Optimally Factored IFIR Filters

A. Mehrnia¹ · A. N. Willson Jr.¹

Received: 28 September 2017 / Revised: 19 May 2018 / Accepted: 22 May 2018 /
Published online: 4 July 2018
© The Author(s) 2018

Abstract This paper presents a new design method and a corresponding architecture for creating FIR filters that are significantly more hardware-efficient than presently known implementations. These *optimally factored IFIR* filters are also easily pipelined, thereby allowing operation at much higher data-rates. Using examples introduced by previous researchers, we show surprisingly better hardware efficiency. Two such examples show hardware reductions in the vicinity of 50%, relative to the conventional Remez structures, whereas previous research targeting this matter reports more modest results. We also show new features and further benefits that can be obtained by using optimally factored IFIR filters.

Keywords FIR filter · Factored filter · IFIR · Interpolated FIR · Optimal FIR filter · Cascade filter · Optimal filter design · Filter hardware complexity

1 Introduction

An interpolated FIR digital filter (often referred to as an “IFIR filter”) is well known to filter designers [6, 7, 11, 17, 19]. It uses a filter architecture that can be very efficient for making narrow-band lowpass (or, by simple mapping, highpass) filters. The IFIR transfer function $H(z)$ is constructed as a cascade connection of two FIR filters $H(z) = G(z^L)I(z)$ where the n -tap FIR filter $G(z)$ (often called the model filter) has its argument

✉ A. N. Willson Jr.
willson@ee.ucla.edu
A. Mehrnia
mehrnia@ucla.edu

¹ Electrical and Computer Engineering Department, University of California, Los Angeles, Los Angeles, California, USA

z replaced by z^L for a positive integer L (called the “stretch factor,” subsequently referred to as “SF”), and this replacement is equivalent to “stretching” the length of filter G to become approximately L times as long—more precisely, it will have $1 + (n - 1)L$ taps, with the majority of the tap coefficients having the value zero, hence having zero hardware cost for such tap-coefficient multipliers and their structural adders. This stretching in the time domain is equivalent to “shrinking” the transfer function $G(e^{j\omega})$ by the factor L in the frequency domain which gives insight as to why such functions can be efficient when used for narrow-band filters. Such frequency-domain shrinking, however, causes unwanted passbands, centered at $\omega = 2\pi/L, 4\pi/L, \dots, 2\pi(L - 1)/L$, to appear and these must be removed (or masked) by using the cheap (due to its wider transition band) lowpass filter $I(z)$, called the *interpolator* or *masking filter*. Refs. [6, 17, 19] provide more details on IFIR filters and their properties, and [7, 11] show how to choose an optimum stretch factor (SF) so that a filter will most efficiently meet given passband and stopband specifications.

Refs. [1, 3, 10, 12–14, 16, 20, 23, 24] suggest benefits from making FIR filters as a cascade of smaller FIR factors. The present work extends such factoring to IFIR filters and explores new features and further benefits that can be obtained. Figure 1 shows the magnitude response (dashed line) of an equiripple lowpass FIR filter of degree 15 (Example 1) that has the following specifications:

Passband edge $\omega_p = 0.042\pi$ rad.; Stopband edge $\omega_s = 0.14\pi$ rad.;
Ripple $\delta_p = 0.0839$ (± 0.7 dB); Attenuation $\delta_s = 0.0794$ (22 dB).

This filter is used in [12] to introduce our optimal FIR factoring algorithm and the filter’s coefficients are easily obtained by using the Parks–McClellan [9] Matlab statement:

$$\text{firpm}(15, [0 \ 0.042 \ 0.14 \ 1], [1 \ 1 \ 0 \ 0])$$

which yields the 16-tap FIR filter, described by a degree-15 polynomial

Fig. 1 Remez order-15 lowpass FIR filter example (dashed line), the same filter specs implemented using IFIR technique with SF = 2 (black solid line) and also with SF = 3 (gray solid line)

Fig. 2 Remez order-15 lowpass FIR filter

Fig. 3 Alternative factored implementation of FIR filter example

Fig. 4 Optimally factored non-IFIR cascade filter implementation (top), zero-map of the optimally-paired zeros and frequency response (magnitude in dB) for the order-15 lowpass filter (Example 1)

$$H(z^{-1}) = h_0(1 + z^{-15}) + h_1(z^{-1} + z^{-14}) + h_2(z^{-2} + z^{-13}) + \dots + h_7(z^{-7} + z^{-8})$$

whose most conventional implementation would be as shown in Fig. 2. Alternatively, by factoring the degree-15 polynomial into its natural factors (pairing-up complex-conjugate roots), we could envision an implementation in the form of a simple cascade having one first-order filter and seven second-order FIR filters, as shown in Fig. 3.

Many other factoring choices would also be possible, corresponding to the many possible combinations of natural factors of the $H(z^{-1})$ polynomial. It is shown in [12] that an “optimal” choice of factors for this filter could be the implementation comprising one first-order factor, three fourth-order factors, and one second-order factor, as shown in Fig. 4.

For additional details on our “optimal factoring algorithm,” we recommend [12, 14]. We now, however, begin an extension of this work to *IFIR* filter implementations.

Fig. 5 Optimal stretch factor for order-15 filter (Example 1)

In Fig. 5, we show that, following [11], the best choice of SF (the IFIR stretch factor) to minimize the number of coefficients is either 2 or 3. The SF=2 design $G_1(z^2)I_1(z)$ for this example (black solid line in Fig. 1) is obtained by using the single Matlab function *ifir* as follows: $[G_1, I_1] = \text{ifir}(2, \text{'low'}, [.042 \ .14], [.0839 \ .0794])$. The IFIR SF=3 design $G_2(z^3)I_2(z)$ can similarly be obtained as follows:

$$[G_2, I_2] = \text{ifir}(3, \text{'low'}, [.042 \ .14], [.0839 \ .0794], \text{'advanced'}) .$$

For this example, Fig. 5 does not make evident which of these two SF values is the clear winner. So one must simply design both and learn, as becomes evident in Row 3 of Table 1 (showing a total hardware complexity of 300 full adders and flip-flops) versus Row 4 (whose total complexity is 246). Thus, the choice of SF=3 happens to be the best.

1.1 In Regard to Optimal Factoring Filter Design

Being a practical algorithm for finding particularly beneficial factors, our so-called “optimal factoring of FIR filters,” introduced in [12, 14], finds *optimal pairings* of the *natural* complex-conjugate zero-pairs of an FIR transfer function. The filter in Fig. 4 is an optimally factored FIR (but, *not* an IFIR) cascade that meets Fig. 1 passband and stopband specs. As indicated by the straight lines in the Fig. 4 zero-map, the optimally factored cascade is obtained by factoring the Parks–McClellan order-15 transfer function $H(z)$ into its natural second-order factors and then (as explained in [12, 14]) these “zero-pair” factors are paired, to constitute “optimal pairings”:

$$(27.9^\circ, 133.2^\circ), \quad (43.7^\circ, 110^\circ), \quad (64.6^\circ, 156.6^\circ).$$

In the Fig. 4 factored filter, this yields the three fourth-order stages and it leaves the 180° zero and the 87° zero-pair to stand alone as first-order and second-order blocks.

