GLCC-uFTB 模块验证报告

李雪涛

October, 2024

版本号：*v1.0*

目录

表格

1 验证对象

uFTB 分支预测器为香山分支预测单元的第一个分支预测器，也是唯一一个能在一个周期内能输出预测结果的分支预测器，其接收来自 s0 通道的分支预测请求与更新请求并向 s1 通道输出分支预测结果。本文旨在说明验证 uFTB 分支预测器验证的步骤以及测试用例。

1.1 uFTB 模块结构介绍

1. 支持基于 FTB 项的预测
2. 支持两比特饱和计数器
3. 支持 s1 通道基础预测结果输出及 meta 信息输出
4. 支持更新请求响应，更新内部 FTB 及两比特饱和计数器。

1.2 uFTB 模块功能介绍

功能点主要分为四个部分，分别为:

1. 接收控制信号
2. 执行更新
3. 维护两比特饱和计数器并根据饱和计数器输出预测结果
4. 提供基础条件分支结果

2 验证方案和验证框架

2.1 验证方案

本文基于 Picker 与 MLVP 库将 Verilog 导出为 python 形式的待测设计（Design Under Test, DUT）。以 python 的形式驱动其顶层接口，观察特定接口行为和结果反馈。

2.2 验证框架

在 Picker 导出的 DUT 基础上，该项目复用 uFTB-with-ftq 所提供的 Bpu-top 模块以及 ftq 模块，并且进行一定程度上的修改使其能够执行测试用例。

参考模型 参考模型用于比对 DUT 的功能行为是否正确。在本次验证中，我们为 uFTB 编写了参考模型，见/env-xs-ov-00-bpu/tests/FTBs/uFTB-func/env/路径下的 uftb_model.py 文件

3 uFTB 功能点和测试点

在本次验证中，拆分了 3 个功能点和 17 个测试点。

3.1 接收控制信号

表 3.1.1: uFTB 功能点 1 接收控制信号

编号	测试点	测试用例
uFTB 1.1	能够接收来自 BPU_top 的 reset 控制信号，若为真，重置预测器和缓存	控制信号测试 1
uFTB 1.2	能够接收来自 BPU_top 的 io_ctrl_ubtb_enable 控制信号，若为真，预测输出均为未命中	控制信号测试 2
uFTB 1.3	能够接收来自 BPU_top 的 io_s0(1、2)_fire_0 控制信号	控制信号测试 3 控制信号测试 4

3.2 执行更新

表 3.2.1: uFTB 功能点 2 执行更新

编号	测试点	测试用例
uFTB 2.1	能够接收更新请求，更新请求若已被缓存则更新命中的缓存，除饱和计数器以外的内容直接覆盖	缓存更新测试 1 缓存更新测试 2 缓存更新测试 3
uFTB 2.2	能够接收更新请求，更新请求若已被缓存则更新命中的缓存，若更新跳转，饱和计数器加一	饱和计数器测试 1-4 饱和计数器测试 6-9 饱和计数器测试 11
uFTB 2.3	能够接收更新请求，更新请求若已被缓存则更新命中的缓存，若更新非跳转，饱和计数器减一	饱和计数器测试 2-5 饱和计数器测试 7-10 饱和计数器测试 11
uFTB 2.4	更新请求若未被缓存，且 FTB 缓存未满，则有空让进	缓存更新测试 5
uFTB 2.5	更新请求若未被缓存，且 FTB 缓存已满，则根据 LRU 算法替换	缓存更新测试 4

3.3 维护两比特饱和计数器并根据饱和计数器输出预测结果

表 3.3.1: uFTB 功能点 3 维护两比特饱和计数器并根据饱和计数器输出预测结果

编号	测试点	测试用例
uFTB 3.1	缓存命中，always taken 为 0，两比特饱和计数器值为 2'b00，需预测为假	饱和计数器测试 4-5 饱和计数器测试 9-10 饱和计数器测试 11

