学习内容

- 5.1 重叠方式
- 5.2 流水方式
- 5.3 向量的流水处理与向量处理机
- 5.4 指令级高度并行的超级处理机
- 5.5 ARM流水线处理器举例

5.2 流水方式

- 流水线的基本概念
- 流水线分类
- 流水线性能
- 流水线相关
- 流水线调度

5.2.1 流水线的基本概念

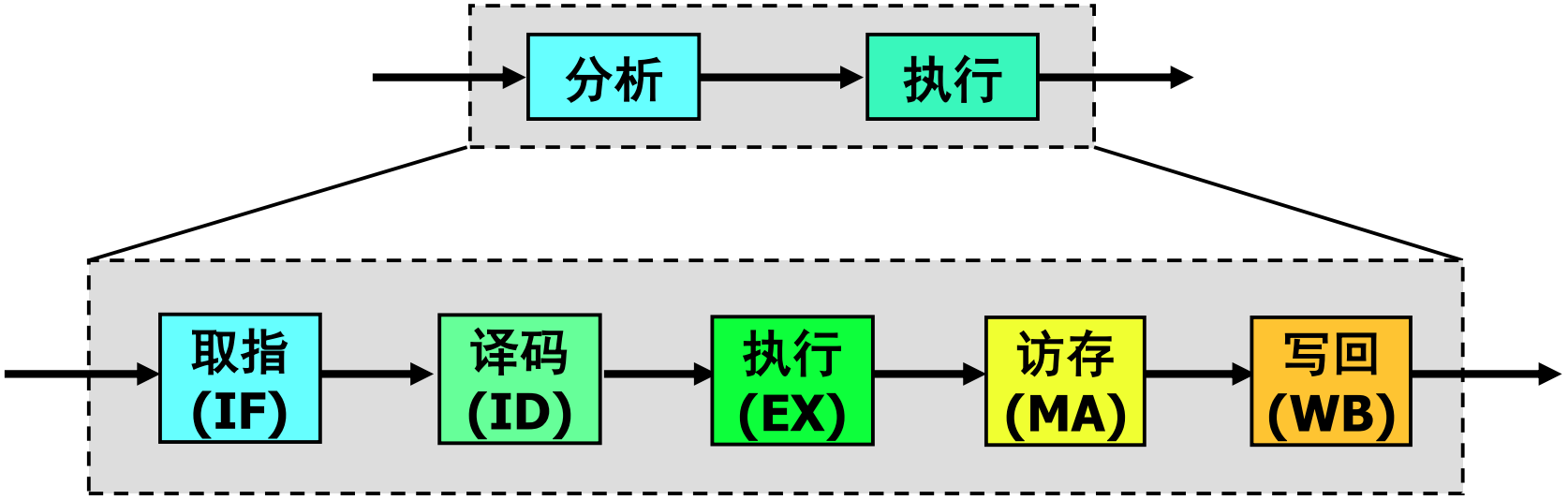
加快机器语言程序的执行可以通过以下两个途径实现：

- **提高每一条指令的执行速度**
  - 通过选用更高速的器件、采取更好的算法，提高指令内部各微操作的并行度。
- **提高多条指令的执行速度**
  - 通过控制机构采用同时解释和执行两条、多条甚至整段程序的控制方式，提高指令间的并行度。
  - **流水线技术是目前广泛应用的一项关键技术 。**

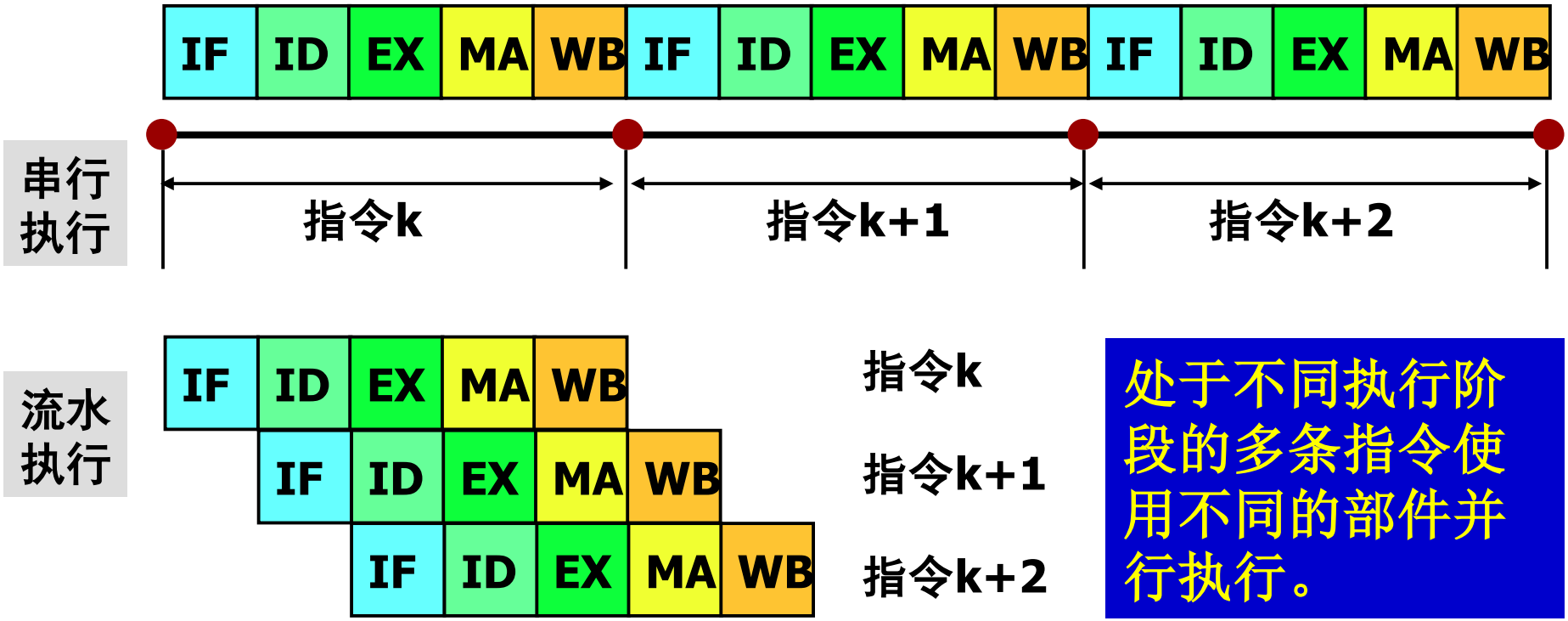
5.2.1 流水线的基本概念

- **流水线技术：流水是重叠的引申。**

将指令的执行过程分解为多个子过程，并让每个子过程分别由专用的部件完成，这些功能部件可以同时工作。让多条指令在时间上错开，依次通过各功能部件，这样，每个子过程就可以与其他的子过程并行进行，从而实现多条指令的并行执行，减少多条指令或一段程序的完成时间。



5.2.1 流水线的基本概念



假设每个子过程的处理时间均为Δt，则：

串行执行完成时间=5×Δt×3=15Δt

流水执行完成时间=7Δt

流水线特点

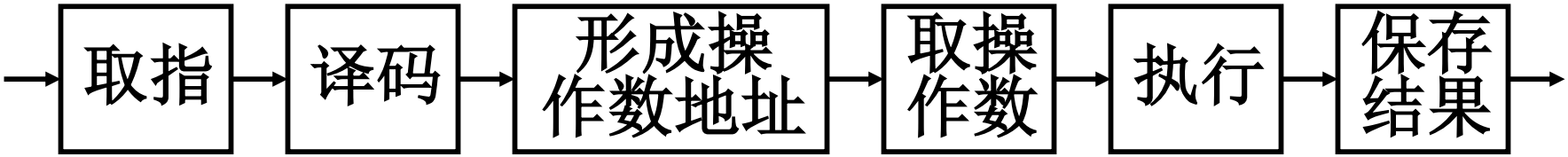
- 是一种基于时间重叠的并行处理技术；不能加快一条指令的执行，能加快多条指令的执行；
- 由多个有联系的子过程组成。这些子过程称为流水线的“级”或“段”。流水线的每一个阶段称为流水步、流水步骤、流水段、流水线阶段、流水功能段、功能段、流水级、流水节拍等。
- 段的数目称为流水线的“深度”；
- 在每一个流水段的末尾或开头必须设置一个寄存器，称为流水寄存器、流水锁存器、流水闸门寄存器等。设置寄存器会增加指令执行时间。
- 为了简化，在一般流水线中不画出流水锁存器。

流水线特点

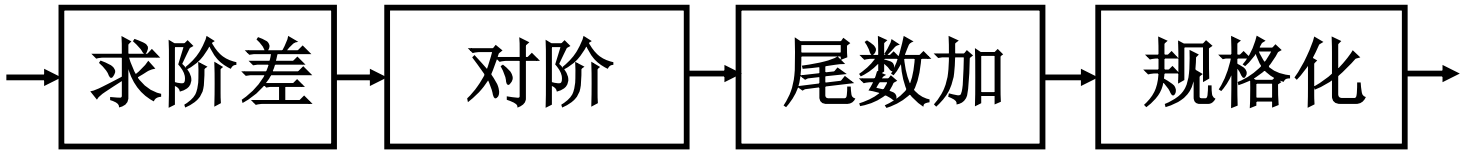
- 每个子过程分别由专用的部件完成。这些部件可以同时工作；
- 各流水段的时间应尽量相等。
- 流水线工作阶段可分为建立、满载和排空三个阶段：
  - 从第一个任务进入流水线到流水线所有部件都处于工作状态这个时期，称为流水线建立阶段；
  - 当所有部件都处于工作状态时，称为流水线满载阶段；
  - 从最后一条指令流入流水线到结果流出，称为流水线的排空阶段。
  - 建立和排空阶段所用的时间分别叫做“装入时间”和“排空时间”。
- 适合大量重复的时序过程。只有连续不断地向流水线输入任务，才能发挥流水线的效力。

流水线特点

- 一般指令流水线有4-12个流水段。
- 等于及大于8个流水段的处理机称为超流水线处理机。



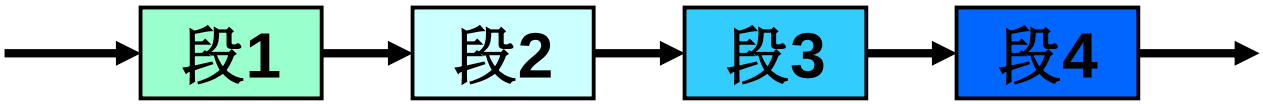
- 复杂指令的执行阶段也采用流水线，例如浮点加法器的流水线。



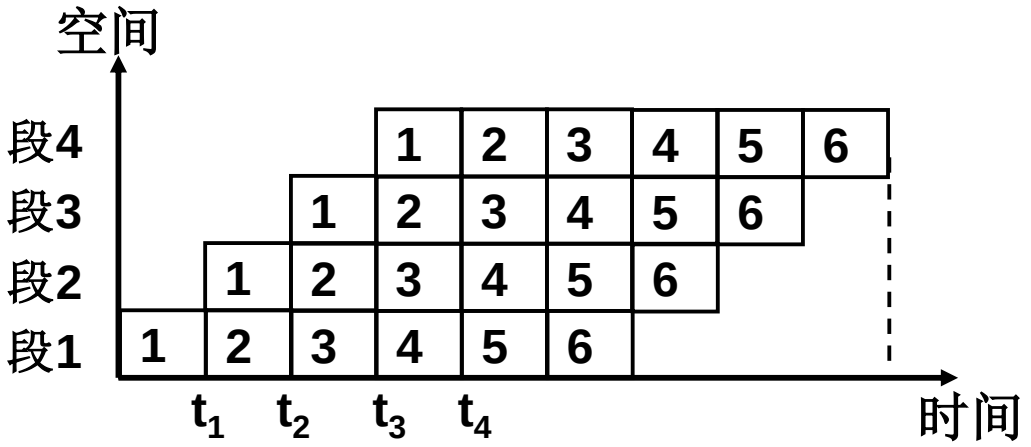
流水线表示方法：

连接图、时空图和预约表

- **连线图：**将各功能段用连接线连接在一起。



- **时空图：**表示指令执行过程。
  - 横坐标表示时间，纵坐标表示空间，即流水线的各个功能段。



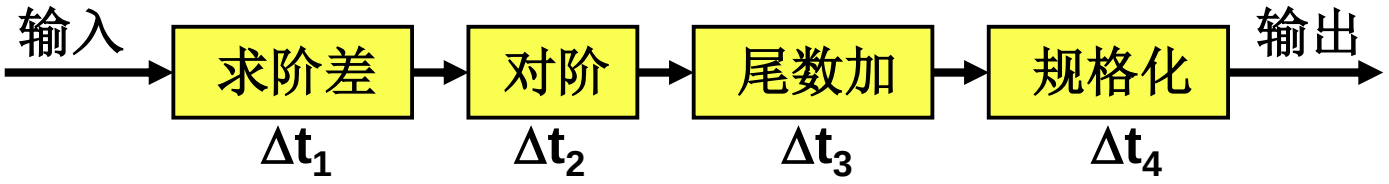
流水线分类

- 部件级/处理机级/处理机间级（宏流水线）
- 单功能/多功能
- 静态/动态
- 线性/非线性
- 标量/向量
- 同步/异步
- 顺序/乱序　等

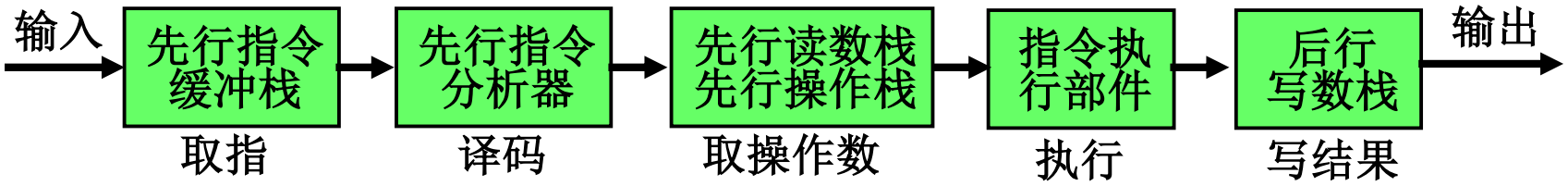
部件级/处理机级/处理机间级

- 根据向上和向下扩展的思想分类。
- 部件级（又称为运算操作流水线 ）
  - 指构成部件内的各子部件之间的流水。
- 处理机级（又称为指令流水线）
  - 指构成处理机的各个部件之间的流水。
- 系统级（又称为宏流水）
  - 指构成计算机系统的多个处理机之间的流水。