Table 1 Hardware-complexity summary and comparison (order-15 Example 1) parameter B is the data width (wordlength)

Hardware complexity for various filter design methods	multiplier adders	structural adders	full adders	D flip-flops	total complexity	hardware reduction relative to Remez
Remez (direct form) $B = 6$ bits	15	15	198	90	288	-----
non-IFIR optimally factored $B = 7$ bits Fig. 4 cascade	7	15	154	105	259	$(288 - 259)/288 = 29/288 = 10.1\%$ reduction
IFIR non-factored (SF= 2)	12	14	150	150	300	$(288 - 300)/288 = \text{not applicable}$
IFIR non-factored (SF= 3)	5	11	96	150	246	$(288 - 246)/288 = 42/288 = 14.6\%$ reduction
optimally factored IFIR (SF = 2) $B = 6$ bits Fig. 11 cascade	0	13	78	150	228	$(288 - 228)/288 = 60/288 = 20.8\%$ reduction
optimally factored IFIR (SF = 3) $B = 6$ bits Fig. 7, Fig. 8	0	11	66	150	216	$(288 - 216)/288 = 72/288 = 25\%$ reduction

Using a similar approach, we now present a method for making optimally factored IFIR filters that has not previously been explored.

Table 1, line 2, gives our assessment of a 10.1% hardware savings to be expected by this Fig. 4 non-IFIR factoring example. This may seem a relatively minor improvement. However, we shall see that this small (degree-15) filter, which does not display many features that would identify it as a particularly good candidate for IFIR implementation (for example, it does not have a very narrow transition band) still achieves a rather impressive 25% hardware reduction once we combine the optimal factoring with the use of the IFIR architecture (as is shown in the *last* row of Table 1). For larger and more demanding filters, we have found, and will demonstrate herein, that even greater percentage-reductions in hardware can be expected. Before discussing this IFIR factoring further, we shall briefly explain our Table 1 computations. (Similar hardware-assessment techniques will be used throughout our subsequent discussions.)

1.2 Concerning Our Assessments of Filter Hardware Costs

In our subsequent discussions on IFIR transfer functions of the form $G(z^L)$, the presence of stretch factors $L > 1$ will cause us to consider FIR filter structures having a

Fig. 6 Approximate cost comparison: D flip-flop and full adder

cascade of numerous z^{-1} delays as alternatives for structures having fewer delays but more numerous (and often more expensive) tap-coefficient multipliers. We consider all FIR filters discussed here to have fixed-point binary multiplier coefficient values, implemented efficiently by circuits that employ hard-wired data shifts and additions (multiplier adders) of this shifted data. This is true (and commonplace) for direct-form as well as transposed-form FIR filter hardware implementations.

The “hardware efficiency” of a circuit will be affected by the number of additions required, which we assess in terms of the number of “multiplier adders” (MA) used. Also, other adders, so-called “structural adders” (SA), will be required to implement, for example, “plus or minus” operations like those shown within the boxes comprising the cascade structure at the top of Fig. 4 or, more generally, the additions for combining data that would take place in a conventional direct-form or a conventional transposed-form FIR filter. Ultimately this “adder hardware” will be assessed as the total number of single-bit “full adders” required to build these multiplier adders and the structural adders. And in doing this for our various examples we also account for the type of simplifications that are routinely employed by one skilled in the art, such as sub-expression sharing.

The amount of hardware needed to build the z^{-1} delays is, of course, also relevant in assessing the hardware cost of an FIR, and especially an IFIR, filter. We assess this in terms of the cost of the circuitry that implements a z^{-1} delay, and the predominant component of this circuitry is the single-bit “D flip-flop.” Again, like the adder costs, this cost will be influenced by the bit-width of the data-stream samples processed by the filter. It is outside the scope of this paper to dwell on the details of such circuitry but, since these circuit components will, for a single filter, all operate at the same data-rate as one-another, a simple transistor-level comparison, given in Fig. 6, showing the structure of a typical “full adder” and that of a typical “D flip-flop,” shows that *roughly* the same hardware complexity would be expected for these two components. Thus, we have chosen (as have various other publications referenced herein, e.g., [2, 12, 14]) to simply assess a filter’s “total complexity” in terms of the sum of the number of full adders and the number of D flip-flops required to implement the filter. We refer the interested reader to [2, 4, 5] for a further in-depth review of this topic.

In Sect. 2, we now use the small order-15 FIR filter, whose transfer-function magnitude $|H(e^{j\omega})|$ is plotted in Fig. 1 (dashed line), to illustrate some of the basic concepts for our *new* optimally factored IFIR filter design and implementation.

Notice that the important *new* concept of *joint* (versus individual) sequencing of the two sets of *model filter* and *interpolator filter* stages will also be introduced. The resulting filter structures are compared with the non-interpolated optimally factored (Fig. 4) design and with the conventional Remez implementation.

Additional *techniques* and *benefits* are also presented in Sects. 3 and 4 by examining the optimally factored IFIR design of two highly cited high-order filters.

2 Degree-15 Filter Example: Choice of Stretch Factor and New Joint Stage-Sequencing Technique

Example 1 Following [11], the choice of an optimum stretch factor for this filter is obtained and, as discussed previously, the two choices, indicated in Fig. 5, are $SF=2$ and $SF=3$, where, as summarized in Table 1, $SF=3$ is the better choice due to its slightly greater hardware efficiency. Notice that the exact hardware costs will ultimately involve the details of any specific implementation.

When $SF=3$ for Example 1, the $G(z)$ and $I(z)$ sub-filters have degrees 7 and 4, respectively (Fig. 5). The cascade of an optimally factored $G(z^3)$ and $I(z)$ is shown in Fig. 7, where we have identified the factors by using the optimal factoring theory and algorithm of [12]. The resulting Fig. 7 structure has *only* trivial coefficients (i.e., all coefficients are exact powers of two) and it needs just seven and four structural adders for $G(z^3)$ and $I(z)$, respectively. If approximately 0-dB DC gain is desired, two more shift-adds (by “shift-add” we mean a hard-wired shift and an addition) are needed to implement the post-filter gain-adjust multiplier 0.111011, as follows:

$$(\text{using } 0.111011 = 0.11110\bar{1} = 1.000\bar{1}0\bar{1}, \text{ where } \bar{1} \Rightarrow -1).$$