表 3.3.1 – 接上页表格

编号	测试点	测试用例
uFTB 3.2	缓存命中，always taken 为 0，两比特饱和计数器值为 2'b01，需预测为假	饱和计数器测试 3-5 饱和计数器测试 8-10 饱和计数器测试 11
uFTB 3.3	缓存命中，always taken 为 0，两比特饱和计数器值为 2'b10，需预测为真	饱和计数器测试 1-3 饱和计数器测试 6-8 饱和计数器测试 11
uFTB 3.4	缓存命中，always taken 为 0，两比特饱和计数器值为 2'b11，需预测为真	饱和计数器测试 1-3 饱和计数器测试 6-8 饱和计数器测试 11
uFTB 3.5	缓存命中，always taken 为 1，无论两比特饱和计数器，需预测为真	饱和计数器测试 12-14

3.4 提供基础条件分支结果

表 3.4.1: uFTB 功能点 4 提供基础条件分支结果

编号	测试点	测试用例
uFTB 4.1	预测结果将在 s1 流水有效时，预测结果通过 s1 通道进行输出	饱和计数器测试 1-11 缓存更新测试 5
uFTB 4.2	当 s1 流水有效时，若 s1_pc 命中读出 uFTB 缓存的对应一项	饱和计数器测试 1-11 缓存更新测试 5
uFTB 4.3	当 s1 流水有效时，若 s1_pc 未命中输出 hit 为 0	控制信号测试 3
uFTB 4.4	预测结果将在 s2 和 s3 流水有效时，预测结果输出为 last_stage_meta	控制信号测试 4

4 测试环境

4.1 硬件环境

1. AMD Ryzen 7 5800H
2. 16GB 内存

4.2 软件环境

操作系统:

1. 物理机：Windows 11 家庭中文版 23H2
2. 虚拟机：Linux-5.15.153.1-microsoft-standard-WSL2-x86_64-with-glibc2.35-WSL2.2.4.0

测试工具集:

名称	版本号
*python	3.10.12
*pytest	8.2.2
*pytest-xdist	3.6.1
*pytest-reporter-html1	0.9.0
*mlvp	0.0.1-master-6624111
*picker	0.1.0-master-36d6882
*cmake	3.15.7
*gcc/g++	11.4.0
*Verilator	4.218 2022-01-17
*verible-verilog-format	v0.0-3716-g914652db
*SWIG	4.2.1

其中，标注 * 的测试工具为必要的环境。测试代码详见FTBs。

5 uFTB 测试用例

在本次验证中，总共有 23 个测试用例。uFTB 的测试用例分为储存及输出测试、饱和计数器测试、缓存更新测试三个部分。

5.1 储存及输出测试

5.1.1 reset 信号

表 5.1.1: uFTB 测试用例组 1 储存及输出测试 1 reset 信号

测试步骤	预期结果
步骤一：uFTB 预测器初始化，将 reset 置 1 步骤二：送入更新请求以缓存 10 条 FTB 项 步骤三：s0_fire 置为有效，送入上面 10 条 FTB 项的 s1_pc，观察输出结果 步骤四：uFTB 预测器初始化，将 reset 置 1 步骤五：s0_fire 置为有效，送入上面 10 条 FTB 项的 s1_pc，观察输出结果	步骤三：预测结果均为命中且跳转 步骤五：预测结果均为不命中
覆盖测试点 接收控制信号	

5.1.2 io_ctrl_ubtb_enable 信号

表 5.1.2: uFTB 测试用例组 1 储存及输出测试 2 io_ctrl_ubtb_enable 信号

测试步骤	预期结果
步骤一：uFTB 预测器初始化，将 reset 置 1 步骤二：送入更新请求以缓存 10 条 FTB 项 步骤三：s0_fire 置为有效，送入上面 10 条 FTB 项的 pc，观察输出结果 步骤四：s0_fire 置为有效，io_ctrl_ubtb_enable 置为无效，送入上面 10 条 FTB 项的 pc，观察输出结果	步骤三：预测结果均为命中且跳转 步骤四：预测结果均为不命中
覆盖测试点 接收控制信号	