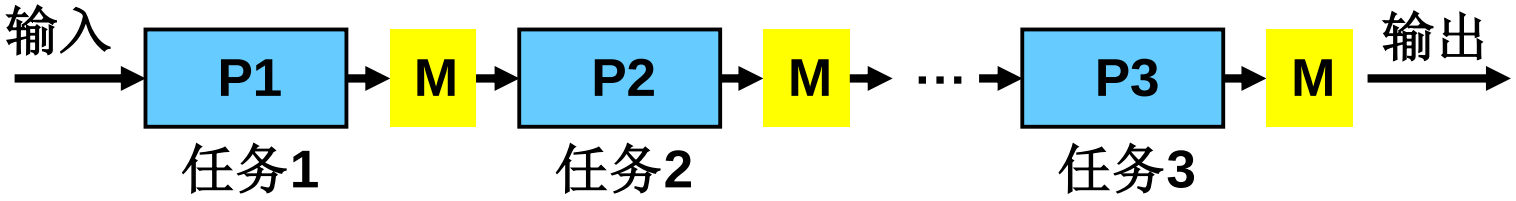
部件级/处理机级/处理机间级



部件级流水线（操作流水线）— 浮点加法器流水线



处理机级流水线 — 先行控制方式中的指令流水线



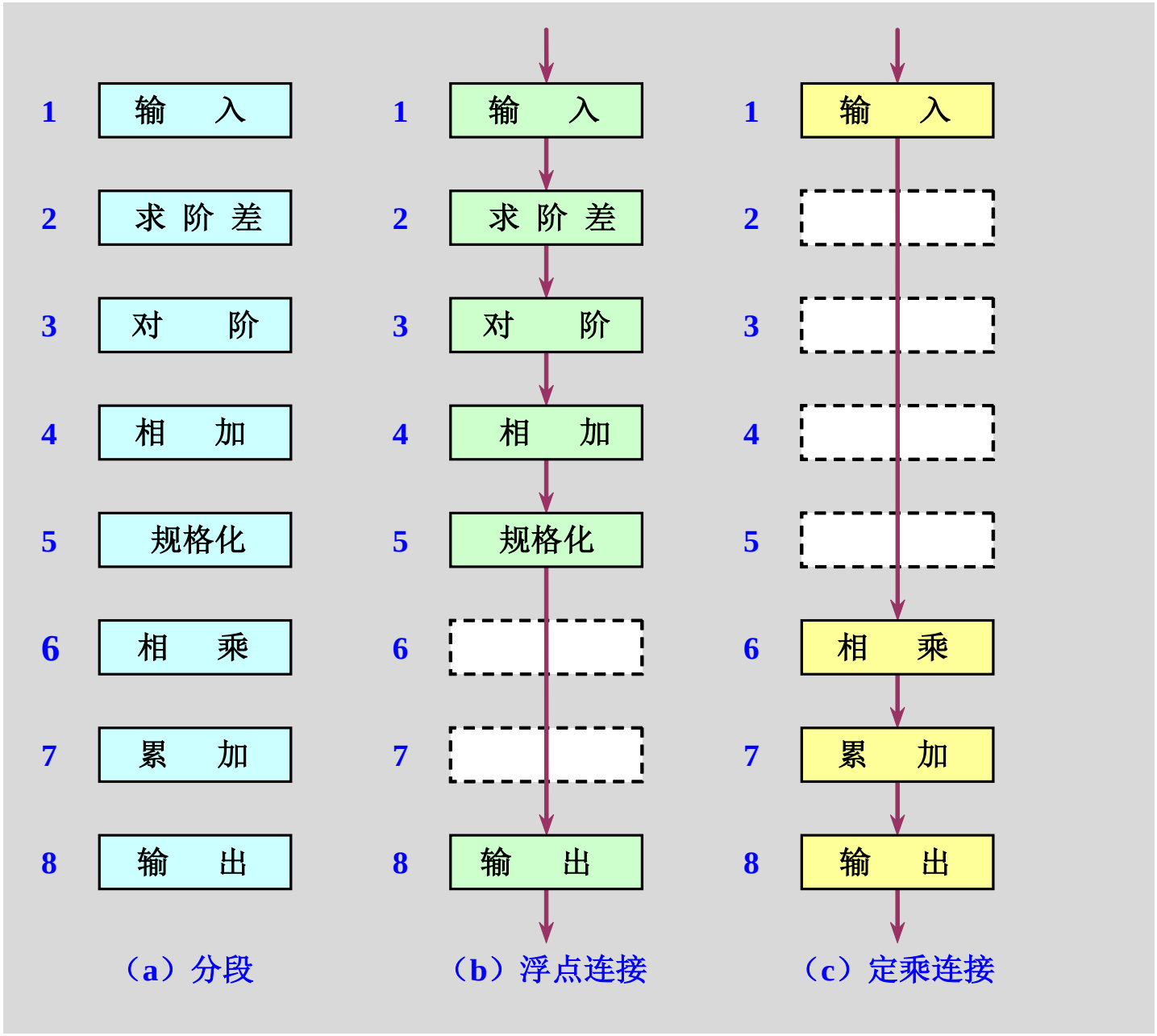
处理机之间的流水线 — 宏流水线

单功能/多功能

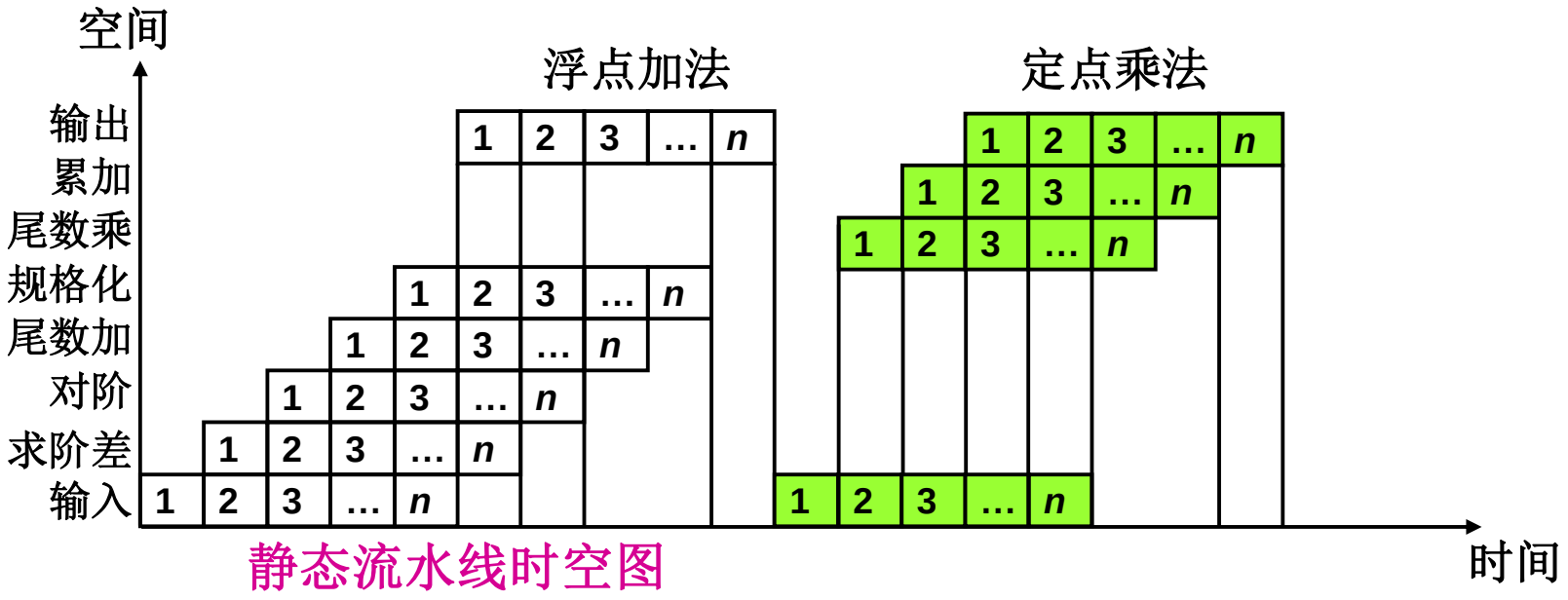
- 从流水线具有的功能分类。
- 单功能流水线
  - 指只能实现一种功能的流水线。
  - 例如：只实现浮点加减的流水线
  - 可以将多条单功能流水线组合起来，完成多功能的流水。
- 多功能流水线
  - 指同一流水线的各个段之间可以有多种不同的连接方式，以实现多种不同的运算或功能。

静态/动态

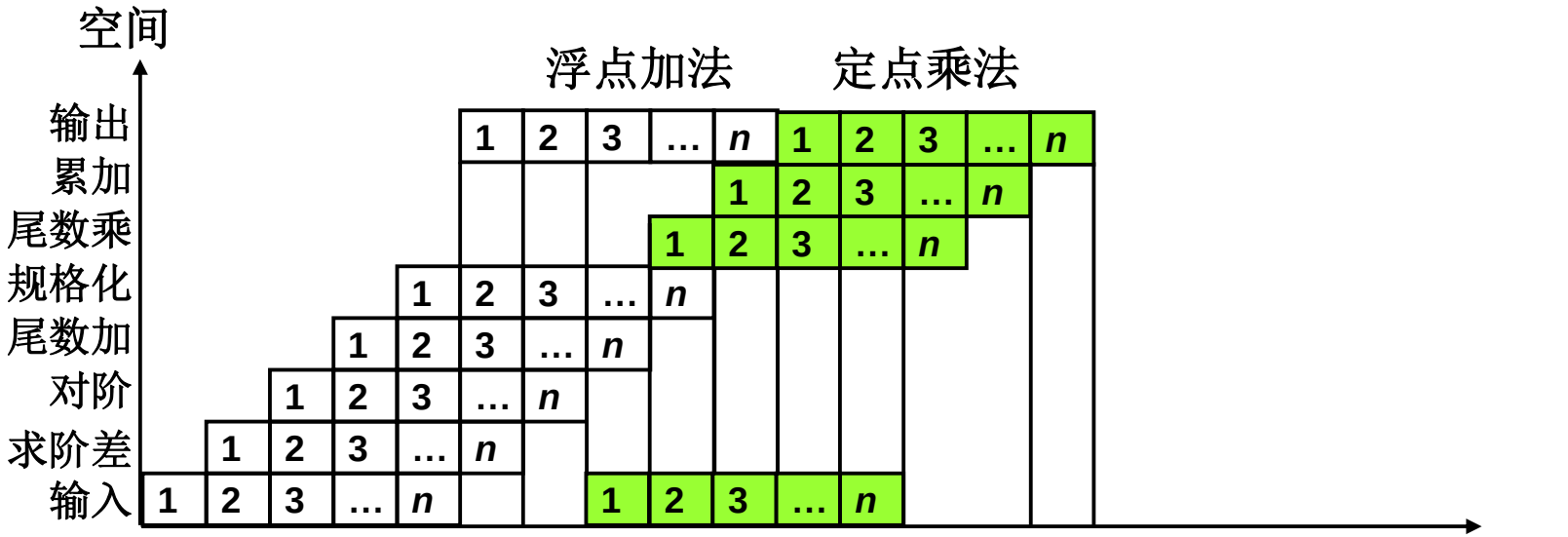
- 从功能部件的连接关系上对多功能流水线分类。
- 静态流水线
  - 在某一时间段内，各功能段只能按一种功能的连接流水。只有等流水线全部流空后，才能切换成按另一种功能来连接流水。
  - 适合于处理一串相同的运算操作。
- 动态流水线
  - 各功能段在同一时间内可根据需要进行连接流水。当某些段正在实现某种运算时，另一些段却在实现另一种运算。
  - 能提高流水的吞吐率和设备的利用率，但控制复杂，成本高。


1 输 入
2 求 阶 差
3 对 阶
4 相 加
5 规格化
6 相 乘
7 累 加
8 输 出
（a）分段
1 输 入
2 求 阶 差
3 对 阶
4 相 加
5 规格化
6
7
8 输 出
（b）浮点连接
1 输 入
2
3
4
5
6 相 乘
7 累 加
8 输 出
（c）定乘连接

静态/动态



静态流水线时空图

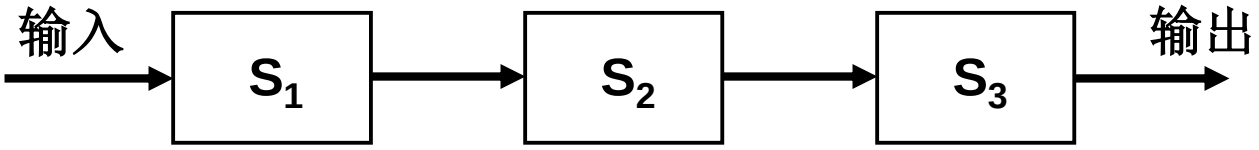


动态流水线时空图

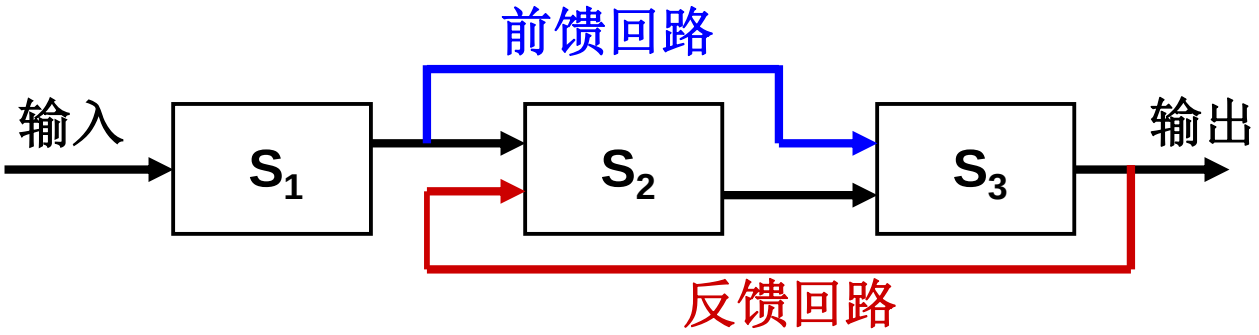
线性/非线性

- 按流水线各功能段之间是否有反馈回路分类。
- 线性流水线
  - 各功能段串行连接，没有反馈回路；
  - 各个段只经过一次。
- 非线性流水线
  - 各功能段不仅有串行连接，还有反馈回路；
  - 一个任务需要多次经过或越过某些段；
  - 其中一个重要的问题是流水线的调度：即何时向流水线输入新的任务，使此任务经过流水线各功能段时，不会与先前进入的任务争用功能段。

线性/非线性



一种简单的线性流水线



一种简单的非线性流水线

标量/向量

- 按机器具有的数据表示分类。
- 标量流水处理机
  - 没有向量数据表示，仅对标量数据进行流水处理。
- 向量流水处理机
  - 具有向量数据表示，能对向量、数组等进行流水处理；
  - 它是向量数据表示和流水技术的结合。

顺序/乱序

- 按照任务流出顺序与流入顺序是否相同分类。
- 顺序流水线
  - 任务流出顺序与任务流入顺序相同的流水线。
  - 顺序流水线又称为同步流水线。
- 乱序流水线
  - 任务流出顺序与任务流入顺序不同的流水线。
  - 乱序流水线又称为无序流水线、错序流水线或异步流水线等。

5.2.2 流水线处理机的主要性能

主要性能指标：3个

- 吞吐率
- 加速比
- 效率

吞吐率

- **定义：** 流水线单位时间内能处理的指令条数或能输出的结果数。

$$TP = \frac{n}{T_K}$$

n：任务数。

T_k：处理完成 n 个任务所用的时间。

吞吐率

- ①最大吞吐率TP_{max}
  - 指流水线单位时间内能处理的最多指令条数或能输出的最多结果数。
  - 显然，只有在流水线满负荷工作、达到稳定状态后才能达到。
- ②实际吞吐率TP
  - 指流水线单位时间内实际处理的指令条数或实际输出的结果数。

线性流水线吞吐率 – 最大吞吐率

- 若流水线各段的时间相等，均为Δt_0，则：

$$TP_{max} = 1/\ \Delta t_0$$

- 若流水线各段时间不等，第 *i* 段时间为Δt_i，则：

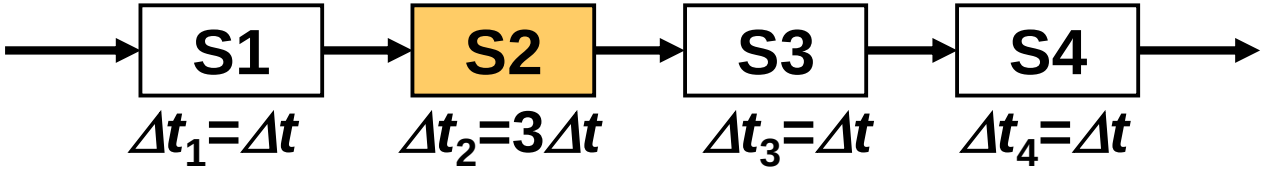
$$TP_{max} = 1/\ \max\{\ \Delta t_i\ \}$$

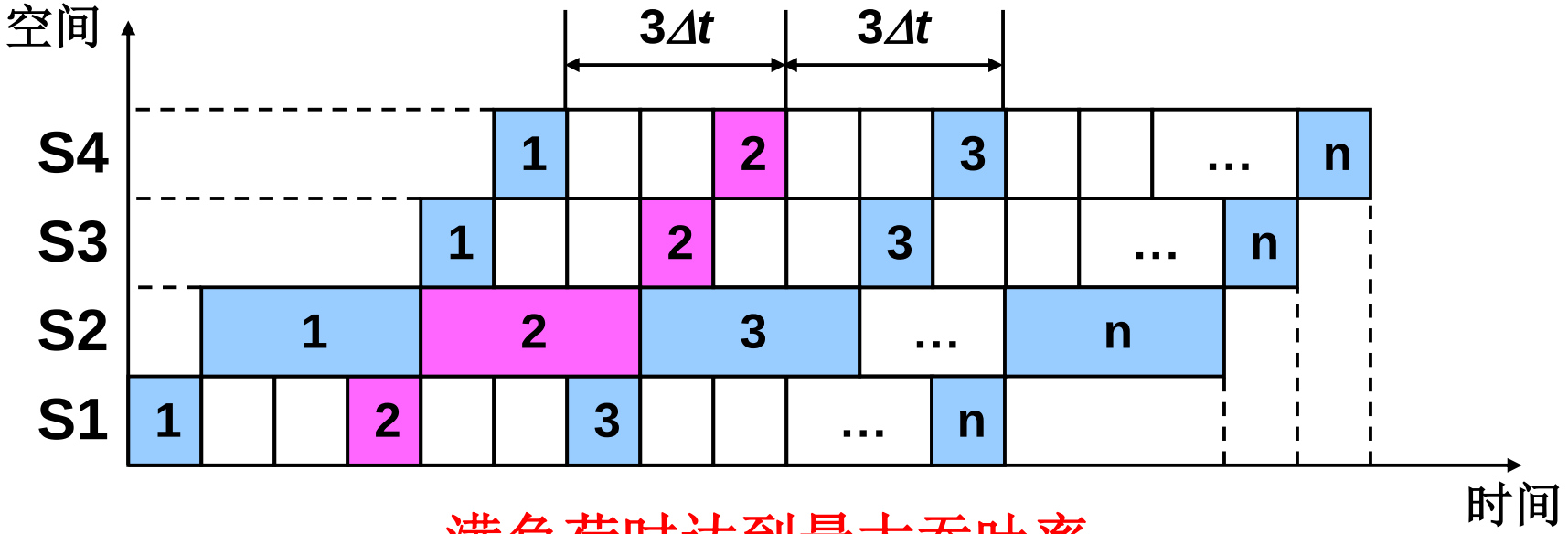
它受限于流水线中最慢子过程所需要的时间。

将流水线中经过时间最长的子过程为瓶颈子过程。

为了提高流水线的最大吞吐率，必须找出瓶颈子过程，并设法消除。

线性流水线吞吐率 – 最大吞吐率





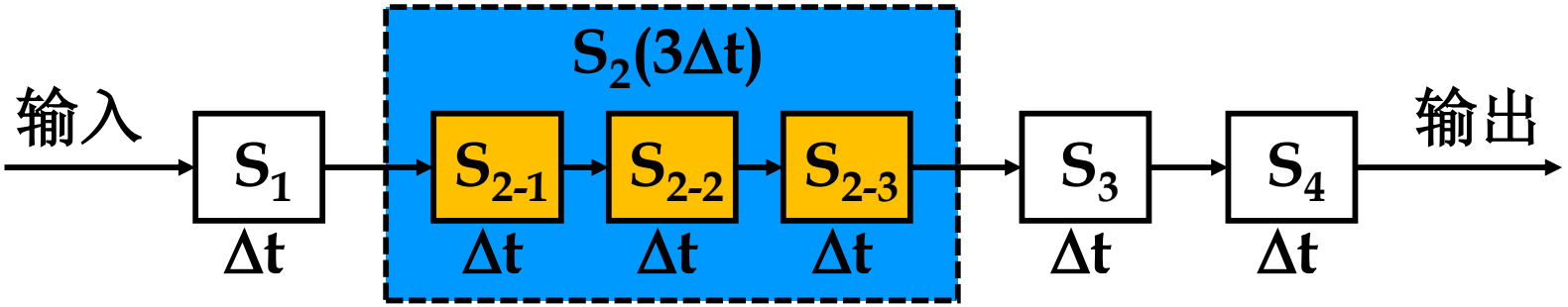
满负荷时达到最大吞吐率。

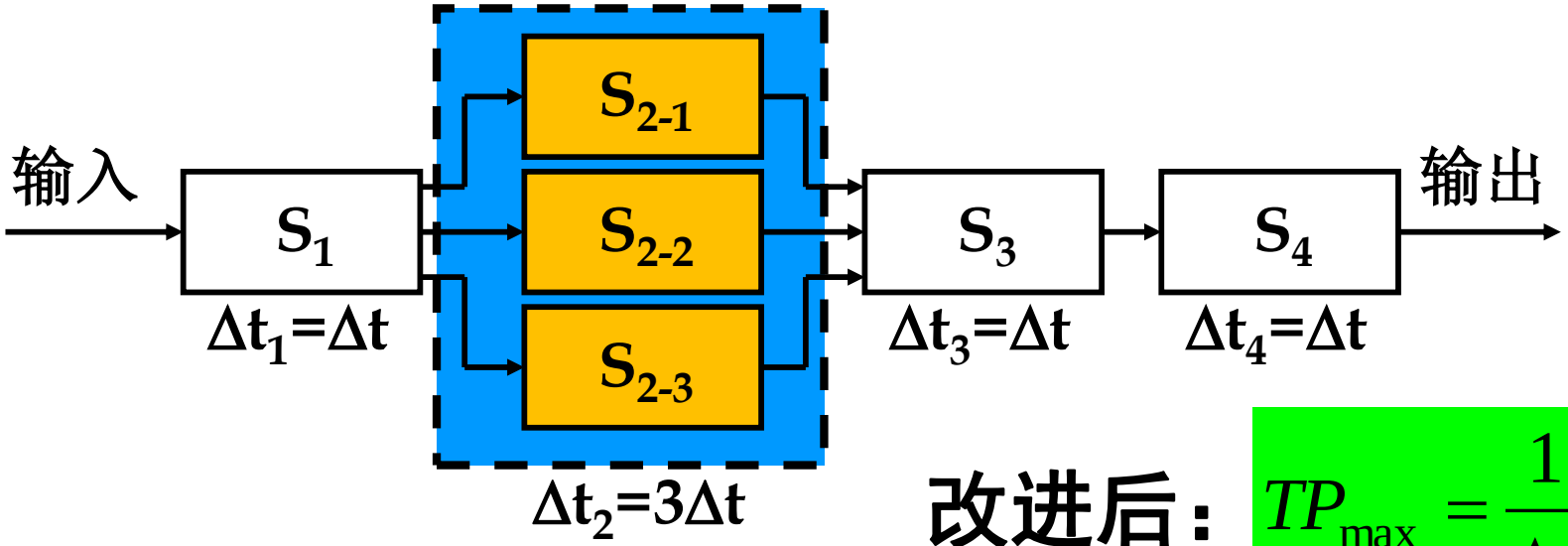
S2为瓶颈段。每隔 $3\Delta t$ 流出一个结果。$TP_{max} = 1/3\Delta t$

