Capstone VP Project

Baseline Predictor Specification

Author: Yuhan Li

Revision: 1.0

Date: Nov 22, 2021

Revision History:

|  |  |
| --- | --- |
| Revision | Description |
| 1.0 | Original release |

1. Overview

The Baseline Predictor in the Capstone VP project is a last-value predictor. It is the fundamental building block of all major types of value prediction algorithms. However, this block is intended to be a standalone predictor that can be directly integrated into the CPU core instead of a subblock within other predictors. Within the Capstone VP project, the Baseline Predictor will be the first VP unit to be integrated to the CPU for testing the VP-CPU interface.

Since the target CPU is the SweRV EH1 core, the baseline predictor will be able to support generating and updating up to two instructions per cycle. Furthermore, the block is parameterized for flexible testing and experimenting, including storage size and confidence. Finally, there could be future work for parameterized output delay of the Baseline Predictor to emulate different predictors for better testing of the VP-CPU interface.

1. Block Parameters

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Acceptable Range | Default | Description |
| P\_STORAGE\_SIZE | ≥1 | 2048 | max number of last-values the predictor stores in the last-value table |
| LP\_INDEX\_WIDTH | log2(P\_STORAGE\_SIZE) | | The address width used to index the last-value table, automatically calculated. |
| P\_CONF \_WIDTH | ≥2 \* P\_NUM\_PRED | 8 | Produces a confident prediction when this bit is set in the confidence counter, i.e., the predictor produces a confident prediction when the estimated probability of error is ≤ 1/(2^P\_CONF \_WIDTH) |
| P\_NUM\_PRED | 1 or 2 | 2 | Max number of concurrent predictions the predictor is able to make and update. |
| P\_OUT\_DELAY | Reserved | | |

1. Block Interfaces and IO Signals

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Width | Interface | Description |
| clk\_i | 1 |  | Main clock |
| rst\_i | 1 |  | Active high reset |
| fw\_pc\_i | P\_NUM\_PRED\*32 | Prediction Input | Instruction address used for prediction |
| fw\_valid\_i | P\_NUM\_PRED | Determines if each input PC is valid and usable |
| pred\_pc\_o | P\_NUM\_PRED\*32 | Prediction Output | PC of the instruction of which the output prediction result is for. The same as fw\_pc\_i delay matched with prediction result output path |
| pred\_result\_o | P\_NUM\_PRED\*32 | The actual prediction result for the instruction address |
| pred\_conf \_o | P\_NUM\_PRED | Indicates if pred\_conf\_o has been saturated |
| pred\_valid\_o | P\_NUM\_PRED | Determines if the prediction output is valid and usable. The same as fw\_valid\_i delay matched with prediction result output path |
| fb\_pc\_i | P\_NUM\_PRED\*32 | Feedback | The address of the instruction prediction to be updated |
| fb\_actual\_i | P\_NUM\_PRED\*32 | The actual execution result of the instruction |
| fb\_mispredict\_i | P\_NUM\_PRED | Indicates the prediction of fb\_pc\_i was wrong |
| fb\_valid\_i | P\_NUM\_PRED | Determines if the feedback input is valid and usable |

1. Functionality Requirements
   1. Forward Prediction
2. The predictor shall index the last-value table and the confidence table using the fw\_pc\_i for each prediction in P\_NUM\_PRED.
3. The predictor shall use the tables’ output as the prediction result.
4. The predictor shall produce valid output for each prediction in P\_NUM\_PRED when its corresponding fw\_valid\_i is set.
5. The input PC and valid bits shall be delay matched with the table read delay.
6. The prediction output may forward the updated confidence counter when valid forward and feedback PCs match.
   1. Feedback and Update
7. Upon valid correct prediction feedback, the confidence counter shall be incremented or saturated to 2P\_CONF\_WIDTH-1.
8. Upon valid misprediction feedback, the confidence counter and saturation bit shall be reset.
9. When two valid updates try to write to the same entry, the block shall merge the two updates into one, by:
   * 1. add 2 to the confidence counter when both predictions were correct, or
     2. reset confidence counter and store the second result when two actual execution results disagree or both predictions were same mispredictions.
10. for each entry in the last-value table, the block shall capture the rising edge of the saturation of the confidence counter and store it for the output pred\_conf\_o.
11. Testing Requirements
12. The tests shall exercise at least all functional requirements under P\_NUM\_PRED=2.
13. The C model shall be bit-accurate.
14. There shall be directed tests for merging concurrent updates, confidence saturation, and confidence reset, and randomized input valid signals (fw\_valid\_i and fb\_valid\_i).