Design of Multistage Decimation Filters Using Cyclotomic Polynomials: Optimization and Design Issues

Massimiliano Laddomada, Member, IEEE

Abstract—This paper focuses on the design of multiplier-less decimation filters suitable for oversampled digital signals. The aim is twofold. On one hand, it proposes an optimization framework for the design of constituent decimation filters in a general multistage decimation architecture. The basic building blocks embedded in the proposed filters belong, for a simple reason, to the class of cyclotomic polynomials (CPs): the first 104 CPs have a z-transfer function whose coefficients are simply \{-1,0,1\}. On the other hand, the paper provides a bunch of useful techniques, most of which stemming from some key properties of CPs, for designing the proposed filters in a variety of architectures. Both recursive and non-recursive architectures are discussed by focusing on a specific decimation filter obtained as a result of the optimization algorithm.

Design guidelines are provided with the aim to simplify the design of the constituent decimation filters in the multistage chain.

Index Terms—A/D converter, CIC, cyclotomic, comb, decimation, decimation filter, multistage, polynomial, sigma-delta, sinc filters.

I. INTRODUCTION AND PROBLEM FORMULATION

The design of multistage decimation filters for oversampled signals is a well-known research topic [1]. Mainly inspired by the need of computationally efficient architectures for wide-band, multi-standard, reconfigurable receiver design, this research topic has recently garnered new emphasis in the scientific community [2]–[5]. Multistage decimation filters are also employed for decimating highly oversampled signals from noise-shaping \(\Sigma\Delta\) A/D converters [6].

Given a base-band analog input signal \(x(t)\) with bandwidth \([-B_x, B_x]\), an A/D converter produces a digital signal \(x(nT_s)\) by sampling \(x(t)\) at rate \(f_o = \frac{1}{T_s} = 2\rho B_x \gg 2B_x\), whereby \(\rho \geq 1\) is the oversampling ratio (notice that \(\rho > 1\) for oversampled signals). The normalized maximum frequency contained in the input signal is defined as \(f_c = \frac{f_o}{f_s} = \frac{1}{2\rho}\), and the digital signal \(x(nT_o)\) at the input of the first decimation filter has frequency components belonging to the range \([-f_c, f_c]\). This setup is pictorially depicted in the reference architecture shown in Fig. 1.

Owing to the condition \(\rho \gg 1\), the decimation of an oversampled signal \(x(nT_o)\) is efficiently accomplished by cascading two (or more) decimation stages as highlighted in Fig. 1 in which a multistage architecture composed by \(m\) decimation stages is shown as reference scheme. Consider an oversampling ratio \(\rho\) which can be factorized as follows:

\[ \rho = \prod_{i=1}^{m} D_i \]

whereby, for any \(i\), \(D_i\) is an appropriate integer strictly greater than zero.

In the general architecture shown in Fig. 1, the sampling rate decreases in \(m\) consecutive stages, whereby the sampling rate at the input of the \(i\)th stage is

\[ f_{i-1} = f_i \cdot D_i, \quad \forall i = 1, \ldots, m \]

while the output sample data rate is:

\[ f_i = \frac{f_o}{\prod_{p=1}^{i} D_p}, \quad \forall i = 1, \ldots, m \]

The design of any decimation stage in a multistage architecture imposes stringent constraints on the shape of the frequency response over the so-called folding bands. Considering the scheme in Fig. 1, the frequency response \(H_i(e^{j\omega})\) of the \(i\)th decimation filter must attenuate the quantization noise (QN) falling inside the frequency ranges defined as

\[
\begin{align*}
\left[ \frac{k}{D_i}, \frac{k-1}{D_i} \right], & \quad k = 1, \ldots, k_M, \\
\left[ \frac{k}{D_i}, \frac{k+1}{D_i} \right], & \quad D_i \text{ even} \\
\left[ \frac{k}{D_i}, \frac{k+1}{D_i} \right], & \quad D_i \text{ odd}
\end{align*}
\]

whereby \(f_c^{-1}\) is the normalized signal bandwidth at the input of the \(i\)th decimation filter. The reason is simple: the QN falling inside these frequency bands will fold down to baseband (i.e., inside the useful signal bandwidth \([-f_c^{-1}, f_c^{-1}]\)) because of the sampling rate reduction by \(D_i\) in the \(i\)th decimation stage, irremedially affecting the signal resolution after the multistage decimation chain.

The normalized maximum frequency contained in the input signal is defined as \(f_c = \frac{f_o}{f_s} = \frac{1}{2\rho}\), and the digital signal \(x(nT_o)\) at the input of the first decimation filter has frequency components belonging to the range \([-f_c, f_c]\). This setup is pictorially depicted in the reference architecture shown in Fig. 1.
Fig. 1. General architecture of a \( m \)-stage decimation chain for A/D converters, along with a pictorial representation of the key frequency intervals to be carefully considered for the design of the \( i \)th decimation stage. The sampling rate at the input of the \( i \)th decimation stage is \( f_{i-1} \), \( \forall i = 1, \ldots, m \).

On the other hand, frequency ranges labelled as don’t care bands in Fig. 1 do not require a stringent selectivity since the QN within these bands will be rejected by the subsequent filters in the multistage chain.

The relation between \( f_c^i \) and \( f_c^o \) is as follows:

\[
f_c^i = f_c^{i-1} D_i, \ \forall i = 1, \ldots, m
\]

whereby it is \( f_c^o = 1/2 \rho \).

