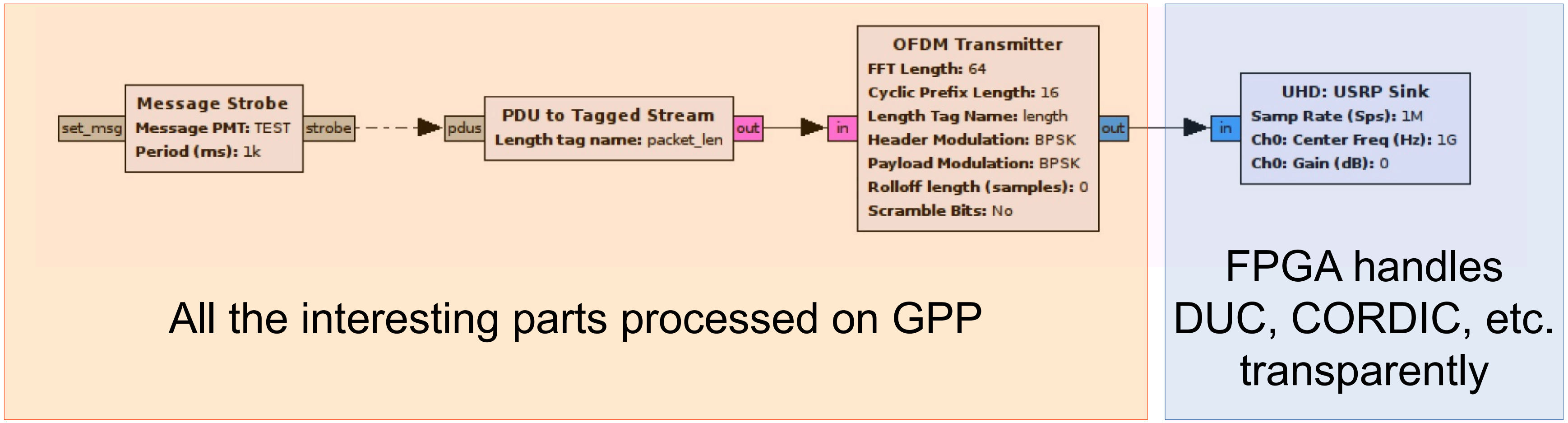
RFNoC™: Evolving SDR Toolkits to the FPGA platform

Martin Braun

31.1.2016

USRP: A White Box?

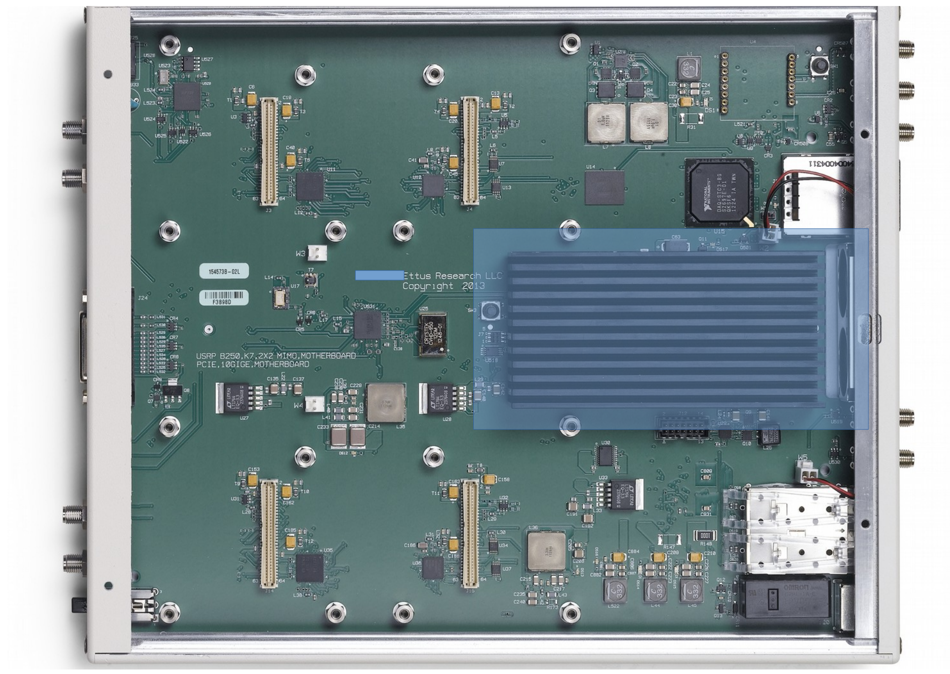
- Simple OFDM Transmitter Development:



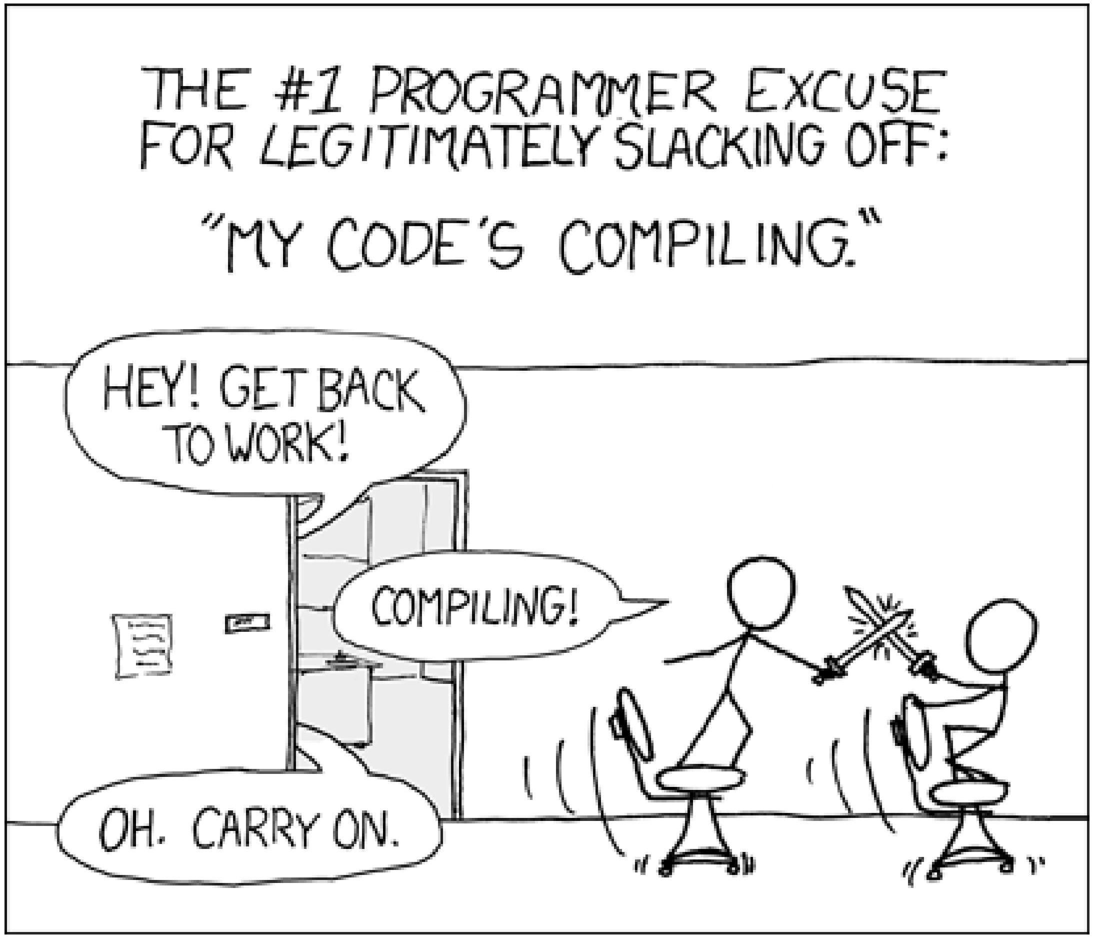
- Entire Hardware stack is treated like a reprogrammable ASIC, Features are used as-is

Open the Box!

