Linear phase approximation of real and complex pole IIR filters using MFIR structures

Vandenbussche Jean-Jacques\(^{(1,2)}\), Lee Peter\(^{(2)}\), Peuteman Joan\(^{(1,3)}\)

22-23 March 2012

\(^{(1)}\) KHBO, Departement Industriële Wetenschappen en Technologie, Zeeheldenkaai 101, 8400 OOSTENDE, Belgium (jeanjacques.vandenbussche@khbo.be and joan.peuteman@khbo.be)

\(^{(2)}\) University of Kent, Faculty of Sciences, School of Engineering and Digital Arts, Canterbury, Kent CT2 7NZ, UK (P.Lee@kent.ac.uk)

\(^{(3)}\) Katholieke Universiteit Leuven, Department Electrical Engineering (ESAT-ELECTA) Kasteelpark Arenberg 10, 3000 Leuven, Belgium

Abstract

This paper discusses the approximation of real and complex pole IIR filters using MFIR structures. MFIR structures are not new but up till recently they were seldom used due their hardware impact. It is shown in this paper that their hardware impact is very acceptable when using state of the art FPGA’s. The MFIR structures fit even very well on FPGA architectures. Moreover they are also ideally suited for the approximation of the magnitude response of IIR filters when linear phase behavior is required.

Keywords: MFIR filter, linear phase filter, pole approximation.

1 Introduction

In the past two decades, there have been two major trends in the wireless communication world. The first trend was to use more and more the phase of the RF carrier to carry the information of the baseband signal. This resulted in a growing popularity of modulation schemes as (D)QPSK, QAM, MSK etc. The second trend was the shift of the digital signal processing part, in the receiver, towards the frontend. Both trends have increased the importance of linear phase digital filters with narrow transition bands. This paper proposes the Multiplicative
Finite Impulse Response (MFIR) filter structure as a hardware efficient method for the implementation of demanding linear phase digital filters in Field Programmable Gate Arrays (FPGA).

MFIR filters are a class of filter structures introduced in the early 1980’s [1]. MFIRs can be used to replace recursive IIR filters with FIR equivalents requiring significantly fewer hardware than classical FIR architectures that fulfill the same specifications [2]. These MFIR filters are able to realize low pass filters, band pass filters, high pass filters and notch filters.

The replacement of a pole that is implemented in a recursive structure, by a non recursive FIR filter, has the advantage that it will always be stable. This is particularly interesting when the original pole is situated close to the unit circle. However, classical FIR pole replacement architectures require a lot of multipliers, adders and delay elements. Although the MFIR structures require approximately the same number of delay elements as the classical FIR implementations, they require logarithmically fewer adders and multipliers [1]. The MFIR structures can also be used to approximate the magnitude response of a pole filter while obtaining linear phase.

Despite the advantages, MFIR structures have not been popular since their introduction. The large number of delay elements required to approximate the behavior of an IIR filter was considered prohibitively expensive, making them impractical for hardware implementation.

Recent advances in VLSI technology in general and in FPGA architectures in particular, make it necessary to re-evaluate these assumptions. Modern FPGA fabrics now contain a large number of distributed memory elements in combination with a range of hardwired multipliers. This makes them an ideal target platform for implementing efficient MFIR filters that are competitive to standard FIR and IIR equivalents implemented on the same fabric.

The present paper focuses on the implementation of linear phase approximations of IIR filters having a real pole $\lambda = r$ or a conjugate pole pair $\lambda = r e^{j\phi}$ and $\lambda^* = r e^{-j\phi}$. The hardware requirements and some optimizations, when implemented in a state of the art FPGA, are considered.

Section 2 discusses the basic MFIR approximations from a mathematical point of view. In section 3, the actual implementation of real and complex pole approximations in FPGA’s is clarified. The impact on the hardware when very demanding filters are implemented is also taken into consideration. Finally a conclusion is drawn.