5.1.3 io_s0_fire_0 信号

表 5.1.3: uFTB 测试用例组 1 储存及输出测试 3 io_s0_fire_0 信号

测试步骤	预期结果
步骤一：uFTB 预测器初始化，将 reset 置 1 步骤二：送入更新请求以缓存 10 条 FTB 项	

步骤三：s0_fire 置为有效，送入上面 10 条 FTB 项的 pc，观察输出结果 步骤四：s0_fire 置为有效，io_ctrl_ubtb_enable 置为无效，送入上面 10 条 FTB 项的 pc，观察输出结果	步骤三：预测结果均为命中且跳转 步骤四：预测结果均为不命中
覆盖测试点 接收控制信号	

5.1.4 io_s1_fire_0 和 io_s2_fire_0 信号

表 5.1.4: uFTB 测试用例组 1 储存及输出测试 4 io_s1_fire_0 和 io_s2_fire_0 信号

测试步骤	预期结果
步骤一：uFTB 预测器初始化，将 reset 置 1 步骤二：送入更新请求以缓存 10 条 FTB 项 步骤三：s0_fire 置为有效，io_s1_fire_0 和 io_s2_fire_0 置为无效，送入上面 10 条 FTB 项的 s1_pc，观察输出结果 步骤四：s0_fire 置为有效，io_s1_fire_0 置为无效 io_s2_fire_0 置为有效，送入上面 10 条 FTB 项的 s1_pc，观察输出结果 步骤五：s0_fire 置为有效，io_s1_fire_0 置为有效 io_s2_fire_0 置为无效，送入上面 10 条 FTB 项的 s1_pc，观察输出结果 步骤六：s0_fire 置为有效，io_s1_fire_0 和 io_s2_fire_0 置为有效，送入上面 10 条 FTB 项的 s1_pc，观察输出结果	步骤三：输出的 io_out_last_stage_meta 为 0 步骤四：输出的 io_out_last_stage_meta 为 0 步骤五：输出的 io_out_last_stage_meta 为 0 步骤六：输出的 io_out_last_stage_meta 含有上一周期预测结果，其源信息符合 io_out_last_stage_meta = 213'h0, resp_meta_pred_way_r_1, resp_meta_hit_r_1 的规则
覆盖测试点 接收控制信号	

5.2 饱和计数器测试

5.2.1 br_slot 饱和计数器的更新（2-3-3）

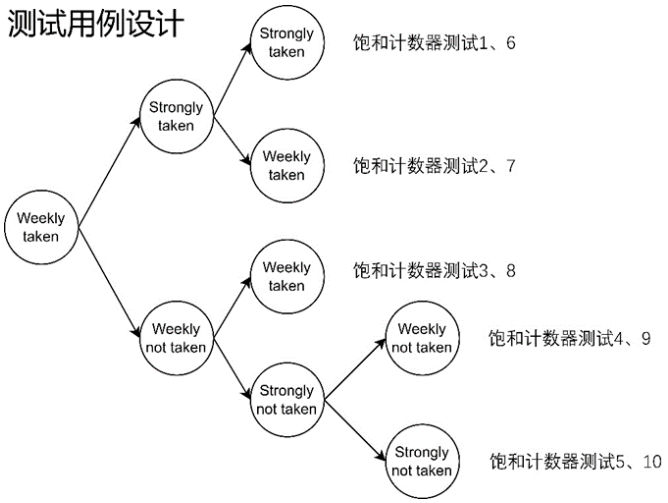

图 1: 饱和计数器测试用例设计

方便起见, 以下内容中, 输入的二元组为 [io_update_bits_br_taken_mask_0, io_update_bits_br_taken_mask_1], 输出的二元组为 [io_out_s1_full_pred_0_br_taken_mask_0, io_out_s1_full_pred_0_br_taken_mask_1]

