#543 uVP: A Power-Efficient Architecture of value prediction via a micro-op cache

PC conflicts
Freddy Gabbay
Leonid Yavits
Lieven Eeckhout
Mikel Lujan
Yoav Etsion

Review preference

Submitted

Submission (800kB) 16 Apr 2021 2:02:41pm PDT · 2d7b7883

Abstract
The potential benefits of value prediction (VP) to enhance out-of-order processor performance have been broadly proven by various past studies. Despite of the potential performance improvement, VP micro-architecture complexity and power overhead introduce a real barrier for deployment in modern microprocessors. This work presents uVP; a novel power-efficient architecture enabling processors that employ a uOP cache to efficiently integrate VP mechanisms. The new proposed scheme can take advantage of the ILP improvement offered by VP while significantly reducing power overhead and integration complexity with respect to other implementations of VP schemes. The uVP architecture suggests dividing the implementation of VP into two phases: a training phase and a deployment phase. In the training phase, a traditional value predictor is used to learn the predicted values and their corresponding confidence levels. In the deployment phase, predicted values with high confidence level are moved into the

[more]

Authors (blind)
D. Recher, A. Mendelson, F. Gabbay [details]

✓ I Confirm That All Authors Have Signed the Author Ethics Form

RevisionLetterSharingOptions
First submission

RevisionLetterVisibleAll (39kB)

Artifacts
Unlikely

Travel
Likely

✓ Please select up to FOUR topics below that best represent the insights and contributions of your submission

▶ Topics

	Nov	Eva	WriQua	OveMer	RevExp	ImpRevReb
Review #543A	3	2	2	2	3	1
Review #543B	2	3	3	2	4	1
Review #543C	2	1	3	2	4	3
Review #543D	3	2	3	3	4	3
Review #543E	1	2	2	2	3	1
Review #543F	2	3	2	3	3	4

You are an **author** of this submission.

Edit submission Add response

Reviews in plain text

Review #543A

Paper Summary
This paper presents uVP, a value-prediction approach that is partially integrated with the uOP cache. The training is performed in a separate piece of hardware that is not on the critical path. The idea is evaluated on an LVP and a VTAGE predictor. This technique provides the standard benefits of value prediction while reducing power and integration complexity.

Strengths
decoupled prediction architecture; integration with uOP cache; good performance

Weaknesses
idea not well described; key results missing; many open questions

Novelty
3. New contribution

Evaluation
2. The evaluation is shallow and there are major weaknesses

Writing Quality
2. Needs improvement

Overall Merit
2. Poor (D) -- Poor-quality paper

Reviewer expertise
3. Knowledgeable

Questions for Revision/Rebuttal:
How are confident predictions moved to the uOP cache? What is the hardware path to do so? Are extra write ports needed in the uOP cache?

VTAGE may predict different consecutive values for the same uOP executed in a loop. How is this done using the uOP cache entry, which holds a fixed prediction value?

Do you account for the energy used by the VP (training) hardware?

Importance of Revision/Rebuttal in Determining Your Post-Response Score
1. Revision/Rebuttal cannot save this paper -- The required changes are so significant that it will become a different paper

Comments to authors
I like the uVP idea, especially the decoupled approach of placing the training hardware away from the critical path. Having said that, the paper does not describe the idea well. Many important details are missing. For example, how are confident predictions moved into the uOP cache? How are predictors supported that predict different consecutive values for the same static instruction? How is the prediction power accounted for in the energy measurements? The paper is not at the page limit, so there should be enough space to add more detailed explanations of how the uVP hardware works.

The results are incomplete. In addition to the prediction accuracy, please also show coverage results. In fact, it would be best to present the 2x2 confusion matrix to provide the full picture (percentage of instructions that were correctly predicted, incorrectly predicted, not predicted but the prediction would have been correct, and not predicted and the prediction would have been wrong).

Also, given the decoupled nature of uVP, I would have expected a sensitivity study on the size of the uOP cache.

Minor comments: Please proofread the paper, check the spelling, and fix the unresolved references.

