

Leveraging Value Equality Prediction for Value Speculation

Kleovoulos Kalaitzidis, André Seznec

▶ **To cite this version:**

Kleovoulos Kalaitzidis, André Seznec. Leveraging Value Equality Prediction for Value Speculation. ACM Transactions on Architecture and Code Optimization, 2021, 18 (1), pp.1-20. 10.1145/3436821 . hal-03097413

HAL Id: hal-03097413
https://inria.hal.science/hal-03097413

Submitted on 25 May 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Leveraging Value Equality Prediction for Value Speculation

KLEOVOULOS KALAITZIDIS and ANDRÉ SEZNEC, Univ Rennes, Inria, CNRS, IRISA, France

Value Prediction (VP) has recently been gaining interest in the research community, since prior work has established practical solutions for its implementation that provide meaningful performance gains. A constant challenge of contemporary context-based value predictors is to sufficiently capture value redundancy and exploit the predictable execution paths. To do so, modern context-based VP techniques tightly associate recurring values with instructions and contexts by building confidence upon them after a plethora of repetitions. However, when execution monotony exists in the form of intervals, the potential prediction coverage is limited, since prediction confidence is reset at the beginning of each new interval.

In this study, we address this challenge by introducing the notion of *Equality Prediction* (EP), which represents the binary facet of VP. Following a twofold decision scheme (similar to branch prediction), at fetch time, EP makes use of control-flow history to predict equality between the last committed result for this instruction and the result of the currently fetched occurrence. When equality is predicted with high confidence, the last committed value is used. Our simulation results show that this technique obtains the same level of performance as previously proposed state-of-the-art context-based value predictors. However, by virtue of exploiting equality patterns that are not captured by previous VP schemes, our design can improve the speedup of standard VP by 19% on average, when combined with contemporary prediction models.

CCS Concepts: • **Computer systems organization** → *Pipeline computing*; **Superscalar architectures**;

Additional Key Words and Phrases: Microarchitecture, general-purpose processor, value prediction, equality prediction, hardware speculation, physical register sharing

ACM Reference format:
Kleovoulos Kalaitzidis and André Seznec. 2020. Leveraging Value Equality Prediction for Value Speculation. *ACM Trans. Archit. Code Optim.* 18, 1, Article 13 (December 2020), 20 pages.
https://doi.org/10.1145/3436821

1 INTRODUCTION

Current flagship processors tend to employ large instruction windows in order to efficiently extract more instruction-level-parallelism (ILP). But even then, performance improvements are inherently limited by the true data dependencies between instructions. Value Prediction (VP) [4, 12, 13] was introduced as a technique to collapse these restricting dependencies by allowing

Now with Huawei Technologies, Zurich Research Center, Switzerland, Oerlikon, Thurgauerstrasse 40, 8050 Zürich.
Extension of Conference Paper: This article is an extension of work originally presented in ICCD'19 [11]. The additional material includes key details about the operation of Equality Prediction with ETAGE, an extensive evaluation based on full experimental results and important insights about VSEP. This article also includes a short analysis of an alternative design using Equality Prediction, but combining Value Prediction and Physical Register Sharing.
Authors' address: K. Kalaitzidis, Huawei Technologies, Zurich Research Center, Thurgauerstrasse 40, Oerlikon, Zurich, Switzerland, 8050; email: kleovoulos.kalaitzidis@huawei.com; A. Seznec, Univ. Rennes, Inria, CNRS, IRISA, Campus de Beaulieu, 263 Avenue du Général Leclerc, Rennes, France, 35042; email: andre.seznec@inria.fr.



This work is licensed under a Creative Commons Attribution International 4.0 License.

© 2020 Copyright held by the owner/author(s).
1544-3566/2020/12-ART13
https://doi.org/10.1145/3436821

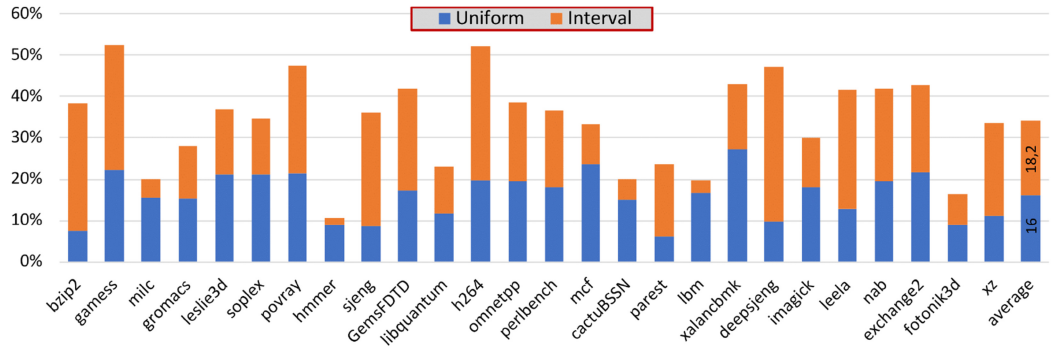


Fig. 1. Classification of programs static instructions according to the form of consecutive value equality they expose. The two classes are mutually exclusive.

dependent instructions to execute ahead of time using speculative sources. Initially proposed in the late 90's, VP has been a primary factor of contention for over a decade due to the high complexity its implementation can impose. However, recent studies [15–17] disclose that such complexity can be highly mitigated.

Value predictors can be classified into two main families: *context-based* and *computation-based* [25]. Context-based predictors aim to generally capture the repetition of the same value, while computation-based predictors compute the predicted value from the last results of an instruction, e.g., stride predictors [5, 36]. In essence, computation-based predictors require the actual or predicted result of previous instruction occurrence(s) to generate a "fresh" prediction. Predicting an instruction's result with a computation-based predictor requires a complex associative search in the speculative window [17] for potential in-flight occurrences of the instruction.

On the other hand, the state-of-the-art context-based Value TAgged GEometric (VTAGE) value predictor [16] does not require such a complex integration in the out-of-order (OoO) execution engine and simply relies on a set of tables read at fetch time. VTAGE aims to capture the correlation of instruction results with the global branch history, i.e., the same value encountered with the same global branch history. To achieve this functionality, VTAGE is implemented with a plurality of tables indexed with different branch history lengths. Each entry in these tables features a full 64-bit value, a partial tag, and a confidence counter, i.e., around 80 bits in total.

In this study, we aim to capture another form of this regularity: *equal values on consecutive occurrences of the same static instruction.*[1] Two different kinds of *value equality* can be discriminated:

(1) **Uniform:** All dynamic instances of a static instruction always produce the same result (e.g., a constant value-sequence [25]).
(2) **Interval-style:** Within an interval of repetitive executions, a static instruction produces the same result. But for each interval, the result is different. Practically, it resembles the uniform pattern limited within an interval.

Equality of the results of two consecutive occurrences of a static instruction is very frequent. We illustrate this in Figure 1, characterizing *uniform equality* and *interval equality* of our benchmarks (discussed in Section 4.2). As reported, interval equality is prevalent across the applications considered in our study, representing on average more than 18% of a program's static instructions and plainly dominating the rates of uniform equality in 12 out of 26 cases. To our knowledge, no

[1]A static instruction is uniquely identified by its address and may be executed more than once for a specific program execution, e.g., in a loop.

previous context-based value prediction scheme has been designed with total awareness of interval equality. Prior work has only considered uniform equality, such as the Last Value Predictor (LVP) [12].

In this work, we introduce a binary-nature VP technique called *Value Speculation through Equality Prediction* (VSEP). VSEP is practically a context-based value predictor that exploits both types of *consecutive value equality* that instruction results may exhibit (i.e., *uniform* and *interval*). Similar to VTAGE, VSEP does not require a complex integration in the OoO execution engine. Though, contrarily to VTAGE, VSEP does not require wide entries in all the predictor tables.

To allow for performance gains with VP, reaching a high-confidence value generally necessitates many successive correct predictions (typically more than 100) [16]. Commonly, in context-based value predictors (from the oldest to the most recent [12, 16]), the prediction confidence is associated with the value to be predicted in the same predictor entry. As a result, on instructions exhibiting interval equality, prediction reaches high confidence only on long intervals. In VSEP, we do not associate the confidence with the value in the same predictor entry, but rather predict the following binary information, i.e., whether the result of the currently-processed occurrence of instruction *K* is very likely to be equal to the last committed value for instruction *K*. More precisely, VSEP consists of a TAGE-like *equality predictor* that we call *ETAGE*, and a table that tracks the last *committed* values of instructions, namely LCVT. ETAGE detects (likely-)equality without explicitly defining the value to be predicted by using the value provided from LCVT. As a consequence, when the VSEP predictor is able to identify the beginning of intervals, it may resume useful (i.e., high-confidence) predictions as soon as it can anticipate with high confidence that the first instruction occurrence in the interval has been committed. Our simulations show that VTAGE and VSEP performance improvements are in the same range, but their prediction coverage is partially orthogonal, in terms of the value patterns that they capture.

