实验 15 报告

学号: 2018K8009929043 2018K8009929035
姓名: 曾冕 张翔雨
箱子号： 06

一、实验任务（10%）

本实验主要在 lab14 的基础上加入了三种 TLB 例外处理，主要更改了 IF（包括 pre-if 和 IF 级）流水级的逻辑和 EXE 级的逻辑。

二、实验设计（40%）

（一）总体设计思路

本次实验主要更新了涉及了虚实地址转换的 IF 级流水线和 EXE 级流水线，其余流水线只是添加了相关总线流水信息，分模块设计思路如下：

1. Pre-IF 模块：产生 tlb_refill 和 tlb_invalid 信号， 将原 next_pc 改为 tlb_nextpc，新 next_pc 会根据 mapped 的情况选择是否映射。定义 pre_ex 信号，当预取指出错会取消发至总线的取指请求，同时添加流水级 readygo 信号判断逻辑使流水级正常运行。添加了 3 个触发器，在 pre-if 向 IF 级流水的时候暂存 pre-if 级的 tlb 例外信息和 tlb_nextpc。
2. IF 模块：根据上述三个触发器在 IF 模块完成 excode 报出和 badvaddr 的更新，同时根据是否发生 tlb 例外更新 fs_pc。
3. EXE 模块：改动较为简单，处理了 refill，invalid，modify 三种 tlb 例外，不涉及流水控制信号，只需要在 es_ex 和 es_excode 等例外相关信号的更新。同时整合 IF 级和 EXE 级的 refill 例外，向下传递直到 wb 级报出

