



# 计算机组成原理实验指导书

马世禹

---

# 目录

计算机组成原理实验指导书 .....	1
实验三 直接相联Cache设计 .....	3
1、实验目的 .....	3
2、实验内容 .....	3
3、实验原理 .....	3
4、实验准备 .....	5
5、实验步骤 .....	8
附录1 IP核的使用步骤 .....	8
附录2 Driver模块设计说明 .....	10

# 实验三 直接相联Cache设计

## 1、实验目的

- 掌握直接相联Cache的基本结构及其设计方法。
- 实现直接相联地址映射的机制
- 实现读命中和读缺失两种情况的处理。

## 2、实验内容

为主存容量8KB的存储系统，设计直接相联Cache，参数如下：

- Cache的总容量为512B
- 每块4个字节
- 包括128个Cache块

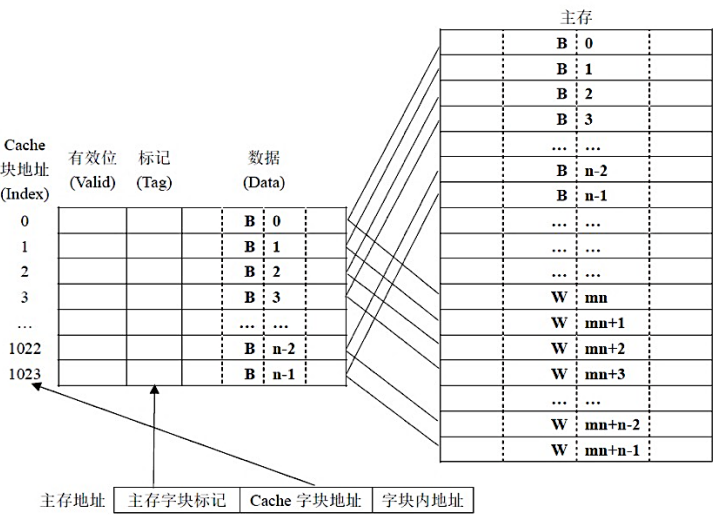
实现在CPU发出读操作请求后的取数据操作，包括读命中和读缺失两种情况。

## 3、实验原理

### 3.1 Cache的结构

Cache由存储体和控制机构两大主要部分构成。

#### Cache的存储体

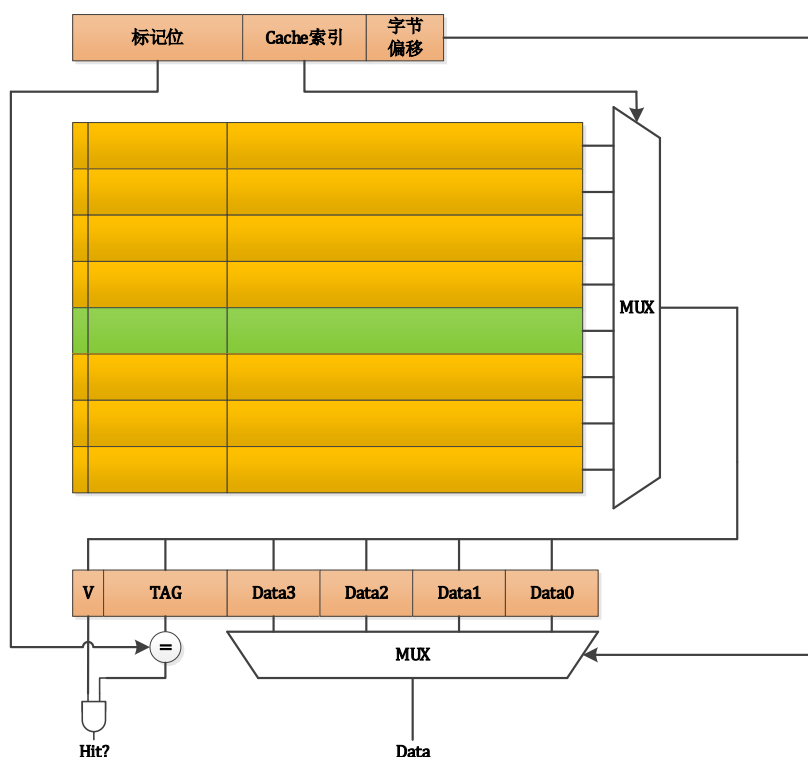


- 有效位 (Valid)：标识该行内存储的数据是否有效，未装入任何数据之前无效。
- 标记 (Tag)：用以唯一地标识某个地址，判断是否访问命中。
- 数据域 (Data)：连续存储的k个相邻的在主存中连续的数据。

