International Journal of Computing and Digital Systems

http://dx.doi.org/10.12785/ijcds/020201

Performance Analysis of a Flexible, Optimized and Fully Configurable FPGA Architecture for Two-Channel Filter Banks

Anthony C. Karloff and Esam Abdel-Raheem

Dept. of Electrical and Computer Engineering, University of Windsor, Windsor, Ontario N9B 3P4, Canada

e-mail: karloff@uwindsor.ca, eraheem@uwindsor.ca

Received 23 Sep. 2012, Revised 9 Feb. 2013, Accepted 15 Mar. 2013

Abstract: This paper presents a fully configurable FPGA architecture for two-channel filter banks which enables rapid quantization error and hardware performance analysis. Lattice designs that eliminate the effects of quantization error do not necessarily exhibit linear phase and may result in excessive delay. This can make them ill-suited for applications such as digital audio. Thus, the effects of quantization on an optimized direct form FIR based filter bank are analyzed. This is accomplished by using a high-level, configurable architecture and parameter driven synthesis for varying coefficient and channel quantization, and filter types. Overall, the presented design targets high-speed optimization through a fully pipelined architecture that reduces complexity by uniquely multiplexing coefficients. This flexible architecture and its supporting tools have enabled rapid filter bank prototyping and analysis of the effects of quantization on performance that drastically reduces design time and cost for realization.

Keywords: FPGA; filter banks; signal processing.

I. INTRODUCTION

Filter banks have a variety of applications in digital signal processing including audio/image compression [1], sub band encoding [2] and adaptive systems [3]. Specifically, the quadrature mirror filter (QMF) is most commonly used to split a signal into separate bands, performed in the analysis bank. The simplest QMF bank is the two-channel filter bank, where the original signal is split into two bands at half the input frequency. The basic architecture for the two-channel filter bank, shown in Fig. 1, was reported on in [4, 5]. Its polyphase decomposition and transform using noble identities (shown in Fig. 2) is well known for reducing the complexity of the QMF. Under perfect reconstruction (PR) conditions, the two channels of the filter bank can be recombined via the synthesis bank to reproduce the original delayed signal with a constant gain. Perfect reconstruction and near perfect reconstruction two-channel filter banks can be achieved in a number of ways, most notably by: factorization, design of complementary filters, Lagrange multiplier approaches [6] and lattice structures [4, 7, 8].

The most basic implementation uses the direct form of the polyphase decomposition of the QMF bank finite impulse response (FIR) filters. This form suffers from high computational complexity since the number of multiplications increases with filter order. To reduce computational complexity, dynamically distributed arithmetic (DDA) implementations of the FIR filters have been proposed that replace the summing of product terms during filtering with a lookup table. However, the look-up table size increases dramatically with increased filter orders and is thus only efficient for low order filters. Field programmable gate array (FPGA) based architectures for DDA, direct form and a hybrid implementation, are reported in [9] and [10]. A variety of application specific integrated circuit (ASIC) lattice structures have been proposed with more recent designs and optimized implementations are reported in [11]. Other recent implementations are reported in [12, 13, 14].

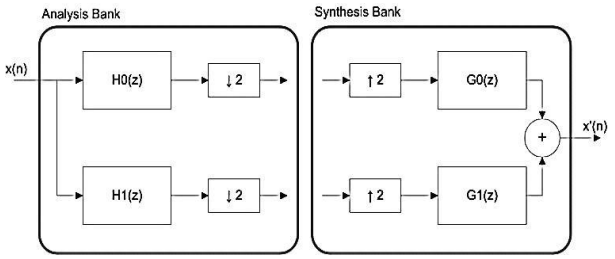

Fig. 1. Quadrature mirror filter bank

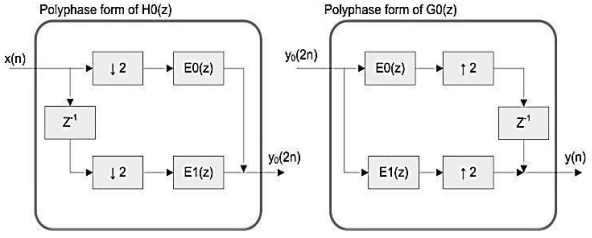

Fig. 2. Polyphase decomposition of QMF bank filters H0 and G0.

With the exception of some lattice realizations, which are not always suitable for applications that require linear phase and low-delay filters (e.g. [15]), the two-channel filter bank is susceptible to quantization error. This error can have a significant effect on the response of the filters and perfect reconstruction. Quantization error can be reduced by using higher bit representations of filter coefficients at the expense of increased resources and reduced performance. However, this is not always easily addressable for the variety of QMF bank structures that can be realized with different hardware optimization methods. Thus, one of the main challenges in implementing such filter banks is being able to rapidly evaluate the cost, performance and effects of quantization that result from hardware implementation. For this reason, a fully configurable architecture is proposed that automatically generates a low-complexity, high-performance system based on reconfigurable system parameters that include filter order and coefficients, bit representation for input data, coefficients, channels and output. The proposed flexible VHDL model is fully capable of automatically synthesizing any FIR based two-channel filter bank using these parameters. This VHDL model for two-channel filter banks currently targets FPGA devices, but has the potential to extend to M-channel filter banks and ASIC implementation.

