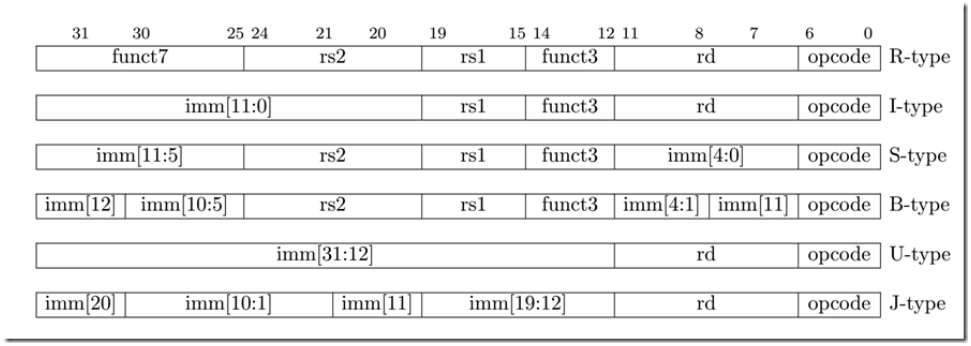
RISC-V 流水线 CPU 设计报告

PB16060130　　顾健鑫

一、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。



二、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。

BrE	JalrE	JalD	PC_In
0	0	0	PCF+4
1	*	*	BrT
*	1	*	JalrT
0	0	1	JalT

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。

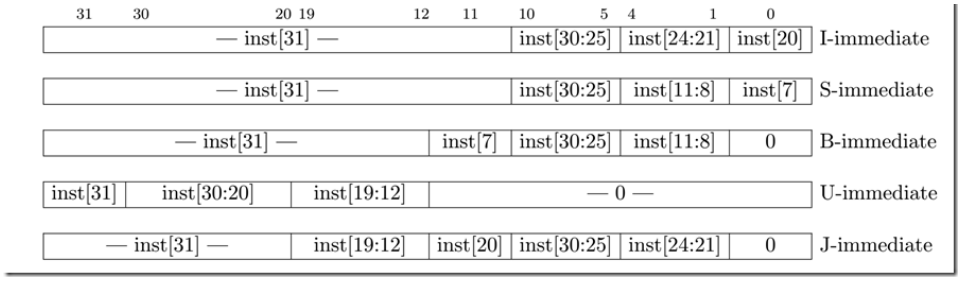
Register File 设计

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：

a. 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。
b. 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.
2. 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。
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？
   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）。
3. 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。
4. 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。
5. 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。
6. 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。
7. 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。
8. AluSrc1E 执行哪些指令时等于 1’b1？
   执行 AUIPC 时 ALUSrc1E 的值为 1’b1。
9. 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。
10. 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。
11. 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。
12. 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。
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？
    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。
14. 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。
15. 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。

	JalD	JalrD	RegReadD[1:0]	MemToRegD	RegWrite[2:0]	MemWrite[3:0]	LoadNpcD	AluControlD[3:0]	BranchType[2:0]	Alu2srcD[1:0]	Alu1srcD	ImmTypeD
LB	0	0	10	1	LB	0000	0	ADD	NOBRANCH	10	0	ITYPE
LH	0	0	10	1	LH	0000	0	ADD	NOBRANCH	10	0	ITYPE
LW	0	0	10	1	LW	0000	0	ADD	NOBRANCH	10	0	ITYPE
LBU	0	0	10	1	LBU	0000	0	ADD	NOBRANCH	10	0	ITYPE
LHU	0	0	10	1	LHU	0000	0	ADD	NOBRANCH	10	0	ITYPE
SB	0	0	11	0	NOREGWRITE	0001	0	ADD	NOBRANCH	10	0	STYPE
SH	0	0	11	0	NOREGWRITE	0011	0	ADD	NOBRANCH	10	0	STYPE
SW	0	0	11	0	NOREGWRITE	1111	0	ADD	NOBRANCH	10	0	STYPE
ADD	0	0	11	0	LW	0000	0	ADD	NOBRANCH	00	0	/
SUB	0	0	11	0	LW	0000	0	SUB	NOBRANCH	00	0	/
OR	0	0	11	0	LW	0000	0	OR	NOBRANCH	00	0	/
XOR	0	0	11	0	LW	0000	0	XOR	NOBRANCH	00	0	/
AND	0	0	11	0	LW	0000	0	AND	NOBRANCH	00	0	/
SLL	0	0	11	0	LW	0000	0	SLL	NOBRANCH	00	0	/
SRL	0	0	11	0	LW	0000	0	SRL	NOBRANCH	00	0	/
SRA	0	0	11	0	LW	0000	0	SRA	NOBRANCH	00	0	/
SLT	0	0	11	0	LW	0000	0	SLT	NOBRANCH	00	0	/
SLTU	0	0	11	0	LW	0000	0	SLTU	NOBRANCH	00	0	/
ADDI	0	0	10	0	LW	0000	0	ADD	NOBRANCH	10	0	ITYPE
ORI	0	0	10	0	LW	0000	0	OR	NOBRANCH	10	0	ITYPE
XORI	0	0	10	0	LW	0000	0	XOR	NOBRANCH	10	0	ITYPE
ANDI	0	0	10	0	LW	0000	0	AND	NOBRANCH	10	0	ITYPE
SLLI	0	0	10	0	LW	0000	0	SLL	NOBRANCH	01	0	/
SRLI	0	0	10	0	LW	0000	0	SRL	NOBRANCH	01	0	/
SRAI	0	0	10	0	LW	0000	0	SRA	NOBRANCH	01	0	/
SLTI	0	0	10	0	LW	0000	0	SLT	NOBRANCH	10	0	ITYPE
SLTIU	0	0	10	0	LW	0000	0	SLTU	NOBRANCH	10	0	ITYPE
LUI	0	0	00	0	LW	0000	0	ADD	NOBRANCH	10	0	UTYPE
AUIPC	0	0	00	0	LW	0000	0	ADD	NOBRANCH	10	1	UTYPE
BEQ	0	0	11	0	NOREGWRITE	0000	0	/	BEQ	00	0	BTYPE

BNE	0	0	11	0	NOREGWRITE	0000	0	/	BNE	00	0	BTYPE
BLT	0	0	11	0	NOREGWRITE	0000	0	/	BLT	00	0	BTYPE
BGE	0	0	11	0	NOREGWRITE	0000	0	/	BGE	00	0	BTYPE
BLTU	0	0	11	0	NOREGWRITE	0000	0	/	BLTU	00	0	BTYPE
BGEU	0	0	11	0	NOREGWRITE	0000	0	/	BGEU	00	0	BTYPE

JAL	1	0	00	0	LW	0000	1	ADD	NOBRANCH	/	/	/
JALR	0	1	10	0	LW	0000	1	ADD	NOBRANCH	10	0	ITYPE

NOP	0	0	00	0	NOREGWRITE	0000	0	ADD	NOBRANCH	00	0	/