FPGA Realization of GDFT-FB Based Channelizers

Fangzhou Wu, Álvaro Palomo-Navarro, Rudi Villing
Department of Electronic Engineering
National University of Ireland Maynooth
Maynooth, Ireland
fwu, apalomo, rudi.villing@eeng.nuim.ie

***Abstract*—Efficient channelization in flexible, reconfigurable communications systems is an ongoing challenge. Our previous work has shown that designs based on the DFT modulated Filter Bank (DFT-FB) and its extension, the Generalized DFT modulated Filter Bank (GDFT-FB) appear to have good computational efficiency and to simplify filter bank design. In this work we examine the design and implementation of the fundamental DFT-FB and GDFT-FB on an FPGA in both critically sampled and oversampled variants. Solutions to various design issues are presented and the FPGA resource usage associated with a concrete example is presented.**

Keywords— GDFT, filterbank, oversampled, FPGA

I. Introduction

In modern communication systems flexibility, reconfigurability, and support for dynamic spectrum allocation techniques continue to gain importance. Support for these can be facilitated by reconfigurable radio and software defined radio (SDR) systems. A key element of these systems is the receiver channelizer which is responsible for extracting independent channels from the received signal through bandpass filtering and down-conversion, prior to subsequent baseband processing of each independent channel. This is particularly challenging in a base station, where the received wideband uplink signal will contain the transmissions from many independent mobile stations and stringent filtering is required to avoid adjacent channel interference.

A number of channelizer architectures have been proposed in the literature to efficiently implement this stringent filtering including Frequency Response Masking (FRM) non-uniform channelizers [1], the FRM based Coffiecient Decimation-based Filter Bank (CDFB) [2] the Tree Quadrature-Mirror Filter (TQMF) Bank [3] and DFT modulated Filter Bank (DFT-FB) channelizers [4]. In our previous work [5] we specifically investigated efficient uniform and non-uniform channelizers based on the DFT-FB and its more general extension, the generalized DFT modulated Filter Bank (GDFT-FB). We subsequently developed even more efficient variations on these [6] but all variations still used the DFT-FB or GDFT-FB as their foundation.

Therefore, the work in this paper is focused on efficient implementation of the fundamental DFT-FB and GDFT-FB in reconfigurable hardware, specifically a Field Programmable Gate Array (FPGA), using multi-rate digital signal processing techniques. In addition to reconfigurability, the primary benefit of an FPGA channelizer implementation is that it should be able to extract a large number of channels simultaneously due to the parallel processing capabilities of the architecture. This work is complementary to research describing the FPGA implementation of other filter bank or channelizer architectures such as the CDFB [2].

In order to facilitate rapid system development and reap the benefits of well tested and optimized FPGA blocks, this work focuses on implementing DFT-FB and GDFT-FB channelizers using reusable FPGA building blocks known as IP (Intellectual Property) cores. In exchange for flexibility and reusability, these IP cores impose certain constraints, which initially appear to prevent their reuse in the GDFT-FB in particular. Therefore one of the objectives of this work is to examine solutions that allow the IP cores to be used and minimize the requirement for additional developments.

In addition, there are three potential problems that this work seeks to address. First, although DFT-FB implementations on an FPGA have been described in the literature (see for example [7]) it does not appear that there has been much work done with FPGA implementations of the GDFT-FB, particularly for channelizer applications. Second, there appears to have been little or no work examining oversampled FPGA implementations of GDFT-FB based channelizers. Although FPGA implementations of the oversampled DFT-FB have been reported (see for example [8]) these have not used IP cores. In general, oversampled channelizers make it easier to use filters to recover the entire channel bandwidth (or even exceed it slightly) without introducing aliasing distortion (as will be explored in more detail in Section II). Finally, our previous work indicated that filter orders for DFT-FB/GDFT-FB filters could be rather high so we wanted to investigate FPGA resource usage for a concrete channelizer example.

The outline of the remainder of this paper is as follows. Section II introduces the background mathematics and architecture of the DFT-FB and GDFT-FB. Section III focuses on the implementation of the critically sampled DFT-FB and GDFT-FB uniform channelizers. Section IV extends this to oversampled implementations. Section V evaluates the FPGA resource usage associated with all implementations for a concrete channelization specification. Finally, conclusions bring the paper to a close.