## 映射机制

如上所示的地址映射机制，低2位为字块内偏移，代表一个Cache行存储有4个字节。中间10位为Cache字块地址，代表Cache有1024个Cache行。高20位为Tag，用以唯一地标识某个主存地址，与Cache中的标记位比对，判断是否访问命中。

## 地址分解



例如，我们系统的主存容量为256KB，则主存地址宽度为18位；我们设计Cache每个字块中存储4个字节，则每个块中需要用2位去寻址相应的字节；Cache容量为256行（ $256 \times 4B = 1024B$ ），且设计的方式是直接相联，则Cache字块地址为8位；剩余的 $18 - 8 - 2 = 8$ 位，则作为Tag使用。此时，我们即完成了地址的分解工作。

如上图所示，一个典型的直接映射Cache访问主要完成几项工作：

- Cache寻址：根据主存地址中的Cache索引，选出对应的Cache块（行）
- 标记比对：主存中的标记位和取出行中的标记位对比，加上Valid位，判断是否命中
- 选出数据：如果命中，根据地址的低位偏移，在读出的Cache Line的数据域中，选择相应的数据，回送给主设备，并采取相应的机制通知主设备，在本次实验中，我们要求拉高Cache模块的hit信号，同时将数据输出到data\_out上。
- 缺失处理：如果不命中的话，还需要进行缺失的处理，在下一部分将会提到。

## 3.2 控制机构

### 缺失处理机制

当Cache的访问发生缺失时，应要有相应的机制，确定缺失的位置，并向下一级存储器发出读字块的请求。

### 缺失动作

在缺失时，Cache向下一级存储器（本实验中为mem\_wrap）发起读请求，请求读取4个连续存储的字，待下级存储器响应后，将返回的数据采集、打上相应的标签，然后写入自身的存储体中。

### 实现原理

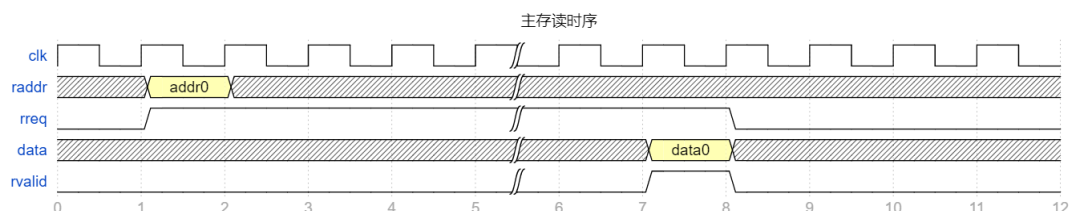
在时序电路中，如果要执行顺序发生的动作，并带有条件跳转，状态机是通用的做法。我们使用状态机生成控制信号，对数据通路进行操纵，从而达到处理各种情况的目的。

## 4、实验准备

### 4.1 熟悉存储系统模型

本次实验采用的存储系统大小为8K，地址宽度为13，字长为1字节。

存储器模块为mem\_wrap.v，为简化实验流程，我们已将Cache缺失时连续读取4个字并拼接的工作完成，具体时序图如下：



在主存空闲阶段，Cache将地址放在raddr口上，并将rreq信号拉高，代表一次的读请求。等待若干个周期，主存将返回4个字拼接而成的数据，并将rvalid信号拉高，代表数据已经准备好，此时可以取走数据。

### 4.2 项目框架概览

#### ■ 设计文件

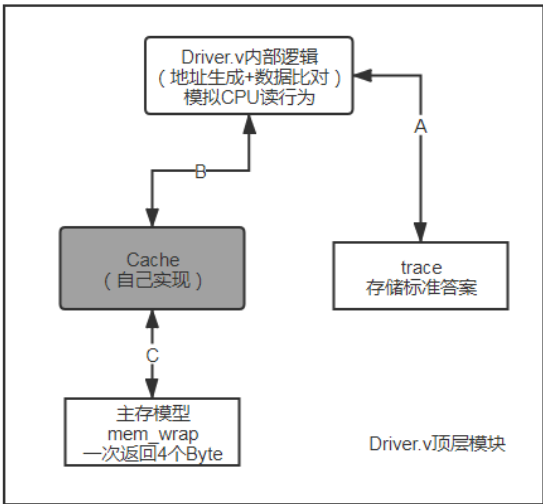
driver (driver.sv) ----- 顶层模块，模拟CPU的访存行为，并进行数据正确性检查，可以看作是CPU

