Understanding Prediction Limits Through Unbiased Branches

Lucian Vintan[1], Arpad Gellert[1], Adrian Florea[1], Marius Oancea[1] and Colin Egan[2]
[1] "Lucian Blaga" University of Sibiu, Computer Science Department, Emil Cioran Street, No. 4, 550025 Sibiu, Romania,
{lucian.vintan, arpad.gellert, adrian.florea, marius oncea}@ulbsibiu.ro
[2] University of Hertfordshire, School of Computer Science, Hatfield, College Lane, AL10 9AB UK,,
c.egan@herts.ac.uk

Abstract The majority of currently available branch predictors base their prediction accuracy on the previous *k* branch outcomes. Such predictors sustain high prediction accuracy but they do not consider the impact of unbiased branches which are difficult-to-predict. In this paper, we quantify and evaluate the impact of unbiased branches and show that any gain in prediction accuracy is proportional to the frequency of unbiased branches. By using the SPECcpu2000 integer benchmarks we show that there are a significant proportion of unbiased branches which severely impact on prediction accuracy (averaging between 6% and 24% depending on the prediction context used).

1. Introduction

Branch instructions are a major bottleneck in the exploitation of instruction level parallelism (ILP) since (in general-purpose code) conditional branches occur approximately every 5 – 8 instructions [5]. With increasing instruction issue rate and depth of the pipeline, accurate dynamic branch prediction becomes more essential. Very high prediction accuracy is required because an increasing number of instructions are lost before a branch misprediction can be corrected. Even a 3% misprediction rate can have a severe impact on MII processor performance [1, 10].

Chang [2] introduced the idea of grouping branches by their bias in an attempt to reduce the impact of aliasing. By profiling, branches were classified between 6 static classes and were then guided to the most appropriate dynamic predictor. Chappell [3] investigated difficult-to-predict branches in a Simultaneous Subordinate Micro-Threading (SSMT) architecture. Chappell constrained microthreading to only difficult-to-predict branches which were identified as those being reached along a 'difficult-path'. We believe that such branches are unbiased. More recently, Desmet [4] applied the concept of *Gini-index* to construct a decision tree based on a number of dynamic and static branch features. In line with our thoughts, Desmet concluded that accurate branch predictors require more than just the type of predictor and history register length to achieve accurate branch prediction.

Alternative methods of dynamic branch prediction are available such as neural branch prediction [6, 15]. Despite a neural branch predictor's ability to achieve a very

high prediction rate and the ability to exploit deeper correlations at linear costs, the associated design complexity due to latency, large quantity of adder circuits, area and power are still obstacles to industrial adoption. As such we therefore consider neural prediction techniques to be outside the scope of this paper.

The main objective of this paper is to highlight the impact of unbiased branches so that they can be considered in the design of two-level predictors. In remainder of this paper we evaluate the impact of unbiased branches, and therefore difficult-to-predict branches, on three commonly used prediction contexts (local, global and global XOR branch address) and their corresponding two-level predictors [8, 9, 10, 13].

2. Identifying difficult-to-predict branches

The majority of branches demonstrate a bias to either the taken or the not-taken path which means branches are highly polarised towards a specific prediction context (a local prediction context, a global prediction context or a path-based prediction context) and such polarised branches are relatively easy-to-predict. However, a minority of branches show a low degree of polarisation since they tend to shuffle between taken and not-taken and are therefore unbiased and difficult-to-predict.

