RFNoC™: RF Network on Chip
Martin Braun, Jonathon Pendulum
GNU Radio Conference 2015
Outline

- Motivation
  - Current situation
  - Goal
- RFNoC
  - Basic concepts
  - Architecture overview
- Summary

- No Demo!
  - See our booth, and our presentation tomorrow
RFNoC is for FPGAs is what GNU Radio is for GPPs.

<table>
<thead>
<tr>
<th>Feature</th>
<th>RFNoC</th>
<th>GNU Radio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Provides Easy-to-use Infrastructure for SDR applications</td>
<td>✅</td>
<td>✅</td>
</tr>
<tr>
<td>Handles Data Movement between blocks</td>
<td>✅ (AXI-Based)</td>
<td>✅ (Circular Buffers)</td>
</tr>
<tr>
<td>Takes care of boring and recurring tasks</td>
<td>✅ (Flow control, addressing, routing)</td>
<td>✅ (R/W pointer updating, tag handling...)</td>
</tr>
<tr>
<td>Provides library of blocks to get started</td>
<td>✅ (Growing)</td>
<td>✅ (Huge and well-tested)</td>
</tr>
<tr>
<td>Works with GNU Radio Companion</td>
<td>✅ (Through gr-ettus)</td>
<td>✅ (Built-in)</td>
</tr>
<tr>
<td>Well-documented</td>
<td>✅ (Right?)</td>
<td>✅ (Right? RIGHT?)</td>
</tr>
<tr>
<td>Writes your blocks for you</td>
<td>❌</td>
<td>❌</td>
</tr>
</tbody>
</table>
How it started: Opening the box

- Simple OFDM Transmitter flow graph:
  - Entire Hardware stack is treated like a reprogrammable ASIC. Features are used as-is (white box).
  - GNU Radio’s flexibility should extend onto the FPGA, all the way up to the ADC/DAC!
What’s the problem?

- Everything USRP is available online (code, schematics)
- X310 contains big and expensive FPGA!
- Seems like everything is there to move flow graphs onto the FPGA?
What’s the problem?

- Everything USRP is available online (code, schematics)
- X310 contains big and expensive FPGA!
- Seems like everything is there to move flow graphs onto the FPGA?

- FPGA Development is difficult:
  - Takes a lot of time
  - Lots of things to consider (timing, clocks, busses, ...)
- Needs transparent integration into software (e.g. GNU Radio, GRC)
Domain vs FPGA Experts

- Know Thy Audience!
- FPGA development is not a requirement of a communications engineering curriculum
- Math is hard too

almost pure-noise channels. This intuition is clarified more by the following inequality. It is shown in [1] that for any B-DMC $W$, 

$$1 - I(W) \leq Z(W) \leq \sqrt{1 - I(W)^2}$$  \hspace{1cm} (2)$$

where $I(W)$ is the symmetric capacity of $W$.

Let $W^N$ denote the channels that results from $N$ independent copies of $W$ i.e. the channel $\langle \{0, 1\}^N, \mathcal{Y}^N, W^N \rangle$ given by

$$W^N(y_1^N \mid x_1^N) \overset{\text{def}}{=} \prod_{i=1}^{N} W(y_i \mid x_i)$$  \hspace{1cm} (3)$$

where $x_1^N = (x_1, x_2, \ldots, x_N)$ and $y_1^N = (y_1, y_2, \ldots, y_N)$. Then the combined channel $\langle \{0, 1\}^N, \mathcal{Y}^N, W \rangle$ is defined with transition probabilities given by

$$\mathcal{W}(y_1^N \mid u_1^N) \overset{\text{def}}{=} W^N(y_1^N \mid u_1^N G_N) = W^N(y_1^N \mid u_1^N R_N G^\otimes n)$$
Example: Wideband Spectral Analysis

- Simple in Theory: 200 MHz real-time, Welch's Algorithm
- In practice: Several stumbling blocks

FPGA: Underutilized

Highly parallelizable operations, basic math => Ideal to shift to FPGA

Transport: Overloaded
- Heterogeneous Processing
- Support composable and modular designs using GPP, FPGA, & beyond
- Maintain ease of use
- Tight integration with GNU Radio
- Heterogeneous Processing
- Support composable and modular designs using GPP, FPGA, & beyond
- Maintain ease of use
- Tight integration with GNU Radio

**Goal**

**FPGA Processing**

**GPP Processing**
RFNoC: RF Network on Chip

- Make FPGA acceleration easier (especially on USRPs)
  - Software API + FPGA infrastructure
    - Handles FPGA – Host communication / dataflow
    - Provides user simple software and HDL interfaces
  - Scalable design for massive distributed processing
  - Fully supported in GNU Radio
- RFNoC + GNU Radio: A perfect match
- Ideal way to use + test RFNoC is with GNU Radio
- Seamlessly interconnects (with some caveats)
Radio block in GNU Radio represents the Radio Core RFNoC block in FPGA.
RFNoC Architecture

User Application – GNU Radio

- RFNoC provides the communication infrastructure

RFNoC: Radio
Radio Select: A
Mode: Rx
Stream Args:
Center Frequency: 1.982G
Sampling Rate: 1M
Gain: 20
Antenna: TX/RX
RFNoC Architecture