The ordering of stages for both $G(z^3)$ and $I(z)$ is determined by the sequencing algorithm as explained in [12]. The down-arrow and power-of-two multiplier at the output of each stage represents a datapath truncation (or rounding). In addition to the complexity reduction evident in Fig. 7, the *optimally factored IFIR implementation* provides an *additional* opportunity to combine and/or jointly sequence the factors of both $G(z^3)$ and $I(z)$. Such flexibility allows one to neutralize (or “tame”) any challenging (typically, large gain) factor in the resulting cascade [14]. Here, particularly, since the Fig. 7 structure is already multiplier-free, no such improvement from a further combining of stages is evident. However, this design *does* happen to benefit from the *joint sequencing of stages* (to be further illustrated in Examples 2 and 3). This yields an *improved* Fig. 8 optimally factored IFIR filter cascade with the overall magnitude (dB)

Fig. 7 The best identified cascade (Example 1) of an optimally factored individually sequenced $G(z^3)$ and $I(z)$ (when $SF=3$)

Fig. 8 Joint sequencing of factors of model filter and interpolator filter (when $SF=3$, yielding Example 1 optimally factored IFIR filter cascade)

Fig. 9 Zero-map and magnitude plots of Example 1 optimally factored IFIR ($SF=3$) filter (solid) versus order-15 Remez filter (dashed)

Fig. 10 Magnitude plots of five stages of the jointly sequenced optimally factored optimally stretched IFIR cascade filter structure in Fig. 8

shown in Fig. 9, demonstrating a superior stopband (greater attenuation, especially at mid and high frequencies) to that of the conventional FIR filter. The zero-map of this optimally factored IFIR filter and the quantized conventional (Remez) filter is shown in Fig. 9. The frequency responses of the five individual factored stages are also shown in Fig. 10.

Fig. 11 Optimally factored IFIR cascade filter of $G(z^2)$ and $I(z)$ (when $SF=2$)

Fig. 12 Zero-map and magnitude plot of Fig. 11 optimally factored IFIR ($SF=2$) filter (solid) versus order-15 Remez (dashed)

When $SF=2$ for Example 1, the $G(z)$ and $I(z)$ sub-filters have degrees 11 and 3, respectively, and the corresponding $G(z^2)$ and $I(z)$ cascade is shown in Fig. 11. Again, the resulting structure has only trivial coefficients and needs just eleven and two structural adders for the IFIR components, in addition to one shift-add for the post-filter multiplier. The sequencing of stages for both $G(z^2)$ and $I(z)$ is again based on [12]. The Fig. 12 frequency response of the resulting ($SF=2$) optimally factored IFIR filter also demonstrates a superior stopband to that of the conventional (dashed line) design. The zero-maps of both implementations are also shown in Fig. 12. The extra lines in Fig. 12 indicate which red zeros are paired together to form each factor of the Fig. 11 structure. There are four complex-conjugate zero-pairs with $\pm 90^\circ$ angles. One pair is combined with two other zero-pairs (having angles of 27.9° and 152.1°), while the other $\pm 90^\circ$ conjugate-pairs stand alone.

Table 1 gives the hardware-complexity comparison for the optimally factored IFIR filters versus the conventional Remez (direct-form) FIR filter, as well as the (non-IFIR) optimally factored filter, and the IFIR non-factored filters. Clearly, the optimally factored $SF=3$ IFIR filter has the fewest adders and lowest total complexity. Due to the modest 22-dB stopband attenuation target in Example 1, the wordlength of the signal path can require as few as just six bits for the optimally factored IFIR cascade implementations.

Table 2 Hardware-complexity comparison for order-59 Example 2 FIR filter (B represents the wordlength of the datapath)

Hardware complexity for alternative filter design methods	multiplier adders	structural adders	total number of adders	full adders	D flip-flops	total complexity
Modified optimally factored IFIR design Ripple = ± 0.1225 dB $H(z) = G(z^3)I(z)$	9	31	40	560 ($B = 14$ bits)	994 1190 for fully pipelined case	1554 or fully pipelined 1750 ($B = 14$)
Optimally factored IFIR design $H(z) = G(z^3)I(z)$	10	31	41	574 ($B = 14$ bits)	994 1190 for fully pipelined case	1568 or fully pipelined 1764 ($B = 14$)
Optimally factored non-IFIR design filter order = 59 [12]	30	58	88	1270 ($B = 14$ bits)	842 1128 for fully pipelined case	2112 or fully pipelined 2398 ($B = 14$)
Genetic algorithm Two-stage cascade filter order = 60 [26]	13	56	69	Our estimate: >1500 ($B = 12$)	Our estimate: > 1300 ($B = 12$)	Our estimate: >2800 ($B = 12$)
Two-stage cascade filter order = 59 [22]	14	55	69	697+69 B [22] =1525 ($B = 12$)	Our estimate: > 1300 ($B = 12$)	Our estimate: >2825 ($B=12$)
MILP method [21, 27] filter order = 59	19	59	78	Our estimate: > 1600 ($B = 12$)	Our estimate: > 1350 ($B = 12$)	Our estimate: >2950 ($B=12$)
Single-stage design filter order = 59 [21]	17	59	76	729+76 B [22] = 1641 ($B = 12$)	Our estimate: > 1350 ($B = 12$)	Our estimate: >2991 ($B=12$)
FIRGAM [2] filter order = 59	27	59	86	Our estimate: 1680 ($B = 12$) 1337 if $B=8$ [2]	Our estimate: > 1410 ($B = 12$) 1180 if $B=8$ [2]	Our estimate: >3090 ($B = 12$)
PMILP [25, 2] filter order = 59	25	59	84	Our estimate: 1678 ($B = 12$) 1342 if $B=8$ [2]	Our estimate: > 1490 ($B = 12$) 1255 if $B=8$ [2]	Our estimate: >3168 ($B = 12$)
CSD [18, 2] filter order = 59	32	59	91	Our estimate: 1819 ($B = 12$) 1455 if $B=8$ [2]	Our estimate: > 1500 ($B = 12$) 1298 if $B=8$ [2]	Our estimate: >3319 ($B = 12$)
Remez [2] filter order = 59	32	59	91	Our estimate: 1858 ($B = 12$) 1494 if $B=8$ [2]	Our estimate: > 1480 ($B = 12$) 1251 if $B=8$ [2]	Our estimate: >3338 ($B = 12$)

3 An Order-59 Filter Example [18]: Factored IFIR Efficiency, and Additional Benefits

Example 2 We now consider a larger (order-59) FIR filter, one that we have examined in [12] using a *non-IFIR* optimally factored cascade. We shall show that the optimally factored *IFIR* filter improves significantly upon the result obtained in [12]. Indeed, as elaborated on in this section, it excels notably in comparison with all previously published methods cited in Table 2, where it promises an approximately 50% reduction

Fig. 13 Finding optimum SF for degree-59 filter (Example 2)

in hardware complexity, compared with the hardware complexity of a conventional (Remez) FIR implementation.

This degree-59 lowpass filter, referred to as filter S2 in [2], has these specifications:

Passband edge $\omega_p = 0.042\pi$ rad.; Stopband edge $\omega_s = 0.14\pi$ rad.;
 Ripple $\delta_p = 0.012$ (± 0.1035 dB); Attenuation $\delta_s = 0.001$ (60 dB).

