TAGE: an engineering cookbook

André Seznec

TAGE: an engineering cookbook

André Seznec

RESEARCH REPORT
N° 9561
Novembre 2024
Project-Teams PACAP

ISSN 0249-6399 ISRN INRIA/RR--9561--FR+ENG

TAGE: an engineering cookbook

André Seznec

Project-Teams PACAP

Research Report n° 9561 — Novembre 2024 — 4 pages

Abstract: CBP2016 TAGE-SC-L is generally considered as the state-of-the-art of branch predictors that have been been proposed in the academic world. This proposition suffers from several drawbacks, that forbids its direct implementation in hardware. However TAGE has been implemented by industry in many processor cores. This report presents a set of tradeoffs that could be used in an effective hardware implementation of TAGE-SC. This proposed implementation would still achieve state-of-the-art branch prediction accuracy.

Key-words: Branch prediction

RESEARCH CENTRE
Centre Inria de l'Université de Rennes

Campus universitaire de Beaulieu
Avenue du Général Leclerc
35042 Rennes Cedex

TAGE: livre de recettes pour l'ingénieur

Résumé :

Mots-clés : Prédiction de branchement

Advertisements

- This note motivates the associated slide set that is incorporated as an annex.
- A branch prediction simulator is associated as an artefact with this note [1]. See slide 2.
- This note assumes that the reader is (very) familiar with the previous literature on TAGE and TAGE-SC-L.

1 Outline

The TAGE branch predictor [9, 3] was introduced in 2006 and has been adopted in many modern processor designs [12]. TAGE and its derivative TAGE-SC-L [5, 4, 6, 10, 7] 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 combines a TAGE predictor with SC, a neural branch predictor component [1] and a loop predictor [11].

However, TAGE and, particularly the TAGE-SC-L version as presented at the Championship Branch Prediction in 2016 are not realistic for hardware implementations. In particular, TAGE-SC-L 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.

In the associated slide set,, we present how a TAGE-SC predictor [2] could be designed for an aggressive instruction front-end predicting an instruction block with up to 4 branches (at most one taken) per cycle. The presented predictor scales smoothly for storage budgets from 64Kbits range to 512Kbits storage range.

The "pure" TAGE predictor that we illustrate uses 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).

While the prediction latency of CBP2016 TAGE-SC-L consists in the TAGE latency plus the SC latency, the prediction latency for the presented TAGE-SC is only marginally higher than the TAGE prediction latency (a extra 2x2 multiplexor). We present three possible versions for the SC component which respectively use only 1, 2, and 5 prediction counter tables and enhancing TAGE accuracy by respectively 2 %, 3.5 % and 5 %. Local branch history is not used.

At the same 512Kbits storage range, the accuracy of the presented TAGE-SC using 7 physical tagged tables and 5 prediction counter tables is within 1 % of the accuracy of the CBP2016 TAGE-SC-L (which features 30 tagged tables, 20 prediction counter tables and uses local history).

Ahead pipelining [8, 2] was proposed to address the latency issue on instruction address generators. We illustrate that the proposed TAGE-SC can be 3-block ahead pipelined with limited accuracy loss (4-6 %) compared with 1-block ahead TAGE-SC predictor, and this for instruction blocks featuring up to 4 branches.

[1] The author apologizes for the old C coding style that he has been using all along his research carreer

[2] We ignore the loop predictor. A loop predictor could be added to the design presented in the note, but would only allow to grab very marginal extra accuracy

References

[1] D.A. Jimenéz and C. Lin. Neural methods for dynamic branch prediction. *ACM Transactions on Computer Systems*, 20(4), November 2002.

[2] A. Seznec and A. Fraboulet. Effective ahead pipelining of the instruction addres generator. In *Proceedings of the 30th Annual International Symposium on Computer Architecture*, June 2003.

[3] André Seznec. The L-TAGE branch predictor. *Journal of Instruction Level Parallelism (http://wwwjilp.org/vol9)*, May 2007.

[4] André Seznec. A 64 kbytes ISL-TAGE branch predictor. In *Proceedings of the 3rd Championship Branch Prediction*, June 2011.

[5] André Seznec. A new case for the tage branch predictor. In *Proceedings of the MICR0 44*, 2011.

[6] André Seznec. Tage-sc-l branch predictors. In *Proceedings of the 4th Championship on Branch Prediction, http://www.jilp.org/cbp2014/*, 2014.

[7] André Seznec. TAGE-SC-L branch predictor again. In *Proceedings of the 5th Championship Branch Prediction*, June 2016.

[8] André Seznec, Stéphan Jourdan, Pascal Sainrat, and Pierre Michaud. Multiple-block ahead branch predictors. In *Architectural Support for Programming Languages and Operating Systems (ASPLOS-VII)*, pages 116–127, 1996.

[9] André Seznec and Pierre Michaud. A case for (partially)-tagged geometric history length predictors. *Journal of Instruction Level Parallelism (http://www.jilp.org/vol8)*, April 2006.

[10] André Seznec, Joshua San Miguel, and Jorge Albericio. 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*, pages 347–357, 2015.

[11] Timothy Sherwood and Brad Calder. Loop termination prediction. In Mateo Valero, Kazuki Joe, Masaru Kitsuregawa, and Hidehiko Tanaka, editors, *High Performance Computing, Third International Symposium, ISHPC 2000, Tokyo, Japan, October 16-18, 2000. Proceedings*, volume 1940 of *Lecture Notes in Computer Science*, pages 73–87. Springer, 2000.

[12] Hosein Yavarzadeh, Mohammadkazem Taram, Shravan Narayan, Deian Stefan, and Dean M. Tullsen. 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*, pages 1220–1237. IEEE, 2023.

RESEARCH CENTRE
Centre Inria de l'Université de Rennes

Campus universitaire de Beaulieu
Avenue du Général Leclerc
35042 Rennes Cedex

Publisher
Inria
Domaine de Voluceau - Rocquencourt
BP 105 - 78153 Le Chesnay Cedex
inria.fr

ISSN 0249-6399

TAGE:
an engineering cookbook

André Seznec

November 2024

INRIA/IRISA

Advertisement

- This slide set assumes that the reader is familiar with the literature on the TAGE predictor.
  - A. Seznec, P. Michaud, " A case for (partially) tagged Geometric History Length Branch Prediction", Journal of Instruction Level Parallelism , Feb. 2006
  - A. Seznec "The L-TAGE predictor", Journal of Instruction Level Parallelism, May 2007
  - A. Seznec, J. San Miguel, J. Albericcio "The Inner Most Loop Iteration counter: a new dimension in branch history", Micro 2015, December 2015
  - A.Seznec, "TAGE-SC-L branch predictors", 4th Championship Branch Prediction, June 2014,
  - A. Seznec, TAGE-SC-L Branch Predictors Again. *5th JILP Championship Branch Prediction (CBP-5)*, June 2016

- Presented simulations are reproducible using CBP 2016 framework, http://www.jilp.org/cbp2016/ and the simulator code available at https://files.inria.fr/pacap/seznec/TageCookBook/predictor.h

TAGE:
TAgged GEometric history length predictor

- Was introduced in 2006
- TAGE (or derivative) is used in most high performance cores
- TAGE-SC-L is the « best » academic branch predictor
  - ➔ « best »:
    - For a given storage budget, it achieves the best accuracy on a set of publiccaly available « representative » traces

This slide set:

- Point out the difficulties of effective hardware implementation of TAGE and TAGE-SC-L

- Present implementable solutions:
  - ➔ With « reasonable » number of tables
  - ➔ With «reasonable » prediction latency
  - ➔ That do not sacrifice accuracy
    - + 1% MPKI compared with TAGE-SC-L 2016

➢ All simulated predictors in 512Kbits storage range

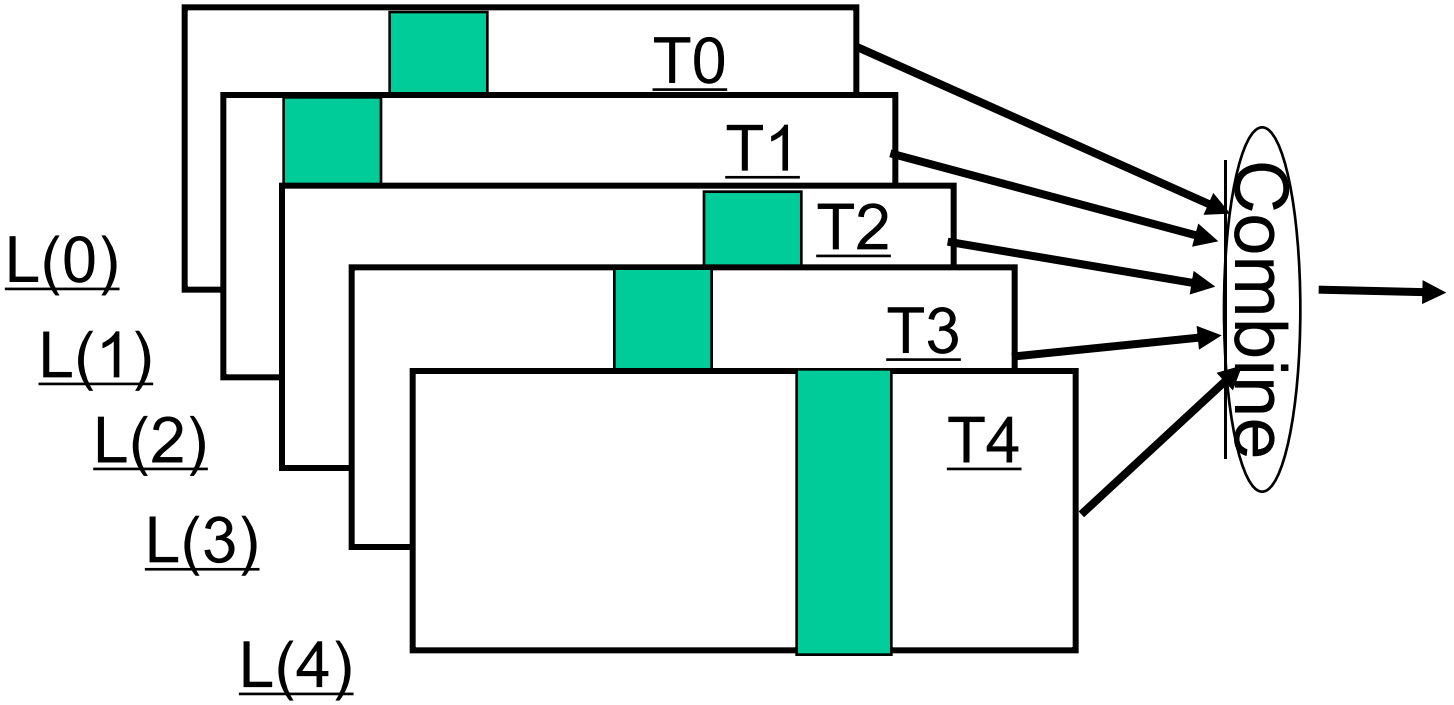
- Organization:
  - « academic » TAGE and TAGE-SC-L
  - Instruction front-end assumptions
  - Dealing with « pure » TAGE implementation
  - Dealing with TAGE-SC implementation
  - 3 Block ahead TAGE-SC prediction

