Building an OFDM Receiver with RFNoC™
Jonathon Pendulum, Martin Braun
GNU Radio Conference 2015
Outline

- Our Goal
  - Build a receiver for WiFi packets
  - ... in RFNoC
  - Find out where RFNoC is falling short
- OFDM Primer
- Implementation
- Summary
- Demo!
Goal: Receive 802.11a packets

- “Develop a useful app in RFNoC”
  - 802.11a is a great candidate
- PHY Layer: 20 MHz wide channels, OFDM signals
- MAC Layer: Low latency processing (DCF, RTS/CTS)
- Various host-side software solutions exist -- gr-ieee-80211
- Implementation should be:
  - Generic and moderately granular
  - Well integrated into GNU Radio
OFDM Primer

- Orthogonal Frequency Domain Multiplexing
  - Multicarrier, wideband modulation
  - Robust
  - LTE, DVB-T, 802.11
OFDM Primer

- OFDM Signal Synthesis:

**QAM Mapping (Header + Payload)**

- Virtual Source
  - Stream ID: Header Bits
- Virtual Source
  - Stream ID: Payload Bits
- Chunks to Symbols
  - Symbol Tables: -1, 1
  - Dimensions: 1
- Chunks to Symbols
  - Symbol Tables: -707...07.107m
  - Dimensions: 1
- Tagged Stream Mux
  - Length tag names: packet_len
- Virtual Sink
  - Stream ID: Pre-OFDM

**Carrier Allocation**

- Virtual Source
  - Stream ID: Pre-OFDM
- OFDM Carrier Allocator
  - FFT length: 64
  - Occupied Carriers: [-... 26]
  - Pilot Carriers: [-2... -7, 21]
  - Pilot Symbols: (1, 1, 1, -1)
  - Sync Words: [0, 0, ..., 0, 0, 0]
  - Length tag key: packet_len

**Modulation**

- FFT
  - FFT Size: 64
  - Forward/Reverse: Reverse
  - Window: Shift: Yes
  - Num. Threads: 1

**Pulse Shaping**

- OFDM Cyclic Prefixer
  - FFT Length: 64
  - CP Length: 16
  - Length Tag Key: packet_len
- Virtual Sink
  - Stream ID: Time Domain
Structure of 802.11a PHY

- 20 MHz wide
- 64 subcarriers
  - FFT size 64
  - 48 data, 4 pilots, 12 guard band
- Simple package structure: Preamble, Header, Data
  - Short & long preamble for timing, frequency, and phase synch
- From the standard:
Structure of 802.11a PHY

- **GI2**: Guard Interval 2
- **T1**: Transmission Interval 1
- **T2**: Transmission Interval 2
- **GI1**: Guard Interval 1
- **SIGNAL**: Signal Data
- **GI1**: Guard Interval 1 for Data 1
- **GI1**: Guard Interval 1 for Data 2
- **SERVICE + DATA**: Service Data
- **DATA**: Data Field

- **Signal Detect, AGC, Diversity Selection**
- **Coarse Freq. Offset Estimation**
- **Channel and Fine Frequency Offset Estimation**
- **Rate Length**
- **Timing Synchronize**

- **8 + 8 = 16 µs**
- **10 × 0.8 = 8 µs**
- **2 × 0.8 + 2 × 3.2 = 8.0 µs**
- **0.8 + 3.2 = 4.0 µs**
- **0.8 + 3.2 = 4.0 µs**
- **0.8 + 3.2 = 4.0 µs**
OFDM Receiver in GNU Radio

UHD: USRP Source
Samp Rate (Sps): 10k
Ch0: Center Freq (Hz): 0
Ch0: Gain Value: 0

Schmidt & Cox OFDM synch.
FFT length: 64
Cyclic Prefix length: 16
freq_offset
detect

Frequency Mod
Sensitivity: -31.25m

Delay
Delay: 80

Multiply

Stream to Vector
Num Items: 64

FFT
FFT Size: 64
Forward/Reverse: Forward
Window:
Shift: Yes
Num. Threads: 1

OFDM Channel Estimation
Synch. symbol 1: sync_word1
Synch. symbol 2: sync_word2
Number of data symbols: 1
Maximum carrier offset: 3

OFDM Frame Equalizer
FFT length: 64
CP length: 16
Equalizer: <gnura...25a030> >
Length Tag Key: frame_len
Propagate Channel State: Yes

OFDM Serializer
FFT length: 64
Occupied Carriers: [-..., 26]
Length Tag Key: frame_len
Input is shifted: True

Constellation Decoder
Constellation Object: ...0>

Repack Bits
Bits per input byte: 2
Bits per output byte: 8

File Sink
File: data.bin
Unbuffered: Off
Append file: Overwrite
- Schmidl & Cox Synchronization
- Timing estimate & frequency correction
- Utilizes auto-correlation of short preamble
OFDM Receiver in GNU Radio

- Time domain to frequency domain conversion
- Demodulate individual subcarriers
OFDM Receiver in GNU Radio

