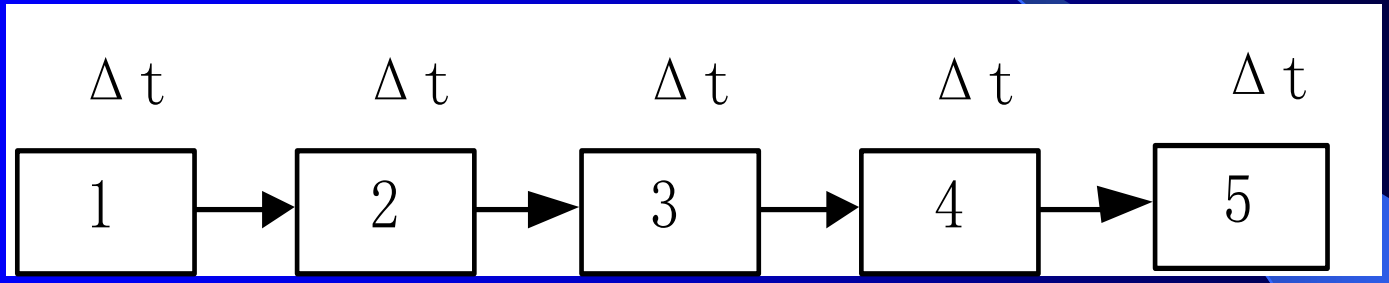
二．流水线的执行过程及性能评价

1 均匀流水线

加法流水线：

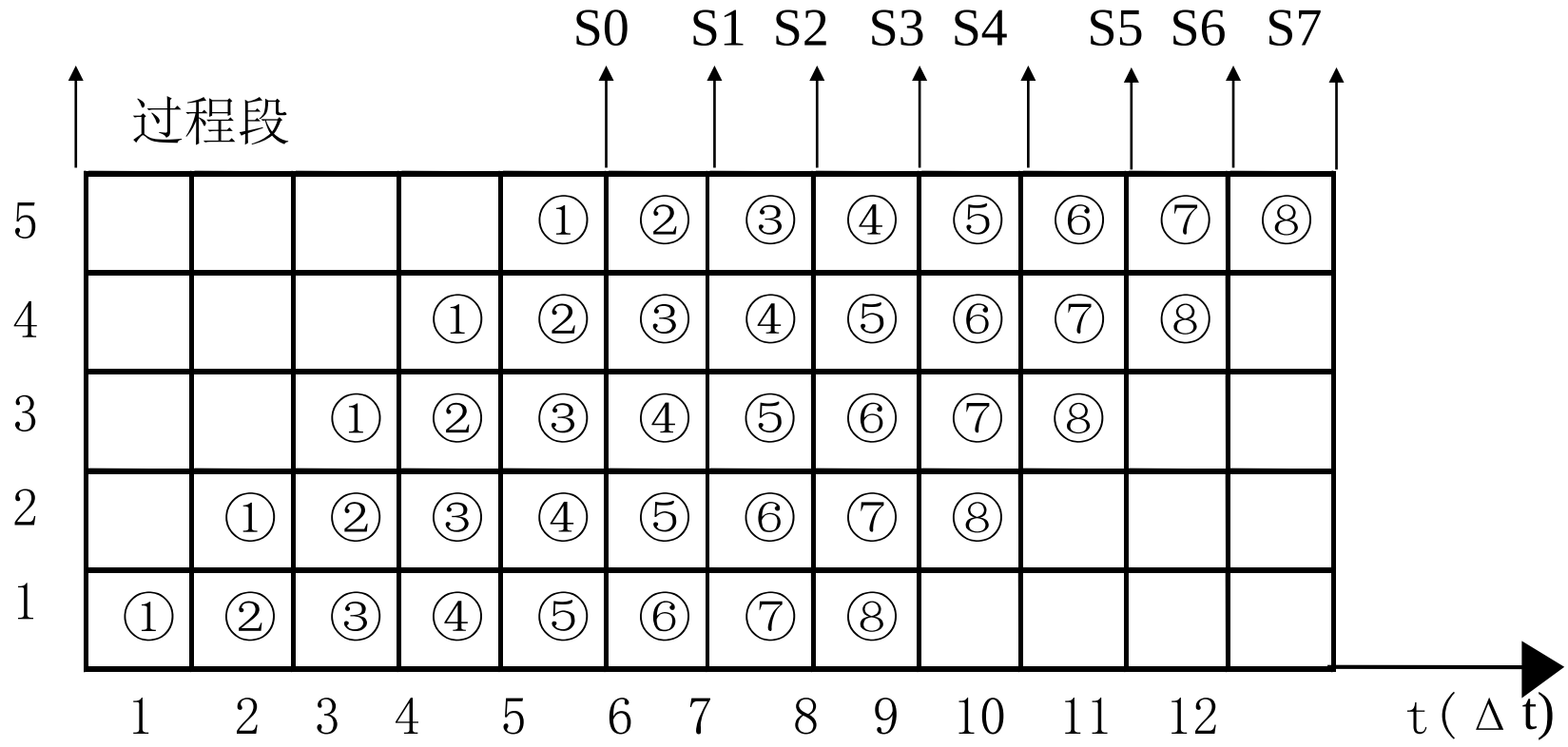


1）不相关算式

计算：$S_i = a_i + b_i$ $(i=0\sim7)$ 共有8个算式

①$S0=a_0+b_0$ ②$S1=a_1+b_1$ … ⑧$S7=a_7+b_7$

画出各算式在流水线上执行过程时空图



完成n个连续任务需要的总时间为：

$$T_k = (k + n - 1)\Delta t$$

其中：k 为流水线的段数，Δt为时钟周期。

性能计算：

吞吐率（TP）：单位时间输出的结果数。

$$TP = \frac{n}{(k+n-1)\Delta t}$$

最大吞吐率为：

$$TP_{\max} = \lim_{n\to\infty} \frac{n}{(k+n-1)\Delta t} = \frac{1}{\Delta t}$$

TP=(输出结果数)/（完成算式总用时）

=8/12=2/3（1/Δt）

一个浮点加法器流水线的时空图

由求阶差、对阶、尾数加和规格化4个流水段组成。

空间

规格化				规格化1	规格化2	规格化3	规格化4	规格化5	
尾数加			尾数加1	尾数加2	尾数加3	尾数加4	尾数加5		
对阶		对阶1	对阶2	对阶3	对阶4	对阶5			
求阶差	求阶差1	求阶差2	求阶差3	求阶差4	求阶差5				
	0	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8 时间

