CSED311 Lab8
Advanced Branch Prediction

Sang-Youn Kwak
ksy109@postech.ac.kr

Spring 2019

Advanced Branch Prediction

- Your goal is to implement the branch predictors below in Verilog
  - As independent modules (not integrated to your Verilog implementation of CPU)

1. Basic branch predictor

- See the next slide
- Or you can implement two-level branch predictor (covered in lecture 10)

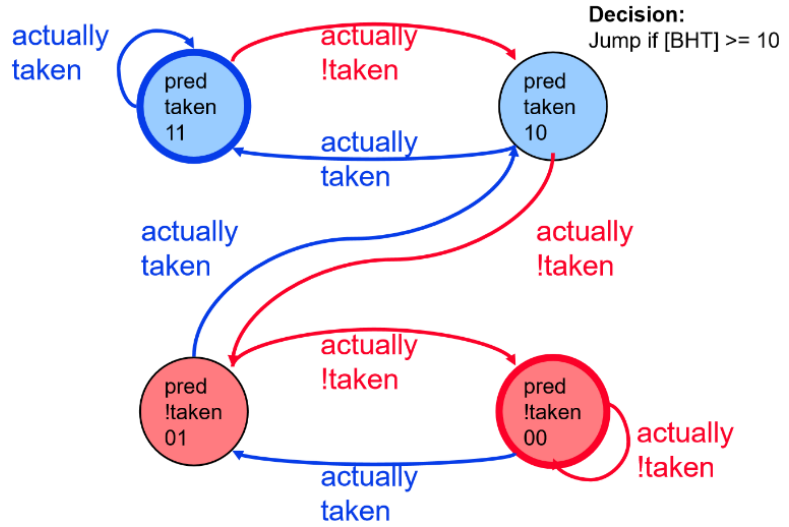
2. Perceptron branch predictor

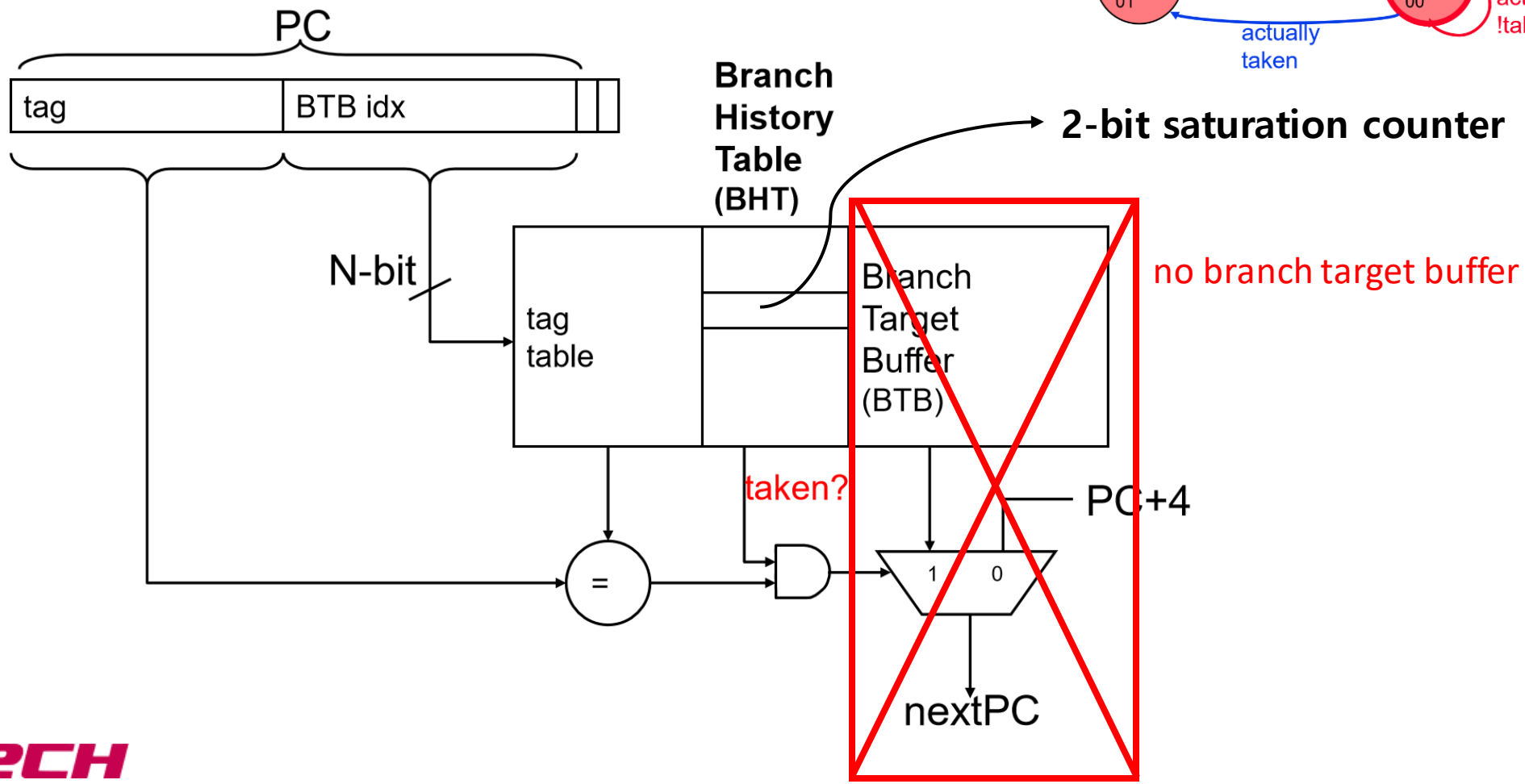
- Read the paper below
- Jiménez and Lin, "Dynamic branch prediction with perceptrons," HPCA'01

