




Branch Predictor for SAT-solver

Some Insights and Results from our project by
Team TNG

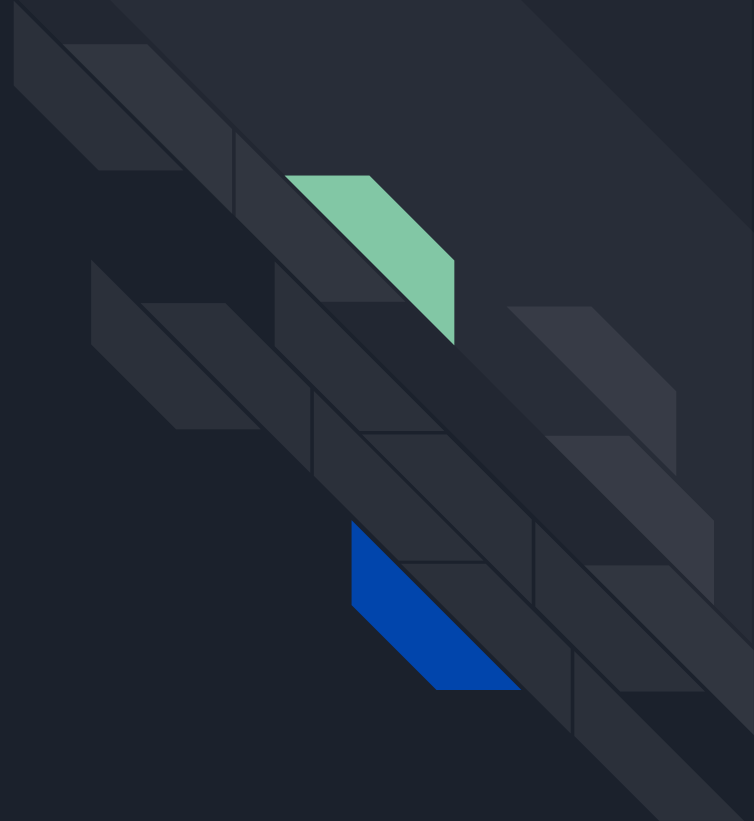


Team TNG :

- Gowtham S (210050059)
- Soni Naman Nirmal (210050151)
- Terli Tulsi Ram(210050157)

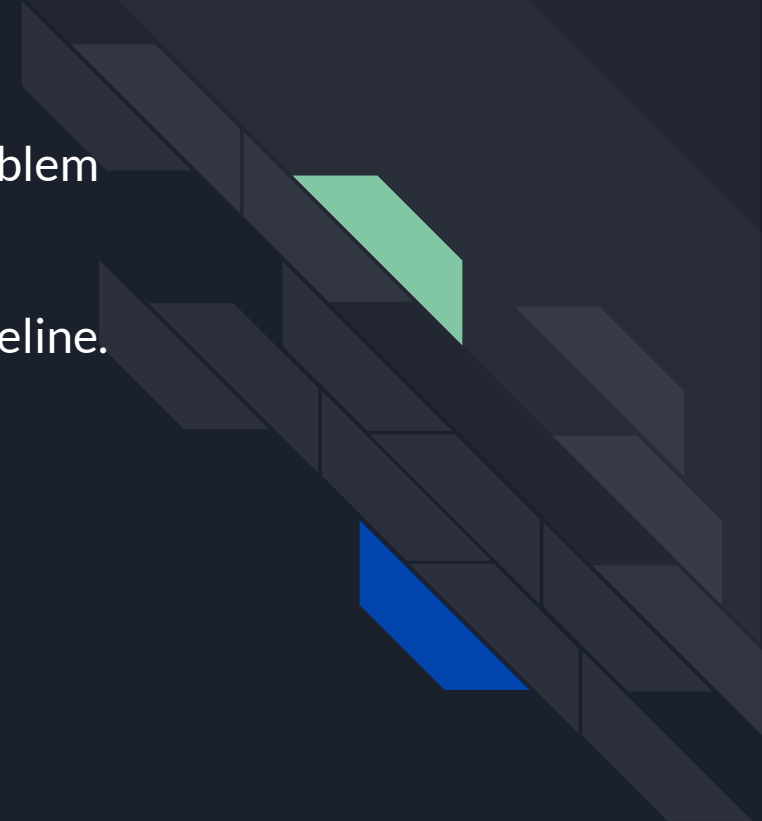
Table of Contents

- Branch Predictors
- Why SAT solvers?
- TAGE Predictor
- L-TAGE Predictor
- Results and Inference
- Resources



What is Branch Predictor ?

