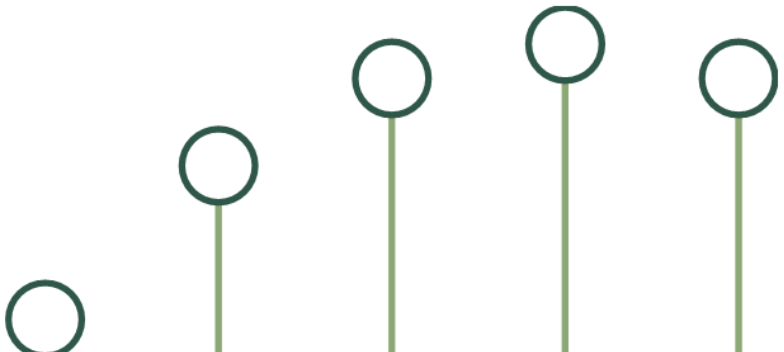
Ettus Research™
A National Instruments Company

RFNoC 4 Workshop

Part 1

Jonathon Pendlum – Ettus Research
Neel Pandeya – Ettus Research

GRCon 2020



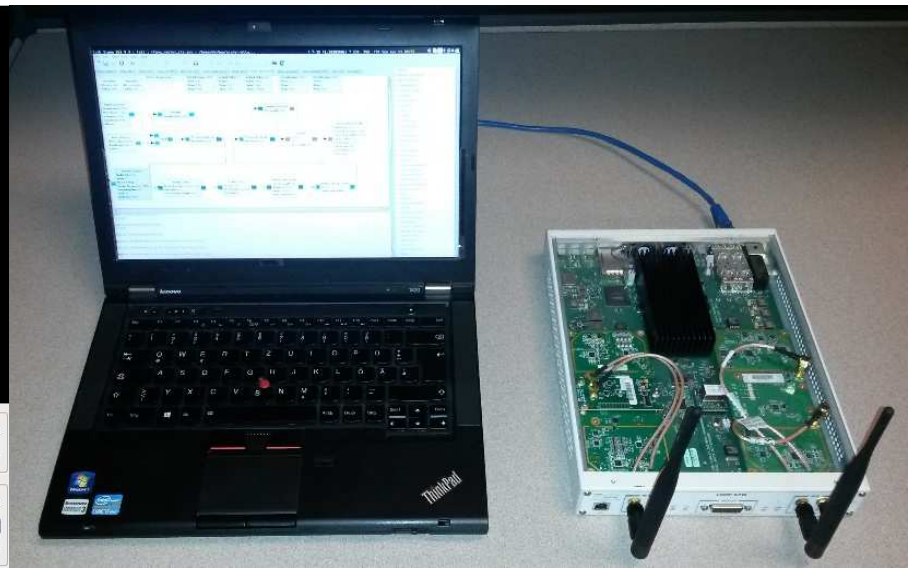
Schedule

- Part 1
  - RFNoC 4 Framework Overview
  - Hands on Demos
- Part 2
  - FPGA Architecture
  - Software Implementation
  - GNU Radio Integration
  - Hands on RFNoC Block Development

Host-Based SDR – Current Situation

Ettus Research™

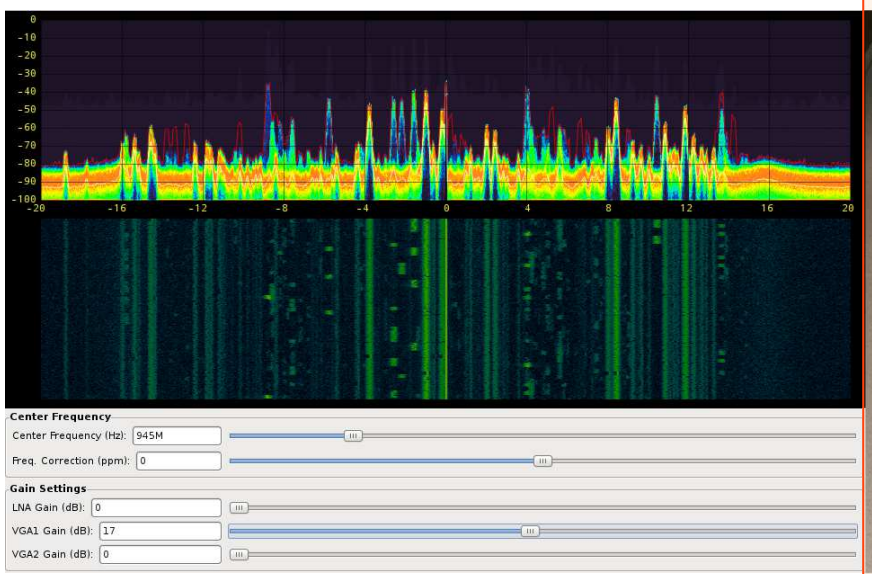
- PC + Flexible RF Hardware + SDR Framework

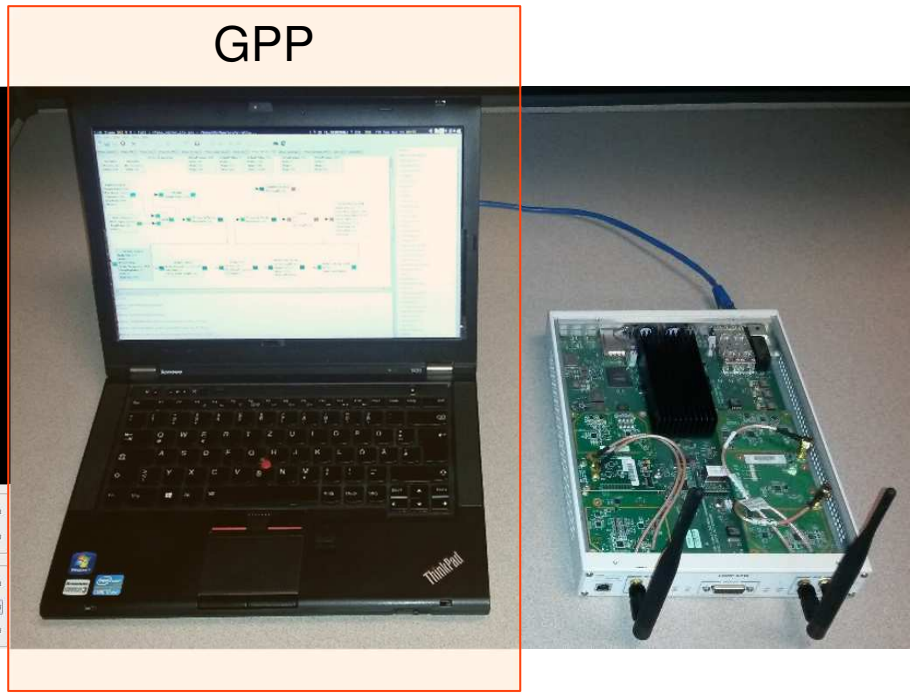


Host-Based SDR – Current Situation

Ettus Research™

- PC + Flexible RF Hardware + SDR Framework
  - GPP: Multi-core + SIMD -- GNU Radio

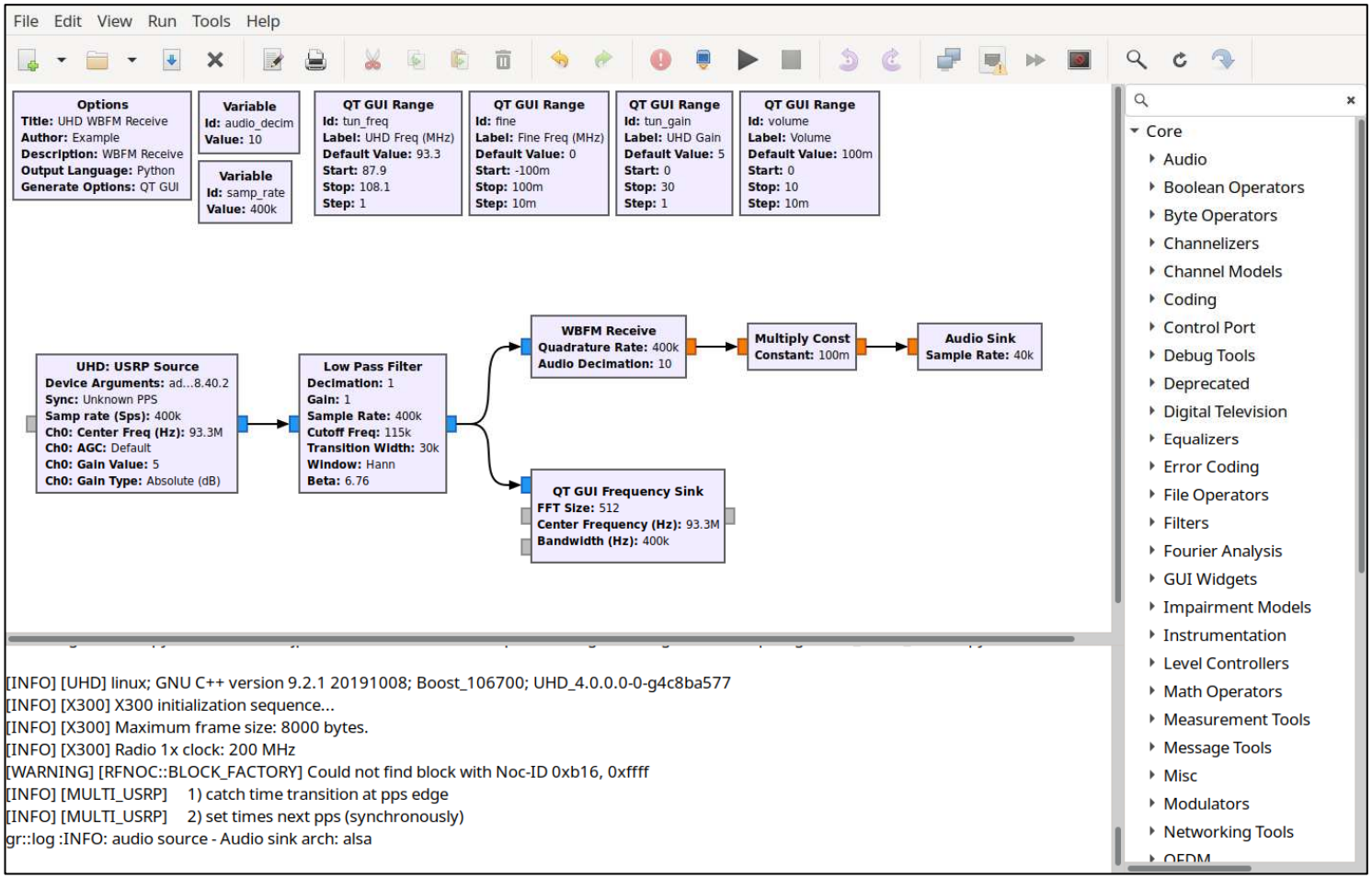




GNU Radio

Ettus Research™

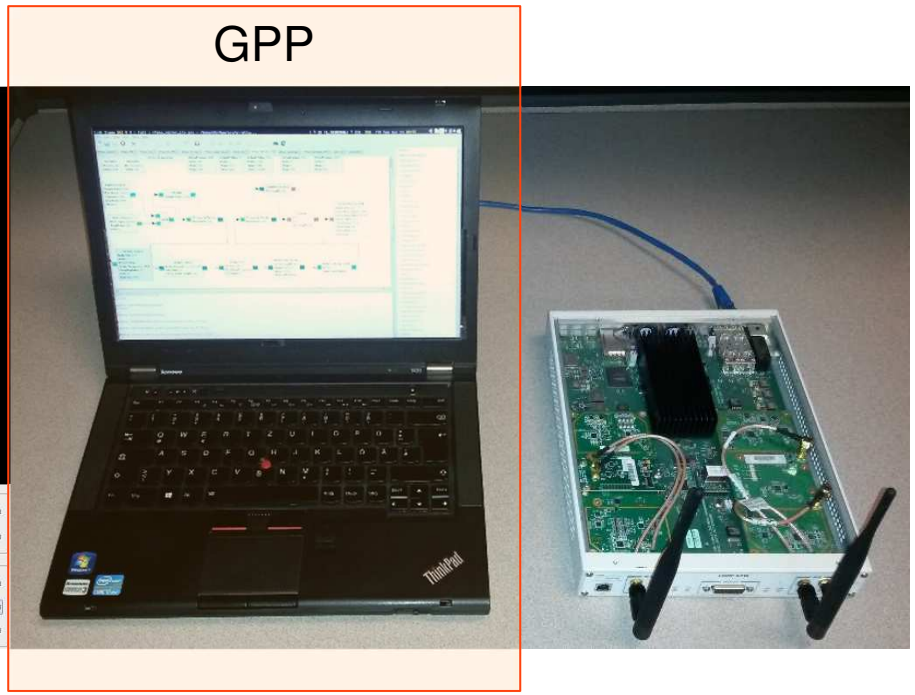
Open source toolkit for developing software radios



