一、 设计说明

1. 处理器应MIPS-C3指令集。

a) MIPS-C3={LB、LBU、LH、LHU、LW、SB、SH、SW、ADD、ADDU、SUB、SUBU、MULT、MULTU、DIV、DIVU、SLL、SRL、SRA、SLLV、SRLV、SRAV、AND、OR、XOR、NOR、ADDI、ADDIU、ANDI、ORI、XORI、LUI、SLT、SLTI、SLTIU、SLTU、BEQ、BNE、BLEZ、BGTZ、BLTZ、BGEZ、J、JAL、JALR、JR、MFHI、MFLO、MTHI、MTLO}。

b) 所有运算类指令均可以不支持溢出。

2. 处理器为流水线设计。

二、 设计要求

3. 集中式控制器或分布式控制器架构均可以。

4. 流水线顶层架构。如果你才有了集中式控制器，则我们建议参考《数字设计和计算机体系结构》中的图7-58作为流水线的顶层视图。

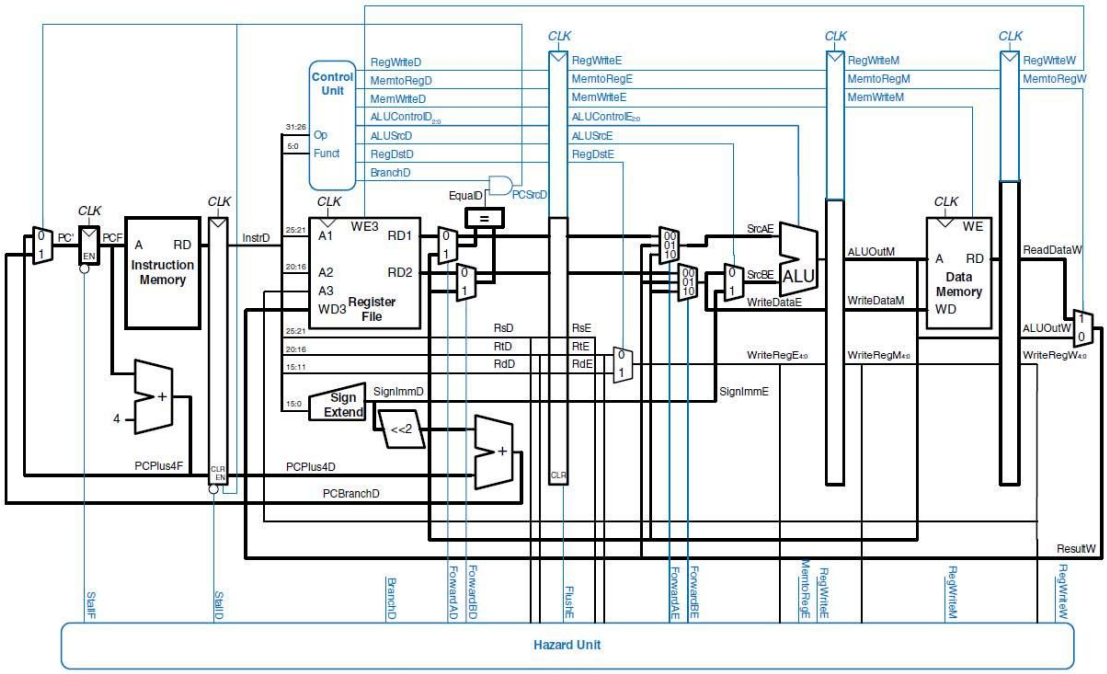

Figure 7.58 Pipelined processor with full hazard handling

a) 该图仅在宏观的结构层面作为参考，并不能支持本project的全部指令。

b) 建议采用3控制器架构，即将图中的Hazard Unit进一步拆分为暂停控制器和转发控制器，即：

1）主控制器：功能如同单周期设计，指令译码、功能部件控制、MUX(不包括转发MUX)控制等。

2）暂停控制器：根据相关检测，只处理暂停IF/ID的指令。

3）转发控制器：根据相关检测，只处理转发。

5. ALU。ALU完成所有的加、减、与或非、移位运算。本project规范ALU的设计接口。ALU接口必须如下，不得修改！

信号名	方向	描述
A[31:0]	I	第1个运算数 当执行移位指令时，A[4:0]为移位位数。
B[31:0]	I	第2个运算数
Op[X:0]	I	运算类型。 具体编号可以自行定义。 X自行定义。
C[31:0]	O	ALU计算结果
Over	O	溢出 0：无溢出 1：有溢出

