实验 7 报告

组员 1：陈飞羽 2018K8009929031
组员 2：彭思叡 2018K8009908040
箱子号：45

一、实验任务（10%）

在 lab6 的 CPU 代码基础上添加指令，具体包括转移指令 BGEZ、BGTZ、BLEZ、BLTZ、J、BLTZAL、BGEZAL、JALR 指令以及访存指令 LB、LBU、LH、LHU、LWL、LWR、SB、SH、SWL、SWR。

二、实验设计（40%）

（一）总体设计思路

在本次实验中，指令 BGEZ、BGTZ、BLEZ、BLTZ 复用了之前 BNE、BEQ 指令的数据通路与控制信号，只是新增了跳转条件信号 rs_lt_zero、 rs_gt_zero；J 指令复用了 JAL 的通路；BLTZAL、BGEZAL、JALR 复用了 JAL 与 JR 指令的 ALU 输入通路与跳转总线（br_taken、br_target）。

LB、LBU、LH、LHU 的访存通路与 LW 相同，LB、LH 符号扩展而 LBU、LHU 零扩展，在 mem_result 处添加了多路选择器，从 wire 类型 ld_bhw_rdata、ld_wr_rdata、ld_wl_rdata 中选择。其中 LWL、LWR 指令的从 data_sram_rdata 与 ms_rt_value 读取并且拼接，结果分别保存在 ld_wl_rdata、ld_wr_rdata。

要实现 SB、SH、SWL、SWR 指令，只要在 EXE 级为这些指令生成各自的 wen 与 wdata 信号，并且向 data_sram_wen 与 data_sram_wdata 添加多路选择器即可。

总体设计图如图 1。分支跳转指令因无新增数据通路在图中无展现，且图中仅展现生成 LOAD/STORE 指令 wdata,wen,rdata 的封装模块，具体实现见重要模块设计 1，2。