Again, we obtain this filter's optimum stretch factor (SF=3), via [11]. As shown in Fig. 13, the orders for model filter $G(z)$ and interpolator $I(z)$ are 20 and 11, respectively.

The optimal factors for $G(z)$ and $I(z)$, found using [12, 14], are shown in Fig. 14. A practical realization of this optimally factored IFIR filter, $H(z) = G(z^3)I(z)$, requires a careful stage sequencing [12] in order to effectively manage the datapath wordlength through the cascade of stages. The cascade structure in Fig. 14 uses *individual sequencing* of factors for $G(z^3)$ and $I(z)$. This design requires a 15-bit datapath (including the sign bit).

Figure 15 shows a better (than Fig. 14) cascade design, using the *joint sequencing* of the $G(z^3)$ and $I(z)$ factors. Its datapath wordlength is reduced to 14-bits (including sign bit) due to better noise performance (discussed further in this section). This structure has just nine non-trivial coefficients, which are realizable with a total of ten shift-adds. There are 20 and 11 structural adders for $G(z^3)$ and $I(z)$, respectively, in addition to the three shift-adds needed to implement the post-filter gain-adjust multiplier shown in Fig. 15. To achieve the highest data-rates, the cascade structure is easily pipelined by inserting registers between the stages. (See discussion in [12], re. Fig. 7 in [12].) We refer to an optimally factored IFIR structure as being “partially pipelined” if pipeline registers are present at the output of some of the stages. The factored structure is “fully pipelined” when a pipeline register is present between *all* adjacent stages.

Fig. 14 Optimally factored IFIR implementation of Example 2 filter $H(z)$ without joint sequencing

Fig. 15 Joint sequencing of the stages to yield the optimally factored IFIR implementation of Example 2 filter $H(z)$

Figures 16 and 17 show the frequency response plot and zero-map for the Fig. 15 optimally factored IFIR implementation. The magnitude plot demonstrates superior stopband characteristics compared to those of the conventional structure (the dashed line), particularly at mid and high frequencies. The zero-map of Fig. 17 illustrates the zero distribution of the optimally factored IFIR cascade versus that of the conventional (direct-form) implementation. Again, straight lines indicate which zeros are paired to form each of the Fig. 15 factors (according to the optimal pairing algorithm of [12, 14]).

Benefit: If desired, the optimally factored IFIR filter easily allows a *non-uniform* datapath wordlength across the stages of the cascade. This can efficiently deliver better noise performance, as the dynamic range of each stage output can easily be optimally and independently adjusted, as will be discussed in Sect. 4.

Table 2 gives a summary and comparison of the various methods of implementing this filter. Clearly, the optimally factored IFIR filter of Fig. 15 has the lowest complexity. Moreover, when this factored IFIR filter is fully pipelined, and hence is capable of operating at data-rates unreachable by conventional FIR implementations (i.e., at speeds attainable only by transposed FIR forms). The optimally factored IFIR filter

Fig. 16 Magnitude plot of Example 2 optimally factored IFIR (SF=3) filter (solid) versus order-59 conventional (dashed) filter

Fig. 17 Zero-map of Example 2 optimally factored IFIR (red circles) filter and order-59 conventional filter (blue dots) (Color figure online)

still achieves a hardware cost (FA+FF) reduction of 47%, in comparison with the conventional Remez filter:

$$(3338 - 1568)/3338 \approx 53\%$$

or, when fully pipelined, as: $(3338 - 1764)/3338 \approx 47\%$.

Parameter B in Table 2 is the datapath wordlength, which should be at least 12 bits (including the sign bit) to allow a single-stage conventional design to provide enough resolution to be able to realize a 60-dB attenuation of the incoming signal. For the Fig. 15 factored IFIR filter, as discussed earlier, a wordlength of 14 bits is required. Table 2 also provides complexity comparisons of the FIRGAM method [2],

the original CSD implementation of this example filter [18] (which, being an early CSD filter, was focused on reducing adder costs only), the PMILP algorithm [25], the minimum-adder MILP [21], the cascade method [22], and the genetic algorithm cascade [26].

Comparisons: Area, Speed, and Power Consumption

Table 3 shows the results of our Verilog implementation and synthesis, using Cadence tools (TSMC 65 nm), of the IFIR filter of Fig. 15 in three forms (non-pipelined, partially pipelined and fully pipelined) to compare the area and power requirements at multiple operating speeds (sampling rates). Here “partially pipelined” refers to a critical path reduction, inserting five pipelining registers between the fifteen stages of Fig. 15 as follows:

$$1\ 2\ 3\ |\ 4\ 5\ 6\ |\ 7\ 8\ 9\ |\ 10\ |\ 11\ 12\ |\ 13\ 14\ 15.$$

The fully pipelined optimally factored structure has fourteen pipelining registers (one register at the output of *each* of the first fourteen stages in Fig. 15). Table 3 shows that when the factored IFIR filter is not pipelined it has the smallest area, but the longest critical path and, as stated earlier, it is suitable only for applications where high speed is not required. Due to its long critical path, the synthesis tool had to increase its logic gate sizes in order to operate at 100 MHz, resulting in slightly higher power consumption than the pipelined designs. Its maximum operating speed was then 160 MHz. In contrast, the fully pipelined optimally factored designs had the shortest critical paths and hence the synthesis tool was able to achieve very high sampling rates using mostly *small* gate cells. While the transposed design and the fully pipelined factored design can both reach, at most, a speed of 900 MHz, notice that the transposed design requires a considerable increase in gate sizes (hence, considerable increases in area and power) in order to operate at this speed.

Notice that the conventional transposed design’s area and power requirements at 900 MHz are, respectively, 3.5 times and 53% higher than those of the optimally factored IFIR filter.

Also the conventional direct-form filter can operate only at speeds up to 500 MHz and even at that relatively low speed, it consumes a 2.3 times larger area and 28% more power than the fully pipelined Fig. 15 optimally factored IFIR filter.

To demonstrate the factored IFIR stage-sequencing effectiveness, we perform the “four-test procedure” described in [12]. The following comprehensive tests use the datapath wordlength $B = 14$, including sign bit, in all cases. We assess the signal RMS values at all stage outputs, normalized to the input signal RMS. The chains of RMS values of the signal at the outputs of the cascade stages for these four tests are reported in Figs. 18 and 19 (where 8000 signal samples are used in each case):

Test 1) The input signal is white Gaussian noise (uniform power across all frequencies). We expect the filter to attenuate by 60 dB the portion of the signal within the stopband.

