并 行 计 算

——结构•算法•编程

主讲教师： 谢磊

第二篇 并行算法的设计

第四章 并行算法的设计基础

第五章 并行算法的一般设计方法

第六章 并行算法的基本设计技术

第七章 并行算法的一般设计过程

第六章 并行算法的基本设计技术

6.1 划分设计技术

6.2 分治设计技术

6.3 平衡树设计技术

6.4 倍增设计技术

6.5 流水线设计技术

6.1 划分设计技术

- 6.1.1 均匀划分技术
- 6.1.2 方根划分技术
- 6.1.3 对数划分技术
- 6.1.4 功能划分技术

归并排序（Merge Sort）

* A merge sort works as follows:
  * If the list is of length 0 or 1, then it is already sorted. Otherwise:
  * Divide the unsorted list into two sublists of about half the size.
  * Sort each sublist recursively by re-applying the merge sort.
  * Merge the two sublists back into one sorted list.

归并排序（Merge Sort）

* Merge sort incorporates two main ideas to improve its runtime:
  * A small list will take fewer steps to sort than a large list.
  * Fewer steps are required to construct a sorted list from two sorted lists than from two unsorted lists. For example, you only have to traverse each list once if they're already sorted.

归并排序示例

均匀划分技术

每组$\frac{n}{p}$个. $A[\frac{(i-1)n}{p}+1, \dots, \frac{i \cdot n}{p}]$

* 划分方法
  n个元素A[1..n]分成p组，每组A[(i-1)n/p+1..in/p]，i=1~p
* 示例： MIMD-SM模型上的PSRS排序

begin
(1)均匀划分：将n个元素A[1..n]均匀划分成p段，每个p_i处理
A[(i-1)n/p+1..in/p]
(2)局部排序：p_i调用串行排序算法对A[(i-1)n/p+1..in/p]排序
(3)选取样本：p_i从其有序子序列A[(i-1)n/p+1..in/p]中选取p个样本元素
(4)样本排序：用一台处理器对p^2个样本元素进行串行排序
(5)选择主元：用一台处理器从排好序的样本序列中选取p-1个主元，并播送给其他p_i
(6)主元划分：p_i按主元将有序段A[(i-1)n/p+1..in/p]划分成p段
(7)全局交换：各处理器将其有序段按段号交换到对应的处理器中
(8)归并排序：各处理器对接收到的元素进行归并排序
end.

均匀划分技术

* 例6.1 PSRS排序过程。N=27，p=3，PSRS排序如下：

(a) 均匀划分：15 46 48 93 39 6 72 91 14 | 36 69 40 89 61 97 12 21 54 | 53 97 84 58 32 27 33 72 20

(b) 局部排序：6 14 15 39 46 48 72 91 93 | 12 21 36 40 54 61 69 89 97 | 20 27 32 33 53 58 72 84 97

p^2

(c) 正则采样：6 39 72 12 40 69 20 33 72

(d) 采样排序：6 12 20 33 39 40 69 72 72

$p-1$

(e) 选择主元：33 69

→ 接近等分($\frac{1}{p}, \frac{2}{p}, \cdots, \frac{p-1}{p}$)

(f) 主元划分：6 14 15 ⋮ 39 46 48 72 91 93 | 12 21 ⋮ 36 40 54 61 ⋮ 69 89 97 | 20 27 32 33 ⋮ 53 58 ⋮ 72 84 97

(g) 全局交换：6 14 15 12 21 20 27 32 33 | 39 46 48 36 40 54 61 69 53 58 | 72 91 93 89 97 72 84 97

(h) 归并排序：6 12 14 15 20 21 27 32 33 | 36 39 40 46 48 53 54 58 61 69 | 72 72 84 89 91 93 97 97

图6. 1

6.1 划分设计技术

- 6.1.1 均匀划分技术
- 6.1.2 方根划分技术
- 6.1.3 对数划分技术
- 6.1.4 功能划分技术

临视 cpu时.

方根划分技术

* 划分方法 $\sqrt{(i-1)n}+1$, $\sqrt{in}$ 划分为$\sqrt{n}$组

n个元素A[1..n]分成A[(i-1)n^(1/2)+1..in^(1/2)]，i=1~n^(1/2)

* 示例： SIMD-CREW模型上的 $k=\lfloor\sqrt{pq}\rfloor$ Valiant归并(1975年发表)

//有序组A[1..p]、B[1..q], (假设p<=q), 处理器数 $k=\lfloor\sqrt{pq}\rfloor$

begin

(1)方根划分: A,B分别按 $i\lfloor\sqrt{p}\rfloor$和 $j\lfloor\sqrt{q}\rfloor$分成若干段（$i=1\sim\lfloor\sqrt{p}\rfloor$、$j=1\sim\lfloor\sqrt{q}\rfloor$）；

(2)段间比较: A划分元与B划分元比较(至多 $\lfloor\sqrt{p}\rfloor\cdot\lfloor\sqrt{q}\rfloor$对),

确定A划分元应插入B中的区段;

(3)段内比较: A划分元与B相应段内元素进行比较，并插入适当的位置;

(4)递归归并: B按插入的A划分元重新分段，与A相应段(A除去原划分元)

构成了成对的段组，对每对段组递归执行(1)~(3)，直至A

组为0时，递归结束; 各组仍按 $k=\lfloor\sqrt{pq}\rfloor$分配处理器;

end.

方根划分技术

■ 示例: A={1,3,8,9,11,13,15,16},p=8; B={2,4,5,6,7,10,12,14,17},q=9

段间比较

(1)(2) A: 1 3 8* 9 11 13* 15 16 $(p=8,\lceil\sqrt{p}\rceil=3,\lfloor\sqrt{p}\rfloor=2)$

B: 2 4 5* 6 7 10* 12 14 17* $(q=9,\lceil\sqrt{q}\rceil=3,\lfloor\sqrt{q}\rfloor=3)$

⇩

段内比较

(3) A: 1 3 8* 9 11 13* 15 16

B: 2 4 5* 6 7 10* 12 14 17*

⇩

A_1: 1 3 $(p_1=2)$ | A_2: 9 11 $(p_2=2)$ | A_2: 15 16

B_1: 2 4 5 6 7 8$(q_1=6)$ | B_2: 10 12 13$(q_2=3)$ | B_3: 14 17

⇩

A_1: 1 3* $(p_1=2)$ | ...... | ......