2 Mathematical background

2.1 The approximation principle

MFIR filters are based on two basic expressions. The first expression is the geometric series expansion given by
\[ \sum_{i=0}^{\infty} q^i = \frac{1}{1-q} \quad \text{for} \ |q| < 1. \tag{2.1} \]

For \(|q| \geq 1\) the series diverges and consequently (2.1) does not hold. When \(q = -1\), the sum of the series is undetermined. The second basic expression is given by [3]

\[ \sum_{i=0}^{\infty} x^i = \prod_{i=0}^{\infty} (1 + x^2) \tag{2.2} \]

which also only applies when the sum converges. When considering a finite sum (2.2) is replaced by

\[ \sum_{i=0}^{P-1} x^i = \prod_{i=0}^{P-1} (1 + x^2). \tag{2.3} \]

In case \(H(z)\) is the transfer function of a real pole IIR filter, the combination of (2.1) and (2.3) yields

\[ H(z) = \frac{1}{1 - \lambda z^{-1}} = \sum_{i=0}^{\infty} (\lambda z^{-1})^i \]

\[ = \prod_{i=0}^{P-1} (1 + (\lambda z^{-1})^2) = \prod_{i=0}^{P-1} M_i(z) = M(z). \tag{2.4} \]

With the obvious definitions for \(M_i(z)\) and \(M(z)\). In (2.4) a real pole at \(\lambda\) is replaced by \(2^P-1\) zeroes in the z-plane, equidistantly spread over a circle with radius \(|\lambda|\) and angles \(i \cdot \frac{2\pi}{2^P} \quad i = 1, 2, \ldots, (2^P - 1)\). Notice a zero is lacking at the place of the pole that is approximated. It is important to remark that (2.4) only holds when \(|\lambda| < 1\) which means the unit circle is in the region of convergence.

An IIR filter with a transfer function \(H(z)\) having a conjugate pole pair \(\lambda\) and \(\lambda^*\) where \(\lambda = re^{j\phi}, \lambda^* = re^{-j\phi}\) and \(|r| < 1\), can be approximated with a cascade MFIR structure [1].
\[ H(z) = \frac{1}{(1-\lambda z^{-1})(1-\lambda^* z^{-1})} \]
\[ \approx \prod_{n=0}^{P-1} \left( 1 + \left( \lambda z^{-1} \right)^n \right) \prod_{n=0}^{P-1} \left( 1 + \left( \lambda^* z^{-1} \right)^n \right) \]
\[ = \prod_{n=0}^{P-1} \left( 1 + 2\left| rz^{-1}\right|^n \cos(2^n \theta) + \left| rz^{-1}\right|^{2n} \right) = \prod_{n=0}^{P-1} M_n(z) = M(z). \] (2.5)

Also here, (2.5) only applies when \(|\lambda z^{-1}|\) and \(|\lambda^* z^{-1}|\) are < 1 i.e. the unit circle must be in the region of convergence.

When considering the real pole approximation of (2.4), an estimate of the required \(P\) to have an approximation that does not deviate by more than 0.01 dB in the magnitude response is given by [2]
\[ P > \left\lceil \frac{\log \left( \frac{0.00115}{\log |\lambda|} \right)}{\log 2} \right\rceil. \] (2.6)

Where \(\lceil . \rceil\) denotes “rounded to the next higher integer”. Substituting \(|\lambda|\) in (2.6) with some realistic values (\(|\lambda| < 1\)) shows that \(P\) remains relatively small. E.g. for \(\lambda = 0.99\) the minimum required \(P\) is only 10. Every stage of the MFIR \(M(z)\) requires 2 (for a real pole approximation) or 3 (for a complex pole approximation) multipliers resulting in a total of 20 or 30 multipliers in this extreme example. This is an acceptable number when the MFIR is implemented in a state of the art FPGA. The required number of delay cells equals \(2^P-1\) for the approximation of a real pole and \(2^{(P+1)-2}\) for the approximation of a complex conjugate pole pair. These are quite large numbers, but in an FPGA implementation, the delays can be implemented in the distributed RAM. I.e. the look-up tables of the slices which are abundantly present in FPGA’s.