Overall, this study makes the following contributions:

- We identify two different forms of *consecutive value equality*, the **uniform** and the **interval**, and show how the latter diminishes value predictability on current context-based value predictors.
- We propose ETAGE, an equality predictor that solely identifies equality between the current execution of a static instruction and its last committed result.
- We propose VSEP, a value predictor that leverages the ETAGE equality predictor to perform value prediction.
- We present a comprehensive analysis of our scheme and compare it against prior work in value prediction, revealing how our solution complements existing techniques by increasing the fraction of predicted instructions that expose interval equality.
- Finally, we propose a practical integration of the state-of-the-art VTAGE with VSEP that delivers approximately 7% of average speedup, i.e., 19% higher than VTAGE/VSEP alone.

In Section 6, we briefly describe an alternative design to VSEP that is also based on Equality Prediction called *Previous Occurrence Equality Prediction*, or POEP. POEP relies on predicting that an instruction produces the same result as its previous occurrence while VSEP relies on predicting that the instruction produces the same result as its last committed occurrence. When equality is predicted with high confidence, if no other occurrence is present in the window of in-flight instructions, then the last committed value read on LCVT is used; if another occurrence is still in-flight, then *physical register sharing* [10] is used. That is, at rename time, the rename register is forced to match the result register of the previous occurrence of the instruction. Our simulation results showed that POEP did not provide better performance than VSEP despite the significant extra hardware complexity (use of physical register sharing, complex equality predictor training).

2 RELATED WORK

According to Sazeides et al., the different value predictors developed over the years can be classified into two broad categories, *Context-based* and *Computation-based* value predictors [25].

Context-based value predictors include LVP [12] which is indexed by the program counter and the Finite Context Method (FCM) predictor indexed by a hash of instructions last values [25]. More recently, VTAGE [16] was introduced. VTAGE leverages global branch history as a context and does not depend on any value history; thus, it does not require a complex integration in the OoO core. It is a TAGE-like value predictor that uses a set of tagged prediction tables that are indexed using a hash of the instruction PC and the global branch history. Each table is accessed by using a different number of bits from the global branch history forming a geometric series.

In the same work, Perais and Seznec pointed out that in-order *Validation* at commit coupled with a full pipeline flush on a misprediction is cost-effective provided that predictions are only used with very high confidence (over 99%). To achieve this, they proposed the use of Forward Probabilistic Counters (FPC) [23] to track confidence of value-entries. In a nutshell, the predictor filters the predictions that are used in the pipeline, sacrificing some coverage for more than 99% accuracy.

Computation-based predictors include the Stride predictor [5], but also the Differential FCM (D-FCM) [7] and Differential VTAGE (D-VTAGE) [17]. These predictors necessitate a complex integration in the OoO core. The computation of the predicted value may depend on in-flight speculative values, thus requiring a complex associative search in the instruction window. With the E-stride component in the Enhanced VTAGE Enhanced Stride (EVES) predictor [27], this issue is partially addressed; one has just to count the number of in-flight occurrences of the instruction.

In all the works above, values are predicted for all register-producing instructions (i.e., regardless of the type of their operation). There are also other studies that focus only on load-value prediction [1, 13, 14, 30, 31, 34, 35]. In particular, in Ref. [13], Lipasti et al. introduced the very first *Load-Value Predictor* that exploits the relation between load instruction addresses and the values being loaded. To tackle the high cost of load-value mispredictions, in Refs [34] and [35], the authors propose a pipeline-flush-free methodology where only the values of those loads that are safe-to-approximate and miss in the cache can be predicted. Furthermore, the recent Focused Value Predictor (FVP) [1] aims to reduce area and power by only predicting loads that produce the inputs of some other long-latency loads that are very likely to stall execution.

On the other hand, in Refs [14], [30], and [31], load-value prediction is implemented indirectly by using a different approach; rather than traditionally predicting the values of loads, these methods are based on predicting the memory addresses from where load-values are fetched. These predicted addresses are eventually used to retrieve the load-values earlier in a speculative manner. Finally, the composite predictor of Sheikh and Hower as shown in Ref. [31] combines this address-prediction approach with other typical prediction models in order to increase coverage at the cost of some additional complexity imposed on the predictor's critical path.

Value prediction can also draw from a wealth of work on the all-year Championship Value Prediction (CVP). However, analyzing all the models proposed in CVP would not serve the purpose of this article. Interested readers are highly encouraged to visit the dedicated site of CVP[2] in order to delve into the plethora of contest proposals.

3 VALUE SPECULATION THROUGH EQUALITY PREDICTION—VSEP

In this section, we describe VSEP, our proposed scheme for leveraging Equality Prediction (EP) in order to perform value prediction. Then, we present a synopsis of its prediction logic by comparing it with already established methods of value prediction.

[2] https://www.microarch.org/cvp1/.

3.1 Value Prediction Using ETAGE

VSEP consists in two distinct components, ETAGE, the equality predictor, and LCVT, the Last Committed Value Table.

ETAGE is a context-based equality predictor that essentially copies the TAGE branch predictor structure [29]. That is, it employs a series of prediction tables that are indexed by a different number of bits of the global branch history, hashed with the instruction PC. These main tables are also backed up by a tag-less base table that is directly indexed with the instruction PC. The entries of the tagged tables contain the 1-bit equality information, a partial tag, a 1-bit usefulness indicator, and a 3-bit confidence counter. Since we leverage EP to guide the speculation of instruction results, high accuracy is essential to avoid performance degradation. Hence, as already established by modern VP-schemes, we use confidence counters to filter the predictions that are promoted to the pipeline in order to guarantee very high prediction accuracy[3] (over 99%). Therefore, a predicted value is used by the pipeline if and only if *equality* is predicted with high confidence. To ensure that high confidence is reached only after a large (i.e., sufficiently safe) number of equality occurrences, ETAGE-entries feature Forward Probabilistic Counters (FPCs) [23] of three bits. Experimentally, we found that using the probability vector $V = \left\{1, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}\right\}$ to control forward transitions provides a good tradeoff. In the case of an *inequality*, the confidence counter is reset.

On the other hand, LCVT records the value of the most recently committed instance of instructions, i.e., their last committed value. The particular LCVT that we employ is a 3-way set associative table of no more than 3K entries, as we found that this size is sufficient to track execution in our framework (discussed in Section 4.3). Along with the full 64-bit committed value, each entry includes a 13-bit tag, which is a folded hash of the instruction PC. In the paragraphs that follow, we detail the prediction and training logic that VSEP follows.

Prediction. ETAGE delivers a prediction after accessing all its main tables in parallel in a search of the tagged entry. When matching entries are present in multiple tables, the prediction arrives from the one accessed with the longest history. In the absence of any matching entry, the base table provides the prediction. Concurrently to the prediction process, LCVT is accessed to provide the last committed value of the relevant instruction. It should be noted that several occurrences of the same instruction may be simultaneously present in the window of in-flight instructions at prediction time. LCVT does not provide the result of the previous occurrence of the instruction, but rather provides the result of the last committed occurrence of the instruction. On a miss in LCVT, *inequality* is assumed. Recall that the predicted value, if any, is used in the pipeline only if *equality* is predicted with high confidence.

Training. At update time, the entry of ETAGE that provided the prediction (i.e., equality or inequality) is updated with the execution outcome, i.e., the actual equality/inequality of the current instruction result with the speculative value retrieved from LCVT at prediction time. Their agreement triggers the increment of the entry's confidence counter, according to the probability vector described above. On a misprediction, the entry's equality-information bit is replaced if its confidence is already equal to zero. If confidence is not equal to zero, it is reset so that the entry's equality-information bit will be corrected in the next execution. Moreover, a new entry is allocated in a "higher" table (i.e., accessed with a longer history), following a similar allocation policy as the TAGE [29] branch predictor. In addition, the usefulness bit that leads the entries replacement is set when the produced prediction is correct and the prediction that could have been

[3]Accuracy is defined as the number of correctly predicted dynamic instructions divided by the total number of predicted dynamic instructions.

alternatively derived from a table with shorter history is inverse. In the TAGE algorithm, this alternative prediction-option is called *alternate prediction* [29]. In the opposite case (i.e., when the alternate prediction is actually correct since it is the inverse of the processed one), the usefulness bit is reset. In any case, the usefulness bits of all the tagged entries are periodically reset to avoid the possible endless protection of certain entries from replacement. Finally, regardless of the execution outcome, LCVT is updated with the committed result.

Pipeline Details. In VSEP, validations are done in-order at commit time to reduce OoO core complexity by banking the physical register file [16]. To enable validation at commit, a FIFO *validation queue* with size equal to that of the instruction window is employed. Each entry holds the 1-bit equality prediction of ETAGE and the value retrieved from LCVT at prediction time (around 2KB of storage space is required considering the 256-instruction window of our framework). We assume the following process:

- At Fetch, leverage the ETAGE equality predictor to generate a high-accuracy prediction that specifies the equality of the current instruction's result with the last committed value. In parallel, index LCVT to acquire the instruction's last committed value. Place both the equality prediction and the potentially predicted value in the validation queue.
- At Rename, if *equality* is predicted with high confidence with a tag-hit in LCVT, the predicted value is written onto the physical register file (PRF). Conversely, when *inequality* is predicted or *equality* does not have sufficiently high confidence, the predicted value is dismissed (i.e., not used).
- At execution, overwrite the predicted value with the computed one in the PRF.
- Before Commit, use the validation queue to validate the predicted value against the computed result. Flush the pipeline on a misprediction if the predicted value was inserted in the pipeline. Similarly, validate the equality prediction in order to adequately update the ETAGE predictor. Finally, update LCVT with the committed result.

