

（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： 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

|  |
| --- |
| 1. **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。 |
| 1. 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  ); |
| 1. **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 |