计算机系统结构
第三章 流水线技术

主　讲：刘 超

中国地质大学（武汉）计算机学院

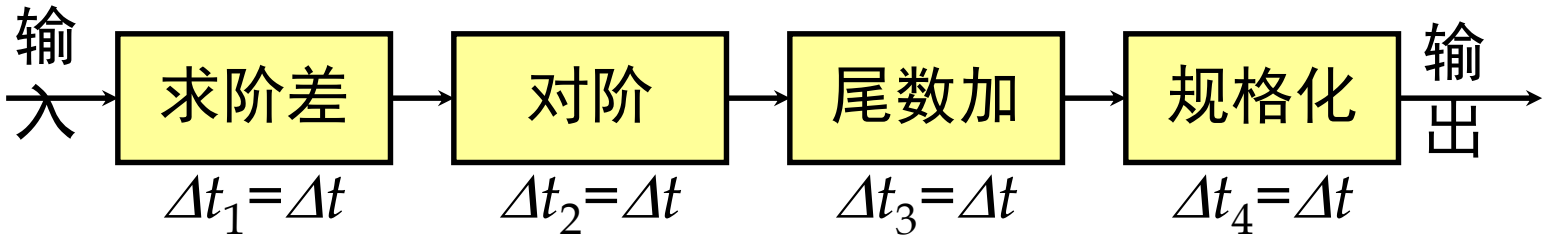
小节目录

流水线性能分析举例二

- 单功能、线性流水线，输入任务不连续的情况，计算流水线的吞吐率、加速比和效率。

- 例：用图中所示的一条4段浮点加法流水线计算8个浮点数的和：

$$Z=A+B+C+D+E+F+G+H$$

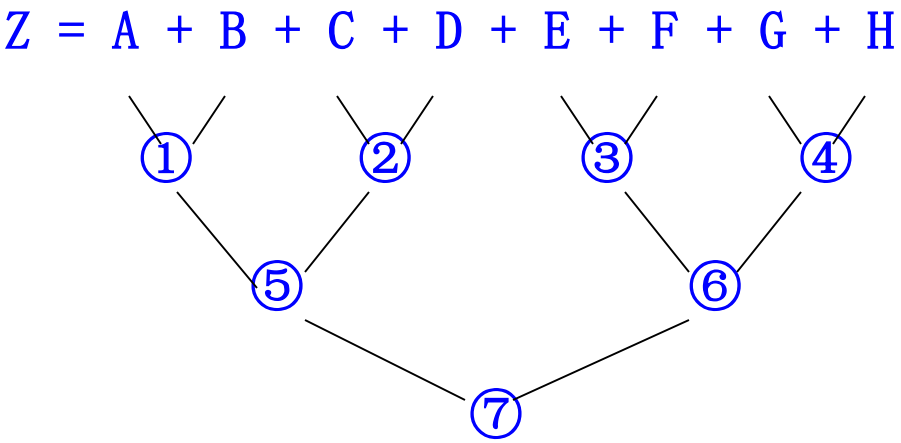


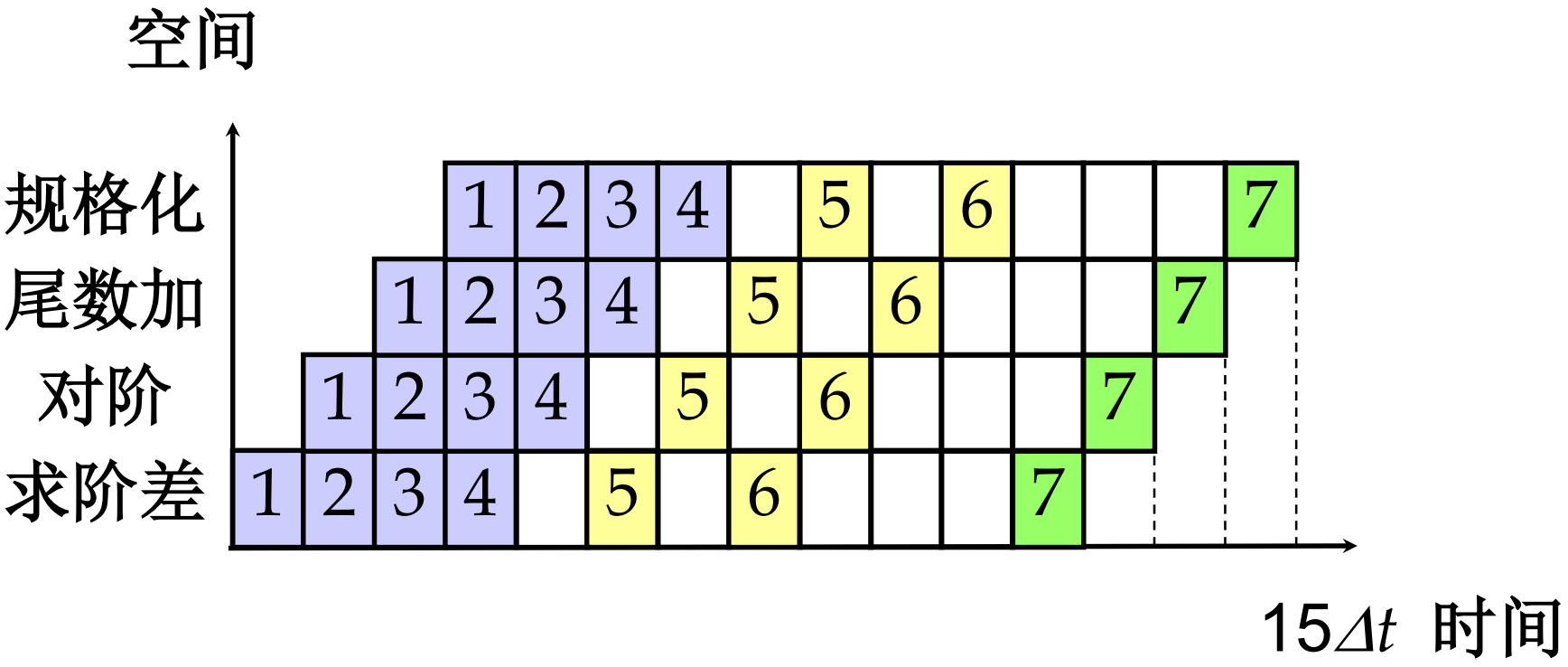
解：

- 先将表达式转换为：

  Z = [(A+B) + (C+D)] + [(E+F) + (G+H)] 通过二叉数算法获得最大的并行度

  可见，共执行7次浮点加法，这7次任务的次序为：
  - 1、A+B
  - 2、C+D
  - 3、E+F
  - 4、G+H
  - 5、 [(A+B) + (C+D)]
  - 6、 [(E+F) + (G+H)]
  - 7、 [(A+B) + (C+D)] + [(E+F) + (G+H)]
- 指令间存在相关，所以指令不能完全顺序流入流水线
  - 5必须在1,2后面执行
  - 6必须在3,4后面执行
  - 7必须在5,6后面执行
- 流水线的时空图见下一页





用一条4段浮点加法流水线求8个数之和的流水线时空图

共有K=4个流水段，n= 7次浮点加法运算共用了15个时钟周期。一次浮点加法运算要4个周期。当每个流水段的延迟都为Δt时：

- 流水线的吞吐率为：

$$TP = \frac{n}{T_k} = \frac{7}{15 \cdot \Delta t} = 0.47 \frac{1}{\Delta t}$$

- 流水线的加速比为：

$$S = \frac{T_0}{T_k} = \frac{4 \times 7 \cdot \Delta t}{15 \cdot \Delta t} = 1.87$$