B_1: 2 4 5* 6 7 8*$(q_1=6)$ | ...... | ......

⇩

A_{11}: 1* | A_{11}: | ...... | ......

B_{11}: 2 3* | B_{12}: 4 5 6 7 8 | ...... | ......

⇩

A:

B: 1 2 3 | 4 5 6 7 8 | 9 10 11 | 12 13 | 14 15 16 | 17

方根划分技术

- 算法分析

(1)算法在并行递归过程中所需的处理器数⩽ $k=\lfloor\sqrt{pq}\rfloor$

段间比较:$\lfloor\sqrt{p}\rfloor\cdot\lfloor\sqrt{q}\rfloor$比较对数⩽ $\lfloor\sqrt{pq}\rfloor=k$；

段内比较: $\lfloor\sqrt{p}\rfloor\cdot(\lfloor\sqrt{q}\rfloor-1)\le\lfloor\sqrt{pq}\rfloor=k$

递归调用：设 **A,B** 分成若干子段对为**(p_1,q_1), (p_2,q_2),……**

则Σ**p_i⩽p**，Σ**q_i⩽q**， 由 Cauchy 不等式=>

$$\sum\lfloor\sqrt{p_i q_i}\rfloor\le\lfloor\sum\sqrt{p_i q_i}\rfloor\le\lfloor\sqrt{\sum p_i\sum q_i}\rfloor\le\lfloor\sqrt{pq}\rfloor=k$$

综上，整个过程可用处理器数 $k=\lfloor\sqrt{pq}\rfloor$完成。

(2)时间分析

记 λ$_i$ 是第 **i** 次递归后的 **A** 组最大长度，=> $\lambda_0=p$， $\lambda_i\le\lfloor\sqrt{\lambda_{i-1}}\rfloor\le\cdots\le\lfloor p^{2^{-i}}\rfloor$

算法在 $\lambda_i=常数C$ 时终止递归，即 $\lfloor p^{2^{-i}}\rfloor\ge常数C$ => $i\le\log\log p+常数C_1$

由**(1)**知算法中其他各步的时间为 **O(1)**，所以 **Valiant** 归并算法时间

$$t_k(p,q)=O(\log\log p)\qquad p\le q$$

6.1 划分设计技术

6.1.1 均匀划分技术

6.1.2 方根划分技术

6.1.3 <u>对数划分技术</u>

6.1.4 功能划分技术

对数划分技术

* 关于划分技术的思考
  * 为了确定A序列中的划分元素在B序列中的全局位置，所以划分元素必须在各段之间实行全局比较。
  * 如果在选取A序列中的划分元素时，就考虑到了它在B序列中的全局位序，那么就不必对划分元素施行段间的全局比较，而可直接对按划分元素所断开的各段组两两进行归并，便可完成两个原序列的归并。

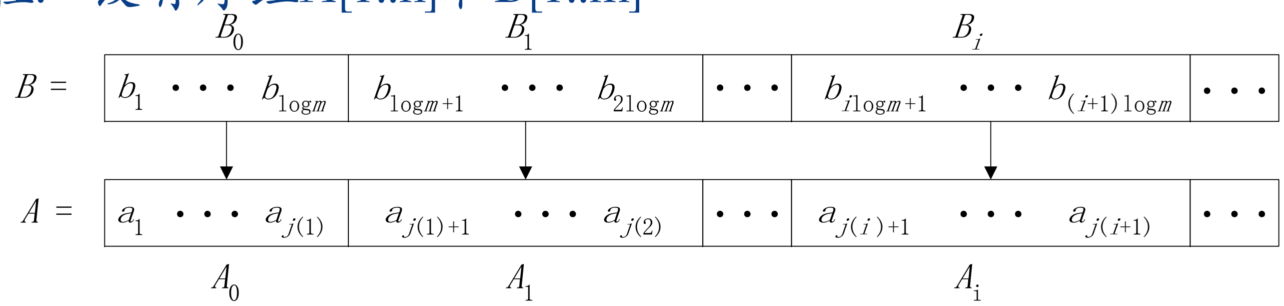
对数划分技术

* 划分方法

n个元素A[1..n]分成A[(i-1)logn+1..ilogn]，i=1~n/logn

* 示例： PRAM-CREW上的对数划分并行归并排序

(1)归并过程：设有序组A[1..n]和B[1..m]



j[i]=rank(b_{ilogm}:A)为b_{ilogm}在A中的位序，即A中小于等于b_{ilogm}的元素个数

(2)例： A=(4,6,7,10,12,15,18,20), B=(3,9,16,21) n=8, m=4

=>logm=log4=2

=> j[1]=rank(b_{logm}:A)=rank(b_2:A)=rank(9:A)=3, j[2]=⋯=8

B_0: 3, 9 B_1: 16, 21

A_0: 4, 6, 7 A_1: 10, 12, 15, 18, 20

A和B归并化为(A_0, B_0)和(A_1, B_1)的归并

6.1 划分设计技术

6.1.1 均匀划分技术

6.1.2 方根划分技术

6.1.3 对数划分技术

6.1.4 功能划分技术

功能划分技术

* 问题背景
  * 假定欲从长为n的序列中选取前m个最小者，此即所谓的(m,n)-选择问题，那么应如何对原序列施行划分以便并行处理呢?
  * 此时可以使用所谓功能划分法，即将长为n的序列划分成等长的一些组，每组中的元素应大于或等于m（最后一组除外）。然后各组可并行处理。

