Branch Prediction

Presented By

Dr. Banchhanidhi Dash

School of Computer Engineering
KIIT University

11/18/2023

Review: How to Handle Control Dependences

- Critical to keep the pipeline full with correct sequence of dynamic instructions.

- Potential solutions if the instruction is a control-flow instruction:

- Stall the pipeline until we know the next fetch address
- Guess the next fetch address (branch prediction)
- Employ delayed branching (branch delay slot)

More Sophisticated Direction Prediction

- Compile time (static)
  - Always not taken
  - Always taken
- Run time (dynamic)
  - Last time prediction (single-bit)predictor
  - Two-bit counter based prediction

Static Branch Prediction

- Always not-taken
  - Simple to implement: no need for BTB, no direction prediction
  - Low accuracy: ~30-40%
  - Compiler can layout code such that the likely path is the "not-taken" path

- Always taken
  - No direction prediction
  - Better accuracy: ~60-70%
    - Backward branches (i.e. loop branches) are usually taken
    - Backward branch: target address lower than branch PC

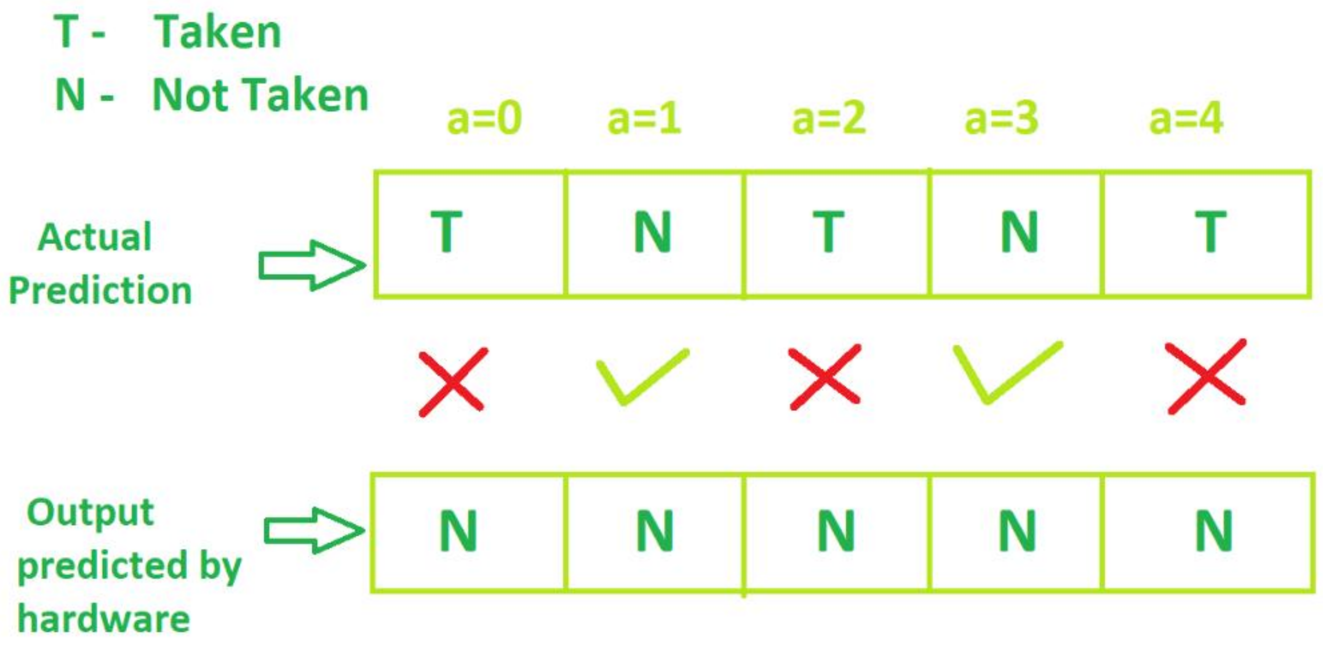
Static branch prediction

In case of Static branch prediction technique underlying hardware assumes that either the branch is not taken always or the branch is taken always.
Let us understand branch prediction with an example code:

```
//Code
int a=0;
while(a<5)
 {
   //branch instruction, condition either true or false
   if(a%2==0)
    {
      b=b+10;
    }
   a++;
 }
```

ASSume branch not taken always(N)

Let us assume that underlying hardware has assumed that branch is not taken always. The output predicted by underlying hardware and actual output is shown in fig:



Dynamic Branch Prediction

- Idea: Predict branches based on dynamic information (collected at run-time)

- Advantages

  + Prediction based on history of the execution of branches

  + It can adapt to dynamic changes in branch behavior
- Disadvantages

  -- More complex (requires additional hardware)

Dynamic Branch Prediction

Dynamic Branch Prediction Technique :
In Dynamic branch prediction technique prediction by underlying hardware is not fixed, rather it changes dynamically.This technique has high accuracy than static technique.