« Academic » TAGE and TAGE-SC-L

GEOMETRIC HISTORY LENGTH PREDICTOR

CBP 2004

A Multiple length global history predictor



With a limited number of tables

Underlying idea

- H and H' two history vectors equal on N bits, but differ on bit N+1
  - e.g. $L(1) \leq N < L(2)$
- Branches (A,H) and (A,H')

  biased in opposite directions

Table T2 should allow to discriminate between (A,H) and (A,H')

GEometric History Length predictor

The set of history lengths forms a geometric series

$$L(0) = 0$$

$$L(i) = \alpha^{i-1} L(1)$$

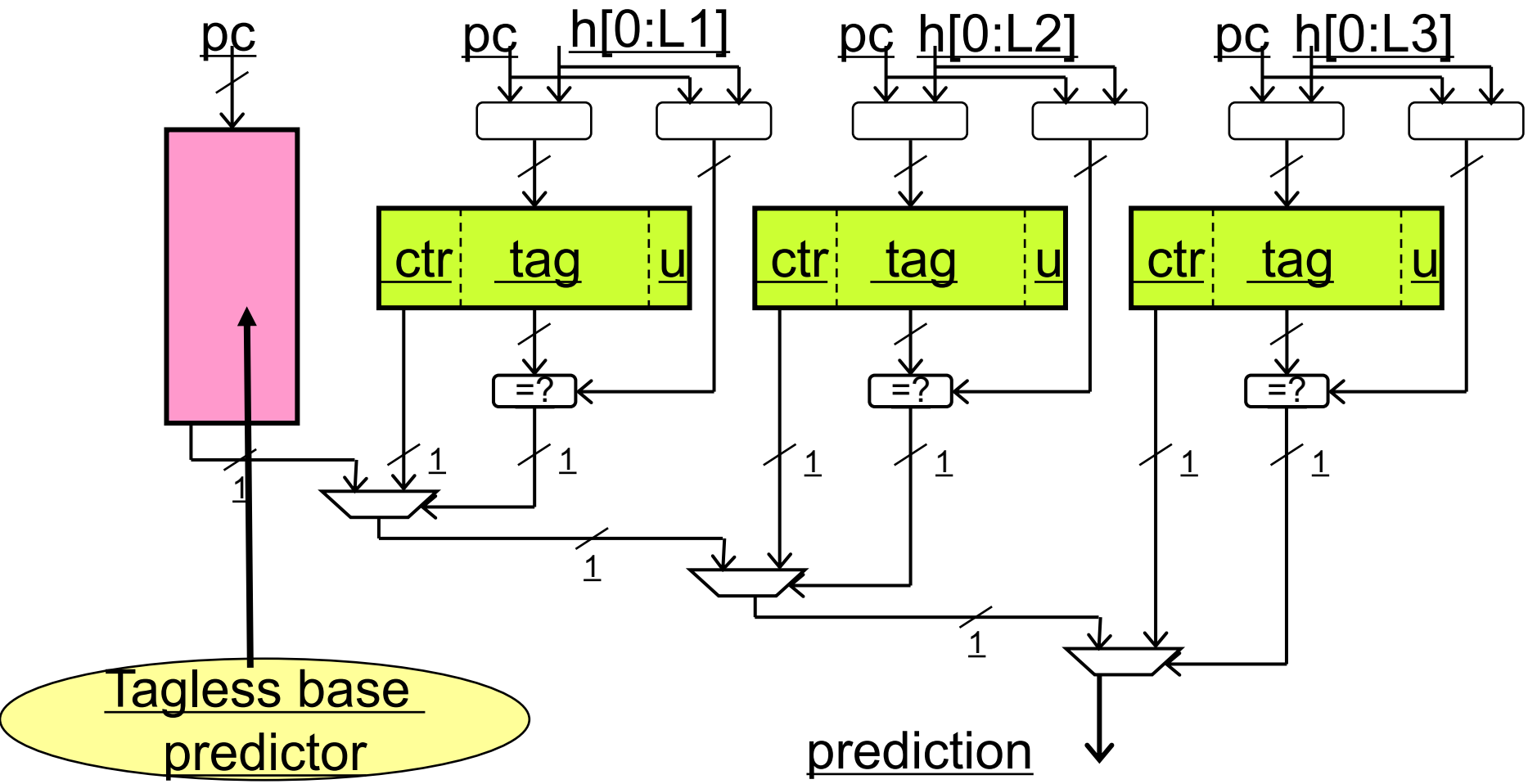
{0, 2, 4, 8, 16, 32, 64, 128}

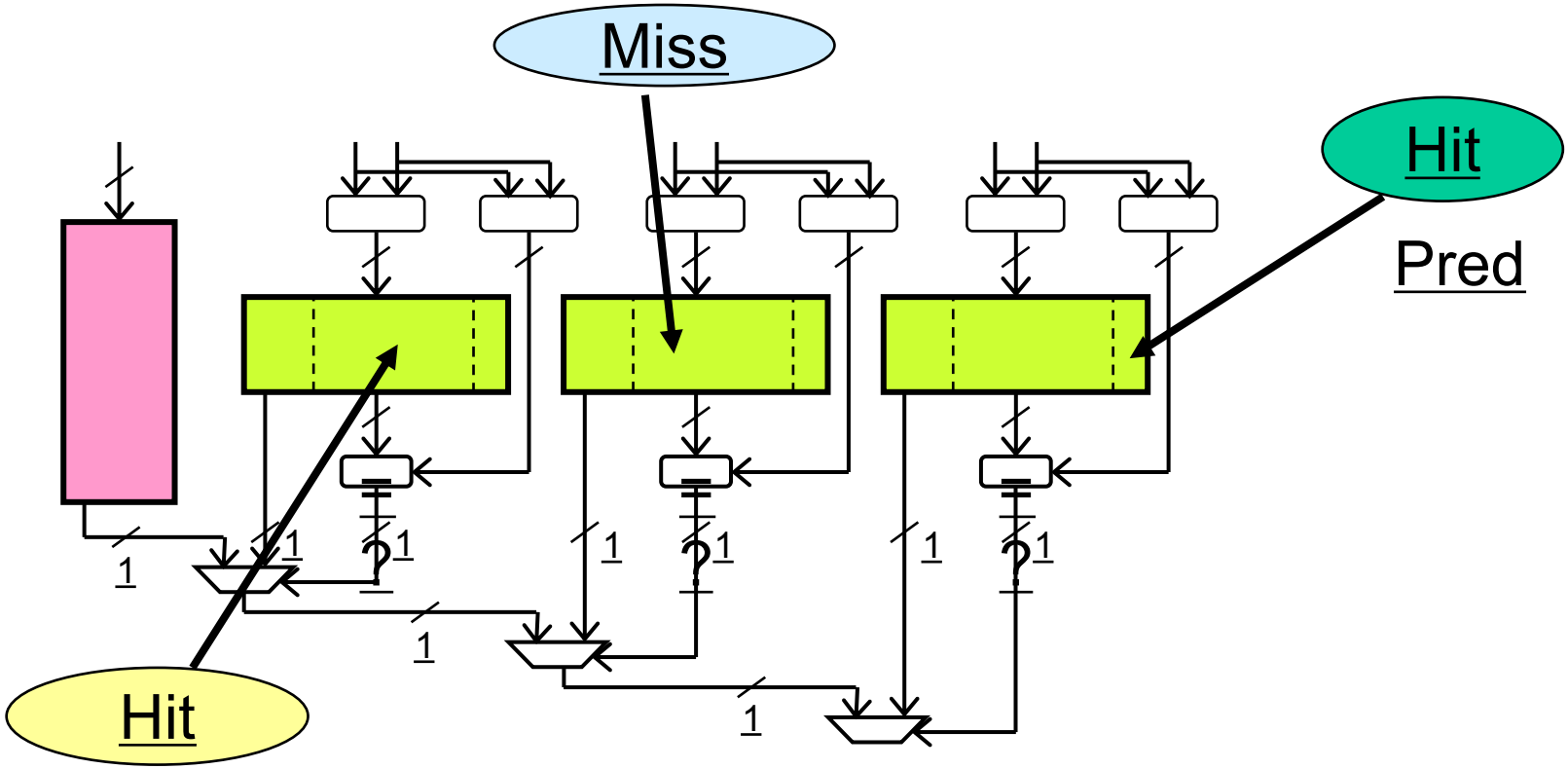
What is important: L(i)-L(i-1) is drastically increasing

Spends most of the storage for short history !!

