Channelization using RFNoC

GRCON 2017

Phil Vallance

Critical Mission Engineering
Company Info

- Engineering services in the areas of embedded hardware, firmware, software development and test.
- Specialties include: RF, DSP, FPGA, C/C++, Python, Linux.
- From initial operational concept, through engineering design & development, to field deployment.
# Table of contents

1. Introduction

2. Channelizer Background

3. Hardware Implementation

4. Filter Generation

5. GNURadio Software Component / Results

6. Future Work
Introduction
Introduction: Goals

- Review the derivation of the $M/2$ channelizer structure found in *A Versatile Multichannel Filter Bank with Multiple Channel Bandwidths* [Harris F.(2010)].
- Provide detailed FPGA implementation optimized for both the Xilinx FPGA architecture and the RFNoC framework.
- Detail implications using block-floating point FFT core.
- Describe current filter tap generation algorithm based on [Lubberhuizen(2010)].
- Discuss limitations and future work.
Channelizer Background
Channelizer Background: Motivation

- Relaxes the filter transition bandwidth requirements.
- Allows perfect reconstruction of the channels.
- Efficient implementation.
Channelizer Background: Identities

Noble Identities

\[ M^{-1} \sum_{r=0}^{M-1} y_r(n) e^{j2\pi Mr} = x(n) \]

Time Expansion

\[ x_k[n] = \begin{cases} x[r], & n = Mr \\ 0, & n \notin MN \end{cases} \Rightarrow X(Z^M) \]
Channelizer Background: Channel Selector

\[ y(n, k) = [x(n)e^{-j\theta_k n}] * h(n) = \sum_{r=0}^{N-1} x(n - r)e^{-j\theta_k n}h(r) \]
Channelizer Background: Channel Selector

\[ \begin{align*}
x(n) & \xrightarrow{\text{M:1}} x(n)e^{-j\theta_k n} \\
& \xrightarrow{H(Z)} y(n, k) \\
y(n, k) & = \sum_{r=0}^{N-1} x(n - r)e^{-j\theta_k n}h(r)
\end{align*} \]

\[ \begin{align*}
x(n) & \xrightarrow{\text{M:1}} x(n)e^{-j\theta_k n} \\
& \xrightarrow{H(Ze^{j\theta_k})} y(n, k) \\
y(n, k) & = \sum_{r=0}^{N-1} x(n - r)e^{-j\theta_k(n-r)}h(r) = e^{-j\theta_k n}\sum_{r=0}^{N-1} x(n - r)h(r)e^{j\theta_k r}
\end{align*} \]
Channelizer Background: Channel Selector

\[ y(nM, k) = \sum_{r=0}^{M} y_r(nM) e^{j2\pi Mk} \]

\[ e^{jM\theta_k n} \]

\( y(n, k) \) is the channel selector index

\( M:1 \)
Channelizer Background : Channel Selector

\[ y(nM, k) = H(Ze^{j\theta_k}) y(n, k) \]

Restrict \( \theta_k \) to a multiple of \( \frac{2\pi}{M} \)

\[ e^{-jM\theta_k n} \]
Channelizer Background : Filter Transformation

\[ H(Z) = \sum_{n=0}^{N-1} h(n)Z^{-n} = \sum_{r=0}^{M-1} Z^{-r} H_r(Z^M) \]

\[ = \sum_{r=0}^{M-1} Z^{-r} \sum_{n=0}^{(N/M)-1} h(r + NM) Z^{-nM} \]

\[ H(z) = h(0) + h(M + 0)Z^{-M} + h(2M + 0)Z^{-2M} + \ldots \]
\[ h(1)Z^{-1} + h(M + 1)Z^{-(M+1)} + h(2M + 1)Z^{-(2M+1)} + \ldots \]
\[ h(2)Z^{-2} + h(M + 2)Z^{-(M+2)} + h(2M + 2)Z^{-(2M+2)} + \ldots \]
\[ h(3)Z^{-3} + h(M + 3)Z^{-(M+3)} + h(2M + 3)Z^{-(2M+3)} + \ldots \]
\[ \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \]
\[ h(M - 1)Z^{-(M-1)} + h(2M - 1)Z^{-(2M-1)} + h(3M - 1)Z^{-(3M-1)} + \ldots \]
Channelizer Background : Filter Transformation

\[ H(z) = h(0) + h(M + 0)Z^M + h(2M + 0)Z^{2M} + h(1)Z^1 + h(M + 1)Z^{M+1} + h(2M + 1)Z^{2M+1} + \cdots \]

\[ x(n) \]

\[ y(n) \]

\[ H_0(Z^M) \]

\[ H_1(Z^M) \]

\[ H_{M-2}(Z^M) \]

\[ H_{M-1}(Z^M) \]

\[ Z^{-1} \]

\[ Z^{-(M-2)} \]

\[ Z^{-(M-1)} \]

\[ + \]

\[ \cdot \cdot \cdot \]

\[ M:1 \]