- 流水线的效率为：

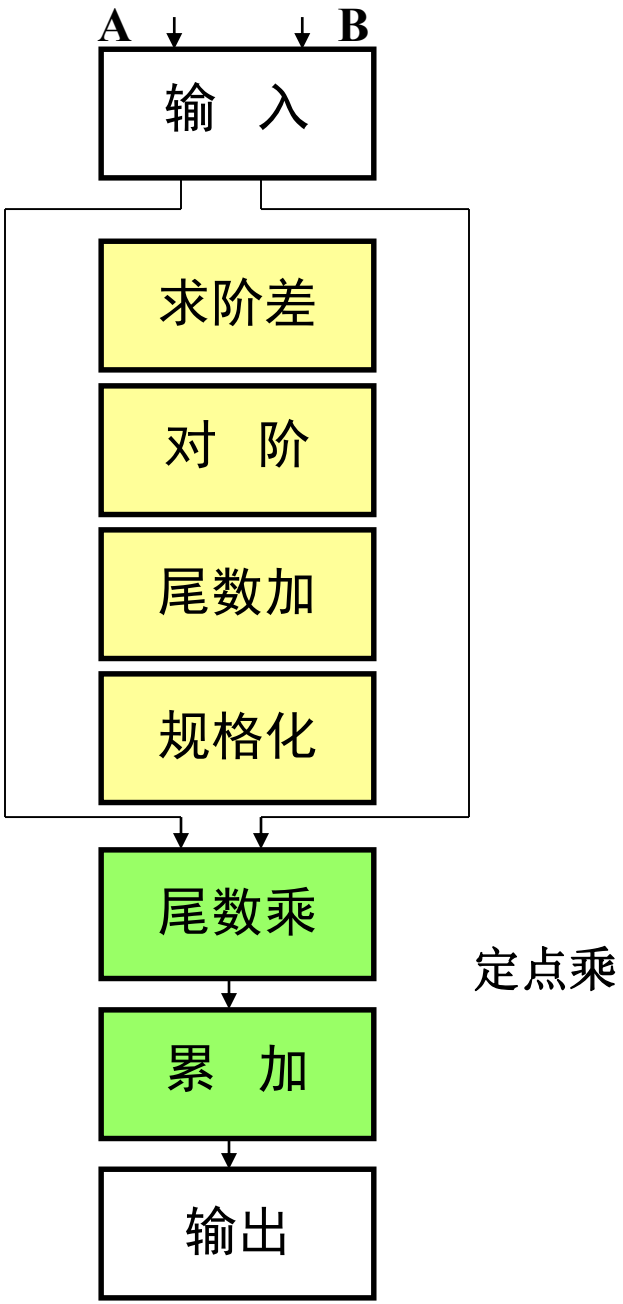
$$E = \frac{T_0}{k \cdot T_k} = \frac{4 \times 7 \cdot \Delta t}{4 \times 15 \cdot \Delta t} = 0.47$$

流水线性能分析举例三

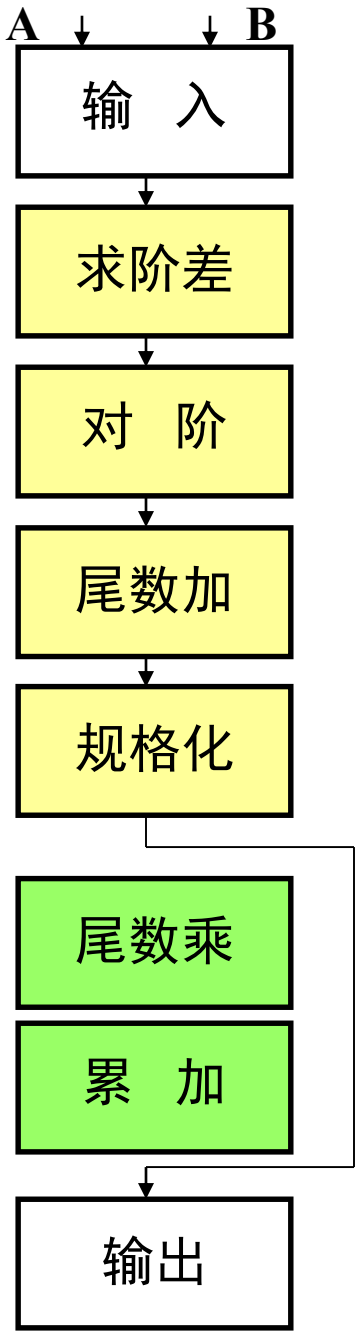
- 多功能、线性流水线，输入任务不连续的情况，计算流水线的吞吐率、加速比和效率。

