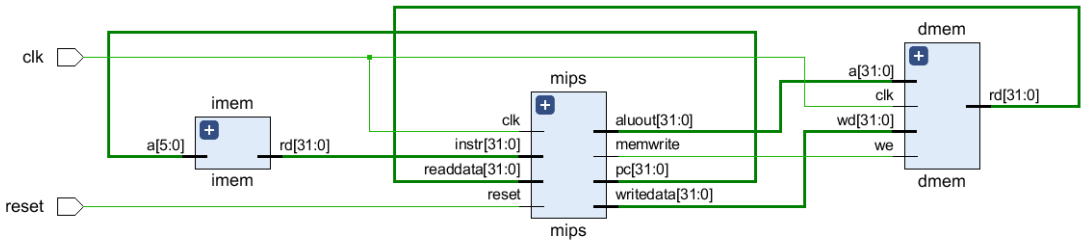
作业 1——单周期 MIPS 处理器

实验报告

贾子安 18307130017

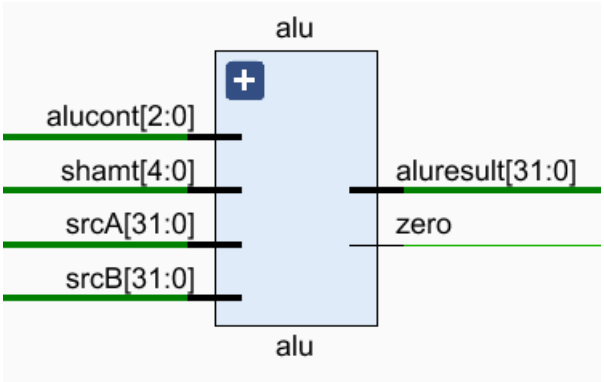
1.总体结构



imem：指令存储器
dmem：数据存储器
clk, reset：输入的时钟与重置信号
mips：处理器核心

2.核心中各元件

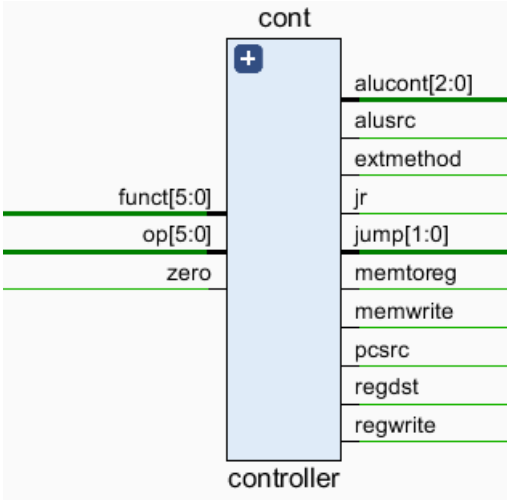
2.1.ALU



此 ALU 的功能表如下：

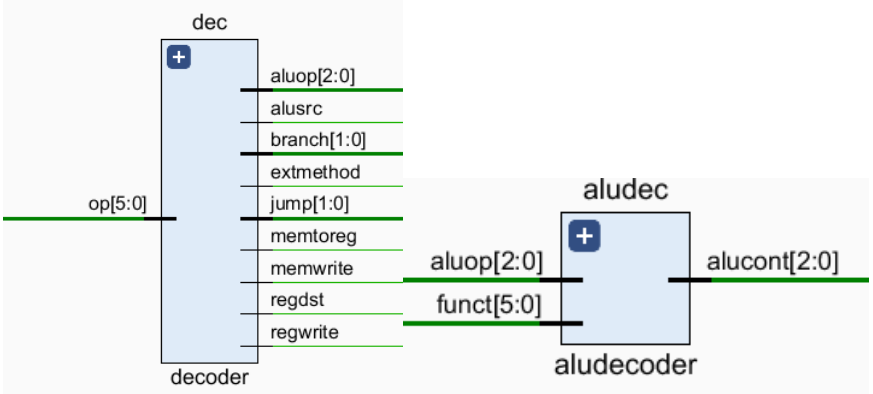
[2:0]alucont	**功能**
000	srcA & srcB
001	srcA | srcB
010	srcA + srcB
011	srcA - srcB
100	SLT srcA srcB
101	srcB SLL shamt
110	srcB SRL shamt
111	srcB SRA shamt

