哈爾濱工業大學（深圳）
HARBIN INSTITUTE OF TECHNOLOGY

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：T2612

学生班级：1901105

学生学号：190110509

学生姓名：王铭

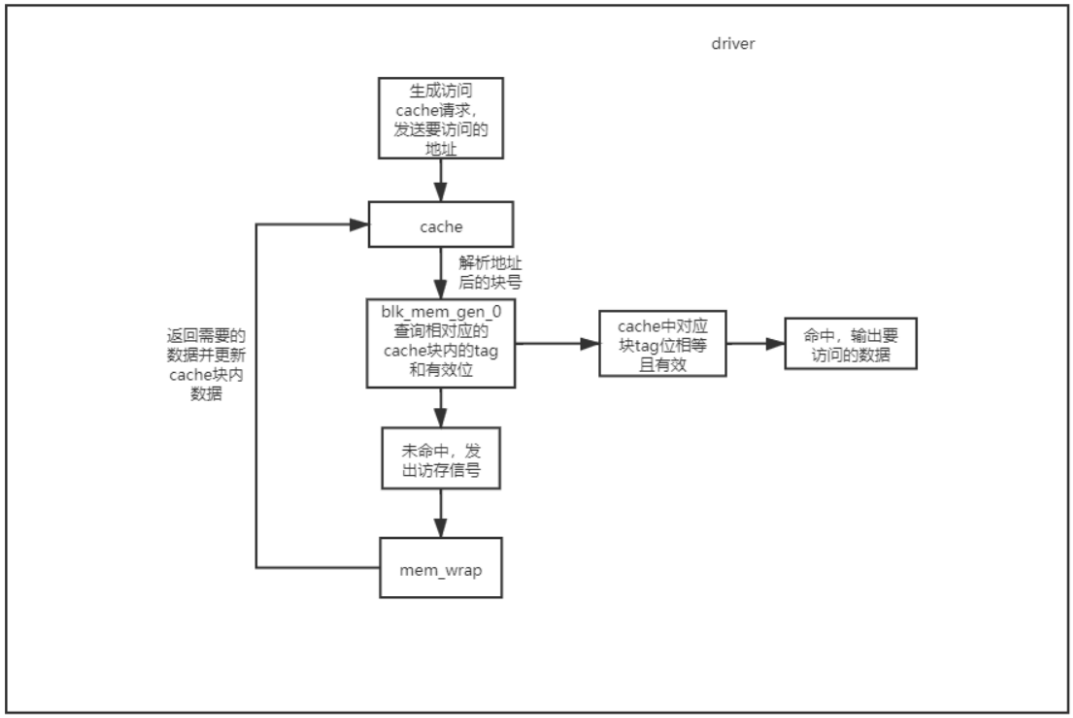
作业成绩：

实验与创新实践教育中心制

2021 年 4 月

一、 系统功能详细设计

系统硬件框图

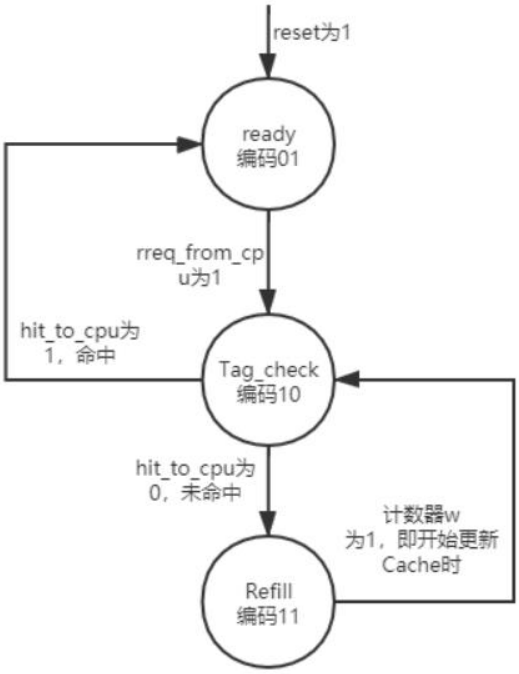


系统的主要功能

用 driver 模拟 cpu 提出的访问存储数据请求，首先，解析地址，判断 cache 块中是否有要访问的数据，若命中，则输出要返回的数据，否则访问主存并将数据更新至对应的 cache 块，然后输出要访问的数据。