The \( i \)th decimation filter \( H_i(e^{j\omega}) \) introduces a pass-band ripple \( \delta_p^i \) which can also be expressed in dB as follows

\[
R_p^i = -20 \log_{10} \left( \frac{1 - \delta_p^i}{1 + \delta_p^i} \right) > 0 \tag{2}
\]

while the selectivity (in dB) corresponds to

\[
A_s = 20 \log_{10} \left( \frac{\delta_s}{1 + \delta_p^i} \right) \approx 20 \log_{10} (\delta_s) \ll 0 \tag{3}
\]

With this background, let us provide a quick survey of the recent literature related to the problem addressed here. This survey is by no means exhaustive and is meant to simply provide a sampling of the literature in this fertile area.

Excellent tutorials on the design of multirate filters can be found in \cite{7}, \cite{8}, while an essential book on this topic is \cite{11}. Recently, Coffey \cite{9}, \cite{10} addressed the design of optimized multistage decimation and interpolation filters.

The design of cascade-integrator comb (CIC) filters was first addressed in \cite{11}, while multirate architectures embedding comb filters have been discussed in \cite{12}. Since then, many papers \cite{13} have focused on the computational optimization of CIC filters even in the light of new wide-band and reconfigurable receiver design applications \cite{14}-\cite{16}. Comb filters have been then generalized in \cite{17}-\cite{20}, especially in relation to the decimation of \( \Sigma \Delta \) modulated signals.

Other works somewhat related to the topic addressed in this paper are \cite{21}-\cite{27}. The use of decimation sharpened filters embedding comb filters is addressed in \cite{20,21}, while in \cite{22} authors proposed computational efficient decimation filter architectures using polyphase decomposition of comb filters. Dolecek et al. proposed a novel two-stage sharpened comb decimator in \cite{23}. The design of FIR filters using cyclotomic polynomial (CP) prefilters has been addressed in \cite{24}, while effective algorithms for the design of low-complexity FIR filters embedding CP prefilters have been proposed in \cite{25}-\cite{27}.

Owing to the discussion on the folding bands presented above, this paper addresses the design of computationally efficient decimation filters suitable for oversampled digital signals. Natural eligible blocks used in filter design are cyclotomic polynomials with order less than 105, since these polynomials possess coefficients belonging to the set \( \{-1, 0, +1\} \). We first recall the basic properties of CPs in Section III since these properties suggest useful hints at the basis of the practical implementation of the designed decimation filters. For conciseness, we address the design of the first stage in the multistage architecture, even though the considerations which follow are easily applicable to any other stage in the chain.

The computational complexity of basic CP filters is discussed in Section III. In Section IV we propose an optimization framework whose main aim is to design an optimal decimation filter (optimal in that the cost function to be minimized accounts for the number of additions required by the chosen CP filter) featuring high selectivity within the folding bands seen from the 4th decimation stage.
The practical implementation of the designed decimation filters is addressed in Section VI, whereby both recursive and non-recursive architectures stemming from a variety of properties of polynomials, are discussed. Finally, Section VII draws the conclusions.

II. BASICS OF CYCLOTONIC POLYNOMIALS AND KEY PROPERTIES

Cyclotomic polynomials (CPs) arose hand in hand with the old Greek problem of dividing a circle in equal parts. Key properties of such polynomials along with the basic rationales can be found in various number theory books (we invite the interested readers to refer to [28], [29]), other than in some recent papers [24]. Given an integer $D$ strictly greater than zero, polynomial $(1 - z^{-D})$ can be factorized as a product of cyclotomic polynomials as follows:

$$1 - z^{-D} = \prod_{q|D} C_q(z)$$

whereby $q : q|D$ identifies the set of integers $q$, less than, or equal to $D$, which divides $D$ (in other words, the remainder of the division between $D$ and $q$ is zero).

For each $q$ as above, there is a unique polynomial $C_q(z)$ whose roots satisfy the following conditions.

- For each $q \leq D$, the roots of $C_q(z)$ constitute a subset of the roots belonging to the polynomial $1 - z^{-D}$.
- The roots of $C_q(z)$ are the primitive $q$th roots of unity, i.e., they all fall on the $z$-plane unit circle.
- The number of roots corresponds to the number of positive integers which are prime with respect to $D$, and smaller than $D$.
- Roots of $C_q(z)$ do not belong to the set of roots of the polynomial $1 - z^{-r}$, $0 < r < q \leq D$.

Based on the observations above, polynomials $C_q(z)$ are defined as:

$$C_q(z) = \prod_{i=(1,q)=1}^q \left(1 - z^{-1}e^{-j2\pi i q}\right)$$

whereby $(i, q) = 1$ is used to mean that $i$ and $q$ are coprime [28]. Notice that, given an integer $q$, (5) allows us to write the $z$-transfer function of any CP indexed by $q$.

Key advantages of CPs in connection to filter design rely on the following property: if $q$ has no more than two distinct odd prime factors, polynomials $C_q(z)$ contains coefficients belonging to the set $\{-1, 0, +1\}$. From a practical point of view, CP coefficients belong to the set $\{-1, 0, +1\}$ if $q \leq 104$ [28], [29].

The degree of polynomial $C_q(z)$ is not $q$ but it is defined as follows:

$$\operatorname{deg} [C_q(z)] = \sum_{d|q} d \cdot \mu\left(\frac{q}{d}\right) = \phi(q)$$

whereby $\phi(q)$ is the totient function (see Table I), i.e., the number of positive integers less or equal to $q$ that are relatively prime to $q$, while $\mu(n)$ is the Möbius function defined as:

$$\mu(n) = \begin{cases} 1, & n = 1 \\ (-1)^k, & n = p_1 \cdot p_2 \cdot \ldots \cdot p_k, \\ 0, & \text{if } n \text{ is divisible by the squares of a prime} \\ \text{with } p_i \text{ prime, } p_i \neq p_j, \forall i \neq j \\ \end{cases}$$

Index $k$ in the second entry stands for the number of distinct prime numbers which decomposes the argument $n$. Values of the Möbius function are shown in Table II for $n \in [1, 104]$. Notice that $\mu(n) \neq 0$ implies that $n$ is squarefree, i.e., its decomposition does not contain repeated factors.