线性流水线吞吐率 – 最大吞吐率

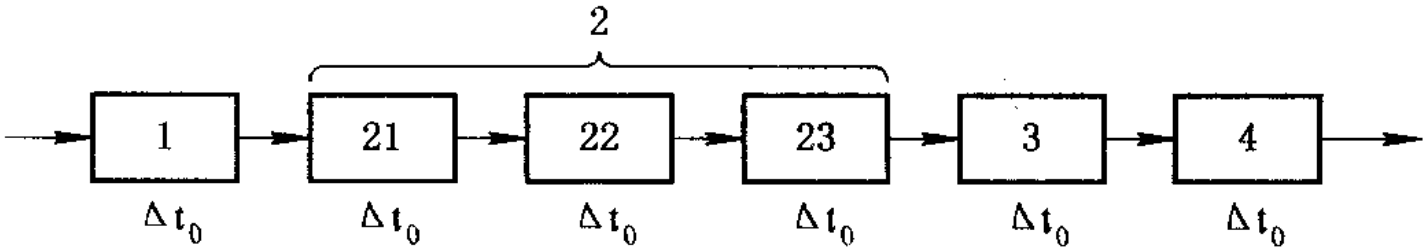
- 消除瓶颈子过程的方法有：
  - **细分瓶颈过程：**使各功能段处理时间尽可能相等。
  - **重复设置瓶颈功能段，并使它们交叉并行工作。**若瓶颈子过程不能再细分，可以采用此方法。但此方法的设备量增加，需要解决好并行子过程的任务分配和同步。

线性流水线吞吐率 – 最大吞吐率

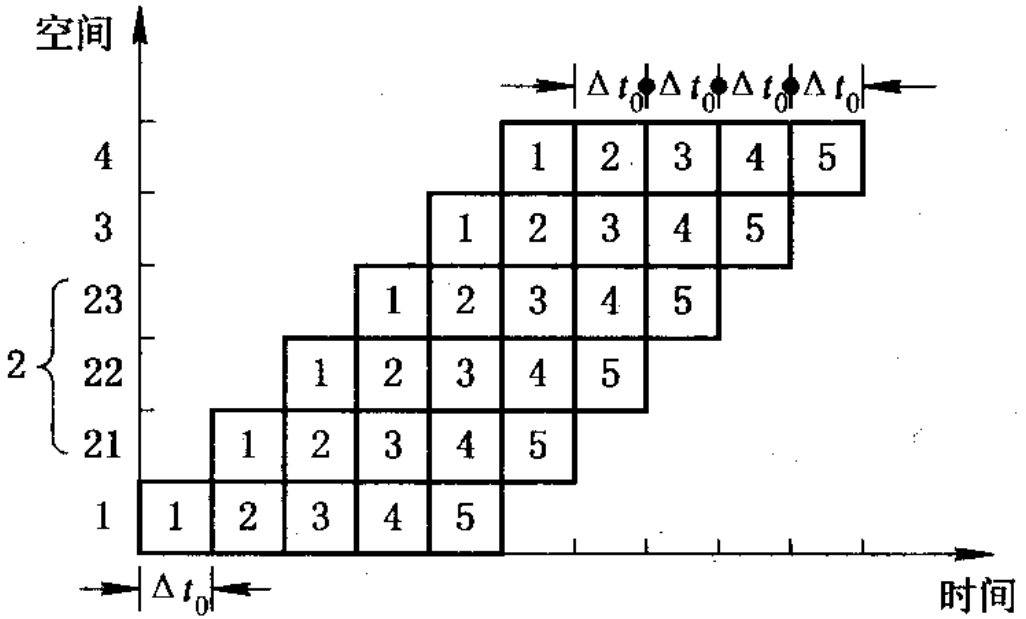




改进后： $TP_{\max} = \dfrac{1}{\Delta t}$

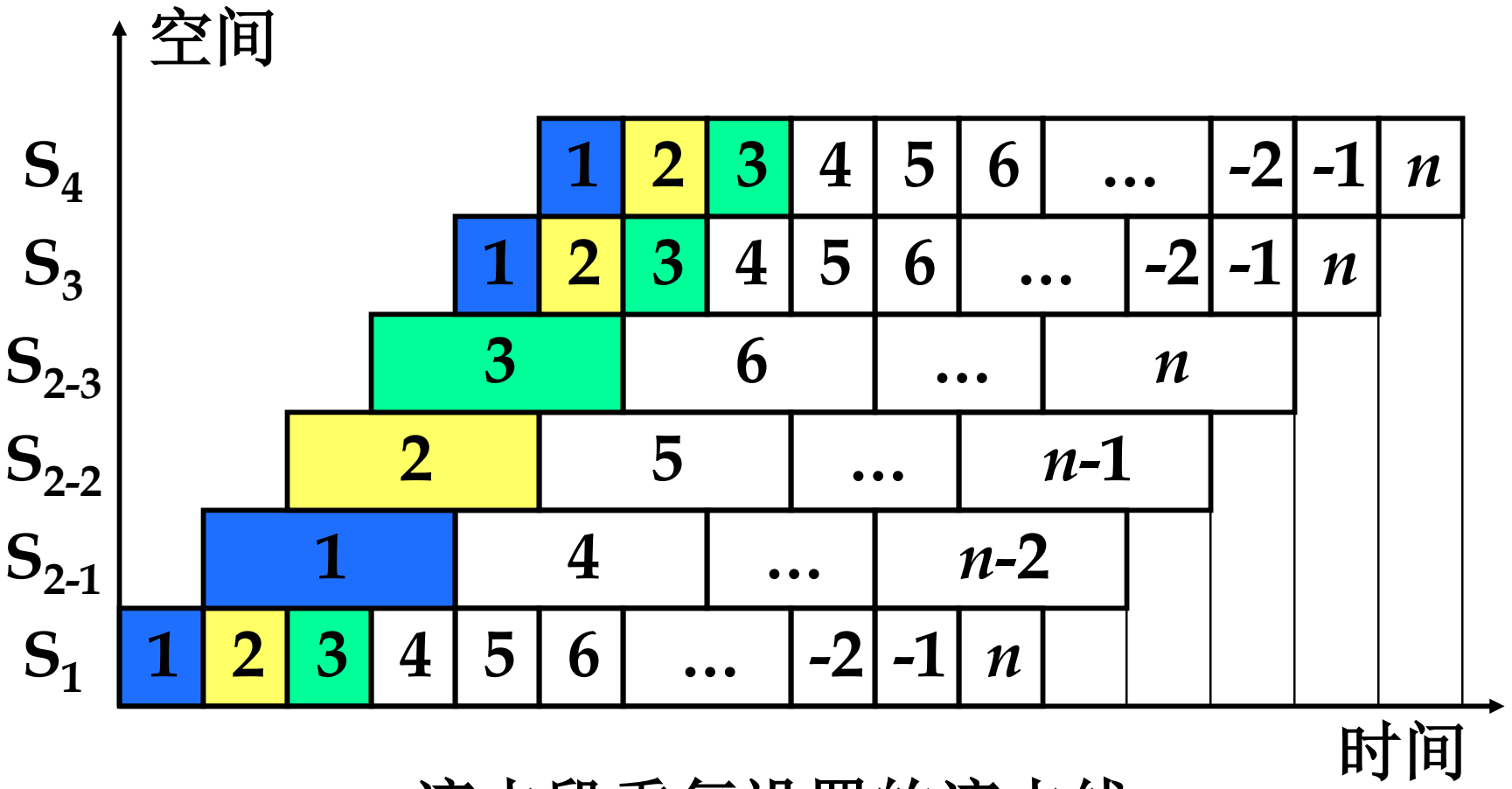

2
1
21
22
23
3
4
Δt0
Δt0
Δt0
Δt0
Δt0
Δt0

(a)


空间
Δt0 Δt0 Δt0 Δt0
4
3
23
22
21
1
2
1
1 2 3 4 5
Δt0
时间

(b)

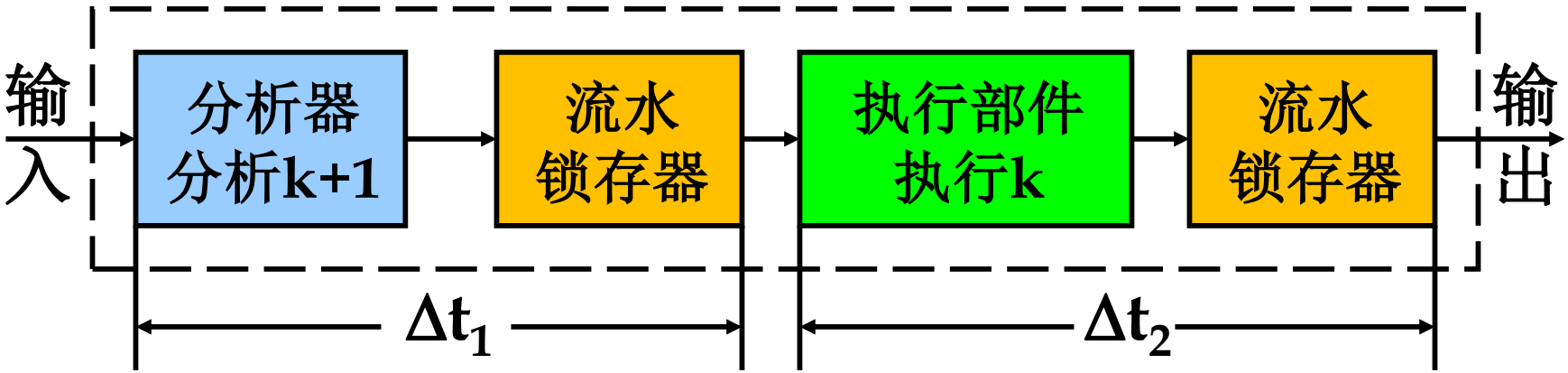
线性流水线吞吐率 – 最大吞吐率



流水段重复设置的流水线

最大吞吐率

- 各子过程所经过的时间会有不同。
- 为平滑各子过程速度差异，一般在子部件设置**高速接口锁存器**。
- 子过程越细分，锁存器数量越多，因此而增大的延迟就越长，从而影响最大吞吐率的提高。



线性流水线吞吐率 – 实际吞吐率

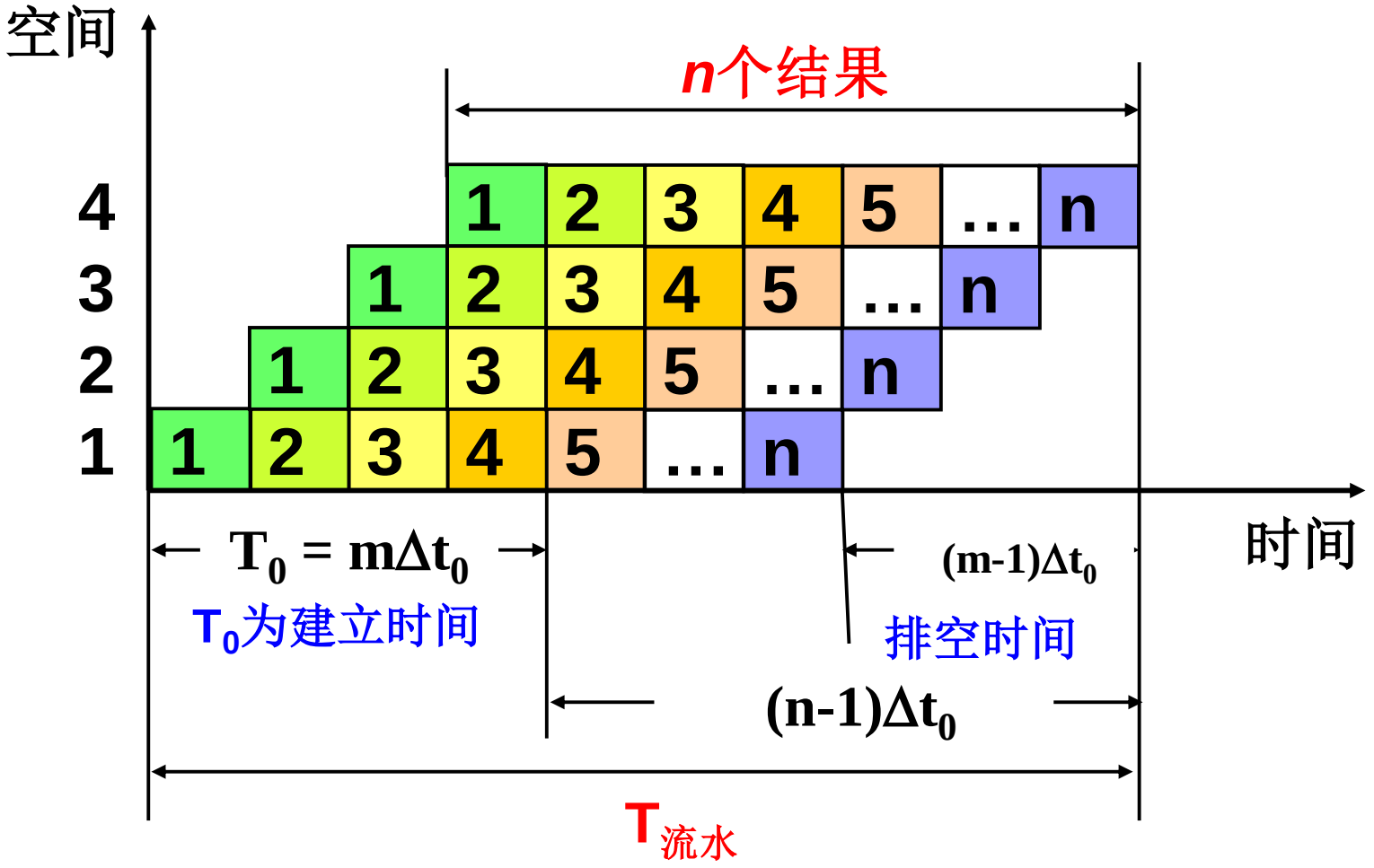
- 若流水线在 $T_{流水}$ 时间内处理了 n 条指令，

$$TP = n / T_{流水}$$

- 通常， $TP < TP_{max}$ ， 因为流水线开始时，总要有一段建立时间。
- **建立时间T0：** 流水线从开始启动工作到流出第一个结果所需要的时间。
- **排空时间T1：** 最后一个任务进入流水线到结果流出所需要的时间。

线性流水线吞吐率 – 实际吞吐率

各段的时间相等，均为Δt_0



线性流水线吞吐率 – 实际吞吐率

- 若流水线各段的时间相等，均为Δt_0，则：

$$TP = \frac{n}{m\Delta t_0 + (n-1)\Delta t_0} = \frac{1}{\Delta t_0(1+\frac{m-1}{n})} = \frac{TP_{\max}}{1+\frac{m-1}{n}}$$

- 若流水线各段的时间不相等，则：

$$TP = \frac{n}{\sum_{i=1}^{m}\Delta t_i + (n-1)\cdot \max(\Delta t_1, \Delta t_2, \cdots, \Delta t_m)}$$

线性流水线吞吐率 – 实际吞吐率

- 流水线的实际吞吐率小于最大吞吐率，它除了与每个段的时间有关外，还与流水线的段数 **m** 以及输入到流水线中的任务数 **n** 等有关。
- 只有当 $n >> m$ 时，才有 $TP \approx TP_{max}$。

加速比

- **定义：** 指流水线速度与等效的非流水线速度之比。

加速比 $S_p = T_{非流水} / T_{流水}$

对于线性流水线，若流水线为m段，每段时间均为 Δt_0，则：

$T_{非流水} = nm\,\Delta t_0$，$T_{流水} = m\,\Delta t_0 + (n-1)\,\Delta t_0$

$$S_p = \frac{mn}{m+n-1} = \frac{m}{1+\frac{m-1}{n}}$$

$$S_{p\max} = \lim_{n\to\infty} \frac{m}{1+\frac{m-1}{n}} = m$$

思考： 流水线的段数愈多愈好？

加速比

- 对于线性流水线，若流水线为m段，每段时间不相等，则：

$$S_p = \frac{n \cdot \sum_{i=1}^{m} \Delta t_i}{\sum_{i=1}^{m} \Delta t_i + (n-1) \cdot \max(\Delta t_1, \Delta t_2, \cdots, \Delta t_m)}$$

效率

- **定义：**指流水线中的设备实际使用时间占整个运行时间之比，也称为流水线的设备时间利用率。
- 由于流水线有通过时间和排空时间，所以在连续完成 *n* 个任务的时间内，各段并不是满负荷地工作。

线性流水线效率

- 若各段时间相等，则各段的效率 η_i 相同：

$$\eta_1 = \eta_2 = \cdots = \eta_m = \frac{n\Delta t}{T_m} = \frac{n}{m+n-1}$$

分母 = 时—空图中m段和总时间T所围成的面积
分子 = 时—空图中n个任务实际占用的总面积

- 整条流水线的效率为：

$$\eta = \frac{\eta_1 + \eta_2 + \cdots + \eta_m}{m} = \frac{m\eta_1}{m} = \frac{mn\Delta t}{mT_m}$$

或

$$\eta = \frac{n}{m+n-1}$$

与吞吐率的关系为：$\eta = TP \bullet \Delta t_0$

- 最高效率为：

$$\eta_{max} = \lim_{n\to\infty} \frac{n}{m+n-1} = 1$$

当 $n >> m$ 时，$\eta \approx 1$。

线性流水线效率

- 若各段时间不相等，则各段的效率 η_i 不相同
- 整条流水线的效率＝？

从时—空图上来看，效率实际上就是 n 个任务所占用的时空区面积与 m 个段所占用的总的时空区的面积之比。

$$\eta = \frac{n\text{个任务实际占用的时空区}}{m\text{个段总的时空区}}$$

- 若各段时间不相等

$$\eta = \frac{n\text{个任务实际占用的时空区}}{m\text{个段总的时空区}} = \frac{n \cdot \sum_{i=1}^{m} \Delta t_i}{m \cdot \left[\sum_{i=1}^{m} \Delta t_i + (n-1)\Delta t\right.}$$

