RISC-V设计报告

陈炜 PB16050567

IF段：

NPC_Generator：

共三个控制信号，四个输入变量以及一个输出。

Branch：BrE使能决定是否使用BrT地址作为下一个NPC，和所有branch命令相关

Jal：JalD使能决定是否使用JalTarget

Jalr:JalrE使能对应JarlTarget

上述使能有优先级，JalrE和Branch优先级相同（不可能同时出现），Jal优先级最低。

PCF是上一次的PC值。

如果上述三个使能任意一个有效，选择对应PC值输出；如果均无效，则输出PCF+4

IF段寄存器：

只要接收PC_IN并将PCF输出到NPC_Generator和nstruction Memory（嵌入ID寄存器中）即可。

Instruction Memory:

将输入的30位地址（因为地址必然是4的整倍数，所以低两位可以省略），转换成有效的11位地址，在RAM中取得对应的指令。

因为Instruction Memory采用block memory的形式，所以是同步读取的而非异步读取，只有在时钟上升沿才能得到数据，所以将此模块嵌入ID寄存器中，才能保证正确的5段流水线模式。

ID段：

ID段寄存器：

内嵌Instruction Memory，得到指令地址，将其[31:2]输入Instruction Memory。输出得到的指令即可。

ControlUnit:

组合逻辑电路，根据输入的指令段确定信号。

JalD:

根据func7，如果为1101111即为1，否则0

JalrD:

与JalD类似

RegWrite:

此信号在DataExt模块起作用，根据指令类型决定写入寄存器的数据要如何组织，是实现非对齐load的组成部分。

MemWriteD:

是实现非对齐store的组成部分。在WBSegReg（内嵌dataRam）起作用，作为WE（写使能）。采用独热码，此处只生成0001，0011或者1111。在写入的时候再根据最低2位地址进行移位即可（数据也要左移）。

LoadNpcD：

和Jal，Jalr有关，当出现这两种指令的时候，需要把PC+4写入RD寄存器，这个信号指示，是否写入，如果此信号为0，则写入AluOut的值，这个信号在WB段起作用。

RegReadD：

用于Hazard模块中的Forward功能，决定是否启动Forward1或2。

BranchTypeD、AluContrlD：

根据输入信号，输出Branch种类和Alu计算种类

AluSrc1D:

1位，决定输入ALU的是从forward1出来的数据还是PCE（Branch）

AluSrc2D：

2位，3中类型：forward2、立即数还是【24:20】（移位指令SLLI等）

ImmType:

共6种，根据func7判断，其中R-type没有立即数

Immediate Operand Unit：**

根据输入的ImmType，对31:7立即数，进行组合，符号扩展，组合方式由指令集确定。

RegisterFile:

寄存器的读写。对时钟信号取反，以便实现异步读取，减少一个周期。

JalNPC:

没有独立的单元，直接在Core(顶层模块)中加上PCD（0或者PCF）

EX段：

EX段寄存器:

无特别的功能。

BranchDecisionMaking:

根据type和两个操作数，决定是否branch

Alu：

根据传入的Type对两个操作数进行运算，包括加减，移位等。

以上两个type信号都在controlUnit生成并传递过来。

MEM段：

MEM段寄存器:

无特别的功能。

DataMemory：

被嵌入WB段寄存器中。与Instruction Ram不同的是，DataRam需要支持，LB,LH，SB,SH等操作，需要非对齐的存取。一个32位的地址，有4个数据，所以采用4位独热码来实现非对齐的存取。

```
always @ (posedge clk)
    if(wea[0] & addra_valid)
        ram_cell[addral][ 7: 0] <= dina[ 7: 0];

always @ (posedge clk)
    if(wea[1] & addra_valid)
        ram_cell[addral][15: 8] <= dina[15: 8];

always @ (posedge clk)
    if(wea[2] & addra_valid)
        ram_cell[addral][23:16] <= dina[23:16];

always @ (posedge clk)
    if(wea[3] & addra_valid)
        ram_cell[addral][31:24] <= dina[31:24];
```