6. CMP。CMP用于实现b类指令的比较操作。CMP位于流水线译码/读寄存器级。本project规范CMP的设计接口。CMP接口必须如下，不得修改！

信号名	方向	描述
A[31:0]	I	第1个运算数
B[31:0]	I	第2个运算数
Op[X:0]	I	比较类型。 具体功能编码可以自行定义。 X自行定义。
Br	O	分支指令比较的结果。 0：条件不成立

		1：条件成立

7. GPR(寄存器堆)。本project规范RF的设计接口。GPR接口必须如下，不得修改！

信号名	方向	描述
A1[4:0]	I	读取的第1个寄存器编号
A2[4:0]	I	读取的第2个寄存器编号
A3[4:0]	I	写入的寄存器编号
RD1[31:0]	O	A1对应的寄存器值
RD2[31:0]	O	A2对应的寄存器值
WD[31:0]	I	写入的数据
We	I	写使能
Clk，Rst	I	时钟，复位

a) GPR必须支持内部转发，即当写入寄存器编号与读出寄存器编号相同时，输出值就是待写入寄存器值。

8. 乘除法部件。为了支持mult、multu、div、divu、mfhi、mflo、mthi及mtlo这些乘除法相关指令，需要设计独立的乘/除功能部件。该部件位于在流水线的EX阶段，如图1所示。

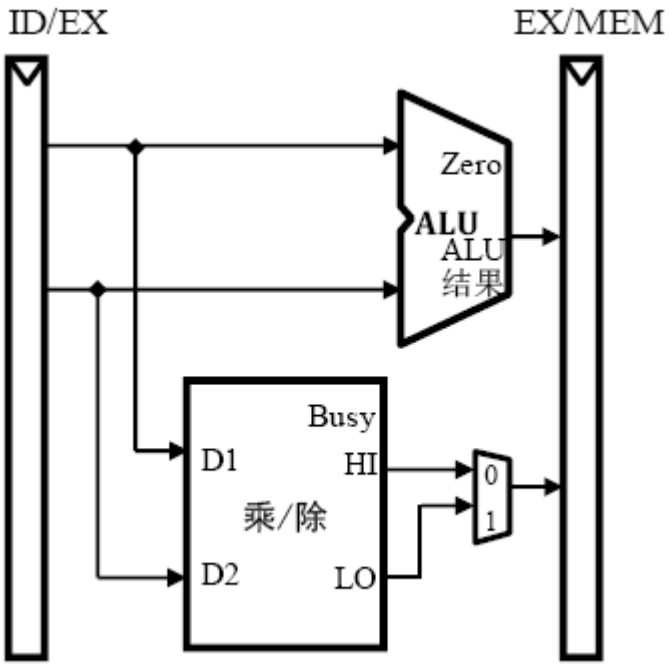

图1 流水线EX阶段的乘/除部件

a) 为了降低实现难度，乘/除法运算的实现可以使用VerilogHDL的内置运算符，而不需要从门级开始建模乘法或除法的硬件算法。

b) 乘除法运算延迟。我们假定乘/除部件的执行乘法的时间为5个cycle(包含写入内部的HI和LO寄存器)，执行除法的时间为10个cycle。你在乘/除部件内部必须模拟这个延迟，即通过Busy标志来反映这个延迟。图2给出了乘法的计算延迟。