Host-Based SDR – Current Situation

Ettus Research™

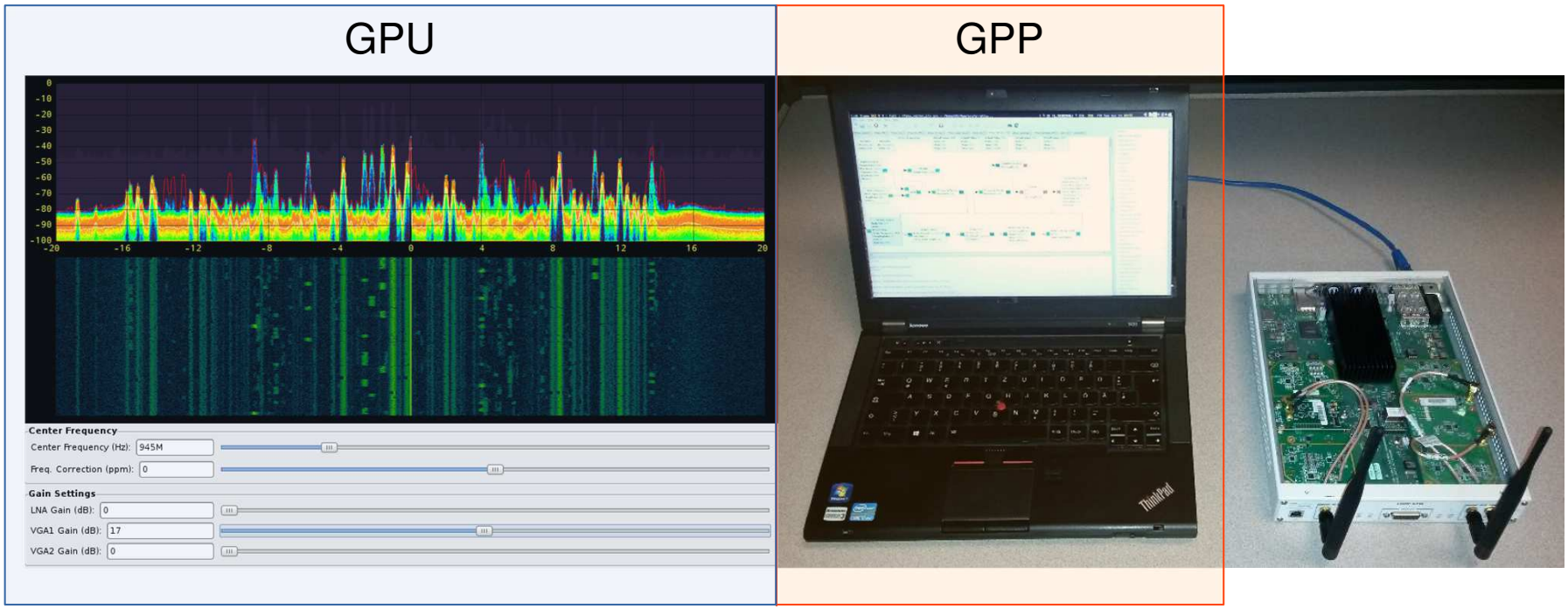
- PC + Flexible RF Hardware + SDR Framework
  - GPP: Multi-core + SIMD -- GNU Radio



Host-Based SDR – Current Situation

Ettus Research™

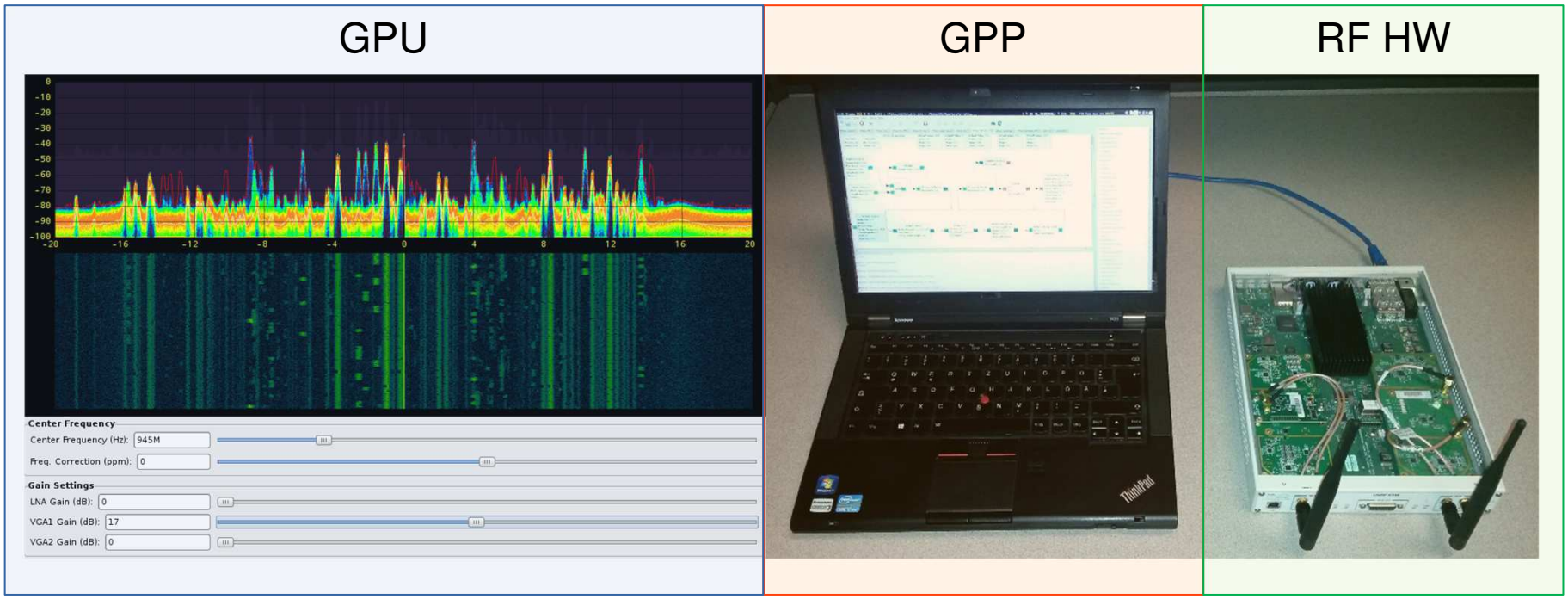
- PC + Flexible RF Hardware + SDR Framework
  - GPP: Multi-core + SIMD -- GNU Radio
  - GPU: High performance FP -- OpenCL, gr-fosphor



Host-Based SDR – Current Situation

Ettus Research™

- PC + Flexible RF Hardware + SDR Framework
  - GPP: Multi-core + SIMD -- GNU Radio
  - GPU: High performance FP -- OpenCL, gr-fosphor
  - RF HW: Wide bandwidth, large FPGA -- Rate change DSP



Universal Software Radio Peripheral

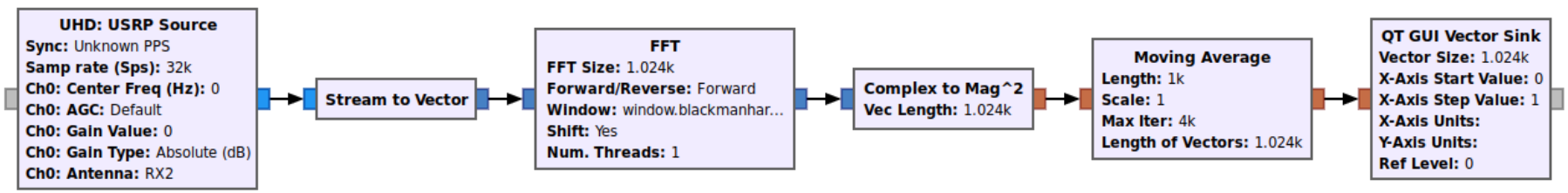
Ettus Research™

	Gen 1	Gen 2	Gen 3 (E310)	Gen 3 (X310)
FPGA	Cyclone 1	Spartan 3	Zynq	Kintex 7
Logic Cells	12K	53K	85K	406K
Memory	26KB	252KB	560KB	3180KB
Multipliers	NONE!	126	220	1540
Clock Rate	64 MHz	100 MHz	200 MHz	250 MHz
RF Bandwidth	8 MHz	50 MHz	128 MHz	640 MHz
Free Space	NONE!	~50%	~60%	~75%

Challenges

Ettus Research™

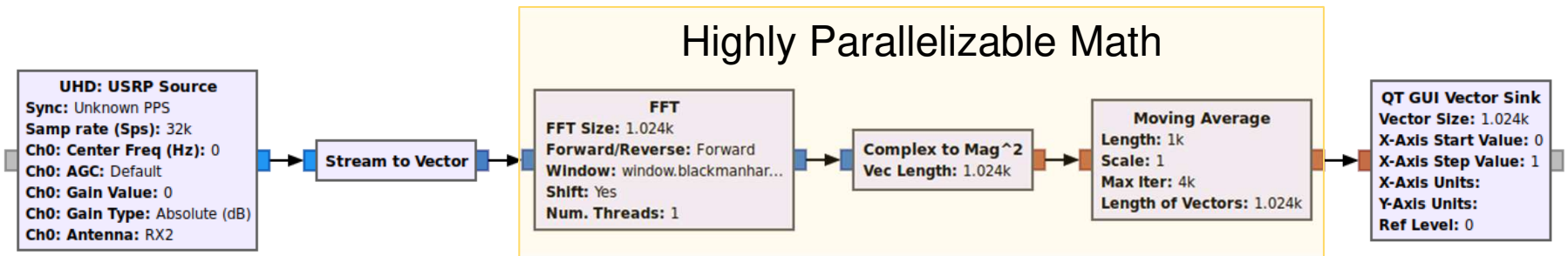
- Massive processing requirements
  - Welsh's algorithm for Power Spectrum Estimation
  - 1024 FFT + $|X|^2$ + Moving Average at 200 MSPS



Challenges

Ettus Research™

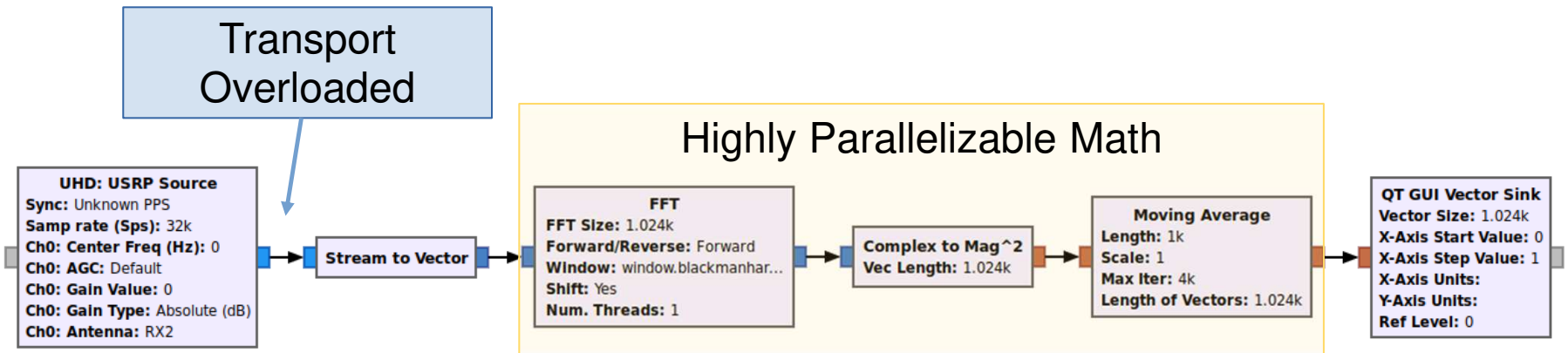
- Massive processing requirements
  - Welsh's algorithm for Power Spectrum Estimation
  - 1024 FFT + $|X|^2$ + Moving Average at 200 MSPS



Challenges

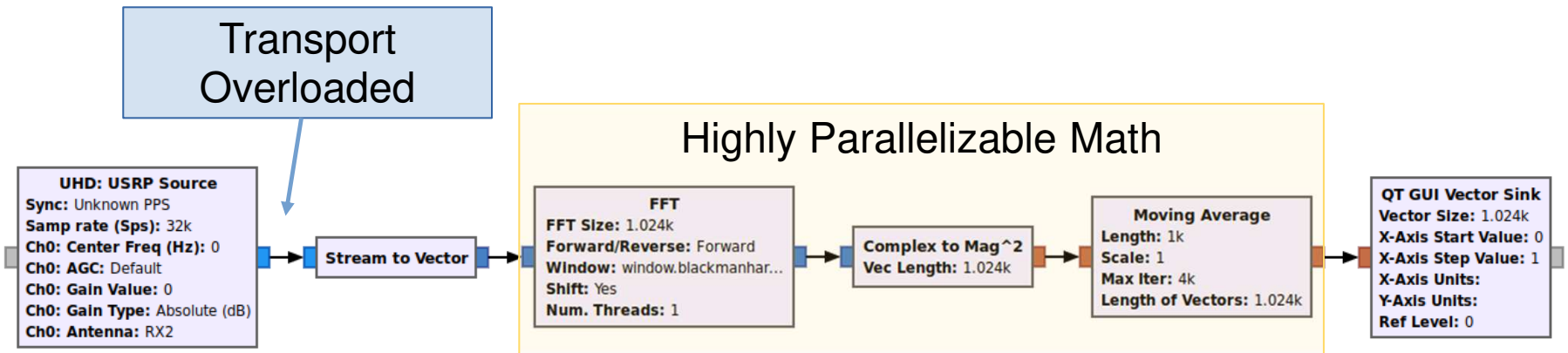
Ettus Research™

- Massive processing requirements
  - Welsh's algorithm for Power Spectrum Estimation
  - 1024 FFT + $|X|^2$ + Moving Average at 200 MSPS
- Overloaded transport
  - 200e6 samp/sec * 32 bits/samp => **6.4 Gb/sec**



Challenges

- Massive processing requirements
  - Welsh's algorithm for Power Spectrum Estimation
  - 1024 FFT + $|X|^2$ + Moving Average at 200 MSPS
- Overloaded transport
  - 200e6 samp/sec * 32 bits/samp => **6.4 Gb/sec**
- Latency and Determinism
  - Ethernet latency, OS scheduling, precise timing



Opportunity: Use the FPGA!

