

384 TMAC/s FIR filtering on an Artix-7 FPGA using Prism signal processing

John Owen^a

^aEntegra Solutions Ltd
Guildford, GU1 1YH, UK.
john.owen@entegra.co.uk

Manus Henry^{b,c}

^bDepartment of Engineering Science
University of Oxford
Oxford, OX1 3PJ, UK.
manus.henry@eng.ox.ac.uk

^cSchool of Electrical Engineering and Computer Science
South Ural State University
Chelyabinsk, Russia

Abstract—The Xilinx Artix-7 family of FPGAs has a peak DSP performance of approximately 1 TMAC/s. This paper describes a floating point FIR bandpass filter instantiation on an Artix-7 with up to 192 million taps, updated at up to 2 MHz, delivering an equivalent performance of up to 384 TMAC/s, while using only one third of the DSP resources on the FPGA. This has been achieved using Prism signal processing, a new recursive FIR filtering technique. Filter design is simple; a PC host HMI facilitates the immediate creation of new filter implementations. Examples of experimental results are provided.

Keywords—Artix-7, FPGA, Recursive FIR filtering, bandpass filtering, Prism signal processing.

I. INTRODUCTION

Since their first inception, Finite Impulse Response (FIR) filters have been viewed as inherently non-recursive calculations. FIR filters are normally formulated as discrete convolutions of the form:

$$y[n] = \sum_{i=0}^N b_i \cdot x[n-i] \quad (1)$$

Here the filter of order N generates an output sequence $y[n]$ from the sum of products of the filter coefficients b_i and the window of the $N+1$ most recent filter input values $x[n-i]$. The calculation is non-recursive as the coefficients are positional: on the next time-step equation (1) must be repeated in full over the shifted data window.

Convolution calculations similar to (1) are so common in digital signal processing (DSP) that the multiply and accumulate (MAC) operation is a widely used metric of device performance. For example, eqn (1) describes a calculation requiring $N+1$ MACs. To achieve real time throughput at a sample rate of f Hz for an FIR filter of order N requires a computational performance of $f \cdot (N + 1)$ MAC/s. The largest device in the Artix-7 FPGA family, the XC7A200T, has a nominal DSP performance of 929 GMAC/s (1 GMAC = 10^9 MACs) [1, 2]. This performance calculation assumes all of the 740 DSP ‘slices’ in the FPGA operate in parallel. In practice such performance is not necessarily achieved for any specific application, due to unavoidable inefficiencies entailed in the details of the implementation.

Various approaches are reported in the literature for developing recursive FIR filters, typically using specific mathematical structures which support efficient computation. These include: adapting IIR filters to approximate FIR behaviour [3, 4], for example the switching and resetting of multiple IIR filters [5]; the Cascade Integrator Comb [6, 7], which cascades the moving average filter (the only ‘naturally’ recursive FIR filter); and the construction of piecewise polynomial approximations to the impulse response [8-11].

In this paper we describe a recursive FIR bandpass filter application implemented on an Artix-7 which achieves the equivalent of up to 384 TMAC/s (3.84×10^{14} MAC/s) i.e. approximately 400 times faster than the nominal performance, and using floating point rather than fixed point arithmetic. The implementation is not optimal – for example only around 1/3 of the DSP slices is used – and further performance improvements are undoubtedly possible. The calculation is organized as a single bandpass filter, operating on ADC data sampled at up to 2 MHz. The high MAC performance is achieved via a high filter order (up to 192 million). The key to making this possible is Prism Signal Processing (PSP) [12], a new type of FIR filter which is fully recursive i.e. the computational effort is independent of the filter length. Our performance claim is therefore of *equivalence*, based on the computational load required to deliver a conventional non-recursive FIR filter of the same order and update rate.

The bandpass filter consists of a chain of six Prisms. The design of each Prism is straightforward, once its required parameters values are known. A simple higher level calculation maps the desired bandpass filter characteristics (central frequency, bandwidth) onto the corresponding parameters for each Prism in the chain. An HMI on the host computer enables new bandpass filters to be designed and implemented on the FPGA instantly.