- Everything USRP is available online (code, schematics)
- Contains big and expensive FPGA!

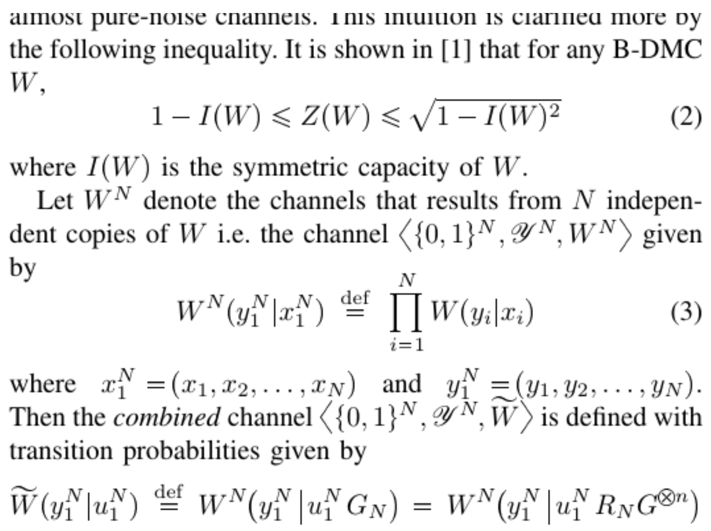


FPGAs: Hard to use... slow to develop



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,

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

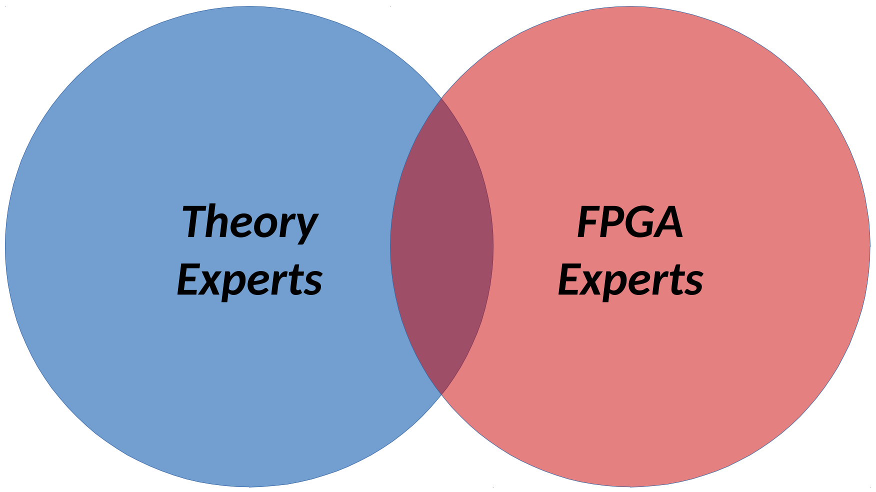
where $I(W)$ is the symmetric capacity of W.

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

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

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

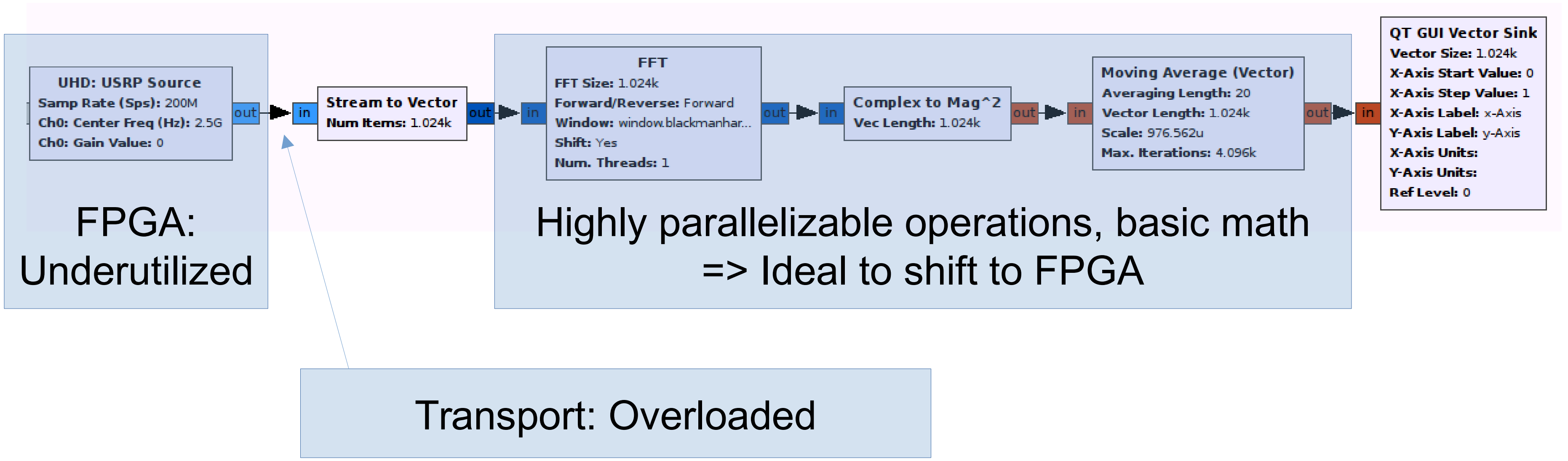
$$\widetilde{W}(y_1^N | u_1^N) \overset{\text{def}}{=} W^N(y_1^N | u_1^N G_N) = W^N(y_1^N | u_1^N R_N G^{\otimes n})$$



Example: Wideband Spectral Analysis

Ettus Research™
A National Instruments Company

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



UHD: USRP Source
Samp Rate (Sps): 200M
Ch0: Center Freq (Hz): 2.5G
Ch0: Gain Value: 0

Stream to Vector
Num Items: 1.024k

FFT
FFT Size: 1.024k
Forward/Reverse: Forward
Window: window.blackmanhar...
Shift: Yes
Num. Threads: 1

