

哈爾濱ノ業大学 (深圳) HARBIN INSTITUTE OF TECHNOLOGY

实验作业

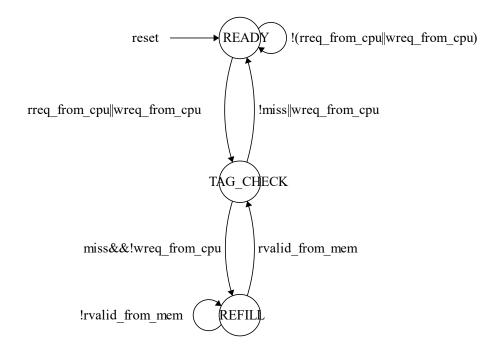
开课学期:	2022 春季
课程名称:	计算机组成原理(实验)
实验名称:	直接映射 Cache 设计
实验性质:	综合设计型
实验学时:	4 地点: T2615
学生班级:	 计算机 6 班
学生学号:	200110619
学生姓名:	 梁鑫嵘
作业成绩:	

实验与创新实践教育中心制 2022 年 5 月

一、 Cache 模块设计

(画出读、写的状态转移图,并描述状态之间的转移关系和转移条件、以及每个状态需要完成什么操作。)

有限状态自动机:

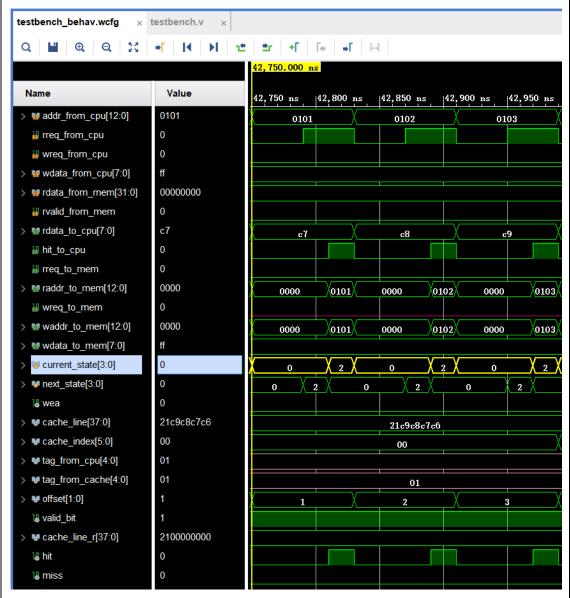


- 1. READY: 待机准备状态
 - a) $r/waddr_to_mem \le 0$
 - b) $r/wreq_to_mem \le 0$
- 2. TAG_CHECK: 当前周期检查是否命中 Cache
 - a) waddr_to_mem <= addr_from_cpu; 写入写地址
 - b) wreq_to_mem <= 0; 检查的时候不能确定是否写缺失,所以不会写
 - c) raddr_to_mem <= addr_from_cpu; 写入读地址
 - d) rreq_to_mem <= miss && rreq_from_cpu && !rvalid_from_mem; 产生缺失
- 3. REFILL: 读内存到 Cache 或者写内容到内存
 - a) raddr_to_mem <= addr_from_cpu; 写入读地址
 - b) rreq_to_mem <= rreq_from_cpu && !rvalid_from_mem; 读完适时拉低 rreq 信号
 - c) waddr_to_mem <= addr_from_cpu; 写入写地址