- The Branches and loops are indeed a problem to a pipelined processor. Wrong branch prediction makes pipelines to flush the instructions in the early stages of the pipeline.
- This saves millions of cycles and nop instructions during execution of large applications.



Why SAT Solvers ?

- Very sophisticated branch predictors are known to work well on **almost** all applications.
- Exceptions are here :
 - Graph analytics - (not in this project)
 - SAT solvers
- SAT solvers are used in automated checking of circuits, model checking and various reasons.

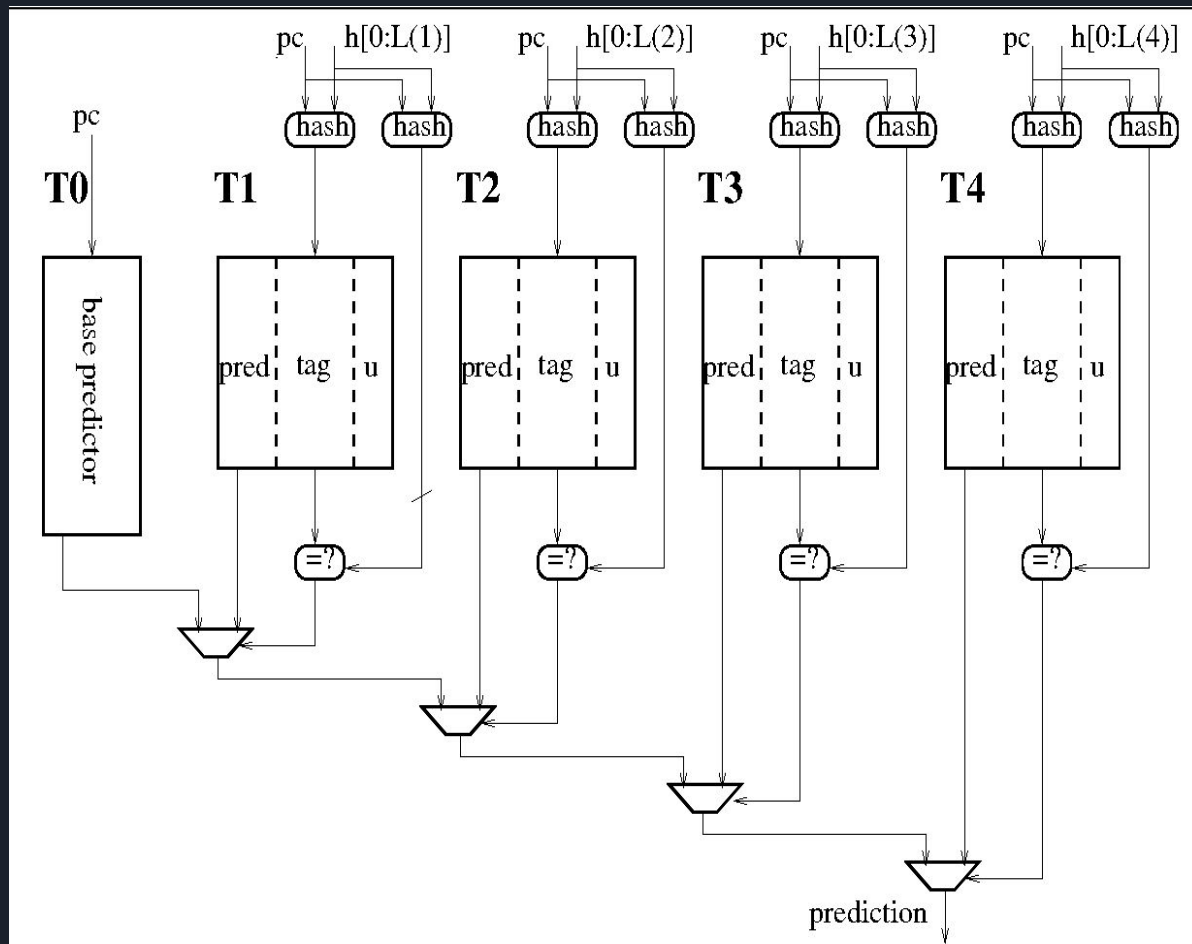


What did we do ? TAGE Predictor

- Components :-
 - Base Bimodal Predictor
 - Counter, Tag, Index, u-bit
 - Tagged History Components
 - Counter, Tag, Index, u-bit
 - Index calculated by hashing instruction address and some variable length global history, have different lengths.