- 例：用下图中所示的TI-ASC计算机的多功能静态流水线计算两个向量的点积

$$Z = A \cdot B + C \cdot D + E \cdot F + G \cdot H$$



定点乘



浮点加

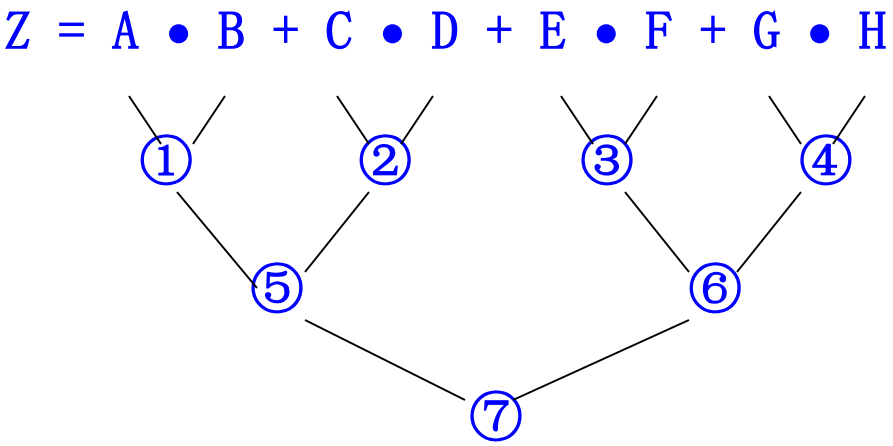
解：

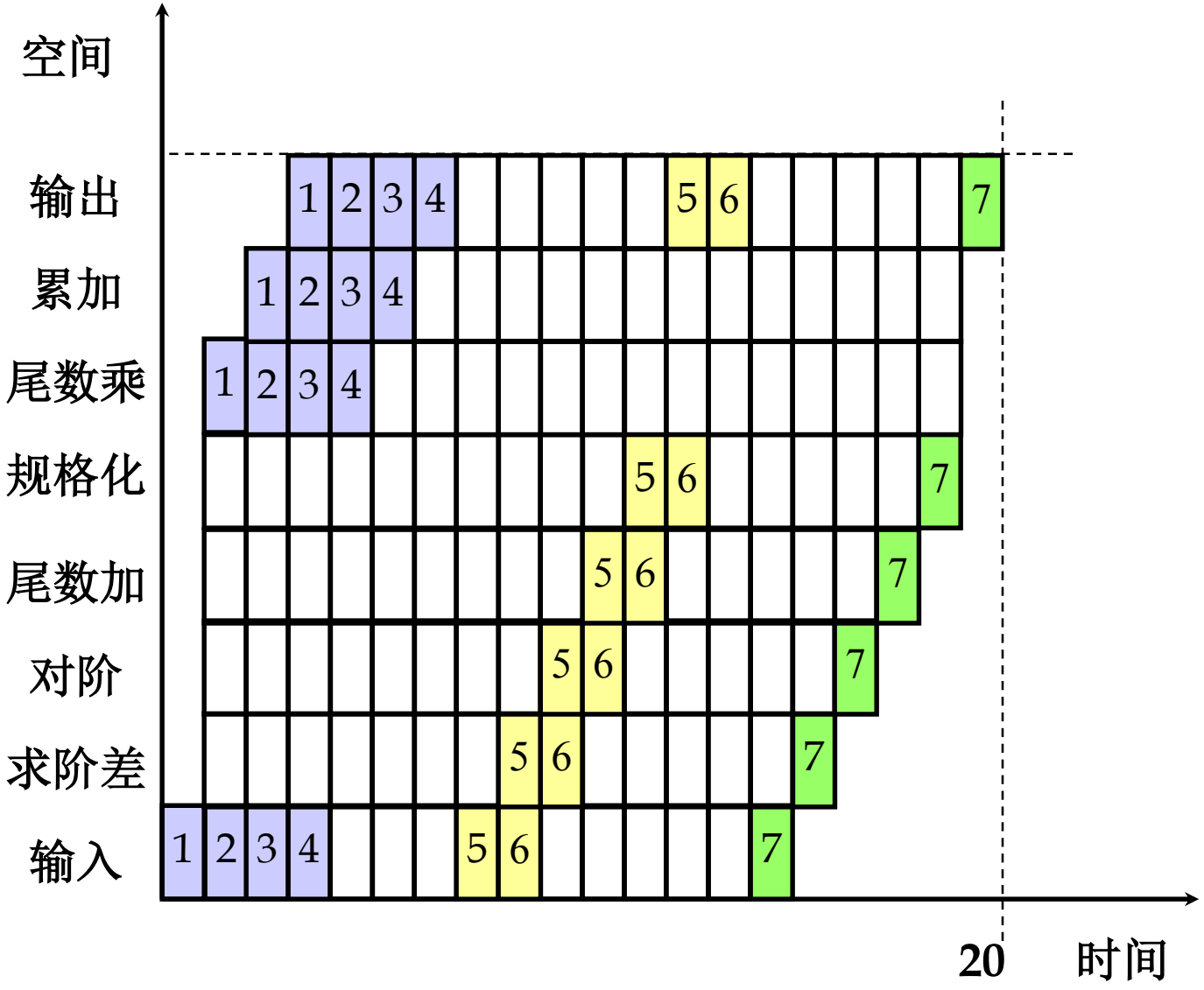
- 先将表达式转换为：

  Z = [(A.B) + (C.D)] + [(E.F) + (G.H)]

  通过二叉数算法获得最大的并行度

  可见，共执行**4**次乘法、**3**次浮点加法，这**7**次任务的次序为：
  - 1、A.B
  - 2、C.D
  - 3、E.F
  - 4、G.H
  - 5、 [(A.B) + (C.D)]
  - 6、 [(E.F) + (G.H)]
  - 7、 [(A.B) + (C.D)] + [(E.F) + (G.H)]
- 指令间存在相关，所以指令不能完全顺序流入流水线
  - 5必须在1,2后面执行
  - 6必须在3,4后面执行
  - 7必须在5,6后面执行
- 流水线的时空图见下一页





用TI-ASC多功能静态流水线求两个向量点积的流水线时空图

共有**K=8**个流水段，**n=7**个运算共用了**20**个时钟周期。一次加法运算要**6**个周期，一次乘法运算需**4**个周期。当每个流水段的延迟都为Δt时：

- 流水线的吞吐率为：

$$TP = \frac{n}{T_k} = \frac{7}{20\Delta t} = 0.35\frac{1}{\Delta t}$$

- 流水线的加速比为：

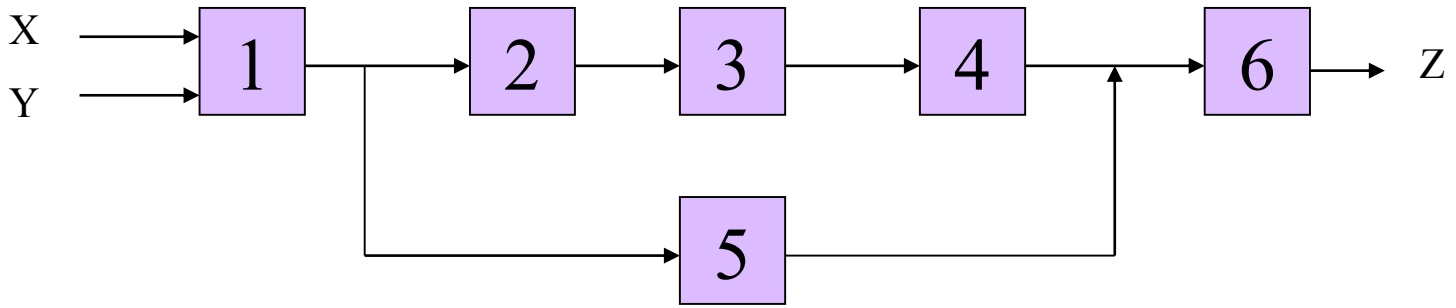
$$S = \frac{T_0}{T_k} = \frac{4\times 4\Delta t + 3\times 6\Delta t}{20\Delta t} = \frac{34\Delta t}{20\Delta t} = 1.70$$

- 流水线的效率为：

$$E = \frac{T_0}{k \bullet T_k} = \frac{34\Delta t}{8\times 20\Delta t} = 0.21$$

流水线性能分析举例四

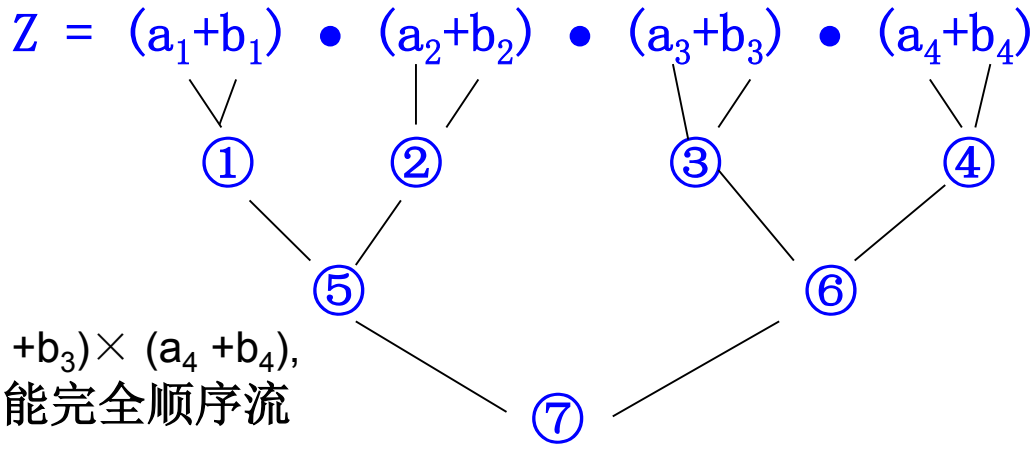
- 一条线性静态多功能流水线由如图所示六个功能段组成, 加法操作使用其中的1, 5, 6功能段, 乘法操作使用其中的1, 2, 3, 4, 6功能段, 每个功能段的延迟时间均为Δt ， 流水线的输出端和输入端之间有直接数据通路, 而且设置有足够的缓冲寄存器。现在用这条流水线计算$(a_1 + b_1) \times (a_2 + b_2) \times (a_3 + b_3) \times (a_4 + b_4)$, 画出流水线的时—空图, 并计算流水线的实际吞吐率, 加速比和效率.