Complex to Mag^2
Vec Length: 1.024k

Moving Average (Vector)
Averaging Length: 20
Vector Length: 1.024k
Scale: 976.562u
Max. Iterations: 4.096k

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

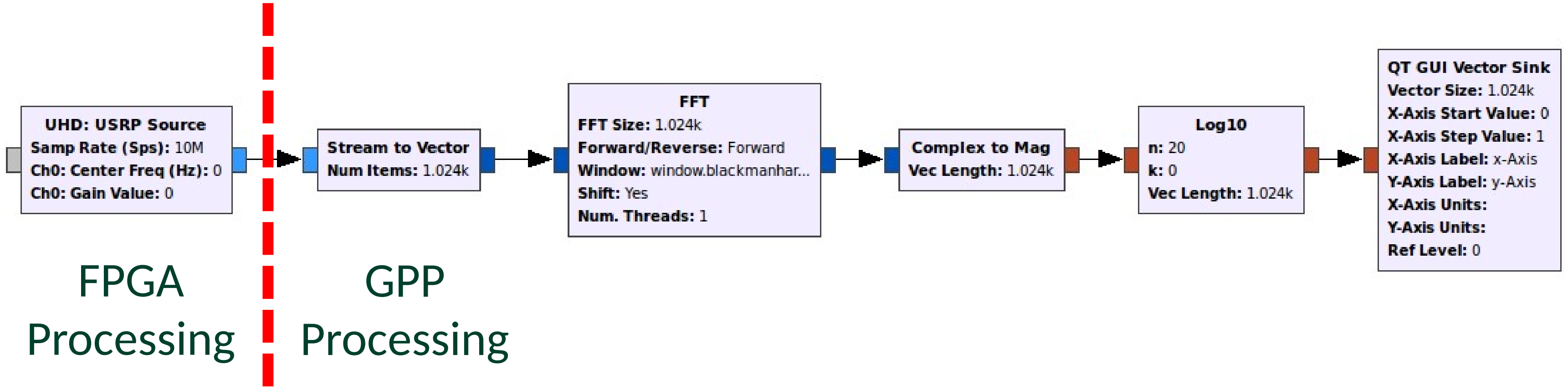
FPGA: Underutilized

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

Transport: Overloaded

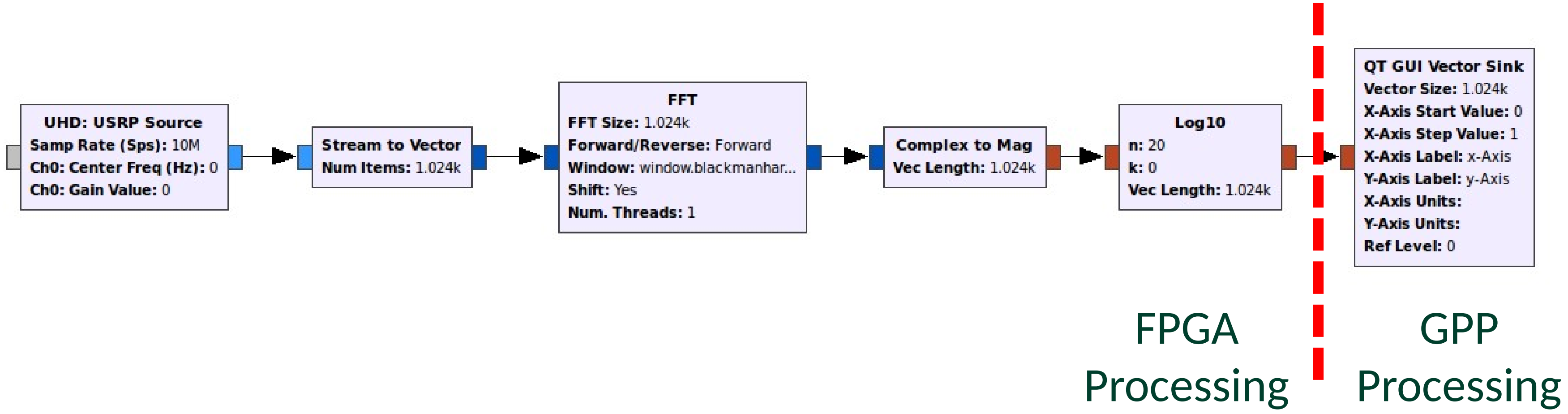
Goal

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



