

# Efficient FPGA implementation of sharp FIR filters using the FRM technique

Shuguo Li<sup>a)</sup> and Jian Zhang

*Institute of Microelectronics, Tsinghua University, Beijing, 100084, China*

*a) lisg@tsinghua.edu.cn*

**Abstract:** A high-performance field programmable gate array (FPGA) implementation of full pipelined computation structure is proposed for sharp finite-impulse -response (FIR) filters using the frequency response masking (FRM) technique. The FRM-based FIR (FFIR) filter consists of a novel symmetrical systolic array of a interpolated FIR (IFIR) filter in cascade to a pair of nonsymmetrical systolic arrays of masking FIR filters mainly. These filters are designed based on inner-product computation involving MAC operation which can be realized by the DSP block in the latest FPGA device efficiently. The realization results on a Xilinx Virtex-5 chip show that the proposed FPGA implementation can obtain higher throughput but consumes less resource compared to the equivalent conventional sharp FIR (CSFIR) filter that developed by the Core Generator software tool.

**Keywords:** frequency response masking technique, FIR filter, field programmable gate array, systolic array

**Classification:** Integrated circuits

## References

- [1] Y. C. Lim, "Frequency-response masking approach for the synthesis of sharp linear phase digital filters," *IEEE Trans., Circuits Syst.*, vol. CAS-33, no. 4, pp. 357–364, April 1986.
- [2] L. Chen and K. P. Chan, "Design of two-dimensional sharp wideband filters using frequency response masking technique and frequency transformation," *Electron. Lett.*, vol. 45, no. 1, pp. 82–83, Jan. 2009.
- [3] Y. C. Lim, Y. Yu, H. Q. Zheng, and S. W. Foo, "FPGA implementation of digital filters synthesized using the FRM technique," *Circuits, Syst., Signal Process.*, vol. 22, no. 2, pp. 211–218, March/April 2003.
- [4] Y. Lian, "A modified frequency-response masking structure for high-speed FPGA implementation of sharp FIR filters," *J. Circuits, Syst. Comput.*, vol. 12, no. 5, pp. 643–654, Oct. 2003.
- [5] R. Mahesh and A. P. Vinod, "Reconfigurable frequency response masking filters for software radio channelization," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 55, no. 3, pp. 274–278, March, 2008.
- [6] R. Lyons, "Interpolated narrowband lowpass FIR filters," *IEEE Signal Process. Mag.*, vol. 20, no. 1, pp. 50–57, Jan. 2003.
- [7] Xilinx Inc., *XtremeDSP Design Considerations User Guide*, 2005.

- [8] J. G. Proakis and D. G. Manolakis, *Digital signal processing : principles, algorithms, and applications*, Prentice Hall, New Jersey, 2007.

## 1 Introduction

The frequency response masking (FRM) technique was developed for realization of finite-impulse-response (FIR) digital filters with very narrow transition bands [1, 2]. The technique utilizes a pair of interpolated band-edge shaping filters to form sharp transition-band and a pair of masking filters to remove the frequency repetitions caused by periodic band-edge shaping filter in the stopband. The major advantages of the FRM approach is that the filter employs an interpolated FIR filter (IFIR) with very sparse coefficient vector and requires low implementation complexity compared to the equivalent conventional sharp FIR (CSFIR) filters with the same frequency characteristic. It has been proven that the FRM technique is efficient for the design of very sharp digital filters with reduced implementation complexity compared to other options [1].

The field-programmable gate array (FPGA) technology is shown to be a very powerful approach to implement the FRM-based FIR filter (FFIR) filter [3, 4, 5]. The technique for reducing amount of coefficient transfer between the external memory and the FPGA is presented to improve the operating speed [3]. The long IFIR filter in the original FRM-based filter is replaced by two or three cascaded short filters to increase the throughput rate of the FPGA implementation [4]. A reconfigurable architecture is combined with the binary subexpression elimination technique to reduce the overall filter complexity [5]. All these structures, however, are not suitable for MAC-based implementation of the FIR filters and they don't make full use of pipelined and parallel characteristic of FPGA devices to obtain the feasible highest throughput.

