New Analytical Model of Value Prediction

J. MARKOVSKI, M. GUSEV

Faculty of Natural Sciences and Mathematics,
Ss. Cyril and Methodius University,
Arhimedova b.b., PO Box 162, 1000 Skopje
FYROM

Abstract: - Value prediction is a technique for speculative execution of data dependent instructions based on predicted outcomes of instructions. The predicted outcomes are supplied by a special hardware device called value predictor. Despite the numerous reports on the benefits of value prediction there is still no commercial processor that employs this technique. One limiting factor of the performance potential of value prediction is the value predictor latency. In this paper we extend an existing analytical model of the processor in order to evaluate the effect of value predictor latency. We conclude that the value predictor latency must be reduced and value prediction latency should be overlapped by some other pipeline stages. In addition, value prediction should be made only on instructions with high execution latencies.

Keywords: ILP, value prediction, analytical model, dataflow, critical path

1 Introduction

Value prediction was introduced almost simultaneously in [9] and [3] as powerful means to overcome true data dependences. It relies on the principles of value locality and value predictability which are employed as a way to collapse true data dependences or a means to produce suitable value predictors according to the observed type of value predictability [3]. In both cases, we can define speculative execution based on value prediction as execution of a true data dependent instruction with an empty or partially computed set of input values [3, 6, 14]. This set is filled with predicted values supplied by the value predictor. Value predictor presents a hardware-based mechanism that produces a predicted outcome value of a given instruction.

Speculative execution based on value prediction introduces new classes of values to the processor: (1) actual or final values, (2) predicted values and (3) speculative values [12, 13, 14]. Actual (or final) values present real values produced by the instructions. Predicted values are values obtained by the value predictor. Speculative values are results of those instructions which execution is based on the predicted result. Predicted values are usually obtained before the beginning of instruction execution stage. They can be (1) speculatively forwarded to all data dependent instructions or (2) used to speculatively execute only the predicted instruction [12, 13].

Numerous propositions are made on how to implement techniques based on value locality and value predictability into contemporary processors. Different techniques propose different approaches, but they achieve similar performance improvements of 5-20% percent when simulated in comparable environments [2, 3, 13, 14].

However, if we compare (1) additional hardware complexity and complex misspeculation recovery techniques required to implement effective value prediction and (2) gained performance improvements, it comes to no surprise why there is still no commercial processor that employs value prediction. Still, significant performance potential of value prediction exists and shows tempting results when observed on perfect or almost perfect dataflow machines [3].

2 Limiting factors

Our previous research in [10] overviews current limiting factors of the performance potential of value prediction: (1) low instruction fetch width and finite instruction window [3, 5, 11, 12, 14], (2) imperfect branch prediction [5, 10, 13], (3) delayed update of the value prediction tables and value predictor latency [1, 2, 4, 8, 13, 14, 15] and (4)

multiple simultaneous value predictor accesses [2, 5, 14].

All above mentioned factors are still not resolved completely and pose very low boundaries on the amount of ILP offered by value prediction techniques.

Instruction fetch width of the processor directly influences the performance potential of value prediction [3, 5, 6, 11, 12, 14]. In case of low instruction fetch width the possibility of fetching data dependent instruction in the same cycle is very small. This imposes serialization of data dependent instructions, which in return provides additional time for computation of inputs of the data dependent instruction.

The effect of the finite instruction window is presented as the most devastating factor that suppresses the amount of extractable ILP using value prediction in several studies [3, 5, 6, 11, 14]. The performance potential of value prediction grows as the instruction window size increases until a certain threshold when observed under fixed instruction fetch width. Afterwards there is no additional performance gain while increasing the instruction window size.

The correlation of branch prediction and value prediction is discussed on multiple occasions [5, 11, 14]. One obvious fact is that branch misspeculation and value misprediction can lead to additional value and branch mispredictions which may result in higher misspeculation penalties and decrease of the performance.

After the value predictor supplies the predicted value for some instruction, its state has to be updated. Usually, (1) the history of previously seen values needs to be updated with the latest actual result and (2) the confidence estimator [12] needs to be adjusted depending on the success of the previous value prediction. However, value predictor update may occur late in case of long execution latency of the predicted instruction. In that case, several other dynamic instances of the same static instruction obtain predicted values with the old state of the predictor which in return affects the value prediction accuracy. For example, this phenomenon is observed during execution of a short loop with long execution latency of the instructions like integer division or load cache misses.

