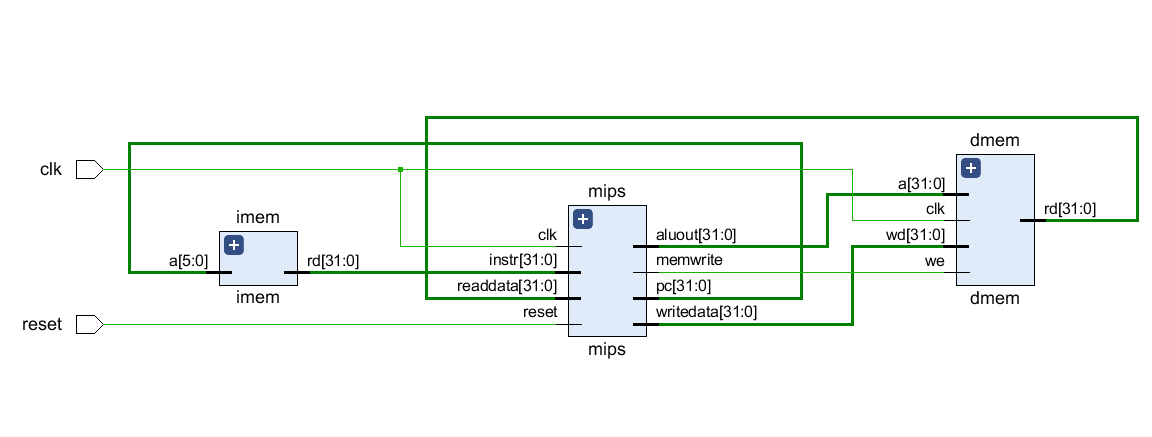
**作业1——单周期MIPS处理器**

**实验报告**

贾子安18307130017

**1.总体结构**



imem：指令存储器

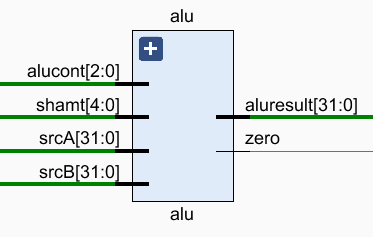
dmem：数据存储器

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**

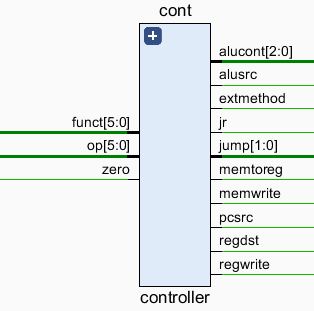
2.1.ALU



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

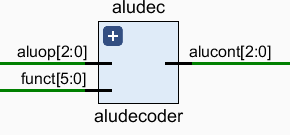
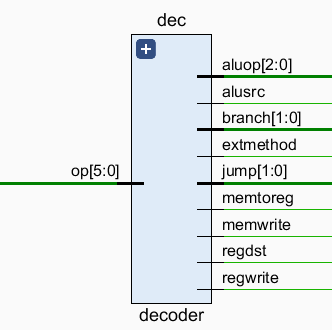
memwrite：是否写dmem

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:



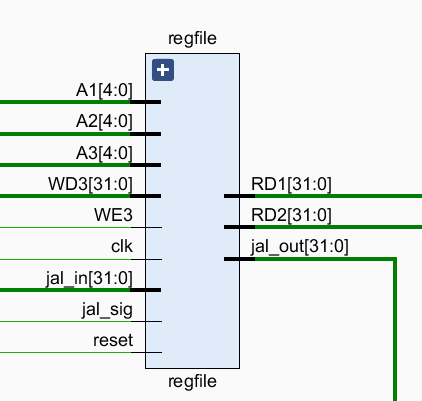
maindecoder分析输入的op，给出除alocont以外的控制信号并输出aluop信号，而aludecoder接受aluop信号并结合输入的funct，输出alucont控制信号给ALU

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

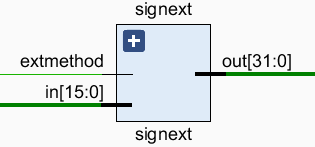
WE3：是否允许写入数据

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

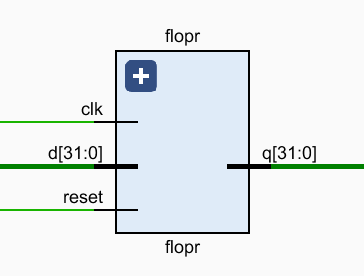
寄存器本体用数组实现

2.5.符号扩展器



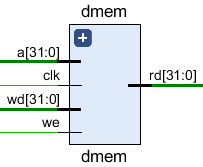
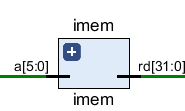
将16位输入值扩展为32位输出，extmethod决定是否按符号位扩展

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

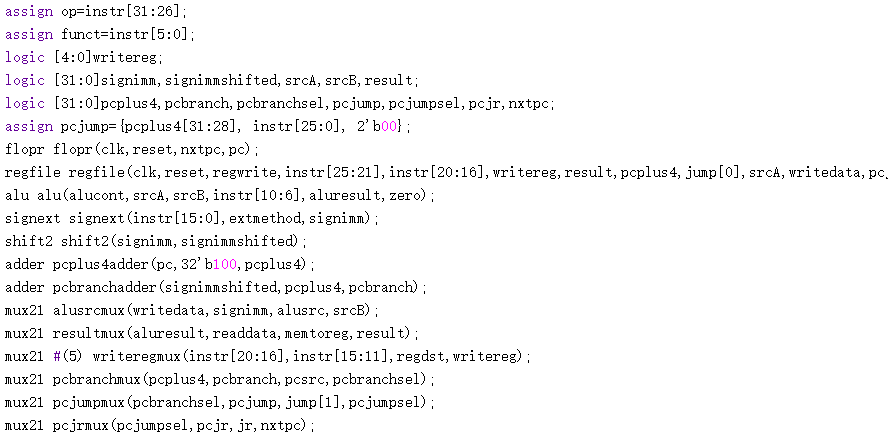
we：是否写入数据

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：



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