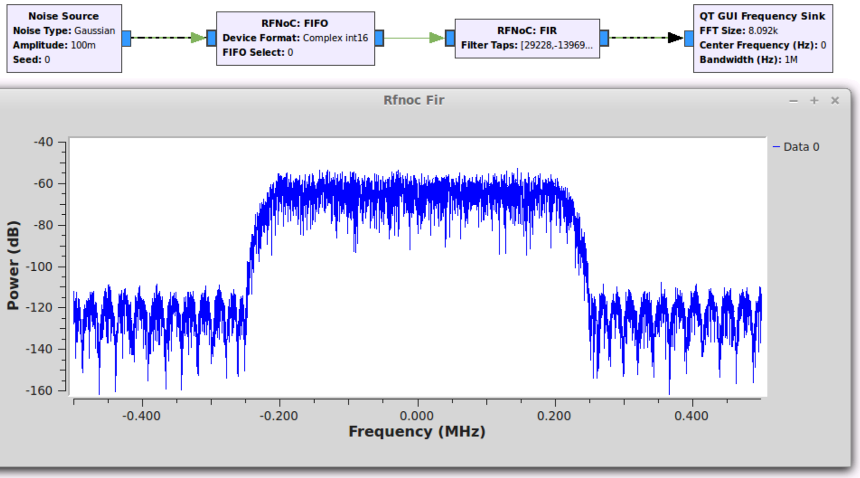
Goal

- 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



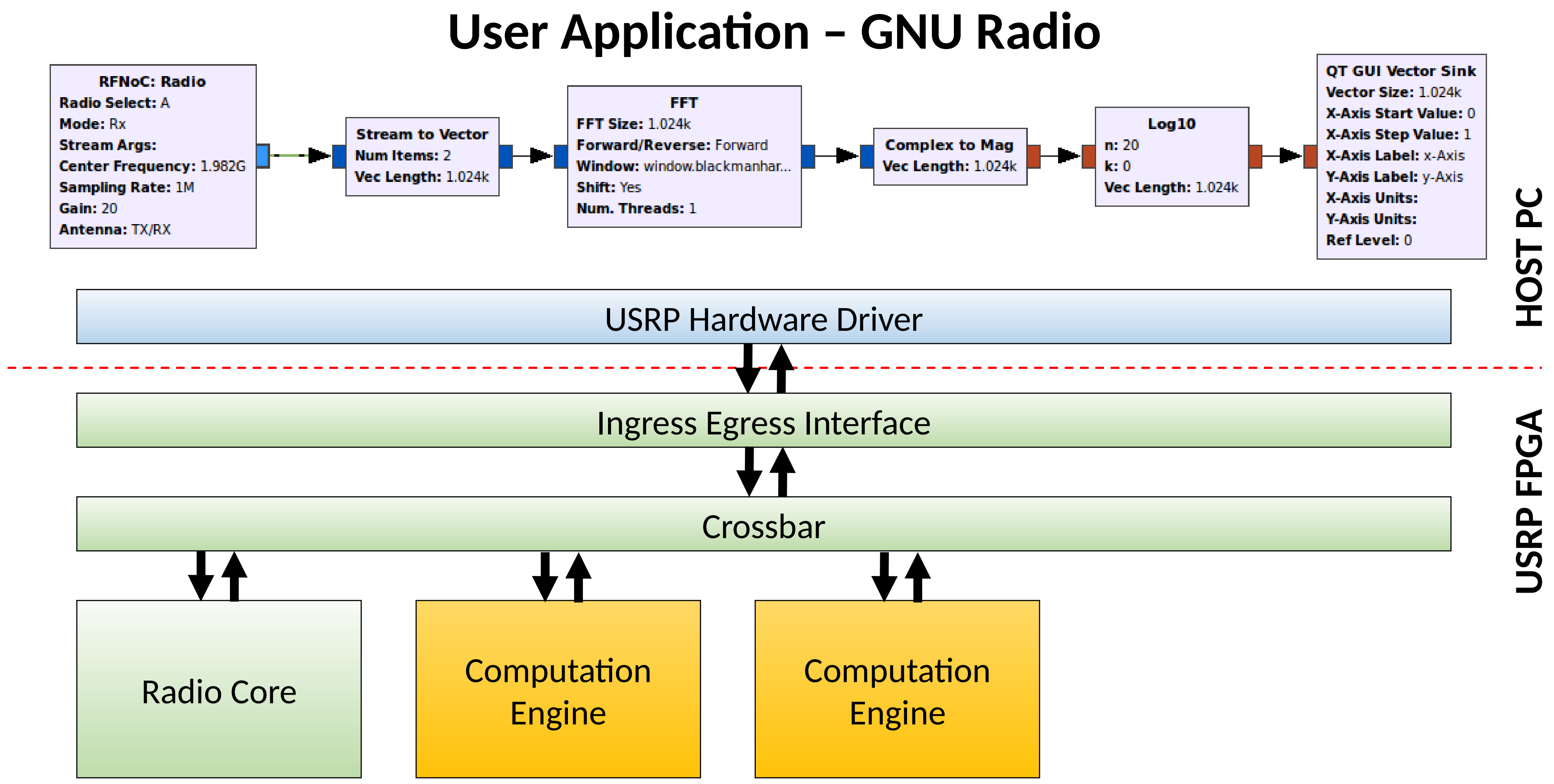
Noise Source
Noise Type: Gaussian
Amplitude: 100m
Seed: 0

RFNoC: FIFO
Device Format: Complex int16
FIFO Select: 0

RFNoC: FIR
Filter Taps: [29228,-13969...

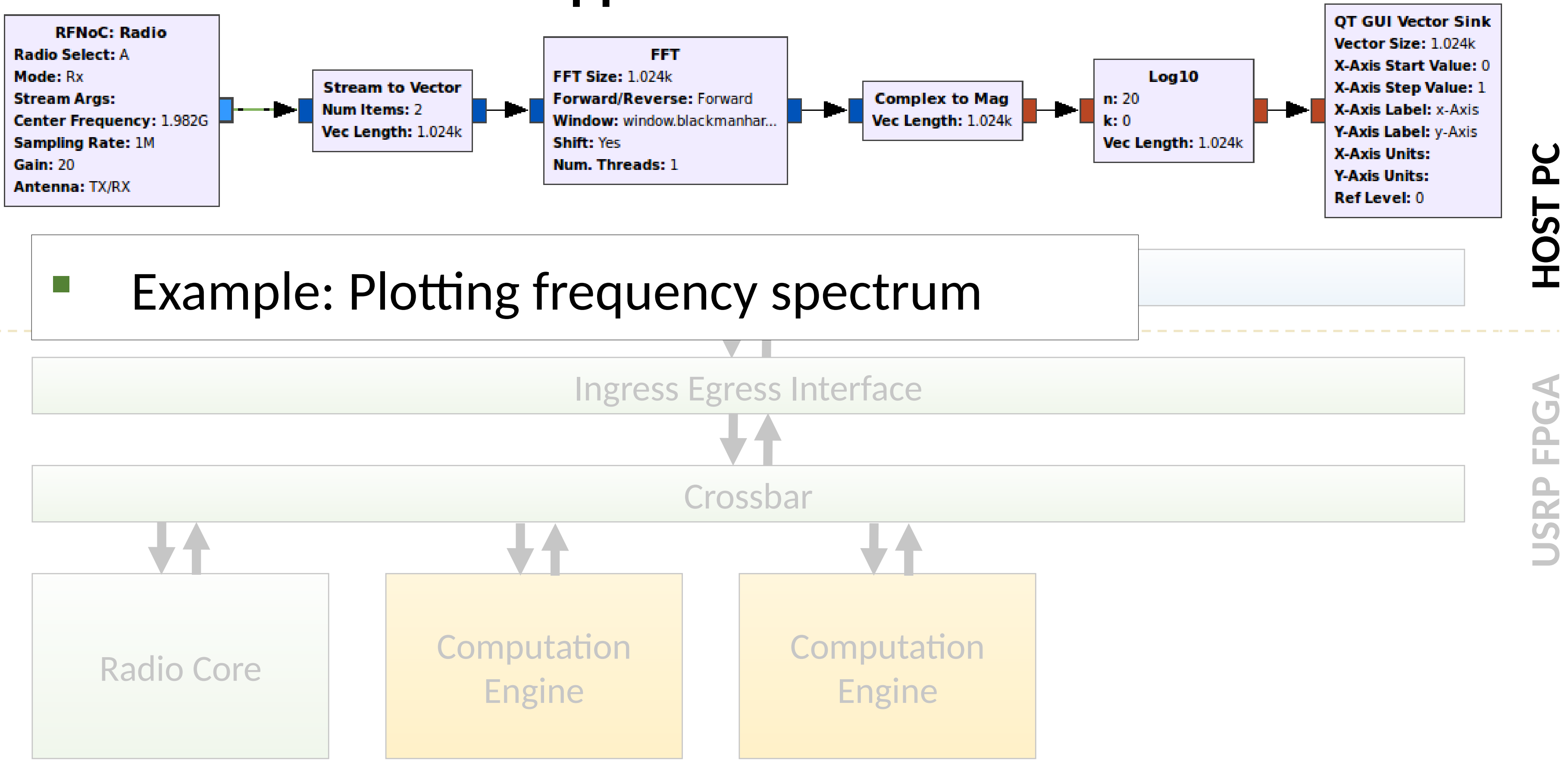
QT GUI Frequency Sink
FFT Size: 8.092k
Center Frequency (Hz): 0
Bandwidth (Hz): 1M