A value predictor can receive a request to make a value prediction on result of some instruction during different stages in the pipeline, which imposes the following classification of the value prediction methods: (1) at-fetch value prediction, (2) post-decode value prediction and (3) decoupled value prediction [1]. This classification is important because it is not reasonable to assume that the predicted value will be available immediately, since the value predictor has to access considerably large memory tables [1]. The benefit from the value prediction greatly depends on the point in the pipeline where the value prediction is requested.

Latency of a value predictor is time required to produce a predicted value. It depends on: (1) size of the value predictor, (2) associativity of the value prediction tables and (3) number of ports of the value prediction tables. It varies from less than 2 (2 ports, 2K table entries, associativity 1) up to more than 24 (8 ports, 16K table entries, associativity 16) processor cycles on a contemporary high frequency processor with working frequency of 3.5 GHz [1]. On contrary, very large percent of instructions have consumers within small number of cycles. Particularly, 78-94% of the loads have a consumer within one processor cycle, whereas 73-99% of the results of integer instructions are requested within one cycle [1].

Even more, predicted values are useless unless they are provided before the predicted instruction finishes its execution and the actual result is obtained. This implies that not all instructions benefit from value prediction since instructions which can have predicted results must have execution latency lower than the latency of the value predictor.

Both, delayed value predictor update and value predictor latency introduce stalls in the process of value prediction and contribute to prolonged generation of predicted values [1, 2, 4, 8, 13, 14, 15]. The former enforces serialized value prediction because stale values in the value prediction tables reduce the value prediction accuracy which has considerable impact on the gained performance. The latter prolongs generation of predicted values which in return decreases the time difference between predicted and actual results and diminishes the importance of the made value prediction.

Non-serialized generation of value predictions provides multiple predicted values per static instruction in case its different dynamic instances require value prediction [5]. It requires multiple value predictor accesses both for (1) generation of predicted values and (2) update of value prediction tables [8]. This problem was

anticipated in [3] and further investigated in [2]. Both studies propose interleaved multiple value predictor banks as a possible implementation of the value prediction tables. The maximum number of banks depends on the maximum number of simultaneous accesses supported by the value predictor.

3 Existing analytical models

Each one of these processor features has been extensively studied under experimental conditions using simulators and generic benchmark programs. However, only few attempts have been made to analytically describe a processor that employs value prediction: (1) performance potential of value prediction on a dataflow machine [3], (2) analytical model that estimates expected performance when employing perfect value prediction [14] and (3) fluid stochastic Petri net model that deals with finite processor resources and realistic branch prediction and realistic value prediction [11].

However, none of these models observes the effect of value predictor latency. Our analytical model of the processor extends the model introduced in [3]. We have chosen this model as basis for our research for the following reasons: (1) we want to analyze the effect of value predictor latency since it is identified as one of the most devastating limits even on perfect dataflow machines [1, 2, 3, 8, 13, 15], (2) the model presented in [14] assumes perfect value prediction and actually studies the effect of branch prediction in that environment, (3) the fluid stochastic Petri net model is far to complex since it introduces stochastic variables. However, it is of interest to us since it confirms some experimentally discovered limits on the performance potential of value prediction.

Dataflow graph presentation of a program is given by a directed graph *G(V, S)* as presented on Fig. 1. Each node $v \in V$ represents one instruction and each arc $s \in S$ presents data dependence between two nodes. Critical path C in a given dataflow graph G is the longest path in the dataflow graph from the entry node to the termination node as presented on Fig. 1a.

Dataflow graph is extended in [3] to speculative dataflow graph since in reality it is not possible to correctly predict all instructions. Speculative dataflow graph is a weighted graph where each $s \in S$ is assigned probability p_s to correctly predict the result of the starting node. Critical path in the speculative dataflow graph is defined analogously to a critical path in dataflow graph as presented on Fig. 1b.

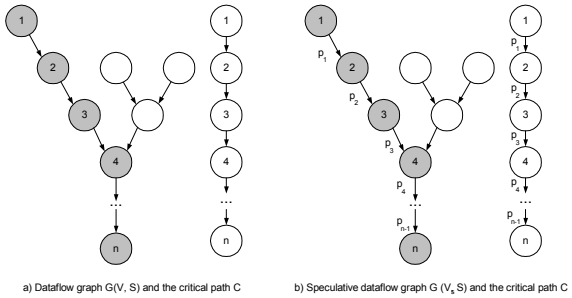

Fig. 1 Dataflow graph vs. Speculative dataflow graph

[3] analyses performance potential of value prediction in dataflow machines using two processor models based on the size of the instruction window: (1) infinite instruction window and (2) limited instruction window. The focus is placed on execution of the critical path because dataflow machines are limited only by the critical path of the dataflow graph. Final expression that estimates the performance potential of value prediction is based on (1) the average value prediction accuracy under assumption that every instruction on the critical path can be value predicted and (2) number of entries in the instruction window for the second model.

