RFNoC™: Evolving SDR Toolkits to the FPGA platform

Martin Braun

31.1.2016
USRP: A White Box?

- Simple OFDM Transmitter Development:
  - Entire Hardware stack is treated like a reprogrammable ASIC, Features are used as-is

  ![Diagram showing OFDM Transmitter Development]

  All the interesting parts processed on GPP

- FPGA handles DUC, CORDIC, etc. transparently
Open the Box!

- Everything USRP is available online (code, schematics)
- Contains big and expensive FPGA!
FPGAs: Hard to use... slow to develop

THE #1 PROGRAMMER EXCUSE FOR LEGITIMATELY SLACKING OFF:
"MY CODE'S COMPILING."

HEY! GET BACK TO WORK!

COMPILING!

OH. CARRY ON.
Know Thy Audience!

FPGA development is not a requirement of a communications engineering curriculum

Math is hard too
Example: Wideband Spectral Analysis

- Simple in Theory: 200 MHz real-time, Welch's Algorithm

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

Transport: Overloaded

FPGA: Underutilized
Goal

- 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 popular SDR frameworks
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 Diagram](image.png)
RFNoC Architecture

User Application – GNU Radio

- Example: Plotting frequency spectrum

Ingress Egress Interface

Crossbar

Radio Core

Computation Engine

Computation Engine

USRP FPGA

HOST PC

Ettus Research

A National Instruments Company
Radio block in GNU Radio represents the Radio Core RFNoC block in FPGA
RFNoC Architecture

User Application – GNU Radio

- RFNoC provides the communication infrastructure

- USRP Hardware Driver
  - Ingress Egress Interface
    - Crossbar
      - Radio Core
      - Computation Engine
      - Computation Engine
RFNoC Architecture

User Application – GNU Radio

- RFNoC provides space for user logic called Computation Engines
RFNoC Architecture

User Application – GNU Radio

- RFNoC provides space for user logic called Computation Engines
RFNoC Architecture

User Application – GNU Radio

- Implement FFT as a Computation Engine in FPGA
RFNoC Architecture

User Application – GNU Radio
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

- **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
  - Y-Axis Units: Ref Level
  - Y-Axis Units: 0

- **HOST PC**

- **USRP FPGA**

- **USRP Hardware Driver**

- **Ingress Egress Interface**

- **Crossbar**

- **Radio Core**

- **FFT**

- **Dogecoin Mining**
Computation Engine

- FIFO to FIFO, packetization, flow control
- Provided by RFNoC infrastructure
### 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
Each block is in their own clock domain
- Improve block throughput, timing
- Interface to Crossbar has clock crossing FIFOs
Many computation engines

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

- Radio Core
- FFT
- FIR
- Demodulator
- Crypto Core
- Compression/Decompression
- Soft Processor MicroBlaze

- 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**

- **USRP Hardware Driver**
- **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
RFNoC Architecture

User Application – GNU Radio

Radio Core

FFT

Computation Engine

Crossbar

Ingress Egress Interface

USRP Hardware Driver

HOST PC

USRP FPGA

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
DEMO
RFNoC Stack (Even Simpler)

Your Application here!

UHD Integration

Block Declaration (XML / NocScript)  Default Block Controller

FPGA Integration

Verilog / VHDL / CoreGen / IP
Summary

- Simple architecture for heterogeneous data flow processing
- Several interesting blocks already exist
- Integrated with GNU Radio
- Portable between all third generation USRPs
  - X3x0, E310, and products soon to come
- Completely open source (within Xilinx toolchains)
- Available on github!
  - github.com/EttusResearch/uhd/wiki/RFNoC:-Getting-Started