CPU 总体设计图如下图所示：

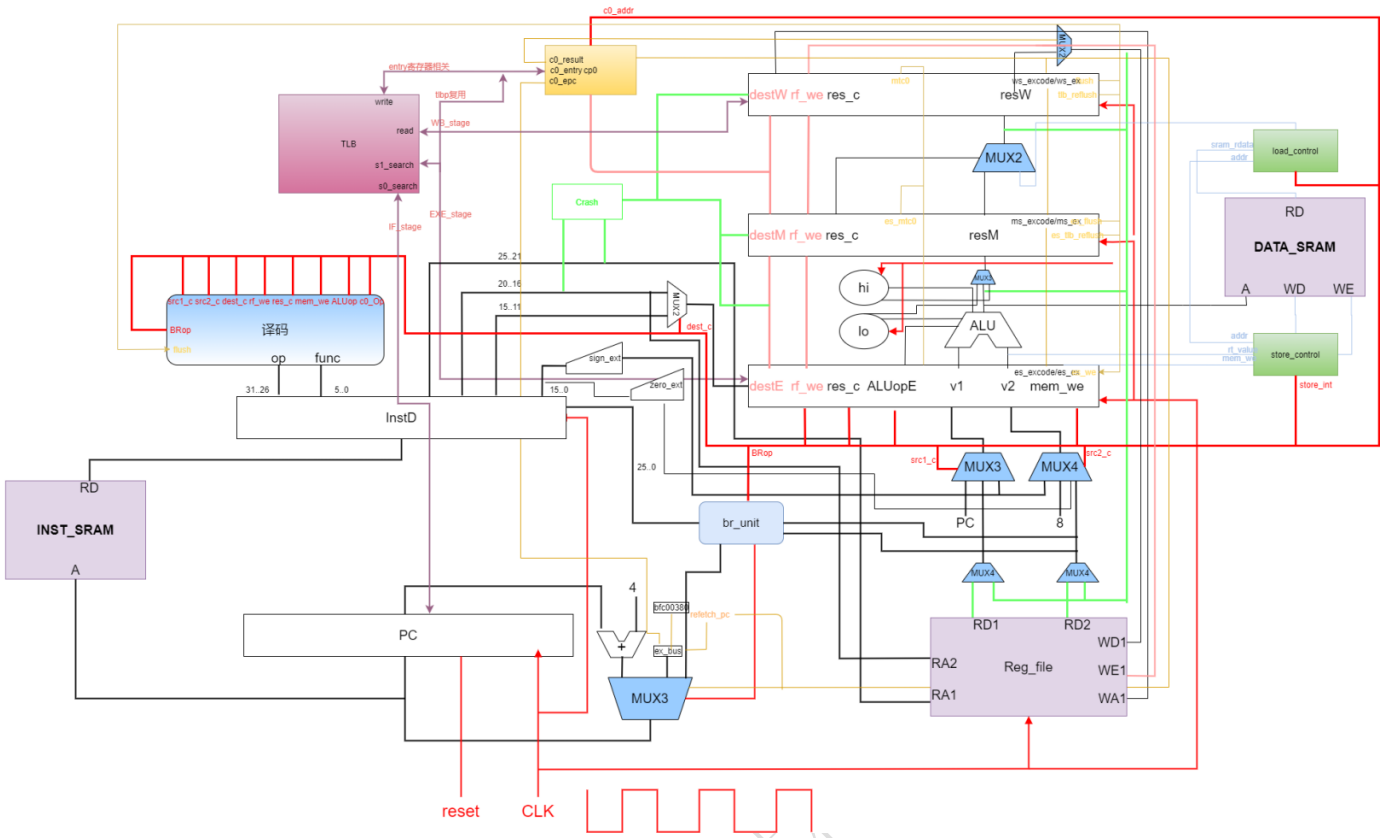

图 01: 整体 CPU 设计

（二）重要模块 1 设计：IF 模块（描述 lab15 更新）

1、工作原理（内容描述）

接收 ws 模块报出的 tlb_refill 错误，跳转到对应例外处理地址。内部的信号更新见详细内容描述

2、接口定义（仅展示 lab14 更新）

名称	方向	位宽	功能描述
Fs_to_ds_bus	OUT	104	向下一级额外流水了 tlb_refill_r 信号
Ws_tlb_refill	IN	1	上一次处理器的 refill 信号，决定例外跳转地址

3、 功能描述

将原本 next_pc 更新为 tlb_nextpc：

```
//modified in Lab15
assign tlb_nextpc    =
                       ex_eret    ? ex_bus:
                       (br_bus_valid & bd_done) ? br_bus_r:
                       (br_bus_valid & ~bd_done) ?  bd_bus_r :seq_pc;
```

图 02:：tlb_nextpc 的设计

Pre-IF 级：

根据 tlb 查找信息，返回取值阶段两种 tlb 例外信号如下：

```
assign tlb_refill  = mapped & !s0_found;
assign tlb_invalid = mapped & s0_found & !s0_v;
```

图 03: tlb 例外信号

用 tlb_nextpc 作为查找的 tlb 映射关系如下，当 pc 信号高位和次高位不为 10 的时候表明不在可映射段，这时候需要用物理地址，物理地址的拼接方式如下图 phaddr 所示：

```
assign s0_vpn2 = tlb_nextpc[31:13];
assign s0_odd_page = tlb_nextpc[12];

assign phaddr =  {s0_pfn, tlb_nextpc[11:0]} ;
assign mapped = (tlb_nextpc[31:30] != 2'b10);
assign nextpc = mapped ? phaddr : tlb_nextpc;
```

图 04: next_pc 选择

由于 tlb 例外在 pre-if 级发生，当指令流动至 if 级，对应的 tlb 信息则是下一条指令的信息，所以在向 if 级流水的时候需要用寄存器暂存 tlb 例外信息和此时的 tlb_nextpc 值，这里仅展示一个触发器的逻辑（当可以向 if 级流水的时候进行暂存）如下所示：

```
always @(posedge clk)
begin
    if (reset)
        tlb_refill_r <= 1'b0;
    else if (to_fs_valid && fs_allowin)
        tlb_refill_r <= tlb_refill;
end
```

图 05:暂存 pre-if 级信息

IF 级部分：

fs_pc 需要在产生 tlb 例外的时候选择 tlb_nextpc。

在整个 IF 级的错误处理分为两个大类，地址错例外和 tlb 例外，这两类例外其实在 preif 级就可以判断出来，对于 badvaddr 的值，当为地址错例外的时候用 fs_pc 作为值，当为 tlb 例外的时候需要用之前暂存的 tlb_nextpc_r 值，整体代码如下图所示：

```
assign fs_addr_error = (fs_pc[1: 0] != 2'b0) ;
assign pre_addr_error = (nextpc[1: 0] != 2'b0) ;
assign fs_ex = fs_valid ? (fs_addr_error|tlb_invalid_r|tlb_refill_r):1'b0;
assign pre_ex = tlb_refill|tlb_invalid|pre_addr_error;
assign fs_excode = fs_ex ? ((tlb_refill_r | tlb_invalid_r) ? 5'h02 :5'h04):5'h0;
assign badvaddr_value = fs_addr_error ? fs_pc :
                        (tlb_invalid_r|tlb_refill_r) ? tlb_nextpc_r : 32'd0;
```