二、 Cache 模块设计（包括状态转移图）

状态转移图



Cache 具体设计

①地址解析

由题目所述，Cache 一个块为 4B，共 128 个块，可判断块号为 7 位，块内地址为 2 位，剩下 4 位用于标记 Tag

②状态转移逻辑

设置两位变量的状态寄存器 current_state 和 next_state，根据上述状态转移图可以设置 next_state 的赋值逻辑。代码如下：

```
if(current_state == 2'b11 && w) next_state = 2'b10;

else if(current_state == 2'b10 && hit_to_cpu == 1'b1) next_state = 2'b01;

else if(current_state == 2'b10 && hit_to_cpu == 1'b0) next_state = 2'b11;

else if(current_state == 2'b01 && rreq_from_cpu == 1'b1) next_state = 2'b10;
```

其中，由于从数据有效到数据更新 Cache 需要一个时钟周期，从 Refill 态向 TC 态转变时需设置一个一位计数器 w，当 w 为 1 时，写入操作开始，次态置为 TC，下一个时钟周期，

现态变为 TC，Cache 更新完毕。

③判断命中逻辑

当 current_state 为 10，即 Tag_check 状态时，若 Cache 中相应块内的 Tag 位与 cpu 所给访存地址的高四位相等，且最高有效位为 1，则已经命中，置 hit_to_cpu 为 1，否则置 0

代码如下：

```
assign hit_to_cpu = ((current_state == 2'b10) && (raddr_from_cpu[12:9] == data_temp [35:32]) && (data_temp[36] == 1'b1)) == 1 ? 1 : 0;//data_temp 为从 Cache 所读数据
```

④输出逻辑

根据 cpu 所给地址的低两位选择对应 Cache 块中的正确字节输出。

代码如下：

```
assign rdata_to_cpu[7:0] = (raddr_from_cpu[1:0] == 2'b00 ? data_temp[7:0] : raddr_from_cpu[1:0] == 2'b01 ? data_temp[15:8] : raddr_from_cpu[1:0] == 2'b10 ? data_temp[23:16] : data_temp[31:24]);
```

⑤模拟 Cache 的 IP 核设计及更新 Cache 逻辑

由地址解析知，Cache 中数据位需 4B 即 32 位，Tag 位需 4 位，有效位需 1 位，共 37 位，共 128 个块，故 IP 核 blk_mem_gen_0 的宽度设为 37，深度设为 128.Cache 输出数据的最高位为有效位，接着 4 位为 Tag 位，最后 32 位为真正的数据。

当 next_state 为 11 即 Refill 态时，拉高访存信号，发出访存地址。

当主存返回读出信号有效时，更新 Cache 内对应块的数据。由 IP 核设计可定义输入数据 temp_reg 如下：

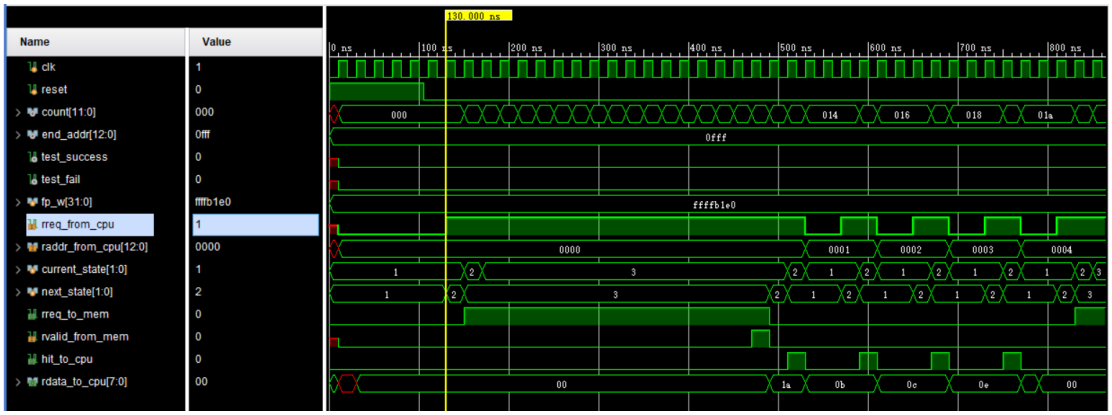
```
assign temp_reg = {1’b1,raddr_from_cpu[12:9],rdata_from_mem};
```