Taking the number of delays and the number of multipliers into account, it is clear that the MFIR approximation is interesting (from a resource requirement point of view) for the approximation of poles close to the unit circle to avoid stability problems. Using MFIR filters to approximate any pole that can be implemented in a stable manner with a classical IIR structure is not that useful.

2.2 Linear phase MFIR filters

The MFIR approximation can also be used to approximate the magnitude behavior of a pole filter (with a pole inside the unit circle) while obtaining linear phase. Ref. [1] proposes the following procedure. If the desired magnitude response is given by \(|H(z)|\):
1. Design an elliptic IIR filter having stable pole(pairs) and realizing \(|H(z)|^{1/2}\).
2. Approximate all poles of this IIR filter using (2.4) and/ or (2.5)
3. Cascade to every zero of the obtained MFIR structure(s) the reciprocal zero with respect to the unit circle. Remark that this procedure must be followed for every pole(pair), not only the possible unstable pairs.
4. The resulting structure must be cascaded with the zeros of the elliptic IIR filter (that realizes $|H(z)|^{1/2}$) each doubled.

Concentrating the discussion on one single pole(pair), [1] suggests that a linear phase approximation of a pole with $|\lambda| < 1$ can be viewed as an approximation of

$$
\bar{H}(z) = \frac{f}{(1-\lambda z^{-1}) (\lambda - z^{-1})}.
$$

(2.7)

Here the dash above $H$ indicates the linear phase behavior and $f$ and $g$ are constant factors. Both IIR’s in (2.7) can be approximated with an MFIR according to [1]. However, [4] argues that the second factor in (2.7): $g/(\lambda - z^{-1})$ does not include the unit circle in the region of convergence and consequently (2.4) must not be used. This remark is correct since

$$
\sum_{i=0}^{\infty} (\lambda z)^i = \frac{1}{\lambda - z^{-1}}
$$

(2.8)

In case $|\lambda| < 1$ and $z$ belongs to the unit circle, the sum in (2.8) does not converge. An equality is only obtained in case $(\lambda z)^i$ is smaller than 1 requiring a $|\lambda| > 1$ when $z$ belongs to the unit circle. However $\lambda$ must always be smaller than 1 to keep $\sum_{i=0}^{\infty} (\lambda z)^i$ convergent. Consequently $(\lambda z)^i < 1$ can never be fulfilled for $z$ on the unit circle. Therefore [4] proposes the following approximation

$$
\frac{1}{\lambda - z^{-1}} = -z \sum_{i=0}^{\infty} (\lambda z)^i = -z \prod_{i=0}^{\infty} (1 + (\lambda z)^i) = -z \prod_{i=0}^{\infty} \left( z^{-\lambda} + \lambda^z \right).
$$

(2.9)

Unfortunately (2.9) is non-causal and must be made causal by discarding the anticipation factor $z^\lambda$.

The approximation proposed in this paper does not use (2.7) nor (2.8), but it still follows the four step procedure of [1] formulated above. If the IIR filter that realizes $|H(z)|^{1/2}$ is indicated by $H'(z)$, the stable poles of $H'(z)$ (indicated by $\lambda'$) can be approximated using (2.4) or (2.5) without any convergence problems. This yields an always stable MFIR filter $M'(z)$ with zeros inside the unit circle on a circle with radius $|\lambda'|$ as shown in Figure 1.

In order to obtain a linear phase filter, $M'(z)$ must be cascaded with a second MFIR $M''(z)$ realizing the reciprocals of the zeros of $M'(z)$. $M''(z)$ will have its zeros on a circle with radius $1/|\lambda'|$. As shown in [5], $M'(z)$ and $M''(z)$ will have the same magnitude response and consequently the cascade of $M'(z)$ and $M''(z)$ will have the magnitude response $|H(z)|$ while obtaining a linear phase behavior.

