We implement the A 2-bit saturating counter.

https://courses.cs.washington.edu/courses/csep548/00sp/lectures/class10/sld008.htm

Firstly, we need create a class of Branch prediction class.

Public class BP {

Private int State=2;//the initial value is weakly taken

//strong not taken:0 weakly not taken:1 weakly taken:2 strongly taken:3

Private int[][] Position=new int[16][100];

//storing the position of branch to implement the backtrack. I am not sure whether we need to use this array, We will create multiple-threading, so we maybe can implement backtrack by eliminating extra thread.

Public void ChangeState(int tag){ //tag only can be 1 and 0

If tag==1

If self.State==0

Self.State=1

Elseif self.State==1

Self.State=2

Elseif self.State=2

Self.State=3

Else:

Else if tag==0

If self.State==3

Self.State=2

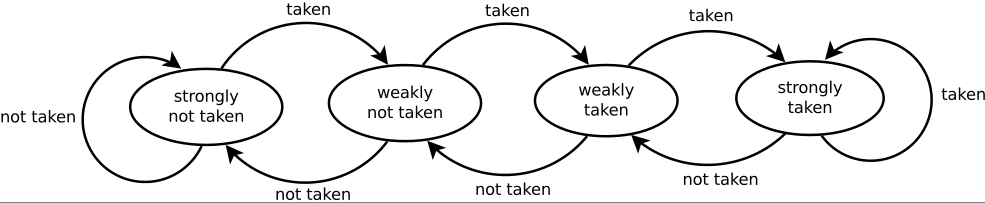
Elseif self.State==2

Self.State=1

Elseif self.State=1

Self.State=0

Else:



Secondly, we need to implement the concurrence. When the decoder decodes the next instruction and finds that it is a branch, the program operate depending the state of Branch prediction Register. If the state is “taken”, program create a new thread. The original thread countinue run. After the getting the real state of branch, comparing real state with state of Branch prediction Register. If it is same, continue. If it is not same, backtrack depending the branch position.

/\*\*\*\*read the next instruction\*\*\*\*\*\*/

/\*\*\*\*find that it is branch instruction\*\*\*\*\*/

Create a new thread to run the code depending on BP.state.

/\*\*\*\*\*run the branch instruction\*\*\*\*\*\*\*/

/\*\*\*\*\*get the judge result of branch instruction\*\*\*\*\*/

Tag=real result

Comparing the real result with BP prediction result

If is different, backtrack

If is same do nothing

BP.ChangeState(Tag)

/\*\*\*\*\*\*read the next instruction\*\*\*\*\*/