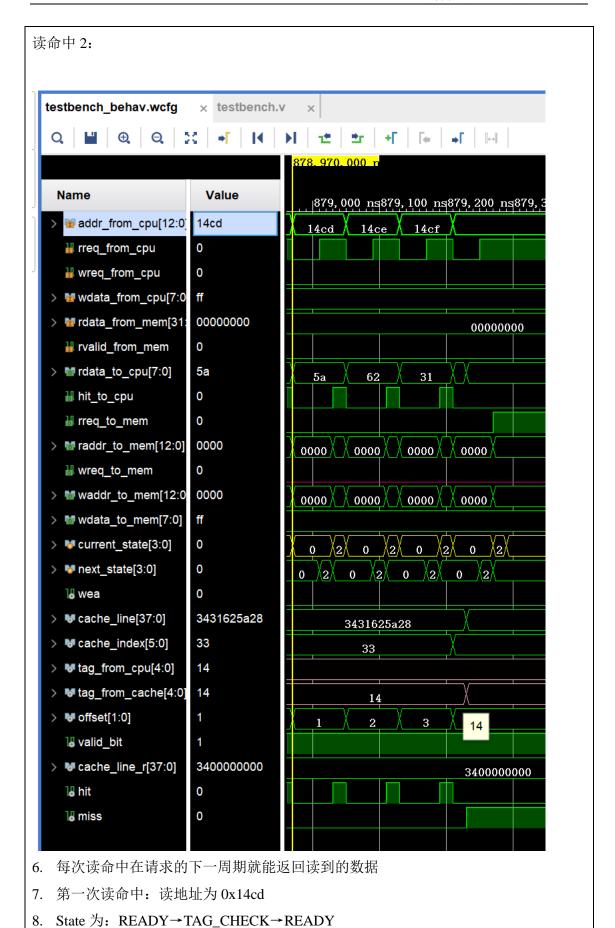
d)	wdata_to_mem <= wdata_from_cpu; 是否写入
e)	wreq_to_mem <= !miss && wreq_from_cpu; 填入 Cache 后拉低 wreq
	\m\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\\
	调试报告

(仿真截图及时序分析,要求包含读命中、读缺失、写命中及写缺失共四种情况的分析, 且每种情况需列举**2个**测试用例进行分析。)

读命中1:



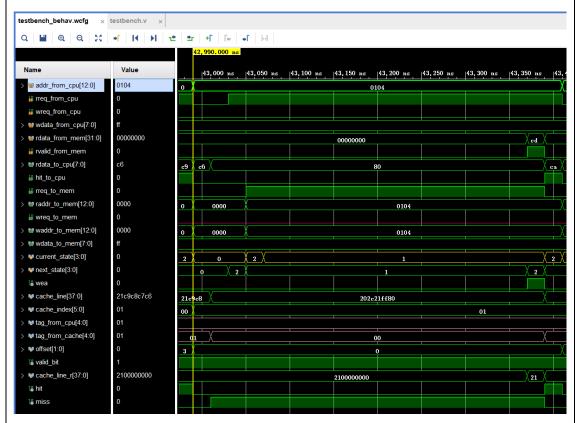
- 1. 每次读命中在请求的下一周期就能返回读到的数据
- 2. 第一次读命中: 读地址为 0x0101
- 3. State 为: READY→TAG_CHECK→READY
- 4. 这三次读取命中同一个 Cache Line, Tag 为 0x01
- 5. Offset 依地址变化为 0x1,0x2,0x3, 故依次返回 Cache Line 的第 2,3,4 字节



9. 这三次读取命中同一个 Cache Line, Tag 为 0x01

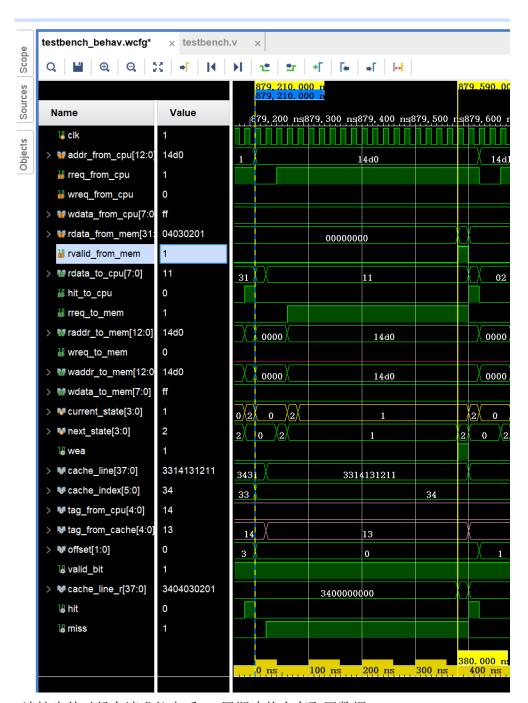
Offset 依地址变化为 0x1,0x2,0x3, 故依次返回 Cache Line 的第 2,3,4 字节