功能划分技术

* 划分方法

  n个元素A[1..n]分成等长的p组，每组满足某种特性。

* 示例：(m, n)选择问题(求出n个元素中前m个最小者)
  * 功能划分：要求每组元素个数必须大于m；
  * 算法：

  输入：A=(a1,⋯,an)；输出：前m个最小者；

  Begin

  (1) 功能划分：将A划分成g=n/m组，每组含m个元素；

  (2) 局部排序：使用Batcher排序网络将各组并行进行排序；

  (3) 两两比较：将所排序的各组两两进行比较，从而形成MIN序列；

  (4) 排序-比较：对各个MIN序列，重复执行第(2)和第(3)步，直至选出m个最小者。

  End

功能划分技术

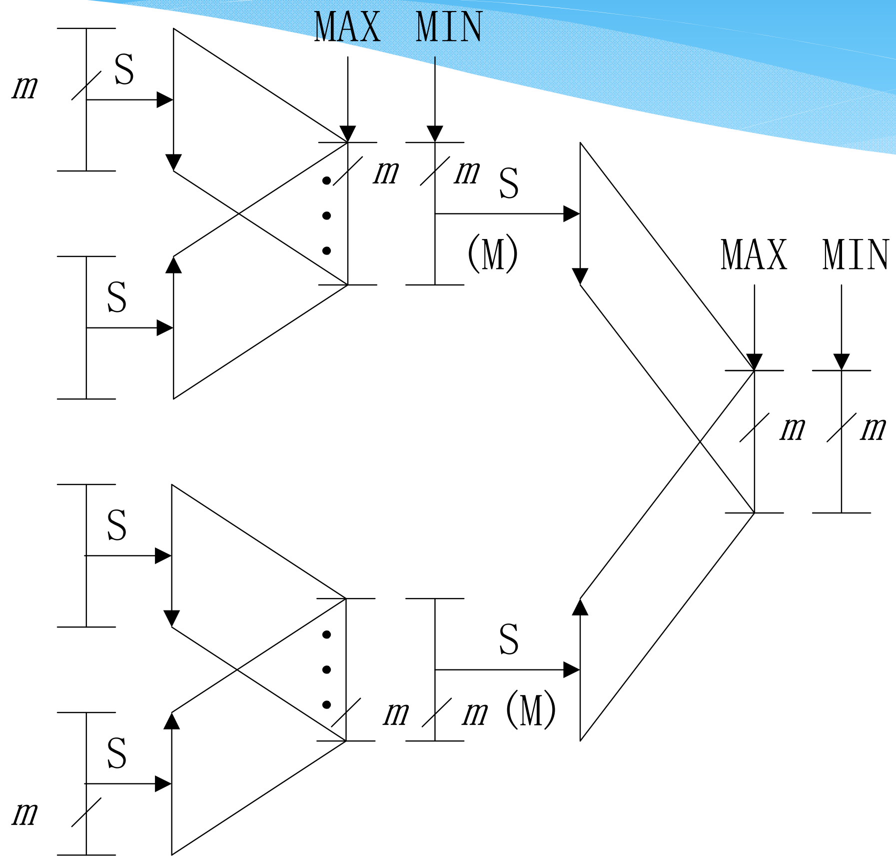


图6.3 (m-n)-选择过程

第六章 并行算法的基本设计技术

6.1 划分设计技术

6.2 分治设计技术

6.3 平衡树设计技术

6.4 倍增设计技术

6.5 流水线设计技术

6.2 分治设计技术

6.2.1 并行分治设计步骤

6.2.2 双调归并网络

6.2.3 并行快速排序

并行分治设计步骤

* 并行分治法分为三步:
  * (1)将输入划分成若干个规模相等的子问题;
  * (2)同时(并行地)递归求解这些子问题;
  * (3)并行地归并子问题的解，直至得到原问题的解。
* 分治法与划分法求解问题的共同点在于:
  * 两者均试图将原问题划分成可并行求解的子问题;
* 分治法与划分法求解问题的不同点在于:
  * 分治法的侧重点在于子问题的归并上;
  * 划分法的注意力则集中在原问题的划分上。

6.2 分治设计技术

6.2.1 并行分治设计步骤

6.2.2 双调归并网络

双调归并网络

* 双调序列(p145定义6.2)

  (1,3,5,7,8,6,4,2,0)

  (8,7,6,4,2,0,1,3,5) n=9

  (1,2,3,4,5,6,7,8)

  以上都是双调序列

  允许数列中出现至多一次顺序转换.

* Batcher定理

  给定双调序列$(x_0,x_1,\cdots,x_{n-1})$, 对于所有的$0\le i \le \frac{n}{2}-1$, 执行x_i和$x_{i+n/2}$的比较交换得到 $s_i=\min\{x_i,x_{i+n/2}\}$和$l_i=\max\{x_i,x_{i+n/2}\}$。则

  (1)小序列$(s_0,s_1,\cdots,s_{n-1})$和大序列$(l_0,l_1,\cdots,l_{n-1})$仍是双调序列;