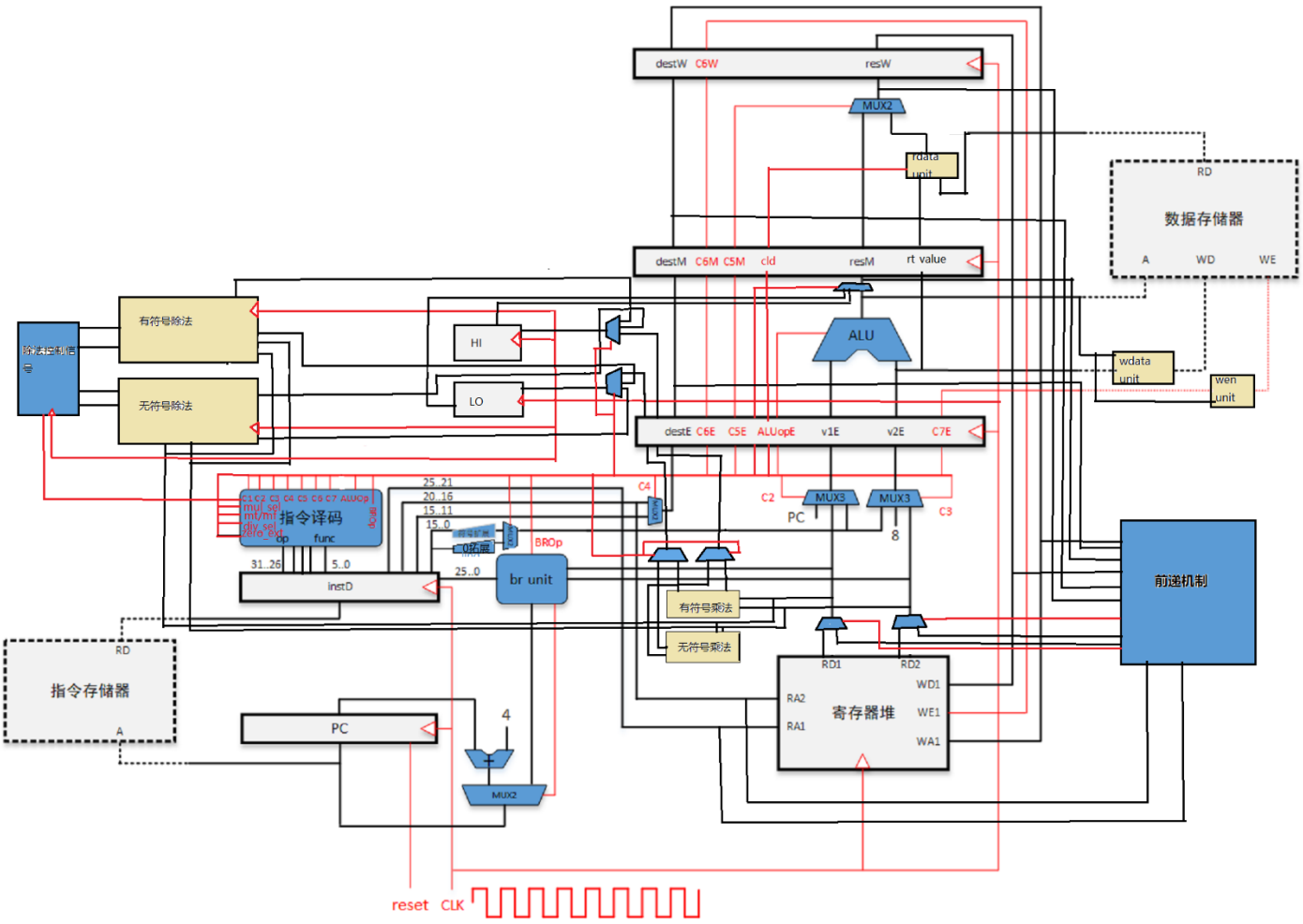

图 1 总体设计图

（二）重要模块 1 设计：LWL/LWR 模块

1、工作原理

实现非对齐内存读取并写入寄存器。

2、接口定义

名称	方向	位宽	功能描述
data_sram_rdata	IN	32	从 4 字节对齐地址读取的内存内容
ms_rt_value	IN	32	目标寄存器原有内容
mem_result	OUT	32	根据指令读取的内存内容
ms_mem_op	IN	7	读取指令的类型，one-hot 编码
ld_bhw_rdata	IN	32	LB、LBU、LH、LHU、LW 的读取结果

3、功能描述

这一模块为组合逻辑实现。

首先确定原地址的低两位地址，低两位地址为 0、1、2、3 对应信号 addr_offset0\1\2\3。随后生成 LWL 与 LWR 的 wdata，生成逻辑主要是 rt_value 与 data_sram_rdata 根据地址偏移量进行不同位数的拼接。LWL 指令与 LWR 指令执行结果分别见表 1，表 2。

访存地址低 2 位	LWL 执行结果			
	[31:24]	[23:16]	[15:8]	[7:0]
2’b00	data_sram_rdata[7:0]	rt_value[23:0]		
2’b01	data_sram_rdata[15:8]	data_sram_rdata[7:0]	rt_value[15:0]	
2’b10	data_sram_rdata[23:16]	data_sram_rdata[15:8]	data_sram_rdata[7:0]	rt_value[7:0]
2’b11	data_sram_rdata[31:24]	data_sram_rdata[23:16]	data_sram_rdata[15:8]	data_sram_rdata[7:0]

表 1 LWL 指令执行结果

访存地址低 2 位	LWR 执行结果			
	[31:24]	[23:16]	[15:8]	[7:0]
2’b00	data_sram_rdata[31:24]	data_sram_rdata[23:16]	data_sram_rdata[15:8]	data_sram_rdata[7:0]
2’b01	rt_value[31:24]	data_sram_rdata[31:24]	data_sram_rdata[23:16]	data_sram_rdata[15:8]
2’b10	rt_value[31:16]		data_sram_rdata[31:24]	data_sram_rdata[23:16]
2’b11	rt_value[31:8]			data_sram_rdata[31:24]

表 2 LWR 指令执行结果

（三）重要模块 2 设计：SWL/SWR 模块

1、工作原理

实现非对齐内存写入。

2、接口定义

名称	方向	位宽	功能描述
data_sram_wdata	OUT	32	准备写入 4 字节对齐地址的内容
data_sram_wen	OUT	4	第 i 位决定第 i 个字节是否有效，只有有效数据才会写入
es_mem_we 与 es_valid	IN	1*2	EXE 级的状态，决定写入是否有效
es_mem_op	IN	7	读取指令的类型，one-hot 编码
es_rt_value	IN	32	源寄存器内容
st_bhw_wdata	IN	32	SB、SH、SW 的写入结果
sb_wen&&sh_wen	IN	4*2	SB、SH 的按字节写使能信号