- Equalization via long preamble
- Phase & further frequency correction
OFDM Receiver in GNU Radio

- Remove pilot tones, guard bands, DC bin
- Decode symbols to bits
OFDM Receiver in GNU Radio

- **UHD: USRP Source**
  - Samp Rate (Sps): 10k
  - Ch0: Center Freq (Hz): 0
  - Ch0: Gain Value: 0

- **Schmidl & Cox OFDM synch.**
  - FFT length: 64
  - Cyclic Prefix length: 16
  - freq_offset
  - detect

- **Frequency Mod Sensitivity:** -31.25m

- **Delay**
  - Delay: 80

- **Multiply**
  - out

- **Stream to Vector**
  - Num Items: 64

---

- **FFT**
  - FFT Size: 64
  - Forward/Reverse: Forward
  - Window:
  - Shift: Yes
  - Num. Threads: 1

- **OFDM Channel Estimation**
  - Synchron. symbol 1: sync_word1
  - Number of data symbols: 1
  - Maximum carrier offset: 3

- **OFDM Frame Equalizer**
  - FFT length: 64
  - CP length: 16
  - Equalizer: <gnura...25a030> >

---

- **OFDM Serializer**
  - FFT length: 64
  - Occupied Carriers: [...., 26]
  - Length Tag Key: frame_len

- **Constellation Decoder**
  - Constellation Object: ...0 >

---

- **Repack Bits**
  - Bits per input byte: 2
  - Bits per output byte: 8

---

- **File Sink**
  - File: data.bin
  - Unbuffered: Off
  - Append file: Overwrite
OFDM Receiver in RFNoC

RFNoC: Radio
- Radio Select: B
- Mode: Rx
- Stream Args:
  - Center Frequency: 2.4G
  - Sampling Rate: 20M
  - Gain: 10
  - Antenna: TX/RX
  - Force Vector Length: 64

RFNoC: FIFO
- Device Format: Complex int16
- FIFO Select: 0
- Force Vector Length: 64

RFNoC: OFDM Sync
- FFT Size: 64
- Cyclic Prefix Length: 16
- Threshold: 850m
- delay: 248
- Max Num Symbols: 8

RFNoC: FFT
- FFT Size: 64
- Forward/Reverse: Forward
- Shift: Yes
- FFT Output: Complex
- FFT Scaling Word: 1.365k

RFNoC: OFDM Equalizer
- FFT Size: 64

RFNoC: OFDM Constellation Demap
- FFT Size: 64
- Modulation: QPSK
- Scaling: QPSK

RFNoC: FIFO
- Device Format: Byte
- FIFO Select: 1
- Force Vector Length: 64

File Sink
- File: data.bin
- Vec Length: 64
- Unbuffered: Off
- Append file: Overwrite
OFDM Receiver in RFNoC

**RFNoC: Radio**
- Radio Select: 8
- Mode: Rx
- Stream Args:
  - Center Frequency: 2.4G
  - Sampling Rate: 20M
  - Gain: 10
  - Antenna: TX/RX
  - Force Vector Length: 64

**RFNoC: FFT**
- FFT Size: 64
- Forward/Reverse: Forward
- Shift: Yes
- FFT Output: Complex
- FFT Scaling Word: 1.365k

**RFNoC: FIFO**
- Device Format: Byte
- FIFO Select: 1
- Force Vector Length: 64

**File:** data.bin
- Vec Length: 64
- Unbuffered: Off
- Append file: Overwrite

**RFNoC: OFDM Constellation Demap**
- FFT Size: 64
- Modulation: QPSK
- Scaling: QPSK
OFDM Receiver in RFNoC

**RFNoC: Radio**
- Radio Select: 8
- Mode: Rx
- Stream Args:
  - Center Frequency: 2.4G
  - Sampling Rate: 20M
  - Gain: 10
  - Antenna: TX/RX
  - Force Vector Length: 64

**RFNoC: FIFO**
- Device Format: Complex int16
- FIFO Select: 0
- Force Vector Length: 64

**RFNoC: OFDM Sync**
- FFT Size: 64
- Cyclic Prefix Length: 16
- Threshold: 850m
- delay: 248
- Max Num Symbols: 8

**Diagram: OFDM Receiver**
- Split
- Delay 80
- X
- Moving Sum
- Round + Clip
- CORDIC
- Phase
- Num
- Samples Out
- Phase inc
- Phase Acc
- Freq. Correction
- Periodic Framer
- Append matrix correction
- Installation Demap
- FFT Output
- FFT Scale
- FFT Size
- Forward
- Shift: Yes
- FFT Output
- Randomization
- Split
- [x]²
- Moving Sum
- Round + Clip
- Denom
- Num
- Denom
- Divider
- Round + Clip
- Peak Detect + AGC
- Samples Out
- Phase inc
- Phase
OFDM Receiver in RFNoC