As hardware multipliers are becoming faster and readily available in FPGAs, the proposed architecture revisits the direct form implementation with optimizations to reduce size and complexity compared to existing hardware designs. The architecture employs a standard cell array, fully pipelined architecture and reduces the number of required multipliers by exploiting the multi-rate nature of the filter bank as well as any symmetry in the filter coefficients (e.g. linear phase filters.) A single high-level FIR building block is used to implement the FIR filters in an optimized direct form. Multipliers share filter coefficients to take advantage of extra clock cycles as a result of the difference in high and low rate clocking schemes in both the analysis and synthesis banks, as well as an input folded path that exploits symmetric coefficients as common factors to delayed inputs. The result is a single, flexible architecture that can implement any FIR based two-channel filter bank in a compact and high-speed design that reduces the number of multiplications by up to a factor of four. The reduction in multipliers allows for higher order filters to be implemented without an excessive increase in memory and resources characteristic of DDA methods, and the fully pipelined results of these multipliers allow for high speed operation of high bit representations comparable to lattice structures and ASIC designs. This optimized direct form structure is later examined in further detail and compared to alternative forms such as DDA QMF bank implementation [10] and its hybrid equivalent [16], as well as recent FPGA implementations of wavelet filters [17].

The main benefit of such a flexible, configurable system in terms of cost and time for implementation is that a wide range of different filters and levels of quantization can be quickly compared without having to spend significant time focusing on the hardware implementation and the performance trade-offs of each system. In addition, a more resource efficient implementation can permit greater bit representations to reduce the effects of quantization. This paper first discusses the proposed architecture and how the various properties of the FIR filters that compose the QMF bank can be exploited to form a low-complexity, high-performance system. Next, various example filter banks are synthesized to demonstrate the flexibility of the reconfigurable system with respect to bit representation for filter coefficient and data quantization. Finally, results for quantization error and performance from various synthesized filter banks are presented for comparison as well as a complexity comparison of the proposed architecture and similar FPGA and hardware targeted implementations.

II. Analysis of Filter Implementation and Optimization

A brief analysis of each filter case (even/odd order filters that are symmetric, anti-symmetric and non symmetric) for both the analysis and synthesis banks was performed to determine the best manner for implementing the desired FIR filters in hardware. The most common simplification is the polyphase decomposition of each filter, requiring N/2 multipliers for each filter bank [5]. Further savings can be made if symmetric coefficients are selected, however, the hardware implementation that can best exploit these

additional savings for every order and symmetry FIR filter case is not always straight forward. To address this, an investigation was performed to replace the polyphase decomposition of an FIR filter by exploiting a combination of extra clocks and filter symmetry while multiplexing filter coefficients. The result is a common structure that further reduces the number of required multipliers in both the analysis and synthesis banks for any order FIR filter with symmetric, anti-symmetric or non-symmetric coefficients.

A. Non-Symmetric FIR Optimization

Instead of decimating and interpolating filter input or output data to reduce the number of multipliers, the FIR filters are optimized by multiplexing coefficients to a multiplier on alternating clocks. By multiplexing even and odd coefficients into one multiplier, the optimized filter has a similar effect to the polyphase decomposition. The direct form block shown in Fig. 3 can be used to implement any FIR filter in a fully pipelined architecture where the partial sum (P SUM) cumulates successive even and odd clock outputs. For non-symmetric coefficients, this structure only increases multiplier savings in the filter bank by a factor of two, but for symmetric and anti-symmetric filter coefficients, the savings in multipliers increases further.

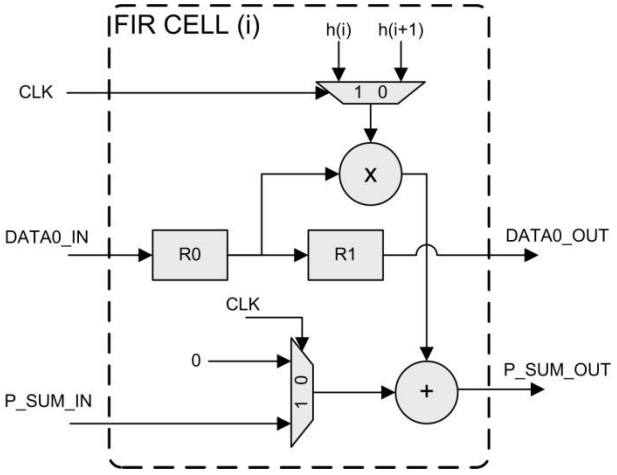

Fig. 3. Basic FIR cell with multiplexed coefficients and a partial sum (P SUM) input to cumulate successive even and odd output results. The basic FIR cell can be connected to pipelined adders for computing the sum of products.

B. Optimized Direct Form Symmetric FIR Implementation

When the coefficients of an FIR filter are either symmetric or anti-symmetric, the number of required multipliers can be further reduced by identifying common factors in the output and reusing previous multiplier outputs. One or both of these methods can be employed depending on the order of the desired filter, and whether it appears in the synthesis or analysis bank. To best demonstrate how this is possible, the polyphase form must be re-expressed. Using an analysis bank filter as an example, instead of splitting the input signal into even and odd samples, filter coefficients are divided into even and odd coefficients with neighboring pairs assigned to a single multiplier as:

$$M_i^{ev}(n) = h_{2i}x(n-2i) \quad (1)$$

$$M_i^{od}(n) = h_{2i+1}x(n-2i) \quad (2)$$

where, $M_i^{ev}$of Equation (1) denotes multiplication of the even coefficient of the i^{th} multiplier and $M_i^{od}$of Equation (2) denotes multiplication of the odd coefficient of the i^{th} multiplier. Input data can be streamed directly into a series of pipelined multipliers such that Equation (1) is performed on alternating clocks with Equation (2). Note that Equations (1) and (2) are not performed simultaneously.

