Chapter 9
Pipeline and Parallel Recursive and Adaptive Filters

- Yin-Tsung Hwang

Introduction (1)

- Any required digital filter spectrum can be realized using FIR or IIR filters
- IIR filters require lower tap order but have potential stability problem
- adaptive filters are used in time varying systems
- coefficients of the adaptive digital filters are adapted at each iteration
- recursive and adaptive filters cannot be easily pipelined or processed in parallel due to the feedback loops

Introduction (2)

- Parallelizing and pipelining recursive digital filters
  - look ahead computation
  - incremental block processing
  - relaxed look ahead transform

Pipeline Interleaving in Digital Filters

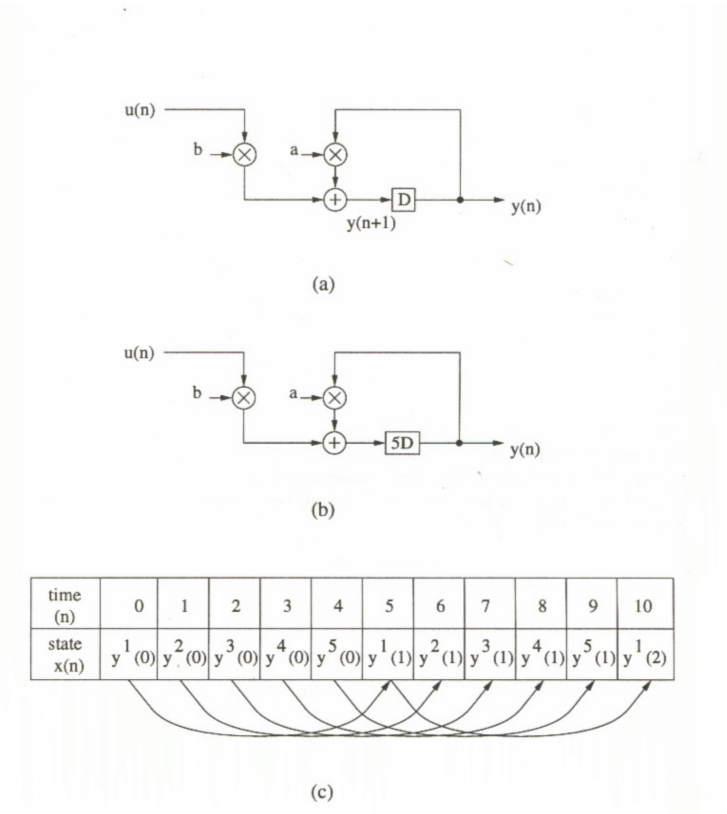
- Three forms of pipeline interleaving
  - inefficient single/multi-channel interleaving
  - efficient single-channel interleaving
  - efficient multi-channel interleaving
- inefficient single/multi-channel interleaving
  - consider a 1st order LTI recursion
  - $y(n+1)=a*y(n)+b*u(n)$

Pipeline Interleaving in Digital Filters

(a) a simple 1st order recursion

(b) inserting M-1 delay

(c) 5-way interleaving



time (n)	0	1	2	3	4	5	6	7	8	9	10
state x(n)	$y^1(0)$	$y^2(0)$	$y^3(0)$	$y^4(0)$	$y^5(0)$	$y^1(1)$	$y^2(1)$	$y^3(1)$	$y^4(1)$	$y^5(1)$	$y^1(2)$

Single/Multi-Channel Interleaving (1)

- Iteration period in (a) is T_m+T_a
- *M-stage* pipelined version in (b), where *M*-1 additional latches are added
  - clock period can be reduced by *M* times
  - sample period must be increased M times for correct operation
  - *M* times rescaling
  - effective initiation interval and computing latency remains unchanged
  - overhead of *M*-1 additional latches

Single/Multi-Channel Interleaving (2)

- in (c), M independent time series are operated on the same hardware
  - may correspond to each cascade stage of a large filter
  - or independent channels requiring identical filtering operations
- also known as *M*-slow circuit
- potential drawbacks
  - sample rate is *M* times slower than the clock rate
  - inefficient processor utilization if *M* independent computing series are not available

Efficient Single Channel Interleaving (1)

- 1-step Look ahead transformation
  - y(n+1)=a*y(n)+b*u(n)
  - y(n+2)=a[a*y(n)+b*u(n)]+b*u(n+1)

    $\Rightarrow$ iteration bound = $2(T_m+T_a)/2$
  - y(n+2)=a^2*y(n)+a*b*u(n)+b*u(n+1)

    $\Rightarrow$ iteration bound = $(T_m+T_a)/2$
- M-step look ahead transformation

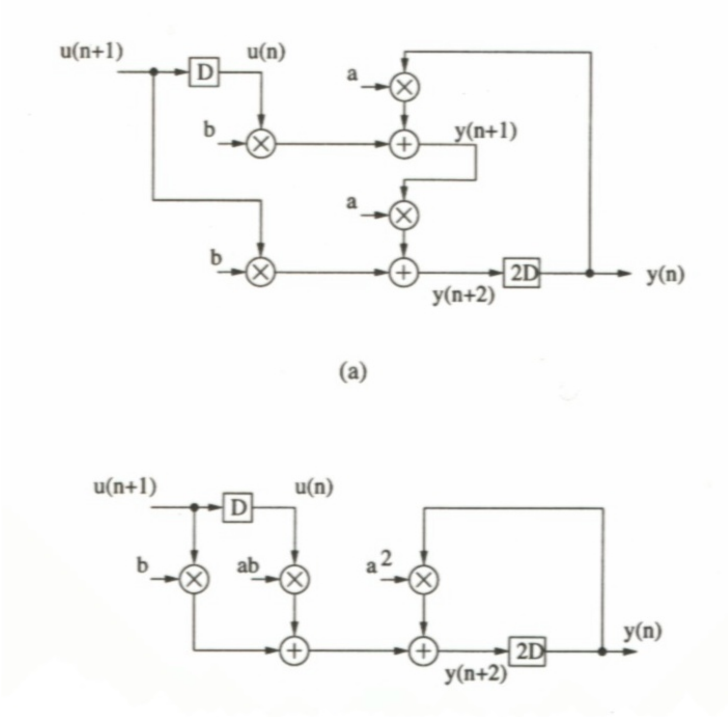
$$y(n+M)=a^M y(n)+\sum_{i=0}^{M-1} a^i \cdot b \cdot u(n+M-1-i)$$

  - coefficients can be pre-computed

Efficient Single Channel Interleaving (2)

(a) equivalent realization of unfolding 1st order IIR filter 1 time

(b) equivalent 1st order recursion after look ahead transform



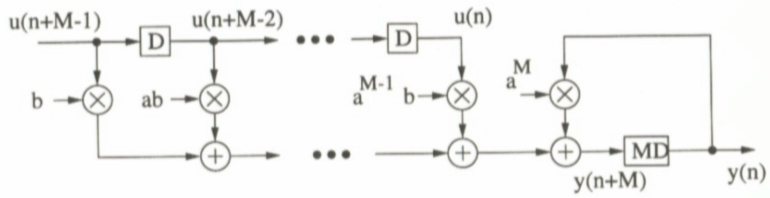
Efficient Single Channel Interleaving (3)

- M-step look ahead transformation
  - allows a single serial computation transformed into M independent concurrent computations
  - loop delay is M $\Rightarrow$ computation is completed in M cycles
  - iteration bound is (T_m+T_a) / M
  - hardware complexity and computing latency are linearly increased
  - M times higher sampling rate than than the original graph
  - initial conditions
  - $y(-i)=a^{-i} * y(0)$, i = 1,2, …., M-1

Efficient Single Channel Interleaving (4)

(a) equivalent 1st order recursion after (M-1) step look ahead transform

(b) a partial scheduling for M = 5



(a)

time (n)	0	1	2	3	4	5	6	7	8	9	10
state x(n)	y(-4)	y(-3)	y(-2)	y(-1)	y(0)	y(1)	y(2)	y(3)	y(4)	y(5)	y(6)

Efficient Multi-Channel Interleaving

- Look ahead transform + pipeline interleaving
  - assume P independent time series (channels)
  - loop is M-stage pipelined
  - $\Rightarrow$ the recursion must be iterated (Q -1) times, where M = P*Q
- Example for M = 6, P = 2

$$y^i(n+3) = a^3 y^i(n) + a^2 b u^i(n) + abu^i(n+1) + bu^i(n+2), \quad i = 1,2$$

- a partial schedule for a 2-channel with 6 pipeline stages obtained using 2-step LA

time (n)	0	1	2	3	4	5	6	7	8	9	10	11	12
state x(n)	$y^1(-2)$	$y^2(-2)$	$y^1(-1)$	$y^2(-1)$	$y^1(0)$	$y^2(0)$	$y^1(1)$	$y^2(1)$	$y^1(2)$	$y^2(2)$	$y^1(3)$	$y^2(3)$	$y^1(4)$