II. Background

In signal processing, an analysis filter bank is equivalent to a set of parallel band-pass filters that divides a wideband input signal into multiple sub-bands at different center frequencies.

In contrast, a synthesis filter bank synthesizes a single wideband output from multiple input sub-bands. For the purpose of channelization in a communications receiver, it is only the analysis filter bank that is relevant.

A modulated analysis filter bank implements the equivalent of multiple parallel band-pass operations using a single low-pass *prototype* filter and an efficient modulation operation that can filter multiple sub-bands at once. In the context of channelization these sub-bands are the desired narrow-band output channels.

One of the simplest modulated filter banks is the DFT modulated Filter Bank (DFT-FB). The prototype filter for a K band filter-bank, $H(z)$, is divided into K poly-phase components, $E_p(z)$, as follows:

$$H(z)=\sum_{n=-\infty}^{\infty} h(n)z^{-n}=\sum_{p=0}^{K-1} z^{-p}E_p(z^K) \qquad (1)$$

Where

$$E_p(z)=\sum_{n=-\infty}^{\infty} h(nK+p)z^{-n} \qquad (2)$$

The K sub-band filters are obtained by complex modulation of the prototype filter using the DFT algorithm [9]as:

$$H_k(z)=\sum_{p=0}^{K-1} E_p(z^K)z^{-p}W_K^{-kp}, \quad p,k=0,...,K-1 \qquad (3)$$

where $W_K=e^{j2\pi/K}$. Figure 1 shows the block diagram representation of a DFT-FB analysis bank suitable for use as a uniform channelizer. (In typical implementations the Fast Fourier Transform (FFT) is used instead of the DFT because of its greater computational efficiency.)

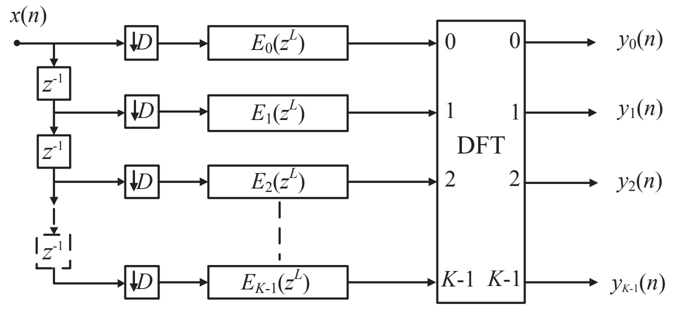

Figure 1. DFT modulated filter-bank (DFT-FB)

In the figure L is the oversampling factor, and D is the downsampling factor of the DFT-FB, defined as

$$L=K/D \qquad (4)$$

and the output sample rate of each sub-band, F_s, is related to the input sample rate by

$$F_s=F_{s,IN}/D \qquad (5)$$

Therefore, when L = 1 the DFT-FB is *critically sampled* whereas when L > 1 the filter bank is *oversampled*. Moreover, although the output of each filter, $H_k(z)$ in (3), is theoretically decimated after filtering, for efficiency this decimation normally takes place before the filtering operation in a polyphase decimated implementation. In this case the polyphase components are also decimated by D so that instead of $E_p(z^K)$ we have $E_p(z^{K/D})=E_p(z^L)$.

A critically sampled filter bank requires a prototype low-pass filter whose filter pass and transition bands does not exceed the decimated Nyquist frequency of the sub-band if aliasing is to be minimized (see Figure 2a). These constraints are relaxed in an oversampled filter bank since the sub-band Nyquist frequency is now larger than the sub-band spacing ($F_{s,IN}/K$) and it is therefore possible to have sub-band filters which overlap in terms of the input signal but whose images do not overlap to cause aliasing after decimation (Figure 2b). More details are available in [10].

