TAGE-SC for CBP2025

André Seznec
SiFive

ABSTRACT

The TAGE branch predictor has been adopted in many modern processor designs. TAGE and its derivative TAGE-SC-L are so far and at a fixed storage budget the most accurate branch predictors that have been published in the academic world. Recently, a "realistic" TAGE-SC predictor was presented.

The CBP2025 TAGE-SC is derived from CBP2016 TAGE-SC-L, and replicates most of the features that would prevent any reasonable direct hardware implementation: huge number of distinct tables, complete table interleaving in TAGE, use of local histories, unrealistic prediction latency, .. It features the new optimizations on allocation/replacement policy as well as the optimizations on the IMLI components in SC that were proposed for the "realistic" TAGE-SC.

On CBP2025 public traces, the submitted predictor achieves 3.363 MPKI and 143.935 CycWpPKI.

ADVERTISEMENTS

- The branch predictor presented in this note is derived from the version of TAGE-SC-L that was presented at CBP 2016 [10] and the "TAGE engineering cookbook" that was published in November 2024 [11].
- An implementation of the 64KB 2016 CBP TAGE-SC-L was provided in the CBP 2025 simulation framework. The branch prediction code is derived from this implementation and the artefact associated with [11].
- This note assumes that the reader is familiar with the previous literature on TAGE and TAGE-SC-L.

1 OUTLINE

The TAGE branch predictor (Figure 1) [6, 13] was introduced in 2006 and has been adopted in many modern processor designs [17]. TAGE and its derivative TAGE-SC-L [7–10, 14] are so far and at a fixed storage budget the most accurate branch predictors that have been published in the academic world. TAGE-SC-L (Figure 2) combines a TAGE predictor with SC, a neural branch predictor component [1] and a loop predictor [15].

However, CBP2016 TAGE-SC-L can not be directly implemented in hardware. In particular, it was designed to win the Championship. Among the severe limitations that forbid a direct hardware implementation of CBP2016 TAGE-SC-L, one can cite the unreasonable number of tables in the TAGE component of CBP2016 TAGE-SC-L, the unreasonable number of tables in the SC component , the unreasonable total prediction latency and the use of local history components in SC.

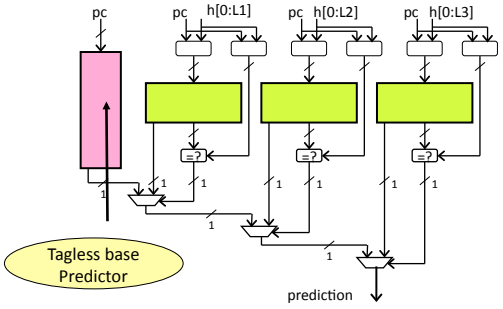

Figure 1: The TAGE predictor [13]

In [11], a "realistic" TAGE-SC predictor was presented [1]. "Realistic" meaning that the author estimated that it could be implemented for an aggressive instruction front-end predicting an instruction block with up to 4 branches (at most one taken) per cycle. "Realistic" TAGE-SC features a limited number of physical tagged tables (7) while it achieves the same accuracy as the TAGE component in CBP2016 TAGE-SC-L (30 tagged tables) on the CBP2016 traces. This is allowed by physical tagged tables sharing among the TAGE logical tables [11] as well as optimizations on the allocation/replacement policy for TAGE entries. The SC component uses a limited number of prediction counter tables (1 to 5) and reduces TAGE misprediction by up to 5 %. In particular, we redefined the IMLI component(s) [14]. Moreover on "realistic" TAGE-SC, the prediction latency is only one extra multiplexor longer than TAGE prediction (Fig 3). Simulations using the CBP 2016 framework showed that the proposed "realistic" TAGE-SC scales smoothly for storage budgets from 64Kbits range to 512Kbits storage range.

The CBP2025 TAGE-SC is derived from CBP2016 TAGE-SC-L, and replicates most of the features that would prevent any reasonable direct hardware implementation: huge number of distinct tables, complete table interleaving in TAGE, use of local histories, unrealistic prediction latency, .. It features the new optimizations on allocation/replacement policy on TAGE-SC proposed in [11] as well as the optimizations on the IMLI components in SC.