## 2 Review of FRM technique

In this section, we provide a brief review of the sharp FIR filter based on the FRM technique. The basic idea behind the FRM technique is to compose the overall sharp transition-band filter using several wide transition-band filters [2]. Let  $H_a(z)$  denote a prototype  $N$ -tap linear-phase FIR filter. An interpolated FIR filter (IFIR)  $H_a(z^M)$  is obtained by replacing each unit delay of  $H_a(z)$  with  $M$ -unit delays ( $M$  is an integer). The complementary filter  $H_c(z^M)$  of the IFIR  $H_a(z^M)$  can be expressed as

$$H_c(z^M) = z^{-M(N-1)/2} - H_a(z^M) \quad (1)$$

So a tap delay line containing  $M(N-1)/2$  delay elements along with the succedent subtractor perform the same operation as the complementary filter  $H_c(z^M)$ . The structure of a filter using the FRM technique consists of the IFIR  $H_a(z^M)$ , a tap delay line, a subtractor and two linear phase masking

filters  $H_{ma}(z)$  and  $H_{mc}(z)$ . The IFIR  $H_a(z^M)$ , the tap delay line and the subtracter are used to form the sharp transition-band of the FFIR filter. Two masking filters are cascaded to the IFIR  $H_a(z^M)$  and the subtracter to remove periodic repetitions of filters  $H_a(z^M)$  and  $H_c(z^M)$  in the stopband respectively. The final outputs  $H(z)$  of FFIR filters is obtained by summing the outputs of two masking filters.

The transfer function of the overall FFIR filter is given by

$$H(z) = H_a(z^M) H_{ma}(z) + H_c(z^M) H_{mc}(z). \quad (2)$$

Thus the FRM filter with the transfer function  $H(z)$  has a transition band width  $N$  times less than the prototype filter  $H_a(z)$ .

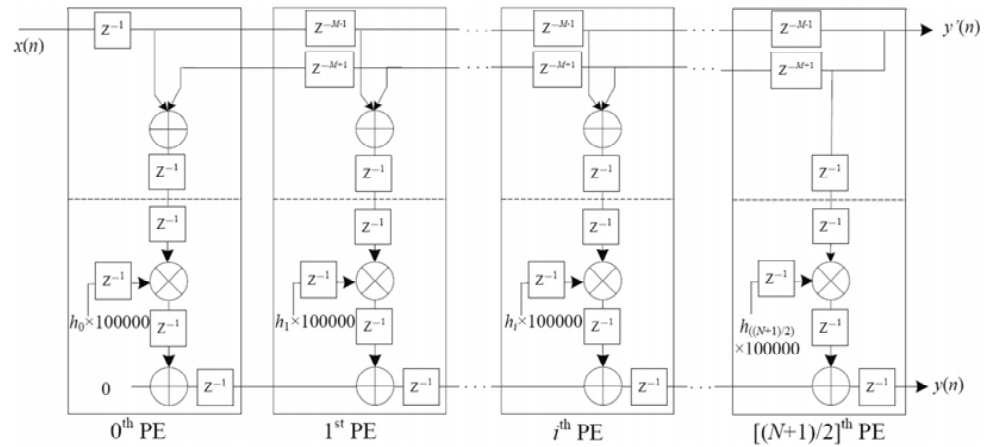
### 3 High-performance FPGA implementation of FFIR filters

From section 2, the FFIR filter consists of one IFIR filter and two masking FIR filters mainly. In this section, we propose a performance-cost efficient FPGA implementation of the symmetrical IFIR filter and introduce the non-symmetrical systolic array of the FIR filter. Further, the full pipelined architecture of FFIR filters is designed to obtain the high throughput of the sampling data.

#### 3.1 Symmetrical systolic array of IFIR filters

An IFIR filters is based on the behavior of an  $N$ -tap nonrecursive linear-phase FIR filter when each of its unit delay are replaced by  $M$ -unit delays, with the expansion factor  $M$  being an integer. In general, the architecture of an IFIR filter of length  $N$  requires  $N$  multiplications and  $(N-1)$  additions even [6]. To reduce the implementation complexity without loss of performance, we design a symmetrical systolic array of the IFIR filters with symmetrical coefficients as shown in Fig. 1. The architecture consists of  $(N+1)/2$  processing elements (PEs), each of which consists of delay elements, an upper adder, a multiplier and a lower adder. All PEs are divided into two parts by a dashed line: the upper part and the lower part and ordered from left to right with the first PE on the left.

In the upper part, the sampling data is fed to the leftmost PE (0th PE) and delayed for one cycle to yield  $x(n-1)$  by the 0th PE. The  $x(n-1)$  propagates toward the right passing one PE per  $(M+1)$  cycles until it reaches the rightmost PE ( $[(N+1)/2]$ th PE) to yield the data  $x(n-4-3M)$ . Then the data  $x(n-4-3M)$  propagates toward inversely the left passing one PE per  $(M-1)$  cycles until it reach the leftmost PE (0th PE). Thus every PE accepts two inputs and yields two outputs except the rightmost and leftmost ones. Two input data in the upper part are added and their sum is delayed and multiplied by the  $i$ th coefficient of the filter ( $i = 0, 1, \dots, (N+1)/2$ ) in the lower part. The product of the multiplier is delayed for one cycle and passed to the lower adder which perform the addition of the product and the result of the previous PE. The sum of the lower adder is delayed for one cycle and passed to the lower part of next PE. So there is a pipelined adder



**Fig. 1.** Symmetrical systolic array of the IFIR filter with the delay element  $z^{-M}$

chain acting as a data buffer to store previously calculated inner products in the lower part of PEs.

