S-38.220 Postgraduate course on signal processing in communications
FALL 1999

Pipelined and parallel recursive and adaptive filters

Kari Seppänen
VTT Information Technology
Kari.Seppanen@vtt.fi

November 16, 1999

Contents

Chapter 1

Introduction

Any desired digital filter spectrum could be realized using either FIR or IIR digital filters. IIR filter is often preferred because specified spectrum could be realized using a much lower order IIR than FIR. In time-varying systems, e.g., noice cancellation, echo cancellation, etc., whrere FIR and IIR filters are not useful adaptive filters are utilized. Instead of constant coefficients their coefficients are adapted at each iteration until they converge.

Using pipelining or parallel processing is straightforward in nonrecursive computations. Recursive and adaptive filters cannot be easily pipelined due to feedback loops in these filters. This presentation introduces some approches based on *look-ahead computation*, *incremental block processing* and *relaxed look-ahead* techniques.

Chapter 2

Pipeline interleaving in digital filters

2.1 Inefficient single/multichannel interleaving

First-order linear time-invariant (LTI) recursion is described by

$$y(n+1) = ay(n) + bu(n)$$

shown in figure 2.1. The iteration period is $(T_m + T_a)$.

M-stage pipelined version is obtained by inserting $(M-1)$ additional latches inside the loop (figure 2.2). The clock period could be reduced by M times but at the same time sample period will increase to M clock periods. For single time series system will be useful only for $1/M$ of the time. If M independent time seriers are available the hardware can be fully utilized.

This slow interleaved system which is often called as *M-slow circuit* is suitable for applications requiring nominal concurency. It is inefficient if there is not M indepentend time series available.

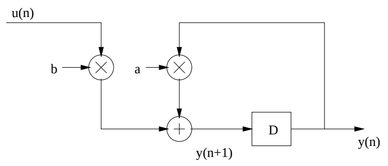

Figure 2.1: A simple 1st-order recursion

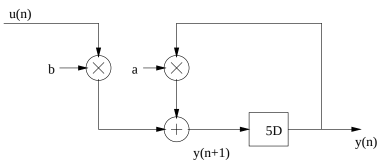

Figure 2.2: A simple 1st-order recursion with 5-way interleaving

2.2 Efficient single-channel interleaving

Look-ahead transformation can be used to overcome the limitations of the M slow system. If 1st-order LTI is recasted to experess $y(n+2)$ as a function of $y(n)$ we get

$$y(n+2) = a[ay(n) + bu(n)] + bu(n).$$

Iteration bound for this system is $2(T_m + T_a)/2$ which is same as in M slow system. However, equivalent recursion is expressed by

$$y(n+2) = a^2 y(n) + abu(n) + bu(n)$$

which have iteration bound of $(T_m + T_a)/2$.

Generally $(M-1)$ steps of look-ahead of the iteration are descibed by

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

Iteration bound is $(T_m+T_a)/M$, which corresponds to M times higher sample rate. Complexity and system latency are lineary increased.

The steady-state behavior of the system is not altered, i.e., for sufficiently old inputs the outputs of the transformed system and the original system will be identical.

Chapter 3

Pipelining in 1st-order IIR digital filters

With look-ahead techinque canceling zeroes and poles with equal angular spacing and a similar distance from the origin than the original pole are introduced. Pipelined realizations are always guaranteed to be stable for 1st-order IIR filters if the original filter is stable.

Decomposition technique is used to implement the nonrecursive portion generated by look-ahead process to obtain logarithmic increase in hardware.

Example: 1st-order IIR with transfer function

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

gives

$$y(n) = ay(n-1) + u(n).$$

The sample rate is limited by the computation time of one multiply-add operation.

3.1 Look-ahead pipelining for 1st-order IIR filters

Basic idea in look-ahead pipelining is to add cancelling poles and zeros to transfer function such that the coefficients of $z^{-1}, \vdots, z^{-(M-1)}$ are zero. Result is that there will be M delay elements in the critical loop.

Example:

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

adding poles and zeroes to $z = ae^{\pm(j2\pi/3)}$

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

3.2 Look-ahead pipelining with power-of-2 decomposition

With power-of-2 decomposition, M-stage pipelined implementation can be obtained by $\log_2 M$ sets of transformations. If original transfer function has a single pole at a, the pipelined version has poles at locations

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

. Total complexity of pipelined implemetation is $\log_2 M + 2$.

