Microprocessors and Microsystems 35 (2011) 595–602

Contents lists available at SciVerse ScienceDirect

Microprocessors and Microsystems

journal homepage: www.elsevier.com/locate/micpro

Recursive FIR filter structures on FPGA

Raija Lehto [a,*], Tarja Taurén [b], Olli Vainio [b]

[a] Tampere University of Technology, Department of Signal Processing, P.O. Box 553, FIN-33101 Tampere, Finland
[b] Tampere University of Technology, Department of Computer Systems, P.O. Box 553, FIN-33101 Tampere, Finland

ARTICLE INFO

Article history:
Available online 3 September 2011

Keywords:
Filter structure
Finite Impulse Response (FIR) digital filters
FPGA
Piecewise polynomial

ABSTRACT

A new approach to piecewise-polynomial approximation and recursive implementation structures for linear-phase Finite Impulse Response (FIR) filters have been recently proposed. In this paper, we describe hardware prototype implementations of the new structures for all four types of linear-phase FIR filters using a Field Programmable Gate Array (FPGA) based platform. Narrowband lowpass filters and narrowband differentiators are used as design examples to demonstrate the functionality and efficiency of the implementations. The required wordlength and resource usage is analyzed.

© 2011 Elsevier B.V. All rights reserved.

1. Introduction

Digital FIR filters are widely used in many signal processing applications due to their many well-known favorable properties and popular design methods. Their main drawback is high arithmetic complexity, i.e., the required number of adders, multipliers and delay elements needed in conventional implementations especially when a narrow transition band is required [1]. An efficient approach to overcome the above-mentioned problem is to synthesize linear-phase FIR filters so that their impulse response is piecewise-polynomial and the implementation is performed using recursive structures [2–8]. The arithmetic complexity of these filters is proportional to the number of impulse-response pieces and the overall polynomial order rather than the actual filter order.

Also, other computationally efficient approaches have been developed to reduce arithmetic complexity but these methods are implemented with non-recursive structures and they are mostly suited for signal-processor applications or software based implementations. The most efficient methods of this kind are Interpolated FIR-filter approach and Frequency–response-masking approach [9,10].

Mostly, the computationally efficient methods are based on the fact that the impulse response of a direct-form narrowband FIR filter generally has a smooth shape, and thus there is a strong correlation between successive coefficient values. Therefore, piecewise-polynomial approximation of the impulse response can give significant savings in arithmetic complexity. On the other hand, polynomial responses can be efficiently generated using recursive structures based on cascaded accumulators [2].

Boudreaux and Parks were among the first to propose a recursive piecewise-polynomial approximation for the impulse response of an FIR filter [3]. This principle was further generalized by Chu and Burrus [4,5]. Piecewise polynomial approximations have also been studied in Refs. [6,8]. More recently, new efficient recursive structures for FIR filter implementation have been developed by Lehto et al. [11,12]. In their approach, the optimization of the coefficients for the piecewise-polynomial approximation is done by linear programming. A recursive structure can be found for all four types of linear-phase FIR filters.

The objective of this work is to demonstrate the new recursive FIR structures on an FPGA board. Some preliminary results were obtained in Ref. [13]. FPGA-circuits are re-programmable and support implementations on finer granularity level than, e.g., software on a signal processor, since the wordlength can be customized. Therefore, the required effective performance of the structures can be obtained. The advantage of the FPGA-boards is that implementation is cheap and can also be used when small amounts of circuits are produced. Therefore, FPGA-circuits are very suitable to implement computationally efficient piecewise-polynomial FIR filter structures. Section 3 gives a summary of the Altera platform used in this work. Section 4 gives results of the implementation examples for all four types of linear-phase FIR filters. Section 5 gives comparative analysis of teh hardware complexity both by analytical formulas and numerical examples. Section 6 concludes the paper.

2. Filter structures

The impulse response is symmetrical for FIR filter Types 1 and 2, and anti-symmetrical for Types 3 and 4. The principle behind the piecewise-polynomial approximation in a nutshell is to divide the overall impulse response into subresponses and to generate

* Corresponding author.
E-mail address: raija.lehto@tut.fi (R. Lehto).

each subresponse with polynomials of a given degree. The subresponses are of different lengths and after summing them up the overall shape is obtained, with a different number of polynomials up to the center of symmetry in each block of the overall impulse response. All the subresponses are centered to the middle. Thus, the first subresponse consists of one polynomial, the second of two polynomials, etc. The center subresponse consist of the highest number, M polynomials. The most advantageous degree of the polynomials is typically $L = 3$, as using a higher degree for the polynomials means needing more coefficients with only slightly increased accuracy.

The general implementation structure for Type 1 and 3 filters is shown in Figs. 1 and 2, respectively. For the sake of clarity of the diagram, the coefficients $\alpha_k^{(m)}$ have been drawn twice. In a practical implementation, the inputs of the left-hand side and right-hand side $\alpha_k^{(m)}$s $\left[-\alpha_k^{(m)}\text{s}\right]$ are added [subtracted] and the result is multiplied by $\alpha_k^{(m)}$. The larger M is, the more delays and α coefficients are needed, and the structure expands horizontally. The polynomial degree is L. Thus increasing L makes the structure grow in the vertical dimension in Fig. 1. The β-coefficients in the middle of the structure are extra coefficients, which are needed to make the pole-zero cancellation. Thus it results in the desired symmetric or antisymmetric shape of the impulse response. The β-coefficients are calculated from the optimized filter coefficients according the rules given in Refs. [11,12]. The structure is slightly modified for the other FIR filter types due to the different symmetry properties, e.g., see Fig. 2.