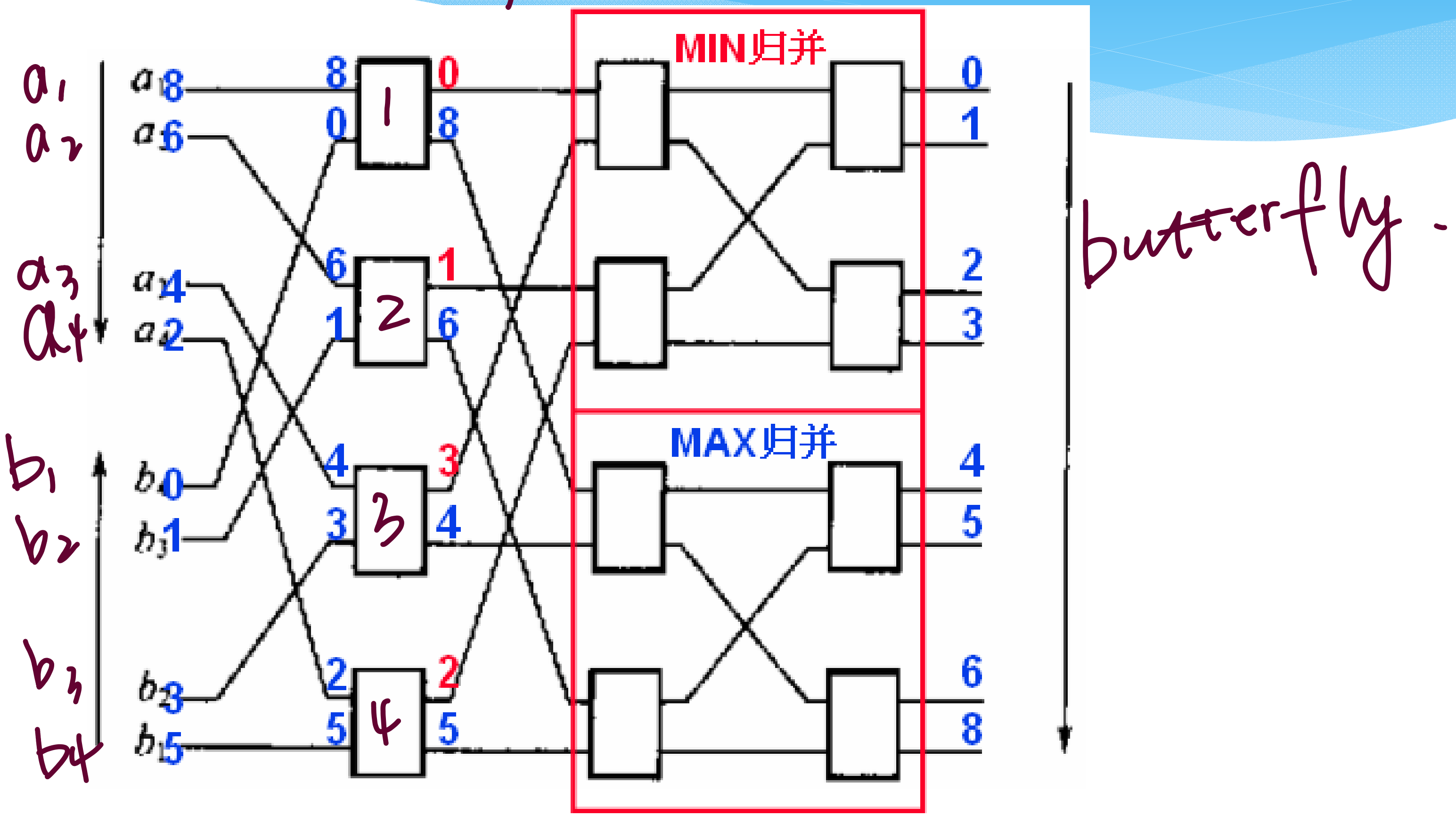
  (2)对于所有的$0\le i,j \le \frac{n}{2}-1$,满足$s_i \le l_j$。

双调归并网络

* 根据Batcher定理，对双调序列(1,3,5,7,8,6,4,2,0)进行划分：
  * n=9,[n/2]=4
  * $x0=1,x4=8 \rightarrow s0=1, l0=8$
  * $x1=3,x5=6 \rightarrow s1=3, l1=6$
  * $x2=5,x6=4 \rightarrow s2=4, l2=5$
  * $x3=7,x7=2 \rightarrow s3=2, l3=7$
  * $x8=0 \rightarrow s4=0$

双调归并网络

* (4,4)双调归并网络 对称点 .



两两比较　　2个(2,2)双调归并网络

双调归并网络

* Batcher双调归并算法

输入：双调序列$X=(x_0,x_1,\ldots,x_{n-1})$

输出：非降有序序列$Y=(y_0,y_1,\ldots,y_{n-1})$

```
Procedure BITONIC_MERG(x)
Begin
  (1)for i=0 to n/2-1 par-do
     (1.1) s_i=min{xi,xi+n/2}
     (1.2) l_i=max{xi,xi+n/2}
     end for
  (2)Recursive Call:
     (2.1)BITONIC_MERG(MIN=(s_0,…,s_{n/2-1}))
     (2.2)BITONIC_MERG(MIN=(l_0,…, l_{n/2-1}))
  (3)output sequence MIN followed by sequence MAX
End
```