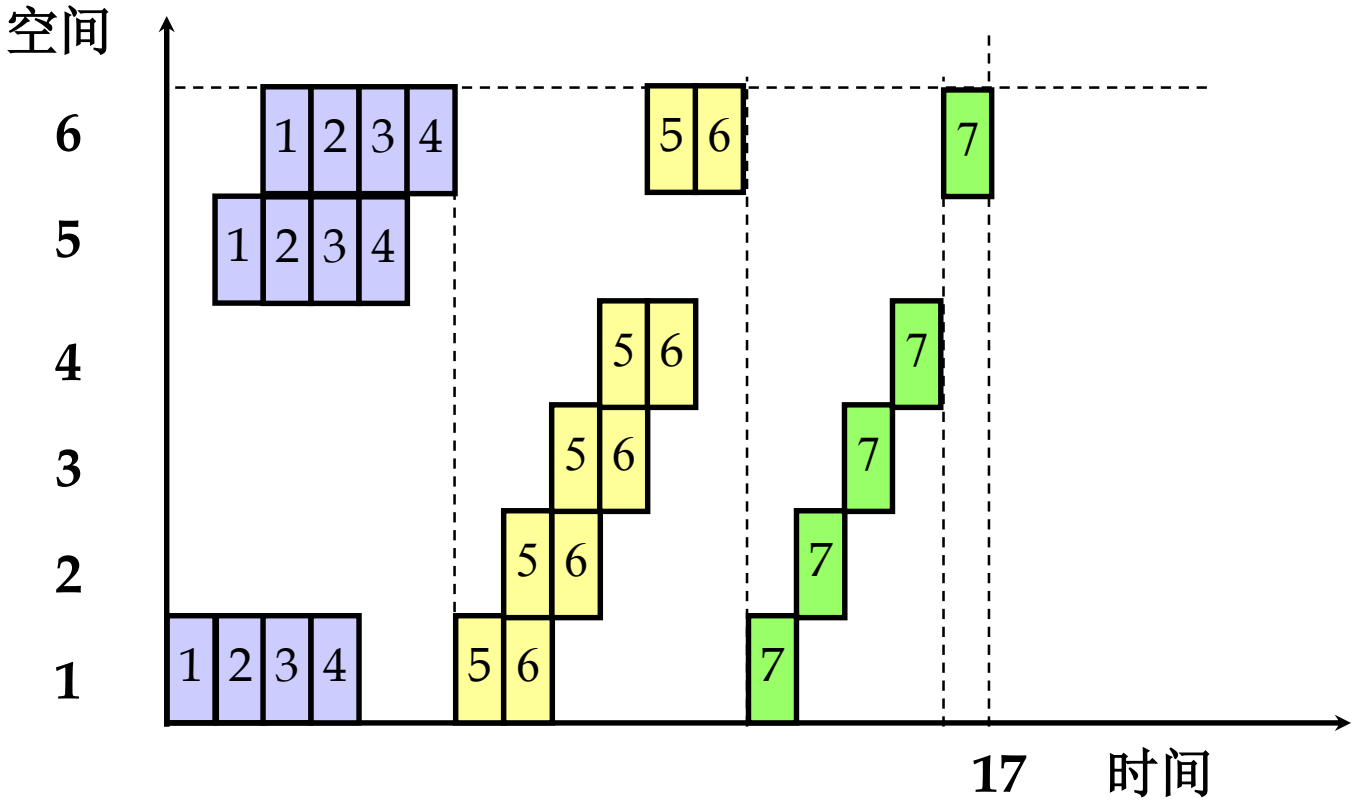
解：

- 先将表达式

$(a_1+b_1)\times(a_2+b_2)\times(a_3+b_3)\times(a_4+b_4)$,进行任务分解，共执行4次加法、3次乘法，这7次任务的次序为：
  - 1、 a_1+b_1
  - 2、 a_2+b_2
  - 3、 a_3+b_3
  - 4、 a_4+b_4
  - 5、 $(a_1+b_1)\times(a_2+b_2)$
  - 6、 $(a_3+b_3)\times(a_4+b_4)$
  - 7、 $(a_1+b_1)\times(a_2+b_2)\times(a_3+b_3)\times(a_4+b_4)$,



- 指令间存在相关，所以指令不能完全顺序流入流水线
  - 5必须在1,2后面执行
  - 6必须在3,4后面执行
  - 7必须在5,6后面执行
- 流水线的时空图见下一页



$$\text{吞吐率} TP = \frac{n}{T_k} = \frac{7}{17\Delta t}$$

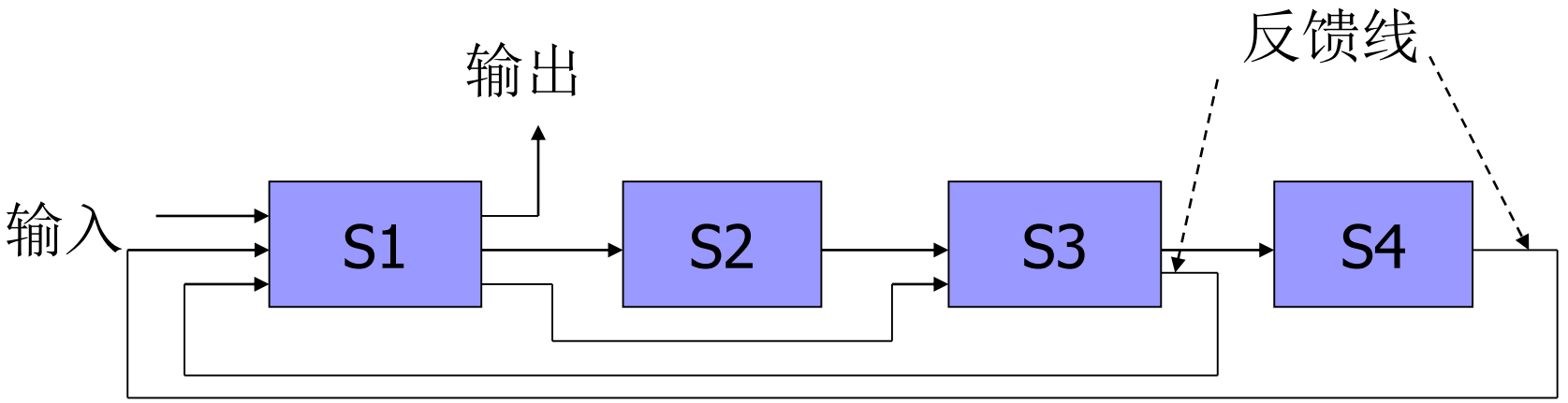
$$\text{加速比} S = \frac{T_0}{T_k} = \frac{4\times 3\Delta t + 3\times 5\Delta t}{17\Delta t} = \frac{27\Delta t}{17\Delta t} = 1.589$$

$$\text{效率} E = \frac{T_0}{k\times T_k} = \frac{27\Delta t}{6\times 17\Delta t} = 0.265 = 26.5\%$$

3.3 非线性流水线的调度

- 非线性流水线：流水线各个段间除了串行连接外，还有反馈回路，从而使处理的任务要多次流过某些段的流水线。
- 非线性流水线的调度要解决的问题：确定任务流入流水线的时间间隔，使其既不发生任务争用流水段的冲突，又有较高的吞吐率和效率。

非线性流水线的连接图



单功能非线性流水线最优调度

- 向一条非线性流水线的输入端连续输入两个任务之间的时间间隔称为非线性流水线的启动距离。
- 会引起非线性流水线功能段使用冲突的启动距离则称为禁用启动距离。
- 启动距离和禁用启动距离一般都用时钟周期数来表示，是一个正整数。

单功能非线性流水线最优调度算法

- 1)根据任务对流水线各段的使用时间建立一个预约表；
- 2)由预约表得出禁止表，禁止表是禁止后续任务流入流水线的时间间隔的集合；
- 3)由禁止表得出初始冲突向量；
- 4)由初始冲突向量得出状态有向图；
- 5)由状态有向图得出最优调度策略；

预约表 (Reservation Table)

- 预约表：表示某任务占用各流水段的预约情况。横坐标表示处理该任务所使用的流水线时钟周期，纵坐标表示流水线的各个流水段，中间有“✓”表示该流水段在这一个时钟周期处于工作状态，空白表示该流水段在这一个时钟周期不工作。

	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9
S_1	✓								✓
S_2		✓	✓					✓	
S_3				✓					
S_4					✓	✓			
S_5							✓	✓	

