BRANCH PREDICTOR DRAC specification version v0.1

Victor Soria Pardos

March 25, 2020

Contents

1 General purpose of the module

Branch predictor is composed by two modules, the top view called branch_predictor, and the bimodal_predictor.

The purpose of this modules is to predict if a given program counter is a branch, and if so predict if the branch is going to be taken and where is going to jump. Then the prediction is used to fetch the following instructions.

Each time Drac fetches an instruction, it verifies at execution stage if the fetched instruction is a branch. Moreover Drac also computes at execution stage what should have been the next program counter after it. This information is forwarded to the branch prediction to improve the prediction of future program counters, and in case of missprediction correct the program counter at fetch stage.

2 Design placement

Branch prediction modules are meant to be placed inside the fetch_stage module.

3 Parameters

All parameters, enums and types used in this module are defined in drac_pkg or risc_pkg.

4 Interface

In this section we describe the interface signals of each module for the different components connected to it.

4.1 Bimodal Predictor Interfaces

4.1.1 Input Interface

Signal name	Width or Struct	Input	Description
clk_i	1	bimodal_predictor <- branch_predictor	Clock for the module
pc_fetch_i	64	bimodal_predictor <- branch_predictor	Program counter at fetch stage
pc_execution_i	64	bimodal_predictor <- branch_predictor	Program counter at execution stage
branch_addr_result_exec_i	64	bimodal_predictor <- branch_predictor	Address generated by branch instruction at execution stage
branch_taken_result_exec_i	1	bimodal_predictor <- branch_predictor	Branch at execution stage is taken or not
is_branch_EX_i	1	bimodal_predictor <- branch_predictor	Instruction at execution stage is a branch

4.1.2 Output Interface

Signal name	Width or Struct	Output	Description
bimodal_predict_taken_o	1	bimodal_predictor -> branch_predictor	Bit that encodes branch taken ’1’ or not ’0’
bimodal_predict_addr_o	64	bimodal_predictor -> branch_predictor	Address predicted to jump

4.2 BRANCH PREDICTOR INTERFACES

4.2.1 INPUT INTERFACE

Signal name	Width or Struct	Input	Description
clk_i	1	branch_predictor <- if_stage	Clock for the module
pc_fetch_i	64	branch_predictor <- if_stager	Program counter at fetch stage
pc_execution_i	64	branch_predictor <- if_stage	Program counter at execution stage
branch_addr_result_exec_i	64	branch_predictor <- if_stage	Address generated by branch instruction at execution stage
branch_taken_result_exec_i	1	branch_predictor <- if_stage	Branch at execution stage is taken or not
is_branch_EX_i	1	branch_predictor <- id_stage	Instruction at execution stage is a branch

4.2.2 OUTPUT INTERFACE

Signal name	Width or Struct	Output	Description
branch_predict_is_branch_o	1	branch_predictor -> if_stage	Bit that encodes if the instruction being fetched is predicted as a branch
branch_predict_taken_o	1	branch_predictor -> if_stage	Bit that encodes branch taken '1' or not '0'
branch_predict_addr_o	64	bimodal_predictor -> branch_predictor	Address predicted to jump