The $z$-transfer function of a CP with squarefree index $q$ is [30]:

$$C_q(z) = \sum_{d=0}^{\phi(q)} c_q,d z^{-d(\phi(q)-d)}$$

whereby coefficients $c_q,d$ can be evaluated with the following recursive relation:

$$c_q,d = -\frac{\mu(q)}{d} \sum_{p=0}^{d-1} c_q,p \cdot \mu\left(g(q,d-p)\right) \phi\left(g(q,d-p)\right)$$

1Two numbers are said to be relatively prime if they do not contain any common factor. Notice that the integer 1 is considered as being relatively prime to any integer number.
using the initial value $c_{q,0} = 1$. Function $g(q, d - p)$ in (2) is the greatest common divisor between $q$ and $d - p$. Notice that (9) represents an effective algorithm for automatically generating the $z$-transfer function of CPs with squarefree indexes $q$.

Perhaps, the main properties useful for deducing the $z$-transfer function of any CP, are the ones summarized in the following [28]. We will discuss the application of such properties in Section III whereby the focus is on the design of low complexity CPs in terms of both additions and delays.

1) Given a prime number $t$, it is

$$C_t(z) = \sum_{i=0}^{t-1} z^{-i} = \frac{1 - z^{-t}}{1 - z^{-1}}$$

2) Let $k$, $n$ and $m$ be three positive integers. Then, it is

$$C_{mn^k}(z) = C_{mn}(z^{nk-1})$$

3) Consider a prime number $p$, which does not divide $q$, then

$$C_{pq}(z) = \frac{C_p(z^p)}{C_q(z)}$$

4) Given any odd integer $n$ greater or equal to 3, then it is

$$C_{2n}(z) = C_n(-z)$$

5) For $z = 1$, the following relation holds:

$$C_q(1) = \begin{cases} 0, & q = 1 \\ p, & q = p^k, \ p \ prime \\ 1, & \text{otherwise} \end{cases}$$

This relation assures us that for indexes $q > 1$, $z$-transfer function of the respective CP presents unity gain in baseband provided that $q \neq p^k$. Otherwise, CP transfer functions have to be normalized by $p$ in order to assure unity gain in baseband.

III. CRITERIA FOR IDENTIFYING LOW COMPLEXITY CPs

The $z$-transfer function of CPs for any index $q$ can be deduced upon employing the relation (5) along with the properties stated in (10)-(13). Different architectures (both recursive and non recursive) for implementing each CP can be obtained, mainly differing in the number of additions and delays required. For conciseness, in this paper we show the $z$-transfer functions of the first sixty CPs in Table IV the $z$-transfer functions of $C_q(z)$ for any $q \in \{1, \ldots, 104\}$ in both non recursive and recursive (if any) form can be found in [31].

Let us discuss some key examples by starting from CP $C_{33}(z)$. Considering that 33 is squarefree and given that $p = 33$ can be written as $3 \times 11$, whereby 3 and 11 are coprimes, there are three possible architectures for implementing such a polynomial. The first one stems from (8) and (9) and it consists of a non recursive architecture (see Table IV) employing 14 additions and 20 delays. On the other hand, two recursive architectures follow upon using property (12) with $p = 3, q = 11$ and $p = 11, q = 3$:

$$C_{11,3}(z) = C_{11}((z^3)) = \frac{1 + z^{-33}}{1 + z^{-1}} = \frac{1 + z^{-33} + z^{-34}}{1 + z^{-1} + z^{-22}}$$

$$C_{3,11}(z) = C_{3}((z^{11})) = \frac{1 + z^{-11} + z^{-22}}{1 + z^{-1} + z^{-22}}$$

As far as the number of additions is concerned, from (15) it easily follows that the architecture $C_{3,11}(z)$ only requires 4 additions, which compares favorably with both the non recursive implementation and $C_{11,3}(z)$. Notice also that, since CP coefficients are simply $\{-1, 0, +1\}$, the recursive architectures can be implemented without coefficient quantization; this in turn suggests that exact pole-zero cancellation is not a concern with these architectures.

On the other hand, the non recursive architecture requires only 20 delays as opposed to the recursive architectures requiring, respectively, 34 and 22 delays. In this work, we suppose that the computational complexity of the filter depends only on the number of additions.

Upon comparing for any $q$ both recursive and non recursive architectures in Table IV (see also the complete list of the first CPs reported in [31]), it easily follows that recursive implementations, where do exist, allow the reduction of the number of additions with respect to non recursive implementations; the price to pay, however, relies on the increased filter delay. As a rule of thumb, non recursive architectures should be preferred to recursive implementations when memory space is a design constraint. On the other hand, recursive architectures can greatly reduce the number of additions.

When $q$ is a prime number, the $z$-transfer function of the related CP corresponds to the first order comb filter, as can be straightforwardly seen from (10). Finally, property (13) can be effectively employed for deducing the $z$-transfer function of CPs with even indexes $q$ which can be written as $2n$, with $n$ an odd number strictly greater than 2. As an example,
notice the following relations: \( C_{30}(z) = C_{15}(-z) \), \( C_{34}(z) = C_{17}(-z) \).

The simple examples presented above are by no means a complete picture of the capabilities and sophistication that can be found in multistage structures for sampling rate conversion. They are merely intended to show why such structures can constitute the starting point for obtaining computationally efficient filters for decimating oversampled signals. The design of computationally efficient decimation filters relies on the combination of an appropriate set of CPs. In oversampled A/D converters, for example, it is very important to contain the computational burden of the first stages in the multistage decimation chain. This motivates the study of an effective algorithm for identifying an appropriate set of CPs that, cascaded, is able to attain a set of prescribed requirements as specified in (2) and (3): this is the topic addressed in the next section.