While 64KB CBP2016 TAGE-SC [2] achieves 3.750 MPKI on the CBP2025 traces, brute force scaling of CBP2016 TAGE-SC to $\tilde{1}92$KB is achieving 3.438 MPKI[3] and the 192KB CBP2025 TAGE achieves 3.363 MPKI[4]..

[1] The loop predictor was ignored. A loop predictor could be added to the design presented in [11], but would only allow to grab very marginal extra accuracy
[2] The loop predictor was disabled
[3] The CBP2025 organizers did a better job than me; my own brute force scaling was resulting in 3.496 MPKI.
[4] MPKI is measured on 50% of the instructions after warming

'CBP 2025', June 21, 2025, Tokyo, Japan
2025. ACM ISBN How.to.remove.this?(Not.ACM)...$15.00
https://doi.org/How.to.remove.this?(Not.ACM)

André Seznec
SiFive

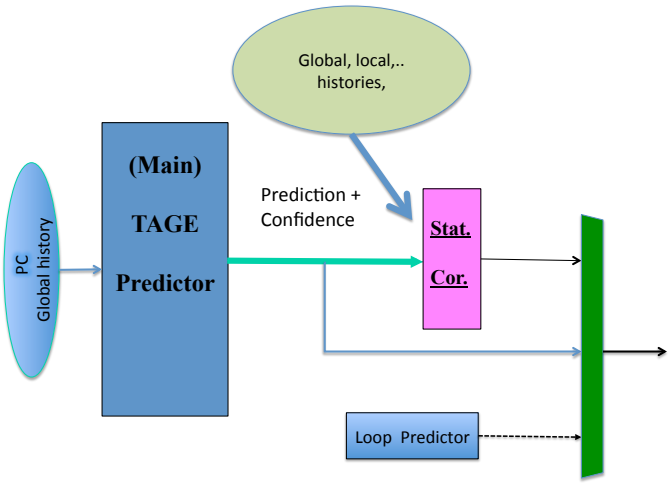

Figure 2: The TAGE-SC-L predictor: a TAGE predictor backed with a Statistical Corrector predictor and a loop predictor

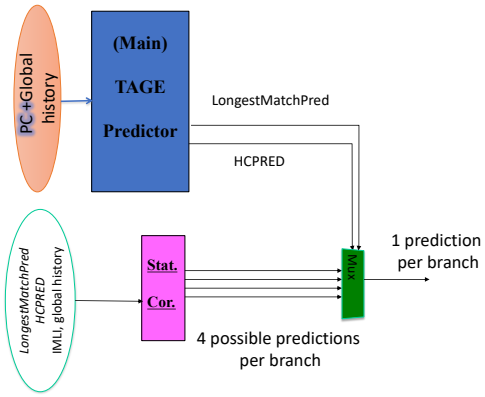

Figure 3: The "realistic" TAGE-SC: TAGE and SC are accessed in parallel, final prediction is selected through a 4-to-1 multiplexor

2 TAGE FEATURES

For a general introduction to TAGE principles, see [13].

2.1 General parameters

TAGE features an untagged direct mapped base table. For the fun, we use 3-bits counters in this table rather than 2-bit counters in previous TAGE propositions. We use low-order bits sharing among two adjacent entries [12]. The base table features 32Kentries for a total of 64Kbits.

The 192 KB budget allowed the use of 28 banks of 2K tagged entries (19 bits) shared among 28 logical tables. The global space is interleaved among the logical tables. Each bank is two-way skewed-associative. The high storage budget pushes to a large tag width (14 bits) to decrease aliasing on table entries[5]. High storage budget makes also 2-bit a better tradeoff for U counters than 1-bit.

In order to avoid path aliasing (i.e. impossibility to discriminate between two distinct paths due to limited representation), 5 bits are used in the history path per branch. Longest branch history is 1000, shortest history is 3. We do not fully respect the geometric series for the history lengths, but favor history lengths in the middle (see [11]).