读缺失1:



- 1. 读缺失的时候在请求拉高后 18 周期才从内存取回数据
- 2. 取到数据的那一周期写入 Cache, 取到数据为 0xcdcccbca, 写入 Cache Line 为 0x21cdcccbca
- 3. 取到数据之后的下一周期将 Cache 内容返回 CPU
- 4. State 变化为: READY→TAG_CHECK→REFILL→TAG_CHECK→READY

读缺失 2:



- 1. 读缺失的时候在请求拉高后 18 周期才从内存取回数据
- 2. 取到数据的那一周期写入 Cache, 取到数据为 0x04030201, 写入 Cache Line 为 0x3404030201
- 3. 取到数据之后的下一周期将 Cache 内容返回 CPU
- 4. State 变化为: READY→TAG_CHECK→REFILL→TAG_CHECK→READY
- 5. 读缺失总共耗时 400ns, 即 20 周期

写命中1: testbench_behav.wcfg* × testbench.v × Q 💾 🙉 Q 💥 📲 🔣 >1 🛨 🖭 +1 📭 🔐 352, 810, 000 r l, 352, 770, 000 r l, 352, 770, 000 r Value Name **₩** clk > 🛂 addr_from_cpu[12:0] 1f04 1f04 # rreq_from_cpu wreq_from_cpu > Www.data_from_cpu[7:0 ff > 💆 rdata_from_mem[31: 00000000 II rvalid_from_mem > Mrdata_to_cpu[7:0] 80 1a 0 ₩ hit_to_cpu 0 > Mraddr_to_mem[12:0] 0000 0000 1f04 000 ₩ wreq_to_mem > Mwaddr_to_mem[12:0 0000 1f04 0000 000 > **!!** wdata_to_mem[7:0] > W current_state[3:0] > ***** next_state[3:0] 0 **™** wea 3f2c21ff80 > **W** cache_line[37:0] 3f2c21ff80 3f0e0c0b1a > W cache_index[5:0] 00 > **W** tag_from_cpu[4:0] 1f 1**f** > Wtag_from_cache[4:0] 1f > **W** offset[1:0] valid_bit

1. 欲写入地址为 0x1f04, 数据为 0xff

3f00000000

0

2. 写访存情况下,CPU 写信号拉高后的下一周期就能将是否写命中返回给CPU

3f00000000

0 ns 10 ns 20 ns 30 ns

3f2c21ffff

3f00000000

40.000 ns 40 ns

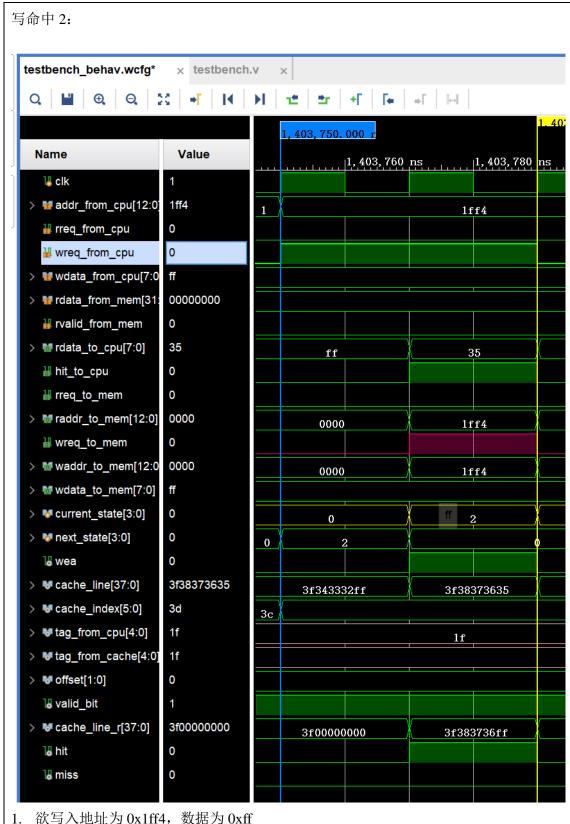
- 3. 返回是否写命中的本周期拉高内存写入信号并写入内存指定地址
- 4. 返回是否写命中的本周期拉高 Cache 写信号 wea 并按照 offset 更新 Cache Line
- 5. State: READY→TAG CHECK→READY
- 6. 总耗时: 2周期