RFNoC Architecture



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

Stream to Vector
Num Items: 2
Vec Length: 1.024k

FFT
FFT Size: 1.024k
Forward/Reverse: Forward
Window: window.blackmanhar...
Shift: Yes
Num. Threads: 1

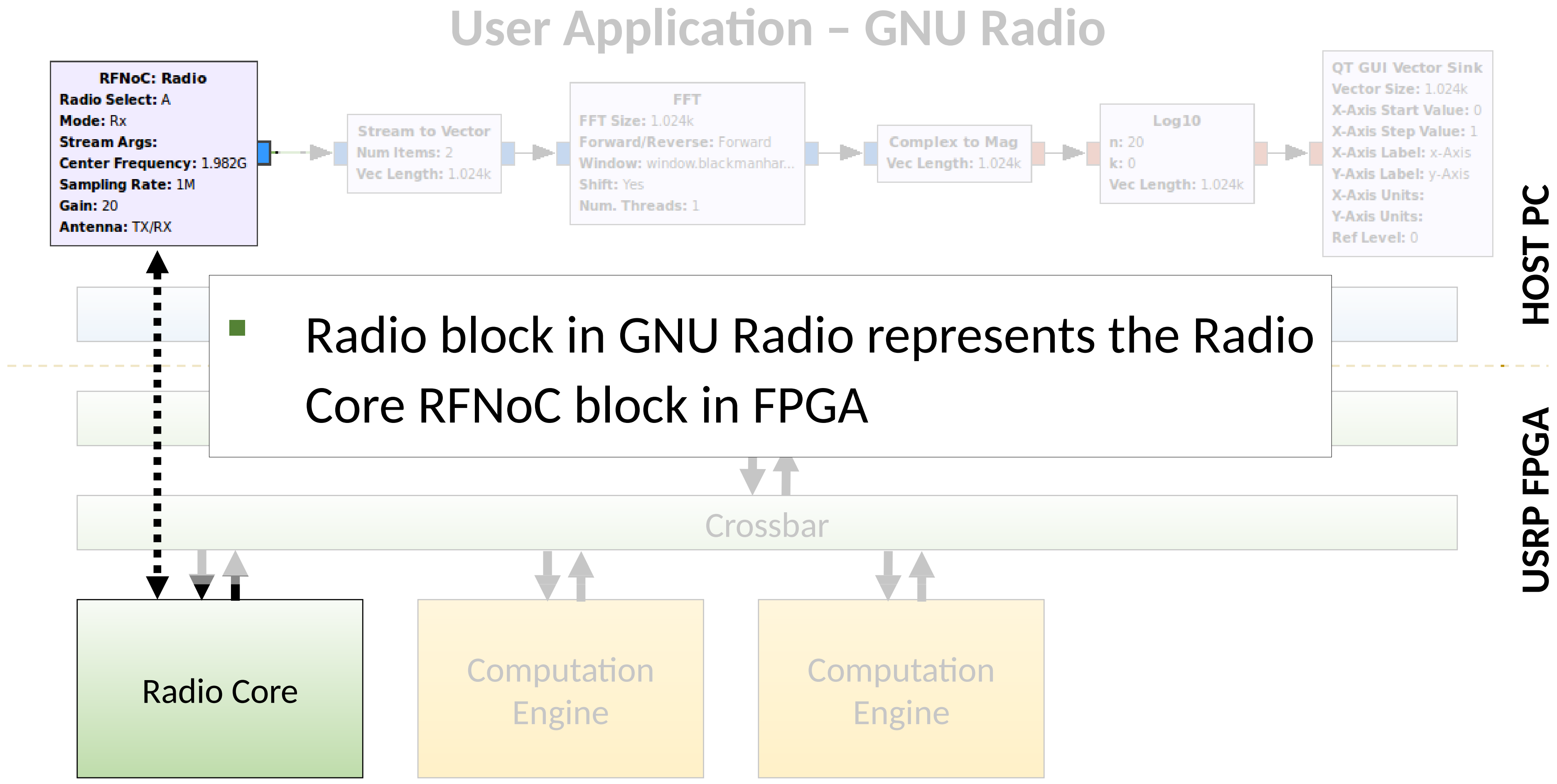
Complex to Mag
