

## VerilogHDL 开发流水线处理器 (2)

V1.0@2014.11.29

高小鹏

### 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) 所有会产生溢出的运算类指令都必须支持溢出，即必须当溢出发生时，ALU 必须输出相应的溢出信号。

2. 处理器为流水线设计。

### 二、 设计要求

3. 流水线顶层架构。你设计的流水线的顶层视图，我们建议采用《数字设计和计算机体系结构》中的图 7-58。

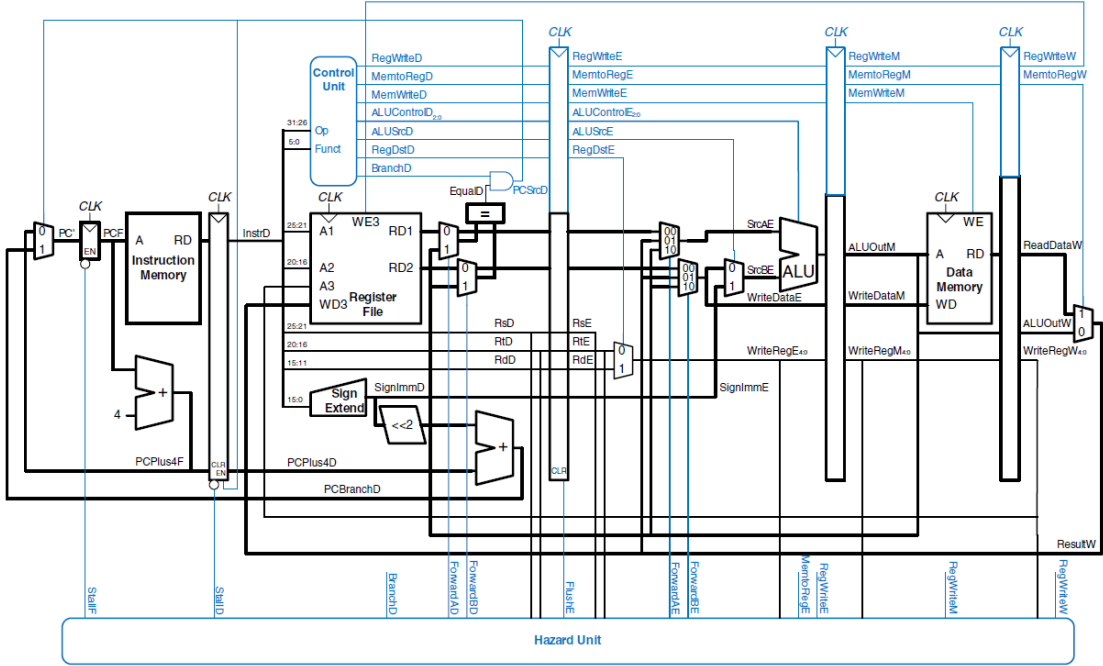


Figure 7.58 Pipelined processor with full hazard handling

- a) 该图仅仅是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。这样：
- 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。
4. 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
B[31:0]	I	第 2 个运算数 当执行移位指令时，B[4:0]为移位位数。
Op[3:0]	I	运算类型。 具体编号可以自行定义。
C[31:0]	O	ALU 计算结果
Great	O	A 大于 B 结果标志 0: A 不大于 B 1: A 大于 B
Less	O	A 小于 B 结果标志 0: A 不小于 B

		1: A 小于 B
Zero	O	A 等于 B 结果标志 0: A 不等于 B 1: A 等于 B
Over	O	溢出 0: 无溢出 1: 有溢出

5. 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]	O	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。
6. 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。

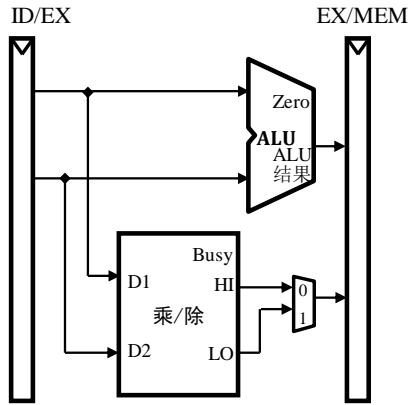


图 1 流水线 EX 阶段的乘/除部件

- 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。

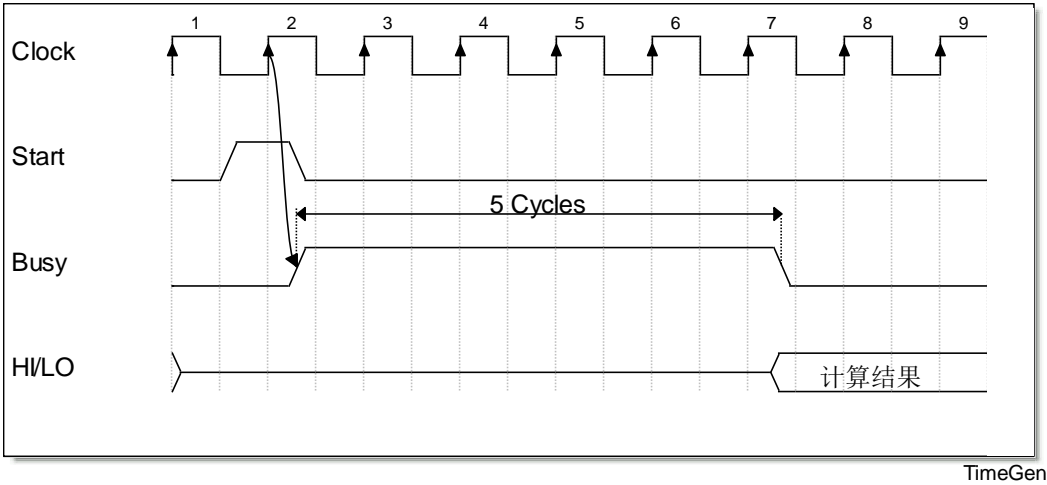


图 2 乘法执行延迟(5 cycles)

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	I	运算种类 0: 乘法 1: 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。
7. 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)。
8. 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

9. 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

10. 参考前个 project。

### 四、测设要求

11. 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。
12. 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。

13. 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。

14. 本 project 不提供基准测试程序。

15. 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

16. 参见前个 project。

## 六、 Project 提交

17. 参见前个 project。

## 七、 开发与调试技巧

18. **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]<sub>15:0</sub>**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]<sub>7:0</sub> 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。

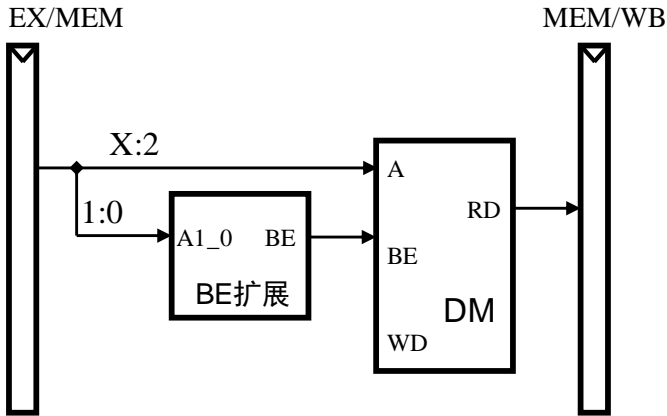


图 3 BE 扩展

19. 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[1:0]	I	数据扩展控制码。 00：符号字节数据扩展 01：无符号字节数据扩展 10：符号半字数据扩展

		11: 无符号半字数据扩展
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。

20. 其他部分参见前个 project。