- This will be the last assignment.

Assignment

"direction" prediction only – no branch target buffer
Use 2-bit saturation counter





Assignment

- Skeleton codes will be provided
  - basic_branch_predictor.v
  - perceptron_branch_predictor.v

- We provide two testbenches
  - basic_branch_predictor_tb.v (for basic branch predictor)
  - perceptron_branch_predictor_tb.v (for perceptron branch predictor)

- and also “f” versions (output to a file)
  - basic_branch_predictor_tb_f.v
  - perceptron_branch_predictor_tb_f.v

Skeleton code

- You don’t have to implement the whole CPU
- Just implement branch predictor
  - “direction” prediction only – no branch target buffer
  - Module has four inputs and one output
  - Table size: 256 entries (default)
- Input
  - Clock, reset
  - 64-bit instruction address (branch instructions only)
  - Branch outcome (1-bit, taken (1) or not taken (0)) for the instruction address given in the previous cycle
    → Use this to update the internal state of the branch predictor
- Output
  - Predicted branch outcome (1-bit, taken (1) or not taken (0))

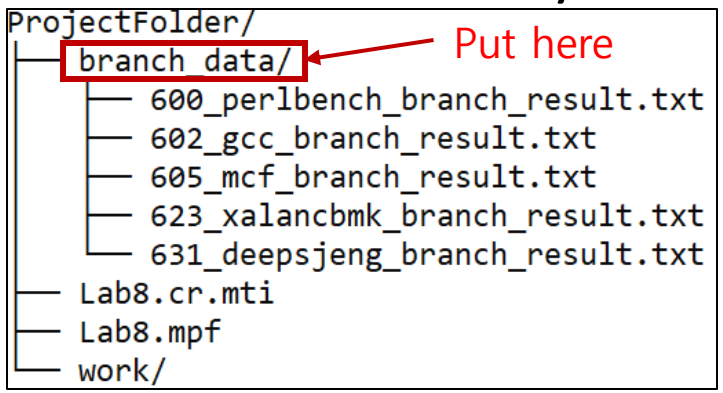
Input and output

- In each cycle
  - Module will get input “ip” (instruction pointer, instruction address)
  - Module should output prediction (predicted branch outcome) in the following cycle.
  - Testbench will compare the output of your module with the actual branch outcome
  - Module receives the actual branch outcome for the previous cycle's IP as the input “taken” and uses it to update its internal states.