The paper is organized as follows. Section II gives a brief overview of PSP and the design calculation for the bandpass filter. Section III describes the FPGA-based implementation of the bandpass filtering demonstrator, while Section IV gives experimental results obtained from the system, including results for a 192 million order, 384 TMAC/s filter.

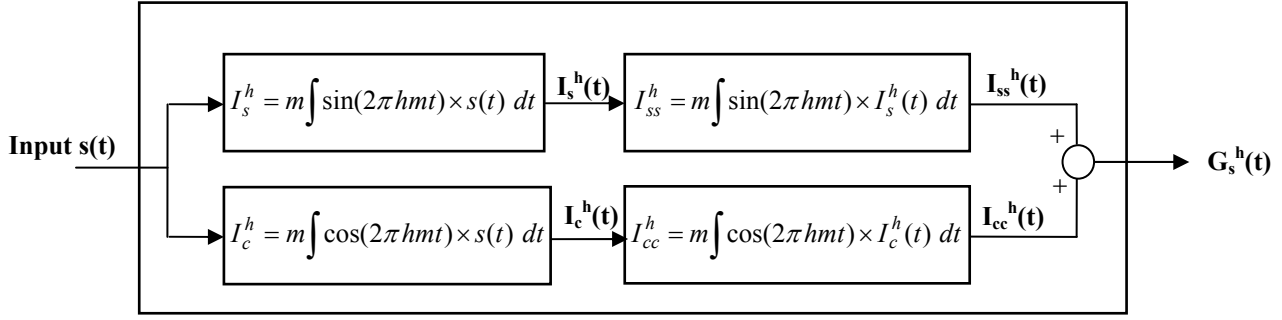


Fig. 1. Prism structure generating a single output G_s^h .

II. PRISM BANDPASS FILTERING

The Prism (Fig. 1) is an FIR filter generating one or two outputs. Its properties are defined by its characteristic frequency m and harmonic number h [12]. Structurally, the Prism consists of four or six integration blocks, where the input to each is multiplied by a sine or cosine function with frequency mh Hz and integrated over the period $1/m$ s to generate the block output. This calculation can be performed recursively [12], as the final Prism outputs are independent of the instantaneous phase of the sinusoidal modulation functions. In the Prism, the equivalent of filter coefficients are linearly spaced sine and cosine values, so that, for desired values of m and h , the design calculation is straightforward.

A companion paper to this conference provides more background on the Prism and sets out the theory of how a bandpass filter can be constructed using a chain of six single output Prisms, arranged as three pairs (Fig. 2). A Prism-based Recursive Signal Tracker (RST) [13] may be used to calculate the frequency, amplitude and/or phase of the filtered signal. Given the desired bandwidth b and central frequency c , simple rules are used to calculate the corresponding values of m and h for each of the six Prisms. Fig. 3 shows how the relative gain of the bandpass filter varies with distance from c , where the gain at $c \pm b/2$ Hz is -3 dB.

III. DEMONSTRATION SYSTEM

The demonstration system consists of a PC hosting an Innovative Integration Inc XA-160M module [14]; this has dual 16 bit ADCs and DACs, an Artix-7 XC7A200T FPGA and a 4 lane PCIe link to the host. Input is provided by a dual channel signal generator: one channel provides the sinusoidal signal to be tracked, while the second provides an interference signal, such as white noise or a sinusoid with an adjacent frequency. The host PC controls the bandpass filter design, filter operation, and data storage of experimental results.

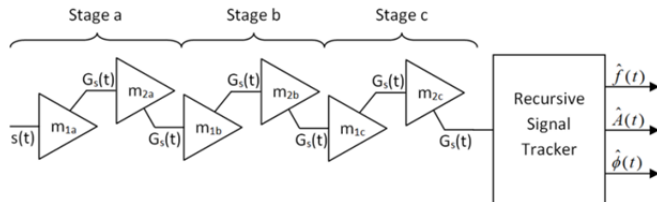


Fig. 2. Bandpass filter from three pairs of single output Prisms, and a Prism-based tracker generating frequency, amplitude and phase estimates.

