注意联系 10.3 节的内容，其实对于一阶 IIR 节，离散超前和聚类超前都是一样的，没有区别。为了得到非递归部分的“优美”分解，往往将 M 设为 2 的幂次，即$M = 2^m$，如下代码，设置 m 为 3，设计 8 级离散超前的 IIR 二阶节，

代码 1　2 阶 IIR 节离散超前流水线设计，M 为 2 的幂次，能得到“优美”的分子分解

```
> restart :
>
> m := 3;
                              m := 3
> M := 2^m;
                              M := 8
> Dz := 1-5/4·p + 3/8·p^2;
                    Dz := 1 - 5/4 p + 3/8 p^2
> k := degree(Dz, p);
                              k := 2
> K := M·k;
                              K := 16
> Pz := 1 + sum(a_i·p^i, i = 1..K - k) :
>
> DP := collect(Dz·Pz, p) :
>
> eqns := {seq(coeff(DP, p, i) = 0, i = 1..K)} :
> eqns := eqns\{seq(coeff(DP, p, i·M) = 0, i = 1..k)} :
>
> vars := [seq(a_i, i = 1..K - k)] :
> sols := solve(eqns, vars) :
>
> subs(op(sols), [op(eqns)]);
      [0 = 0, 0 = 0, 0 = 0, 0 = 0, 0 = 0, 0 = 0, 0 = 0, 0 = 0, 0 = 0, 0 = 0, 0
         = 0, 0 = 0, 0 = 0, 0 = 0]
>
>
> DP := subs(op(sols), DP);
              DP := 1 + 6561/16777216 p^16 - 6817/65536 p^8
> Nz := factor(subs(op(sols), Pz));
```

$$Nz := \frac{1}{2097152}\,(3\,p+4)\,(p+2)\,\left(p^2+4\right)\,\left(9\,p^2+16\right)\,\left(81\,p^4+256\right)\,\left(p^4+16\right)$$

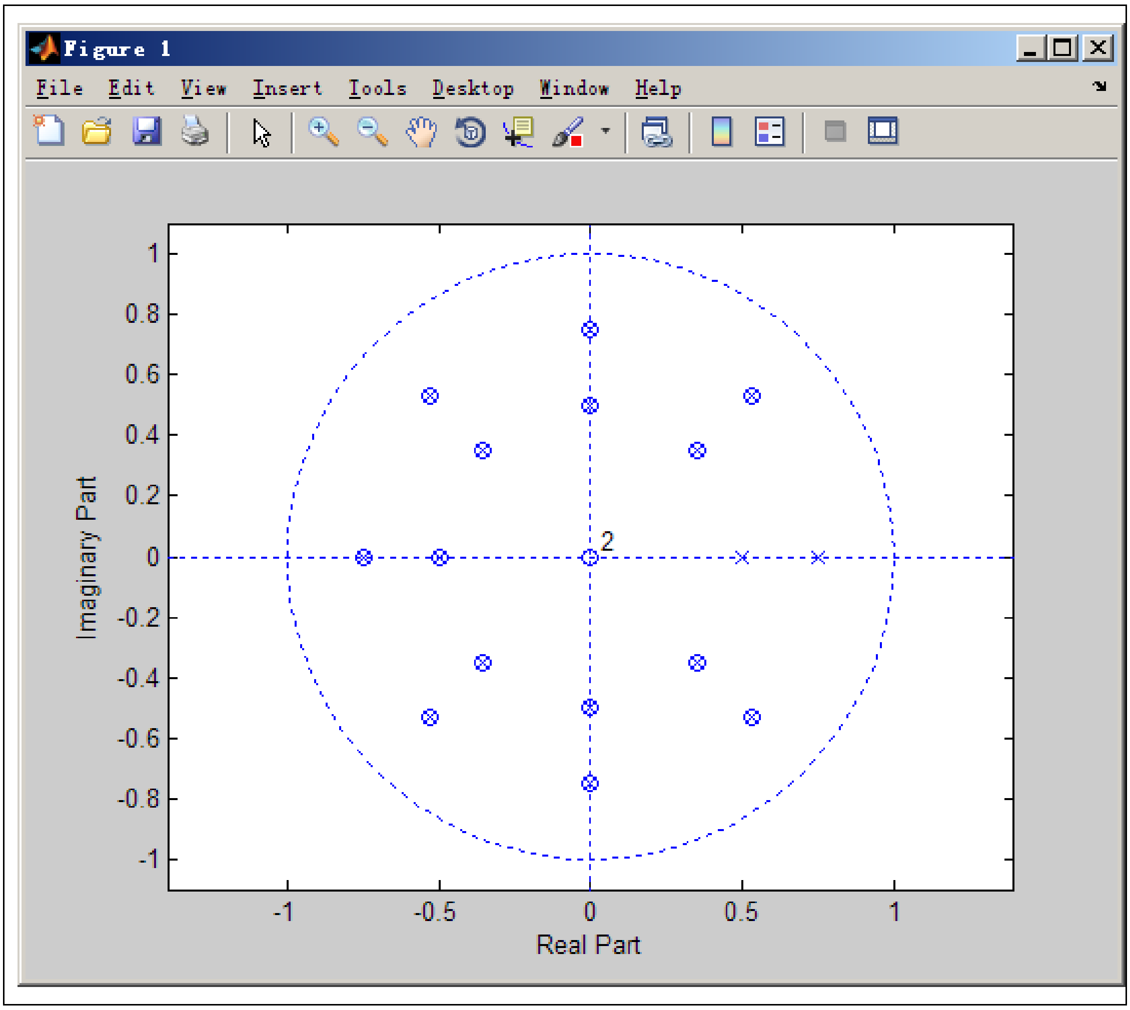
> $maplematrix\_b := Matrix([seq(coeff(Nz,\ p,\ i),\ i = 0..K - k)]);$