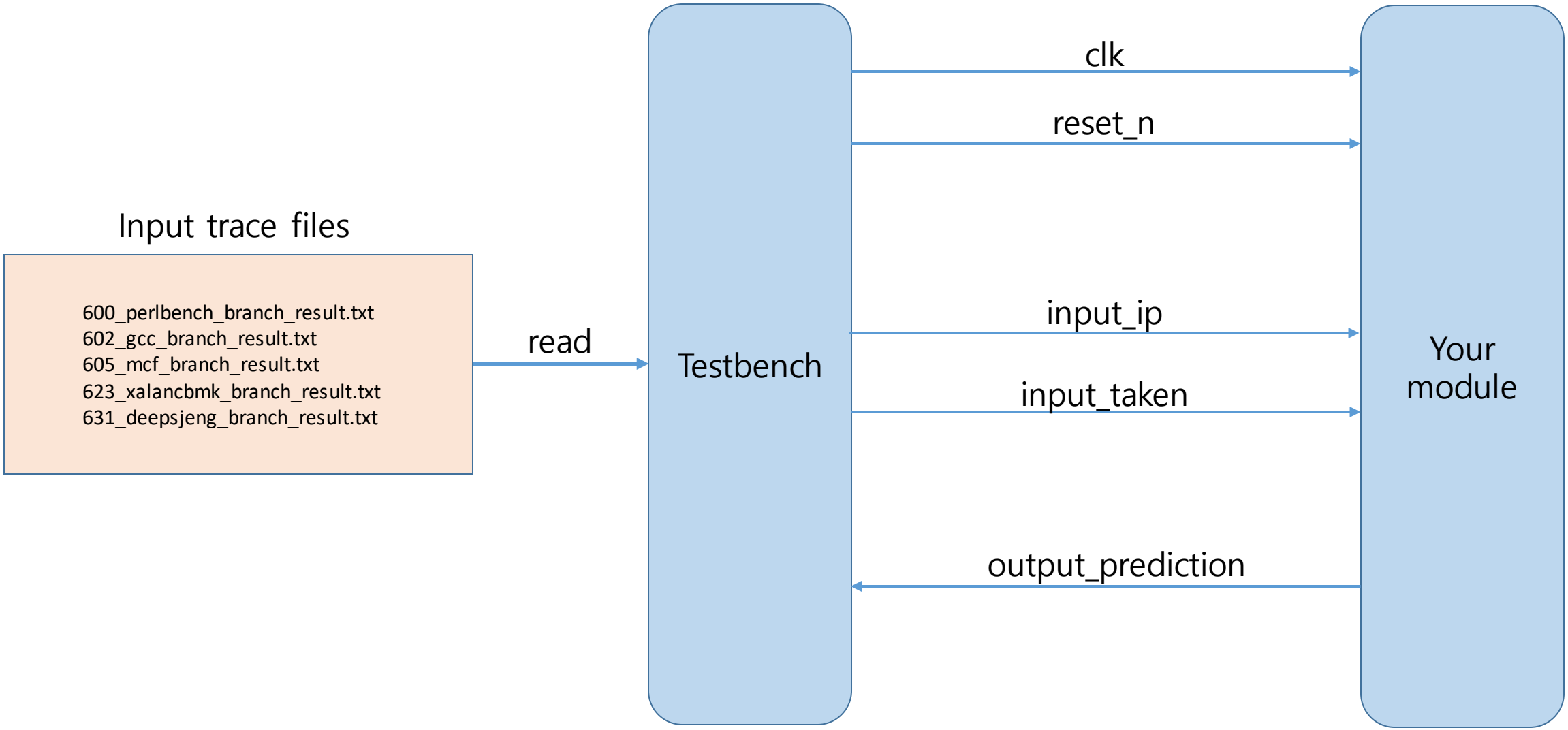
- The process continues until the end of input trace files.

