INVESTIGATING NEW BRANCH PREDICTORS THROUGH QUANTITATIVE APPROACHES

Lucian N. VINTAN, Adrian FLOREA

University "L. Blaga", Department of Computer Science, Str. E. Cioran, No. 4, Sibiu-2400, ROMANIA,
Tel./Fax: ++40-69-212716, E-mail: vintan@cs.sibiu.ro,

Abstract: During this work we investigated through a trace driven simulation method two distinct approaches in branch prediction. In our first approach the main idea is that the most predictable branches to be predicted based on a Static Prediction Table and the other branches – most difficult predictable **-** to be predicted based on a classical Two Level Adaptive Branch Prediction Scheme. The obtained results are quite encouraging because the proposed hybrid predictor is less complex and obtains a similar performance compared to a classical dynamic predictor. Also we investigated through the same trace driven simulation method some Markov – PPM branch predictors, based on the general PPM (Prediction by Partial matching) algorithm. We studied in this context the influence of the HRg's (Global History Register's) length and we developed a comparison between a Markov predictor and a classical one. Also we compared a complete PPM predictor with a simplified one, most feasible to be implemented in hardware and the obtained prediction accuracies are quite identical. Therefore we conclude that similar simplified PPM predictors could be a better performance/cost alternative to the classical Two Level Adaptive Branch Predictor.

Key Words: Multiple Instruction Issue, Branch Prediction, Prediction by Partial Matching, Trace Driven Simulation

1. INTRODUCTION

As the average instruction issue rate and depth of the pipeline in multiple instruction issue (MII) processors increase, the necessity of an efficient hardware branch predictor becomes essential. Very high prediction accuracies are necessary, because taking into account the MII processors characteristics as pipeline depth or issue rates, even a prediction miss rate of a few percent involves a substantial performance loss.

The first efficient approach in hardware (dynamic) branch prediction consists in Branch Target Buffer (BTB) structures [Per93]. BTB is a small associative memory, integrated on chip, that retains the addresses of recently executed branches, their targets and optionally other information (e.g. target opcode). Due to some intrinsic limitations, BTB's accuracies are limited on some benchmarks having unpropitious characteristics (e.g. correlated branches).

In order to improve BTB's efficiency, Yeh and Patt (1992) and independently Pan et al (1992) generalised it through a new approach called Two Level Adaptive Branch Prediction. According to [Yeh92], the Two Level Adaptive Branch Prediction uses two distinct levels of branch history information to make predictions. The first level consists in the History Register (HR), that contains the last k branches encountered (taken/ not taken) or the last k occurrences of the same branch instruction. The second level consists in the branch behaviour of the last l occurrences of the specific pattern of these branches. It is implemented by a Pattern History Table (PHT), that contains essentially the branch prediction automaton (usually 2 bit saturating counters).

HR shifts left with a binary position when updated according to the actual branch behaviour (taken=1/ not taken=0). There is a corresponding entry in the PHT for each of the 2^k HR's patterns. The prediction of the branch (P) is a function (f) of the actual prediction automaton state S_t.

$$\mathbf{P = f(S_t)} \tag{1}$$

After the branch is resolved, HR is shifted left and the prediction automaton state becomes S_{t+1}.

$$\mathbf{S_{t+1} = g(S_t ,B_t)} \tag{2}$$

where g represents the automaton's transition function and B_t represents the behaviour of the last branch encountered (taken/ not taken). A lot of interesting implementations of these correlated branch prediction schemes are known [Yeh92, Pan92, Ega98].

These Two Level Adaptive Branch Prediction schemes are very effective in predicting correlated branches with high accuracy. It's well known that the average prediction rate for these schemes, measured on nine of the ten Spec benchmarks, is about 97%, while BTB schemes achieved at most 94% on the same benchmarks [Yeh92]. An interesting generalisation of these Two Level Adaptive Branch Prediction schemes, based on the universal compression/prediction algorithm called "prediction by partial matching", is given in [Mud96].

Our simulation work has been centered on the Stanford integer benchmark suite, a collection of eight C programs designed by Professor John Hennessy (Stanford University), to be representative of non - numeric code while at the same time being compact. The benchmarks are computationally intensive with higher dynamic instruction counts. All these benchmarks were compiled by the HSA gnu C compiler which targets the HSA instruction set. A dedicated HSA simulator [Ste96] that generates the corresponding traces simulated the resulted HSA object code. Some characteristics of the used traces are given in Table 1.