表 5.2.1: uFTB 测试用例组 2 饱和计数器测试 1

测试步骤	预期结果
步骤一：uFTB 预测器初始化，将 reset 置 1 步骤二：送入更新请求以缓存 1 条 FTB 项，io_update_valid 有效，输入 [1, 1]，空转两个时钟周期等待 FTB 项更新完毕 步骤三：输出该 FTB 项的预测结果 步骤四：io_update_valid 有效，输入 [1, 1]，空转两个时钟周期等待 FTB 项更新完毕 步骤五：输出该 FTB 项的预测结果	步骤三：预测结果为 [1, 1] 步骤五：预测结果为 [1, 1]
覆盖测试点 执行更新; 维护两比特饱和计数器并根据饱和计数器输出预测结果; 提供基础条件分支结果	

5.2.2 br_slot 饱和计数器的更新（2-3-2）

表 5.2.2: uFTB 测试用例组 2 饱和计数器测试 2

测试步骤	预期结果
步骤一：uFTB 预测器初始化，将 reset 置 1 步骤二：送入更新请求以缓存 1 条 FTB 项，io_update_valid 有效，输入 [1, 1]，空转两个时钟周期等待 FTB 项更新完毕 步骤三：输出该 FTB 项的预测结果	步骤三：预测结果为 [1, 1]

步骤四：io_update_valid 有效，输入 [0, 1]，空转两个时钟周期等待 FTB 项更新完毕	
步骤五：输出该 FTB 项的预测结果	步骤五：预测结果为 [1, 1]
覆盖测试点 执行更新; 维护两比特饱和计数器并根据饱和计数器输出预测结果; 提供基础条件分支结果	

5.2.3 br_slot 饱和计数器的更新（2-1-2）

表 5.2.3: uFTB 测试用例组 2 饱和计数器测试 3

测试步骤	预期结果
步骤一：uFTB 预测器初始化，将 reset 置 1	
步骤二：送入更新请求以缓存 1 条 FTB 项，io_update_valid 有效，输入 [0, 1]，空转两个时钟周期等待 FTB 项更新完毕	
步骤三：输出该 FTB 项的预测结果	步骤三：预测结果为 [0, 1]
步骤四：io_update_valid 有效，输入 [1, 1]，空转两个时钟周期等待 FTB 项更新完毕	
步骤五：输出该 FTB 项的预测结果	步骤七：预测结果为 [1, 1]
覆盖测试点 执行更新; 维护两比特饱和计数器并根据饱和计数器输出预测结果; 提供基础条件分支结果	

5.2.4 br_slot 饱和计数器的更新（2-1-0-1）

表 5.2.4: uFTB 测试用例组 2 饱和计数器测试 4

测试步骤	预期结果
步骤一：uFTB 预测器初始化，将 reset 置 1	
步骤二：送入更新请求以缓存 1 条 FTB 项，io_update_valid 有效，输入 [0, 1]，空转两个时钟周期等待 FTB 项更新完毕	
步骤三：输出该 FTB 项的预测结果	步骤三：预测结果为 [0, 1]
步骤四：io_update_valid 有效，输入 [0, 1]，空转两个时钟周期等待 FTB 项更新完毕	
步骤五：输出该 FTB 项的预测结果	步骤五：预测结果为 [0, 1]
步骤六：io_update_valid 有效，输入 [1, 1]，空转两个时钟周期等待 FTB 项更新完毕	
步骤七：输出该 FTB 项的预测结果	步骤七：预测结果为 [0, 1]
覆盖测试点 执行更新; 维护两比特饱和计数器并根据饱和计数器输出预测结果; 提供基础条件分支结果	