Execution of the critical path is observed by using speculative dataflow graph $G_E(V_E, S_E)$ which presents all possible execution sequences, as presented on Fig. 2. All nodes which are executed with speculated values are denoted with subscript *s*.

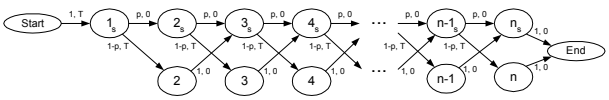

Fig. 2 Execution graph of the critical path

For simplicity, several assumptions are made in [3]: (1) all instructions on the critical path are predicted correctly by some value predictor with the same average probability p, (2) in case of correct value prediction execution time is zero cycles, (3) in case of misprediction, misspeculation penalty is considered to be T, which presents the average execution time of all instructions on the critical path and (4) after the misprediction recovery, execution proceeds with value prediction of the subsequent instruction.

Probability to execute a certain path $\sigma = (s_1, s_2, s_3, \ldots, s_n) \in G_E(V_E, S_E)$ is given by $P_\sigma = \prod_{i=1}^{n} P_{s_i}$, where P_{s_i} denotes probability to correctly or incorrectly predict the outcome of the operation

defined by the starting node of the arc s_i. Entire execution time of σ is defined as T_σ and it can be obtained as $T_\sigma = T + T_p$, because T is execution of one instruction and random variable T_p presents the execution time of the nodes in σ which can be overlapped in case of correct value prediction, i.e. $T_p = \sum_{t=1}^{n-1} t_{s_i}$, where t_{s_i} is execution time of the operation defined by s_i.

The random variable T_p has binomial distribution as illustrated on Eq. 1 [3]:

$$P(T_p = k \cdot T) = (1-p)^k \cdot p^{n-k-1} \cdot \binom{n-1}{k},\ 0 \le k \le n\text{-}1$$

Eq. 1 Binomial distribution of T_p

T_p has a binomial distribution because occurrence of value misprediction in the linear graph C (see Fig. 1b) is equivalent to choosing k occurrences out of $(n-1)$ with a probability to choose $(1-p)$. Thus, average execution time of the critical path can be calculated as mathematical expectation of T_σ as presented on Eq. 2 *Average execution time of the critical path*

.

$$\begin{aligned} E(T_\sigma) &= E(T + T_p) = E(T) + E(T_p) \\ &= T + \sum_{i=0}^{n-1} i \cdot T \cdot (1-p)^i \cdot p^{n-i-1} \cdot \binom{n-1}{i} \\ &= T + T \cdot (n-1) \cdot (1-p) \end{aligned}$$

Eq. 2 Average execution time of the critical path

As expected, Eq. 2 shows that average execution time of the critical path is prolonged due to value mispredictions. This is evident from the second part of the expression of $E(T_\sigma)$ which contains the probability of value misprediction $(1-p)$. Average execution time of the critical path is used to calculate average ILP boost in the ideal case of employing value prediction on a dataflow machine as presented on Eq. 3.

$$ILPboost_{VP} = \frac{n \cdot T}{E(T_\sigma)} = \frac{n \cdot T}{T \cdot (1 + (n-1) \cdot (1-p))}$$

$$ILPboost_{VP} \approx \frac{1}{(1-p)}$$

Eq. 3 Average ILP boost by employing value prediction on a dataflow machine

The expression presented on Eq. 3 shows that in ideal case the speedup obtained by employing value prediction is proportional to the accuracy of value prediction.

4 Value Prediction Latency

Reduced performance of value prediction due to latency of the value predictor is reported on multiple occasions [1, 2, 4, 8, 13, 15]. We modify the analytical model of the processor introduced in [3] in order to better understand the impact of value predictor latency. We change the assumption made in the model presented on Fig. 2 that execution time of correct value prediction is zero cycles. More precisely, we change execution time of correct value prediction from *0* to *L* cycles, where *L* denotes value predictor latency.

The modified execution graph of the critical path is presented on Fig. 3.