In this study, we identify unbiased branches by cascading branches through the three different prediction contexts and their respective predictors: a PAg, a GAg and a Gshare predictor. We also increase the history register lengths in units of 4-bits from 16-bits to 28-bits as shown in Figure 1. Within our prediction contexts, a feature is the binary context on p bits of prediction information. Finally, each static branch has associated k dynamic contexts in which it can appear ($k \le 2^p$). We define the polarisation index (P) of a certain dynamic branch context as equation (1):

$$P(S_i) = \max(f_0,\ f_1) = \begin{cases} f_0, & f_0 \ge 0.5 \\ f_1, & f_0 < 0.5 \end{cases} \qquad (\mathbf{1})$$

where:

- $S = \{S_1, S_2, ..., S_k\}$ = the set of prediction contexts that appear during all branches instances;
- k = the number of distinct prediction contexts that appear during the branch's execution instances, $k \le 2^p$, and p is the history register length;
- $f_0 = \frac{T}{T+NT}$, $f_1 = \frac{NT}{T+NT}$, NT = the number of "not-taken" branch instances corresponding to context S_i, T = the number of "taken" branch instances corresponding to context S_i, $(\forall) i = 1, 2, ..., k$, and therefore $f_0 + f_1 = 1$;
- if $P(S_i) = 1, (\forall) i = 1, 2, ..., k$, then the context S_i is completely biased (100%) and the branch is highly predictable;
- if $P(S_i) = 0.5, (\forall) i = 1, 2, ..., k$, then the context S_i is totally unbiased and the branch might be difficult to predict.

Consider the following trivial examples, a branch in a certain dynamic context shows the following behaviour: TTTTTT… or NNNNNN… in which case the

transitions are always taken or always not-taken, and would be biased and easy-to-predict. However, a branch in a certain context that is stochastic will show a highly shuffled behaviour, which would result in the branch being unbiased and difficult-to-predict with its transitions toggling between T and N. We therefore consider that the rate of transition between branch outcomes is an important feature that can be applied to branch prediction. We introduce the distribution index which is the based on the rate of transitions as shown by equation (2):

$$D(S_i) = \begin{cases} 0, & n_t = 0 \\ \dfrac{n_t}{2 \cdot \min(NT, T)}, & n_t > 0 \end{cases} \quad (2)$$

where:

- n_t = the number of branch outcome transitions in context S_i;
- $2 \cdot \min(NT, T)$ = the maximum number of possible transitions;
- k = the number of distinct dynamic contexts, $k \leq 2^p$, and p is the history register length;
- if $D(S_i) = 1, (\forall) i = 1, 2, ..., k$, then the behaviour of the branch in context S_i is "contradictory" (the most unfavourable case), and the predictor cannot learn it;
- if $D(S_i) = 0, (\forall) i = 1, 2, ..., k$, then the behaviour of the branch in context S_i is constant (the most favourable case), and the predictor can be learned.

A branch with a low distribution index (tending to 0) will show a repeating pattern and there will be few transitional changes. In contrast, a branch that exhibits many transitional changes will show a shuffled pattern and will have a high distribution index (tending to 1). Hence, the greater the distribution index means that the branch becomes more difficult-to-predict in a given predictor. We consider any branch for a given prediction context that has a distribution index of ≤0.2 to be easy-to-predict and define a difficult-to-predict branch for a given prediction context to be a branch with a low polarisation index (P<0.95 as derived from equation 1 (an unbiased branch)) and with a distribution index of >0.2. We chose this value because for a given branch context with a polarisation index >0.95 will be easy-to-predict and will achieve a high prediction accuracy. Consequently branches with a polarisation index of <0.95 will be difficult-to-predict.

We identify and reduce the number of unbiased branches (Figure 1) by passing unbiased branches through successive cascades of different prediction contexts with increasing history information (from 16- to 28-bits).

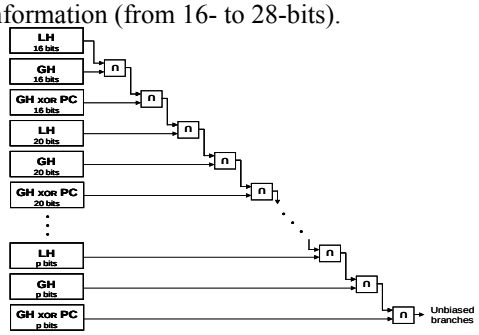

Figure 1. Unbiased branches cascading through the prediction contexts

The number of unbiased branches is reduced from one prediction context to the next because an unbiased branch in one prediction context is not necessarily unbiased in a different prediction context. By the time our final prediction context (28-global history bits XORed with 28-bits of the branch address) is iterated the only remaining unbiased branches are those that have been unbiased throughout all iterations of all of the previous prediction contexts and these remaining unbiased branches are therefore identified as difficult-to-predict. We therefore predict with a short history prediction context before a long history prediction context to remove biased branches (those that are easy-to-predict) as early as possible.