On even clocks, when the even coefficients are multiplexed as multiplier inputs, the even result $y_0(n)$ is generated as the sum of products. On odd clocks, when the odd coefficients are multiplexed, the odd result $y_1(n)$is generated similarly. The result of even and odd outputs are summed in the partial sum block of each FIR cell shown in Fig. 3 and the result is generated at the appropriate clock depending on the position of the filter cell in either the analysis or synthesis bank. In this manner, the same multiplier architecture for the analysis and synthesis banks can be utilized to implement the optimized direct form structure, operating on the same input/output clock and channel half clock. Fig. 4 demonstrates the multiplier configuration for a 7th order symmetric or anti-symmetric FIR filter for both an analysis and synthesis FIR filter. Although seemingly incomplete, the correct output of this multiplier configuration depends on redundancies in either the multiplier results or input folded data that will be discussed in the next sections.

To determine the configuration of the multipliers for any order and symmetric form, an expression can be written for the equivalent optimized direct form. For example, the odd ordered, symmetric (or anti-symmetric) 7th order filter of Fig. 4 can express even and odd clock outputs as:

$$y_0(n) = M_0^{ev}(n) + M_1^{ev}(n) + M_1^{ev}(n-2) + M_0^{ev}(n-6) \quad (3)$$

$$y_1(n) = M_0^{od}(n) + M_1^{od}(n) + M_0^{od}(n-4) \quad (4)$$

where the complete output for the analysis and synthesis banks is expressed as:

$$y(n/2) = y_0(n) + y_1(n-1) \quad (5)$$

$$y(n) = \begin{cases} y_0(n/2) & n \in even \\ y_1\big((n+1)/2\big) & otherwise \end{cases} \quad (6)$$

In Equation (3) and Equation (4), the filter output is either a delayed and summed output of the two terms at half the input clock rate for the analysis bank as shown in

Equation (5), or an alternating switched output of the two terms at the normal clock rate for the synthesis bank as shown in Equation (6).

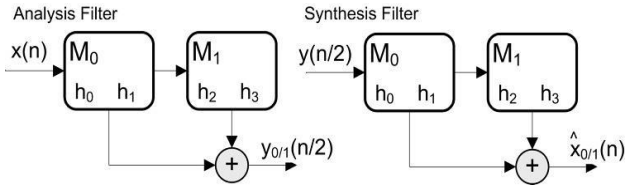

Fig. 4. Multiplier configurations for an analysis and synthesis 7th order symmetric FIR filter using two multipliers. In the analysis bank, coefficients multiplex at the same rate as the input data and add at half the rate. In the synthesis bank, coefficients are multiplexed and results added at twice the channel rate (same as the input data). Folded paths for complete filter implementation are omitted.

The outputs for an odd order symmetric filter generated on alternating clocks (shown in equation 3) can be expressed in a more general form for a Nth order filter as:

$$y_0(n) = \sum_{i=0}^{(m-r)/2} M_i^{ev}(n) \pm \sum_{i=0}^{\frac{(m-r)}{2}-(1-r)} M_i^{ev}(n-2m+4i) \quad (7)$$

$$y_1(n) = \sum_{i=0}^{\frac{(m-r)}{2}-(1-r)} M_i^{od}(n) \pm \sum_{i=0}^{(m-r)/2} M_i^{od}(n-2(m-1)+4i) \quad (8)$$

where $m = (N-1)/2$ indicates the center coefficient index and $r = m(mod\ 2)$ identifies an even or odd number of shared multipliers, where *N* indicates the filter length. Similarly, an even order symmetric filter can be described by:

$$y_0(n) = \sum_{i=0}^{\frac{(m-r)}{2}-(1-r)} M_i^{ev}(n) \pm \sum_{i=0}^{\frac{(m-r)}{2}-(1-r)} M_i^{ev}(n-2m+4i+1) \quad (9)$$

$$y_1(n) = \sum_{i=0}^{\frac{(m-r)}{2}-r} M_i^{od}(n) \pm \sum_{i=0}^{\frac{(m-r)}{2}-r} M_i^{od}(n-2m+4i+3) \quad (10)$$

with $m = N/2$ and $r = m(mod\ 2)$. The anti-symmetric variants of these two cases simply impose a subtraction of multiplier sums instead of addition. The main observation that can be made is that the total number of required multipliers is equal to $(m+r)/2$ since multiplier blocks are re-used as delayed outputs. These redundant, delayed occurrences omitted in Fig. 4 can be implemented in two manners: as either delayed results to be summed, or delayed inputs to be added before multiplication by a common coefficient. The first is quite apparent in Equation (3) and Equation (4). The later becomes apparent when the expansion of the multiplication term is expressed as:

$$M_0^{ev}(n) + M_0^{ev}(n-6) = h_0x(n) + h_0x(n-6) \quad (11)$$

C. Delayed Multiplier Outputs (DMO)

The main premise behind delayed multiplier outputs is that for filters with symmetric coefficients, the sum of products of future outputs often includes some of the products of the current output. In this case, it is not necessary to compute the same product twice, but to simply delay the existing products to be included in later sums. This is shown demonstratively for the cases of even and odd symmetric filters in both the analysis and synthesis banks shown in Fig. 5. In this tabular representation, each row represents a new clock cycle. On each clock, inputs are shifted to the right. Columns are labeled by multipliers and delayed multiplier outputs are shown in bold boxes with an arrow indicating the source of the output.