The caption of Fig. 9 does not agree with the chart title. Also, you can probably remove the (redundant) chart title in this and other figures.

Which of the two VPs does Figure 8 use?

Review #543B

Paper Summary
The paper tries to optimize the implementation of value prediction by using the micro-op cache to store the value predicted data. This can simplify the prediction and learning.

Strengths
Neat idea to use an existing resource (uop cache) to accomplish value prediction easily in a modern OOO processor pipeline.

Weaknesses
The idea is more of a design improvement and does not give any insight into how value prediction can be fundamentally improved. Adding data to each uop cache entry can be area intensive and may degrade uop-cache capacity benefits (at same area).

Novelty
2. Incremental improvement

Evaluation
3. There are some minor issues with the evaluation, but they can be solved

Writing Quality
3. Adequate

Overall Merit
2. Poor (D) -- Poor-quality paper

Reviewer expertise
4. Expert

Questions for Revision/Rebuttal:
Since only one data can be stored in the u-op cache, how will context based value prediction (VTAGE) work ? How much additional performance gains does context value prediction give over last value prediction ?

Importance of Revision/Rebuttal in Determining Your Post-Response Score
1. Revision/Rebuttal cannot save this paper -- The required changes are so significant that it will become a different paper

Comments to authors
The ideas in the paper are interesting. However I felt the novelty is somewhat low. There are no real insights on how value prediction could be improved. Also putting data to every uop cache entry may not be a very area friendly solution.

A lot of work has shown that value prediction can be restricted to just loads. Only a small fraction of loads will actually occupy the uop cache entries. But adding a storage for every uop may be an overkill. You may want to have storage for every N uops, (assuming you may not want to value predict nearby loads, and the chance of loads bunched together is small). That could reduce area requirements somewhat.

Also uop cache capacity is very strained. SPEC may be fine, but high footprint server workloads may suffer. Instead of increasing storage of uop cache for VP, a better tradeoff would be to increase its capacity for the same area.

There was also a recent work in ISCA'20 where the authors reduced storage of value prediction (using smart techniques to focus value prediction on only a small subset of critical loads). They showed much smaller area for VP than existing solutions, which should make search and implementation easy for VP.

Review #543C

Paper Summary
This paper discusses integrating value prediction in the context of a uop cache. Uop caches introduce unique challenges with mapping pcs (uops vs instructions) as well as unique opportunities, directly integrating value predictions into the uop itself when stored in the cache. Integrating value prediction into a uop cache architecture requires that uop sequences must be built using value predictions that come from stable, previously trained value predictors. In light of this, uop sequences are built in two phases, first a training phase in which value predictions are pulled from the value predictor, second a deployment phase in which stable value predictions are fetched directly from the uop cache. The authors report performance/power benefits from the both integration of the uop cache (figure 8), as well the benefits of the integration of a value predictor over a baseline with just a uop cache (figure 10).

Strengths
The paper discusses interesting tradeoffs on integrated two techniques. These techniques seem to be complementary.

Weaknesses
The paper proposes integration of two well known techniques and seems fairly incremental.

Novelty
2. Incremental improvement

Evaluation
1. The idea is not appropriately evaluated

Writing Quality
3. Adequate

Overall Merit
2. Poor (D) -- Poor-quality paper

Reviewer expertise
4. Expert

Questions for Revision/Rebuttal:
This paper discusses the integration of a uop cache and value prediction and reports the performance benefit that these deliver in concert. As a baseline, the authors choose a baseline where one or the other of these well known techniques is not used. Aren't there more naive, straightforward integration of these techniques that could be used as a better baseline?

Importance of Revision/Rebuttal in Determining Your Post-Response Score
3. Revision/Rebuttal would be somewhat useful -- I am open to revising my score based on how satisfactorily the concerns are addressed.

Comments to authors
Are you sure you haven't conflated uop caches with trace caches? Uop caches aren't necessarily constructed around basic blocks.
You're discussing the integration of two well known techniques and you've described the some of the engineering that is required to efficiently integrate these techniques. The performance benefit you cite is against a baseline missing either one or both of the known techniques you're evaluating. Is there a way to isolate the performance benefit your innovations contribute? It seems like you could do this by comparing to a naive integration of uop caches and value predictors.