Table 3 Detailed performance comparison (area and power consumption) of the optimally factored IFIR (Fig. 15) and optimally factored non-IFIR [12] structures versus direct-form and transposed-form designs. Cadence results—TSMC 65 nm library

Sampling rate		Direct Form	Transposed Form	Optimally Factored IFIR (Fig. 15) Not pipelined	Optimally Factored IFIR (Fig. 15) Partially pipelined	Optimally Factored IFIR (Fig. 15) Fully pipelined	Optimally Factored Non-IFIR [12] Fully pipelined
100 MHz	Area (μm^2)	21995	32462	12800	13284	14278	19968
	Power (mw)	6.224	5.991	4.188	3.697	3.404	5.337
	Time slack (ps)	5869	6319	0	6445	7476	7170
200 MHz	Area (μm^2)	21997	32498	N/A	13284	14278	19968
	Power (mw)	7.478	8.095	N/A	5.582	5.079	7.153
	Time slack (ps)	869	1323	N/A	1453	2490	2170
400 MHz	Area (μm^2)	26356	33785		13525	14291	20029
	Power (mw)	10.22	12.229		8.493	8.253	10.123
	Time slack (ps)	0	0		0	36	0
500 MHz	Area (μm^2)	33340	35702		15778	14331	20411
	Power (mw)	12.6	14.477		10.181	9.844	11.629
	Time slack (ps)	0	0		0	1	0
600 MHz	Area (μm^2)	N/A	39986		17907	14709	21628
	Power (mw)	N/A	16.642		12.097	11.407	13.248
	Time slack (ps)	N/A	0		0	0	0
800 MHz	Area (μm^2)		50343		N/A	15354	24010
	Power (mw)		22.387		N/A	14.666	16.827
	Time slack (ps)		0		N/A	0	0
900 MHz	Area (μm^2)		56891			16090	27840
	Power (mw)		25.165			16.412	19.135
	Time slack (ps)		0			0	0

Test 2) The input signal is colored Gaussian noise with uniform power within the stopband. It is a sum of 100 random-phase sinusoids, uniformly distributed across the stopband ($\omega \geq 0.14\pi$). We expect a 60-dB attenuation of the entire signal.

Test 3) The input signal is one sinusoid at the passband edge.

Test 4) The input signal is one sinusoid at the stopband edge.

Figures 18 and 19 show that the Fig. 15 optimally factored IFIR filter is able to fully attenuate (by at least 60 dB) the stopband portions of the input signal (including a sinusoid at the edge of the stopband) and it is able to perfectly pass the passband signals (including a sinusoid at passband edge) with negligible (less than 0.1-dB) attenuation. Figure 19 shows the progress of the RMS stage outputs throughout the cascade for the two sinusoidal test cases at the passband and stopband edges (Test 3 and Test 4).

Fig. 18 RMS output of stages of Fig. 15 jointly sequenced optimally factored IFIR filter (Tests 1 and 2)

Fig. 19 RMS output of stages of Fig. 15 jointly sequenced optimally factored IFIR filter (Tests 3 and 4)

An Additional *Benefit* provided by the inherent flexibility of the optimally factored IFIR filter in Fig. 15:

If a (very modest) 0.019-dB increase is allowed in the passband ripple (i.e., changing from ± 0.1035 to ± 0.1225 dB), then the 8th-stage $[1 - 0.46875z^{-3} + z^{-6}]$ in the Fig. 15 structure can be further simplified to become $[1 - 0.5z^{-3} + z^{-6}]$, while the rest of the cascade factors can remain intact. The resulting modified stage has *only* trivial coefficients, which yields a further reduction in the shift-add operations necessary for implementing the Fig. 15 filter coefficients (a reduction by 10% from ten down to nine multiplier adds). The importance of this observation is that:

In general, we have found that, given a minor (usually acceptable) allowance in some of the target filter specifications, it is *often* possible to exploit it to further simplify a specific stage (or stages!) of the optimally factored IFIR filter. In particular, this can be done *without the need to change any of the other stages* in order to reduce the filter's overall hardware complexity.

Fig. 20 Choice of optimum stretch factor for Example 3 IFIR filter

4 A Hardware-Efficient Wideband Filter Design Via Optimally Factored IFIR Implementation: Order-62 Filter Example from [2, 8, 27]

Example 3 We now consider a highly cited order-62 wideband filter [8] having the following specifications:

Passband edge $\omega_p = 0.2\pi$ rad.; Stopband edge $\omega_s = 0.28\pi$ rad.;
 Ripple $\delta_p = 0.028$ (± 0.24 dB); Attenuation $\delta_s = 0.001$ (60 dB).

Similar to the order-59 filter in Sect. 3, this filter, referred to as filter L2 in [2], is a convenient example because several previous publications [2, 8, 18, 21, 22, 25, 27] have chosen to use it when presenting their own filter design and implementation methods. These include the FIRGAM and Remez algorithms [2], an algorithm (LIM) from [8], the Partial Mixed-Integer Linear Programming (PMILP) algorithm of [25] and the single-stage and dual-stage designs using the coefficient optimization algorithms in [21, 22].

We first demonstrate an optimally factored IFIR implementation of filter L2, and we compare its complexity with the above-cited designs. Our filter implementation will also provide the opportunity to demonstrate: ANOTHER BENEFIT of our optimally factored filters: i.e., due to the *relatively small size* of our FIR factors, it is *often* possible to find some (otherwise not particularly obvious) opportunities to further reduce the number of add operations required for implementing some FIR coefficients.

The optimum stretch factor (SF = 2) for this order-62 filter, via [11] (as illustrated in Fig. 20), leads to orders of 34 and 9 for the model filter $G(z)$ and the interpolator filter $I(z)$, respectively. Filter L2 has the 62 zeros shown as blue dots in Fig. 21b, of which sixteen are off the unit circle (representing four fourth-order factors), and 46 zeros are on the unit circle (representing 22 complex-conjugate zero-pairs and two zeros at $\omega = \pi$). Attempting an exhaustive pairing and factoring of all complex-

Fig. 21 Zero-maps of IFIR components $G(z)$ and $I(z)$ and optimal pairings for the optimally factored IFIR versus zeros of order-62 filter L2 (Example 3)

conjugate zero-pairs would, of course, be impractical since there are more than 6.5×10^{14} possible factoring choices for model filter $G(z)$. However, by employing our optimal factoring algorithm for this filter, the best identified factors for $G(z)$ and $I(z)$ (Tables 4, 5 and Fig. 22) are found. The results are illustrated in Fig. 21a which

Table 4 Model filter $G(z)$: quantized stages and binary representations, L2 filter using optimal pairing identified in Fig. 21a

Optimally factored $G(z)$	Binary representation absolute value of taps	
1: $(1+z^{-4}) - 1.25(z^{-1}+z^{-3})$	1.01	
2: $1 + 0.375z^{-1} + z^{-2}$	0.011	
3: $1 + 0.5z^{-1} + z^{-2}$	0.1	
4: $1 + 0.75z^{-1} + z^{-2}$	0.11	
5: $(1+z^{-7}) - 0.96875(z^{-1}+z^{-6})$ $- 0.15625(z^{-2}+z^{-5}) + 1.21875(z^{-3}+z^{-4})$	0.11111	
6: $(1+z^{-6}) - 2.59375(z^{-1}+z^{-5})$ $+ 1.25(z^{-2}+z^{-4}) + 0.875z^{-3}$	0.00101	1.00111
7: $1 + 1.75z^{-1} + z^{-2}$	10.10011	
8: $(1+z^{-6}) - 0.125(z^{-1}+z^{-5})$ $+ 0.625(z^{-2}+z^{-4}) + 2z^{-3}$	1.01	0.111
9: $1 + 1.875z^{-1} + z^{-2}$	1.11	
10: $1 + z^{-1}$	0.001	
	0.101	10.0
	1.111	
	none	