Some dynamic branch prediction techniques are:

1-bit predictor(last time predictor)
2-bit predictor(2 bit saturating counter)

Review: Branch Prediction

- Idea: Predict the next fetch address (to be used in the next cycle)

- Requires three things to be predicted at fetch stage:
  - Whether the fetched instruction is a branch
  - (Conditional) branch direction
  - Branch target address (if taken)

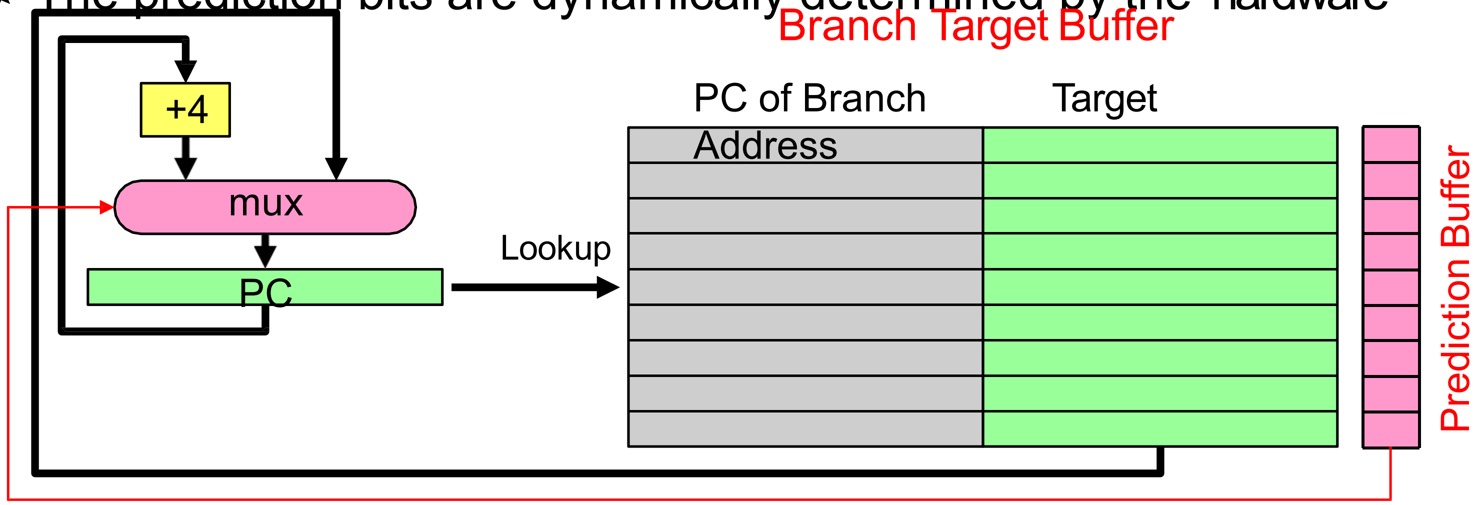
- Observation: Target address remains the same for a conditional direct branch across dynamic instances
  - Idea: Store the target address from previous instance and access it with the PC
  - Called Branch Target Buffer (BTB) or Branch Target Address Cache

