实验 12 报告

学号：2017K8009922026，2017K8009922032
姓名：康齐瀚,赵鑫浩
箱子号：63

一、实验任务（10%）

完成 TLB 模块的读，写，查找这三个部分的接口（其中有两个查找接口）。

二、实验设计（40%）

（一）总体设计思路

我们首先观察了 TLB 模块的几组接口，分别为 1 个写口，1 个读口和 2 个查找端口，因为这四部分相对独立，所以我们分开实现了这 4 个模块的功能。

（二）重要模块 1 设计：读，写，查找模块

1、工作原理

我们通过组合逻辑直接实现读和查找端口，用时序逻辑实现写端口。

2、功能描述

```
assign r_vpn2 = tlb_vpn2[r_index];
assign r_asid = tlb_asid[r_index];
assign r_g = tlb_g[r_index];
assign r_pfn0 = tlb_pfn0[r_index];
assign r_c0 = tlb_c0[r_index];
assign r_d0 = tlb_d0[r_index];
assign r_v0 = tlb_v0[r_index];
assign r_pfn1 = tlb_pfn1[r_index];
assign r_c1 = tlb_c1[r_index];
assign r_d1 = tlb_d1[r_index];
assign r_v1 = tlb_v1[r_index];
```

读端口根据给定的 r_index 读取出对应 index 的 TLB 项，例如 vpn2,asid 等等。

```
always@(posedge clk)
begin
    if(we) begin
        tlb_vpn2[w_index] <= w_vpn2;
        tlb_asid[w_index] <= w_asid;
        tlb_g[w_index] <= w_g;
        tlb_pfn0[w_index] <= w_pfn0;
        tlb_c0[w_index] <= w_c0;
        tlb_d0[w_index] <= w_d0;
        tlb_v0[w_index] <= w_v0;
        tlb_pfn1[w_index] <= w_pfn1;
        tlb_c1[w_index] <= w_c1;
        tlb_d1[w_index] <= w_d1;
        tlb_v1[w_index] <= w_v1;
    end
end
```

写端口首先判断使能信号，若为 1，则向对应 index 项写入对应的值。

```
assign match0[ 0] = (s0_vpn2==tlb_vpn2[ 0]) && ((s0_asid==tlb_asid[ 0]) || tlb_g[ 0]);
assign match0[ 1] = (s0_vpn2==tlb_vpn2[ 1]) && ((s0_asid==tlb_asid[ 1]) || tlb_g[ 1]);
assign match0[ 2] = (s0_vpn2==tlb_vpn2[ 2]) && ((s0_asid==tlb_asid[ 2]) || tlb_g[ 2]);
assign match0[ 3] = (s0_vpn2==tlb_vpn2[ 3]) && ((s0_asid==tlb_asid[ 3]) || tlb_g[ 3]);
assign match0[ 4] = (s0_vpn2==tlb_vpn2[ 4]) && ((s0_asid==tlb_asid[ 4]) || tlb_g[ 4]);
assign match0[ 5] = (s0_vpn2==tlb_vpn2[ 5]) && ((s0_asid==tlb_asid[ 5]) || tlb_g[ 5]);
assign match0[ 6] = (s0_vpn2==tlb_vpn2[ 6]) && ((s0_asid==tlb_asid[ 6]) || tlb_g[ 6]);
assign match0[ 7] = (s0_vpn2==tlb_vpn2[ 7]) && ((s0_asid==tlb_asid[ 7]) || tlb_g[ 7]);
assign match0[ 8] = (s0_vpn2==tlb_vpn2[ 8]) && ((s0_asid==tlb_asid[ 8]) || tlb_g[ 8]);
assign match0[ 9] = (s0_vpn2==tlb_vpn2[ 9]) && ((s0_asid==tlb_asid[ 9]) || tlb_g[ 9]);
assign match0[10] = (s0_vpn2==tlb_vpn2[10]) && ((s0_asid==tlb_asid[10]) || tlb_g[10]);
assign match0[11] = (s0_vpn2==tlb_vpn2[11]) && ((s0_asid==tlb_asid[11]) || tlb_g[11]);
assign match0[12] = (s0_vpn2==tlb_vpn2[12]) && ((s0_asid==tlb_asid[12]) || tlb_g[12]);
assign match0[13] = (s0_vpn2==tlb_vpn2[13]) && ((s0_asid==tlb_asid[13]) || tlb_g[13]);
assign match0[14] = (s0_vpn2==tlb_vpn2[14]) && ((s0_asid==tlb_asid[14]) || tlb_g[14]);
assign match0[15] = (s0_vpn2==tlb_vpn2[15]) && ((s0_asid==tlb_asid[15]) || tlb_g[15]);
```

两个查找端口互相独立，各自查找，将 s0_vpn2 和 s0_asid 和 TLB 中的部分比较，依次比较，看 16 项 TLB 中是否有命中的，若有命中，则将该位的 match 信号置 1，如果有其中 1 个 match 信号值为 1，则置 found 信号为 1，表示找到匹配的 TLB 项。

```
assign s0_index    =  ({4{match0[ 0]}} & 4'h0)
                    | ({4{match0[ 1]}} & 4'h1)
                    | ({4{match0[ 2]}} & 4'h2)
                    | ({4{match0[ 3]}} & 4'h3)
                    | ({4{match0[ 4]}} & 4'h4)
                    | ({4{match0[ 5]}} & 4'h5)
                    | ({4{match0[ 6]}} & 4'h6)
                    | ({4{match0[ 7]}} & 4'h7)
                    | ({4{match0[ 8]}} & 4'h8)
                    | ({4{match0[ 9]}} & 4'h9)
                    | ({4{match0[10]}} & 4'ha)
                    | ({4{match0[11]}} & 4'hb)
                    | ({4{match0[12]}} & 4'hc)
                    | ({4{match0[13]}} & 4'hd)
                    | ({4{match0[14]}} & 4'he)
                    | ({4{match0[15]}} & 4'hf);
```

我们用于多位选择器来判断命中的 TLB 的 index。

```
assign s0_pfn = (s0_odd_page) ? tlb_pfn1[s0_index] : tlb_pfn0[s0_index];
assign s0_c = (s0_odd_page) ? tlb_c1[s0_index] : tlb_c0[s0_index];
assign s0_d = (s0_odd_page) ? tlb_d1[s0_index] : tlb_d0[s0_index];
assign s0_v = (s0_odd_page) ? tlb_v1[s0_index] : tlb_v0[s0_index];
```

在得到 index 我们还需要通过 odd_page 信号判断取奇数页还是偶数页。

三、实验过程（50%）

（一）实验流水账

2019/11/28 14：00 ~ 15：30 写完代码，仿真通过，上板通过

（二）错误记录

1、错误 1：读数据出错

（1）错误现象

仿真对比显示，读出 TLB 表项时出错

（2）分析定位过程

并没有从波形来找，直接看的代码，发现读出的 g 位这一位没有相应的读出的组合逻辑

（3）错误原因

G 位没有对应的组合逻辑来读出。

（4）修正效果

增加 G 位读出的组合逻辑

```
assign r_g      = tlb_g   [r_index];
```

增加之后就能顺利通过仿真测试了。

（5）归纳总结（可选）

粗心导致的，每次写完代码之后对于一些重要的位置要反复检查，查看其在逻辑上是否有漏洞。