Although the pipelined recursive filters are stable under infinite precision condition, they are sensitive to coefficients under finite precision. The pole location is more sensitive for small values but this is not a problem since a filter with poles close to origin are more stable.

Finite precision causes also inexact pole-zero cancellation which leads to phase and magnitude errors which can be reduces by increasing wordlength.

3.3 Look-ahead pipelining with general decomposition

The idea of decomposition can be extended to to any arbitrary numeber of M. If $M = M_1 M_2 \ldots M_p$, nonrecursive stages implement $(M_1 - 1), M_1(M_2 - 1), \ldots, M = M_1 M_2 \ldots M_{p-1}(M_p - 1)$ zeros totaling $(M - 1)$ zeros.

Chapter 4

Pipelining in Higer-order IIR digital filters

For higher order filters there are two techiques for pipelining: *clustered* and *scattered look-ahead* techniques. First technique require linear complexity but does not always quarantee stability. 2nd technique can be use to derive stable pipelined filters. Pipelining can be achieved with out pole-zero cancellation by using *constrained filter design technique.*

Direct form N-th order recursive filter is described by

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

giving

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

4.1 Clustered look-ahead pipelining

Basic idea of clustered look-ahead pipelining is to add cancelling poles and zeros to the filter transfer function such that the coefficients of $z^{-1}, \ldots, z^{-(M-1)}$ are zero. Thus, loop can be pipelined by M stages.

The numerator can be implemented with (N+M) multiplications and the denominator with N multipications. Thus, total complexity is (N+N+M) which is linear to speedup. However, the additional poles may lie outside the unit circle, which changes the stability of the filter.

4.2 Stable clustered look-ahead filter design

It has been shown that clustered look-ahead transformation always produces a stable filter at some critical delay M_c such that stability will be assured for $M > M_c$. So, if desired pipeline delay M does not produce stable filter M should be increased until a stable filter is obtained.

When the number of denominator multipliers is large, the filter will suffer from roundoff noise.

4.3 Scattered look-ahead pipelining

In this look-ahead for each pole in the original filter $(M-1)$ cancelling poles and zeros with equal angular spacing and a same distance from orgining as the original pole are added. This look-ahead approach produces stable pipelined filters if the original filter is stable. The complexity of numerator is (NM+1) multiplications and N multiplications for denumerator. Total complexity is (NM+N+1) which is linear with respect to M but much greater than that of clustered look-ahead. It should be noted that the latch complexity is M^2.

4.4 Scattered look-ahead pipelining with power-of-2 decomposition

A 2-stage pipelined implementation can be obtained by multiplying the numerator and denominator by

$$1 - \sum_{i=1}^{N} (-1)^i a_i z^{-i}.$$

Similarily, subsequent transformations lead to 4, 8, and 16 stage pipelined implementations. Each transformation leads to an increase in multiplication by N while doubling the speed or sample rate. M-stage pipelining can be achieve applying $\log_2 M$ transformations, which leads to multiplication complexity of $(2N + N\log_2 M + 1)$. Total number of delays is $NM(\log_2 M + 1)$.

4.5 Scattered look-ahead pipelining with general decomposition

General decomposition for arbitrary number of pipeline stages can be done in similar way as in 1st-order IIR.

4.6 Constrained filter design techinques

The idea behind constrained filter design is to constrain the denominator to be polynomial in z^M rather than z, i.e., to expess it in scattered look-ahead from. The constrained filter design requires less complexity and produces less roundoff noise. Two methods that can be used are "Modified Deczky's filter design" and "Martinez-Parks decimation filter desing".

Chapter 5

Parallel processing for IIR filters

1st-order IIR

$$y(n+1) = ay(n) + u(n)$$

can be parallelized (here 4-parallel) by iterating the recursion or by applying look-ahead technique

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

and substituting $n = 4k$ to get single stage

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

Block processing structure can be obtained by subsituting $n = 4k+4, 4k+5, 4k+6, 4k+7$.

Hardware complexity of such system is squared in respect of parallelism. Such system is hardware expensive but it is robust to roundoff noise.

By using incremental block processing hardware complexity is reduced but delay and roundoff noise are increaced. The idea is to use $y(4k)$ to compute $y(4k+1)$, $y(4k+1)$ to compute $y(4k+2)$, etc.

Appendix A

Exercise

Homework is 10.4.