Rheinland-Pfälzische Technische Universität Kaiserslautern Landau

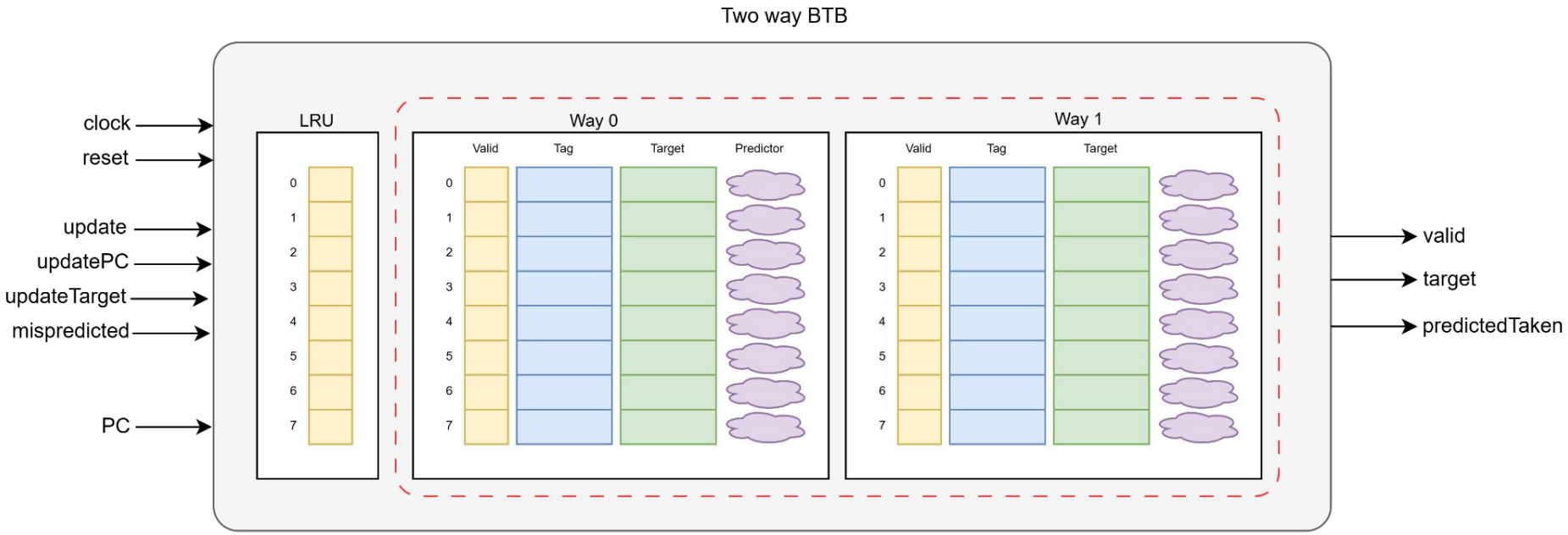
Entwurf informationstechnischer Systeme

BTB Specifications

ADS1 Class Project WS24/25

Tharindu Samarakoon - 430891

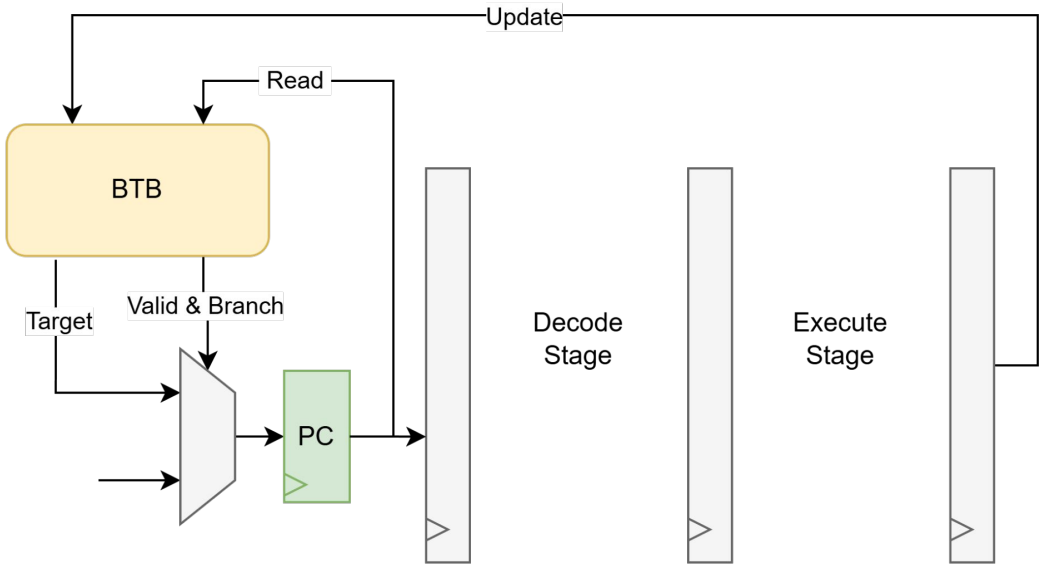
Architecture



Integration with Pipelined Processor

Update:
- Create / update a set in way0/way1
- Initialize LRU

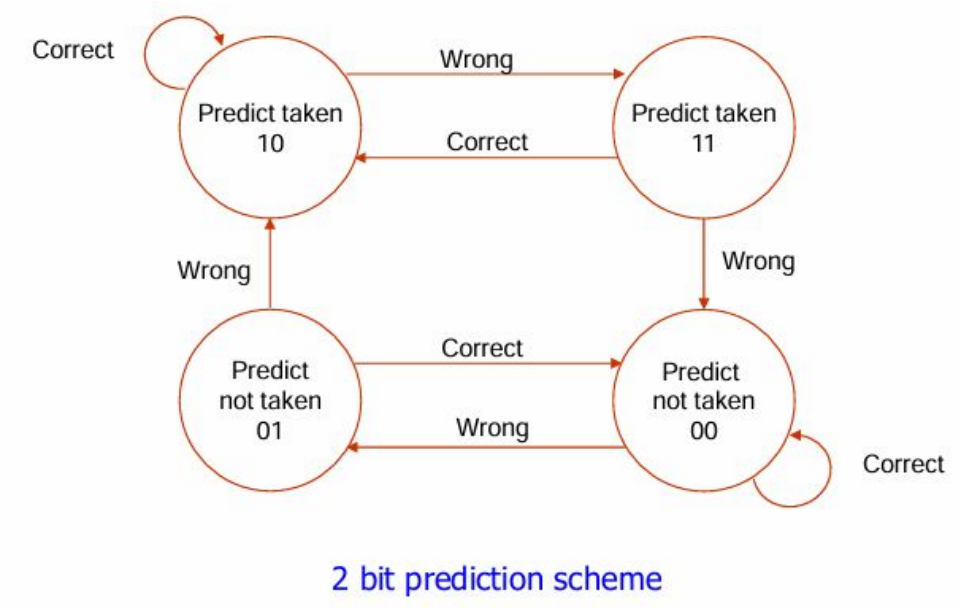
Read:
- Output Target
- Update LRU



Branch Predictor

Initial update request:

1. If (mispredicted == 0)
   - State := Strong not taken (00)
2. If (mispredicted == 1)
   - State := Strong taken (10)



2 bit prediction scheme

Ref: ADS1 slides: arch-2025-6_Instruction-Level_Parallelism

Chisel Testbench

Four scenarios are verified.

1. Add a new value (and read the same value)
   - Cover writing to different sets
2. Update existing value multiple times
   - Cover mispredicted = 1, 0 cases
   - Cover all 4 prediction states
3. Add 2nd new value to same set
   - New value should go to other way
   - Replace previous value only it is not read yet
4. Add 3rd new value to same set
   - Oldest value should be replaced (Cover LRU logic)

```
// TEST2: Update existing value multiple times
// a. Initialize with branch = 0
//1. update set:3, addr: 0x36C, target: 0x7370, mispredicted: 0
dut.clock.step(1)
dut.io.update.poke(1.U)
dut.io.updatePC.poke(0x36C.U)
dut.io.updateTarget.poke(0x7370.U)
dut.io.mispredicted.poke(0.U)

//2. check set:3, addr: 0x36C, target: 0x7370, prediction: 0 (strong not take
dut.clock.step(1)
dut.io.update.poke(0.U)
dut.io.PC.poke(0x36C.U)
dut.io.valid.expect(1.U)
dut.io.target.expect(0x7370.U)
dut.io.predictedTaken.expect(0.U)
```

Note: Other than these tests, there are **Chisel Assertions** inside the BTB for **formal verification**

SystemVerilog testbench

- Driver to send **randomized read requests and update requests**
  - This act as a CPU which generate Branch, Jump and other types of instructions
- **Golden model** of Two Way BTB to compare with the DUT

```
175  class TwoWayBTB_driver #(int NUM_INSTRUCTIONS = 40);