The problem that arises from this method is that even order symmetric filters in the analysis bank require that some of the delayed products be calculated on extra clocks, making it impossible to multiplex the coefficients.

D. Common Coefficient (CC)

Another way to eliminate a multiplication operation in a symmetric filter is by summing input values that share a coefficient in the output before performing multiplication. The multiplier expansion of Equation (11) can be expressed in general as:

$$Y_0(z) = \sum_{n=0}^{N/2-1} h(2n)\left[x(2n) + x\left(\frac{N}{2}-2n\right)\right]z^{-n} \quad (12)$$

$$Y_1(z) = \sum_{n=0}^{N/2-1} h(2n+1)\left[x(2n+1) + x\left(\frac{N}{2} - 2n + 1\right)\right] z^{-n} \quad (13)$$

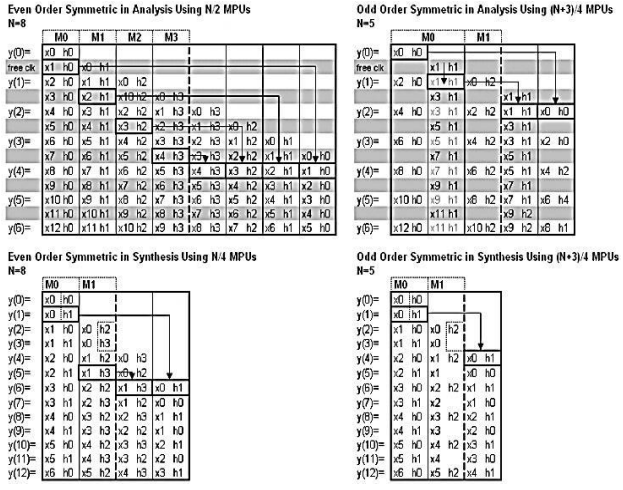

Fig. 5. Summary of DMO computations. Multipliers M0 and M1 multiplex coefficients on alternating clocks. Delayed products are boxed.

Equations (12) and (13) show how folded input can result in N/4 multipliers sharing N/2 unique coefficients. Again, $y_0(n)$ and $y_1(n)$ are computed on alternate clock cycles which permits this sharing, as previously discussed for Equation (7) and Equation (8). Examples of this can be observed in Fig. 6 where some common coefficients are highlighted in bold boxes.

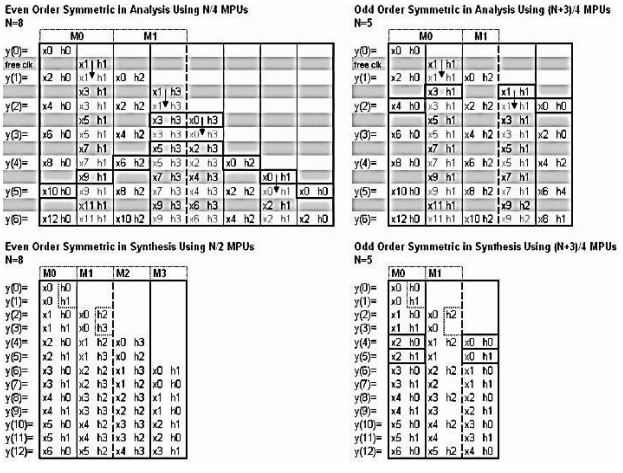

Fig. 6. Summary of CC computations. Multipliers M0 and M1 multiplex coefficients on alternating clocks. Terms with common coefficients are boxed.

This is extremely effective for all cases of odd order filters, but further minimizing even order synthesis filters fails since each coefficient of the symmetric filters appears once in every output on every clock cycle. In this case, the number of multiplications is still only reduced by a factor of two and cannot be reduced further for even symmetric filters in the synthesis bank.

E. Complexity Comparison

Since the DMO and CC methods of filter implementation fail to further optimize a different filter, a combination of these two methods can be used to reduce all cases of symmetric filters. To determine which method to use predominately, the complexity of implementing each method must be compared.

Since the CC method uses feedback of previous coefficients, N registers of the same bit width of the input data are required to store the feedback plus a number of output registers equal to:

$$nr \approx (N^2 - 2N)/8 \quad (14)$$

for fully pipelined computation of the sum.

Alternatively, to implement the DMO method, results from the multipliers must be stored with bit width equal to twice the input data width and require twice as many of these output registers when compared to CC, expressed as:

$$nr \approx N^2/4 \quad (15)$$

where Equation (14) and Equation (15) were derived from the multiplier structures that result from Equation (7) through Equation (10). As a result, it is far more efficient to implement the CC method, however, this does not mean that the DMO method cannot be used for an even symmetric filter in the synthesis bank. In fact, for this specific case, since the multiplexing of multiplier coefficients is achievable, the number of required delayed registers for the product outputs is slightly less than Equation (15), although still not comparable to the fully pipelined summation block required for implementing the CC method, seen in Equation (14).