[11] pointed out that using HCpred, the highest matching not low confidence prediction as the alternate prediction allows to shorten the critical path in TAGE prediction computation. We use this technique.

2.2 Optimizations on the TAGE allocation/replacement policy

Allocation/replacement in the TAGE predictor tagged tables occurs at predictor update time, typically at retirement [6]. Therefore allocation/replacement is not on the critical path for computing TAGE prediction.

The following optimizations are implemented on top of the classical TAGE allocation/replacement policy (see [11]).

- On a TAGE misprediction with a weak longest match counter, the alternate prediction entry is also updated.
- On a misprediction, several entries from different tables are allocated at the same time. By setting the U counter of the first allocated entry to 1, the entry is protected against replacement before the next smooth U resetting.
- We filter allocation based on the probability that the allocated entry will provide a correct prediction on its next access as the longest matching entry. To evaluate this probability, we use probabilistic counters [2] to evaluate the confidence in the prediction provided by the longest matching counter when it is weak. When the confidence is lower than 1/2, we allocate only with probability $\frac{1}{16}$, when the confidence is lower than 16/31, we allocate with probability 1/4 and when it is lower than 8/15 we allocate with probability 1/2. Moreover we increase the number of allocated entries when the confidence is high.
- We protect recently useful entries against fast eviction by setting its U counter directly to 2. The entry will survive a complete interval between two smooth U counter resetting.
- We use 2-way skewed associativity [4] on the tagged tables and reorganization at replacement/allocation (a la Elbow Cache [16] or Zcache [3]).

3 OPTIMIZATIONS ON THE STATISTICAL CORRECTOR

In the SC component for CBP2025 TAGE-SC predictor, we essentially use the SC components that are proposed in [11] and a few local history based components, two global history GEHL-based components [5] and a few other tables playing with different outputs of the TAGE predictor (confidence, longest hitting bank number, base prediction, ..) . These later components bring very marginal

[5]For smaller storage budgets, one should consider to decrease the number of logical tables to decrease the aliasing probability and allows using a shorter tag.
[6]The CBP2025 framework implements it at branch resolution time !!

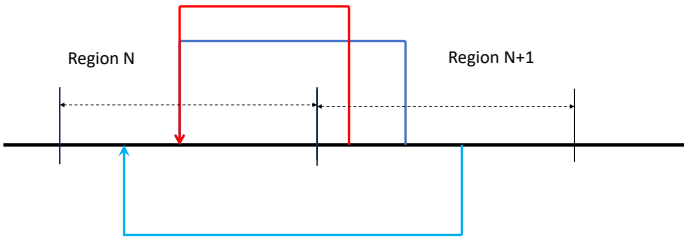

Figure 4: BrIMLI counts the number of consecutive taken backward branches within the same region.

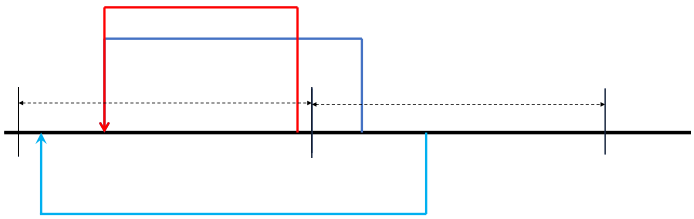

Figure 5: TaIMLI counts the number of consecutive taken backward branches with their target in the same region.

accuracy benefits and represent in total less than 1 Kentries. All counter tables in SC implement 6-bit counters.

3.1 The "TAGE-SC cookbook" components

3.1.1 SC tables using only PC and TAGE outputs.

- 2 2Kentries tables indexed respectively with PC and (PC xor (HCPRED,LongestMatchPred))
- a 4-entry table indexed with (HCPRED,LongestMatchPred).

