FPGA Polyphase Filter Bank Study & Implementation

Raghu Rao
Matthieu Tisserand
Mike Severa
Prof. John Villasenor

Image Communications/Reconfigurable Computing Lab.
Electrical Engineering Dept.
UCLA
Introduction

This document describes a polyphase filter bank and summarizes the results of a feasibility study of its implementation on FPGA-based architectures with respect to size, timing and bandwidth requirements.

Under the SLAACC program, UCLA and Los Alamos National Labs have collaborated in mapping new adaptive algorithms to configurable computing platforms.
The current portion of the collaboration has involved the feasibility and implementation of a Polyphase Filter bank using various FPGAs and hardware architectures.

The Polyphase implementation is a multi-rate filter structure combined with a DFT designed to extract subbands from an input signal.

It is an optimization of the standard approaches and offers increased efficiency in both size and speed, aspects that are well suited to reconfigurable computing.

Task heretofore implemented only in ASIC; offers a good opportunity as an example of migration from ASIC to an Adaptive platform.
Basic Project Parameters

- 128 Megasamples/sec input signal
- 16 distinct subband outputs
- Implement using Polyphase filter and DFT structure

![Diagram showing Poly-DFT process with 128 Mss input and multiple outputs]
Polyphase Filter Architecture

Input samples → Commutator → Polyphase filter bank → FFT → 16 positive frequency bins
Polyphase Filter Architecture

Commutator:
- distributes signal to \( n \) lines
- reduces clock speed by factor of \( n \)

Polyphase Filter bank:
- 32 1-input, 1-output polyphase filters or
- 16 1-input, 2-output optimized polyphase filters

FFT:
- \( 2n \)-point real FFT
- \( n \)-point complex FFT
System Requirements

- 128MHz system speed
- Note: 4 samples at 32MHz equivalent to one sample at 128 MHz
- All lines are buses equal to sample precision (from 8 to 16 bits)
- Precision has been implemented as a generic in VHDL
  - makes precision configurable
  - allows easy assessment of precision’s affect on feasibility
What Happens Inside?

- Data will be sent to 32 filters...
- i.e., need to be latched and further demuxed by factor of 8
- Clock speed reduced also by factor of 8 to 4MHz
And then?

- Some work gets done
  - Polyphase filtering, DFT: @ 4MHz
  - note: using resource-sharing filter structures, initial decimation only by factor of 4, smaller filter bank, work gets done @ 8MHz (slides on this method later)
And finally?

- 16 samples at 4MHz are available to the remuxing logic
- 16 samples are required for the system
- Re-Mux runs at 16MHz and samples 4 DFT outputs at a time
- Results data has latency of a minimum of 12 clock cycles due to demux/remux (plus polyphase/DFT latency)
The following slides describe the regular polyphase filter bank, the transpose form FIR filter, and optimizations based on symmetry.

This is a symmetric FIR filter, i.e., the first $n/2$ and the last $n/2$ coeffs are the same, albeit in reverse order.

We can exploit this symmetry to implement an optimal form of the filter bank, using resource sharing.

We also describe two methods of exploiting resource sharing. The advantage of these schemes is the reduction in the size of both the filter bank and the commutator.
First step is to design a prototype low-pass, FIR filter $h(n)$ with the desired filter parameters.

The $I$ polyphase filters $p_k$, each of integer length $K = M/I$ are derived from the length $M$ FIR filter $h(n)$ via

- $p_k(n) = h(k + nI)$, $k = 0..I-1$, $n = 0..K-1$
- ($M$ is selected to be a multiple of $I$)
Our Polyphase Design

- $K = M/I : M = 128, K = 4, I = 32$
- $p_k(n) = h(k + nI), k = 0..I-1, n = 0..K-1$
  - $p_0 = h(0 + 0), h(0 + 32), h(0 + 64), h(0 + 96)$
  - $p_1 = h(1 + 0), h(1 + 32), h(1 + 64), h(1 + 96)$
  - $p_{31} = h(31 + 0), h(31 + 32), h(31 + 64), h(31 + 96)$
Our Polyphase Design