图 06:整体例外处理代码设计

增加了 pre_ex 错误，包含 pre-if 级的 tlb 例外和地址错例外，这个信号会用于阻塞向总线发送请求和拉低

cancel 信号，涉及代码如下所示：

```
assign to_fs_valid  = ~reset & ((inst_sram_addr_ok & inst_sram_req) || pre_ex);//lab15
assign inst_sram_req = !br_stall & !reset & fs_allowin & !request_control & !fs_ex & !pre_ex;
else if((ws_ex|eret_flush |tlb_reflush)&((to_fs_valid & !pre_ex) | (!fs_allowin & !fs_ready_go)))
 //valid 和 ready_go 用的同一个信号 所以需要保证为高的时候拉低 cancel
```

在本小组设计中，to_fs_valid 信号相当于同时作为了 pre-if 级的 valid 和 ready_go 信号，这里会牵扯到需要 cancel 的情况的第一种，需要做特别设计，所以这里需要用 pre_ex 来控制拉低 cancel 信号，具体的逻辑见错误记录 1。

最后，由于之前使用 ex_bus 寄存器存储发生例外时的跳转地址，tlb_refill 是特别的例外处理地址，所以对于添加了 tlb 例外的 ex_bus 寄存器代码如下所示：

```
always @(posedge clk)
begin
    if(eret_flush)
        ex_bus <= c0_epc;
    else if (ws_ex & ws_tlb_refill)
        ex_bus <= 32'hbfc00200;
    else if (ws_ex & !ws_tlb_refill)
        ex_bus <= 32'hbfc00380;
    else if (tlb_reflush)
        ex_bus <= refetch_pc;
end
```

图 07:例外跳转地址寄存器逻辑

（三）重要模块 2 设计：EXE 模块

1、工作原理

添加了三种 tlb 例外处理，同时将 IF 阶段和 EXE 阶段都可能出现的 refill 信号进行整合，向下一级流水传递

2、接口定义（lab14 更新）

名称	方向	位宽	功能描述
Es_to_ms_bus	OUT	134	增添了向下一级传递的综合 refill 信号的总线
ds_to_es_bus	IN	214	增添了带有 IF 阶段 refill 的总线

3、功能描述

EXE 阶段的例外处理基本与 IF 阶段类似，而且不需要对流水线控制信号进行太多处理，同时多了一个 modify 例外，具体的处理代码如下所示：

```
//Lab15
assign phaddr =  (inst_swl|inst_lwl) ? {s1_pfn, es_alu_result[11:2],2'b00}:{s1_pfn,es_alu_result[11:0]};
assign mapped =  (es_alu_result[31:30] != 2'b10 );
assign vaddr  =(inst_swl|inst_lwl) ? {es_alu_result[31:2],2'b00} : es_alu_result;

assign es_tlb_refill = !s1_found & mapped & (es_load_op|es_store);
assign es_tlb_invalid = s1_found & !s1_v & mapped & (es_load_op|es_store);
assign tlb_modify = s1_found & s1_v & !s1_d & mapped & es_store;
```

图 08:EXE 阶段 tlb 例外信号逻辑

在这里按照之前的讲义要求，在指令为 swl 和 lwl 指令的时候，把物理地址和虚拟地址低两位都强制抹 0，refill 和 invalid 只会在涉及到 load 或者 store 的时候触发，modify 时在 store 时映射段对应的 dirty 位不为 1 的时候触发。