F. DMO and CC Architectures

For CC, the cell shown in Fig. 3 was modified to include a folding path and extra adder before multiplication. This modification is shown in Fig. 7. The resulting cell can be arrayed to suit the desired filter order with a specific feedback path of the last cell configured according to the filter type being implemented. This new implementation can effectively replace the polyphase and direct form implementation as well as DDA methods with greater hardware savings. Fig. 8 shows the various foldings depending on the type of filter being implemented for analysis and synthesis bank filters, as determined by Equation (7) through Equation (10). The VHDL implementation of this proposed architecture can easily array the required number of such cells and add the

appropriate ending accordingly. Pipelined addition is then performed on the partial sum outputs.

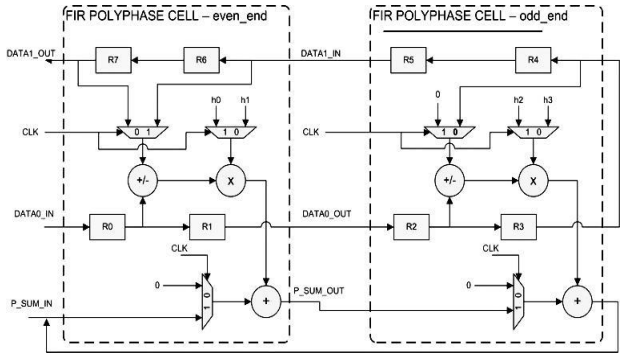

Fig. 7. Modified FIR cell and its variant. The new cell can accommodate both common factor feedback and multiplexed coefficients with a partial sum register for addition of products calculated on extra clocks.

The dominate advantage of this architecture is that any bit width can be utilized for the channel data and parameters of the FPGA synthesizer can optimize the data paths, multipliers and supporting logic to suit the targeted device. In this regard, the effects of data quantization on overall resource utilization and performance across a number of devices is not easy to directly determine, but it can now be quickly evaluated with an overall optimized structure.

For a more general and flexible architecture overall, the CC block was solely implemented. However, it should be noted that there are some features of the DMO block architecture that can be highly advantageous in certain PR filter bank designs. Specifically, the DMO had the unique feature that the same filter realization for the analysis block can be used for the synthesis block and vice versa under certain conditions. This effectively permits the half implementation of a filter bank that can perform both the functions of the analysis and synthesis banks with minimal supporting hardware. This is not investigated in further detail in this paper.

III. FPGA IMPLEMENTATION

The proposed architecture was implemented in VHDL using Xilinx ISE 10.1. Synthesis of the VHDL core targeted the Xilinx Virtex 5 for estimation of resource utilization and timing, however, it is flexible enough to accommodate any FPGA device with sufficient resources. In order to create configurable VHDL code, a library file was generated that contained the parameters of the filter bank, including bit representation, filter lengths, symmetries and quantized filter coefficient values. This library file was generated via a MATLAB interface, in which filters could be designed and converted into the appropriate fixed point representation. Arrangement of the filter structures shown in Fig. 8 were automatically generated based on filter length and symmetric properties.

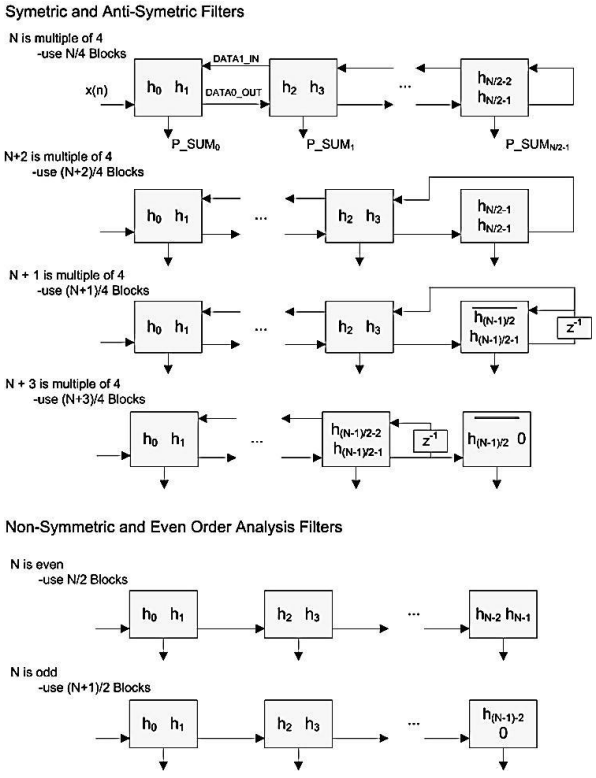

Fig. 8. Configurations of the modified FIR cell for filter bank implementations.

The VHDL module was organized into the following structure. The top level module utilized generic parameters to configure the data input, channel input/output, and output data bus widths accordingly. The top level filter bank module then created four instances of FIR filters for the analysis and synthesis bank filters using filter property parameters and coefficients specified in the library file (generated by MATLAB).

The next level module was the analysis FIR and synthesis FIR. This module instantiated and connected the FIR cell blocks using generic parameters and generate statements according to the configurations shown in Fig. 8. Depending on the filter lengths and symmetry, various generic parameters were passed to FIR cell blocks and the DATA x and P SUM folding were connected accordingly.

Generic parameters allowed for either of the two structures shown in Fig. 7 to be implemented with positive or negative input folded data depending on symmetry of the filter being implemented. Finally, the size of multipliers and adders was determined based on the specified coefficient and data channel bit representations, and implemented according to variable device synthesis parameters (ideally DSP multipliers).