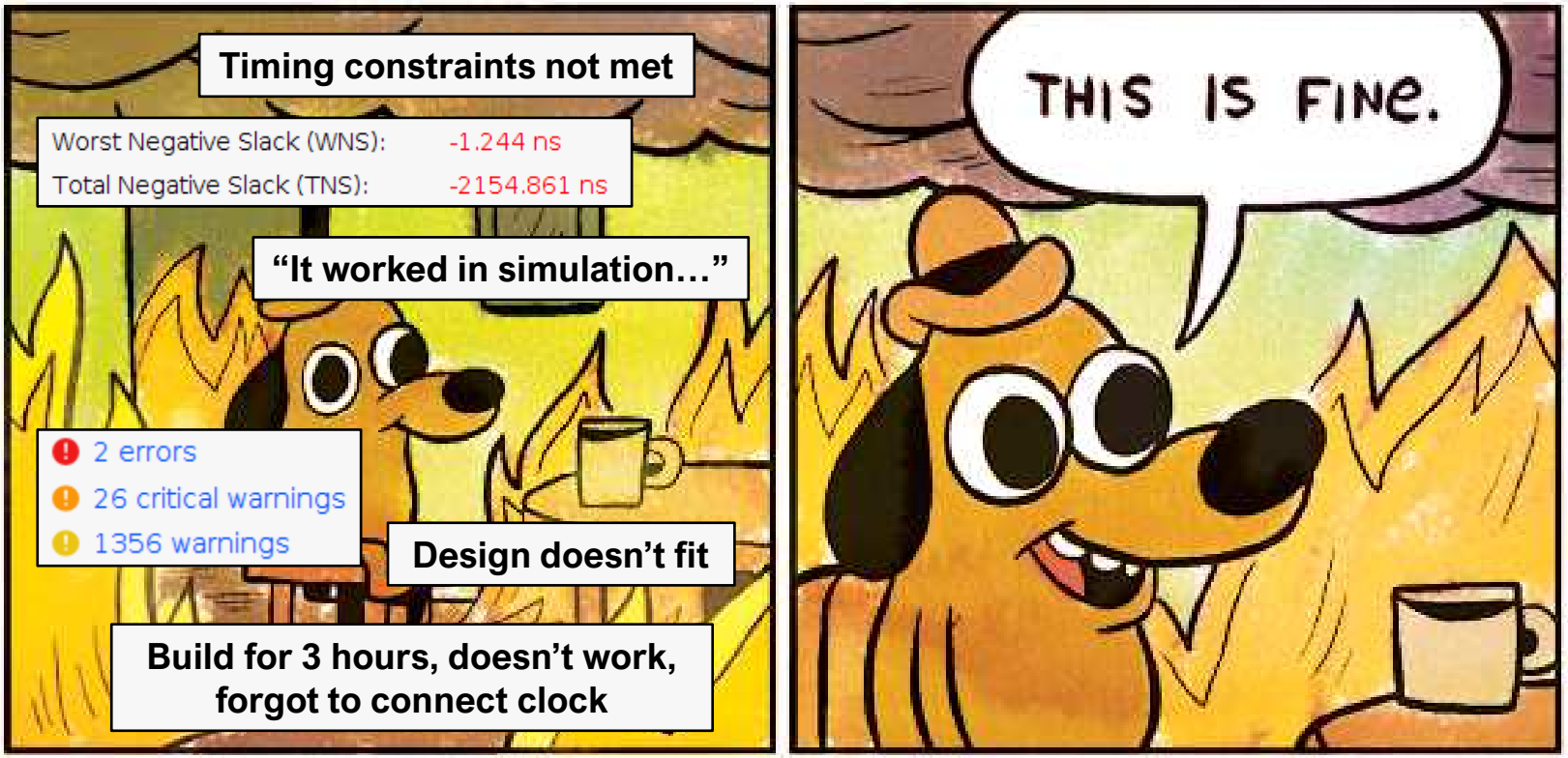
Ettus Research™

- Everything USRP is open source, available online (code, firmware, schematics)
- Contains big and expensive FPGA!
- Why do customers not use it?

FPGAs are Hard

Ettus Research™

FPGA Design Process



Domain vs FPGA Experts

Ettus Research™

- FPGA development is not a requirement of a communications engineering curriculum
- Math in FPGAs is hard
- Complicated system architecture

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

$$1 - I(W) \leqslant Z(W) \leqslant \sqrt{1 - I(W)^2} \tag{2}$$

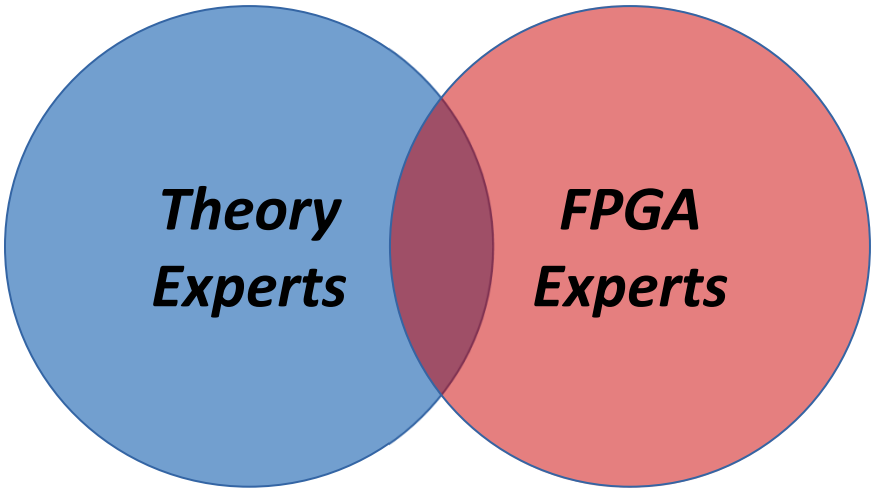
where $I(W)$ is the symmetric capacity of W.

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

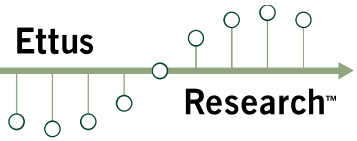
$$W^N(y_1^N | x_1^N) \stackrel{\text{def}}{=} \prod_{i=1}^{N} W(y_i | x_i) \tag{3}$$

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 $\langle \{0,1\}^N, \mathscr{Y}^N, \widetilde{W} \rangle$ is defined with transition probabilities given by

$$\widetilde{W}(y_1^N | u_1^N) \stackrel{\text{def}}{=} W^N(y_1^N | u_1^N G_N) = W^N(y_1^N | u_1^N R_N G^{\otimes n})$$



RFNoC: RF Network on Chip

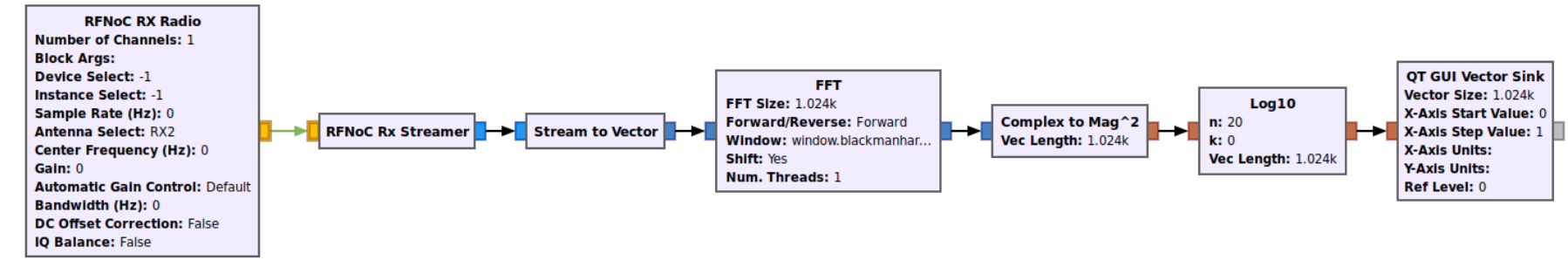


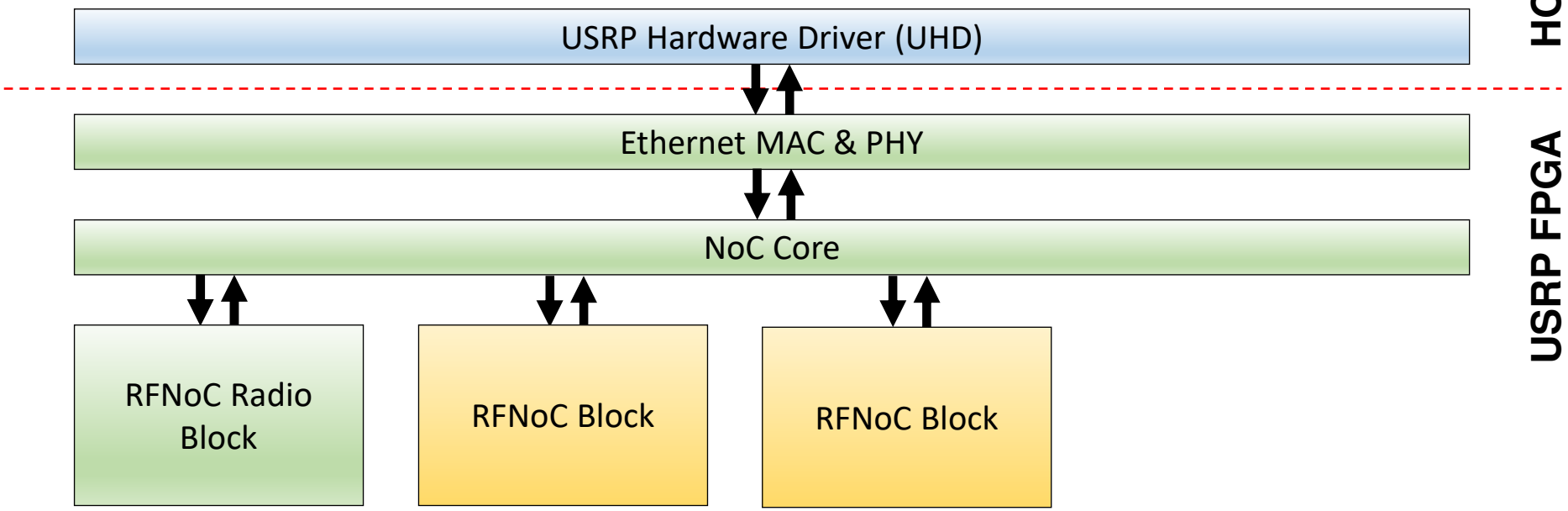
- Make USRP FPGA acceleration more accessible
- Software API + FPGA infrastructure
  - Handles FPGA – Host communication / dataflow
- Provides users simple software and HDL interfaces
  - Infrastructure transparent to user -- reusable code
  - Trade off flexibility versus resource utilization
- Open source
- Fully supported in GNU Radio
  - Modularity and composability

RFNoC Architecture

Ettus Research™

User Application – GNU Radio





RFNoC Architecture

Ettus Research™

User Application – GNU Radio

- Users implement custom FPGA logic in “RFNoC Blocks”
  - Architecturally independent of other logic
  - Easy to add and remove RFNoC Blocks

HOST PC

USRP FPGA

NoC Core

RFNoC Radio Block

RFNoC Block

RFNoC Block

RFNoC Architecture

Ettus Research™

User Application – GNU Radio

- Users implement custom FPGA logic in “RFNoC Blocks”
  - Architecturally independent of other logic
  - Easy to add and remove RFNoC Blocks
- Ettus provides a library of pre-made RFNoC Blocks
  - RFNoC Radio Block connects to the RF Frontend, SPI, GPIO

HOST PC

USRP FPGA

NoC Core

RFNoC Radio Block

RFNoC Block

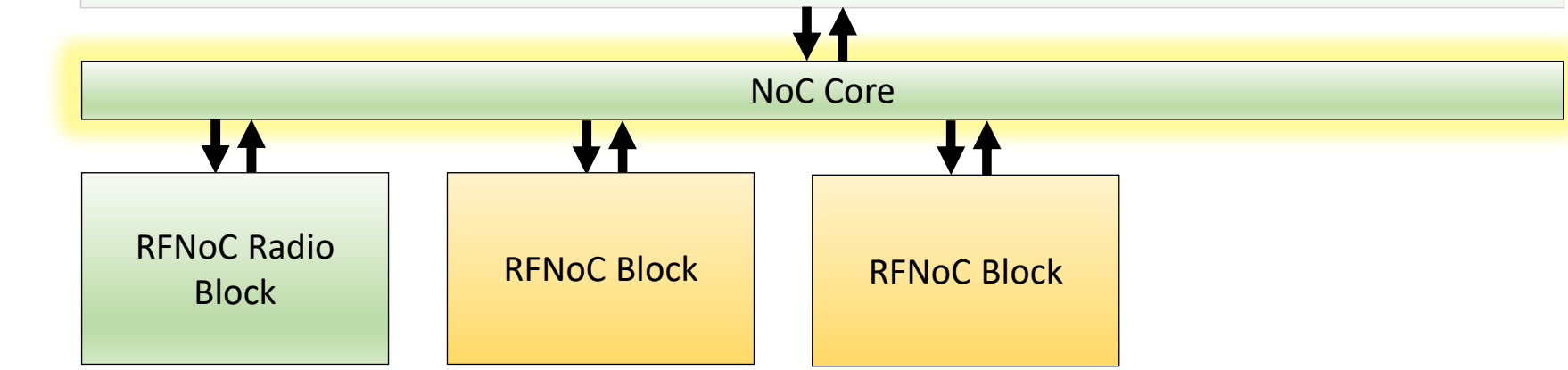
RFNoC Block

RFNoC Architecture

Ettus Research™

User Application – GNU Radio

- NoC Core connects RFNoC Blocks to each other and I/O interfaces (e.g. Ethernet, PCIe, etc.)
- Supports both full crossbar connections and static routing
  - Trade off lower resource utilization and latency for flexibility
- Autogenerated based on RFNoC Block topology



