Chapter 10: Pipelined and Parallel Recursive and Adaptive Filters

Keshab K. Parhi

Outline

- Introduction
- Pipelining in 1st-Order IIR Digital Filters
- Pipelining in Higher-Order IIR Digital Filters
- Parallel Processing for IIR Filters
- Combined Pipelining and Parallel Processing for IIR Filters

Look-Ahead Computation

First-Order IIR Filter

- Consider a 1st-order linear time-invariant recursion (see Fig. 1)

$$y(n+1) = a \cdot y(n) + b \cdot u(n) \qquad (10.1)$$

- The iteration period of this filter is $\{T_m + T_a\}$, where $\{T_m, T_a\}$ represent word-level multiplication time and addition time
- In *look-ahead transformation*, the linear recursion is first iterated a few times to create additional concurrency.
- By recasting this recursion, we can express y(n+2) as a function of y(n) to obtain the following expression (see Fig. 2(a))

$$y(n+2) = a\left[ay(n) + bu(n)\right] + bu(n+1) \qquad (10.2)$$

- The iteration bound of this recursion is $2\left(T_m + T_a\right)/2$, the same as the original version, because the amount of computation and the number of logical delays inside the recursive loop have both doubled

- Another recursion equivalent to (10.2) is (10.3). Shown on Fig.2(b), its iteration bound is $(T_m + T_a)/2$, a factor of 2 lower than before.

$$y(n+2) = a^2 \cdot y(n) + ab \cdot u(n) + b \cdot u(n+1) \qquad (10.3)$$

- Applying *(M-1)* steps of look-ahead to the iteration of (10.1), we can obtain an equivalent implementation described by (see Fig. 3)

$$y(n+M) = a^M \cdot y(n) + \sum_{i=0}^{M-1} a^i \cdot b \cdot u(n+M-1-i) \qquad (10.4)$$

  - Note: the loop delay is z^{-M} instead of z^{-1}, which means that the loop computation must be completed in *M clock cycles (not 1 clock cycle)*. The iteration bound of this computation is $(T_m + T_a)/M$, which corresponds to a sample rate *M* times higher than that of the original filter
  - The terms $\{ab, a^2 b, \cdots, a^{M-1} b, a^M\}$ in (10.4) can be pre-computed (referred to as *pre-computation terms*). The second term in RHS of (10.4) is the look-ahead computation term (referred to as the *look-ahead complexity*); it is non-recursive and can be easily pipelined

Fig. 1

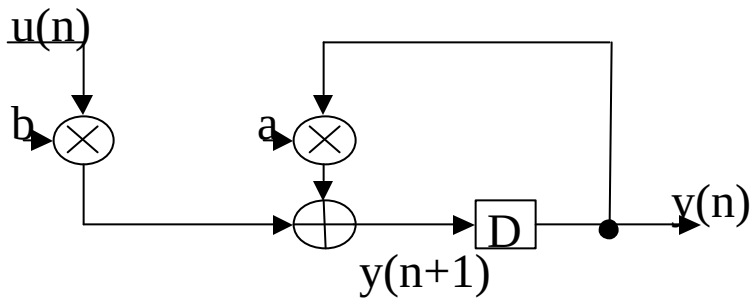


Fig.2.(a)

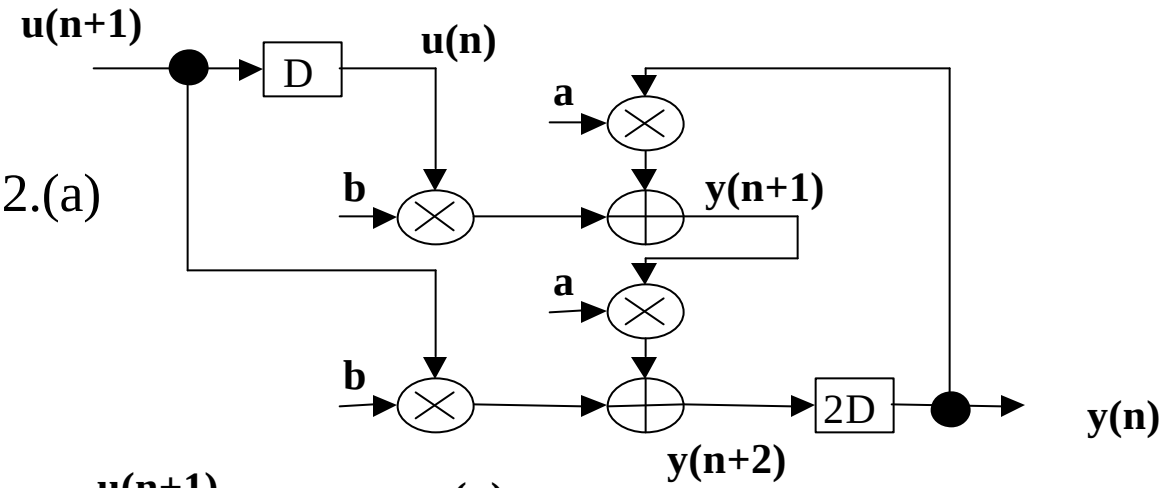
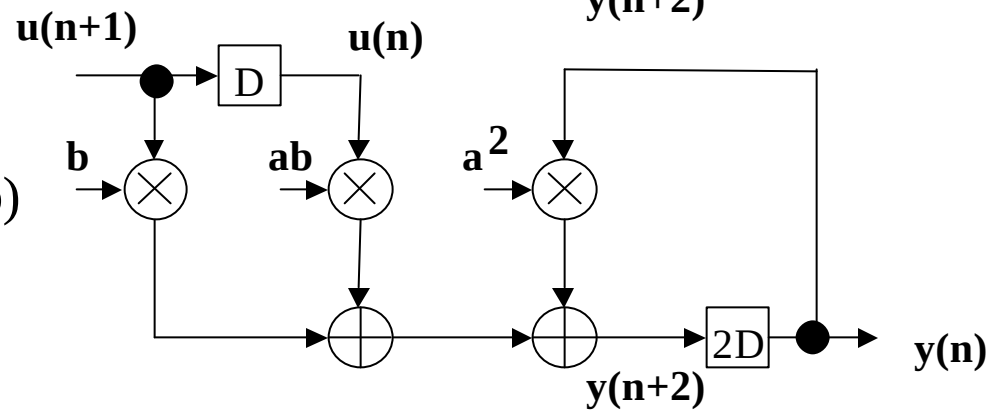


Fig.2.(b)



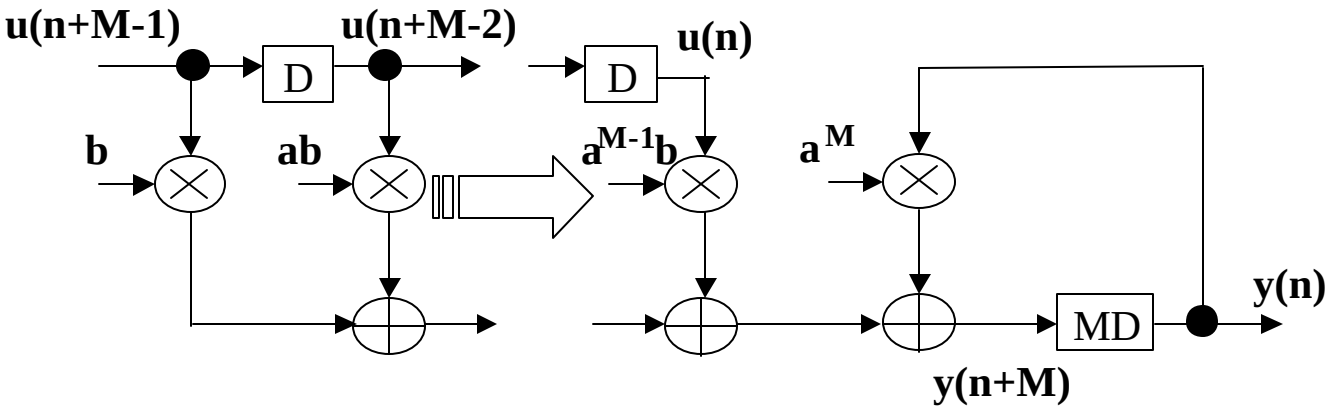


Fig. 3: M-stage Pipelinable 1st-Order IIR Filter

- Look-ahead computation has allowed a single serial computation to be transformed into M independent concurrent computations, and to pipeline the feedback loop to achieve high speed filtering of a single time series while maintaining full hardware utilization.
- Provided the multiplier and the adder can be conveniently pipelined, the iteration bound can be achieved by *retiming* or *cutset transformation* (see Chapter 4)

Pipelining in 1st-Order IIR Digital Filters

- **Example: Consider the 1st-order IIR filter transfer function**

$$H(z) = \frac{1}{1 - a \cdot z^{-1}} \qquad (10.5)$$

  - **The output sample y(n) can be computed using the input sample u(n) and the past output sample as follows:**

$$y(n) = a \cdot y(n-1) + u(n) \qquad (10.6)$$

  - **The sample rate of this recursive filter is limited by the computation time of one multiply-add operation**

- *Look-ahead techniques* add canceling poles and zeros with equal angular spacing at a distance from the origin which is same as that of the original pole. The pipelined filters are always stable provided that the original filter is stable
- The pipelined realizations require a *linear increase in complexity* but decomposition techniques can be used to obtain an implementation with logarithmic increase in hardware with respect to the number of loop pipeline stages