For the leftmost PE, the sampling data is pumped in continuously at the rate of one sampling data per cycle. It only outputs one delayed data to the next PE. In the rightmost PE, there are only one input port and one output port. Thus the rightmost PE doesn't need an upper adder.

An IFIR filter of length  $N$  implemented by the above systolic array requires only  $(N + 1)/2$  multiplications and approximately  $N$  additions. It reduces the number of multiplier by about 50% compared to the existing architecture [6]. A further benefit of the symmetric implementation is the reduction in latency, due to the adder chain being half the length  $N$ . Thus the symmetrical systolic array is formed to offer an efficient choice for the implementation of the IFIR filter.

Eq.3 and Eq.4 demonstrate how the sampling data are processed by the IFIR filter:

$$\begin{aligned}
 y(n) = & h_0 * \left( x \left( n - \frac{N+7}{2} \right) + x \left( n - \frac{N+7}{2} - (N-1) * M \right) \right) + \dots \\
 & + h_i * \left( x \left( n - \frac{N+7}{2} - i * M \right) + x \left( n - \frac{N+7}{2} - (N-1-i) * M \right) \right) + \dots \\
 & + h_{\left( \frac{N-3}{2} \right)} * \left( x \left( n - \frac{N+7}{2} - \frac{N-3}{2} * M \right) + x \left( n - \frac{N+7}{2} - \frac{N+1}{2} * M \right) \right) \\
 & + h_{\left( \frac{N-1}{2} \right)} * x \left( n - \frac{N+7}{2} - \frac{N-1}{2} * M \right)
 \end{aligned} \tag{3}$$

$$y'(n) = x \left( n - \frac{N+1}{2} - \frac{N-1}{2} * M \right). \tag{4}$$

where  $x(n)$  denotes the sampling data at sampling time  $n$ ; and  $y(n)$  and  $y'(n)$  denote the output data of IFIR filters at sampling time  $n$ ;  $h_i$  is the  $i$ th coefficients ( $i = 0, 1, \dots, N/2$ );  $M$  is the number of delay elements.

To achieve the full speed of the FPGA device, the delay elements and the upper adder in the upper part can be implemented by the LUT-FF pair efficiently and the lower part of every PE can be mapped to the DSP block which can be easily and efficiently chained together using dedicated routing between DSP blocks.

Truncating errors are another problem of the hardware implementation of filters. The filter coefficients produced by the MATLAB tool are decimal fractional coefficients. However, the hardware implementation of filters only can accept binary coefficients. Larger the bit number of the decimal fraction is, larger the truncating error is. Thus every decimal fractional coefficient is multiplied by a constant 100000 to reduce the bit number of the decimal fraction so that the truncating error can decrease evidently.

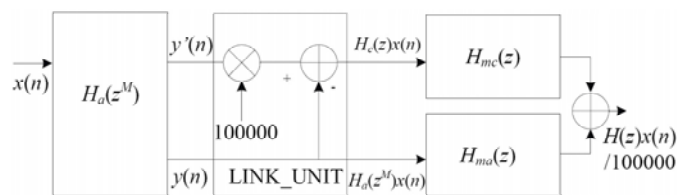
### 3.2 Nonsymmetrical systolic array of normal FIR filter

Symmetrical architecture of FIR filter must broadcast the delayed sampling data to every processing elements (PEs). Thus the high fanout circuit is formed and the feasible clock rate is reduced. To obtain the highest throughput, the nonsymmetrical systolic array is adopted to realize the masking FIR filter as shown in [7].

### 3.3 Overall FFIR filter architecture

Based on the symmetrical systolic array of IFIR filters and the nonsymmetrical systolic array of FIR filters, the overall FFIR filter architecture is designed as shown in Fig. 2.

