This paper presents the design and implementation of high-speed, multiplierless, arbitrary bandwidth sharp FIR filters based on frequency-response masking (FRM) technique. The FRM filter structure has been modified to improve the throughput rate by replacing long band-edge shaping filter in the original FRM approach with two to three cascaded short filters. The proposed structure is suitable for FPGA as well as VLSI implementation for sharp digital FIR filters. It is shown by an example that a near 200-tap equivalent Remez FIR filter can be implemented in a single Xilinx XC4044XLA device that operates at sampling frequency of 5.5 MHz.

Keywords: FIR filter design; high-speed low-power filters.

1. Introduction

With the advance in VLSI technologies, digital signal processing techniques are being applied to wireless communication systems, digital TVs, and multimedia systems. One of the building blocks for digital signal processing is the linear phase FIR filter. The FIR filter possesses some very desirable features like guaranteed stability, free from phase distortion, and low coefficient sensitivity. The drawbacks for the FIR filter are its high implementation cost due to the large number of arithmetic operations needed for each sample, large storage for input samples and coefficients, and high round-off error. For years considerable attention and effort have been placed on reducing implementation cost of FIR filters. One of the most computationally efficient ways to design arbitrary bandwidth sharp FIR filter is frequency-response masking techniques.\textsuperscript{1–16} The FRM technique utilizes a pair of interpolated band-edge shaping filter to form sharp transition-band and a pair of masking filter to remove the frequency repetitions caused by periodic band-edge shaping filter in the stopband. It was reported in Ref. 2 that the FRM approach is able to achieve more than 98% savings in the number of multipliers at the cost of
about 42% increase in group delay when synthesizes a very sharp FIR filter. This prompted us that FRM technique is a suitable candidate for VLSI implementation of high-speed sharp FIR filters as the number of multipliers is dramatically reduced. Another benefit for using FRM is the reduction in round-off error. It is a well-known fact that the longer is the filter, the higher is the round-off error. It is pointed out in Refs. 17–22 that round-off error caused by the filter coefficients can be greatly reduced if the coefficients are quantized into signed powers-of-two (SPOT) values. The quantization process requires a lot of computer resources and takes very long time to search for the optimal SPOT coefficients, especially for long filters. By synthesizing a sharp filter with FRM approach, a long filter is broken into three relative short subfilters. Hence, it is easy to quantize coefficients of each subfilter into SPOT value. With the help of FRM and SPOT techniques, we are able to implement a sharp FIR filter using a multiplierless architecture and achieve high throughput.

Traditionally, FIR filters are implemented as software solutions on dedicated DSP chips and usually limited to low speed applications. To achieve high performance, ASIC approach has to be used. With the extremely fast as well as impressive technological advance in VLSI technologies, more and more FIR filters have been implemented and fabricated into single chip. The advantages of VLSI implementation are high-speed, small die area, and low power dissipation if low power techniques are employed in the design. The disadvantages are high layout costs, delivery time, and limited flexibility in design changes. With recent advancements in Field Programmable Gate Array (FPGA) technology, FPGA becomes an alternative to implement real-time signal processing algorithms. The powerful synthesis tools provided by FPGA vendors together with FPGA’s regular array structure made it very attractive to be used in data path designs. Several techniques were reported in Refs. 24–28 for implementing digital FIR filters using FPGAs. These techniques are suitable for implementing short to medium length FIR filters. In this paper, a modified FRM structure is proposed for implementing multiplierless high-speed sharp FIR filters. This structure allows FIR filter with substantial high order to be implemented in a moderate-sized FPGA. An implementation of a near 200-tap equivalent Remez FIR filter with maximum permitted normalized peak ripple magnitude of 0.01 is presented to illustrate the design process. The filter with 10-bit signed 2’ complement input has been realized on a single Xilinx XC4044XLA device that operates at sampling frequency of 5.5 MHz.

This paper is organized as follows: In Sec. 2 the FRM technique is reviewed. A modification to the FRM approach has been presented in Sec. 3 to improve throughput and round-off error. A discussion on design procedure is followed. In Sec. 4, an FPGA implementation scheme is proposed to maximize the utilization of the FPGA resources. An example is included to illustrate the proposed techniques. Finally, the conclusions are drawn.
2. Frequency-Response Masking Techniques

