Reconfigurable Polyphase Filter Bank Architecture for Spectrum Sensing

Suhaib A. Fahmy[1] and Linda Doyle[2]

[1] School of Computer Engineering, Nanyang Technological University, Nanyang Avenue, Singapore 639798
sfahmy@ntu.edu.sg

[2] Trinity College Dublin, College Green, Dublin 2, Ireland
linda.doyle@tcd.ie

Abstract. This paper presents a brief tutorial and background on implementing filter banks for spectrum sensing. It discusses the advantages of this approach over standard FFT-based spectral estimation. A general architecture for implementation of filter banks on FPGAs is then presented, exploiting heterogeneous resources. It takes advantage of the fact that subband filters run at a reduced sample rate, and hence can share the same computational resources. We show how to facilitate this sharing of resources by saving state for each subband. The architecture is fully reconfigurable, allowing sensing parameters to be changed at runtime. Resource usage figures are given, showing the efficiency of the architecture. We finally discuss how this architecture can be adapted to signal reception.

1 Introduction

The Fourier transform (FT) is perhaps the most widely used, and generally applicable component of the signal processing toolbox. It allows us to obtain the frequency domain representation of a time domain signal. Unfortunately, using the FT for signal analysis in DSP systems, we face a problem due to the discrete nature of digital systems. The abrupt edges in the time domain translate into sinc functions in the frequency domain. This results in what is called spectral leakage; where some of the energy in a bin "leaks" into adjacent bins. This gives an inaccurate spectral footprint, especially when adjacent bins have vastly differing amplitudes, as illustrated in Fig. 1. This inaccurate representation can underestimate available spectrum for a cognitive radio. This can be overcome by windowing in the time-domain. Polyphase filter banks are an efficient way of accomplishing this.

Filter banks are an established structure used in multirate signal processing and have applications in compression and image processing. [1] When trying to decompose signals into subbands, we typically modulate a prototype low-pass filter to the appropriate centre frequencies. This requires as many different low-pass filters as there are subbands, and each of these must be run at the input sample rate. Fig. 2 shows a frequency domain representation of the filters.

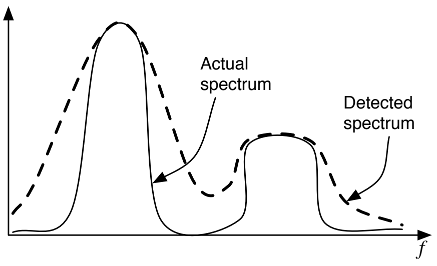

Fig. 1. Spectral leakage using PSD estimation

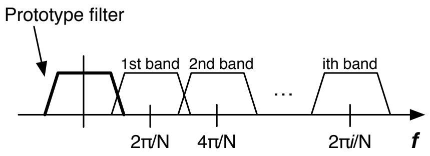

Fig. 2. Frequency representation of a filter bank

Polyphase filter banks enable us to do this more efficiently, by decomposing a single *prototype filter*, into subfilters and effectively modulating these using an FFT. These filters can then be run at a fraction of the original sampling rate. While filter banks are not new, a number of exciting candidate applications, specifically in the area of cognitive radio, are reawakening research in the area, while demanding much larger dimensions than have been previously required. In this paper, we present a brief tutorial on the theory behind polyphase filter banks, we analyse the computational requirements, then present a reconfigurable FPGA-based architecture for polyphase filter banks for spectrum sensing.

Using filter banks for spectrum sensing offers a number of advantages. Firstly, we can scan wide portions of the spectrum (high input sampling rate), efficiently (using fewer resources at a lower sampling rate). Secondly, the results of spectrum sensing using filter banks are far more accurate than basic energy detection using an averaged FFT. Finally, the use of filter banks for transmission and reception of signals is an area that has gained renewed interest at present. It is envisaged that such techniques could supersede current multicarrier transmission methods. [2] It would then be possible to use the same computational structure for sensing and receiving; an attractive proposition for resource limited cognitive radios. The main contribution of this paper is the scheme by which we are able to share the filtering circuitry between subbands while maintaining state for each of them.

This architecture will be used within an FPGA-based cognitive radio framework [3] to explore aspects of spectrum sensing and wideband transmission; hence the importance of a reconfigurable design. We present a brief background in Section 2, followed by an overview of related work in Section 3. We then present the proposed architecture and implementation results in Sections 4 and 5, respectively.

2 Background

Frequency division multiplexing (FDM), allows us to combine multiple transmission streams by arranging them adjacently in the frequency domain. Typically, we split the overall bandwidth of the FDM signal into N subbands, modulating data onto each subcarrier independently. Protocols like Orthogonal Frequency Division Mutiplexing (OFDM) allow us to modulate several independent streams of data across a wide band, overcoming the weaknesses of narrow band techniques applied to wide bands, in terms of robustness.