计算：S=a0+a1+a2+a3+a4+a5+a6+a7

2）相关算式

计算：**S=a0+a1+a2+a3+a4+a5+a6+a7**

对相关算式要合理分解算式——尽量分解为少相关算式：

① S0=a0+a1

② S1=a2+a3

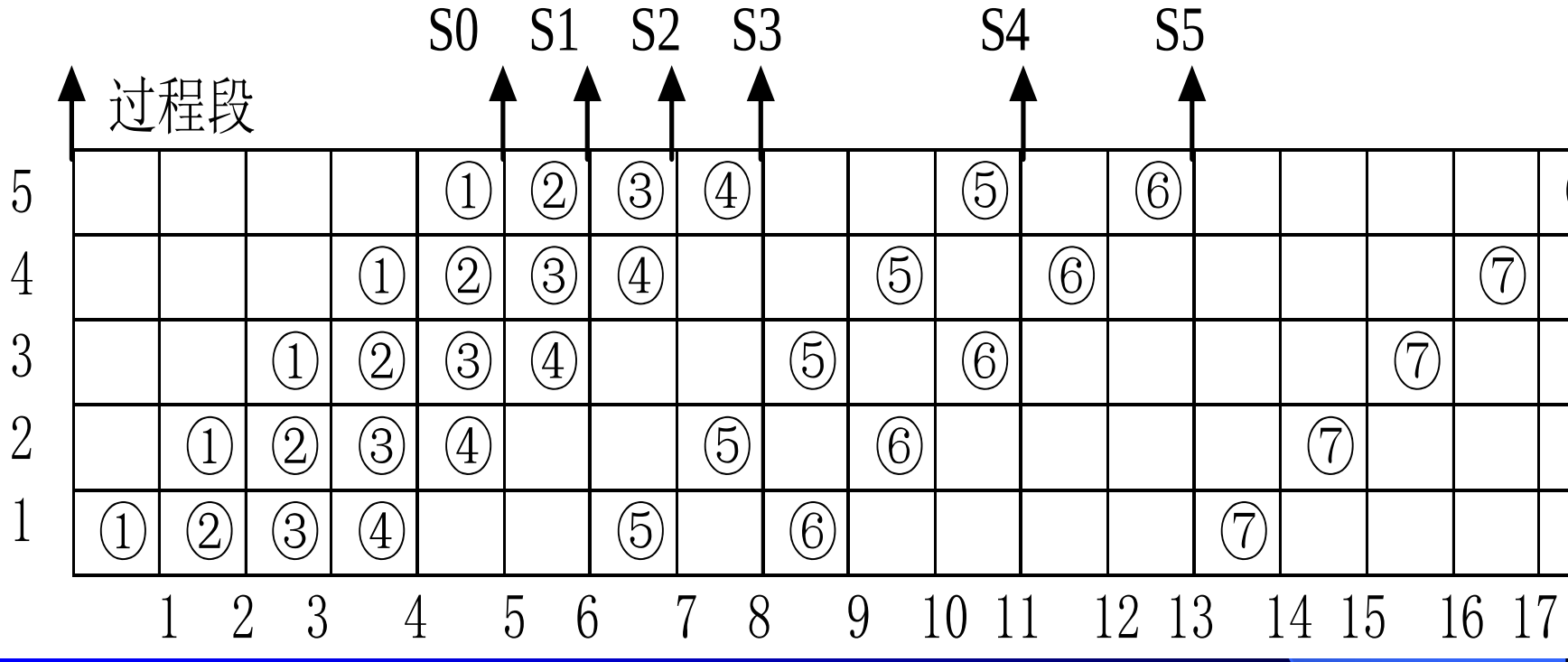
③ S2=a4+a5

④ S3=a6+a7

⑤S4=S0+S1

⑥S5=S2+S3

⑦S6=S4+S5



TP=(输出结果数)/（完成算式总用时）

TP=7/18 （1/Δt）

效率（η）：即流水线上部件的利用率

η =（作用区域面积）/（完成运算所需时间矩形面积）

=（7*5Δt）/（18Δt*5）=7/18

结论：相关发生时，对单条流水线而言会降低流水线性能。

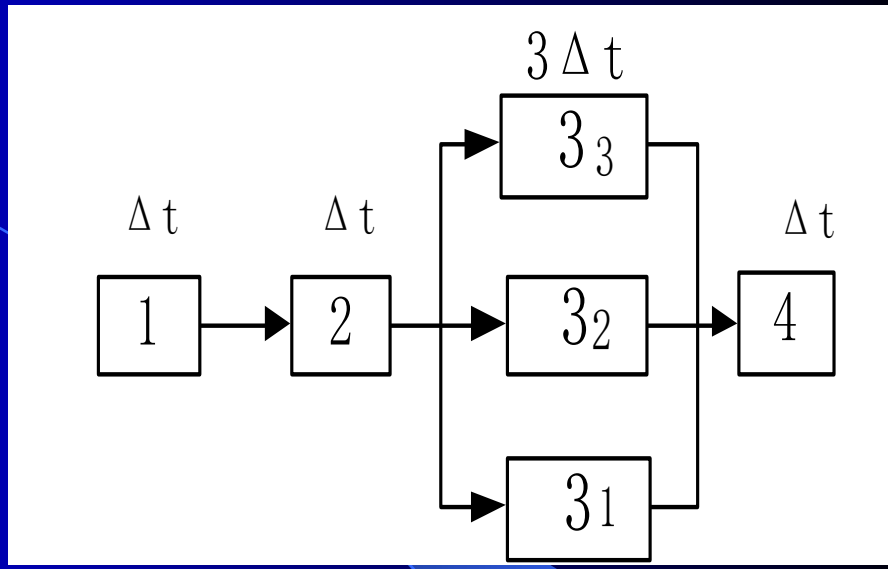
4 时间匹配的非均匀流水线

右图所示乘法流水线

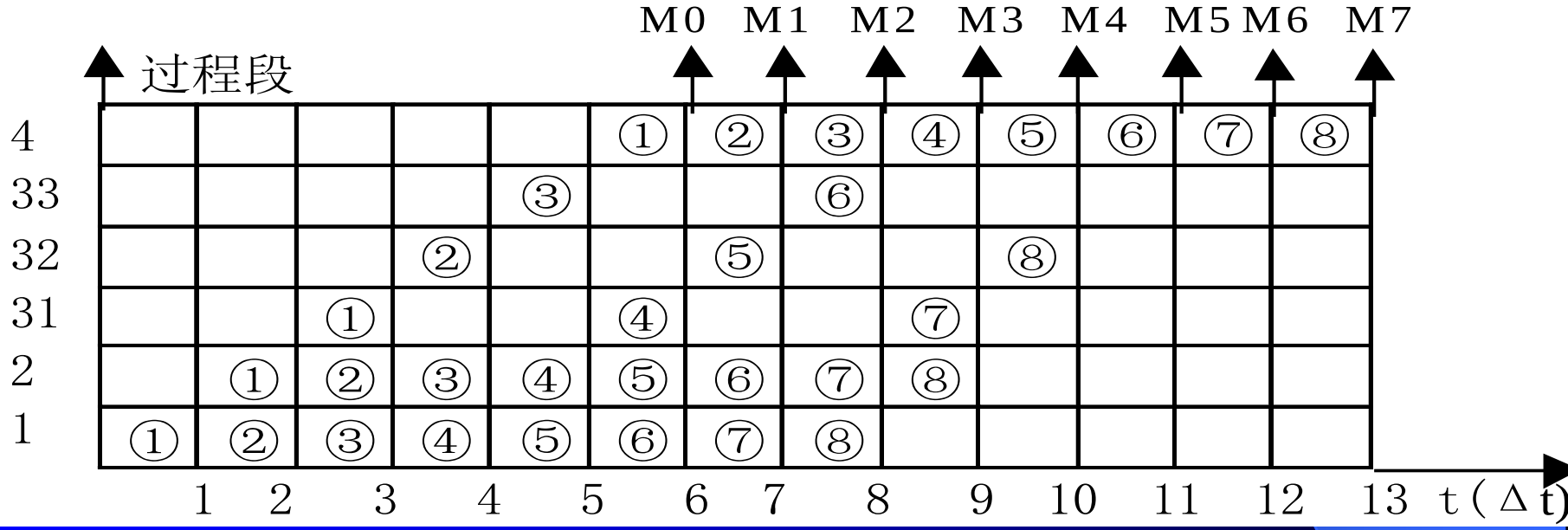
完 成 计 算 :Mi=ai*bi (i=0~7) （也是不相关式子）

1）① M0=a0*b0 …

⑧ M7=a7*b7

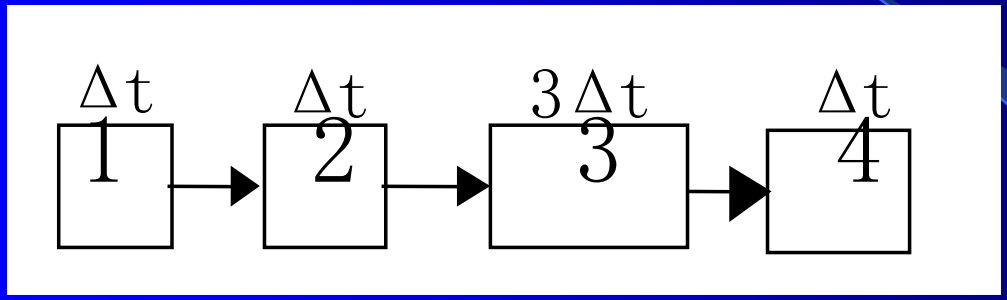


2）画出各算式在流水线上执行过程示意图



- 3）性能：TP=8/13 （1/Δt）
- η=（8*6Δt）/（13Δt*6）=8/13

3 时间不匹配的非均匀流水线



按上图所示乘法流水线完成算式:

M=a0*a1*a2*a3*a4*a5*a6*a7

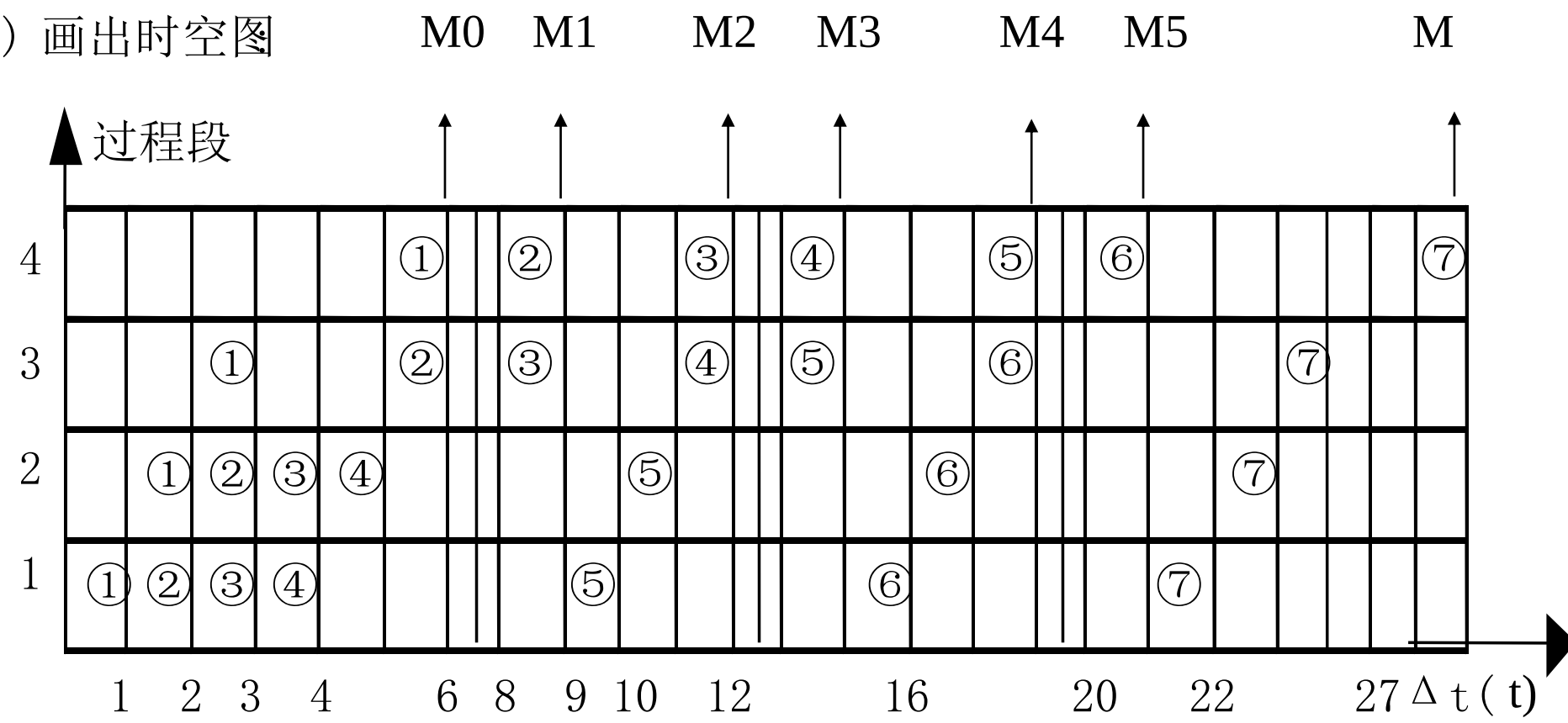
1）合理分解算式

①M0=a0*a1　　②M1=a2*a3

③M2=a4*a5　　④M3=a6*a7

⑤M4=M0*M1　　⑥M5=M2*M3　　⑦M6=M4*M5

2）画出时空图

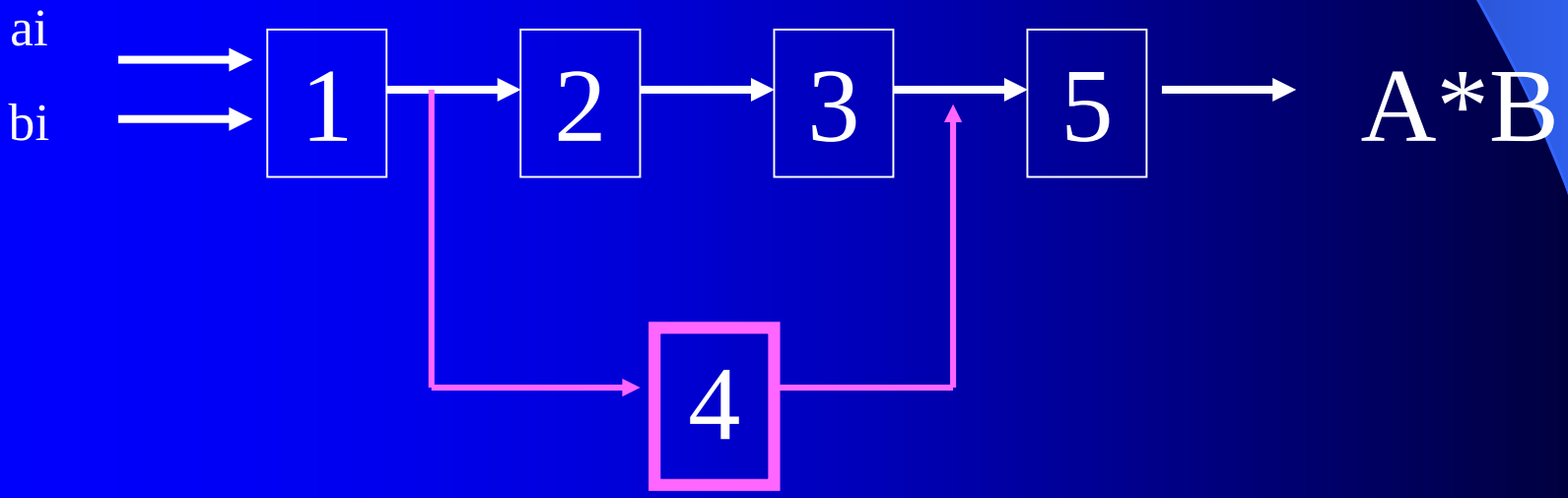


- **3）性能:**
- **TP=7/27 （1/Δt）**
- **η=（7*6Δt）/（27Δt*4）=7/18**

4、静态多功能流水线

- 计算 A*B=

$$\sum_{i=1}^{4} a_i.b_i$$



- 1—2—3---5 做加法
- 1—4—5　　做乘法

解：分解算式

①S1=a1.b1

②s2=a2.b2

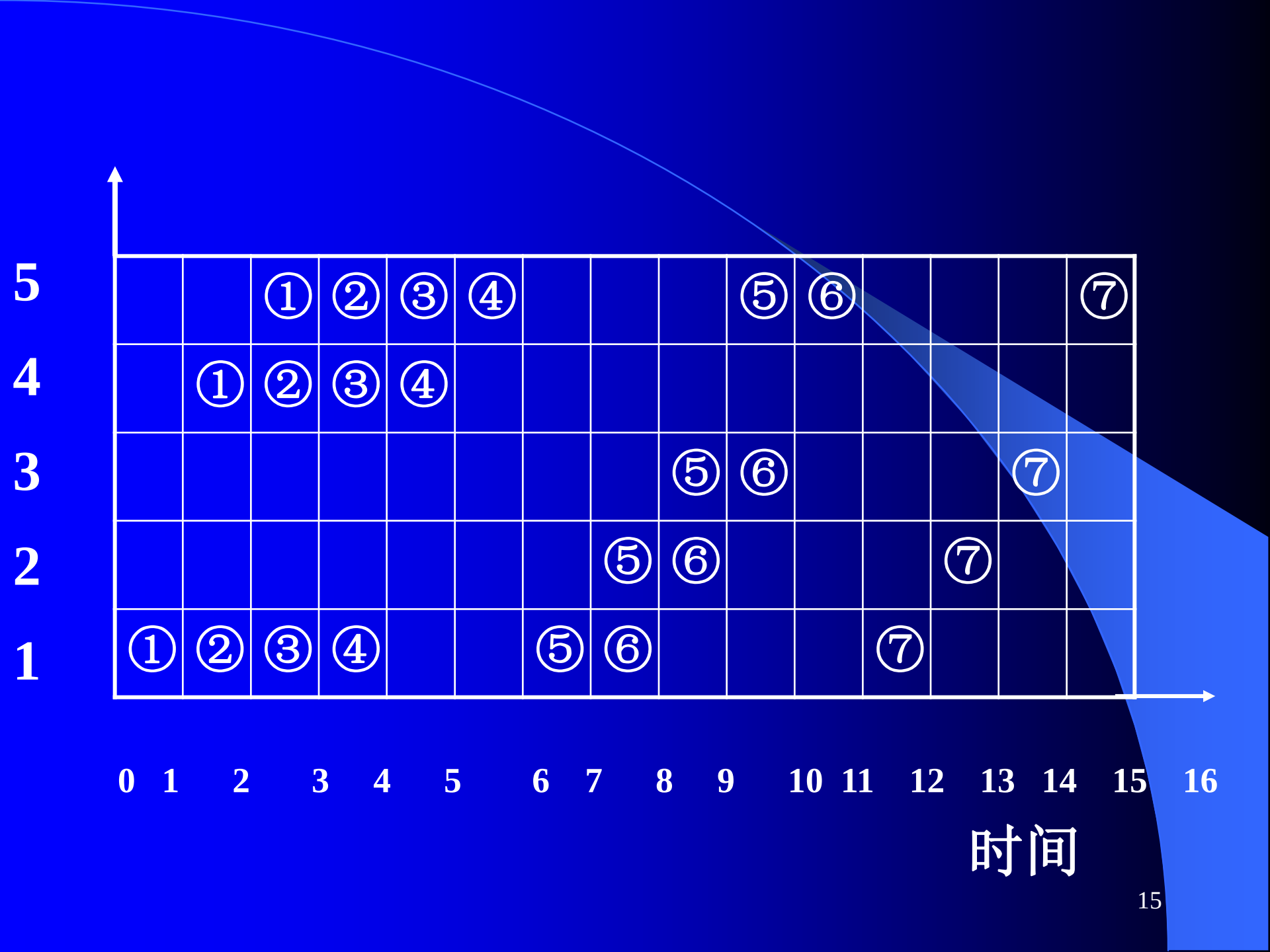
③s3=a3.b3

④s4=a4.b4

⑤s5=s1+s2

⑥s6=s3+s4

⑦s7=s5+s6


5
4
3
2
1
①
②
③
④
⑤
⑥
⑦
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
时间

- 吞吐率

$$TP = \frac{7}{15} \quad (1/\triangle T)$$

$$效率 = \frac{3*4+4*3}{5*15} = 32\%$$

5、单功能流水线与多功能流水线

- 单功能流水线：只能完成一种固定功能的流水线。

  Cray-1计算机有12条

  YH-1计算机有18条

  Pentium有一条5段的定点和一条8段的浮点流水线。

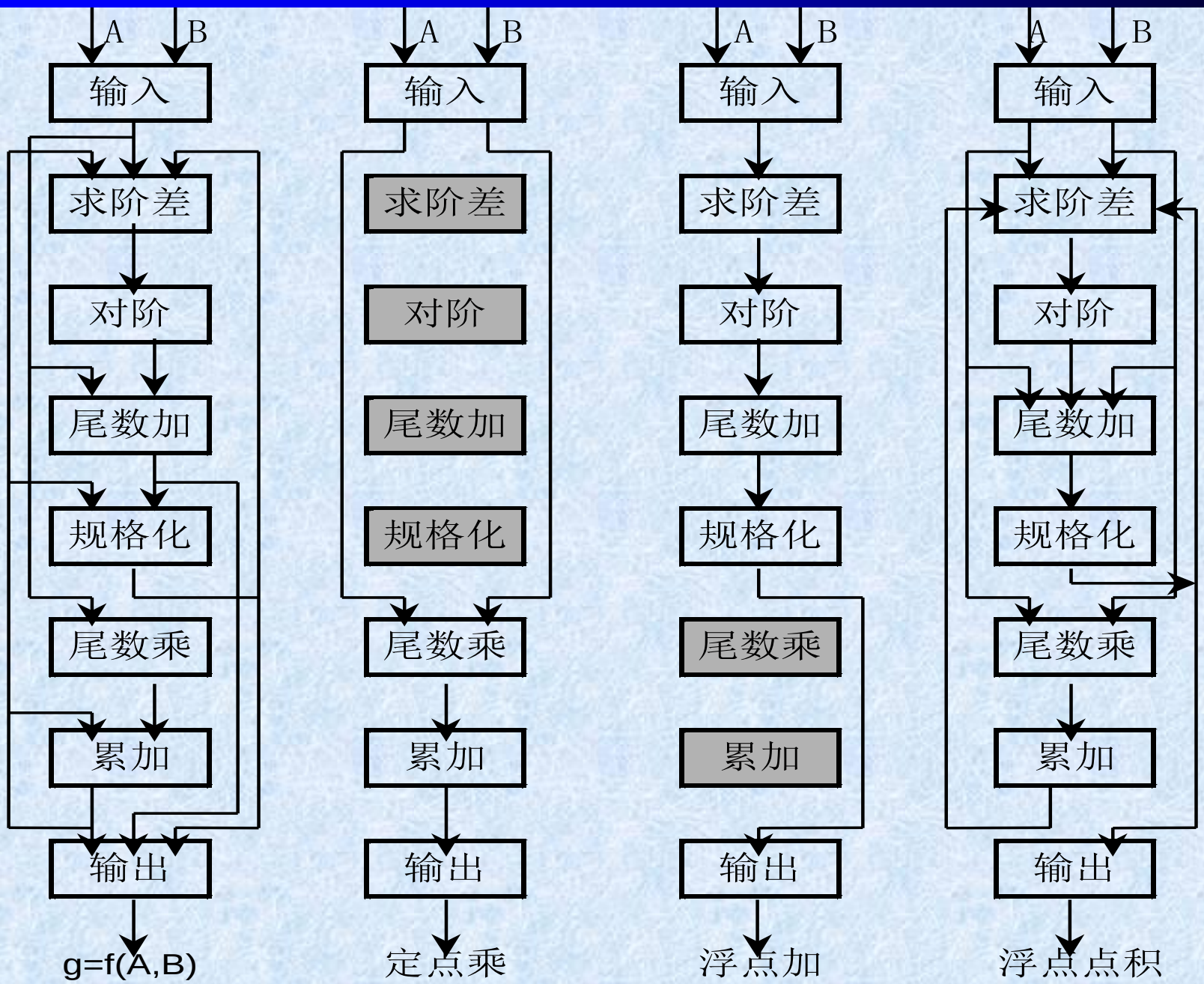
  PentiumⅢ有两条定点指令流水线，一条浮点指令流水线。

- 多功能流水线：

  流水线的各段通过不同的连接实现不同的功能。

  Texas公司的ASC计算机中的8段流水线，能够实现：

  定点加减法、定点乘法、
  浮点加法、浮点乘法，
  逻辑运算、移位操作、
  数据转换、向量运算等。