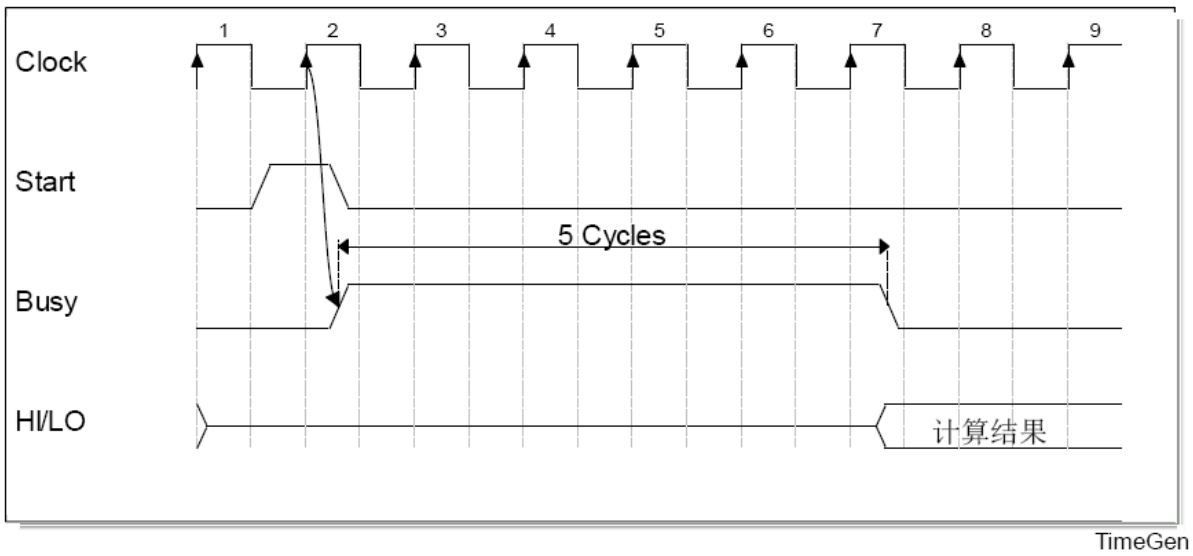

图2 乘法执行延迟(5 cycles)

c) 乘/除部件与ALU可以并行工作，这意味着你可以在mult/multu/div/divu指令后面放入若干无关指令，从而充分利用乘/除部件的执行延迟。这点非常类似于编译器针对分支指令的延迟槽技术和针对lw指令的指令调度优化。

d) 乘/除部件接口必须如下。不得修改接口定义！

信号名	方向	描述
D1[31:0]	I	①执行乘除法指令时的第1个操作数 ②mthi/mtlo 指令的写入数据
D2[31:0]	I	执行乘除法指令时的第2个操作数
HiLo	I	待写入的寄存器 0：LO寄存器 1：HI寄存器
Op[1:0]	I	运算种类 00：无符号乘法 01：符号乘法 10：无符号除法 11：符号除法
Start	I	运算启动。该信号只能有效1个cycle 1：启动
We	I	HI或LO寄存器的写使能
Busy	O	乘除单元忙标志。 0：乘除单元未执行运算 1：乘除单元正在执行运算
HI[31:0]	O	HI寄存器的输出值
LO[31:0]	O	LO寄存器的输出值
Clk，Rst	I	时钟，复位

e) 自Start信号有效后的第1个clock上升沿开始，乘除部件开始执行运算，同时Busy置位为1。

f) 在运算结果保存到HI和LO后，Busy位清除为0。

g) 当Busy为1时，mfhi、mflo、mthi、mtlo、mult、multu、div、divu均被阻塞，

即被阻塞在IF/ID。

h) 数据写入HI或LO，均只需1个cycle。

9. 指令存储器(IM，instruction memory)和数据存储器(DM，data memory)：

a) IM：容量为8KB(32bit/word×2Kword)。

b) DM：容量为8KB(32bit/word×2Kword)。

10. 为了支持lb、lbu、lh、lhu、sb、sh指令，DM模块的接口必须如下，不得修改接口！

信号名	方向	描述
A[X:2]	I	DM的地址。位数自定义。
BE[3:0]	I	4位字节使能，分别对应4个字节。 BE[x]为1：对应的WD[31:0]的第X字节可以被写入 BE[x]为0：对应的WD[31:0]的第X字节禁止被写入
WD[31:0]	I	32位写入数据
RD[31:0]	O	32位输出数据
We	I	写使能
Clk	I	时钟