- Tables having lengths in GP with common ratio $1.6(\alpha)$

$$T(i) = \text{int}(\alpha * T(i-1) + 0.5)$$





Prediction Algorithm

- We find the *longest history matching table* among the tagged - history components. If no table matches, we take the prediction given by bimodal predictor as our *final prediction*.
- We also keep track of the matching table with second longest history and call that prediction ***alternate prediction***, we use it when our prediction is weak and the longest matching entry is newly allocated.



Why to use alternate prediction ?

- A prediction is termed weak if its prediction for next time changes if the branch takes opposite direction to the direction predicted this time
- So, if the prediction is weak, there are greater chances of it to be wrong also if an entry is newly added into the history table, the prediction made by it will also be most likely weak,
- So intuitively it makes sense to use the prediction made by second longest matching table, just to be careful.



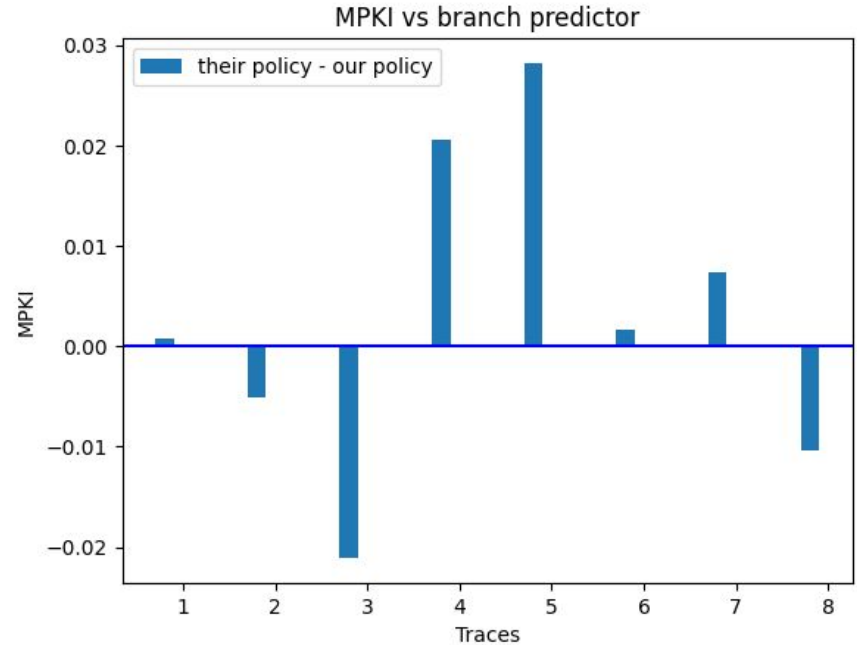
Updates