196      constraint instruction_c{
197          foreach (instructions[j]) {
198              instructions[j] dist {BRANCH_INS := 5, JUMP_INS := 1, OTHER_INS:= 20};
199          }
200          instructions[NUM_INSTRUCTIONS-1] == JUMP_INS; // last instruction should jump back to 0th instruction
201          foreach(target_addresses[i]){
202              target_addresses[i] inside {[0:NUM_INSTRUCTIONS-1]};
203          }
204          target_addresses[NUM_INSTRUCTIONS-1] == 0; // last instruction should jump back to 0th instruction
205      }
```

```
95   class TwoWayBTB_model #(int NSETS = 8);
109      function automatic void read(input int PC, ref bit valid, ref int target, ref bit predictTaken);···
138      endfunction
139
140      function automatic bit check_hit(input int PC, output bit wayID);···
152      endfunction
153
154      function automatic void write(input bit update, input int updatePC, input int updateTarget, input bit mispredicted);···
163      endfunction
```

```
452      assert property (valid_check) else begin $error("Dut_valid: %0d != Model_valid: %0d", dut_valid, model_valid); $stop; end
453      assert property (target_check) else begin $error("Dut_target: %0d != model_target: %0d", dut_target, model_target); $stop; end
454      assert property (predictedTaken_check) else begin $error("dut_predictedTaken: %0d != model_predictTaken: %0d", dut_predictedTaken,
```

Design Choices

1. Initial prediction:
   - Decided based on misprediction value.
     i. (Mispredicted == 1) -> Strong_Taken
     ii. (Mispredicted == 0) -> Strong_Not_Taken
2. Read and write the same address at once
   - The read response will be based on older information
     i. This simplifies the implementation
3. When LRU should be updated
   - LRU will be updated **only when reading** not writing.
   - Ex1:- write without reading
     i. Write A: writes to way0
     ii. Write B: writes to way0 (replace old value)
   - Ex2:- write after at least 1 read
     i. Write A: writes to way0
     ii. Read A: update LRU
     iii. Write B: write to way1