Pipelining in 1st Order IIR Filters

- Look ahead transform introduces canceling poles and zeros
  - if the introduced poles and zeros do not cancel each other exactly due to e.g. finite precision HW limitation
  - the poles outside unit circle will cause stability problem
  - pipelined realization for a 1st order IIR filter is always stable provided the original is stable
- decomposition technique
  - to implement the non-recursive portion due to look ahead transform
  - to achieve logarithmic HW complexity increase w.r.t. the No. of pipeline stages

Look Ahead for 1st order IIR Filters

- Consider a first order filter
  - contains a pole at $z = a$, $a \le 1$ $\quad H(z) = \dfrac{1}{1 - a \cdot z^{-1}}$
- a 3-stage pipelined version filter
  - contains added poles and zeros at $z = ae^{\pm(j2\pi/3)}$

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

  - because the introduced poles have magnitude always less than 1
  - $\Rightarrow$ the filter is always stable even the added poles are not exactly canceled out

Look Ahead Pipelining with Power of 2 Decomposition (1)

- Non-recursive part of an M (power of 2) -stage pipeline is decomposed into $\log_2 M$ sets of transformation
- consider a 1st order recursive filter $H(z) = \dfrac{b \cdot z^{-1}}{1 - a \cdot z^{-1}}$
  - has a single pole at $z = a$
- after look ahead

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

  - adding (M - 1) poles and zeros at identical locations
  - has poles at locations

$$a,\ ae^{j2\pi/M},\ ae^{j2(2\pi)/M}, \cdots,\ ae^{j(M-1)(2\pi)/M}$$

Power of 2 Decomposition (2)

- The decomposition of the canceling zeros
  - the i-th stage implements 2^i zeros located at

$$z = ae^{j(2n+1)\pi/2^i},\ n = 0,1,\cdots,(2^i - 1)$$

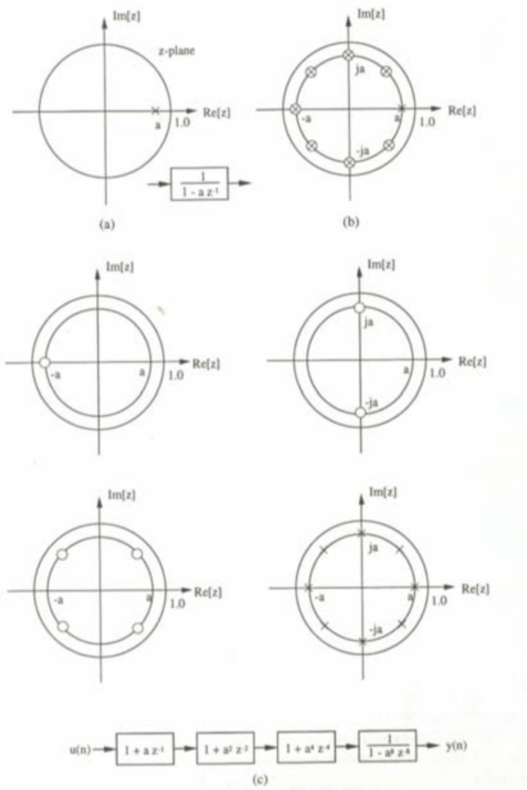
  - requires a single pipelined multiplication operation independent of the stage number i.
  - A total of ($\log_2$M+2) multiplication is needed

Power of 2 Decomposition (3)

(a) pole representation of a 1st order recursive filter

(b) pole zero representation of a 1st order LTI recursive system with 8 loop pipelining stages

(c) decomposition based pipeline implementation for M =8



Power of 2 Decomposition (4)

- Time domain explanation

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

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

- for M=8 example ,where

$$y(n+8) = a^8 \cdot y(n) + \sum_{i=0}^{7} a^i \cdot b \cdot u(n+7-i)$$

$$= a^8 \cdot y(n) + \sum_{i=0}^{7} a^i \cdot f_0(n+7-i)$$

$$= a^8 \cdot y(n) + \sum_{i=0}^{3} a^{2i} \cdot f_1(n+7-2i)$$

$$= a^8 \cdot y(n) + \sum_{i=0}^{1} a^{4i} \cdot f_2(n+7-4i)$$

$$f_0(n) = b \cdot u(n)$$

$$f_1(n) = a \cdot f_0(n-1) + f_0(n)$$

$$f_2(n) = a^2 \cdot f_1(n-2) + f_1(n)$$

Power of 2 Decomposition (5)

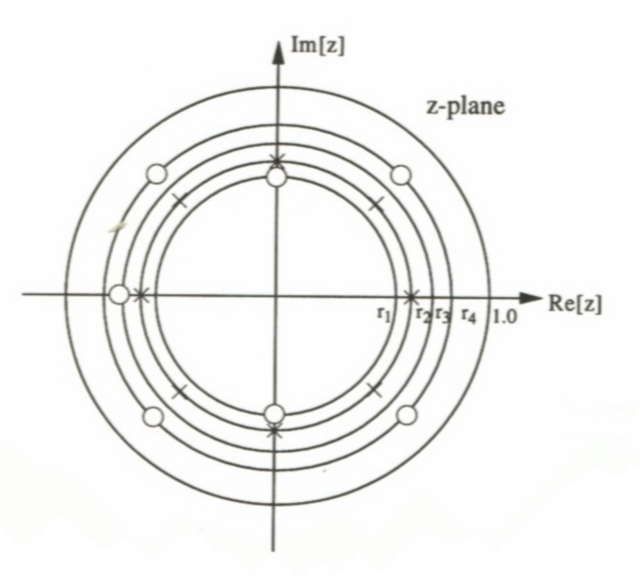
- In finite precision implementation, the poles are located at

$$p = (a^M + \Delta)^{1/M} \approx a(1 + \frac{\Delta}{M \cdot a^M})$$

- where Δ corresponds to finite precision error in representing a^M
  - the deviation becomes severe when |a| <<1
  - not a problem
- inexact pole-zero cancellation
  - leads to phase and magnitude error
  - can be reduced by increasing the word length

Power of 2 Decomposition (6)

- Inexact pole zero cancellation due to finite precision implementation



General Decomposition (1)

- If $M = M_1 M_2 \cdots M_p$
  - $\Rightarrow$ the non-recursive stages implement (M_1-1), $M_1(M_2-1)$, $\cdots$, $M_1 M_2 \cdots (M_p-1)$ zeros
- a 12-stage pipeline decomposition example
  - $12 = 2 \times 3 \times 2$

$$H(z) = \frac{\sum_{i=0}^{11} a^i \cdot z^{-i}}{1 - a^{12} z^{-12}}$$

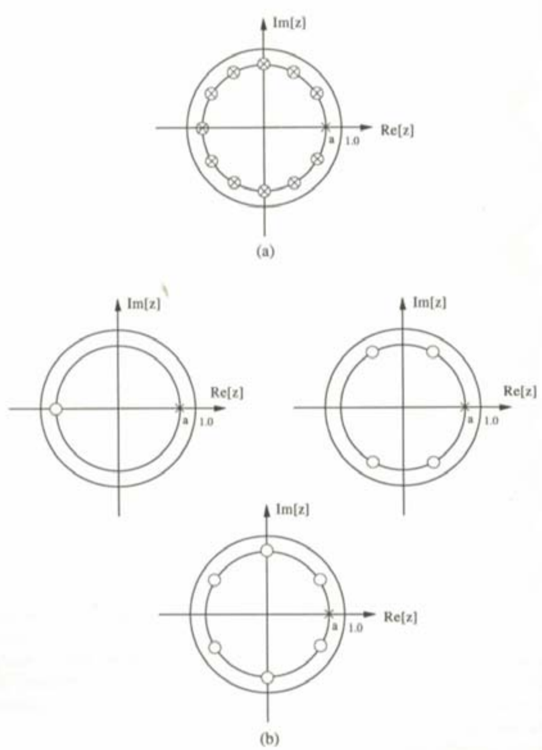
$$= \frac{(1 + az^{-1})(1 + a^2 z^{-2} + a^4 z^{-4})(1 + a^6 z^{-6})}{1 - a^{12} z^{-12}}$$

  - 1st stage implements the zero at $-a$
  - 2nd stage implements 4 zeros at $a \cdot e^{\pm j\pi/3},\ a \cdot e^{\pm j2\pi/3}$
  - 3rd stage implements 6 zeros at $\pm ja, a \cdot e^{\pm j\pi/6}, a \cdot e^{\pm j5\pi/6}$

General Decomposition (2)

(a) pole zero location of a 12-stage pipelined 1st order recursive filter

(b) decomposition of the zeros of the pipelined filter for a $2 \times 3 \times 2$ decomposition



General Decomposition (3)