3.2 Dissecting the Prediction Scenario of VSEP

In previous context-based value predictors, the predicted value is precisely defined by the predictor's entries that tightly associate the predicted value with its confidence. In VSEP, the predicted value is not directly specified by the equality predictor ETAGE. ETAGE solely defines the potential equality/inequality of the current instruction-instance result with the last committed value of the same static instruction. When high-confidence *equality* is predicted, the corresponding last committed value constitutes the predicted value. In this case, the value that is used to perform the prediction of the instruction's result is provided from a cache-like table of committed values, namely LCVT. Overall, the unique feature of the VSEP prediction scheme is that ETAGE makes a *de facto* decision of whether value speculation can take place independently from the predicted value.

Being a context-based predictor that leverages the global branch history as context, ETAGE can exploit control-flow history, similarly to VTAGE. Both predictors implement the TAGE algorithm [29]; therefore, they are both able to identify precise positions in the control flow path. Below, we describe the general prediction scenario for LVP, VTAGE, and VSEP on instructions featuring interval value equality. This is illustrated in Figure 2, where we assume that the examined instruction K returns the value X on the first interval, and then flips to value Y on the second interval.

LVP learns the equality in the first interval and reaches high confidence. At this point, the prediction from LVP can be used in the pipeline. When the result flips to Y, a misprediction is encountered and a misprediction penalty must be paid. Thereafter, LVP has to re-train on the second interval before reaching high confidence and becoming used in the pipeline again.

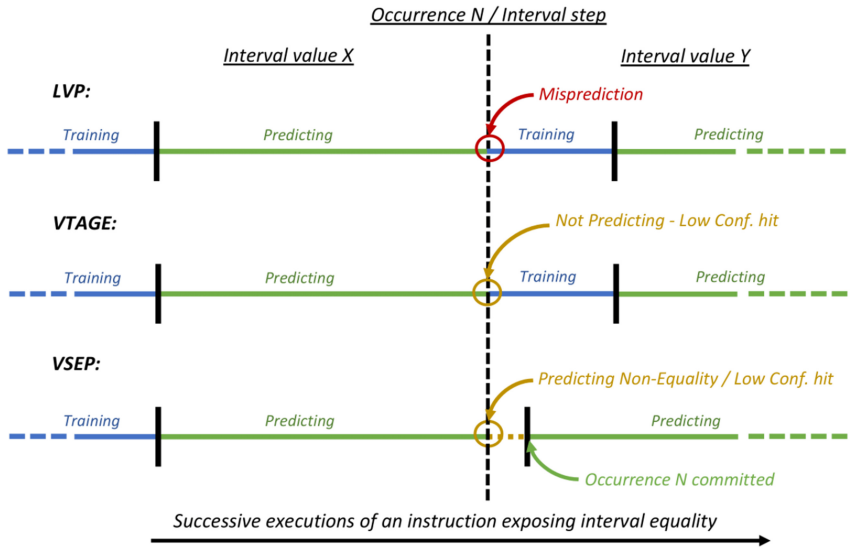


Fig. 2. Prediction scenarios in presence of interval equality.

VTAGE will also learn of equality in the first interval, and likely of the value change. However, the new value *Y* has to be re-learned by the predictor entries, and, therefore, the prediction does not reach high confidence before a large number of occurrences of the instruction *K* (generally more than 100 [16]).

Similarly, ETAGE learns the equality in the first interval. When the value change is correlated with the control flow path, ETAGE also learns to detect it and therefore will not suffer a misprediction on the flip to *Y*. When fetching after *Occurrence N* of the instruction that flips to *Y*, the value read from LCVT remains *X* until *Occurrence N* has retired. ETAGE is able to identify this in the control-flow history. Essentially, the entries that are associated with the first occurrences do not reach high-confidence levels, and the respective predicted values are not used. When Occurrence N has retired, or more precisely, is very likely to have retired (in practice when the distance between the currently fetched occurrence and *Occurrence N* is larger than the instruction window size), LCVT provides the value *Y*, and ETAGE identifies the presence of *equality*. Since ETAGE entries for the first occurrence and subsequent occurrences are distinct, prediction from ETAGE exhibits high confidence as soon as the first occurrence has been committed. At an interval change, the number of occurrences that are not predicted is roughly proportional to the size of the instruction window divided by the average distance (in number of instructions) between two consecutive occurrences of the instruction.

For a better understanding of the ETAGE description provided above, Figure 3 depicts a snapshot of an execution pipeline augmented with ETAGE for value prediction, i.e., employing the VSEP scheme. In this particular example, five consecutive occurrences of the instruction *K* coexist in the window of in-flight instructions, processed in different pipeline stages. Among these occurrences, K_1 is the oldest and represents the respective *Occurrence N* that flips the instruction's result to the value Y in Figure 2. During the depicted execution phase, we assume that K_5 predicts *equality* with high confidence, right after K_1 has committed and updated LCVT with its value (i.e., when the distance between K_5 and K_1 surpasses the length of the instruction window). In parallel, occurrences K_2 and K_3 (orange color) have executed, and they are sequencing toward Commit, while K_4 (blue color) is found at the Decode phase. Note that occurrences K_2 to K_4 have predicted *inequality*, since the value of K_1 was not yet written in the LCVT during their prediction.

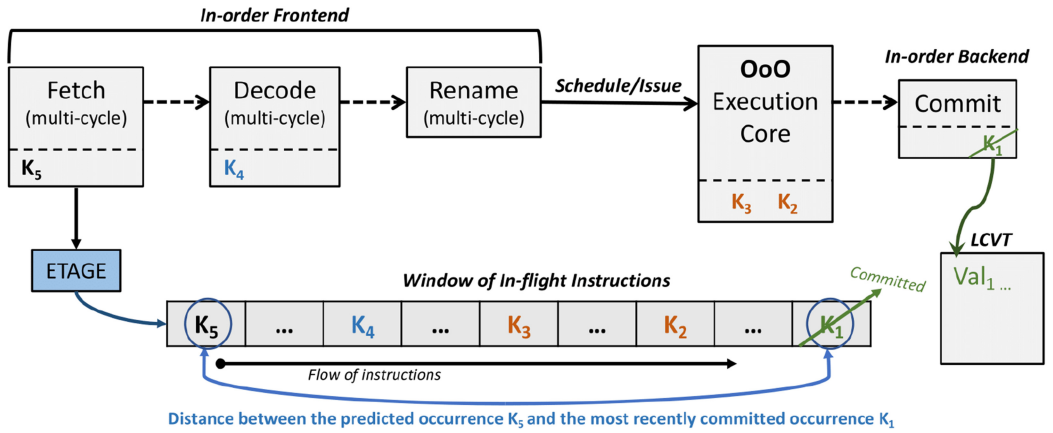


Fig. 3. Execution phase with a total of five occurrences of the instruction K in the window of in-flight instructions that are being processed simultaneously in different stages of the pipeline. K_5 has predicted high-confidence equality and will be performing value prediction using the committed value (Val_1) of occurrence K_1.

In summary, for instructions that exhibit interval equality, VTAGE and LVP need very long intervals to learn the equality (in the sense of reaching high confidence), and, as a result, predicted data are not sent to the pipeline. VTAGE is often able to learn of interval changes. Although VSEP also learns of interval changes, it also learns the paths leading to the first instruction occurrences in the interval, and when the value from LCVT is safe to be used in the pipeline. As it follows, VSEP generally predicts more occurrences than LVP and VTAGE on instructions featuring interval equality. Nonetheless, it should be pointed out that VTAGE can learn some other scenarios that VSEP is not able to track, even on instructions featuring interval equality, as will be shown in the performance analysis of Section 5. One of these scenarios is the case of repeatable short intervals (for instance of five occurrences) with distinct values (X0, . . . , Xk). Contrary to VSEP, VTAGE will be able to track these cases.

4 EVALUATION METHODOLOGY

4.1 Examined Predictors

Together with VSEP, we study the behavior of two context-based value predictors, namely, the classic LVP and the well-established model of VTAGE predictor. We use these two models as they have virtually set the fundamental principles of modern VP. Therefore, by comparing against these models, we first demonstrate how VP can generally become more efficient by leveraging *Equality Prediction*, and later, in Section 5.2, how a modern model such as VTAGE can seamlessly adapt and exploit EP. The more recent proposals that we mentioned in Section 2 can also adopt and exploit EP in a similar way to VTAGE. However, employing all of them to evaluate our technique is beyond the scope of this article; rather than advocating for a new value predictor that outperforms all previous works, our study's objective is to reveal that converting/adapting into predicting consecutive value equality is essential to further enhance VP performance. Note that the authors of this article have already submitted a model to the all-year *CVP* that combines previous proposals of the contest with EP and successfully placed top of the leaderboard [28].