TAGE:

Tagged and prediction by the longest history matching entry





Altpred: the prediction if Pred was not hitting

A tagged table entry

- Ctr: 3-bit prediction counter
- U: 1 or 2-bit counters
  - ➔ Was the entry recently useful ?
- Tag: partial tag, 11-12 bits good trade-off

Prediction computation

- General case:
  - ➔ Longest matching component provides the prediction
- Special case:
  - ➔ Many mispredictions on newly allocated entries: weak Ctr

On many applications, **Altpred** more accurate than **Pred**

➔ Property dynamically monitored through a 5-bit counter

Allocate entries on mispredictions

- Allocate entries in longer history length tables
  - ➔ On tables with U unset
- Set Ctr to Weak and U to 0
- Limited storage budget:
  - ➔ Allocate 2 entries

    (when 15 to 20 different history lengths)

Managing the (U)seful counter

- Increment when avoids a misprediction
  - (Pred = taken) & (Altpred ≠ taken)

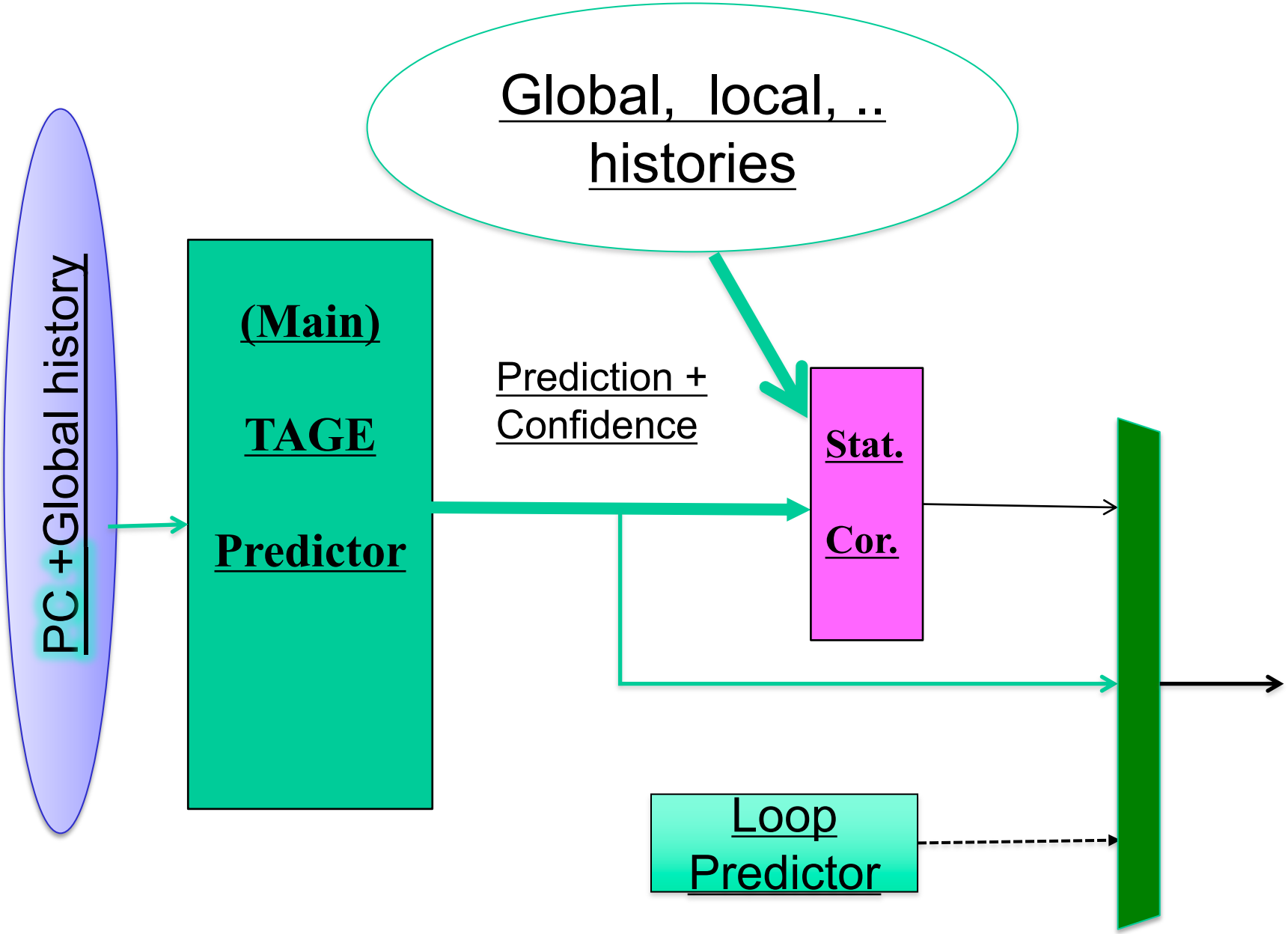
  Becomes « useful »

- Global decrement when it becomes « difficult » to allocate:
  - Many possible heuristics

    « difficult » ≈ 2/3 of the entries useful

TAGE-SC-L

- Was introduced at CBP 3, refined at CBP 4 and CBP5 (2016):
  - ➔ Objectives:
    - Enhance the TAGE prediction accuracy when possible
    - To win the championships
  - ➔ Main problem:
    - realistic implementation was not really considered


Global, local, .. histories
PC +Global history
(Main)
TAGE
Predictor
Prediction + Confidence
Stat.
Cor.
Loop Predictor

Evaluation framework

- Traces from CBP 2016

- Stop simulation at 10 000 000 branches
  - ➔ Does not change the relative accuracy
  - ➔ Save (my) time (and the planet 😇)

TAGE-SC-L accuracy (CBP 2016, ~512 Kbits)

- All: SC with local history, IMLI, global history (512 Kbits)
  - ➔ 4.140 MPKI
- No local history, no loop predictor (482 Kbits/512 Kbits: 1 extra tag bit)
  - ➔ 4.239 MPKI/ 4.220 MPKI
- SC tage ouput only (458 Kbits/517 Kbits: 2 Ubits 1 extra tag bit)
  - ➔ 4.361 MPKI/ 4.323 MPKI
- Pure TAGE (453 Kbits/513 Kbits: 2 Ubits 1 extra tag bit)
  - ➔ 4.451 MPKI/ 4.391 MPKI

At iso-storage budget
TAGE-SC-L 5.7 % MPKI gain over TAGE

TAGE-SC-L 2016 is not realistic: was designed to win CBP 😇

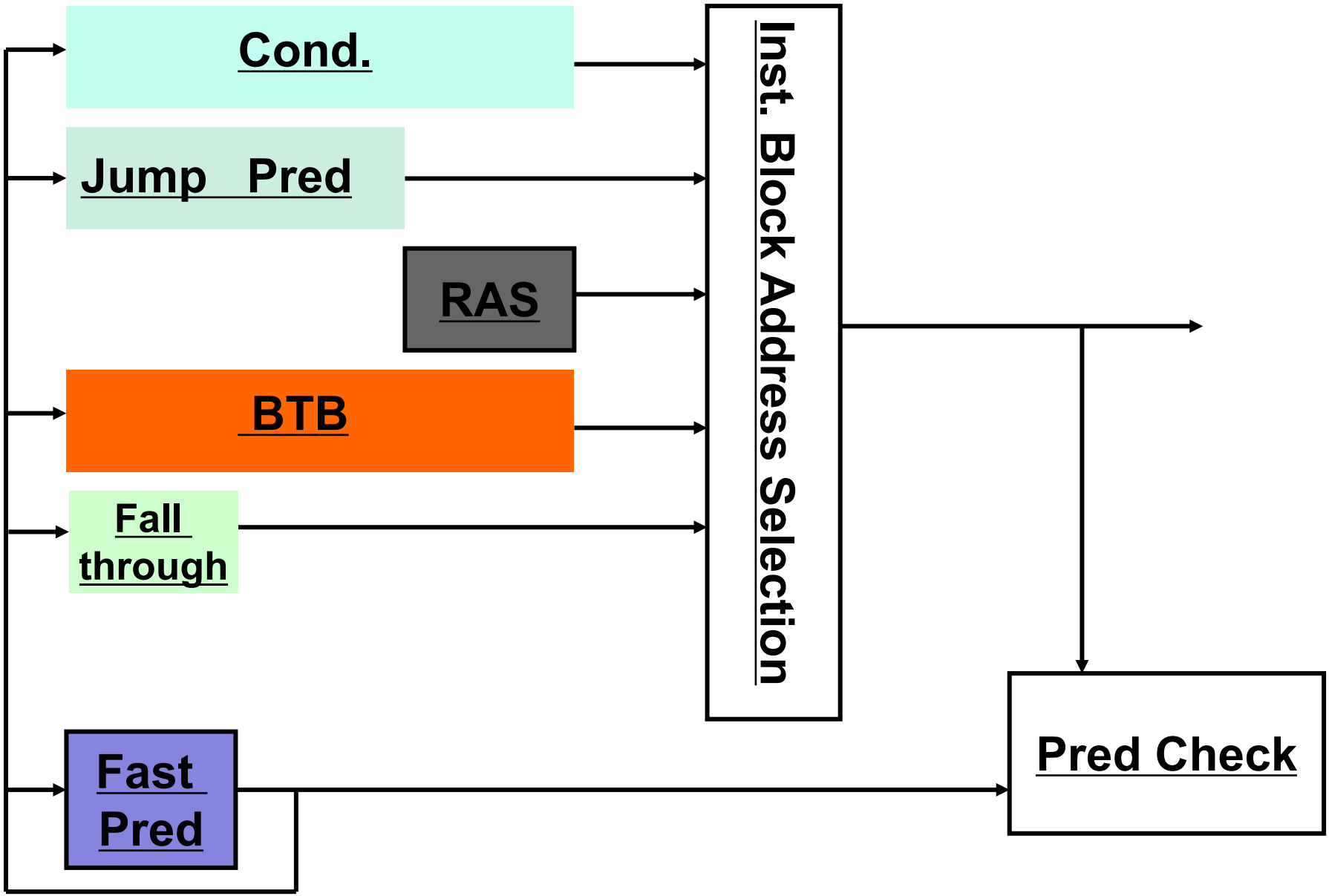
- Unrealistic number of tables on TAGE :
  - ➔ 30 interleaved banks.
- Unrealistic number of tables in the Statistical Corrector:
  - ➔ 20 tables
- Use of local histories:
  - ➔ 3 different local histories
  - ➔ Managing speculative local history is untractable with a 500+ instruction window
- Untractable prediction latency:
  - ➔ TAGE prediction followed by SC prediction:
    - SC prediction needs TAGE prediction, alternate prediction, confidence