A. Testing VHDL code

A VHDL testbench was developed to implement the top level module. The testbench read input data from a binary file and wrote the resulting output to another binary file. The test data was generated via MATLAB where the input data was represented in the appropriate

signed fixed point representation. The same MATLAB program read the resulting output data from the VHDL testbench and calculated the signal to noise ratio (SNR). Any arbitrary input data can be specified to test the effects of quantization on compression or perfect reconstruction.

B. Critical Path, Timing, and Latency

Since the filter bank operates in a fully pipelined architecture, critical timing is limited by the speed of the most complex operation; the multiplication of a signed fixed point number with a constant signed fixed point number. In this case, the maximum speed of the filter bank is affected by the bit length of the values being multiplied and the speed of the targeted FPGA device. For 16 bit signed fixed point numbers, the maximum operating frequency of the filter bank on a Virtex 5 is 200 MHz. For smaller bit representation, such as 12 bit signed fixed point, the maximum operating frequency can run as fast as 260 MHz. Another implication of the filter's fully pipelined nature is that latency is a function of filter length. Exact latency depends on the order and symmetry of each filter.

C. Quantization and Resource Utilization

Resource utilization is a function of both the filter length and the desired bit representation. It should be noted that the odd order filters dramatically reduce the overall complexity, because far less FIR cells were required to implement the desired filters in the CC architecture. Since a variety of parameters exist for synthesis in FPGA devices, it is difficult to accurately predict the effects different levels of quantization will have on the size of implementation. Having a flexible architecture that inherently optimizes the general structure can be very useful for fine tuning the size and performance requirements of implementing a two-channel filter bank for a specific application.

D. Signal and Noise

Signal to noise ratio (SNR) is independent of synthesis and a direct function of bit representation (assuming filter coefficients adhere to perfect reconstruction requirements.) SNR was measured using a ramp input to the filter bank and compared to the output with pertinent delay compensation as described in [6].

E. Complexity Comparison

To demonstrate the flexibility and advantages of the proposed architecture, a few applications of two-channel filter banks were compared to recent FPGA based architectures in literature. Table I shows a comparison with the Biorthogonal 9/7 Tap filter reported on in [17]. For the full CDF-9/7 implementation, the proposed architecture reduces the number of multipliers and significantly improves performance of the filter at the expense of larger implementation for speed optimized synthesis on a Virtex 5 device. Table II shows a comparison with two DDA methods that utilize look up tables in place of multipliers. The proposed architecture reduces size and increases speed for the targeted Virtex device even for a low order filter realization. Savings will continue to increase as filter orders increase.

TABLE I: COMPARISON WITH BIORTHOGONAL 9/7 TAP IMPLEMENTATION ON VIRTEX 5 DEVICE [17].

FEATURES	CDF-9/7 [17]	PROPOSED
MULTIPLIERS	16	10
COEFFICIENT BITS	7	7
SLICES (REGISTERS)	144	749
CLOCK FREQUENCY	106.98 MHZ	239.257 MHZ
RECONFIGURABLE	N	Y

TABLE II: COMPARISON WITH HYBRID DDA* [16] AND DDA [10] WITH 4 TAP FILTERS. *EXTRAPOLATED FROM SINGLE FIR RESULTS.

FEATURES	HYBRID DDA	DDA	PROPOSED
MULTIPLIERS	0	0	4
COEFFICIENT BITS	4	4	4
SLICES (REGISTERS)	584	364	223
CLOCK FREQUENCY	188 MHZ	75 MHZ	56.881 MHZ
RECONFIGURABLE	N	N	Y

IV. QUANTIZATION OF PERFECT RECONSTRUCTION TWO-CHANNEL FILTER BANKS

The main drawback of any variant of a direct form implementation or spectral factorization is that quantization of filter coefficients affects the perfect reconstruction condition. However, unlike lattice realizations, symmetry in filter coefficients will still guarantee linear phase response. The problem is being able to measure the effect of quantization on the filter bank's perfect reconstruction and weigh the compromise of performance for using higher bit representations in hardware realization of the system. This is now fast and simple, since the proposed architecture can easily implement any FIR based filter bank for any user specified quantization within the capacity of the target FPGA device for synthesis. To demonstrate, a few perfect reconstruction two-channel filter banks were synthesized for the Virtex 5 with varying levels of quantization in both the channels and the filter co-efficients.

A. Example PR Two-Channel Filter Banks in [6]

Two sets of perfect reconstruction Wilson filter coefficients [6] were implemented with various different bit representations (analysis bank input bits : channel bits : synthesis bank output bits) where filter coefficient quantization was matched to the input channel. The direct form complexity of the two different Wilson filters considered is shown in Table III. The results of each implementation, shown in Table IV, clearly shows the reduction in multipliers achievable with the proposed

architecture and how there is a greater reduction for odd order filters. In addition, the FPGA utilization in terms of slices is shown. Channel and output bit representation were selected to reduce truncation error so that the SNR was representative of filter quantization.

TABLE III: FILTER COMPLEXITY.

Filter Bank	Filter	N	Total QMF Multipliers
Example 2.1 [6]	H0	23	96
	H1	25	
Example 2.3 [6]	H0	16	88
	H1	28	

TABLE IV: RESULTS OF COEFFICIENT IMPLEMENTATION.