Table 5 Interpolator filter $I(z)$: quantized stages and binary representations, L2 filter using optimal pairing identified in Fig. 21a

Optimally factored $I(z)$	Binary representation absolute value of taps
11: $1 - 4(z^{-2}+z^{-3}) + z^{-5}$	100.0
12: $1 + 1.5z^{-1} + z^{-2}$	1.1
13: $1 + 2z^{-1} + z^{-2}$	10.0

Fig. 22 Optimally factored interpolated FIR implementation of filter L2 (Example 3)

shows the optimal pairing of zeros for the $G(z)$ and $I(z)$ filters. Figure 21b compares zeros of the resulting *optimally factored IFIR* design with the 62 zeros (blue dots) of the original 63-tap filter L2 according to [8].

The binary values of coefficients for $G(z)$ and $I(z)$, listed in Tables 4 and 5, indicate that most factors can be implemented very cheaply. Indeed, only Factor 5 and Factor 6 (the two largest factors) have coefficients that require more than one MA (multiplier adder) in their implementation. The Appendix explains how we can implement each of these factors with just two MA. (Admittedly, we do somewhat blur the distinction between MA and SA: we increase the number of SA.) Overall, however, we achieve a net reduction of one addition for each factor: we need 2 MA and 8 SA for Factor 5, and the same for Factor 6.

Table 4 shows that 33 structural adders (SA) are needed to realize the order-34 $G(z)$, and Table 5 shows that the order-9 $I(z)$ needs merely seven SA. (Just count the number of plus/minus signs in Tables 4 and 5.) Our above-mentioned modifications increase SA by *one* for Factor 5 and by *two* for Factor 6 (with corresponding reductions of *two* MA for Factor 5 and *three* MA for Factor 6). Therefore, the resulting *optimally factored IFIR* requires a total of 11 MA and 43 SA as is also summarized in Table 6.

Similar to our previous examples, a practical realization of the $G(z^2)I(z)$ cascade requires a careful sequencing of stages to effectively manage the datapath wordlength through the cascade. The best identified stage-order using *joint* sequencing of the $G(z^2)$ and $I(z)$ stages, according to the sequencing algorithm given in [12], when applied to this *optimally factored IFIR* implementation of filter L2, is:

$$\text{Stage Order} = 8 \quad 12 \quad 4 \quad 6 \quad 3 \quad 7 \quad 5 \quad 2 \quad 9 \quad 1 \quad 10 \quad 13 \quad 11$$

where the numbers 1 through 13 correspond to row numbers given in Tables 4 and 5.

Table 6 Hardware-complexity comparison for order-62 wideband FIR filter (B represents the wordlength of the datapath)

Hardware complexity for alternative filter design methods	multiplier adders	structural adders	total number of adders	full adders	D flip-flops	total complexity
Optimally factored IFIR design $H(z)=G(z^2)I(z)$ Fig. 22	11	43	54	789 ($B = 14$ bits)	1081 1255 for fully pipelined case	1870 fully pipelined: 2044 ($B = 14$)
Optimally factored non-IFIR design when filter order = 64 [12, 14]	31	60	91	1350 ($B = 14$)	920 1130 for fully pipelined case	2270 fully pipelined: 2480 ($B = 14$)
Optimally factored non-IFIR design filter order = 62 [12, 14]	47	62	109	1580 ($B = 14$)	$62B+24 = 892$ 1102 for fully pipelined case	2472 fully pipelined: 2682 ($B = 14$)
Two-stage cascade filter order = 62 [22]	11	56	67	$623+67B$ [22] $=1427$ ($B = 12$)	Our estimate: > 1350 ($B = 12$)	Our estimate: >2777 ($B = 12$)
Single-stage design filter order = 62 [21]	17	56	73	$666+73B$ [22] $= 1542$ ($B = 12$)	Our estimate: > 1350 ($B = 12$)	Our estimate: >2892 ($B = 12$)
MILP method [21, 27] filter order = 62	17	62	79	Our estimate: > 1550 ($B = 12$)	Our estimate: > 1400 ($B = 12$)	Our estimate: >2950 ($B = 12$)
FIRGAM [2] filter order = 62	18	62	80	Our estimate: > 1550 ($B = 12$) 1213 if $B = 8$ [2]	Our estimate: > 1400 ($B = 12$) 1154 if $B = 8$ [2]	Our estimate: >2950 ($B = 12$)
PMILP [25, 2] filter order = 62	19	62	81	Our estimate: 1609 ($B = 12$) 1285 if $B = 8$ [2]	Our estimate: > 1430 ($B = 12$) 1182 if $B = 8$ [2]	Our estimate: >3039 ($B = 12$)
LMS method [8, 2] filter order = 62	23	62	85	Our estimate: 1651 ($B = 12$) 1311 if $B = 8$ [2]	Our estimate: > 1400 ($B = 12$) 1154 if $B = 8$ [2]	Our estimate: >3051 ($B = 12$)
Remez [2] filter order = 62	30	62	92	Our estimate: > 1800 ($B = 12$) 1510 if $B = 8$ [2]	Our estimate: > 1530 ($B = 12$) 1281 if $B = 8$ [2]	Our estimate: >3330 ($B = 12$)

MA multiplier adders, SA structural adders

The resulting optimally factored IFIR structure for Filter L2 is shown in Fig. 22, and its magnitude plot is shown in Fig. 23. This filter's stopband behavior exceeds specifications, especially at mid and high frequencies. Its peak-to-peak passband ripple is 0.395 dB, compared to the target 0.48 dB. The frequency responses for each of the 13 jointly sequenced stages are shown in Fig. 24. Given the target stopband attenuation of $20\log_{10}(\delta_s) = 60$ dB, it can be shown that the truncation level (i.e., wordlength) B should be at least 14 bits (including sign bit) for a practical realization of Fig. 22 13-stage factored IFIR filter. For a conventional single-stage design that meets the specification of Example 3, a 12-bit (including sign bit) wordlength would suffice.

Fig. 23 Magnitude plots of Example 3 (Fig. 22) optimally factored IFIR filter (solid) versus order-62 filter [8] (dash)

Fig. 24 Magnitude plots of the 13 stages in Fig. 22 (Example 3)

To illustrate the effectiveness of the Fig. 22 stage sequencing, we use the following comprehensive tests. We then measure the output RMS values of all cascade stages.

Test 1) Input signal is an ensemble of 50 random-phase in-band sinusoids ($\omega \leq 0.2\pi$). We expect the signal to traverse the factored filter unaffected, and the output to be a delayed version of the input.