Testbench

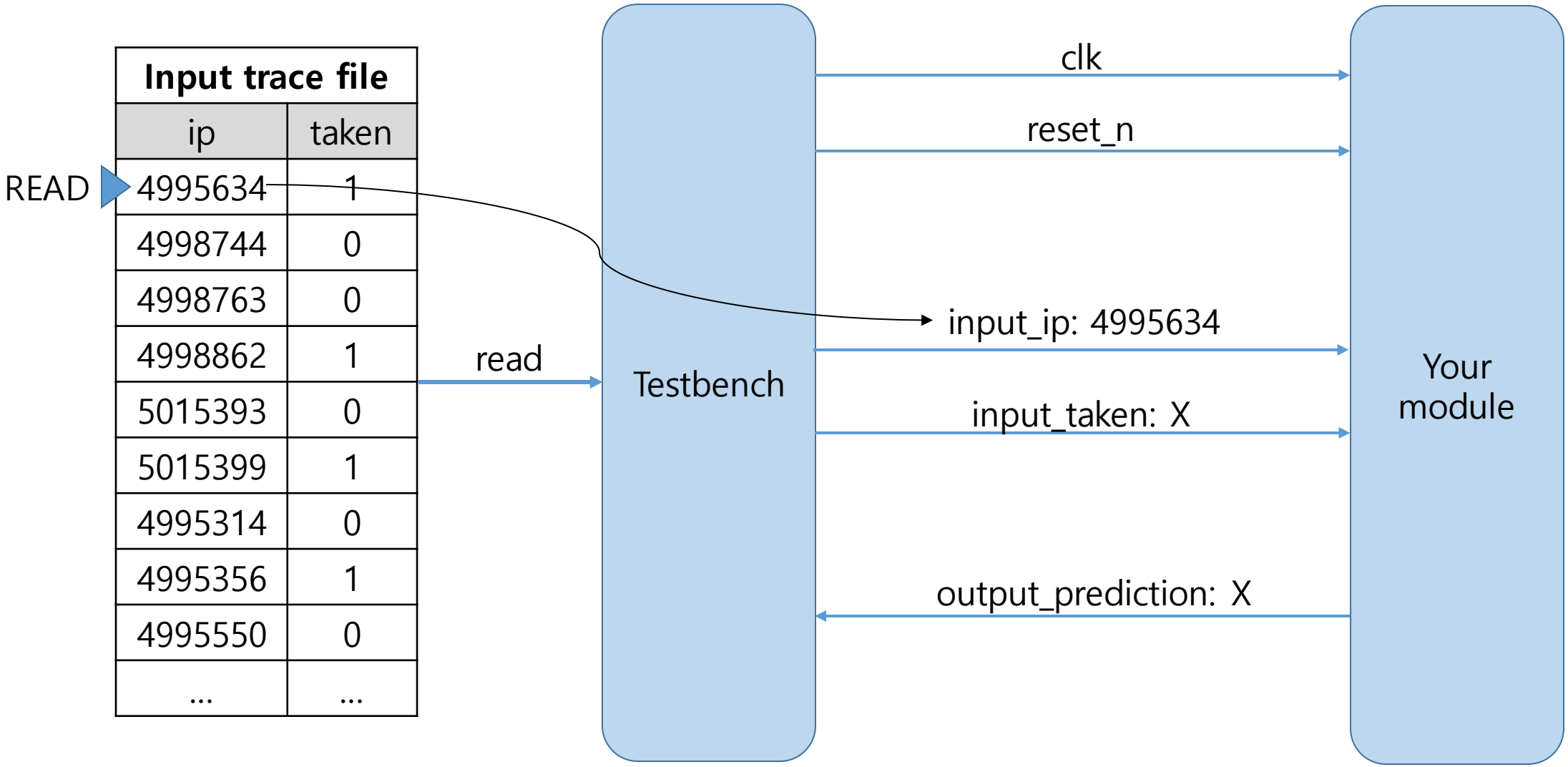
- 5 test cases - will be provided in text file (input trace files)
  - 600_perlbench_branch_result.txt (about 15MB)
  - 602_gcc_branch_result.txt (about 22MB)
  - 605_mcf_branch_result.txt (about 11MB)
  - 623_xalancbmk_branch_result.txt (about 28MB)
  - 631_deepsjeng_branch_result.txt (about 12MB)
- Put those files in your project folder
  - Those files contain address and taken/not taken information
  - Testbench will read those files and test your module



