INTRODUCTION

Infinite impulse response (IIR) filters are used in applications where the computational efficiency is the highest priority. It is well known that an IIR filter transfer function is of a considerably lower order than the transfer function of an FIR equivalent. The drawbacks of an IIR filter are the nonlinear phase characteristic and sensitivity to quantization errors.

In multirate applications, the computational requirements for FIR filters can be reduced by the sampling rate conversion factor as demonstrated in Chapter IV. However, such a degree of computation savings cannot be achieved in multirate implementations of IIR filters. This is due to the fact that every sample value computed in the recursive loop is needed for evaluating an output sample. Based on the polyphase decomposition, several techniques have been developed which improve the efficiency of IIR decimators and interpolators as will be shown later on in this chapter.

In this chapter, we consider first the direct implementation structures for IIR decimators and interpolators. In the sequel, we demonstrate the computational requirements for direct form IIR decimators and interpolators. The polyphase decomposition of an IIR transfer function is explained with its application to decimation and interpolation. Then, we demonstrate an efficient IIR polyphase structure based on all-pass subfilters, which is applicable to a restricted class of decimators and interpolators. In this chapter, we discuss the application of the elliptic minimal Q factor (EMQF) filter transfer function in constructing high-performance decimators and interpolators. The chapter concludes with a selection of MATLAB exercises for the individual study.
DIRECT IMPLEMENTATION STRUCTURES FOR IIR FILTERS FOR DECIMATION AND INTERPOLATION

For studying the efficiency of IIR decimators and interpolators let us consider the factor-of-$M$ decimator depicted in Figure 5.1, where the decimation filter $H(z)$ is an $N$th order IIR transfer function.

For an IIR decimation filter, the $H(z)$ is the transfer function of the form

$$H(z) = \frac{V(z)}{X(z)} = \frac{B(z)}{A(z)},$$  \hspace{1cm} (5.1)

where $A(z)$ and $B(z)$ are $N$th order polynomials in $z^{-1}$,

$$B(z) = \sum_{n=0}^{N} b_n z^{-n}, \quad A(z) = 1 + \sum_{n=1}^{N} a_n z^{-n}.$$  \hspace{1cm} (5.2)

The input-output relations (5.1) can be rearranged in a product of the recursive section $1/A(z)$ and non-recursive section $B(z)$,

$$V(z) = \frac{1}{A(z)} X(z) B(z).$$  \hspace{1cm} (5.3)

Denoting the output of the recursive part with $W(z)$, i.e.,

$$W(z) = \frac{1}{A(z)} X(z)$$  \hspace{1cm} (5.4)

we present relation (5.3) as the output of the nonrecursive part inputted by $W(z)$,

$$V(z) = W(z) B(z).$$  \hspace{1cm} (5.5)

In time domain, input-output relations (5.4) and (5.5) are expressed by the difference equations,

$$w[n] = x[n] - a_1 w[n-1] - a_2 w[n-2] \ldots - a_N w[n-N].$$  \hspace{1cm} (5.6)

$$v[n] = b_0 w[n] + b_1 w[n-1] + b_2 w[n-2] \ldots + b_N w[n-N].$$  \hspace{1cm} (5.7)

The recursive equation (5.6) computes the intermediate sequence $\{w[n]\}$, which is used in the nonrecursive equation (5.7) for computing the output samples $v[n]$.

Figure 5.1. General structure of factor-of-$M$ decimator

![Figure 5.1. General structure of factor-of-$M$ decimator](image-url)
Related Content

Open Source Software Systems: Understanding Bug Prediction and Software Developer Roles
www.irma-international.org/chapter/open-source-software-systems/37047/

Mitigating Type Confusion on Java Card
www.irma-international.org/article/mitigating-type-confusion-java-card/77915/

Security Issues of CPS
www.irma-international.org/chapter/security-issues-of-cps/121254/

Supporting Consistency during the Development and Evolution of Quality Unified Use-Misuse Case Models

Lean, Light, Adaptive, Agile and Appropriate Software Development: The Case for a Less Methodical Methodology
www.irma-international.org/chapter/lean-light-adaptive-agile-appropriate/28109/