Test 2) Input signal is white Gaussian noise (uniform power across all frequencies). We expect to attenuate the portion of the signal that falls within the stopband ($\omega \geq 0.28\pi$) by 60 dB.

Test 3) Input signal is colored Gaussian noise with uniform power only in the stopband. We realize this using a sum of 100 random-phase sinusoids, uniformly distributed in the stopband ($\omega \geq 0.28\pi$). We expect our filter to attenuate the entire signal by at least 60 dB.

Test 4) Input signal is a sinusoid at $\omega_p = 0.2\pi$ passband edge.

Test 5) Input signal is a sinusoid at $\omega_s = 0.28\pi$ stopband edge.

The results of the tests and corresponding signal RMS values at the outputs of all stages (normalized to the input signal RMS) are illustrated in Fig. 25, and we see that the RMS level at a few stage outputs increases above the input signal RMS level, requiring slightly larger dynamic range at the output of these few stages. This shows that if a uniform wordlength is desired (for design simplicity) then B should be 15 bits (including sign bit). The one extra bit accommodates the aforementioned RMS increases shown in Fig. 25 plots.

A slightly more efficient realization is also possible, employing the inherent flexibility of the factored structure which can (as mentioned for Example 2) accommodate a non-uniform datapath wordlength (i.e., truncation/rounding levels) throughout the cascade. According to Fig. 25, while 15 bits are needed for truncation at the outputs of stages #1, #2, #3, #10, #11, #12 and #13 (to accommodate up to a 6-dB increase in the stage-output RMS values, compared to the RMS of the filter input), only 14 bits are needed for truncation at the outputs of stages #4, #5, #6, #7, #8, and #9.

A summary of hardware complexity and a comparison with the previously reported methods of implementing this order-62 L2 filter are given in Table 6, and it is evident that the optimally factored IFIR filter has the lowest complexity. The complexity reduction, relative to Remez, can be seen as:

$$(3330 - 1870)/3330 \approx 44\%$$

or, when fully pipelined, as: $(3330 - 2044)/3330 \approx 39\%$.

Noise analysis for the factored IFIR structure in Fig. 22:

We now examine the noise performance of the Fig. 22 optimally factored IFIR structure. The truncation (or rounding) event at each output of the 13 cascaded stages injects quantization noise into the datapath. These truncation events can be approximately modeled by 13 independent and identically distributed additive uniform noise sources at the stage outputs. Figure 26 shows the effective total magnitude response that each of the 13 noise sources experiences from the point of truncation (noise generation) to the output of Fig. 22 factored IFIR structure. It confirms that none of the 13 noise sources experiences considerable out-of-band noise amplification, compared to the in-band signal power level. The overall effect of truncation noise from all cascade stages at the Fig. 22 filter output is illustrated in Fig. 27. Its bottom two plots are a histogram and a normalized PSD plot of the total noise at the Fig. 22 output (taking into account the contribution of all noise sources). The top plot shows the RMS of the total noise at the output of each stage. It demonstrates that the overall output noise in

Fig. 25 RMS of stage outputs in Fig. 22 for test scenarios (Example 3)

Fig. 26 Effective frequency response that each of the 13 noise sources experiences from the point of truncation (at the output of each stage) to the final output in the factored IFIR structure in Fig. 22

Fig. 27 Overall effect of truncation noise at the output of Fig. 22 optimally factored IFIR structure, taking into account the noise contributions from all stages (top plot: RMS of stage outputs, bottom: histogram and PSD)

the stopband is well below the target – 60 dB stopband level that the filter is required to realize.

5 Conclusion

In this paper, an apparently quite superior general method, and a corresponding structure, for achieving significantly more hardware-efficient implementations of FIR filters has been presented. This advancement employs our recently announced “optimal factoring of FIR filters.” We have demonstrated that by applying optimal factoring to well-designed *IFIR* filters we can implement much better (more hardware-efficient) FIR digital filters. When assessing hardware cost as the sum of the required full adders and flip-flops, we have demonstrated that such optimally factored IFIR filters can provide *substantially lower* hardware cost than that achieved by the methods presented in previous research publications. (Two of our examples show hardware reductions in the vicinity of 50%, in comparison to conventional Remez implementations. Indeed, the recent publication [15] shows these results to be quite close to a new “lower bound” for the hardware complexity of *any* FIR implementation that meets the specifications of these two FIR filters.) As shown in Table 3, our optimally factored IFIR filters can be particularly beneficial when specifications that push the technology speed limits are required, and in these cases the area and power savings for our optimally factored IFIR filters still appear quite substantial. Further properties, benefits, and alternative implementations of these filters have also been demonstrated when implementing well-known examples. This further confirms the utility of the optimally factored IFIR filters in comparison with more conventional implementations.

An extension of this paper’s optimal factoring of IFIR filters to the optimal factoring of FRM (frequency response masking) filters is also evident. (Please see [6] for FRM details.) Basically, the FRM structure is an extension of the IFIR structure which includes additional FIR-type hardware (for the purpose of facilitating a broader class of FIR filters, including certain highpass and bandpass FIR filters whose direct implementation via an IFIR structure could seem problematic). In Figs. 3(a) and 5 of [6] it is shown that one can start with an IFIR structure and include two more FIR blocks to obtain an FRM filter implementation that may seem more suited for some desired filters, primarily bandpass FIR structures. While certain complications may arise when attempting to implement the FIR factoring efficiently in an FRM filter (i.e., one basic issue could concern a desire to preserve a pure delay chain z^{-L_n} with what could have a substantial length L , and which may thus seem inconsistent with FIR factoring), it can still be envisioned that the type of FIR factoring that we have presented here could be extended to FRM filters. This would, of course, be a possible topic for future research.

Open Access This article is distributed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits unrestricted use, distribution, and reproduction in any medium, provided you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

Appendix

We now demonstrate the Benefit mentioned in Sect. 4 which *often* seems to be available when using our optimally factored IFIR filters. It comes about due to the relatively small size (low degree) of our individual IFIR factors. (A feature that is *often* present.) Here, we shall reduce the number of add operations required in the implementation of the MA and SA for Factors 5 and 6 of Example 3 (using data shown in Table 4).

Regarding Factor 5, it is evident that one adder (or one subtraction) would suffice for realizing the first coefficient

$$0.11111 = 1.0000\bar{1}, \quad \text{where } \bar{1} \Rightarrow -1$$

and we could also build the second coefficient (0.00101) with one adder.

It would seem, however, that two adders are needed for the third coefficient: $1.00111 = 1.0100\bar{1}$. Factor 5 thus seems to need four MA which, when added to the factor's seven SA, suggests that a total of 11 additions must be performed for Factor 5. Figure 28a shows what we may call the “conventional” realization of Factor 5.

Actually Factor 5 can be built with just two MA, but (although we somewhat blur the distinction between MA and SA) we must increase the SA count by one, thus saving a net total of one adder for making Factor 5.