Instruction front end organization
assumptions

In the first part:
A decoupled instruction fetch front end

- One instruction block predicted per cycle
  - The instruction block ends on:
    - A taken branch
    - Or after the 4th conditional branch

- Up to four not-taken capture most of the benefit of bypassing all not taken


Cond.
Jump Pred
RAS
BTB
Fall through
Fast Pred
Inst. Block Address Selection
Pred Check

Dealing with « pure » TAGE implementation

Predicting up to four branches per cycle

- Index the predictor with the address of instruction block:
  - No noticeable difference with the branch address
- Tag the TAGE entries with the number of the branch:
  - Not an extra 2-bit tag, but embedded in the tag
  - An extra XOR in the tag computation (not on the critical path)
  - An extra 4-4 permutation at the end of the prediction
- Design options:
  - Only one prediction per instruction block per table
  - Two or four predictions per instruction block per table (direct mapped)
    - Two predictions: - ~1.1 % MPKI
    - Four predictions: - ~ 0.8 % MPKI

Revisiting the alternate prediction

- In the initial definition of TAGE:

"Altpred is the prediction if Pred was not hitting"

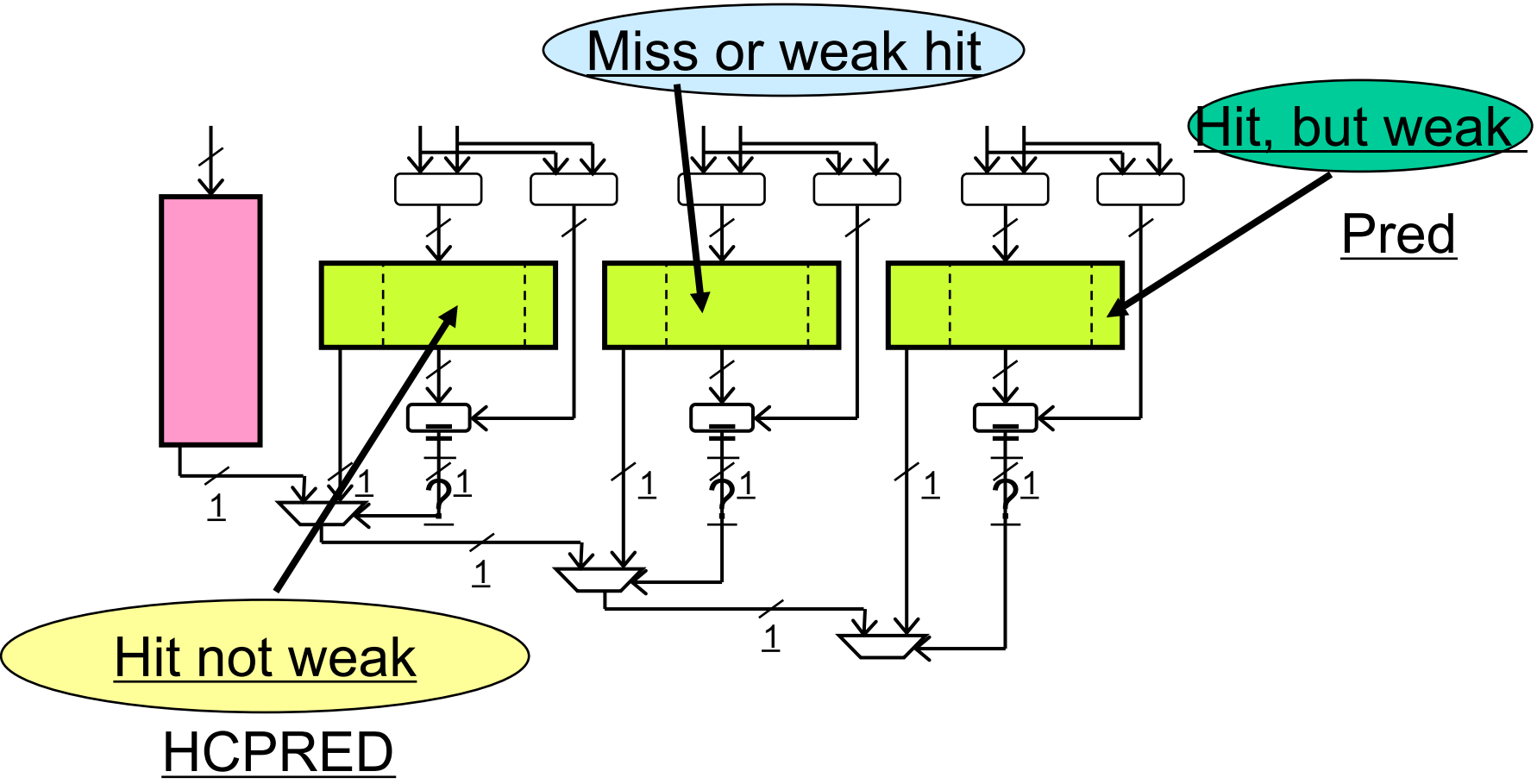
In practice, Altpred is more difficult than Pred.

This lengthens the prediction time.

An alternative is to use HCPRED the longest hitting "not weak" entry:

If Pred is not weak then HCPRED is identical to Pred

If Pred is weak then HCPRED is distinct



HCPRED= Longest Match hitting not weak prediction

Revisiting the alternate prediction (2)

- Still the need to compute the 2nd hitting entry for U counter update:

  if (Pred==taken) U+= (altpred≠taken)

  ➔ altpred is only needed at update time:
    - not on critical prediction path

- The good news:
    - marginal accuracy benefit vs the initial definition

Medium large number of tables is better for accuracy

- With 250 taken branch history and 12 tag bits (509 Kbits)
  - ➔ 7 tables: 4.704 MPKI
  - ➔ 14 tables: 4.527 MPKI
  - ➔ 28 tables: 4.519 MPKI (would be better with 13 tag bits)

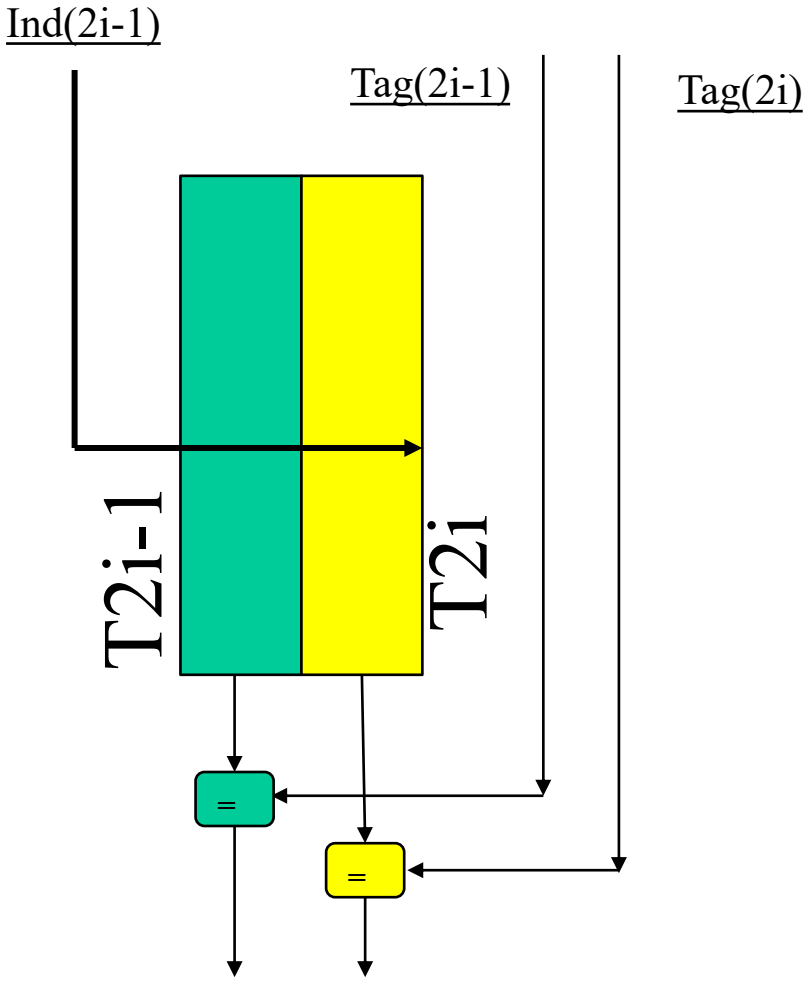
- 14 tables: probably to many for implementation 😱