trace (IP核) ----- 存储标准答案

cache (cache.v) ----- Cache模块（需要完成）

mem\_wrap(mem\_wrap.v) ----- 主存存储器模型

模块间的关系如下图所示：



■ 仿真文件

使用说明

完成设计后，可以运行all\_sim.v文件中的仿真，运行仿真时，下方的Tcl Console也会打印相应的调试信息，帮助你定位出错点。



如果测试全部通过，控制台会显示相应字样，同时仿真将会停止在对应位置。

```
34 | ○ if(g0.next_state == 8'b0000_0001) $display("该地址测试正确，将测试下一个地址\n -
35 | ○ end
36 | ○ always begin
37 | ○ #10 clk = ~clk;
38 | ○ end
39 | ○ always @(posedge clk) begin
40 | ○ if(test_success) begin
41 | ○
42 | ○
43 | ○
44 | ○ $display("=====测试全部通过=====");
45 | ○ $stop;
```

4.3 模块接口规范

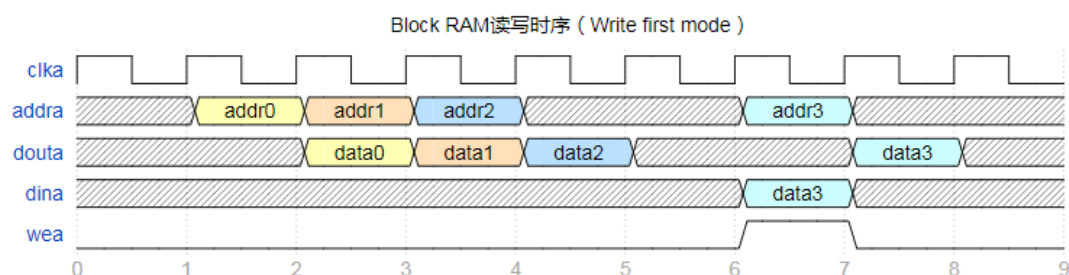
属性	名称	含义	位宽
----	----	----	----

输入	clk	时钟	1
输入	reset	复位（高电平有效）	1
输入	raddr_from_cpu	CPU的读地址	13
输入	rreq_from_cpu	CPU的读请求	1
输出	rdata_to_cpu	Cache读出的数据	8
输出	hit_to_cpu	命中标记	1
输入	rdata_from_mem	主存模块读取的连续4字节	32
输入	rvalid_from_mem	主存读取完毕标记	1
输出	rreq_to_mem	读主存请求	1
输出	raddr_to_mem	读主存首地址	13

#### 4.4 Cache模块时序规范

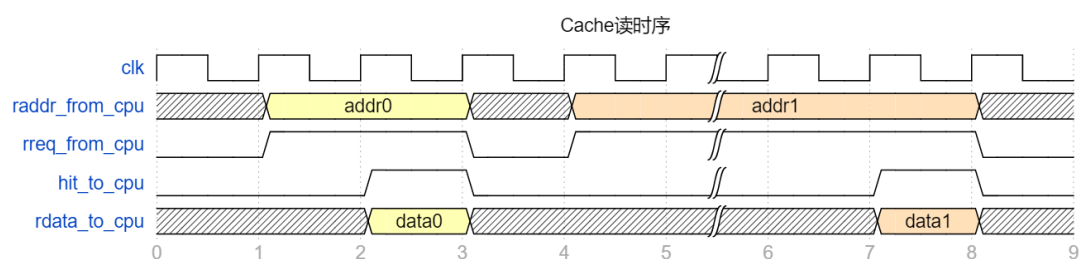
Cache存储体调用Block RAM的IP核实现，因此，我们需要了解Block RAM的读时序。IP核的使用见后文附录。

##### Block RAM的读写时序：



- 读时序：上一周期给出地址，下一周期输出数据，可连续读取。（周期1，2，3）
- 写时序：上一周期给数据、写地址，拉高wea信号，下一周期成功写入数据，刚刚写入的数据出现在douta口上。