One of the primary tasks of a cognitive radio (CR) is spectral estimation, that is, determining portions of the spectrum that can be used for transmission. This allows the CR node to establish communication with other nodes without interfering with primary users. This is most simply done by using an FFT to provide a spectral estimate. However, an FFT only provides an instantaneous representation of the spectrum, so this must be averaged over a number of windows to give a spectral estimate that is of any use. When taking an FFT over a fixed sized window, spectral leakage can occur. This is due to the sharp edges of a square window causing what should be delta functions in the frequency domain to become sinc functions, the sidelobes of which can interfere with adjacent frequency bins.

The standard solution for obtaining more accurate estimates is to use windowing to weight the samples in the input window. Windowing over multiple overlapping windows provides more accurate results, and can be implemented using a polyphase filter bank. [4]. An N-band polyphase filter bank consists of a commutator that switches input samples into one of N subband filters successively. The coefficients of the rth subband filter, $h_r(n)$ are derived from the prototype filter, $h_\mathrm{P}(n)$, using the following identity:

$$h_r(n) = h_P(r + nN) \tag{1}$$

Hence, for a prototype filter with $N \times M$ coefficients, decomposed into N bands, each filter has M coefficients. This can be seen as a lexicographic reordering; rearranging the 1-dimensional $h_\mathrm{P}(n)$ coefficients row-wise into N rows, then reading back the coefficients column-wise. The outputs of the filters are then connected to an N-point Discrete Fourier transform, which creates the modulation effect on the filter responses. When N is a power of 2, we can use an FFT, for which highly efficient architectures exist. A general polyphase filter architecture is shown in Fig. 3.

Each subband filter only operates on $1/N$th of the input samples. So filter $h_0(n)$ operates on inputs x_0, x_N, x_{2N}, $\cdots$, filter $h_9(n)$ operates on inputs x_9, x_{N+9}, x_{2N+9}, $\cdots$, and so on. These subband filters are thus run at $1/N$th of the input sample rate. [5] This is what makes this method so attractive from the computational perspective: the performance requirement in the subband filters is $1/N$th of that for running them at full sample rate. Or alternatively, we can share the same computational circuitry between the subband filters.

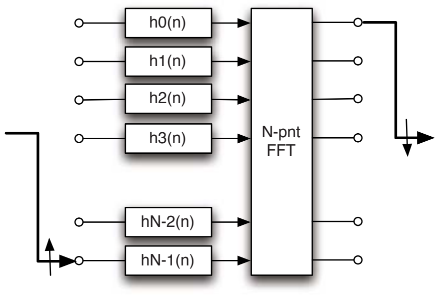

Fig. 3. The polyphase decomposition

3 Previous Work

FPGAs are ideally suited to accelerating filters, due primarily to the provision of embedded multipliers and multiply accumulate (MAC) units on modern devices. Typically, finite impulse-response (FIR) filters are accelerated in hardware by parallelising the MACs; a filter with k coefficients requires k MACs to run at maximum speed. A more compact implementation can use fewer MACs, but at a cost of more clock-cycles per result. Implementing FFTs on FPGAs is straightforward nowadays, with efficient, feature-rich cores provided by device vendors.

Most hardware implementations of polyphase filter banks so far have focussed on much smaller problems. For spectrum sensing or filter bank-based communications, we typically consider 128–2048 subbands and prototype filters of increased dimensions. In [6], a 16-band polyphase filter bank architecture is presented, along with basic optimisations of wordlengths and multiplications. In [7], a wideband channeliser is implemented on a Xilinx Virtex-4 FPGA, to extract frequencies of interest and recombine them in a narrower bandwidth. The number of bands in this implementation is 32, and synthesis results are given. In [8], we are presented with a polyphase filter bank implementation used in MP3 decoding, again with a 32-band decomposition. This implementation is focussed on the MP3 standard, and little discussion of architectural optimisations is presented.

The Xilinx FIR Compiler core [9] does offer a Channeliser mode, however this only allows for a fixed implementation, and the specific architecture used for the implementation is not discussed. We have been unable to identify any work that focusses on polyphase filter bank implementations for spectrum sensing or for reception. The dimensions of the problem differ significantly, and this is what we tackle in this paper. This architecture facilitates much larger prototype filters, a significantly increased number of subbands, and allows for runtime reconfiguration of the filter bank parameters. We also focus on how state can be preserved between application of the computational core to the different subbands in turn.

4 Proposed Architecture

4.1 Overview

Our aim is to create a general filter bank architecture that can be used for both spectrum sensing and for reception as part of a cognitive radio system. The basic structure of polyphase filter banks was discussed in Section 2. An input signal is commutated to a bank of FIR filters, the coefficients of which are derived from the prototype filter using the method explained in Section 2. If we have N subbands, then each subfilter is only activated once for every N inputs. Hence, we can design a compact architecture by sharing the same filter hardware among the subband filters and running it at the input sample rate. The outputs of the filters are fed through to an FFT for combination.