It has been reported in several recent publications\(^{1-16}\) that the FRM technique is one of the most computationally efficient methods for the synthesis of sharp FIR filters with arbitrary bandwidth. The FRM technique employed a delay-complementary concept to realize a sharp FIR filter using four subfilters, i.e., a pair of complementary band-edge shaping filter \(H_a(z)\) and \(H_c(z)\), and two masking filters \(H_{Ma}(z)\) and \(H_{Mc}(z)\). Given a prototype symmetrical impulse response linear phase low-pass filter \(H_a(z)\) of odd length \(N_a\), its complementary filter \(H_c(z)\) can be expressed as

\[
H_c(z) = z^{-(N_a-1)/2} - H_a(z) .
\]  

Replacing each delay elements of both \(H_a(z)\) and \(H_c(z)\) by \(M\) delays, a pair of periodic band-edge shaping filter \(H_a(z^M)\) and \(H_c(z^M)\) are formed. The transition-band widths of \(H_a(z^M)\) and \(H_c(z^M)\) are a factor of \(M\) narrower than that of \(H_a(z)\).

In the frequency-response masking technique, two so-called masking filters \(H_{Ma}(z)\) and \(H_{Mc}(z)\) are cascaded to \(H_a(z^M)\) and \(H_c(z^M)\), respectively, to remove periodic repetitions of \(H_a(z^M)\) and \(H_c(z^M)\) in the stopband. The outputs of \(H_a(z^M)H_{Ma}(z)\) and \(H_c(z^M)H_{Mc}(z)\) are then summed to form \(H(z)\).

\[
H(z) = H_a(z^M)H_{Ma}(z) + H_c(z^M)H_{Mc}(z) .
\]  

Note that the group delay of \(H_{Ma}(z)\) and \(H_{Mc}(z)\) must be equal. The frequency responses of various subfilters are shown in Fig. 1, and one of the possible realization structures for frequency-response masking approach is shown in Fig. 2.

3. A Modified FRM Structure

Although frequency-response masking approach is computationally efficient in terms of the number of arithmetic operations and will lower the power consumption due to the sparse coefficients used in \(H_a(z)\), it suffers several drawbacks when implemented on an FPGA using direct-form. First, we noticed that the filter length of band-edge shaping filter \(H_a(z)\) can be much longer than that of \(H_{Ma}(z)\) and \(H_{Mc}(z)\) in a single stage frequency-response masking design. In some designs, the filter length of \(H_a(z)\) can be 2–4 times longer than that of \(H_{Ma}(z)\) or \(H_{Mc}(z)\). With the unmatched filter length among \(H_a(z)\), \(H_{Ma}(z)\), and \(H_{Mc}(z)\), the overall throughput depends on the longest filter in a direct-form implementation without employing pipelining. It is possible to balance the filter length of three subfilters by changing the interpolation factor. But the chosen interpolation factor may increase the overall filter complexity. This is not acceptable for an FRM design as the goal of using FRM is to achieve maximum savings in the number of arithmetic operations. Another way for increasing the throughput is to apply pipelining and parallel techniques to \(H_a(z)\), which requires a larger number of latches to accomplish the task. This will demand more FPGA resources as each CLB in an FPGA provides limited number of latches.\(^{29}\)
Second, each delay elements in $H_a(z)$ was replaced by $M$ delays. As a result, a large amount of storage elements are required in the overall design. For an FRM design, it normally requires 5-50% more storage elements than a Remez design.

Fig. 1. The frequency responses of various subfilters used in an FRM filter.
depending on filter’s specifications and wordlength requirement. For example, a 200-tap Remez filter with 10-bit input will need additional 400–800 D flip-flops if it is implemented by an FRM structure with each subfilter implemented in direct form. The additional number of CLBs will translate into high cost as large-size FPGA has to be used. To address these two problems, two approaches were proposed: (1) modify the frequency-response masking structure; (2) use memory blocks instead of D flip-flops to store the samples. With the new approach, we will show in Sec. 4 that a 200-tap Remez equivalent FIR filter can be implemented into a single Xilinx XC4044XLA device.

To make FRM structure effective for an FPGA implementation, let us first consider a way to improve the throughput of FRM filter without involving pipelining and parallel technique. Since the longest filter in an FRM approach is band-edge shaping filter $H_a(z)$, a reasonable conclusion is to factor $H_a(z)$ into several short filters. With the factorization, the length of each subfilter in an FRM approach is balanced. Hence, overall throughput is improved. The $z$-transform transfer function of the overall filter using modified FRM structure is given by:

$$ H(z) = [H_{Mc}(z) - H_{Ma}(z)] \prod_{i=1}^{p} H_{ai}(z^M) + H_{Mc}(z), $$

(3)

where $p$ is number of short filters factorized from $H_a(z)$. One of the possible realizations of the modified FRM structure is illustrated in Fig. 3, where $H_a(z)$ is factorized into two subfilters, $H_{a1}(z)$ and $H_{a2}(z)$. The factorization of band-edge shaping filter into several small filters will not only improve the throughput but also yield a design with better round-off noise property. The question is how to fac-

Fig. 2. A realization structure for frequency-response masking approach.