From a structural viewpoint, the predictors we use in our evaluation have a storage budget in the 64KB range, i.e., a storage budget in the same range as the one of branch predictors in high-end processors. More precisely, our LVP predictor employs 4K entries for a total budget of

Table 1. Basic Layout and Size of the Examined Predictors

Predictor:	Structure and Storage Budget:
LVP [12]	4K-entry LVT, 13bit-tags, *40KB*
VTAGE [16]	4K-entry Base Component, *33,5KB* 6 x 512-entry tagged tables, tags 12+rank bits, *30,5KB* **Branch history lengths:** 2, 4, 8, 16, 32, 64
VSEP	**ETAGE:** 8K-entry Base Component, *4 KB* 13 x 1024-entry tagged tables, tags 8+rank bits, *32KB* **LCVT:** 3K entries, 3-way associative, 13bit-tags, *28KB* **Branch history lengths:** 2, 3, 4, 5, 6, 8, 11, 15, 20, 27, 36, 48, 64
Hybrid VSEP-VTAGE	**ETAGE:** 8K-entry Base Component, *4 KB* 13 x 512-entry tagged tables, tags 8+rank bits, *16KB* **LCVT:** 3K entries, 3-way associative, 13bit-tags, *28KB* **VTAGE Tagged tables:** 6 x 256-entry, tags 12+rank bits, *15KB* **Branch history lengths:** same as above for each respective predictor

Table 2. Applications Used in Our Evaluation

Benchmark Suite	Applications
SPEC2K6 INT/FP	bzip2, gamess, milc, gromacs, leslie3d, soplex, povray, hmmer, sjeng, GemsFDTD, libquantum, h264, omnetpp
SPEC2K17 INT/FP Rate	perlbench, mcf, cactuBSSN, parest, lbm, xalancbmk, deepsjeng, imagick, leela, nab, exchange2, fotonik3d, xz

40 KB. Doubling this size does not bring any significant benefit in our experiments. Then, we derive a proportional design of VTAGE with a 4K-entry base component, since this component is practically a tag-less LVP. VTAGE also includes 6 tagged tables of 512 entries each, amounting to a total budget of 64KB. In the implementation of LVP and VTAGE, we use 3-bit FPCs [23] controlled by the probability vector $V = \left\{1, \frac{1}{16}, \frac{1}{16}, \frac{1}{16}, \frac{1}{16}, \frac{1}{32}, \frac{1}{32}\right\}$ that we found to be performance-effective, i.e., same with the one used in the study that introduced VTAGE [16].

Accordingly, we design our ETAGE predictor to reach the same storage budget as VTAGE. The shortest and longest history lengths are 2 and 64, respectively, roughly following a geometric series.[4] We consider a predictor featuring 13 history components together with an 8K-entry tag-less table (i.e., a base component that basically holds *last equality* of instructions), that accounts for the 36KB of required storage. This, added with a 3K-entries LCVT (i.e., 28KB) completes the storage requirements of our design. Table 1 summarizes the design parameters and the size of all the predictors that we consider in our analysis. Note that the *Hybrid VSEP-VTAGE* predictor that is shown in Table 1 will be presented later in Section 5.2.

4.2 Benchmarks

We cast a wide range of workloads from the suites of SPEC2K6 [32] and SPEC2K17 [33] in order to expose as many value patterns as possible. For the workloads duplicated in both suites, we consider the ones from SPEC2K17. Table 2 lists the benchmarks considered in our study. To get relevant numbers, we identify a region of interest in the benchmarks using Simpoints 3.2 [20], as value prediction is highly sensitive to phase behavior. We simulate the resulting slice of 150M

[4]Each successive history length can be obtained by multiplying the previous one by 1.2~1.5 (see Table 1).

Table 3. Simulator Configuration Overview

Front-End	**Fetch through Rename width:** 8 insts/cycle **L1I:** 8-way, 32KB, 1 cycle, 128-entry ITLB, 64B fetch buffer **Branch Pred.:** State-of-the-art TAGE-SC-L [26] 64KB, 2-way 8K-entry BTB, 32-entry RAS, 20 cycles min. mis penalty.
Execution	**Issue through Commit width:** 8 insts/cycle **ROB/IQ/LQ/SQ:** 256/128/48/48 (Store To Load Forwarding (STLF) latency: 4 cycles) 256/256 INT/FP physical registers (4-bank PRF), 1K-SSID/LFST Store Sets [3], not rolled-back on squash and cleared every 30K access, 4ALU(1c), 1Mul/Div(3c/25c*), 2FP(3c), 2FPMulDiv(5c/10c*), 2Ld/Str Ports, 1Str Port, Full bypass
Memory Hierarchy	**L1D:** 4-way, 32KB, 4 cycles load-to-use, 64 MSHRs, 2 reads & 2 writes/cycle, 64-entry DTLB **L2:** Unified private, 8-way 256KB, 11 cycles, 64 MSHRs, no port constrains, Stream Prefetcher, (degree 1) **L3:** Unified shared, 16-way 2MB, 34 cycles, 64 MSHRs, no port constrains, Stream Prefetcher, (degree 1) All caches have 64B lines and LRU Replacement Policy **Memory:** Dual Channel DDR4-2400 (17-17-17) 2 ranks, 8banks/rank, 8K row-buffer, tREFI 7.8us, Across a 64B bus Min Read lat.: 75 cycles, Max.: 185 cycles.

*Not pipelined.

instructions by warming up the processor (caches, predictors) for 50M instructions and then collecting statistics for 100M instructions. For ease of reading, an extra bar was added to present the average value of the evaluation metrics we use. For this average, we present results as the arithmetic average across all workloads, except for the relative speedup where we calculate the geometric mean.

4.3 Simulator Framework

In our experiments, we use the *gem5* cycle-level simulator [2] implementing the x86_64 ISA. However, it should be noted that contrary to modern x86 implementations, *gem5-x86* does not support optimizations like *move elimination* [10, 22] and *μ-op fusion* [6]. Nevertheless, value prediction is orthogonal with most of these optimizations; therefore, readers should not assume that the speedup obtained through VP may be shadowed by them. Also, in the *gem5-x86* version used in our study, we have implemented branches as a single μ-op rather than three μ-ops to eliminate false dependencies that would not exist in a realistic implementation.

We model a relatively aggressive 4GHz, 8-issue superscalar baseline with a fetch-to-commit latency of 19 cycles. The in-order front-end, the out-of-order backend, as well as the in-order Commit stage of the pipeline are all properly dimensioned to support up to 8 μ-ops per cycle. We model a deep front-end that spans 15 cycles, accompanied by a fast backend of 4 cycles to obtain realistic branch/value misprediction penalties. Table 3 describes the details of the baseline pipeline structure we use. Since μ-ops are known at Fetch in *gem5*, all the widths given in Table 3 are in μ-ops, including the Fetch stage. Independent memory instructions (as predicted by the Store Set predictor [3]) are allowed to issue OoO and the entries in the Instruction Queue (IQ) are released upon issue.

When value prediction is employed, the corresponding predictor makes a prediction at *Fetch* for any μ-op that produces a register (i.e., not for branch or store instructions). To index the predictor, we XOR the instruction's PC-address left-shifted by one with the μ-op number inside the instruction. With this mechanism, we ensure that more than one μ-op of the same instruction will

generate a different index and therefore will have their own entry in the predictor. We assume that all the examined predictors can deliver *fetch-width* predictions by the *Fetch* stage. The predicted values with high confidence are written in the PRF at Rename time, and they are replaced by their non-speculative counterparts when they are regularly computed by the OoO execution engine. Also, a pre-commit stage responsible for validation/training (VT) is added to the pipeline. The VT stage lengthens the pipeline by one cycle, resulting in a value misprediction penalty of at least 21 cycles, while minimum branch misprediction latency remains unchanged. Note that predictors are effectively trained after commit, but values are read from the PRF during the VT stage.

5 EXPERIMENTAL RESULTS AND ANALYSIS

In this section, we demonstrate the strengths of VSEP. We compare our model to related prior work, the fundamental LVP and the state-of-the-art VTAGE, both implemented as described in Section 4.1. Overall, our evaluation shows that VSEP is partly orthogonal to traditional context-based VP techniques, regarding the instructions that it can predict (see *interval coverage* below). This leads us to define a hybrid predictor, that of VSEP and VTAGE as a unified design. We propose a practical implementation of this combination, and finally compare the hybrid predictor with VTAGE and VSEP alone.

5.1 VSEP vs. VTAGE and LVP

Figure 4 reports our simulation results by comparing the three different value predictors that we consider: LVP, VTAGE and VSEP. In particular, Figure 4(a) shows the relative amount of instructions per cycle (IPC) of the three variants normalized to the baseline. Figure 4(b) and (c) show the fraction of predicted dynamic instructions from those exposing either *uniform* or *interval* value equality. That is, *uniform coverage* (Figure 4(b)) describes the portion of instructions that exhibit uniform equality and their result was predicted, while *interval coverage* (Figure 4(c)) similarly expresses the predicted portion of the instructions that exhibit interval equality. These two metrics allow us to quantify dynamic instruction counts as a ratio of *consecutive value equality* (measured in static instruction counts in Figure 1) that the examined predictors can cover. However, they should not be confused with the general prediction coverage (depicted in Figure 6), which is defined as the total number of predicted dynamic instructions divided by the total number of dynamic instructions that are eligible for value prediction. Recall here that only high-confidence predictions are used in the pipeline.