3.1.2 Other SC tables. In [11], more refined forms of IMLI [14] were introduced. Two IMLI-based tables are used, respectively indexed with the BrIMLI and the TaIMLI counters. Region "branch" IMLI is illustrated in Figure 4. The BrIMLI counter counts the number of consecutive backward taken branches within the same region (64 bytes in our proposition). The counter is reset on the first taken backward branch that does not belong to the region. Region "target" IMLI is illustrated in Figure 5. The TaIMLI counter counts the number of consecutive backward taken branches with target in the same region (64 bytes in our proposition). The counter is reset on the first taken backward branch which target that does not belong to the region. TaIMLI and BrIMLI allows to track branches which outputs are correlated with their values. Single entry/single exit innermost loops are covered, but also loops with multiple exit points, nested loops, etc.

For CBP2025 TAGE-SC, we double the IMLI-based mechanism by adding another pair of tables, with region size being only 4 bytes, respectively the exact same branch address and the exact same branch target: benefit is marginal over using twice as large IMLi-based tables. The four tables have each 2 Kentries.

[11] also used two tables indexed with the "forward path" and "backward path". The "forward path" is the path of taken forward branches. The "backward path" is the path of taken backward branches where loops are eliminated.

3.2 Other global history based tables

CBP2025 TAGE-SC implement two global history based components based on the GEHL predictor [5]. A first component is indexed using the PC and the global history. The second component uses the PC xored with (LongestMatchPred,HCPRED) and global history. For CBP2025, the components feature respectively 5 and 4 2Kentry tables.

We add a 1Kentry table indexed by the history of the 256B block encountered in the path. This component has a very marginal benefit (0.004 MPKI) and was essentially added to spend part of the remaining storage budget.

3.3 Local histories SC tables

Implementing local history components in real hardware would be very difficult, since managing speculative local history is very difficult in an out-of-order execution processor.

For the CBP2025, the framework allows to checkpoint the local history at no accounted cost in the storage budget. Therefore, CBP2025 TAGE-SC implements several local history components with different number of histories and counter tables. One of these local history components is concatenating 2 bits of global history in addition of the current branch direction. Another one is indexed by the index of the basic block and accumulates all the histories of the branches in the basic block in the history.

The overall accuracy benefit brought by the use of local history components is modest (about 3%) and would not justify the complexity of managing speculative local history.

3.4 Miscelleanous optimizations on SC

The overall contribution of local history components (respectively the IMLI components) is emphasized/deemphasized through multiplicative factors that are updated at execution. The multiplicative factors are increased when the contribution was allowing a correct prediction and decreased otherwise.

The dynamic threshold fitting derived from the GEHL predictor [5] is optimized. The threshold(s) is only updated when the SC sum is higher than half of the threshold.

4 ACCURACY EVALUATION

We report here only the arithmetic mean of MPKI on the distributed traces for different configurations:

- **Full CBP2025 TAGE-SC: 3.363 MPKI**
- TAGE only: 3.781 MPKI
- TAGE + SC limited to the use of TAGE outputs: 3.612 MPKI
- TAGE-SC without IMLI and without local history: 3.522 MPKI
- TAGE-SC without local history: 3.472 MPKI

As already mentioned, the accuracy benefit coming from using local history is limited (3.1%). We also report the accuracy for scaled size CBP2025 TAGE-SC[7], respectively for 2x, 3.252 MPKI, for 0.5x, 3.532 MPKI and for 0.25x, 3.734 MPKI.

CBP2016 TAGE-SC (64KB) is achieving 3.751 MPKI and scaled 192KB CBP2016 TAGE-SC is achieving 3.438 MPKI. On this benchmark set, CBP2025 TAGE-SC achieves 2% lower misprediction rate better than the 192 KB CBP2016 TAGE-SC.