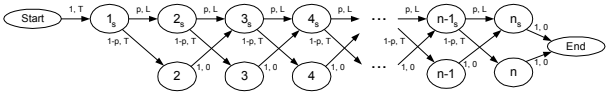

Fig. 3 Modified execution graph with additional value predictor latency

Obviously, change of execution time of correct value prediction introduces changes in the set of possible values of T_p as presented on **Error! Reference source not found.**.

$$P(T_p = k \cdot T + (n-k-1) \cdot L) = (1-p)^k \cdot p^{n-k-1} \cdot \binom{n-1}{k}$$

Eq. 4 Modified variable T_p

This affects the average execution time of T_σ, as presented on Eq. 5.

$$\begin{aligned} &E(T_\sigma) = E(T + T_p) = E(T) + E(T_p) = \\ &= T + \sum_{i=0}^{n-1} (i \cdot T + (n-i-1) \cdot L) \cdot (1-p)^i \cdot p^{n-i-1} \cdot \binom{n-1}{i} \\ &= T + T \cdot (n-1) \cdot (1-p) - L \cdot (n-1) \cdot (1-p) + L \cdot (n-1) \end{aligned}$$

Eq. 5 Average execution time of the critical path with value predictor latency (part 1)

The last expression can be rearranged as presented on Eq. 6 *Average execution time of the critical path with value predictor latency (part 2)*

.

$$E(T_\sigma) = T + (n-1)\cdot T\cdot(1-p) + (n-1)\cdot L\cdot p$$

Eq. 6 Average execution time of the critical path with value predictor latency (part 2)

The execution time of the critical path increases significantly with respect to the latency of the value predictor. Average ILP boost gained by employing value prediction with value predictor latency L on a dataflow machine is presented on Eq. 7.

$$ILPboost_{VPL} = \frac{n\cdot T}{E(T_\sigma)} = \frac{n\cdot T}{T + T\cdot(n-1) + (L-T)\cdot(n-1)\cdot p}$$
$$\approx \frac{T}{T + (L-T)\cdot p} = \frac{T}{T\cdot(1-p) + L\cdot p}$$

$$ILPboost_{VPL} \approx \frac{T}{T\cdot(1-p) + L\cdot p}$$

Eq. 7 Average ILP boost when employing value predictor with latency L

In case value predictor latency is zero cycles, we obtain the same expression for the ILP boost as [3]. However, if the latency is greater than zero, than the gain of value prediction is reduced to a speedup of only several times. Even with perfect value prediction (value prediction accuracy $p = 1$) leads to ILP boost of only several times as presented on Eq. 8.

$$ILPboost_{VPL} \approx \frac{T}{T\cdot(1-1) + L\cdot 1} = \frac{T}{L}$$

Eq. 8 ILP boost with perfect value prediction and value predictor latency L

This result suggests that value predictor latency should be reduced as low as possible, which excludes use of large value prediction tables and suggests overlapping of value prediction with other pipeline stages. Also, higher ILP boost is gained by predicting instruction with longer execution time T which suggest value prediction on load instructions, especially ones that miss in L1 data cache and long latency integer and floating point arithmetic instructions.

Further analysis of the effect of the value prediction latencies are presented on Fig. 4. We assume that $T = 7$ as it is suggested in [15], where the average execution time of instructions on the critical path is obtained using critical path profiling.

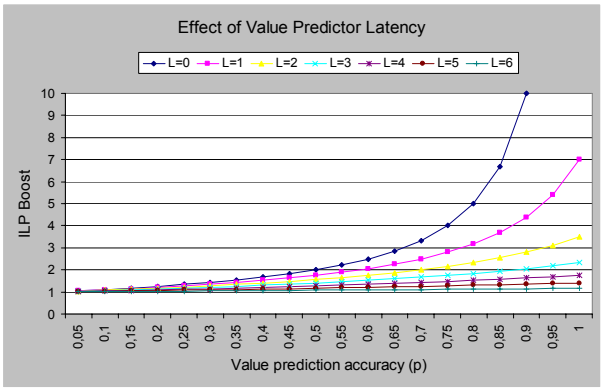

Fig. 4 Effect of value predictor latency

For zero cycle predictor latencies the ILP boost is hyperbolical as expected from Eq. 1. In this ideal case, which is practically impossible, value prediction has the potential to collapse all true data dependences in case value prediction accuracy $p = 1$. In this case, each program could be executed in two steps: (1) predict outcomes of all instructions and (2) validate predicted results.