Performance Analysis. Our evaluation indicates that VSEP benefits the same benchmarks and heavily competes with the state-of-the-art VTAGE. Specifically, it improves 14 out of 26 benchmarks by more than 1%, with a maximum speedup of 45% on *imagick* and an average speedup of 5.8%. In addition, VSEP succeeds in always obtaining equal or higher performance to the fundamental LVP. On several benchmarks, VSEP outperforms VTAGE, e.g., on *leslie3d* and on *GemsFDTD*. On the other hand, VTAGE outperforms VSEP on some other benchmarks, e.g., on *h264*, and *xz*. Finally, the two predictors achieve average performance in the same range.

Readers will notice that the reported speedup of VTAGE is lower than that of LVP for two applications, namely *leslie3d* and *GemsFDTD*, even though their two-fold coverage is in the same range. After a closer examination, we found that the potential performance gain of VTAGE for these benchmarks is limited by bursts of mispredictions associated with the interval transitions. This peculiar behavior may happen in VTAGE when applications expose very long value equality intervals. In Figure 5, we display a thorough classification of all the instructions exposing interval equality depending on the relevant interval length. The length of an interval is defined as the number of successive executions of a static instruction that produce the same result.

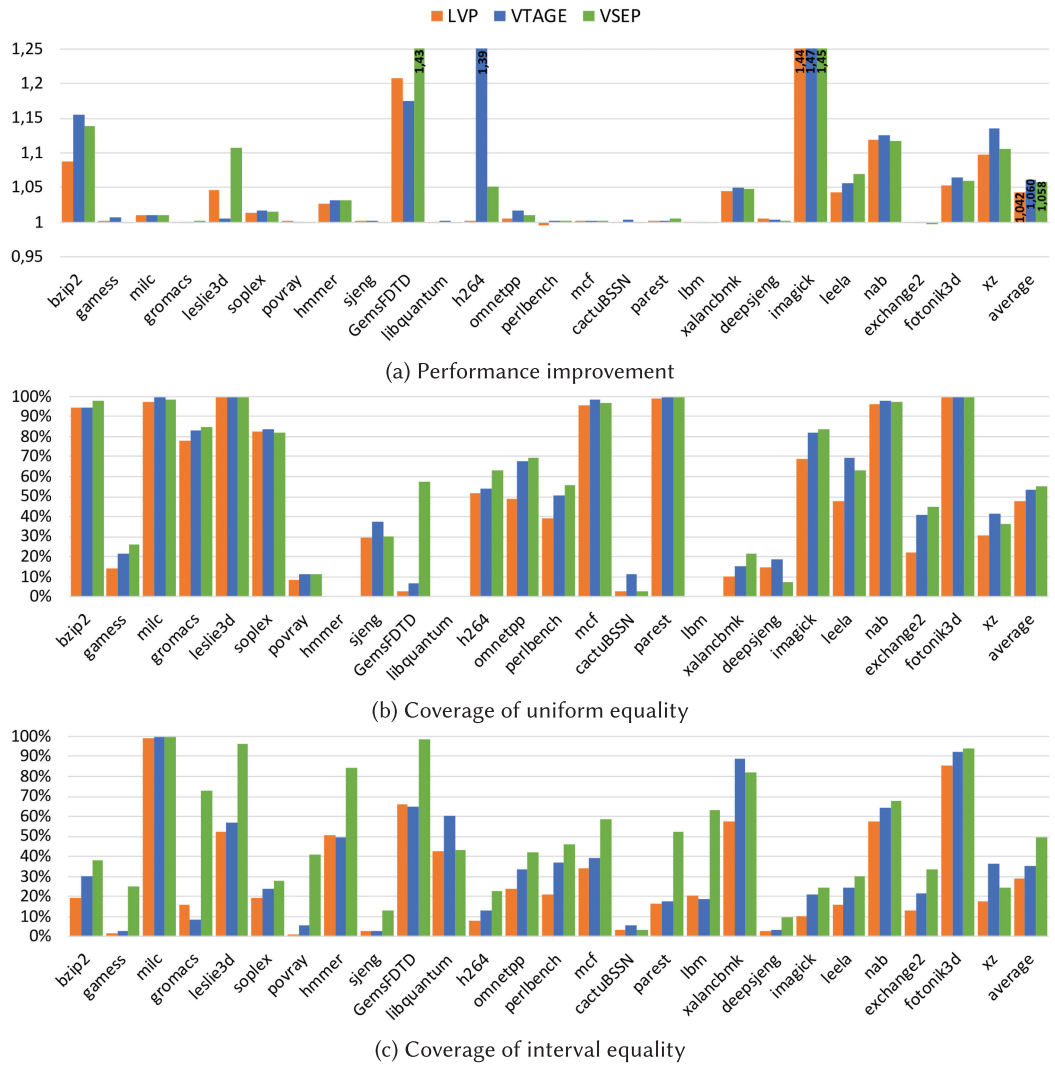


Fig. 4. Comparison of the three prediction schemes: LVP, VTAGE, and VSEP.

As we mentioned in Section 3.2, VTAGE can identify value switches without mispredicting when the correlation with the control-flow path is high enough. However, during very long intervals VTAGE may establish the confidence of the same recurrent value in several of its tagged-entries. Hence, upon an interval step, a burst of mispredictions will occur when these entries will be successively hit. In our simulations, both *leslie3d* and *GemsFDTD* suffer from this phenomenon, as the overwhelming majority of their *interval equalities* occur in intervals of more than 100 occurrences. Seznec experienced a similar anomaly in his VTAGE-like value predictor, EVES [27], and proposed the use of a prediction filter to defeat such bursts of mispredictions (no prediction, high-confidence or not, is promoted to the pipeline for 128 μ-ops after a misprediction).

On the contrary, such a prediction filter is not required by VSEP. Similar to VTAGE, ETAGE may also establish the confidence of *equality* in multiple entries during long intervals. When instruction results correlate with the control-flow history, ETAGE also identifies value switches by

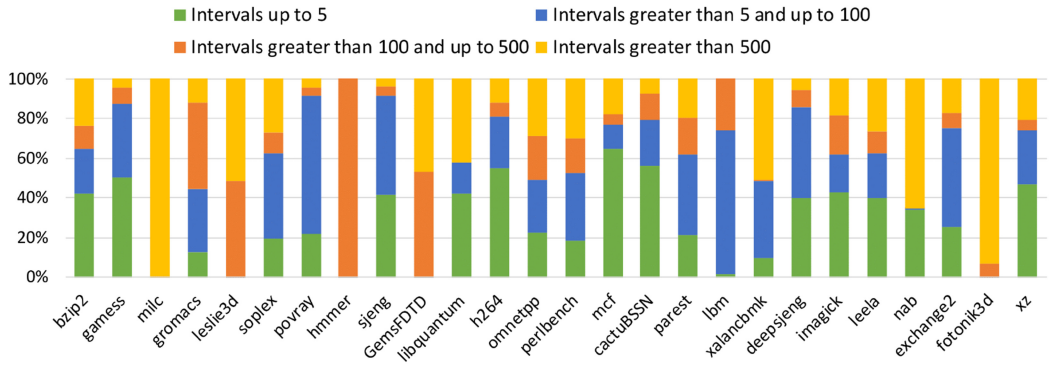


Fig. 5. Segmentation of instructions that exhibit interval equality in very short, moderate, and very long intervals.

matching with entries that either possess low confidence or indeed predict *inequality*. However, when ETAGE starts, respectively, hitting entries with already high-confidence *equality* from the previous interval, it allows VSEP to immediately re-enter prediction mode using the correct value retrieved from the LCVT. Therefore, due to its construction, VSEP not only captures instructions that expose this behavior, but also benefits highly by delivering significantly larger speedup in both aforementioned applications, i.e., in *leslie3d* and *GemsFDTD*.

Coverage Analysis. Figure 4(b) indicates that both VSEP and VTAGE achieve similar coverage of uniform equality of around 54% on average, which is marginally better than LVP that attains 48%. This behavior is expected since *uniform equality* is relatively simple to capture, even for predictors that do not leverage an elaborate context to distinct dynamic instructions (e.g., the global branch history used both by VTAGE and VSEP). Nonetheless, when it comes to the coverage of interval equality, Figure 4(c) illustrates that VSEP successfully surpasses both LVP and VTAGE on 22 out of 26 workloads, achieving 50% on average, versus 35% for VTAGE and 29% for LVP. As a matter of fact, the behavior of VSEP that we have detailed in Section 3 is verified by our experimental results, i.e., that VSEP is able to generally cover cases of interval equality that both LVP and VTAGE miss. In this way, VSEP is capable of complementing the established way that VP is performed.

Yet higher interval/uniform coverage for VSEP on many benchmarks does not always mean higher speedup. As in the case of *h264*, VTAGE significantly outperforms VSEP when it only has half of its interval coverage and similar uniform. In fact, the general prediction coverage of VTAGE is not composed only by the instructions that expose one of the two flavors of value equality that we have identified. In reality, the prediction range of VTAGE also includes independent patterns, e.g. sequences of repeatable strided values [25], that VSEP can not equally capture. For that reason, in Figure 6, we compare the general prediction coverage of the examined predictors, expressed in the two different types of consecutive value equality (i.e., *uniform* and *interval*) and also in any other independent value pattern that instruction results may expose. For each benchmark, we plot three different bars for our three predictors LVP, VTAGE, and VSEP from left to right, respectively. Readers should note that the numbers presented in Figure 6 refer to dynamic instructions, and, therefore, may be higher than those of Figure 1 representing static instruction counts.