第六章 并行算法的基本设计技术

6.1 划分设计技术

6.2 分治设计技术

6.3 平衡树设计技术

6.4 倍增设计技术

6.5 流水线设计技术

6.3 平衡树设计技术

- 6.3.1 设计思想
- 6.3.2 求最大值
- 6.3.3 计算前缀和

平衡树设计技术

* 设计思想

  以树的叶结点为输入，中间结点为处理结点，由叶向根或由根向叶逐层进行并行处理。

* 示例
  * 求最大值
  * 计算前缀和

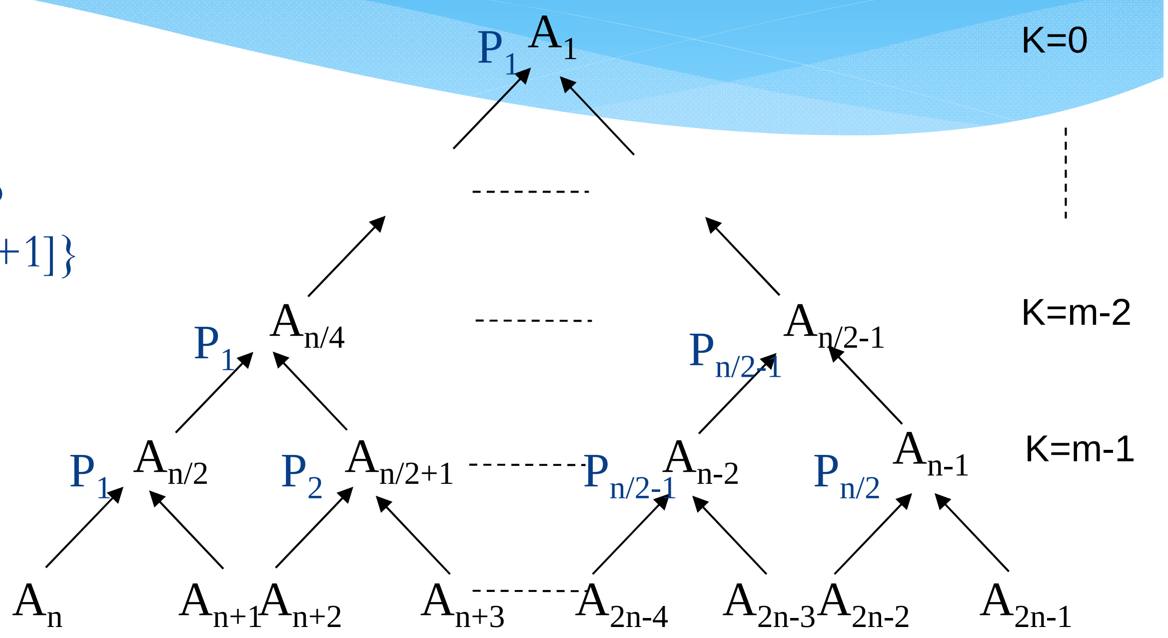
6.3 平衡树设计技术

- 6.3.1 设计思想
- 6.3.2 <u>求最大值</u>
- 6.3.3 计算前缀和

求最大值

* 算法6.8: SIMD-TC(SM)上求最大值算法

```
Begin
  for k=m-1 to 0 do
    for j=2^k to 2^(k+1)-1 par-do
      A[j]=max{A[2j], A[2j+1]}
    end for
  end for
end
```



* 图示
* 时间分析

$t(n)=m\times O(1)=O(\log n)$

$p(n)=n/2$

6.3 平衡树设计技术

6.3.1 设计思想

6.3.2 求最大值

6.3.3 计算前缀和

计算前缀和

- 问题定义

n个元素{$x_1,x_2,\cdots,x_n$}，前缀和是n个部分和：

$S_i=x_1*x_2*\cdots*x_i$，$1\leqslant i\leqslant n$　这里*可以是＋或×

- 串行算法：　$S_i=S_{i-1}*x_i$　　计算时间为O(n)
- 并行算法：p150算法6.9 SIMD-TC上非递归算法

令$A[i]=x_i$, $i=1\sim n$,

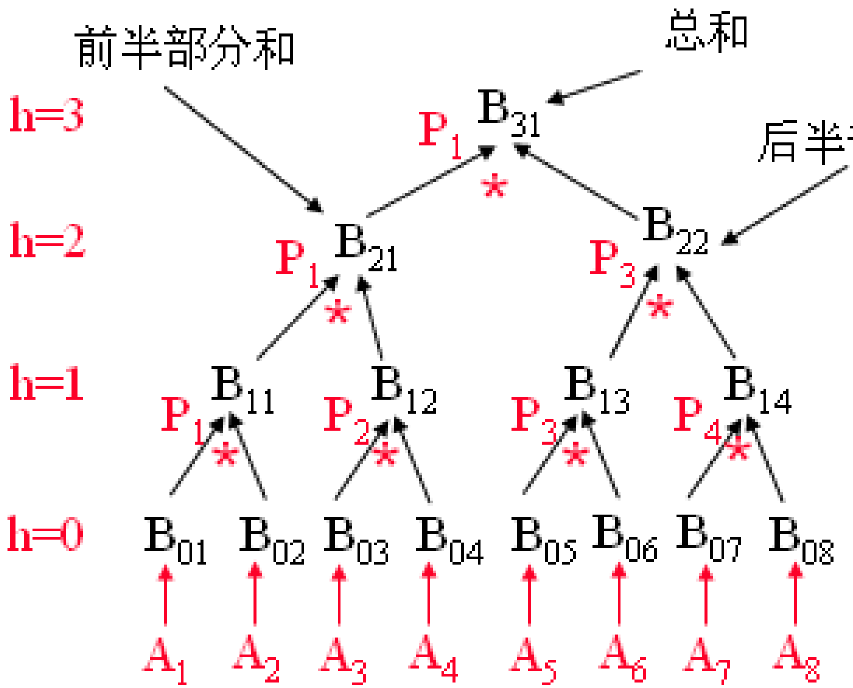
B[h,j]和C[h,j]为辅助数组($h=0\sim\log n$, $j=1\sim n/2^h$)

数组B记录由叶到根正向遍历树中各结点的信息(求和)

数组C记录由根到叶反向遍历树中各结点的信息(播送前缀和)