This method will never have any stability problems since it starts from the always stable MFIR filter with its zeros inside the unit circle and not from the possible unstable IIR filter which could have a divergent geometric series approximation.

Consequently, the linear phase approximation of an IIR filter $H(z)$ with a real stable pole at $\lambda$, can be designed by determining the real (stable) pole $\lambda'$ realizing $|H'(z)| = |H(z)|^{1/2}$. The resulting $\bar{H}(z)$ is then given by
\[
\bar{H}(z) = \prod_{i=0}^{P-1} \left(1 + \left(\lambda' z^{-1}\right)^2 \right) \left(1 + \left(\frac{z^{-1}}{\lambda'}\right)^2\right)
\]

(2.10)

\[
\bar{H}(z) = \prod_{i=0}^{P-1} \left(1 + \left(\lambda'\right)^2 + \frac{1}{(\lambda')^2}\right) z^{-2i} + z^{-2i}
\]

For a \( H(z) \) realizing an IIR filter with a complex conjugate pole pair \( \lambda = re^{j\theta} \), the (stable) poles \( \lambda' = r'e^{j\theta'} \) of \(|H'(z)| = |H(z)|^{1/2}\) will be determined. The MFIR approximation yielding the linear phase \( \bar{H}(z) \) filter is then given by

\[
\bar{H}(z) = \prod_{i=0}^{P-1} \left(1 + 2(r'z^{-1})^2 \cos(2\theta') + (r'z^{-1})^2 + 2\left(\frac{1}{r'}z^{-1}\right)\cos(2\theta') + \left(\frac{1}{r'}z^{-1}\right)^2\right)
\]

(2.11)

\[
\bar{H}(z) = \prod_{i=0}^{P-1} \left[1 + \left(2\left(r'\right)^2 + 4\cos^2(2\theta') + \left(r'\right)^2\right)z^{-2i}\right]
\]

Figure 2: z-plane of an MFIR filter approximating a complex pole pair \( \lambda' = 0.9 e^{j0.1} \) with \( P = 6 \) and linear phase behavior

Figure 2 shows the z-plane of a linear phase MFIR filter approximating the magnitude response of a filter with a complex pole pair \( \lambda' = 0.9 e^{j0.1} \) with \( P = 6 \). In (2.10) and (2.11), all stages used in the MFIR cascade are each linear phase FIR filters with an odd number of symmetric coefficients. Consequently, the cascade of these stages will always realize a linear phase filter. Remark that an odd number of
symmetric coefficients allows the realization of any type of linear phase filter (low pass, high pass, band pass or band reject).
When comparing (2.4) with (2.10) and (2.5) with (2.11), it is clear that the linear phase approximations do not require more multipliers than their non-linear phase counterparts. However, the linear phase approximations require twice the number of delay elements when compared with their non-linear phase counterparts.

3 The implementation
The general RTL (Register Transfer Level) implementations of MFIR structures shown here are vendor independent. However, major FPGA vendors supply DSP optimized blocks which offer the possibility to implement the MFIR structure more efficiently. Therefore this paper presents both the general implementations and the Xilinx DSP48A optimized implementations. All reference designs have been implemented in a Xilinx Spartan3A-DSP: XC3SD1800A.

3.1 The approximation of a real pole
Using (2.10) a linear phase filter can be realized. The architecture of a general stage $i$ is shown in Figure 3. In order to avoid overflow, each stage $i$ is scaled with a scaling factor. The type of scaling can be absolute bound, infinity bound or L2 bound as defined in [6]. In Figure 3 the multiplier indicated by $S_i$ is the scaled multiplier of the first and last term in (2.10). The $S_{bi}$ multiplier is the scaled

$$\left(\lambda^2 + \frac{1}{(\lambda')^2}\right)$$

factor in (2.10).