Pipelining in 1st-Order IIR Digital Filters (continued)

1. Look-Ahead Pipelining for 1st-Order IIR Filters

- Look-ahead pipelining adds canceling poles and zeroes to the transfer function such that the coefficients of $\{z^{-1}, \cdots, z^{-(M-1)}\}$ in the denominator of the transfer function are zero. Then, the output sample *y(n)* can be computed using the inputs and the output sample *y(n-M)* such that there are *M* delay elements in the critical loop, which in turn can be used to pipeline the critical loop by *M* stages and the sample rate can be increased by a factor *M*
- Example: Consider the 1st-order filter, $H(z) = 1/\left(1 - a \cdot z^{-1}\right)$, which has a pole at z=a (a≤1). A 3-stage pipelined equivalent stable filter can be derived by adding poles and zeroes at $z = ae^{\pm(j2p/3)}$, and is given by

$$H(z) = \frac{1 + a \cdot z^{-1} + a^2 \cdot z^{-2}}{1 - a^3 \cdot z^{-3}}$$

Pipelining in 1st-Order IIR Digital Filters (continued)

2. Look-Ahead Pipelining with Power-of-2 Decomposition

- With power-of-2 decomposition, an M-stage (for power-of-2 M) pipelined implementation for 1st-order IIR filter can be obtained by $\log_2 M$ sets of transformations
- Example: Consider a 1st-order recursive filter transfer function described by $H(z) = \left(b \cdot z^{-1}\right) / \left(1 - a \cdot z^{-1}\right)$. The equivalent pipelined transfer function can be described using the decomposition technique as follows

$$H(z) = \frac{b \cdot z^{-1} \prod_{i=0}^{\log_2 M - 1} \left(1 + a^{2^i} \cdot z^{-2^i}\right)}{1 - a^M \cdot z^{-M}} \qquad (10.7)$$

  - This pipelined implementation is derived by adding (M-1) poles and zeros at identical locations.
  - The original transfer function has a single pole at $z = a$ (see Fig.4(a)).

– The pipelined transfer function has poles at the following locations (see Fig.4(b) for M=8):

$$\left\{a, ae^{j2p/M}, \cdots, ae^{j(M-1)\cdot(2p)/M}\right\}$$

– The decomposition of the canceling zeros is shown in Fig.4(c). The i-th stage of the decomposed non-recursive portion implements 2^i zeros located at:

$$z = a\exp\left(j(2n+1)p/\left(2^i\right)\right), \quad n = 0,1,\cdots,\left(2^i - 1\right) \qquad (10.8)$$

– The i-th stage of the decomposed non-recursive portion requires a single pipelined multiplication operation independent of the stage number i

– The multiplication complexity of the pipelined implementation is

$$\left(\log_2 M + 2\right)$$

– The finite-precision pipelined filters suffer from inexact pole-zero cancellation, which leads to magnitude and phase error. These errors can be reduced by increasing the wordlength (see p. 323)

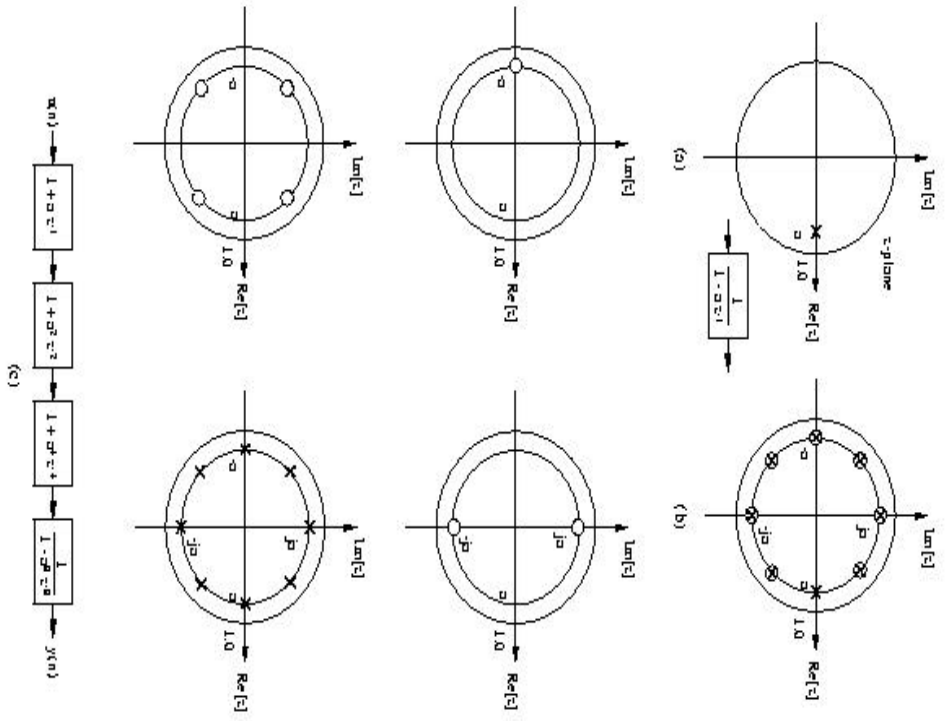


Fig. 4

- Pole representation of a 1^{ST}-order recursive filter
- Pole/zero representation of a 1^{ST}-order IIR with 8 loop stages
- Decomposition based on pipelined IIR for M = 8

Pipelining in 1st-Order IIR Digital Filters (continued)

3. Look-Ahead Pipelining with General Decomposition

- The idea of decomposition can be extended to any arbitrary number of loop pipelining stages M. If $M = M_1 M_2 \cdots M_p$, then the non-recursive stages implement $(M_1 - 1)$, $M_1(M_2 - 1)$, $\cdots$, $M_1 M_2 \cdots M_{p-1}(M_p - 1)$ zeros, respectively, totaling (M-1) zeros
- Example (Example 10.3.3, p.325) Consider the 1st-order IIR

$$H(z) = \frac{1}{1 - a \cdot z^{-1}}$$

  - A 12-stage pipelined decomposed implementation is given by

$$H(z) = \frac{\sum_{i=0}^{11} a^i \cdot z^{-i}}{1 - a^{12} \cdot z^{-12}} = \frac{\left(1 + az^{-1}\right)\left(1 + a^2 z^{-2} + a^4 z^{-4}\right)\left(1 + a^6 z^{-6}\right)}{1 - a^{12} \cdot z^{-12}}$$

- This implementation is based on a $2 \times 3 \times 2$ decomposition (see Fig. 5)

– The first section implements 1 zero at –a, the second section implements 4 zeros at $\left\{ae^{\pm jp/3}, ae^{\pm j2p/3}\right\}$, and the third section implements 6 zeros at $\left\{\pm ja, ae^{\pm jp/6}, ae^{\pm j5p/6}\right\}$

– Another decomposed implementation ($2\times2\times3$ decomposition) is given by

$$H(z)=\frac{\left(1+az^{-1}\right)\left(1+a^2z^{-2}\right)\left(1+a^4z^{-4}+a^8z^{-8}\right)}{1-a^{12}\cdot z^{-12}}$$

• The first section implements 1 zero at –a, the second section implements 2 zeros at $\pm ja$, and the third section implements 8 zeros at $\left\{ae^{\pm jp/6}, ae^{\pm jp/3}, ae^{\pm j2p/3}, ae^{\pm j5p/6}\right\}$

– The third decomposition ($3\times2\times2$ decomposition) is given by

$$H(z)=\frac{\left(1+az^{-1}+a^2z^{-2}\right)\left(1+a^3z^{-3}\right)\left(1+a^6z^{-6}\right)}{1-a^{12}\cdot z^{-12}}$$

• The first section implements 2 zeros at $\left\{ae^{\pm j2p/3}\right\}$, the second section implements 3 zeros at $\left\{-a, ae^{\pm jp/3}\right\}$, and the third section implements 6 zero at $\left\{ae^{\pm jp/6}, \pm ja, ae^{\pm j5p/6}\right\}$

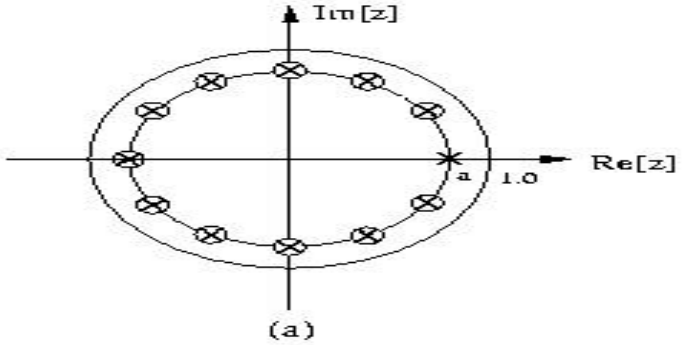


Fig.5(a)