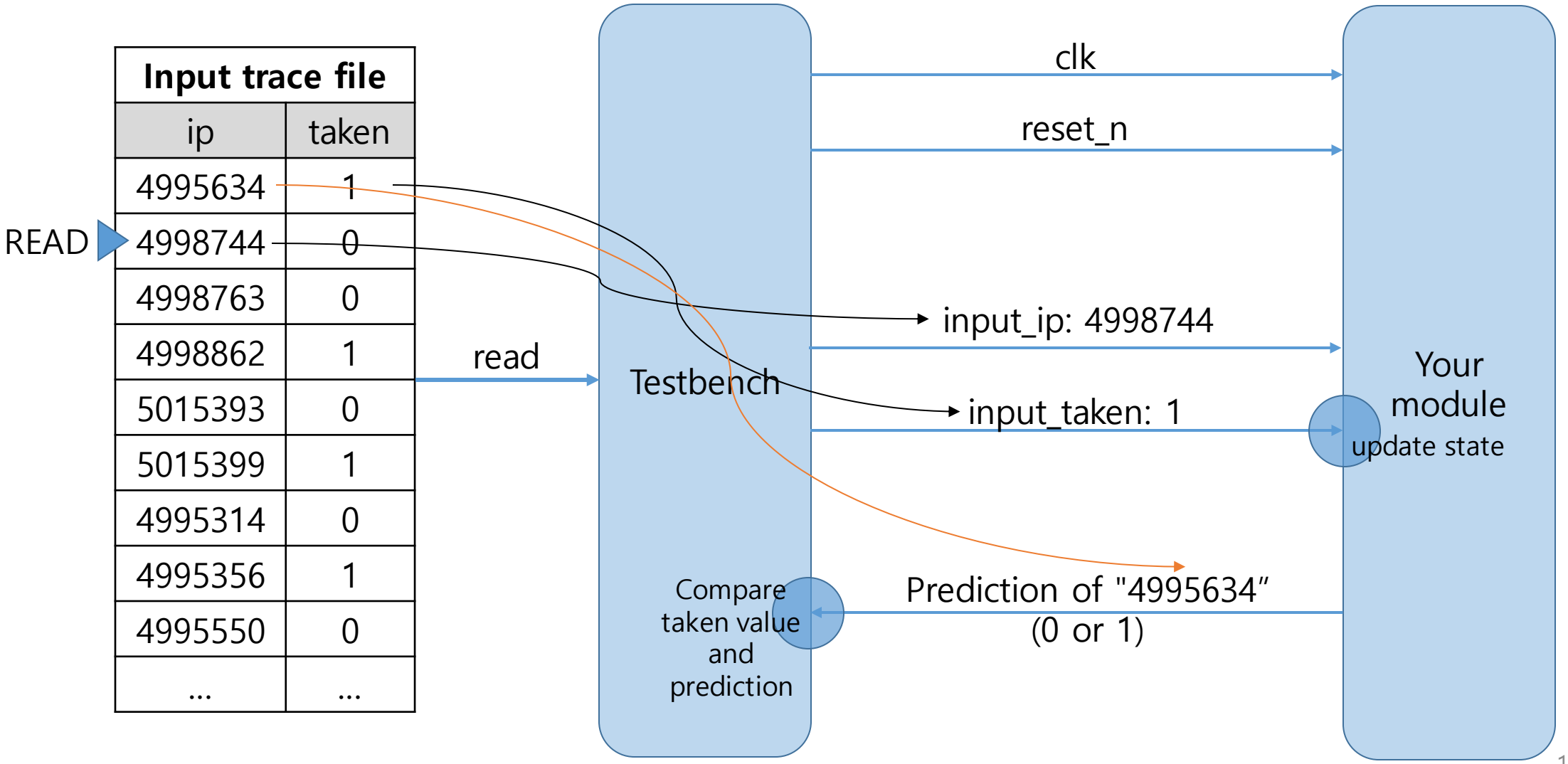
Overall structure