非线性流水线的预约表示例

由预约表得出禁止表F(Forbidden List)

- 禁止表F：一个由禁用启动距离构成的集合。
- 将预约表的每一行中任意两个“✓”之间的距离都计算出来，去掉重复的，这种数组成的一个数列就是这条非线性流水线的禁止向量。
- 例如：前述的非线性流水线，
  - 第一行的差值只有一个：8；
  - 第二行的差值有3个：1，5，6；
  - 第3行只有一个√，没有差值；
  - 第4和第5行的差值都只有一个：1；
- 其禁止向量，**F = {1, 5, 6, 8}**

由禁止表得出初始冲突向量 (Collision Vector)

- 初始冲突向量$C_0=(C_nC_{n-1}\cdots C_k\cdots C_2C_1)$，其中

$$C_k=\begin{cases}1 & \text{禁止间隔}k\Delta t\text{流入一个后续任务}\\ 0 & \text{允许间隔}k\Delta t\text{流入一个后续任务}\end{cases}$$

Δt为流水线各段的同步时钟周期。

冲突向量的位数为禁止表中的最大量

- 例如：前述的禁止表为 F ={1, 5, 6, 8}

  得到初始冲突向量 C_0=(10110001)

根据初始冲突向量C_0画出状态转换图

- 当第一个任务流入流水线后，初始冲突向量C_0决定了下一个任务需间隔多少个时钟周期才可以流入。
- 在第二个任务流入后，新的冲突向量是怎样的呢？
  - 假设第二个任务是在与第一个任务间隔j个时钟周期流入，这时，由于第一个任务已经在流水线中前进了j个时钟周期，其相应的禁止表中各元素的值都应该减去j，并丢弃小于等于0的值。
  - 对冲突向量来说，就是逻辑右移j位（左边补0）。

- 对它们的冲突向量进行“或”运算。

$$SHR^{(j)}（C_0）\vee C_0$$

其中：$SHR^{(j)}$表示逻辑右移j位

- **推广到更一般的情况**

假设: C_k：当前的冲突向量

j: 允许的时间间隔

则新的冲突向量为：

$$SHR^{(j)}（C_k）\vee C_0$$

- **对于所有允许的时间间隔都按上述步骤求出其新的冲突向量，并且把新的冲突向量作为当前冲突向量，反复使用上述步骤，直到不再产生新的冲突向量为止。**

- **从初始冲突向量C_0出发，反复应用上述步骤，可以求得所有的冲突向量以及产生这些向量所对应的时间间隔。由此可以画出用冲突向量表示的流水线状态转移图。**
  - 有向弧：表示状态转移的方向
  - 弧上的数字：表示引入后续任务（从而产生新的冲突向量）所用的时间间隔（时钟周期数）

对于上面的例子

（1） C_0=（10110001）

引入后续任务可用的时间间隔为：2、3、4、7个时钟周期

如果采用2，则新的冲突向量为：

（00101100）∨（10110001）= （10111101）

如果采用3，则新的冲突向量为：

（00010110）∨（10110001）= （10110111）

如果采用4，则新的冲突向量为：

（00001011）∨（10110001）= （10111011）

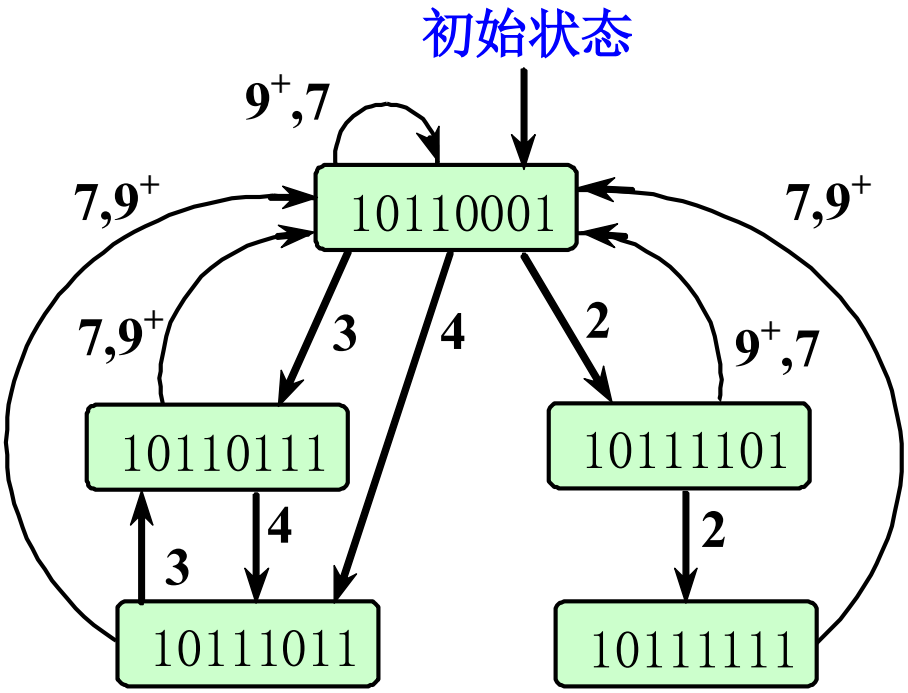
如果采用7，则新的冲突向量为：

（00000001）∨（10110001）= （10110001）

（2）对于新向量（10111101），其可用的时间间隔为2个和7个时钟周期。用类似上面的方法，可以求出其后续的冲突向量分别为（10111101）和（10110001）。

（3）对于其他新向量，也照此处理。

（4）在此基础上，画出状态转移示意图。



由初始冲突向量得出状态有向图小结

F = {1, 5, 6, 8}

C_0 10110001
K=2,3,4,7

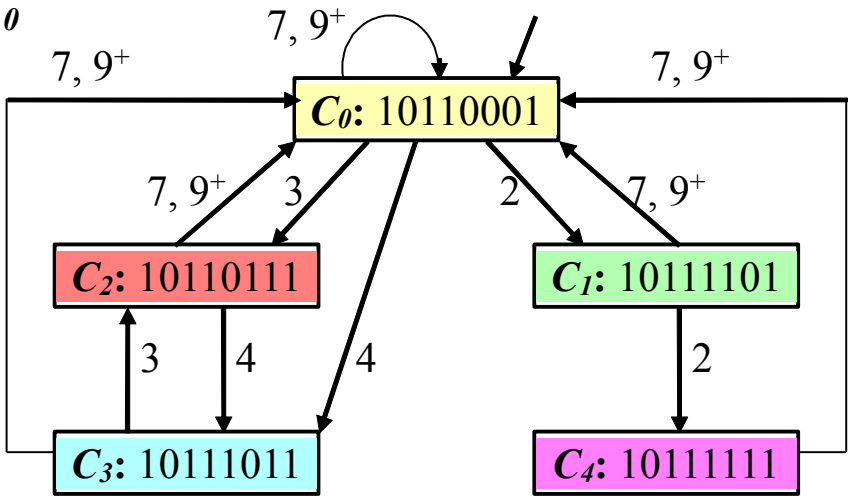
00101100	00010110	00001011	00000001
10110001	10110001	10110001	10110001
10111101	10110111	10111011	10110001
K=2 得C_1	*K=3* 得C_2	*K=4* 得C_3	*K=7* 得C_0