### IV. Optimization Algorithm and Design Examples

This section presents an optimization framework for designing low complexity decimation filters, \( H_i(z) \), as a cascade of CP subfilters. For the derivations which follow, consider the design of the \( i \)th decimation filter in the multistage chain depicted in Fig. [1] with a frequency response that can be represented as follows:

\[
H_i(f_d) = \prod_{q=1}^{\left|S_{c,p}\right|} C_{m+q}^{m+q}(f_d)
\]

whereby \( f_d \) is the digital frequency normalized with respect to the sampling frequency \( f_{s-1} \) as discussed in Section [2]. \( S_{c,p} \) is a suitable set of eligible CPs to be used in the optimization framework \( \left(S_{c,p}\right) \) is the cardinality of the set, i.e., the number of eligible CPs, \( C_{g}(f_d) \) is the frequency response of the CP indexed by \( q \) and \( m_q \) is its integer order in the cascade constituting \( H_i(f_d) \) (it is \( m_q \geq 0 \), \( \forall q \)).

A suitable cost function accounting for the complexity of the \( i \)th decimation filter can be defined as a weighted combination of the number of adders and delays required by the overall filter \( H_i(z) \) [26]:

\[
A_{d} = \max_{f_d \in \left[0; f_{s-1}\right]} 20 \log_{10} \left(|C_{g}(f_d)|_{n}\right)
\]

\[
A_{s}(k, q) = \min_{f_d \in \left[\frac{k}{M} - f_{s-1}; \frac{k}{M} + f_{s-1}\right]} 20 \log_{10} \left(|C_{g}(f_d)|_{n}\right)
\]

whereby \( N_{a,q} \) and \( N_{d,q} \) are known once the set \( S_{c,p} \) of eligible CPs has been appropriately identified. Notice also that \( N_{a,q} \) and \( N_{d,q} \) can be straightforwardly obtained by Table [IV] (see also [31] for a list of all 104 CPs).

Let us address the choice of the eligible CPs in the set \( S_{c,p} \). This is one of the most important design step since the complexity of the optimization framework discussed below, is tied tightly to the number of eligible CPs. By virtue of the discussion on the folding bands spanned by the \( i \)th decimation filter, we choose the eligible CPs between the 104 CPs in such a way that 1) at least 20% of zeros falls within the folding bands defined in (1), 2) no zero falls in the signal passband ranging from 0 to \( f_s \). As a result of extensive tests, we adopted such a threshold which is capable to reject about 20 – 60 initial CPs depending on \( D \). Of course, lower thresholds can increase the number of eligible CPs at the cost of an increased complexity of the optimization framework discussed below. On the other hand, when designing the \( i \)th decimation filter in a multistage architecture, only the so-called folding bands must be spanned by zeros, since don’t care frequency bands will be appropriately spanned by the zeros belonging to the subsequent decimation filters in the cascade.

Before presenting the optimization algorithm, let us discuss the requirements imposed to the frequency response \( H_i(f_d) \) of the \( i \)th decimation filter in the cascade. Mask specifications [1] are given as for classical filters as far as the passband ripple is concerned. In particular, for the optimization algorithm we use the passband ripple expressed in dB as specified in (2). The main difference between the design proposed in this work and classical FIR filter design techniques relies on the fact that in our setup specifications are only imposed in the folding bands [1]. To this end, we evaluated the lowest attenuations (worst-case) attained by each CP belonging to \( S_{c,p} \) in each folding band:

\[
A_{d} = \max_{f_d \in \left[0; f_{s-1}\right]} 20 \log_{10} \left(|C_{g}(f_d)|_{n}\right)
\]

\[
A_{s}(k, q) = \min_{f_d \in \left[\frac{k}{M} - f_{s-1}; \frac{k}{M} + f_{s-1}\right]} 20 \log_{10} \left(|C_{g}(f_d)|_{n}\right)
\]
The optimization problem can be formulated as follows:

\[
\min_{m_1, \ldots, m_{|S_{cp}|}} F \left( m_1, \ldots, m_{|S_{cp}|} \right) \quad |y_0 = 0 \text{ in } (17)
\]

subject to:

\[
\begin{align*}
0) & \quad \sum_{q=1}^{S_{cp}} m_q A_d q \leq R_p \text{ (ripple)} \\
1) & \quad \sum_{q=1}^{S_{cp}} m_q A_s (1, q) \leq A_s \text{ (selectivity)} \\
& \quad \cdots \\
& \quad \sum_{q=1}^{S_{cp}} m_q A_s (k, q) \leq A_s \\
& \quad \cdots \\
& \quad \sum_{q=1}^{S_{cp}} m_q A_s (k_M, q) \leq A_s
\end{align*}
\]

(19)

The optimization problem can be also solved for different prescribed selectivities, \( A_s \) (as specified in (3)), around the various folding bands. In this work we do not pursue this approach. However, notice that such an approach can be effective for noise shaping ΣΔ A/D converters which present an increasing noise power spectra density for higher and higher values of the digital frequency \( f_d \) [6, 13]. Setting increasing values of \( A_s \) in correspondence of successive folding bands can mitigate noise folding due to the decimation process.

The solution to the optimization problem (19) is the set of CP orders \( m = [m_1, \ldots, m_{|S_{cp}|}]^T \), whereby \( m_i = 0 \) signifies the fact that the \( i \)th CP in \( S_{cp} \) is not employed for synthesizing \( H_i(f_d) \).