5.2.5 br_slot 饱和计数器的更新（2-1-0-0）

表 5.2.5: uFTB 测试用例组 2 饱和计数器测试 5

测试步骤	预期结果
步骤一：uFTB 预测器初始化，将 reset 置 1	
步骤二：送入更新请求以缓存 1 条 FTB 项	
步骤三：输出该 FTB 项的预测结果，io_update_valid 有效，输入 [0, 1]，空转两个时钟周期等待 FTB 项更新完毕	步骤三：预测结果为 [0, 1]
步骤四：io_update_valid 有效，输入 [0, 1]	
步骤五：输出该 FTB 项的预测结果	步骤五：预测结果为 [0, 1]
步骤六：io_update_valid 有效，输入 [0, 1]	
步骤七：输出该 FTB 项的预测结果	步骤五：预测结果为 [0, 1]
覆盖测试点 执行更新; 维护两比特饱和计数器并根据饱和计数器输出预测结果; 提供基础条件分支结果	

5.2.6 tail_slot 饱和计数器的更新（2-3-3）

表 5.2.6: uFTB 测试用例组 2 饱和计数器测试 6

测试步骤	预期结果
步骤一：uFTB 预测器初始化，将 reset 置 1	
步骤二：送入更新请求以缓存 1 条 FTB 项，tail_slot 有效且为 sharing 状态，io_update_valid 有效，输入 [0, 1]，空转两个时钟周期等待 FTB 项更新完毕	
步骤三：输出该 FTB 项的预测结果	步骤三：预测结果为 [0, 1]
步骤四：io_update_valid 有效，输入 [0, 1]	
步骤五：输出该 FTB 项的预测结果	步骤五：预测结果为 [0, 1]
覆盖测试点 执行更新; 维护两比特饱和计数器并根据饱和计数器输出预测结果; 提供基础条件分支结果	

5.2.7 tail_slot 饱和计数器的更新（2-3-2）

表 5.2.7: uFTB 测试用例组 2 饱和计数器测试 7

测试步骤	预期结果
步骤一：uFTB 预测器初始化，将 reset 置 1	
步骤二：送入更新请求以缓存 1 条 FTB 项，tail_slot 有效且为 sharing 状态，io_update_valid 有效，输入 [0, 1]，空转两个时钟周期等待 FTB 项更新完毕	
步骤三：输出该 FTB 项的预测结果	步骤三：预测结果为 [0, 1]
步骤四：io_update_valid 有效，输入 [0, 0]	
步骤五：输出该 FTB 项的预测结果	步骤五：预测结果为 [0, 1]
覆盖测试点 执行更新; 维护两比特饱和计数器并根据饱和计数器输出预测结果; 提供基础条件分支结果	

5.2.8 tail_slot 饱和计数器的更新（2-1-2）

表 5.2.8: uFTB 测试用例组 2 饱和计数器测试 8

测试步骤	预期结果
步骤一：uFTB 预测器初始化，将 reset 置 1	
步骤二：送入更新请求以缓存 1 条 FTB 项，tail_slot 有效且为 sharing 状态，io_update_valid 有效，输入 [0, 0]，空转两个时钟周期等待 FTB 项更新完毕	
步骤三：输出该 FTB 项的预测结果	步骤三：预测结果为 [0, 0]
步骤四：io_update_valid 有效，输入 [0, 1]	
步骤五：输出该 FTB 项的预测结果	步骤五：预测结果为 [0, 1]
覆盖测试点 执行更新; 维护两比特饱和计数器并根据饱和计数器输出预测结果; 提供基础条件分支结果	

5.2.9 tail_slot 饱和计数器的更新（2-3-0-1）

表 5.2.9: uFTB 测试用例组 2 饱和计数器测试 9

测试步骤	预期结果
步骤一：uFTB 预测器初始化，将 reset 置 1	
步骤二：送入更新请求以缓存 1 条 FTB 项，tail_slot 有效且为 sharing 状态，io_update_valid 有效，输入 [0, 0]，空转两个时钟周期等待 FTB 项更新完毕	
步骤三：输出该 FTB 项的预测结果	步骤三：预测结果为 [0, 0]
步骤四：io_update_valid 有效，输入 [0, 0]	
步骤五：输出该 FTB 项的预测结果	步骤五：预测结果为 [0, 0]
步骤六：io_update_valid 有效，输入 [0, 1]	
步骤七：输出该 FTB 项的预测结果	步骤七：预测结果为 [0, 0]
覆盖测试点 执行更新; 维护两比特饱和计数器并根据饱和计数器输出预测结果; 提供基础条件分支结果	