Filter Bank	MCUs	# Bits	SNR (dB)	Max. Clock	Slices
Ex.2.1 [6]	26	6:8:10	39.9792	250.815 MHz	1697
		8:12:16	86.0192	242.742 MHz	2507
		10:18:32	121.1571	238.960 MHz	3672
Ex. 2.3 [6]	33	6:8:10	46.6023	250.611 MHz	1735
		8:12:16	85.7358	241.838 MHz	2667
		10:18:32	142.1724	238.975 MHz	3985

Observations can now be made regarding the performance and size compromise for obtaining a higher SNR with higher bit representations. In this particular case, the synthesizer targeted performance, maximizing the operating frequency at the expense of resources. Thus, as bit representation increases, so do resources without significantly affecting the maximum operating frequency. Fig. 9 and Fig. 10 show two example outputs from the behavioral simulation of the proposed VHDL filter bank architecture from Table IV.

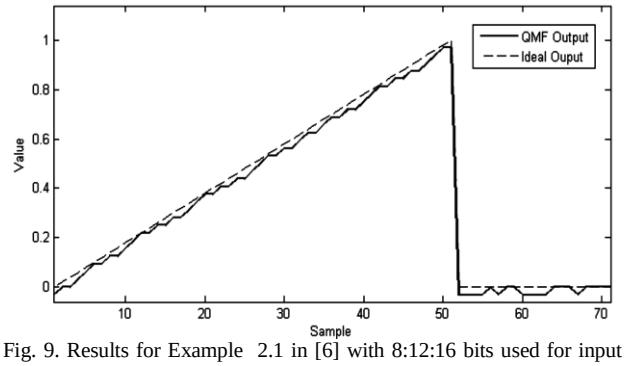

Fig. 9. Results for Example 2.1 in [6] with 8:12:16 bits used for input data, channel data and output data, respectively.

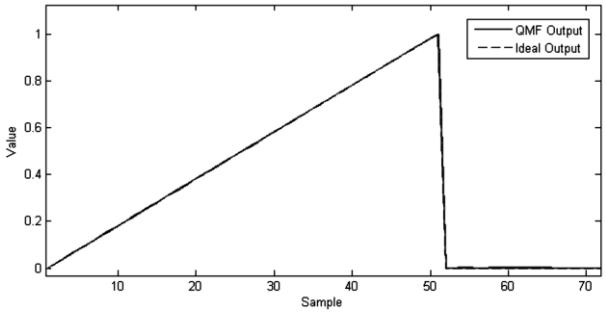

Fig. 10. Results for Example 2.3 in [6] with 10:18:32 bits used for input data, channel data and output data, respectively.

Finally, the timing diagram generated in ISE 11.1 for the implementation of Example 2.3 filter bank in [6] is shown in Fig. 11. Here, the registers for first FIR CELL of the H0 synthesis filter is shown in addition to the data input and Y1 and Y0 outputs. The m_reg0 register holds the current multiplier coefficient and the results of the delays and summed cumulated multiplier outputs are present in c_sum_in and c_sum_out.

V. CONCLUSION

This paper has shown the successful design and implementation of a FPGA architecture for two-channel filter banks in VHDL. The architecture of the VHDL module optimizes the complexity depending on the length and symmetry of the filters being used for the analysis and synthesis banks of the filter bank. In addition to the multiplier optimization, the design also has the flexibility to implement any bit representation for the input data, channel data and output data. The VHDL module and test bench have been tested with a variety of different two-channel filter banks including perfect reconstruction, DDA implementation of 4 tap filters and a CFM-9/7 filter, all showing improvement in size and/or speed. Test results for the two perfect reconstruction filters in [6] were implemented with various bit representations, showing the great reduction of multipliers made possible by multiplexing coefficients to exploit extra clocks and filter coefficient symmetry. The analysis of quantization and its effects on FPGA resource utilization, speed and SNR were examined for different synthesis parameters. In addition, with minimal work, the current two-channel architecture can be easily adapted for the M-channel case using the new optimized implementation of FIR filters. Overall, the proposed structure proved not only to be an ideal implementation for filter banks, but a valuable tool for rapid evaluation of quantization.