Benchmark	Total instr.	% Branches (%Taken)	Short description
Puzzle	804.620	25(91)	Solves a cube packing problem
Bubble	206.035	20(75)	Bubble sorts an array
Matrix	231.814	9(97)	Matrix multiplication
Permute	355.643	15(80)	Recursive computation of permutations
Queens	206.420	19(50)	Solves the eight queens problem
Sort	72.101	17(65)	Quick sorts a randomised array
Towers	251.149	15(76)	Solves Towers of Hanoi problem (recursive)
Tree	136.040	24(73)	Performs a binary tree sort

Table 1. Characteristics of the HSA traces

The average instructions number is about 273.000 and the average percentage of total instructions that are branches is about 18%, with about 76% of them being taken. Derived from HSA traces, special traces were obtained, containing exclusively all the processed branches. Each branch belonging to these modified HSA traces is stored in the following format: branch's type, the PC of the branch and it's target address.

2. COMBINING STATIC AND DYNAMIC BRANCH PREDICTOR

In this paragraph we tried to combine into a new prediction scheme two distinct branch prediction principles: static and respectively dynamic branch prediction. Firstly we developed a statistic for each benchmark in order to learn how many time a static branch is taken and respectively not taken. As an example, below it's presented a very short fragment belonging to these laborious statistics (derived from *Tower* benchmark). As it can be seen, branch at PC (Program Counter)=64 was 4021 times taken (T) and only 132 times not taken (NT), therefore having a strong polarised behaviour (predictable branch). In contrast, the next static branch at PC=65 seems to be most hardly to be static predicted because it was 2190 times not taken and 4021 times taken. Based on these profiling information we proposed to implement a Static Prediction Table (SPT), addressed in parallel with the Dynamic Prediction Table implemented as a modified GAg (Global History Register - HR and Global Pattern History Table) scheme, like in Figure 1 [Yeh92]. If the prediction is found in SPT the branch is accordingly predicted, if not, it will be predicted dynamically based on the GAg scheme (through one classical 2 bit saturating counter).

PC=64	NT	T
4153	132 (3.18%)	4021 (96.82%)
PC=65		
6211	2190 (35.26%)	4021 (64.74%)

Of course, a SPT entry will contain principally a tag, a static prediction bit (taken/not taken) and the target address. SPT will be addressed by the branch's PC and obviously it will be loaded based on the profiling information obtained after running the benchmark (program). In our experiments we used small SPT tables having only 12 entries and loaded with the most polarised branches (being taken or not taken over 85%). Therefore the main idea is that the most predictable branches to be predicted based on the SPT table and the "lower polarised" branches – probably less predictable - to be predicted based on a classical Two Level Adaptive Branch Prediction Scheme (GAg here).

Figure 2 presents obtained prediction accuracies for a hybrid prediction scheme having a 12 entries SPT table and a 32 entries GAg table (see also Figure 1) vs. a GAg scheme having a Prediction History Table (PHT) of 64 entries. At average through the hybrid scheme it was obtained a prediction accuracy of 77.21% (including correct targets) and for the pure dynamic scheme an accuracy of 77.32%, practically the same performance for both schemes. This is very encouraging from our point of view taking into account that the hybrid scheme is much simpler and also cheaper comparing to the "equivalent" dynamic scheme. Therefore we consider that putting together static and dynamic schemes could be a feasible solution, in special related to cheaper microprocessors. As our laborious simulation experiments point out, similar hybrid branch predictors could be even better than dynamic branch predictors, at the same cost.

3. A MARKOVIAN - PPM BRANCH PREDICTOR

During this section we investigated through the same trace driven simulation method a Markovian branch prediction scheme, first introduced by Trevor Mudge [Mud96]. The prediction is based here on the PPM (Prediction by Partial Matching) algorithm that represents an universal compression/prediction algorithm. PPM has been theoretically proven optimal in data compression and prefetching and also in some speech recognition problems. The bases of the PPM algorithm of order m are a set of (m+1) Markov predictors. A Markov predictor of order j predicts the next bit based upon the j immediately preceding bits pattern (a simple Markov chain). More precisely, the prediction process counts every time when that pattern on j bits was found, if it was followed by a '1' or respectively by a '0'. The prediction is according to the most frequent bit that follows the searched and founded pattern. The prediction algorithm is illustrated in Figure 3. PPM uses the m immediately preceding bits to search a pattern in the highest order Markov model, in this case m. If the search succeeds, which means the pattern appears in the input sequence seen so far, PPM predicts the next bit using this m-th order Markov predictor. However, if the pattern is not found, PPM uses the (m-1) immediately preceding bits to search the next lower order (m-1)-th order Markov predictor. Whenever a search misses, PPM reduces the pattern by one bit and uses it to search in the next lower order Markov predictor. This process continues until a match is found and the corresponding prediction can be made.