Here, the useful bits come into play

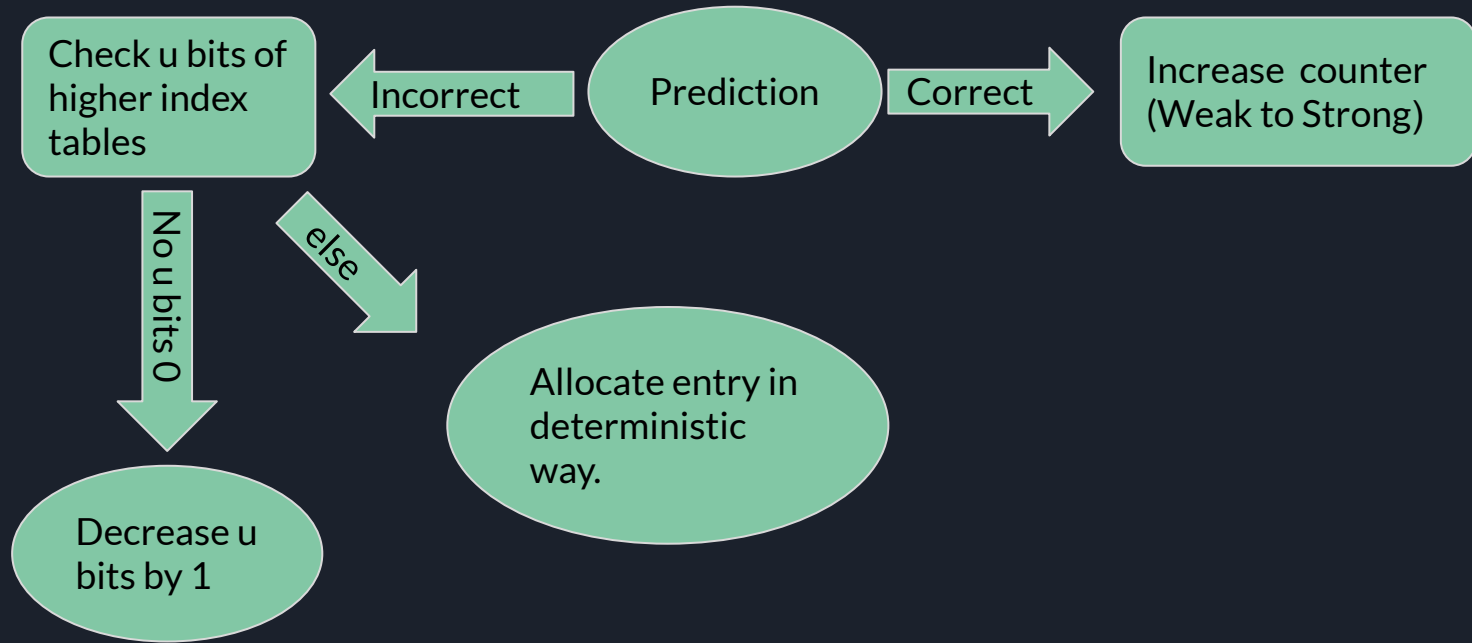
- They are updated when $\text{alt prediction} \neq \text{final prediction}$. If the final prediction is correct, then u bits are incremented else these are decremented.
- In case of misprediction and hit in tagged component X , then going through the tagged components from $X+1$ till the last tagged component
 - In case if there are multiple tagged components with $u = 0$, then replace the entry in a tagged component with a probability in decreasing order from the components from $X+1$ to the last tagged component. This was a change done by us from the TAGE predictor. (results in next slide)
 - Else, then we decrement all the u bits of those tagged components
- After every 512K instruction we are resetting our useful bits.

Improvement from the update policy

- Their improvement policy was for only 3 components.
- So, we generalized it for all the tagged components with a entry with resetted u bit.
- The improvements can be seen in the graph beside where the difference between the MPKI are plotted.