$$maplematrix\_b := \left[\begin{array}{c} \mathit{1\ x\ 15\ Matrix} \\ \mathit{Data\ Type{:}\ anything} \\ \mathit{Storage{:}\ rectangular} \\ \mathit{Order{:}\ Fortran\_order} \end{array}\right]$$

> $maplematrix\_a := Matrix([seq(coeff(DP,\ p,\ i),\ i = 0..K)]);$

$$maplematrix\_a := \left[\begin{array}{c} \mathit{1\ x\ 17\ Matrix} \\ \mathit{Data\ Type{:}\ anything} \\ \mathit{Storage{:}\ rectangular} \\ \mathit{Order{:}\ Fortran\_order} \end{array}\right]$$

```
> Matlab[setvar]("matlabmatrix_b", maplematrix_b);
> Matlab[setvar]("matlabmatrix_a", maplematrix_a);
> Matlab[evalM]("zplane(matlabmatrix_b,matlabmatrix_a);");
```



8 级流水的分子分解同图 14，其实从 maple 的默认因式分解即可看出，

$$Nz := \frac{1}{2097152}\,(3\;p+4)\;(p+2)\;\left(p^2+4\right)\;\left(9\;p^2+16\right)\;\left(81\;p^4+256\right)\;\left(p^4+16\right)$$

最后，给出一段通用的求解初始值的程序，只需将设计好的流水线 IIR 节输入，并给出原始 IIR 节的初值，即可自动求出新设计的 M 级流水 IIR 节初值，代码如下

代码 2　流水 IIR 节初值计算（限公式(10)的形式）

```
> restart :
>
> a := [-5/4, 3/8];
```

$$a := \left[-\frac{5}{4}, \frac{3}{8}\right]$$

```
> Y := [1/2, 1/2];
```

$$Y := \left[\frac{1}{2}, \frac{1}{2}\right]$$

$$B := convert\left(\left[\begin{array}{cccccccc} 1 & \frac{5}{4} & \frac{19}{16} & \frac{65}{64} & \frac{211}{256} & \frac{665}{1024} & \frac{2059}{4096} & \frac{6305}{16384} \end{array}\right], list\right):$$

$$A := convert\left(\left[\begin{array}{cccccccccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\frac{19171}{65536} & \frac{18915}{131072} \end{array}\right], list\right):$$

$M := nops(B);\ K := nops(A);$

$$M := 8$$

$$K := 10$$

$f := y(n) = -a_1 \cdot y(n-1) - a_2 \cdot y(n-2) + x(n):$
$T := [seq(0 = 0,\ i = 1..K-1)]:$
for k from 0 to $K-2$ do
$T_{K-k-1} := subs(n = k,\ f);$
od:
$S := [seq(0 = 0,\ i = 1..K-1)]:$
$S_{K-1} := T_{K-1}:$
for k from 1 to $K-2$ do
$S_k := subs\left(seq\left(T_i,\ i = k+1..K-1\right),\ T_k\right);$
od:

$f := y(n) = sum\left(-A_i \cdot y(n-i+1),\ i = 2..K\right) + sum\left(B_i \cdot x(n-i+1),\ i = 1..M\right):$
$eqns_0 := \{seq(x(i) = 0,\ i = -M+1..-1)\}:$
$eqns_1 := \{\}:$
for k from 0 to $K-4$ do
$eqns_1 := eqns_1$ union $\{subs(n = k,\ f)\};$
od:
$eqns_1 := subs\left(eqns_0,\ eqns_1\right):$
$eqns_1 := subs\left(S,\ eqns_1\right):$
$eqns_1 := subs\left(y(-1) = Y_1,\ y(-2) = Y_2,\ eqns_1\right):$

$solve\left(eqns_1,\ [seq(y(i),\ i = -K+1..-3)]\right);$

$$\left[\left[y(-9) = -\frac{123584}{2187},\ y(-8) = -\frac{19232}{729},\ y(-7) = -\frac{2864}{243},\ y(-6) = -\frac{392}{81},\ y(-5) = -\frac{44}{27},\ y(-4) = -\frac{2}{9},\ y(-3) = \frac{1}{3}\right]\right]$$

，，，课本上 10.4.6 介绍了两个 IIR 的受限设计技术，这个就留给数字信号处理高手自行研究吧。如果哪位研究透了，希望能共享一些可运行的代码，造福大家伙！

至此，我们基本把基于超前计算的递归流水线技术过了一遍，并且给出了一阶和二阶 IIR 节的例子。

课本中对于递归流水线提供了 M 倍降速和超前计算两种设计方法，两种方法各有其优缺点，大家应根据实际情况进行评估。关于设计结果的复杂度方面我们没有过多讨论，但那是很实用的内容，希望大家能自学一遍！

某些情况下，将 M 倍降速和超前计算结合起来将更为高效，这里给出一个简要的说明（课本 10.2.3 节），大家动手实践：

设计双通道复合的 4 级递归流水线 IIR 滤波器，传递函数如下

$$H\left(e^{j\omega}\right)=\frac{1}{\left(1+a_1z^{-1}+a_2z^{-2}\right)}\cdot\frac{1}{\left(1+a_3z^{-1}+a_4z^{-2}\right)} \tag{1}$$

假设系统时钟“硬”要求将乘法单元做成 4 级流水；现在要实现的系统，也就是公式(13)为两个 IIR 2 阶节级联。一个可行的设计方案是，利用 1 次超前构造 2 级流水的 IIR 2 阶节（注意，优先使用聚类超前，但如果所得系统不稳定，则改用离散超前）。2 级流水 IIR 节设计好了以后，再进行 2 倍降速，就能将两路系统复合在一起。解决问题的诀窍在于：先用超前计算构造“自交织”的流水 IIR 节，然后再用 M 倍降速构造“独立交织”的 IIR 节。

第一节、 IIR 滤波器的并行处理

回顾第三章的并行处理设计方法，其实很简单，L 路并行，就是将$n=k\cdot L+i$，其中$i=0,...,L-1$，代入原始迭代公式，，至少第三章设计并行 FIR 是这么做的。如果是 IIR 滤波器，也“照搬”会怎么样呢？以一阶 IIR 节为例，迭代公式如下

$$y\left(n\right)=a\cdot y\left(n-1\right)+b\cdot x\left(n\right) \tag{2}$$

设计三并行结构，L=3，如下

$$\begin{aligned} y\left(3k\right) &= a\cdot y\left(3k-1\right)+b\cdot x\left(3k\right)\\ y\left(3k+1\right) &= a\cdot y\left(3k\right)+b\cdot x\left(3k+1\right)\\ y\left(3k+2\right) &= a\cdot y\left(3k+1\right)+b\cdot x\left(3k+2\right)\end{aligned} \tag{3}$$

电路结构如下，看起来也是某种形式的并行了，但是这种形式有点“蠢”