Vec Length: 1.024k

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

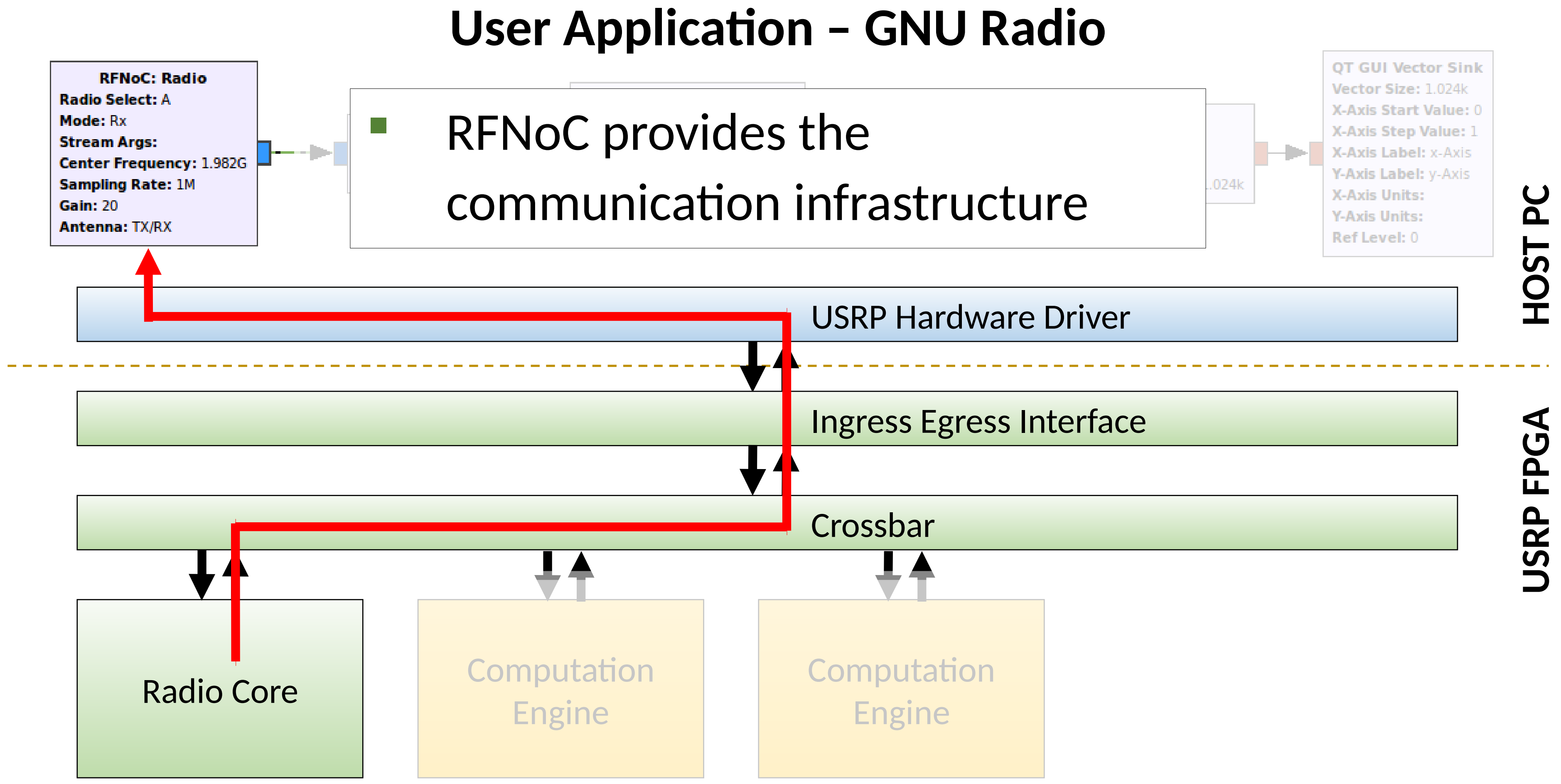
- Example: Plotting frequency spectrum

RFNoC Architecture



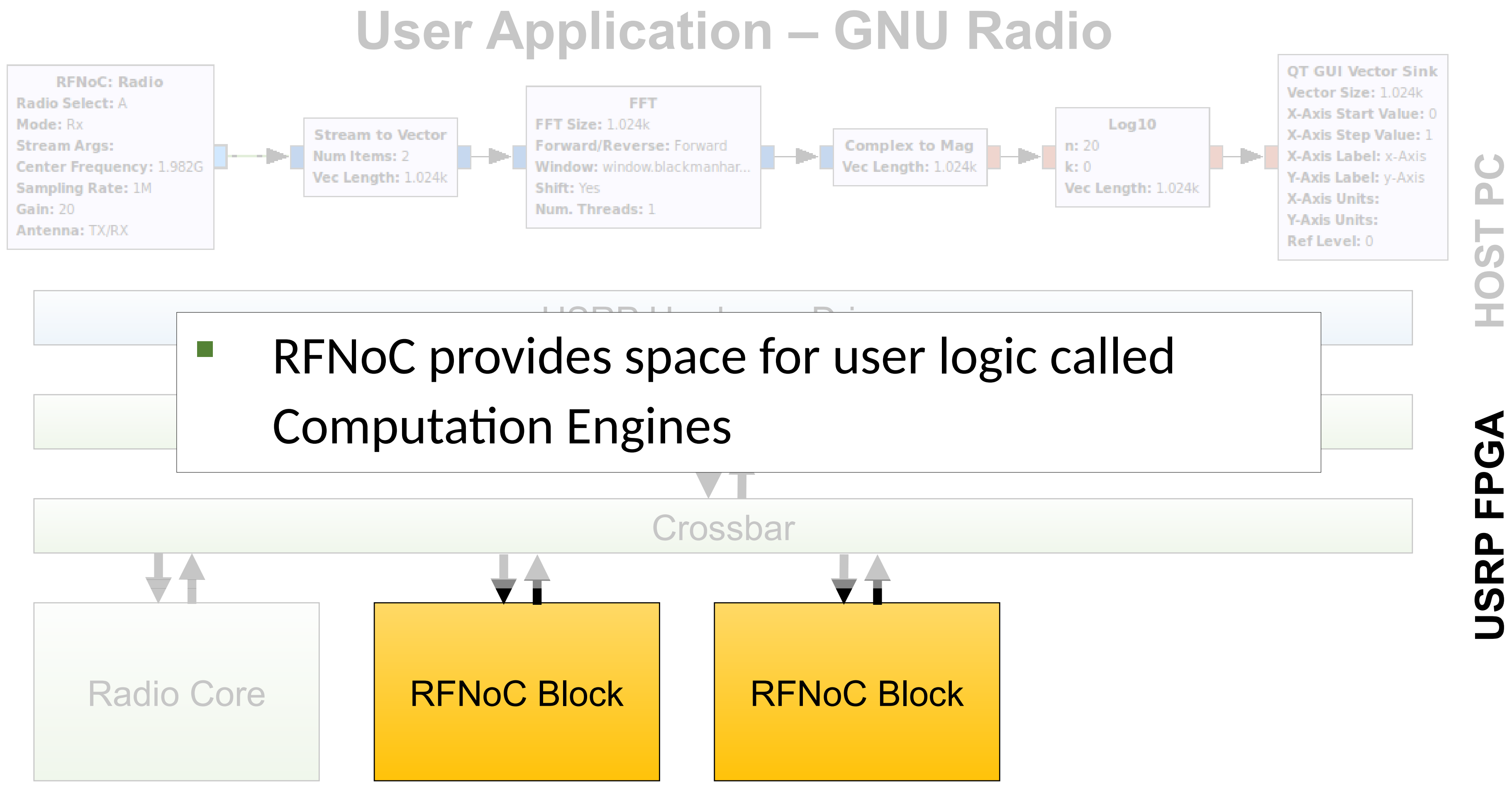
- Radio block in GNU Radio represents the Radio Core RFNoC block in FPGA