Zero-Delayed Branch

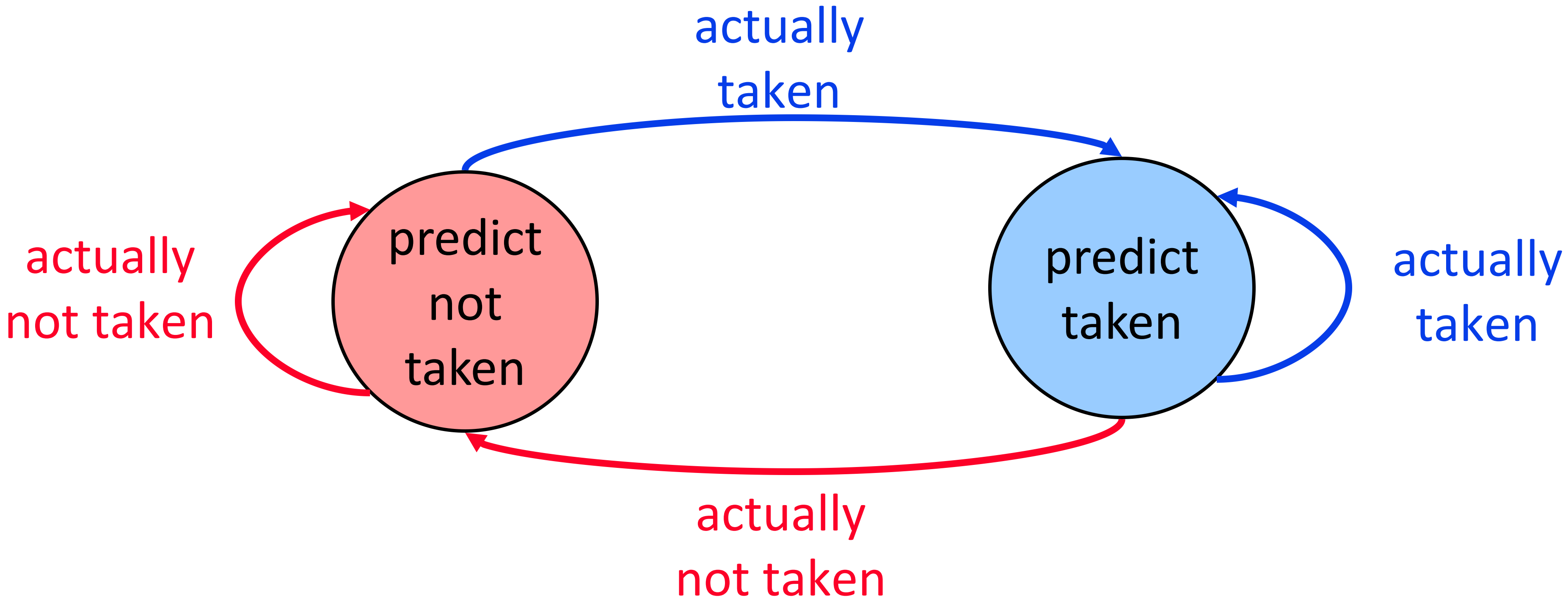
- How can we achieve zero-delay for a taken branch …
  - If the branch target address is computed in the ID stage ?
- Solution
  - Check the PC to see if the instruction being fetched is a branch
  - Store the branch target address in a table in the IF stage
  - Such a table is called the branch target buffer
  - If branch is predicted taken then
    - Next PC = branch target fetched from target buffer
  - Otherwise, if branch is predicted not taken then
    - Next PC = PC + 4
  - Zero-delay is achieved because Next PC is determined in IF stage

Branch Target and Prediction Buffer

- The branch target buffer is implemented as a small cache
  - That stores the branch target address of taken branches
- We also have a branch prediction buffer
  - To store the prediction bits for branch instructions
  - The prediction bits are dynamically determined by the hardware

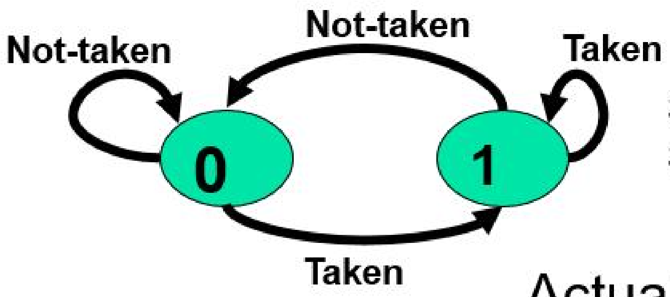


State Machine for 1 bit predictor



1 bit predictor

1-bit branch prediction technique: There are two states 0 and 1. Each state represents one prediction. If anyone makes a wrong prediction then move to another state.



State 0: Predict branch not to be taken.
State 1: Predict branch has to be taken.

Actual sequence: T, NT, NT, NT, T
initial state : T

Prediction: T, T, NT, NT, NT
Accuracy= 3/5=60%

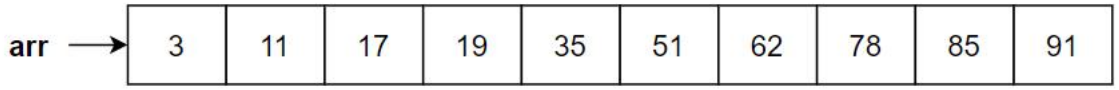
```
for( int i = 0; i < arraySize ;i++)
{
    if( arr[i] >= 50)
    {
        // Inner section
    }

    // Outer section
}
```

Code

1-bit dynamic

This relies on the most recent result of the conditional operator. If it is the first time conditional operator is being executed the prediction can be made randomly