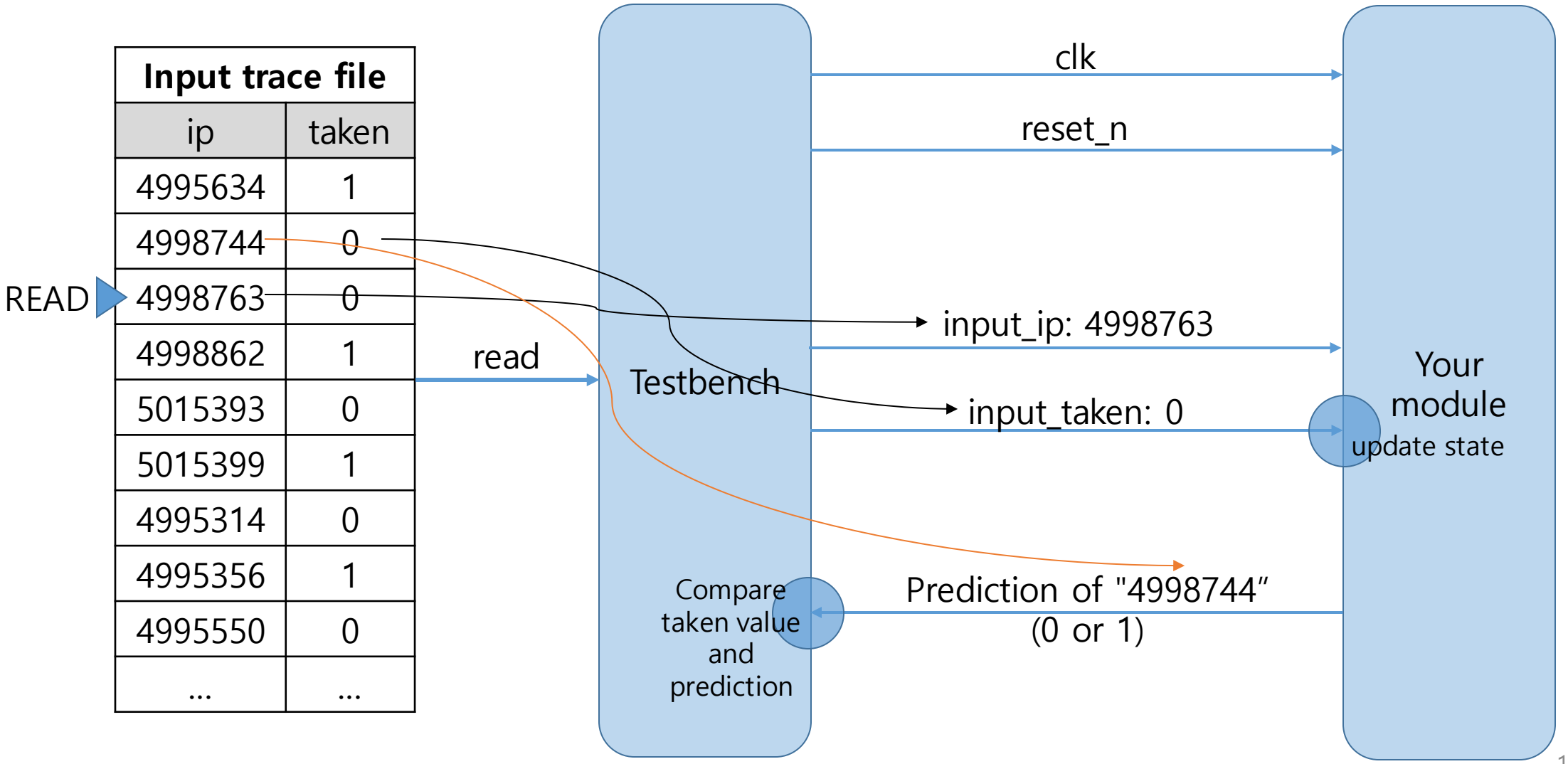
Example (clock cycle 0)