The implementation of a real pole $\lambda'$ at 0.99 with 9 stages and 17 bits wide data input uses 1632 slices which is only 9% of the total available resources of this mid-range FPGA. The estimated maximum frequency after synthesis equals 164 MHz in speed grade -4. Figure 4 shows the magnitude and phase response of the entire filter without scaling.

Figure 3. MFIR filter with linear phase approximating a real pole filter: architecture of stage $i$
Figure 4. Magnitude and phase response of the MFIR filter approximation for $\lambda'$ at 0.99 with linear phase behavior; 9 stages

A DSP48A optimized implementation of the same filter stage $i$ is shown in Figure 5. The corresponding device utilization for the entire MFIR filter is shown in Figure 6. The number of slices reduces from 1632 (9%) to 773 (4%) while the estimated maximum frequency after synthesis increases from 164 MHz to 172.4 MHz in speed grade -4.

Figure 5: MFIR with linear phase behavior approximating a real pole filter: DSP48A optimized architecture of stage $i$
Figure 6: MFIR filter with linear phase behavior approximating a $\lambda'$ at 0.99; 9 stages, 17 bits wide data; DSP48A optimized

In case absolute bound scaling is used, there is no risk of overflow and the clipping block is not implemented, usually resulting in a higher maximum clock frequency. In the example above, the estimated maximum frequency after synthesis increases to 235 MHz in speed grade -4.

3.2 The approximation of a conjugate pole pair

Figure 7 shows the structure of a general stage $i$ of an MFIR filter with a linear phase behavior approximating the magnitude response of a filter having a conjugate pole pair. This structure corresponds with (2.11) and in Figure 7 no hardware optimization has been performed. $S_i$ equals the scaled first and last term in (2.11). $S_{ai}$ equals the scaled $\left(2\left(r^i\right)^2 + \left(r^i\right)^{-2}\right)\cos\left(2\theta^i\right)$ factor and $S_{bi}$ equals the scaled $\left(\left(r^i\right)^{-2}\cos^2\left(2\theta^i\right) + \left(r^i\right)^{2}\cos^2\left(\theta^i\right)\right)$ factor. As in the real pole approximation case, the scaling can be any type of scaling given in [6]. The implementation of this non-optimized design for a conjugate pole pair $\lambda' = 0.99 \ e^{i0.008}$ with 9 stages and 17 bits wide data in a Spartan3A-DSP requires 2884 slices (17%) and 27 multipliers. The estimated maximum frequency after synthesis equals 155 MHz in speed grade -4.
A DSP48A optimized implementation of the same filter is shown in Figure 8. The number of required slices reduces from 2884 (17%) to 1373 (8.25%) and the estimated maximum frequency after synthesis increases from 155 MHz to 172 MHz in speed grade -4. When absolute bound scaling is used, the clipping function in the “round and clip” block can be omitted, resulting in an estimated maximum frequency after synthesis of 231 MHz in speed grade -4.

Figure 5 and Figure 8 show that the MFIR structures fit very well in the DSP48A blocks of Xilinx®. The high maximum clock frequencies prove that the structures can be very effectively mapped in the Spartan3A-DSP FPGA. The relatively large amount of required memory can be implemented in an effective way in the FPGA.
slices. This results in a limited resource impact of about 4 to 8% when realizing very demanding filters with large bus widths.

4 Conclusion

It has been shown that the MFIR structure is very well suited for the approximation of real and complex pole IIR filters using FPGA’s. The MFIR implementation approach is fairly straightforward and requires no special HDL code structures. To the best of the author’s knowledge, there is no other more efficient structure that approximates the magnitude behavior of pole filters with the additional linear phase property. The hardware impact on the FPGA in terms of the number of slices and the number of multipliers is limited and a high maximum clock speed is possible.

References