To generate responses of finite length from a structure that includes feedback loops, pole-zero cancellation is essential. Therefore the following important considerations arise. Quantization of the polynomial coefficients should be done before deriving the coefficients in the structure. The structure does not automatically recover from any temporary data errors in accumulators due to feedback loops at the end of the structure. To ensure pole-zero cancellation in the long run, a switching and resetting principle as discussed in Ref. [10] can be used, i.e., two structures are implemented in parallel such that the data registers are doubled. One of the modules is operational while the other is being reset, and the roles are periodically switched. Such a structure is shown in Fig. 3. Two's complement arithmetic is used that has the useful property of allowing overflows in intermediate additions as long as the final result is in the range $[-1, 1-2^{-b}]$, where b is the number of fractional bits. It is also beneficial to use double wordlength in accumulators to minimize quantization errors. Worst-case scaling is required, i.e., the sum of the absolute values of the impulse response has to be less than or equal to unity. If necessary, the input signal or the coefficients are scaled accordingly.

3. FPGA platform

The hardware platform used in this work is the Altera DE2 development and education board that is based on the Altera Cyclone II EP2C35 FPGA [14]. Some of the features of the board are the following:

- 16-Mbit serial configuration device.
- Built-in USB interface.
- 8-MBytes SDRAM, 512 K SRAM, 4-MBytes Flash.
- 18 toggle switches, four pushbutton switches.
- 18 red LEDs, 9 green LEDs.
- 16 × 2 LCD display, eight 7-segment displays.
- 50 MHz and 27 MHz crystal oscillators.
- RS232, Infrared port, PS/2, 10/100 Ethernet.
- Video in and video out.
- Expansion headers (76 signal pins).

The key features of the Cyclone II EP2C35 FPGA are the following:

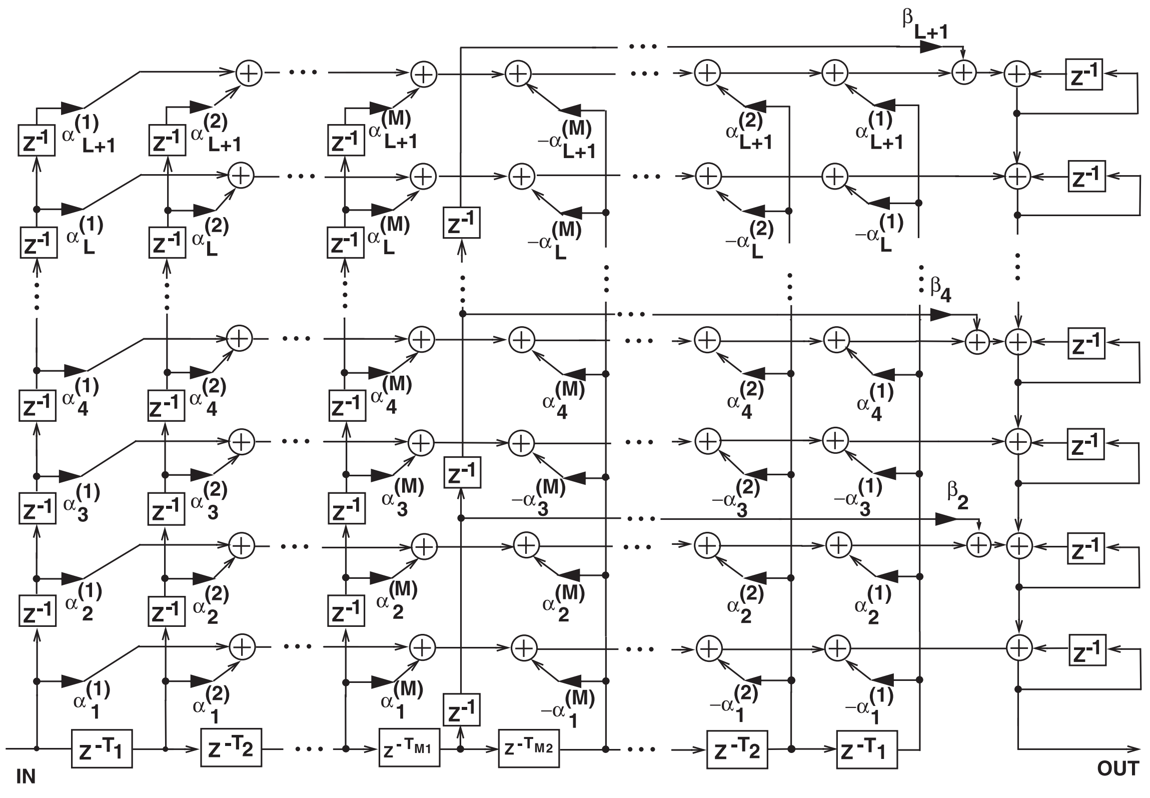

Fig. 1. Implementation structure of the Type-1 FIR filters [11].