Fig. 4 outlines the arrangement. The XA-160M card samples the two inputs separately (so they can be recorded individually) but then sums the digitized values and feeds the sum into the six Prism filter. One disadvantage of the Prism compared with a conventional FIR filter is its relatively high memory requirements. Given a sampling rate f_s and Prism parameter m , and defining $l = f_s/m$, from Fig 1 it can be deduced that approximately $8l$ storage locations are required for each Prism, as follows: sine and cosine vectors of length l , and a shifting window of l product values for each of the six integration stages; this is in addition to a relatively small number of totalizers, control variables etc. Viewed as a conventional FIR filter, however, the equivalent order of the Prism is only $2l$ (consider the concatenation of the two layers of integration blocks in Fig. 1). To achieve a balance between precision and execution speed in the FPGA, the sine/cosine and product vectors are stored as 32 bit floating point values, while the totalizers are 64 bit floating point values.

To facilitate the creation of long bandpass filters, the data needed for each filtering Prism is held in the 8 Gbyte RAM of the host PC. This arrangement is one of the limiting factors on total system data throughput (~ 2 M samples/s). The faster 1 Gbyte RAM on the XA-160M card is currently used for buffering of PCIe transfers; higher throughput, especially for smaller filters, may be possible with a different arrangement.

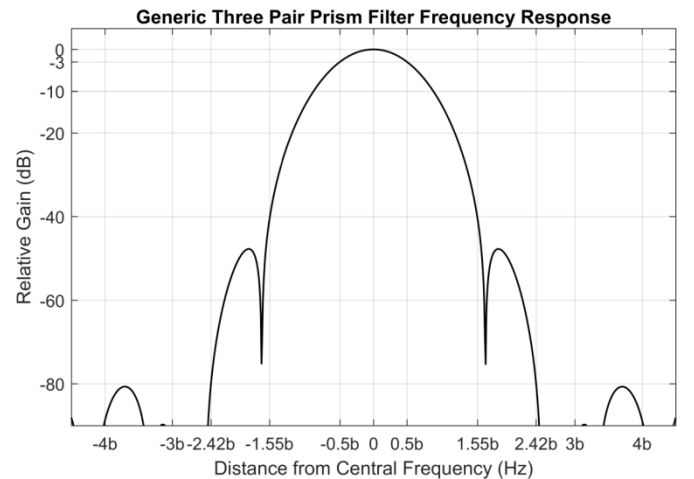


Fig. 3. Frequency response of bandpass filter formed from three Prism pairs.

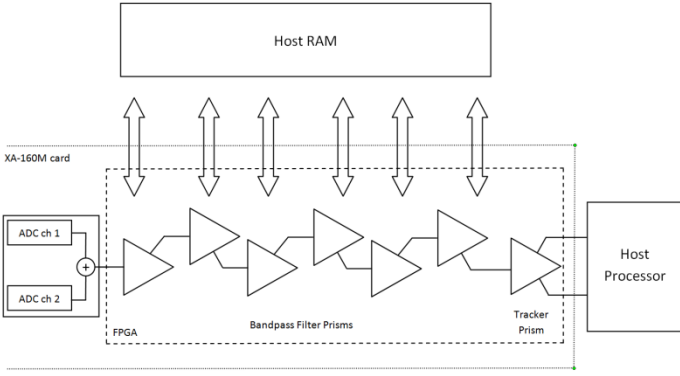


Fig. 4. FPGA-based implementation of bandpass filtering.

The functionality of the tracker in Fig. 2 (used to generate frequency, phase and amplitude values from the filter output) is split between the Prism and host PC. As explained in [13], the RST consists of a dual output Prism (here instantiated within the FPGA), and further calculations based on the Prism outputs. As the latter include calls to trigonometric and square root functions, here these are carried out on the host PC.

Prism algorithms previously coded in C++ were used as the basis for the FPGA coding in VHDL. The logic development tool Vivado™ includes a simulator enabling the VHDL code to be debugged by comparing outputs with the C++ code. Table 1 shows the Artix-7 resource usage for the FPGA design, which includes the 6 + 1 Prisms and the interfaces to the ADCs, host, and host memory interfaces. Note that less than 30% of the DSP48 slices are currently used, so there is scope for further design improvements, for example introducing additional pipelining to improve data throughput.

Fig. 5 shows a section of the PC HMI: a new filter design can be created by supplying the desired central and bandpass frequencies. The corresponding values of m and h for each of the Prisms are calculated and downloaded to the bandpass

Fig. 5. Section of PC host HMI interface.

filter on the FPGA, along with the linearly spaced sine and cosine values for the modulation functions.