FLOWCHART



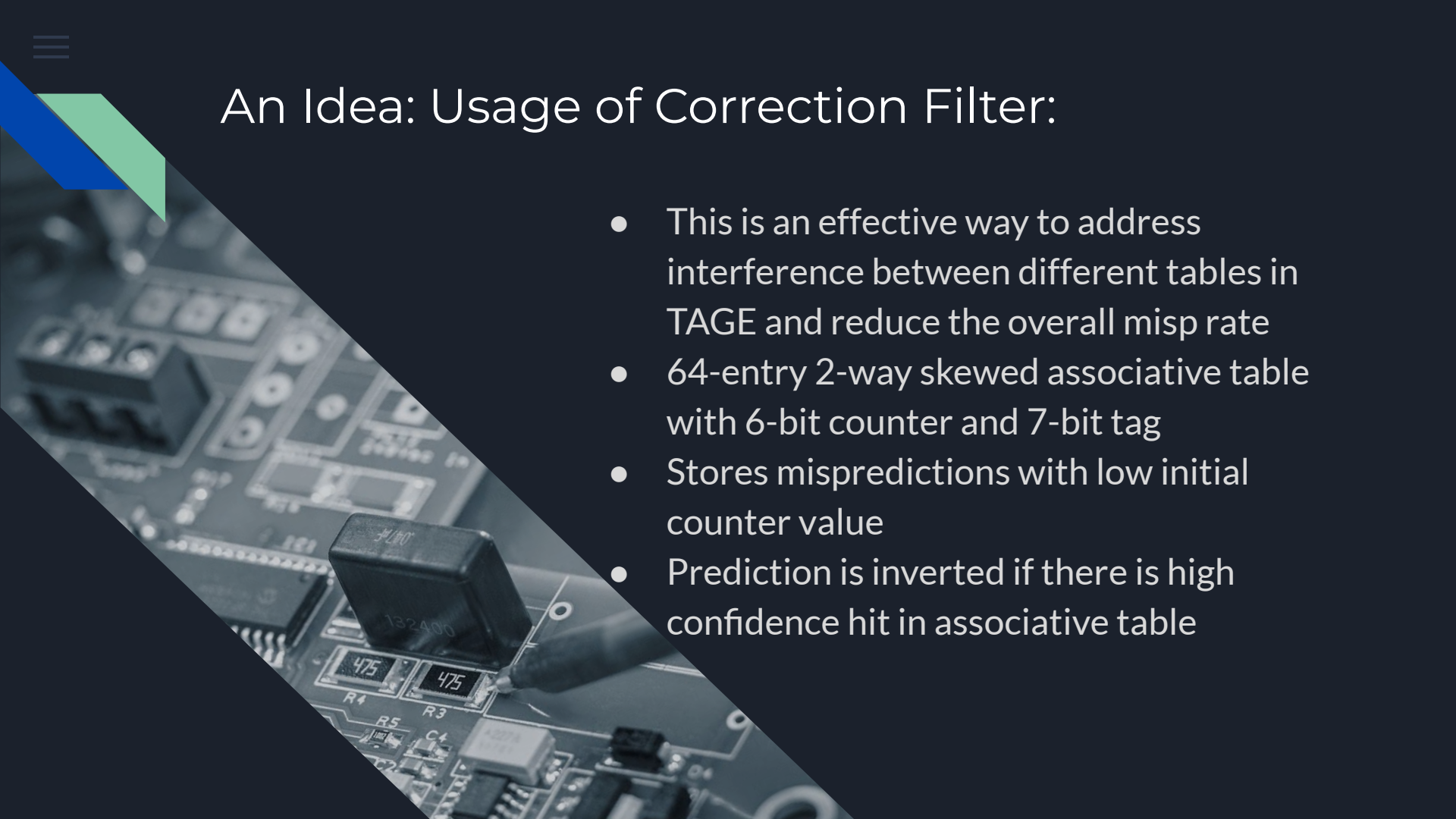


TAGE → L-TAGE

- Loop Predictor
 - Provides global prediction when a loop has been executed successfully with same number of iterations thrice.
 - Prediction is taken over TAGE prediction when it has high confidence and if the loop predictor is performing consistently.
 - Update policy is based on age.