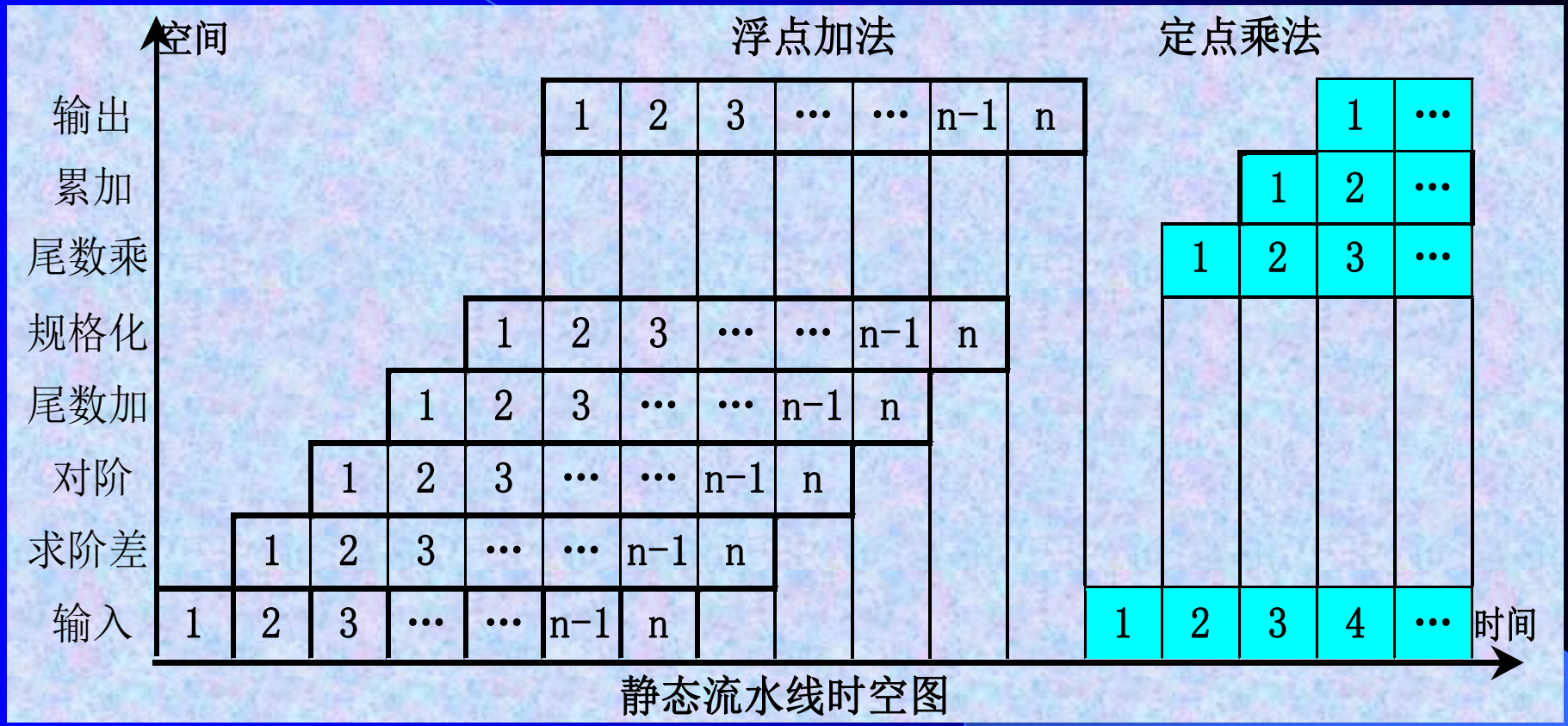


(a) 功能段间的互连　(b) 定点乘法　(c) 浮点加法　(d) 浮点点积

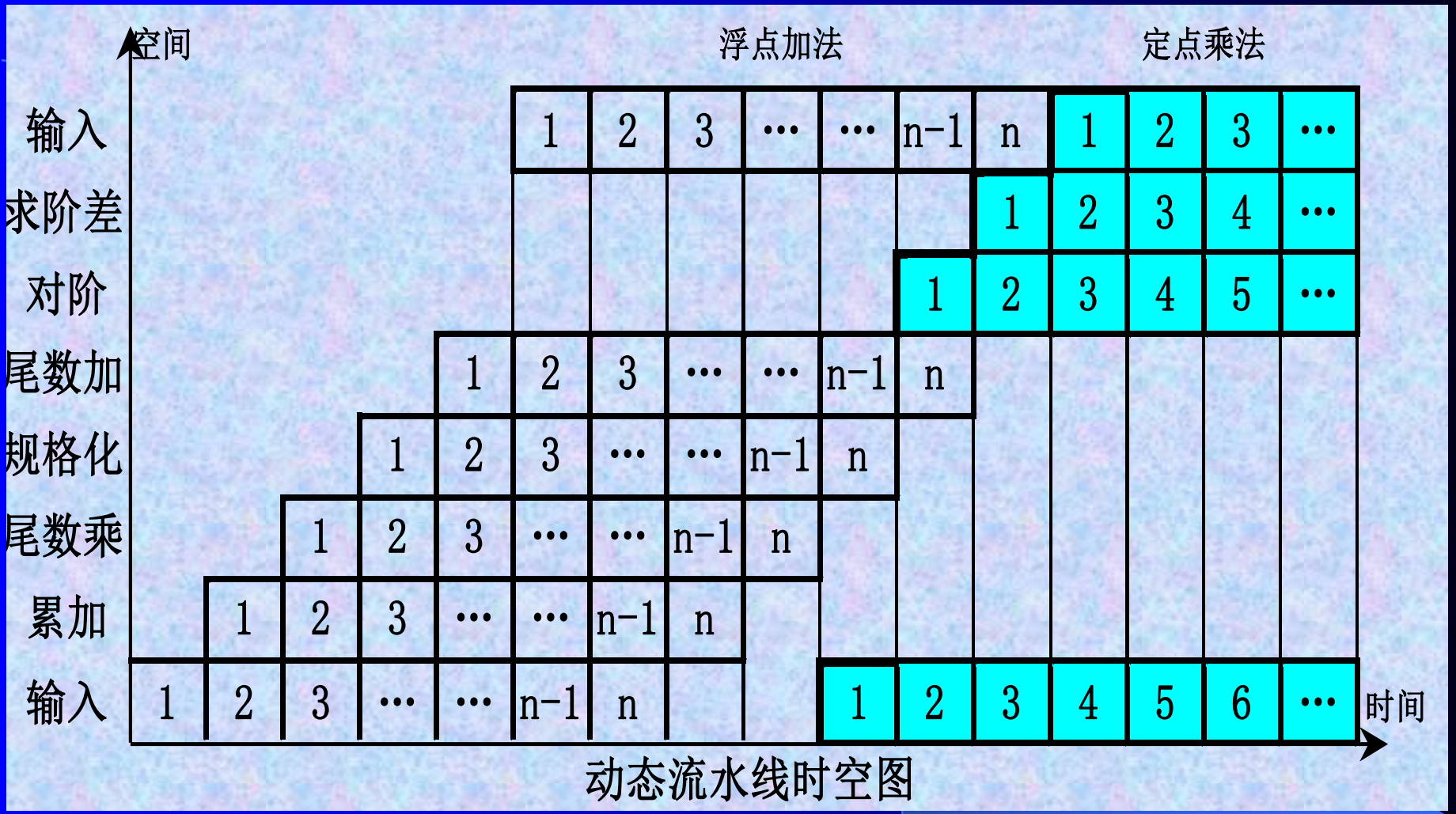
6、静态流水线与动态流水线

- 静态流水线：同一段时间内，多功能流水线中的各个功能段只能按照一种固定的方式连接，实现一种固定的功能。

只有连续出现同一种运算时，流水线的效率才能得到充分的发挥。



- 动态流水线：在同一段时间内，多功能流水线中的各段可以按照不同的方式连接，同时执行多种功能。



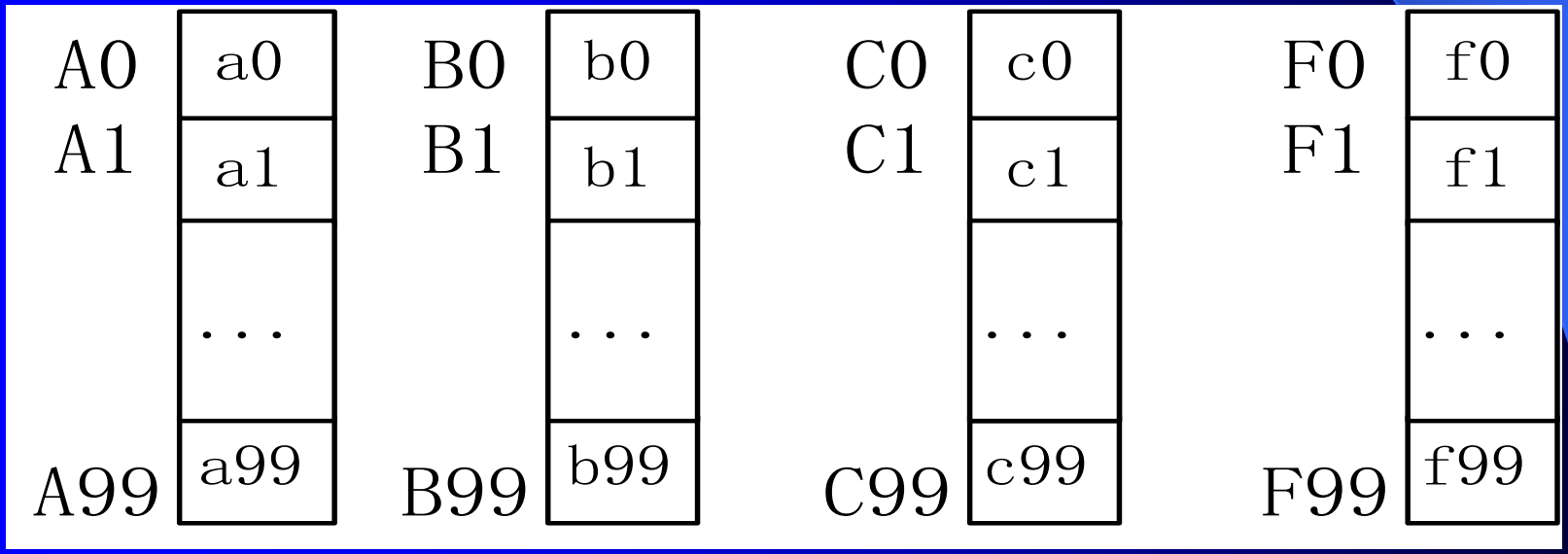
动态流水线时空图

第二节 向量处理技术

1 向量的处理方式

计算：**fi=ai*bi+ci**

设各向量分别放在大写字母单元中：



1）横向处理

按照算式一个一个地进行计算，即按行计算

第一步计算：**f0=a0*b0+c0**

```
LD    R , A0
MUL   R , B0
ADD   R , C0
ST    R , F0
```

第二步计算：**f1=a1*b1+c1**

即将第一步中的脚标0改为1，同样用上述四条指令。

……

直到第一百步，**f99**

优点：作为工作单元的通用寄存器少（本例仅用一个**R**）

缺点：条条指令发生相关，因而无人采用。

2）纵向处理

将所有算式列出后，按列进行计算。如对f0~ f99可分为四大步完成。

第一大步：取向量

```
LD   R0 , A0
 :
LD   R99 , A99
```

第二大步：向量乘

```
MUL  R0 , B0
 :
MUL  R99 , B99
```

第三大步：向量加

```
ADD  R0 , C0
 :
ADD  R99 , C99
```

第四大步：送结果

```
ST    R0 , F0
  :
ST    R99 , F99
```

优点：解决了相关问题，将原来条条发生相关改为条条不相关。

缺点：在向量数据较多时，所用的寄存器数目多。

如本例共用了一百个寄存器（R0~R99)

