

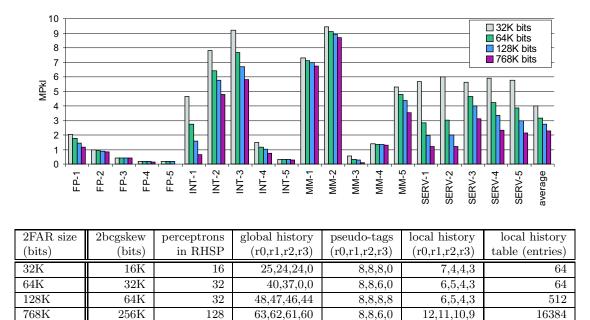


Figure 5: Up: Performance for various 2FAR predictor sizes. Down: Details on configuration settings—r0,r1,r2,r3 denote the lengths (bits) for the pure, 1st, 2nd, 3rd redundancy level respectively.

2bcgskew prediction bit. We distinguish two series. The first—labeled with *bias with hysteresis*—includes both bias and hysteresis bits of the 2bcgskew components in the RHSP predictor, while *bias without hysteresis* only includes the bias bits. Both also include the 2bcgskew prediction bit. Worst case scenarios are identified as (i) only the state of the meta predictor is provided to the RHSP predictor and (ii) a predictor without connections between 2bcgskew and RHSP, i.e. a pure RHSP. The most important outcome bit is the 2bcgskew prediction bit and the egskew outcome is useful if none of the component states (i.e., the bias and/or hysteresis bits) are transferred to the RHSP predictor. To conclude, the internal state of the two gshares and the bimodal components are equally important, but the meta predictor is not important.

- Minor variations are measured by varying the global history length used in the 2bcgskew.
- Skewing, which has no impact on predictor size, is of paramount importance.

## 4.3 Scalability

Figure 5 illustrates the scalability of the 2FAR predictor to other bit budgets. As with all predictor designs, the amount of available bits to store information has a major impact on its performance.

| Program     | Ref Input     | Static Branches |  |  |
|-------------|---------------|-----------------|--|--|
| 164.gzip    | random        | 150             |  |  |
| 175.vpr     | route         | 598             |  |  |
| 176.gcc     | scilab        | $14,\!880$      |  |  |
| 181.mcf     | ref           | 346             |  |  |
| 186.crafty  | ref           | 1,396           |  |  |
| 197.parser  | ref           | 1,974           |  |  |
| 252.eon     | kajiya        | 281             |  |  |
| 253.perlbmk | splitmail 535 | 2,055           |  |  |
| 254.gap     | ref           | 566             |  |  |
| 255.vortex  | lendian2      | $1,\!603$       |  |  |
| 256.bzip2   | program       | 339             |  |  |
| 300.twolf   | ref           | 569             |  |  |

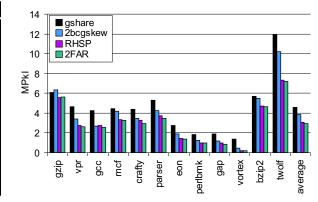


Figure 6: Left: SPEC INT 2000 benchmarks, their inputs, and the number of static branches executed. Right: The 2FAR predictor evaluated on the SPEC INT 2000 traces.

The smaller 2FAR predictor (32K bits) also spends half its budget on 2bcgskew, and compared to the baseline, it halves the numbers of entries in the perceptron table down to 16. In addition, it has a global history length of 25 (compared to 40 at 64K bits) and benefits from one extra local history bit.

At 128K bits, the predictor design scales to a global history of 48 bits. It keeps the 6 bits of fully redundant local history from the baseline, but seriously increases the entries in its local history table to 512 entries. For a bit budget of 768K bits we measure 2.26 MPkI with a 2FAR configuration that additionally feeds the egskew outcome into the fusion predictor.

# 4.4 SPEC INT 2000

In this section we evaluate the branch predictor on traces which are commonly used in branch prediction studies, i.e. exclusive evaluation on the SPEC CPU 2000 integer benchmark suite. For each of the benchmarks shown in Figure 6, we skip the first 50M conditional branches and take statistics for the following 250M conditional branches. The resulting traces are much longer than the CBP traces.

The results in Figure 6 show that the overall conclusion drawn from the CBP-traces— 40% improvement over gshare—holds for the longer SPEC INT 2000 traces, with 2FAR performing 35% better than gshare on average. For completeness, we also show the individual results for the 2bcgskew and the pure skewed perceptron predictor from section 4.1. The 2FAR predictor outperforms the RHSP, although not by much, on all traces, except for gzip.

# 5. 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 standalone versions. Our 2bcgskew predictor fused by an alloyed redundant history skewed perceptron predictor, which we call 2FAR, uses multiple predictions to obtain a more accurate overall 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 some local history, the global history, and pseudo-tags these predictions serve as input to our fusion predictor, which is an alloyed redundant history skewed perceptron predictor.

The presented baseline 2FAR branch predictor implementation needs exactly (64K + 192) bits and reaches 3.15 MPkI, which is significantly better than 2bcgskew and RHSP. In addition, we examined the sensitivity of the proposed 2FAR predictor to its various design parameters. Finally, we demonstrate how the 2FAR design scales to a wide range of predictor sizes, and prove its robustness to longer traces taken from SPEC INT 2000 suite, which is frequently used in branch prediction studies.

# Acknowledgements

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. The authors are indebted to Toon De Pessemier for his programming work.

# References

- M. Evers, P.-Y. Chang, and Y. 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*, pp. 3–11, 1996.
- [2] S. McFarling, "Combining branch predictors," Tech. Rep. TN-36, Digital Western Research Laboratory, June 1993.
- [3] A. Seznec, S. Felix, V. Krisnan, and Y. Sazeides, "Design tradeoffs for the Alpha EV8 conditional branch predictor," in *Proceedings of the 29th Annual International Symposium on Computer Architecture*, pp. 295–306, May 2002.
- [4] G. H. Loh and D. S. Henry, "Predicting conditional branches with fusion-based hybrid predictors," in *Proceedings of the 11th International Conference on Parallel Architectures and Compilation Techniques*, pp. 165–176, Sept. 2002.
- [5] D. A. Jiménez, "Neural methods for dynamic branch prediction," ACM Transactions on Computer Systems, vol. 20, pp. 369–397, Nov. 2002.
- [6] 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*, pp. 292–303, June 1997.
- [7] A. Seznec, "An optimized 2bcgskew branch predictor," Sept. 2003.
- [8] A. Seznec, "Redundant history skewed perceptron predictors: Pushing limits on global history branch predictors," Tech. Rep. PI-1554, Institut de Recherche en Informatique et Systèmes Aléatoires, May 2003.
- [9] J. A. Fisher and S. M. Freudenberger, "Predicting conditional branch directions from previous runs of a program," in *Proceedings of the 5th International Conference on Architectural Support* for Programming Languages and Operating Systems, pp. 85–95, Oct. 1992.