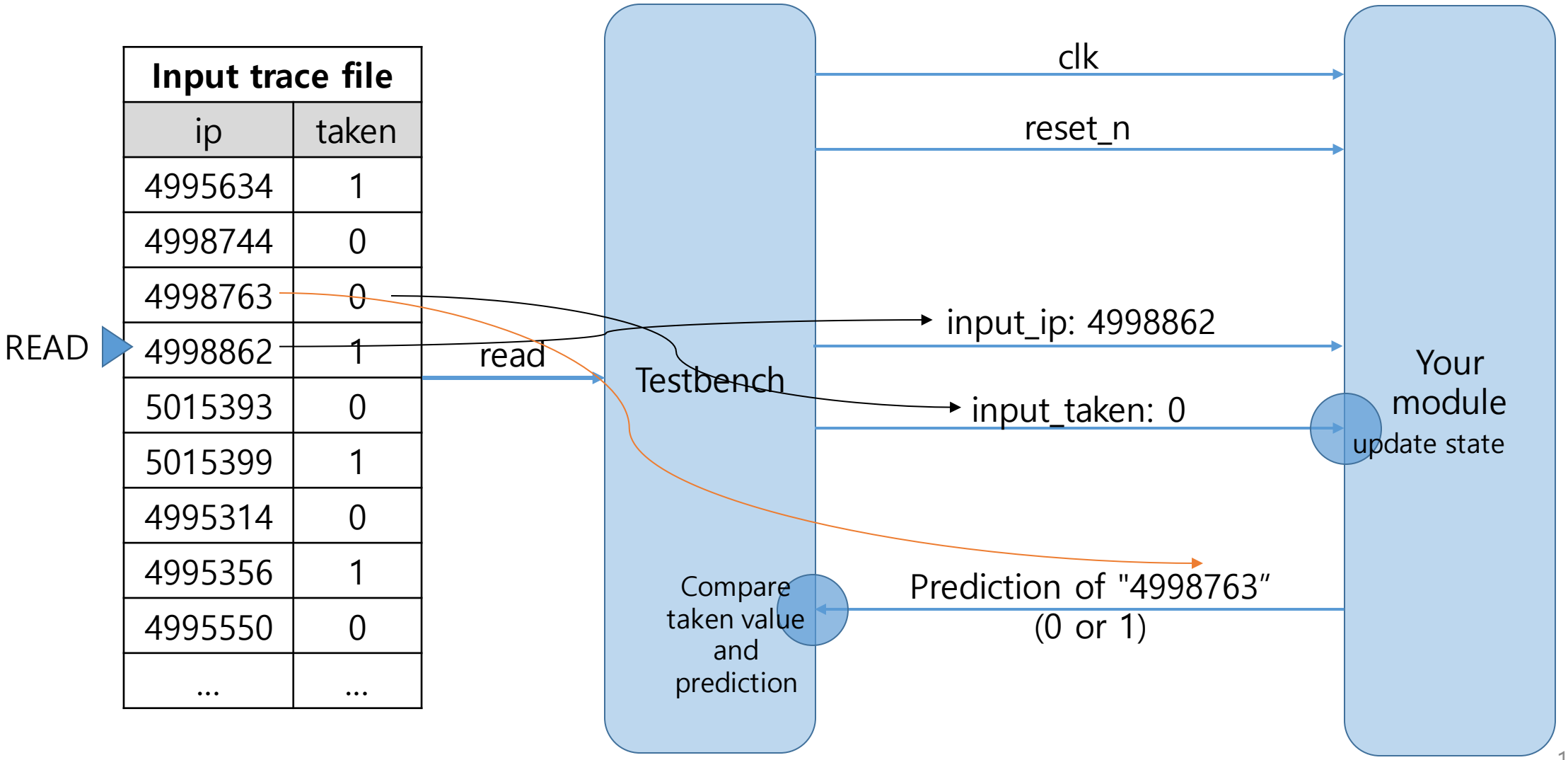
Example (clock cycle 1)