5.2.2 流水线处理机的主要性能

- 问题：流水线功能段的数量应为多少？
- 增加流水线段数时，流水线的吞吐率和加速比都能提高。但随着段数的增多，总延迟时间将增加，流水线的价格也会增加。因此要综合考虑各方面的因素，根据性能价格比来选择流水线最佳段数。
- 目前，一般处理机中的流水线段数在3到12之间，极少有超过15段的流水线。
- 一般把8段或超过8段的流水线称为超流水线，采用8段以上流水线的处理机有时也称为超流水线处理机。

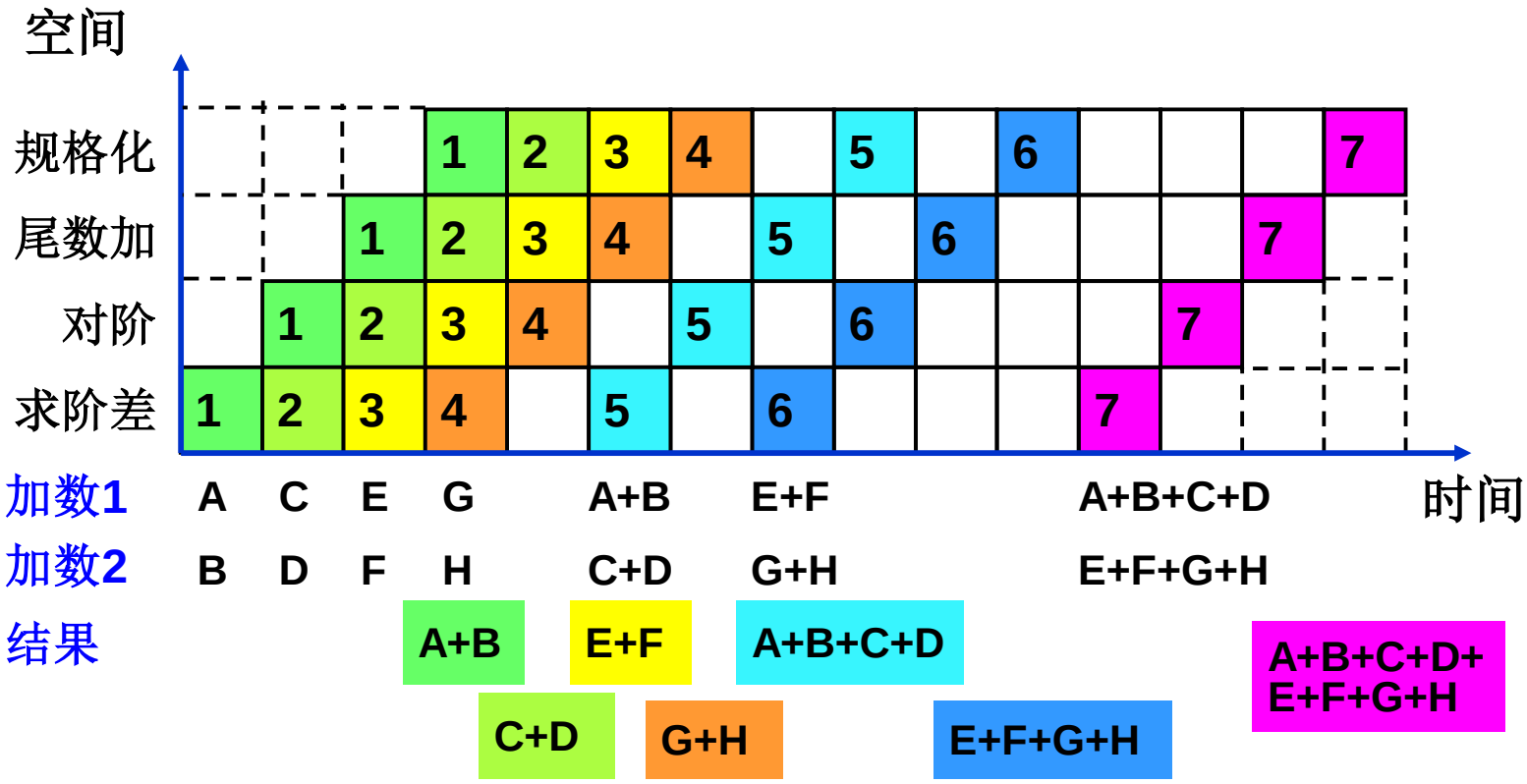
流水线性能举例

- 例5-2：用一条4段浮点加法器流水线求8个浮点数的和：Z＝A＋B＋C＋D＋E＋F＋G＋H。计算其吞吐率、加速比和效率。4段浮点加法器分别为：求阶差、对阶、尾数加和规格化。各段处理时间相等，都为Δt。

流水线性能举例

- 例5-2 解：拆为多个加法，以便连续进行计算：

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



流水线性能举例

- 例5-2 解：进行了7次浮点加法，共用了15个时钟周期。

$$\text{吞吐率 } TP = \frac{n}{T_{\text{流水}}} = \frac{7}{15\,\Delta t}$$

$$\text{加速比 } S = \frac{T_{\text{非流水}}}{T_{\text{流水}}} = \frac{7 \times 4\,\Delta t}{15\,\Delta t} = 1.87$$

$$\text{效率 } E = \frac{\text{7个加法的时空区}}{\text{4个段总的时空区}} = \frac{7 \times 4\,\Delta t}{4 \times 15\,\Delta t} = 47\%$$

流水线性能举例

- 例1：在DLX的非流水实现和基本流水线中，5个功能单元的执行时间分别为：10/8/10/10/7ns。流水线额外开销为1ns，求相对于非流水指令实现而言，基本DLX流水线的加速比是多少？
- 解：

$$T_{非流水} = 10 + 8 + 10 + 10 + 7 = 45ns$$

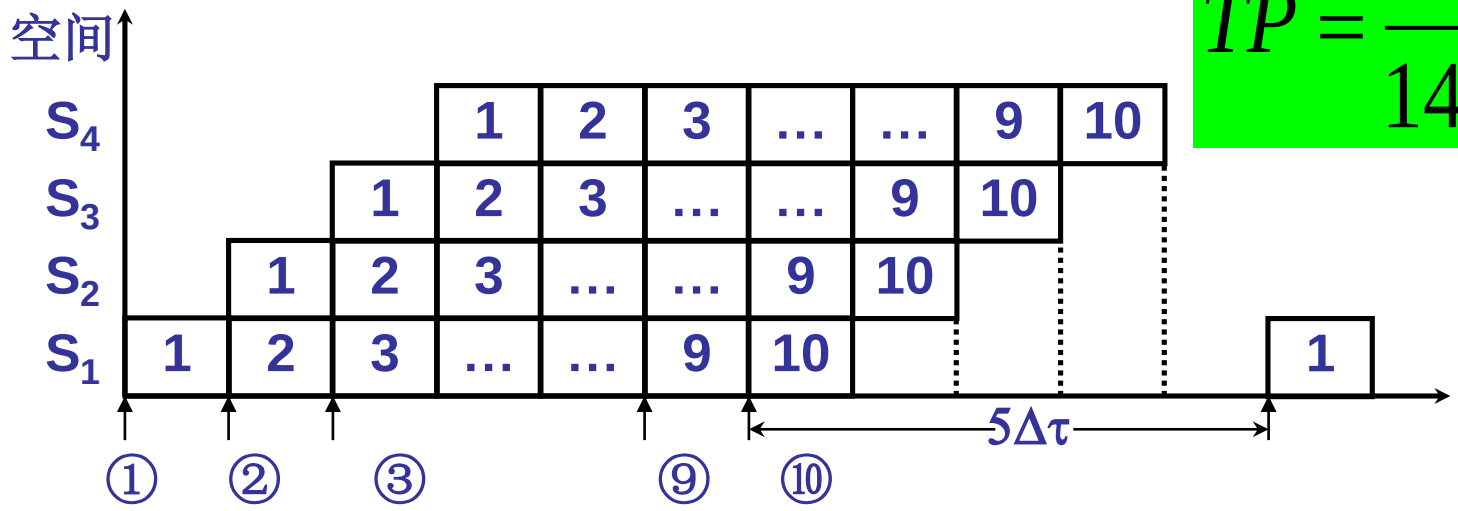
$$T_{流水} = 10ns + 1ns = 11ns$$

$$加速比S = 45/11 \approx 4.1$$

流水线性能举例

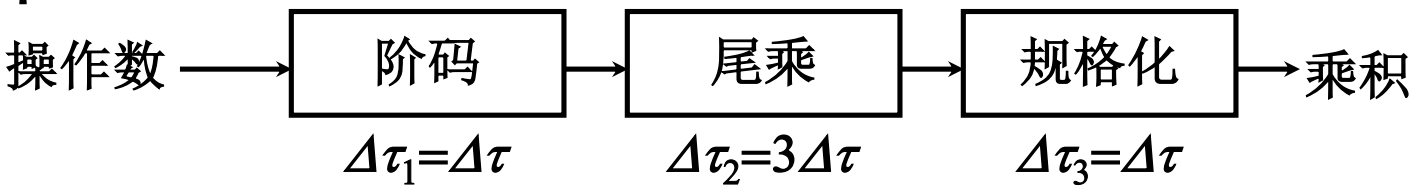
- 例2：流水线由4个功能部件组成，每个功能部件的延迟时间为Δt。当输入10个数据后，间歇5Δt，又输入10个数据，如此周期地工作。求此时流水线的吞吐率，并画出时空图。
- 解：

$$TP = \frac{10}{14\Delta t} = \frac{5}{7\Delta t}$$

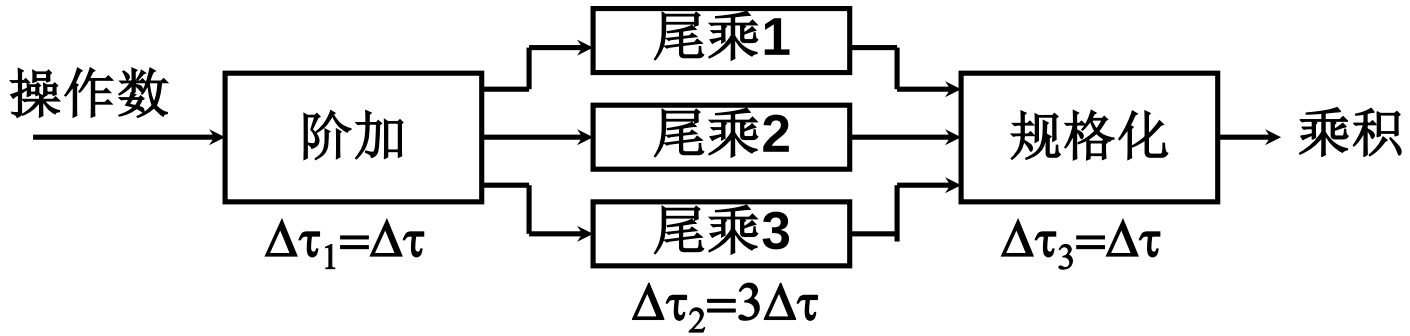


流水线性能举例

- 例3：有一个浮点乘流水线如图a所示，其乘积可直接返回输入端或暂存于相应缓冲寄存器中，画出实现A*B*C*D的时空图以及输入端的变化，并求出该流水线的吞吐率和效率；当流水线改为图b形式实现同一计算时，求该流水线吞吐率和效率。



图a



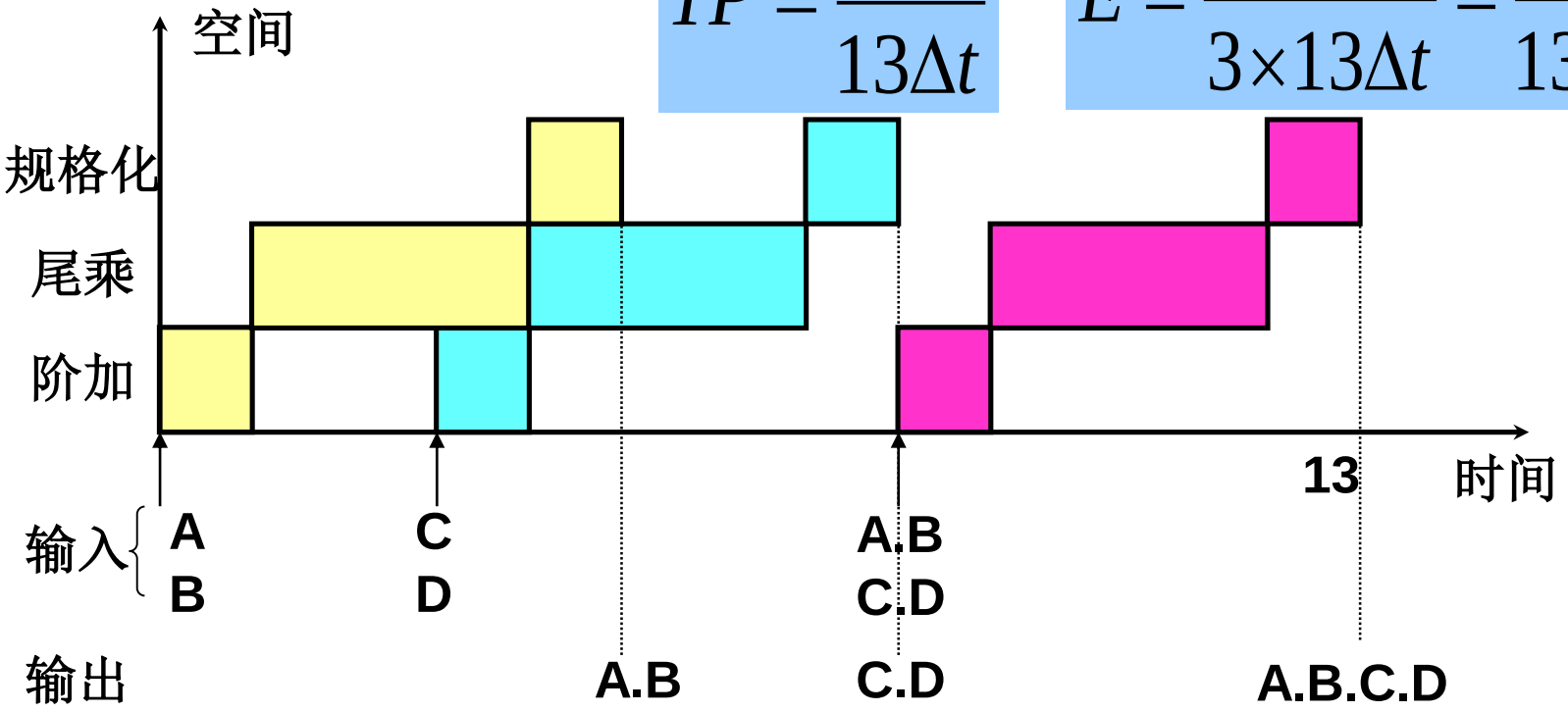
图b

流水线性能举例

- 解：为减少运算过程操作数相关，将A*B*C*D改成(A*B)*(C*D)。图a

$$TP = \frac{3}{13\Delta t}$$

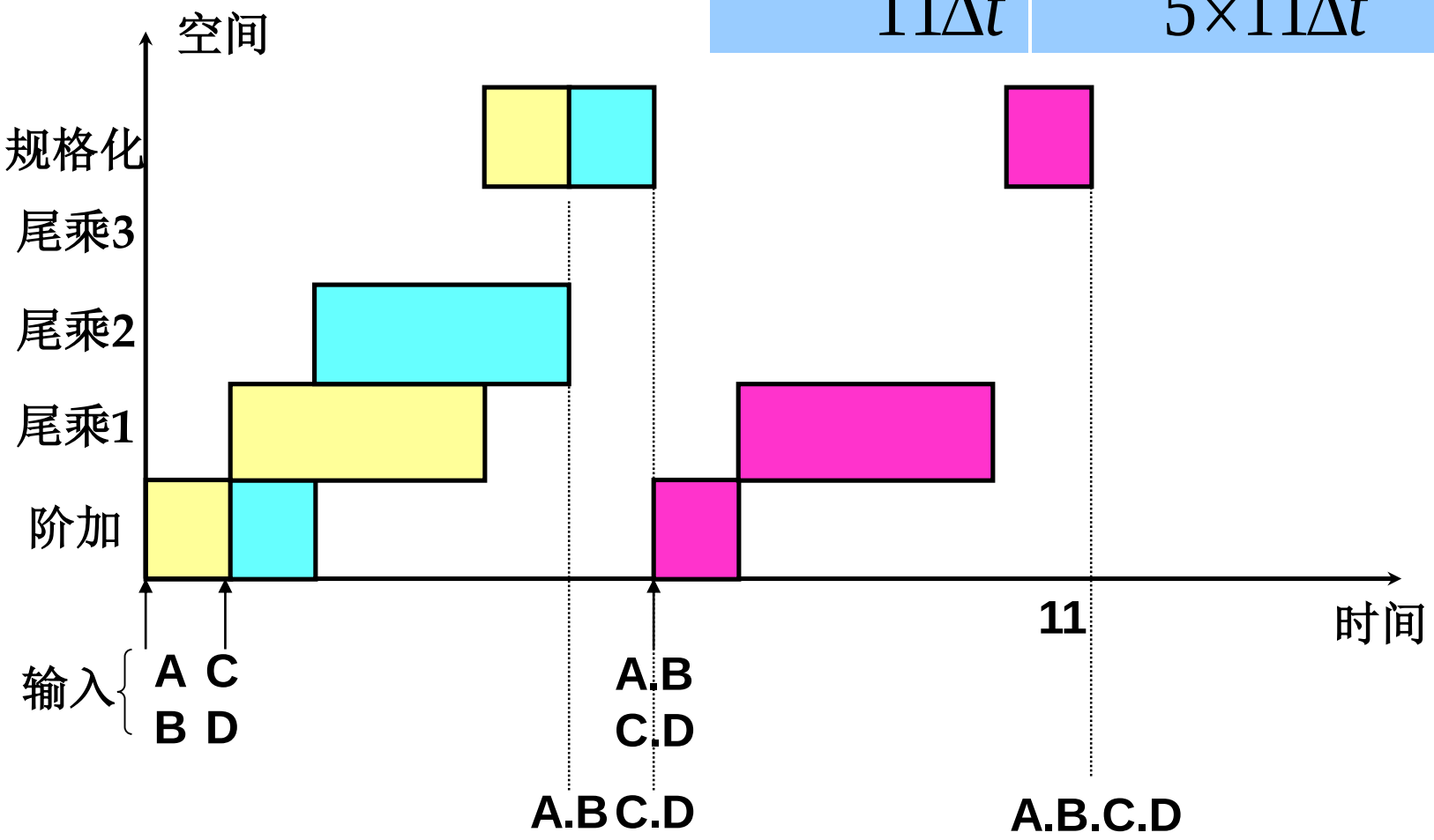
$$E = \frac{3 \times 5\Delta t}{3 \times 13\Delta t} = \frac{5}{13}$$



