示。然后在各个阶段之间放上流水线寄存器(pipeline register)，这样每条指令都会按照三步经过这个系统，从头到尾需要三个完整的时钟周期。如图 4-33 中的流水线图所示，只要 I1 从 A 进入 B，就可以让 I2 进入阶段 A 了，依此类推。在稳定状态下，三个阶段都应该是活动的，每个时钟周期，一条指令离开系统，一条新的进入。从流水线图中第三个时钟周期就能看出这一点，此时，I1 是在阶段 C，I2 在阶段 B，而 I3 是在阶段 A。在这个系统中，我们将时钟周期设为 100＋20＝120ps，得到的吞吐量大约为 8.33 GIPS。因为处理一条指令需要 3 个时钟周期，所以这条流水线的延迟就是 3×120＝360ps。我们将系统吞吐量提高到原来的 8.33/3.12＝2.67 倍，代价是增加了一些硬件，以及延迟的少量增加(360/320＝1.12)。延迟变大是由于增加的流水线寄存器的时间开销。

4.4.2　流水线操作的详细说明

为了更好地理解流水线是怎样工作的，让我们来详细看看流水线计算的时序和操作。图 4-34 给出了前面我们看到过的三阶段流水线(图 4-33)的流水线图。就像流水线图上方指明的那样，流水线阶段之间的指令转移是由时钟信号来控制的。每隔 120ps，信号从 0 上升至 1，开始下一组流水线阶段的计算。

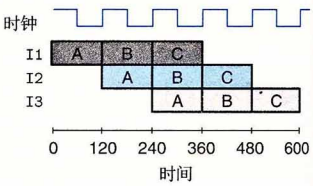

图 4-34　三阶段流水线的时序。时钟信号的上升沿控制指令从一个流水线阶段移动到下一个阶段

图 4-35 跟踪了时刻 240～360 之间的电路活动，指令 I1 经过阶段 C，I2 经过阶段 B，而 I3 经过阶段 A。就在时刻 240(点 1)时钟上升之前，阶段 A 中计算的指令 I2 的值已经到达第一个流水线寄存器的输入，但是该寄存器的状态和输出还保持为指令 I1 在阶段 A 中计算的值。指令 I1 在阶段 B 中计算的值已经到达第二个流水线寄存器的输入。当时钟上升时，这些输入被加载到流水线寄存器中，成为寄存器的输出(点 2)。另外，阶段 A 的输入被设置成发起指令 I3 的计算。然后信号传播通过各个阶段的组合逻辑(点 3)。就像图中点 3 处的曲线化的波阵面(curved wavefront)表明的那样，信号可能以不同的速率通过各个不同的部分。在时刻 360 之前，结果值到达流水线寄存器的输入(点 4)。当时刻 360 时钟上升时，各条指令会前进经过一个流水线阶段。

从这个对流水线操作详细的描述中，我们可以看到减缓时钟不会影响流水线的行为。信号传播到流水线寄存器的输入，但是直到时钟上升时才会改变寄存器的状态。另一方面，如果时钟运行得太快，就会有灾难性的后果。值可能会来不及通过组合逻辑，因此当时钟上升时，寄存器的输入还不是合法的值。

根据对 SEQ 处理器时序的讨论(4.3.3 节)，我们看到这种在组合逻辑块之间采用时钟寄存器的简单机制，足够控制流水线中的指令流。随着时钟周而复始地上升和下降，不同的指令就会通过流水线的各个阶段，不会相互干扰。

4.4.3　流水线的局限性

图 4-33 的例子给出了一个理想的流水线化的系统，在这个系统中，我们可以将计算分成三个相互独立的阶段，每个阶段需要的时间是原来逻辑需要时间的三分之一。不幸的是，会出现其他一些因素，降低流水线的效率。