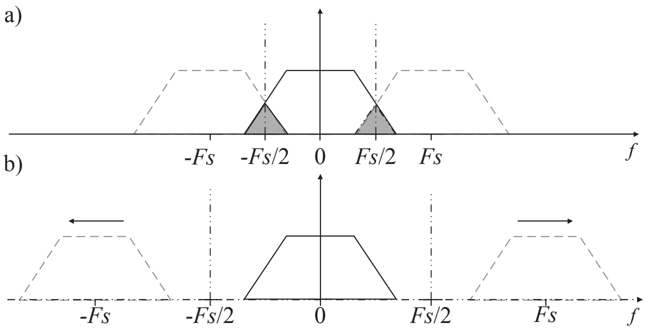

Figure 2. The interaction of a filter with its images in the decimated sub-band output a) exhibits aliasing when critically sampled due to overlapping images whereas b) oversampling greatly reduces aliasing.

A DFT-FB channelizer can only filter an even-stacked set of uniformly spaced filters as shown in Figure 3a. An even stacked filter bank has one channel centred at DC and one (typically unusable) channel that is centred and split in two at the wideband input signal Nyquist frequency.

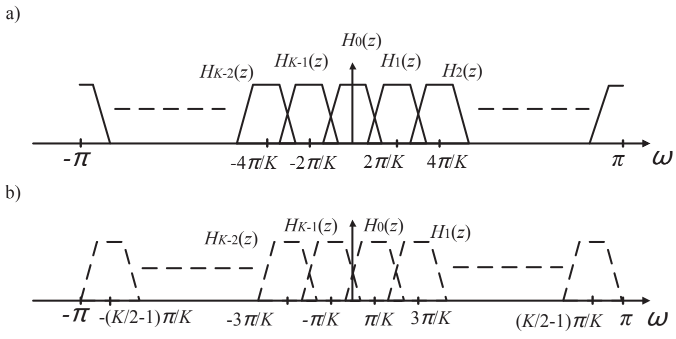

Figure 3. a) Even stacked channels, b) odd stacked channels

The Generalized DFT-FB (GDFT-FB) is an extension of the DFT-FB that offers more flexible channel stacking and phase shifts. In particular the GDFT-FB supports odd-stacked channels as shown in Figure 3b. The effect of the odd-stack design is that the wideband input spectrum is rotated right by half of one sub-band bandwidth. This eliminates the half sub-bands from either end of the even stacked design and may be more useful than the even stacked design in some cases.

The GDFT-FB achieves the phase shift and channel stacking flexibility by implementing the sub-band filters as:

$$H_k(z) = W_K^{-(k+k_0)n_0} \sum_{p=0}^{K-1} E'_p(z^K) z^{-p} W_K^{-kp} \tag{6}$$

where

$$E'_p\left(z^K\right) = E_p\left(z^K W_K^{-k_0 D}\right) W_K^{-k_0 p} \tag{7}$$

and D is the decimation factor, n_0 is a possible phase shift which can be applied to filter bank outputs (usually $n_0 = 0$) and k_0 determines the even or odd stacking of the filter bands ($k_0 = 0$ for even stacked and $k_0 = \frac{1}{2}$ for odd stacked). The GDFT-FB block diagram is shown in Figure 4. The frequency shift term in the figure does not appear in (6) which only describes the input signal band pass filtering—the frequency shift is responsible for shifting the sub-band output from being centered at $F_s/2$ to being centred at DC.

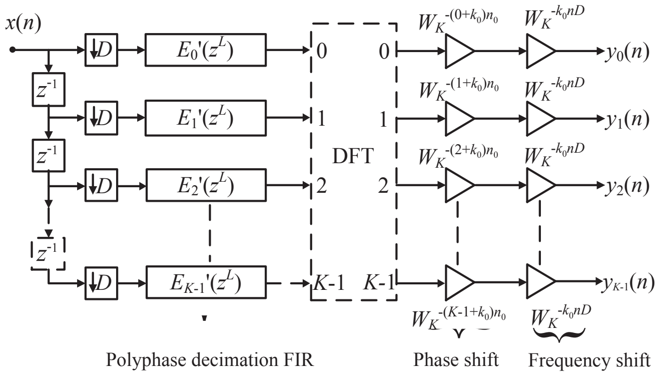

Figure 4. GDFT modulated filter bank (GDFT-FB)

Comparing the GDFT-FB and DFT-FB it is clear that the DFT-FB is just a special case of the GDFT-FB where k_0 and n_0 are both equal to 0 [11]. When n_0 is zero (as it is when the filter-bank is used to implement a channelizer) the phase shift term simplifies to multiplication by 1. Unfortunately the complex modulation terms in the definition of $E'_p\left(z^K\right)$ mean that the polyphase components of the prototype filter now have complex rather than real coefficients. In general it is clear that the flexibility of the GDFT-FB results in some additional complexity and computation relative to the DFT-FB.