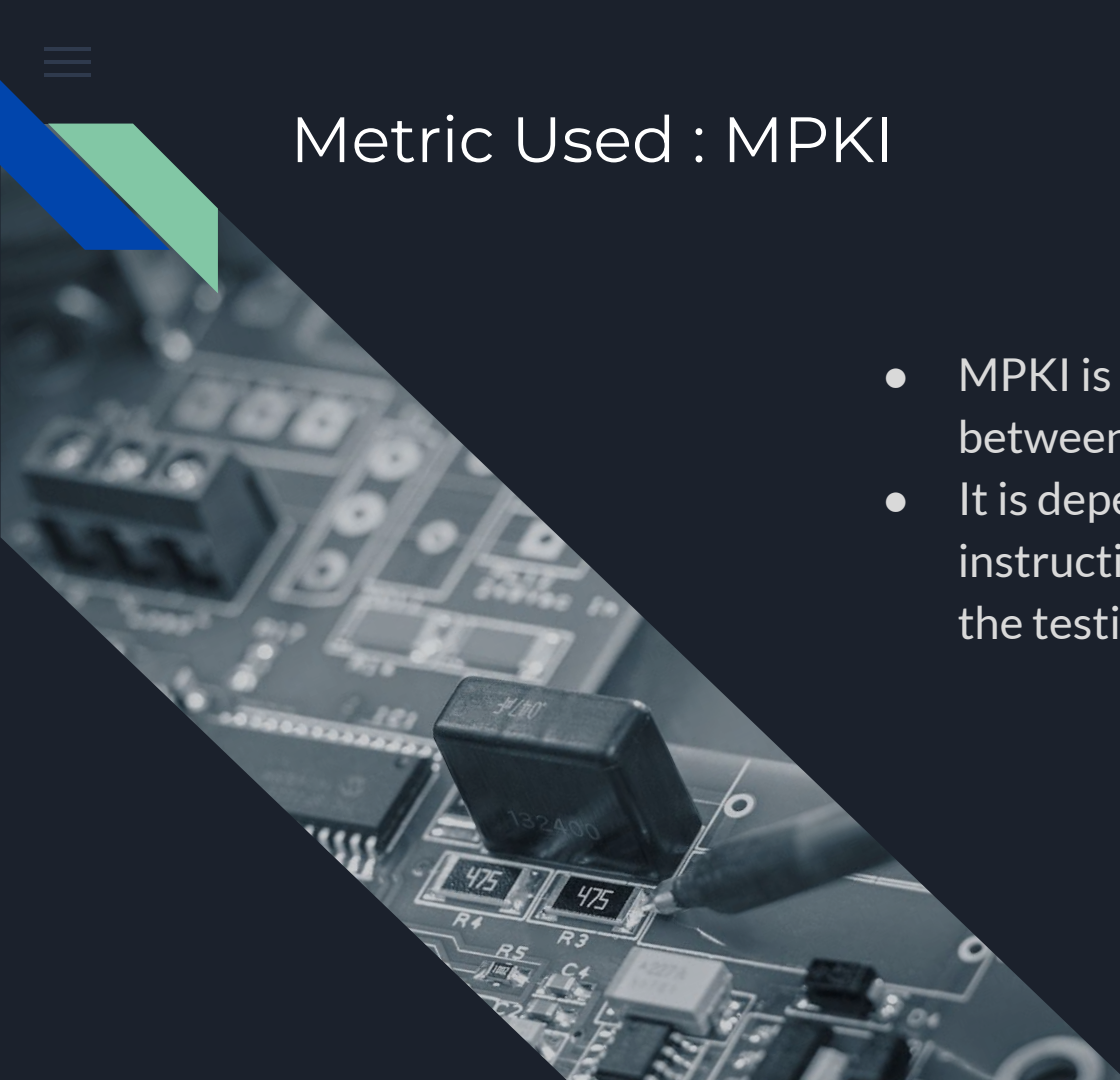
An Idea: Usage of Correction Filter:

- This is an effective way to address interference between different tables in TAGE and reduce the overall misp rate
 - 64-entry 2-way skewed associative table with 6-bit counter and 7-bit tag
 - Stores mispredictions with low initial counter value
 - Prediction is inverted if there is high confidence hit in associative table
- 
- A detailed background image of a printed circuit board (PCB) with various electronic components. A black probe is shown touching a component on the board. Labels like 'R4', 'R5', 'R3', 'C4', and '475' are visible on the board.



Metric Used : MPKI

- MPKI is used as it has high correlation between the accuracy and IPC.
- It is dependent on the number of instructions which is constant across the testing and experiments done.