If we consider value predictor latencies greater than zero, it is obvious that the performance potential drops significantly. Particularly, the ILP boost is at most $\frac{T}{L}$, as implied by Eq. 8. Similar behavior is observed in [4] while examining the performance potential of data reusing techniques. [4] reports performance decrease from about 20 to 2 when the reuse buffer latency is increased from zero to one.

We make additional observation on the way the ILP boost increases with the increase of the value prediction accuracy. When considering higher value predictor latencies the ILP boost assumes quasi-linear tendency of growth as reported on multiple occasions [1, 2, 5, 11, 13, 14, 15].

5 Conclusion and Future Work

Techniques based on principles of value locality and value predictability have potential to break true data dependencies. This is exploited in several manners: (1) to reduce or eliminate memory access latencies, (2) to execute data dependent instructions in parallel and (3) to buffer instruction results for later reuse. Numerous propositions are made on how to implement value prediction (and instruction reuse) into contemporary processors. All techniques have different approaches, but they achieve similar performance improvements of 5-20% when simulated in comparable environments.

The limiting factors on the performance potential of value prediction have been widely explored using experimentally obtained results. Few attempts have been made to analytically describe a processor that employs value prediction, none of which observes the effect of value predictor latency.

We introduced a new analytical model of value prediction, by extending the existing model of a dataflow machine which employs value prediction [3]. The analytical results confirm some previously experimentally observed behaviors, so that we can conclude that although our model is not detailed, it gives a general idea of the effect of value predictor latency.

Three general conclusions can be made: (1) value predictor latency must be reduced either by simplifying the way predictions are made or by reducing the value prediction tables and (2) value prediction latency should be overlapped by some other pipeline stages in order to reduce the value predictor latency as much as possible and (3) value prediction should be made only on instructions with high execution latencies. Our future work will study the third aspect of predicting high execution latency instructions, especially loads.

References

[1] R. Bhargava, L. K. John, "Latency and Energy Aware Value Prediction for High-Frequency Processors", *ICS'02*, New York, USA, 2002

[2] M. Burtscher, "Improving Context-Based Load Value Prediction", *PhD Thesis*, University of Colorado, USA, 2000

[3] F. Gabbay, A. Mendelson, "Using Value Prediction to Increase the Power of Speculative Execution Hardware", *ACM Transactions on Computer Systems*, Vol. 16, No. 3, pp. 234-270, 1998

[4] A. González, J. Tubella, C. Molina, "The Performance Potential of Data Value Reuse", *Technical Report UPC-DAC-1998-23*, University of Politecenica of Catalunya, 1998

[5] J. González, A. González, "Limits of Instruction Level Parallelism with Data Speculation", *Proceedings of the VECPAR Conf.*, pp. 585-598, 1998

[6] J. L. Hennessy, D. A. Patterson, "*Computer Architecture: A Quantitive Approach – third edition*", Morgan – Kaufmann Publishers, San Francisco, USA, 2003

[7] M. S. Lam, R. P. Wilson, "Limits of Control Flow on Parallelism", *in Proceedings of the 19th International Symposium on Computer Architecture*, Gold Coast, Australia, pp. 46-57, 1992

[8] S. Lee, P. Yew, "On Some Implementation Issues for Value Prediction on Wide-Issue ILP Processors", *International Conference on Parallel Architecture and Compiler Techniques*, 2000

[9] M. H. Lipasti, C. B. Wilkerson, J. P. Shen, "Value Locality and Load Value Prediction", *In Proceedings of the Second International Conference on Architectural Support for Programming Languages and Operating Systems*, pp. 138-147, 1996

[10] J. Markovski, M. Gusev, "Why Value Prediction is Limited?", *Technical Report PMF-II-01-2004*, FNSM, University of Sts. Cyril and Methodius, Skopje, FYROM

[11] P. Mitrevski, "Prediction and Speculation in ILP Processors", *PhD Thesis*, University of Sts. Cyril and Methodius, Skopje, Republic of Macedonia, 2002

[12] R. A. Moreno, "Using Value Prediction as a Complexity-effective Solution to Improve Performance", *Technical Report, DACYA-UCM 5/98*, 1998

[13] B. Rychlik, J. Faistl, B. Krug, J. P. Shen, "Efficacy and Performance Impact of Value Prediction", *In International Conference on Parallel Architectures and Compilation Techniques*, 1998

[14] Y. Sazeides, "An Analysis of Value Predictability and Its Application to a Superscalar Processor", *PhD Thesis*, University of Wisconsin, Madison, USA, 1999

[15] D. M. Tullsen, B. Calder, "Computing Along the Critical Path", *UCSD Technical Report*, 1998