2.2.控制器



alusrc：选择 srcB 的来源(符号扩展器 or 寄存器输出)
extmethod：选择符号扩展方式(全填 0 or 全填符号位)
jr：是否为 jr 指令，通往一个 pc 多路选择器
jump[1]：是否需要 jump，通往一个 pc 多路选择器
jump[0]：是否是 jal 指令，前往寄存器
memtoreg：是否正在把 dmem 中读出的数据存入寄存器
memwrite：是否写 dmem
pcsrc：branch 是否跳转，通往一个 pc 多路选择器
regdst：将要被写的寄存器地址
regwrite：是否写寄存器

控制器包含两部分：maindecoder 和 aludecoder:



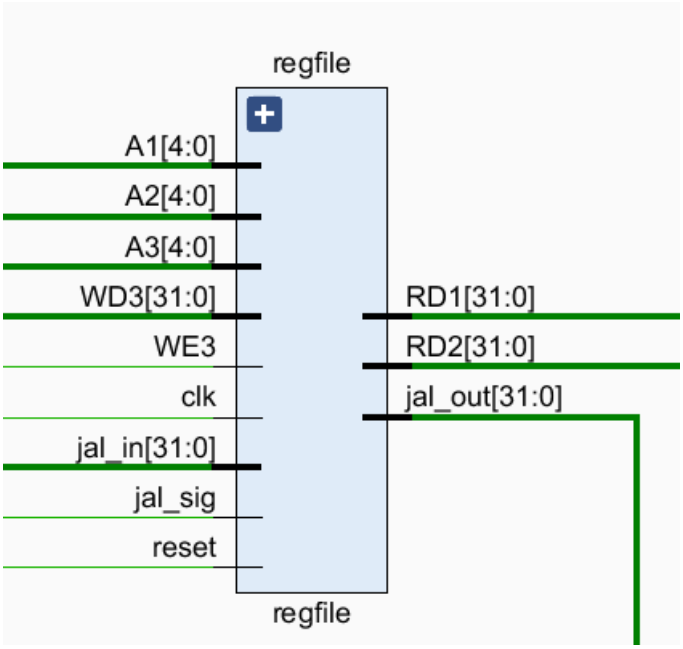
maindecoder 分析输入的 op，给出除 alocont 以外的控制信号并输出 aluop 信号，而 aludecoder 接受 aluop 信号并结合输入的 funct，输出 alucont 控制信号给 ALU

以下是 maindecoder 和 aludecoder 的真值表：

Maindecoder										
指令	op	aluop	alusrc	branch	extmethod	jump	memtoreg	memwrite	regdst	regwrite
R-type	000000	010	0	00	0	00	0	0	1	1
addi	001000	011	1	00	1	00	0	0	0	1
andi	001100	100	1	00	0	00	0	0	0	1
ori	001101	101	1	00	0	00	0	0	0	1
slti	001010	110	1	00	1	00	0	0	0	1
sw	101011	000	1	00	1	00	0	1	0	0
lw	100011	000	1	00	1	00	1	0	0	1
j	000010	000	0	00	0	10	0	0	0	0
beq	000100	001	0	01	1	00	0	0	0	0
bne	000101	001	0	10	1	00	0	0	0	0
jal	000011	000	0	00	0	11	0	0	0	0

Aludecoder		
aluop	funct	alucont
000#default	/	010#add
001#branch	/	011#sub
010#R-Type	100100#and	000
010#R-Type	100101#or	001
010#R-Type	100000#add	010
010#R-Type	100010#sub	011
010#R-Type	101010#slt	100
010#R-Type	000000#sll	101
010#R-Type	000010#srl	110
010#R-Type	000011#sra	111
011#addi	/	010#add
100#andi	/	000#and
101#ori	/	001#or
011#slti	/	100#slt