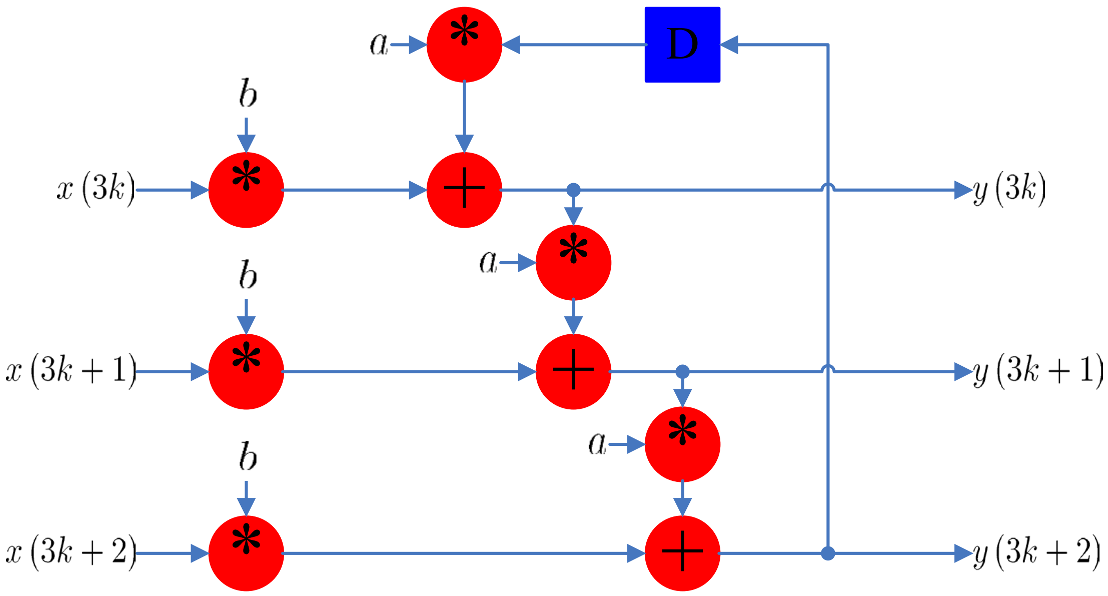


图 1“有点蠢”的 3 并行 IIR 电路

注意体会图 22 的电路，每一个输出点都依赖于其前一个输出点，也就是在输出点之间形成了一个串行的“进位链”，从这一点上说该电路不能算是并行处理。但是，另一方面，三个输入点是同时进行乘法（乘于 b）计算的，算是并行处理。总的说来，图 22 电路具有一定的并行性，但是不是“完全”并行。其实并行的部分只是非递归环路部分，而递归环路是串行的。

图 22 电路的不足在于，计算$y(3k+1)$需要$y(3k)$，计算$y(3k+2)$需要$y(3k+1)$;；如果能去掉这些依赖关系，也就是说计算$y(3k+1)$和$y(3k+2)$只需要$y(3k)$之前的输出，那么就能实现完全并行处理！怎样才能实现这种“超前运算”呢？其实解决方法就是超前计算，同样设计 L=3 并行电路，首先对公式(14)进行 2 步超前迭代，得

$$y(n) = a^3 \cdot y(n-3) + b \cdot x(n) + a \cdot b \cdot x(n-1) + a^2 \cdot b \cdot x(n-2) \tag{4}$$

表示成并行迭代式，如下

$$\begin{aligned} y(3k) &= a^3 \cdot y(3k-3) + b \cdot x(3k) + a \cdot b \cdot x(3k-1) + a^2 \cdot b \cdot x(3k-2) \\ y(3k+1) &= a^3 \cdot y(3k-2) + b \cdot x(3k+1) + a \cdot b \cdot x(3k) + a^2 \cdot b \cdot x(3k-1) \\ y(3k+2) &= a^3 \cdot y(3k-1) + b \cdot x(3k+2) + a \cdot b \cdot x(3k+1) + a^2 \cdot b \cdot x(3k) \end{aligned} \tag{5}$$