If we consider the search pattern for every Markov predictor as a HRg (Global History Register) pattern, it is proved [Mud96] that the complete PPM predictor can be viewed as a set of Two Level Predictors, having not one size of HRg but a set that spans m down to 0. It's obviously that this PPM branch predictor generalise the Two Level Predictors. To implementing this Markov predictor in hardware would require a doubling of the hardware, straining the limits of practicality. Trying to avoid this circuit complexity, we investigate – for the first time according to our knowledge – a simplified PPM branch predictor consisted by only a Markov predictor of order m, quite feasible to be implemented in hardware. In this case, if the pattern (HRg content, on m bits) is not found in the global history string (taken/not taken), automatically the branch will be predicted as no taken. Figure 4 presents the influence of the HRg's length (10 to 40 bits) on the prediction accuracy offered by this simplified Markov predictor. The global history string was considered here on 300 bits. More interesting, Figure 5 shows clearly that a PPM branch predictor outperforms an equivalent Two Level Branch Predictor (GAg – having a HRg on k=5 bits like in Figure 1, equal with the Markov predictor's order), results placed in a perfect concordance with those published by other researchers [Mud96]. Table 2 presents an interesting comparison between a complete PPM predictor and our proposed simplified PPM predictor, earlier described. Surprisingly, in both cases the obtained prediction accuracies are quite similar that shows clearly that a simplified PPM predictor outperforms a classical scheme and also is feasible to be implemented in hardware. We strongly believe that some similar simplified PPM predictors or our recently proposed neural branch predictors [Vin99a,b], could be original ideas in finding new efficient branch prediction approaches.

4. CONCLUSIONS AND FURTHER WORK

Firstly, we propose a new branch prediction scheme that putts working together a static and a dynamic branch predictor. At a lower cost, this hybrid predictor obtained practically the same prediction accuracy like a Two Level Adaptive Branch Predictor. This is very encouraging from our point of view taking into account that the hybrid scheme is much simpler and cheaper comparing to the "equivalent" dynamic scheme. Therefore we consider that putting together static and dynamic schemes could be a feasible solution, in special related to cheaper microprocessors.

Secondly, we investigated through the same trace driven simulation method some Markov – PPM branch predictors, based on the general PPM algorithm. We studied in this context the influence of the HRg's length and we developed a comparison between a Markov predictor and a classical one. Also we compared a complete PPM predictor with a simplified one, most feasible to be implemented in hardware and the obtained prediction accuracies are quite similar for both prediction structures. We conclude that using general prediction algorithms like PPM and - based on these mature approaches - implementing some simplified hardware schemes, could be a good alternative to the well known Two Level Adaptive Predictors.

Because of the additional warm up time our models are likely to perform less successfully with relatively small benchmarks like Stanford. Anyway, the obtained prediction accuracies are in a perfect concordance with those obtained by other researchers that used Stanford benchmarks in evaluating branch prediction [Ega98]. We would expect to show some improvement in prediction accuracy using larger benchmarks like Spec '95,98. Unfortunately, from financial reasons, at this moment we aren't able to use Spec or other larger benchmarks.

As a further work we intend to continue to develop and implement in more depth new hybrid branch prediction schemes as an alternative to the complex pure dynamic prediction schemes. Also we'll intend to develop, analyse and optimise through complex simulation some more efficient simplified PPM branch prediction schemes. As a distinct new approach in branch prediction, we'll also try together with Professor's G. B. Steven Research Group in Advanced Computer Architectures (University of Hertfordshire, UK) to enlarge and improve our Neural Branch Predictor [Vin99a], through a joint Royal Society Research Grant.