- An alternative 2 × 2 × 3 pipeline decomposition
  - $$H(z)=\frac{(1+az^{-1})(1+a^2z^{-2})(1+a^4z^{-4}+a^8z^{-8})}{1-a^{12}z^{-12}}$$
  - 1st stage: $-a$ 2nd stage: $\pm ja$
  - 3rd stage: $a\cdot e^{\pm j\pi/6},\ a\cdot e^{\pm j\pi/3},\ a\cdot e^{\pm j2\pi/3},\ a\cdot e^{\pm j5\pi/6}$
- An alternative 3 × 2 × 2 pipeline decomposition
  - $$H(z)=\frac{(1+az^{-1}+a^2z^{-2})(1+a^3z^{-3})(1+a^6z^{-6})}{1-a^{12}z^{-12}}$$
  - 1st stage: $a\cdot e^{\pm j2\pi/3}$ 2nd stage: $-a, a\cdot e^{\pm j\pi/3}$
  - 3rd stage: $a\cdot e^{\pm j\pi/6}, \pm ja, a\cdot e^{\pm j5\pi/6}$

Pipelining in High Order IIR Filters

- *Clustered* look ahead transforms
  - linear hardware complexity w.r.t. pipeline stages
  - but not always guaranteed to be stable
- *scattered* look ahead transforms
  - guaranteed to be stable
  - higher hardware complexity, i.e. N*M
- both transforms reduce to the same form in the 1st order IIR filter case

Clustered Look-Ahead Pipelining (1)

- Consider an N-th order direct form IIR filter

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

$$y(n)=\sum_{i=1}^{N} a_i y(n-i)+\sum_{i=0}^{N} b_i u(n-i)$$
$$=\sum_{i=1}^{N} a_i y(n-i)+z(n)$$

  - the sample rate is limited by the throughput of 1 multiplication and 1 addition

Clustered Look-Ahead Pipelining (2)

- adding canceling poles and zeros such that the coefficients of $z^{-1}, \cdots, z^{-(M-1)}$ in the denominator becomes zero
- look ahead the *cluster* of past N outputs by M-1 steps
  - y(n-1), y(n-2), ··· , y(n-N)
  - ⇒ y(n-M), y(n-M-1), ··· , y(n-M-N+1)
  - the critical loop contains M delay elements and a single multiplication
  - sample rate can be increased by a factor M
  - *M-stage clustered look-ahead pipelining*

Clustered Look-Ahead Pipelining (3)

- Consider an all-pole 2nd order IIR
  - with poles at z = 1/2 and z = 3/4
  - original transfer function $H(z) = \dfrac{1}{1 - \frac{5}{4}z^{-1} + \frac{3}{8}z^{-2}}$
  - for 2-stage pipelining, the coefficient of z^{-1} must be nullified
  - multiply the numerator and denominator by $(1+5/4z^{-1})$
  - I.e. adding canceling pole and zero at z = -5/4
  - the transformed transfer function

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

Clustered Look-Ahead Pipelining (4)

- 3-stage pipelining
  - multiply the numerator and denominator by
  - $(1+5/4z^{-1}+19/16z^{-2})$
  - the transformed transfer function $H(z) = \dfrac{1 + \frac{5}{4}z^{-1} + \frac{19}{16}z^{-2}}{1 - \frac{65}{64}z^{-3} + \frac{57}{128}z^{-4}}$
- general form of multiplying factors
  - $\sum_{i=0}^{M-1} r_i \cdot z^{-i}$, where

$$r_{-i} = 0, \text{ for } i = 1, 2, \cdots, N-1$$
$$r_0 = 1$$
$$r_i = \sum_{k=1}^{N} a_k r_{i-k}, \quad \text{for } i > 0$$

Clustered Look-Ahead Pipelining (5)

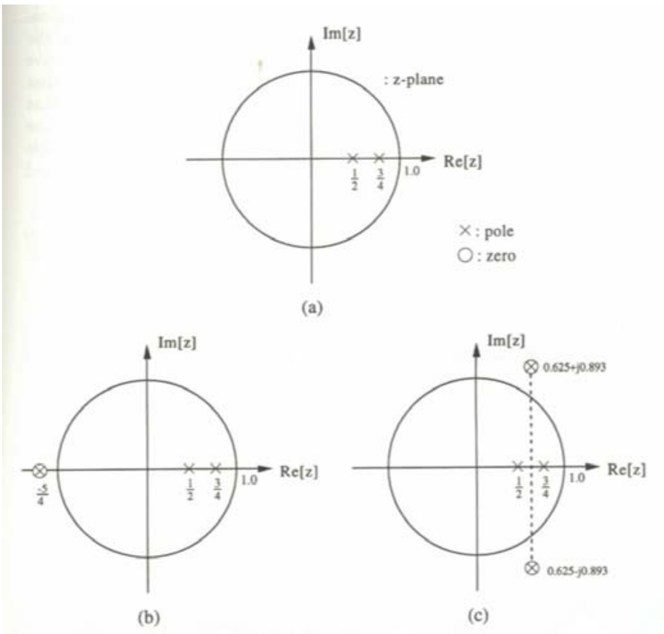
- Hardware implementation complexity
  - coefficients of the filters are pre-computed off-line
  - numerator (non-recursive) part requires N+M MPYs
  - denominator (recursive) part requires N MPYs
  - total (N + N + M) multiplications $\Rightarrow$ linear HW implementation complexity
- Filter stability problem
  - adding poles may lie outside the unit circle
  - example: assume the 2nd order IIR has poles at z = 0.5 and z = 0.75
  - 2-stage pipelining introduce an additional pole at z = -1.25

Clustered Look-Ahead Pipelining (6)

(a) pole zero representation of a stable 2nd order recursive filter

(b) poles & zeros after 2-stage pipelining using clustered look ahead

(c) poles & zeros after 3-stage pipelining using clustered look ahead



Clustered Look-Ahead Pipelining (7)

- Filter stability problem (cont.)
  - 3-stage pipelining introduce two additional poles at $z = 0.625 \pm j\,0.893$
  - note that the original filter is stable!

Stable Clustered Look ahead Filter Design (1)

- Consider a stable recursive digital filter

$$H(z) = \frac{N(z)}{D(z)} \Rightarrow H(z) = \frac{N(z)P(z)}{D(z)P(z)} = \frac{N(z)P(z)}{1 - z^{-M}Q(z)}$$

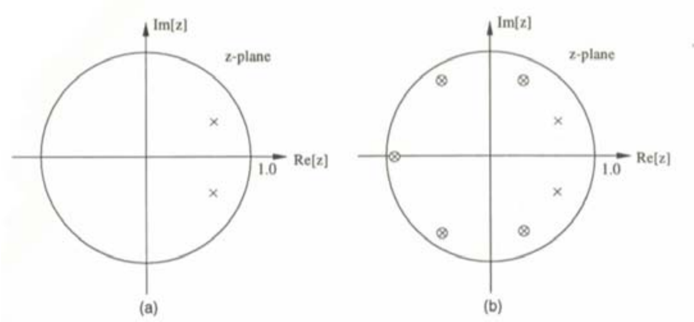
  - P(z) represents superfluous poles needed to create the desired pipeline delay M
  - for $M > M_c$ (some critical delay), the filter is always stable
  - numerical search methods are needed to obtain optimal pipelining level M and P(z)
  - Example: consider $H(z) = \dfrac{1}{1 - 1.5336 \cdot z^{-1} + 0.6889 \cdot z^{-2}}$
  - $M_c = 5$ and for $M = 6$,

$$H(z) = \frac{1 + 1.5336 \cdot z^{-1} + 1.663 \cdot z^{-1} + 1.4939 \cdot z^{-1} + 1.1454 \cdot z^{-1} + 0.7275 \cdot z^{-1}}{1 - 1.3265 \cdot z^{-6} + 0.5011 \cdot z^{-7}}$$

Stable Clustered Look ahead Filter Design (2)

(a) original poles

(b) poles & zeros of a 6-stage pipelined filter using a stable clustered look ahead



- when the no. of denominator multipliers is larger, pipelined filter suffers from large round-off noise
  - the denominator cannot be decomposed in order to maintain M level of pipelining

Scattered Look Ahead Pipelining (1)

- The denominator of H(z) is transformed to containing N terms, Z^{-M}, Z^{-2M}, ··· , Z^{-NM}
- equivalently, y(n) is computed in terms of N past scattered states y(n-M), y(n-2M), ··· , y(n-NM)
- for each pole, (M-1) canceling poles and zeros are introduced
  - with equal angular spacing
  - same distance from the origin as that of the original pole
  - e.g. if the original pole is z = *p*
  - added poles and zeros are $z = pe^{j2\pi k/M}$, for $k = 1,2,\cdots,(M-1)$

Scattered Look Ahead Pipelining (2)

- Assume the denominator of H(z) can be factorized as

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

- after scattered look-ahead transform

$$H(z)=\frac{N(z)}{D(z)}=\frac{N(z)\prod_{i=1}^{N}\prod_{k=1}^{M-1}(1-p_i e^{j2\pi k/M}z^{-1})}{\prod_{i=1}^{N}\prod_{k=0}^{M-1}(1-p_i e^{j2\pi k/M}z^{-1})}=\frac{N'(z)}{D'(Z^M)}$$

- consider the 2nd order filter, for M = 3

$$H(z)=\frac{1}{1-a_1z^{-1}-a_2z^{-2}}\Rightarrow$$

$$H(z)=\frac{1+a_1z^{-1}+(a_1^2+a_2)z^{-2}-a_1a_2z^{-3}+a_2^2z^{-4}}{1-(a_1^3+3a_1a_2)z^{-3}-a_2^3z^{-6}}$$