Pole-zero location of a 12-stage pipelined 1^{ST}-order IIR

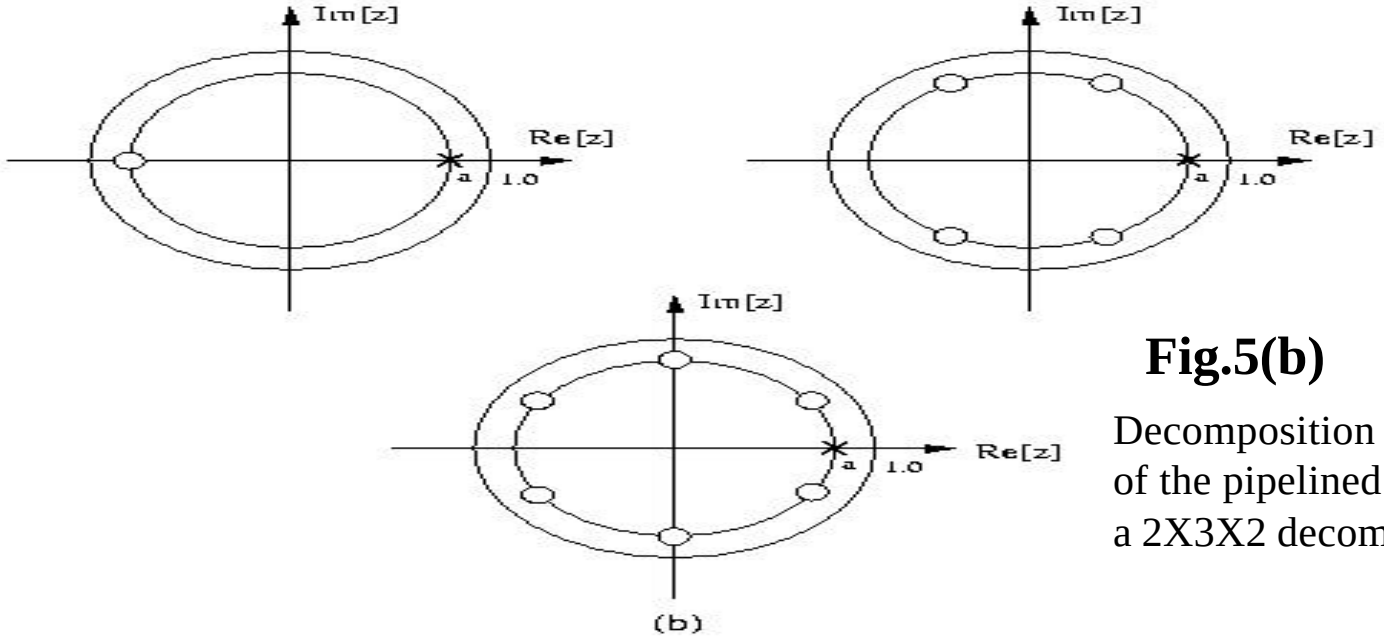


Fig.5(b)

Decomposition of the zeros of the pipelined filter for a 2X3X2 decomposition

Pipelining in Higher-order IIR Digital Filters

- Higher-order IIR digital filters can be pipelined by using clustered look-ahead or scattered look-ahead techniques. (For 1st-order IIR filters, these two look-ahead techniques reduce to the same form)
  - Clustered look-ahead: Pipelined realizations require a linear complexity in the number of loop pipelined stages and are not always guaranteed to be stable
  - Scattered look-ahead: Can be used to derive stable pipelined IIR filters
  - Decomposition technique: Can also be used to obtain area-efficient implementation for higher-order IIR for scattered look-ahead filters
  - Constrained filter design techniques: Achieve pipelining without pole-zero cancellation

- The transfer function of an N-th order direct-form recursive filter is described by

$$H(z) = \frac{\sum_{i=0}^{N} b_i z^{-i}}{1 - \sum_{i=1}^{N} a_i z^{-i}} \qquad (10.9)$$

- Equivalently, the output y(n) can be described in terms of the input sample u(n) and the past input/output samples as follows

$$\begin{aligned} y(n) &= \sum_{i=1}^{N} a_i y(n-i) + \sum_{i=0}^{N} b_i u(n-i) \\ &= \sum_{i=1}^{N} a_i y(n-i) + z(n) \end{aligned} \qquad (10.10)$$

  - The sample rate of this IIR filter realization is limited by the throughput of 1 multiplication and 1 addition, since the critical path contains a single delay element

Pipelining in Higher-order IIR Digital Filters (cont'd)

1. Clustered Look-Ahead Pipelining

- The basic idea of clustered look-ahead:
  - Add canceling poles and zeros to the filter transfer function such that the coefficients of $\{z^{-1}, \cdots, z^{-(M-1)}\}$ in the denominator of the transfer function are 0, and the output samples y(n) can be described in terms of the cluster of N past outputs:

    $\{y(n-M), \cdots, y(n-M-N+1)\}$
  - Hence the critical loop of this implementation contains M delay elements and a single multiplication. Therefore, this loop can be pipelined by M stages, and the sample rate can be increased by a factor M. This is referred to as *M-stage clustered look-ahead pipelining*

- **Example:** (Example 10.4.1, p.327) Consider the all-pole 2nd-order IIR filter with poles at $\{1/2, 3/4\}$. The transfer function of this filter is

$$H(z) = \frac{1}{1 - \frac{5}{4} z^{-1} + \frac{3}{8} z^{-2}} \qquad (10.11)$$

  - A 2-stage pipelined equivalent IIR filter can be obtained by eliminating the z^{-1} term in the denominator (i.e., multiplying both the numerator and denominator by $\left(1 + 5/4\ z^{-1}\right)$. The transformed transfer function is given by:

$$H(z) = \frac{1}{1 - \frac{5}{4} z^{-1} + \frac{3}{8} z^{-2}} \cdot \frac{1 + \frac{5}{4} z^{-1}}{1 + \frac{5}{4} z^{-1}}$$

$$= \frac{1 + \frac{5}{4} z^{-1}}{1 - \frac{19}{16} z^{-2} + \frac{15}{32} z^{-3}} \qquad (10.12)$$

  - From, the transfer function, we can see that the coefficient of z^{-1} in the denominator is zero. Hence, the critical path of this filter contains 2 delay elements and can be pipelined by 2 stages

– Similarly, a 3-stage pipelined realization can be derived by eliminating the terms of $\{z^{-1}, z^{-2}\}$ in the denominator of (10.21), which can be done by multiplying both numerator and denominator by

$$\left(1+5/4\,z^{-1}+19/16\,z^{-2}\right)$$

– The new transfer function is given by:

$$H(z) = \frac{1+\frac{5}{4}z^{-1}+\frac{19}{16}z^{-2}}{1-\frac{65}{64}z^{-3}+\frac{57}{128}z^{-4}} \qquad (10.13)$$

- **Computation complexity:** The numerator (non-recursive portion) of this pipelined filter needs (N+M) multiplications, and the denominator (recursive portion) needs N multiplications. Thus, the total complexity of this pipelined implementation is (N+N+M) multiplications
- **Stability:** The canceling poles and zeros are utilized for pipelining IIR filters. However, when the additional poles lie outside the unit circle, the filter becomes unstable. Note that the filters in (10.12) and (10.13) are unstable.

2. Stable Clustered Look-Ahead Filter Design

- If the desired pipeline delay M does not produce a stable filter, M should be increased until a stable pipelined filter is obtained. To obtain the optimal pipelining level M, numerical search methods are generally used
- Example (Example 10.4.3, p.330) Consider a 5-level (*M=5*) pipelined implementation of the following 2nd-order transfer function

$$H(z)=\frac{1}{1-1.5336z^{-1}+0.6889z^{-2}} \qquad (10.14)$$

  - By the stability analysis, it is shown that (*M=5*) does not meet the stability condition. Thus M is increased to *M=6* to obtain the following stable pipelined filter as

$$H(z)=\frac{1+1.5336z^{-1}+1.6630z^{-2}+1.4939z^{-3}+1.1454z^{-4}+0.7275z^{-5}}{1-1.3265z^{-6}+0.5011z^{-7}} \qquad (10.15)$$

3. Scattered Look-Ahead Pipelining

- Scattered look-ahead pipelining: The denominator of the transfer function in (10.9) is transformed in a way that it contains the N terms $\{z^{-M}, z^{-2M}, \cdots, z^{-NM}\}$. Equivalently, the state y(n) is computed in terms of N past scattered states y(n-M),y(n-2M),…, and y(n-NM)
- In scattered look-ahead, for each poles in the original filter, we introduce (M-1) canceling poles and zeros with equal angular spacing at a distance from the origin the same as that of the original pole.
  - Example: if the original filter has a pole at $z=p$, we add (M-1) poles and zeros at $\{z = p\exp(j2pk/M), \quad k = 1,2,\cdots,M-1\}$ to derive a pipelined realization with M loop pipeline stages
- Assume that the denominator of the transfer function can be factorized as follows:

$$D(z) = \prod_{i=1}^{N}(1 - p_i z^{-1}) \qquad (10.16)$$

(continued)

– Then, the pipelining process using the scattered look-ahead approach can be described by

$$H(z) = \frac{N(z)}{D(z)}$$

$$= \frac{N(z)\prod_{i=1}^{N}\prod_{k=1}^{M-1}\left(1 - p_i e^{j2kp/M} z^{-1}\right)}{\prod_{i=1}^{N}\prod_{k=0}^{M-1}\left(1 - p_i e^{j2kp/M} z^{-1}\right)} = \frac{N'(z)}{D'(z^M)} \quad (10.17)$$