##### Cache的读时序



下面以 Cyc# 代表周期号，详细叙述Cache和CPU之间的通信约定。

- CPU发来rreq信号，同时把地址放在raddr上(Cyc #1)，代表启动一次读取，在得到Cache的hit响应(Cyc #2)之前，CPU会保证：rreq\_from\_cpu信号不会撤下地址线raddr\_from\_cpu上的地址不会改变。
- 在得到Cache的hit响应之后(Cyc #2)，CPU会保证：在hit信号到来的下一个周

期(Cyc #3), rreq信号马上撤下Cache需要做到: hit响应信号和读出的数据只需持续一个周期(Cyc #2 - Cyc #3), 同时有效

- CPU未发rreq信号的时候, Cache需要做到: hit信号始终为0, 不得置高, 数据输出可以是任意值

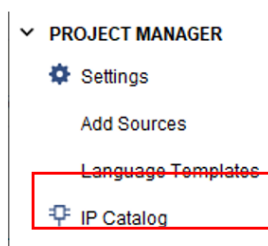
## 5、实验步骤

- 进行地址各个字段的分解
- 计算Cache各个参数, 选择大小合理的Block RAM, 创建IP核并实例化
- 编写命中判断的组合逻辑
- 设计控制机制的状态机 (不局限于Moore或Mealy, 也不局限于我们提供的几个状态, 可以自由发挥)
- 自己编写Testbench, 验证相关功能正确性。
- 使用提供的all\_sim.v测试模块进行测试、调试, 根据输出的调试信息, 定位错误点。

## 附录1 IP核的使用步骤

- 加入IP: Block memory generator

点击Flow Navigator中的IP Catalog, 打开窗口添加IP核。



在Memories & Storage Elements -> RAM & ROMs & BRAM文件夹, 双击Block Memory Generator。

Name	AXI4	Status	License	VLNV
Memories & Storage Elements				
ECC		Prod...	Included	xilin...
FIFOs				
Memory Interface Generators				
Memory Interface Generator (MIG 7...	AXI4	Prod...	Included	xilin...
RAMs & ROMs				
RAMs & ROMs & BRAM				
Block Memory Generator	AXI4	Prod...	Included	xilin...
Partial Reconfiguration				
SDAccel DSA Infrastructure				

- 点击Basic选项卡配置RAM为简单单口RAM

点击Port A Options设置Write Width为36, Read Width为36, Write Depth为64, Read Depth为64, 端口使能选择Always Enable, 去掉勾选Primitives Output Register。点击Other Options, 勾选Fill Remaining Memory Locations。点击OK等待片刻即可生成IP。



Re-customize IP

### Block Memory Generator (8.4)

Documentation IP Location Switch to Defaults

IP Symbol Power Estimation

☒ Show disabled ports

Component Name: blk\_mem\_gen\_0

Basic Port A Options Other Options Summary

Interface Type: Native ☐ Generate address interface with 32 bits

Memory Type: Single Port RAM ☐ Common Clock

ECC Options

ECC Type: No ECC

☐ Error Injection Pins: Single Bit Error Injection

Write Enable

☐ Byte Write Enable

Byte Size (bits): 9

Algorithm Options

Defines the algorithm used to concatenate the block RAM primitives. Refer datasheet for more information.

Algorithm: Minimum Area

Primitive: Bx2

OK Cancel

Re-customize IP

### Block Memory Generator (8.4)

Documentation IP Location Switch to Defaults

IP Symbol Power Estimation

☒ Show disabled ports

Component Name: blk\_mem\_gen\_0

Basic Port A Options Other Options Summary

Memory Size

Write Width: 36 Range: 1 to 4098 (bits)

Read Width: 36

Write Depth: 64 Range: 2 to 1048576

Read Depth: 64

Operating Mode: Write First Enable Port Type: Always Enabled

Port A Optional Output Registers

☐ Primitives Output Register ☐ Core Output Register

☐ SURECC Input Register ☐ REGCEA Pin

Port A Output Reset Options

☐ RSTA Pin (set/reset pin) Output Reset Value (Hex): 0

☐ Reset Memory Latch Reset Priority: CE (Latch or Register Enable)

READ Address Change A

☐ Read Address Change A

OK Cancel

Re-customize IP

### Block Memory Generator (8.4)

Documentation IP Location Switch to Defaults

IP Symbol Power Estimation

☒ Show disabled ports

Component Name: blk\_mem\_gen\_0

Basic Port A Options Other Options Summary

Pipeline Stages within Mux: 0 Mux Size: 1x1

Memory Initialization

☐ Load Init File

Coe File: no\_coe\_file\_loaded Browse Edit

☒ Fill Remaining Memory Locations

Remaining Memory Locations (Hex): 0

Structural/UnitSim Simulation Model Options

Defines the type of warnings and outputs are generated when a read-write or write-write collision occurs.

