Reconfigurable FPGA-Based Channelization Using Polyphase Filter Banks for Quantum Computing Systems

Johannes Pfau, Shalina Percy Delicia Figuli, Steffen Bähr, and Jürgen Becker

Karlsruhe Institute of Technology, Institute for Information Processing Technologies,
Engesserstr. 5, 76131 Karlsruhe, Germany
{johannes.pfau,shalina.ford,steffen.baehr,juergen.becker}@kit.edu

Abstract. Recently proposed quantum systems use frequency multiplexed qubit technology for readout electronics rather than analog circuitry, to increase cost effectiveness of the system. In order to restore individual channels for further processing, these systems require a demultiplexing or channelization approach which can process high data rates with low latency and uses few hardware resources. In this paper, a low latency, adaptable, FPGA-based channelizer using the Polyphase Filter Bank (PFB) signal processing algorithm is presented. As only a single prototype lowpass filter needs to be designed to process all channels, PFBs can be easily adapted to different requirements and further allow for simplified filter design. Due to reutilization of the same filter for each channel they also reduce hardware resource utilization when compared to the traditional Digital Down Conversion approach. The realized system architecture is extensively generic, allowing the user to select from different numbers of channels, sample bit widths and throughput specifications. For a test setup using a 28 coefficient transpose filter and 4 output channels, the proposed architecture yields a throughput of 12.8 Gb/s with a latency of 7 clock cycles.

Keywords: quantum computing, FPGA, signal processing, channelization

1 Introduction

"Is there a boundary to the never ending, data hungry applications that emerge afresh every day?" is the constant question confronted in the world of science and technology. On one hand, modern applications ranging from data-streaming to video conferencing, high-resolution detectors to quantum computing have the urgency to process a huge amount of data with input frequencies in the range of THz and beyond. On the other hand, Moore's law, the engine that has powered the semiconductor industry is descending towards obsoleteness, as the state-of-the-art Field Programmable Gate Arrays (FPGAs) are struggling to be clocked above 1 GHz. Even in the analog domain, although recent Analog to Digital Converters (ADCs) have been pushed to GHz range, dividing the ever-widening

This is the accepted version.
The final publication is available at Springer via https://doi.org/10.1007/978-3-319-78890-6_49.

multiplexed input spectrum into narrower bandwidths through analog front-ends gets more complicated, requires higher cost, more power and greater area. An alternate method to tackle this problem is to turn the focus towards available channelization techniques which can be further optimized and made scalable to meet our requirements. Channelizers, also known as filter banks, perform the task of processing an input signal with a certain bandwidth into one or multiple derived signals covering a subset of the input signal's bandwidth. Through the lineage of channelization commencing from the latter half of 1970s has its trail all along the way from television receivers to Wi-Fi, the recent, more sophisticated and hot off the fire application cases such as Cryogenic Particle Detectors (CPDs), multi-qubit Quantum Computing (QC) and the like use frequency multiplexed readouts and thereby require a channelizer which can fulfill the following requirements:

1. Allow for high input sampling and data rates required by wide-bandwidth inputs and high-resolution samples.
2. Process data with low latency, as certain applications in QC use feedback loops and therefore limit the highest tolerable latency.
3. Utilize as few resources as possible to fit in state-of-the-art or even low budget FPGAs.
4. Allow scaling for different, and large number of channels to adapt to a variety of application use cases.
5. Allow reconfiguration to allow for rapid development and easy adaption to other applications.

Among the popular channelization techniques available, such as Digital Down Conversion (DDC), Weighted Overlap Add (WOLA) and Pipelined Frequency Transform (PFT) [1], this paper focuses on the implementation of PFBs on FPGAs. PFBs allow scaling with larger number of channels and performing filtering at lower sample rates. Combined with advantages of FPGAs such as reconfigurability, flexibility and fast I/O interfaces, this enables PFBs to meet the aforementioned requirements. The main design scheme of our channelizer is not only a solution for applications with high data throughput but also for the ones in QC where maintaining low latency is crucial. Thereby, the objective is to make appropriate trade-offs between supported sample rate, required resources and filter specification in order to embrace high data rates with lower latency. The rest of the paper is structured as follows: Section 2 briefs the available common channelization concepts with Sect. 3 discussing the related work. The proposed channelizer architecture is elaborated in Sect. 4 and its implementation in FPGA technology along with validation results are presented in Sect. 5. Finally, conclusions are summarized in Sect. 6.