3. Simulations

In this study we identify unbiased branches in the SPEC2000 benchmark suite [12] by cascading branches through the three different prediction contexts and their respective predictors: a PAg predictor, a GAg predictor and a Gshare predictor. We use the SimpleScalar simulator [11] and all results are reported on 1 billion dynamically executed instructions, skipping the first 300 million instructions.

Figure 2 shows the prediction accuracy achieved by the 16-local history bit prediction context using the PAg predictor. The average prediction accuracy of this local prediction context is around 91%, which is limited by the impact of the unbiased branches which have an average prediction accuracy of around 76%. The frequency of unbiased branches (Table 1) varies between 5.76% (*mcf*) and 44.98% (*twolf*) with an average of 24.55%.

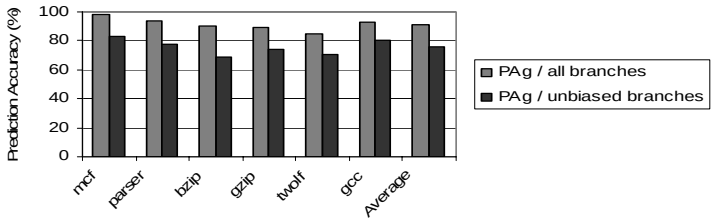

Figure 2. PAg prediction accuracy with the 16-local history bits prediction context

Table 1. Percentage of unbiased branches (16-local history bits prediction context)

Benchmark	*mcf*	*parser*	*bzip*	*gzip*	*twolf*	*gcc*	Avg.
Unbiased branches (P<0.95)	5.76%	20.60%	26.42%	38.73%	44.98%	10.80%	24.55%

The unbiased branches are now cascaded through the 16-global history bit prediction context and its corresponding GAg predictor. The average prediction accuracy of this global prediction context is around 93%, which again is limited by the impact of the unbiased branches. The average prediction of accuracy unbiased branches is around 72% and the frequency of these unbiased branches (Table 2) varies between 3.28% (*mcf*) and 32.41% (*twolf*) with an average of 17.48%.

Table 2. Percentage of unbiased branches (16-global history bits prediction context)

Benchmarks	*mcf*	*parser*	*bzip*	*gzip*	*twolf*	*gcc*	Avg.
Unbiased branches (P<0.95)	3.28%	12.95%	23.4%	28.89%	32.41%	3.92%	17.48%

The remaining unbiased branches are now cascaded through the 16-global history prediction context XORed with 16-bits of the branch address and its associated

Gshare predictor. The prediction accuracy of the 16-history bit Gshare predictor improved by around 1% in comparison with the 16-history bit GAg predictor. However, the number of unbiased branches remained the same as those of the GAg predictor apert from *gcc* which showed a marginal reduction to 3.91%.

The history register length was increased by 4-bits to 20-bits and then the remaining unbiased branches were cascaded through the 20-local history bit prediction context and its associated PAg predictor. We continued to cascade through our remaining prediction contexts (local, global, global XOR branch address), increasing the amount of history information by 4-bits at a time (to a maximum of 28-history bits) as shown by Figure 1 thereby gradually reducing the number of unbiased branches through each context and decreasing the number of unbiased branches.

A distribution index tends to 0 for a branch that is not shuffled (and is easy-to-predict) and tends to 1 (and is difficult-to-predict) for a shuffled branch. We partitioned the percentage of branches into 5 distribution index intervals: (0.0 - 0.2), (0.2 - 0.4), (0.4 - 0.6), (0.6 - 0.8) and (0.8 - 1.0).