Scattered Look Ahead Pipelining (3)

- Consider the 2nd-order filter with complex conjugate poles at $z=r\cdot e^{\pm j\theta}$

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

- for 3-stage pipelining

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

- when $\theta=2\pi/3$, only 1 pole & zero at z = r are added

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

Scattered Look Ahead Pipelining (4)

- Consider the 2nd-order filter with poles at z = r_1, r_2

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

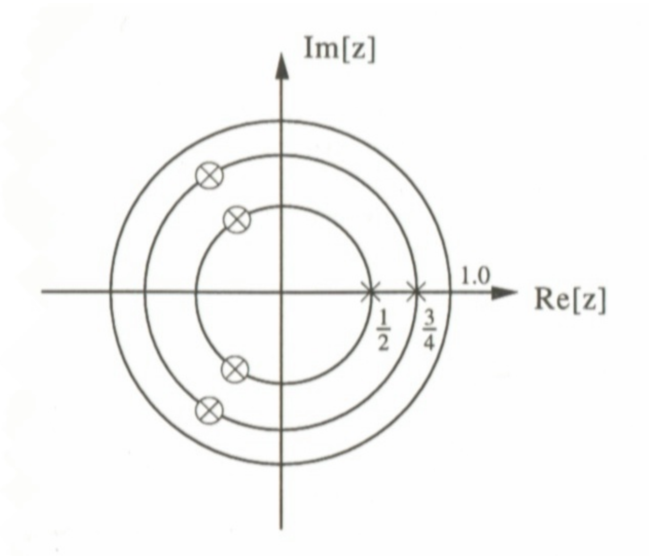
- for 3-stage pipelining adding poles at

$$z = r_1 \cdot e^{\pm j2\pi/3},\ z = r_2 \cdot e^{\pm j2\pi/3}$$

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

Scattered Look Ahead Pipelining (5)

- Pole zero representation of a 3-stage pipelined equivalent stable filter using scattered look ahead



Power of 2 Decomposition in Scattered Look Ahead Scheme (1)

- The multiplication complexity in scattered look ahead transform
  - non-recursive portion : (NM+1)
  - recursive portion is : N
  - much greater than that in clustered look ahead scheme
- consider a recursive filter

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

- by multiplying $(1-\sum_{i=1}^{N}(-1)^i a_i z^{-i})$ in both numerator and denominator

Power of 2 Decomposition (2)

- For 2-stage pipelining

$$H'(z)=\frac{\sum_{i=0}^{N} b_i \cdot z^{-1}(1-\sum_{i=1}^{N}(-1)^i a_i \cdot z^{-i})}{[1-\sum_{i=1}^{N} a_i \cdot z^{-i}][1-\sum_{i=1}^{N}(-1)^i a_i \cdot z^{-i}]}=\frac{N'(z)}{D'(z^2)}$$

- the denominator contains terms of z^{-2}, z^{-4}, … ,z^{-2N}
- subsequent transforms can be applied to obtain 4,8 and 16 stage pipelined implementations
- $\log_2 M$ sets of transforms are needed to achieve M-stage pipelining
- each transform double the speed but also increase the hardware complexity by N

Power of 2 Decomposition (3)

- Applying ($\log_2 M$-1) sets of transforms
- leads to M-stage pipelining
- requires (2N+N* $\log_2 M$+1) multiplications
- logarithmic complexity w.r.t. M (speed up)
- total number of delays $\approx$NM*($\log_2 M$+1)
- NM delays are used in the non-recursive portion
- NM*$\log_2 M$ delays are required to pipeline each N*$\log_2 M$ multipliers by M stages

Power of 2 Decomposition (4)

- N(M-1) poles and zeros are added
- N(M-1) zeros are decomposed and implemented in $\log_2 M$ stages
- 1st stage realizes N-th order nonrecursive section
- following stages implements 2N, 4N, ··· , NM/2-order non-recursive sections
- each section requires N multiplication
  - due to the symmetry of coefficients
  - independent of the order of the section

Power of 2 Decomposition (5)

- Consider a 2nd-order recursive filter example

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

- the poles are located at $re^{\pm j\theta}$
- the pipelined function

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

- the 2M poles are located at

$$z=re^{\pm j(\theta+i(2\pi/M))},\qquad i=0,1,2,\cdots,(M-1)$$

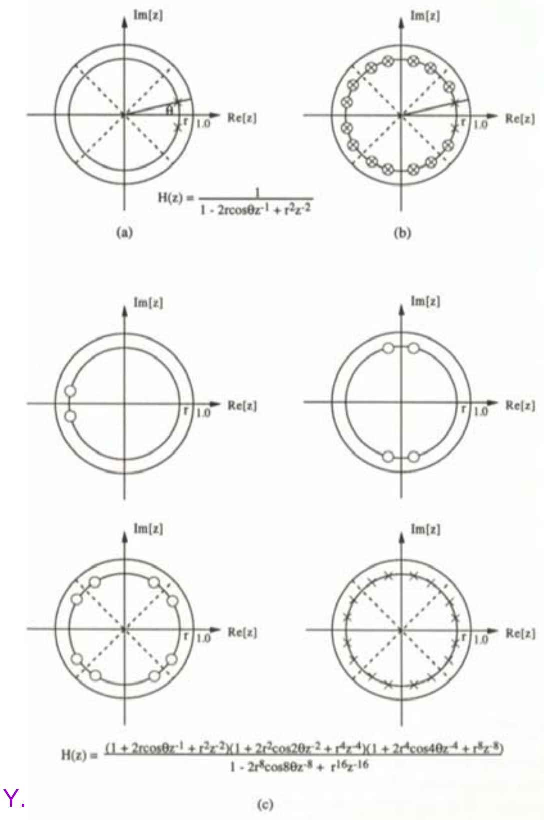
- implementation complexity is $(2\log_2 M+5)$ multiplications

Power of 2 Decomposition (6)

(a) pole diagram of the 2nd order filter

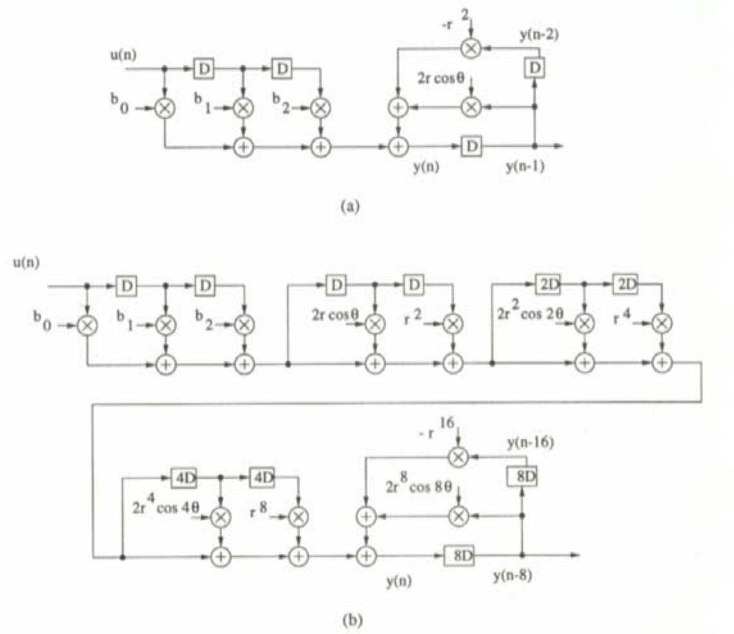
(b) poles & zeros of the pipelined 2nd order filter with 8 loop pipelining stages

(c) decomposition of poles and zeros of the pipelined filter



Power of 2 Decomposition (7)

- Implementation of the original and the pipelined scattered look-ahead recursive filters



Scattered LA Pipelining with General decomposition

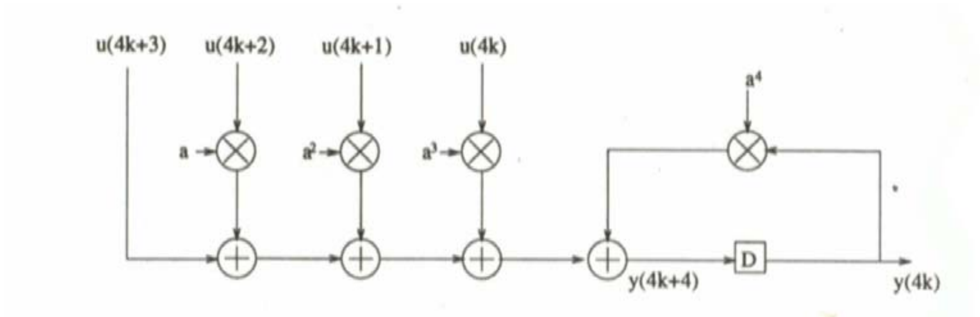
- For an N-th order filter with M levels of pipelining
  - N(M-1) canceling zeros
  - for $M=M_1M_2 \cdots M_p$ decomposition
  - the 1st stage implements $N(M_1-1)$ zeros
  - the 2nd stage implements $NM_1 (M_2-1)$ zeros
  - the Pth stage implements $NM_1M_2 \cdots M_{p-1}(M_p-1)$ zeros
- the non-recursive portion requires NM delays and $N \cdot \sum_{i=1}^{P}(M_i - 1)$ multipliers
- generation decomposition of high order IIR filter
  - decompose into cascade of 1st order sections
  - apply general decomposition to each 1st order section