2 Common Channelization Concepts

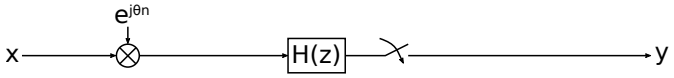
Fast implementations of well-studied Digital Down Conversion systems have been presented by Meyer et al. [2], but one of the limitations of such systems is

resource sharing. The components cannot be shared between multiple channels, even though each channel requires the same components for data processing. As shown in Fig. 1a, each channel's input signal has to be multiplied with a heterodyne to shift the desired channel's center frequency to the base band. This is followed by a low pass filter to limit the signal bandwidth to the channel width. The filtered signal is then downsampled to reduce the sample rate for further processing. This same chain of working principle is followed for all the other channels. As the number of channels increases, duplication of heterodyne multiplication and filtering operation also increases proportional to the number of channels and thereby makes this approach cost ineffective. When certain characteristics about the input signal are met, specialized Orthogonal Frequency Division Multiplexing (OFDM) demultiplexing systems are also a valid option [3]. But as our system has to deal with arbitrary input signals, such a solution can not be used here.

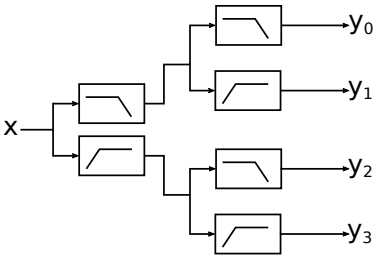
Shown in Fig. 1b, the Pipelined Frequency Transform channelizer recursively applies half band filtering to split the current signal into two new signals consisting of the lower and upper half of the original bandwidth. When equal bandwidth channels are desired, the number of channels supported by PFT is limited to powers of two. As the number of channels increases, the hardware resource requirements increase on a logarithmic scale with data processing latency being increased for every added filter stage. The absolute processing latency and resource usage depends largely on the used half band filters. The design of efficient half band filters such as Infinite Impulse Response (IIR) filters depends on output characteristics, for instance on linear phase requirements [4]. As our channelizer needs to be easily adaptable to different applications, approaches such as PFT depending on filters to be manually optimized for each application are not further considered.

In Weighted Overlap Add filter banks, a block of input data is multiplied with a signal processing window and the multiplied data is sliced into multiple buffers. These are then overlapped and added. This summed data is further block processed by a Discrete Fourier Transform (DFT) or Fast Fourier Transform (FFT) block processor as depicted in Fig. 1c. Although arbitrary resampling rates can be adopted for polyphase channelizers as exhibited in [5] and [6], it demands modifications in the channelizer's structure, whereas WOLA channelizers require only a different overlap and therefore no change to the structure is needed [7]. On the other hand, adjusting channel filter characteristics is more straightforward in PFBs, as the filter which shapes individual channels is designed directly, whereas WOLA channelizers require a windowing function which is a more abstract way to describe channel characteristics. As simple reconfigurability for different channel shaping filters is a requirement and arbitrary oversampling is not mandatory for our use case, PFB is chosen over WOLA channelizer.

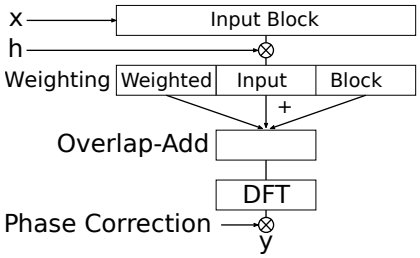
Polyphase Filter Banks. PFBs are an extension of polyphase filters to process multiple channels in parallel using a single prototype filter. Polyphase filter partitioning is a special structure used to describe FIR filters, leading to new per-



(a) One channel of Digital Down Conversion (DDC)



(b) Pipelined Frequency Transform (PFT)



(c) Weighted Overlap Add (WOLA)