如图，根据wea独热码来决定对应的数据能否被写入。结合写使能，对数据进行移位，就实现了非对齐的Store。

关于移位：首先独热码的1一定是连续的，所以只需要将数据左移到最低位与独热码的最低位对齐即可。

WB段：

WB段寄存器：

内嵌DataRam，非对齐Store的移位操作在这里进行，内嵌的原因与InstructionRam内嵌的原因类似。

DataExt：

根据RegWriteW（最早由ControlUnit生成）决定load的类型，目前的想法是，和store类似，用独热码和读出来的数据与，LB，LH分别对应0001和0011，LW对应1111。

LoadedBytesSelect指示最低2位地址，即是块内偏移地址。这个信号为几，就将Load出来的数据右移几个字节对齐即可。移位以后，再与独热码与操作即可。

以上就是我当前实现各个模块的思路。

问题回答：

\1. 为什么将DataMemory和InstructionMemory嵌入在段寄存器中?

因为这两者是block memory，只能在时钟上升沿读出数据，为了节省周期，所以要嵌入段寄存器中。

\2. DataMemory和InstructionMemory输入地址是字（32bit）地址，如何将访存地址转化为字地址输入进去?

PCF本就是32位，不过只传入前30位，instruction memory不支持非对齐的读取，所以后两位必须为0。

访问DataMemory的指令只有store和load，这两种指令的地址都是经过ALU计算得到的32位数据，后两位置为0即可。

\3. 如何实现DataMemory的非字对齐的Load?

我在DataExt中阐述了我的想法，根据RegWriteW确定独热码是0001，0011还是1111。

根据LoadedBytesSelect决定将数据右移几位，最后将独热码与右移后的数据做与操作即可。

\4. 如何实现DataMemory的非字对齐的Store?

根据独热码最低的1在哪一位对数据进行左移，然后将独热码与移位后的数据做与操作

\5. 为什么RegFile的时钟要取反?

为了能在一个周期内完成读取寄存器的操作，所以要在下一次时钟上升沿之前将数据输出。

\6. NPC_Generator中对于不同跳转target的选择有没有优先级?

有优先级，Jal优先级最低，因为要实现最早的跳转

\7. ALU模块中，默认wire变量是有符号数还是无符号数?

无符号，因为溢出的时候会直接舍弃高位。

\8. AluSrc1E执行哪些指令时等于1’b1?

Branch指令，将地址输入ALU进行运算

\9. AluSrc2E执行哪些指令时等于2‘b01?

移位指令。

\10. 哪条指令执行过程中会使得LoadNpcD==1?

Jal和Jalr

\11. DataExt模块中，LoadedBytesSelect的意义是什么?

意义是最后两位偏移地址，在非对齐访问的时候，决定到底要取哪一个字节或者字。

\12. Harzard模块中，有哪几类冲突需要插入气泡?

写后读，写入寄存器的数据不是直接来自ALU而是来自LOAD操作。对IF和ID都插入气泡

\13. Harzard模块中采用默认不跳转的策略，遇到branch指令时，如何控制flush和stall信号?

因为branch信号的触发是在EX级的，当BrE指令被设为1的时候，EX寄存器内是branch指令的有关数据和信号，IF和ID内都是不应该继续执行的指令，对ID和EX段寄存器输入Flush即可。下一个时钟上升沿到来的时候，这两个段寄存器都会被置为0，而MEM段寄存器会继续执行Branch指令，新的地址存入PC_In中而不是IF中。

\14. Harzard模块中，RegReadE信号有什么用?

通过RegReadE信号可以判断是不是真的冲突了，免得进行了错误的forward，把真正需要的数据冲掉了。

\15. 0号寄存器值始终为0，是否会对forward的处理产生影响?

如果前一个操作是对0号寄存器的写入，当前动作是读取0号寄存器，forward会把写入值传入ALU而不是0。所以rd为0号寄存器的操作应该是禁止的。