实验 4 报告

学号：2017K8009922027
姓名：张磊
箱子号：39

一、实验任务（10%）

任务：在实验 3 的 CPU 代码基础上，加入适当的逻辑处理寄存器写后读数据引发的流水线冲突（阻塞方式），运行 func_lab4，要求成功通过仿真和上板验证；

二、实验设计（40%）

（一）总体设计思路

1. 首先通过将 ES，MS，WS 三个阶段的寄存器写地址，通过端口送回 DS 阶段，若此时发现 DS 阶段需要读取的寄存器号与从 ES，DS，WS 三个阶段返回的写寄存器号相等，则修改 DS 阶段的 ready_go 信号，达到阻塞 DS 阶段流水的目的（此时并未考虑 DS,ES,MS,WS 阶段指令是否有效，也没有考虑特殊指令）；
2. 在 1 的基础上，添加判断每一级流水的指令是否有效（即通过判断每一级的 valid 信号来确认），若某一级的指令无效，则直接返回 5’b0（即 0 号寄存器，因为 0 号寄存器不存在写后读数据相关）；
3. 在 2 的基础上，通过区分不同的指令来判断当前流水级是否需要阻塞，特殊的指令可分为：
   （1）无目的操作数，有目的操作数；
   （2）无源操作数，有 1 个源操作数，有 2 个原操作数；

完成指令区分后可以达到避免“伪流水线阻塞”的目的，加快流水线进程；

三、实验过程（50%）

（一）实验流水账

1. 2019 年 9 月 20 日 13:30-15:00 完成了总体设计思路的第 1 阶段；
2. 2019 年 9 月 15 日 18:00-20:00 完成了总体设计思路的第 2，3 阶段；

（二）重要模块 1 设计：写寄存器号返回模块；

这里仅以 WS 阶段为例（ES，MS 阶段相同）；

```
module wb_stage(
    input                           clk           ,
    input                           reset         ,
    //allowin
    output                          ws_allowin    ,
    //from ms
    input                           ms_to_ws_valid,
    input  [`MS_TO_WS_BUS_WD -1:0]  ms_to_ws_bus  ,
    //to rf: for write back
    output [`WS_TO_RF_BUS_WD -1:0]  ws_to_rf_bus  ,
    //trace debug interface
    output [31:0] debug_wb_pc     ,
    output [ 3:0] debug_wb_rf_wen ,
    output [ 4:0] debug_wb_rf_wnum,
    output [31:0] debug_wb_rf_wdata,

    //pipeline block
    output [4:0] reg_dest_ws
);
```

1、工作原理

先判断当前流水级信号是否有效（ws_valid），若无效，直接返回 5’b0，否则根据该条指令是否需要写寄存器（ws_dest_type，若 ws_dest_type == 1 则说明需要写返回 ws_dest），否则返回 5’b0；

```
assign reg_dest_ws = ws_valid ? ((ws_dest_type)?(ws_dest):(5'b0)):
                                5'b0;
```

其中 ws_dest_type 的值在 ID 阶段由指令译码逻辑决定，然后跟随 bus 一起流水；

```
//pipeline
assign ds_dest_type = (inst_beq | inst_bne | inst_jr) ? `NO_DEST : `DO_DEST;
assign ds_src_type = (inst_jal | inst_lui) ? `NO_SRC : `DO_SRC;
```

宏定义在头文件中：

```
`define NO_DEST 0
`define DO_DEST 1
```

2、功能描述

将 ES，MS，WS 阶段的写寄存器号返回 ID 阶段；

（三）重要模块 2 设计：ID 阶段判断是否阻塞模块

阻塞方法最关键的一步就在于 ID 阶段的 ready_go 信号；

1、工作原理

首先通过 ds_src_type 判断当前处于 ID 阶段的指令是否需要读寄存器（这里仅简单区分为需要读寄存器和不需要读寄存器）；

```
//pipeline
assign ds_dest_type = (inst_beq | inst_bne | inst_jr) ? `NO_DEST : `DO_DEST;
assign ds_src_type = (inst_jal | inst_lui) ? `NO_SRC : `DO_SRC;
```

对 ready_go 信号进行修改：

```
assign ds_ready_go = (ds_src_type)?( (reg_dest_es | reg_dest_ms | reg_dest_ws) ?
                        ( (reg_dest_es == rs || reg_dest_es == rt ||
                           reg_dest_ms == rs || reg_dest_ms == rt ||
                           reg_dest_ws == rs || reg_dest_ws == rt)?(1'b0):
                           (1'b1) ):(1'b1)):(1'b1);
```

2、功能描述

若 ID 阶段不需要源操作数（ds_src_type == 0），则赋值为 1，否则进一步判断，若 ES,MS,WS 三个阶段写寄存器号均为 0 则也不用阻塞（注意，此时并不一定是真的写 ES,MS,WS 阶段寄存器号为 0，也有可能是处于 ES,MS,WS 阶段的指令不需要写寄存器，因为如果不用写寄存器的话，我们也返回 0 号寄存器）直接赋值为 1，否则再判断是否有 ID 阶段的读寄存器号与 ES,MS,ES 阶段的写寄存器号相等，若相等，则说明有写后读数据相关，赋值为 0，否则，赋值为 1；

四、实验总结（可选）

1. 本次实验设计仍有可以改进的地方，如在区分指令是否需要原操作数时，仅进行了简单地 2 类（有源操作数，无源操作数），实际上可以继续细分，进一步优化流水线阻塞；

2. 在 ES,MS,WS 阶段判断是否需要写寄存器时，代码逻辑稍显复杂，其实可以直接通过写使能信号判断；