可以看到，三个输出点之间不存在依赖关系。对应的电路结构为

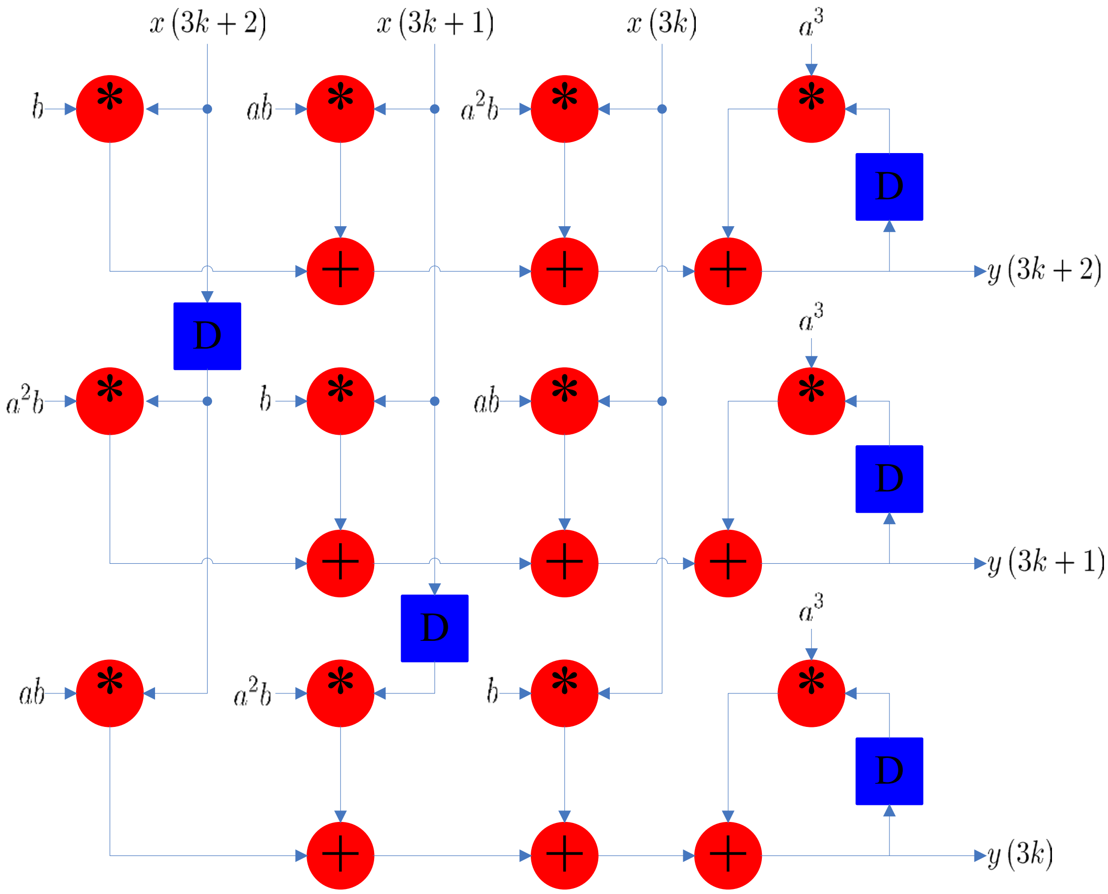


图 2全并行 IIR 结构

从图 23 可以看出，全并行 IIR 结构消耗资源非常多。很多时候将图 22 和图 23 的结构结合起来，得到一个折中的结构是不赖的，如图 24，

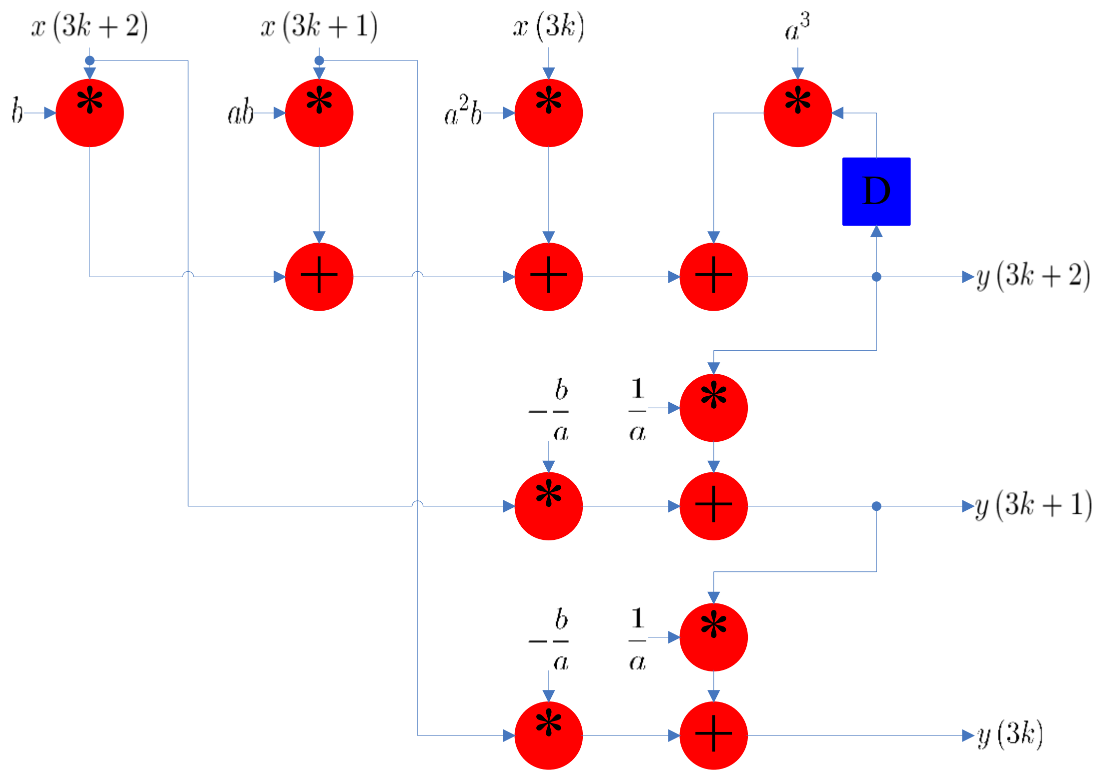


图 3增量块滤波器结构

说实在的，我自己还不是很清楚增量块滤波器结构的好处，一时看不出来图 24 结构到底比图 22 好在哪里？看起来图 22 更节约资源，而且图 24 各个输出点之间也是存在依赖关系，存在一个串行“进位链”。

不论是图 22/23/24，新结构存在环路，必须重新分析其稳定性，课本 10.5 节末尾给出了一个判据，这里我们就不追究是为什么了，直接利用其结论：系统矩阵的特征值的对应于并行系统的极点，如果这些特征值都在单位圆内，那么系统就是稳定的。

将整个构造并行 IIR 电路和稳定性分析的过程编制为程序如下，求解课本例 10.5.3，

代码 3　并行 IIR 求解

```
> restart :
>
> L := 3;
                    L := 3
> A := [1, -5/4, 3/8];
                    A := [1, -5/4, 3/8]
> B := [1, 2, 1];
                    B := [1, 2, 1]
> N := nops(A);
```

$$N := 3$$

```
> M := nops(B);
```

$$M := 3$$

```
> F[0] := f(n) = sum(B[i]·x(n - i + 1), i = 1 .. M);
  Y[0] := y(n) = sum(-A[i]·y(n - i + 1), i = 2 .. N) + f(n);
```

$$F_0 := f(n) = x(n) + 2\ x(n-1) + x(n-2)$$

$$Y_0 := y(n) = \frac{5}{4}\ y(n-1) - \frac{3}{8}\ y(n-2) + f(n)$$

```
>
> T := [seq(0 = 0, i = 1 .. L - 1)] :
   for k from 1 to L - 1 do