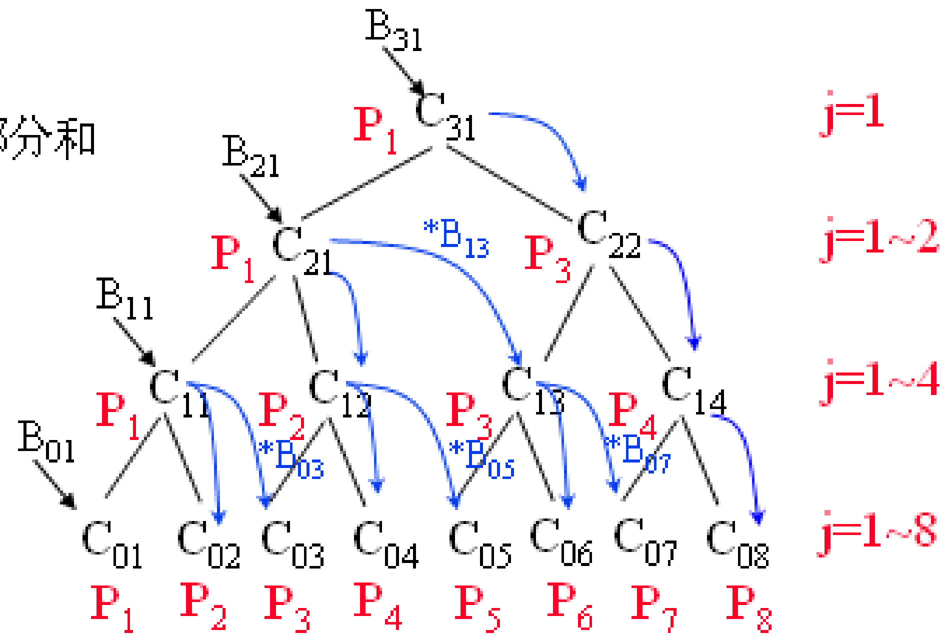
计算前缀和

* 例： $n=8, p=8, C_{01} \sim C_{08}$为前缀和



计算：B[h,j]=B[h-1,2j-1]*B[h-1,2j]

正向遍历(求和)



计算：C[h,1]=B[h,1]　　j=1

C[h,j]=C[h+1, j/2]　　j=even

C[h,j]=C[h+1,(j-1)/2]*B[h, j]　　j=odd>1

反向遍历(播送前缀和)

第六章 并行算法的基本设计技术

6.1 划分设计技术

6.2 分治设计技术

6.3 平衡树设计技术

6.4 倍增设计技术

6.5 流水线设计技术

6.4 倍增设计技术

6.4.1 设计思想

6.4.2 表序问题

6.4.3 求森林的根

倍增设计技术

* 设计思想
  * 又称指针跳跃(pointer jumping)技术，特别适合于处理链表或有向树之类的数据结构；
  * 当递归调用时，所要处理数据之间的距离逐步加倍，经过k步后即可完成距离为2^k的所有数据的计算。
* 示例
  * 表序问题
  * 求森林的根

6.4 倍增设计技术

6.4.1 设计思想

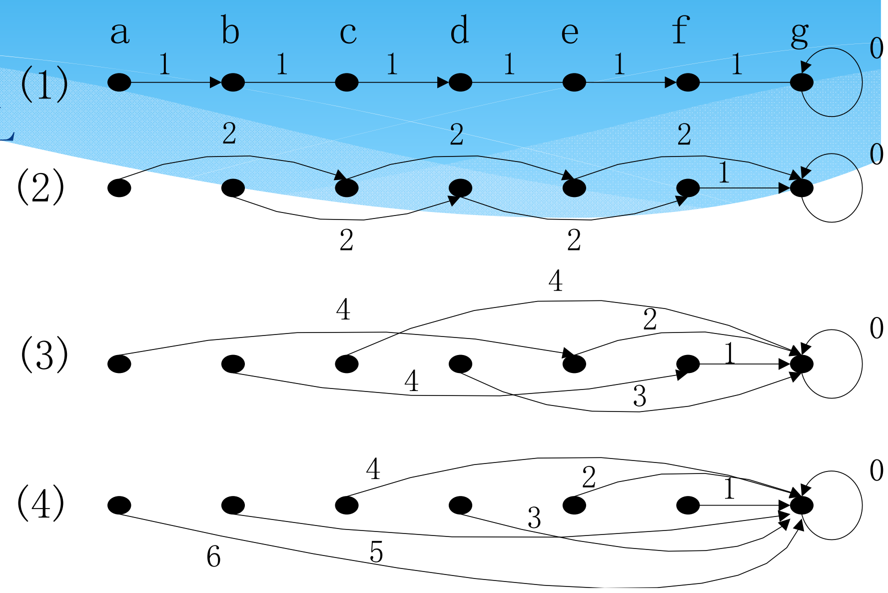
6.4.2 表序问题

6.4.3 求森林的根

表序问题

* 问题描述

n个元素的列表L，求出每个元素在L中的次第号(秩或位序或rank(k))，rank(k)可视为元素k至表尾的距离；



* 示例： n=7

(1)p[a]=b, p[b]=c, p[c]=d, p[d]=e,
p[e]=f, p[f]=g, p[g]=g
r[a]=r[b]=r[c]=r[d]=r[e]=r[f]=1, r[g]=0

(2)p[a]=c, p[b]=d, p[c]=e, p[d]=f, p[e]=p[f]=p[g]=g
r[a]=r[b]=r[c]=r[d]=r[e]=2, r[f]=1, r[g]=0

(3)p[a]=e, p[b]=f, p[c]=p[d]=p[e]=p[f]=p[g]=g
r[a]=4, r[b]=4, r[c]=4, r[d]=3, r[e]=2, r[f]=1, r[g]=0

(4)p[a]=p[b]=p[c]=p[d]=p[e]=p[f]=p[g]=g
r[a]=6, r[b]=5, r[c]=4, r[d]=3, r[e]=2, r[f]=1, r[g]=0

表序问题

- 算法：P151算法6.10

(1)并行做：初始化p[k]和distance[k] //O(1)

(2)执行 $\lceil \log n \rceil$次 //O(logn)

(2.1)对k并行地做 //O(1)

如果k的后继不等于k的后继之后继，则

(i) distance[k]= distance[k]+ distance[p[k]]

(ii) p[k]=p[p[k]]

(2.2)对k并行地做

rank[k]=distance[k] //O(1)

