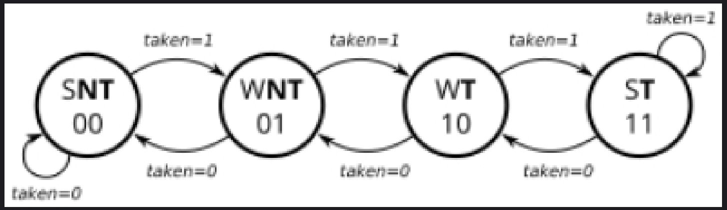
Branch direction predictors are often structured as tables of counters indexed by the program counter and branch history. Each table entry usually uses two-bits of state because one-bit of state (remember last outcome) does not have enough hysteresis and flips states too easily.



State diagram of a two-bit saturating counter. The four states are **S**trong/**W**eak Taken (**T**)/Not-Taken (**NT**).

A two-bit state machine that works fairly well is a saturating counter[1], which counts up to 3 (or 2'b11) or down to 0 (or 2'b00) but does not wrap around. A "taken" result increments the counter, while a "not-taken" result decrements the counter. A branch is predicted to be taken when the count is 2 or 3 (or 2'b1x). Adding some hysteresis prevents a flipping of the prediction when a strongly-biased branch occasionally takes a different direction, requiring two increments in the opposite direction before the prediction is flipped.

References

1. ↑ R. Nair, "Optimal 2-bit branch predictors", *IEEE Trans. Computers*, vol. 44 no. 5, May, 1995

Description

Build a two-bit saturating counter.

The counter increments (up to a maximum of 3) when `train_valid` = 1 and `train_taken` = 1. It decrements (down to a minimum of 0) when `train_valid` = 1 and `train_taken` = 0. When not training (`train_valid` = 0), the counter keeps its value unchanged.

`areset` is an asynchronous reset that resets the counter to weakly not-taken (2'b01). Output `state[1:0]` is the two-bit counter value.