Turn-around to limit to 7 physical tables, but 14 logical tagged TAGE tables

- Use a single physical table for H(2i-1) and H(2i)
  - ➔ 4 predictions for H(2i-1) and 4 predictions for H(2i)
  - ➔ Index computed with H(2i-1)
  - ➔ Respective tags computed with H(2i-1) and H(2i)
  - ➔ Adapt history lengths :
    - newL(2i) = L(2i-1) + (L(2i)-L(2i-1))/2
- 4.603 MPKI vs 4.527 MPKI:

  but only 7 tagged physical tables

A single physical table for two logical TAGE tables



Sharing tables: limiting unbalanced loads on the tables

- Some applications need more entries on small histories and underutilize the tables with large histories
- Other applications would need more entries on large histories and underutilize the tables with small histories

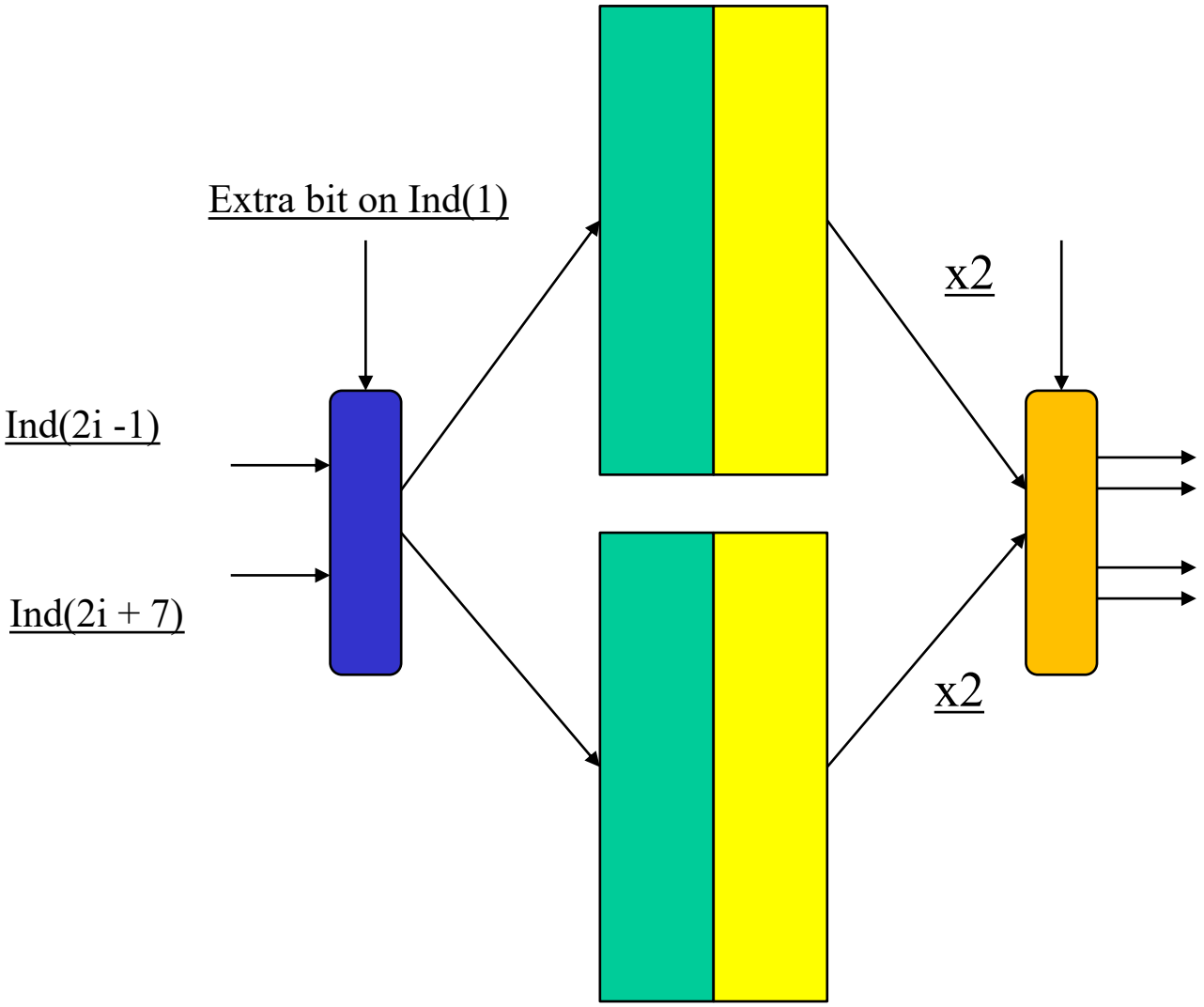
➢ To balance the load on the different physical tables, logical table T(i) and T(i+8) share two tables based on one bit index computed with H(1).

Back from 4.603 to 4.527

but adds two 2x2 MUX on the prediction path

- Note: Sharing space between T(2i-1) and T(2i) is not worthwhile

Sharing tables



Geometric series is not that optimal

Seems that:

- ➔ Better to concentrate the history lengthes in the middle of the spectrum

For 14 tables:

-use a geometric series of 18 elements

-use x1, x3, x5, x6, x7, x8, x9, x10, x11, x12, x13, x14, x16, x18

- From 4.527 to 4.494 MPKI

Dimensioning the tagged entry

- U counter :
  - ➔ 1 bit: 4.528 MPKI
  - ➔ 2 bits: 4.494 MPKI
- Tag width:
  - ➔ 11 bits: 4.515 MPKI
  - ➔ 12 bits: 4.494 MPKI

Optimizing allocation/replacement

- Replacement is not on the critical path

- Base replacement policy uses U counter:
  - ➔ Monitors if ratio of « useful » entries is below 2/3
  - ➔ Reaching 2/3 triggers smart U counter reset
- Several optimizations are possible (details in next slides)
  - ➔ Filtering allocation : -0.04 MPKI
  - ➔ First U counter setting: -0.02 MPKI
  - ➔ Protecting recently useful allocated entries: -0.01 MPKI
  - ➔ Updating HCPRED and AltPred on weak Pred mispredicting: -0.015 MPKI

4.406 MPKI: 2 % gain

Filtering allocation

- Two optimizations:
  - ➔ If the misprediction ratio is high then reduce the possibility of allocation on long history tables (T8 and higher)
    - Counter monitoring if (misp_ratio > 1/9)
  - ➔ If the ratio of correct predictions on weak counters is low, reduce the allocations:
    - If ratio < 50% then allocate only with probability 1/8
    - If ratio < 16/31 then allocate with probability ½

Use of probabilistic counters

First U counter setting

- Basic policy on allocation is setting U=0
- When filtering allocation:
  - ➔ Set the U counter of the 1st allocated entry to 1
  - ➔ Reset randomly the U counter if the prediction is weak when allocation is denied on the table (see code for probability)

Protecting recently useful allocated entries when useful

- Set the U counter to 2 instead of 1 to allow the entry to survive the smooth resetting

- Only resets the U counters when the ratio of useful entries is 4/5 if the mispisprediction rate is high.

Updating HCPRED and Altpred on weak Pred mispredicting

- When Pred is weak and mispredicts, also update HCPRED and Altpred (if on distinct tables)

Summary on « realistic » pure TAGE

- Chose prediction between largest history hitting entry and largest history hitting not weak entry
- 1 bimodal table + 7 tagged physical tables
- 14 logical tables:
  - ➔ 1 logical table spans over two physical tables
  - ➔ 2 adjacent logical tables share the same index

- Allocation/replacement is not on the prediction computation path:
  - ➔ Several optimizations decrease the misprediction rate

4.406 MPKI

within 0.5 % of unrealistic CBP 2016 512 Kbits TAGE

Table associativity

- Trading aliasing against associativity benefit:
  - ➔ 4.383 MPKI for associativity 2 at 12 tag bits
  - ➔ Associativity 4 require extra tag bits
- For the fun:
  - ➔ Use partial skewed associativity (on 2 bits)
  - ➔ Optimize the allocation/replacement
    - 4.358 MPKI : Same accuracy gain that assoc 4 with 13 tag bits

Conflicts on way 0 and 2 are different:

- Prediction 2 for block B conflicts on prediction 0 for block A on way 0
- Prediction 3 for block B conflicts on prediction 0 for block A on way 1
  - A0, B2 and B3 can coexist !!