III. FPGA IMPLEMENTATION OF CRITICALLY SAMPLED DFT-FB AND GDFT-FB CHANNELIZERS

This section develops critically sampled FPGA channelizer implementations on the Xilinx family of FPGAs using the ISE (Integrated Software Environment) tool suite and the library of reusable IP cores, providing solutions where is necessary to IP core limitations.

A. Wordlength Consideration

In this implementation, the input samples have 16-bit signed in-phase and quadrature parts, the coefficients are also in 16-bit signed representation. This allows us to make efficient use of the embedded DSP Blocks on the FPGA. The architecture allows for 16-bit coefficients with a scale value. The scale value must be computed in advance by the user, but is simply a case of finding the maximum dynamic range for each sub-band filter and scaling by a power of 2.

B. Basic DFT-FB Channelizer Implementation

As is typical when implementing communications systems, the complex input signal is not processed directly in complex form in the FPGA but must instead be divided into its in-phase (I) component, corresponding to the real part of the signal, and quadrature (Q) component, corresponding to the imaginary part of the signal, before input to the FPGA channelizer.

Xilinx provides an FFT IP core which can be used in the DFT-FB and GDFT-FB. They also provide an FIR compiler IP core whose polyphase decimation mode efficiently implements the filtering operation required for the DFT-FB (see left hand side of Figure 1). In this mode the delay chain and decimation operations are implemented using a commutator [12].

The FIR compiler IP core only generates filters which implement a *real filter* operation (that is, the prototype filter coefficients must be real). Nevertheless it does apply this real filter identically to two inputs which are normally the I and Q components of a signal to be filtered. Therefore the IP core directly supports real filtering of a complex signal (which has been separated into its I and Q components).

Finally it is worth noting that both the FIR compiler and FFT IP cores use time division multiplexed serial inputs and outputs rather than the notional parallel paths suggested by Figure 1 and Figure 4. Putting all this together we find that the FIR compiler and FFT IP cores facilitate the straightforward implementation of a DFT-FB channelizer capable of extracting 8–1024 channels as shown in Figure 5.

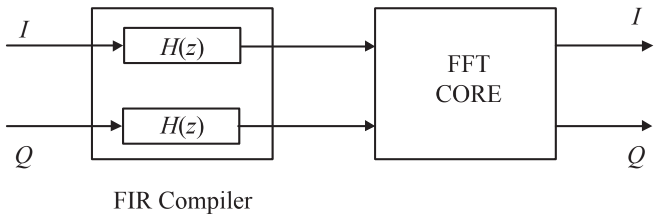

Figure 5. DFT-FB implementation using FIR compiler and FFT IP cores

C. GDFT-FB Channelizer Implementation

1) Complex modulation of prototype filter coefficients

Equation (7) indicates the coefficients of the (real) prototype filter are subject to complex modulation for the GDFT-FB. This modulation is applied offline at design time so that the modulated coefficients (separated into their I and Q parts) are supplied to the FIR compiler IP cores. To see how this is done, first consider the time domain equivalent of (7) without any interpolation

$$e'_p(n) = e_p(n) e^{j\frac{2\pi}{K}k_0 Dn} e^{-j\frac{2\pi}{K}k_0 p} \tag{8}$$

where

$$e_p(n) = h(nK + p), \qquad p = 0,\ldots,K-1 \tag{9}$$

In the case of a critically sampled GDFT-FB, where $D = K$ and $k_0 = \frac{1}{2}$ then (8) reduces to

$$e'_p(n) = e_p(n)e^{j\pi n}e^{-j\pi p/K} \tag{10}$$

The modulation of coefficients is applied to each polyphase component independently. Thereafter the modulated component coefficients are interpolated and reassembled as indicated by (1), substituting $E'_p(z^K)$ for $E_p(z^K)$, to form the appropriate arrangement of prototype filter coefficients. These coefficients are then divided into their I and Q parts for use with the complex filter implementation to be discussed next.

