

（深圳）

实验作业

开课学期： 2021春季

课程名称：计算机组成原理（实验）

实验名称： 直接相连Cache设计

实验性质： 综合设计型

实验学时： 4 地点： T2

学生班级： 7

学生学号： 190110716

学生姓名： 朱海峰

作业成绩：

实验与创新实践教育中心制

2021年4月

|  |
| --- |
| 1. **系统功能详细设计** |
| **（**系统主要功能及各模块之间的相互关系，用硬件框图描述）  本次实验中，由driver控制着访问主存的地址生成，并在trace和cache中同时访问相同的地址。在cache未命中时，cache会读取主存mem\_wrap，重新装载相应的数据，然后命中返回读取的数据。整体硬件框图如图一：    Figure  // 地址 raddr\_from\_cpu  // 标记位 cache索引  字节地址    总共  //  4       7           2       13  // 12:9     8:2         1:0  // cache\_reg位分配  // 有效位  标记       数据      总共  //  1       4           8\*4     37  // 36       35:32       31:0 |
| 1. Cache模块设计（包括状态转移图） |
| 在cache内部，状态转移图如图2，其中输出为1表示命中，同时传出对应的数据，输出为0表示未命中，输出无效：    Figure  当cpu发出req信号：   1. 通过读取地址的cache索引段找到对应的cache行； 2. 通过读取地址的字节偏移段找到对应的字节数据； 3. 通过读取地址的标记位段和对应cache行的标记位段是否匹配以及cache行中的有效位判断是否命中。   逻辑图如下：    Figure |
| 1. **调试报告** |
| （自己写的testbench核心代码，仿真截图及时序分析。根据测试的完备性评分）  本次实验没有另外特地去写testbench，而是直接使用实验包附带的all\_sim\_v.v源文件进行仿真调试。  //状态分配：  //00  ready  //01  TAG\_CHECK  //10  REFILL  以下是时序分析。  测试开始时，具体波形如图4：   1. rreq\_from\_cpu拉高，访问地址0，状态机状态由ready变为tag\_check，进行tag\_check； 2. 并没有命中，于是进入refill状态开始从主存装入，拉高rrep\_to\_mem； 3. 若干个时钟周期后，rvalid\_from\_cpu有效，装入4个字节读到的数据； 4. 状态再次进入tag\_check，并且命中，拉高hit\_to\_cpu，传输数据，波形图中可见得到的数据与trace\_rdata相匹配。 5. 对于接下来的地址1、2、3，cache均能直接命中返回，因为在第2步已经装入     Figure  测试即将结束时，具体波形如图5，控制台输出如图6：  状态、重装等相关信号与上面提到的相似。此外能清晰地看到cache的寄存器对应的变化，并且在测试完成后，test\_success有效，测试成功。  图6中，等待返回数据的时候，显然cache在从主存中重装数据，于是，之后连续的地址也能直接在cache中命中读出。    Figure    Figure |