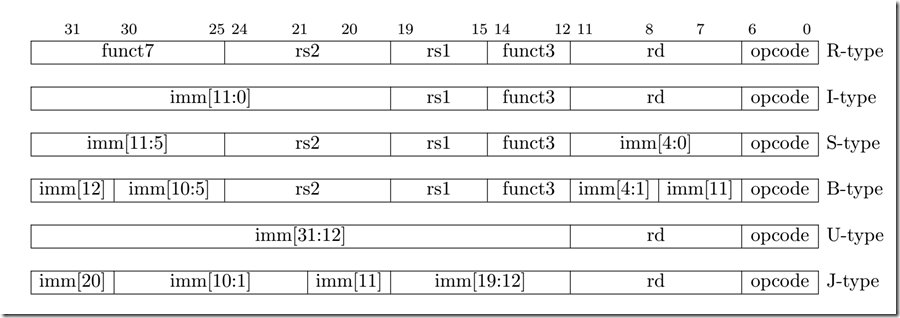
**RISC-V 流水线CPU设计报告**

**PB16060130 顾健鑫**

1. 指令结构

RV32I指令格式包括以下6种，每种指令格式都是固定的32位指令，所以指令在内存中必须4字节对齐，否则将触发异常。其中rd表示目的寄存器，rs1是源操作数寄存器1，rs2是源操作数寄存器2。



1. 数据通路设计

NPC Generator设计

NPC Generator每个周期通过输入的的PCF对其加4得到PC\_In作为NPC输出。

对于三个不同类型跳转的使能信号，当信号输入为一时，将对应的PC输入作为NPC从PC\_in输出。考虑到跳转优先级的情况，流水段深的指令优先级更高。根据数据通路，可得到如下表格作为NPC的生成原则。



Instruction Memory设计

根据提供的文件框架，指令Memory由于调用的为同步读的Memory模块，因此为了使得仅用一个周期就讲数据传输到Ex段，需要将Memory模块嵌入到ID段寄存器中，段寄存器使用组合逻辑，则可以实现一个周期的Memory读写任务。

Control Unit设计

此模块整体为组合逻辑电路，根据输入的指令的Op段和funct3段来决定最终的信号输出结果。具体取值可参考尾页的表格截图。

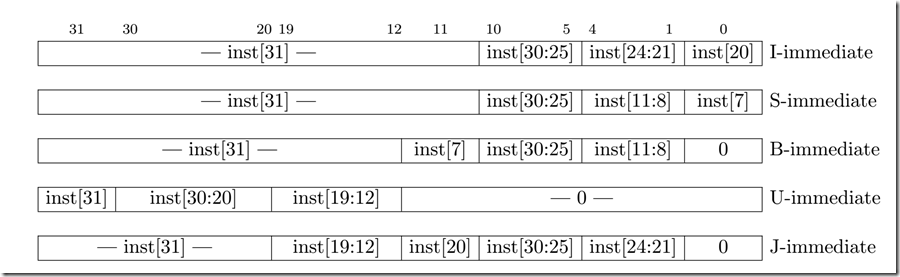
Register File设计

寄存器文件的时钟信号使用反转信号，同时采用下降沿触发进行写操作。写操作为时钟同步写，读操作则为异步读。整体为32bit\*32个寄存器。使用已有的模块，不做修改。

Immediate Operand Unit设计

此模块整体为组合逻辑电路，利用在同一个流水段Control Unit的ImmTypeD输出信号作为标志，生成不同类型的立即数用于后续的计算。

不同的指令类型需求的类型不同，在要求实现的指令中，需要立即数的指令具体的类型可参考尾页的表格的ImmType对应列取值。其中在要求的是有符号立即数时，最高位的inst[31]即为对应的符号扩展位。



ALU设计

此模块依旧为纯组合逻辑电路，根据段寄存器中的ALUControl信号决定运算操作，来完成要求的计算并输出。基本语句结构用case语句实现即可。

Branch Decision Making设计

采用组合逻辑电路，依据输入的BranchTypeE对输入的r1和r2的值进行不同的比较操作，输出是否分支成立以进行跳转工作。因为统一采取预测分支失败的方式，故当分支预测为成功时，需要对IF段和ID段进行flush操作排空流水段以保证程序的正确执行。

Ex和Mem段寄存器设计

Ex和Mem段寄存器均采用提供好的源码模版，其需求均一致，即在stall信号为0时（即en为1），flush为0时正常工作，flush为1则全部置0；当stall为1时不做任何操作，即保持上一个周期的值即可。

WB段寄存器设计

类似于段寄存器ID的做法，为了使得能够一个周期获取数据并传入WB段，此处仍旧要讲同步读写的Data Memory模块嵌入到WB段寄存器中，同时寄存器内全部使用组合逻辑而非时序逻辑。在从Memory中获取数据后有Raw变量负责导出，再根据stall和flush信号做与ID段寄存器同样的赋值操作。对于Memory模块的WB输入，单纯的MemWriteM[3:0]无法达到要求。对于sb，sh指令，要获得具体的写入地址编码，还需将其做左移位操作，移位量为AluOutM的最后两位的大小。同时AluOutM的[31:3]作为字对齐地址输入Memory模块。