2.3.寄存器文件



A1, A2：被读取数据的寄存器地址

A3：被写入数据的寄存器地址

WD3：写入的数据

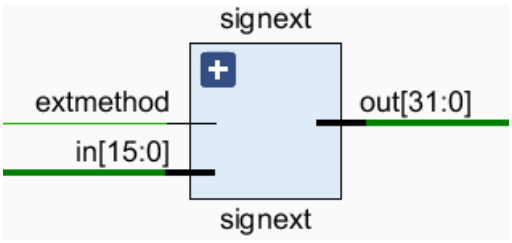
WE3：是否允许写入数据

clk：时钟

jal_in, jal_out, jal_sig(jump[0])：当指令为 jal 时将 jal_in 存入$ra 中。jal_out 始终输出$ra 的值，通往 pc 的多路选择器，当指令为 jr 时成为下一周期的 pc

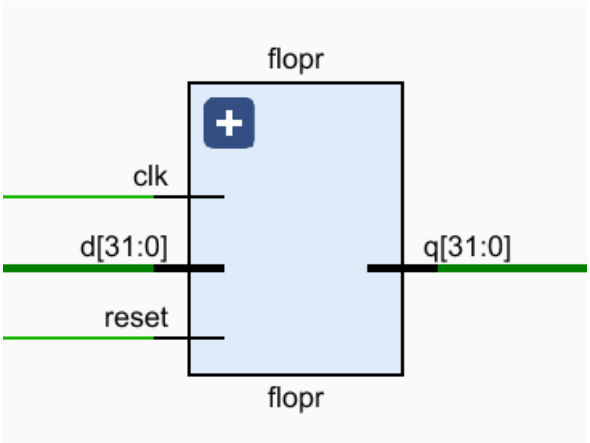
寄存器本体用数组实现

2.5.符号扩展器



将 16 位输入值扩展为 32 位输出，extmethod 决定是否按符号位扩展

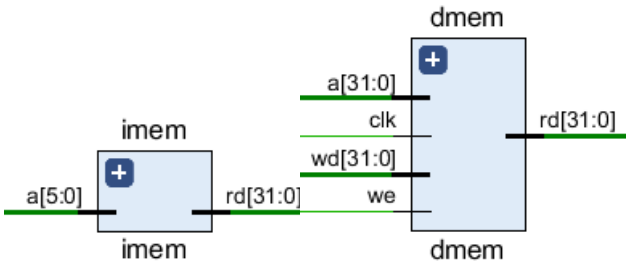
2.6.PC 寄存器



当时钟上升沿到来时更新下一条执行的指令。下一条指令由该寄存器前的三个 mux 选择，分别决定

branch(beq, bne)、jump(jump, jal)和 jr 是否跳转

2.7.存储器



a：地址

wd：写入的数据

we：是否写入数据

rd：输出数据

存储器内部使用数组实现

2.8.数据通路

将核心中的部件连接起来，并和外部相连，由于原理图过于庞大，截图后完全看不清，因此给出

systemverilog 代码如下：

```
assign op=instr[31:26];
assign funct=instr[5:0];
logic [4:0]writereg;
logic [31:0]signimm,signimmshifted,srcA,srcB,result;
logic [31:0]pcplus4,pcbranch,pcbranchsel,pcjump,pcjumpsel,pcjr,nxtpc;
assign pcjump={pcplus4[31:28], instr[25:0], 2'b00};
flopr flopr(clk,reset,nxtpc,pc);
regfile regfile(clk,reset,regwrite,instr[25:21],instr[20:16],writereg,result,pcplus4,jump[0],srcA,writedata,pc
alu alu(alucont,srcA,srcB,instr[10:6],aluresult,zero);
signext signext(instr[15:0],extmethod,signimm);
shift2 shift2(signimm,signimmshifted);
adder pcplus4adder(pc,32'b100,pcplus4);
adder pcbranchadder(signimmshifted,pcplus4,pcbranch);
mux21 alusrcmux(writedata,signimm,alusrc,srcB);
mux21 resultmux(aluresult,readdata,memtoreg,result);
mux21 #(5) writeregmux(instr[20:16],instr[15:11],regdst,writereg);
mux21 pcbranchmux(pcplus4,pcbranch,pcsrc,pcbranchsel);
mux21 pcjumpmux(pcbranchsel,pcjump,jump[1],pcjumpsel);
mux21 pcjrmux(pcjumpsel,pcjr,jr,nxtpc);
```

3.参考文献

David Money Harris, Sarah L. Harris　Digital Design and Computer Architecture

https://github.com/SunflowerAries/ICS-Spring20-Fudan

https://github.com/SunflowerAries/MIPS