Parallel Processing for IIR Filters (1)

- 1st order IIR filter case
  - $H(z) = \dfrac{z^{-1}}{1 - az^{-1}}$ where $|a| \le 1$
  - y(n+1) = ay(n)+u(n)
  - for a 4-parallel architecture implementation
  - $y(n+4)=a^4y(n)+a^3u(n)+a^2u(n+1)+au(n+2)+u(n+3)$
  - by substituting n = 4k
  - $y(4k+4)=a^4y(4k)+a^3u(4k)+a^2u(4k+1)+au(4k+2)+u(4k+3)$
  - pole of the original system is at z = a
  - pole of the parallel system is at $z = a^4$
  - pole is moved toward the origin ⇒ improved robustness to the round-off error

Parallel Processing for IIR Filters (2)

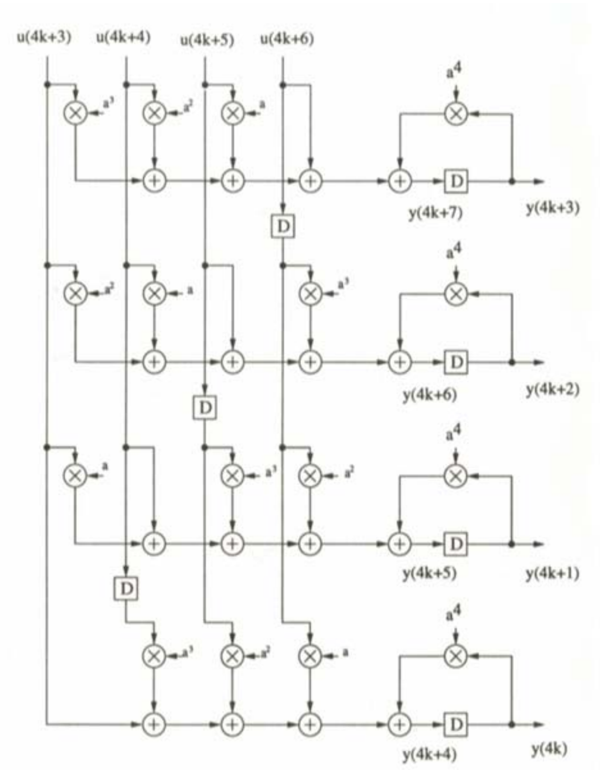
- Note that in pipelined processing case, canceling poles and zeros are introduced



- substituting n with 4k+4, 4k+5, 4k+6, 4k+7 leading to 4 parallel processing blocks

Parallel Processing for IIR Filters (3)

- require L^2 MAC operations
- one delay element represents L sample delays



Parallel Processing for IIR Filters (4)

- Incremental block processing
  - use y(4k) to compute y(4k+1),
  - i.e. $y(4k+1) = a \cdot y(4k) + u(4k)$
  - and then use y(4k+1) to compute y(4k+2), then y(4k+3)
  - the hardware complexity is reduced from L^2 to 2L-1
  - increased computation time for y(4k+1), y(4k+2), y(4k+3)

Parallel Processing for IIR Filters (5)

- Incremental block filter structure with L = 4

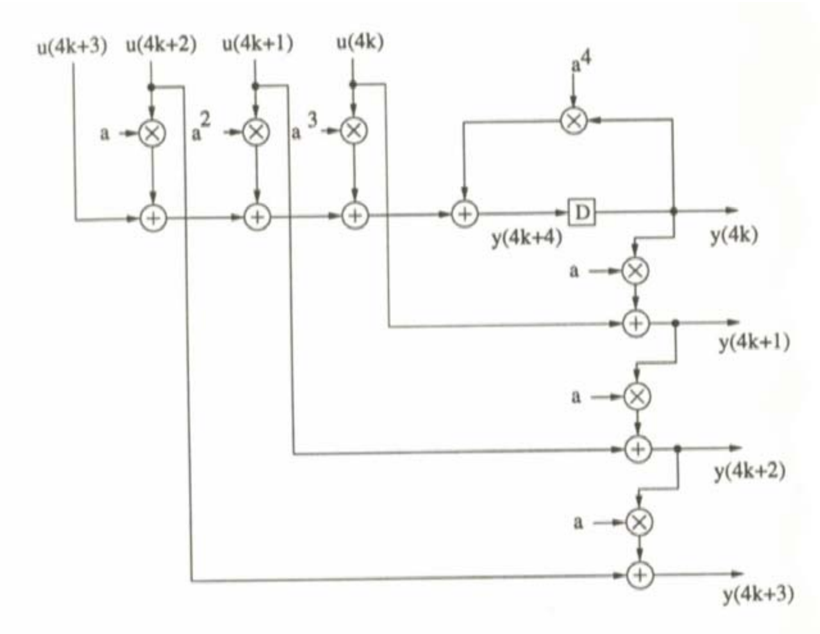


Fig. 10.16 Incremental block filter structure with $L = 4$.

Parallel Processing for IIR Filters (6)

- A second order IIR example for block processing
  - $$H(z) = \frac{(1+z^{-1})^2}{1 - \frac{5}{4}z^{-1} + \frac{3}{8}z^{-2}}$$

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

    where $u(n) + 2u(n-1) + u(n-2)$
- 3-parallel system
  - compute y(3k+3) and y(3k+4) using y(3k) and y(3k+1)
  - for y(3k), y(3k+1) $\rightarrow$y(3k+3) $\Rightarrow$ 2-stage clustered LA
  - for y(3k), y(3k+1) $\rightarrow$y(3k+4) $\Rightarrow$ 3-stage clustered LA

Parallel Processing for IIR Filters (7)

- 2 & 3-stage clustered look ahead

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

- 2-loop update equation

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

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

Parallel Processing for IIR Filters (8)

- Pole zero plots of the transfer function
- loop update for block size = 3
- relationship of recursion outputs

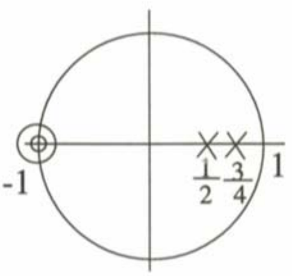
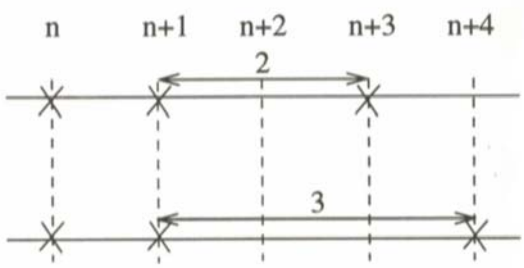


Fig. 10.17 Pole zero plots for the transfer function.



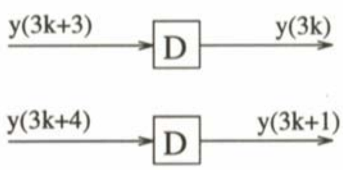
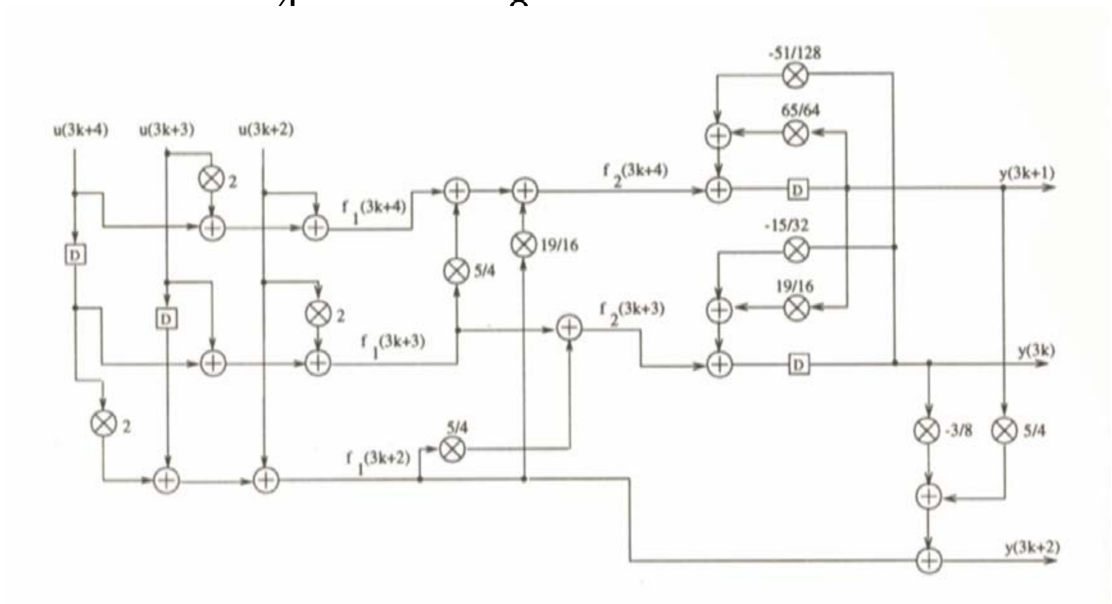


Fig. 10.18 Loop update for block size = 3.