结论：在向量数据不多时，可用纵向处理，而向量数据较多时，可用纵横处理。

3）纵横处理

基本思想：将所有算式分为若干组进行如f0~ f99可分为10组：

第一组：，第二组，……第十组

组内采用纵向处理，组间采用横向处理。

如第一组：①取向量

```
LD   R0 , A0
 :
LD   R9 , A9
```

②向量乘

```
MUL  R0 , B0
 :
MUL  R9 , B9
```

③向量加

```
ADD  R0 , C0
  :
ADD  R9 , C9
```

④送结果

```
ST   R0 , F0
  :
ST   R9 , F9
```

其余各组与第一组类似，因而总共用了10个寄存器（R0~ R9）

2 CRAY-1机有关问题

1）向量指令类型

①取向量： $V_i \leftarrow$ 存储器

②存向量： 存储器 $\leftarrow V_i$

③向量与向量运算：

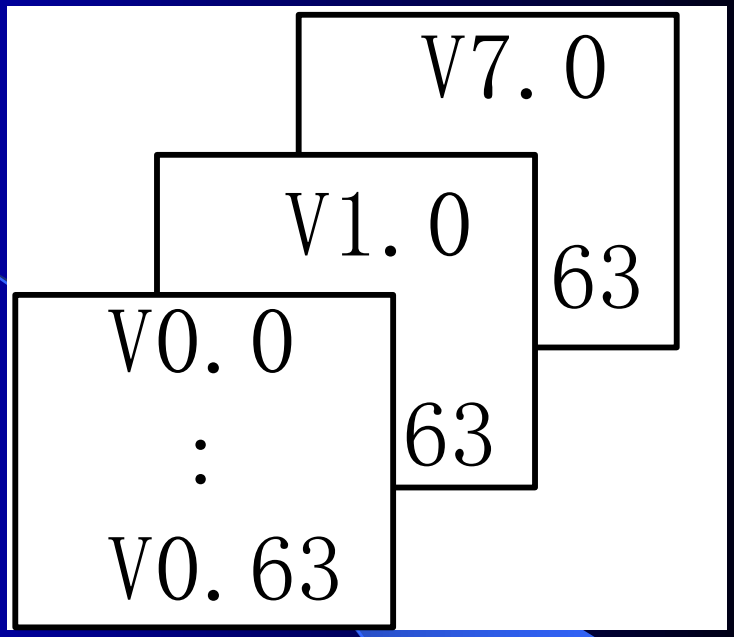
$$V_i \leftarrow V_j \ OP \ V_k$$

④向量与数据运算：

$$V_i \leftarrow V_j \ OP \ B$$

2）多向量寄存器组结构

共有8个向量寄存器组（V0～V7），每个组可存放64个长度为64位的二进制数的向量数据。



3）多功能部件

每个部件都以

1τ=10ns=10^{-8}S为单位的流水线结构。

①逻辑运算：

②定点加：

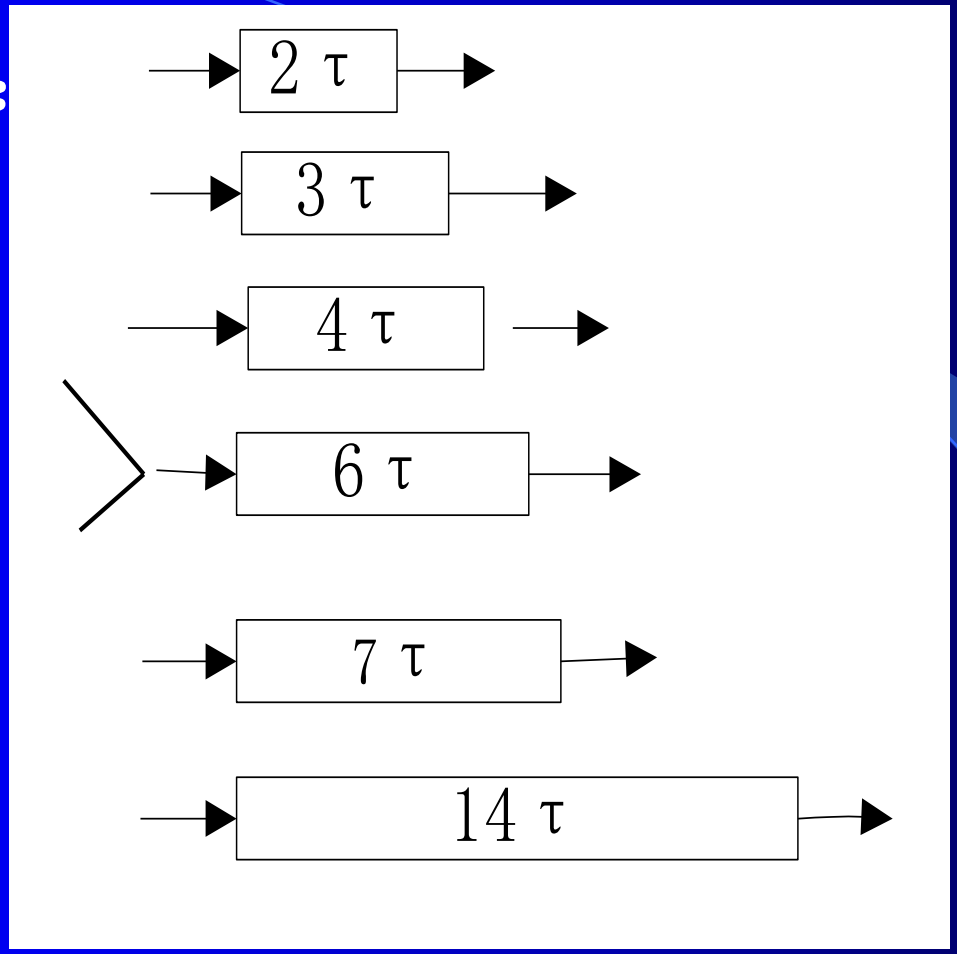
③移位：

④浮点加：

⑤访存储器

⑥浮点乘：

⑦除法 ：



此外，在功能部件和向量寄存器组之间相互传送也用1 τ 。

4）独立总线结构

每个向量寄存器组到每个功能部件之间都有单独总线连接，在不冲突条件下，可实现功能部件之间并行运行。

3 向量指令的执行过程及性能计算

已知向量指令：**V2← V1 + V0**（浮点加）
向量长度为**64**，实际上是**64**组向量数据求和。

1）写出64组算式

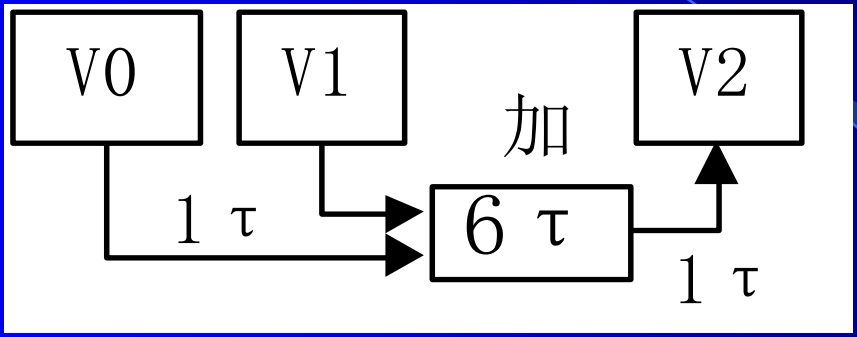
①V2.0←V1.0 + V0.0

② V2.1←V1.1 + V0.1