Collision Warnings: All

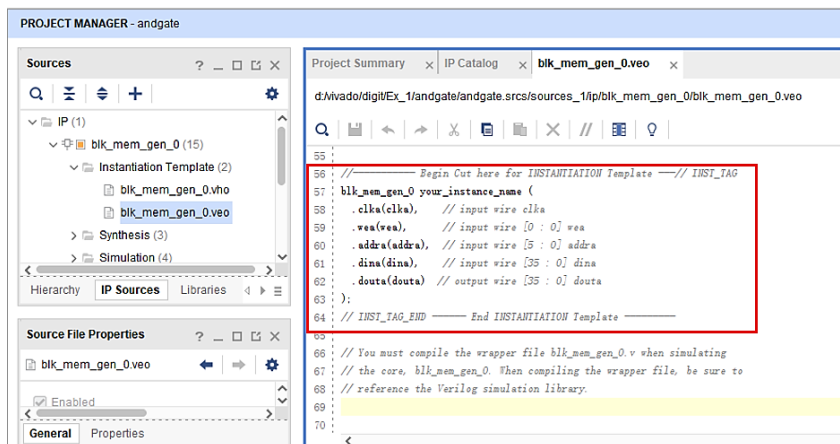
Behavioral Simulation Model Options

☐ Disable Collision Warnings ☐ Disable Out of Range Warnings

OK Cancel

## ■ 例化IP

生成好IP后，点击IP Sources展开IP，进入Instantiation Template下的blk\_mem\_gen\_0.veo文件下复制实例化模板到自己的设计文件下将此IP实例化。

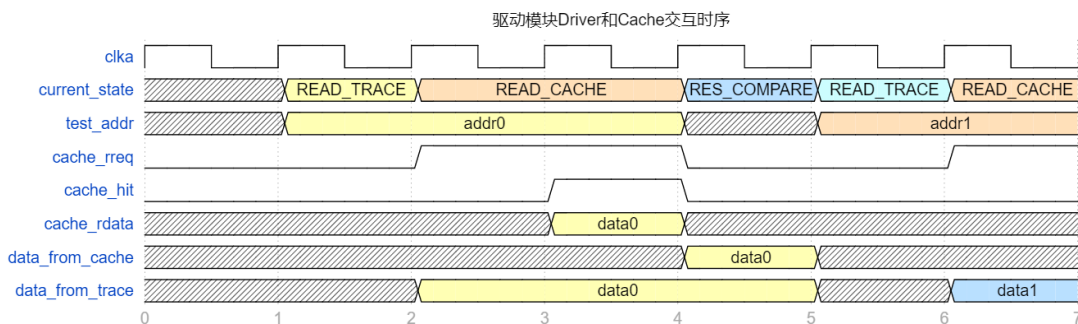


## 附录2 Driver模块设计说明

模拟CPU的读主存行为，与主存直接连接，与实现的Cache也直接连接。该模块生成一系列的测试地址，在对每一个测试地址进行测试时，先从主存中取出相应的数据(READ\_TRACE)，然后再对Cache提出访问请求(READ\_CACHE)，待Cache的数据可用后，将之前取出的数据和Cache给的数据进行比较(RES\_COMPARE)，如果正确，则进行下一个地址的测试。如果错误，则会停留在出错的位置(TEST\_FAIL)。

上层测试模块模拟CPU的取指令行为，其流程（状态转移）如下：

对一个地址进行测试时，首先进入READ\_TRACE状态（周期1），一个周期后，trace内读出的数据锁存到了data\_from\_trace寄存器中（周期2），并进入READ\_CACHE（周期2-3）状态，此时Cache\_rreq信号拉高，CPU向Cache发起读请求。



如图所示，可以看到，在hit信号到来的时候（周期3），上层的CPU取走了数据(周期4: data\_from\_Cache发生了改变)，并转移到RES\_COMPARE状态，对数据的正确性进行比较，如果正确，则进行下一轮的测试。