As expected, LVP can not practically capture other value patterns, since it monitors instruction results in a PC-indexed table, and, thus, it can only recall constant or frequently constant values. This case is similar to VSEP; however, since it tracks previous values with a 3-way associative table (i.e., LCVT), it can predict a greater, yet marginal amount of instructions that follow value patterns not fitting in consecutive value equality. On the other hand, VTAGE is by construction able to

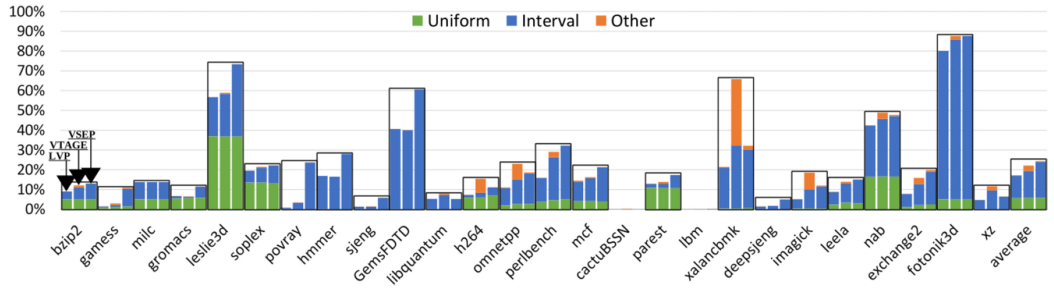


Fig. 6. Stacked representation of the general prediction coverage expressed in *uniform* and *interval* value equality and any other pattern. For each application and from left to right, the three bars represent LVP, VTAGE, and VSEP, respectively.

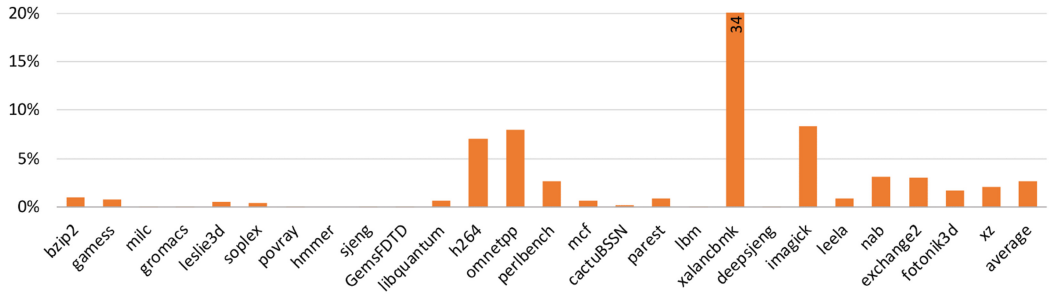
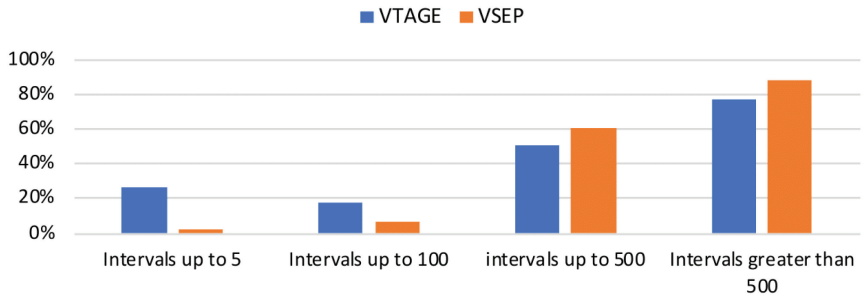


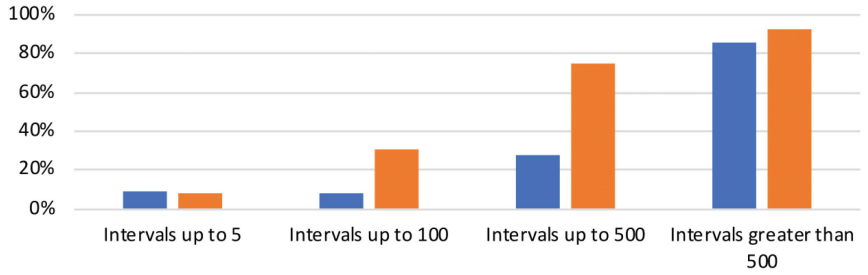
Fig. 7. Proportion of instructions that were predicted by VTAGE and do not exhibit either *uniform* or *interval* equality.

capture other patterns when they are short enough compared to the employed maximum history length (e.g., short strided patterns), with each value of the pattern residing in a different entry [16]. Therefore, in several applications, VTAGE achieves either a border-line but non-negligible (e.g., *bzip2*, *perlbench*, or *xz*) or a substantial (e.g., *h264*, *omnetpp*, *imagick*, and highest on *xalancbmk*) amount of predicted instructions that do not exhibit either type of consecutive value equality. By focusing more on this aspect, Figure 7 presents solely the fraction of these predicted instructions of VTAGE, which correspond to up to 34% and on average 3% of its global prediction coverage. As our analysis reveals, this additional coverage explains why VTAGE can dominate over VSEP in cases of similar or even lower uniform/interval coverage that were encountered previously in the performance comparison of Figure 4.

Moreover, VTAGE is occasionally able to capture some interval equality that is missed by VSEP, e.g., on *libquantum* and *xz*. By selectively focusing on the case of *xz*, where there is generally a noticeable performance improvement from VP, we present the synthesis of the interval coverage for VSEP and VTAGE on this specific benchmark in Figure 8(a). As we mentioned in Section 3.2, VTAGE can obtain higher interval coverage than VSEP when instructions rotate between repeatable short intervals. Unlike the binary-content VSEP, VTAGE is able to exploit the re-appearance of the same value, even in non-consecutive intervals, and continue enhancing the confidence of corresponding value-entries. Indeed, as illustrated in Figure 5, a significant portion of *consecutive value equalities* found in *xz* occur in intervals that are shorter than five occurrences. As reported by Figure 8(a), the coverage superiority of VTAGE is concentrated in these short intervals, where VTAGE achieves around 26% while VSEP can only reach 2%.



(a) Analysis of interval coverage explicitly for application *xz*



(b) Analysis of total interval coverage across all workloads

Fig. 8. Anatomy of interval coverage for VTAGE and VSEP, explicitly for application *xz* and totally for all the examined applications.

Finally, Figure 8(b) displays the segmented representation of the total interval coverage of VTAGE and VSEP, considering all applications listed in Table 2. Both schemes accomplish roughly equal levels of coverage for very short intervals, but VSEP far outweighs VTAGE in all the remaining situations. In particular, VSEP achieves very high rates of coverage in both moderate (i.e., up to 100 and 500 occurrences) and very long intervals (i.e., more than 500 occurrences), which corresponds to around 75% and more than 90%, respectively. On the other hand, VTAGE can merely approximate these rates only for very long intervals. Above all, this observation exposes the operational asset of VSEP over VTAGE under the state of interval value equality. More specifically, it confirms our thesis that the established VP methods cannot sufficiently capture interval equality unless intervals are very long.

Altogether, our performance evaluation suggests that both VSEP and the state-of-the-art VTAGE can individually obtain noticeable speedups, but none of the two can plainly outperform the other. As this event advocates for *hybridization*, in the following section, we present and evaluate a compound model of VSEP and VTAGE.

5.2 Combining VSEP with VTAGE

As verified by our experimental analysis, VSEP can accomplish meaningful performance improvements, either comparable or higher to an established value predictor such as VTAGE, by eliminating its essential weakness to make use of interval value equality. Our performance analysis also shows that VSEP and VTAGE can independently capture different cases. Since the two methods can be considered as *partly orthogonal* with respect to the cases they can capture, we study the combination of VSEP with a short-scale adaptation of VTAGE, in order to provide the best coverage and boost performance.

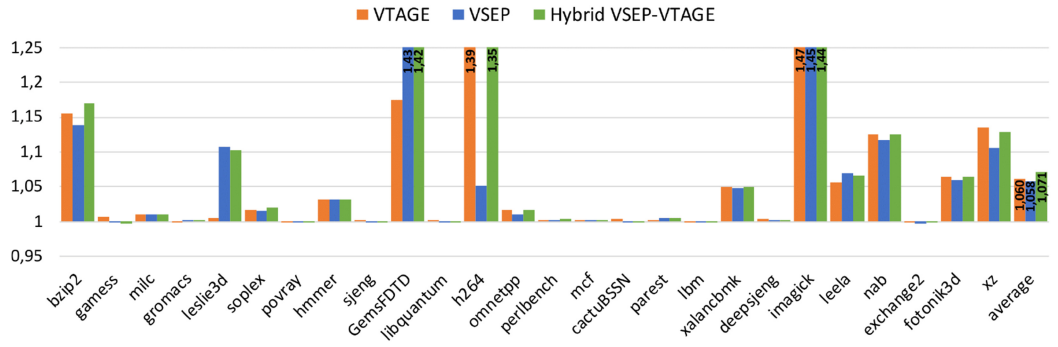


Fig. 9. Performance improvement of the compound model.