Fig. 1: Block diagrams of various channelizer systems

spectives in signal processing. These structures are possible when an FIR filter is followed by an interpolator or decimator which respectively increases or reduces the sample rate after filtering. Since splitting one wide-band signal into multiple narrow-band signals with lower sampling rates is of interest in this context, only decimating filters are described here. The well-known Finite Impulse Response (FIR) filtering equation is combined with the equation of a downsampler [5] to yield equations (1) to (3) with (1) describing the filter phases p_ρ, (2) illustrating the filter input signals x_ρ and (3) the restructured filtering equation for an M phase polyphase filter. The equations represent a counterclockwise commutator structure, where $h(n)$ represents the FIR filter coefficients and $x(n)$ represents the input signal. Alternatively, equations for the clockwise commutator model can be deduced as well. The models are mathematically equivalent and neither provides an advantage in implementation, but it is important not to mix filter phase signals and filter input signals of the different models.

$$p_\rho(n) = h(nM + \rho) \tag{1}$$

$$x_\rho(n) = x(nM - \rho) \tag{2}$$

$$y(n) = \sum_{\rho=0}^{M-1} p_\rho(n) * x_\rho(n) \tag{3}$$

A block diagram for a two phase polyphase filter ($M = 2$) is shown in Fig. 2. The commutator depicted by a bent arrow takes input signal $x(n)$ to produce filter input signals $x_\rho(n)$ according to (2). The phase filters p_0 and p_1 are standard FIR filters using coefficients as described by (1). The inputs are filtered using the phase filters and are summed at the output according to (3).

The polyphase channelizer concept extends polyphase filters to form a channelizer. A polyphase lowpass filter produces the downsampled output of the

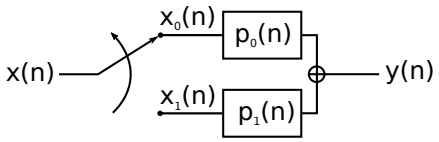


Fig. 2: Block diagram of a polyphase filter

input spectrum filtered by a lowpass spectrum centered on DC frequency. This produces a single output channel, whereas a polyphase channelizer duplicates and shifts the prototype filter's spectrum to generate multiple output channels. Equation (4) as derived in [4] describes the output signals of a polyphase channelizer. In this equation, y_r is the r-th polyphase filter phase output signal and M is the number of total output signals. When comparing this output equation to the well-known DFT transform equation, the polyphase channelizer structure can be deduced as shown in Fig. 3 using a combination of polyphase filter and DFT. It should be noted that the center frequencies of the channels in a polyphase channelizer are distributed equidistantly and the distance between adjacent center frequencies is entirely determined by the number of output channels and the input bandwidth. The center frequencies of the channels can therefore only be further modified by shifting all the channels simultaneously such as in the Generalized Discrete Fourier Transform (GDFT) channelizer [5].

$$y(nM, k) = \sum_{r=0}^{M-1} y_r\,(nM)\, e^{j\frac{2\pi}{M}rk} \tag{4}$$

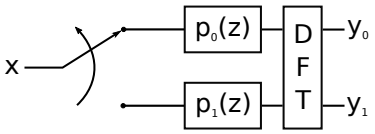


Fig. 3: Block diagram of a polyphase channelizer

3 Related Work

PFBs have been first studied in digital signal processing fields [5] and are often applied in audio applications for signal processing, composition and decomposition [8]. More recently, PFBs are introduced in the domain of communication technology, where they are called polyphase channelizers [4], and in the field of astrophysics [9]. They have also been used as spectrum analyzers, as described by Fahmy in 2010 [10]. It is interersting to note that in [11] PFBs are combined

with Frequency Response Masking (FRM) filters instead of FIR filters. Wu also described an FPGA based polyphase channelizer with odd or even stacking support and optional oversampling [12]. A polyphase channelizer for many-core CPU systems has been proposed in [13] and Chennamangalam et al. have developed a polyphase channelizer system for astrophysics running on GPUs [14].

Previous work has not been concerned with latency introduced by the channelization process and thereby provides no latency measurements. As our channelizer needs to be used in QC applications, where the channelization output is used as feedback to the system, it needs to have well-defined and well-known latency characteristics. Using polyphase channelizers in such systems has the potential to significantly reduce hardware resource consumption while obeying the latency requirements.