Fig. 3. One of the possible realization structures for modified frequency-response masking approach.
torize $H_a(z)$ such that the overall filter achieves best performance. Unfortunately, this is a very time consuming process for finding the optimum solution.\textsuperscript{31,32} From our experience, the maximum number of subfilters factorized from $H_a(z)$ should be less than or equal to three. This is because the synthesis of the overall filter will be more difficult when factorize $H_a(z)$ into four or more subfilters. If there is a need for more short filters, a multi-stage FRM approach should be considered. To further reduce the hardware complexity and increase the speed, the SPOT coefficient values should be employed. This will eliminate the need for a high-speed multiplier that takes up most of the hardware resources in an FPGA.

To synthesis a sharp FIR filter using the proposed FRM structure, we need to design the filter in two stages. In the first stage, the filter is synthesized using the original FRM approach\textsuperscript{1} with coefficients of band-edge shaping filter in infinite precision and two masking filter in SPOT values. In the second stage, the band-edge shaping filter $H_a(z)$ is factorized into two or three short filters and the coefficients of each short filter are quantized into SPOT values. A design procedure for design each subfilters in Fig. 3 is given below:

**Step 1.** Design band-edge shaping filter $H_a(z)$ with infinite precision coefficients and two masking filters $H_{Ma}(z)$ and $H_{Mc}(z)$ with SPOT coefficients using original frequency-response masking technique.

**Step 2.** Find zeros of $H_a(z)$ using method in Ref. 33 or MatLab and plot all zeros. From the plotting, separate the zeros into group of four and group of two, and number each group anti-clockwise around the unit circle starting from x-axis. Collect odd number group to form a short filter $H_{a1}(z)$ and even number group to form a short filter $H_{a2}(z)$.

**Step 3.** Design the $H_{a1}(z)$ in the discrete space using $H_{a2}(z)$ and $H_{Ma}(z)$ and $H_{Mc}(z)$ as prefilters. Let us denote the amplitude of $H_{a1}(e^{j\omega})$ as

$$A_{a1}(e^{jM\omega}) = \sum_i h_{a1}(n)\text{trig}(M\omega, i), \quad (4)$$

where \text{trig}(\omega, i) is a proper trigonometric function depending on the type of filter under consideration, $h_{a1}(n)$ is the impulse response of $H_{a1}(z)$. Applying the same definition to all subfilters and substituting Eq. (4) into Eq. (3) and let $p = 2$, we have

$$A(e^{j\omega}) = [A_{Ma}(e^{j\omega}) - A_{Mc}(e^{j\omega})] \times A_{a2}(e^{jM\omega}) \sum_i h_{a1}(n)\text{trig}(M\omega, i) + A_{Mc}(e^{j\omega}). \quad (5)$$

By evaluating Eq. (5) on a dense grid of frequencies, a set of inequalities as shown in Eqs. (6) and (7) is produced. In the passband, we have

$$1 - \delta_p(\omega) - A_{Mc}(e^{j\omega}) \leq [A_{Ma}(e^{j\omega}) - A_{Mc}(e^{j\omega})] \times A_{a2}(e^{jM\omega}) \sum_i h_{a1}(n)\text{trig}(M\omega, i) \leq 1 + \delta_p(\omega) - A_{Mc}(e^{j\omega}), \quad (6)$$
where $\delta_p(\omega)$ is the required passband ripple. In the stopband, we have
\[
-\delta_s(\omega) - A_{Mc}(e^{j\omega}) \leq \left[A_{Ma}(e^{j\omega}) - A_{Mc}(e^{j\omega})\right] \\
\times A_{a2}(e^{jM\omega}) \sum_i h_{a1}(n) \text{trig}(M\omega, i) \leq \delta_s(\omega) - A_{Mc}(e^{j\omega}),
\]
(7)
where $\delta_s(\omega)$ is the required stopband ripple. Linear programming\textsuperscript{18} or any other suitable techniques may be used to optimize Eqs. (6) and (7).

**Step 4.** Exchange the role of $H_{a1}(z)$ with $H_{a2}(z)$ in Step 3, and repeat Steps 3 and 4 until there is no improvement in the peak ripple. If the overall filter meets the given specification, stop. Otherwise increase the filter length of $H_a(z)$ and go back to Step 1.