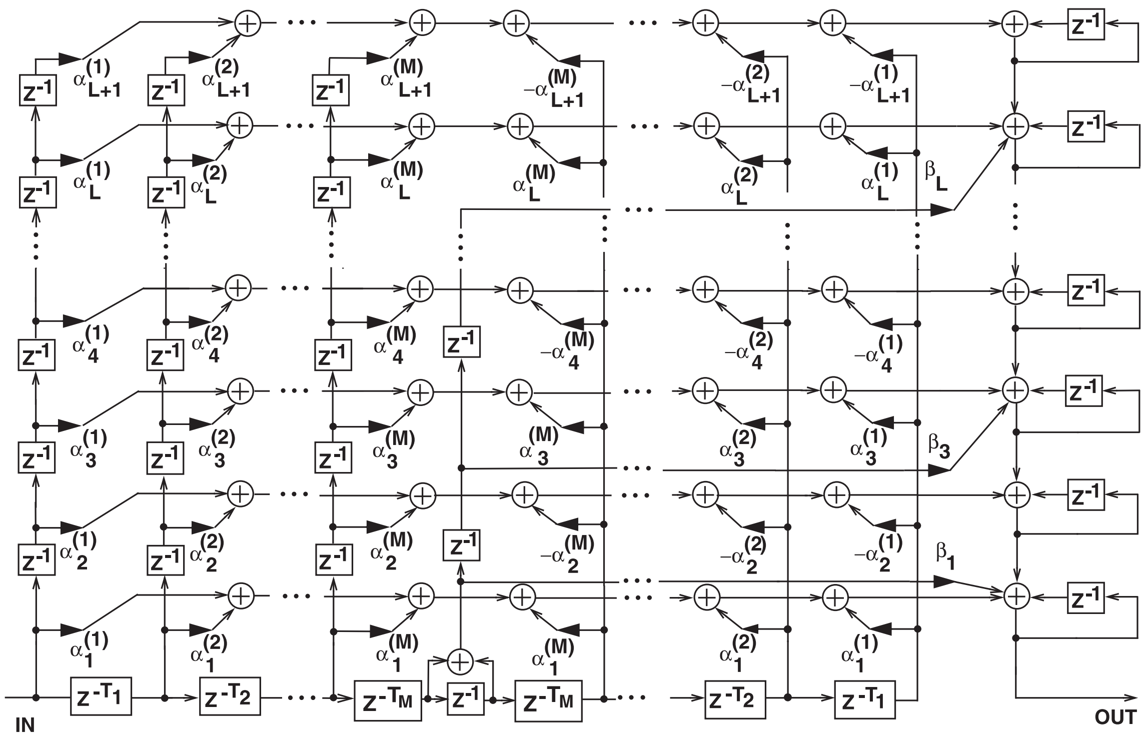

Fig. 2. Implementation structure of the Type-3 FIR filters [11].

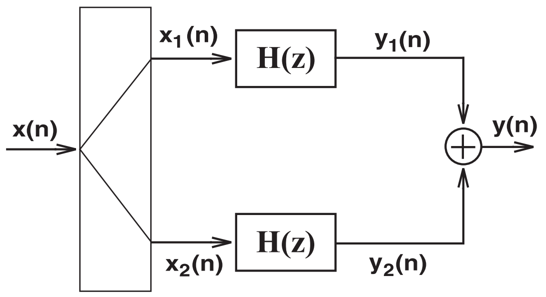

Fig. 3. Practical implementation based on switching and resetting, where a demultiplexer is used to decompose the input $x(n)$ into two signals $x_1(n)$ and $x_2(n)$ [11].

- 33216 Logic elements.
- 105 M4K RAM blocks.
- 483,840 total RAM bits.
- 35 embedded 18 × 18 multipliers.
- Four PLLs.
- 475 user I/O pins.

VHDL description was used for design entry in a PC environment. Simulations were carried out using Modelsim from Mentor Graphics [15]. Matlab [16] was used for coefficient optimization, early structural verification and for converting the implementation results between the time domain and frequency domain. The synthesis tool used in this work was Altera's Quartus II. The software also includes a tool for downloading the synthesized design into the FPGA. The functionality of the implementation was inspected by a software-based logic analyzer called SignalTap Logic Analyzer.

4. Filter implementations

In this section, we show the implementation results for design examples of the four types of linear-phase FIR filters. Narrowband lowpass filters are implemented as Types 1 and 2, and narrowband differentiators as Types 3 and 4. The design specifications are the following:

Passband edge $\omega_p = 0.025\pi$.
Stopband edge $\omega_s = 0.050\pi$.
Passband ripple $\delta_p = 0.01$.
Stopband ripple $\delta_s = 0.001$ (−60 dB).

For all of these designs, the polynomial degree $L = 3$ and the number of slices (see def. in Ref. [8], Eqs. (16a) and (16b)) $M = 5$, i.e., the overall impulse response consists of totally 10 blocks and 5 subresponses. Altogether 22 coefficients are needed in the implementation so that the number of α coefficients is 20 and the number of β coefficients is 2. The number of implementation coefficients depends only on the polynomial degree and the number of slices, not on the actual filter length.

4.1. Type-1 FIR filter

A Type-1 FIR filter is characterized by the symmetry property: $h(N - n) = h(n)$, $n = 0, 1, \ldots, N$, where N is even Using the piecewise-polynomial approximation, the specifications are met by a filter of length 223. The minimum wordlength for which the specifications are met is 1 + 33 bits. The impulse response and the magnitude response are shown in Fig. 4. Table 1 shows the quantized implementation coefficients in two's complement arithmetic for the structure in Fig. 4. As seen in Table 1, the quantized coefficients have quite small values, which means that there are several leading zeros, namely 12 in this case. This results to the fact that the effective number of bits is 21. Table 2 shows the delays used in our example.

The roundoff error in the finite wordlength implementation of the impulse response is shown in Fig. 5. This figure thus shows the difference of the impulse response between the design result from Matlab and the actual implementation with two's complement arithmetic.