    T[k] := subs(n = n - k, Y[0]);
   od:
  Y[1] := simplify(subs(seq(T[i], i = 1 .. L - 1), Y[0]));
```

$$Y_1 := y(n) = \frac{65}{64}\ y(n-3) - \frac{57}{128}\ y(n-4) + \frac{19}{16}\ f(n-2) + \frac{5}{4}\ f(n-1) + f(n)$$

```
>
>
> P := [seq(0 = 0, i = 1 .. L)] :
  for k from 0 to N - 2 do
   P[k+1] := subs(n = L·n + k, Y[1]);
  od:

  for k from N - 1 to L - 1 do
   P[k+1] := subs(n = L·n + k, Y[0]);
   od:

>
> T := [seq(0 = 0, i = 1 .. N - 2)] :
  for k from -L + 1 to N - L - 2 do
   S := subs(n = L·n + k, Y[0]);
   S := y(L·n + k - N + 1) = solve(S, y(L·n + k - N + 1));
   T[k+L] := S;
   od:

>
> for k from 1 to N - 2 do
   P[k] := subs(seq(T[i], i = k .. N - 2), P[k]);
   od:
```

```
>
> for k from 1 to L do
    print(P[k]);
    print("-------------------------------");
  od;
```

$$y(3\ n) = -\frac{15}{32}\ y(3\ n-3) + \frac{19}{16}\ y(3\ n-2) + \frac{5}{4}\ f(3\ n-1) + f(3\ n)$$

"-------------------------------"

$$y(3\ n+1) = \frac{65}{64}\ y(3\ n-2) - \frac{57}{128}\ y(3\ n-3) + \frac{19}{16}\ f(3\ n-1) + \frac{5}{4}\ f(3\ n) + f(3\ n+1)$$