Hybrid VSEP-VTAGE employs the ETAGE equality predictor, accompanied with LCVT, and a moderately-sized VTAGE, which does not encompass its base component (i.e., a simple last-value table or LVT). The objective behind the balanced sizes of the two integrated predictors is to preserve their philosophy and to eliminate the systematic covering of the same cases. Hybrid VSEP-VTAGE works as follows: both predictor components are indexed in parallel early in the front-end. No elaborate or random chooser is necessary to clarify which predictor makes the final prediction. Simply, if *value equality* is predicted by ETAGE, the prediction of VSEP is the one that proceeds; otherwise, the prediction of VTAGE is considered for use to cover the cases missed from VSEP. For validation/update, each entry of the *validation queue* in VSEP (as described in Section 3.1) is augmented with the predicted value of VTAGE.

In Figure 9, we present the speedup brought by *Hybrid VSEP-VTAGE* in comparison with VTAGE and VSEP alone at similar storage budgets (see Table 1 for a size analysis of the hybrid model and the other predictors). As it appears, our hybrid solution can efficiently combine the two schemes by retaining their independent benefits. For any benchmark individually, the perceived speedup is in the range of the highest one between VSEP and VTAGE, and, on average, is augmented by 19% from VSEP/VTAGE alone, obtaining an overall average of 7.1%. Note that the extra storage budget required from the composite scheme (compared to a single VSEP predictor) solely consists of the additional entry required of the *validation queue* for the prediction of VTAGE. Hence, such meaningful performance improvements are acquired for minimum storage/complexity overhead.

In this section, we showed that Hybrid VSEP-VTAGE forms a smooth combination that overcomes the different limitations imposed by its two counterparts, since both predictors work complementary to each other. As such, *equality prediction* is a technique that can effectively enhance the performance gains that modern context-based VP-schemes can achieve. The use of equality prediction in VSEP is the one that we found the most cost-effective. In the next section, we shortly describe our initial proposition, POEP, that exploits the combination of *value prediction* with *physical register sharing* [10]. While relying on a more intuitive equality prediction concept, i.e., predicting the equality of the results of two strictly consecutive occurrences of an instruction, POEP was found to be much less cost-effective than VSEP.

6 COMBINING VALUE PREDICTION AND PHYSICAL REGISTER SHARING THROUGH EQUALITY PREDICTION IS NOT COST-EFFECTIVE

VSEP is based on predicting the equality of the result of the currently fetched occurrence of an instruction with the result of the last committed occurrence of the instruction. In practice, VSEP does not capture all opportunities to exploit interval equality. In VSEP, ETAGE does not directly

capture the equality of strictly consecutive occurrences of an instruction when their distance is shorter than the instruction window size. This can be visualized by revisiting the execution scenario depicted in Figure 3. In that particular example, we had assumed that the examined instruction *K* exposes interval value equality for an interval that begins from the occurrence K_1. K_5 is the first occurrence that predicts *equality*, right after the value Val_1 of K_1 has been written in LCVT. On the other hand, for occurrences K_2 to K_4, *inequality* is predicted because Val_1 was not available in LCVT at prediction time. However, they all produce the same value as their respective preceding occurrence Val_1. Nevertheless, predicting this "last" equality would still be very feasible.

In reality, when launching the VSEP study, our initial proposition was to exploit the equality of results for strictly consecutive instruction occurrences by combining two techniques that depended on whether other in-flight occurrences were present. In particular, if inflight occurrences exist, then the two consecutive occurrences share their result register, i.e., *physical register sharing* is performed, else the *last committed value* is used as the predicted result of the current occurrence, like in VSEP. We refer to this original proposition as POEP.

In POEP, an ETAGE-like predictor predicts the equality of results of consecutive occurrences of instructions. When equality is predicted with high confidence, if the previous occurrence of the instruction is still speculative (i.e., in-flight), then the result of the instruction is not predicted, but its target register is renamed as the one of the previous occurrence. This technique is known as *Physical Register Sharing* (RS) [10] and needs to be implemented in the register-renaming stage. To handle the cases where the previous instruction occurrence has committed, POEP implements an LCVT table just as VSEP. Therefore, when no occurrence of the instruction to be predicted is in-flight, the result of the previous occurrence sits in the corresponding entry of LCVT.

We simulated POEP with the same LCVT and ETAGE configurations as VSEP. The performance results that we obtained for POEP were only marginally better than those of VSEP, while the hardware complexity of POEP is much higher, as described below.

First, POEP requires the same hardware as VSEP (i.e., value prediction support in the OoO core and ETAGE and LCVT tables), but introduces important modifications to support physical register sharing. In general, Physical RS has been proposed for improving sequential performance by enabling various optimizations, like *move elimination* [10], *speculative memory bypassing* (SMB) [18], and other rename-based techniques [21, 22]. Sharing of physical registers between instructions is not trivial. In particular, the classical scheme for de-allocating a physical register is not working anymore. *Register reference counting* is needed and might be fairly complex to implement. Some recent works [18, 24] support that conventional register reference counting models of modern architectures can adapt for that purpose with limited cost. For instance, the Inflight Shared Register Buffer (ISRB) [18], one of the most up-to-date schemes to our knowledge, requires the use of a relatively small fully-associative buffer. But still, each instruction would need to perform a fully-associative search of this buffer both at Rename and at Commit. In the context where we consider RS, eligible instructions (i.e., those that have predicted high-confidence equality) would need to identify their previous in-flight occurrence (if any) in order to define their shared register. As a result, the instruction window would need to be PC-based fully-associative. Therefore, all predicted instructions would induce another complex fully-associative search, but in the in-flight instruction window, i.e., on a much larger structure.

Moreover, when employing physical register sharing in the context of a speculative use, validation is a major issue. The solution proposed in Ref. [18] necessitates the comparison of the shared register with the result of the instruction, i.e., the instruction needs to have an extra source-operand, the shared register, thus entailing additional read ports on the physical register file. Finally, the training of the equality predictor requires an extra read of LCVT at commit time, i.e.,

three accesses per instruction in LCVT (read at prediction time, read at commit time, and write at commit time) against only two accesses on VSEP.

As it appears, implementing POEP instead of VSEP would come together with a non-negligible storage and complexity overhead. Since only marginal performance gains can be obtained with POEP over VSEP, the latter is a much more cost-effective design.

7 CONCLUSION

Context-based value predictors represent an important class of value predictors [25]. They exploit the recurrent occurrences of the same result by a single instruction. Some instructions always deliver the same result and can be predicted by simple VP methods like LVP. Other instructions produce the same result on intervals. Overall, these instructions represent the most significant part of the predictions covered by state-of-the-art context-based predictors, such as VTAGE.

The VSEP predictor presented in this article was introduced to specifically target interval-style value equality, while covering the uniform category as well. Instead of predicting a wide 64-bit value, VSEP predicts some binary information, namely whether "*the instruction's current result is likely to be equal to its last committed value.*" Thus, unlike regular VP schemes that embody structures of 64-bit wide entries, VSEP features only a single Last Committed Value Table with full words (i.e., LCVT) and a binary-content equality predictor ETAGE. Compared to the state-of-the-art VTAGE, VSEP captures the two forms of value equality more efficiently. In practice, when it comes to interval equality, after a value switch from X to Y, VSEP is able to predict equality with high confidence as soon as the first occurrence of the new value Y has been committed. On the other hand, VTAGE has to reconstruct high confidence for each of the entries that has been predicting X. As a result, VSEP outperforms VTAGE on several applications. Nonetheless, VTAGE is able to predict value patterns that do not fit in consecutive occurrences and therefore evenly outperforms VSEP on various applications. In fact, the effective hybrid combination of VSEP and VTAGE introduced in this work leads to some extra performance gains by retaining their particularities.

Our proposition for the equality predictor has been inspired by the TAGE branch predictor. However, other options also could be considered and other types of equality predictors could be designed. For instance, other conditional branch predictors such as the family of perceptron-based predictors [8, 9] could be used. Finally, one could also consider the ETAGE equality predictor combined with the Omnipredictor [19], which may predict branches and memory dependencies within the same predictor.

ACKNOWLEDGMENTS

The authors would like to thank Dr. Anita Tino for helping to polish this article. This research was partially supported by an Intel research grant.

REFERENCES

[1] S. Bandishte, J. Gaur, Z. Sperber, L. Rappoport, A. Yoaz, and S. Subramoney. 2020. Focused value prediction. In *Proceedings of the ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA'20)*. 79–91. DOI: https://doi.org/10.1109/ISCA45697.2020.00018

[2] Nathan Binkert, Bradford Beckmann, Gabriel Black, Steven K. Reinhardt, Ali Saidi, Arkaprava Basu, Joel Hestness, Derek R. Hower, Tushar Krishna, Somayeh Sardashti, et al. 2011. The gem5 simulator. *ACM SIGARCH Computer Architecture News* 39, 2 (2011), 1–7.