ACKNOWLEDGMENTS

This work was supported in part by the Romanian Ministry of Research and Technology grant **MCT No. 4086/1998, 1999,** respectively by the Romanian National Council of Academic Research grants **CNCSU No. 391/1998 and CNCSU No. 489/1999.** Our gratitude to **Professor Gordon B. Steven** from the University of Hertfordshire, UK, for providing HSA Stanford traces and for his very useful suggestions, guidance and encouragement related to our Instruction Level Parallel Processors research.

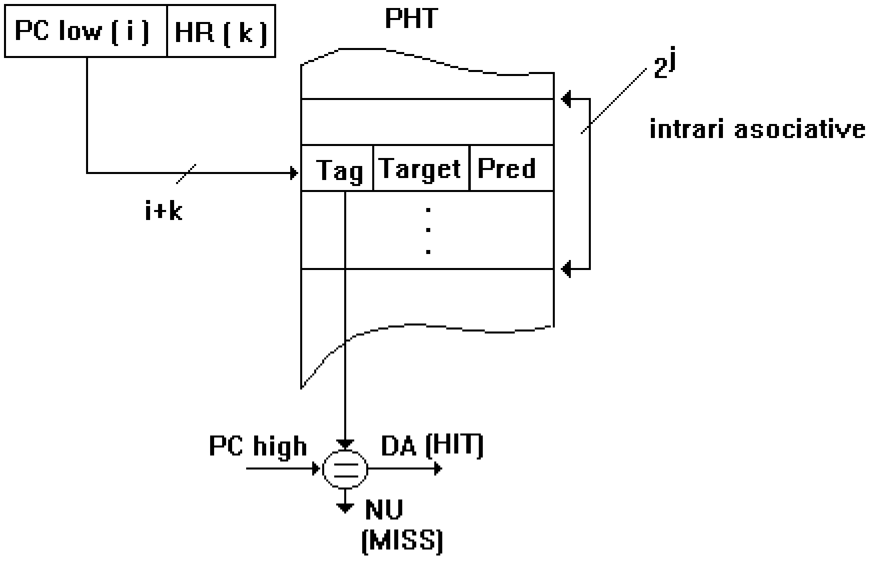

Figure 1. A Modified GAg (Dynamic) Prediction Scheme

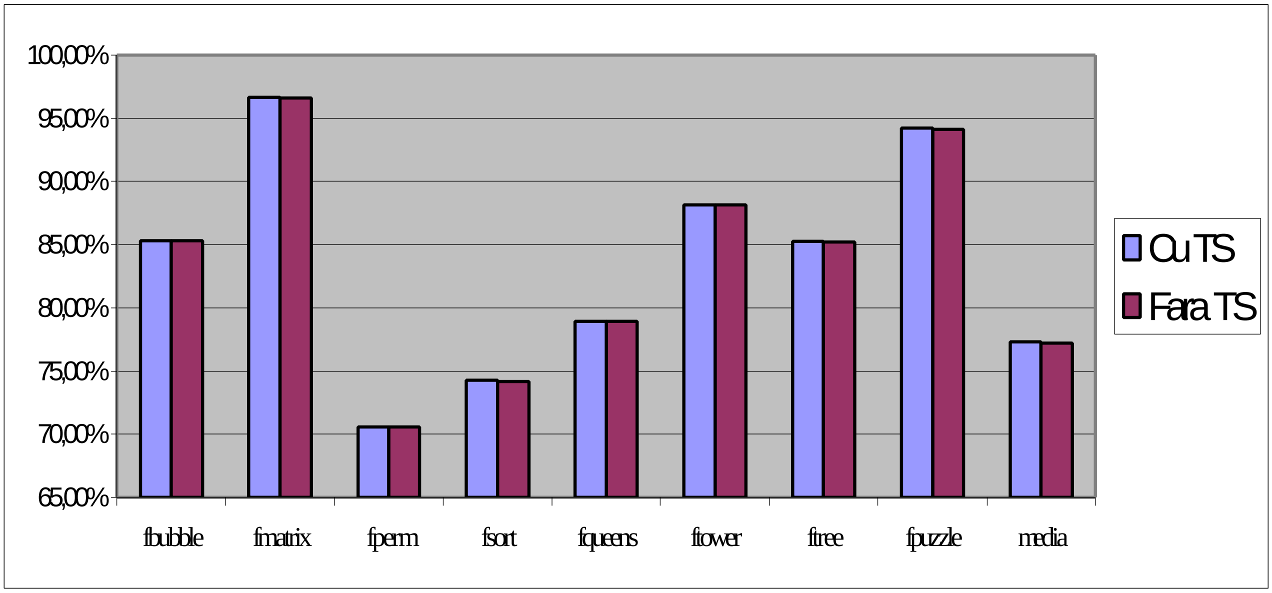

Figure 2. A comparison between dynamic and hybrid (static-dynamic) branch prediction