4 Proposed Channelizer Architecture

System Overview. An overview of the proposed channelizer architecture is shown in Fig. 4. The data flow closely follows the channelizer block diagram presented in Fig. 3 with some additional modules for signal processing. As this is a pipelined approach, there is no need for any control signals except for local signals handling data flows between modules. In order to allow combinations of commutator and filter modules to either pass the input data in parallel to polyphase branches or to implement the branches in a single filter and pass the input samples serially, interfacing between the modules is kept generic.

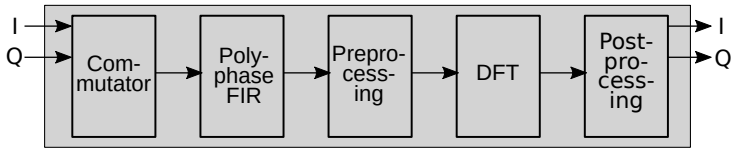


Fig. 4: Overview of modules in the channelizer architecture

The commutator being the first module maps the incoming complex-valued data stream (I and Q) to the polyphase filter phases. If the input data is already available as parallel bitstreams and in the correct format, then the commutator can be skipped with the bitstreams directly given to the respective polyphase filters. These filters are implemented in the polyphase FIR module. In this case, the sample rate of each filter branch is the same as the data rate of a single input bitstream. When the input data is a single bitstream, then the commutator uses a time based demultiplexing approach with the sample rate equal to $(data\,rate\,/\,number\,of\,channels)$. For an optimized implementation, FIR filters can be either clocked at a lower frequency or some filters can be combined into a single filter clocked at higher frequency. The FFT module is a standard FFT or DFT component and can be implemented either as a pipelined FFT processing

data serially or as a parallel FFT processing data in parallel for achieving maximum throughput. The preprocessing and postprocessing modules are responsible for additional processing at the DFT input and output. They are required only for advanced cases such as oversampled PFBs or GDFT PFBs which are not further discussed in this paper.

Truncation and Scaling of Intermediate Values. As filtering and FFT operations are based on multiple additions and multiplications, the output bit width increases significantly due to fixed point data processing. In order to allow maximum precision in channelizer's result, the output samples are supported with full bit growth. For reduced resource usage, each module can optionally apply truncation and scaling to its output data.

5 Implementation, Test Application and Results

In addition to hardware implementation on Xilinx Virtex-7 VC707 evaluation board, a complete MATLAB model has been developed to pre-evaluate the expected behavior and to serve as a reference for the implemented system.

5.1 FPGA Implementation

Commutator Module. The commutator module splits the incoming bit stream into parallel streams to be processed by the polyphase filter branches using a counter driven demultiplexer. Whenever a valid input sample is processed, the counter is incremented and the next sample is delivered to a different filter branch. When data is transferred from a fast ADC, it is already available as parallel streams. In such cases, if the number of parallel streams and the number of channels (filter branches) match, then the commutator is replaced by a simple pass-through entity. If the input is available as parallel streams but does not match the number of channels, a more advanced remuxing concept has been implemented. Two clock sources have been utilized with the commutator making use of the faster one.

FIR Filter Bank. Each implemented FIR phase filter is structurally equivalent to other filters but with different coefficients. For our use case, the incoming bit stream is split into multiple parallel streams and thereby all the filter branches are processed in parallel. This in turn will increase the hardware requirements especially when the number of channels is very high. In order to reduce resource usage, various filters can be merged into a single hardware filter implementation using time domain multiplexing. When using fully parallel processing, the data rate and clock rate of the individual phase filters are reduced compared to the input signal sample rate. As the filters are built using Digital Signal Processing (DSP) slices, having lower clock frequency will allow for using less pipelining and thereby introduce less latency.