<table>
<thead>
<tr>
<th>$p_0$</th>
<th>h(0), h(32), h(64), h(96)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p_1$</td>
<td>h(1), h(33), h(65), h(97)</td>
</tr>
<tr>
<td>$p_2$</td>
<td>h(2), h(34), h(66), h(98)</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>$p_{31}$</td>
<td>h(31), h(63), h(95), h(127)</td>
</tr>
</tbody>
</table>
Polyphase filter bank, 32 filters with 4 taps each

Decimate by 32
Symmetry - how is it useful?
Symmetric filter bank

A1  B1  C1  D1
A2  B2  C2  D2
A3  B3  C3  D3
A4  B4  C4  D4

24 more filters

D4  C4  B4  A4
D3  C3  B3  A3
D2  C2  B2  A2

D1  C1  B1  A1

24 more filters
Symmetry - how is it useful?

• Given an \( n \)-tap filter with coefficients \( h(0..n) \)

\[
h_0 \ h_1 \ h_2 \ h_3 \ h_4 \ h_5 \ h_6 \ h_7 \ h_8 \ h_9 \ h_{10} \ h_{11} \ h_{12} \ h_{13} \ h_{14} \ h_{15}
\]

• In a symmetric filter of \( n \) taps, coefficient \( h(i) = h(n-1-i) \), i.e., we can re-label the above filter coefficients as

\[
h_0 \ h_1 \ h_2 \ h_3 \ h_4 \ h_5 \ h_6 \ h_7 \ h_7 \ h_6 \ h_5 \ h_4 \ h_3 \ h_2 \ h_1 \ h_0
\]

• What does this mean for our polyphase structure?
Symmetry - how is it useful?

- What does this mean for our polyphase structure?

```
|   h0  h8  |   h0  h7  |
|   h1  h9  |   h1  h6  |
|   h2  h10 |   h2  h5  |
|   h3  h11 |   h3  h4  |
|   h4  h12 |   h4  h3  |
|   h5  h13 |   h5  h2  |
|   h6  h14 |   h6  h1  |
|   h7  h15 |   h7  h0  |
```
Symmetry - how is it useful?

- What does this mean for a polyphase structure?
- We can reduce number of coefficient multipliers

\[ \begin{array}{cc}
  h0 & h7 \\
  h1 & h6 \\
  h2 & h5 \\
  h3 & h4 \\
  h5 & h2 \\
  h6 & h1 \\
  h7 & h0 \\
\end{array} \]
Symmetry - how is it useful?

• What does this mean for a polyphase structure?

.. x15 .. x7 → h0 h7 → x0 .. X8 ..

.. x14 .. x6 → h1 h6 → x1 .. X9 ..

.. x13 .. x5 → h2 h5 → x2 .. X10..

.. x12 .. x4 → h3 h4 → x3 .. X11 ..
The commutator is half the size for this architecture. After feeding 8 filters, it reverses direction.

Samples 1 to 8

It goes to filters 1, 2, 3, 4, 5, … 8 and then 8, 7, 6, 5, 4, 3, 2, 1 and reverses direction again.
Symmetry - how is it useful?

Hardware Implementation
Transpose Form of the FIR filter

\[ x(n) \rightarrow h0 \rightarrow h1 \rightarrow h2 \rightarrow h3 \rightarrow y(n) \]

- register
- adder
- multiplier
Resource Sharing Optimization - Scheme 1

- Convolution of even samples
- Convolution of odd samples

Clocked for even samples
Clocked for odd samples
Resource Sharing Optimization - Scheme 2

- $x(n)$
- $h_0$, $h_1$, $h_2$, $h_3$
- Even sample convolution
- Odd sample convolution