### 4. The FPGA Implementation

We know that one of the subfilters, $H_a(z)$, needs excessive delay elements in the frequency-response masking approach. Although current FPGA generation has abundance of flip-flops, they come with a penalty. The larger is a device, the higher is the cost. In a Xilinx device, each CLB contains two flip-flops and three function generators (for XC4000X the function generators can be configured as latches)\textsuperscript{29} while two of the function generators in the CLB can be configured as 16X1 dual-port RAM. So it is more cost effective to use RAM to implement delay elements than use flip-flops. Figure 4 shows a block diagram to implement one of the subfilters in Fig. 3 which consists of data input and output, two dual-port RAM blocks to accommodate the input sample values, two programmable shifters and adders to perform the SPOT multiplication, a pipelined accumulator unit to get the final output, and a control logic unit to generate the data path control signals. The advantages of using this implementation scheme are three folds. First, it implements
delay elements by RAM blocks instead of flip-flops, which results great savings in the FPGA resources; second, it uses much less adders than a direct form implementation; third, it is programmable. The drawback is that the speed is depended on the number of filter taps. With the modified FRM structure, the speed of the filter can be increased by 2–3 times as the longest band-edge shaping filter has been factorized into 2–3 short filters. Most of the building blocks shown in Fig. 4 can be found in Xilinx’s Foundation Series design tools, which is a great help for shortening the design cycle.

To illustrate the proposed technique, let us consider the design of a narrow transition width low-pass filter with 10 bits input. The passband and stopband edges are at 0.2 and 0.21 sampling frequencies, respectively; the permitted maximum passband and stopband deviations are 0.01. The length of the Remez design meeting this set of specification with infinite precision coefficients is 195. Using the original frequency-response masking approach with three terms of 11 bits SPOT coefficients, the filter length of $H_a(z)$ is 53 with interpolation factor of 4. The filter length of $H_{Ma}(z)$, and $H_{Mc}(z)$ are 17 and 31, respectively, where $H_{Mc}(z)$ is a half-band filter. Each of them uses three terms of 10 bits SPOT coefficients. It is clear that $H_a(z)$ consists about three times more taps than the masking filters $H_{Ma}(z)$ and $H_{Mc}(z)$. Hence, $H_a(z)$ can be factorized into three short filters, $H_{a1}(z)$, $H_{a2}(z)$, and $H_{a3}(z)$. The filter length of $H_{a1}(z)$, $H_{a2}(z)$, and $H_{a3}(z)$ are 19, 19, and 19, respectively. The coefficients of $H_{a1}(z)$ use three terms of 11 bits SPOT, $H_{a2}(z)$ and $H_{a3}(z)$ use two terms of 9 bits SPOT, respectively. The frequency responses of $H_{a1}(z)$, $H_{a2}(z)$, $H_{a3}(z)$, and overall band-edge shaping filter $H(z)$ are shown in Fig. 5. The filter length of $H_{Ma}(z)$ and $H_{Mc}(z)$ are the same as in the original FRM,

![Fig. 5. The frequency responses of the band-edge shaping filter and three factorized subfilters.](image-url)
and their frequency responses are shown in Fig. 6. A comparison on FPGA resource utilization and operating speed between the original and modified FRM structures is listed in Table 1. Five building blocks as shown in Fig. 4 are needed together with additional two adders to perform the summations in Fig. 3. Additional RAM blocks are required to produce the complementary filter $H_c(z)$, and another seven delay components are needed to match the group delay between $H_{Ma}(z)$ and $H_{Mc}(z)$. The overall filter including five subfilters can be fitted into a Xilinx XC4044XLA device that operates at sampling frequency of 5.5 MHz. The frequency response of the overall filter is shown in Fig. 7.

From Table 1, the modified FRM structure increases the speed by 2-3 times of the original FRM while uses about 12% more resources than the original FRM. This is mainly due to the additional delay elements needed for generating the complement of band-edge shaping filter. Comparing to a direct-form implementation with D flip-flops as delay elements, the modified structure achieves about 74% savings in terms of the FPGA resources.
5. Conclusions

A modified frequency-response masking structure that is suitable for implementing sharp FIR filters in FPGA is presented. The success of the proposed structure is due to the use of cascading of short filters each with coefficients that are sum or difference of power-of-two terms to replace a long band-edge shaping filter in the original frequency-response masking approach. A near 200 taps equivalent Remez filter is realized using a single Xilinx XC4044XLA device that operates above 5 MHz.

References


Copyright of Journal of Circuits, Systems & Computers is the property of World Scientific Publishing Company and its content may not be copied or emailed to multiple sites or posted to a listserv without the copyright holder's express written permission. However, users may print, download, or email articles for individual use.