Dealing with a « realistic » TAGE-SC

(LET US IGNORE LOOP PREDICTOR)

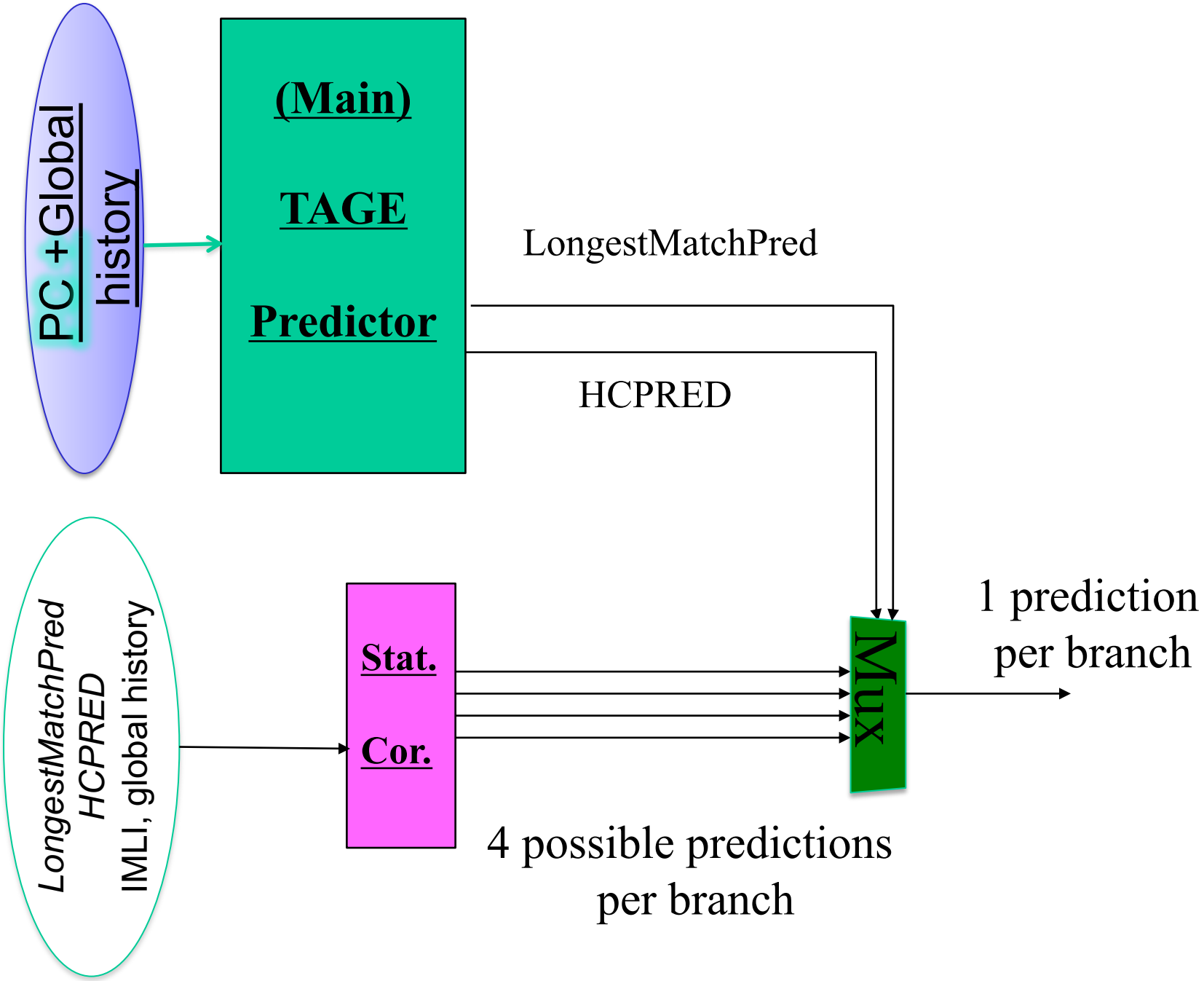
Dealing with the latency issue

SC prediction is computed with information flowing out from TAGE:

- ➔ SC sum computation follows TAGE prediction

To decrease latency

- ❖ Speculative computation of all possible SC sums with all possible TAGE output (LongestMatchPred, HCPRED, confidence, ..) and final selection through a MUX:
- ➔ on CBP 2016 predictor: 8 bits !!
- ➔ To be realistic:
  - only use two bits: LongestMatchPred and HCPRED


PC +Global history
(Main)
TAGE
Predictor
LongestMatchPred
HCPRED
LongestMatchPred
HCPRED
IMLI, global history
Stat.
Cor.
Mux
1 prediction per branch
4 possible predictions per branch

SC using just PC and TAGE output

- A single physical table combining 2 logical tables:
  - ➔ indexed respectively with:
    - PC + branch number
    - PC + branch number +(LongestMatchPred,HCPRED)
      - Same index apart 2 bits
- A 4-entry table indexed with (LongestMatchPred,HCPRED)
- A 2-entry table indexed with LongestMatchPred
- A 2-entry table indexed wit HCPRED

➢ A 2x1K 6-bit counter table: 520 Kbits, 4.283 MPKI

Equivalent gain over Pure TAGE as for CBP 2016 TAGE-SC
much smaller prediction latency

Which other information are worth the effort ?

- From CBP 2016
  - ➔ Global history related
  - ➔ IMLI related: was found as dissapointing
  - ➔ ~~Local history: not implementable due to speculative history management~~

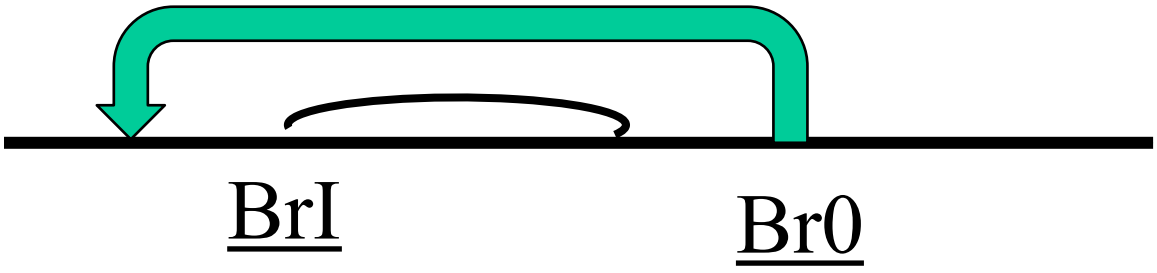
- So what ?

What is IMLI ?

IMLI= Inner Most Loop Iteration number

A. Seznec, J. San Miguel, J. Albericcio "The Inner Most Loop Iteration counter: a new dimension in branch history", Micro 2015, December 2015

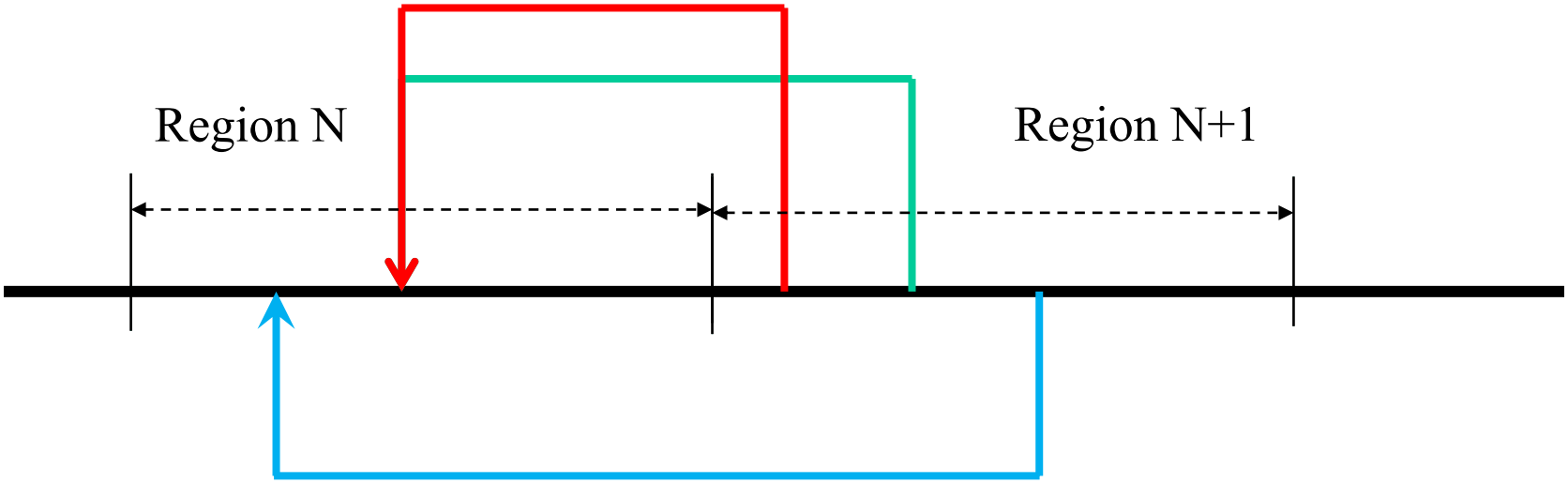
- IMLI counter: counts the number of iterations of the last backward branch



- Efficient to predict branches in the loop body on CBP 2014 traces
- Was not that great on CBP 2016 traces

Reengineering IMLI: « branch » IMLI

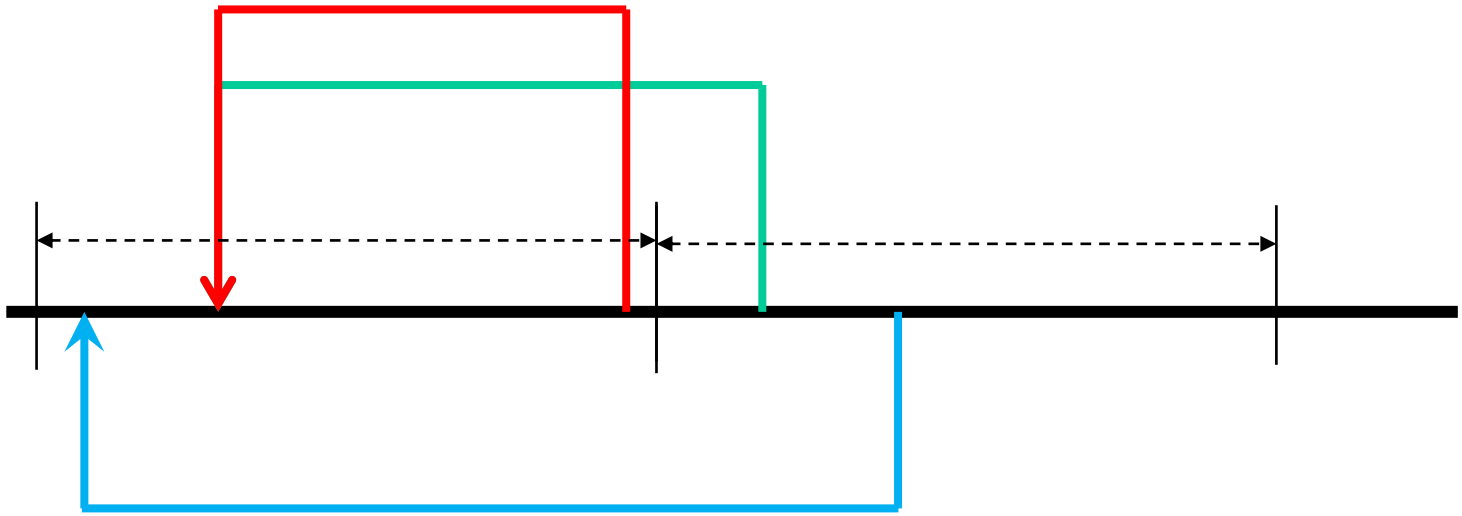
- BrIMLI counts the number of consecutive taken backward branches within the same « region »