RFNoC Architecture



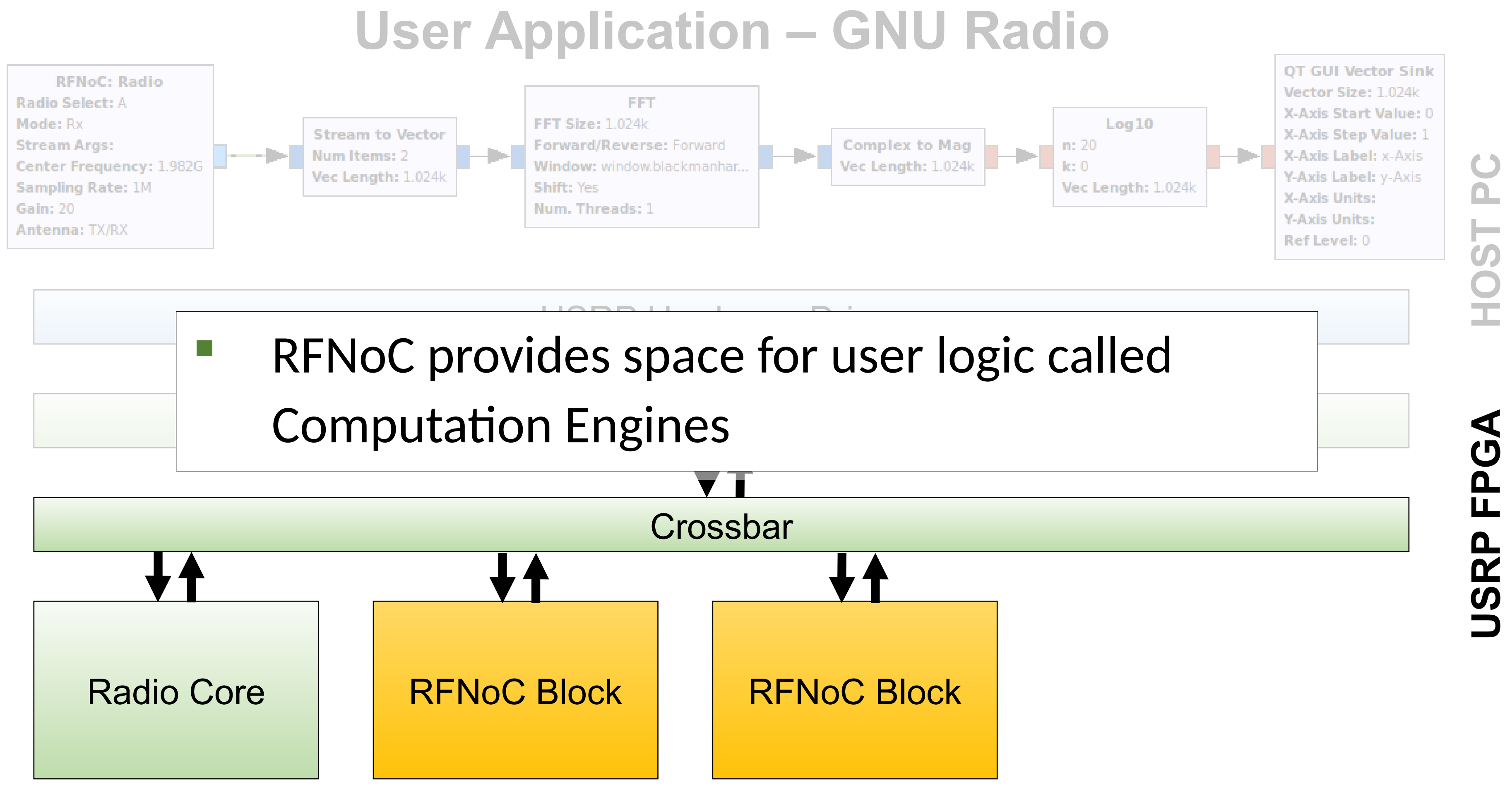
- RFNoC provides the communication infrastructure

RFNoC Architecture



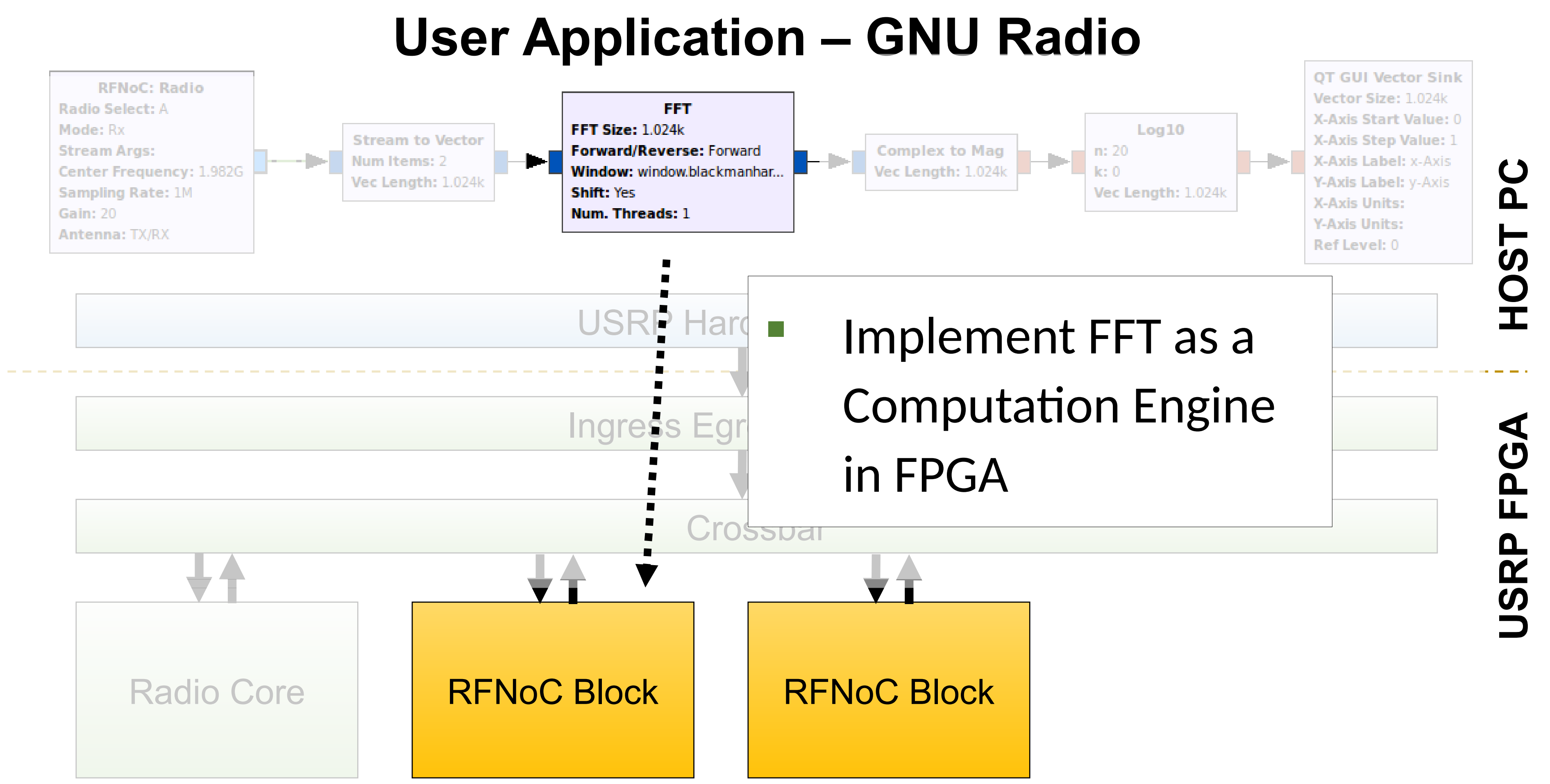
- RFNoC provides space for user logic called Computation Engines