IV. EXPERIMENTAL RESULTS

A series of tests have been carried out to evaluate the performance of the Prism-based bandpass filtering system. The signal generator used was an Agilent 33522B Waveform Generator; a reference clock signal was shared between the Agilent and the test system to ensure timing consistency.

The first set of two tests had the following common characteristics: a sample rate of 2 MHz, a central frequency of 5000 Hz, and a passband of 0.02 Hz. This results in a bandpass filter of order 192 million, and an equivalent DSP performance of 384 TMAC/s.

In both experiments, the frequency component to be tracked is at 5000 Hz, with an amplitude of 1 mV zero-peak. In the first experiment, the interference signal, which is added to the desired signal, is at 5000.1 Hz (i.e. at a distance $5b$ from the central frequency, c) and has an amplitude of 1 V zero-peak i.e. 1000 times greater than the adjacent signal component to be tracked. Fig. 6 shows the frequency spectrum of the combined input signal around the central frequency 5000 Hz; the two adjacent frequency components with their respective amplitudes are clearly discernable.

Fig. 7 shows the frequency spectrum of the bandpass filtered signal. The noise floor, which in Fig. 6 is steady at around $1e-5$ V, has dropped to around $1e-8$ V at a distance of 0.5 Hz or greater from c . As intended, the interfering signal at 5000.1 Hz has been effectively attenuated, leaving the desired signal component at 5000Hz as the highest peak. Fig. 8 shows the amplitude calculated by the RST which remains with 5% of its nominal value of 1 mV.

Figs. 9, 10 and 11 are similar plots for the second test using $b = 0.02$ Hz, where the interference signal is white noise with an amplitude of 0.5 V and a bandwidth of 10 MHz. The plots show a ± 10 Hz region around the central frequency. The high noise floor (Fig. 9) is readily suppressed by the filter (Fig 10), with the amplitude approaching $1e-16$ V at a distance of 10 Hz from the central frequency. Amplitude tracking based on the filter output (Fig. 11) yields similar results to those from the previous example (Fig 8).

While this first pair of examples demonstrates a very narrow passband filter and high DSP performance, the dynamic response of the filter is of the order of 100 seconds, which limits the system's ability to demonstrate successful tracking of changes in component parameter values. Accordingly, a second pair of examples is presented which is

TABLE I. ARTIX-7 XC7A200T RESOURCE UTILIZATION

Resource	Available	Utilization	
		Quantity	%
LUT	133800	59087	44.16
LUTRAM	46200	3169	6.86
FF	267600	102832	38.43
BRAM	365	144.50	39.59
DSP	740	218	29.46
IO	386	283	73.32
GT	8	8	100.00
BUGF	32	16	50.00
MMCM	10	4	40.00
PLL	10	1	10.00
PCIe	1	1	100.00

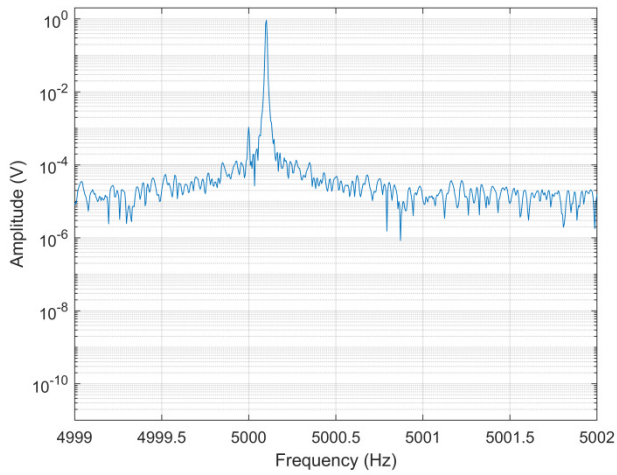


Fig. 6. Spectrum of input signal; components at 5000 Hz and 5000.1 Hz.