"-------------------------------"

$$y(3\ n+2) = \frac{5}{4}\ y(3\ n+1) - \frac{3}{8}\ y(3\ n) + f(3\ n+2)$$

"-------------------------------"

```
>
> T := 'T':
  for k from 1 to N-1 do
    T[k] := [seq(coeff(op(2, P[k]), y(L·n + i), 1), i = -L..N-2-L)];
  od:
  linalg[eigenvalues](Matrix([seq(T[i], i = 1..N-1)]));
```

$$\frac{27}{64},\ \frac{1}{8}$$

其实说白了，并行处理无非就是按一定规则进行公式代换，大家从例题中可以看明白，依样画葫芦即可。令人不解的是，这样构造的电路到底比前面所说的蠢方法好在哪里？哪位大侠看懂了，请指点指点！谢谢！

第二节、 组合流水线和并行处理的 IIR 滤波器

总的原则是“先并行，后流水”。下面通过一个例子来说明这个原则的实际意义，弄明白这一点，大家就能灵活的构造混合并行处理和流水线的 IIR 滤波器。

以课本上例 10.6.1 为例，这里值得注意的是：公式 10-64 是如何得出的。也许你可以先进行 12 步超前，并$n = 3k + 12$带入，得到 4 级流水和 3 并行的结构，但是，你可能得不到非递

归部分的高效组织方式。也许是课本上内容跳跃幅度过大，有点让人迷惑，不知公式 10-64 到底如何导出，而通过观察得出那是不太可能的；以下的做法能让你明白高效的非递归部分是如何组织的。

原始迭代公式如下，设计 3 并行且 4 级流水 IIR 滤波器，即 L=3，M=4，

$$H\left(e^{j\omega}\right)=\frac{1}{1-a\cdot z^{-1}} \tag{6}$$

时域表示为，

$$y\left(n\right)=a\cdot y\left(n-1\right)+x\left(n\right) \tag{7}$$

按照原则“先并行，后流水”，先对公式(19)进行 3 次超前，并得到如下的 3 并行迭代式，

$$y(3n)=a^3\ y(3n-3)+a^2\ f(3n-2)+a\ f(3n-1)+f(3n)$$

$$y(3n+1)=a\ y(3n)+f(3n+1)$$

$$y(3n+2)=a\ y(3n+1)+f(3n+2)$$

其中，$f(n)=x(n)$。

后两个迭代式是由增量处理技术算出，与构造 4 级流水环路没啥关系，可以不管，只看第一个公式，

$$y(3n)=a^3\ y(3n-3)+a^2\ f(3n-2)+a\ f(3n-1)+f(3n)$$

先进行变量代换，其实就是为了后面推导 4 级流水做准备，

令$\widehat{y}\left(n\right)=y\left(3n\right)$ 和 $g\left(n\right)=f\left(3n\right)+a\cdot f\left(3n-1\right)+a^2\cdot f\left(3n-2\right)$,可将 3 并行迭代式化简为

$$\widehat{y}\left(n\right)=a^3\cdot\widehat{y}\left(n-1\right)+g\left(n\right) \tag{8}$$

构造公式(20)的 4 级流水线结构，4 为 2 的幂次，可以得到高效的分子分解，如下

$$\widehat{H}\left(e^{j\omega}\right)=\frac{\left(1+a^3\cdot z^{-1}\right)\left(1+a^6\cdot z^{-2}\right)}{1-a^{12}\cdot z^{-4}} \tag{9}$$