Example (clock cycle 2)



Example (clock cycle 3)



Testbench

- You will get output like this →

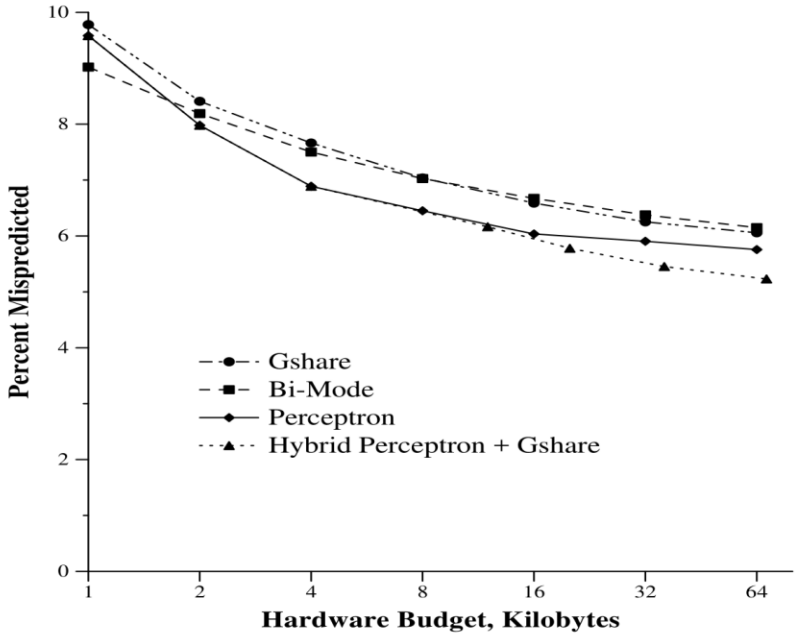
- Test files are large, so it will take a long time

(you can shorten input trace files for debugging)

```
# two level predictor
# test 600 finished
# test 602 finished
# test 605 finished
# test 623 finished
# test 631 finished
# all finished
# ================================
# test - 600:
#       instruction:           5
#       misprediction:           3
#       prediction accuracy: 40.00%
# ================================
# test - 602:
#       instruction:           5
#       misprediction:           2
#       prediction accuracy: 60.00%
# ================================
# test - 605:
#       instruction:           5
#       misprediction:           3
#       prediction accuracy: 40.00%
# ================================
# test - 623:
#       instruction:           5
#       misprediction:           3
#       prediction accuracy: 40.00%
# ================================
# test - 631:
#       instruction:           5
#       misprediction:           0
#       prediction accuracy: 100.00%
# ================================
```

Extra credit (+5%)

- Design space exploration
  - Compare the performance of the basic predictor (or two-level branch predictor) and the perceptron predictor for different table sizes (64, 128, 256, 512, and 1024 entries).



Perceptron vs. other techniqes, Harmonic Mean

x axis: # of table entries
y axis: misprediction rate

Schedule

- Due date: May 28th 18:30
- We will meet for quiz and demo

Q&A

Lab7 Demo