流水线性能举例

- 解：图b

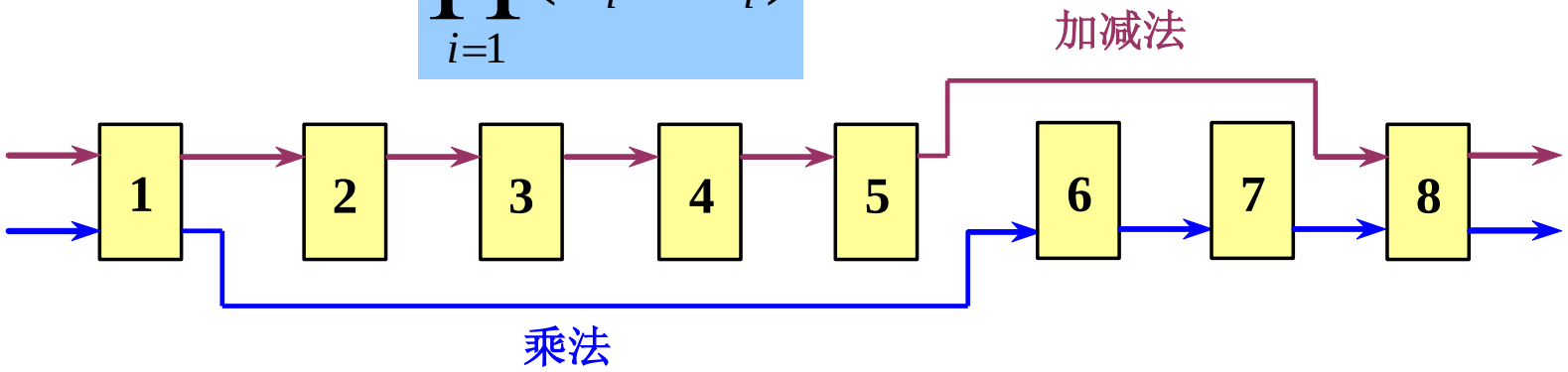
$$TP = \frac{3}{11\Delta t} \qquad E = \frac{3 \times 5\Delta t}{5 \times 11\Delta t} = \frac{3}{11}$$



流水线性能举例

- 例4：设在下图所示的静态流水线上计算：

$$\prod_{i=1}^{4}(A_i + B_i)$$



- 流水线的输出可以直接返回输入端或暂存于相应的流水寄存器中，试计算其吞吐率、加速比和效率。

流水线性能举例

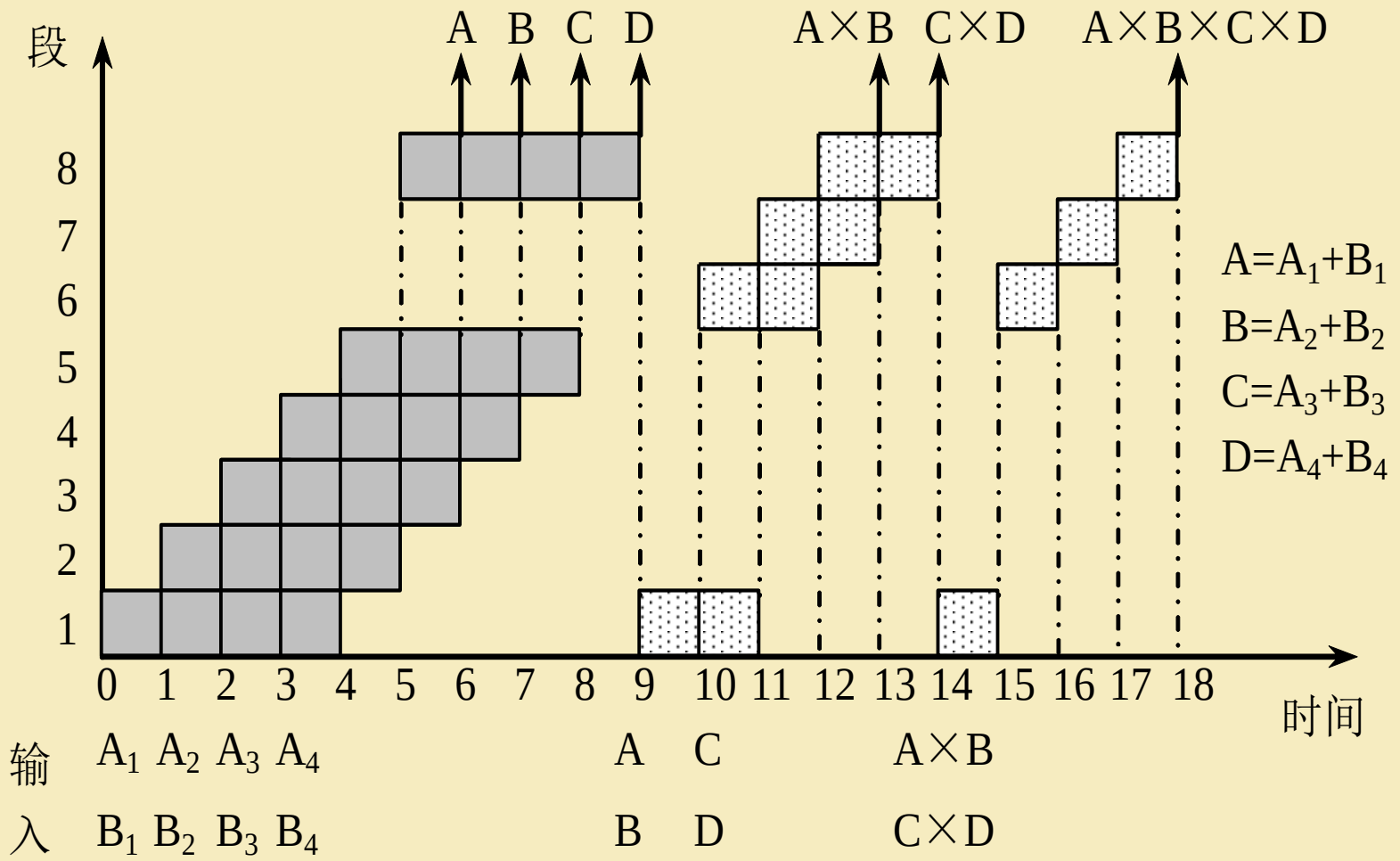
- 例4：解

（1）选择适合于流水线工作的算法

- 先计算A1+B1、A2+B2、A3+B3和A4+B4；
- 再计算(A1+B1)×(A2+B2)和(A3+B3)×(A4+B4)；
- 然后求总的乘积结果。

（2）画出时空图

流水线性能举例


段
8
7
6
5
4
3
2
1
A B C D
A×B C×D
A×B×C×D
A=A1+B1
B=A2+B2
C=A3+B3
D=A4+B4
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
时间
输入
A1 A2 A3 A4
B1 B2 B3 B4
A C
B D
A×B
C×D

流水线性能举例

- 例4：解

（3）计算性能

在18个Δt时间中，给出了7个结果。吞吐率为：

$$TP = \frac{7}{18\Delta t}$$

不用流水线，由于一次求和需6△t，一次求积需4△t，则产生上述7个结果共需（4×6+3×4）△t = 36△t，加速比为：

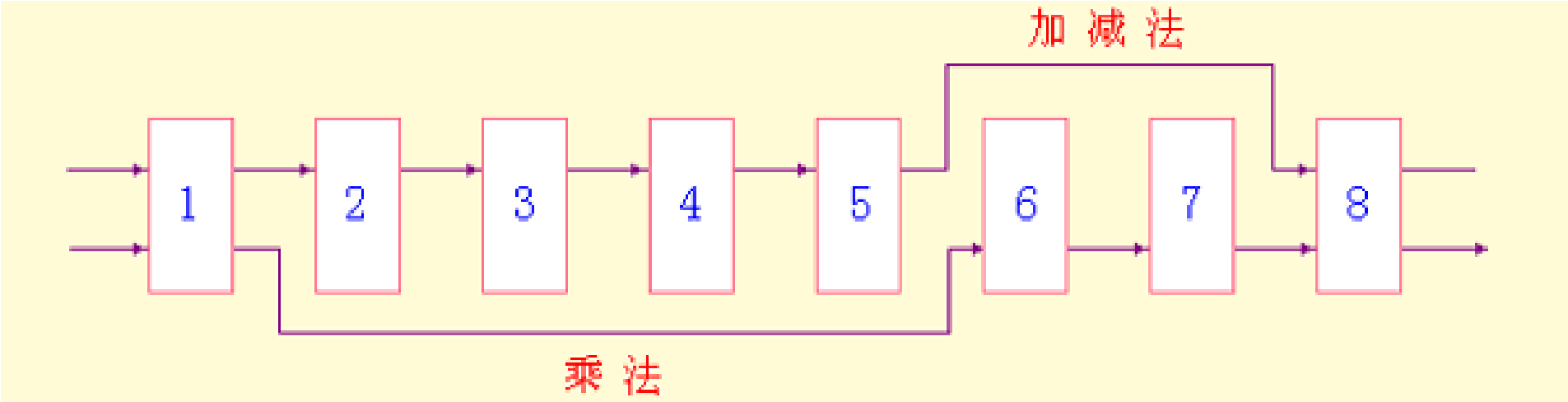
$$S = \frac{36\Delta t}{18\Delta t} = 2$$

流水线的效率：

$$\eta = \frac{4\times 6 + 3\times 4}{8\times 18} = 0.25$$

流水线性能举例

- 例5：在静态流水线上计算 $\Sigma A_i B_i$ ， i=1-4。
  求：吞吐率，加速比，效率。



流水线性能举例

- 例5：解

（1）选择适合于流水线工作的算法

（2）画出时空图

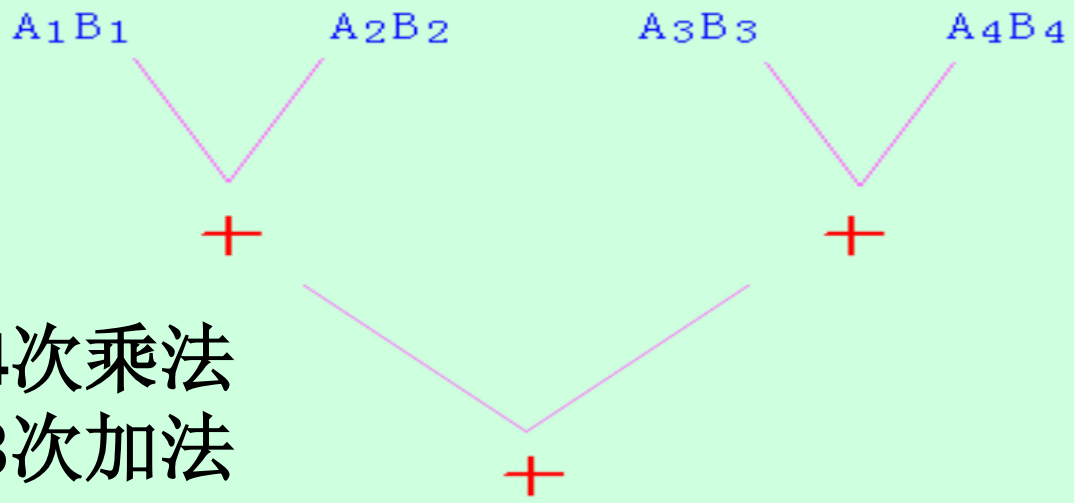
（3）计算性能

流水线性能举例

- 例5：（1）选择适合于流水线工作的算法

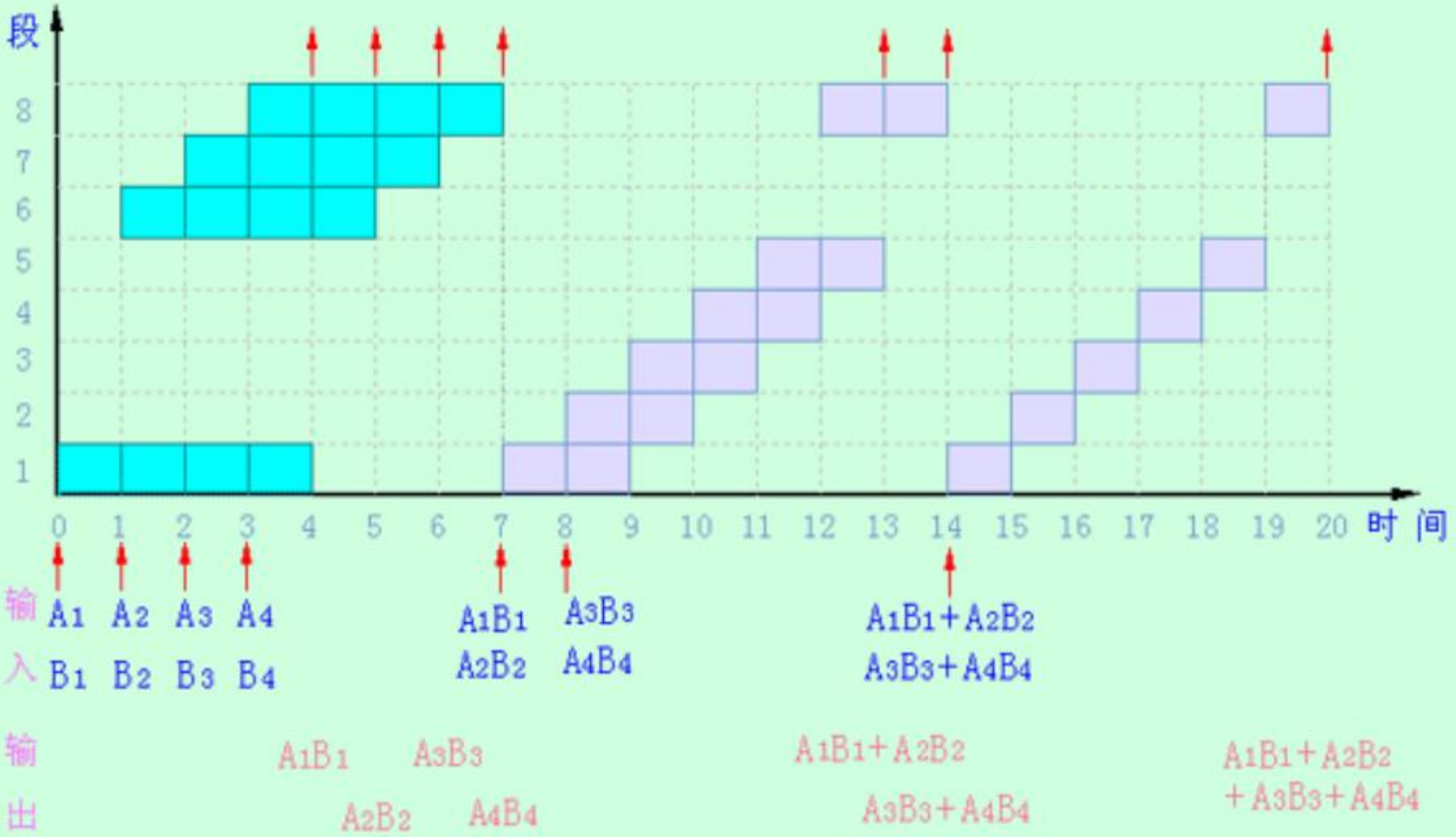
$\sum_{i=1}^{4} A_i B_i$ 的计算过程

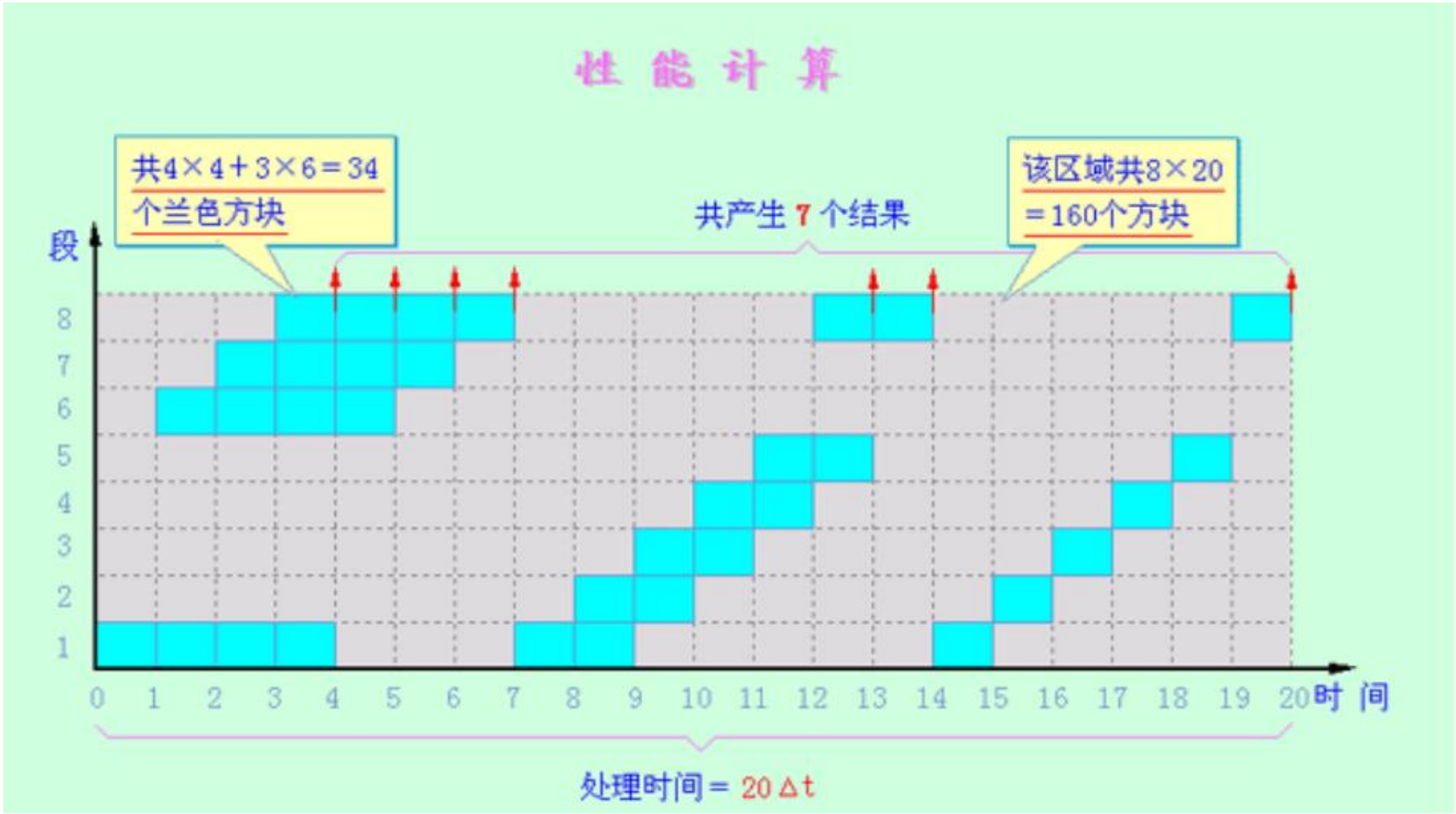
$$\sum_{i=1}^{4} A_i B_i = A_1B_1 + A_2B_2 + A_3B_3 + A_4B_4$$