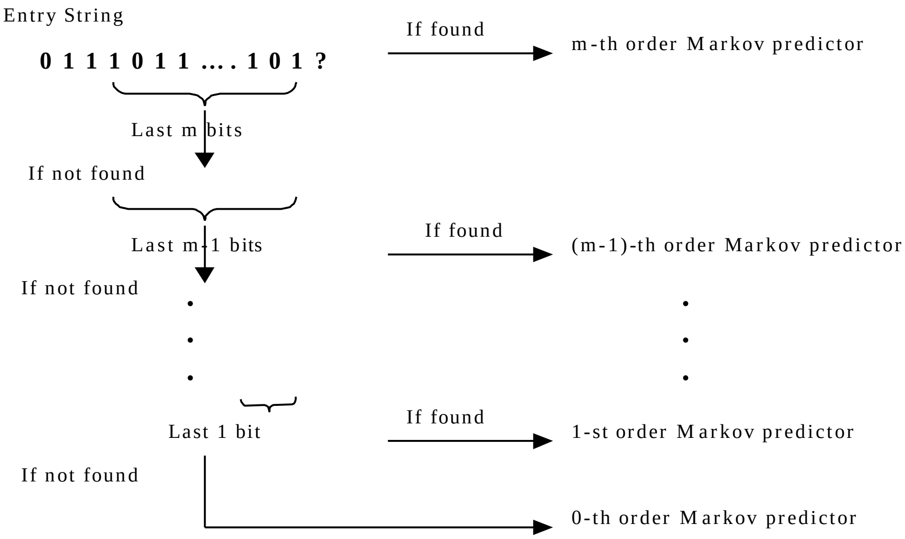

Figure 3. Prediction flowchart of a Markov - PPM predictor of order m

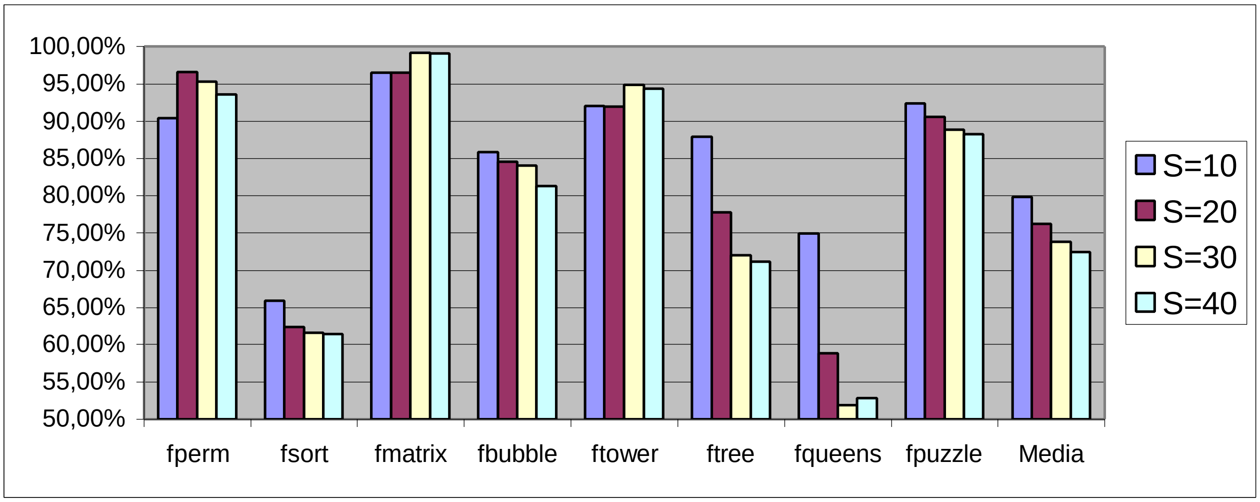

Figure 4. The influence of the search pattern length (HRg) in a Markov branch predictor