Data Ext设计

由于Mem段的地址都是字对齐的，因此读出来的数据总是四个字节为一组的32位数据对于RegWriteW的取值为非LW的情形，分情况讨论如下：

1. RegWriteW == LB或LBU；此时为load一个有（无）符号的8bit数据到指定的32bit寄存器中。根据ALU计算结果的最后两位（LoadBytesSelect），将对应的字节(3，2，1，0)赋值为OUT的低8位在进行对应的有符号或者无符号扩展。
2. RegWriteW = LH或LHU；此时为load一个有（无）符号的16bit数据到指定的32bit寄存器中。同样根据ALU计算结果的最后两位，将对应的两个连续字节的数据赋值为OUT的低8为并进行有符号或者无符号扩展。对于LH的情况，有可能出现非半字对齐的情况及最后两位是11或者01，由于本次实验为考虑异常处理等信息的处理。在此默认输入的指令中不会有类似的非法指令存在。

Harzard Unit设计

该模块用于探测数据冒险并做forward和stall流水线处理。

对于旁路转发，整体分为两种情况讨论：

当RegRead不为0时，需要检查数据相关。

1. 对于RegRead为1的位所对应的寄存器，与RdM和RdW做比较（不插入气泡）

* 当与RdM或RdW相同时，则此时有数据相关的寄存器的正确值已经得出，即为ALU的输出，故将此寄存器对应的数据通路上的ForwardXE设为10或01。
* 当与RdM和RdW均相同时，此时逻辑上正确的结果应该处于Mem段，故相应的Rs寄存器的ForwardingXE应设为10.

1. 对于需插入气泡的情形，仅有一种，即load指令+用到load结果的指令。根据流水线的信号结构，此种情形在ID段和EX段进行检测和气泡处理。当Rs1D或Rs2D与RdE相同时，且MemToRegE为1（即load指令在Ex段）。此时需要stall流水线的IF和ID段一周期。之后的数据转发可归结为上述情况的第二类。

对于分支指令的处理设计如下

1. 当BranchE或JalrE为1时，即分支成功，由于统一采用预测失败的策略，因此要flush流水线的ID和IF段。
2. 当JalD为1且BranchE和JalrE均为0时，根据优先级判断，JalD分支成功，此时仅flush流水线的IF段即可。

问题回答：

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

因为Block Memory IP核和段寄存器都是同步读写，因此如果不嵌入段寄存器中会导致当读取Memory时会多出一个时钟周期。

1. 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？

字宽设置为32bit，输入的是字节地址。由于默认为对齐输入，因此直接抹去后两位即可（设为0）。

1. 如何实现DataMemory的非字对齐的Load？

已知从Mem中读取出的是一个32位长的字数据。根据load的类型，利用Data Ext模块，选择出需要写入的对应的bytes并进行符号扩展。

1. 如何实现DataMemory的非字对齐的Store？

IP核支持按字节写入，通过设置四位宽的写使能信号来写入一个四字节数据的的不同位置。

1. 为什么RegFile的时钟要取反？

在写回的时候，如果不取反，则可能会需要6周期才能完成load操作。除此之外可能还会导致额外的数据冒险。

1. NPC\_Generator中对于不同跳转target的选择有没有优先级？

当不同指令在不同的段出现跳转需求时，执行越靠后的指令的优先级越高。即流水深度越深的指令，其分支优先级越高。

1. ALU模块中，默认wire变量是有符号数还是无符号数？

可以直接指定为有符号数还是无符号数。

1. AluSrc1E执行哪些指令时等于1’b1？

执行AUIPC时ALUSrc1E的值为1’b1。

1. AluSrc2E执行哪些指令时等于2’b01？

使用立即数的移位类指令（SLLI，SRLI，SRAI）要用到ALUSrc2E为2’b01的情况。

1. 哪条指令执行过程中会使得LoadNpcD==1？

JALR和JAL，会将NPC的值写入rd寄存器。

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

指示对于非字对齐的load操作时，具体需要的是32bit数据中的哪些字节。

1. Harzard模块中，有哪几类冲突需要插入气泡？

load指令+有指令需要用到load的结果时需要加入气泡。具体对应流水线的结构即为在ID段时如果EX端的RdE与ID段的Rs1D和Rs2D有冲突，且MemToRegE为1，则stall流水线的IF和ID段。

1. 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？

对于branch采用默认不跳转的策略，当实际要跳转时，才对前两段实行flush操作。

1. Harzard模块中，RegReadE信号有什么用？

判断当指令处于ex段时，rs1和rs2寄存器是否被用到，没用到的话就不用做forwarding的判断。

1. 0号寄存器值始终为0，是否会对forward的处理产生影响？

由于0号寄存器始终为0，需要判断forward，否则可能会写入R0，会产生影响。因为向0号寄存器内写入数据的写入根本不会发生，但是如果下一条指令使用了0号寄存器的值，那么就会使流水线停顿，产生负面影响。因此要使写入R0的操作失效。