5.2.10 tail_slot 饱和计数器的更新（2-3-0-0）

表 5.2.10: uFTB 测试用例组 2 饱和计数器测试 10

测试步骤	预期结果
步骤一：uFTB 预测器初始化，将 reset 置 1	
步骤二：送入更新请求以缓存 1 条 FTB 项，tail_slot 有效且为 sharing 状态，io_update_valid 有效，输入 [0, 0]，空转两个时钟周期等待 FTB 项更新完毕	
步骤三：输出该 FTB 项的预测结果	步骤三：预测结果为 [0, 0]
步骤四：io_update_valid 有效，输入 [0, 0]	

步骤五：输出该 FTB 项的预测结果 步骤六：io_update_valid 有效，输入 [0, 0] 步骤七：输出该 FTB 项的预测结果	步骤五：预测结果为 [0, 0] 步骤七：预测结果为 [0, 0]
覆盖测试点 执行更新; 维护两比特饱和计数器并根据饱和计数器输出预测结果; 提供基础条件分支结果	

5.2.11 饱和计数器更新统一测试

表 5.2.11: uFTB 测试用例组 2 饱和计数器测试 11

测试步骤	预期结果
步骤一：uFTB 预测器初始化，将 reset 置 1 步骤二：送入更新请求以缓存 32 条 FTB 项，tail_slot 有效且为 sharing 状态，io_update_valid 有效，输入 [1, 1] 步骤三：输出该 FTB 项的预测结果 步骤四：io_update_valid 有效，输入 [1, 1] 步骤五：输出该 FTB 项的预测结果 步骤六：io_update_valid 有效，输入 [0, 0] 步骤七：输出该 FTB 项的预测结果 步骤八：io_update_valid 有效，输入 [0, 0] 步骤九：输出该 FTB 项的预测结果 步骤十：io_update_valid 有效，输入 [0, 0] 步骤十一：输出该 FTB 项的预测结果 步骤十二：io_update_valid 有效，输入 [0, 0] 步骤十三：输出该 FTB 项的预测结果 步骤十四：io_update_valid 有效，输入 [1, 1] 步骤十五：输出该 FTB 项的预测结果 步骤十六：io_update_valid 有效，输入 [1, 1] 步骤十七：输出该 FTB 项的预测结果	 步骤三：预测结果为 [0, 0] 步骤五：预测结果为 [1, 1] 步骤七：预测结果为 [1, 0] 步骤九：预测结果为 [0, 0] 步骤十一：预测结果为 [0, 0] 步骤十三：预测结果为 [0, 0] 步骤十五：预测结果为 [0, 0] 步骤十七：预测结果为 [1, 0]
覆盖测试点 执行更新; 维护两比特饱和计数器并根据饱和计数器输出预测结果; 提供基础条件分支结果	

5.2.12 io_update_bits_ftb_entry_always_taken_0

表 5.2.12: uFTB 测试用例组 2 饱和计数器测试 12

测试步骤	预期结果
步骤一：uFTB 预测器初始化，将 reset 置 1 步骤二：送入更新请求以缓存 1 条 FTB 项，tail_slot 有效且为 sharing 状态，io_update_valid 有效，输入 [0, 0]，且 io_update_bits_ftb_entry_always_taken_0 有效，空转两个时钟周期等待 FTB 项更新完毕	