…

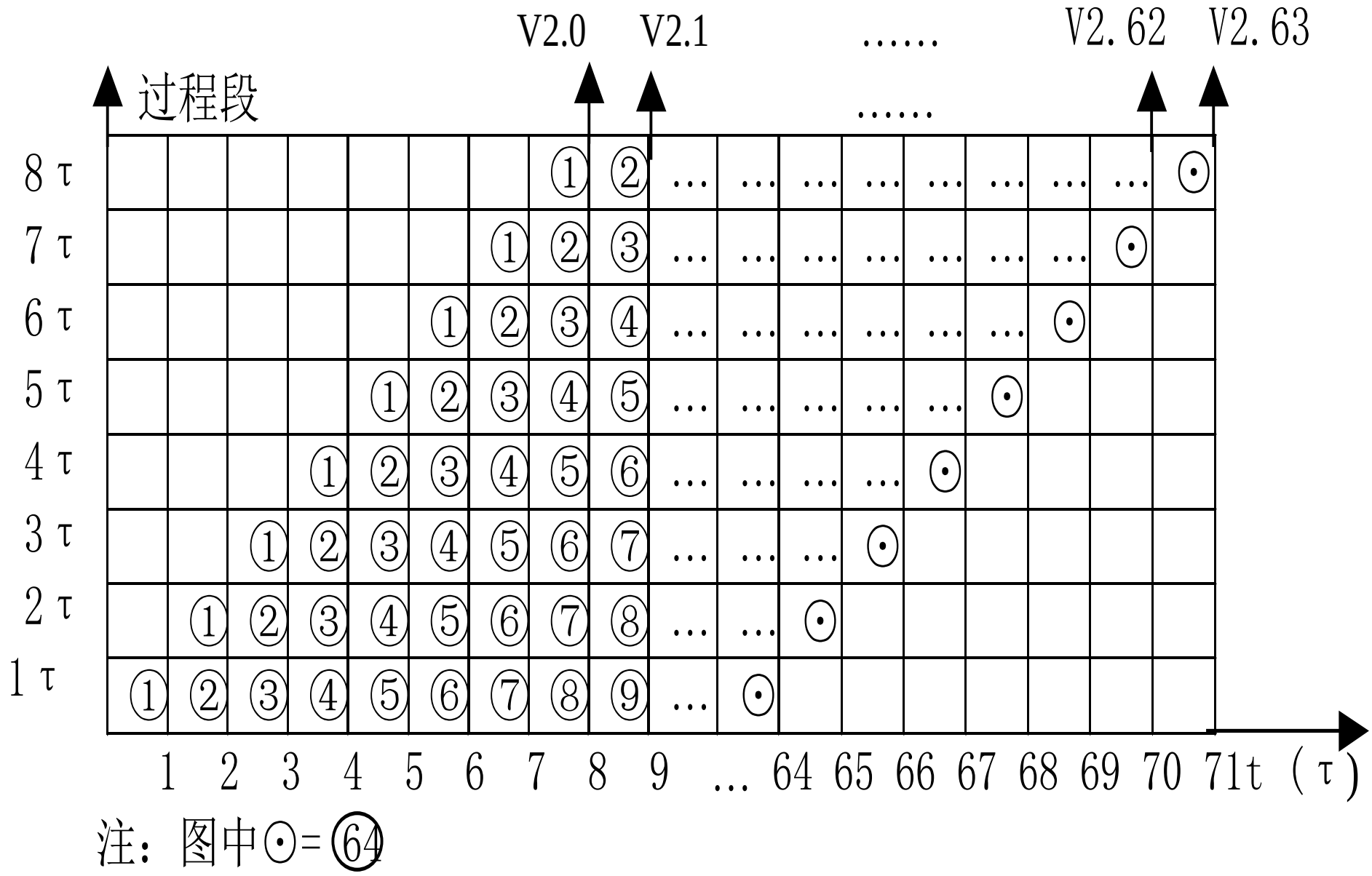
64 V2.63← V1.63 + V0.63

2）画出向量指令结构图（如图所示）



3）画出各算式执行过程示意图

送数1τ，加法6τ，输出结果1τ，共8τ。


V2.0
V2.1
……
V2.62
V2.63
过程段
8τ
7τ
6τ
5τ
4τ
3τ
2τ
1τ
1 2 3 4 5 6 7 8 9 … 64 65 66 67 68 69 70 71t (τ)
注：图中⊙=⑥④

4）完成运算时间

第一个结果时间 +（长度-1） τ

=(1+6+1) τ +(64-1) τ

=71 τ

5）向量数据处理速度计算

（向量指令条数*长度）/（完成运算用时）

=（1*64）/（71*10^{-8}S）

=90MFOLPS

4 多条向量指令的执行过程

若有多条向量指令，且可并行执行时，完成运算用时，可选用时最多的那条向量指令。

如：

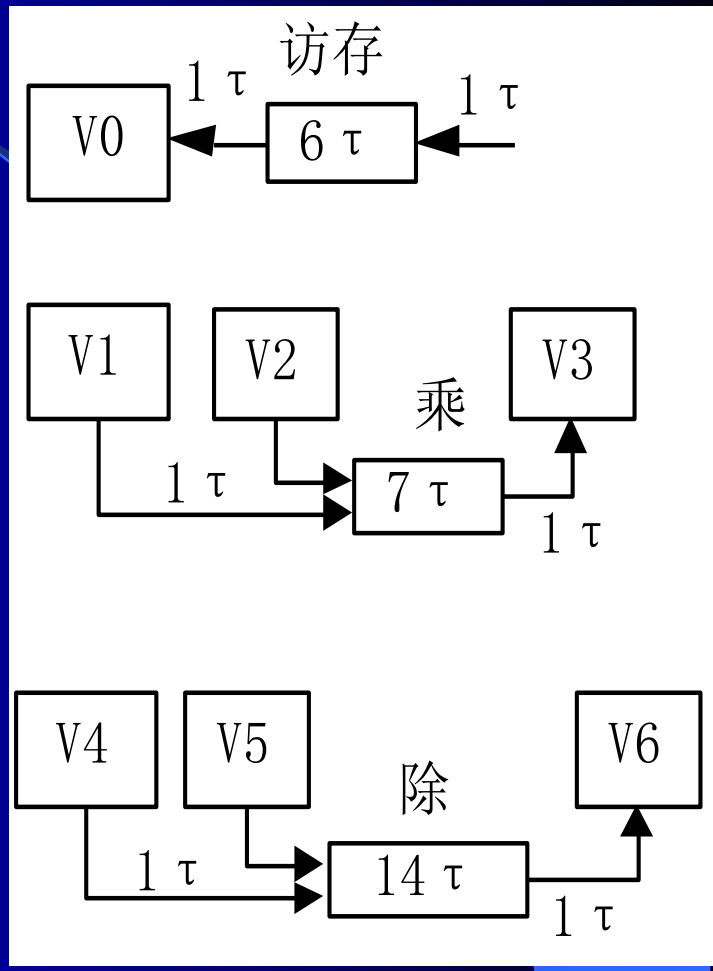
V0←存储器
V3← V2×V1
V6← V5÷V4

} 可并行执行，向量长度为64

由于除法用时最长，以它为准。

1+14+1+(64-1)=79(τ)

3*64/(79*10^{-8}S)≈244MFLOPS



四　向量的链接特性

1　链接：将多条相关的向量指令链接起来组成更大规模的流水线，从而进一步提高向量数据处理速度，这种链接称为向量链接。

2　向量指令之间的几种情况

1）既不相关，又无冲突

不能链接，但可并行执行（执行时间以最长向量指令时间为准）

2）条条指令相关，且无冲突

可顺利链接

3）条条指令相关，但有冲突不能顺利链接，执行时间往往需要推迟。

2)条条指令相关，且无冲突

3　可顺利链接的情况

有如下向量指令：

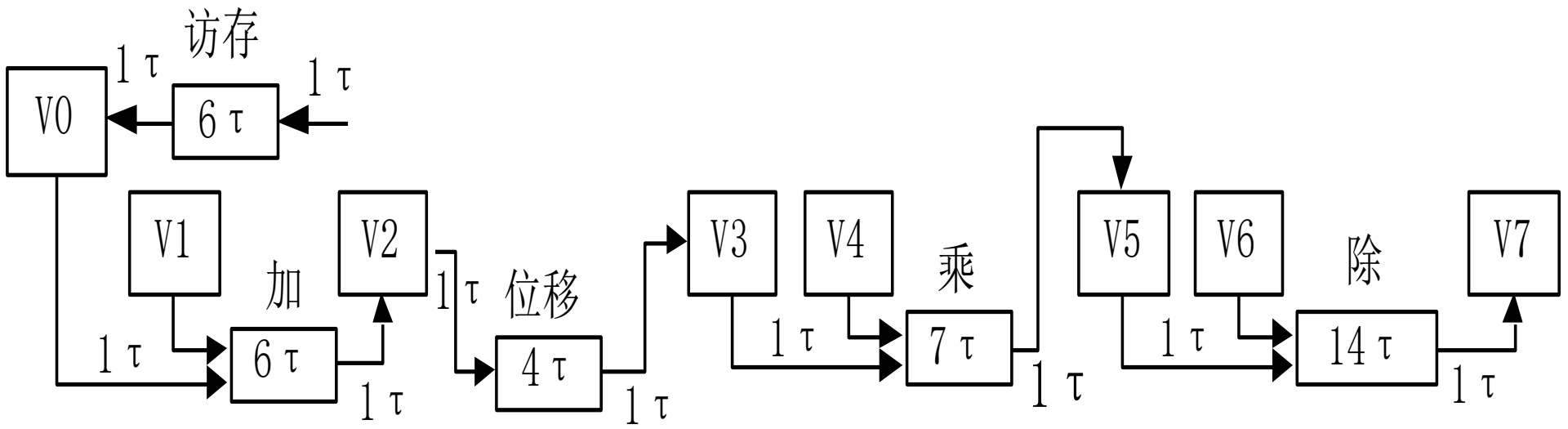
①V0←存储器；②V2←V0+V1；③V3←V2位移；

④ V5← V3×V4；⑤ V7← V5÷V6

向量长度64

相关：上一条向量指令的结果作下一条指令的一个源操作数。

1）画出向量链接特性图



2）完成运算时间

6+2+6+2+4+2+7+2+14+2+（64-1）=110(τ)