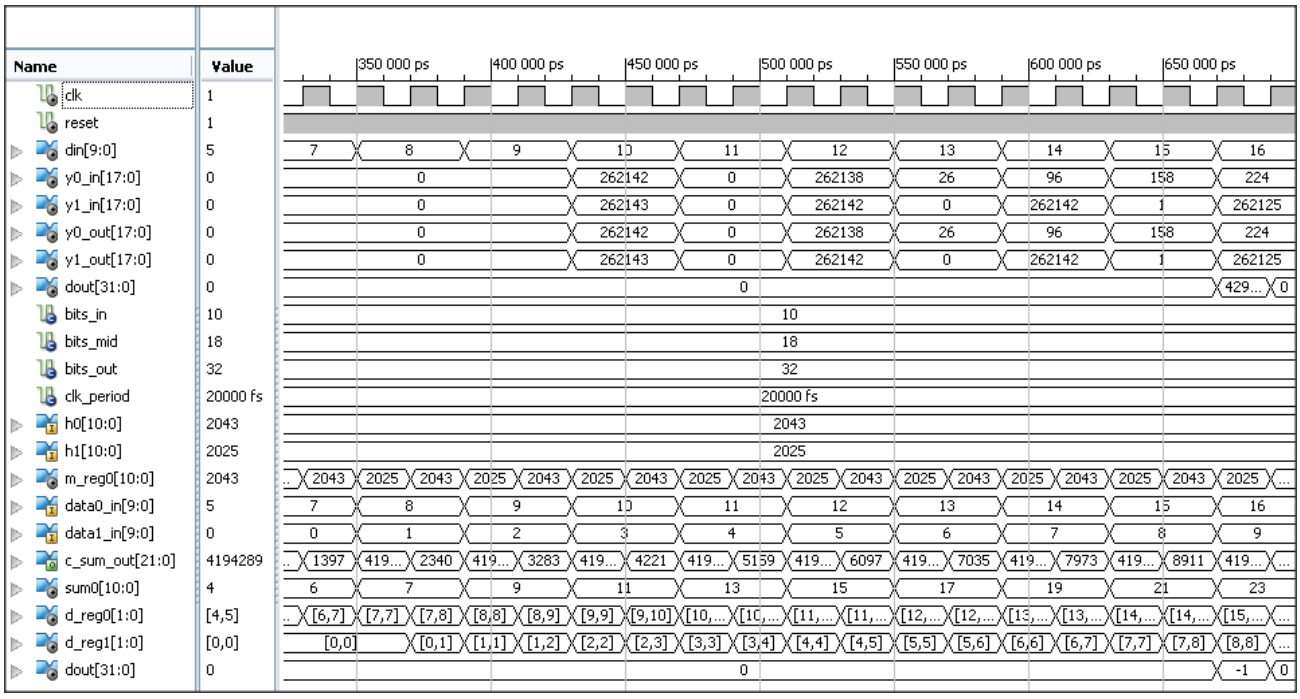

Fig. 11. Timing results for Example 2.3 in [6] showing FIR CELL 1 of Fig. 3. In this graph, h0 and h1 indicate even and odd coefficients of the filter H0, respectively.

REFERENCES

[1]. S. Bishop, S. Rai, B. Gunturk, J. Trahan, and R. Vaidyanathan, "Reconfigurable implementation of wavelet interger lifting transforms for image compression," in IEEE International Conference on Reconfigurable Computing and FPGAs, September 2006, pp. 1–9.

[2]. D. LeGall, "U.S. Pat. 4 829 378: Sub-band coding of images with low computational complexity," Patent, May, 1989.

[3]. A. Wu, K. Liu, and A. Raghupathy, "System architecture of an adaptive reconfigurable DSP computing engine," IEEE Transactions on Circuits and Systems for Video Technology, vol. 8, no. 1, pp. 54–73, 1998.

[4]. P. P.Vaidyanathan, "Multirate digital filters, filter banks, polyphase networks, and applications: A tutorial," IEEE Proceedings, vol. 78, no. 1, pp. 56–93, 1990.

[5]. P. P.Vaidyanathan, "Quadrature mirror filter banks, M-Band extensions and perfect-reconstruction techniques," IEEE ASSP Magazine, pp. 4–20, 1987.

[6]. B. Horng and A. Willson, "Lagrange multiplier approaches to the design of two-channelperfect-reconstruction linear-phase FIR filter banks," IEEE Transactions on Signal Processing, vol. 40, no. 2, pp. 364–374, 1992.

[7]. P. P. Vaidyanathan, T. Nguyen, and T. Saramaki, "Improved approach for design of perfect reconstruction FIR QMF banks, with lossless lattice structures," in Proceedings of the IEEE International Conference Acoustics, Speech, and Signal Processing, April 1988, pp. 1471–1474.

[8]. S. Park and N. Cho, "Design of multiplierless lattice QMF: Structure and algorithm development," IEEE Transaction on Circuits and Systems-II: Express Briefs, vol. 55, no. 2, pp. 173–177, 2008.

[9]. A. Al-Haj, "FPGA-based parallel distributed arithmetic implementation of the 1-D discrete wavelet transform," J. Inform., vol. 29, pp. 241–247, 2005.

[10]. T. Vigneswaran and P. Reddy, "Design of digital FIR filter based on dynamic distributed arithmetic algorithm," J. Applied Sciences, vol. 7, no. 19, pp. 2908–2910, 2007.

[11]. C. Lu and S. Summerfield, "Design and VLSI implementation of QMF banks," IEE Proc. Vision, Image and Signal Processing, vol. 151, no. 5, pp. 421–427, 2004.

[12]. M. D. Valdes, M. J. Moure, J. Dieguez, and S. Antelo, "Hardware solution of a polyphase filter bank for MP3 audio processing,' IEEE Int. Symp. on Industrial Electronics, 2008, pp. 1225 – 1229.e

[13]. X. Huang, L. Zhang, Z. Wei, and F. Fang, "Implementation of DFT filter banks based on FPGA," Int. Conf. on Comp. Distributed Control and Intelligent Environmental Monitoring, 2012, pp. 369 – 372.

[14]. D. –M. Pham, A. B. Premkumar and A. S. Madhukumar, "Design of low hardware complexity filter banks for communications systems employing folding number systems," IEEE GLOBECOM, 2009.

[15]. P. P. Vaidyanathan and P. Hoang, “Lattice structures for optimal design and robust implementation of two-channel perfect-reconstruction QMF banks,” IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 36, no. 1, pp. 81–94, 1988.

[16]. A. Al-Haj, “FPGA-based quadrature mirror filters for DSP applications,” in Third International IEEE Conference on Signal-Image Technologies and Internet-Based Systems, December 2007, pp. 581–584.

[17]. A. Pande and J. Zambreno, “Design and analysis of efficient reconfigurable wavelet filters,” in IEEE International Conference on Electro/Information Technology, December 2008, pp.327 – 33.