Review #543D

Paper Summary
Paper proposes to embed value predictions in the uop cache (only applicable to cores with uop cache). This reduces predictor pressure/power since predictable instructions are not training the predictor (once installed in the uop cache), and the integration of value-prediction in the pipeline is simplified. The idea is to have a value-predictor trained by an instruction after an instruction is decoded in the uop cache. Then when an instruction output is considered predictable copy the prediction in the uop cache and stop training the value-predictor. The value in the uop cache is sticky does not change but the prediction is used as long as it is correct.

Strengths
Interesting idea to leverage uop cache for value prediction

Weaknesses
Many details missing and some of the paper claims are not quantified.

Novelty
3. New contribution

Evaluation
2. The evaluation is shallow and there are major weaknesses

Writing Quality
3. Adequate

Overall Merit
3. Average (C) -- Average-quality paper with many deficiencies that are difficult to overlook. Describe changes needed for this paper to become Good (B).

Reviewer expertise
4. Expert

Questions for Revision/Rebuttal:
see comments to authors

Importance of Revision/Rebuttal in Determining Your Post-Response Score
3. Revision/Rebuttal would be somewhat useful -- I am open to revising my score based on how satisfactorily the concerns are addressed.

Comments to authors
I liked the idea of leveraging the uop cache as means to propagate value predictions. Although I am against accepting paper for publication in its current form, due to the papers many deficiencies, I find the idea promising and with some more thought/effort it can become more attractive with potential to impact industry.

The paper claims simpler integration but it does not discuss/explain how and when the predicted value is propagated to the dependent instructions. To me this seems to require widening the datapath, adding muxes or something else or .... Explain. Discuss why this help simplify integration.

Another missing detail is what is the key used to train for vtage? If we go by the text, it is the bb address + uop-offset. But this is not the info used to index the predictor that combines ghist and IP. May be the authors do something else but it is not described in the paper.

The work suggests that the values can piggy-back existing fields but don't discuss this further. As this is central to the cost of the approach, you need to elaborate as to whether indeed this is feasible and if not what is the cost of the approach and impact on uop access time/power due to wider fields.

State clearly what is the index to each predictor (numbe rof bits and what info), and what are sizes of the different fields. Including the key into uop for each entry seems very costly. I am not sure if you accounted for it.

For ulvp I do not understand why you need a key in each predictor entry and uop entry - you know what is the ip of the predicted instruction, so yo uhave all info need to train predictor and also install in the uop cache?

The work seems to suggest that is not using FPC for LVP, why? This is an easy fix and will help reduce mispredicts.

It is not clear when flushing is done in the simulated pipeline, at resolution time or commit time. Doing at commit it aggravates the mispredict penalty. Most state of the art cores use resolution time for control flow mispredicts.

Need to quantify the benefit from stopping training of predictable instructions as compared to training all ips all the time. This is to confirm that indeed doing helps reduce predictor pressure. Updating all the time helps install faster an instruction that is predictable and got evicted.

What type of instructions are value predicted?

other issues

Error reference

aa

simulated core used seems dated

simulated regions how long are they and they representative?

u bit in tage is for usefulness and helps guide replacement (dont say LRU)

in your results include prediction coverage

fig. 10 and fig. 11 use consistent bench order and colors - try to look into x264 that had a huge performance improvement where the benefits come from. Also provide a more detailed explanation for the energy increase of exchange.

duplicated refs 8 and 10, some refs capitalized.

Review #543E

Paper Summary
This paper proposes doing value prediction for uOPs via the combination of uOP cache and existing value prediction techniques like Last Value Predictor (LVP) and VTage. They achieve a 13% performance gain when using uOP cache + LVP and 28% when using uOP cache + VTage against an old Intel processor -- Intel Gainestown core.

Weaknesses
The proposed mechanism is quite similar to the HPCA 2015 paper: "BeBoP: A Cost Effective Predictor Infrastructure for Superscalar Value Prediction". Perhaps the only difference is that while the previous work looks at a generic instruction block size, e.g. 16B, this paper proposes doing Value Prediction based on basic block granularity, which is quite small to justify a new contribution.
Paper writing is poor and needs a lot of work.