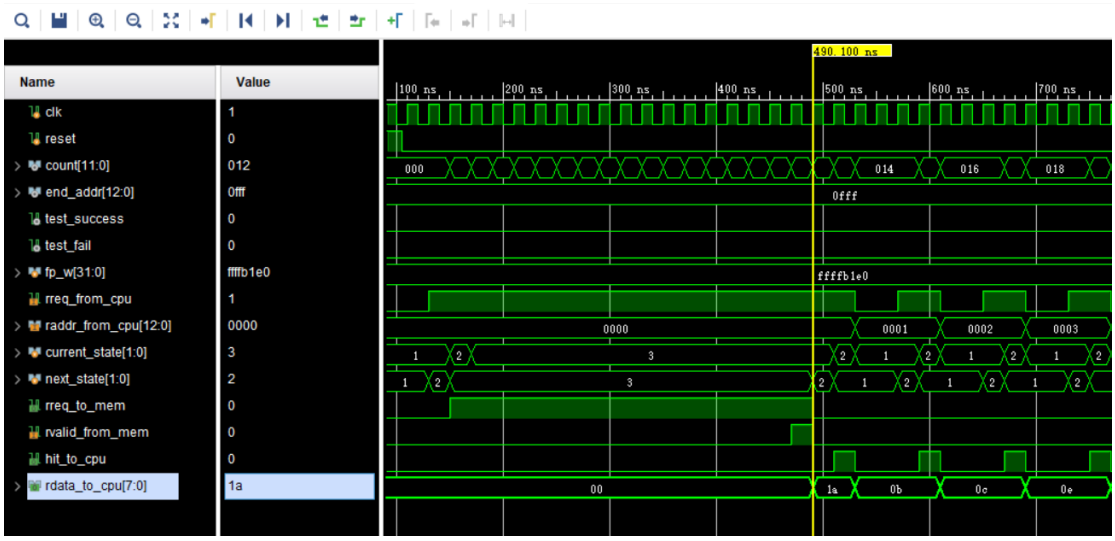
通过拉高 IP 核 blk_mem_gen_0 的写入信号，即可完成对 Cache 的更新。调用 IP 核模块代码如下：

```
blk_mem_gen_0 memory (
    .clka(clk),       // input wire clka
    .wea(rvalid_from_mem),          // 只有在从主存读取且数据有效的时候写使能有效
    .addra(raddr_from_cpu[8:2]),   // cache 块号为 cpu 发送地址的 8 到 2 位
    .dina(temp_reg),       // 更新 cache 的数据
    .douta(data_temp)   // 从 cache 读取的数据
);
```

