Outline

- Host-based Software Defined Radio
  - Current situation
  - Goal
- RFNoC
  - Architecture overview
- Demo
- Conclusion
Host-Based SDR

- PC + Flexible RF Hardware + SDR Framework
- Typical SDR frameworks excel at software reconfigurability & composability
- Less true for available hardware, especially FPGAs
Example GNU Radio + USRP

- Simple spectrum monitor:
  - USRP FPGA underutilized; Features used as-is
  - Highly parallelizable algorithms + large FPGA!
  - Software & FPGA source code are open source
  - Why are so few using FPGA acceleration?
- Simple spectrum monitor:

- USRP FPGA underutilized; Features used as-is
- Highly parallelizable algorithms + large FPGA!
- Software & FPGA source code are open source
- Why are so few using FPGA acceleration?
FPGAs: Hard to use... slow to develop
Domain vs FPGA Experts

- Experts in DSP & algorithms may not have practical FPGA design experience
- Also must learn overall architecture:
  - Dataflow, signal timing, clocking
  - Software & Hardware interfacing (PCIe, GigE, DRAM)
Heterogeneous Processing

Support composable and modular designs using GPP, FPGA, & beyond

Maintain ease of use

Tight integration with popular SDR frameworks
- 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 Architecture

User Application – GNU Radio

- Example: Spectrum monitoring
Radio block in GNU Radio represents the Radio Core RFNoC block in FPGA.
RFNoC Architecture

User Application – GNU Radio

- RFNoC provides the communication infrastructure

Radio Core

Computation Engine

Computation Engine

HOST PC

USRP FPGA

USRP Hardware Driver

Ingress Egress Interface

Crossbar
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: 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

USRP Hardware Driver

Ingress Egress Interface

Crossbar

Radio Core

FFT

Computation Engine
Computation Engine

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

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

To Host PC

RX Sample Data
- 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
- Routes packets between Host PC, Radio Cores, Computation Engines, & other devices
  - Fully connected crossbar supporting arbitrary topologies
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 CEs

- Low latency protocol processing in FPGA
RFNoC Architecture

- 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)

User Application – GNU Radio

- Ingress Egress Interface
- Crossbar
- Radio Core
- FFT
- Computation Engine
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
RFNoC Architecture

User Application – GNU Radio

HOST PC

USRP FPGA

USRP Hardware Driver

Ingress Egress Interface

Crossbar

Radio Core

FFT

Computation Engine

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

GNU Radio
RFNoC Architecture

User Application – GNU Radio

HOST PC

USRP FPGA

USRP Hardware Driver

Ingress Egress Interface

Crossbar

Radio Core

Window

FFT

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

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

RFNoC: Window
Coeffs: [0.2,7,16,29,45,65,...]

FFT Size: 1024
FFT Output: Complex

Complex to Mag
Vec Length: 1.024k

Log10
n: 20
k: 0
Vec Length: 1.024k
How to add a Computation Engine

**FPGA**

- IP
  - (e.g. Xilinx Coregen, Hand-written Verilog...)

**UHD**

- XML Block Description
- C++ Block Controller

**GNU Radio**

- GRC XML Block Description
- C++ Block Controller
How to add a Computation Engine

FPGA
- IP
  (e.g. Xilinx Coregen, Hand-written Verilog...)

UHD
- XML Block Description
- C++ Block Controller

GNU Radio
- GRC XML Block Description
- C++ Block Controller
Conclusion

- Simple architecture for heterogeneous data flow processing
- Implemented several interesting CEs
- Integrated with high level SDR framework
- Portable between all third generation USRPs
  - X3x0, E310, and products soon to come
- Completely open source
- Beta release available!
  - github.com/EttusResearch/uhd/wiki/RFNoC:-Getting-Started