- Example (Example 10.4.5, p.332) Consider the 2nd-order filter with complex conjugate poles at $z = re^{\pm jq}$. The filter transfer function is given by

$$H(z) = \frac{1}{1 - 2r\cos q z^{-1} + r^2 z^{-2}}$$

– We can pipeline this filter by 3 stages by introducing 4 additional poles and zeros at $\left\{z = re^{\pm j(q+2p/3)},\ \ z = re^{\pm j(q-2p/3)}\right\}$ if $q \neq 2p/3$

– (cont'd) The equivalent pipelined filter is then given by

$$H(z)=\frac{1+2r\cos q\cdot z^{-1}+(1+2\cos 2q)r^2\cdot z^{-2}+2r^3\cos q\cdot z^{-3}+r^4z^{-4}}{1-2r^3\cos(3q)\cdot z^{-3}+r^6z^{-6}}$$

– When $q=2p/3$, then only 1 additional pole and zero at $z=r$ is required for 3-stage pipelining since $z=re^{\pm j(q+2p/3)}=re^{\pm jq}$ and $z=re^{\pm j(q-2p/3)}=r$. The equivalent pipelined filter is then given by

$$H(z)=\frac{1-r\cdot z^{-1}}{(1+r\cdot z^{-1}+r^2\cdot z^{-2})(1-r\cdot z^{-1})}=\frac{1-r\cdot z^{-1}}{1-r^3\cdot z^{-3}}$$

- Example (Example 10.4.6, p.332) Consider the 2nd-order filter with real poles at $\{z=r_1, z=r_2\}$. The transfer function is given by

$$H(z)=\frac{1}{1-(r_1+r_2)\cdot z^{-1}+r_1r_2\cdot z^{-2}}$$

– (cont'd) A 3-stage pipelined realization is derived by adding poles(and zeros) at $\left\{z = r_1 e^{\pm j2p/3}, \quad z = r_2 e^{\pm j2p/3}\right\}$. The pipelined realization is given by

$$H(z) = \frac{1 + (r_1 + r_2)z^{-1} + (r_1^{\,2} + r_1 r_2 + r_2^{\,2})z^{-2} + r_1 r_2 (r_1 + r_2)z^{-3} + r_1^{\,2} r_2^{\,2} z^{-4}}{1 - (r_1^{\,3} + r_2^{\,3}) \cdot z^{-3} + r_1^{\,3} r_2^{\,3} z^{-6}}$$

– The pole-zero locations of a 3-stage pipelined 2nd-order filter with poles at z=1/2 and z=3/4 are shown in Fig. 6

- **CONCLUSIONS:**
  – If the original filter is stable, then the scattered look-ahead approach leads to stable pipelined filters, because the distance of the additional poles from the original is the same as that of the original filter
  – Multiplication Complexity: (NM+1) for the non-recursive portion in (10.17), and N for the recursive portion. Total pipelined filter multiplication complexity is (NM+N+1)

– (cont'd) The multiplication complexity is linear with respect to M. and is much greater than that of clustered look-ahead
– Also the latch complexity is square in M, because each multiplier is pipelined by M stages

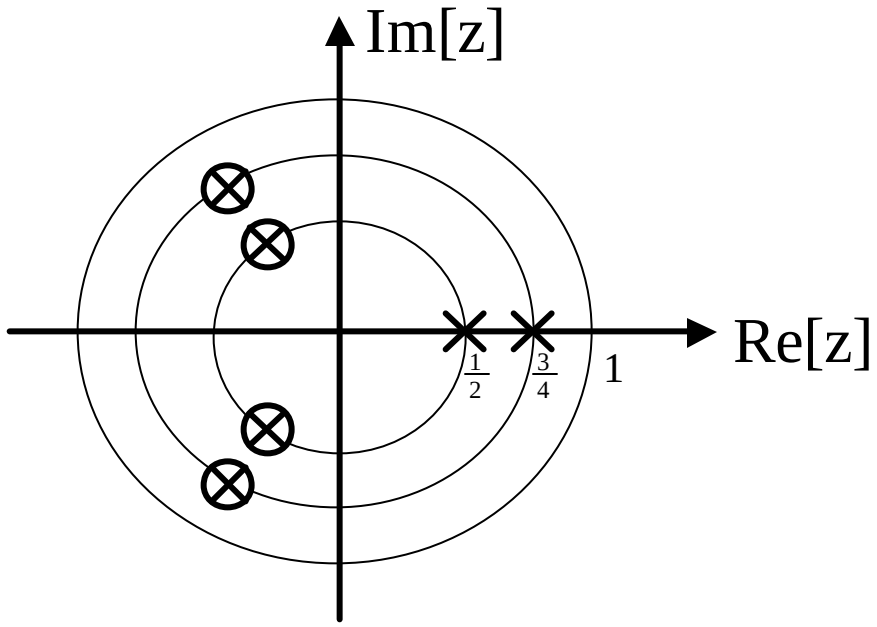


Fig. 6: Pole-zero representation of a 3-stage pipelined equivalent stable filter derived using scattered look-ahead approach

4. Scattered Look-Ahead Pipelining with Power-of-2 Decomposition

- This kind of decomposition technique will lead to a logarithmic increase in multiplication complexity (hardware) with respect to the level of pipelining
- Let the transfer function of a recursive digital filter be

$$H(z) = \frac{\sum_{i=0}^{N} b_i z^{-i}}{1 - \sum_{i=1}^{N} a_i \cdot z^{-i}} = \frac{N(z)}{D(z)} \qquad (10.18)$$

  - A 2-stage pipelined implementation can be obtained by multiplying by $\left(1 - \sum_{i=1}^{N} (-1)^i a_i \cdot z^{-i}\right)$ in the numerator and denominator. The equivalent 2-stage pipelined implementation is described by

$$H(z) = \frac{\left(\sum_{i=0}^{N} b_i z^{-i}\right)\left(1 - \sum_{i=1}^{N} (-1)^i a_i \cdot z^{-i}\right)}{\left(1 - \sum_{i=1}^{N} a_i \cdot z^{-i}\right)\left(1 - \sum_{i=1}^{N} (-1)^i a_i \cdot z^{-i}\right)} = \frac{N'(z)}{D'(z)} \qquad (10.19)$$

- (cont'd) Similarly, subsequent transformations can be applied to obtain 4, 8, and 16 stage pipelined implementations, respectively
- Thus, to obtain an M-stage pipelined implementation *(for power-of-2 M)*, $\log_2 M$ sets of such transformations need to be applied.
- By applying $(\log_2 M - 1)$ sets of such transformations, an equivalent transfer function (with M pipelining stages inside the recursive loop) can be derived, which requires a complexity of $(2N + N\log_2 M + 1)$ multiplications, a logarithmic complexity with respect to M.
- Note: the number of delays (or latches) is linear: the total number of delays (or latches) is approximately $NM(\log_2 M + 1)$, about NM delays in non-recursive portion, and $NM\log_2 M$ delays for pipelining each of the $N\log_2 M$ multipliers by M stages
- In the decomposed realization, the 1st stage implements an N-th order non-recursive section, and the subsequent stages respectively implement 2N, 4N,…, NM/2-order non-recursive sections

- Example (Example 10.4.7, p.334) Consider a 2nd-order recursive filter described by

$$H(z)=\frac{Y(z)}{U(z)}=\frac{b_0+b_1z^{-1}+b_2z^{-2}}{1-2r\cos q\cdot z^{-1}+r^2z^{-2}}$$

  - The poles of the system are located at $\left\{z=re^{\pm jq}\right\}$ (see Fig. 7(a)).
  - The pipelined filter requires $\left(2\log_2 M+5\right)$ multiplications and is described by $\left\{where\quad q\neq 2p/M\right\}$

$$H(z)=\frac{\sum_{i=0}^{2}b_iz^{-i}}{1-2r^M\cos Mq\cdot z^{-M}+r^{2M}z^{-2M}}\times$$
$$\prod_{i=0}^{\log_2 M-1}\left(1+2r^{2^i}\cos 2^iq\cdot z^{-2^i}+r^{2^{i+1}}z^{-2^{i+1}}\right)$$

  - The 2M poles of the transformed transfer function (shown in Fig. 7(b)) are located at

$$z=re^{\pm j\left(q+i\left(2p/M\right)\right)},\quad i=0,1,2\cdots,\left(M-1\right)$$