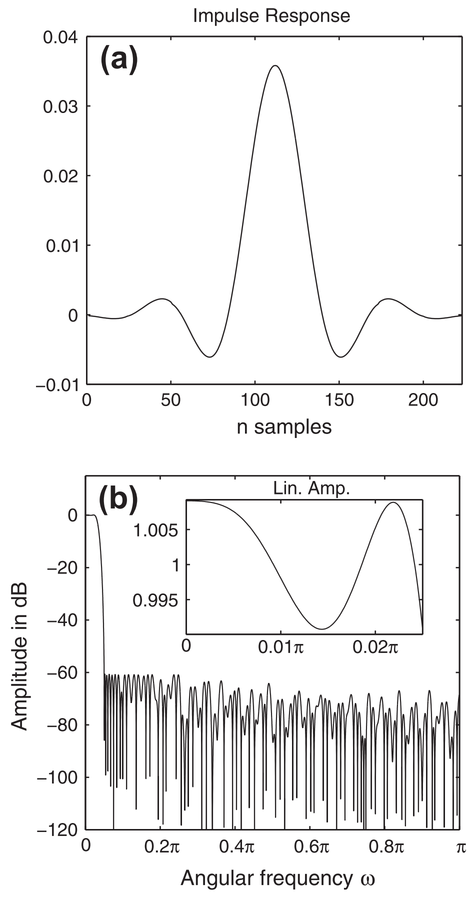

Fig. 4. (a) Impulse response and (b) magnitude response of the Type-1 filter example.

Table 1
Quantized implementation coefficients with two's complement form of the Type-1 lowpass filter.

Slice M	α_k
1	$\alpha_1 = -1.343471230939030\text{E}-04$
1	$\alpha_2 = -2.206047065556050\text{E}-05$
1	$\alpha_3 = -4.925997927784920\text{E}-06$
1	$\alpha_4 = 9.255018085241320\text{E}-07$
2	$\alpha_1 = 2.699275501072410\text{E}-05$
2	$\alpha_2 = -2.851814497262240\text{E}-05$
2	$\alpha_3 = 4.577450454235080\text{E}-06$
2	$\alpha_4 = -3.453344106674190\text{E}-06$
3	$\alpha_1 = -1.333394320681690\text{E}-04$
3	$\alpha_2 = 7.375888526439670\text{E}-05$
3	$\alpha_3 = -1.375330612063410\text{E}-05$
3	$\alpha_4 = 9.295530617237090\text{E}-06$
4	$\alpha_1 = -9.722949471324680\text{E}-05$
4	$\alpha_1 = -1.047406112775210\text{E}-04$
4	$\alpha_3 = -3.310199826955800\text{E}-05$
4	$\alpha_4 = -1.595565117895600\text{E}-05$
5	$\alpha_1 = -1.252338988706470\text{E}-04$
5	$\alpha_2 = -7.988919969648120\text{E}-05$
5	$\alpha_3 = -4.153233021497730\text{E}-05$
5	$\alpha_4 = 6.272457540035250\text{E}-07$
	β_k
	$\beta_2 = 7.022568024694920\text{E}-05$
	$\beta_4 = 1.712143421173100\text{E}-05$

The implementation of a single instance of the structure uses 2265 Logic Elements (LE), corresponding to 7% of the FPGA capacity. The number of memory bits needed is 44032 (9%) and the number of registers is 1204. A switching and resetting structure uses 4206 LEs (13%), 44,032 memory bits, and 1580 registers.

Table 2
Delays T_ks of each slice of the lowpass Type-1 filter.

Delay T_k				
$T_1 = 23$	$T_2 = 27$	$T_3 = 31$	$T_4 = 17$	$T_5(1) = 13$
$T_5(2) = 14$	$T_4 = 17$	$T_3 = 31$	$T_2 = 27$	$T_1 = 23$

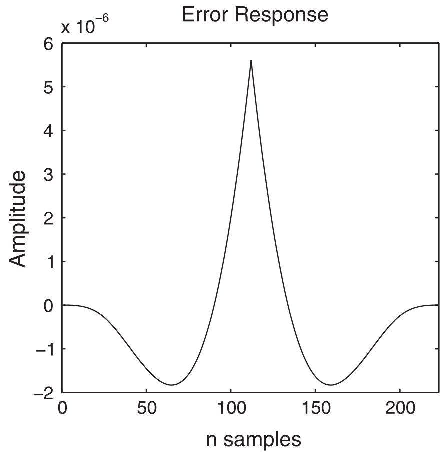

Fig. 5. Roundoff error in the impulse response of the Type-1 filter.

4.2. Type-2 FIR filter

A Type-2 FIR filter is characterized by the symmetry property: $h(N-n) = h(n)$, $n = 0, 1, \ldots, N$, where N is odd. The specifications are met by a filter of length 224. The minimum wordlength for which the specifications are met is 1 + 34 bits. The impulse response and the magnitude response are shown in Fig. 6. The implementation uses 2338 LEs (7%), 44,032 bits of memory (9%) and the number of registers is 1210.

4.3. Type-3 FIR filter

A Type-3 FIR filter is characterized by the antisymmetry property: $h(N-n) = -h(n)$, $n = 0, 1, \ldots, N$, where N is even. Because the filter should behave as a narrowband differentiator, the desired zero-phase frequency response on the passband is $D(\omega) = \omega$, and the minimum stopband attenuation is 60 dB.

The specifications are met by a filter of length 231, and the needed wordlength is 1 + 38 bits. The impulse response and the zero-phase magnitude response are shown in Fig. 7. The implementation uses 2590 LEs (8%), 46080 bits of memory (10%), and 1257 registers.