Through selectively deactivating the SC components on the 192KB CBP2016 TAGE-SC, we found that the CBP2025 TAGE component (enhanced by optimizations on allocation/replacement policy) is outperformed by the 192KB CBP2016 TAGE component provided by the CBP2025 organizer after the submission (3.781 MPKI vs 3.715 MPKI). This loss (unexpected by the author) is due to the use of a huge bimodal table in 192 KB CBP2016 TAGE-SC and the use of wider tags. The most spectacular benefit on CBP2025 TAGE-SC comes from the better implementation of the SC parts using only TAGE outputs. The benefit over TAGE only is 0.169 MPKI versus 0.056 MPKI for the 192KB CBP2016 TAGE-SC. CBP2016 TAGE-SC was relying on SC global history components to capture this behavior. Therefore the contribution of global history SC components is smaller on CBP2025 TAGE-SC (0.90 MPKI vs 0.103 MPKI). The reenginered IMLI component also brings a significant part of the benefit (0.050 MPKI instead of 0.013 MPKI for the 192 KB CBP2016 TAGE-SC). This new form of IMLI is able to correctly predict most loops with constant number of iterations. The contributions of the SC local history components are in the same range (0.109 MPKI vs 0.107 MPKI) despite a much larger size on CBP 2025 TAGE-SC.

5 NO LOOP PREDICTOR

The CBP2025 framework includes a loop predictor model. We do not include this loop predictor in the final CBP2025 TAGE-SC predictor: its accuracy gain is marginal (0.003 MPKI vs 0.010 MPKI for 192 KB CBP2016 TAGE-SC-L). In practice, the new version of IMLI components capture most of the benefits that could bring a loop predictor. Moreover hardware implementation of a loop predictor could be quite tricky. For instance, the model provided by the CBP 2025 organizers does not completely respect the rules of the competition.

6 SUMMARY

CBP2025 TAGE-SC has been defined based on the "realistic" TAGE-SC predictor presented in [11]. It outperforms the scaled version of CBP2016 TAGE-SC, the winner of the previous championship [10].

The championship rules pushed to use some unrealistic features in order to be a strong competitor in the championship. The author does not recommend the presented design for effective hardware implementation: the number of tables has be kept relatively small, the use of local history components should be avoided and the prediction computation latency has to be limited to a few cycles.

However competitively accurate TAGE-SC predictors can be realistically implemented in hardware as illustrated in [11].

[7]Sizes of the TAGE tables and the SC counter tables are scaled, not the history tables

ACKNOWLEDGMENTS

The author thanks the organizers for providing an opportunity to promote TAGE-SC. He thanks SiFive for allowing him to compete and INRIA/IRISA where he has continuously worked on branch prediction related topics for nearly 30 years.