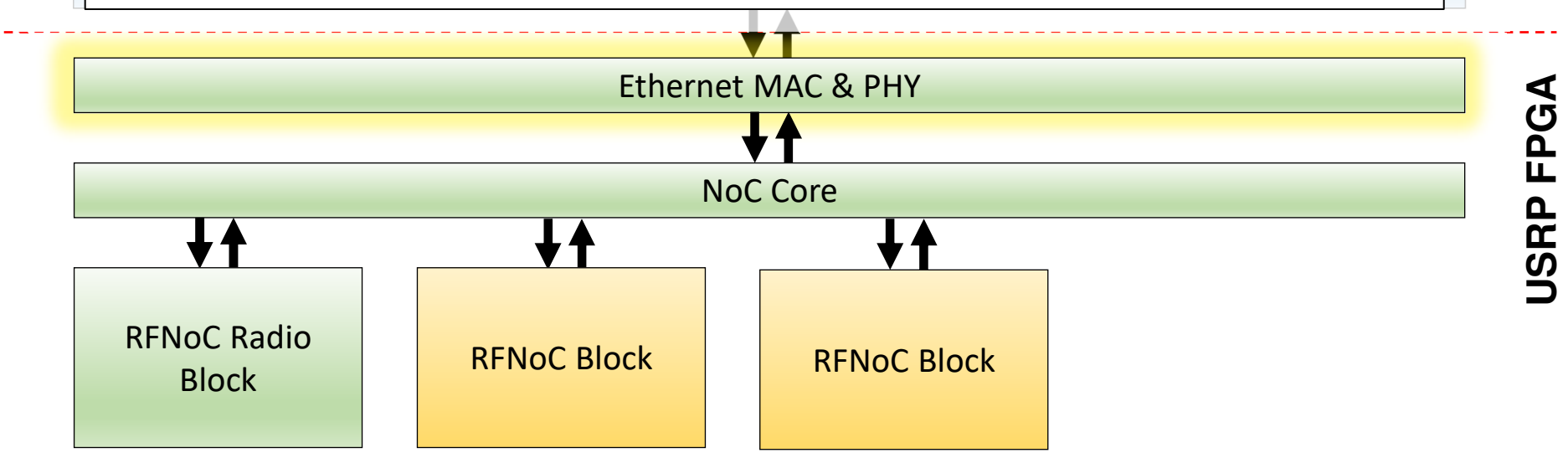
HOST PC

USRP FPGA

RFNoC Architecture

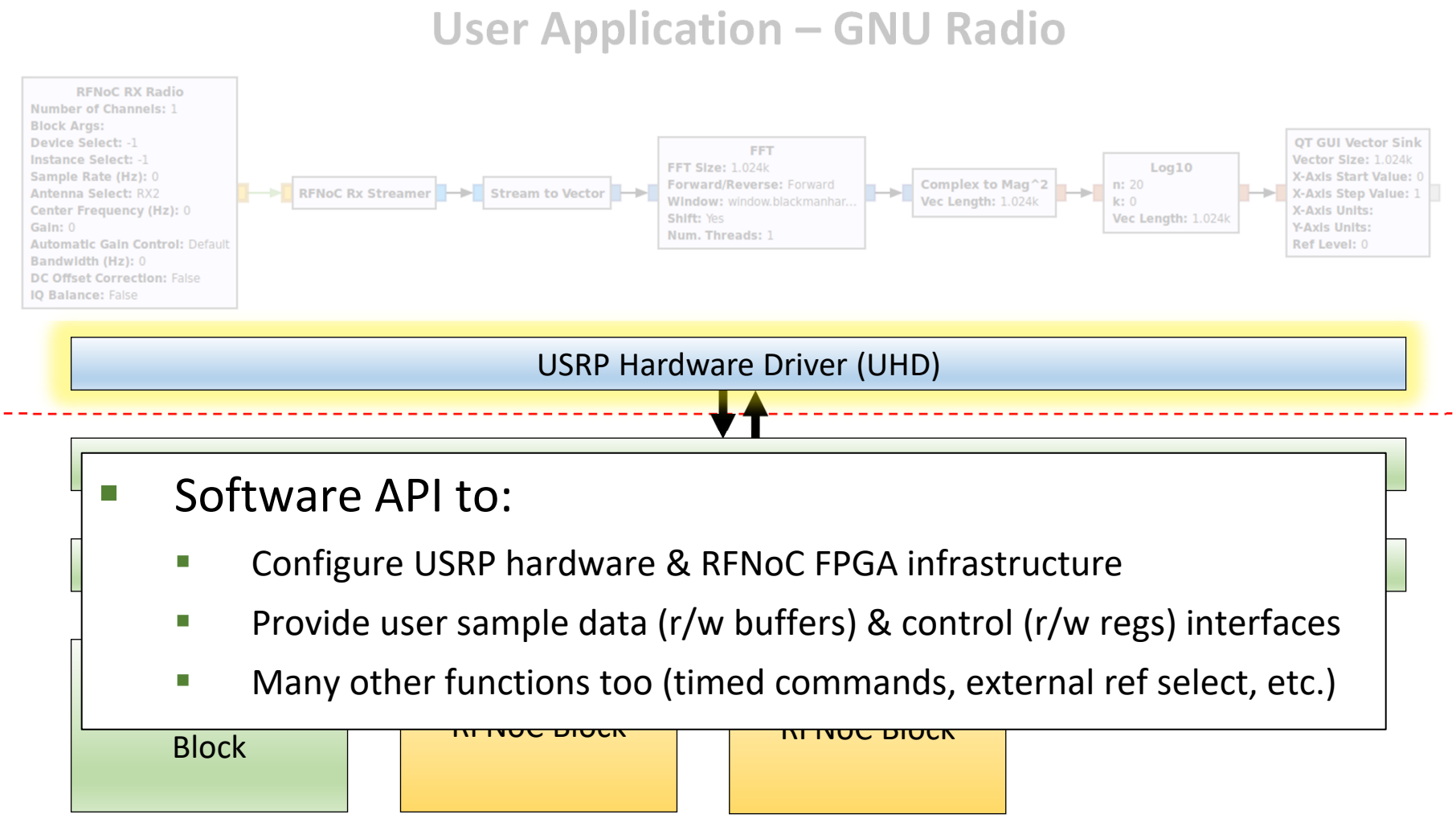
Ettus Research™

- FPGA connection back to the host
  - External connection: 1 GigE, 10 GigE, PCIe, Aurora
  - Internal connection: AXI4 DMA to Zynq ARM processor (e.g. N310)
  - Parallel interfaces (e.g. X310 has 2 x 10 GigE)
- Connected to NoC Core, but not a RFNoC Block
- Transparent protocol conversion



RFNoC Architecture

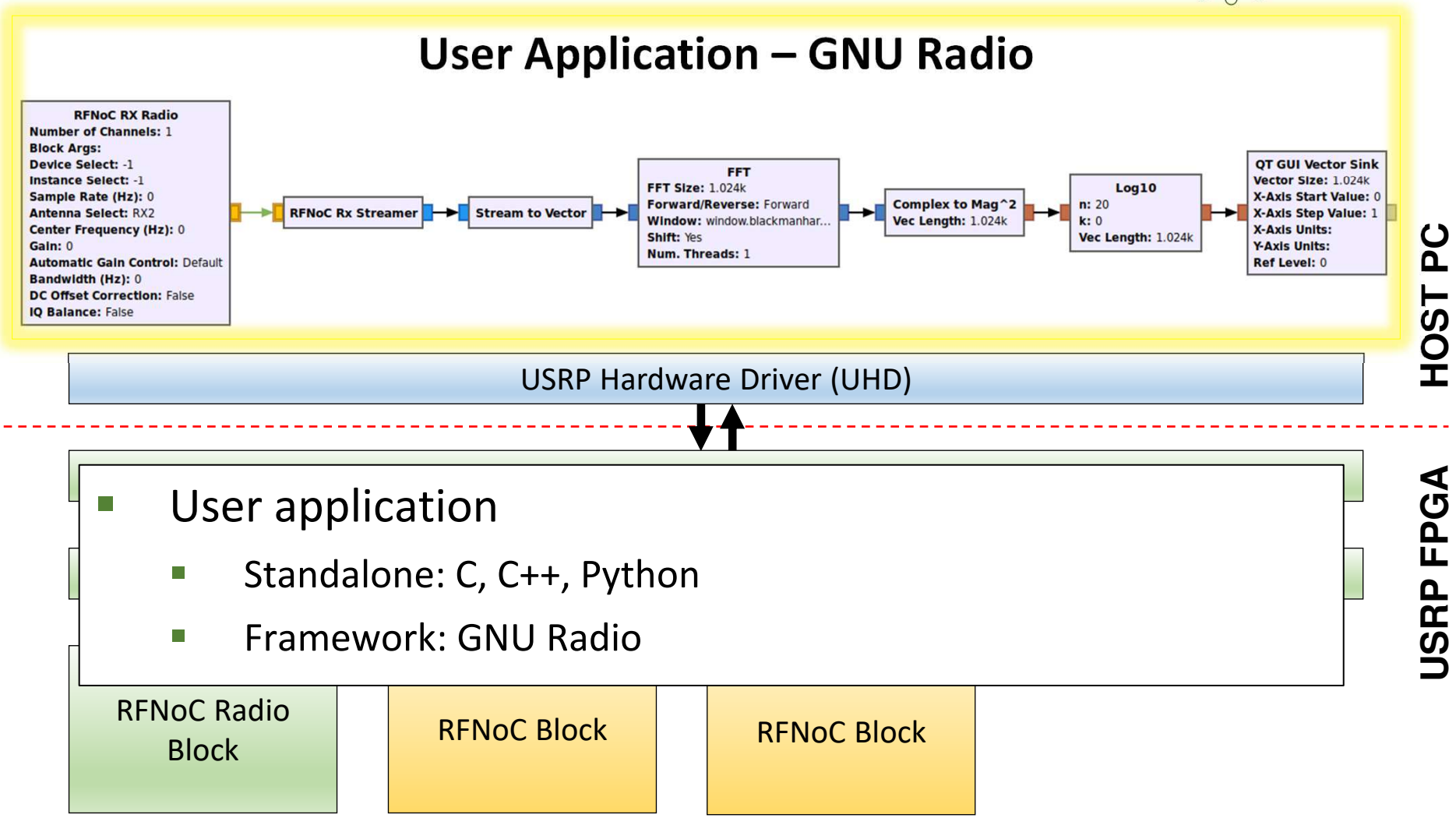
Ettus Research™



- Software API to:
  - Configure USRP hardware & RFNoC FPGA infrastructure
  - Provide user sample data (r/w buffers) & control (r/w regs) interfaces
  - Many other functions too (timed commands, external ref select, etc.)