Parallel Processing for IIR Filters (9)

- y(3k+2) can be obtained incrementally

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



Comments

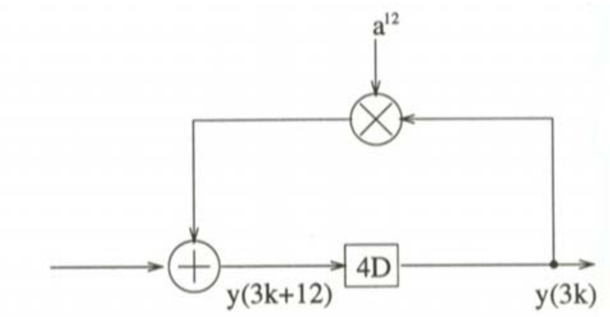
- The original system has 2 poles at 1/2 and 3/4
- transformed system

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

  - the matrix has eigenvalues $(1/2)^3$ and $(3/4)^3$
  - the parallel system is more stable
  - the parallel system has the same number of poles as the original system
- hardware complexity
  - for a 2nd order IIR filter (N=2)
  - 3L + [(L-2) + (L-1)] + 4 + 2(L-2) = 7L -3 MPY operations

Combined Parallel & Pipelining Processing (1)

- To achieve a speed up L × M
  - L is the block size and M is the no. of pipelining stage
  - for $H(z)=1/(1-a\cdot z^{-1})$ with M = 4 and L = 3
  - H(z) is 1st order ⇒ 1 loop update equation is required and the other two can be computed incrementally
  - M = 4 ⇒ loop contains 4 delay elements
  - L = 3 ⇒ y(3k+12) ← y(3k)



Combined Parallel & Pipelining Processing (2)

- 12-step look ahead transform

$$y(n) = ay(n-1) + u(n)$$
$$= a^{12}y(n-12) + a^{11}u(n-11) + a^{10}u(n-10) + \cdots + u(n)$$

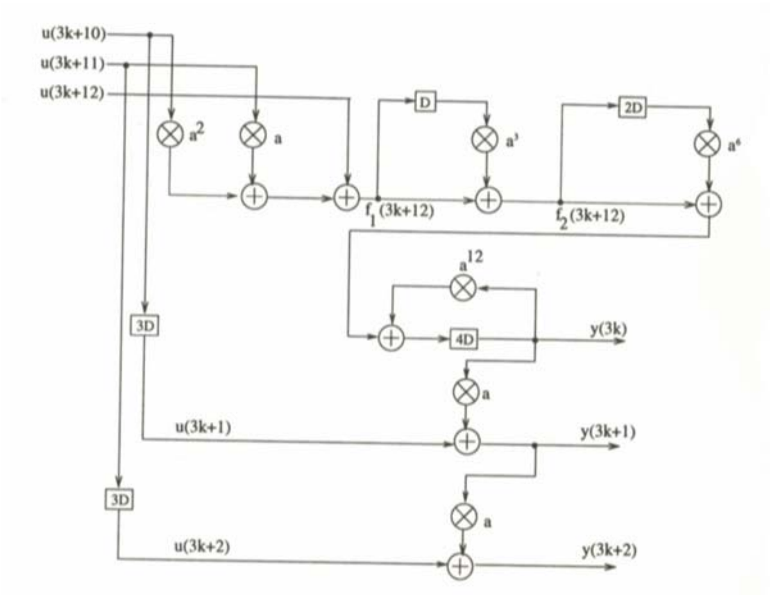
  - substituting n=3k+12

$$\begin{aligned} y(3k+12) = a^{12}y(3k) & \\ & + a^{11}u(3k+1) + a^{10}u(3k+2) + a^{9}u(3k+3) \\ & + a^{8}u(3k+4) + a^{7}u(3k+5) + a^{6}u(3k+6) \\ & + a^{5}u(3k+7) + a^{4}u(3k+8) + a^{3}u(3k+9) \\ & + a^{2}u(3k+10) + au(3k+11) + u(3k+12) \\ & = a^{12}y(3k) + a^{6} f_2(3k+6) + a^{3} f_1(3k+9) + f_1(3k+12) \end{aligned}$$

Combined Parallel & Pipelining Processing (3)

- Where

$$f_1(3k+12) = a^2u(3k+10) + au(3k+11) + u(3k+12)$$
$$f_2(3k+12) = a^3 f_1(3k+9) + f_1(3k+12)$$

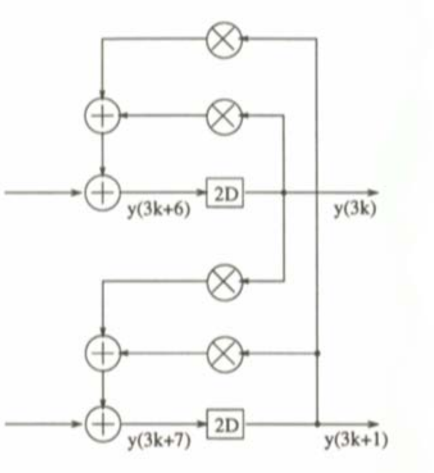


Combined Parallel & Pipelining Processing (4)

- Comments
  - has 4 poles a^3, $-a^3$, ja^3, $-ja^3$
  - decomposition is used for the non-recursive portion
  - multiplication complexity is $2L-1+\log_2 M$
- 2nd order filter example with L = 3 and M = 2

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

  - loop update operations
  - original poles 1/2, 3/4
  - new poles $\pm(1/2)^3$, $\pm(3/4)^3$



Comments

- Systematic approach to compute pole locations
  - write loop update equations using LM-level look ahead
  - write the state space representation of the parallel pipelined filter, where state matrix $A_{N \times N}$
  - compute the eienvalues λ_i of matrix A, $1 \leq I \leq N$
  - NM poles of the new parallel pipelined system correspond to the M-th roots of the eigenvalues of A, $\lambda_i^{\frac{1}{M}}$

Low Power IIR Filter Design

- Parallel and pipelined processing can be used for low power design
- consider a 4-th order Chebyshev low-pass filter

$$H(z) = \frac{0.001836(1+z^{-1})^4}{(1-1.5548z^{-1}+0.6493z^{-2})(1-1.4996z^{-1}+0.8482z^{-2})}$$

  - assume multiplier capacitance is dominant
  - assume Vdd = 5V and Vt = 1v
  - using scattered look ahead with power of 2 decomposition

Pipelined processing to reduce power (1)

- 4-level pipelinable transfer function N(z)/D(z)

$$N(z) = 0.001836(1+z^{-1})^4 \times$$
$$(1+1.5548z^{-1}+0.6493z^{-2})(1+1.4996z^{-1}+0.8482z^{-2})\times$$
$$(1+1.1188z^{-2}+0.4216z^{-4})(1+0.5524z^{-2}+0.7194z^{-4})$$
$$D(z) = (1-0.4085z^{-4}+0.1777z^{-8})(1+1.1337z^{-4}+0.5175z^{-8})$$

  - pipelined design insert delays into the recursive
  - shorter critical path after retiming
  - smaller amount of charging/discharging capacitance in each pipeline stage
  - use lower power supply yet maintain the original speed
  - the pipelined supplied βV_0, with $\beta < 1$

Pipelined processing to reduce power (2)

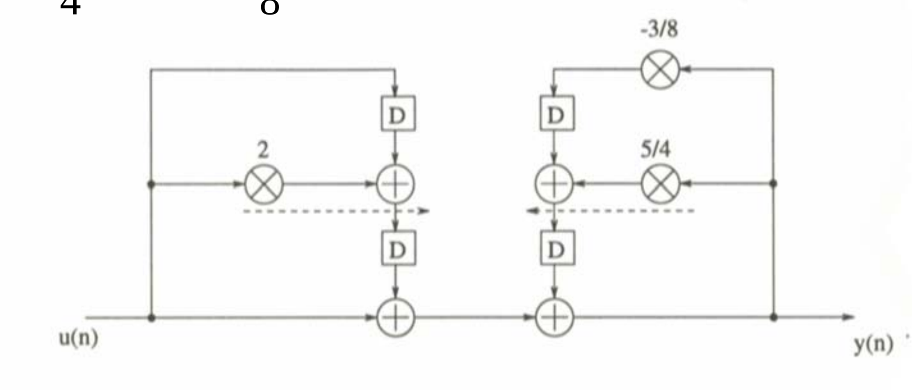
- propagation delay
  - sequential system is $T_{pd} = \dfrac{C_{charge} \times V_{dd}}{k(V_{dd}-V_t)^2} = \dfrac{C_{charge} \times 5}{k(5-1)^2}$
  - 4-level pipelined system $T_{pd} = \dfrac{C_{charge}}{4} \dfrac{5\beta}{k(5\beta-1)^2}$
  - $\Rightarrow$ β=0.476 and new Vdd = 2.38V
- power dissipation
  - the sample period is equal to T_{pd}
  - sequential system is $P_{seq} = \dfrac{C_{total}^{(seq)} \times 5^2}{T_{pd}} = m_{seq} C_M 5^2 f_s$
  - 4-level pipelined system $P_{pip} = \dfrac{C_{total}^{(pip)} \times 2.38^2}{T_{pd}} = m_{pip} C_M 2.38^2 f_s$