> **W** cache_line_r[37:0]

Tol Console Messages Log

l⊌ hit

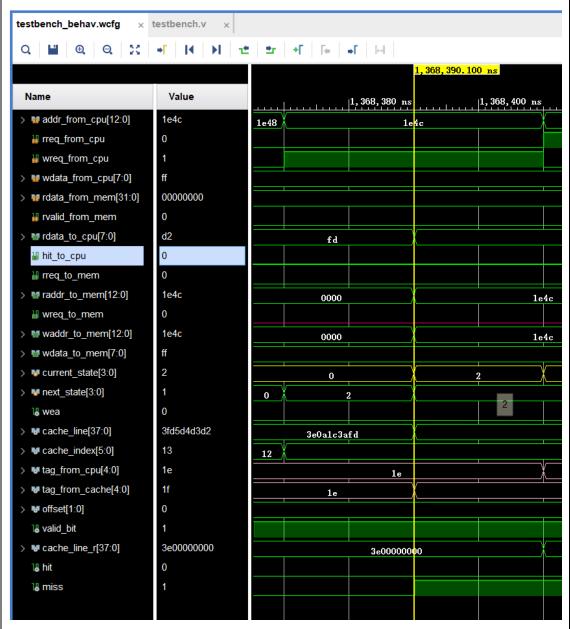
le miss



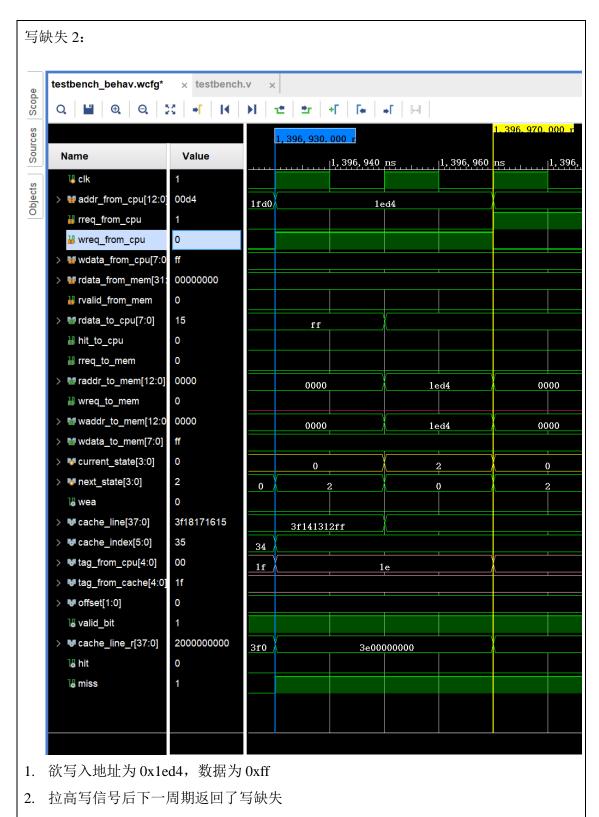
- 2. 写访存情况下,CPU 写信号拉高后的下一周期就能将是否写命中返回给CPU
- 3. 返回是否写命中的本周期拉高内存写入信号并写入内存指定地址
- 4. 返回是否写命中的本周期拉高 Cache 写信号 wea 并按照 offset 更新 Cache Line

- 5. State: READY→TAG_CHECK→READY
- 6. 总耗时: 2周期

写缺失1:



- 1. 欲写入地址为 0x1e4c,数据为 0xff
- 2. 拉高写信号后下一周期返回了写缺失
- 3. 写缺失则什么也不做,不更新 Cache 也不更新内存内容
- 4. State: READY→TAG CHECK→READY



- 3. 写缺失则什么也不做,不更新 Cache 也不更新内存内容
- 4. State: READY→TAG CHECK→READY