2) Complex filter implementation using FIR compiler

The GDFT-FB is not as straightforward to implement as the DFT-FB when the parameter k_0 is non-zero [10]. (We will not examine cases where the parameter n_0 is non-zero in this work since it is not required for channelizer design.) When k_0 is non-zero then the coefficients of the prototype filter are subject to complex modulation (see (7)) yielding complex filter coefficients which the FIR compiler IP core does not support. A complex FIR may be implemented by cross-coupling two real FIRs [13] and this approach may be used with the FIR compiler IP core as shown in Figure 6.

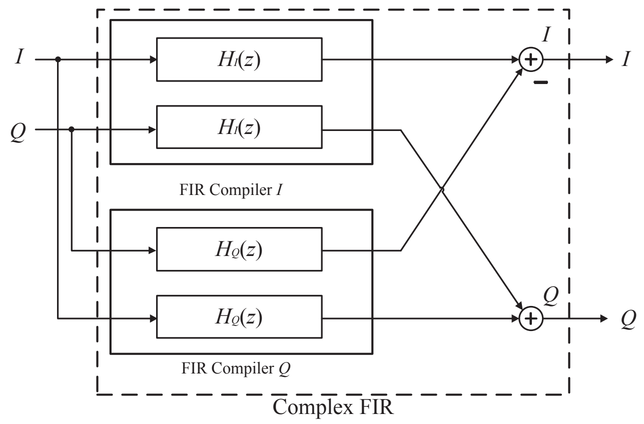

Figure 6. Complex FIR implemented using cross-coupled FIR compiler IP core

To see how this cross-coupled approach works it is useful to consider the minimal case: multiplication of a single input sample, $x = x_i + x_q j$, with a single filter coefficient, $h = h_i + h_q j$, which yields

$$xh = (x_i h_i - x_q h_q) + (x_i h_q + x_q h_i) j \tag{11}$$

3) Frequency shift state machine

From Figure 4 it is clear that the output of the DFT is followed by two separate complex multiplications. One of these, the phase shift operation, reduces to multiplication by 1 (that is no operation required) when n_0 is zero. The frequency shift operation, $W_K^{-k_0 nD}$, shifts the output of each channel as

$$W_K^{-k_0 nD} = e^{-\frac{j2\pi}{K}(k_0 nD)} \qquad n \in \mathbb{N} \tag{12}$$

When $k_0 = \frac{1}{2}$ and $K = D$ (as is the case when $L = 1$ in a critically sampled filter bank) then this reduces to

$$W_K^{-k_0 nD} = e^{-j\pi n} = \begin{cases} 1, & n \text{ even} \\ -1, & n \text{ odd} \end{cases} \quad \text{for } n \in \mathbb{N} \tag{13}$$

where n is the output sample number. This has the effect of shifting the sub-band output by half the output sampling rate so that it is centred at DC and it can be efficiently implemented using a simple state machine which either passes the output through unchanged (for even numbered samples) or negates the output (for odd numbered samples). Since the FFT outputs 1 sample from each output sub-band sequentially, the state machine changes state only after K samples have been output from the FFT.

4) Final design

Figure 6 Figure 7 shows the FPGA implementation block diagram of the theoretical GDFT-FB design shown in Figure 4 incorporating each of the steps described above. Because the GDFT-FB is a general design it could be used to implement both even-stacked and odd-stacked channelizers. Nevertheless, the DFT-FB is the more efficient design for even stacked designs since it does not require the complex FIR or the output mixer state machine.

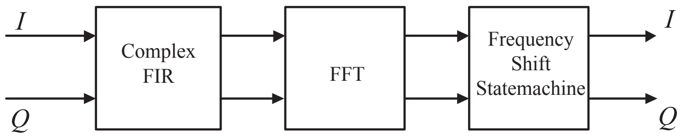

Figure 7. FPGA implementation of the GDFT-FB

IV. FPGA IMPLEMENTATION OF OVERSAMPLED DFT-FB AND GDFT-FB CHANNELIZERS

A. High Level Design

In general the high level design of the oversampled DFT-FB and GDFT-FB is very similar to the critically sampled designs shown in Figure 1 and Figure 4. In particular the overall structure shown in these figures does not change.

