细节)。我们看到一组和前面一样的硬件单元，但是现在线路看得更清楚了。这幅图以及其他的硬件图都使用的是下面的画图惯例。

- 白色方框表示时钟寄存器。程序计数器 PC 是 SEQ 中唯一的时钟寄存器。
- 浅蓝色方框表示硬件单元。这包括内存、ALU 等等。在我们所有的处理器实现中，都会使用这一组基本的单元。我们把这些单元当作“黑盒子”，不关心它们的细节设计。
- 控制逻辑块用灰色圆角矩形表示。这些块用来从一组信号源中进行选择，或者用来计算一些布尔函数。我们会非常详细地分析这些块，包括给出 HCL 描述。
- 线路的名字在白色圆圈中说明。它们只是线路的标识，而不是什么硬件单元。
- 宽度为字长的数据连接用中等粗度的线表示。每条这样的线实际上都代表一簇 64 根线，并列地连在一起，将一个字从硬件的一个部分传送到另一部分。
- 宽度为字节或更窄的数据连接用细线表示。根据线上要携带的值的类型，每条这样的线实际上都代表一簇 4 根或 8 根线。
- 单个位的连接用虚线来表示。这代表芯片上单元与块之间传递的控制值。

图 4-18～图 4-21 中所有的计算都有这样的性质，每一行都代表某个值的计算(如 valP)，或者激活某个硬件单元(如内存)。图 4-24 的第二栏列出了这些计算和动作。除了我们已经讲过的那些信号以外，还列出了四个寄存器 ID 信号：srcA，valA 的源；srcB，valB 的源；dstE，写入 valE 的寄存器；以及 dstM，写入 valM 的寄存器。

阶段	计算	OPq rA, rB	mrmovq D(rB), rA
取指	icode:ifun rA,rB valC valP	icode:ifun ← M_1[PC] rA:rB ← M_1[PC+1] valP ← PC+2	icode:ifun ← M_1[PC] rA:rB ← M_1[PC+1] valC ← M_8[PC+2] valP ← PC+10
译码	valA, srcA valB, srcB	valA ← R[rA] valB ← R[rB]	 valB ← R[rB]
执行	valE Cond. codes	valE ← valB OP valA Set CC	valE ← valB+valC
访存	Read/write		valM ← M_8[valE]
写回	E port, dstE M port, dstM	R[rB]← valE	 R[rA]← valM
更新 PC	PC	PC ← valP	PC ← valP

图 4-24　标识顺序实现中的不同计算步骤。第二栏标识出 SEQ 阶段中正在被计算的值，或正在被执行的操作。以指令 OPq 和 mrmovq 的计算作为示例

图中，右边两栏给出的是指令 OPq 和 mrmovq 的计算，来说明要计算的值。要将这些计算映射到硬件上，我们要实现控制逻辑，它能在不同硬件单元之间传送数据，以及操作这些单元，使得对每个不同的指令执行指定的运算。这就是控制逻辑块的目标，控制逻辑块在图 4-23 中用灰色圆角方框表示。我们的任务就是依次经过每个阶段，创建这些块的详细设计。

4.3.3　SEQ 的时序

在介绍图 4-18～图 4-21 的表时，我们说过要把它们看成是用程序符号写的，那些赋值是从上到下顺序执行的。然而，图 4-23 中硬件结构的操作运行根本完全不同，一个时