运行时间：t(n)=O(logn)　p(n)=n

6.4 倍增设计技术

6.4.1 设计思想

6.4.2 表序问题

6.4.3 求森林的根

求森林的根

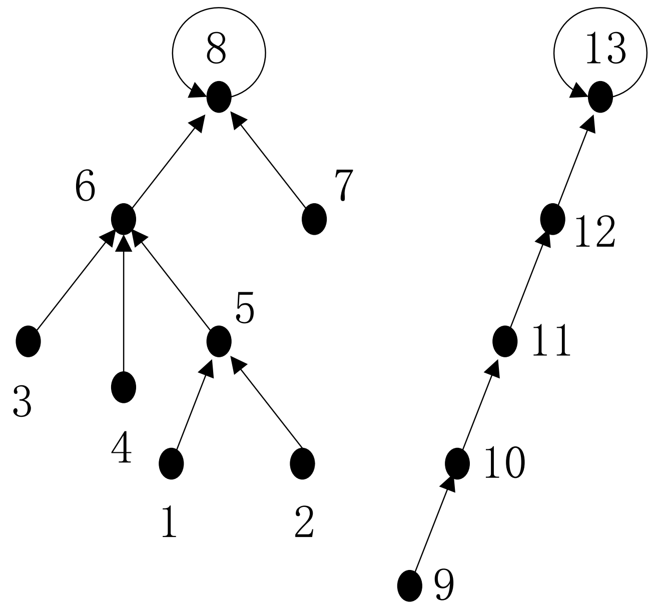
* 问题描述

  一组有向树F中，如果<i, j>是F中的一条弧， 则p[i]=j(即j是i的双亲)； 若i为根， 则p[i]=i。求每个结点j(j=1~n)的树根s[j].

* 示例

  初始时

  P[1]=p[2]=5   p[3]=p[4]=p[5]=6
  P[6]=p[7]=8   p[8]=8 P[9]=10
  p[10]=11 p[11]=12 p[12]=13 p[13]=13
  s[i]=p[i]

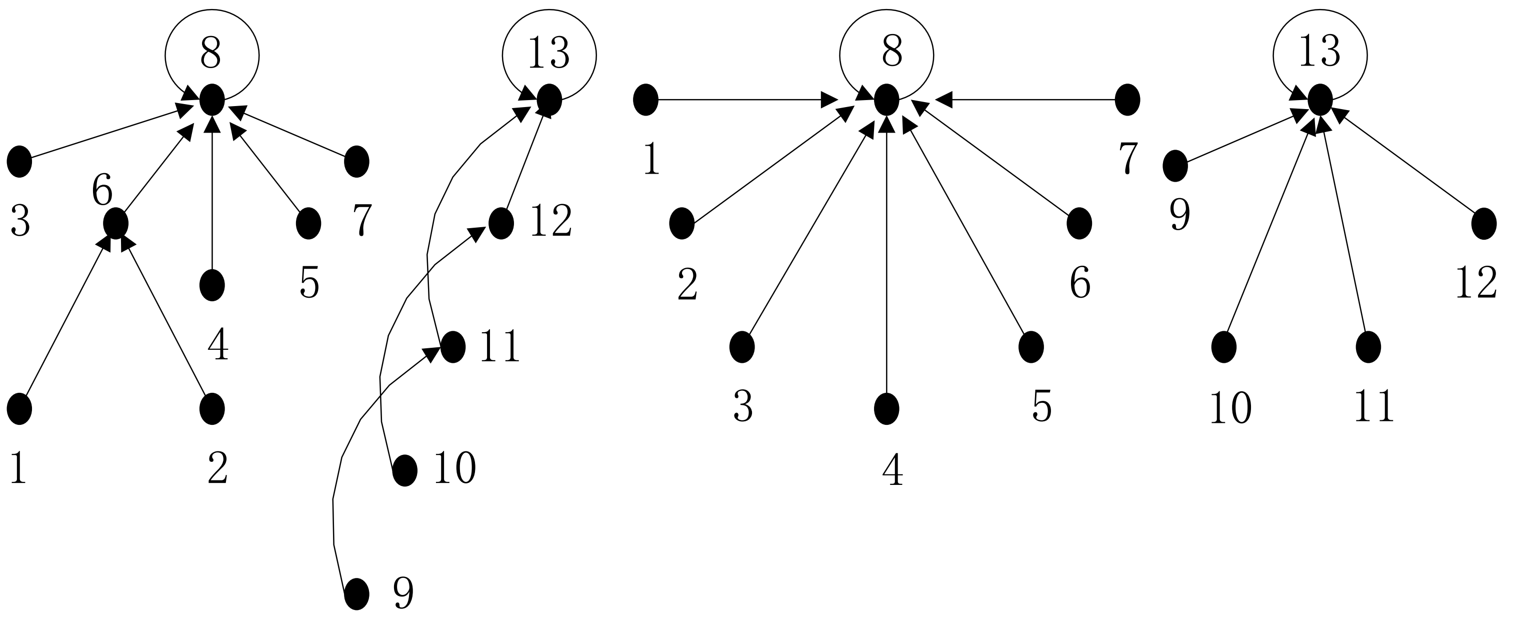


(a)

求森林的根

* 示例

第一次迭代后　　　　　　第二次迭代后



(b)　　　　　　(c)

求森林的根

- 算法：SIMD-CREW上求森林根的算法

输入：森林F，弧由（i,P(i)）指定，$1 \le i \le n$

输出：对每个节点i,输出包含i的树的根S(i)

Begin

for $1 \le i \le n$ par-do

(1) S(i)=P(i)

(2) while S(i)≠S(S(i)) do

S(i)=S(S(i))

endwhile

endfor

End

令h是森林中树的最大高度，不难看出算法将迭代O(logn)次，每次迭代做了O(n)次运算而花费了O(1)时间。所以算法的运行时间：t(n)=O(logn)，总运算量W(n)=O(nlogn)。

第六章 并行算法的基本设计技术

6.1 划分设计技术

6.2 分治设计技术

6.3 平衡树设计技术

6.4 倍增设计技术

6.5 流水线设计技术

6.5 流水线设计技术

- 6.5.1 设计思想
- 6.5.2 5-point DFT的计算

流水线设计技术

* 设计思想
  * 将算法流程划分成p个前后衔接的任务片断，每个任务片断的输出作为下一个任务片断的输入；
  * 所有任务片断按同样的速率产生出结果。
* 评注
  * 流水线技术是一种广泛应用在并行处理中的技术；
  * 脉动算法(Systolic algorithm)是其中一种流水线技术；

6.5 流水线设计技术

6.5.1 设计思想

6.5.2 5-point DFT的计算

5-point DFT的计算

* 问题描述
  * 一个n点的离散傅利叶变换（DFT），可定义为:给定序列(a0,a1,⋯,an-1),按如下规则变换成序列(b0,b1,⋯,bn-1):

$$b_j = \sum_{k=0}^{n-1} a_k w^{kj} \quad (0 \le j \le n)$$

做不同的加权，

参数

用统一逻辑门?