The roundoff error in the finite wordlength implementation of the impulse response is shown in Fig. 8. This figure thus shows the difference of the impulse response between the design result of Type 3 from Matlab and the actual implementation with two's complement arithmetic. As seen in Table 3, also in Type-3 case the quantized coefficients have quite small values, which means that there are several leading zeros, namely 12 in this case. This results to the fact that the effective number of bits is 26. Table 4 shows the delays used in our example.

4.4. Type-4 FIR filter

A Type-4 FIR filter is characterized by the antisymmetry property: $h(N-n) = -h(n)$, $n = 0, 1, \ldots, N$, where N is odd. Also in this case a differentiator is desired.

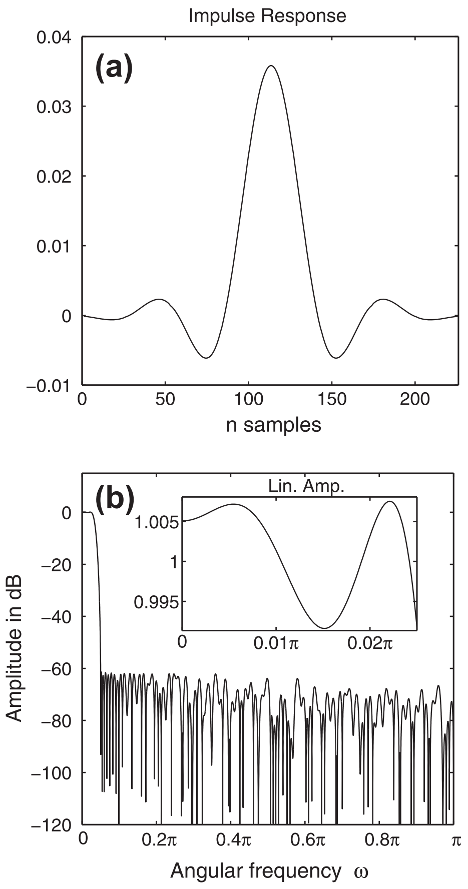

Fig. 6. (a) Impulse response and (b) magnitude response of the Type-2 filter example.

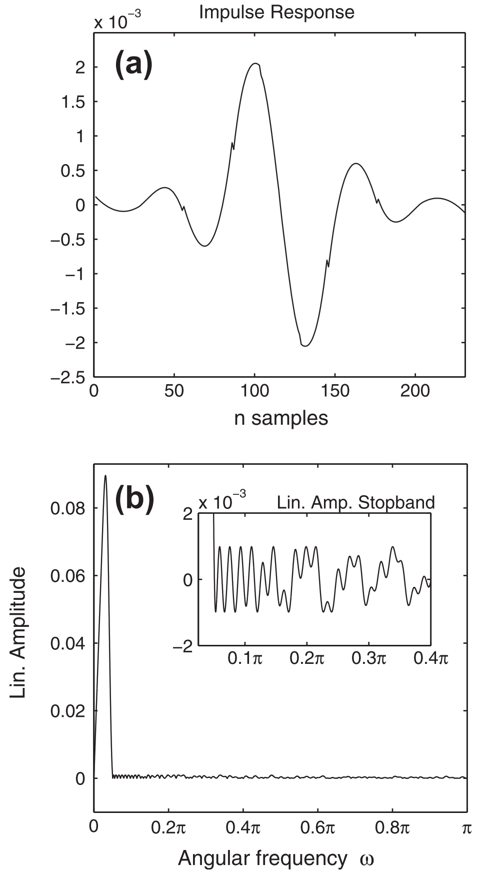

Fig. 7. (a) Impulse response and (b) zero-phase frequency response of the Type-3 differentiator.

The specifications are met by a filter of length 232, and the needed wordlength is 1 + 37 bits. The impulse response and the zero-phase magnitude response are shown in Fig. 9. The implementation uses 2409 Logic Elements (7%), 46,080 bits of memory (10%), and 1252 registers.

4.5. Implementation considerations

Implementation requires accurately optimized filter coefficients. In order to obtain desired accuracy in the coefficients, the SeDuMi 1.1 [17] was used in the optimization with the following parameters:

```
parsSDM.vplot = 0;
parsSDM.eps = 1e−22;
parsSDM.bigeps = 1e−20;
parsSDM.stepdif = 2.
```

Implementation itself should be carried out thoroughly regarding conversion into two's complement form of the coefficients and arithmetic operations and the required word-length. If the filter is implemented accurately according the given guidelines in Section 2, the minimum number of delays, adders and multiplications are obtained as indicated in Section 2, otherwise the complexity can be slightly higher. The filter requirements regarding the number of bits for all four types are a strict requirement. If teh number of bits is less, implementation does not meet the filter specification. Thereby, there is no trade-off but this is a strict requirement. The FPGA includes built-in 18 × 18-bit multipliers. However, in these structures the required wordlengths are so large, i.e., the

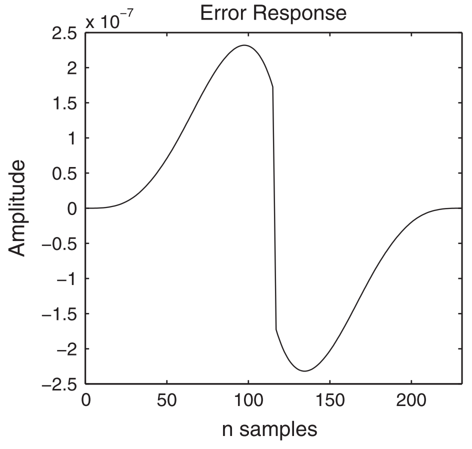

Fig. 8. Roundoff error in the impulse response of the Type-3 differentiator.

effective number of bits is at least 21, see Section 4, that the multipliers cannot be directly used. Instead, the multipliers for full word length are constructed as combinational logic using the logic elements. The synthesis tools support such constructs automatically.