RFNoC Architecture

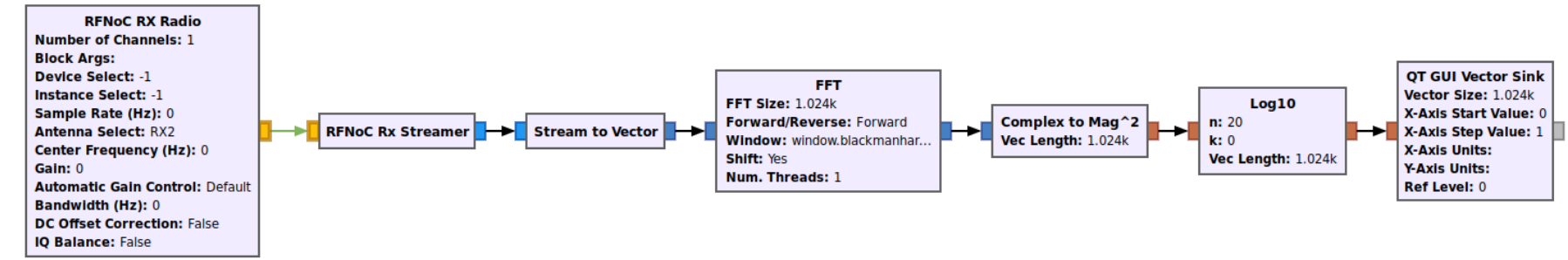
Ettus Research™



RFNoC Architecture

Ettus Research™

User Application – GNU Radio



HOST PC

- Example: Plotting Frequency Spectrum

USRP FPGA

NoC Core

RFNoC Radio Block

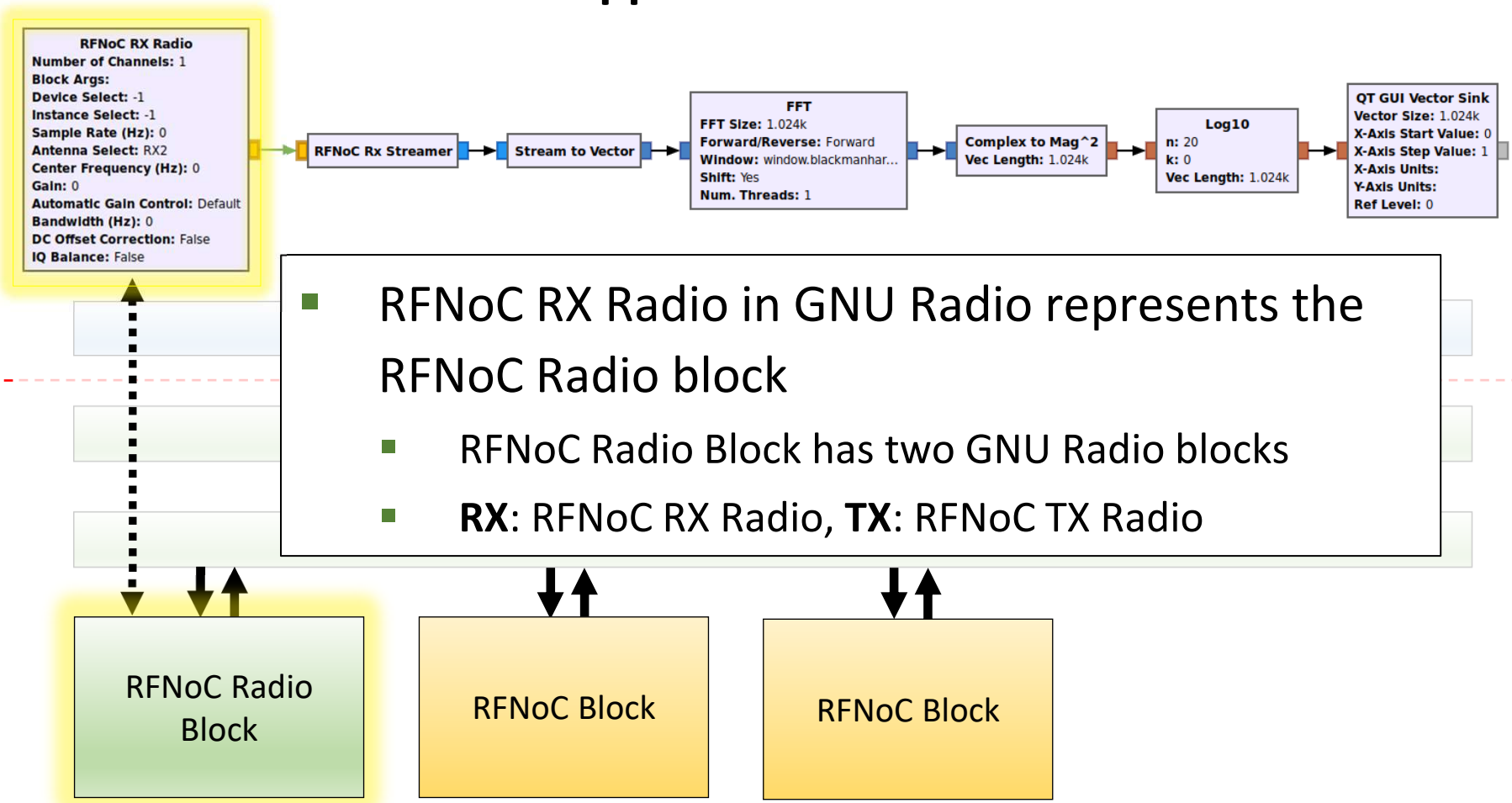
RFNoC Block

RFNoC Block

RFNoC Architecture

Ettus Research™

User Application – GNU Radio

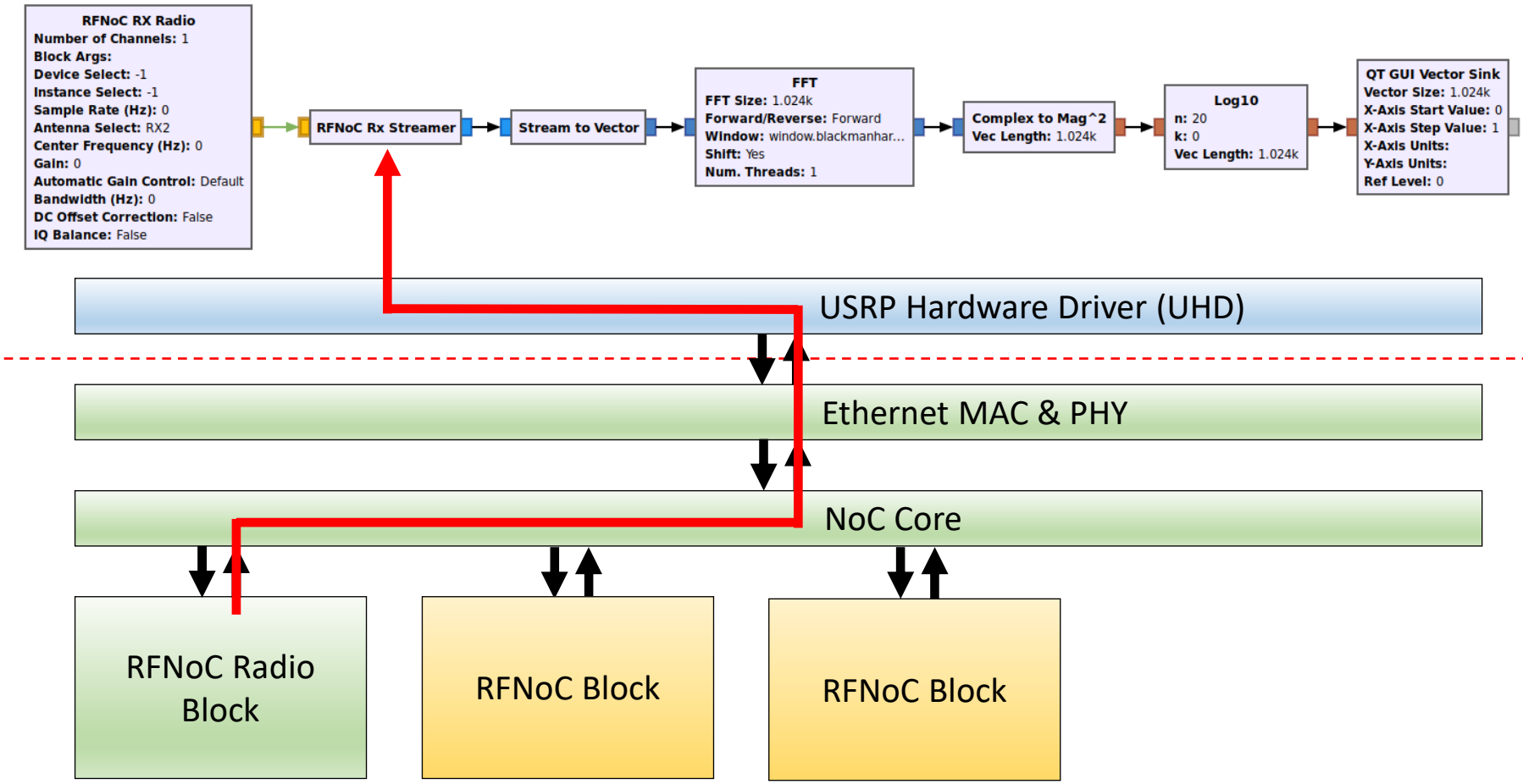


- RFNoC RX Radio in GNU Radio represents the RFNoC Radio block
  - RFNoC Radio Block has two GNU Radio blocks
  - **RX**: RFNoC RX Radio, **TX**: RFNoC TX Radio