4.2 Efficient Implementation

The challenge in sharing a single filter engine between the subband filters arises because we need to be able to access a new set of coefficients in each clock cycle, and store the intermediate state of the filter once the next sample has been received into the filter. This would allow an N-band filter bank, with M coefficients per subband filter, to consume the same area as an M point FIR filter plus whatever resources are necessary for storing N different states. Fig. 4 shows an overview of the proposed architecture.

Prior to processing, we must load the coefficient memories. To preclude the need for a large memory to store the prototype filter prior to distribution, we distribute the coefficients directly into the combined memories. The *load_coeffs* input is pulsed. In the following cycle, the filter coefficients are fed in, one per clock cycle on the *coeff_in* input. The values of *num_bands* and *num_sbcoeffs*

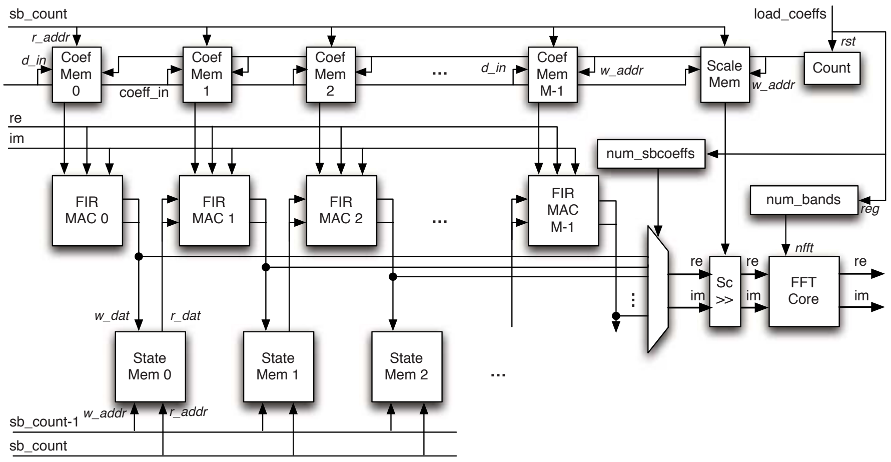

Fig. 4. Proposed architecture

determine how the memories are written to. A shift register with a single 1 is enabled when loading coefficients. This is connected to the write enable signal of each of the memories, and shifts in each cycle, writing into subsequent memories. *num_sbcoeffs* determines when this shift register resets to the first index, hence writing into the second address of the first memory. A counter that counts up after each complete row of coefficients serves as the write address. Once the coefficients are loaded, the circuit switches to processing mode, which is depicted in Fig. 4.

The filters in our system have real coefficients but operate on complex inputs; effectively, we have a duplicate filter for the imaginary part of the input, as for the real part. We implement the filter core in transposed form. Each MAC has a single cycle latency. To manage the sharing of the coefficients and storage of intermediate data, we keep track of the current subband index in the register *sb_count*. This is a counter which counts up in every clock cycle. This counter is used to address the coefficient memory, loading the correct coefficients for the subband being processed.

Storing the intermediate state of the filter is accomplished in a similar fashion to the coefficient control. Each MAC has an associated state memory, which stores the output of the MAC into a location one cycle-delayed from the coefficient index. While we store the result for one subband, we are loading the state for the next. Since we have dual-port memories on the FPGA, this arrangement does not present any problems. By ensuring the index count loops at the correct maximum count (*num_bands*−1), the data is correctly dealt with from one complete iteration through the bands to the next.

The output of the filters is then rescaled (discussed in Section 4.3), before being passed to the FFT core, the result of which, represents the system output. The serial output of the FFT represents the combination of subband outputs as per the polyphase definition.

4.3 Wordlength Considerations

When dealing with very large filters, we expect to see some very small coefficients. In the absence of floating point arithmetic units, we can either use wide fixed point representations or manage scaling manually. Rather than make the resource sacrifice necessary for wide filters, we choose to take advantage of a property of the subband filters. Since each filter is independent of the next, it is feasible to scale the coefficients of each filter independently of the others for the filtering operation, as long as the scales are restored when the outputs of the subband filters are combined. This gives us a significant area saving, while maintaining filter accuracy.

In this implementation, the input samples have 16-bit signed real and imaginary 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 coefficients to be input as 32-bit numbers along with a scale value. The scale value for each MAC is stored in a memory, and is used at the output of the filter to correctly rescale the samples before they are passed to the

FFT. The scale value must be computed in advance by the user, but is simply a case of finding the maximum dynamic range for each subband filter and scaling by a power of 2.