5. Comparisons

This chapter shows the computational complexity of direct-form FIR filter realization compared to the proposed filter realizations. Table 5 shows the number of multipliers, adders and delays

Table 3
Quantized implementation coefficients with two's complement form of the Type-3 differentiator.

Slice M	α_k
1	α_1 = 119.797587103676e−006
1	α_2 = −23.8748070842121e−006
1	α_3 = 1.38318864628673e−006
1	α_4 = 4.12546796724200e−009
2	α_1 = 1.84456075658090e−006
2	α_2 = 4.40525764133781e−006
2	α_3 = −345.884473063052e−009
2	α_4 = −383.799488190562e−009
3	α_1 = 130.616983369691e−006
3	α_2 = 3.26719964505173e−006
3	α_3 = 13.4794536279514e−006
3	α_4 = 714.841007720679e−009
4	α_1 = −297.423819574760e−006
4	α_2 = −12.0516415336169e−006
4	α_3 = −35.5654046870768e−006
4	α_4 = 546.198862139136e−009
5	α_1 = −118.354084406747e−006
5	α_2 = −38.0758610845078e−006
5	α_3 = −18.7287223525345e−006
5	α_4 = 1.90710125025362e−006
	β_k
	β_1 = −34.5946937159169e−006
	β_3 = −6.13347947364673e−00

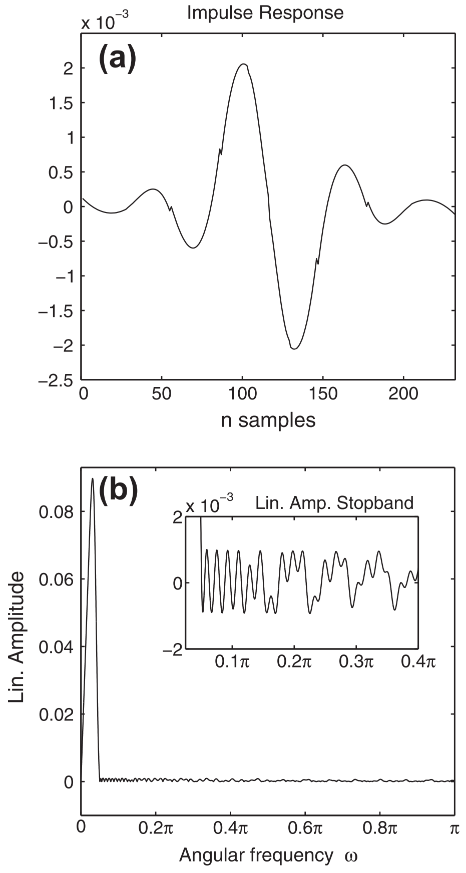

Fig. 9. (a) Impulse response and (b) zero-phase frequency response of the Type-4 differentiator.

Table 4
Delays T_ks of each slice of the Type-3 differentiator.

Delay T_k				
$T_1 = 28$	$T_2 = 27$	$T_3 = 31$	$T_4 = 17$	$T_5(1) = 12$
$T_5(2) = 12$	$T_4 = 17$	$T_3 = 31$	$T_2 = 27$	$T_1 = 28$

Table 5
Arithmetic complexity of Implementation of Piecewise Polynomial FIR filters for Types 1, 2, 3 and 4.

Number of	Type 1
Multipliers	$M(L+1) + \lfloor (L+1)/2 \rfloor$
Adders	$2M(L+1) + L + \lfloor (L+1)/2 \rfloor$
Delays	$N + L + 1$
Number of	Type 2
Multipliers	$M(L+1) + \lfloor (L+1)/2 \rfloor$
Adders	$2M(L+1) + L + 1 + \lfloor (L+1)/2 \rfloor$
Delays	$N + L + 1$
Number of	Type 3
Multipliers	$M(L+1) + \lceil (L+1)/2 \rceil$
Adders	$2M(L+1) + L + 1 + \lceil (L+1)/2 \rceil$
Delays	$N + L + 1$
Number of	Type 4
Multipliers	$M(L+1) + \lceil (L+1)/2 \rceil$
Adders	$2M(L+1) + L + \lceil (L+1)/2 \rceil$
Delays	$N + L + 1$

required for each proposed FIR filter realization. Table 6 gives the same for direct-form FIR filter [18]. Tables 8 and 9 show some numerical examples of the proposed filter structures for lowpass FIR filters. Tables 10 and 11 show numerical examples of the proposed filter structures for differentiators. Table 7 compares the complexity of each FIR filter type and the direct-form FIR filter type. As it can be observed in Tables 8 and 9 for lowpass Type 1 FIR filter the most advantageous design parameters are the case, where the polynomial degree is $L = 3$ and the number of slices is $M = 5$. The optimization of the filter coefficients is usually easiest for the polynomial degree $L = 3$.

Table 12, with design criteria $\omega_p = 0.025$, $\omega_s = 0.05$, $d_p = 0.01$ and $d_s = 0.001$, shows a comparison between the proposed Type 1 structure and the equivalent direct-form FIR filter in terms of wordlength and the number of transistors for delays, adders and multipliers. The number of transistors in ASIC implementations can be estimated in a technology-independent way by using the formulas given in Ref. [19].