**User Application – GNU Radio**

- RFNoC provides space for user logic

**RFNoC: Radio**
- Radio Select: A
- Mode: Rx
- Stream Args:
  - Center Frequency: 1.982G
  - Gain: 20
  - Antenna: TX/RX

**HOST PC**

**USRP FPGA**

**USRP Hardware Driver**

**Crossbar**

**Ingress Egress Interface**

**Radio Core**

**Custom RFNoC Block**

**Custom RFNoC Block**
RFNoC Architecture

User Application – GNU Radio

- RFNoC provides space for user logic

USRP Hardware Driver

Ingress Egress Interface

Radio Core

Custom RFNoC Block

Custom RFNoC Block
RFNoC Architecture

User Application – GNU Radio

Implement FFT as an RFNoC block in FPGA

- FFT
  - FFT Size: 1.024k
  - Forward/Reverse: Forward
  - Window: window.blackmanharris...
  - Shift: yes
  - Num. Threads: 1

- Complex to Mag
  - Vec Length: 1.024k

- Log10
  - m: 20
  - k: 0
  - Vec Length: 1.024k

- QT GUI Vector Sink
  - Vector Size: 1.024k
  - X-Axis Start Value: 0
  - X-Axis Step Value: 1
  - X-Axis Label: x-Axis
  - X-Axis Units: x-Axis
  - Y-Axis Label: y-Axis
  - Y-Axis Units: y-Axis
  - Ref Level: 0
RFNoC Architecture

User Application – GNU Radio

RFNoC: Radio
- Radio Select: A
- Mode: Rx
- Stream Args:
  - Center Frequency: 1.982G
  - Sampling Rate: 1M
  - Gain: 20
  - Antenna: TX/RX

FFT
- FFT Size: 1024
- FFT Output: Complex

Complex to Mag
- Vec Length: 1.024k

Log10
- n: 20
- k: 0
- Vec Length: 1.024k

QT GUI Vector Sink
- Vector Size: 1.024k
- X-Axis Start Value: 0
- X-Axis Step Value: 1
- X-Axis Label: x-Axis
- Y-Axis Label: y-Axis
- X-Axis Units:
- Y-Axis Units:
- Ref Level: 0

HOST PC

USRP FPGA

USRP Hardware Driver

Ingress Egress Interface

Crossbar

Radio Core

FFT

Jose’s Mariachi-Block
RFNoC Architecture

User Application – GNU Radio

HOST PC

USRP FPGA

Radio Core

FFT

Twitter Parser

USRP Hardware Driver

Ingress Egress Interface

Crossbar

RFNoC: Radio
Radio Select: A
Mode: Rx
Stream Args:
Center Frequency: 1.982G
Sampling Rate: 1M
Gain: 20
Antenna: TX/RX

RFNoC: FFT
FFT Size: 1024
FFT Output: Complex

Complex to Mag
Vec Length: 1.024k

Log10
n: 20
k: 0
Vec Length: 1.024k

QT GUI Vector Sink
Vector Size: 1.024k
X-Axis Start Value: 0
X-Axis Step Value: 1
X-Axis Label: x-Axis
Y-Axis Label: y-Axis
X-Axis Units:
Y-Axis Units:
Ref Level: 0
Anatomy of an RFNoC Block

- **Radio Core**
  - Depacketizer
  - Packetizer
  - FIFO
  - TX DSP
  - RX DSP

- **FFT**
  - Depacketizer
  - Packetizer
  - FIFO
  - Xilinx FFT IP
  - AXI-Stream

- **Crossbar**

- **To Host PC**

- **RX Sample Data**
Anatomy of an RFNoC Block

- FIFO to FIFO, packetization, flow control
- Provided by RFNoC infrastructure
Anatomy of an RFNoC Block

- User interfaces to RFNoC via AXI-Stream
  - Industry standard (ARM), easy to use
  - Large library of existing IP cores
User writes their own HDL or drops in IP
- Multiple AXI-Streams, Control / Status registers
Anatomy of an RFNoC Block

- Each block is in their own clock domain
  - Improve block throughput, timing
  - Interface to Crossbar has clock crossing FIFOs
Anatomy of an RFNoC Block

- All you need to do is fill this part!
Many computation engines

Not limited to one crossbar, one device
- Scales across devices for massive distributed processing
Many Types of Blocks

- Low latency protocol processing in FPGA
RFNoC Architecture

User Application – GNU Radio

- Transparent protocol conversion
- Multiple standards PCI-E, 10 GigE, AXI
  - Could be wire through -- forwarding to another crossbar
- Parallel interfaces (example: X300 has 2 x 10 GigE)
RFNoC Architecture

User Application – GNU Radio

- Software API to:
  - Configure USRP hardware & RFNoC FPGA infrastructure
  - Provide user sample data (r/w buffers) & control (r/w regs) interfaces
  - Transparent mapping of settings e.g. from GRC to FPGA regs
  - C++ or NocScript
Summary

- Simple architecture for heterogeneous data flow processing
  - The GNU Radio of FPGAs!
- Several interesting blocks exist as reference and examples
- Integrated with GNU Radio + GRC
- Portable between all third generation USRPs
  - X3x0, E310, and products soon to come
- Completely open source (UHD + FPGA codes)
  - github.com/EttusResearch/uhd/wiki/RFNoC:-Getting-Started