- Monitors the number of iterations in a loop with multiple exits
- Monitors the number of iterations in loop nests
- Region = 256 bytes (seems the best trade-off)

Reengineering IMLI: « Target » IMLI

- TaIMLI the number of consecutive taken backward branches with their target in the same « region »



- Capture the same phenomena as BrIMLI, but adds some coverage

Global forward path and Filtered backward path

- Global Forward Path: the taken path with only the forward branches
- Filtered Backward Path: the global backward path but only when BrIMLI or TaIMLI is reset
- Using a single extra table per feature.

With all tricks (see the code, for the details)

- BrIMLI and TaIMLI are null in many cases:
  - ➔ Share the tables with Global path
- When LongestMatchPred is high confidence and SC very low confidence use LongestMatchPred
  - ➔ this replaces the final 4x1 MUX by a 5x1 MUX
  - ➔ Necessitates to check SC confidence
  - ➔ Improve pure SC by ~0.01 MPKI
  - ➔ Improve full SC by ~0.02 MPKI

4.146 MPKI TAGE-SC with a 2x1K counter table + 4 1K counter tables for SC (545 Kbits)

4.179 MPKI at 511 Kbits (halves the bimodal and SC tables)

Summary on « realistic » full TAGE-SC

- Can be built with:
  - ➔ 7 large tagged tables and a large bimodal for TAGE
  - ➔ 4 1Kcounter tables + a 2x1Kcounter table for SC

- Overall prediction latency equivalent to pure TAGE

- Only use global information and 2 bits from TAGE
- Use forward taken history and backward taken history
- Use Reengineered IMLI

MPKI within 1 % of unrealistic CBP 2016 512 Kbits TAGE-SC-L

At 511 Kbits and only global information !!

TAGE-SC with only two physical SC tables

- Most benefit from non-PC comes from a single table:
  - ➔ BrIMLI (+ global history when BrIMLI==0)

- 4.197 MPKI TAGE-SC with a 2x1K counter table + a 1K counter table for SC (527 Kbits)

Scaling

- Scaling down to 64 Kbits:
  - ➔ Just dividing able size by 8:
    - In the same accuracy range as CBP2016 8KB TAGE-SC-L
    - Probably would slightly benefit from larger SC budget
      - – Not done the exercise …
- Increasing/decreasing the number of tagged physical tables:
  - ➔ 6 or 8 are good design points

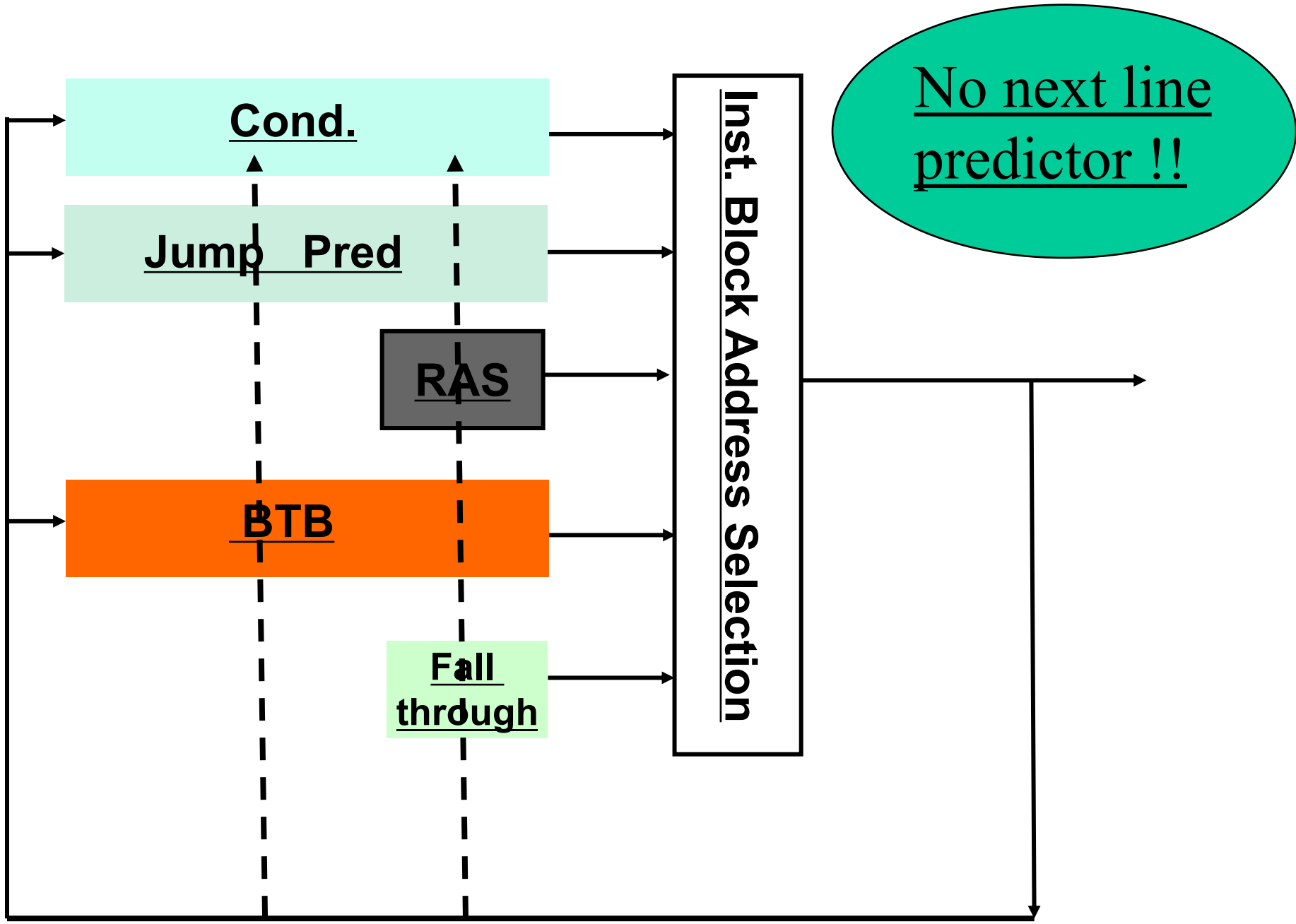
Dealing with prediction latency

3 BLOCK AHEAD TAGE-SC PREDICTION

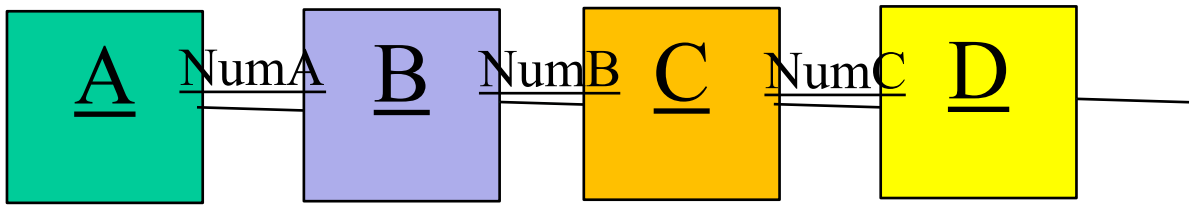
2nd part: Ahead pipelined front-end

- Instruction address generation takes N cycles:
  - ➔ Begin to predict/compute instruction block address with N cycles ahead information
  - ➔ Use intermediate information

- This presentation assumes 3 cycles.