4.4 Reconfiguration

Changing the number of subbands or the size of each subband filter is facilitated at runtime. The *load_coeffs* signal is simply asserted with new values for *num_bands* and *num_sbcoeffs*. New coefficients are then fed into the system. Such a reconfiguration is applied by changing the count limit for the commutator, adjusting which MAC output feeds into the FFT, and changing the FFT point size (which is a supported feature of the Xilinx FFT Core). In this way, we can dynamically switch the properties of the architecture to suit different sensing and reception requirements.

5 Initial Results

The architecture has been synthesised for the Xilinx Virtex-5 XCVFX70T, as found on the ML507 Development Board which is used for our cognitive radio implementations. The architecture itself can achieve a maximum clock speed of 150MHz (or 125MHz as we exceed 80% device usage), which exceeds the 100MHz bus requirement for our implementation framework. We initially provide results for a reconfigurable implementation with a maximum subband count of 1024.

Resource usage can be estimated in advance, based on the maximum allowable values for *num_bands* and *num_sbcoeffs*. While these values can be changed at runtime, we must set limits, which determine the resource usage. The maximum number of subbands determines the depth of the memories used for storing coefficients and state information. The maximum number of subband coefficients determines the number of MAC units and hence the number of memories required. For 16-bit coefficients, we can store up to 1024 entries in a single 18Kb Block RAM. Note that the Block RAM count returned by the tools for a Virtex-5 device counts 36Kb Block RAMs (which can be used as two 18Kb Block RAMs). Hence, reducing the number of bands does not reduce the memory usage for storing the coefficients.

The state memories store 64-bit data (concatenated 32-bit real and imaginary parts), hence for 1024 bands, we would require four 18Kb (or two 36Kb) Block RAMs per state memory. For 512 bands, we would require two 18Kb (one 32Kb) Block RAMs, and for 256 or fewer bands, we would require one 18Kb Block RAM per state memory. We expect logic usage of the filter portion not to change with a reduced number of bands, as the only difference will be the width of the address counters.

The design's area is dominated by BlockRAM and DSP48E usage, though the FFT core adds considerable logic usage. For a 1024 band implementation, we would be restricted to a maximum filter size of 57 taps, though we expect the clock speed to be reduced when nearing 100% usage.

Table 1. FPGA resource utilisation

Filter Coeffs	LUT/FF Pairs	BlockRAMs	DSP48Es
8	1,120	20	16
16	1,760	40	32
32	3,170	80	64
48	4,480	120	96
1024pt FFT	7,100	4	13
Available	44,800	148	128

We believe performance can be increased by further pipelining of the memories and MAC units, though this is not required for our purposes.

6 Conclusion

By exploiting the embedded memories and multipliers on modern FPGAs, we have been able to design an efficient generalised architecture for polyphase filter banks. It allows a single filter structure to be used by alternate subbands in consecutive clock cycles. The architecture scales up to the large dimensions required for spectrum sensing, and due to the memory-based implementation facilitates reconfiguration at runtime. By incorporating this design into our Cognitive Radio framework [3], we can time-multiplex sensing with other functions of a cognitive radio system on a smaller device. The reconfigurability of this architecture makes it ideally suited to explorative research in the area.

References

1. Vaidyanathan, P.: Multirate Systems And Filter Banks. Prentice Hall PTR, Englewood Cliffs (1992)
2. Farhang-Boroujeny, B., Kempter, R.: Multicarrier communication techniques for spectrum sensing and communication in cognitive radios. IEEE Communications Magazine 46(4), 80–85 (2008)
3. Fahmy, S., Lotze, J., Noguera, J., Doyle, L., Esser, R.: Generic software framework for adaptive applications on FPGAs. In: IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM), Napa, CA (April 2009)
4. Harris, F.: On detecting white space spectra for spectral scavenging in cognitive radios. Wireless Personal Communications 45(3), 325–342 (2008)
5. Harris, F.: Multirate Signal Processing for Communication Systems. Prentice Hall PTR, Englewood Cliffs (2004)
6. Fiore, P.: Low-complexity implementation of a polyphase filter bank. Digital Signal Processing 8(2), 126–135 (1998)
7. Egg, B., Harris, F., Dick, C.: Ultra-wideband 1.6GHz channelizer: Versatile FPGA implementation. In: Proceedings of the SDR 2005 Technical Conference and Product Exposition (2005)
8. Valdés, M., Moure, M., Diéguez, J., Antelo, S.: Hardware implementation of a polyphase filter bank for MP3 decoding. In: Proceedings of Southern Conference on Programmable Logic (SPL), pp. 19–24 (2008)
9. Xilinx Inc.: FIR Compiler v4.0 Product Specification (DS534) (June 2008)