Using a direct form FIR filter is favorable when implementing hardware sharing by using multiple filter coefficient sets with one filter implementation. This kind of filter however requires additional pipelining at the output stage due to the large adder tree. As lower latency is one of the key criteria in QC, a simple transpose FIR filter as shown in Fig. 5 has been implemented using DSP48 slices. In transpose filters, delay registers are not used at the input stage as shift registers, but are instead interposed into the adder chain. This makes the adder chain intrinsically pipelined. By using dedicated routing channels on the FPGA, this structure can be implemented efficiently. Even though increased complexity in realizing multiple filters in such a transpose filter hardware structure is a disadvantage, hardware sharing is not a concern for our fully parallel filter bank. While higher fanout at the circuit driving filter inputs may limit the filter performance, testing shows a simple transpose filter to be a better option for our parallel filter bank implementation.

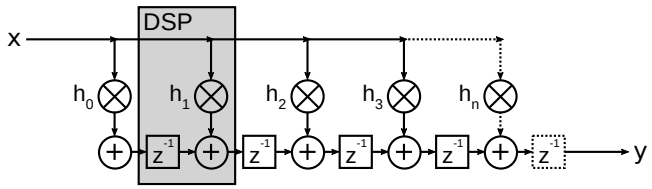


Fig. 5: Transpose form FIR filter

DFT. Considering a four channel configuration in the QC system, an optimized four channel channelizer using the well-known radix-2 Cooley-Tukey decimation algorithm has been employed. Explicit equations for real and imaginary components of each output channel derived from the standard DFT equation (5) yield a simple FFT structure, which can be implemented using two stages of adders. In this special case of a four channel DFT, all phase shifts degenerate to multiplications by 1, -1, j or $-j$ which can be interpreted as sign inversion and coupling between real and imaginary channels. Therefore, the need for multipliers can be eliminated. For example, derived equations for channel 0 depicted in (6) and (7) can be directly used to implement a 4-point parallel DFT where the input data samples x_n are passed as two independent values $Re(x_n)$ and $Im(x_n)$.

$$X(k) = \frac{1}{N} \sum_{n=0}^{N-1} x_n e^{-j\frac{2\pi nk}{N}} \tag{5}$$

$$Re\,(X_0) = (Re\,(x_0) + Re\,(x_2)) + (Re\,(x_1) + Re\,(x_3)) \tag{6}$$

$$Im\,(X_0) = (Im\,(x_0) + Im\,(x_2)) + (Im\,(x_1) + Im\,(x_3)) \tag{7}$$

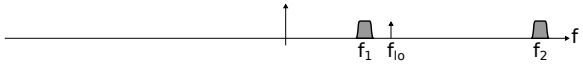
The 4-point DFT equations can therefore be simplified as a signed addition of the outputs of two other signed additions, which results in an adder tree. Though pipelining the adder stages leads to shorter critical paths and thereby higher

clock frequencies, it comes at the cost of introduced cycle delays. Consequently, the 4-point FFT has been configured to use zero, one or two pipeline stages and also supports both DSP slices and fabric logic implementation. Although Xilinx synthesis tool sets fabric logic as default, DSP slice implementation has been preferred as their utilization ratio is very small and fabric usage can be minimized by using DSP slices.

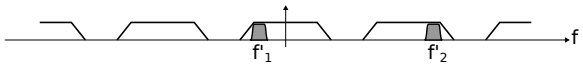
5.2 Integration into the Testing System

Although the channelizer is primarily meant to target QC systems, we use a generic, non-application specific test setup to enable testing and evaluation with different system parameters and for different application cases. The channelizer has been integrated into an existing data processing system[1] to test its functionality when the center frequencies of the channel do not exactly match the center frequencies dictated by the PFB structure. The test system needs to channelize two frequencies at $f_1 = 4492.63\,\text{MHz}$ and $f_2 = 4627\,\text{MHz}$ with the input signal being down-mixed using a configurable local oscillator (f_{lo}). The down-mixed signal is given as input to the 500 MHz ADC, which converts it into four parallel input data streams with 16-bit per sample at a sample rate of 125 MHz per channel.

Fig. 6a shows the absolute frequencies of the input signals before mixing with the oscillator frequency f_{lo}. By setting $f_{\text{lo}} = 4497.315\,\text{MHz}$ the input frequencies f_1 and f_2 are translated to f_1' and f_2' as shown in Fig. 6b. As each channel has a sample rate of 125 MHz, the center frequencies of the channels are placed at multiples of 125 MHz which in turn offsets the input frequencies by 4.685 MHz compared to the channel centers. The prototype filter's bandwidth must be large enough to include these frequencies and therefore, a passband width of 6 MHz with an acceptable passband ripple of 5 dBand a stopband starting at 7.5 MHz with stop band attenuation set to 40 dB are chosen as system parameters. This results in a 337-tap FIR prototype filter.