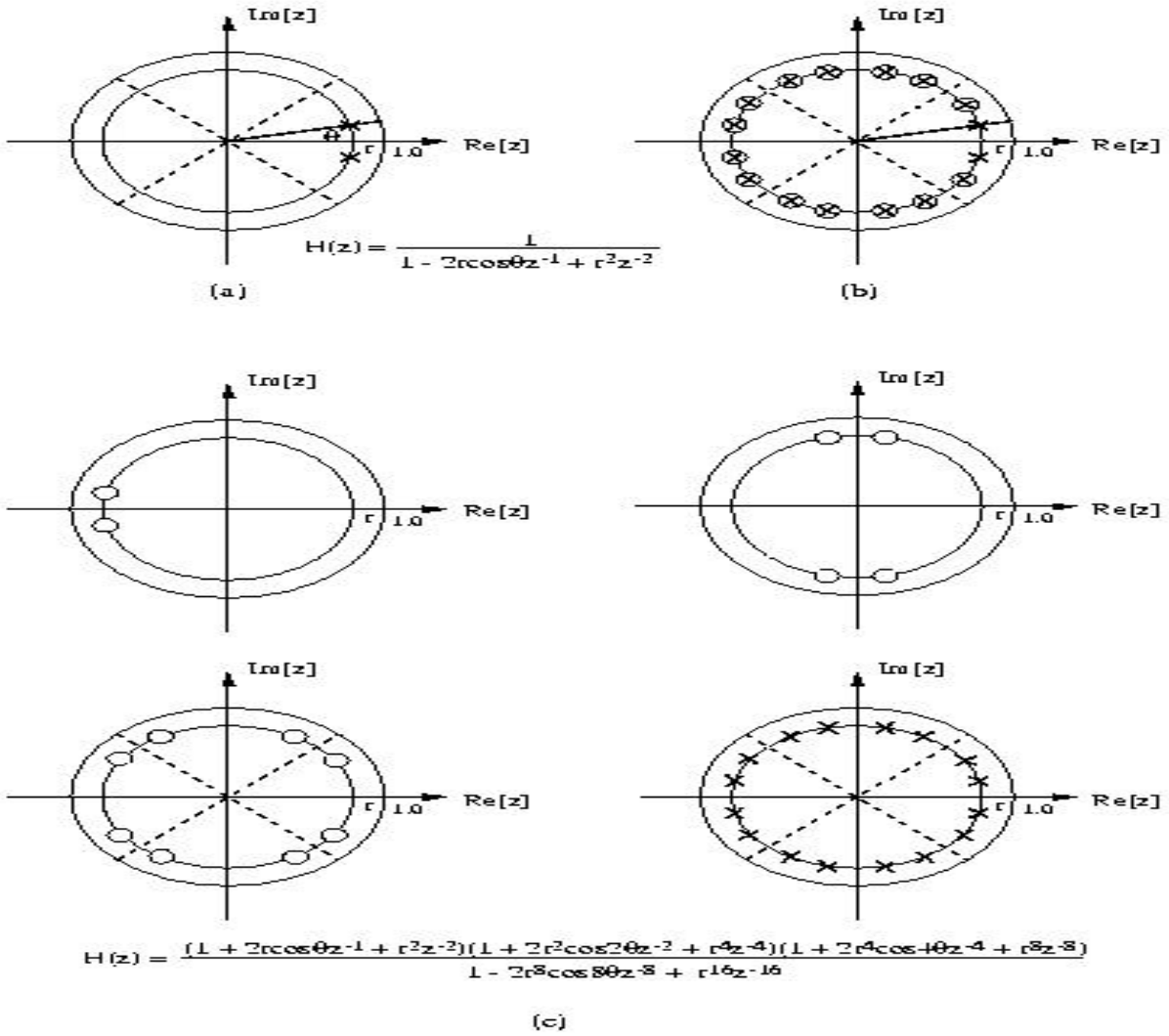


Fig. 7: (also see Fig.10.11, p.336) Pole-zero representation for Example 10.4.7, the decompositions of the zeros is shown in (c)

Parallel Processing in IIR Filters

- Parallel processing can also be used in design of IIR filters
- First discuss parallel processing for a simple 1st-order IIR filter, then we discuss higher order filters
- Example: (Example 10.5.1, p.339) Consider the transfer function of a 1st-order IIR filter given by

$$H(z) = \frac{z^{-1}}{1 - az^{-1}}$$

  - where $|a| \leq 1$ for stability. This filter has only 1 pole located at $z = a$. The corresponding input-output can be written as $y(n+1)=ay(n)+u(n)$
  - Consider the design of a 4-parallel architecture (L=4) for the foregoing filter. Note that in the parallel system, each delay element is referred to as a block delay, where the clock period of the block system is 4 times the sample period. Therefore, the loop update equation should update $y(n+4)$ by using inputs and $y(n)$.

– By iterating the recursion (or by applying look-ahead technique), we get

$$y(n+4) = a^4 y(n) + a^3 u(n) + a^2 u(n+1) + a u(n+2) + u(n+3) \qquad (10.20)$$

– Substituting $n = 4k$

$$y(4k+4) = a^4 y(4k) + a^3 u(4k) + a^2 u(4k+1) + a u(4k+2) + u(4k+3)$$

– The corresponding architecture is shown in Fig.8.

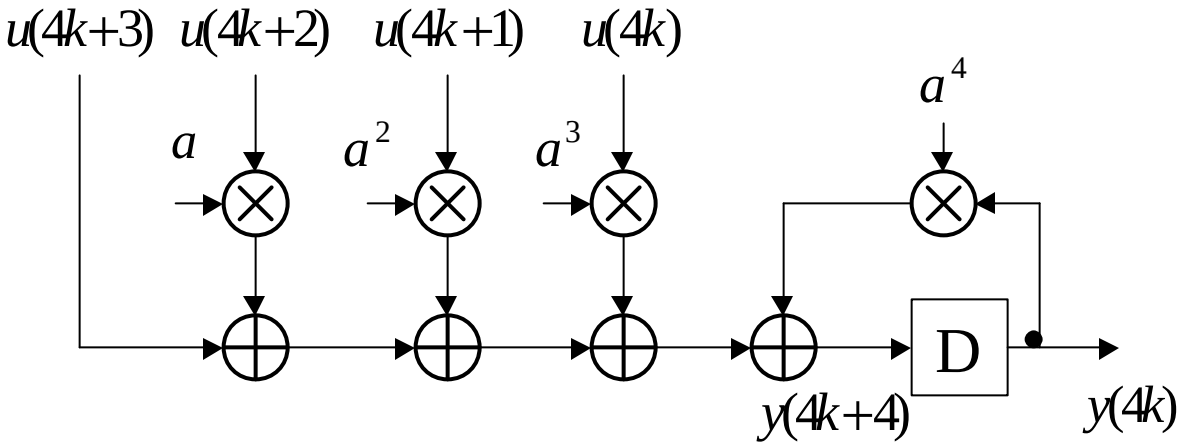


Fig. 8: (also see Fig.10.14, p. 340)

- The pole of the original filter is at $z = a$, whereas the pole for the parallel system is at $z = a^4$, which is much closer to the origin since

$$\left\{ \left| a^4 \right| \le |a|, \quad \text{since} \quad |a| \le 1 \right\}$$

- An important implication of this pole movement is the improved robustness of the system to the round-off noise
- A straightforward block processing structure for L=4 obtained by substituting n=4k+4, 4k+5, 4k+6 and 4k+7 in (10.20) is shown in Fig. 9.
- Hardware complexity of this architecture: L^2 multiply-add operations (Because L multiply-add operations are required for each output and there are L outputs in total)
- The square increase in hardware complexity can be reduced by exploiting the concurrency in the computation (the decomposition property in the scattered look-ahead mode can not be exploited in the block processing mode because one hardware delay element represents L sample delays)