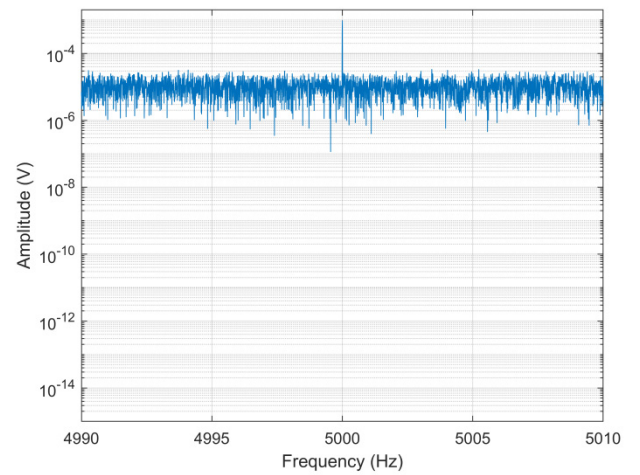


Fig. 9. Spectrum of input signal; component at 5000 Hz with white noise.

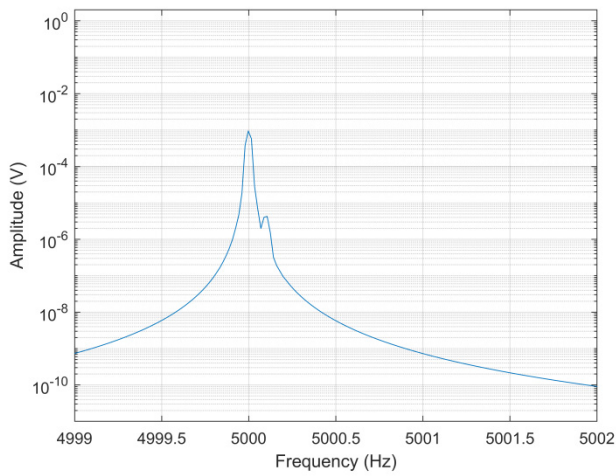


Fig. 7. Filtered signal: 5000.1 Hz component has been attenuated.

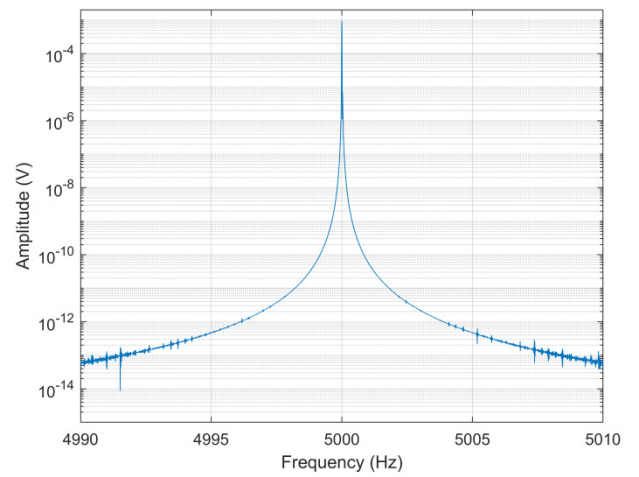


Fig. 10. Spectrum of filtered signal; white noise has been attenuated.

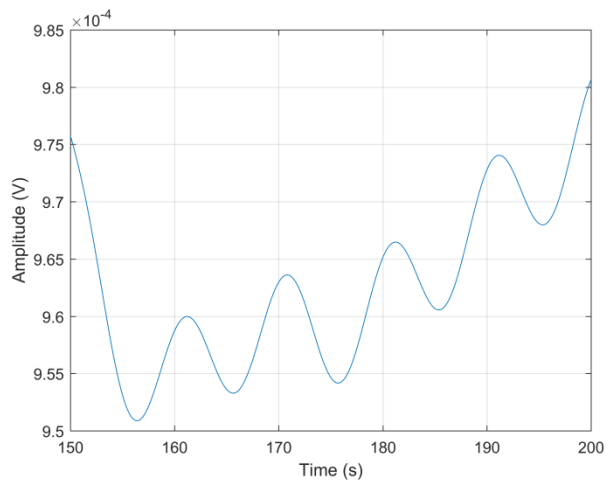


Fig. 8. Amplitude value calculated by tracker from filtered signal.