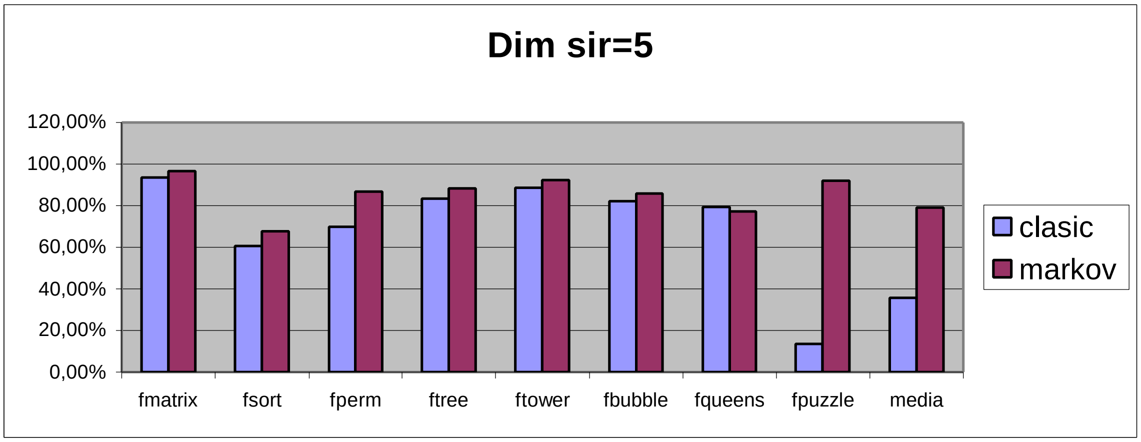

Figure 5. A comparison between a dynamic and an "equivalent" Markov branch predictor

No of bits of entry string	No bits of HRg (Search pattern)	Average Prediction Accuracy	
		Complete PPM Implementation	Simplified PPM Implementation
300	10	79.841%	79.783%
300	20	76.209%	76.063%
300	30	73.888%	73.682%
300	40	72.413%	72.165%
300	50	70.781%	70.498%
300	60	68.278%	67.976%
300	70	66.581%	66.253%
300	80	66.114%	65.760%

Table 2. Complete Markov vs. Simplified Markov branch predictor implementation

REFERENCES

[Cha97] **Chang P.Y., Hao E., Patt Y.N**. - *Target Prediction for Indirect Jumps*, ISCA '97 - Ann. Int.'l Symp. Computer Architecture

[Ega98] **Egan C. -** *Branch Predictor Report*, University of Hertfordshire, Department of Computer Science, UK, November, 1998

[Eve96] **Evers M., Chang P.Y., Patt Y.N**. - *Using Hybrid Branch Predictors to Improve Branch Prediction Accuracy in the Presence of Context Switches*, ISCA '96 (Ann. Int.'l Symp. on Computer Architecture)

[Mud96] **Mudge T.N., Chen I., Coffey J. -** *Limits of Branch prediction*, Technical Report, Electrical Engineering and Computer Science Department, The University of Michigan, Ann Arbor, Michigan, USA, January, 1996

[Pan92] **Pan S.T., So K., Rahmeh J.T. -** *Improving the Accuracy of Dynamic Branch Prediction Using Branch Correlation*, ASPLOS V Conference, Boston, October, 1992

[Per93] **Perleberg C., Smith A. J**. - *Branch Target Buffer Design and Optimization*, IEEE Transactions on Computers, No. 4, 1993

[Ste96] **Steven G. B. et al. -** *A Superscalar Architecture to Exploit Instruction Level Parallelism*, Proceedings of the Euromicro Conference, 2-5 September, Prague, 1996.

[Vin99a] **Vintan L., Steven G.B.** – *A New Branch Prediction Approach Using Neural Networks*, The 6th International Symposium on High Performance Computer Architectures, Toulouse, France, January 2000

[Vin99b] **Vintan L.** – *Predicting Branches through Neural Networks: A LVQ and a MLP Approach*, University "L. Blaga" of Sibiu, Faculty of Engineering, Dept. of Comp. Sc., Technical Report, February, 1999

[Vin99c] **Vintan L., Egan C.** – *Extending Correlation in Branch Prediction Schemes*, International Euromicro Conference, Milano, Italy, 8-10 September, 1999

[Yeh92] **Yeh T., Patt Y. N. -** *Alternative Implementations of Two Level Adaptive Branch Prediction*, 19th Ann. Internationall Symposium on Computer Architecture, 1992.