4次乘法
3次加法

时 空 图


段
8
7
6
5
4
3
2
1
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
时 间
输入
A1 A2 A3 A4
B1 B2 B3 B4
A1B1 A3B3
A2B2 A4B4
A1B1+A2B2
A3B3+A4B4
输出
A1B1 A3B3
A2B2 A4B4
A1B1+A2B2
A3B3+A4B4
A1B1+A2B2
+A3B3+A4B4



吞吐率TP = $7/20 \cdot \Delta t_0$

加速比 S = $34 \cdot \Delta t_0 / 20 \cdot \Delta t_0 = 1.7$

效率η=(4×4+3×6)/(8×20)=0.21

讨论：为何两题的流水线效率较低？

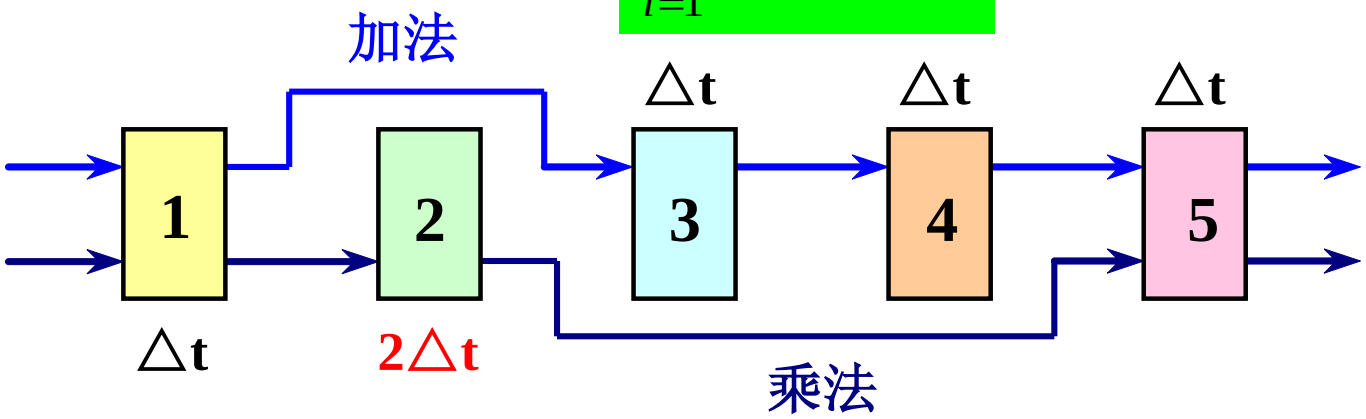
流水线性能举例

- 讨论：主要原因
  - 多功能流水线在做某一种运算时，总有一些段是空闲的。
  - 静态流水线在进行功能切换时，要等前一种运算全部流出流水线后才能进行后面的运算。
  - 运算之间存在关联，后面有些运算要用到前面运算的结果。
  - 流水线的工作过程有建立与排空部分。

流水线性能举例

- 例6：有一条**动态多功能流水线**由5段组成，加法用1、3、4、5段，乘法用1、2、5段，第2段的时间为2△t，其余各段时间均为△t，而且流水线的输出可以直接返回输入端或暂存于相应的流水寄存器中。若在该流水线进行以下计算，试计算其吞吐率、加速比和效率。

$$\sum_{i=1}^{4}(A_i \times B_i)$$



流水线性能举例

- 例6：解

（1）选择适合于流水线工作的

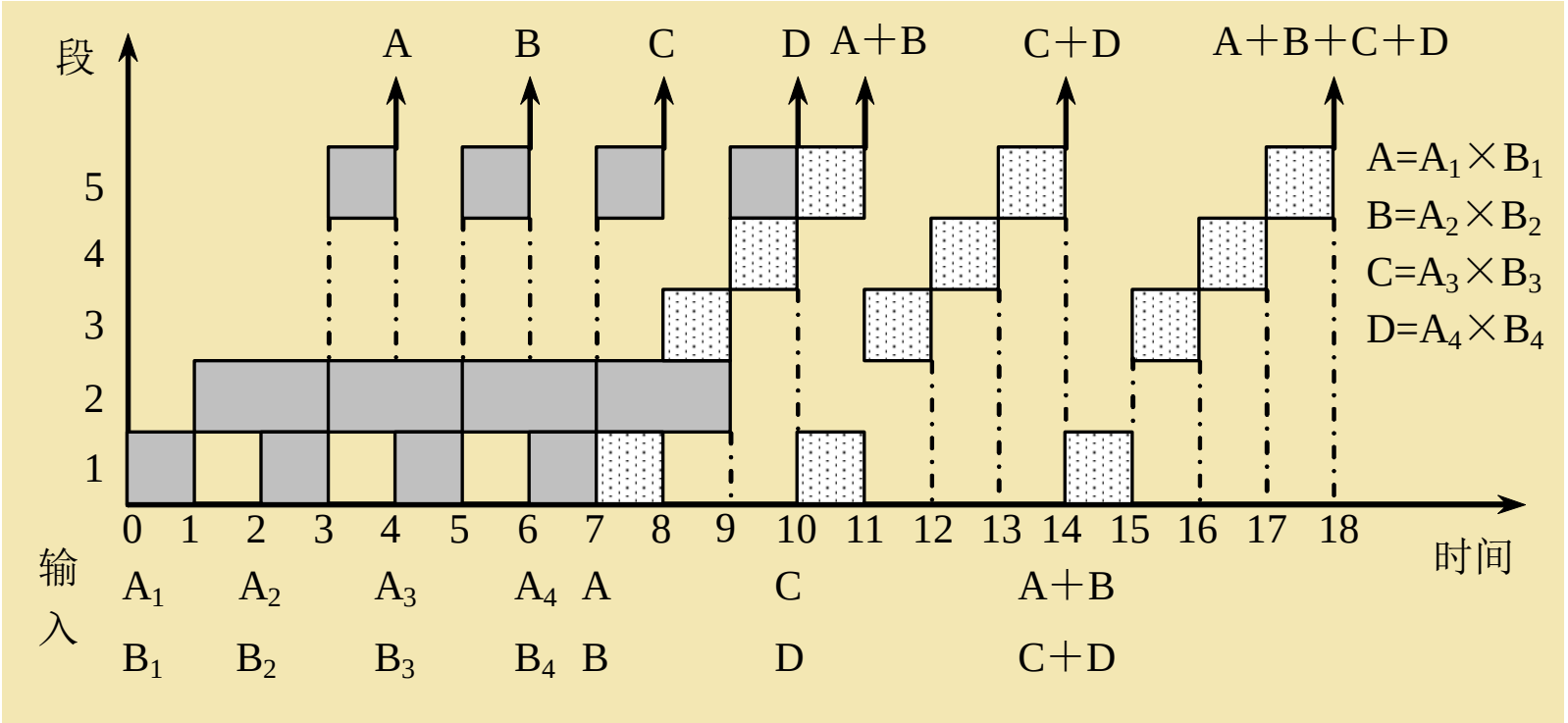
- 应先计算A1×B1、A2×B2、A3×B3和A4×B4；
- 再计算(A1×B1)＋(A2×B2) 和(A3×B3)＋(A4×B4)；
- 然后求总的累加结果。

（2）画出时空图

（3）计算性能

流水线性能举例

- 例6：解



$$TP = \frac{7}{18\Delta t}$$

$$S = \frac{28\Delta t}{18\Delta t} \approx 1.56$$

$$E = \frac{4\times 4 + 3\times 4}{5\times 18} \approx 0.31$$

5.2.3 流水线调度

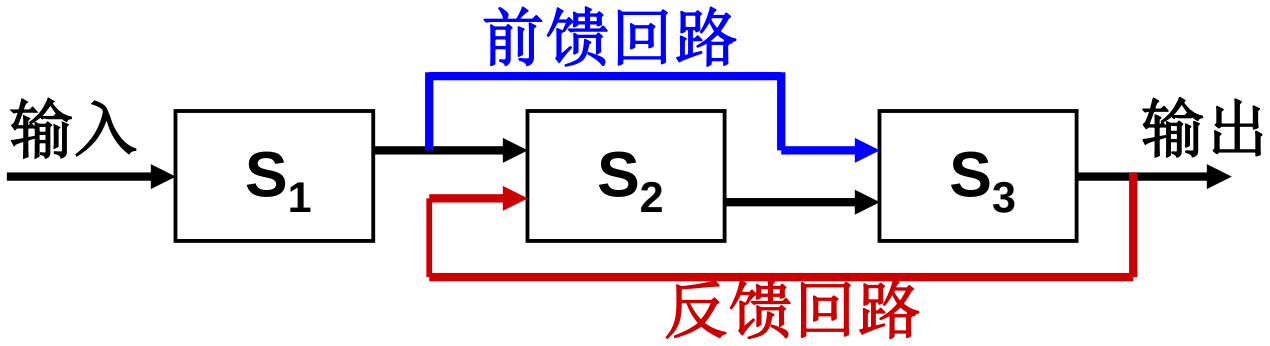
- 由于功能部件一般只有一套，所以难免会发生资源冲突。
- 资源冲突
  - 指几个任务争用一个功能段的现象。
- 因此，如何调度任务，使之合理、高效地利用流水线，使流水线的吞吐率和效率不致下降，是非常关键的。

1. 线性流水线的调度

- 线性流水线无反馈，每个任务只通过每个段一次，不会发生冲突，因此只需每隔Δt（或数个Δt）输入一个任务即可。
- 注意：需解决好其他相关问题！

2. 非线性流水线的调度

- 非线性流水线段间有反馈回路，一个任务在流水执行的全过程中，可能会多次通过同一段，或越过某些段。



一种简单的非线性流水线

2. 非线性流水线的调度

- 如果每拍向流水线输入任务，将会发生资源冲突。
  - 注意：可能还有其他相关问题！
- 问题：那么究竟间隔几拍向流水线输入任务，才能既不发生功能段使用冲突，又能使流水线有较高的吞吐率呢？
- 这就是流水线调度要解决的问题！

2. 非线性流水线的调度

- 在一般情况下，间隔的拍数当然应该越小越好，而且往往不是一个常数。
- 非线性流水线的调度的任务：

  找出一个最小的循环周期，按照这个周期向流水线输入新任务，流水线的各个功能段都不会发生冲突，而且流水线的吞吐率和效率最高。

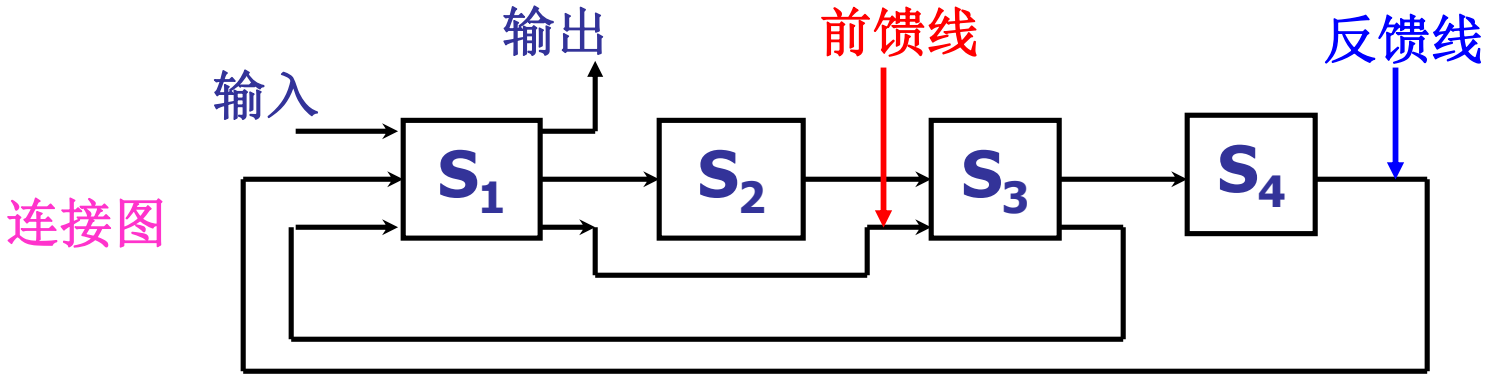
2. 非线性流水线的调度

- 二维预约表法(Reservation Table)
  - E. S. Davidson 于1971年提出。
  - 假设有一个由 K 段组成的单功能非线性流水线，每个任务通过流水线需要 N 拍。
  - 利用类似时—空图的方法，可以得到一个任务使用流水线各段的时间关系表（二维预约表）。
  - 如果任务在第 n 拍用到第 K 段，就在相应第 n 列和第 K 行的交叉点处用 √ 表示。

2. 非线性流水线的调度

- 现有一个由4个功能段组成非线性流水线，每个任务通过流水线需要7拍。该流水线的二维预约表如下：

	1	2	3	4	5	6	7
S1	√			√			√
S2		√			√		
S3		√			√		
S4			√				

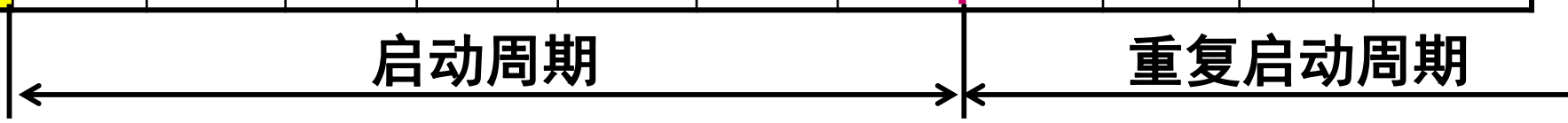


2. 非线性流水线的调度

- **流水线的启动距离：**
  - 也称为**等待时间**
  - 向流水线连续输入**两个任务之间的时间间隔**
- **流水线的冲突：**
  - 几个任务争用同一个流水段

2. 非线性流水线的调度

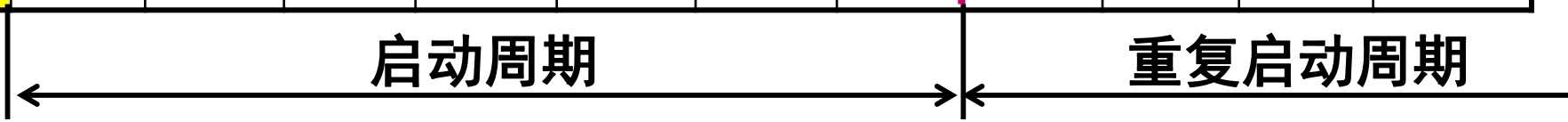
	1	2	3	4	5	6	7	8	9	10	11
S1	X1			X1X2			X1X3 X2	X1		X2X3	X1
S2		X1			X1X2			X2X3	X1		X3
S3		X1			X2	X1		X3	X1X2		
S4			X1			X2			X3	X1	



启动距离＝3时，发生冲突

2. 非线性流水线的调度

	1	2	3	4	5	6	7	8	9	10	11
S1	X1		X2	X1	X3	X2	X1	X1X3	X2		X1X3 X2
S2		X1		X2	X1	X3	X2		X1X3		
S3		X1		X2		X1X3		X2	X1	X3	
S4			X1		X2		X3			X1	



启动距离＝2时，发生冲突

2. 非线性流水线的调度

- 根据二维预约表可以确定非线性流水线的调度方案。
- 5个步骤：
  - ① 构建延迟禁止向量F
  - ② 构建初始冲突向量 C
  - ③ 计算流水线的新的冲突向量
  - ④ 构造用冲突向量表示的流水线状态图
  - ⑤ 确定调度方案

2. 非线性流水线的调度

- 例1：现有一个由 5 个功能段组成的单功能非线性流水线，每个任务通过流水线需要 9 拍。该流水线的二维预约表如下：

	1	2	3	4	5	6	7	8	9
S1	√								√
S2		√	√					√	
S3				√					
S4					√	√			
S5							√	√	

2. 非线性流水线的调度

- ① 根据预约表可以得出一个任务使用各功能段所需间隔的拍数，构建延迟禁止向量F 。
  - 间隔这些拍数就会产生争用或冲突。引起流水线冲突的启动距离为禁止启动距离。
    - 例如：1段为8，2段有1、5、6三种
  - 将流水线中所有各功能段对一个任务流过时会产生争用的节拍间隔数汇集在一起，构成延迟禁止向量F。
  - 上述例子的 F={1，5，6，8}，这些间隔拍数应当禁止使用。

2. 非线性流水线的调度

- ② 将禁止向量(或禁止表)转换成m位的、用二进位表示的初始冲突向量 C。
  - Collision vector
  - 冲突向量C（$C_m\ C_{m-1}\cdots C_i\cdots C_1$）的第 *i* 位表示与“当时”相隔 *i* 拍给流水线送入后续任务，是否会发生功能段冲突
    - 若C_i = 1 间隔 *i* 拍会产生冲突
    - 若C_i = 0 间隔 *i* 拍不会产生冲突
  - m = ?

m=禁止向量F中的最大值

2. 非线性流水线的调度

- 根据延迟禁止表F ={1，5，6，8}，可以得到任务刚进入流水线时的初始冲突向量：

  C = (10110001)

  - 其中C_2、 C_3 、 C_4 、 C_7为0
  - 所以，第 2 个任务可以距第 1 个任务2、3、4或7拍流入流水线，而不会与第 1 个任务冲突。

2. 非线性流水线的调度

- ③ 计算流水线的新的冲突向量
  - 当第 2 个任务进入流水线后，应当产生新的流水线冲突向量，以便决定第 3 个任务相隔多少拍流入流水线，才不会与前面的第 1 个和第 2 个任务争用功能段。
  - 随着流水线中的任务每拍向前推进一个功能段，原先禁止后续任务进入流水线的间隔拍数应相应地减1。

2. 非线性流水线的调度

- ③ 计算流水线的新的冲突向量（续）
  - 这意味着可以将冲突向量放在一个右移位器中，每拍右移1位，让左面移出的空位补“0”
  - 用移位器中的值与初始冲突向量作“按位或”运算，得到一个新的冲突向量
  - 随着任务在流水线中向前推进，会不断形成任务当时的冲突向量
  - 如此重复，这样的操作共进行 n 次

2. 非线性流水线的调度

例1：

- 初始冲突向量 C1 = (10110001)

如果选择第 2 个任务间隔 2 拍时流入流水线，则：

- 第 1 个任务的初始冲突向量右移 2 位，成为 C1= (00101100)
- 第 2 个任务刚流入流水线，其初始冲突向量 C2 = (10110001)