3、功能描述

这一模块为组合逻辑实现。

addr_offset0\1\2\3 信号生成与 LWL\LWR 模块相同。随后生成 SWL 与 SWR 的 wdata，生成逻辑略有不同，只确保了有效数据的正确性。SWR,SWL 的输出数据具体见表 3，表 4。

访存地址低 2 位	SWR 输出数据			
	[31:24]	[23:16]	[15:8]	[7:0]

2’b00	rt_value[31:0]		
2’b01	rt_value[23:0]		8’b0
2’b10	rt_value[15:0]	16’b0	
2’b11	rt_value[7:0]	24’b0	

表 3 SWR 指令输出数据

访存地址低 2 位	SWL 输出数据			
	[31:24]	[23:16]	[15:8]	[7:0]
2’b00	24’b0			Rt_value[31:24]
2’b01	16’b0		rt_value[31:16]	
2’b10	8’b0	rt_value[31:8]		
2’b11	rt_value[31:0]			

表 4 SWL 指令执行结果

此外，还生成了写使能信号，具体实现逻辑如下：

```
data_sram_wen=     ({4{es_mem_we&&es_valid}}&({4{es_mem_op_b}} & sb_wen  |
                   {4{es_mem_op_h}}                            & sh_wen  |
                   {4{es_mem_op_wr}}                           & swr_wen |
                   {4{es_mem_op_wl}}                           & swl_wen |
                   {4{es_mem_op_w}}) ;
```

三、实验过程（50%）

（一）实验流水账

10.25 22：00 – 23：30 完成分支跳转指令的添加，并通过前 58 个测试。

10.26 15：50 – 18：00 完成 LOAD/STORE 指令的添加，通过全部测试且上板成功。

10.26 20：00 – 22：00 完成实验报告撰写。

10.27 15：00 – 16：00 修改实验报告。

（二）错误记录

1、错误 1：写入值错误

（1）错误现象

报错信息如图 7。

```
[1258527 ns] Error!!!
    reference: PC = 0xbfc405f0, wb_rf_wnum = 0x1f, wb_rf_wdata = 0xbfc405f8
    mycpu    : PC = 0xbfc405f0, wb_rf_wnum = 0x1f, wb_rf_wdata = 0x00000000
```

图 2 错误 1 报错信息

（2）分析定位过程

首先，追溯至 ds_pc=0xbfc405f0 的时刻，发现该指令为 bltzal 指令。然后从 rf_wdata 一路向前追溯，发现错误数据来自 alu_result。拉取全部 alu 相关信号，发现 alu_op 信号异常，为全零。

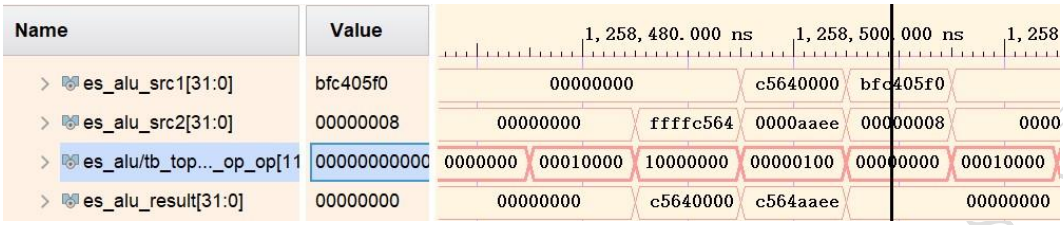

图 3 错误 1 异常波形

（3）错误原因

本次添加的指令未设置 ALU 操作。

（4）修正方法

将 BLTZAL,BGEZAL,JALR,LB,LBU,LH,LHU,LWL,LWR,SB,SH,SWL,SWR 的 ALU 操作设置为加法。

具体实现如下：

```
assign alu_op[ 0] = inst_addu   | inst_addiu | inst_add    | inst_addi | inst_lw  | inst_sw | inst_jal
                    | inst_bgezal | inst_jalr  | inst_bltzal | inst_lwr  | inst_lwl | inst_lb | inst_lbu
                    | inst_lh     | inst_lhu   | inst_sb     | inst_sh   | inst_swl | inst_swr;
```

四、实验总结（可选）

通过本次实验，我意识到电路设计时仔细认真，仿真前检查一遍代码，可以大大减少 debug 工作量。