Nevertheless, there are two significant changes required. First, since $L = K / D > 1$ in the oversampled case, the decimation factor no longer matches the number of channels and the implementation of both the real FIR (in the DFT-FB) and complex FIR (in the GDFT-FB) must be replaced with an oversampled version as described in the following section. Second, for similar reasons the output frequency shift state machine design must be modified since there are more states to implement for an oversampled design.

B. Oversampled Polyphase Decimation FIR

The FIR compiler IP core implements the polyphase decimation structure using a commutator. The decimation factor in this structure must, by definition of how a commutator works, match the number of polyphase branches. However, for an oversampled filter bank, the decimation factor must be less than the number of branches. The question then, is how to implement an oversampled polyphase decimation FIR when only critically sampled polyphase decimation FIR blocks are available?

Consider a filter bank where $L = 2$. Expanding (1) we get

$$H(z) = E_0\left(z^K\right) + z^{-1}E_1\left(z^K\right) + \ldots + z^{-(K-1)}E_{K-1}\left(z^K\right) \quad (14)$$

which can be re-written as

$$\begin{aligned}H(z) &= E_0\left(z^K\right) + z^{-1}E_1\left(z^K\right) + \ldots + z^{-(K/2-1)}E_{K/2-1}\left(z^K\right) \\ &+ z^{-K/2}\left[E_{K/2}\left(z^K\right) + z^{-1}E_{K/2+1}\left(z^K\right) + \ldots + z^{-(K/2-1)}E_{K-1}\left(z^K\right)\right]\end{aligned} \quad (15)$$

In general, the polyphase decomposition of the DFT-FB prototype filter in (15) can be re-written as

$$\begin{aligned}H(z) &= \sum_{i=0}^{L-1} z^{-iD}\left(\sum_{p=0}^{D-1} z^{-p}E_{p+iD}\left(z^K\right)\right) \\ &= \sum_{i=0}^{L-1} z^{-iD}H_{FIRi}(z)\end{aligned} \quad (16)$$

where

$$H_{FIRi}(z) = \sum_{p=0}^{D-1} z^{-p}E_{p+iD}\left(z^K\right) \quad i = 0,1,\ldots,L-1 \quad (17)$$

The benefit of this decomposition is that each $H_{FIRi}(z)$ now contains just D polyphase branches so that it may easily be decimated by D using a commutator making it compatible with FIR compiler IP cores. The same decomposition can be applied to the GDFT-FB prototype filter by substituting $E'_{p+iD}\left(z^K\right)$ for $E_{p+iD}\left(z^K\right)$ in (16).

The overall solution, therefore, is that the oversampled polyphase decimated FIR is implemented using L, the oversampling factor, number of polyphase decimation FIR blocks (real or complex as needed by the DFT-FB or GDFT-FB respectively). The sub-prototype filter supplied to each of these FIR blocks is assembled from a subset of the interpolated polyphase components of the overall prototype filter in accordance with (17).

Furthermore, since each of the FIR blocks executes in a parallel and produces outputs simultaneously, it is necessary to add an FIR selector state machine which implements the time division multiplexing of FIR block outputs onto the single I and Q input to the FFT IP core. Specifically, to implement the DFT-FB or GDFT-FB correctly the oversampled FIR outputs from branch 0 to branch K of the overall polyphase decomposition must be supplied to the FFT sequentially. First the D samples from 0 to D-1 are selected from FIR_1, then D samples from D to $2D$-1 are selected from FIR_2, and so on, until the final D samples from (L-1) D to LD-1 are selected from FIR_L at which point the sequence begins again.

Since the outputs of all FIR blocks are produced simultaneously it is necessary to add a FIFO to the outputs of each FIR block so that samples can be stored until they are selected for output from the overall oversampled FIR.

The resulting implementation of the oversampled polyphase decimation FIR is shown by Figure 8.