(a) Input signal before local oscillator



(b) Signal after local oscillator mixing

Fig. 6: Input signal and channel filter layout for the test system

[1] Our special thanks to Nick Karcher and Oliver Sander of IPE, KIT for providing the test setup.

Figure 7 shows a block diagram of the integrated system. As the ADC streams out four complex data streams in parallel, the commutator module has been omitted and the channelizer is implemented using only PFB and DFT modules. As the DFT outputs data at the same sampling rate of 125 MHz with 16-bit

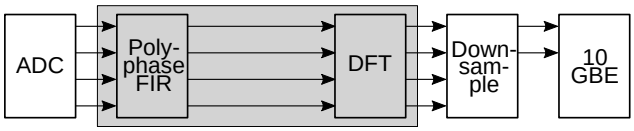


Fig. 7: Block diagram of polyphase channelizer integration

complex-valued samples, a data rate of 125 MHz $\times$ 16 bit $\times$ 2 $\times$ 4 channels is too demanding for the 10 Gbit/s Ethernet interface. Hence, a simple decimator module has been employed to downsample the output data to a sample rate of 125 MHz / 8 = 15.625 MHz by removing seven out of eight samples. No additional band-limiting filters are required as the bandwidth is sufficiently limited by the PFB. To evaluate the setup, a linear combination of complex phasors fed to a Digital to Analog Converter produced the input signal for the system. Verification of the results is done by examining the output data of the PFB channels using spectrum analysis techniques.

5.3 Results

Latency and Throughput. For hardware utilization analysis, a simpler test setup with only the channelizer has been synthesized using Vivado 2016.2 on a Virtex-7 FPGA platform. The setup uses a 28 coefficient FIR filter with 8 bit sample size and four channels. A single input bitstream with 200 MHz data rate is fed to the commutator. Then each channel transfers data to a prototype filter of length 7.

Table 1: Clock cycle latency

Configuration	FIR	Pre FFT		Post	Total
Transpose DFT	6	0	1	0	7
Transpose GDFT	7	0	1	1	9
Xilinx serial	-	-	-	-	16–19

As shown in Table 1, transpose DFT configuration requires only 6 clock cyles for FIR filter with $n-1$ stages being fully pipelined, where n denotes filter length and 1 additional clock cycle for FFT module. The GDFT implementation used for odd channel stacking has 1 clock cycle overhead due to the implementation of

complex filter coefficients. The Xilinx serial filter is included as a reference serial implementation. It processes the polyphase filter branches using a partially time multiplexed, optimized filter component and our standard, parallel DFT module. Because of some inherent parallelism in the Xilinx filter core combined with serial processing of branches, generating the output for all the n channels requires 16 to 19 clock cycles. The input samples are passed serially, but the output is provided in parallel so measured latency for different channels varies. The tested configuration does not require any preprocessing, so the latency introduced in this module is zero. Latency is calculated as the time required for the first input sample to be available at the output. Depending upon the channelizer's use case, another relevant metric is the group delay depending the FIR filter. It is generally lower than the presented delay metric. The throughput of the system can be calculated as the data input sample rate multiplied by the bit width of the complex samples: $50\,\text{MHz} \times 8\,\text{bit} \times 2 \times 4\,\text{channels}$.

Resource Usage. The resource usage per module has been charted down in Table 2. As expected since GDFT filter banks modulate the filters and therefore require complex filter coefficients, a higher LUT and DSP utilization can be observed for FIR filter in the GDFT configuration.