RFNoC Architecture

Ettus Research™

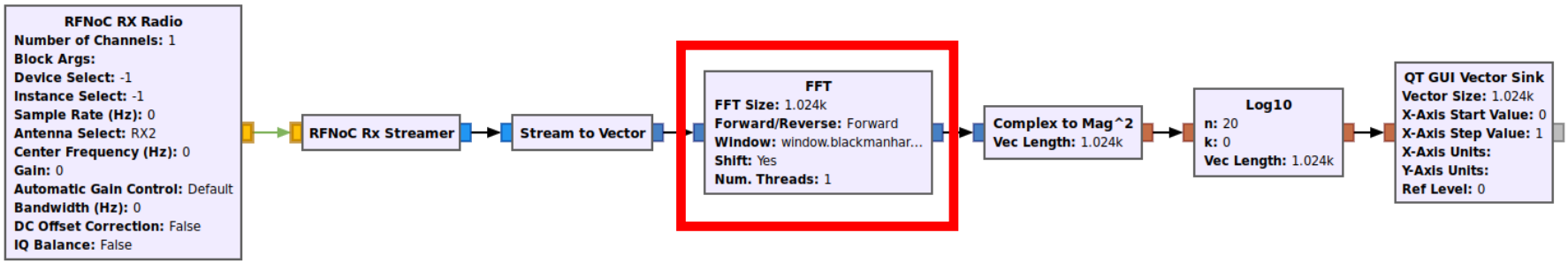
User Application – GNU Radio



RFNoC Architecture

Ettus Research™

User Application – GNU Radio



- Profiling reveals FFT is a hotspot

HOST PC

USRP FPGA

NoC Core

RFNoC Radio Block

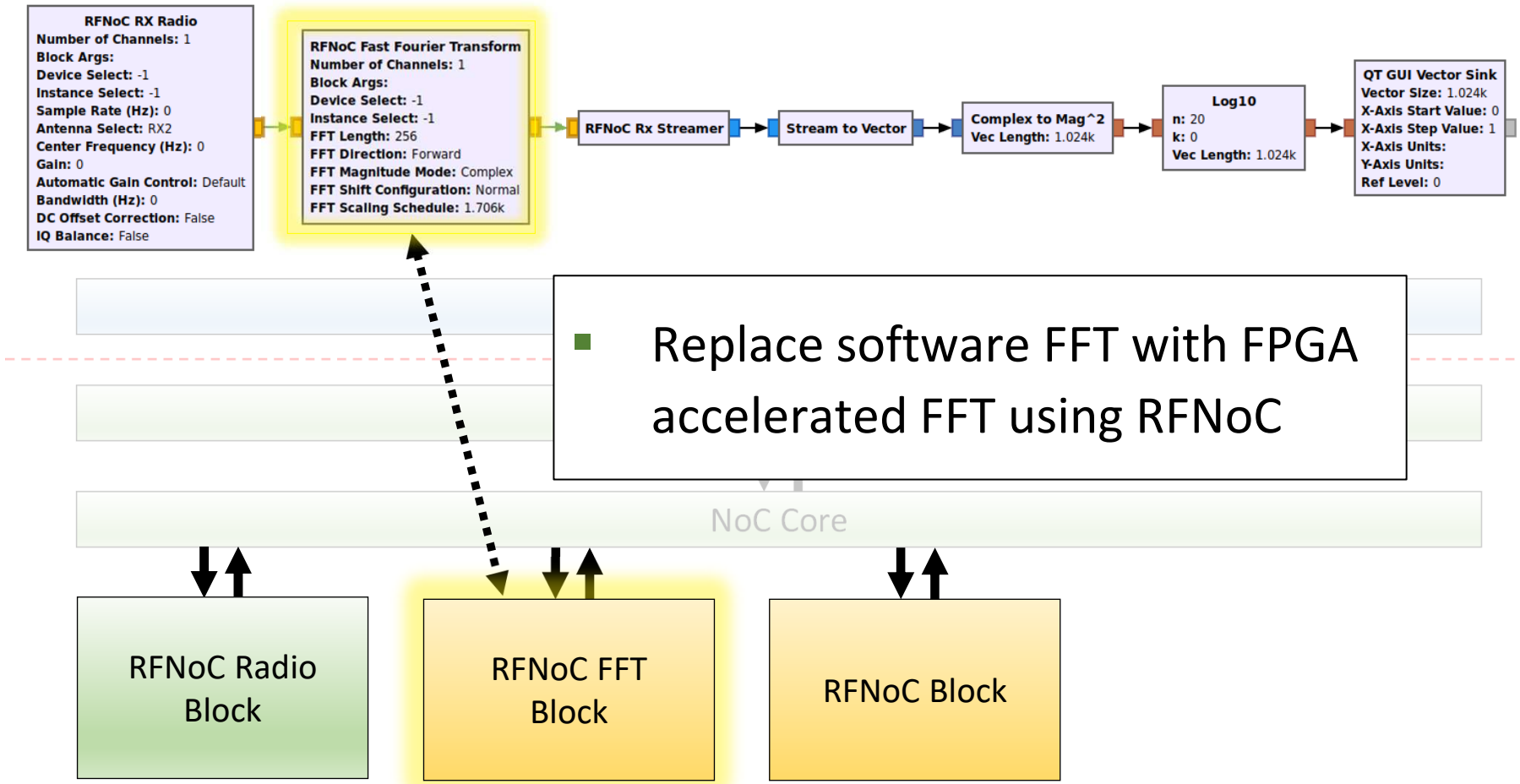
RFNoC Block

RFNoC Block

RFNoC Architecture

Ettus Research™

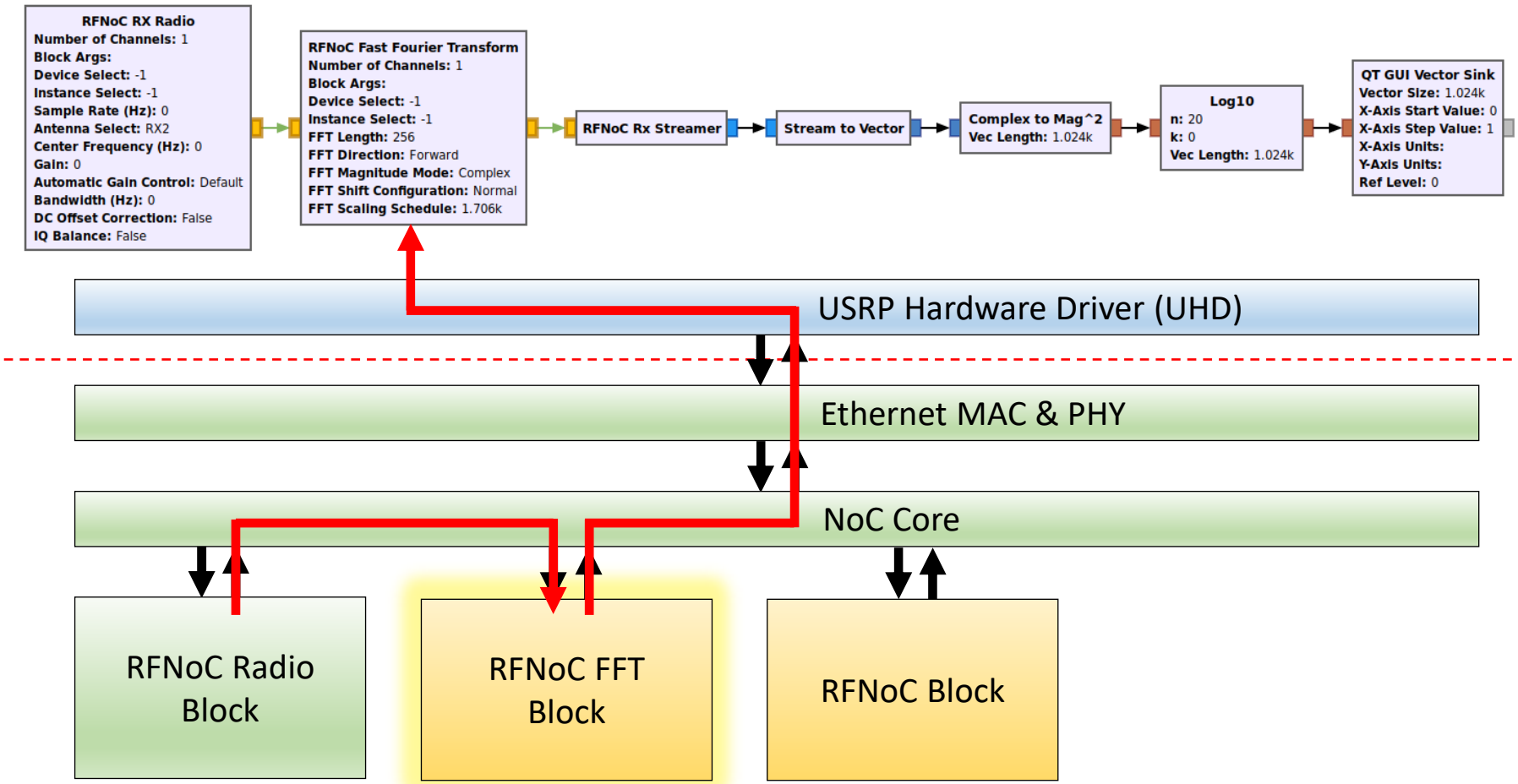
User Application – GNU Radio



RFNoC Architecture

Ettus Research™

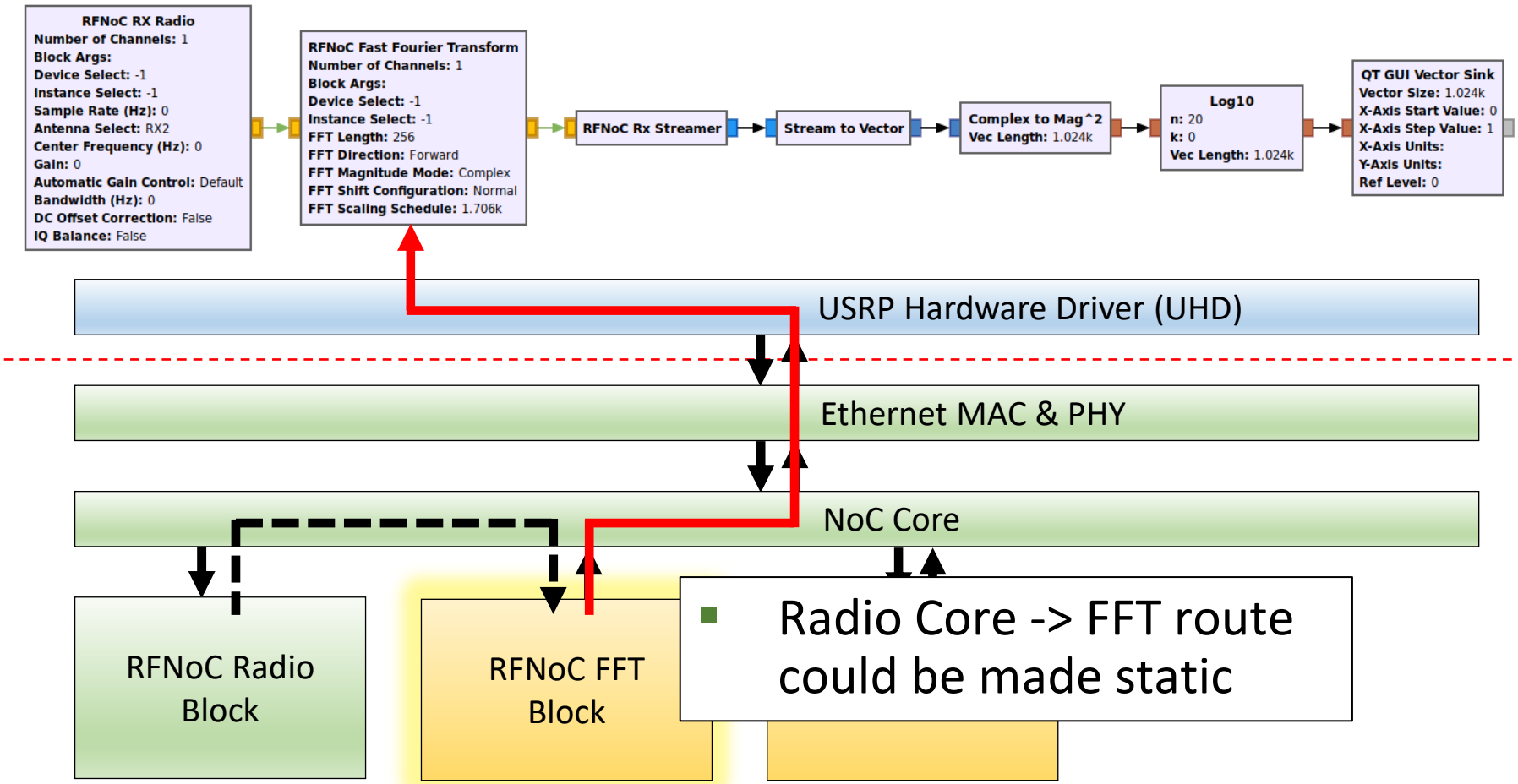
User Application – GNU Radio



RFNoC Architecture

Ettus Research™

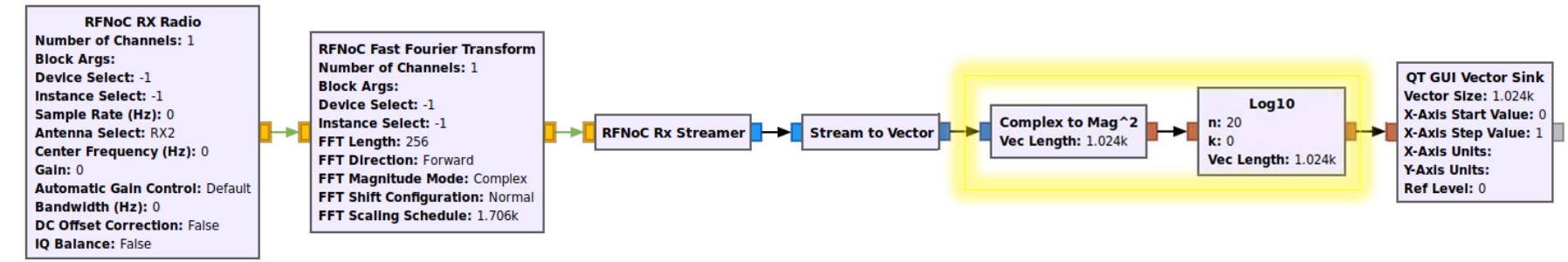
User Application – GNU Radio



RFNoC Architecture

Ettus Research™

User Application – GNU Radio



HOST PC

- Profile again and decide if more FPGA acceleration is needed

NoC Core

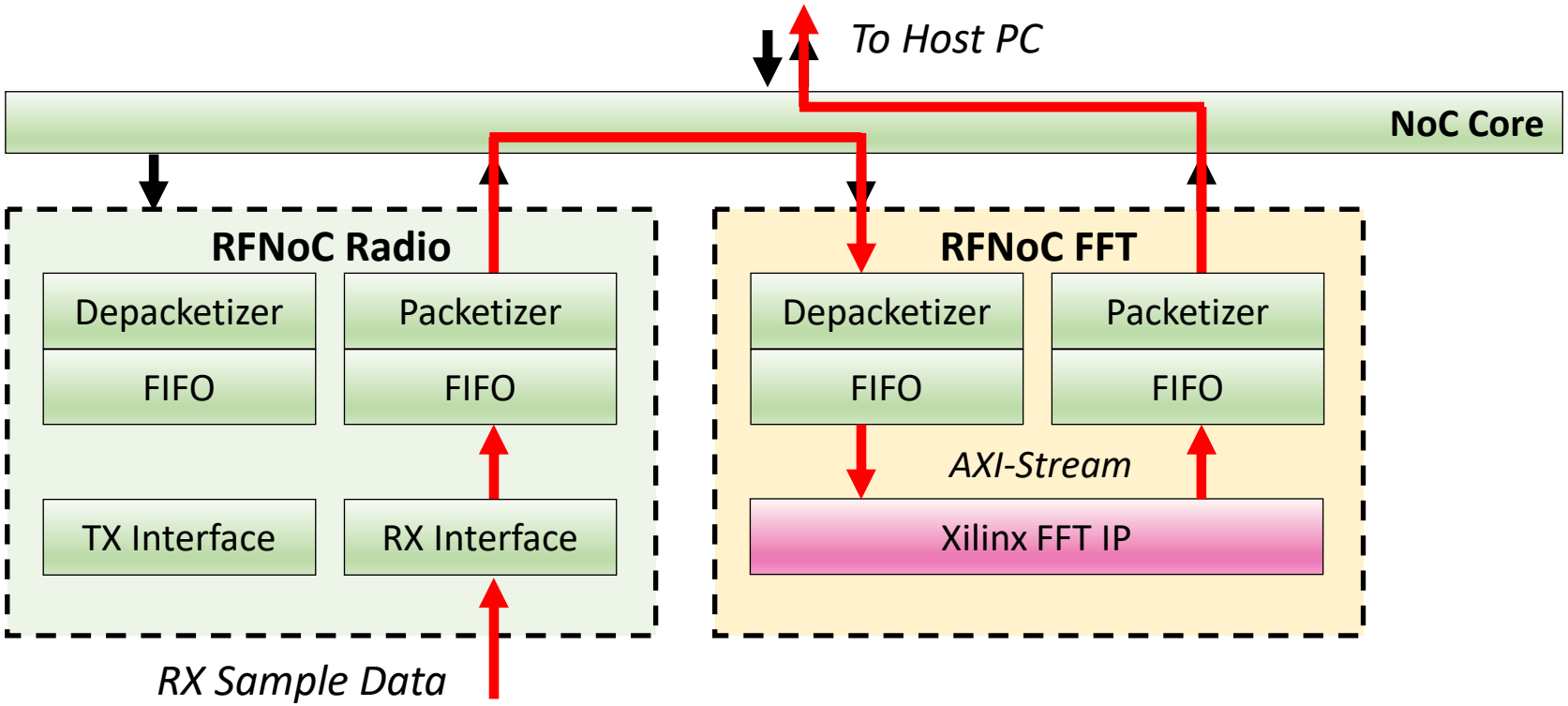
USRP FPGA

RFNoC Radio Block