Control:
- 1 - odd
- 0 - even

Clocked for even samples

Clocked for odd samples
## Comparison of Schemes

<table>
<thead>
<tr>
<th></th>
<th>Multipliers</th>
<th>Adders</th>
<th>Registers</th>
<th>Multiplexer</th>
<th>Demultiplexer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regular</td>
<td>128</td>
<td>96</td>
<td>128</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Scheme 1</td>
<td>64</td>
<td>96</td>
<td>128</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Scheme 2</td>
<td>64</td>
<td>48</td>
<td>256</td>
<td>112</td>
<td>16</td>
</tr>
</tbody>
</table>

**NOTE:** schemes 1 and 2, also reduce the size of the commutator. With these schemes only a N/2 commutator is needed (decimate by 16).
Since each filter convolves alternate samples, giving two outputs, one a convolution of even samples and the other a convolution of odd samples, it also acts to decimate by 2. So, the initial decimator needs to decimate only by 16.
The commutator is half the size for this architecture. After feeding 16 filters, it reverses direction.

Samples 1 to 16

It goes to filters 1, 2, 3, 4, 5, … 16 and then 16… 6, 5, 4, 3, 2, 1 and reverses direction again.
A clocking scheme to enable flipflops alternately

The flipflops in different colors need to be latch data alternately.

When blue is on, green is off. This can be accomplished by a 2 phase clocking scheme.
Instead of positive and negative DFFs, enable FFs can be used to convolve alternate samples. This clock enable also can be used as the select line to the muxes and demuxes.
Initial Studies

The initial work involved approaching the topic from a theoretical standpoint

- understand polyphase theory
- implement polyphase structure simulation
  - DSP Canvas
  - MatLab
- create filter based on design specs from Fiore’s paper
- generate initial size estimates based on knowledge of the size of components and number of CLB’s necessary to implement them on an FPGA
Feasibility Experiments

These experiments evaluated the feasibility of implementing the polyphase filter bank on an Altera Flex10K250A (part EPF10K250AGC599-1) a Xilinx XC40150 (part XC40150XV-09-BG560) and a Xilinx VirtexXCV1000 (part XCV1000-4-BG560)

All experiments were synthesized using Synplify 5.1.4 and placed and routed with Maxplus2 9.1

The filter bank consisted of a decimator at the input, feeding a bank of either 16 or 32, 4 tap filters (filters optimized for symmetry have 2 outputs). The outputs of the filter bank feed a commutator that “re-muxes” data onto 4 lines that will feed a DFT (assumption that the DFT is on another chip).
Results for non-symmetry optimized filter bank

Flex10K250A, part EPF10k250AGC599-1, does not fit.

The critical resource on an Altera Flex10K is the carry chain (fast interconnect) routing.

32 filters, with 1 output each, not optimized for symmetry
Results for non-symmetry optimized filter bank

**Xilinx Virtex, part XCV1000-4-BG560**

<table>
<thead>
<tr>
<th>AREA</th>
<th>MAX FREQ</th>
</tr>
</thead>
<tbody>
<tr>
<td>2130 of 12288 CLB slices, 17% of chip</td>
<td>66.578 MHz</td>
</tr>
<tr>
<td>3740 of 12288 CLB slices, 30% of chip</td>
<td>64.583 MHz</td>
</tr>
<tr>
<td>4030 of 12288 CLB slices, 33% of chip</td>
<td>72.343 MHz</td>
</tr>
</tbody>
</table>

This has 32 filters, with 1 output each, not optimized for symmetry

- **D** - data precision
- **C** - coeff precision
Results for symmetry optimized filter bank