Figure 3 shows the intervals for the 16-local history bit context and that around 41% of the unbiased branches have a distribution index interval (0.4 - 0.6), making their branch behaviour relatively shuffled and around 21% of the branches have a distribution index between (0.8 - 1.0), making their behaviour highly shuffled and therefore difficult-to-predict. Using the cut-off distribution index of <0.2, our simulations show that only around 16% of the unbiased branches are easy-to-predict with the 16-history bit local prediction context. *Gcc* has the greatest percentage of unbiased and easy-to-predict branches (around 39%) and *gzip* the greatest percentage of unbiased and difficult-to-predict branches (around 30%).

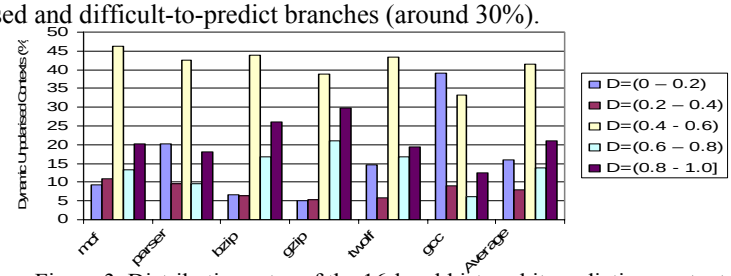

Figure 3. Distribution rates of the 16-local history bit prediction context

Similarly, we determined the intervals for the 16-global history bit prediction context and the intervals for the 16-global history bit XORed with 16-bits of the branch address prediction context (intervals were the same for both contexts). Since both of these are global prediction contexts, it is not surprising that their distribution indices are similar. With the 16-history bit global prediction contexts only around 3% of the unbiased branches have a distribution index <0.2, around 33% have a distribution index of (0.4 - 0.6), and around 28% have a distribution index of (0.8 – 1). As with the local prediction context, *gcc* has the greatest percentage of unbiased but easy-to-predict branches (around 8%), but *twolf* has the greatest percentage of unbiased and difficult-to-predict branches (around 39%).

Figure 4 shows the reduction in the number of unbiased branches as they cascade through the three prediction contexts. The percentage reduction in the number of unbiased branches decreases from around 25% to around 6%. We consider that this value of 6% is still too high and further investigations are required.

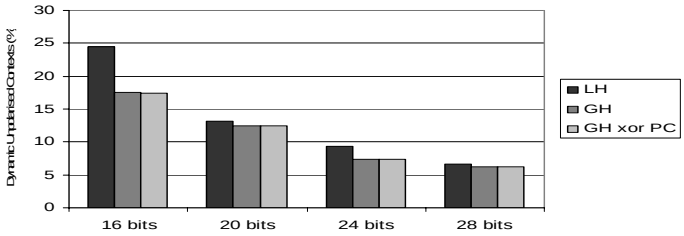

Figure 4. Reducing the number of unbiased branches with increasing history register length

In an earlier paper [14], we explored the benefits of adding sufficient information, in the form of successive branch addresses, to uniquely identify each program path. We continue that work in this study evaluating, on all branches, paths of different lengths (*p* branches) used together with global histories of the same length (*p* bits). The results are presented in Figure 5, where they are compared with the results obtained using only global history prediction context.

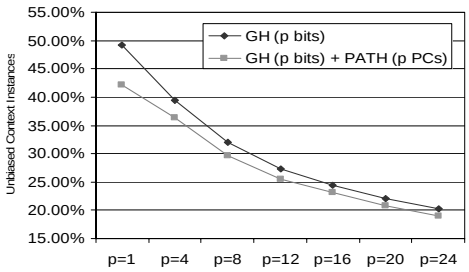

Figure 5. The gain introduced by the path for different context lengths

Our simulations show that the best gain is achieved with short history lengths ($p<16$) and there is only marginal gain with longer history lengths, meaning that long global history (*p* bits) approximates very well the longer path information (*p* branches).

We have also undertaken similar simulations with neural predictors [6, 15] and in addition to the SPEC benchmarks we have used the Championship Branch Prediction benchmarks [7, 16]. However, due to space limitations we have not shown the results of these simulations in this paper.