Upon collecting the set of \( k_M + 1 \) conditions in the matrix \( A \):

\[
A = \begin{pmatrix}
A_{d1} & \cdots & A_{d,|S_{cp}|} \\
A_s(1, 1) & \cdots & A_s(1, |S_{cp}|) \\
\vdots & \vdots & \vdots \\
A_s(k_M, 1) & \cdots & A_s(k_M, |S_{cp}|)
\end{pmatrix}
\]

and the requirements \( b = [R_p, A_s \ldots A_s]^T \), the constraints in (19) can be rewritten as:

\[ A_m \leq b \]

By this setup, the optimization problem in (19) with respect to \( m_1, \ldots, m_{|S_{cp}|} \) can be rewritten as:

\[
\min_{m_1, \ldots, m_{|S_{cp}|}} F \left( m_1, \ldots, m_{|S_{cp}|} \right) \quad |y_0 = 0
\]

subject to:

\[
\begin{align*}
& \quad m_i \geq 0, \quad m_i \text{ integer, } \forall i = 1, \ldots, |S_{cp}|
\end{align*}
\]

and solved by mixed integer linear programming techniques [32]. We solved the optimization problem using the Matlab function linprog along with a new matlab file capable of managing integer constrained solutions (the latter file is available online [33]).

The results of the previous optimization problem are summarized in Table III for various \( A_s \) specifications and two different values of \( R_p \), namely \( R_p = 1 \) and 2 dB. We solved the problem for three different values of the decimation factor \( D \) of the first stage in the decimation chain depicted in Fig. 1 by assuming that the residual decimation factor is \( \nu = 4 \) (in other words, we
assumed that $\rho = D \cdot 4$). Notice that such an approach is quite usual in practice in that the first decimation filter accomplishes the highest possible decimation in order to reduce the sampling rate, while the subsequent decimation stages are usually accomplished with half-band filters each one decimating by 2 [1].

The first row related to any decimation factor shows the set of eligible CPs found in the preliminary design step discussed above, while the $z$-transfer functions of the CPs can be found in Table IV (see also [31] for a list of all 104 CPs).

It is worth comparing the frequency responses of the optimized filters $H_{8,1}(f_d)$ and $H_{16,1}(f_d)$ (for $i = 1, 2, 3$) in Table III with the specifications $R_p = 1$dB and various $A_s$. To this end, Figs. 2 and 3 show, respectively, the behaviours of the frequency responses $H_{8,1}(f_d)$ and $H_{16,1}(f_d)$ along with the imposed selectivity $A_s$ around the various folding bands (identified by horizontal bold lines).

V. IMPLEMENTATION ISSUES

This section addresses the design of optimized CP-based decimation filters. For conciseness, we will focus on the design of decimation filter $H_{8,2}(z)$ shown in Table III even though the considerations which follow can be applied to any other decimation filter quite straightforwardly. The decimation stage related to $H_{8,2}(z)$ is depicted in Fig. 4a: this decimation filter will be designed through a variety of architectures following from different mathematical ways to simplifies the analytical relation defining $H_{8,2}(z)$.

First of all, notice that upon substituting the appropriate equations of the constituent CP filters in $H_{8,2}(z)$, the designed filter takes on the following expression:

$$H_{8,2}(z) = C_2^2(z)C_3^2(z)C_8^3(z) = (1 + z^{-1})^2 (1 + z^{-2})^3 (1 + z^{-4})^3$$

which can be rewritten as follows:

$$H_{8,2}(z) = \prod_{i=0}^{2} (1 + z^{-2^i})^3$$

From the commutative property employed in [12], the cascaded implementation shown in Fig. 4b easily follows. The $r$th stage in Fig. 4b operates at the sampling rate $f_{i-1}/2^r$, whereby $f_{i-1}$ is the data sampling frequency at the filter input as shown in the multistage architecture in Fig. 1. Further power consumption reduction can be achieved by applying polyphase decomposition to the architecture shown in Fig. 4b. To this aim, consider the $z$-transfer function of the 3rd order cell:

$$(1 + z^{-1})^3 = 1 + 3z^{-2} + z^{-1} (3 + z^{-2}) = E_0(z^2) + z^{-1}E_1(z^2)$$

$$E_0(z) = 1 + 3z^{-1}$$

$$E_1(z) = 3 + z^{-1}$$

The polyphase architecture for $(1 + z^{-1})^3$ easily follows from the commutative property applied to the two filters $E_0(z^2)$ and $E_1(z^2)$ in (22), and it is shown in Fig. 4c along with the architectures for implementing both $E_0(z)$ and $E_1(z)$. Notice that the multipliers appearing in $E_0(z)$ and $E_1(z)$ can be implemented in the form of shift registers as depicted in Fig. 4d.

The actual complexity of the architecture shown in Fig. 4b is fully defined once the data wordlength in any substage is well characterized, since the power consumption of a filter cell can be approximated as the product between the data rate, the number of additions
performed at that rate, and the data wordlength. While the data rate along with the number of additions are well defined, data wordlength in each substage in Fig. 4b is not. Given the input data wordlength, \( R \) (in bits), the data size at the output of the first decimation substage in Fig. 4b is equal to \( R + 2 \) bits since two carry bits have to be allocated for the two additions involved in that substage. With a similar reasoning, data wordlength increases at the output of each subsequent substage in Fig. 4b in order to take into account the increase of data size due to the involved additions.

As a reference example, if the decimation filter depicted in Fig. 4b is the first decimation stage at the output of a \( \Sigma \Delta \) A/D converter embedding a 1-bit quantizer into the loop, it is \( R = 1 \). Thus, data wordlength is as low as 3 bits after the first decimation substage, and so on.