The number of transistors in typical implementations of the building blocks are:

- $16b$ for a delay element,
- $28b$ for an adder/subtractor,
- $23b^2 + 26b - 58$ for a Booth array multiplier,

where b is the wordlength. Direct-form FIR filter Type 1 requires at least 16 bits minimum wordlength with filter degree $N = 215$. The proposed structure with filter degree 222 needs at least 21 effective bits due to small coefficients values (see Tables 1 and 3), which give 12 leading zeros in two's complement representation. Direct-form FIR filter requires 22% of the FPGA capacity and the sampling frequency is approximately 13.5 MHz compared to proposed FIR filters which require only 7% of the FPGA capacity. The number of delays can be implemented that the delays required by the polynomials (delays between filter coefficients) can be shared with the overall delay line (the bottom line in Figs. 1 and 2). Therefore, the number of delays in the implementation for Type 1, in Table 9 is only 226 while the filter degree is 222.

Table 6
Arithmetic complexity of direct-form FIR filter. N is the filter order.

Number of	Direct-form (DF)
Multipliers	$N+1$
Adders	N
Delays	N
Number of	with coeff. symmetry for Types 1, 2, 3 and 4
Multipliers	$N/2+1$, $(N+1)/2$, $N/2$, $(N+1)/2$
Adders	N
Delays	$N+2$

Table 7
Filter designs with the criteria: $\omega_p = 0.025$, $\omega_s = 0.05$, $\delta_p = 0.01$, $\delta_s = 0.001$. The number of slices, $M = 5$, the polynomial degree, $L = 3$.

Filter type	Number of multipliers	Number of adders	Number of delays
Type 1	22	45	226
Type 2	22	46	227
Type 3	22	45	235
Type 4	22	45	235
DF, lowp.	108	215	217
DF, diff.	112	223	225

Table 8
Filter designs for Type 1 lowpass filter with the criteria: $\omega_p = 0.025$, $\omega_s = 0.05$, $\delta_p = 0.01$, $\delta_s = 0.001$.

Polynomial degree	Number of slices	Number of multipliers	Overall ripple value
$L = 2$	$M = 8$	25	9.94893E−003
$L = 3$	$M = 5$	22	9.03716E−003
$L = 4$	$M = 4$	22	8.78672E−003
$L = 5$	$M = 4$	28	8.09919E−003

Table 9
Filter implementations for Type 1 lowpass filter with the criteria: $\omega_p = 0.025$, $\omega_s = 0.05$, $\delta_p = 0.01$, $\delta_s = 0.001$.

Polynomial L, M	Number of multipliers	Number of adders	Number of delays
$L = 2$, $M = 8$	25	51	227
$L = 3$, $M = 5$	22	45	226
$L = 4$, $M = 4$	22	46	227
$L = 5$, $M = 4$	28	57	228

Table 10
Filter designs for Type 3 lowpass differentiator with the criteria: $\omega_p = 0.025$, $\omega_s = 0.05$, $\delta_p = 0.01$, $\delta_s = 0.001$.

Polynomial degree	Number of slices	Number of multipliers	Overall ripple value
$L = 2$	$M = 7$	23	9.810202E−004
$L = 3$	$M = 5$	22	9.870342E−004
$L = 4$	$M = 4$	23	9.975399E−004

Table 11
Filter implementations for Type 3 lowpass differentiator with the criteria: $\omega_p = 0.025$, $\omega_s = 0.05$, $\delta_p = 0.01$, $\delta_s = 0.001$.

polynomial L, M	Number of multipliers	Number of adders	Number of delays
$L = 2$, $M = 7$	23	47	233
$L = 3$, $M = 5$	22	45	234
$L = 4$, $M = 4$	23	46	235

Table 12
The cost of Type 1 FIR filter. b is the number of bits.

A Filter	Delays $16b$	Adders $28b$	Multipliers $23b^2 + 26b - 58$
Proposed	75936	26460	233882
Direct-form	55552	96320	674568

6. Conclusions

Recursive FIR filter implementation structures have been successfully demonstrated on FPGA for all four types of linear-phase FIR filters. The observed responses satisfy the design specifications. Significant computational savings are achieved compared to direct-form structures as the number of needed multiplications is much lower. Only a small fraction of the FPGA capacity is used in the example cases for filter lengths exceeding 200. There is a very little difference in resource usage between the four types.

The implementations are fully parallel such that each new sample is processed in one clock cycle. Because of the cascaded additions, the critical timing path is quite long. The achievable sampling rate for these designs is about 35 MHz.

It was found that the structures are numerically quite sensitive, i.e., requires 21 effective bits in multipliers (e.g. Type 1). Therefore, a high numerical accuracy is needed in all stages of design process. Higher the numerical sensitiveness is, more bits are required in the implementation. In FIR filter case, when the number of multipliers decreases, more bits are required in the implementation and thereby, larger wordlengths. This is not a problem in customized FPGA implementations where resources can be allocated as needed. The structures implemented here could also be implemented by using ASIC-circuit technology.

In addition, as seen in Figs. 5 and 8, the error between the designed and quantized impulse response is very small, i.e., around 10^{-6} and 10^{-7} for each sample of the impulse response for types 1 and 3, respectively. The error in the impulse response samples resembles the form of the impulse response of each type.

Acknowledgement

This work was supported by the Academy of Finland under Grant 127858.