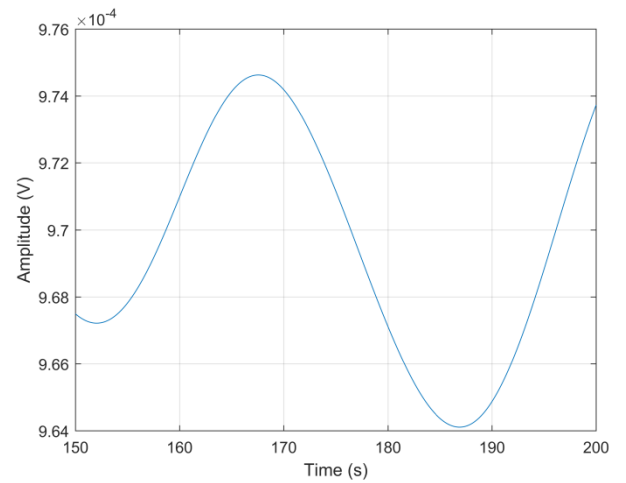


Fig. 11. Amplitude value calculated by tracker from filtered signal.

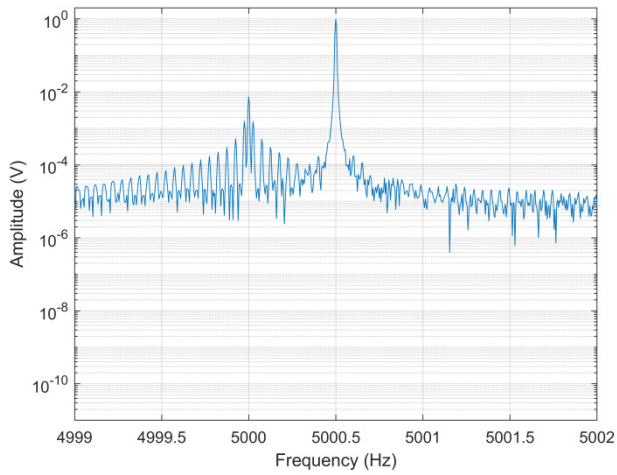


Fig. 12. Spectrum of input signal; components at 5000 Hz and 5000.5 Hz.

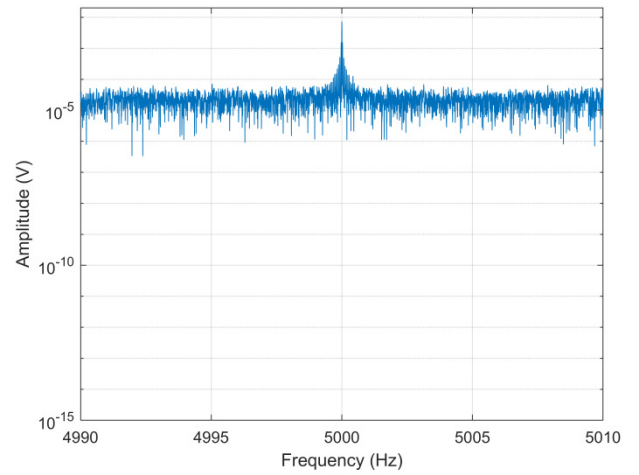


Fig. 15. Spectrum of input signal; component at 5000 Hz with white noise.

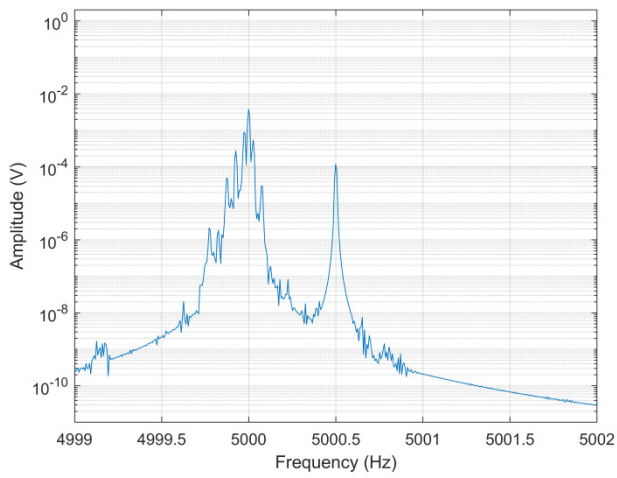


Fig. 13. Filtered signal: 5000.5 Hz component has been attenuated.