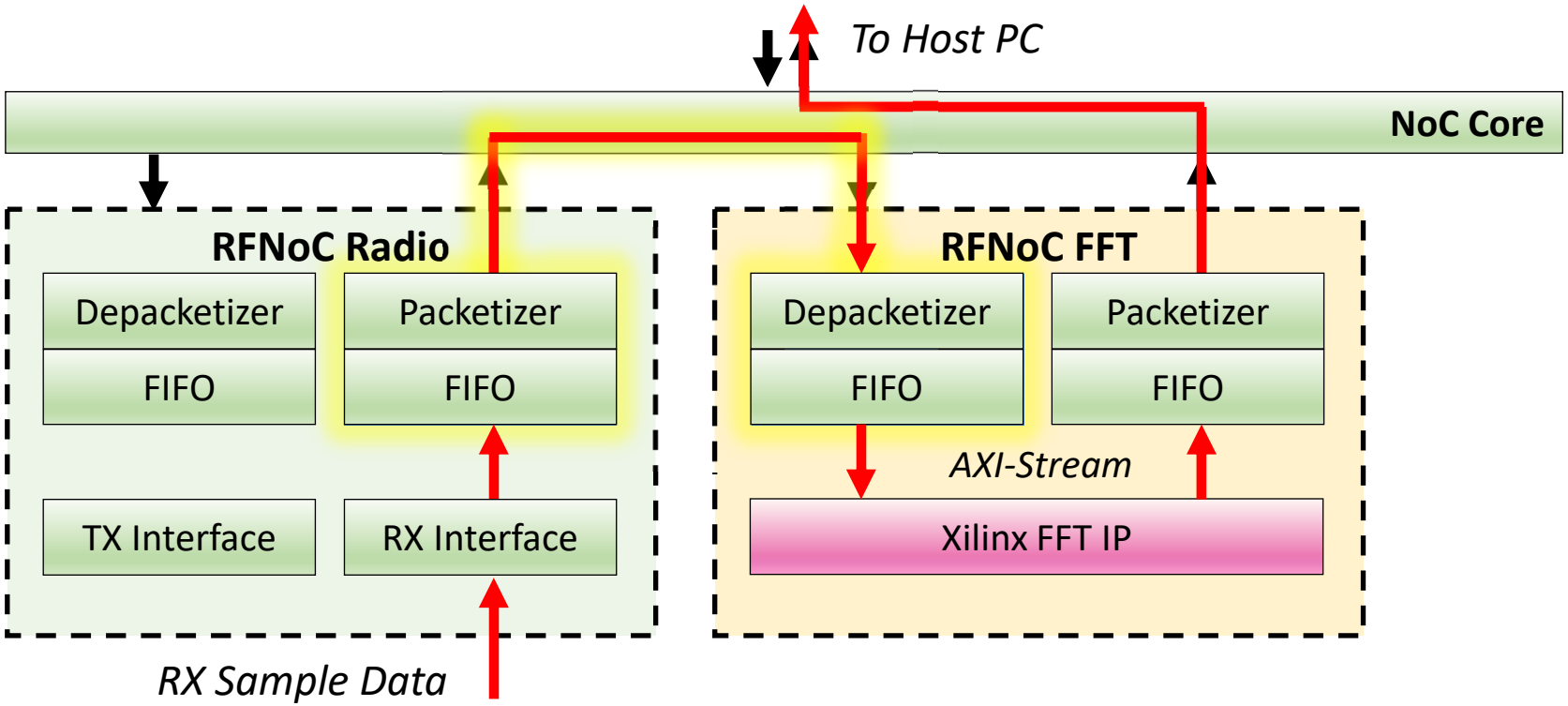
RFNoC FFT Block

RFNoC Block

RFNoC Block Overview

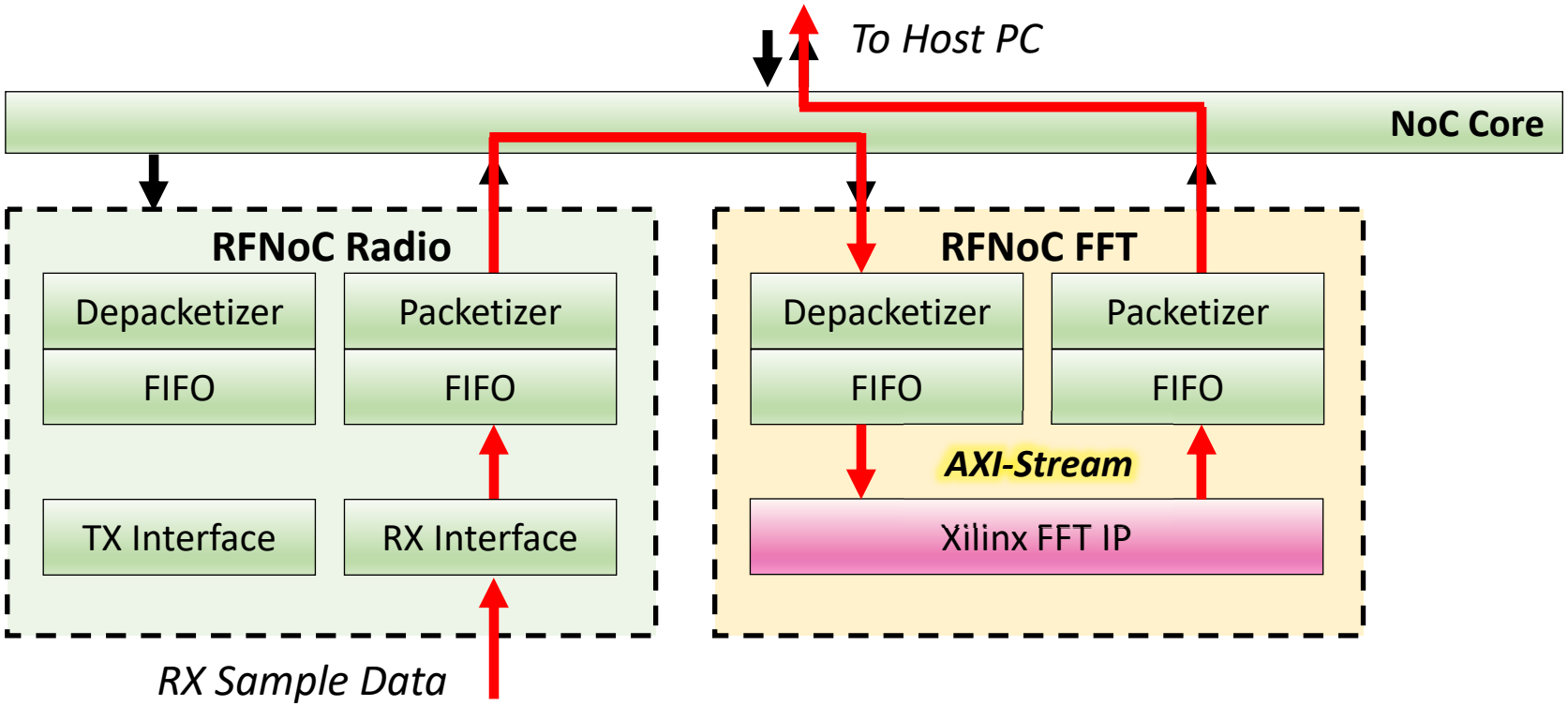


RFNoC Block Overview



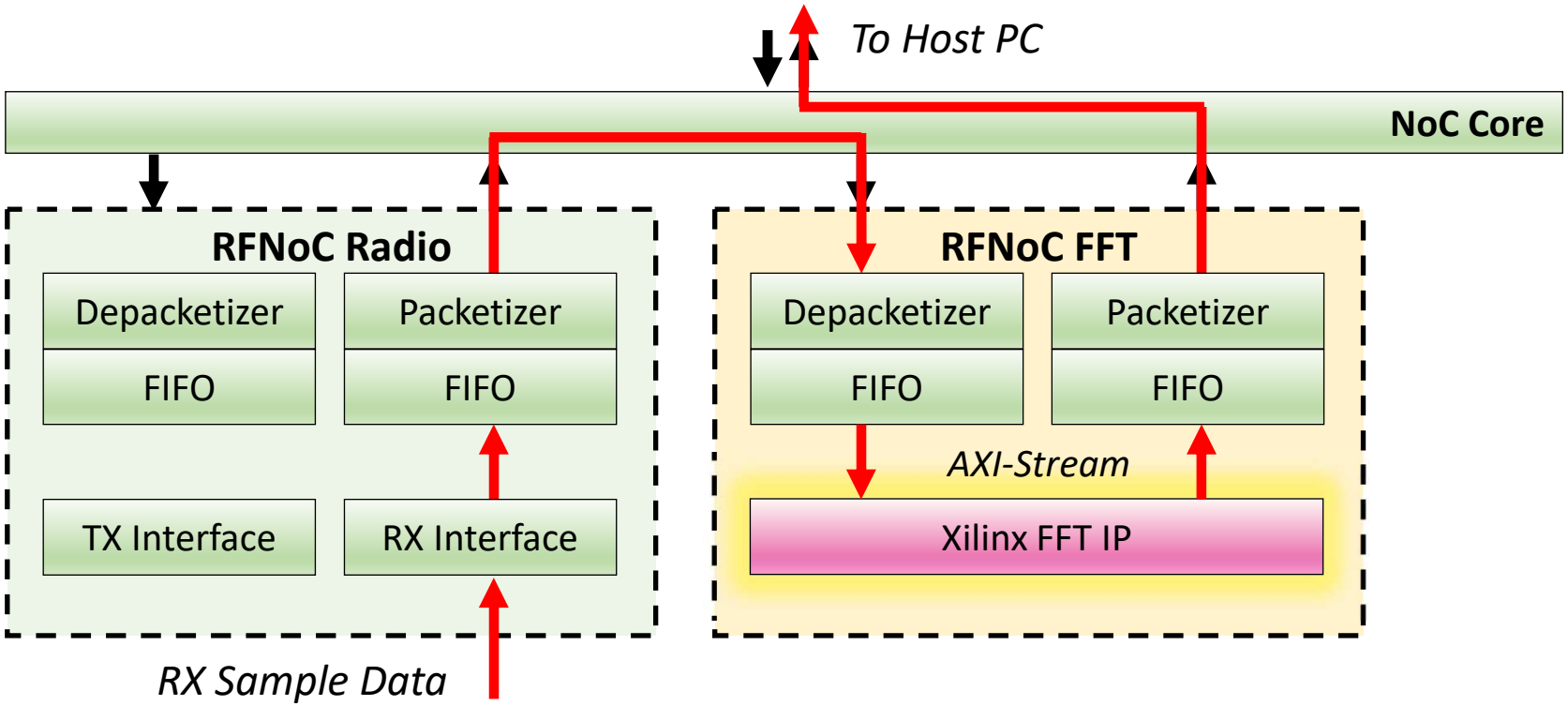
- Block to block communication:
  - FIFO to FIFO, packetized, flow control (unless static route)
  - Transparent to user – built into RFNoC infrastructure

RFNoC Block Overview



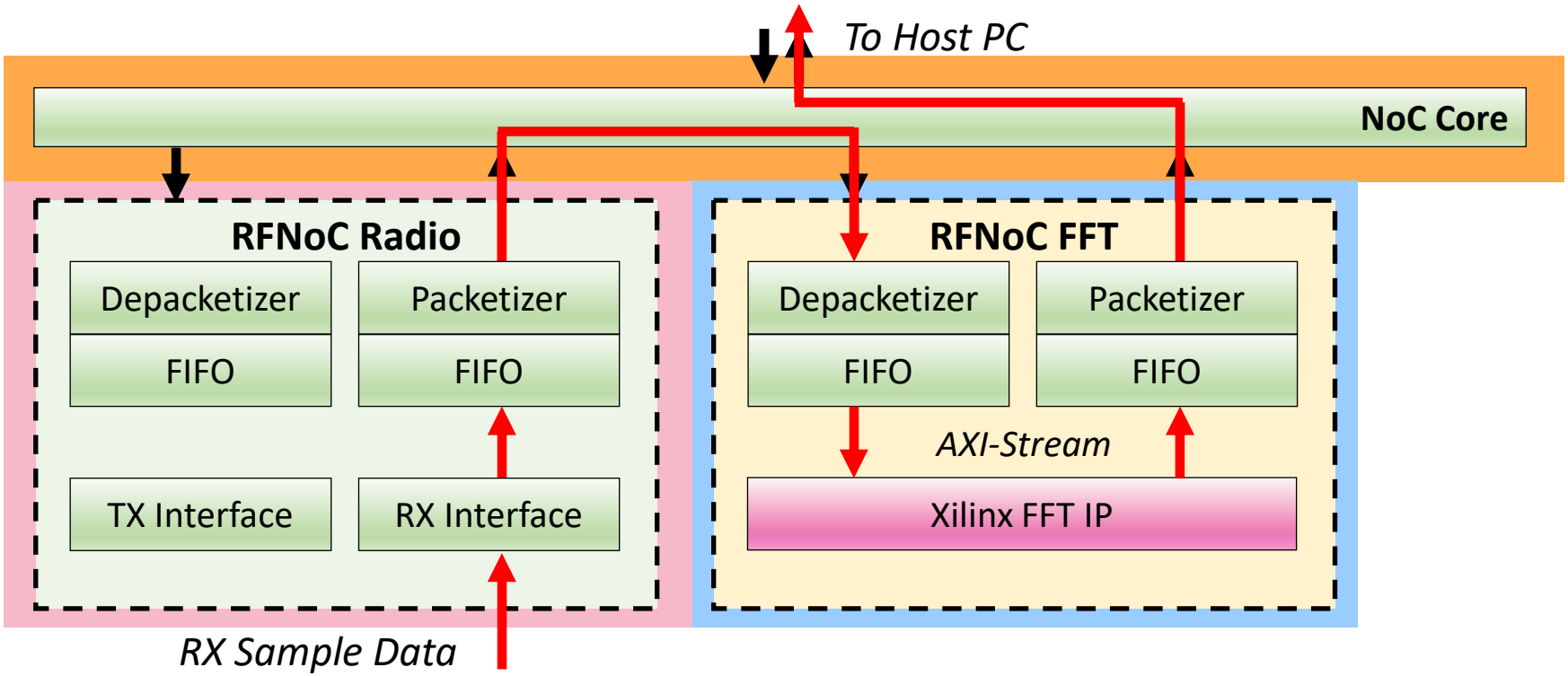
- User interfaces to RFNoC via AXI-Stream
  - Industry standard (ARM), easy to use
  - Large library of existing IP cores

RFNoC Block Overview



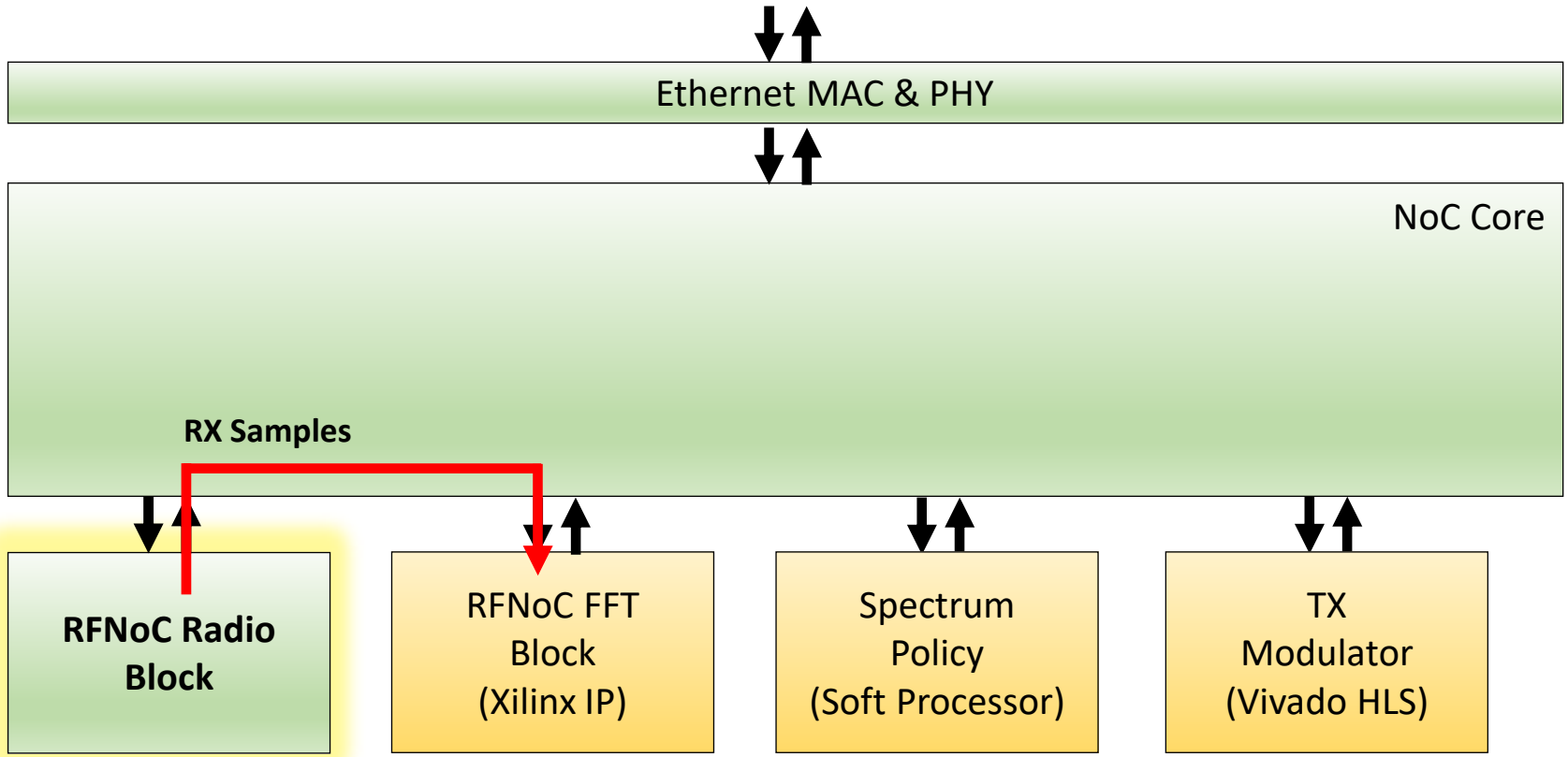
- User writes their own custom HDL or drops in IP
  - VHDL, Verilog, SystemVerilog, Vivado HLS
  - Xilinx IP, Vivado Block Diagram

RFNoC Block Overview



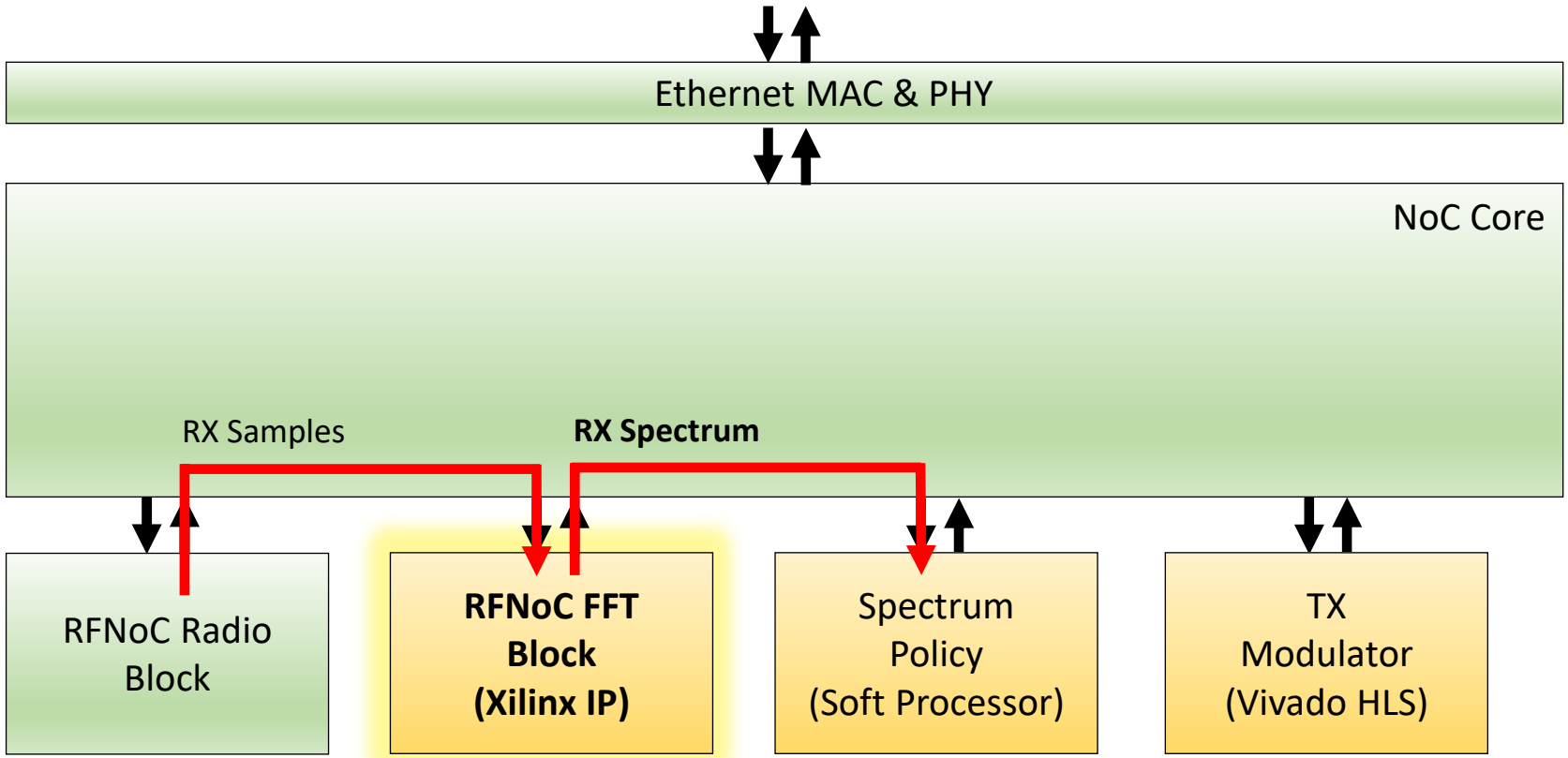
- Each block is in their own clock domain
  - Improves throughput
  - Easier timing closure