11. 流水线设计指导思、延迟槽、数据转发来源、PC复位初始值(0x0000_3000)等要求与前个project要求。

三、 命名规范化

12. 参考前个project。

四、 测设要求

13. 功能与性能测试。由于开发的指令较多，因此测试用例建议分为功能测试和性能测试两部分为好。

a) 功能测试：主要用于测试单条指令的正确性，注意测试指令的数据边界以数据性质发生变化的情况。例如mult指令测试，应该至少测试正×正、正×负、负×正、负×负、数×0、0×数等情况。

b) 性能测试：主要针对流水线中的各类冒险。

14. 乘除法引发的相关测试。由于乘除部件加入，因此数据相关发生了一些变化。此时控制系统应该根据Busy标志来判断是否允许相关指令继续执行还是被阻塞。

15. 相关性测试。参见前个project。需要说明的是：

a) 当你完成了指令的功能测试正确后，在进行相关测试时，你不需要进行指令的全覆盖组合，而是应该按大类进行组合。

b) 这样虽然不是非常严格的测试方法，但这是高效的测试方法。

16. 本project不提供基准测试程序。

17. 详细说明你的相关性测试程序原理及测试结果。【WORD】

c) 应明确说明相关性测试程序的测试期望，即应该得到怎样的运行结果。

d) 每条汇编指令都应该有注释。

五、 成绩及实验测试要求

18. 参见前个project。

六、 **Project**提交

19. 参见前个project。

七、 开发与调试技巧

20. BE[3:0]是字节使能，分别与WD[31:24]、WD[23:16]、WD[15:8]及WD[7:0]对应。当We有效时，对于Addr寻址的那个word来说，BE[3]为1则WD[31:24]被写入byte3，类似的BE[2]对应WD[23:16]和byte2，依此类推。

a) BE[3:0]主要用于支持sb、sh、sw这3条指令。当处理器执行sb、sh、sw指令时，通过对EX/MEM保存的ALU计算结果的位1和位0(保存的是ALU计算的32位地址)的解读后产生相应的BE[3:0]，就可以“通知”DM该写入哪些字节。

b) sw指令：GPR[rt]写入对应的字。

地址[1:0]	BE[3:0]	用途
XX	1111	WD[31:24]写入byte3 WD[23:16]写入byte2

		WD[15:8]写入byte1 WD[7:0]写入byte0

c) sh指令：GPR[rt]15:0写入对应的半字。

地址[1:0]	BE[3:0]	用途
0X	0011	WD[15:8]写入byte1 WD[7:0]写入byte0
1X	1100	WD[15:8]写入byte3 WD[7:0]写入byte2

d) sb指令：GPR[rt]7:0写入对应的字节。

地址[1:0]	BE[3:0]	用途
00	0001	WD[7:0]写入byte0
01	0010	WD[7:0]写入byte1
10	0100	WD[7:0]写入byte2
11	1000	WD[7:0]写入byte3

e) 图3给出了增加BE扩展的数据通路局部参考设计。显然，BE扩展功能部件还需要有来自控制器的控制信号。注意：由于DM容量有限，因此ALU计算出来的32位地址没有必要也不可能都用上。