Table 2: Resource utilization per component

	Configuration	Commutator	FIR	FFT	Post	Other	Total
LUT	Transpose DFT	6	2	0	0	11	19
	Transpose GDFT	6	46	30	18	10	110
Register	Transpose DFT	131	8	1	0	35	175
	Transpose GDFT	131	8	1	35	35	210
DSP	Transpose DFT	0	50	6	0	0	56
	Transpose GDFT	0	67	6	0	0	73

6 Conclusion

The proposed reconfigurable FPGA-based polyphase channelizer has been tested and synthesized using Vivado 2016.2 on a Virtex-7 FPGA platform. The reconfigurability of FPGAs allows for fast development iterations and adaption to changing system requirements. When compared to per-channel approaches like traditional Digital Down Conversion, PFB operates filters at lower clock frequency. This allows PFBs to process large input bandwidths as made available by high throughput data interfaces of modern FPGAs. The PFB can be adapted to different application cases by configuring the number of channels, data width and throughput specifications. Different configurations using transpose filters

and DFT or GDFT transforms have been analyzed for a test setup with a 28 coefficient FIR filter and 4 channels. In order to balance latency against throughput, the user has the choice of optional pipelining. Additionally, resource utilization can be balanced against throughput by configuring the hardware parallelism of the FIR filter and DFT. The test results show the transpose PFB with DFT to yield throughput of 12.8 Gb/s with a latency of 7 clock cycles for a sampling frequency of 200 MHz. These results suggest that a PFB may well be used in high-bandwidth, latency critical QC systems. Additionally, the possibility to gauge spectral channel shape, latency and resource requirements is especially useful for these systems.

References

1. Lillington, J.: Comparison of Wideband Channelisation Architectures. International Signal Processing Conference (ISPC), Dallas (2003)
2. Meyer, J. et al.: Ultra High Speed Digital Down Converter Design for Virtex-6 FPGAs. 17th International OFDM Workshop 2012 (InOWo'12), 1–5 (2012)
3. Meyer, J. et al.: A Novel System on Chip for Software-Defined, High-Speed OFDM Signal Processing. 2013 26th Symposium on Integrated Circuits and Systems Design (SBCCI), 1–6 (2013)
4. harris, f.j.: Multirate Signal Processing for Communication Systems. Prentice Hall PTR, Upper Saddle River, NJ, USA (2004)
5. Crochiere, R.E., Rabiner, L.R.: Multirate Digital Signal Processing. Prentice-Hall, Eaglewood Cliffs, NJ, USA (1983)
6. harris, f.j., Dick, C., Rice, M.: Digital Receivers and Transmitters using Polyphase Filter Banks for Wireless Communications. IEEE Transactions on Microwave Theory Techniques 51, 1395–1412 (2003)
7. Wang, H., Lu, Y., Wang, X.: Channelized Receiver with WOLA Filterbank. 2006 CIE International Conference on Radar (2006)
8. Löllmann, H.W., Vary, P.: Low Delay Filter-Banks for Speech and Audio Processing. Speech and Audio Processing in Adverse Environments, Springer Berlin Heidelberg, Berlin, Heidelberg, 13–61 (2008)
9. Tuthill, J., Hampson, G., Bunton, J.D., harris, f.j., Brown, A., Ferris, R., Bateman, T.: Compensating for Oversampling Effects in Polyphase Channelizers: A Radio Astronomy Application. 2015 IEEE Signal Processing and Signal Processing Education Workshop (SP/SPE), 255–260 (2015)
10. Fahmy, S.A., Doyle, L.: Reconfigurable Polyphase Filter Bank Architecture for Spectrum Sensing. Reconfigurable Computing: Architectures, Tools and Applications: 6th International Symposium, ARC 2010, Bangkok, Thailand, March 17–19, 2010. Proceedings, Springer Berlin Heidelberg, Berlin, Heidelberg, 343–350 (2010)
11. Wu, F., Villing, R.: FPGA based FRM GDFT Filter Banks. 2016 27th Irish Signals and Systems Conference (ISSC)
12. Wu, F., Palomo-Navarro, Á., Villing, R.: FPGA Realization of GDFT-FB Based Channelizers. 2015 26th Irish Signals and Systems Conference (ISSC)
13. Adámek, K., Novotný, J., Armour, W.: A Polyphase Filter for Many-Core Architectures. Astronomy and Computing 16, 1-–16 (2016)
14. Chennamangalam, J. et al.: A GPU-Based Wide-Band Radio Spectrometer. Publications of the Astronomical Society of Australia 31 (2014)