3）计算向量数据处理速度：

5*64/(110*10^{-8}S)≈291MFLOPS

此处结论：

相关在向量链接中有利于向量数据处理速度的提高。

4 不能顺利链接的情况

有如下向量指令：

①V0←存储器；　　②V2←V0×V1；

③V4←V2+V3；　　④ V5←V4位移；

⑤ V7←V5÷V6；　　⑥ V0←V7×V1

条条指令相关，但有冲突不能顺利链接

向量长度64，上述向量指令条条相关，有冲突：

冲突 {
- V0—目寄存器冲突
- V1—源寄存器冲突
- * —功能部件冲突

故不能顺利链接

1）不能顺利链接时，对画向量链接特性图的影响。

①源冲突：第一次送出画实线，第二次送出画虚线

②目冲突：第一次接收画实线，第二次接收画虚线

③功能部件冲突：第一次出现画实线，第二次出现画虚线

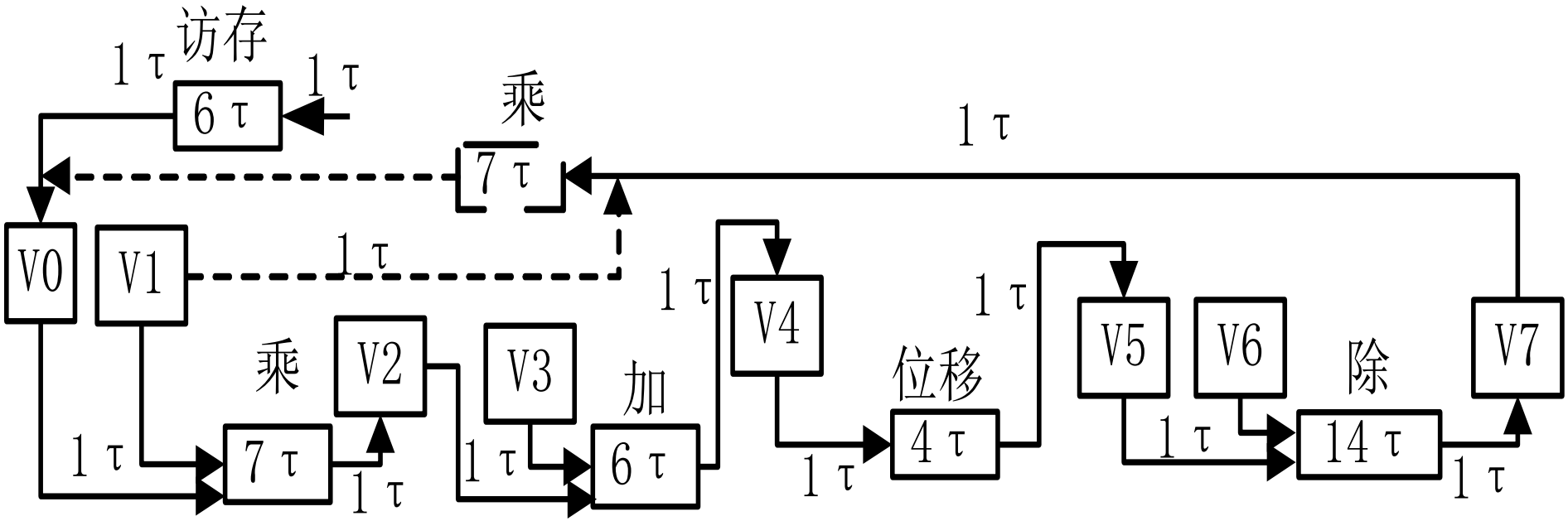
V0×V1→V2　实线
V7×V1→V0　虚线

V0←存储器　实线
V0← V7×V1　虚线

①V0←存储器；　②V2←V0×V1；

③V4←V2+V3；　④ V5←V4位移；

⑤ V7←V5÷V6；　⑥ V0←V7×V1



2）为了计算是否需要推迟时间，以及推迟多少时间，先计算冲突部件的　有关时间。

①源冲突：

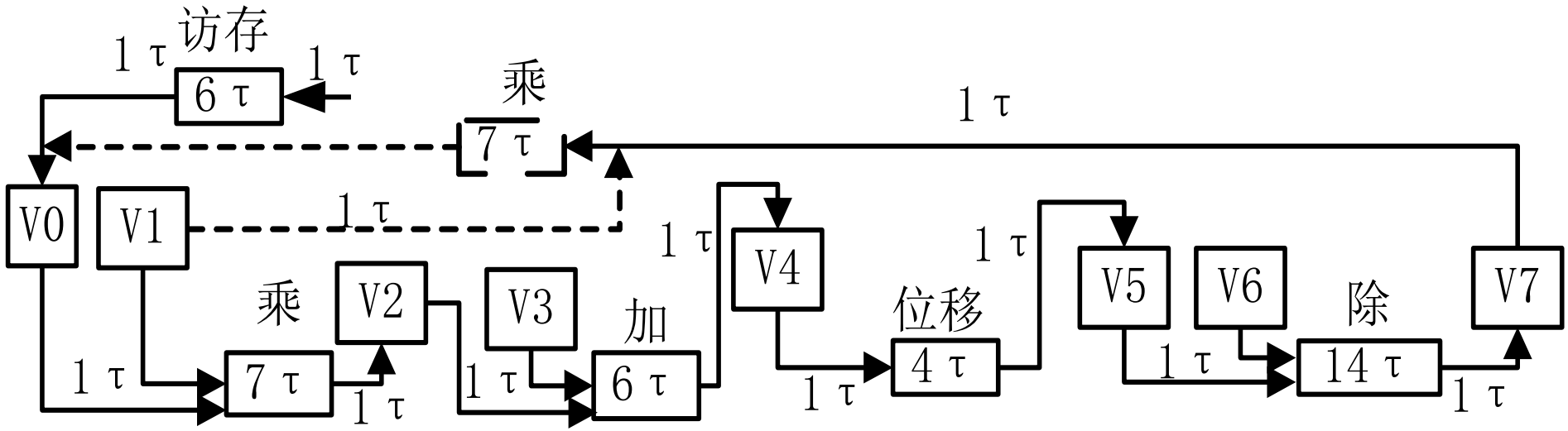
从第一次送出到第二次送出之前1 τ

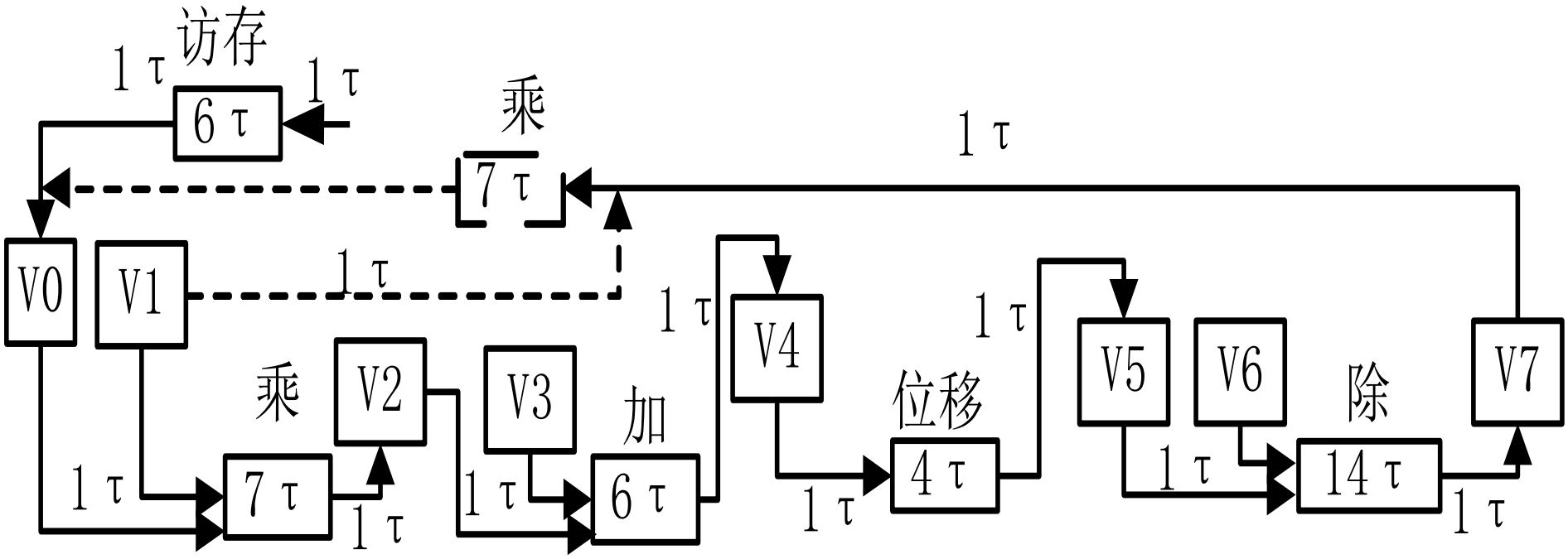
②目冲突：

从第一次接收到第二次接收之前1 τ

③功能块：

从第一次送出到第二次送入之前1 τ





源冲突（V1）：

1+7+1+1+6+1+1+4+1+1+14+1=39（τ）

目冲突（V0）：

1+1+7+1+1+6+1+1+4+1+1+14+1+1+7=48（τ）

功能块（×）：

1+1+6+1+1+4+1+1+14+1=31（τ）

说明：乘法功能部件冲突最严重，上述三个时间以最短时间为准（仅适用本例）。

3）推迟时间计算：

①当长度大于最短有关时间时，实际需要推迟时间为：

向量时间 － 有关时间

②当长度小于等于有关时间时，实际不用推迟可视为表面冲突。

本例推迟时间为：64-31=33（τ）

4）完成运算用时计算：顺利连接时间+推迟时间

1+6+1+1+7+1+1+6+1+1+4+1+1+14+1+1+7+1+（64-1）+33=152(τ)