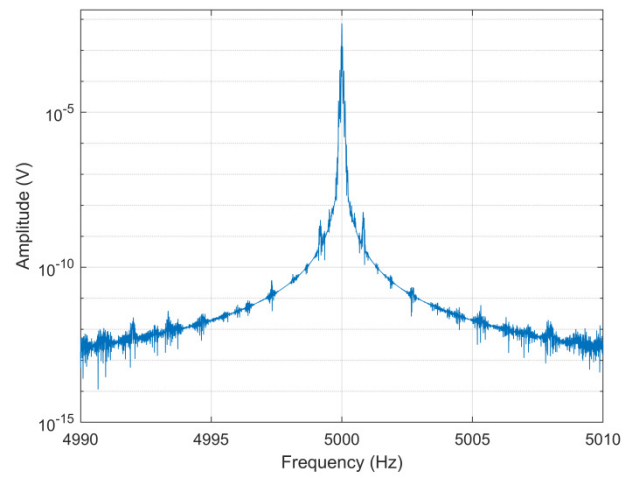


Fig. 16. Spectrum of filtered signal; white noise has been attenuated.

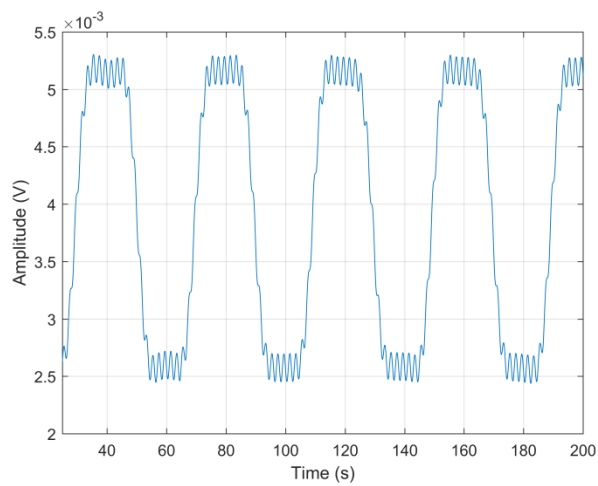


Fig. 14. Tracking amplitude step changes in filtered signal.

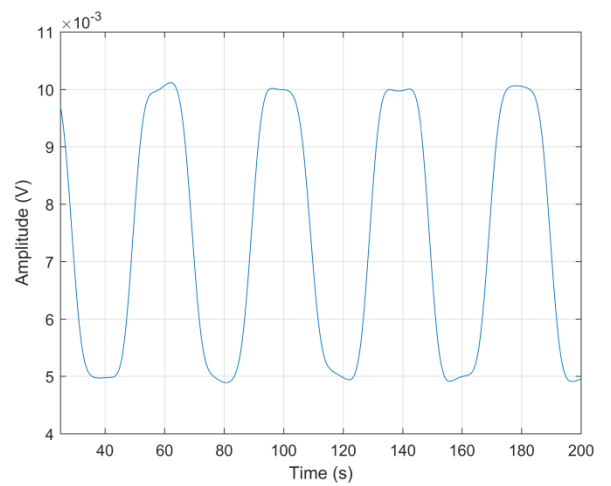


Fig. 17. Tracking amplitude step changes in filtered signal.

similar to the first pair, but where the filter bandwidth is extended so that step changes in the amplitude of the tracked signal component can be tracked.

In the second pair of tests, the sampling rate remains 2 MHz and the bandpass central frequency c remains 5000 Hz, but the bandwidth b is increased to 0.1 Hz. The corresponding filter order is 38 million samples, so the equivalent DSP performance is 76 TMAC/s.

In the first of these tests, the amplitude of the 5000 Hz frequency component steps between 2.5 mV and 5 mV every 20 s, while the interference component is at 5000.5 Hz (i.e. $5b$ away) with a fixed amplitude of 1 V. Fig. 12 shows the spectrum of the input signal, which has additional peaks at intervals of 0.05 Hz due to the period of the amplitude step changes. The spectrum of the filter output (Fig. 13) shows that the frequency component at 5000.5 Hz has been significantly attenuated. Finally, in Fig. 14 the tracker output shows the amplitude step changes every 20s.