2. 非线性流水线的调度

例1（续）：

- 则流水线新的冲突向量C就应当是第 1 个任务当前的冲突向量与第 2 个任务初始的冲突向量的按位“或”

	1	0	1	1	0	0	0	1	(C2)
∨	0	0	1	0	1	1	0	0	(C1)
	1	0	1	1	1	1	0	1	(C)

若让第 3 个任务流入流水线后，即不与第 1 个任务发生功能段冲突，也不与第 2 个任务发生功能段冲突，则应与第 2 个任务间隔 2 拍或 7 拍。

2. 非线性流水线的调度

如果选择第 3 个任务间隔 2 拍时流入流水线，则：

- 第 1 个任务的冲突向量 (00101100) 继续右移 2 位，成为C1 = (00001011)
- 第 2 个任务的冲突向量 (10110001) 右移2位，成为 C2 = (00101100)
- 第 3 个任务刚流入流水线，其初始冲突向量 C3 = (10110001)

2. 非线性流水线的调度

- 流水线新的冲突向量C就应当是第 1 个任务当前的冲突向量与第 2 个任务的当前冲突向量和第 3 个任务初始的冲突向量的按位“或”

	1	0	1	1	0	0	0	1	(C3)
	0	0	1	0	1	1	0	0	(C2)
∨	0	0	0	0	1	0	1	1	(C1)
	1	0	1	1	1	1	1	1	(C)

2. 非线性流水线的调度

- 也可以这样计算流水线新的冲突向量C：将流水线的当前的冲突向量 C = (10111101) 右移两位，成为 C = (00101111)，然后与第 3 个任务初始的冲突向量 C3 = (10110001) 的按位“或”

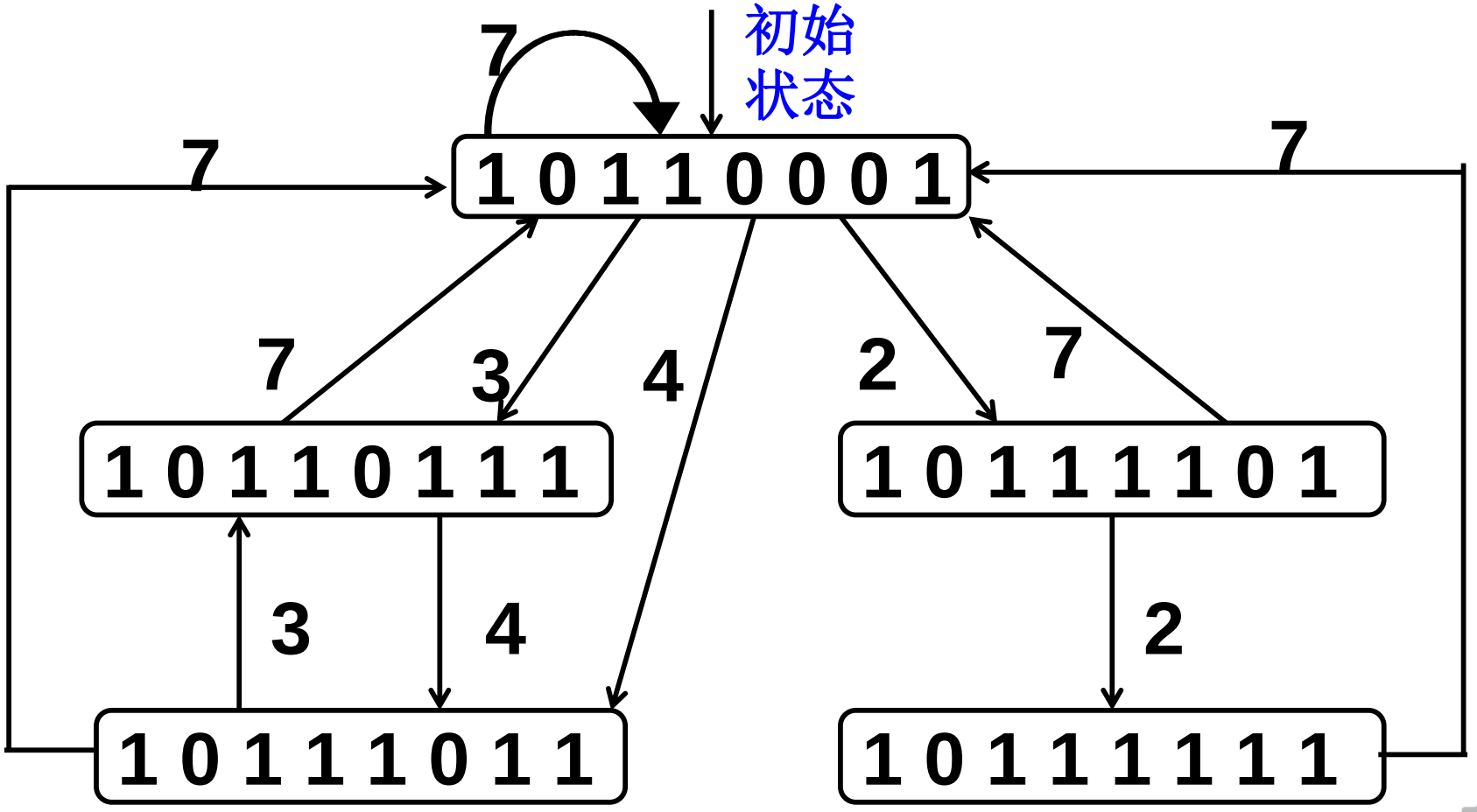
	1	0	1	1	0	0	0	1	(C3)
∨	0	0	1	0	1	1	1	1	(C)
	1	0	1	1	1	1	1	1	(C)

2. 非线性流水线的调度

- 按照这样的思路，选择各种可能的拍数输入新任务，又可以产生新的冲突向量，一直到不再产生不同的冲突向量为止，这样就考虑了所有的可能性。

2. 非线性流水线的调度

- ④ 构造用冲突向量表示的流水线状态图



2. 非线性流水线的调度

⑤ 确定调度方案

- 从状态图中的初始状态出发，找到一个间隔拍数呈周期性重复的方案，并按此方案进行调度，就不会发生功能段冲突
- 显然，可以找到多个调度方案

2. 非线性流水线的调度

- 问题：但哪一种调度方案最佳呢？
- 所谓最佳是指：
  - 流水线吞吐率最高
  - 控制简单
- 在这些调度方案中，找出平均延迟最短的调度方案，就是流水线的最佳调度方案。

2. 非线性流水线的调度

- 根据上述状态图，可以找出所有的调度方案，并计算出每种方案的平均间隔拍数

调度方案	平均间隔拍数	调度方案	平均间隔拍数
(2，2，7)	3.67	(3，7)	5.00
(2，7)	4.50	(4，3，7)	4.67
(3，4)	3.50	(4，7)	5.50
(4，3)	3.50	(7)	7.00
(3，4，7)	4.67	…	…

2. 非线性流水线的调度

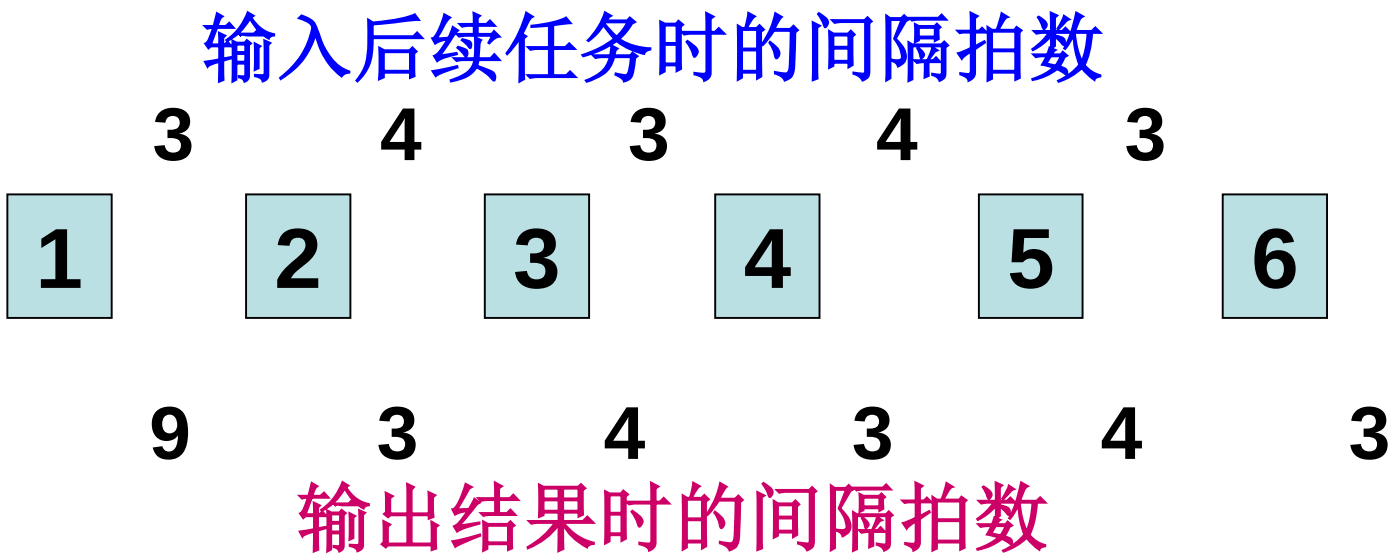
- 可以看出，(3，4) / (4，3) 调度方案平均间隔拍数最小，吞吐率最高，是最佳调度方案
- 为了简化控制，也可以采用等间隔调度，但常常会使吞吐率和效率下降
- 本例中只有一个等间隔调度方案：每隔7拍输入一个任务

2. 非线性流水线的调度

- (3，4) / (4，3) 哪种调度方案更好？
- 需要具体分析！
- 例：连续输入6个任务，分别计算采用(3，4) / (4，3) 调度方案时的吞吐率和效率

2. 非线性流水线的调度

- 采用 (3，4) 时



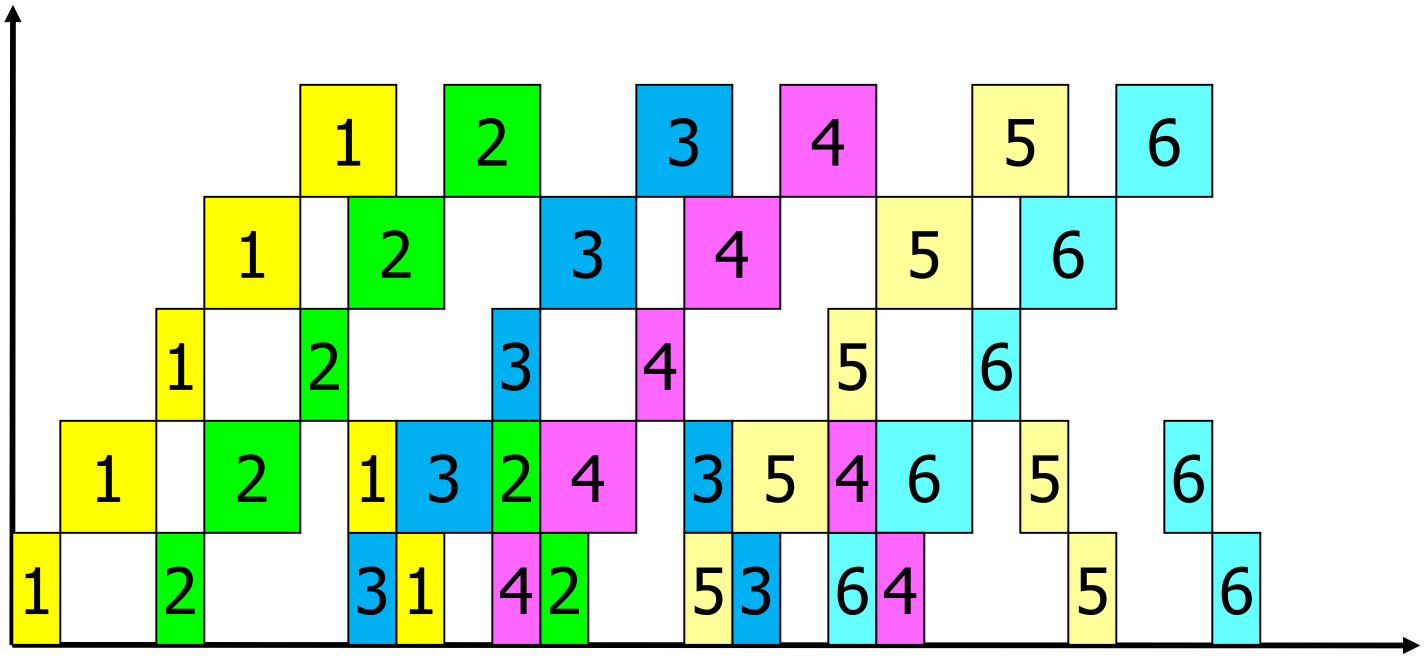
$TP_{max} = 1/3.5\Delta t$　　**Sp = ?**

$TP = 6/(9+3+4+3+4+3)=6/26\Delta t$

效率 $= 6\times10\ \Delta t\ /5\times26\ \Delta t = 46\%$

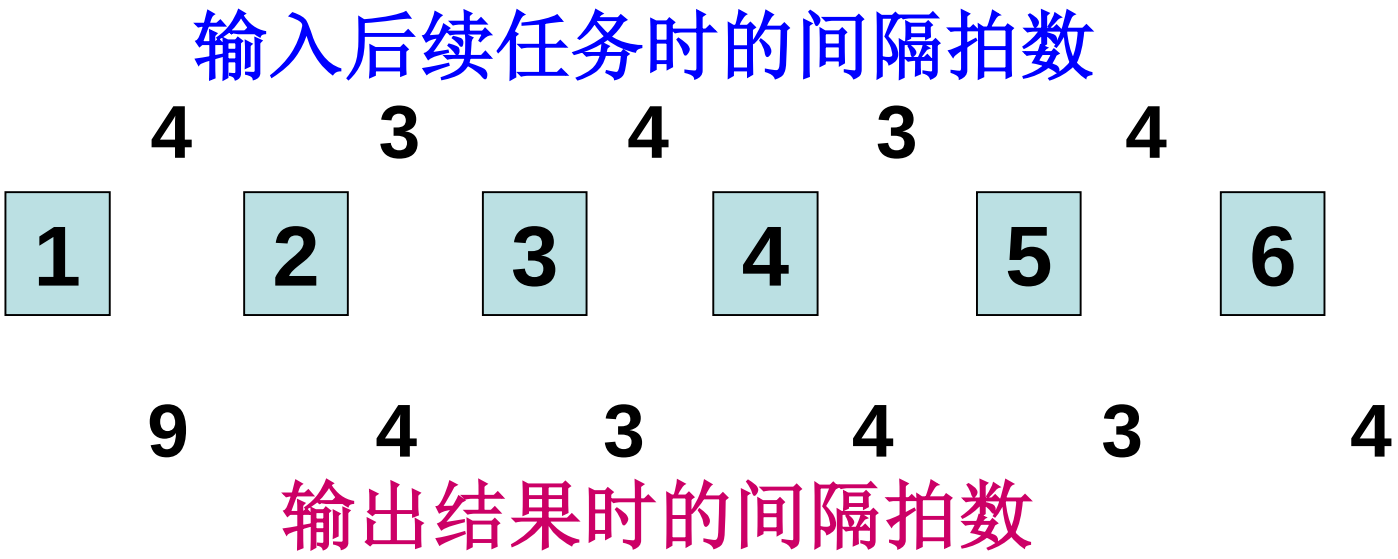
2. 非线性流水线的调度

- 采用 (3，4) 时



2. 非线性流水线的调度

- 采用 (4，3) 时



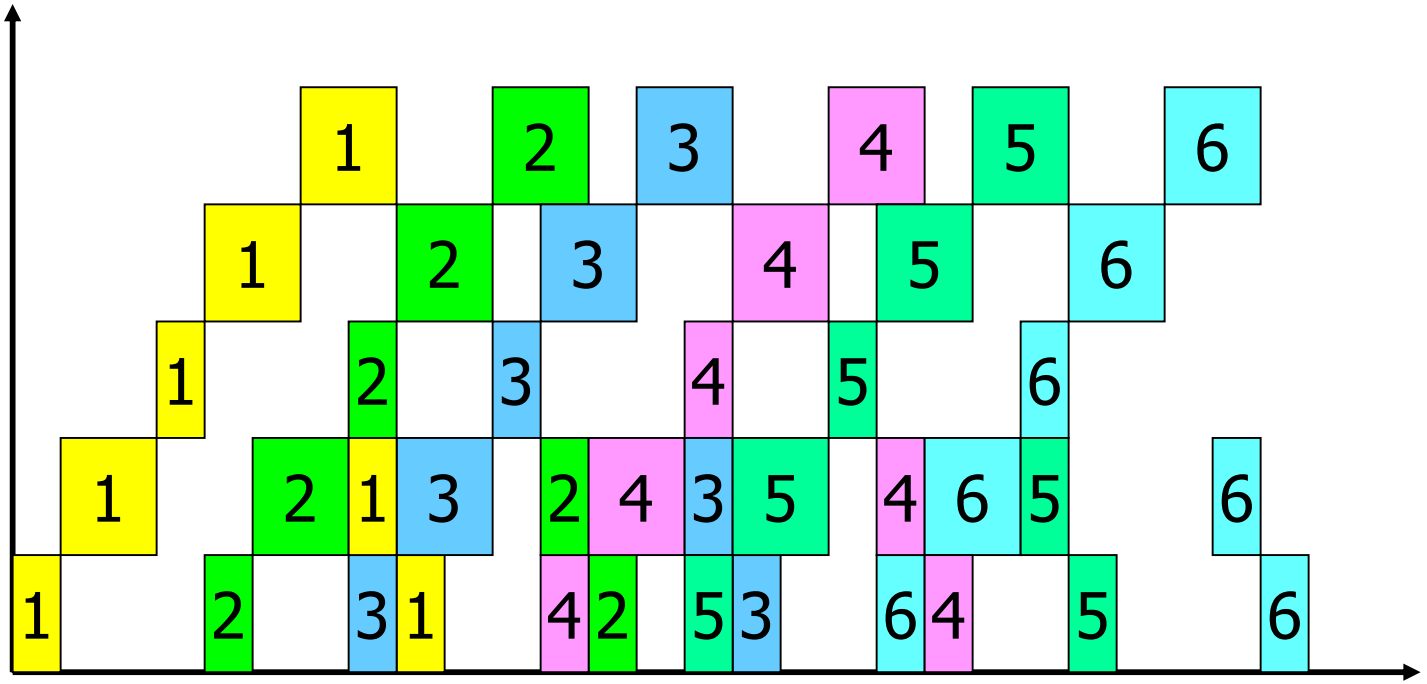
$TP_{max} = 1/3.5\Delta t$ **Sp = ?**

$TP=6/(9+4+3+4+3+4)=6/27\ \Delta t$

效率＝ $6\times10\ \Delta t\ /5\times27\ \Delta t$＝44%

2. 非线性流水线的调度

- 采用 (4，3) 时



2. 非线性流水线的调度

- 例2：一条有 4 个功能段的非线性流水线，每个功能段的延迟时间都相等，它的预约表如下

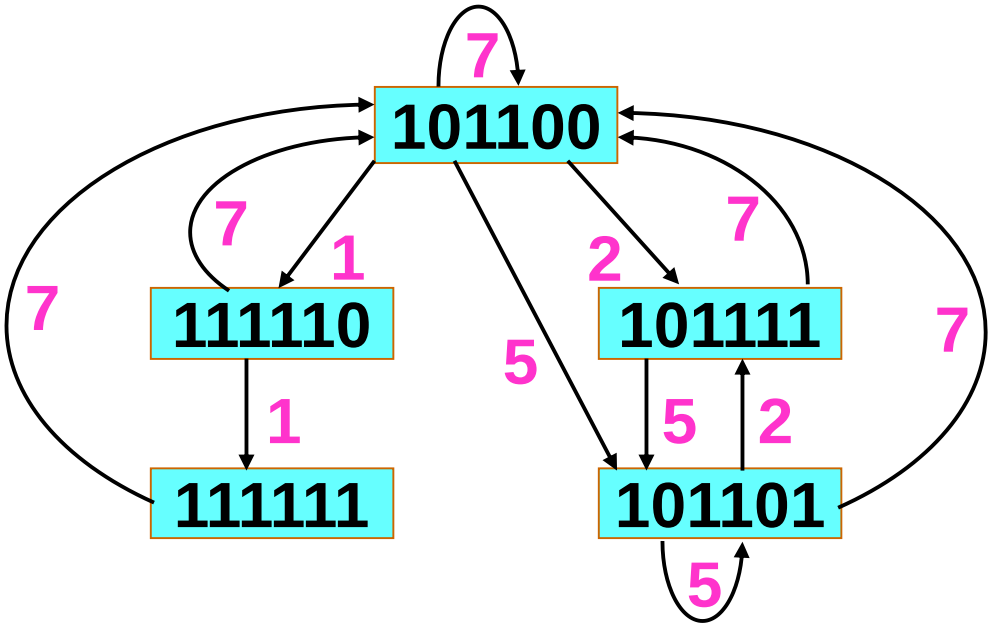
	1	2	3	4	5	6	7
S1	√			√			√
S2		√			√		
S3		√				√	
S4			√				