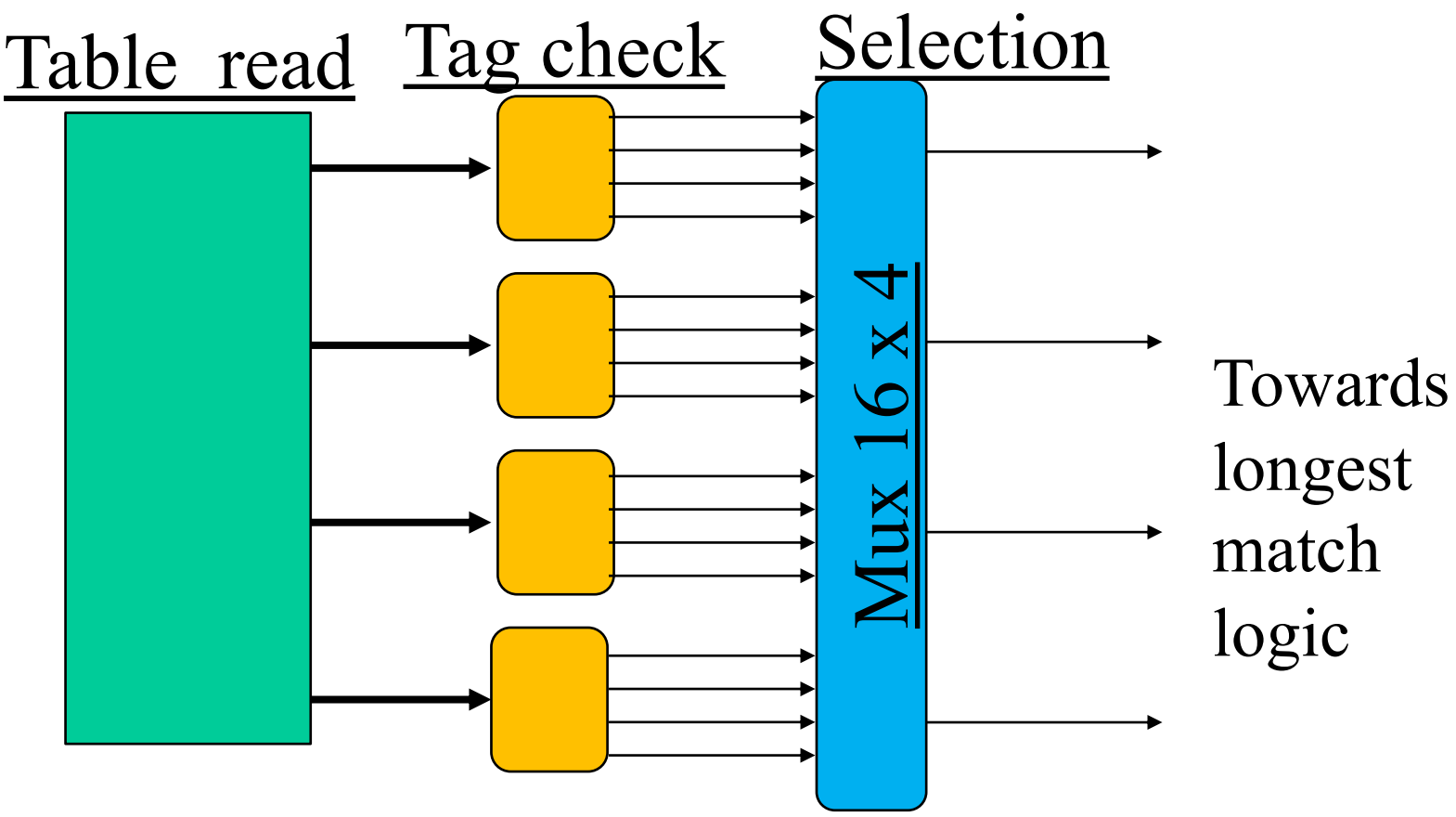

Cond.
Jump Pred
RAS
BTB
Fall thrdugh
Inst. Block Address Selection
No next line predictor !!

TAGE prediction computation

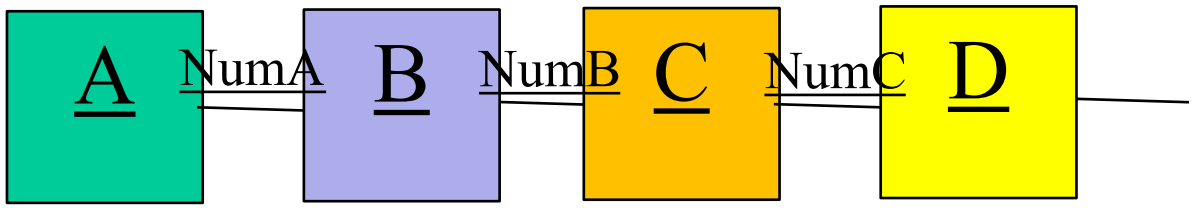


- To predict D in 3 cycles:
  1. Use (A,Ha) to index the tables, read 2**M entries
     - M bits are unknown: hashed from (NumA, B,NumC)
  2. Use (B,Hb) for the tag check unknown bits for (NumB, C)
     1. Perform 2**N // tag checks per read entry
  3. Use C to read the bimodal while computing LongestMatchPred and HCPRED:
     - Select 4 out of 2**(N+M) exits for each tagged table
     - Flow through the longest match selection logic

Assuming 4 reads per table and 4 tag check per entry



TAGE ahead prediction



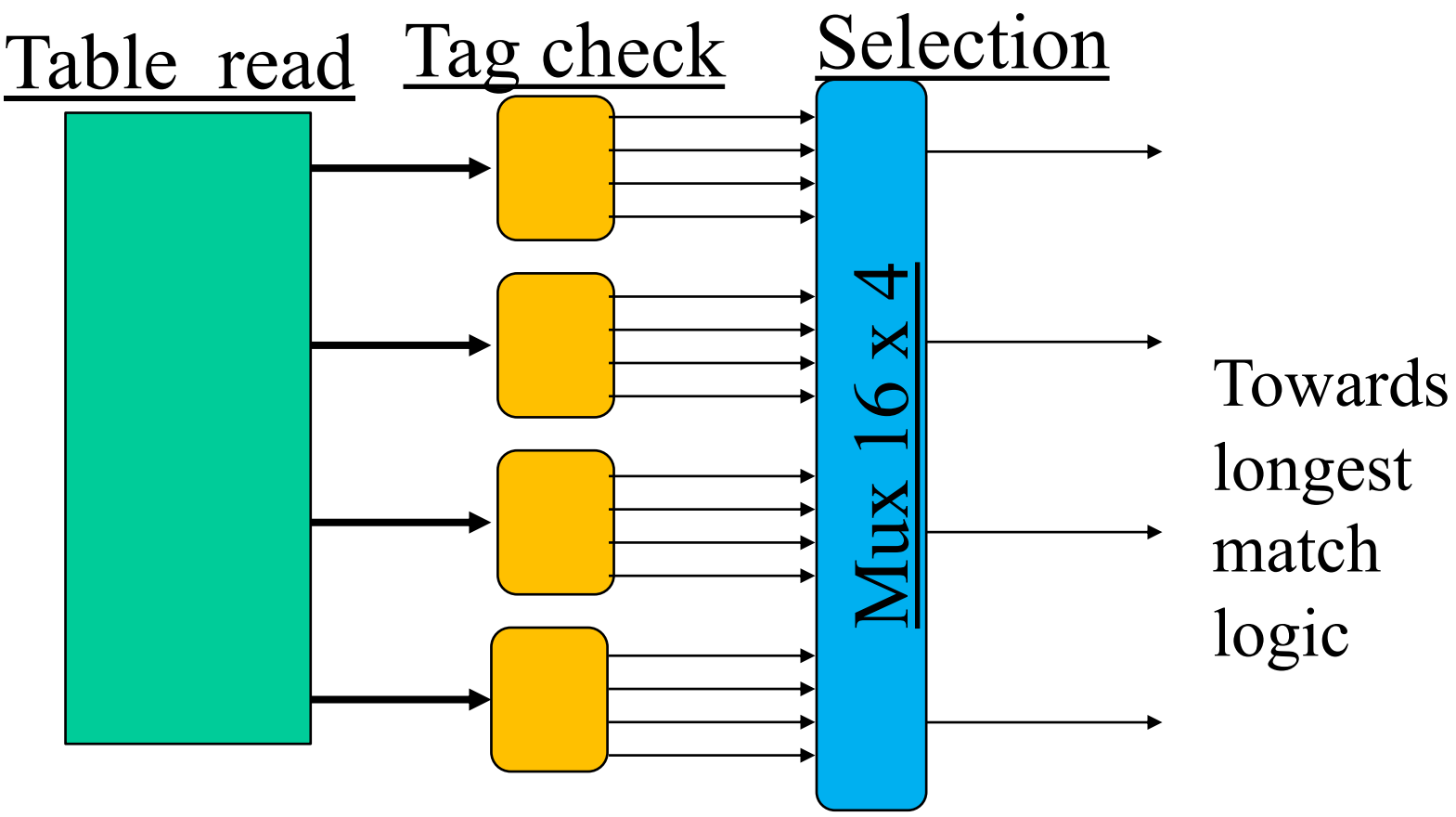
- Many path possibilities from A to D:
  - up to 4 branches in each basic block
  - The taken branch in A or B may be returns or indirects

- 8 * 8 or 16 * 4 are good design points:
  - Cost is some logic

4.641 MPKI against 4.358 MPKI for 1-block ahead (C->D) :

~6% accuracy loss

Assuming 4 reads per table and 4 tag check per entry



The SC component

- Can probably be done in two cycles:
  - ➔ Tables read
  - ➔ Adder tree
- To predict block D:
  - ➔ Use information available with B
- PC + LongestMatchPred+ HCPRED only:
  - ➔ 4.462 MPKI against 4.270 MPKI for 1-block ahead: extra ~4.5 %
- All:
  - ➔ 4.326 MPKI against 4.146 MPKI for 1-block ahead: extra ~4.4 %

The SC component (better with extra logic)

- Use speculatively informationH(NumB, C) e.g. 2 bits:
  - ➔ Read 4 times more entries on SC
  - ➔ Compute 4 times more SCsum
  - ➔ Select 1 prediction out 4 with H(NumB,C):

    - 1 % misprediction reduction

Final summary

- « Realistic » TAGE-SC:
  - ➔ achieving accuracy close to unrealistic CBP 2016 TAGE-SC-L
    - With number of TAGE tagged tables around 7
    - With number of SC tables around 5
    - With prediction latency of TAGE
  - ➔ should be implementable in hardware