三、 调试报告

未命中时的仿真：



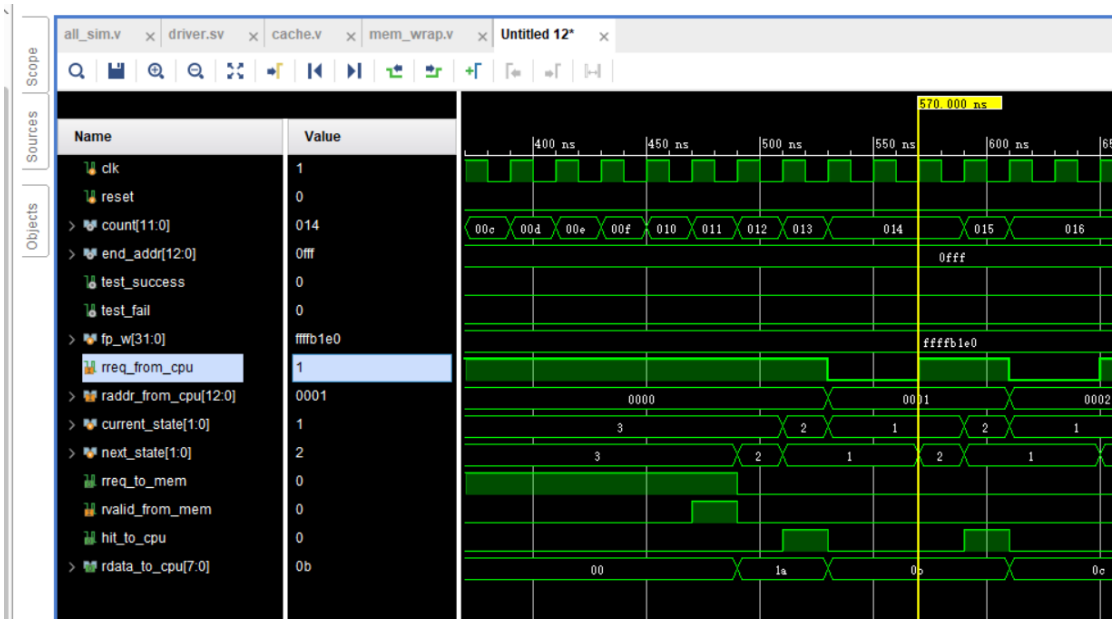


时序分析：

130ns 时，cpu 发出访问申请，状态机次态变为 2，下一个时钟周期现态变为 2，开始比较 cpu 要访问的地址与 Cache 相应块内的 Tag 位以及最高有效位，此时 hit_to_cpu 为 0，未命中，次态变为 3，置访存信号 rreq_to_mem 为 1。下一个时钟周期，进入访问主存阶段。

490ns 时，主存返回数据有效信号，此时次态变为 2，同时在这个时钟周期对 Cache 块内数据进行更新。下一个时钟周期，数据更新成功，现态变为 2，进入比对阶段，hit_to_cpu 变为 1，输出正确数据 1a(26)。下一个时钟周期，现态恢复为 Ready。

命中时的仿真：



时序分析：在 570ns 时，cpu 发出访问申请，状态机次态变为 2，下一个时钟周期现态变为 2，开始比较 cpu 要访问的地址与 Cache 相应块内的 Tag 位以及最高有效位，此时 hit_to_cpu 为 1，已经命中，次态变为 1。本次访问 Cache 命中，在下个时钟周期恢复现态为 1。

仿真代码

```
module all_sim();

reg clk,reset;

wire [11:0] count;

wire [12:0] end_addr = 13'b0_1111_1111_1111;

wire test_success, test_fail;

driver g0(

    clk,

    reset,

    end_addr,

    count,
```

```
        test_success,
        test_fail
        );
    integer fp_w;

    initial begin
    #0
    fp_w=$fopen("result","w");
    reset = 1;
    clk = 0;
    #105
    reset = 0;

    end

    always @(posedge clk)    begin
        if(g0.current_state == 8'b0000_0001)      $display("访问地址为",g0.test_addr);
        if(g0.current_state == 8'b0000_0010 && g0.next_state == 8'b0000_0010 )
$display("等待 Cache 响应，应得到数据",g0.data_from_trace);
        if(g0.cache_hit)      $display("Cache 访问命中!");
        if(g0.current_state == 8'b0010_0000)      $display("Cache 已取回数据",
g0.data_from_cache);
        if(g0.next_state == 8'b0000_0001)          $display("该地址测试正确，将测试下一
个地址\n ---------- ");
```

```
    end

    always begin

        #10 clk = ~clk;

    end

    always @(posedge clk)    begin

        if(test_success)      begin

            $fwrite(fp_w,"TEST PASSED\n");

            $fwrite(fp_w,"Cycles spent on reading cache: %d\n", count);

            $fclose(fp_w);

            $display("======================测试全部通过
========================");

            $stop;

        end

        if(test_fail)    begin

            $fwrite(fp_w,"TEST FAILED\n");

            $fwrite(fp_w,"At read addr %x, expect %x, but get %x\n", g0.test_addr,
g0.data_from_trace, g0.data_from_cache);

            $fclose(fp_w);

            $display("=========================测试未通过，具体请看上面调
试信息=========================");

            $stop;

        end

    end

endmodule
```