Our more efficient implementation, in Fig. 28b, allows us to share multiplier hardware between two filter-taps, even though the taps operate on different data. (Notice that one non-trivial tap processes data delayed by z^{-3} and z^{-4} while another processes data delayed by z^{-2} and z^{-5} , while a third processes data delayed by z^{-1} and z^{-6} . Nonetheless, as shown at the top of Fig. 28b, we can easily implement the rightmost

Fig. 28 **a** Conventional realization of Factor 5 (Table 4). **b** More efficient implementation of Fig. 28a filter

Fig. 29 Demonstration of simple improvements for Factor 6 (Table 4)

Fig. 28a tap in the form of two simpler taps, operating in parallel, with their outputs added together. The cost of this implementation is just one MA, but the addition operation required then restores the total cost to two adders. However, notice that the first of these two parallel paths employs a multiplier that is the same as that needed for the leftmost of the three Fig. 28a non-trivial taps. Thus, we may improve our efficiency by using this leftmost tap to process not only the data for that tap but also provide one of the partial paths for the rightmost tap. As shown in Fig. 28b, this can result in a *net saving of one addition* for the complete implementation of Factor 5 (10 adds rather than 11 adds). We actually have increased by one the number of SA in our modified structure (from seven to eight), but there is a net saving because the MA needed for Factor 5 is reduced by two (from four down to two).

Figure 29 shows that we can similarly save one addition in an improved implementation of Factor 6.¹

References

1. J.W. Adams, A.N. Willson Jr., A new approach to FIR digital filters with fewer multipliers and reduced sensitivity. *IEEE Trans. Circuits Syst.* **30**(5), 277–283 (1983)
2. M. Aktan, A. Yurdakul, G. Dündar, An algorithm for the design of low-power hardware-efficient FIR filters. *IEEE Trans. Circuits Syst. I Regul. Pap.* **55**(6), 1536–1545 (2008)
3. D.S.K. Chan, L.R. Rabiner, An algorithm for minimizing roundoff noise in cascade realizations of finite impulse response digital filters. *Bell Syst. Tech. J.* **52**(3), 347–385 (1973)
4. A.G. Dempster, M.D. Macleod, Use of minimum-adder multiplier blocks in FIR digital filters. *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.* **42**(9), 569–577 (1995)
5. M. Faust, C. H. Chang, Optimization of structural adders in fixed coefficient transposed direct form FIR filters, in *Proceedings of the IEEE International Symposium on Circuits and Systems* (2009), pp. 2185–2188
6. Y.C. Lim, Frequency–response masking approach for the synthesis of sharp linear phase digital filters. *IEEE Trans. Circuits Syst.* **33**(4), 357–364 (1986)
7. Y.C. Lim, Y. Lian, The optimum design of one- and two-dimensional FIR filters using the frequency response masking technique. *IEEE Trans. Circuits Syst. II* **40**(2), 88–95 (1993)

¹ This paper's optimally factored IFIR structure and the design methods for finding optimal factors, scaling and sequencing them are patent pending.

8. Y.C. Lim, S.R. Parker, Discrete coefficient FIR digital filter design based upon an LMS criteria. *IEEE Trans. Circuits Syst.* **30**(10), 723–739 (1983)
9. J.H. McClellan, T.W. Parks, L.R. Rabiner, A computer program for designing optimum FIR linear phase digital filters. *IEEE Trans. Audio Electroacoust.* **21**(6), 506–526 (1973)
10. A. Mehrnia, M. Dai, A.N. Willson Jr., Efficient halfband FIR filter structures for RF and IF data converters. *IEEE Trans. Circuits Syst. II Express Briefs* **63**(1), 64–68 (2016)
11. A. Mehrnia, A. N. Willson, Jr., On optimal IFIR filter design, in *Proceedings of the IEEE International Symposium on Circuits and Systems*, vol 3 (2004), pp. 133–136
12. A. Mehrnia, A.N. Willson Jr., Optimal factoring of FIR filters. *IEEE Trans. Signal Process.* **63**(3), 647–661 (2015)
13. A. Mehrnia, A.N. Willson Jr., Further desensitized FIR halfband filters. *IEEE Trans. Circuits Syst. I Regul. Pap.* **62**(7), 1815–1824 (2015)
14. A. Mehrnia, A.N. Willson Jr., FIR filter design via extended optimal factoring. *IEEE Trans. Signal Process.* **64**(4), 1061–1075 (2016)
15. A. Mehrnia, A.N. Willson Jr., A lower bound for the hardware complexity of FIR filters. *IEEE Circuits Syst. Mag.* **18**(1), 10–28 (2018)
16. S. Nakamura, S.K. Mitra, Design of FIR digital filters using tapped cascaded FIR subfilters. *Circuits Syst. Signal Process.* **1**(1), 43–56 (1982)
17. Y. Neuvo, C.-Y. Dong, S.K. Mitra, Interpolated finite impulse response filters. *IEEE Trans. Acoust. Speech Signal Process.* **32**(3), 563–570 (1984)
18. H. Samueli, An improved search algorithm for the design of multiplierless FIR filters with powers-of-two coefficients. *IEEE Trans. Circuits Syst.* **36**(7), 1044–1047 (1989)
19. T. Saramaki, Y. Neuvo, S.K. Mitra, Design of computationally efficient interpolated FIR filters. *IEEE Trans. Circuits Syst.* **35**(4), 70–88 (1988)
20. W. Schüssler, On structures for nonrecursive digital filters. *Arch. Elek. Übertragung* **26**(6), 255–258 (1972)
21. D. Shi, Y.J. Yu, Design of linear phase FIR filters with high probability of achieving minimum number of adders. *IEEE Trans. Circuits Syst. I Regul. Pap.* **58**(1), 126–136 (2011)
22. D. Shi, Y.J. Yu, Design of discrete-valued linear phase FIR filters in cascade form. *IEEE Trans. Circuits Syst. I Regul. Pap.* **58**(7), 1627–1636 (2011)
23. P.P. Vaidyanathan, G. Beitman, On prefilters for digital FIR filter design. *IEEE Trans. Circuits Syst.* **32**(5), 494–499 (1985)
24. A.N. Willson Jr., Desensitized halfband filters. *IEEE Trans. Circuits Syst. I* **57**(1), 152–165 (2010)
25. C.-Y. Yao, C.-J. Chien, A partial MILP algorithm for the design of linear phase FIR filters with SPT coefficients. *IEICE Trans. Fundam.* **E85-A**, 2302–2310 (2002)
26. W.B. Ye, Y.J. Yu, Single-stage and cascade design of high order multiplierless linear phase FIR filters using genetic algorithm. *IEEE Trans. Circuits Syst. I Regul. Pap.* **60**(11), 2987–2997 (2013)
27. Y.J. Yu, Y.C. Lim, Design of linear phase FIR filters in subexpression space using mixed integer linear programming. *IEEE Trans. Circuits Syst. I Regul. Pap.* **54**(10), 2330–2338 (2007)