Pipelined processing to reduce power (3)

- Power ratio
  - number of multiplications, m_{seq}=5, m_{pip} = 13

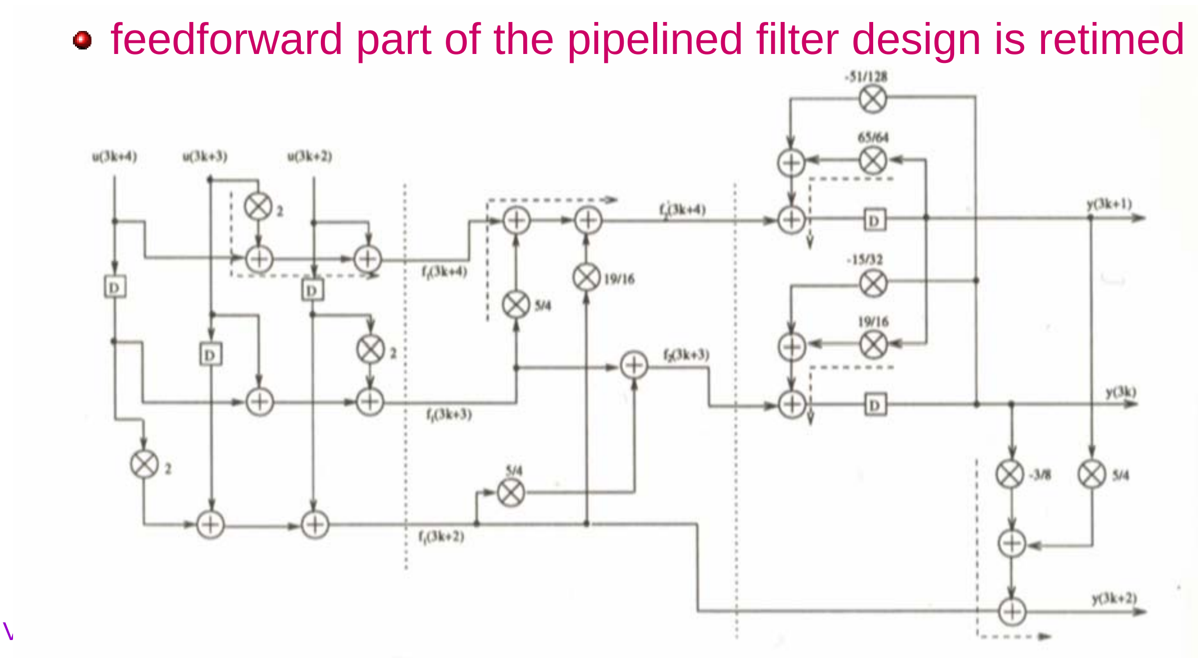
$$Ratio = \frac{P_{pip}}{P_{seq}} \left(\frac{13}{5}\right)\left(\frac{2.38}{5}\right)^2 = 0.5891$$

- Example: a second order IIR filter
  - $y(n) = \frac{5}{4}y(n-1) - \frac{3}{8}y(n-2) + u(n) + 2u(n-1) + u(n-2)$



Parallel processing to reduce the power (1)

- 3-parallel filter design ⇒ 3 outputs per clock cycle
- clock cycle can be three times the sample cycle
- the critical path of the sequential filter is 1 MPY + 1 Add
- feedforward part of the pipelined filter design is retimed



Parallel processing to reduce the power (2)

- Propagation delay
  - The critical path of the parallel design is 1 MPY + 2 Add
  - assume the charge/discharge capacitances and the delays are dominated by the multipliers
  - assume the parallel filter supplied βV_0, with $\beta < 1$ and the threshold voltage V_t=0.6V
  - propagation delay (clock cycle) for sequential filter is

$$T_{seq} = \frac{C_M V_0}{k(V_0 - V_t)^2}$$

  - propagation delay (clock cycle) for parallel filter is

$$T_{par} = \frac{C_M \beta V_0}{k(\beta V_0 - V_t)^2}$$

Parallel processing to reduce the power (3)

  - Choose $T_{par} = 3\, T_{seq} \Rightarrow \quad 3 \cdot \frac{V_0}{(V_0 - V_t)^2} = \frac{\beta V_0}{(\beta V_0 - V_t)^2}$
  - we get β=0.4673 and βV_0=2.3365V
- power consumption

$$P_{seq} = 3 C_M V_0^2 f_s$$

$$P_{par} = 12 C_M (\beta V_0)^2 \frac{f_s}{3} = 4\beta^2 C_M V_0^2 f_s$$

$$Ratio = \frac{P_{par}}{P_{seq}} = \frac{4\beta^2}{3} = 29.116\%$$

- comments
  - the parallel or pipelined design can be used either to achieve high speed or low power operations

Pipelined Adaptive Digital Filters (1)

- Adaptive digital filters are difficult because
  - presence of long feedback loop
  - look ahead transform leads to prohibitively large hardware overhead
  - note that look ahead transform maintains exact input-output mapping
  - since coefficients of the adaptive filters continue to change $\Rightarrow$ exact input-output mapping is not necessary
- adaptive filter measurement indexes
  - mis-adjustment error
  - adaptation time constant

Pipelined Adaptive Digital Filters (2)

- Relaxed look-ahead transform
  - aims to pipeline adaptive filter with little hardware overhead and at the expense of marginal degradation of adaptation behavior
- product relaxation
- sum relaxation
- delay relaxation

  - the adaptation behavior of the pipelined adaptive filters should be analyzed using stochastic techniques

Relaxed Look Ahead (1)

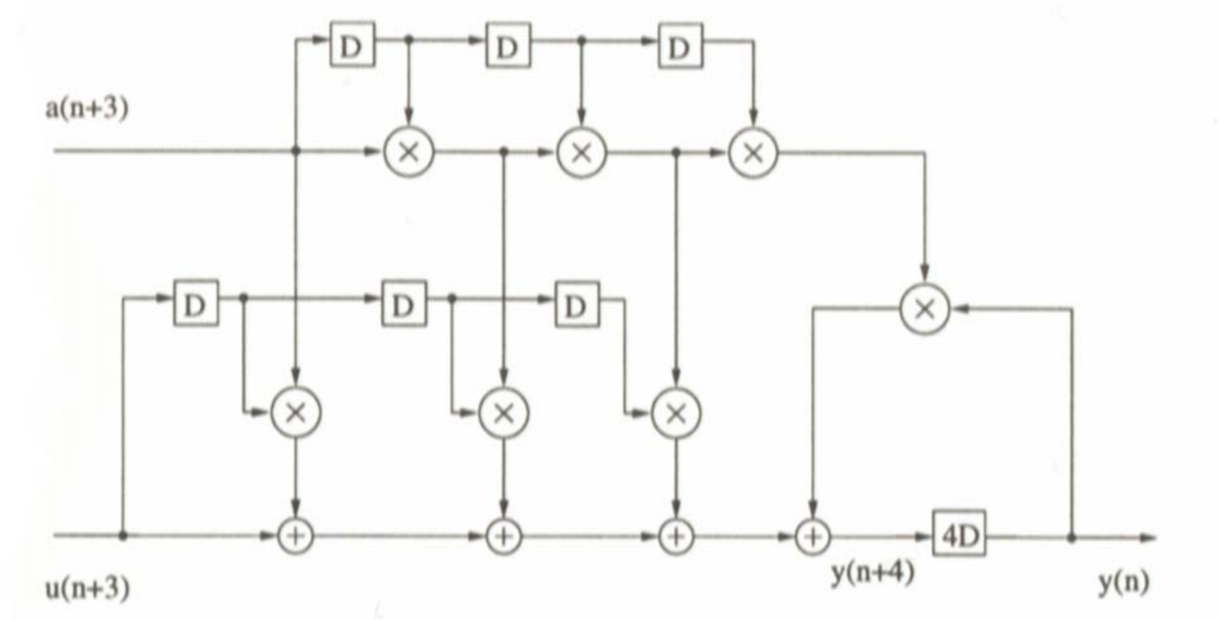
- Consider the 1st order time varying recursion
  - y(n+1)=a(n)y(n)+u(n), where coefficient a(n) is time varying
  - using look ahead transform

$$y(n+M) = \left(\prod_{i=0}^{M-1} a(n+M-1-i)\right)\cdot y(n) + \sum_{i=1}^{M-1}\left[\prod_{j=0}^{i-1} a(n+M-1-j)\right]\cdot u(n+M-1-i) + u(n+M-1)$$