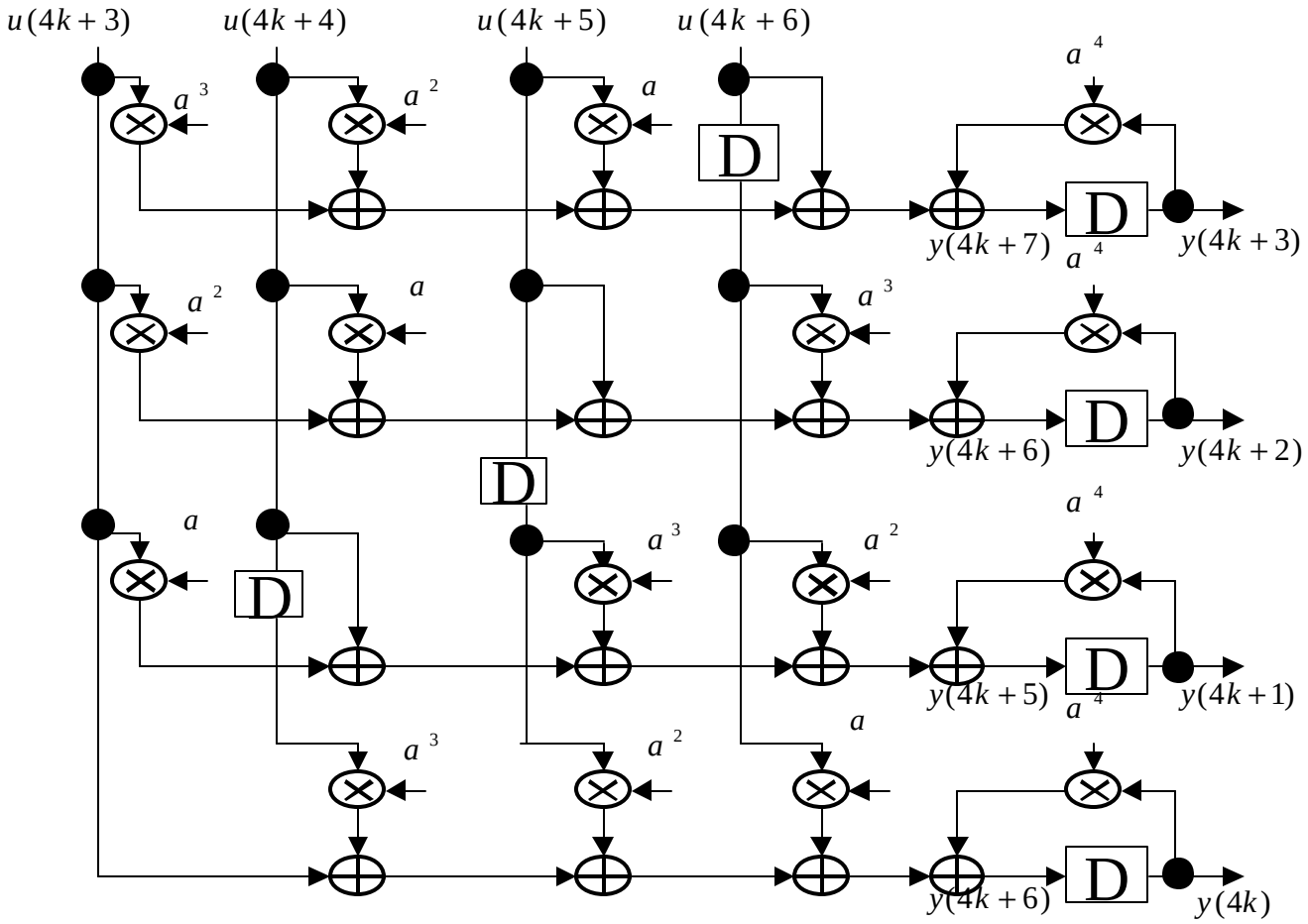


Fig. 9: (also Fig.10.15, p.341) A 4-parallel 1st-order recursive filter

- Trick:
  - Instead of computing *y(4k+1)*, *y(4k+2)* & *y(4k+3)* independently, we can use *y(4k)* to compute *y(4k+1)*, use *y(4k+1)* to compute *y(4k+2)*, and use *y(4k+2)* to compute *y(4k+3)*, at the expense of an increase in the system latency, which leads to a significant reduction in hardware complexity.
  - This method is referred as *incremental block processing*, and *y(4k+1)*, *y(4k+2) and y(4k+3)* are computed *incrementally*.
- Example (Example 10.5.2, p.341) Consider the same 1st-order filter in last example. To derive its 4-parallel filter structure with the minimum hardware complexity instead of simply repeating the hardware 4 times as in Fig.15, the incremental computation technique can be used to reduce hardware complexity
  - First, design the circuit for computing *y(4k)* (same as Fig.14)
  - Then, derive *y(4k+1)* from *y(4k)*, *y(4k+2)* from *y(4k+1)*, *y(4k+3)* from *y(4k+2)* by using

$$\begin{cases} y(4k+1) = ay(4k) + u(4k) \\ y(4k+2) = ay(4k+1) + u(4k+1) \\ y(4k+3) = ay(4k+2) + u(4k+2) \end{cases}$$

– The complete architecture is shown in Fig.10
– The hardware complexity has reduced from L^2 to *(2L-1)* at the expense of an increase in the computation time for *y(4k+1)*, *y(4k+2)* and *y(4k+3)*

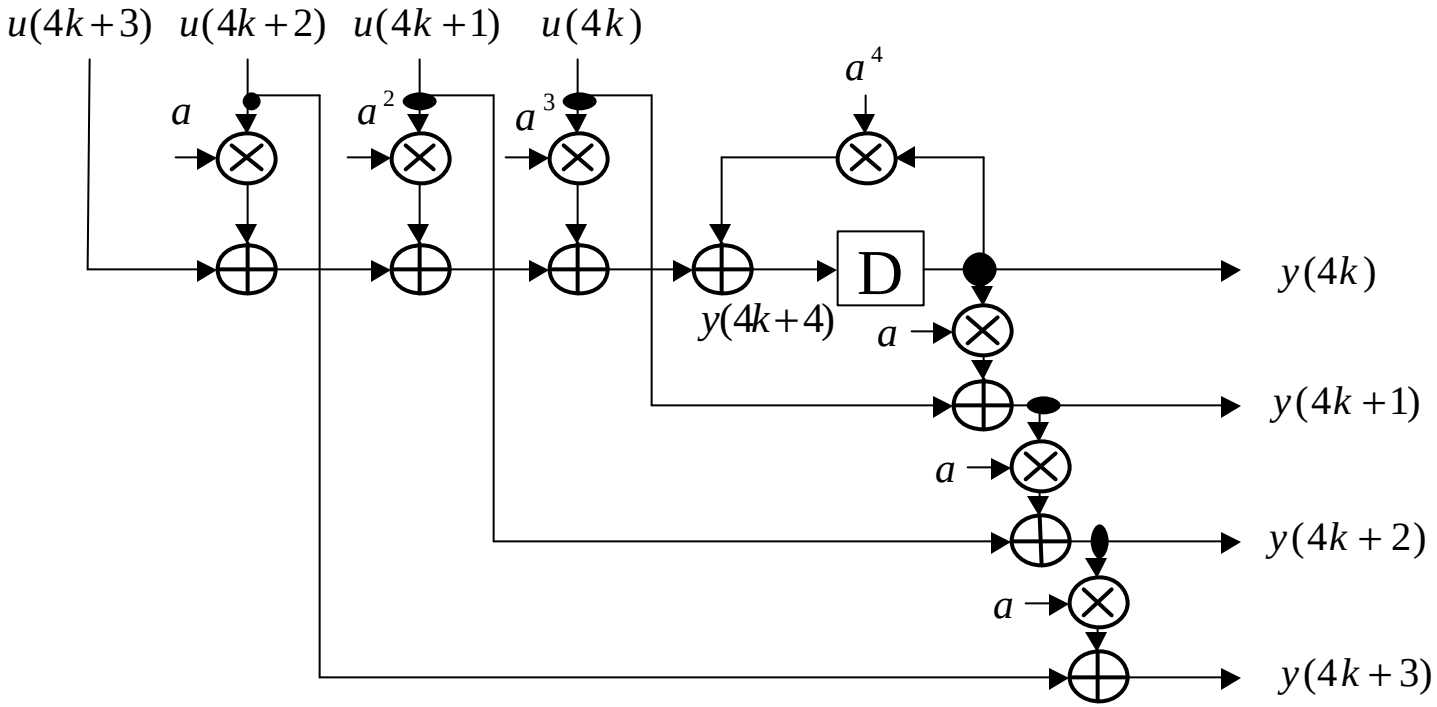


Fig.10: (also see Fig.10.16, p.342) Incremental block filter structure with L=4

- Example (Example 10.5.3, p.342) Consider a 2nd-order IIR filter described by the transfer function (10.21). Its pole-zero locations are shown in Fig.11. Derive a 3-parallel IIR filter where in every clock cycle 3 inputs are processed and 3 outputs are generated

$$H(z) = \frac{(1+z^{-1})^2}{1-\frac{5}{4}z^{-1}+\frac{3}{8}z^{-2}} \qquad (10.21)$$

  - Since the filter order is 2, 2 outputs need to be updated independently and the 3rd output can be computed incrementally outside the feedback loop using the 2 updated outputs. Assume that y(3k) and y(3k+1) are computed using loop update operations and y(3k+2) is computed incrementally. From the transfer function, we have:

$$\begin{aligned} y(n) &= \tfrac{5}{4}y(n-1) - \tfrac{3}{8}y(n-2) + f(n); \\ f(n) &= u(n) + 2u(n-1) + u(n-2) \end{aligned} \qquad (10.22)$$

  - The loop update process for the 3-parallel system is shown in Fig.12 where *y(3k+3)* and *y(3k+4)* are computed using *y(3k)* and *y(3k+1)*

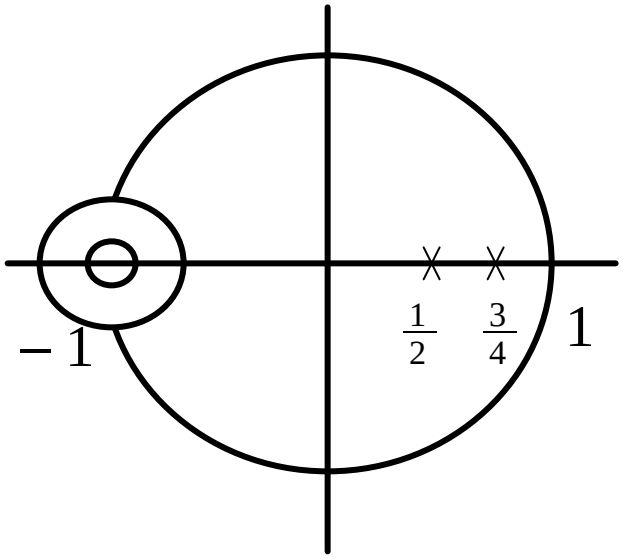


Fig.11: Pole-zero plots for the transfer function

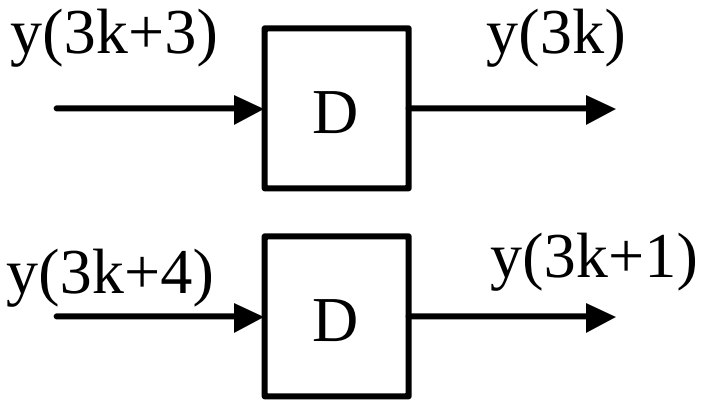


Fig.12: Loop update for block size=3

– The computation of *y(3k+3)* using *y(3k) & y(3k+1)* can be carried out if *y(n+3)* can be computed using *y(n) & y(n+1).* Similarly *y(3k+4)* can be computed using *y(3k) & y(3k+1)* if *y(n+4)* can be expressed in terms of *y(n) & y(n+1)* (see Fig.13). These state update operations correspond to clustered look-ahead operation for M=2 and 3 cases. The 2-stage and 3-stage clustered look-ahead equations are derived as:

$$
\begin{aligned}
y(n) &= \tfrac{5}{4}\, y(n-1) - \tfrac{3}{8}\, y(n-2) + f(n) \\
&= \tfrac{5}{4}\left[\tfrac{5}{4}\, y(n-2) - \tfrac{3}{8}\, y(n-3) + f(n-1)\right] - \tfrac{3}{8}\, y(n-2) \\
&\quad + f(n) \\
&= \tfrac{19}{16}\left[\tfrac{5}{4}\, y(n-3) - \tfrac{3}{8}\, y(n-4) + f(n-2)\right] - \tfrac{15}{32}\, y(n-3) \\
&\quad + \tfrac{5}{4}\, f(n-1) + f(n)
\end{aligned}
\qquad (10.23)
$$

– Substituting *n=3k+3 & n=3k+4* into (10.23), we have the following 2 loop update equations:

$$\begin{cases} y(3k+3) = \frac{19}{16}\, y(3k+1) - \frac{15}{32}\, y(3k) + \frac{5}{4}\, f(3k+2) \\ \qquad\qquad + f(3k+2) \\ y(3k+4) = \frac{65}{64}\, y(3k+1) - \frac{57}{128}\, y(3k) + \frac{19}{16}\, f(3k+2) \\ \qquad\qquad + \frac{5}{4}\, f(3k+3) + f(3k+4) \end{cases} \tag{10.24}$$

– The output y(3k+2) can be obtained incrementally as follows:

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

– The block structure is shown in Fig. 14

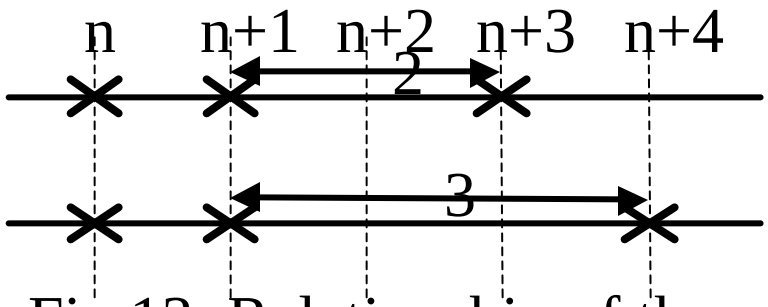


Fig.13: Relationship of the recursive outputs

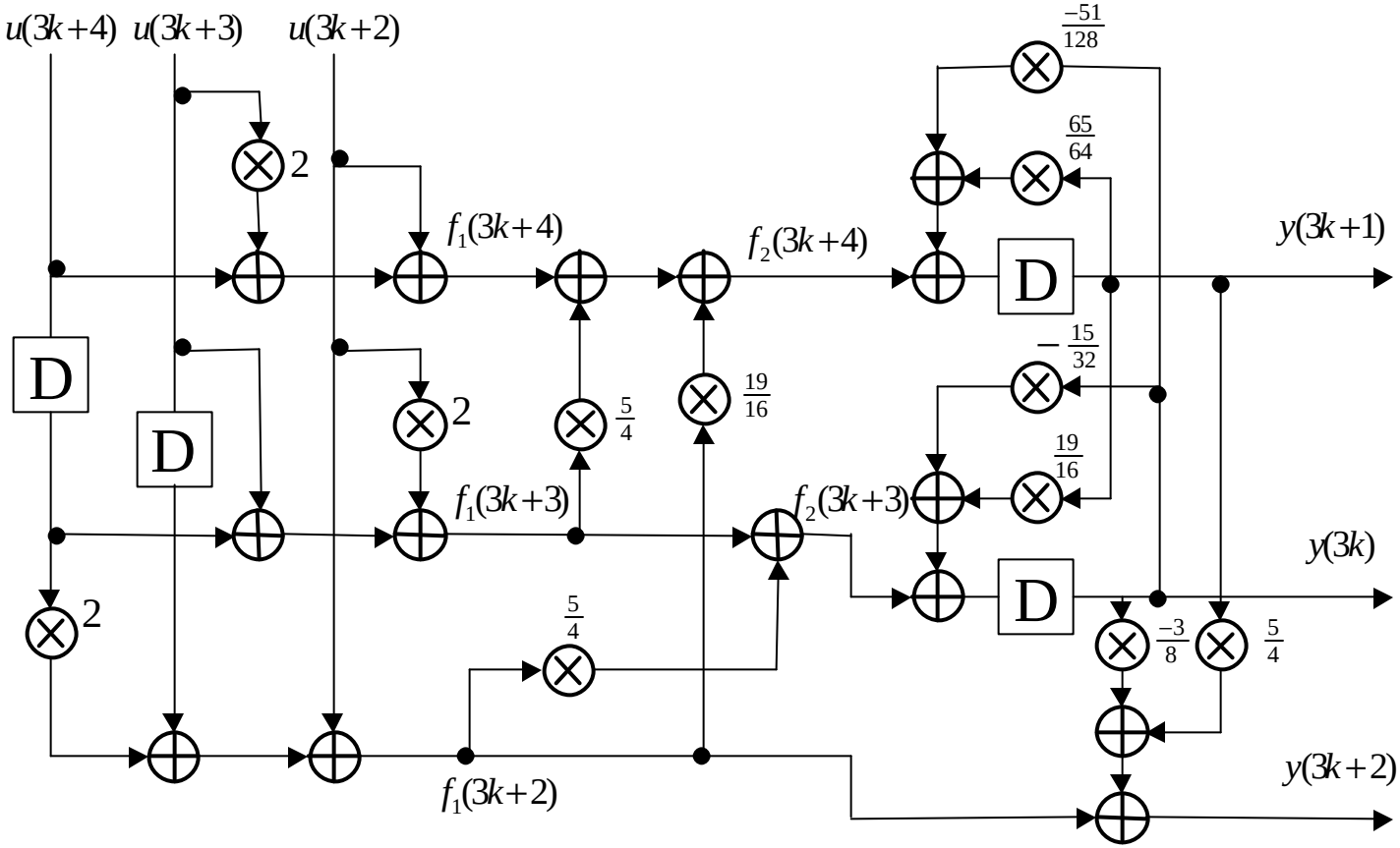


Fig. 14: Block structure of the 2nd-order IIR filter (L=3)
(also see Fig.10.20, p.344)

- Comments
  - The original sequential system has 2 poles at {1/2, 3/4}. Now consider the pole locations of the new parallel system. Rewrite the 2 state update equations in matrix form: ***Y(3k+3)=AY(3k)+F, i.e.***

$$\begin{bmatrix} y(3k+3) \\ y(3k+4) \end{bmatrix} = \begin{bmatrix} \frac{-15}{32} & \frac{19}{16} \\ \frac{-57}{128} & \frac{65}{64} \end{bmatrix} \cdot \begin{bmatrix} y(3k) \\ y(3k+1) \end{bmatrix} + \begin{bmatrix} f_1 \\ f_2 \end{bmatrix} \qquad (10.25)$$

  - The eigenvalues of system matrix A are $\left(\frac{1}{2}\right)^3$, $\left(\frac{3}{4}\right)^3$, which are the poles of the new parallel system. Thus, the parallel system is more stable. **Note**: the parallel system has the same number of poles as the original system
  - For a 2nd-order IIR filter (N=2), there are total *3L+[(L-2)+(L-1)]+4+2(L-2)=7L-3* multiplications, **(**the numerator part — *3L*; the overhead of loop update — *[(L-2)+(L-1)];* the loop multiplications — *4*; the incremental computation — *2(L-2)***)**. The multiplication complexity is linear function of block size L. This multiplication complexity can be further reduced by using fast parallel filter structures and substructure sharing for the incrementally-computed outputs

Combined Pipelining and Parallel Processing For IIR Filters

- Pipelining and parallel processing can also be combined for IIR filters to achieve a speedup in sample rate by a factor L×M, where L denotes the levels of block processing and M denotes stages of pipelining, or to achieve power reduction at the same speed
- Example (Example 10.6.1, p.345) Consider the 1st-order IIR with transfer function (10.26). Derive the filter structure with 4-level pipelining and 3-level block processing (i.e., M=4, L=3)

$$H(z) = \frac{1}{1 - a \cdot z^{-1}} \qquad (10.26)$$

  - Because the filter order is 1, only 1 loop update operation is required.The other 3 outputs can be computed incrementally.

– Since pipelining level *M=4*, the loop must contain 4 delay elements (shown in Fig.15). Since the block size *L=3*, each delay element represents a block delay (corresponds to 3 sample delays). Therefore, *y(3k+12)* needs to be expressed in terms of *y(3k)* and inputs (see Fig. 15).

