浙江大学

本科实验报告

课程名称：计算机体系结构

姓　　名：刘思锐

学　　院：计算机科学与技术学院

系：

专　　业：计算机科学与技术

学　　号：3200102708

指导教师：陈文智

2022 年　11 月　20 日

浙江大学实验报告

课程名称：计算机体系结构 实验类型：综合

实验项目名称：Cache Design

学生姓名：刘思锐 专业：计算机科学与技术 学号：3200102708

同组学生姓名：陈镛屹 指导老师：陈文智

实验地点：曹西 301 实验日期：2022 年 11 月 15 日

一、 实验目的和要求

1. Understand Cache Line.
2. Understand the principle of Cache Management Unit (CMU) and State Machine of CMU.
3. Master the design methods of CMU.
4. Master the design methods of Cache Line.
5. Master verification methods of Cache Line.

二、 实验内容和原理

1. Design of Cache Line and CMU.
2. Verify the Cache Line and CMU.
3. Observe the Waveform of Simulation.

三、 实验过程和数据记录

3.1 生成 cache 内部信号

本实验采用两路组关联，因此各种信号都对称的分为 1、2 两路。

```
assign word2 = inner_data[addr_word2];                //need to fill in
assign half_word1 = addr[1] ? word1[31:16] : word1[15:0];
assign half_word2 = addr[1] ? word2[31:16] : word2[15:0];          //need to fill in
assign byte1 = addr[1] ?
               addr[0] ? word1[31:24] : word1[23:16] :
               addr[0] ? word1[15:8] :  word1[7:0]   ;
assign byte2 = addr[1] ?
               addr[0] ? word2[31:24] : word2[23:16] :
               addr[0] ? word2[15:8] :  word2[7:0]   ;               //need to fill in

assign recent1 = inner_recent[addr_element1];
assign recent2 = inner_recent[addr_element2];             //need to fill in
assign valid1 = inner_valid[addr_element1];
assign valid2 = inner_valid[addr_element2];               //need to fill in
assign dirty1 = inner_dirty[addr_element1];
assign dirty2 = inner_dirty[addr_element2];               //need to fill in
assign tag1 = inner_tag[addr_element1];
assign tag2 = inner_tag[addr_element2];                  //need to fill in

assign hit1 = valid1 & (tag1 == addr_tag);
assign hit2 = valid2 & (tag2 == addr_tag);                //need to fill in
```

3.2 生成 cache 输出信号

注意的 valid、dirty 与 tag 信号，从实验 4 中可以知道他们的作用是确定换页时是否需要写回，又因为本实验要求 cache 使用 LRU 替换策略，所以需要传出的并不是当前访问的地址所在的块的值，而是即将被换出的块的值，因此当近期使用的是 1 号块时，传出 2 号块的数 valid、dirty 与 tag，反之亦同。

而 hit 信号与最近使用的是哪一个块无关，只要两路中任意一路命中即 cache hit。

```
always @ (posedge clk) begin
    valid <= recent1 ? valid2 : valid1;                  //need to fill in
    dirty <= recent1 ? dirty2 : dirty1;                  //need to fill in
    tag <= recent1 ? tag2 : tag1;                      //need to fill in
    hit <= hit1 | hit2;                        //need to fill in
```

3.3 编写 cache read hit 行为

Cache read hit 时根据 u_b_h_w 的要求取出结果中对应的位输出，同时更新 recent。

```
else if (hit2) begin
    //need to fill in
    dout <=
        u_b_h_w[1] ? word2 :
        u_b_h_w[0] ? {u_b_h_w[2] ? 16'b0 : {16{half_word2[15]}}, half_word2} :
        {u_b_h_w[2] ? 24'b0 : {24{byte2[7]}}, byte2};

    inner_recent[addr_element1] <= 1'b0;
    inner_recent[addr_element2] <= 1'b1;
end
```

3.4 编写 cache write hit 行为

Cache write hit 时根据 u_b_h_w 的要求写入对应的位，并且更新 recent 和 dirty 的值。

```
else if (hit2) begin
    //need to fill in
    inner_data[addr_word2] <=
        u_b_h_w[1] ?        // word
            din
        :
            u_b_h_w[0] ?    // half word
                addr[1] ?       // upper / lower?
                    {din[15:0], word2[15:0]}
                :
                    {word2[31:16], din[15:0]}
            :   // byte
                addr[1] ?
                    addr[0] ?
                        {din[7:0], word2[23:0]}    // 11
                    :
                        {word2[31:24], din[7:0], word2[15:0]} // 10
                :
                    addr[0] ?
                        {word2[31:16], din[7:0], word2[7:0]}    // 01
                    :
                        {word2[31:8], din[7:0]} // 00
    ;
    inner_dirty[addr_element2] <= 1'b1;
    inner_recent[addr_element1] <= 1'b0;
    inner_recent[addr_element2] <= 1'b1;
end
```

input wire [31:0]