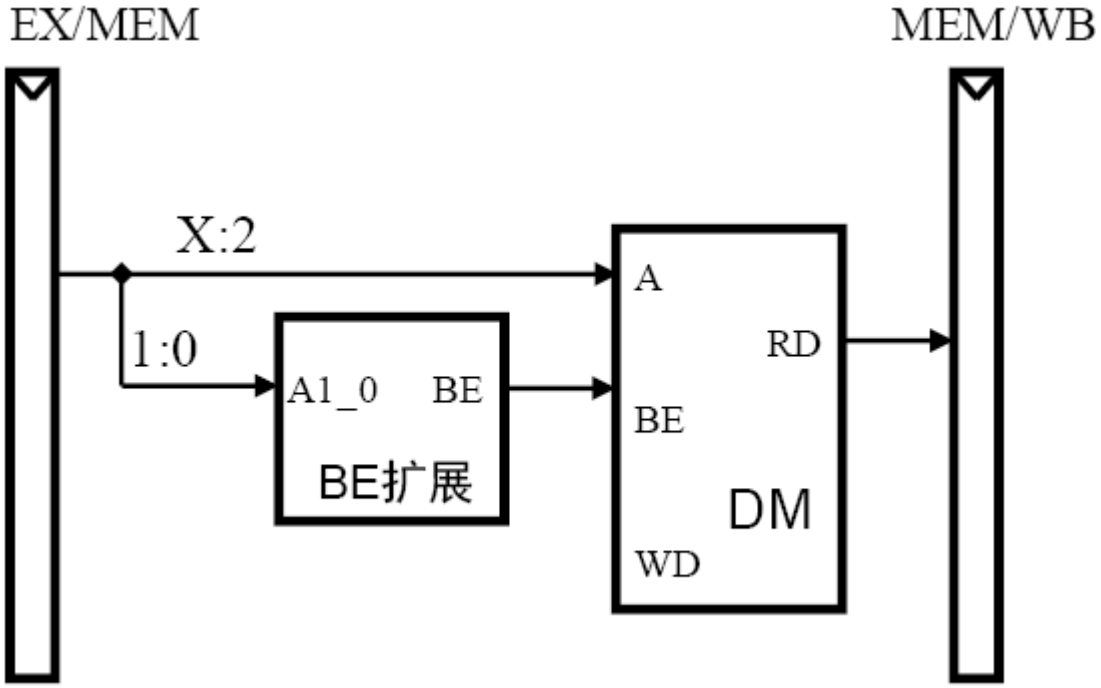

图3 BE扩展

21. 对于lb、lbu、lh、lhu来说，你必须增加一个数据扩展模块。这个模块把从DM读出的数据做符号或无符号扩展。

a) 以lb为例，数据扩展模块输入数据寄存器的32位数据，根据ALU计算出来的地址最低2位从中取出特定的字节，并以该字节的最高位为符号位做符号扩展。

b) 数据扩展模块的接口定义如下，不得修改：

信号名	方向	描述
A[1:0]	I	最低2位地址。
Din[31:0]	I	输入32位数据
Op[2:0]	I	数据扩展控制码。 000：无扩展 001： 无符号字节数据扩展 010：符号字节数据扩展 011：无符号半字数据扩展 100：符号半字数据扩展
DOut[31:0]	O	扩展后的32位数据

c) 数据扩展模块应在MEM/WB之后，而不能在DM之后。

22. 宏定义。宏定义不仅有助于提高可读性，而且不易出错。对于下列表达式，在表述方面显然前者优于后者，而在门电路实现方面两者则是等价的。

```
assign beq = (op == `BEQ) ;
assign beq = !op[5] & !op[4] & !op[3] & op[2] & !op[1] & !op[0] ;
```

23. 其他部分参见前个project。

八、 技术解释

24. 怎么实现乘除法？

a) 《计算机组成与设计~~硬件/软件接口》(第4版)的第3章讲述了硬件完成运算的方法。

25. 为什么乘除单元不分离？

a) 见《计算机组成与设计~~硬件/软件接口》(第4版)的3.4.4。

26. 为什么数据扩展模块不放在M阶段？

a) 如果以《计算机组成与设计~~硬件/软件接口》(第4版)的图4-26为设计依据，可以看出DM延迟最大。因此如果放置在Mem阶段，则Mem阶段的延迟将进一步增加。而放置在WB阶段，则由于RF延迟小，因此可能仍然小于放置在Mem阶段后的延迟，这可以使得时钟频率不会因此而降低。

27. bgez等指令的rt域的编码是什么意思？

a) 这个编码就如同R型指令的function域一样，用于解释具体是哪条分支指令。

b) 个人认为MIPS在b类指令编码方案上似乎不是很理想，搞出来2类格式，其实还是有些复杂的。也许不如把所有b类都用同一个opcode来引导比较好？

28. CMP模块的op与指令的opcode什么关系？

a) 没有直接关系！你应该用指令的opcode(对于bgez等指令还需要增加funct)来生成op。当然，你也可以把opcode/funct直接输入到CMP模块，然后在CMP模块完成译码。