步骤三：输出该 FTB 项的预测结果	步骤三：预测结果为 [1, 0]
步骤四：io_update_valid 有效，输入 [0, 0]	
步骤五：输出该 FTB 项的预测结果	步骤五：预测结果为 [1, 0]
步骤六：io_update_valid 有效，输入 [0, 0]	
步骤七：输出该 FTB 项的预测结果	步骤七：预测结果为 [1, 0]
覆盖测试点 执行更新; 维护两比特饱和计数器并根据饱和计数器输出预测结果; 提供基础条件分支结果	

5.2.13 io_update_bits_ftb_entry_always_taken_1

表 5.2.13: uFTB 测试用例组 2 饱和计数器测试 13

测试步骤	预期结果
步骤一：uFTB 预测器初始化，将 reset 置 1	
步骤二：送入更新请求以缓存 1 条 FTB 项，tail_slot 有效且为 sharing 状态，io_update_valid 有效，输入 [0, 0]，且 io_update_bits_ftb_entry_always_taken_1 有效	
步骤三：输出该 FTB 项的预测结果	步骤三：预测结果为 [0, 1]
步骤四：io_update_valid 有效，输入 [0, 0]	
步骤五：输出该 FTB 项的预测结果	步骤五：预测结果为 [0, 1]
步骤六：io_update_valid 有效，输入 [0, 0]	
步骤七：输出该 FTB 项的预测结果	步骤七：预测结果为 [0, 1]
覆盖测试点 执行更新; 维护两比特饱和计数器并根据饱和计数器输出预测结果; 提供基础条件分支结果	

5.2.14 tailSlot_sharing

表 5.2.14: uFTB 测试用例组 2 饱和计数器测试 14

测试步骤	预期结果
步骤一：uFTB 预测器初始化，将 reset 置 1	
步骤二：送入更新请求以缓存 1 条 FTB 项，tail_slot 有效且不处于 sharing 状态，io_update_valid 有效，输入 [0, 0]，且 io_update_bits_ftb_entry_always_taken_1 有效	
步骤三：输出该 FTB 项的预测结果	步骤三：预测结果为 [0, 1]
步骤四：io_update_valid 有效，输入 [0, 0]	
步骤五：输出该 FTB 项的预测结果	步骤五：预测结果为 [0, 1]
步骤六：io_update_valid 有效，输入 [0, 0]	
步骤七：输出该 FTB 项的预测结果	步骤七：预测结果为 [0, 1]
覆盖测试点 执行更新; 维护两比特饱和计数器并根据饱和计数器输出预测结果; 提供基础条件分支结果	

5.3 缓存更新测试

5.3.1 io_update_valid

表 5.3.1: uFTB 测试用例组 3 缓存更新测试 1

测试步骤	预期结果
步骤一：uFTB 预测器初始化，将 reset 置 1 步骤二：送入更新请求以缓存 1 条 FTB 项 步骤三：送入更新请求以更新该 FTB 项，io_update_valid 置为无效，空转两个时钟周期等待更新完成 步骤四：输出该 FTB 项的预测结果 步骤五：送入更新请求以更新该 FTB 项，io_update_valid 置为有效，空转两个时钟周期等待更新完成 步骤六：输出该 FTB 项的预测结果	步骤四: 输出的 FTB 项与送入的 FTB 项更新后的内容一致 步骤六: 输出的 FTB 项未被更新
覆盖测试点 执行更新; 提供基础条件分支结果	

5.3.2 FTB 项更新

表 5.3.2: uFTB 测试用例组 3 缓存更新测试 2

测试步骤	预期结果
步骤一：uFTB 预测器初始化，将 reset 置 1 步骤二：送入更新请求以缓存 1 条 FTB 项 步骤三：送入更新请求以更新该 FTB 项，除了 pc 外的所有值均置随机数 步骤四：输出该 FTB 项的预测结果	步骤四: 输出的 FTB 项与送入的 FTB 项更新后的内容一致
覆盖测试点 执行更新; 提供基础条件分支结果	