  - for M = 4, the iteration period is limited to $(T_m+T_a)/4$
  - under certain circumstances, we can substitute approximation expressions to simplify the terms in the RHS

Relaxed Look ahead Transform (2)

- A 4-level look head pipelined 1st order



Product Relaxation (1)

- Assumptions
  - a(n) is close to 1 and a(n) = (1-ε(n)), where ε(n) is close to zero

$$\prod_{i=0}^{M-1} a(n+i) \approx a(n+M-1)^M = (1-\varepsilon(n+M-1))^M$$

$$= 1 - M\varepsilon(N+M-1) = 1 - M(1-a(n+M-1))$$

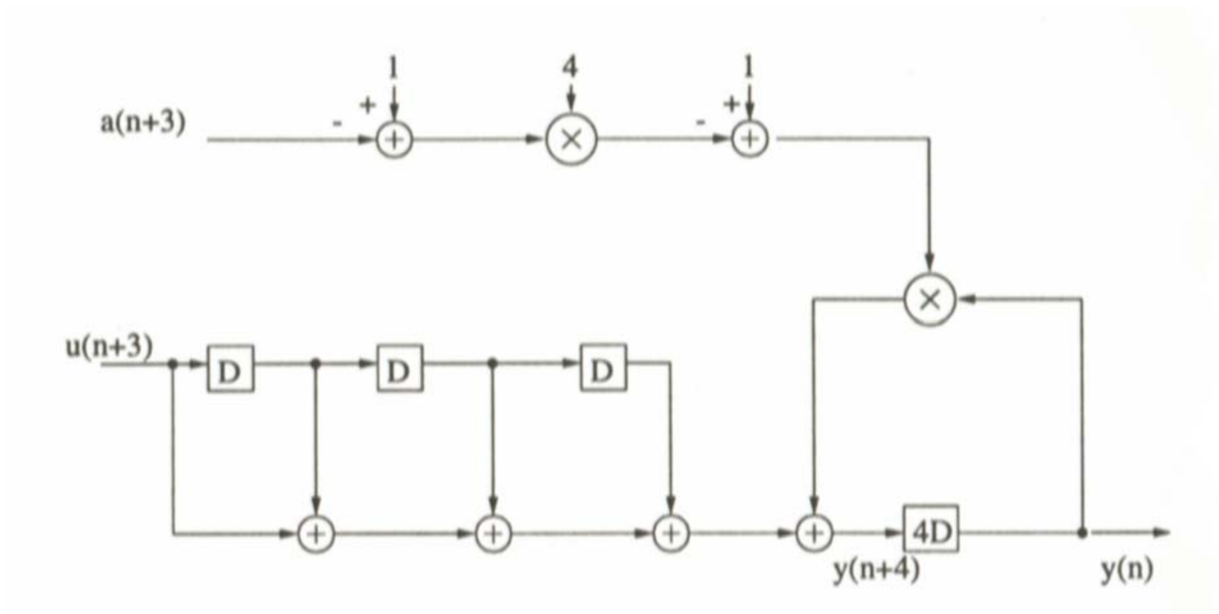
  - if $u(n+i)$ is close to zero, then

$$\left[\prod_{j=0}^{i-1} a(n+M-1-j)\right] \cdot u(n+M-1-i) \approx u(n+M-1-i)$$

$$\Rightarrow y(n+M) = (1-M(1-a(n+M-1)))y(n) + \sum_{i=0}^{M-1} u(n+M-1-i)$$

Product Relaxation (2)

- A 4-level product relaxed lookahead pipelined 1st order recursion



国立中興大學
National ChungHsing University

Product Relaxation (3)

- Another approximation
  - a(n) is assumed to be close to 1

$$\prod_{j=0}^{i-1} a(n+M-1-j) \approx a(n+M-1)$$

$$\Rightarrow y(n+M) = a(n+M-1)y(n) + \sum_{i=0}^{M-1} u(n+M-1-i)$$

国立中興大學
National ChungHsing University

Sum Relaxation (1)

- If the input u(n) varies slowly over M cycles

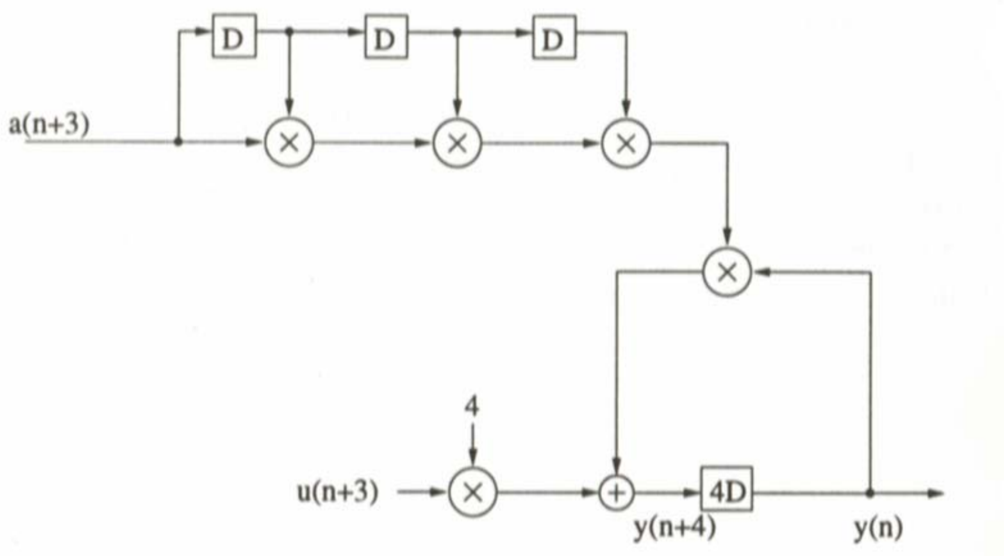
$$\begin{aligned} y(n+M) &= \prod_{i=0}^{M-1} a(n+M-1-i)y(n) \\ &\quad + \sum_{i=1}^{M-1}\left[\prod_{j=0}^{i-1} a(n+M-1-j)\right]\cdot u(n+M-1-i) + u(n+M-1) \\ &\approx \prod_{i=0}^{M-1} a(n+M-1-i)y(n) + Mu(n) \end{aligned}$$

  - if u(n) is close to zero, then $M\,u(n) \approx u(n)$

$$y(n+M) \approx \prod_{i=0}^{M-1} a(n+M-1-i)y(n) + u(n)$$

Sum Relaxation (2)

- A 4-level sum-relaxed look-ahead pipelined 1st-order recursion



Delay Relaxation (1)

- Consider the recursion y(n)=y(n-1)+a(n)u(n)
- M-level look-ahead pipelined version

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

  - delay relaxation involves the use of delayed input u(n-M') and delayed coefficient a(n-M')
  - assume the product a(n)u(n) is more or less constant over M' samples
  - a reasonable assumption for stationary or slowly varying product a(n)u(n)

$$y(n) \approx y(n-M) + \sum_{i=0}^{M-1} a(n-M'-i)u(n-M'-i)$$

Comments of relaxation techniques

- Three relaxation techniques can be applied individually or in different combinations to derive a rich variety of architectures
- each derived architecture has different convergence characteristics
- the relaxed look-ahead is a transform in the stochastic sense

Pipelined LMS Adaptive Filter (1)

- LMS algorithm

$$e(n) = d(n) - \hat{d}(n) = d(n) - W^T(n-1)U(n)$$
$$W(n) = W(n-1) + \mu \cdot e(n) \cdot U(n)$$

- 2 recursive loops in the LMS architecture
  - weight update loop
  - error feedback loop
- direct look ahead

$$W(n) = W(n-1) + \mu \cdot e(n) \cdot U(n)$$
$$= W(n-M_2) + \sum_{i=0}^{M_2-1} \mu \cdot e(n-i) \cdot U(n-i)$$

  - M_2 is the delay in the weight update loop

Pipelined LMS Adaptive Filter (2)

- By delay relaxation
  - assume the gradient estimate e(n)U(n) varies slowly

$$W(n) = W(n-M_2) + \sum_{i=0}^{M_2-1} \mu \cdot e(n-i) \cdot U(n-i)$$
$$= W(n-M_2) + \sum_{i=0}^{M_2-1} \mu \cdot e(n-M_1-i) \cdot U(n-M_1-i)$$

  - the hardware complexity is $N(M_2-1)$ MACs
  - further sum relaxation taking only the first M'_2 sum terms, where $M'_2 \le M_2$

$$e(n) = d(n) - W^T(n-1)U(n)$$
$$= d(n) - [W^T(n-M_2-1)$$
$$+ \mu \sum_{i=0}^{M'_2-1} e(n-M_1-i-1)U(n-M_1-i-1)]U(n)$$

Pipelined LMS Adaptive Filter (3)

- Assume μ is sufficiently small and replacing $W(n-M_2-1)$ by $W(n-M_2)$

$$e(n) = d(n) - W^T(n-M_2)U(n)$$