查阅指令集手册后，更新 excode 的逻辑如下图所示：

```
assign  es_excode_temp = tlb_modify           ? 5'h01 :
                         (es_tlb_invalid | es_tlb_refill) & es_load_op ? 5'h02:
                         (es_tlb_invalid | es_tlb_refill) & es_store   ? 5'h03:
                         exception_overflow   ? 5'h0c :
                         es_adel              ? 5'h04 :
                         es_ades              ? 5'h05 : 5'b0 ;
```

图 09:EXE 阶段 excode 逻辑代码

同时将新加入的三种例外填入 es_ex 错误中，（在 es_we 中有！es_ex 选项，从而在发生例外时阻塞请求等，这些处理之前的实验报告中都有提及，这里就不重复展示代码了）

```
assign  es_ex =  es_valid ? (ds_ex | exception_overflow | es_ades | es_adel
                           | es_tlb_invalid |es_tlb_refill|tlb_modify) : 1'b0;
```

图 10:EXE 阶段 es_ex 逻辑代码

三、实验过程（50%）

（一）实验流水账

12.7 晚上 8:00-23:00 完成代码的设计和编写，仿真出现错误，暂时搁置

12.8 早 7:30 起床修改了一处代码逻辑，通过仿真。

12.21 开始写实验报告，零零碎碎大约花费 3 小时完成。

（二）错误记录

1、错误 1：例外处理的下一条指令被错误的取消

（1） 错误现象

cancel 错误拉高导致例外处理的第一条指令被取消掉了

（2）分析定位过程

观察波形，发现是 cancel 信号错误的被额外拉高了几拍，导致例外处理结束后下一条指令也被取消掉，波形如下图所示：

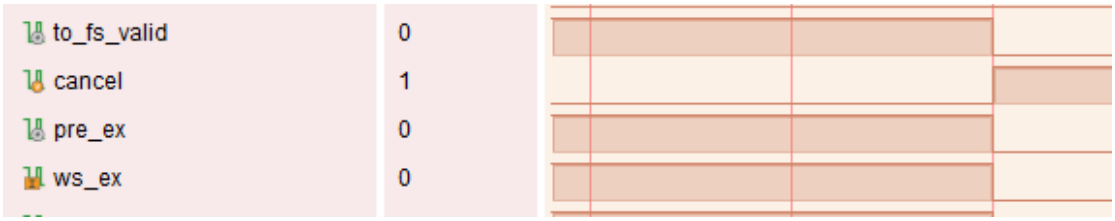

图 11:cancel 错误拉高示意图

（3）错误原因

在之前的实验设计中，to_fs_valid 信号与 pre_if 级的 ready_go 信号，并无区别，因此代码中将二者并为一个信号。

但在这个实验中，当 preif 级发生 TLB 例外时，需要将 preif 级的 ready_go 信号一直拉高保证流水级正常运转。在一开始的代码处理中认为 pre_if 级不会产生例外，所以当例外刷新 if 级时会发现 to_fs_valid 信号拉高，错误的判断为需要取消的情况。

（4）修正效果，

修改前后代码变化如下图所示：

```
else if((ws_ex|eret_flush |tlb_reflush)&(to_fs_valid | (!fs_allowin & !fs_ready_go)))
else if((ws_ex|eret_flush |tlb_reflush)&((to_fs_valid & !pre_ex) | (!fs_allowin & !fs_ready_go)))
begin
    cancel <= 1'b1;
```

图 12:gitee 代码修改记录

在修改后仿真通过，上板通过。

四、实验总结（可选）

本次实验主要涉及到三种 tlb 例外的添加，其中 exe 阶段数据访问的 tlb 例外添加是比较简单的，因为在 exe 阶段的只有一条指令，所有的信息都是当前指令生成的，不需要使用额外的寄存器去存相应的信息，而在取指级，由于 preif 级和 if 级对应的是两条 pc，则需要使用一些寄存器存储相关信息以保证对应的例外标志不会出错，还是比较麻烦的。

同时在这次实验中，发现讲义中要求出现地址相关例外时不能向总线发出请求，第一版的代码设计中，虽然发出了请求但也能通过测试，因为被标记例外的指令实际上是什么也做不了的。但是这与设计思想不符，因为这会导致软件设计人员无法控制的地址访问，因此又设计了发生例外取消访存的逻辑，顺便把之前的取指错例外加上，感觉实验的测试可以完善上这一部分。