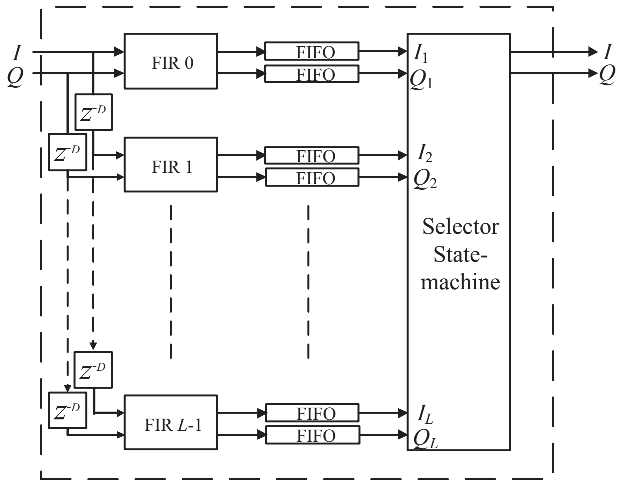

Figure 8. Oversampled polyphase decimation FIR implemented using real or complex critically sampled polyphase decimation FIR blocks

In the case of the DFT-FB, each of the FIR blocks in the oversampled FIR implements a real filter and therefore can be implemented using the FIR compiler directly. Consequently, the oversampled DFT-DB requires L FIR compiler IP cores. As for the critically sampled case, the GDFT-FB will require complex filters and therefore each FIR block must implement a complex filter. This means that the GDFT-DB requires $2L$ FIR compiler IP cores.

C. Oversampled Frequency Shift State Machine

As was the case for the critically sampled GDFT-FB, the final frequency shift can be implemented using a state machine, albeit one with more states. The number of states depends on the oversampling factor, $L = K/D$. Substituting $k_0 = \frac{1}{2}$ and $K = LD$ into (12) yields

$$W_K^{-k_0 nD} = e^{-j\pi n/L} \quad n \in \mathbb{N} \quad (18)$$

In the case that $L = 2$ this reduces to just four unique values

$$W_K^{-k_0 nD} = \begin{cases} -j & n = 4m \\ -1 & n = 4m+1 \\ j & n = 4m+2 \\ 1 & n = 4m+3 \end{cases} \quad n \in \mathbb{N},\ m \in \mathbb{N} \quad (19)$$

and therefore the frequency shift can be replaced by a state machine with 4 states. All 4 of these multiplications can be implemented without any multipliers since the operations amount to passing through, negating, or swapping the I and Q components.

Similar state machines can be derived for larger interpolation factors but it is worth noting that multipliers will be required in this case.

V. EVALUATION

A. Equipment and Materials

In order to evaluate the FPGA designs described in this paper it was necessary to construct working implementations. These were developed on the Xilinx Virtex-6 FPGA prototyping board. The Xilinx Virtex-6 XC6VLX240T has over 240,000 gates, 768 DSP48 slices, and 416 block RAMs. Every DSP48 contains a 25×18 multiplier, an adder, and an accumulator. Block RAMs are 36 Kb in size and can also be used as two independent 18 Kb blocks [14]. Xilinx ISE Foundation Series tools and IP cores (FIR compiler and FFT) were used.

B. Test and Results

A 16-channel channelizer was implemented using each of the four different designs examined in this paper, namely the critically sampled DFT-FB and GDFT-FB, and the oversampled DFT-FB and GDFT-FB. In all cases the correct operation of the channelizer was validated and the resource usage for the design was examined so that the scalability of the design to larger numbers of channels could be understood.

The wideband input signal was digitized at 2 megasamples/second with a resolution of 16 bits per sample. The channel spacing in the wideband input signal was 125 kHz. A length 512 low pass prototype filter was designed such that the pass band end frequency was 57.5 kHz and stop band start frequency was 67.5 kHz. Each of the sub-band filters in the filter bank therefore overlapped its adjacent sub-band filters by approximately 28 kHz on both sides.

The FPGA was operating with a 600 MHz clock and therefore had 300 clock cycles on average to process each input sample. An FPGA is organized into slices and each slice has some lookup table (LUT) resources (used to implement predefined logic) and register resources (usually used to maintain state and synchronize input/output). Block RAM will be used to store data and be embedded throughout the FPGA, for example filter coefficients. All the multiplications and additions were implemented by DSP48s for maximum performance. Therefore the usage of the block RAM and DSP48 resources are the most important to evaluate.