5-point DFT的计算。应用秦九韶(Horner)法则，

out/in

$$\begin{cases} y_0 = b_0 = a_4\omega^0 + a_3\omega^0 + a_2\omega^0 + a_1\omega^0 + a_0 \\ y_1 = b_1 = a_4\omega^4 + a_3\omega^3 + a_2\omega^2 + a_1\omega^1 + a_0 \\ y_2 = b_2 = a_4\omega^8 + a_3\omega^6 + a_2\omega^4 + a_1\omega^2 + a_0 \\ y_3 = b_3 = a_4\omega^{12} + a_3\omega^9 + a_2\omega^6 + a_1\omega^3 + a_0 \\ y_4 = b_4 = a_4\omega^{16} + a_3\omega^{12} + a_2\omega^8 + a_1\omega^4 + a_0 \end{cases} \Rightarrow \begin{cases} y_0 = (((a_4\omega^0 + a_3)\omega^0 + a_2)\omega^0 + a_1)\omega^0 + a_0 \\ y_1 = (((a_4\omega^1 + a_3)\omega^1 + a_2)\omega^1 + a_1)\omega^1 + a_0 \\ y_2 = (((a_4\omega^2 + a_3)\omega^2 + a_2)\omega^2 + a_1)\omega^2 + a_0 \\ y_3 = (((a_4\omega^3 + a_3)\omega^3 + a_2)\omega^3 + a_1)\omega^3 + a_0 \\ y_4 = (((a_4\omega^4 + a_3)\omega^4 + a_2)\omega^4 + a_1)\omega^4 + a_0 \end{cases}$$

等比 → 递归表示 | 递归定义更清爽

流水线{上一轮的输出→下一轮输入}

5-point DFT的计算

* 示例： p(n)=n-1, t(n)=2n-2=O(n)

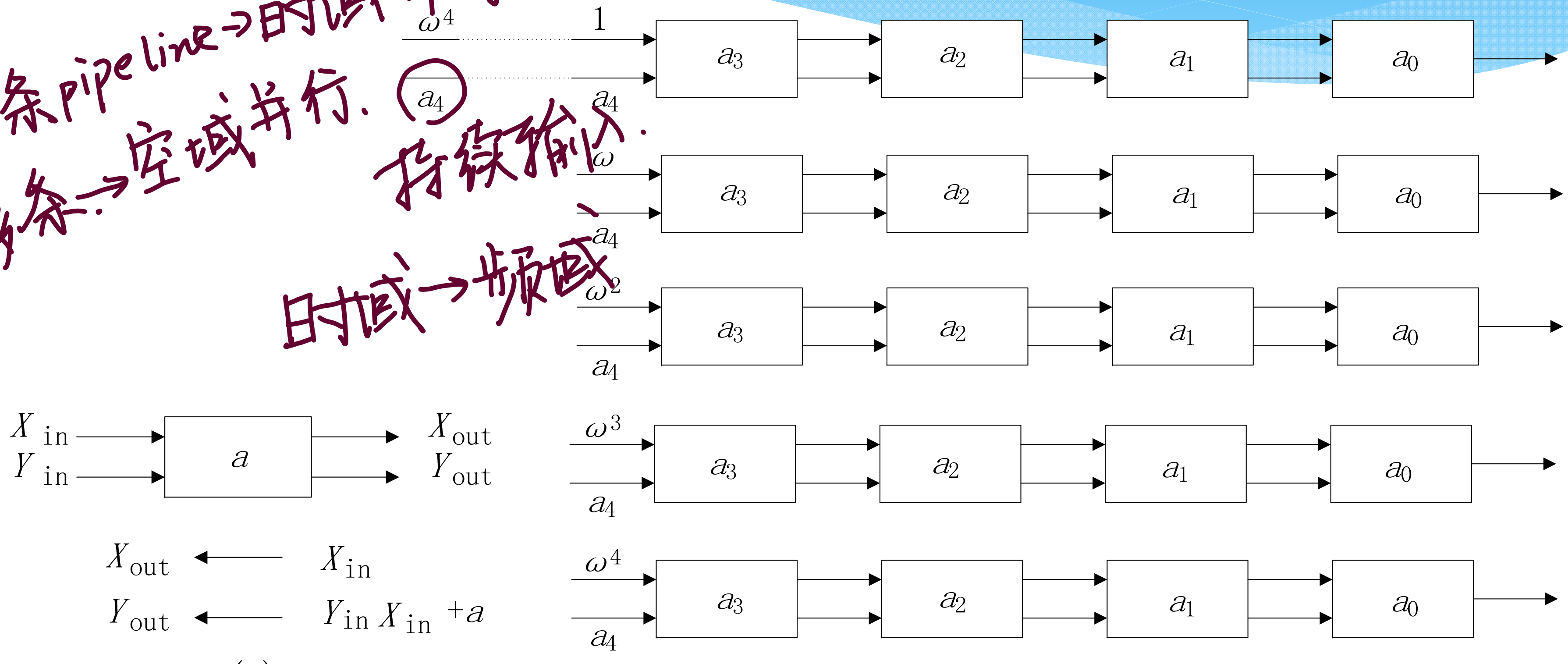
不同的部分放在不同的寄存器.

单条pipeline→时域并行.

多条→空域并行.

持续输入.

时域→频域



(a)

(b)

脉动算法.

pipeline特性.