设计到此结束，接下来只需根据公式(21)和公式(20)画出电路结构即可。由公式(21/20)得，

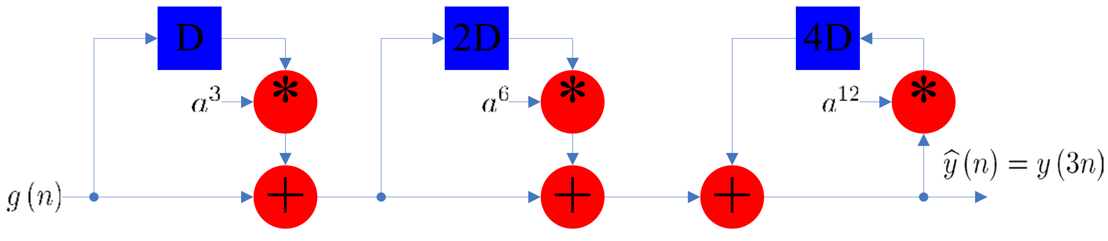


图 4

而$g\left(n\right)$结构为，

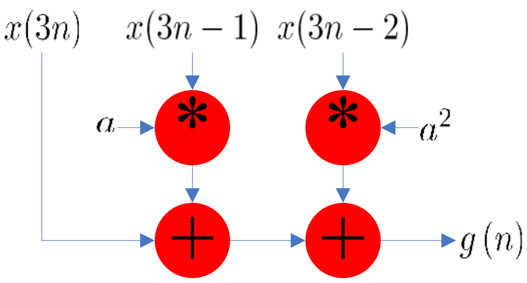
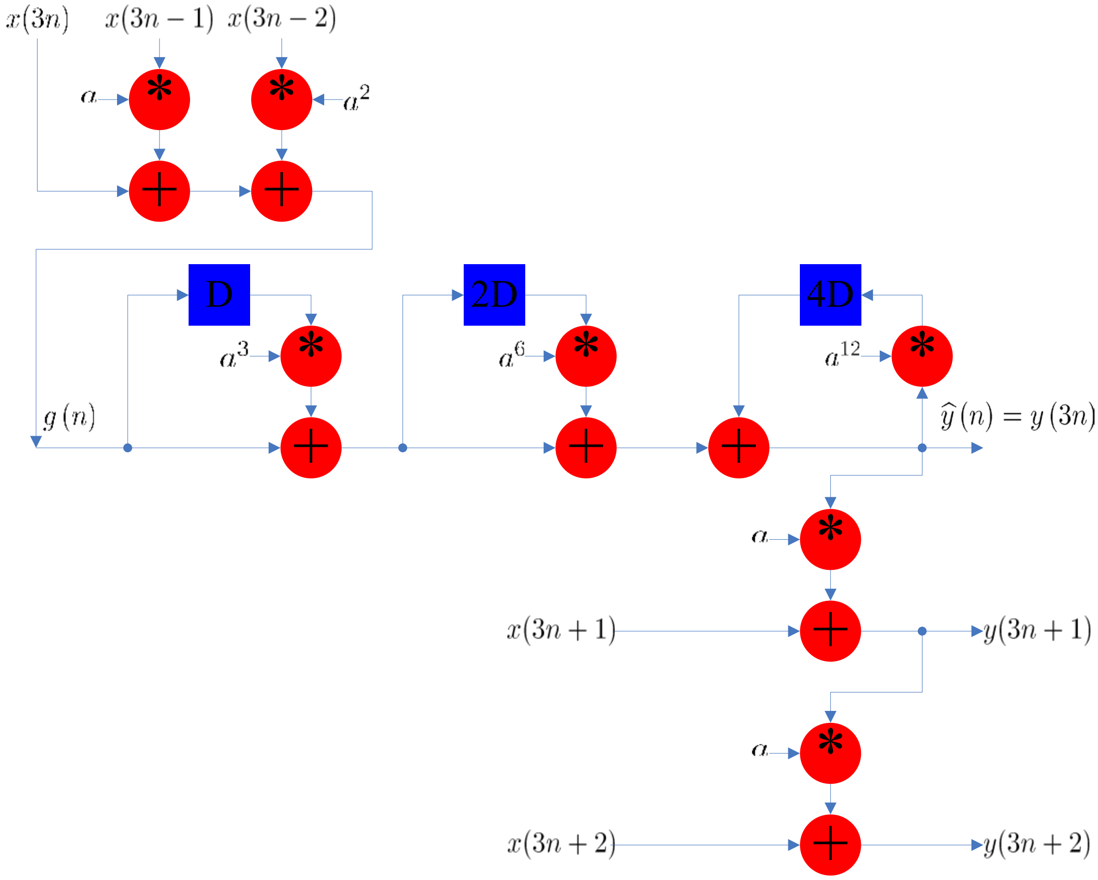


图 5

结合图 26 和图 25，并补全增量计算的部分，有



按照这里给出的推导方式，很自然就得到了非递归部分的高效实现，如果不是这样，可能会耗费更多的资源，大家可以自己试试。

大结：

本章的内容就讲那么多，关于自适应滤波器的内容有兴趣的可以自学，弄懂以上所讲的内容，自适应滤波器部分就很容易理解。课本上的内容基本上来源于参考文献中的文章，从书的出版（1999 年）到现在已过 10 年，也许出现了更新更强的技术，大家不妨关心关心 IEEE/IET 上电路系统和信号处理期刊，了解行业内最新的一些技术动向，以便跟进。

写作匆忙，谬误在所难免，请各位大虾不吝指正，谢谢大家！