In Fig. 2, the IFIR filter  $H_a(z^M)$  accepts the sampled input data  $x(n)$  and output the interim resulting data  $y(n)$  and  $y'(n)$ . Since the coefficients in the IFIR filter  $H_a(z^M)$  are multiplied by the constant 100000 to reduce the truncating errors, the data  $y'(n)$  should be multiplied by the same constant and subtracted by the data  $y(n)$  to form the outputs of the complementary filter  $H_c(z^M)$ . The multiplication along with the subtraction form the link unit between the IFIR filter and the two masking FIR filters. Whereafter the masking filters  $H_{mc}(z)$  and  $H_{ma}(z)$  accept the interim results produced by the link unit and produce two output data which are added together then divided by the constant 100000 to yield the final resulting data.



**Fig. 2.** The overall architecture of FFIR filter using the FRM technique

Without loss of generalization, suppose the prototype filter  $H_a(z)$  is of length  $N_a$ , every delay element in filter  $H_a(z)$  is replaced by  $M$  delay elements to form the IFIR filter, the masking FIR filter  $H_{ma}(z)$  and  $H_{mc}(z)$  are of length  $N_c$ . Then the IFIR filter requires  $(N_a + 1)/2$  DSP blocks and  $N_a$  shift-registers to delay the sampling data. The masking filter  $H_{ma}(z)$  and  $H_{mc}(z)$  both require  $N_c$  DSP blocks. The link unit can be implemented on a DSP block. Thus the FPGA implementation of the overall FFIR filter requires

$((N_a + 1)/2 + 1 + 2N_c)$  DSP blocks and  $N_a$  shift-registers. In all filters the full precision of the calculation is carried internally and is truncated to be 18-bit at the output because a restriction on the output is placed by the real output width of a DSP block.

#### 4 Experiment results and comparison

In this section, we compare the proposed FPGA implementation of the FFIR filter against the equivalent CSFIR filter that is developed by the Core Generator software tool. Various key performance metrics such as occupied resources, maximum usable throughput are estimated for different filter orders. The implemented FFIR filter is a network of a IFIR filter  $H_a(z^M)$  in cascade to two masking FIR filters  $H_{ma}(z)$  and  $H_{mc}(z)$  as shown in Fig. 2. Its equivalent CSFIR filter can be obtained by the linear convolution operation of coefficients of filters [8].

Without loss of generalization, suppose two FIR filters are both length 14; the prototype filter  $H_a(z)$  is of length 11 or 23; the number of delay elements are 4 or 8. All PEs are configured as shown in Fig. 1 or [7]. Then the equivalent CSFIR filters are of length 64, 104, 124, and 212 respectively. These FFIR filter and CSFIR filters are described by verilog program and implemented on the Xilinx Virtex-5 XC5VSX50T-3ff665 chip for minimum delay with ISE software tool. The implementation results are presented in Table I in terms of occupied resources and maximum usable throughput.

From Table I, the FFIR filter represents a saving of 33% and 65% in numbers of occupied slices and DSP blocks but obtains 14% higher throughput compared to the CSFIR in the case the filter tap number is 124. Thus the proposed implementation significantly outperforms the existing implementations in terms of the resources occupied and maximum usable throughput. The superior performance of the proposed architecture is due to the fact that the spare coefficients of the IFIR filters save much MAC operations for the FFIR filters as opposed to the CSFIR filters developed by the Core Generator software tool.

**Table I.** Implementation results for resources and throughput

Filter Parameter		Resource		Throughput(MSPS)	
FFIR	CSFIR	FFIR	CSFIR	FFIR	CSFIR
$M=4$ $N_a=11$	$TN=64$	129Slices +37DSP48Es	166Slices +64DSP48Es	550	550
$M=8$ $N_a=11$	$TN=104$	138Slices +37DSP48Es	336Slices +104DSP48Es	547	510
$M=4$ $N_a=23$	$TN=124$	210Slices +43DSP48Es	315Slices +124DSP48Es	543	476
$M=8$ $N_a=23$	$TN=212$	234Slices +43DSP48Es	619Slices +212DSP48Es	532	455

## 5 Conclusion

A full pipelined architecture of the FFIR filter is proposed that is suitable for implementing on the FPGA device containing the DSP blocks. The proposed architecture makes full use of the DSP blocks so that it requires lower complexity and less power but obtains higher throughput compared to the CSFIR filter that meets the same set of frequency response specifications. Moreover, the architecture also can be modified to realize other FRM-based filters and be extended to design sharp transition highpass, bandpass and bandstop filters. To the author's knowledge this is the first time that the useful combination of the FRM mechanization and special DSP blocks of FPGA have been presented.

## Acknowledgments

This work was supported by the National Natural Science foundation of China (no. 60476015, no. 60276016), national High-Tech Research and Development Program of China (no. 2006AA01Z418), and Ministry Commission Project of China (no. 20054500224).