**RFNoC: Radio**
- Radio Select: 8
- Mode: Rx
- Stream Args:
  - Center Frequency: 2.4G
  - Sampling Rate: 20M
  - Gain: 10
  - Antenna: TX/RX
  - Force Vector Length: 64

**RFNoC: FIFO**
- Device Format: Complex int16

**RFNoC: OFDM Sync**
- FFT Size: 64
- Cyclic Prefix Length: 16
- Threshold: 850m
- Delay: 248
- Max Num Symbols: 8

**OFDM Constellation Demap**
- Size: 64
- Constellation: QPSK
- QPSK

**RFNoC: FFT**
- FFT Size: 64
- Forward/Reverse: Forward
- Shift: Yes
- FFT Output: Complex
- FFT Scaling Word: 1.365k

**File Sink**
- File: data.bin
- Vec Length: 64
- Unbuffered: Off
- Append file: Overwrite
OFDM Receiver in RFNoC

- Zero forcing / One tap equalizer
- \[ Y(F) = H(F) \times X(F), \quad X(F) = \text{Long Preamble} = 0, -1, +1 \]
- \[ H(F) = (0, -1, +1)/Y(F) \]
- FIFO, Complex Invert, & 2 x Multipliers
OFDM Receiver in RFNoC

RFNoC: Radio
- Radio Select: 8
- Mode: Rx
- Stream Args:
  - Center Frequency: 2.4G
  - Sampling Rate: 20M
  - Gain: 10
  - Antenna: TX/RX
  - Force Vector Length: 64

RFNoC: FFT
- FFT Size: 64
- Forward/Reverse: Forward
- Shift: Yes
- FFT Output: Complex
- FFT Scaling Word: 1.365k

RFNoC: OFDM Equalizer
- FFT Size: 64

RFNoC: OFDM Sync
- FFT Size: 64
- Cyclic Prefix Length: 16
- Threshold: 850m
- delay: 248
- Max Num Symbols: 8

RFNoC: OFDM Constellation Demap
- FFT Size: 64
- Modulation: QPSK
- Scaling: QPSK

RFNoC: FIFO
- Device Format: Byte
- FIFO Select: 1
- Force Vector Length: 64

File Sink
- File: data.bin
- Vec Length: 64
- Unbuffered: Off
- Append file: Overwrite
- Remove Guard bands, DC bin, & Pilot tones
- Scaling
- Symbol to (gray coded) Bits
Summary

- Built an OFDM Receiver using RFNoC
- Developing DSP blocks always brings fun challenges
  - Having RFNoC + a standardized testbench infrastructure available makes this much easier
- Integrated with GNU Radio + GRC
- Similar workflow to developing GNU Radio applications
- All code available as part of our RFNoC repository
The Joy of Digital Signal Processing
Detection - Block Diagram

- OFDM Detection
Block Diagram

- OFDM Detection

- Split
  - Delay 80
  - $|x|^2$
  - $|x|$

- Moving Sum
  - Moving Sum
  - Moving Sum

- Round + Clip
  - Round + Clip
  - Round + Clip

- CORDIC
  - Phase
  - Num

- Peak Detect + AGC
  - Samples Out
  - Phase inc

- Phase inc
  - Phase Acc
  - Freq. Correction
  - Periodic Framer
  - Samples Out
## Does this look familiar?

<table>
<thead>
<tr>
<th>Feature</th>
<th>RFNoC</th>
<th>GNU Radio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graphical Debugging Capabilities</td>
<td>Modelsim for Simulations, GNU Radio widgets when streaming</td>
<td>Graphical widgets</td>
</tr>
<tr>
<td>Languages</td>
<td>Verilog, VHDL, (+ others)</td>
<td>C++, Python</td>
</tr>
<tr>
<td>Testing Infrastructure</td>
<td>Through Modelsim</td>
<td>Boost + Python unit test infrastructure (make test)</td>
</tr>
<tr>
<td>Works with GNU Radio Companion</td>
<td>Yes, through gr-ettus</td>
<td>Yes, obviously</td>
</tr>
<tr>
<td>How much you have to care about the framework while you’re writing your DSP code</td>
<td>Very little</td>
<td>Very little</td>
</tr>
<tr>
<td>Using this framework will avoid all pitfalls and requires no DSP knowledge</td>
<td>lol</td>
<td>rofl</td>
</tr>
</tbody>
</table>
Testbench Infrastructure

- In GNU Radio, you’d never install a module before your exhaustive suite of unit tests passes (right?)
- For RFNoC development, the same is true
  - Unless you feel inclined to invest a few hours per build, and then use ChipScope
- Testbench infrastructure provides:
  - Simulated streaming environment
  - Macros to generate packets and connect RFNoC blocks
- SystemVerilog based
- Requires Modelsim (not free)
  - Vivado sim support soon (freeish)
Graphical Debugging

- Imagine debugging GNU Radio applications without using graphical widgets (FFT Plot, Scope Plot, etc.)
- GUI Widgets can be used in RFNoC when streaming through devices