Substitute \( G \)
Channelizer Background: Filter Transformation

\[ H(z) = h(0) + h(M+0)z^{-M} + h(2M+0)z^{-2M} + h(1)z^{-1} + h(M+1)z^{-(M+1)} + h(2M+1)z^{-(2M+1)} \]

With the advantage of periodicity of complex exponential \( e^{jk\frac{2\pi}{M}} \),

\[ x(n) \]

\[ y(n) \]
Channelizer Background: Filter Transformation

\[ H(z) = h_0 + h(M+0)z^M + h(2M+0)z^{2M} + \cdots \]

\[ x(n) \]

\[ y(n) \]

\[ x(n) \]

\[ y(n) \]

\[ x(n) \]

\[ y(n) \]

\[ x(n) \]

\[ y(n) \]
Channelizer Background: $M$-Filter Transformation

Substitute $Z^{-1}$ with $Z^{-1}e^{j\theta}$ and $Z^{-M}$ with $Z^{-M}e^{jM(2\pi/M)}$

$$G(Z) = \sum_{n=0}^{N-1} h(n)[e^{-j\theta}Z]^{-n}$$

$$= h(0) + h(1)e^{j\theta}Z^{-1} + h(2)e^{2j\theta}Z^{-2} + \cdots + h(N-1)e^{j(N-1)\theta}Z^{-(N-1)}$$

$$G(z) = \begin{cases} 
  h(0) \\
  h(1)Z^{-1}e^{j(2\pi/M)} \\
  h(2)Z^{-2}e^{j(2\pi/M)} \\
  \vdots \\
  h(M-1)Z^{-(M-1)}e^{j(M-1)(2\pi/M)} \\
  h(M)Z^{-M}e^{jM(2\pi/M)} \\
  h(M+1)Z^{-(M+1)}e^{j(M+1)(2\pi/M)} \\
  \vdots \\
  h(2M-1)Z^{-(2M-1)}e^{j(2M-1)(2\pi/M)} \\
  h(2M)Z^{-2M}e^{j2M(2\pi/M)} \\
  \vdots \\
  h(3M-1)Z^{-(3M-1)}e^{j(3M-1)(2\pi/M)} \\
  \vdots \\
  \vdots \\
  \vdots \\
  \vdots 
\end{cases}$$
Channelizer Background

Standard Decimate by M Channelizer

\( y(n, k) = [x(n)e^{-j\theta_k n}] \ast h(n) = \sum_{r=0}^{N-1} x(n - r)e^{-j\theta_k n}h(r) \)
Channelizer Background: $M/2$ Filter Transformation

The diagram represents the channelization process for an $M/2$ channelizer. The input signal $x(n)$ is processed through a series of filters $H_i(Z^M)$, where $i = 0, 1, \ldots, M/2-1$, and $H_{M-1}(Z^M)$. Each filter is applied to the input signal after a delay of $Z^{-i}$, resulting in outputs $y(n)$.

- $H_0(Z^M)$
- $H_1(Z^M)$
- $\ldots$
- $H_{M/2-1}(Z^M)$
- $H_{M-1}(Z^M)$

The final output is obtained by summing these filtered signals:

$$y(n) = H_0(Z^M) + H_1(Z^M) + \ldots + H_{M-1}(Z^M)$$

For an $M/2$ channelizer, the correction is a circular shift of the filter output data. Samples are delivered are first vacated by their former contents. There is a serpentine progression up the stack to Port 0. The $M/2$ addresses to which the new $M/2$ inputs are directed are M/2:1.
Channelizer Background: \( M/2 \) Filter Transformation
Channelizer Background: $M/2$ Filter Transformation

$\text{Serpentine Shift}$

$\text{Polyphase Filter}$

$\text{Circular Buffer}$

$\text{M-Point IFFT}$

$\text{Input Data Buffer}$

$\text{Polyphase Filter Bank}$

$y(n) = h(n + 3M/2) + h(n + 5M/2) + h(n + 7M/2)$

Must compensate for this phase shift by applying the appropriate phase correction to the spectral estimates directly into the polyphase filter bank.