Flex10K250A, part EPF10k250AGC599-1

<table>
<thead>
<tr>
<th>AREA</th>
<th>MAX FREQ</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>D=8, C=13</strong></td>
<td>4932 LEs</td>
</tr>
<tr>
<td></td>
<td>22.47 MHz</td>
</tr>
<tr>
<td>40% of chip</td>
<td></td>
</tr>
<tr>
<td><strong>D=13, C=13</strong></td>
<td>8109 LEs</td>
</tr>
<tr>
<td></td>
<td>22.42 MHz</td>
</tr>
<tr>
<td>66% of chip</td>
<td></td>
</tr>
<tr>
<td><strong>D=14, C=13</strong></td>
<td>8523 LEs</td>
</tr>
<tr>
<td></td>
<td>22.02 MHz</td>
</tr>
<tr>
<td>70% of chip</td>
<td></td>
</tr>
<tr>
<td><strong>D=15, C=13</strong></td>
<td>8975 LEs.</td>
</tr>
<tr>
<td></td>
<td>No Fit.</td>
</tr>
<tr>
<td>76% of chip. No Fit, routing resources</td>
<td></td>
</tr>
</tbody>
</table>

This has 16 filters, with 2 outputs each, optimized for symmetry
Results for symmetry optimized filter bank

**Xilinx XC40150XV-09-BG560**

<table>
<thead>
<tr>
<th>AREA</th>
<th>MAX FREQ</th>
</tr>
</thead>
<tbody>
<tr>
<td>( D=8, \ C=13 )</td>
<td>2946 of 5184 CLB slices, 56% of chip</td>
</tr>
</tbody>
</table>

This has 16 filters, with 2 outputs each, optimized for symmetry
Xilinx Virtex XCV1000-4-BG560

<table>
<thead>
<tr>
<th>AREA</th>
<th>MAX FREQ</th>
</tr>
</thead>
<tbody>
<tr>
<td>(D=13, C=13)</td>
<td>3478 of 12288 CLB</td>
</tr>
<tr>
<td></td>
<td>slices, 29% of chip</td>
</tr>
</tbody>
</table>

This has 16 filters, with 2 outputs each, optimized for symmetry.
The following slides describe some optimizations of the FFT and how its inclusion into the system logic affects size and speed.

- Goal of system is 16 distinct positive frequency bins
- An N-point FFT produces N/2+1 distinct bins
- Our input sequence is real
- The FFT of a real valued sequence of 2N points can be computed efficiently by employing an N-point complex FFT
X(n), the 2N point real sequence is divided into 2, N-point sequences as follows:

\[ h(k) = x(2k), \quad k = 0, 1, \ldots, N - 1 \]

\[ g(k) = x(2k + 1), \quad k = 0, 1, \ldots, N - 1 \]

i.e.. The function \( h(k) \) is equal to the even-numbered samples of \( x(k) \), and \( g(k) \) is equal to the odd-numbered samples.

A N-point complex valued sequence \( y(k) \) can be written as

\[ y(k) = h(k) + j \, g(k) \]

The DFT of \( y(k) \) is then computed.
FFT cont’d.

\[ Y(k) = H(k) + W^k_{2N} G(k), \quad k = 0, 1, \ldots, N-1 \]

\[ Y(k + n) = H(k) - W^k_{2N} G(k), \quad k = 0, 1, \ldots, N-1 \]

To compute the real and imag. parts of the output, \( H(k) \) and \( G(k) \) can be expressed in terms of even and odd components.

\[ H(k) = R_e(k) + j I_o(k) \]
\[ G(k) = I_e(k) - j R_o(k) \]

Substituting this in \( Y(k) \), we get,

\[ Y(k) = Y_r(k) + j Y_i(k), \text{ where} \]
FFT of a $2N$ point real sequence from a $N$ point complex FFT

- Even samples
- Odd samples
- 32 point real sequence
- 16 point complex FFT
- Real
- Imag
- $G(k)$
- $G(k + N)$
- $W_{2N}^k$

Reconfigurable Computing Lab
Area and delay numbers for the 32-point real FFT

<table>
<thead>
<tr>
<th>Data precision</th>
<th>Timing</th>
<th>Area</th>
<th>Percentage of chip</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>16.03 MHz</td>
<td>2165</td>
<td>17% of chip</td>
</tr>
<tr>
<td></td>
<td>62.3 ns</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>16.47 MHz</td>
<td>2413</td>
<td>19% of chip</td>
</tr>
<tr>
<td></td>
<td>60.7 ns</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>8.89 MHz</td>
<td>2654</td>
<td>21% of chip</td>
</tr>
<tr>
<td></td>
<td>112.4 ns</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>15.31 MHz</td>
<td>2876</td>
<td>23% of chip</td>
</tr>
<tr>
<td></td>
<td>65.3 ns</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Altera Flex10K-250A GC599-1

Xilinx xc40150-09-bg560: Area 2530 out of 5184 (48% of chip), 20.001 MHz.