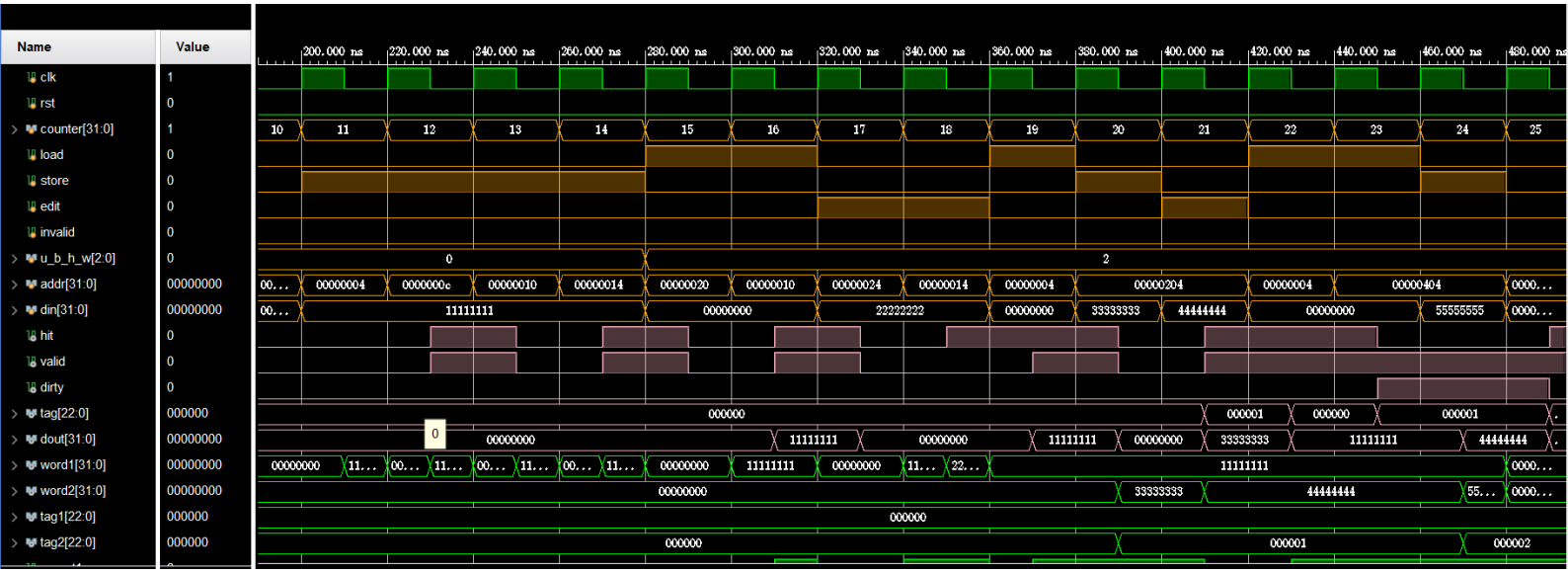
3.5 编写 cache 换页行为

发生换页时，整个块中的内容改为输入的数据，并重置 valid、dirty 和 tag 信息。

```
if (store) begin
    if (recent1) begin  // replace 2
        inner_data[addr_word2] <= din;
        inner_valid[addr_element2] <= 1'b1;
        inner_dirty[addr_element2] <= 1'b0;
        inner_tag[addr_element2] <= addr_tag;
    end else begin
        // recent2 == 1 => replace 1
        // recent2 == 0 => no data in this set, place to 1
        //need to fill in
        inner_data[addr_word1] <= din;
        inner_valid[addr_element1] <= 1'b1;
        inner_dirty[addr_element1] <= 1'b0;
        inner_tag[addr_element1] <= addr_tag;
    end
```

四、 实验结果分析

本次实验不需要上板验证，只需要对照仿真结果。



对照实验指导 PPT，可见所得仿真波形与参考波形相同。cache 对 read/write+hit/miss 共计四种情况都可以正确响应。

五、 讨论与心得

出于本实验使用两路组关联而其中一路的代码已经给出，大部分时候我们只需要对照所给代码写出与之结构对称的另一路，本次实验的难度并不高。唯一的困惑之处在于 valid 与 dirty 信号，最初自然的认为地址访问哪一个块就输出哪一个块的相关信息，结果仿真波形始终有所偏差，在下发 lab4 之后才理解这两个信号的用处。