Channelizer outputs samples from each frequency bin at a rate of $kM/2:1$.
Channelizer Background: \( M/2 \) Filter Transformation

\[
H(Z) = \sum_{r=0}^{M/2-1} Z^{-r} H_r(Z^M) + Z^{-(r+M/2)} H_{r+M/2}(Z^M)
\]

where

\[
H_r(Z^M) = \sum_{n=0}^{(N/M)-1} h(r + nM)Z^{-nM}
\]

- Delay in Polyphase filter bank effectively implements a circular shift of \( M/2 \)
- This shifting of origin will need to be compensated before PFB outputs are sent to IFFT.
Channelizer Background: Origin Compensation

Channelization using RFNoC

---

M-Point FFT Output Buffer Polyphase Data Buffers Cyclically Shifted Buffer

$H(0) \rightarrow n+4 \leftrightarrow n-4 \leftrightarrow n-12 \leftrightarrow n-20$

$H(M-2) \rightarrow n+3 \leftrightarrow n-5 \leftrightarrow n-13 \leftrightarrow n-21$

$H(M-1) \rightarrow n+2 \leftrightarrow n-6 \leftrightarrow n-14 \leftrightarrow n-22$

$H(M) \rightarrow n+1 \leftrightarrow n-7 \leftrightarrow n-15 \leftrightarrow n-23$

$n \leftrightarrow n-8 \leftrightarrow n-16 \leftrightarrow n-24$

$r(nM, 0) \rightarrow r(nM, M-2) \rightarrow r(nM, M-1)$

---

GRCON 2017
Channelizer Background: Origin Compensation

- Need to implement *Circular Shift*
Channelizer Background: System Diagram

\[ x(n) \rightarrow \text{Input Data Buffer} \rightarrow \text{Polyphase Filter} \rightarrow \text{Circular Buffer} \rightarrow \text{M-Point IFFT} \rightarrow y(n) \]
Hardware Implementation
Hardware Implementation: Input Buffer

- Utilizes AXI flow to conform to the RFNoC interface.
- Provides a ping-pong buffer interface for continuous streaming.
- Each dual-port RAM is read twice. (Produces the time domain sequence required by the PFB)
Hardware Implementation: Polyphase Filter Bank

\[ H_0(Z^2) \]

\[ x(n) \rightarrow Z^{-2} \rightarrow h(k) \rightarrow h(k + M) \rightarrow Z^{-2} \rightarrow Z^{-2} \rightarrow y(n) \]

\[ Z^{-1}H_{M/2+1}(Z^2) \]

\[ x(n) \rightarrow Z^{-1} \rightarrow Z^{-2} \rightarrow h(k + \frac{M}{2}) \rightarrow h(k + \frac{3M}{2}) \rightarrow Z^{-2} \rightarrow Z^{-2} \rightarrow Z^{-2} \rightarrow y(n) \]
• Take advantage of Systolic Architecture of PFB.
• Fully pipeline DSP internals to maximize FMax.
• Utilize dedicated routing for optimal placement.
• Utilizes built in routing of Xilinx architecture
• Samples can be streamed continuously.
• Achieves high FMax value > 500 MHz.
- Utilizes ping-pong buffering to allow streaming.
- Offset counter logic offsets the read pointer by $M/2$ samples every other block of $M$ samples.
- Using block floating point IFFT (common exponent for block of samples).
- Averages the exponent of 64 consecutive IFFT frames.
- Current block is shifted by the integer difference between the current exponent and the moving average.
- Enforces headroom in output to allow for loud, bursty transmissions.
Filter Generation
Filter Generation

- Based on the work of Wessel Lubberhuizen found in [Lubberhuizen(2010)].
- Using root raised erf functions to perform filter coefficient calculation.
- Stable even for very narrow filter bandwidths.
- Simple Calculation.
- Not as flexible as Remez.
- Provides flat passband and low stopband with linear phase.
GNURadio Software Component / Results
- This setup is using a 256 channel channelizer.
- Four sub-bands specified in the mask vector argument to `poly_channelizer` block.
- The `tap_update` python block configures the taps at start-up and when number of channels is adjusted.
- `chanmux` is the block controller for the RFNoC block.
- The `tap_update` is being converted to C++ and integrated in `chanmux`. 
GNURadio Software Component / Results

Channelization using RFNoC
Future Work
Future Work

• Word sizes larger than 16 bit I/Q samples, current head space is diminishing DR.
• Narrow-band signals would also benefit.
• Second task is to convert the python code found in the tap_update module in to C++.
Questions?
Harris F., McGwier R., Egg B.

A versatile multichannel filter bank with multiple channel bandwidths.


Lubberhuizen, Wessel.

Near perfect reconstruction polyphase filterbank.