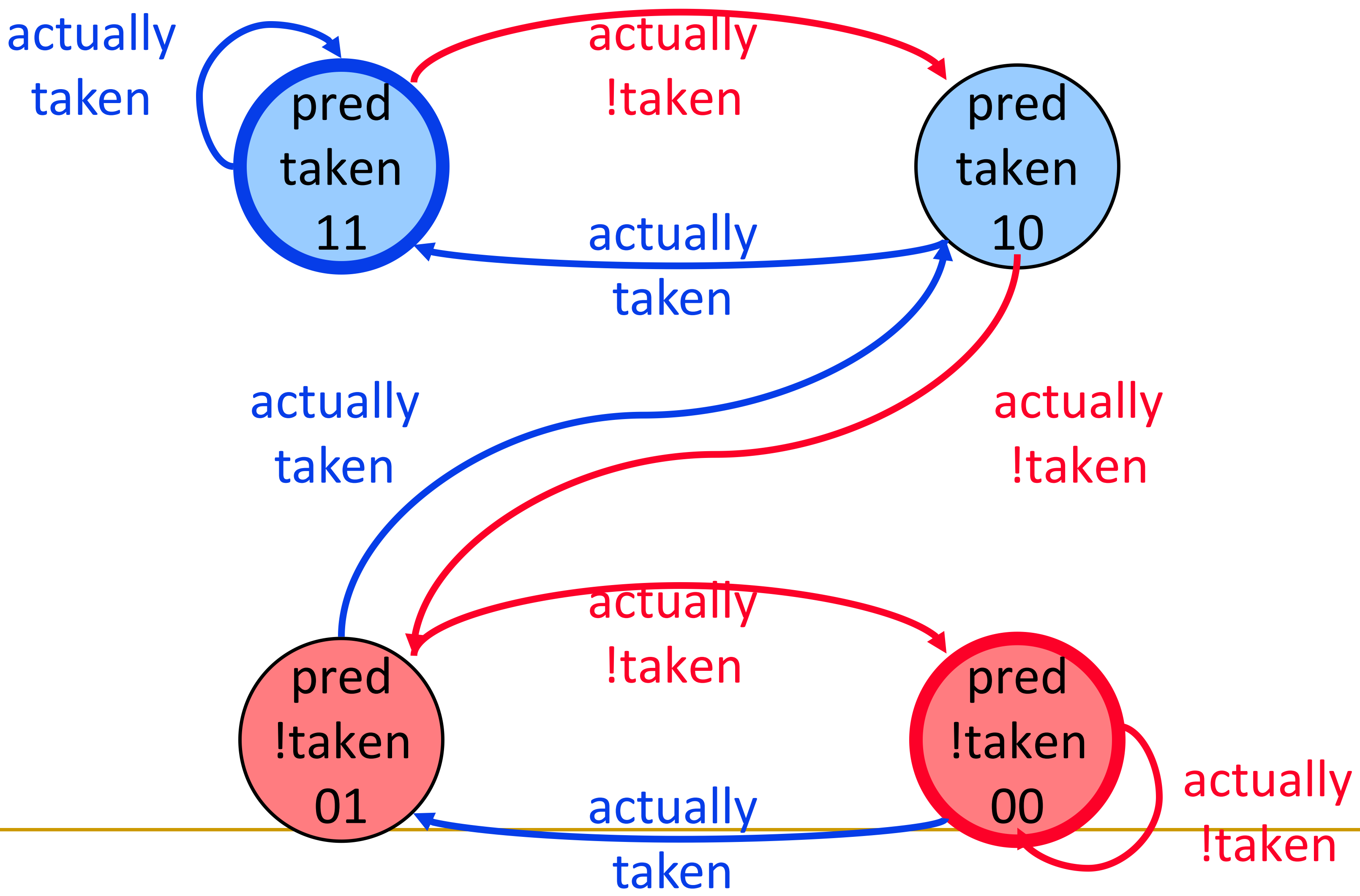
Improving the Last Time Predictor

- Problem: A last-time predictor changes its prediction from T→NT or NT→T too quickly
  - even though the branch may be mostly taken or mostly not taken

- Solution Idea: Add hysteresis to the predictor so that prediction does not change on a single different outcome
  - Use two bits to track the history of predictions for a branch instead of a single bit
  - Can have 2 states for T or NT instead of 1 state for each

State Machine for 2-bit predictor

Change prediction after 2 consecutive mistakes



```
for( int i = 0; i < 100 ;i++)
{
    a = 0;
    while(a < 10)
    {
        // Inner section
        a++;
    }
    // Outer section
}
```

Code

2-bit dynamic

This predictor predicts the branch according to the two most recent results of the conditional operator.

If it is the first time conditional operator is being executed the prediction can be made randomly

- The outcome of a branch is T, NT, T, NT, T and this pattern is repeated 3 times by an outer loop. Find out the no of miss and correct predictions using 2-bit predictor scheme.Assuming the initial
- state is weakly taken(T)and 2 bits counter value is 10_2. ANS:
- Solution:
- Initial = T
- Actual= T,NT,T,NT,T, T,NT,T,NT,T, T,NT,T,NT,T
- Predictions= T,T,T,T,T, T,T,T,T,T, T,T,T,T,T
- No.of correct predictions=9
- No.of correct predictions=6