C_1 10111101
K=2,7

00101111	00000001
10110001	10110001
10111111	10110001
K=2 得C_4	*K=7* 得C_0

对*C2*、*C3*、*C4*分别计算
发现不再产生新的状态向量。

从而得出状态向量图：



状态有向图

后续状态向量的计算公式小结

后续状态计算公式：

$C_j = SHR^{(k)}(C_i) \vee C_0$

当前状态C_i

k为C_i允许输入下任务的一个时间间隔

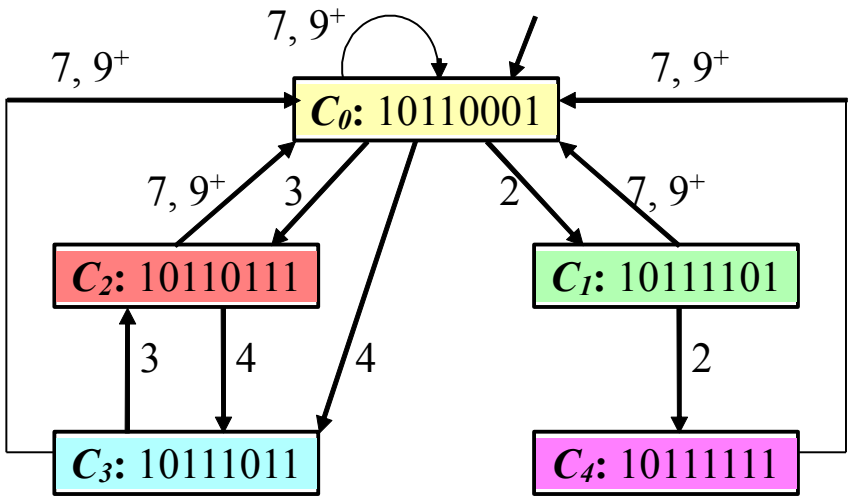
$SHR^{(k)}$表示右移k位，高位补0

后继状态C_j由向量$SHR^{(k)}(C_i)$和初始冲突向量C_0“或”运算得出

循环计算，直到不再有新的状态向量产生为止

最后用有向线段连接各个状态，并标记k值

由状态有向图得出各种调度方案，比较后，得出最优调度策略



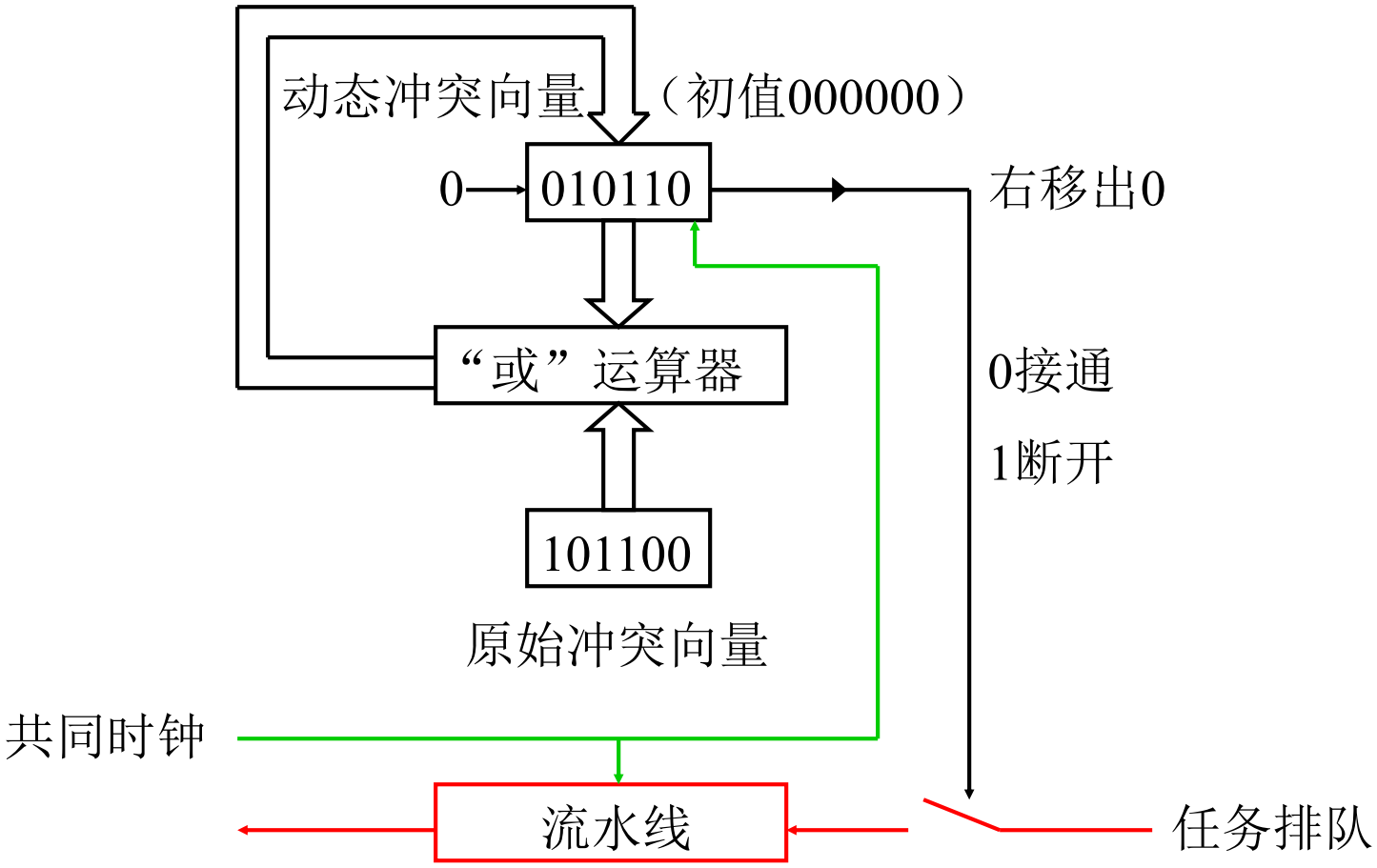
单功能非线性流水线状态有向图

调度策略	平均间隔周期
bc: (3,4)	3.50
ad: (2,7)	4.50
ade: (2,2,7)	3.67
abcb: (3,4,3,7)	4.25
abc: (3,4,7)	4.67
acb: (4,3,7)	4.67
ab: (3,7)	5.00
ac: (4,7)	5.50
a: (7)	7.00

- 找出所有的环路，既可得到调度方案
- 比较各个方案的平均时间间隔，最小的为最佳调度策略
- 本例中 bc 为最佳调度策略，平均间隔时间3.5个时钟周期（吞吐率最高）。注意：方案（4，3）不是最佳方案，why？

- 方案（3，4）是一种不等时间间隔的调度方案，与等间隔的调度方案相比，在控制上要复杂得多。为了简化控制，也可以采用等间隔时间的调度方案，但吞吐率和效率往往会下降不少。
  - 在上述例子中，等时间间隔的方案只有一个：（7），其吞吐率下降了一半。

非线性流水线调度的实现



3.3.2多功能非线性流水线的调度

本小节内容不做要求！

3.6 习题补充

[习题1]

假设一条指令的执行过程分为“取指令”、“分析”和“执行”三段，每一段的执行时间分别为Δt、$2\Delta t$和$3\Delta t$。在下列各种情况下，分别写出连续执行n条指令所需要的时间表达式。

（1）顺序执行方式。

（2）仅“取指令”和“执行”重叠。

（3）“取指令”、“分析”和“执行”重叠

[习题1解答]