5.3.3 缓存更新命中

表 5.3.3: uFTB 测试用例组 3 缓存更新测试 3

测试步骤	预期结果
步骤一：uFTB 预测器初始化，将 reset 置 1 步骤二：送入更新请求以缓存 32 条 FTB 项 步骤三：每个周期 s0_fire 置为有效，依次送入步骤二中所有 FTB 项，观察输出结果	步骤三: 输出的 FTB 项与送入的 FTB 项更新后的内容一致
覆盖测试点 执行更新; 提供基础条件分支结果	

5.3.4 缓存更新未命中，LRU 更新测试

表 5.3.4: uFTB 测试用例组 3 缓存更新测试 4

测试步骤	预期结果
步骤一：uFTB 预测器初始化，将 reset 置 1	
步骤二：送入更新请求以缓存 32 条 FTB 项	
步骤三: 每个周期 s0_fire 置为有效, 依次更新步骤二前 16 条 FTB 项	步骤三: 输出的 FTB 项与送入的 FTB 项更新后的内容一致
步骤四：送入更新请求以缓存新的 16 条 FTB 项	
步骤五：每个周期 s0_fire 置为有效，依次送入步骤二的后 16 条 FTB 项，观察输出结果	步骤五：输出未命中
步骤六: 每个周期 s0_fire 置为有效, 依次送入步骤四的 16 条 FTB 项，观察输出结果	步骤六: 输出的 FTB 项与步骤四中送入的 FTB 项内容一致
覆盖测试点 执行更新; 提供基础条件分支结果	

5.3.5 缓存更新未命中，LRU 不更新测试

表 5.3.5: uFTB 测试用例组 3 缓存更新测试 5

测试步骤	预期结果
步骤一：uFTB 预测器初始化，将 reset 置 1	
步骤二：送入更新请求以缓存 32 条 FTB 项	
步骤三: 每个周期 s0_fire 置为有效, 依次送入步骤二前 16 条 FTB 项，观察输出结果	步骤三: 输出的 FTB 项与送入的 FTB 项更新后的内容一致
步骤四：送入更新请求以缓存新的 16 条 FTB 项	
步骤五：每个周期 s0_fire 置为有效，依次送入步骤二的后 16 条 FTB 项，观察输出结果	步骤五：输出未命中
步骤六: 每个周期 s0_fire 置为有效, 依次送入步骤四的 16 条 FTB 项，观察输出结果	步骤六: 输出的 FTB 项与步骤四中送入的 FTB 项内容一致
覆盖测试点 执行更新; 提供基础条件分支结果	

6 uFTB 结果分析

这里是对整个测试结果的简单分析和介绍。

6.1 测试用例分析

测试用例全部通过

6.2 行覆盖率分析

表 6.2.1: uFTB 行覆盖率

名称	命中行数	总行数	行覆盖率
FauFTBWay.sv	71	72	98.6%
DelayN_2.sv	15	15	100%
FauFTB.v	1056	1058	99.8%

FauFTBWay.sv 文件的行覆盖率为 **98.6%**。

未覆盖到的代码仅有一行，为 initial 块中的判断语句

由于 initial 块中使 reset 生效需要设置 dut.reset.AsImmWrite 将引脚置成立即写入，故在一次测试中不能覆盖两种情况，该行语句无法被覆盖。实际上，为了测试该行语句的功能，我通过进行两次测试，引脚写入两种值，验证结果均符合预期，可以视为该行语句已经被覆盖。

FauFTB.v 文件的行覆盖率为 **99.8%**。

未覆盖到的代码仅有两行，其中一行未覆盖的原因同上，另一行未覆盖的原因为判断语句的条件不可达，所以无法覆盖。

6.3 功能覆盖率分析

表 6.3.1: uFTB 功能覆盖率

命中功能点	总功能点	功能覆盖率
17	17	100%

所有功能点均被覆盖，功能覆盖率达到 100%。

7 测试结论

针对 FTB 项缓存、更新、预测的用例测试显示，uFTB 功能符合预期。综上所述，在验证内容范围内，uFTB 功能正确。