5）性能：

6*64/（152*10^{-8}）≈253M FLOPS

练习题：:在CRAY-1机上，在下列指令组中，组内哪些指令可以链接？哪些不可以链接？不能链接的原因是什么？完成各指令所需的拍数（设向量长度均为64,打入寄存器及启动功能部件各需1τ）。

（1）V0←存储器(6τ)；V1←V2+V3(6τ)；
V4←V5×V6(7τ)

（2）V2←V0×V1；V3←存储器；V4←V2+V3

（3）V0←存储器；
V2←V0×V1；V3←V2+V0；V6←V3+V4

（4）V0←存储器；
V1←1/V0(14τ)　　；　　V3←V1×V2；V5←V3+V4

解：

（1）既不相关又不冲突—并行执行（不可链接）

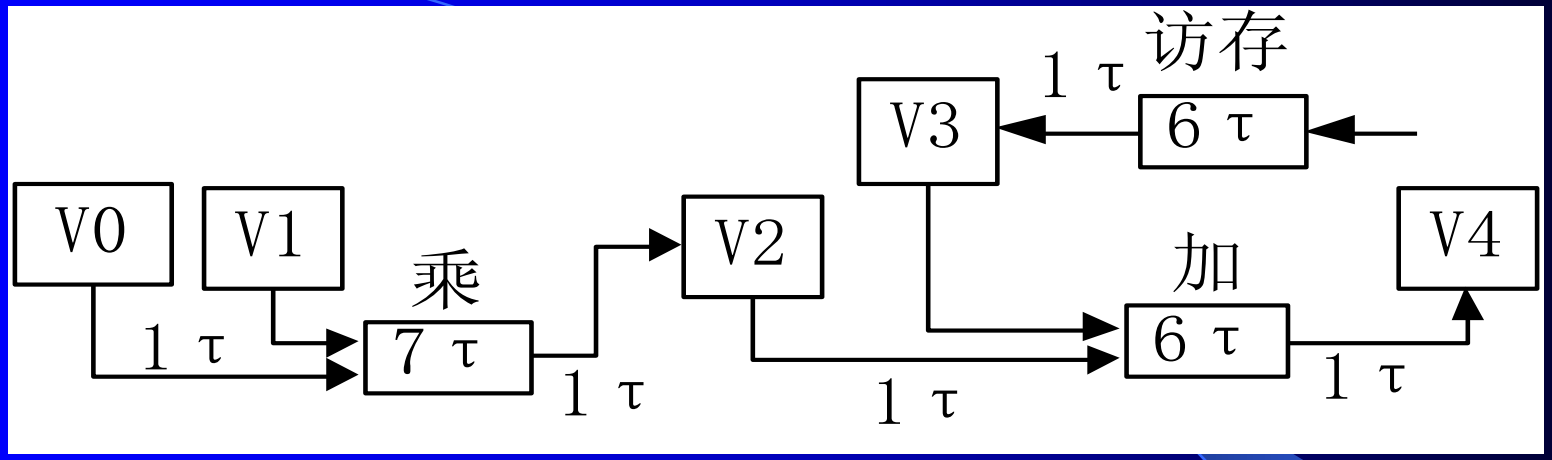
1+7+1+（64-1）=72（τ）

3*64/（72*10^{-8} ）≈267MFLOPS

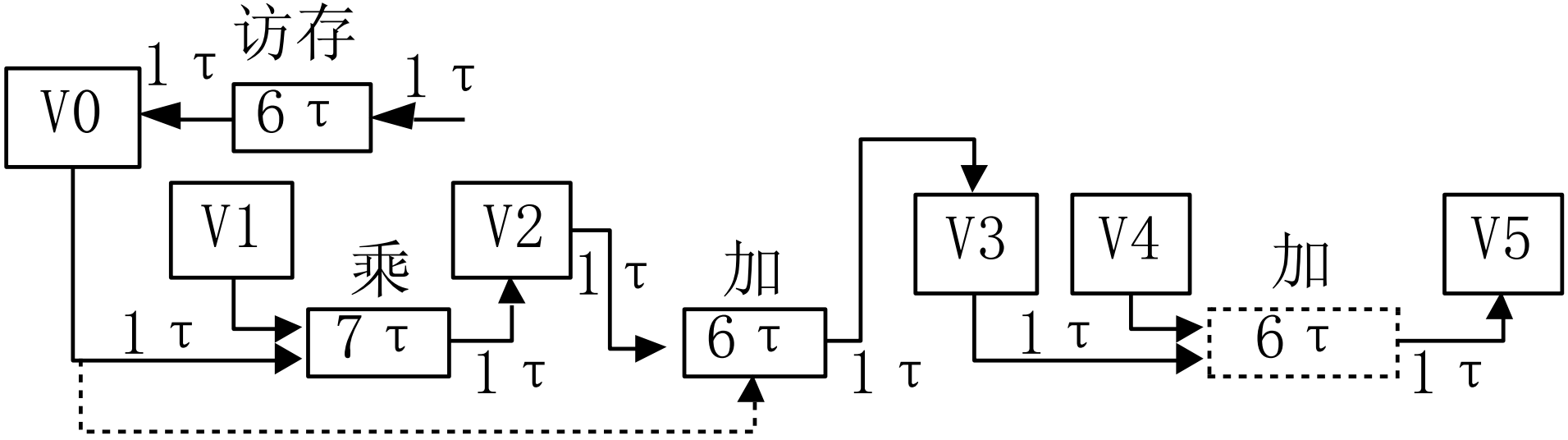
（2）有相关，不冲突——可链接

1+7+1+1+6+1+（64-1）=80（τ）

3*64/（80*10^{-8}）= 240 MFLOPS



（3）条条指令相关，但有冲突——不能顺利链接



源冲突（V1）：

1+7+1=9（τ） →推迟 64-9=55 τ

功能块冲突（加）：

1 τ →推迟 64-1=63 τ

总推迟：

55+63=118（τ）

用时：

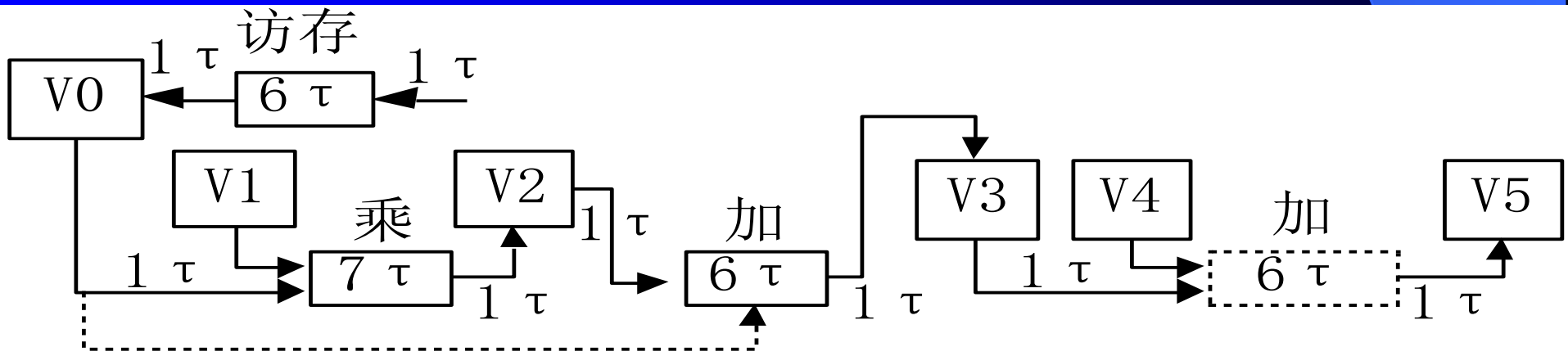
1+6+2+7+2+6+2+6+1+（64-1）+118

=214（τ）

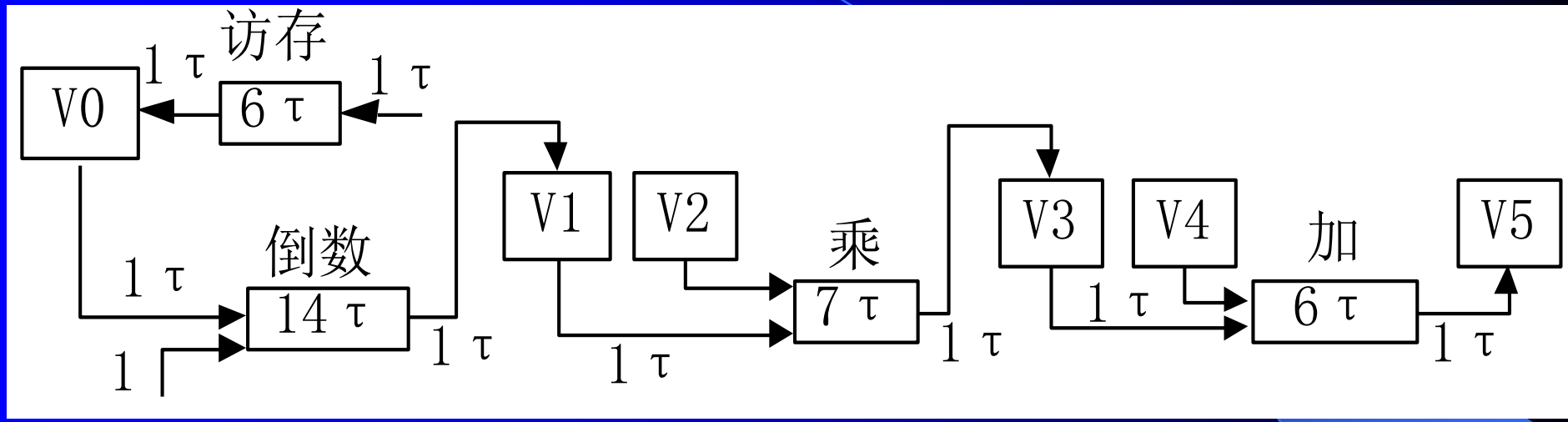
性能：

$4*64/(214*10^{-8})$

≈120M FLOPS



（4）条条相关，且无冲突——可顺利链接



用时：

1+6+2+14+2+7+2+6+1+（64-1）=104（τ）

性能：

$4*64/(104*10^{-8}) \approx 246M$ FLOPS