(1) 写出流水线的禁止向量和初始冲突向量。
(2) 画出调度流水线的状态图。
(3) 求流水线的最小启动循环和最小平均启动距离。
(4) 求平均启动距离最小的恒定循环。

2. 非线性流水线的调度

例2 解：

- 延迟禁止向量：F = (3，4，6)
- 初始冲突向量：C = (101100)
- 状态图



2. 非线性流水线的调度

例2 解：

- 可能的调度方案：
- 平均延迟最小的启动循环被称为最小启动循环。
- 最小启动循环为 (1，1，7)。
- 最小平均启动距离为 3。
- 启动距离最小的恒定循环是(5)。

调度方案	平均延迟
（1，7）	4
（1，1，7）	3
（2，7）	4.5
（2，5）	3.5
（2，5，7）	4.7
（5，7）	6
（5）	5
（7）	7
（5，2，7）	4.7

2. 非线性流水线的调度

例2 解：

最小启动循环（1，1，7）的流水线预约表

	1	2	3	4	5	6	7	8	9	10	11	12
S1	√	▲	〓	√	▲	〓	√	▲	〓	▼		
S2		√	▲	〓	√	▲	〓				▼	
S3		√	▲	〓		√	▲	〓			▼	
S4			√	▲	〓							▼

3. 多功能非线性流水线的调度

- 可以利用上述单功能流水线的调度的基本思想和方法，解决多功能流水线的调度。
- 方法：
  - 将每种功能的预约表都叠加在一起，然后构造交叉冲突向量，以反映多功能动态流水线的各个后继任务流入流水线时应禁止使用的拍数间隔。

5.2.4 流水机器的相关处理和控制机构

- 有关流水线性能的若干问题：
  - 流水线并不能减少（而且一般是增加）单条指令的执行时间，但能够提高吞吐率；
  - 增加流水线的深度可以提高流水线性能；
  - 流水线深度受限于流水线的延迟和额外开销；
  - 需要用高速锁存器作为流水线寄存器；
    - Earle锁存器
  - 流水线只有连续流动不断流，才能获得较高效率；

但是，指令之间存在的相关，限制了流水线的性能。

流水线中的相关

- 流水线只有连续流动不断流，才能获得较高效率。
- 造成流水线断流的原因很多，例如：
  - 编译形成的目的程序不能发挥流水线结构的作用；
  - 存储系统无法提供连续流动所需要的指令和操作数；
  - 相关；
  - 中断等。

指令之间存在的相关，限制了流水线的性能。

流水线中的相关

- **流水线中的相关**是指相邻或相近的两条指令因存在某种关联，它使得指令序列中下一条指令无法按照设计的时钟周期开始执行。
- 三种不同类型的相关
  - **结构相关：**当指令在重叠执行过程中，硬件资源满足不了指令重叠执行的要求，发生**资源冲突**时将产生结构相关。
  - **数据相关：**因一条指令需要用到前面指令的**结果**，而无法与产生结果的指令重叠执行时，就发生了数据相关。
  - **控制相关：**当流水线遇到**分支指令**和其它会改变PC值的指令时就发生控制相关。

流水线中的相关

- 解决结构相关的基本方法
  - 暂停；
  - 重复设置资源；
- 解决数据相关的基本方法
  - 暂停；
  - 定向(旁路)技术：将计算结果从其产生的地方直接送到真正需要它的地方；
- 解决控制相关的基本方法
  - 暂停；
  - 尽早判断出分支转移是否成功；
  - 尽早计算出分支成功转移时的PC值；

流水线中的相关

- **流水线中的相关**是指相邻或相近的两条指令因存在某种关联，它使得指令序列中下一条指令无法按照设计的时钟周期开始执行。

相关	局部性相关	● 指令相关，主存数相关，通用寄存器组数据相关/基址(编址)相关等； ● 只影响数条指令，不影响指缓中预取的指令。 ● 同步流动：WR 引起； ● 异步流动：WW、RW 引起；
	全局性相关	● 转移、中断等引起； ● 影响后续指令； ● 影响指缓中预取的指令； ● 使流水线断流，使吞吐率和效率下降等。

1. 局部相关处理

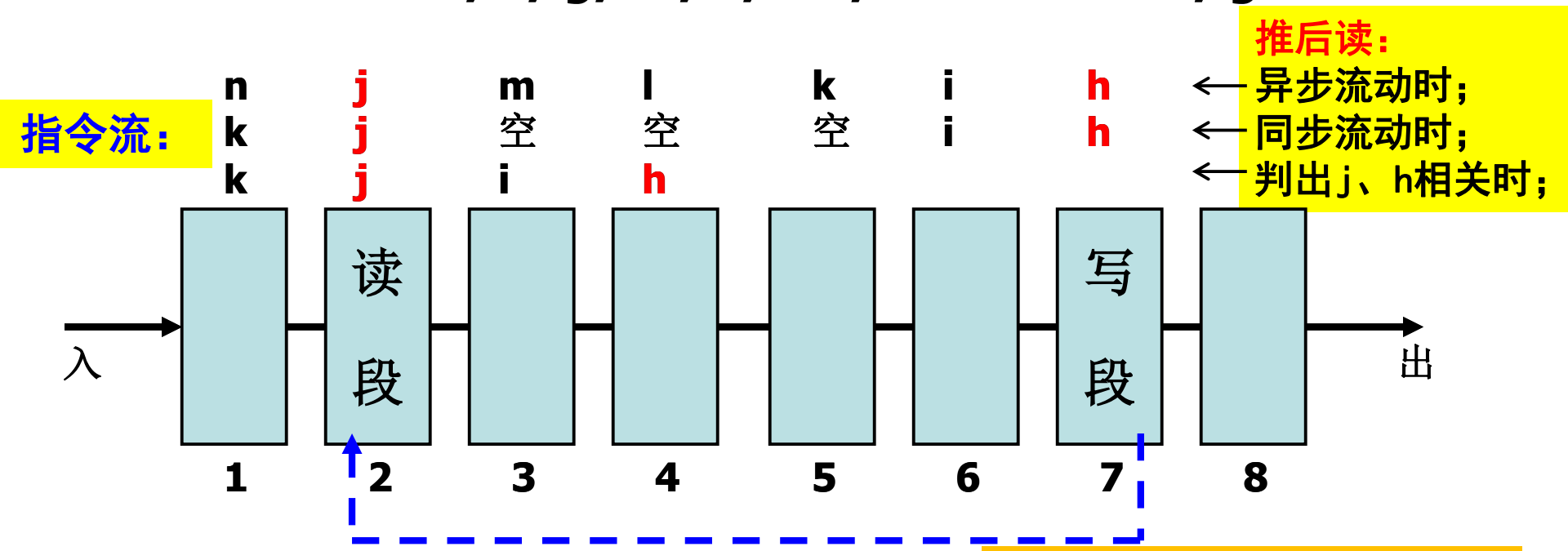
- **原因：** 由于机器同时解释执行多条指令时，这些指令对同一存储单元要求“先写后读”而造成的。
- **解决方法：** 2种
  - 推后读。安排流动顺序。
    - 同步流动
    - 异步流动
  - 设置相关专用通路

1. 局部相关处理

- 任务在流水线中流动顺序的安排和控制
  - **同步流动方式**（顺序流动方式）：任务流出流水线的顺序保持与流入流水线的顺序一致
    - 控制简单，但相关后吞吐率和效率下降
  - **异步流动方式**：任务流出流水线的顺序与流入流水线的顺序不一致

1. 局部性相关处理

- 8段流水线，第2段为读段，第7段为写段
- 指令流：h, i, j, k, l, m, n。其中h, j 相关。



异步方式下，可能会出现“写—写”相关和“先读后写”相关。

顺序流动和异步流动

1. 局部相关处理

设置相关专用通路

- 硬件的方法，可以免去对相关单元的写入和重新读出两个子程。
- **缺点：** 由于流水线同时解释执行多条指令，需要在各个功能部件之间为每种局部性相关都设置单独的相关专用通路，硬件耗费大，控制复杂。

2. 全局性相关处理

- **原因：** 由分支转移指令引起的相关
  - 影响后续指令
  - 影响指缓中的指令
- 对流水线的影响是全局的，包括：
  - 指缓中的指令可能要全部作废
  - 流水线断流，使吞吐率和效率下降等

2. 全局性相关处理

解决方法

- 为了使流水线的吞吐率和效率不致下降太多，关键是能正确判断相关，即能正确判断转移分支
- 无条件转移指令的转移分支是已知的
- 条件转移指令的转移分支是不确定的，需要等指令流出流水线以后才能确定。为判断条件转移分支，可以采取下述方法：

2. 全局性相关

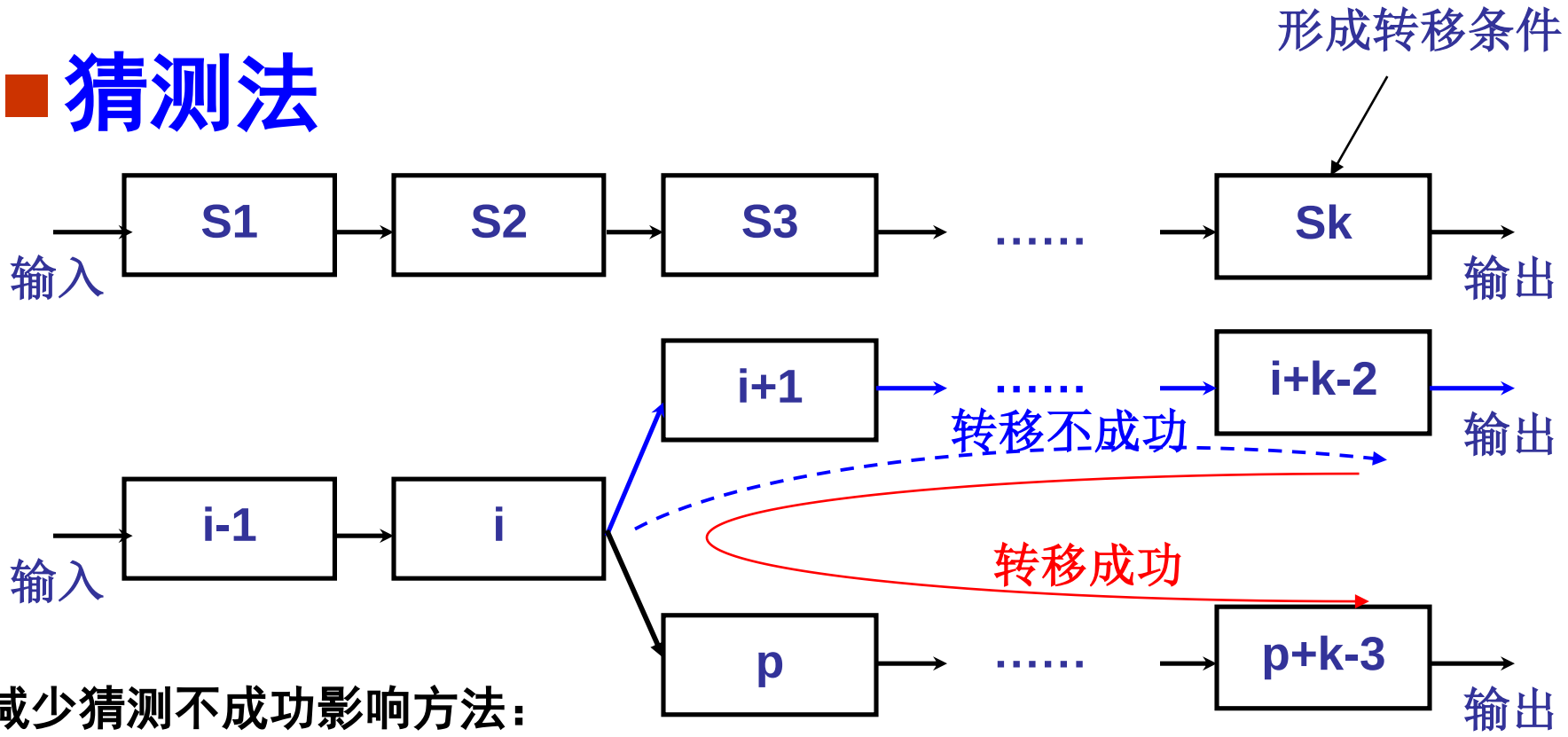
- 条件转移分支判断方法
  - 猜测法
  - 加快和提前形成条件码
  - 采用延迟转移
  - 加快短循环程序的处理

2. 全局性相关

- 猜测法
  - 选取一个分支继续执行
  - 选取哪个分支？
    - 若知道分支的概率，则选取概率高的分支
    - 若不知道概率，或两个分支的概率相近时，宜选取不成功的分支

2. 全局性相关

猜测法



减少猜测不成功影响方法：

1) 猜测时只译码、取操作数，在条件未形成之前不执行。
2) 执行，但不送结果。
3) 执行，送结果，使用后援寄存器。
4) 为了尽可能快地回到原分支点，可以预取转移成功分支的头几条指令。

2. 全局性相关

- 加快和提前形成条件码
  - 加快单条指令条件码形成
  - 加快一段程序的条件码形成
- 采用延迟转移
  - 用软件方法进行静态指令调度，将转移指令与其前面不相关的一条或多条指令交换位置

2. 全局性相关

- 加快短循环程序的处理
  - 将短循环程序全部装入指缓
  - 对循环程序出口分支的处理，恒猜选环分支，因为循环分支的概率高。一般可以使循环时的流水加快1/3到3/4

3. 流水机器的中断处理

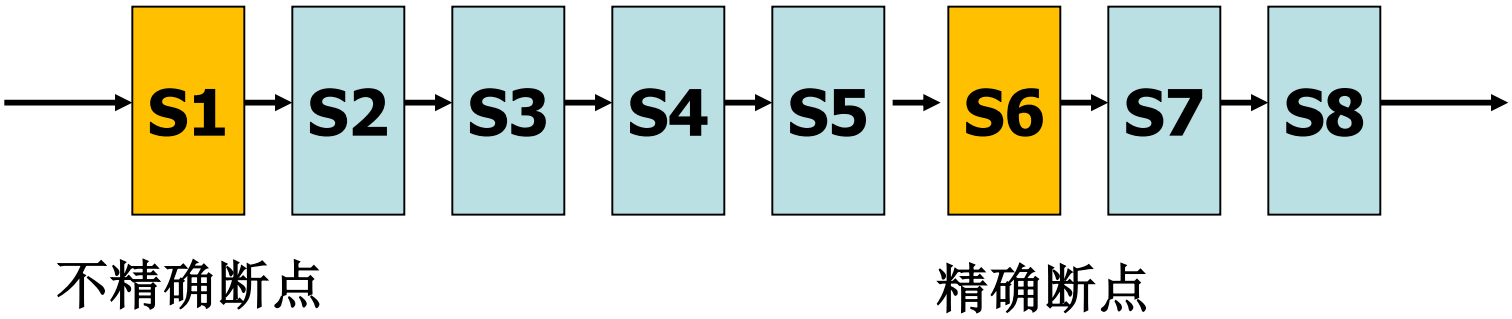
- 中断会使流水线断流
- 中断的特点：
  - 不经常发生
  - 不可预测
  - 一旦进入中断，可能持续很长时间
- 流水机器中断的处理主要是断点现场的保护与恢复

3. 流水机器的中断处理

- 流水线中断有其自己的特点：
  - 当在第 i 条指令发生中断时，由于流水线同时处理多条指令，中断断点的现场可能已经不是第i+1条指令尚未开始的地方，而是第i+1，i+2，…等已经被部分解释执行了的地方。
  - 对于异步流水线，这些指令有些可能已经流到了指令i的前面去了

3. 流水机器的中断处理

- 可以采取以下解决方法：
  - 不精确断点法
  - 精确断点法



流水线处理机的中断处理

3. 流水机器的中断处理

- 不精确断点法
  - 当在第i条指令发生中断时，不再允许尚未进入流水线的后续指令再进入，但允许已经进入流水线的指令全部执行完毕，然后再转入中断处理程序，这样，断点就不一定是第i条，有可能是第i+1，i+2，…，i+m，即断点是不精确的
  - 优点：中断处理简单
  - 缺点：调试困难

3. 流水机器的中断处理

- 精确断点法
  - 多数流水机器采用此方法
  - 不论指令i在流水线的哪一段发生中断，只保护第i条指令的现场。i之后流入流水线的指令的现场都能恢复
  - 此方法需要设置大量的后援寄存器，但这些后援寄存器也是“指令复执”所需要的，不必另外设置