Xilinx Virtex xcv1000-4-bg560: Area 1754 out of 12288 (14% of chip), 48.96 MHz.

(virtex precision 13 & 13, XC40150: 8 & 13)
The entire polyphase filter bank along with the FFT does not fit on an Altera Flex device. But it does fit on the Xilinx XC40150 and Virtex.

Decimation factor = 32, 17 positive frequency bins
Data precision = 13, Coeff precision = 13

**Xilinx xc40150-09-bg560 (D=8, C=13)**
4581 CLBs out of 5184 - 88% of chip.
Freq: 20.492 MHz

**Xilinx Virtex - xcv1000-4-bg560**
7156 CLB slices out of 12288 - 58% of chip.(11631 LUTs).
Freq: 56.715 MHz.
### Design Comparison

<table>
<thead>
<tr>
<th>DESIGN</th>
<th>XILINX XC40150XV-09-BG560</th>
<th>XILINX VIRTEX XCV1000-4-BG560</th>
<th>ALTERA FLEX10K250AGC599-1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>AREA</td>
<td>TIMING</td>
<td>AREA</td>
</tr>
<tr>
<td><strong>Fir filterbank</strong></td>
<td>8 point data precision</td>
<td>8 point data precision</td>
<td>8 point data precision</td>
</tr>
<tr>
<td></td>
<td>13 point coeff precision</td>
<td>13 point coeff precision</td>
<td>13 point coeff precision</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>2946/5184</strong></td>
<td>Freq 31.448 MHz</td>
<td></td>
<td>4932/12160</td>
</tr>
<tr>
<td><strong>CLBs 56% of chip</strong></td>
<td></td>
<td></td>
<td><strong>LEs 40% of chip</strong></td>
</tr>
<tr>
<td><strong>32 point real FFT</strong></td>
<td>2530/5184</td>
<td>Freq 20.001 MHz</td>
<td>2092/12160</td>
</tr>
<tr>
<td></td>
<td>48% of chip</td>
<td></td>
<td><strong>LEs 17% of chip</strong></td>
</tr>
<tr>
<td><strong>Full design (polyphase filter bank+fft)</strong></td>
<td>4581/5184</td>
<td>Freq 20.492 MHz</td>
<td>4195/12288</td>
</tr>
<tr>
<td></td>
<td>88% of chip</td>
<td></td>
<td><strong>34% of chip</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>4581/5184</td>
<td></td>
<td>4195/12288</td>
</tr>
<tr>
<td></td>
<td>88% of chip</td>
<td></td>
<td>34% of chip</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>50.594 MHz</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Area** and **Timing** for each design are provided.
- **Freq** values are given in MHz.
- **CLBs** and **LEs** indicate the percentage of the chip utilized.
Verify VHDL by checking the RTL level schematics, checking the number of adders, multipliers and registers.
RTL level schematics and design browser from Synplify
Future Work

Simulate and test polyphase VHDL implementation using LANL test vectors

Work together with LANL to facilitate possible demo of polyphase work

Implement Scheme 2 of resource sharing symmetrical filter bank

Study the advantages and disadvantages with regards to system goals of FFT replacing the FFT with a DCT

Look into adaptive filtering techniques

Modifying our current polyphase design to accommodate configurable or even programmable rate
Conclusions

Very productive initial phase of collaboration between UCLA and LANL

Our work has resulted in some innovations at the algorithmic level

Task migration from ASIC to FPGA

This study has provided useful sizing information for the Altera Flex and Xilinx Virtex families as well as some initial benchmarks of basic DSP methods used in UWB