5 RESET BEHAVIOR

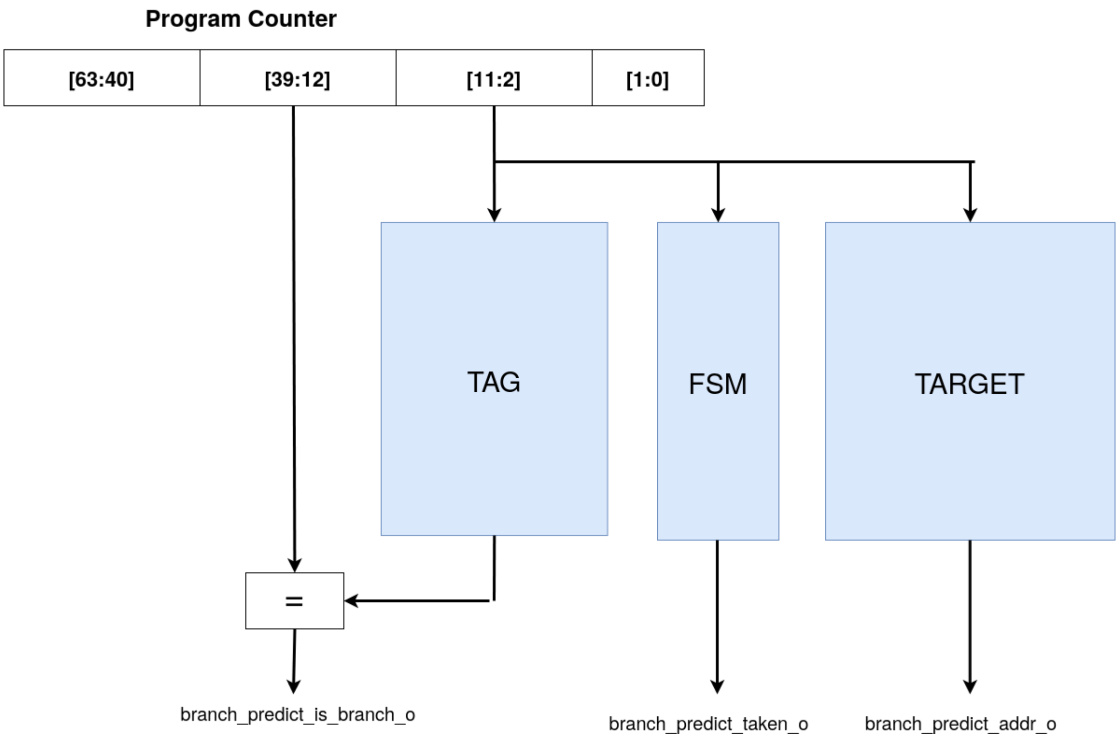
Branch predictor has no reset signal.

6 WHAT COULD NOT HAPPEN

There are no restrictions on inputs or output values. All combinations are possible.

7 Behavior

Branch predictor works as a memory module that contains past branches information, and uses them to predict new values. There are three memory tables, with 1024 entries. These entries are accessed with the bits 11 to 2 of the program counter of the fetch stage.



The first table stores the tags of the branches. A tag is a subset of the bits from 39 to 12 of the PC. We do not store bits 40 to 64 since our processor has support only for 40 bit virtual addresses. Therefore, the table stores 40x1024 bits. The tag read from the table is compared with the equivalent bits of the PC, and if they are equal the BP predicts that the fetched address is a branch. This is done through signal branch_predict_is_branch_o.

The second table stores the target addresses of the branches, the address where the branches jump to. Again this table stores 1024x40 bits. This address is output to signal bimodal_predict_addr_o and to branch_predict_addr_o. Since this signal is 64 bit long, bits 39 to 63 are filled with zeros.

The third table stores the finite machine state of the branches. This machine consists on a 2-bit saturating counter. Therefore the table stores 1024x2 bits. When the branch prediction is doing predictions it reads the table and uses the counter to predict if the branch will be taken or not. If the upper bit is 0 the branch is not taken (values 0 or 1), and if the upper bit is 1 then the branch is taken (values 2 or 3). The decision is output though signals branch_predict_taken_o and bimodal_predict_taken_o.

In order to make good predictions, the tables must be updated with the latest information of the branches. Therefore, the branch prediction is connected to the output of the ALU. In case a branch arrives to the ALU, signal is_branch_EX_i is set to 1. And signals pc_execution_i, branch_addr_result_exec_i, branch_taken_result_exec_i are respectively updated with the program counter of the instruction at execution stage, the address the branch should have jump and the result of the branch (taken or not taken).

8 Special cases, corner cases

No corner cases should be considered