REFERENCES

[1] D.A. Jimenéz and C. Lin. 2002. Neural Methods for Dynamic Branch Prediction. *ACM Transactions on Computer Systems* 20, 4 (Nov. 2002).
[2] Nicholas Riley and Craig B. Zilles. 2006. Probabilistic counter updates for predictor hysteresis and stratification. In *12th International Symposium on High-Performance Computer Architecture, HPCA-12 2006, Austin, Texas, USA, February 11-15, 2006*. IEEE Computer Society, 110–120. https://doi.org/10.1109/HPCA.2006.1598118
[3] Daniel Sánchez and Christos Kozyrakis. 2010. The ZCache: Decoupling Ways and Associativity. In *43rd Annual IEEE/ACM International Symposium on Microarchitecture, MICRO 2010, 4-8 December 2010, Atlanta, Georgia, USA*. IEEE Computer Society, 187–198. https://doi.org/10.1109/MICRO.2010.20
[4] A. Seznec. 1993. A Case for Two-Way Skewed-Associative Caches. In *Proceedings of the 20th Annual International Symposium on Computer Architecture.*
[5] A. Seznec. 2005. Analysis of the O-GEHL branch predictor. In *Proceedings of the 32nd Annual International Symposium on Computer Architecture.*
[6] André Seznec. 2007. The L-TAGE branch predictor. *Journal of Instruction Level Parallelism (http://wwwjilp.org/vol9)* (May 2007). http://www.jilp.org/vol9
[7] André Seznec. 2011. A 64 Kbytes ISL-TAGE branch predictor. 3rd Championship Branch Prediction. In *Proceedings of the 3rd Championship Branch Prediction.*
[8] André Seznec. 2011. A new case for the TAGE branch predictor. In *Proceedings of the MICR0 44.*
[9] André Seznec. 2014. TAGE-SC-L branch predictors. In *Proceedings of the 4th Championship on Branch Prediction, http://www.jilp.org/cbp2014/.*
[10] André Seznec. 2016. TAGE-SC-L branch predictor again. 5th Championship Branch Prediction. In *Proceedings of the 5th Championship Branch Prediction.*
[11] André Seznec. 2024. *TAGE: an engineering cookbook.* Technical Report 9561. Inria. 1–73 pages. https://hal.science/hal-04804900
[12] A. Seznec, S. Felix, V. Krishnan, and Y. Sazeides. 2002. Design tradeoffs for the Alpha EV8 conditional branch predictor. In *Proceedings of the 29th Annual International Symposium on Computer Architecture.*
[13] André Seznec and Pierre Michaud. 2006. A case for (partially)-tagged geometric history length predictors. *Journal of Instruction Level Parallelism (http://www.jilp.org/vol8)* (April 2006). http://www.jilp.org/vol8
[14] André Seznec, Joshua San Miguel, and Jorge Albericio. 2015. The inner most loop iteration counter: a new dimension in branch history. In *Proceedings of the 48th International Symposium on Microarchitecture, MICRO 2015, Waikiki, HI, USA, December 5-9, 2015*. 347–357. https://doi.org/10.1145/2830772.2830831
[15] Timothy Sherwood and Brad Calder. 2000. Loop Termination Prediction. In *High Performance Computing, Third International Symposium, ISHPC 2000, Tokyo, Japan, October 16-18, 2000. Proceedings (Lecture Notes in Computer Science)*, Mateo Valero, Kazuki Joe, Masaru Kitsuregawa, and Hidehiko Tanaka (Eds.), Vol. 1940. Springer, 73–87. https://doi.org/10.1007/3-540-39999-2_8
[16] Mathias Spjuth, Martin Karlsson, and Erik Hagersten. 2005. Skewed Caches from a Low-Power Perspective. In *Proceedings of Computing Frontiers.* Ischia, Italy.
[17] Hosein Yavarzadeh, Mohammadkazem Taram, Shravan Narayan, Deian Stefan, and Dean M. Tullsen. 2023. Half&Half: Demystifying Intel's Directional Branch Predictors for Fast, Secure Partitioned Execution. In *44th IEEE Symposium on Security and Privacy, SP 2023, San Francisco, CA, USA, May 21-25, 2023*. IEEE, 1220–1237. https://doi.org/10.1109/SP46215.2023.10179415

A COST ANALYSIS

We present the storage budget including all miscelleanous counters and a single instance of the different histories that are used for indexing the different tables.

- TAGE and its management: 1169710 bits (28 2K 19-bit entry banks for the tagged tables, 32Kbits for the base table, 5000

bits of global history and 9638 bits for the counter tables used update management)

- SC components using TAGE outputs: 29531 bits (2 K 6-bit entry tables + 4 6-bit entry tables with respectively 512, 128, 64 and 4 entries, 65 counters for managing update thresholds)
- SC components using various forms of global history: 141398 bits (11 2K 6-bit entry tables + 1 1K 6-bit entry table + 86 bits for the different global histories)
- SC components using IMLI counters: 49628 bits (4 2K 6-bit entry tables + 64 multiplicative 6-bit factors + 104 bits for monitoring the IMLI counters)
- SC components using local history: 177840 bits (14 2K 6-bit entry tables + 64 multiplicative 6-bit factors + 5424 bits of local history)

The total is **1568807** bits, i.e. approximately 1531 Kbits, i.e less than 192 Kbytes.

Large storage budget modifies the storage portions to be used respectively for TAGE and SC. In CBP2025 TAGE-SC, these ratios are respectively 74 % for TAGE and 26% for SC, while on CBP2025 TAGE-SC they were 91% and 9%.

The volume of information that has to be checkpointed for each branch is huge, particularly for local history.

- For local history SC components: 5424 bits
- For IMLI SC components: 104 bits
- For global history SC components: 86 bits
- For TAGE: only a 13-bit pointer on the global history circular buffer has to be checkpointed.