[3] George Z. Chrysos and Joel S. Emer. 1998. Memory dependence prediction using store sets. In *Proceedings of the 25th Annual International Symposium on Computer Architecture (Cat. No.98CB36235)*. 142–153. DOI: https://doi.org/10.1109/ISCA.1998.694770

[4] Freddy Gabbay. 1996. *Speculative Execution Based on Value Prediction*. Technion-IIT, Department of Electrical Engineering.

[5] Freddy Gabbay and Avi Mendelson. 1998. Using value prediction to increase the power of speculative execution hardware. *ACM Trans. Comput. Syst.* 16, 3 (Aug. 1998), 234–270. DOI: https://doi.org/10.1145/290409.290411

[6] Simcha Gochman, Ronny Ronen, Ittai Anati, Avraham Berkovits, Tsvika Kurts, Alon Naveh, Amer Saeed, Zeev Sperber, and Raymond D. Valentine. 2003. The Intel Pentium M processor: Microarchitecture and performance.

[7] Bart Goeman, Hans Vandierendonck, and Koenraad De Bosschere. 2001. Differential FCM: Increasing value prediction accuracy by improving table usage efficiency. In *Proceedings of the HPCA 7th International Symposium on High-Performance Computer Architecture*. IEEE, 207–216.

[8] Daniel A. Jiménez. 2016. Multiperspective perceptron predictor. In *JWAC-5: Championship on Branch Prediction*. Retrieved from https://www.jilp.org/cbp2016/.

[9] Daniel A. Jiménez and Calvin Lin. 2001. Dynamic branch prediction with perceptrons. In *Proceedings of the HPCA 7th Int. Symp. High-Performance Computer Architecture*. 197–206. DOI: https://doi.org/10.1109/HPCA.2001.903263

[10] Stephan Jourdan, Ronny Ronen, Michael Bekerman, Bishara Shomar, and Adi Yoaz. 1998. A novel renaming scheme to exploit value temporal locality through physical register reuse and unification. In *Proceedings of the 31st Annual ACM/IEEE International Symposium on Microarchitecture*. IEEE Computer Society Press, 216–225.

[11] Kleovoulos Kalaitzidis and André Seznec. 2019. Value speculation through equality prediction. In *Proceedings of the IEEE 37th Int. Conf. Computer Design (ICCD)*. 694–697. DOI: https://doi.org/10.1109/ICCD46524.2019.00101

[12] Mikko H. Lipasti and John Paul Shen. 1996. Exceeding the dataflow limit via value prediction. In *Proceedings of the 29th Annual IEEE/ACM Int. Symp. Microarchitecture. MICRO 29*. 226–237. DOI: https://doi.org/10.1109/MICRO.1996.566464

[13] Mikko H. Lipasti, Christopher B. Wilkerson, and John Paul Shen. 1996. Value locality and load value prediction. *SIGPLAN Not.* 31, 9 (Sept. 1996), 138–147. DOI: https://doi.org/10.1145/248209.237173

[14] Lois Orosa, Rodolfo Azevedo, and Onur Mutlu. 2018. AVPP: Address-first value-next predictor with value prefetching for improving the efficiency of load value prediction. *ACM Trans. Archit. Code Optim.* 15, 4, Article 49 (Dec. 2018), 30 pages. DOI: https://doi.org/10.1145/3239567

[15] Arthur Perais and André Seznec. 2014. EOLE: Paving the way for an effective implementation of value prediction. In *Proceedings of the ACM/IEEE 41st Int. Symp. Computer Architecture (ISCA)*. 481–492. DOI: https://doi.org/10.1109/ISCA.2014.6853205

[16] Arthur Perais and André Seznec. 2014. Practical data value speculation for future high-end processors. In *Proceedings of the IEEE 20th International Symposium on High Performance Computer Architecture (HPCA'14)*. IEEE, 428–439.

[17] Arthur Perais and André Seznec. 2015. BeBoP: A cost effective predictor infrastructure for superscalar value prediction. In *Proceedings of the IEEE 21st Int. Symp. High Performance Computer Architecture (HPCA)*. 13–25. DOI: https://doi.org/10.1109/HPCA.2015.7056018

[18] Arthur Perais and André Seznec. 2016. Cost effective physical register sharing. In *Proceedings of the IEEE Int. Symp. High Performance Computer Architecture (HPCA)*. 694–706. DOI: https://doi.org/10.1109/HPCA.2016.7446105

[19] Arthur Perais and André Seznec. 2018. Cost effective speculation with the omnipredictor. In *Proceedings of the 27th International Conference on Parallel Architectures and Compilation Techniques (PACT'18)*. ACM, New York, NY, Article 25, 13 pages. DOI: https://doi.org/10.1145/3243176.3243208

[20] Erez Perelman, Greg Hamerly, Michael Van Biesbrouck, Timothy Sherwood, and Brad Calder. 2003. Using SimPoint for accurate and efficient simulation. *SIGMETRICS Perform. Eval. Rev.* 31, 1 (June 2003), 318–319. DOI: https://doi.org/10.1145/885651.781076

[21] Vlad Petric, Anne Bracy, and Amir Roth. 2002. Three extensions to register integration. In *Proceedings of the (MICRO-35) 35th Annual IEEE/ACM Int Symp. Microarchitecture*. 37–47. DOI: https://doi.org/10.1109/MICRO.2002.1176237

[22] Vlad Petric, Tingting Sha, and Amir Roth. 2005. RENO: A rename-based instruction optimizer. In *Proceedings of the 32nd Int. Symp. Computer Architecture (ISCA'05)*. 98–109. DOI: https://doi.org/10.1109/ISCA.2005.43

[23] Nicholas Riley and Craig Zilles. 2006. Probabilistic counter updates for predictor hysteresis and stratification. In *Proceedings of the 12th Int. Symp. High-Performance Computer Architecture*. 110–120. DOI: https://doi.org/10.1109/HPCA.2006.1598118

[24] Amir Roth. 2008. Physical register reference counting. *IEEE Computer Architecture Letters* 7, 1 (Jan. 2008), 9–12. DOI: https://doi.org/10.1109/L-CA.2007.15

[25] Yiannakis Sazeides and James E. Smith. 1997. The predictability of data values. In *Proceedings of the 30th Annual Int. Symp. Microarchitecture*. 248–258. DOI: https://doi.org/10.1109/MICRO.1997.645815

[26] André Seznec. 2016. TAGE-SC-L branch predictors again. In *JWAC-5: Championship on Branch Prediction*. Retrieved from https://www.jilp.org/cbp2016/.

[27] André Seznec. 2018. Exploring value prediction with the EVES predictor. In *First Championship Value Prediction, CVP-1 2018, Los Angeles, June 3, 2018*. Retrieved from https://www.microarch.org/cvp1/cvp1online/contestants.html.

[28] André Seznec and Kleovoulos Kalaitzidis. 2020. Exploring value prediction limits. In *the All-Year Championship on Value Prediction, CVP, February* (2020). Retrieved from https://www.microarch.org/cvp1/cvp1online/contestants.html.

[29] André Seznec and Pierre Michaud. 2006. A case for (partially) TAgged GEometric history length branch prediction. *Journal of Instruction Level Parallelism* 8 (2006), 1–23.

[30] Rami Sheikh, Harold W. Cain, and Raguram Damodaran. 2017. Load value prediction via path-based address prediction: Avoiding mispredictions due to conflicting stores. In *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO-50'17)*. ACM, New York, NY, 423–435. DOI: https://doi.org/10.1145/3123939.3123951

[31] Rami Sheikh and Derek Hower. 2019. Efficient load value prediction using multiple predictors and filters. In *Proceedings of the IEEE Int. Symp. High Performance Computer Architecture (HPCA)*. 454–465. DOI: https://doi.org/10.1109/HPCA.2019.00057

[32] Standard Performance Evaluation Corporation. 2006. The SPEC CPU 2006 benchmark suite. Retrieved from http://www.spec.org.

[33] Standard Performance Evaluation Corporation. 2017. The SPEC CPU 2017 benchmark suite. Retrieved from http://www.spec.org.

[34] B. Thwaites, G. Pekhimenko, H. Esmaeilzadeh, A. Yazdanbakhsh, J. Park, G. Mururu, O. Mutlu, and T. Mowry. 2014. Rollback-free value prediction with approximate loads. In *Proceedings of the 2014 23rd International Conference on Parallel Architecture and Compilation Techniques (PACT)*. 493–494.

[35] Amir Yazdanbakhsh, Gennady Pekhimenko, Bradley Thwaites, Hadi Esmaeilzadeh, Onur Mutlu, and Todd C. Mowry. 2016. RFVP: Rollback-free value prediction with safe-to-approximate loads. *ACM Trans. Archit. Code Optim.* 12, 4, Article 62 (Jan. 2016), 26 pages. DOI: https://doi.org/10.1145/2836168

[36] Huiyang Zhou, Jill Flanagan, and Thomas M. Conte. 2003. Detecting global stride locality in value streams. *SIGARCH Comput. Archit. News* 31, 2 (May 2003), 324–335. DOI: https://doi.org/10.1145/871656.859656

Received March 2020; revised November 2020; accepted November 2020