The second test once more uses 1 V of white noise, while the 5000 Hz component steps between 10 mV and 20 mV every 20s. Figs. 15 and 16 show the spectra of the input signal and filter output respectively, while Fig. 17 demonstrates that the step changes in the 5000 Hz signal are successfully tracked.

V. SUMMARY

This paper has described an implementation of a Prism-based bandpass filter demonstrator using an FPGA card and a PC host. Filters with orders as high as 192 million can be designed in real time and instantiated on the FPGA. Experimental results using a signal generator demonstrate that the bandpass filters are effective in isolating desired signal components and removing unwanted components. While the Artix-7 FPGA DSP performance is rated at less than 1 TMAC/s, the recursive FIR Prism-based filters achieve an equivalent DSP performance of up to 384 TMAC/s, using only a fraction of the resources of the FPGA.

There is significant scope for improving the performance described in this paper. In particular, it is desirable to increase

the throughput of the filter implementation so that faster sampling rates can be used. Most obviously this can be achieved by using more FPGA resources, and providing access to faster memory resources.

VI. REFERENCES

- [1] Xilinx. 7 Series FPGAs Data Sheet Overview. DS180 (v2.6) Feb 27, 2018. https://www.xilinx.com/support/documentation/data_sheets/ds180_7_Series_Overview.pdf
- [2] <https://www.xilinx.com/support/answers/41803.html>
- [3] R. Lyons and A. Bell, "The swiss army knife of digital networks," in IEEE Signal Processing Magazine, vol. 21, no. 3, pp. 90-100, May 2004.
- [4] T. G. Campbell and Y. Neuvo, "Predictive FIR filters with low computational complexity," in IEEE Transactions on Circuits and Systems, vol. 38, no. 9, pp. 1067-1071, Sep 1991.
- [5] T. Saramaki and A. T. Fam, "Properties and structures of linear-phase FIR filters based on switching and resetting of IIR filters," IEEE International Symposium on Circuits and Systems, New Orleans, LA, 1990, pp. 3271-3274 vol.4.
- [6] E. Hogenauer, "An economical class of digital filters for decimation and interpolation," in IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. 29, no. 2, pp. 155-162, Apr 1981.
- [7] M. Mottaghi-Kashitaban, A. Jalali, "FIR filters involving shifts and only two additions, efficient for short word-length signal processing," Microelectronics Journal, Volume 49, 2016, Pages 57-63
- [8] S. Chu and S. Burrus, "Efficient recursive realizations of FIR filters, Part I: The filter structures," Circuits Syst. Signal Process., vol. 3, no. 1, pp. 3-20, 1984.
- [9] S. Chu and S. Burrus, "Efficient recursive realizations of FIR filters, Part II: Design and Applications," Circuits Syst. Signal Process., vol. 3, no. 1, pp. 21-57, 1984.
- [10] R. Lehto, T. Tauren, and O. Vainio, "Recursive FIR filter structures on FPGA," Microprocess. Microsyst., vol. 35, pp. 595-602, Oct. 2011.
- [11] K. Mukumoto and T. Wada, "Realization of Root Raised Cosine Roll-Off Filters Using a Recursive FIR Filter Structure," in IEEE Transactions on Communications, vol. 62, no. 7, pp. 2456-2464, July 2014. doi: 10.1109/TCOMM.2014.2329672
- [12] MP Henry, F Leach, M Davy, O Bushuev, MS Tombs, FB Zhou, and S Karout, "The Prism: Efficient Signal Processing for the Internet of Things", IEEE Industrial Electronics Magazine, pp 2-10, December 2017. DOI: 10.1109/MIE.2017.2760108.
- [13] F. Leach, S. Karout, F.B. Zhou, M.S. Tombs, M. Davy, and M.P. Henry, "Fast Coriolis mass flow metering for monitoring diesel fuel injection", Flow Measurement and Instrumentation, 58 (2017), pp 1-5.
- [14] <http://innovative-dsp.com/product/xa-160m-two-160-msps-16-bit-adc-two-615-msps-16-bit-dac-artix-7-fpga/>