Cognitive Radio



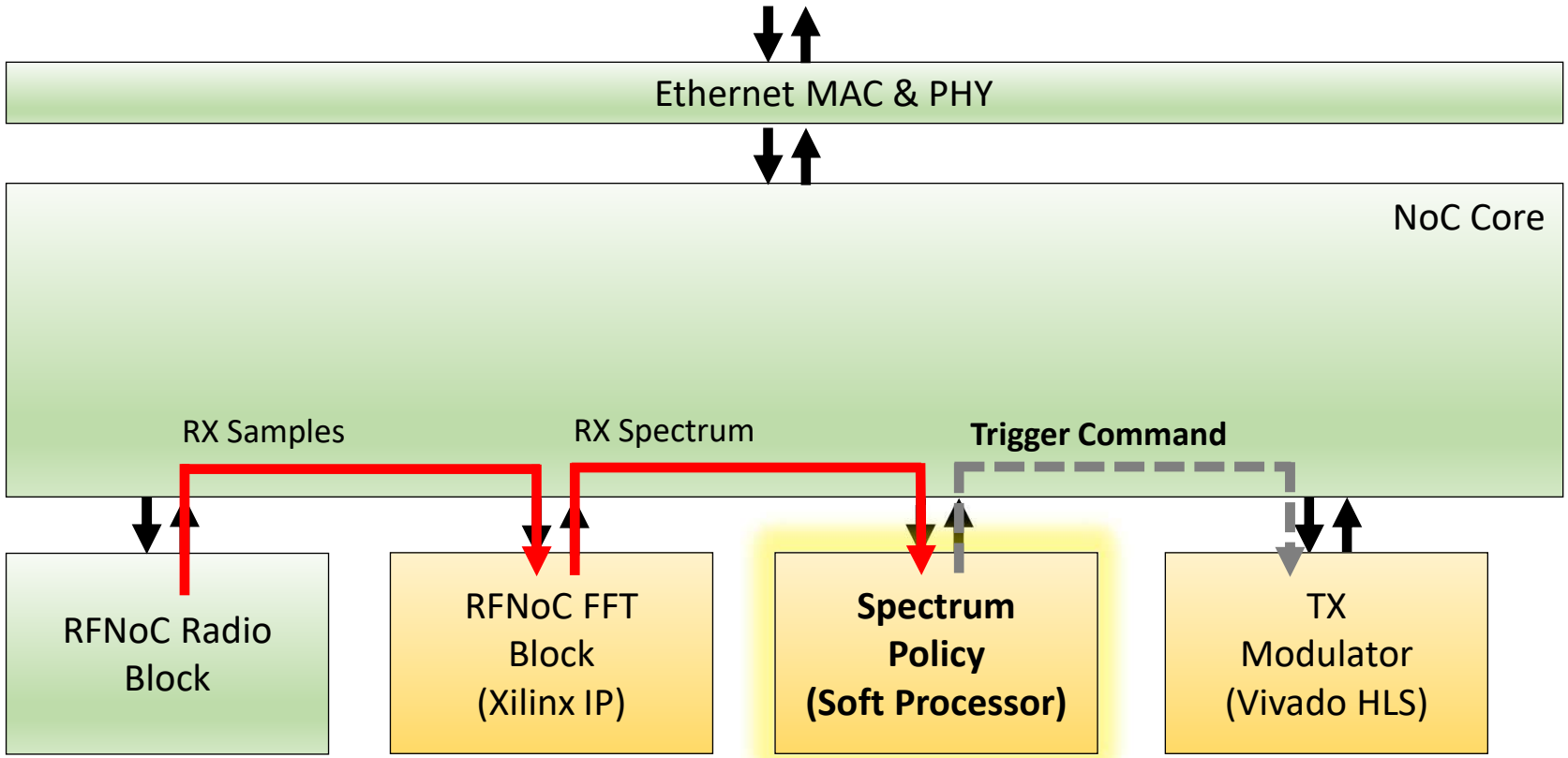
Cognitive Radio

Ettus Research™

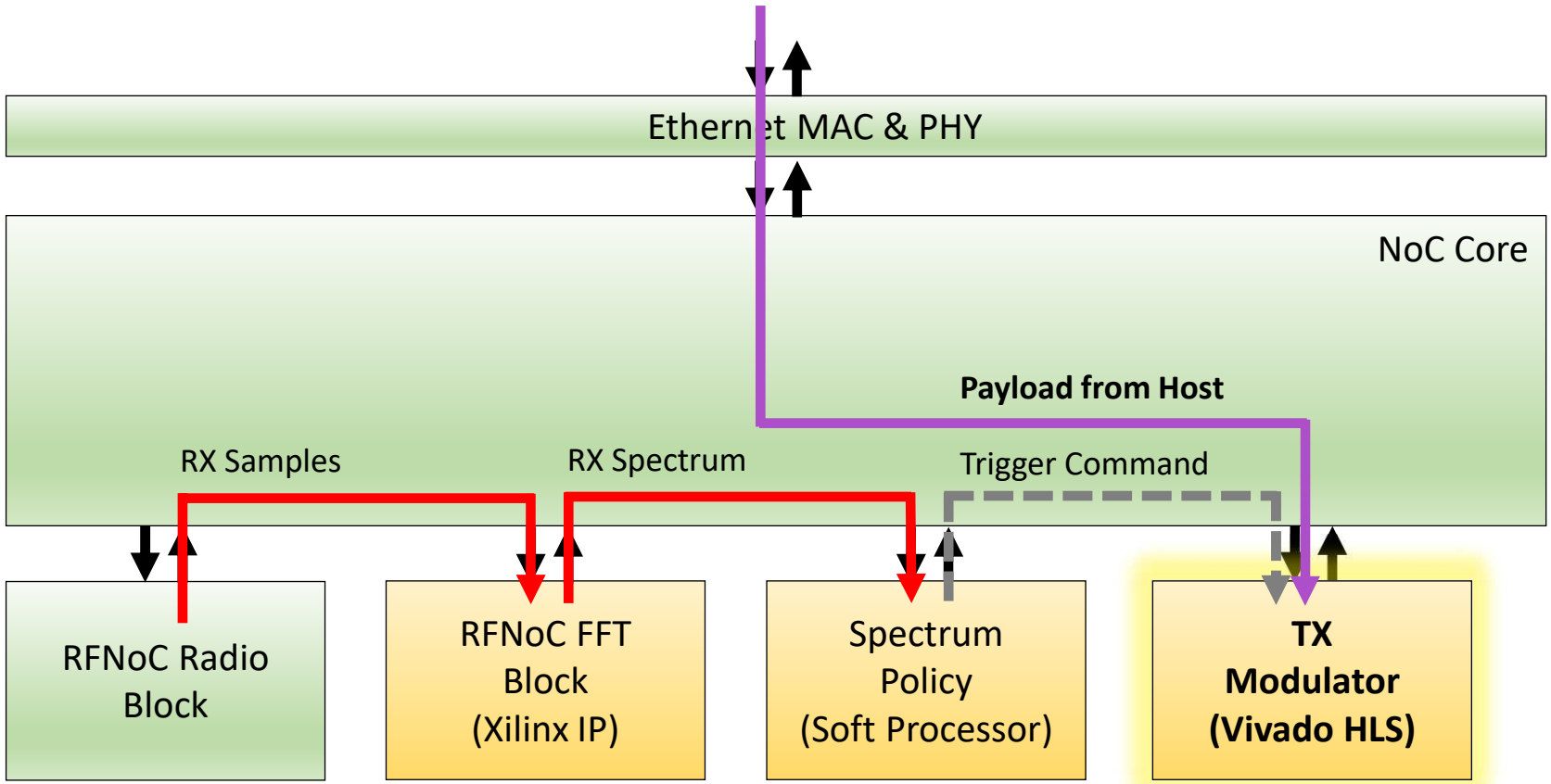


Cognitive Radio

Ettus Research™

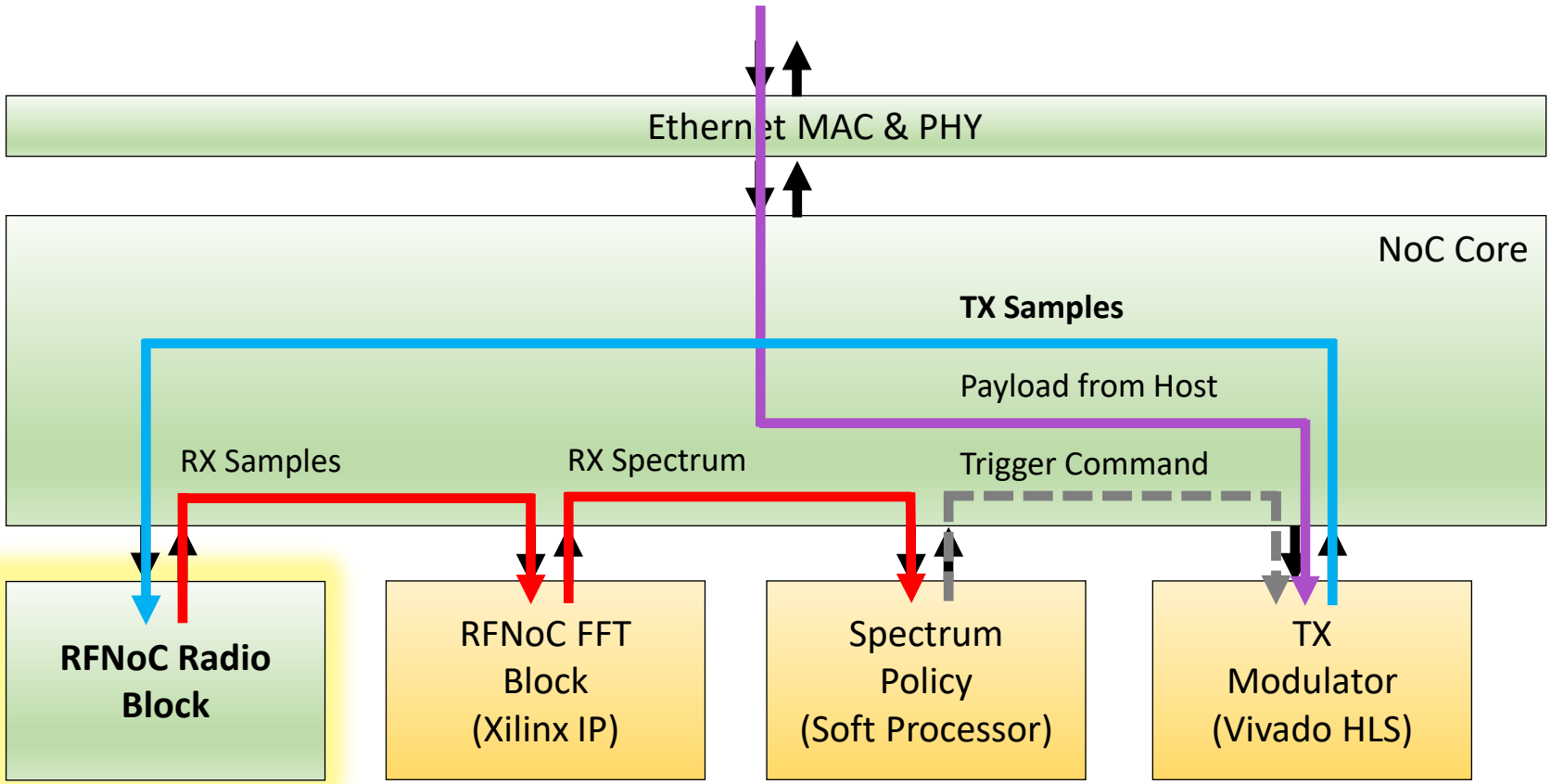


Cognitive Radio

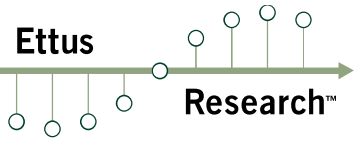


Cognitive Radio

Ettus Research™

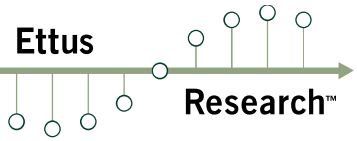


Summary



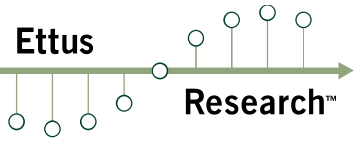
- Make FPGA acceleration more accessible on USRPs
- Tightly integrated with GNU Radio
- Library of existing RFNoC Blocks
  - FFT, FIR, Signal Generator, Fosphor
- Portable between all third generation USRPs
  - X3x0, E3xx, N3xx
- Completely open source
- kb.ettus.com/RFNoC_Getting_Started_Guides
- Next: FPGA & Software Development

What's New in RFNoC 4



- Better Documentation
  - Spec: http://files.ettus.com/app_notes/RFNoC_Specification.pdf
- Software Enhancements
  - Stability and Testing
  - Python support for RFNoC
- FPGA Improvements
  - Scalable to faster sampling rates (250 MSPS+)
  - Instantiate far more RFNoC Blocks
  - Static routing between RFNoC blocks
    - Trade off latency and resource utilization versus flexibility
- GNU Radio 3.8 Support

RFNoC 4 Demo - Fosphor



- Fosphor is a real-time GPU-accelerated or FPGA-accelerated spectrum display tool
- Running on a USRP X310 with a WBX daughterboard
- The system is running Ubuntu 20.04 with GNU Radio 3.8.2.0
- All calculations for the FFT and waterfall are being done on the FPGA, not on the CPU
- The CPU is minimally loaded, even for large bandwidths