IMPLEMENTING TAGE PREDICTOR FOR SAT-SOLVERS

BY CORE CRAFTERS

M.RISHI NAIK - 210050101

ABHIRAM -210050042

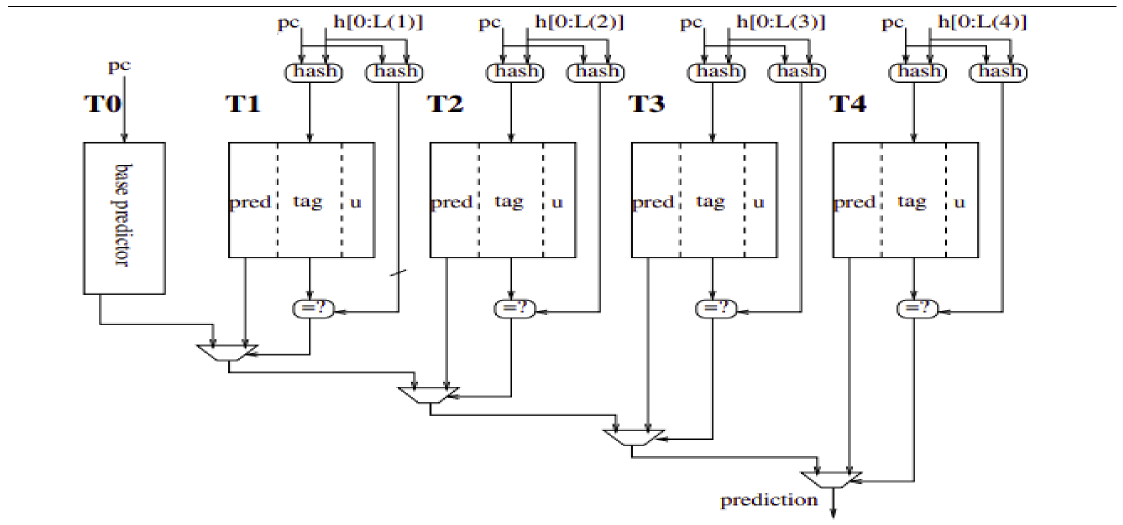
MAHARSHI ERATA - 210050049

RAMASWAROOP RATHI - 210050133

WHY ANOTHER PREDICTOR?

- *A one percent increase in the branch accuracy can help can save millions of cycles in the programs*
- *Therefore, Implementation of new branch predictors are critical in improving performance*

TAGE PREDICTOR



- *Consists of base bimodal predictor backed up with a several tagged component predictors*
- *Entries in the tagged predictors are indexed using different history lengths which form a geometric series*
- *As we can check, there are a multiple predictions. Use the prediction given by the component with longest history with prediction bits not weak. Else use the alternate prediction*
- *Each entry in a tagged components has the info about pred(3bits), tags(#bits depends on the component) and u(2 bits)*

OVERVIEW OF TAGE

Making a Prediction

Observations regarding the prediction

Update the tables (based on outcome)

MAKING A PREDICTION

- *The tagged predictor components are indexed using different history lengths that form a geometric series.*
- *The provider component is the matching component with the longest history. The alternate prediction is the prediction that would have occurred if there had been a miss on the provider component.*
- *The base predictor provides a default prediction. The tagged components provide a prediction only on a tag match.*
- *the overall prediction is provided by the hitting tagged predictor component that uses the longest history, or in case of no matching tagged predictor component, the default prediction is used*

OBSERVATIONS REGARDING THE PREDICTION

- *It was observed that for newly allocated entries, using the alternate prediction results in higher accuracy. On the predictor an entry is classified as "newly allocated" if its prediction counter is weak.*

- *Predictor components are indexed using a hash function of the program counter(ip) , path history register and tag inside for every entry is calculated based on hash function using program counter(ip) and global history register.*

UPDATE THE TABLES(ACCORDING TO RULES)

Updating useful bit of entry hit:

- *Updated if alternate pred != final prediction. Increment if the final prediction is right, else decrement it.*
- *The 1st bit of all the useful bits are reset after a certain clock cycles. Reset the 0th bits after the same clock cycles afterwards. This alternate resetting is done periodically*
- *If the prediction is correct. update the prediction counter*
- *At most one entry is allocated on misprediction*

UPDATE THE TABLES

- *Make a prediction. Whether it's right or wrong check if it's because of the alternate prediction (USE_ALT_ON_NA>8 && pred is weak)and update USE_ALT_ON_NA.*
- *If the prediction is right, increment the update prediction counter.*
- *If the prediction is incorrect, First we update the provider component prediction counter.*
- *As a second step, if the provider component Ti is not the component using the longest history (i.e., i < M) ,*
- *we try to allocate an entry on a predictor component Tk using a longer history than Ti (i.e., i < k < M).*

UPDATE THE TABLES

- *The M-i-1 uj counters are read from predictor components Tj, i < j < M. Then we apply the following rules.*
  - *Is there a k such that u_k = 0?*
  - *If exactly 1 such k, Allocate the component T_k*
  - *More than 1 such k, if u_j=u_k=0 for j<k then allocate to T_j with 2x the probabilty of T_k*

  - If there is no such k, decrement the u counters for all T_j, I<j<M

LOOP PREDICTOR

- *Identifies regular loops with constant number of iterations*
- *Provides a global prediction when loop has been already executed 3 times successively with same number of iterations*
- *We use this when confidence of loop predictor is high enough*
- *Replacement policy of loop predictor tables is based on age*

REFERENCES

- https://jilp.org/vol9/v9paper6.pdf
- http://www.irisa.fr/caps/people/seznec/JILP-COTTAGE.pdf
- https://jilp.org/cbp2016/program.html