Let us address the design of a recursive architecture for \( H_{8,2}(z) \) in (20). First of all, consider the following equality chain

\[
\log_2(D-1) = \sum_{i=0}^{D-1} \left(1 + z^{-2^i}\right) = \left[1 - z^{-2^D-1}\right] \quad \text{(23)}
\]

whereby the first equality holds for any \( D \) that can be written as an integer power of 2, i.e., \( D = 2^n \). On the other hand, the last equality holds for any integer value of \( D \). Notice that decimation factors of the form \( 2^n \) are quite common in practice. Upon using (23) with \( t = 3 \) and \( D = 2^3 \), (20) can be rewritten as follows:

\[
H_{8,2}(z) = \prod_{i=0}^{2} \left(1 + z^{-2^i}\right) = \left[1 - z^{-2^D-1}\right] \quad \text{(24)}
\]

The last relation in (24) can be simplified as follows:

\[
\begin{align*}
(1 - z^{-8})^3 &= \frac{(1 - z^{-8})^3}{(1 - z^{-1})^3(1 + z^{-1})} \\
&= \frac{(1 - z^{-8})^3}{1 - 2z^{-1} + 2z^{-3} - z^{-4}} \quad \text{(25)}
\end{align*}
\]

A recursive implementation of filter \( H_{8,2}(z) \) in (25) is shown in Fig. 4d. It is obtained in the same way as for a classic cascade integrator-comb (CIC) implementation. In other words, the numerator in (25) corresponds to the comb sections at the right of the decimator by \( D \), while the denominator is responsible for the integrator sections at the left of the decimator by \( D = 8 \).

The derivations yielding (25) upon starting from (24) can also be accomplished by following another reasoning based on the following relation:

\[
1 + z^{-n} = \frac{1 - z^{-2n}}{1 - z^{-n}} \quad \text{(26)}
\]

\(^2\)Notice that \((1 - z^{-8})^3\) becomes \((1 - z^{-1})^3\) upon its shifting through the decimator by \( D = 8 \).

\(^3\)We discuss this other approach for completeness, since it can be effective for deriving an appropriate architecture for other decimation filter shown in Table III.
An efficient architecture for implementing each polyphase component $E_i(z)$ stems from the decomposition of each integer as the summation of power-of-two coefficients as shown in (30) for the first two polyphase components $E_0(z)$ and $E_1(z)$. By doing so, and employing coefficient sharing arguments, practical architectures featuring a minimum number of shift registers easily follow as depicted in Fig. 5. Similar considerations can be employed for obtaining the architectures of the remaining polyphase components $E_2(z), \ldots, E_7(z)$.

VI. Conclusions

This paper addressed the design of multiplier-less decimation filters suitable for oversampled digital signals. The aim was twofold. On one hand, it proposed an optimization framework for the design of constituent decimation filters in a general multistage decimation architecture using as basic building blocks cyclotomic polynomials (CPs), since the first 104 CPs have simple coefficients ($\{-1, 0, +1\}$). On the other hand, the paper provided a bunch of useful techniques, most of which stemming from some key properties of CPs, for designing the optimized filters in a variety of architectures. Both recursive and non-recursive architectures have been discussed by focusing on a specific decimation filter obtained as a result of the optimization algorithm. Design guidelines were provided with the aim to simplify the design of the constituent decimation filters in the multistage chain.