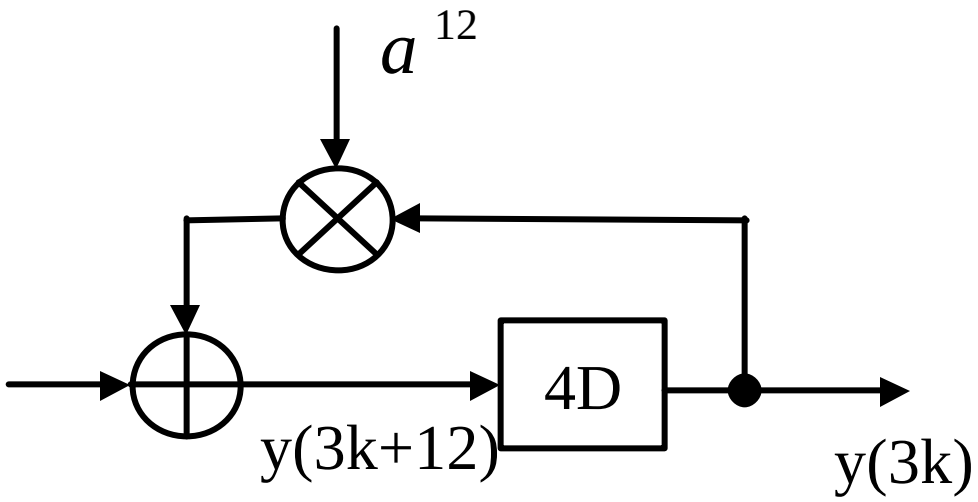


Fig.15: Loop update for the pipelined block system
(also see Fig.10.21, p. 346)

$$y(n) = ay(n-1) + u(n)$$
$$= \cdots\cdots$$
$$= a^{12} y(n-12) + a^{11} u(n-11) + \cdots\cdots + u(n)$$

– Substituting *n=3k+12*, we get:

$$y(3k+12) = a^{12} y(3k) + a^{11} u(3k+1) + \cdots\cdots + u(3k+12)$$

where
$$= a^{12} y(3k) + a^{6} f_2(3k+6) + a^{3} f_1(3k+9) + f_1(3k+12)$$

$$\begin{cases} f_1(3k+12) = a^2 u(3k+10) + au(3k+11) + u(3k+12) \\ f_2(3k+12) = a^3 f_1(3k+9) + f_1(3k+12) \end{cases}$$

– Finally, we have:

$$\begin{cases} y(3k+12) = a^{12} y(3k) + a^{6} f_2(3k+6) + a^{3} f_1(3k+9) + f_1(3k+12) \\ y(3k+1) = ay(3k) + u(3k+1) \\ y(3k+2) = ay(3k+1) + u(3k+2) \end{cases} \quad (10.27)$$

– The parallel-pipelined filter structure is shown in Fig. 16

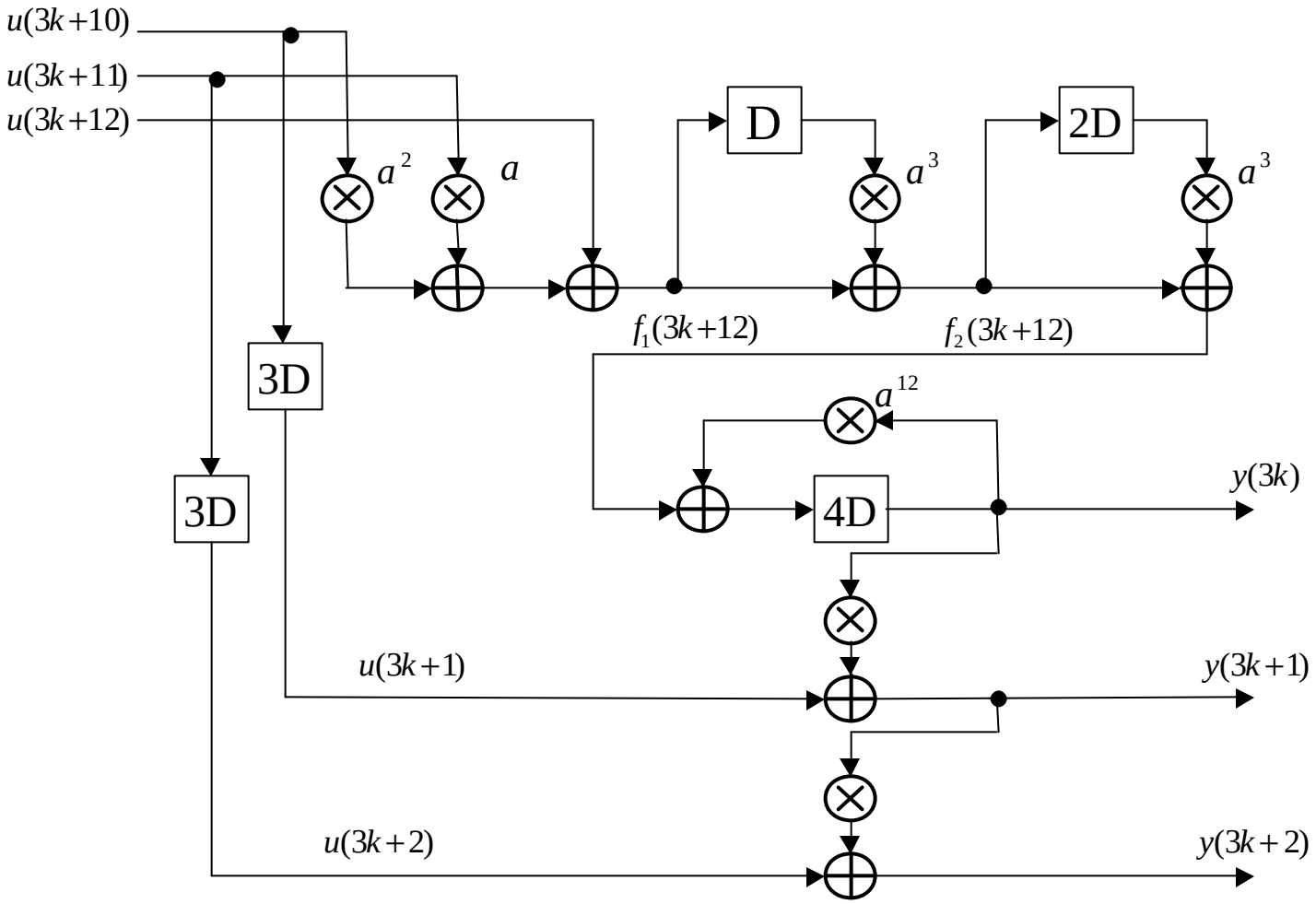


Fig. 16: The filter structure of the pipelined block system with L=3 & M=4 (also see Fig.10.22, p.347)

- Comments
  - The parallel-pipelined filter has 4 poles: $\left(a^3, -a^3, ja^3, -ja^3\right)$. Since the pipelining level is 4 and the filter order is 1, there are total 4 poles in the new system, which are separated by the same angular distance. Since the block size is 3, the distance of the poles from the origin is $\left|a^3\right|$.
  - Note: The decomposition method is used here in the pipelining phase.
  - The multiplication complexity (assuming the pipelining level M to be power of 2) can be calculated as (10.28), which is linear with respect to L, and logarithmic with respect to M:

$$(L-1)+\log_2 M + 1 + (L-1) = 2L - 1 + \log_2 M \qquad (10.28)$$

- Example (Example 10.6.2, p. 347) Consider the 2nd-order filter in Example 10.5.3 again, design a pipelined-block system for *L=3 and M=2*

$$\begin{aligned} y(n) &= \tfrac{5}{4} y(n-1) - \tfrac{3}{8} y(n-2) + f(n); \\ f(n) &= u(n) + 2u(n-1) + u(n-2) \end{aligned} \qquad (10.29)$$

- A method similar to clustered look-ahead can be used to update y(3k+6) and y(3k+7) using y(3k) and y(3k+1). Then by index substitution, the final system of equations can be derived.
- Suppose the system update matrix is A. Since the poles of the original system are $\left(\frac{1}{2}, \frac{3}{4}\right)$, the eigenvalues of A can be verified to be $\left(\frac{1}{2}\right)^6, \left(\frac{3}{4}\right)^6$
- The poles of the new parallel-pipelined second-order filter are the square roots of eigenvalues of A, i.e., $\left(\frac{1}{2}\right)^3, -\left(\frac{1}{2}\right)^3, \left(\frac{3}{4}\right)^3, -\left(\frac{3}{4}\right)^3$

• Comments: In general, the systematic approach below can be used to compute the pole location of the new parallel pipelined system:

- 1. Write the loop update equations using LM-level look-ahead, where M and L denote the level of pipelining and parallel processing, respectively.
- 2. Write the state space representation of the parallel pipelined filter, where state matrix A has dimension N×N and N is the filter order
- 3. Compute the eigenvalues λ_i of matrix A,
- 4. The NM poles of the new parallel-pipelined system correspond to the M-th roots of the eigenvalues of A, i.e.,

$$\left(\lambda_i\right)^{\frac{1}{M}} \quad 1 \le i \le N$$