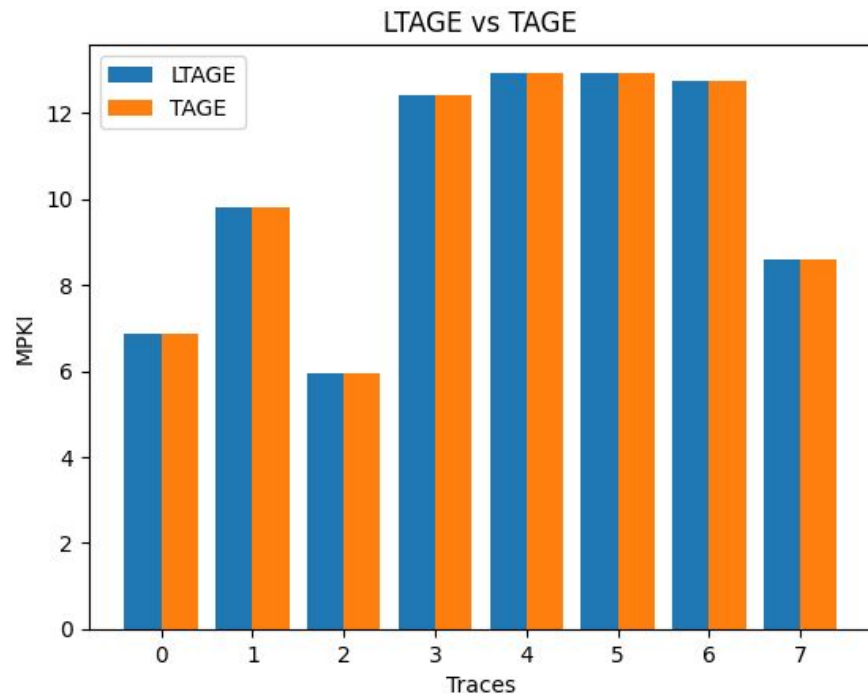
Experiments Done and Results

1. TAGE vs LTAGE
2. Comparison between Hashed Perceptron, Bimodal and LTAGE
3. Variation of MPKI vs Number of Tables
4. Variation of MPKI vs Alpha values
5. Uniform Tag Indexes and History lengths vs Non uniform Tag Indexes and History lengths.

All the results are in the sub-folders in the results_30M folder and are ran for 30M warm-up instructions and 30M simulated instructions.

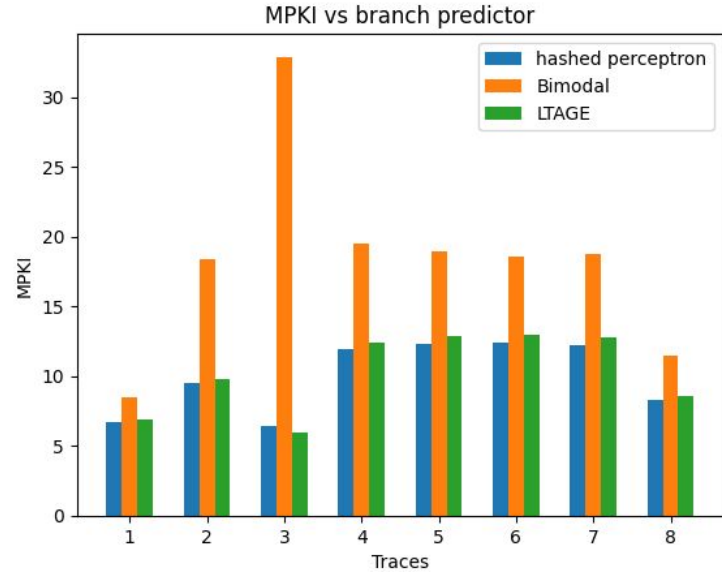
Loop Predictor for SAT solvers ?

- SAT Solvers  Constant Iterations
 - The number of iterations depends on the structure and complexity of formula.
 - No fixed iteration length
No significant increase.



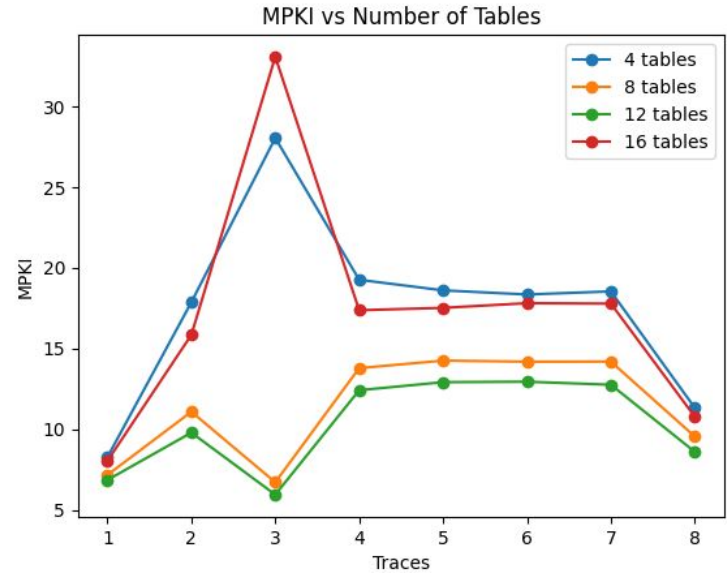
Inferences from this graph

- For SAT Solvers, Hashed perceptron works better than L-TAGE slightly.
- The basic reasoning is that L-TAGE is basically TAGE if the loop predictor is of no use in this case.
- Bimodal Branch Predictor is used as basic benchmark for MPKI.
- In general, Hashed Perceptron is better than L-TAGE and L-TAGE is better than Bimodal Branch Predictor.