REFERENCES

[1] R. E. Crochiere and L. R. Rabiner, Multirate Digital Signal Processing, Prentice-Hall PTR, 1983.
[2] J. Mitola, “The software radio architecture,” IEEE Comm. Magazine, Vol.33, No.5, pp. 26-38, May 1995.
[3] M. Laddomada, F. Daneshgaran, M. Mondin, and R.M. Hickling, “A PC-based software receiver using a novel front-end technology,” IEEE Comm. Magazine, Vol.39, No.8, pp.136-145, Aug. 2001.
[4] F. Daneshgaran and M. Laddomada, “Transceiver front-end technology for software radio implementation of wideband satellite communication systems,” Wireless Personal Communications, Kluwer, Vol.24, No.12, pp. 99-121, December 2002.
[5] A.A. Abidi, “The path to the software-defined radio receiver,” IEEE Journal of Solid-State Circuits, Vol.42, No.5, pp. 954-966, May 2007.
[6] S. R. Norsworthy, R. Schreier, and G. C. Temes, Delta-Sigma Data Converters, Theory, Design, and Simulation, IEEE Press, 1997.
[7] R. E. Crochiere and L. R. Rabiner, “Interpolation and decimation of digital signals,” A tutorial review,” Proceedings of the IEEE, Vol.69, No.3, pp. 300-331, March 1981.
[8] P.P. Vaidyanathan, “Multirate digital filters, filter banks, polyphase networks, and applications: a tutorial,” Proceedings of the IEEE, Vol.78, No.1, pp. 56-93, Jan. 1990.
[9] M.W. Coffey, “Optimizing multistage decimation and interpolation processing-Part I,” IEEE Signal Proc. Letters, Vol.10, No.4, pp. 107-110, April 2003.
[10] M.W. Coffey, “Optimizing multistage decimation and interpolation processing-Part II,” IEEE Signal Proc. Letters, Vol.14, No.1, pp. 24-26, Jan. 2007.
[11] E. B. Hogenauer, “An economical class of digital filters for decimation and interpolation,” IEEE Trans. on Ac., Speech and Sign. Proc., vol. ASSP-29, pp. 155-162, No. 2, April 1981.
[12] S. Chu and C. S. Burrus, “Multirate filter designs using comb filters,” IEEE Trans. on Circuits and Systems, vol. CAS-31, pp. 913924, Nov. 1984.
[13] R.A. Losada and R. Lyons, “Reducing CIC filter complexity,” IEEE Signal Proc. Mag., Vol.23, No.4, pp. 124-126, July 2006.
[14] Y. Gao, J. Tenhunen, and H. Tenhunen, “A fifth-order comb decimation filter for multi-standard transceiver applications,” In Proceedings of ISCAS 2000, May 28-31, 2000, Geneva, Switzerland, pp. III-89-III-92.
[15] F.J.A. de Aquino, C.A.F. da Rocha, and L.S. Resende, “Design of CIC filters for software radio system,” Proceedings of IEEE ICASSP 2006, Vol.3, 2006.
[16] T. Ze and S. Signell, “Multi-standard delta-sigma decimation filter design,” Proceedings of IEEE APCCAS 2006, pp. 1212-1215, 4-7 Dec. 2006.
[17] L. Lo Presti, “Efficient modified-sinc filters for sigma-delta A/D converters,” IEEE Trans. on Circ. and Syst.-II, Vol. 47, pp. 1204-1213, No. 11, November 2000.
[18] M. Laddomada, “Generalized comb decimation filters for ΣΔ A/D converters: Analysis and design,” IEEE Trans. on Circuits and Systems I, Vol.54, No. 5, pp. 994-1005, May 2007.
[19] M. Laddomada and M. Mondin, “Decimation schemes for ΣΔ A/D converters based on Kaiser and Hamming sharpened filters,” IEE Proceedings of Vision, Image and Signal Processing, Vol. 151, No. 4, pp. 287-296, August 2004.
[20] M. Laddomada, “Comb-based decimation filters for ΣΔ A/D converters: Novel schemes and comparisons,” IEEE Trans. on Signal Processing, Vol.55, No. 5, Part 1, pp. 1769-1779, May 2007.
[21] A.Y. Kwentus, Z. Jiang, and A.N. Willson Jr., “Application of filter sharpening to cascaded integrator-comb decimation filters,” IEEE Trans. on Signal Proc., Vol. 45, pp. 457-467, No. 2, February 1997.
[22] H. Abouzahy, Y. Dumonteix, M. Louîrat, and H. Mehrez, “Efficient polyphase decomposition of comb decimation filters in ΣΔ analog-to-digital converters,” IEEE Trans. on Circ. and Syst.-II, Vol. 48, pp. 898-903, No. 10, October 2001.
[23] G. Jovanovic-Dolecek and S.K. Mitra, “A new two-stage sharpened comb decimator,” IEEE Trans. on Circ. and Syst.-I, Vol. 52, pp. 1414-1420, No. 7, July 2005.
[24] R.J. Hartnett and G.F. Boudreaux-Bartels, “On the use of cyclotomic polynomial prefilters for efficient FIR filter design,” IEEE Trans. on Signal Processing, Vol.41, No.5, pp. 1766-1779, May 1993.
[25] H.J. Oh and Y.H. Lee, “Design of efficient FIR filters with cyclotomic polynomial prefilters using mixed integer linear programming,” IEEE Signal Proc. Letters, Vol.3, No.8, pp. 239-241, Aug. 1996.
[26] H.J. Oh and Y.H. Lee, “Design of discrete coefficient FIR and IIR digital filters with prefilter-equalizer structure using linear programming,” IEEE Trans. on Circuits and Systems II, Vol.47, No.6, pp. 562-565, June 2000.
[27] K. Supramaniam and Yong Lian, “Complexity reduction for frequency-response masking filters using cyclotomic polynomial prefilters,” Proceedings of IEEE ISCAS 2006, 21-24 May 2006.
[28] M.R. Schroeder, Number Theory in Science and Communication: With Applications in Cryptography, Physics, Digital Information, Computing, and Self-Similarity, Springer-Verlag, 3rd ed., 1997.
[29] J.H. McClellan and C.M. Rader, Number Theory in Digital Signal Processing, Prentice-Hall, 1979.
[30] http://mathworld.wolfram.com/CyclotomicPolynomial.html
[31] M. Laddomada, “Some Properties along with the z-transfer functions of the first 104 Cyclotomic Polynomials,” Internal report, available at http://www.tlc.polito.it/dcc_team/research.php?id=4
[32] C.H. Papadimitriou and K. Steiglitz, Combinatorial Optimization, Algorithms and Complexity, Dover Publications, Mineola, New York, 1998.
[33] Mixed Integer Linear Programming Matlab file, available at http://www.mathworks.com/matlabcentral/fileexchange/loadFile.do?objectId=6990&objectType=file
[34] A. Antoniou, Digital Signal Processing: Signals, Systems, and Filters, McGraw-Hill, 2005, ISBN 0-07-145425-X.
TABLE IV

THE FIRST SIXTY CYCLOTOmic POLYNOMIALS.