4. Conclusions and Further Work

In this paper, we have shown that the design of branch predictors should consider the identification of difficult-to-predict branches. Different branches exhibit different behaviours for given prediction contexts and predictors, and the amount of shuffling impacts on prediction accuracy. Even after cascading branches through a series of prediction contexts there remains a significant number of difficult-to-predict branches and the frequency of these difficult-to-predict branches varies between different programs and between different prediction contexts. Computer Architects cannot therefore continue to expect a prediction accuracy improvement with conventional predictors and alternative approaches are necessary. We have briefly investigated the

use of increased correlation information by recording path information as well as history information and have shown that some gain can be obtained with short history register lengths (<16), but path information with longer history register lengths only achieves marginal gain.

This work demonstrates that current branch predictors use limited prediction contexts (local, global correlation and path information) due to the degree of polarisation. We have therefore shown that the use of more prediction contexts is required to further improve prediction accuracies. Our current thoughts are to use a particularly relevant "piece" of the dynamic CPU context or alternatively some HLL code information. In order to efficiently use such information we consider it will be necessary to have a significant amount of compiler support.

References

1. Burger, D. and Goodman, J. R.: Billion Transistor Architectures. IEEE Computer. September 1997, 46 – 49.
2. Chang P.-Y., Hao E., Yeh T.-Y., Patt Y. N.: Branch Classification: a New Mechanism for Improving Branch Predictor Performance, Proceedings of the 27th International Symposium on Microarchitecture, San Jose, California, (1994).
3. Chappell R., Tseng F., Yoaz A., Patt Y.: Difficult-Path Branch Prediction Using Subordinate Microthreads, The 29th Annual International Symposia on Computer Architecture, Alaska, USA, (May 2002).
4. Desmet V., Eeckhout L., De Bosschere K.: Evaluation of the Gini-index for Studying Branch Prediction Features. Proceedings of the 6th International Conference on Computing Anticipatory Systems (CASYS). American Institute of Physics. AIP Conference Proceedings. Vol. 718. (2004) 376-384.
5. Hennessy J. and Patterson D.: Computer Architecture: A Quantitative Approach, Third Edition, Morgan Kaufmann Publishers, (2003).
6. Jiménez D. A., Lin C.: Dynamic Branch Prediction with Perceptrons, Proceedings of the 7th International Symposium on High Performance Computer Architecture, (January 2001).
7. Loh G. H.: Simulation Differences Between Academia and Industry: A Branch Prediction Case Study, International Symposium on Performance Analysis of Software and Systems (ISPASS), pp.21-31, Austin, TX, USA, March 2005.
8. McFarling S.: Combining Branch Predictors, WRL Technical Note TN-36, Digital Equipment Corporation, (June 1993).
9. Pan S. T., So K. and Rahmeh J. T.: Improving the accuracy of dynamic branch prediction using branch correlation. Proceedings of ASPLOS V, Boston, MA, October (1992) 76-84.
10. Patt, Y. N., Patel, S. J., Friendly, D. H. and Stark, J.: One Billion Transistors, One Uniprocessor, One Chip. IEEE Computer 1 (September 1997) 51–57.
11. Simplescalar The SimpleSim Tool Set, ftp://ftp.cs.wisc.edu/pub/sohi/Code/simplescalar.
12. SPEC, The SPEC benchmark programs, http://www.spec.org.
13. Yeh T. Y. and Patt Y. N.: Two-level adaptive branch prediction. In Proceedings of the 24-the ACM/IEEE International Symposium on Microarchitecture, (November 1991).
14. Vintan L. and Egan C.: Extending Correlation in Branch Prediction Schemes, International Euromicro'99 Conference, Italy, (September 1999).
15 Vintan L., Iridon M.: Towards a High Performance Neural Branch Predictor, International Joint Conference on Neural Networks, Washington DC, USA, July 1999.
16. The 1st JILP Championship Branch Prediction Competition (CBP-1). http://www.jilp.org/cbp, 2004.