References

[1] T. Saramäki, Finite impulse response filter design, in: S.K. Mitra, J.F. Kaiser (Eds.), Handbook for Digital Signal Processing, Wiley, New York, 1993, pp. 155–277 (Chapter 4).
[2] T. Saramäki, O. Vainio, Structures for generating polynomial responses, in: Proc. 37th Midwest Symp. Circuits and Systems, vol. 2, August 1994, pp. 1315–1318.
[3] G.F. Boudreaux, T.W. Parks, Thinning digital filters: a piecewise-exponential approximation approach, IEEE Trans. Acoust. Speech, Signal Process. ASSP-31 (1983) 105–113.
[4] S. Chu, S. Burrus, Efficient recursive realizations of FIR filters. Part I: The filter structures, Circ. Syst. Signal Process. 3 (1984) 2–20.
[5] S. Chu, S. Burrus, Efficient recursive realizations of FIR filters. Part II: Design and applications, Circ. Syst. Signal Process. 3 (1984) 21–57.
[6] T.G. Campbell, T. Saramäki, Recursive linear-phase FIR filter structures with piecewise-polynomial impulse response, in: Proc. 6th Int. Symp. Networks, Syst. Signal Process., Zagreb, Yuogoslavia, June 1989, pp. 16–19.
[7] S.K. Mitra, A. Mahalanobis, T. Saramäki, A generalized structural subband decomposition of FIR filters and its application in efficient FIR filter design and implementation, IEEE Trans. Circ. Syst. II: Analog Digit. Signal Process. 40 (1993) 363–374.
[8] T. Saramäki, S.K. Mitra, Design and implementation of narrow-band linear-phase FIR filters with piecewise polynomial impulse response, in: Proc. IEEE Int. Symp. Circuits Syst., ISCAS'99, Orlando, FL, vol. 3, July 1999, pp. 456–461.
[9] T. Saramäki, Y. Neuvo, S.K. Mitra, Design of computationally efficient interpolated FIR filters, IEEE Trans. Circ. Syst. 35 (1) (1988) 70–88.
[10] Y.C. Lim, Frequency–response masking approach for the synthesis of sharp linear phase digital filters, IEEE Trans. Circ. Syst. 33 (4) (1986) 357–364.

[11] R. Lehto, T. Saramäki, O. Vainio, Synthesis of narrowband linear-phase FIR filters with a piecewise-polynomial impulse response, IEEE Trans. Circ. Syst. I 54 (10) (2007) 2262–2276.
[12] R. Lehto, Synthesis Methods for Linear-Phase FIR Filters with a Piecewise-Polynomial Impulse Response, Doctoral Dissertation, Tampere University of Technology, Tampere, Finland, 2009 (Publications 814).
[13] T. Taurén, O. Vainio, R. Lehto, Recursive FIR filter structures on FPGA, in: Proceedings of the Nordic Microelectronics Event NORCHIP 2009, 16–17 November 2009, Trondheim, Norway, 2009, 6p.
[14] http://www.altera.com/.
[15] http://www.mentor.com/.
[16] http://www.mathworks.com/.
[17] http://www.sedumi.ie.lehigh.edu/.
[18] S.K. Mitra, Digital Signal Processing: A Computer-based Approach, McGraw Hill, New York, 2006.
[19] P. Pirsch, Architectures for Digital Signal Processing, Wiley, New York, 1998.

Raija Lehto studied mathematics and computer science at the University of Lund, Sweden in the 1980s and received the Master of Science (Engineering) degree in information technology and the Doctor of Science (Tech.) degree in signal processing from Tampere University of Technology (TUT), Tampere, Finland, in 2004 and 2009, respectively, for her contribution to the study of digital filters. Her education includes also Teacher Training Studies, University of Tampere, Finland in 2004–2005. In 1989–2002 she served in the industry mostly in Sweden and also Finland in the field of computer science and data communication among others with Ferring Läkemedel AB, Telecom Finland Sweden AB and Sonera Plc. She had her residence in Sweden during 1980–1997. Since 1997 she has had her residence in Tampere, Finland. She is currently a Senior Researcher at Tampere University of Technology in the field of digital signal processing. Her research interests include among others digital filters with applications and transform techniques in systems biology. She has been teaching several courses in the field of digital signal processing since 2003 at TUT and she has also been a lecturer of digital linear filtering since 2006 at TUT. In 2004–2006 she served as a member of the University Degree Reform group at Tampere University of Technology with duties such as curriculum planning and course scheduling. She has also served as a reviewer of IEEE conferences and IEEE Transactions of Circuits and Systems-I: regular papers in 2006–2007 and other journals and conferences in 2009–2010. She has been a member of IEEE Circuits and Systems Society since 2006. She is also a member of both IEEE CAS Education and Outreach (CASEO) Technical Committee (TC) and IEEE Digital Signal Processing TC.

Tarja Tauren received the Master of Science degree in information technology from Tampere University of Technology, Tampere, Finland, in 2009, majoring in digital and computer systems. She is currently working as an information system technician on the private sector.

Olli Vainio received the degrees of Diploma Engineer and Doctor of Technology in electrical engineering from the Tampere University of Technology, Tampere, Finland, in 1984 and 1988, respectively. He has held research and teaching positions with this university and the Academy of Finland. In 1986–1987 he was a visiting scholar at the University of California, Santa Barbara, working on high-speed integrated circuit design. He has twice been a visitor at the Royal Institute of Technology, Stockholm, Sweden. In 2001–2002, he was senior scientist of the Academy of Finland. He is currently a professor of computer systems engineering in the Tampere University of Technology. He is also a docent of microelectronics in the Lappeenranta University of Technology. He has authored or coauthored 180 papers in journals and conferences internationally, and is a coholder of two patents. He has been a reviewer of several conferences, journals, and book publishers. His research interests are in computationally efficient digital filters, predictive and nonlinear filters, asynchronous circuits, and low-power DSP implementations.