（1）顺序执行需要的时间如下：

$$T = (\Delta t + 2\Delta t + 3\Delta t) \times n = 6n\Delta t$$

（2）取指令和执行重叠，即一次重叠执行方式，我们假设第n+1条指令的取指令和第n条指令的执行同时结束，那么所需要的时间为：

$$T = \Delta t + (2\Delta t + 3\Delta t) \times n = 5n\Delta t + \Delta t$$

（3）取指令、分析和执行重叠

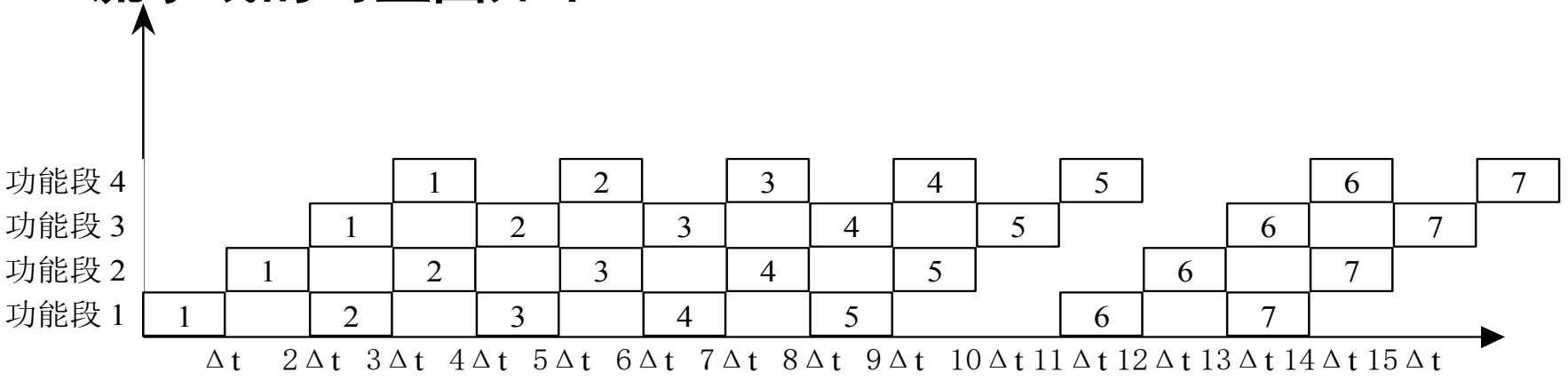
$$T = (\Delta t + 2\Delta t + 3\Delta t) + (n-1)3\Delta t = 5\Delta t + 3n\Delta t$$

[习题2]

一条线性流水线有4个功能段组成，每个功能段的延迟时间都相等，都为 Δt。开始5个任务是每间隔一个 Δt输入一个任务进流水线，然后停顿2个 Δt再输入下5个任务，如此循环往复。求流水线的实际吞吐率、加速比和效率。

[习题2解答]

流水线的时空图如下：



在（11n+1）Δt的时间内，可以输出5n个结果，如果指令的序列足够长（n→∞），并且指令间不存在相关，那么，吞吐率可以认为满足：

$$Tp = \frac{5n}{(11n+1)\Delta t} = \frac{5}{(11+1/n)\Delta t} = \frac{5}{11\Delta t}(n \to \infty)$$

加速比为：

$$S = \frac{5n \times 4\Delta t}{(11n+1)\Delta t} = \frac{20n}{11n+1} = \frac{20}{11+1/n} = \frac{20}{11}(n \to \infty)$$

从上面的时空图很容易看出，效率为：

$$E = \frac{T_0}{k \times T_k} = \frac{20n\Delta t}{4 \times (11n+1)\Delta t} = \frac{5}{11+1/n} = \frac{5}{11}(n \to \infty)$$

[习题3]

用一条5个功能段的浮点加法器流水线计算

$$F = \sum_{i=1}^{10} A_i$$

每个功能段的延迟时间均相等，流水线的输出端和输入端之间有直接数据通路，而且设置有足够的缓冲寄存器。要求用尽可能短的时间完成计算，画出流水线时空图，并计算流水线的实际吞吐率、加速比和效率。

求10个数的和需要做9次加法。在尽可能快的的情况下，要注意前后指令之间的相关。

I1：　　R1←A1+A2

I2：　　R2←A3+A4

I3：　　R3←A5+A6

I4：　　R4←A7+A8

I5：　　R5←A9+A10

I6：　　R6←R1+R2

I7：　　R7←R3+R4

I8：　　R8←R5+R6

I9：　　F←R7+R8

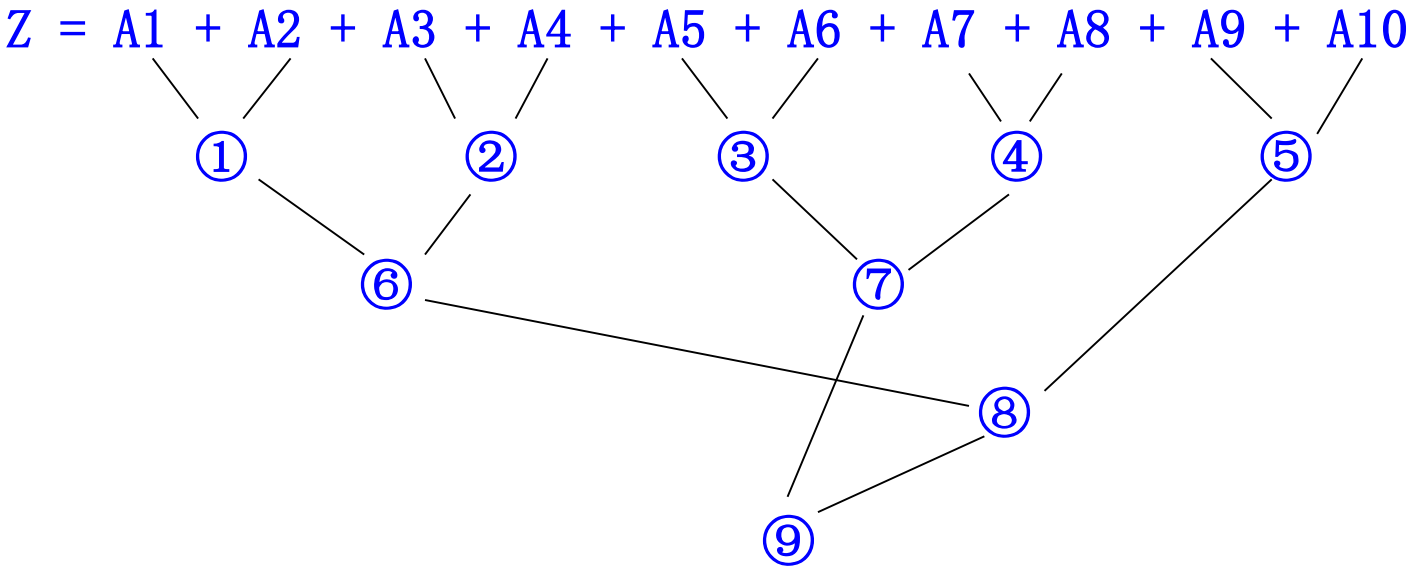
指令间存在相关:

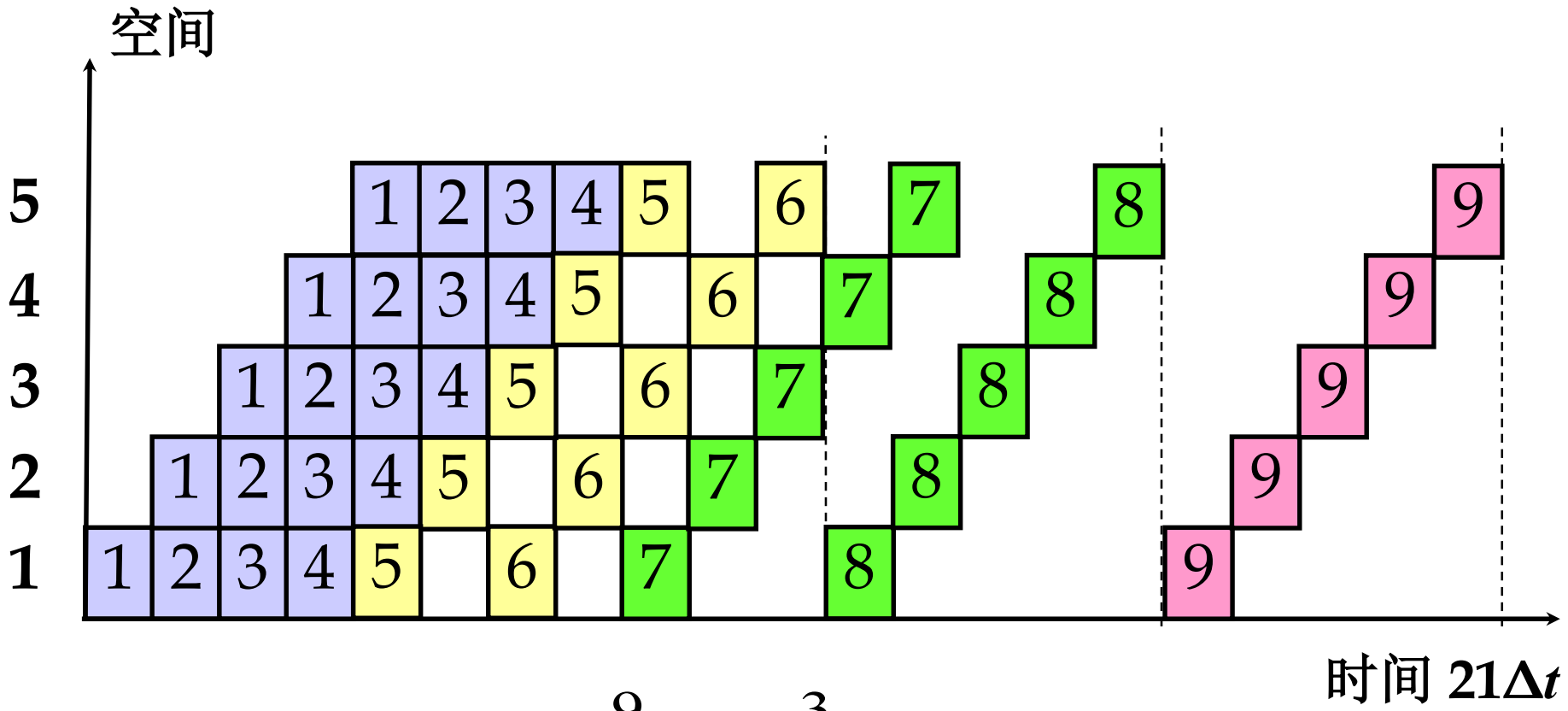
6必须在1,2后面执行

7必须在3,4后面执行

8必须在5,6后面执行

9必须在7,8后面执行





$$Tp = \frac{9}{21\Delta t} = \frac{3}{7\Delta t}$$

$$S = \frac{9 \times 5\Delta t}{21\Delta t} = \frac{45}{21} = 2.1429$$

$$E = \frac{T_0}{k \times T_k} = \frac{9 \times 5\Delta t}{5 \times 21\Delta t} = \frac{3}{7}$$

[习题4]

一条线性动态多功能流水线由6个功能段组成，加法操作使用其中的1、2、3、6功能段，乘法操作使用其中的1、4、5、6功能段，每个功能段的延迟时间均相等。流水线的输入端与输出端之间有直接数据通路，而且设置有足够的缓冲寄存器。现在用这条流水线计算：

$$F = \sum_{i=1}^{6} (A_i \times B_i)$$

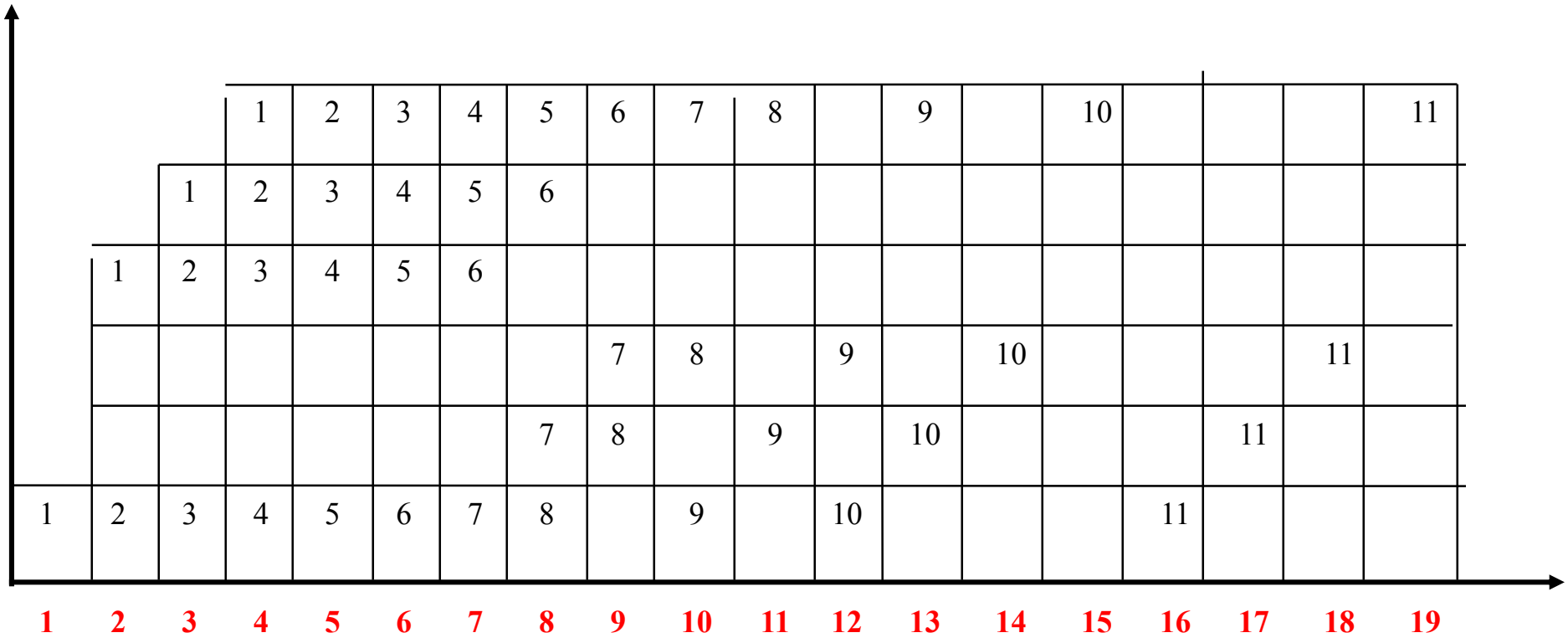
画出流水线时空图，并计算流水线的实际吞吐率、加速比和效率。

[习题4解答]

为了取得较高的速度，我们需要一次将乘法作完，设源操作数存放在寄存器A、B中，中间结果存放在寄存器R中，最后结果存放在寄存器F中，则执行的指令序列如下所示：

I1：　　R1←A1*B1
I2：　　R2←A2*B2
I3：　　R3←A3*B3
I4：　　R4←A4*B4
I5：　　R5←A5*B5
I6：　　R6←A6*B6
I7：　　R7←R1+R2
I8：　　R8←R3+R4
I9：　　R9←R5+R6
I10：　　R10←R7+R8
I11：　　F←R9+R10

这并不是唯一可能的计算方法。假设功能段的延迟为Δt。时空图（不完全）如下，图中的数字是指令号。


1 2 3 4 5 6 7 8 9 10 11
1 2 3 4 5 6
1 2 3 4 5 6
7 8 9 10 11
7 8 9 10 11
1 2 3 4 5 6 7 8 9 10 11
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19

整个计算过程需要19Δt，所以吞吐率为：

$$Tp = \frac{11}{19\Delta t}$$

加速比为：

$$S = \frac{11 \times 4\Delta t}{19\Delta t} = \frac{44}{19}$$

效率为：

$$E = \frac{T_0}{k \times T_k} = \frac{11 \times 4\Delta t}{6 \times 19\Delta t} = \frac{22}{57}$$

思考一下：如果改为静态流水线呢？