| $q$ | $C_q(z^{-1})$ | $q$ | $C_q(z^{-1})$ | $q$ | $C_q(z^{-1})$ |
|-----|----------------|-----|----------------|-----|----------------|
| 1   | $1 - z^{-1}$   | 11  | $\sum_{i=0}^{10} z^{-i} = \frac{1 + z^{-11}}{1 - z^{-1}}$ | 21  | $1 - z^{-1} + z^{-3} - z^{-4} + z^{-6}$ |
| 2   | $1 + z^{-1}$   | 12  | $1 - z^{-2} + z^{-4} = \frac{1 + z^{-6}}{1 + z^{-2}}$ | 22  | $1 - z^{-8} + z^{-9} - z^{-11} + z^{-12}$ |
| 3   | $1 + z^{-1} + z^{-2} = \frac{1 + z^{-3}}{1 + z^{-1}}$ | 13  | $\sum_{i=0}^{12} z^{-i} = \frac{1 + z^{-15}}{1 - z^{-1}}$ | 23  | $\frac{1 + z^{-14} + z^{-2}}{1 - z^{-2}}$ |
| 4   | $1 + z^{-2}$   | 14  | $\sum_{i=0}^{6} (-1)^i z^{-i} = \frac{1 + z^{-7}}{1 + z^{-1}}$ | 24  | $1 - z^{-4} + z^{-8} = \frac{1 + z^{-12}}{1 + z^{-2}}$ |
| 5   | $\frac{1 + z^{-5}}{1 + z^{-1}}$ | 15  | $1 - z^{-1} + z^{-3} - z^{-4} + z^{-5}$ | 25  | $\sum_{i=0}^{4} z^{-5i} = \frac{1 + z^{-20}}{1 + z^{-4}}$ |
| 6   | $1 - z^{-1} + z^{-2} = \frac{1 + z^{-3}}{1 + z^{-1}}$ | 16  | $1 + z^{-8}$ | 26  | $\sum_{i=0}^{12} (-1)^i z^{-i} = \frac{1 + z^{-13}}{1 + z^{-2}}$ |
| 7   | $\frac{1 + z^{-7}}{1 + z^{-2}}$ | 17  | $\sum_{i=0}^{16} z^{-i} = \frac{1 + z^{-17}}{1 + z^{-1}}$ | 27  | $1 + z^{-9} + z^{-18} = \frac{1 + z^{-27}}{1 + z^{-1}}$ |
| 8   | $1 + z^{-4}$   | 18  | $1 - z^{-3} + z^{-6} = \frac{1 + z^{-9}}{1 + z^{-3}}$ | 28  | $\sum_{i=0}^{6} (-1)^i z^{-2i} = \frac{1 + z^{-14}}{1 + z^{-3}}$ |
| 9   | $1 + z^{-3} + z^{-6} = \frac{1 + z^{-9}}{1 + z^{-3}}$ | 19  | $\sum_{i=0}^{18} z^{-i} = \frac{1 + z^{-19}}{1 + z^{-3}}$ | 29  | $\sum_{i=0}^{24} (-1)^i z^{-2i} = \frac{1 + z^{-26}}{1 + z^{-3}}$ |
| 10  | $\sum_{i=0}^{4} (-1)^i z^{-i} = \frac{1 + z^{-5}}{1 + z^{-1}}$ | 20  | $1 + z^{-2} + z^{-4} - z^{-6} - z^{-8}$ | 30  | $1 + z^{-1} - z^{-3} - z^{-4} - z^{-5}$ |
| 31  | $\frac{1 + z^{-31}}{1 + z^{-1}}$ | 41  | $\frac{1 + z^{-2}}{1 + z^{-1}}$ | 51  | $1 - z^{-1} + z^{-3} - z^{-4} + z^{-6} - z^{-7}$ |
| 32  | $1 + z^{-16}$  | 42  | $1 + z^{-1} - z^{-3} - z^{-4} + z^{-6}$ | 52  | $\sum_{i=0}^{12} (-1)^i z^{-2i} = \frac{1 + z^{-26}}{1 + z^{-3}}$ |
| 33  | $1 - z^{-1} + z^{-3} - z^{-4} + z^{-6}$ | 43  | $\frac{1 + z^{-1}}{1 + z^{-1}}$ | 53  | $\frac{1 + z^{-53}}{1 + z^{-2}}$ |
| 34  | $\sum_{i=0}^{16} (-1)^i z^{-i} = \frac{1 + z^{-17}}{1 + z^{-3}}$ | 44  | $\sum_{i=0}^{10} (-1)^i z^{-2i} = \frac{1 + z^{-12}}{1 + z^{-2}}$ | 54  | $1 - z^{-9} + z^{-18}$ |
| 35  | $1 - z^{-1} + z^{-5} - z^{-6} - z^{-7}$ | 45  | $1 - z^{-3} + z^{-6} = \frac{1 + z^{-9}}{1 + z^{-3}}$ | 55  | $1 - z^{-1} - z^{-5} - z^{-6} + z^{-10} - z^{-12}$ |
| 36  | $1 - z^{-8} - z^{-9} - z^{-11} + z^{-12}$ | 46  | $\sum_{i=0}^{22} (-1)^i z^{-i} = \frac{1 + z^{-23}}{1 + z^{-4}}$ | 56  | $\sum_{i=0}^{6} (-1)^i z^{-4i} = \frac{1 + z^{-28}}{1 + z^{-4}}$ |
| 37  | $\frac{1 + z^{-37}}{1 + z^{-1}}$ | 47  | $\sum_{i=0}^{20} (-1)^i z^{-i} = \frac{1 + z^{-20}}{1 + z^{-4}}$ | 57  | $1 - z^{-1} - z^{-3} - z^{-4} - z^{-6} - z^{-7} - z^{-9} - z^{-10} + z^{-12} - z^{-13} + z^{-15}$ |
| 38  | $\sum_{i=0}^{18} (-1)^i z^{-i} = \frac{1 + z^{-19}}{1 + z^{-1}}$ | 48  | $1 - z^{-8} + z^{-16}$ | 58  | $\sum_{i=0}^{28} (-1)^i z^{-2i} = \frac{1 + z^{-29}}{1 + z^{-3}}$ |
| 39  | $1 - z^{-1} + z^{-3} - z^{-4} + z^{-6}$ | 49  | $\sum_{i=0}^{6} z^{-7i} = \frac{1 + z^{-49}}{1 + z^{-1}}$ | 59  | $\sum_{i=0}^{28} (-1)^i z^{-4i} = \frac{1 + z^{-29}}{1 + z^{-1}}$ |
| 40  | $\sum_{i=0}^{4} (-1)^i z^{-5i} = \frac{1 + z^{-20}}{1 + z^{-4}}$ | 50  | $\sum_{i=0}^{4} (-1)^i z^{-5i} = \frac{1 + z^{-25}}{1 + z^{-4}}$ | 60  | $1 + z^{-2} - z^{-6} - z^{-8} - z^{-10}$ |