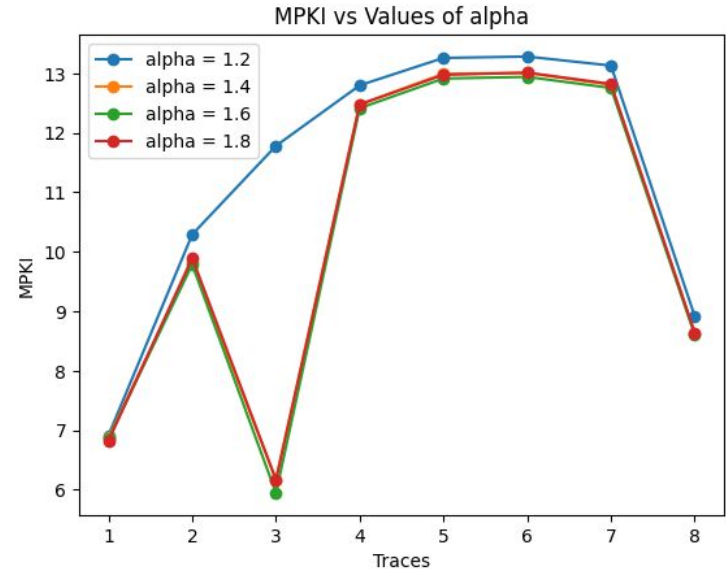
Inferences

- From the graph beside, The branch predictor with 12 components works better than other branch predictors.
- The branch predictors with 8 or 12 tagged components are better.
- Here, the tag Indexes for branch predictors were used from the paper.



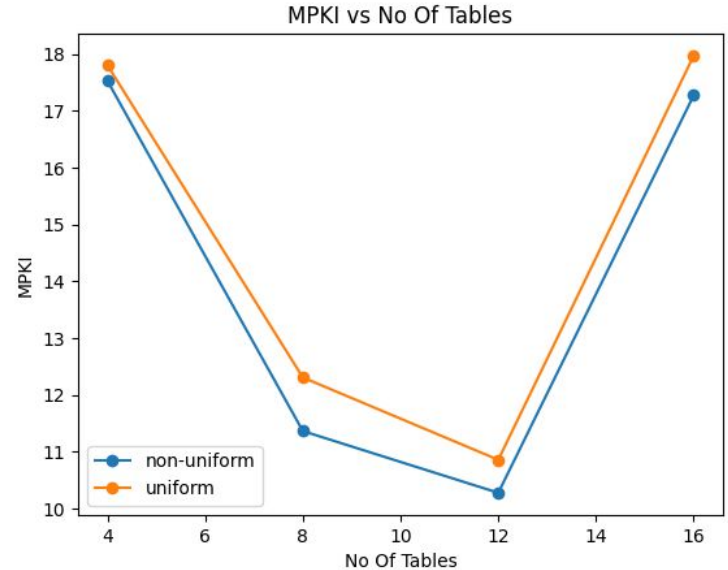
Inferences

- This graph shows how the value of the alpha influences the MPKI or Prediction accuracy
- The value of alpha is found to be optimal at 1.6
- This experimented with the number of tagged components being 12 .



Inferences

- This is an experiment, keeping the tag lengths equal and index lengths equal in all the tagged components.
- This shows that optimal values for each tagged components may not be equal.
- The tag lengths were set to 8 and the index bits were 10 bits long





Pros

- Works better than bimodal and works as good as hashed perceptron for SAT solver traces.
- The prediction given always has high confidence.
- It is easily extensible or It can be easily extended into various other versions as well.

Cons

- More prediction time, It may cause late branch prediction.
- The loop predictor doesn't work well for SAT solver traces. It has to be improved



Where is our project ?

- The GitHub Repo link for our project : [link](#)
- The implementation of LTAGE is in the folder branches and file: LTAGE.bpred.
- The folder has other files that were used for experimentation.
- To run the branch predictor for all the traces for 30 M warmup instructions and 30M simulated instructions, run the following command
 - `./run.sh branchpredictor`



Contributions

Gowtham	Code, Graph-generation
Naman	Code, Slides
TulsiRam	Code, Slides



Resources Used

- [LTAGE Predictor](#)
- [Branch Prediction Github Repo](#)
- [PPM-like TAGE predictor](#)
- [COTTAGE - Advances in TAGE](#)