Novelty
1. No novelty - (work has been published before or openly commercialized)

Evaluation
2. The evaluation is shallow and there are major weaknesses

Writing Quality
2. Needs improvement

Overall Merit
2. Poor (D) -- Poor-quality paper

Reviewer expertise
3. Knowledgeable

Questions for Revision/Rebuttal:
1. What's the difference between what's proposed here and what has been done in the past work "Block Based Value Prediction -- BeBoP"?
2. What are the possible values of PV and TV? Can they both be 0 or both be 1?
3. What exactly is included in the Key/Value field?
4. In the evaluation section, why is the comparison done against Intel Gainestown, a very old Intel CPU instead of a more recent CPU?

Importance of Revision/Rebuttal in Determining Your Post-Response Score
1. Revision/Rebuttal cannot save this paper -- The required changes are so significant that it will become a different paper

Comments to authors
Paper writing needs a lot of work. I'd suggest instead of describing uOP cache and VTage in length, the authors should focus on the core idea that they propose. Besides, section 2.2 was written in a different font than the rest of the paper and has lots of typos. Figure formatting needs to be improved as it's quite blurry at the moment.

Review #543F

Paper Summary
The paper gives a value prediction optimization in which predicted values are stored in the micro-op cache. The idea is described and its effect on performance and energy quantified through experiments.

Strengths
Value prediction seems to have great potential for improving performance and this paper helps demonstrate that potential. The idea is relatively simple if the processor already incorporates a micro-op cache.

Weaknesses
The writing is rough. The penalty of a misprediction is underestimated.

Novelty
2. Incremental improvement

Evaluation
3. There are some minor issues with the evaluation, but they can be solved

Writing Quality
2. Needs improvement

Overall Merit
3. Average (C) -- Average-quality paper with many deficiencies that are difficult to overlook. Describe changes needed for this paper to become Good (B).

Reviewer expertise
3. Knowledgeable

Questions for Revision/Rebuttal:
What would be the performance improvement for a range of misprediction penalties from, say, 10 to 30 cycles?

Importance of Revision/Rebuttal in Determining Your Post-Response Score
4. Revision/Rebuttal would be very useful -- My "post-response" score will mainly be determined by how satisfactorily the concerns are addressed

Comments to authors
The idea is simple. There is previous work the authors fail to cite, e.g. the recent spate of value prediction papers like "Focused Value Prediction" from ISCA 2020 and so forth but the work I have seen is somewhat orthogonal to this approach, which gives an improved mechanism for using value prediction without modifying the predictor too much.

The writing is quite rough. There are misspellings, awkward phrasing, and a weird error of "Error! Reference source not found."

I think a big problem in this paper is that the misprediction penalty is far underestimated. Section 4.2 gives the value misprediction penalty as 5 cycles. Section 2 and Table 1 say that mispredictions are handled the same way as branch mispredictions, i.e., with squashing the pipeline. Real processors based on current microarchitectures that might benefit from value prediction have much higher branch misprediction penalties. A pipeline flush will cost at least 20 cycles (or more depending on the pipeline details and recovery cost), but that doesn't account for extra delay introduced by cache pollution and wasted instruction prefetch bandwidth by going far down the wrong path. With value prediction there are even more opportunities for this kind of penalty as data addresses can be predicted down the wrong path and pollute the cache even more. So, it would be necessary to model both a modern deep pipeline with a commensurate branch misprediction penalty as well as to model wrong path execution. My understanding is that Sniper currently doesn't go down the wrong path although I have heard of extensions that do so. So I have difficulty accepting the paper without this improved methodology. It is possible that, by improving the methodology, experiments would show that the performance benefit evaporates as mispredictions become more costly and confidence estimation has to be tuned to avoid a slowdown rather than a speedup over all benchmarks. I understand that predictor accuracy is high, but just as with branch prediction, even a low misprediction rate yields a large number of cycles wasted to mispeculation.

Add response