Table 1 Resource usage comparison of 3 different designs

	Slice Register	Slice LUT	DSP48s	Block RAMs
Critically sampled, Even stack	1125	819	5	4
Oversampled x2, even stack	1445	1503	7	6
Critically sampled, odd stack	1551	1175	7	6
Oversampled x2, odd stack	2121	2123	15	6
Available	301440	150720	768	416

Table 1 shows the resource usage for each of the four channelizer designs. Although the resources used depend on factors such as input signal sample rate, the number of channels, *K*, and the specifications of the prototype filter, the results suggest that the upper bound capacity of the Virtex-6 would be most constrained by the availability of DSP48 resources. In future work we intend to examine this upper bound capacity in more detail and to explore the relationship between various parameters of the filter bank design and the number of resources consumed.

CONCLUSION

In this work, we have shown how to implement even and odd stacked channelizers in critically sampled and oversampled variants on an FPGA using reusable IP cores. Although the GDFT-FB is more flexible than the DFT-FB since it supports both even and odd stacked channels, its resource consumption is higher than the DFT-FB, particularly in the oversampled configuration. This does however ignore the fact that filter design may be less stringent with oversampled designs than with critically sampled designs because of the greatly reduced aliasing problem. In general, if maximum spectrum utilization is required (suggesting the odd stacked channels) and lack of aliasing distortion is important, then the oversampled GDFT-FB would be the preferred channelizer design.

REFERENCES

[1] R. Mahesh and A. P. Vinod, "Reconfigurable Low Area Complexity Filter Bank Architecture Based on Frequency Response Masking for Nonuniform Channelization in Software Radio Receivers," *Aerospace and Electronic Systems, IEEE Transactions on,* vol. 47, pp. 1241-1255, 2011.

[2] S. Darak, A. Vinod, and E. K. Lai, "A Low Complexity Reconfigurable Non-uniform Filter Bank for Channelization in Multi-standard Wireless Communication Receivers," *Journal of Signal Processing Systems,* vol. 68, pp. 95-111, 2012/07/01 2012.

[3] T. C. Farrell and G. Prescott, "A low probability of intercept signal detection receiver using quadrature mirror filter bank trees," in *Acoustics, Speech, and Signal Processing, 1996. ICASSP-96. Conference Proceedings., 1996 IEEE International Conference on*, 1996, pp. 1558-1561 vol. 3.

[4] H. Xiaohong, Z. Lin, W. Zhicheng, and F. Fang, "Implementation of DFT Filter Banks Based on FPGA," in *Computer Distributed Control and Intelligent Environmental Monitoring (CDCIEM), 2012 International Conference on*, 2012, pp. 369-372.

[5] A. Palomo, R. Villing, and R. Farrell, "Overlapped Polyphase DFT Modulated Filter Banks Applied to TETRA/TEDS SDR Base Station Channelization," presented at the RIA Colloquium on Communications and Radio Science 2010, Dublin, Ireland, 2010.

[6] A. P. Navarro, R. Villing, and R. J. Farrell, "Practical Non-Uniform Channelization for Multistandard Base Stations," *ZTE Communications,* vol. 09, pp. 15-24, 2011.

[7] S. A. Fahmy and L. Doyle, "Reconfigurable polyphase filter bank architecture for spectrum sensing," presented at the Proceedings of the 6th international conference on Reconfigurable Computing: architectures, Tools and Applications, Bangkok, Thailand, 2010.

[8] P. L. De Leon, "On the Use of Filter Banks for Parallel Digital Signal Processing."

[9] P. P. Vaidyanathan, *Multirate systems and filter banks*: Prentice-Hall, Inc., 1993.

[10] R. E. Crochiere and L. R. Rabiner, *Multirate Digital Signal Processing*: Prentice-Hall, 1983.

[11] A. P. Navarro, T. Keenan, R. Villing, and R. Farrell, "Non-uniform channelization methods for next generation SDR PMR base stations," in *Computers and Communications (ISCC), 2011 IEEE Symposium on*, 2011, pp. 620-625.

[12] Xilinx, inc. "IP LogiCORE FIR Compiler v5.0 Product Specification."

[13] K. W. Martin, "Complex Signal Processing is Not Complex," *IEEE Transactions on Circuits and Systems I: Regular Papers,* vol. 51, pp. 1823-1836, 2004.

[14] Xilinx, inc. "Virtex-6 Family Overview," ed, 2011.