RFNoC Architecture



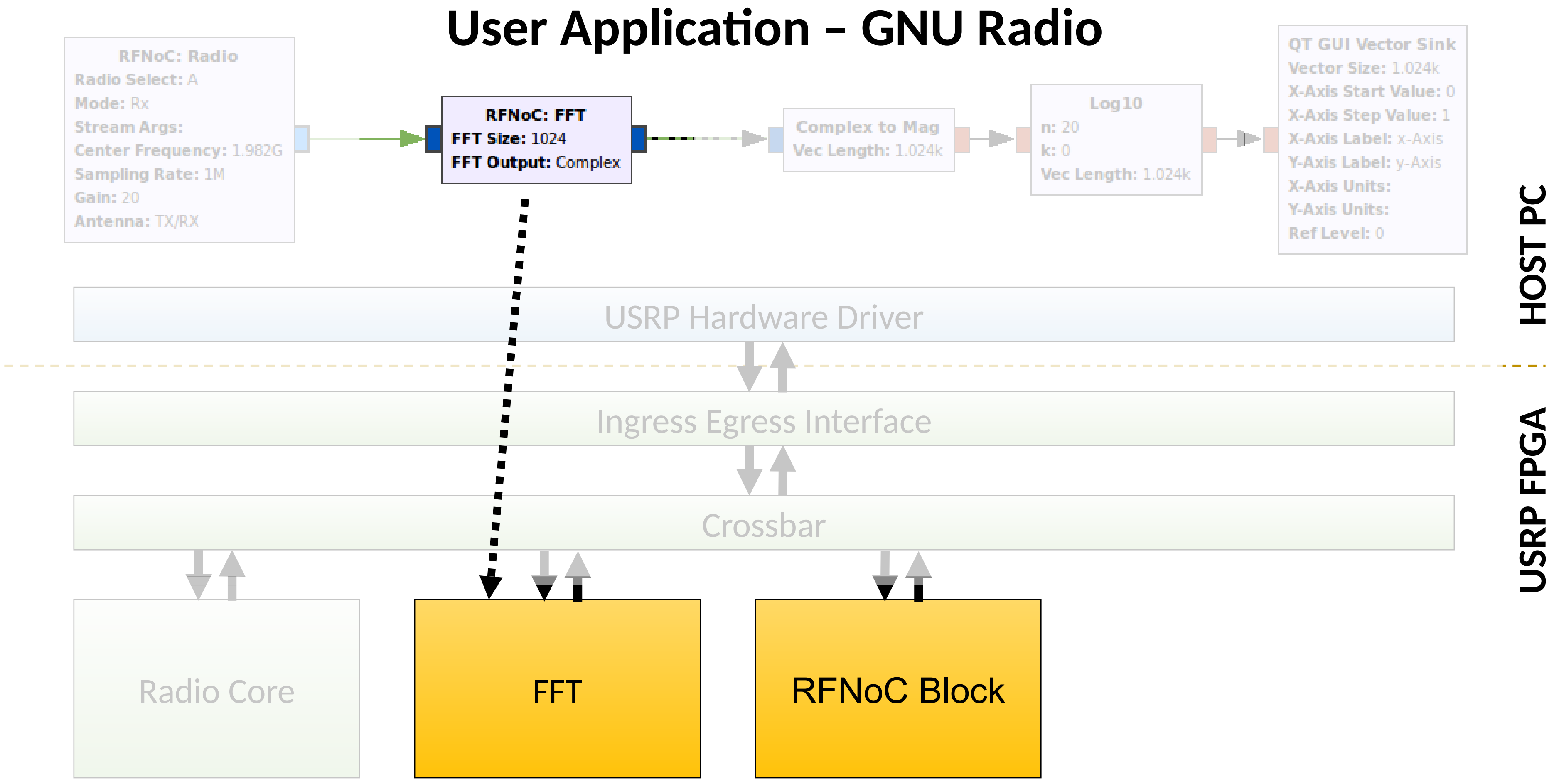
- RFNoC provides space for user logic called Computation Engines

RFNoC Architecture

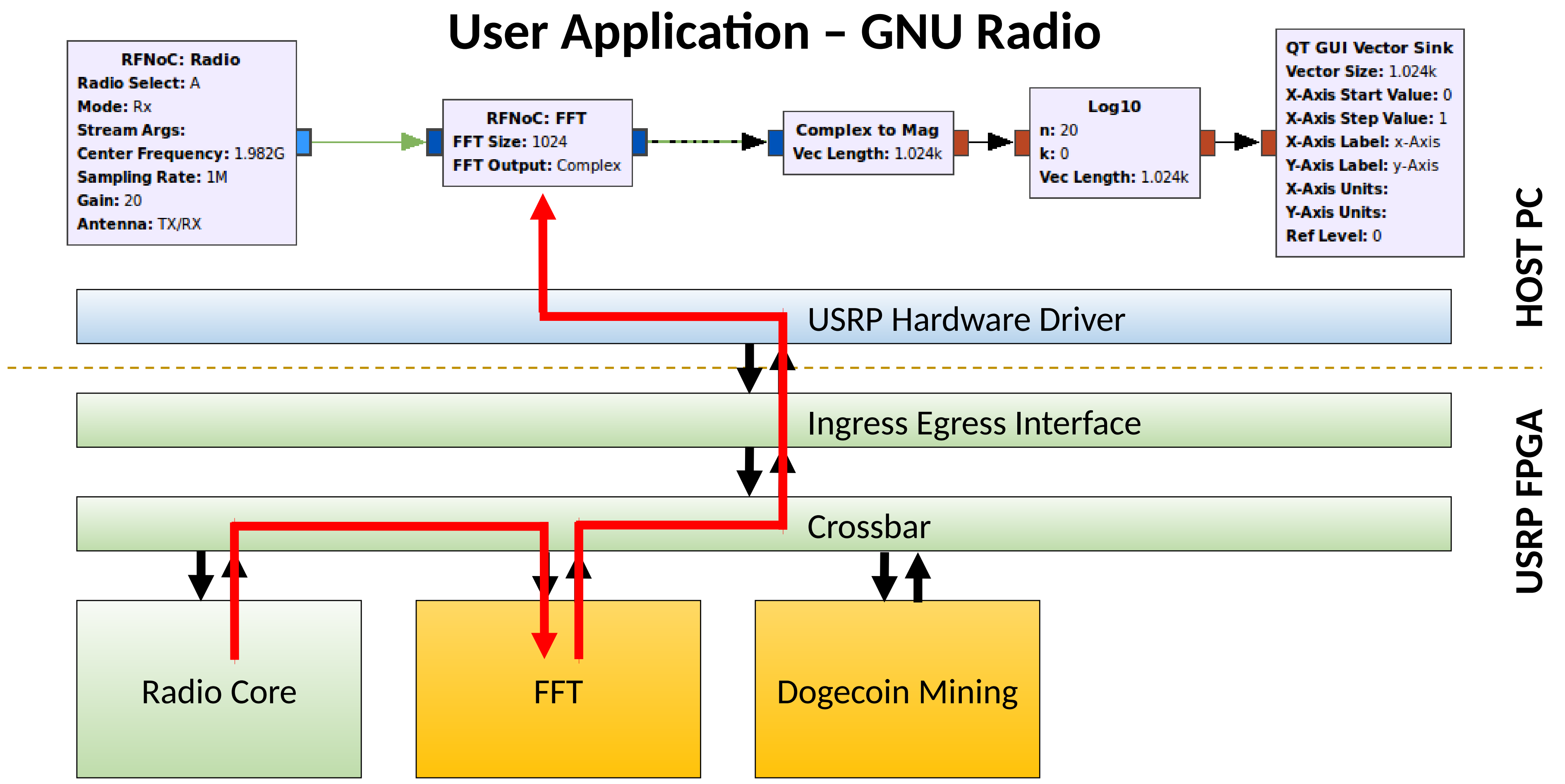


- Implement FFT as a Computation Engine in FPGA

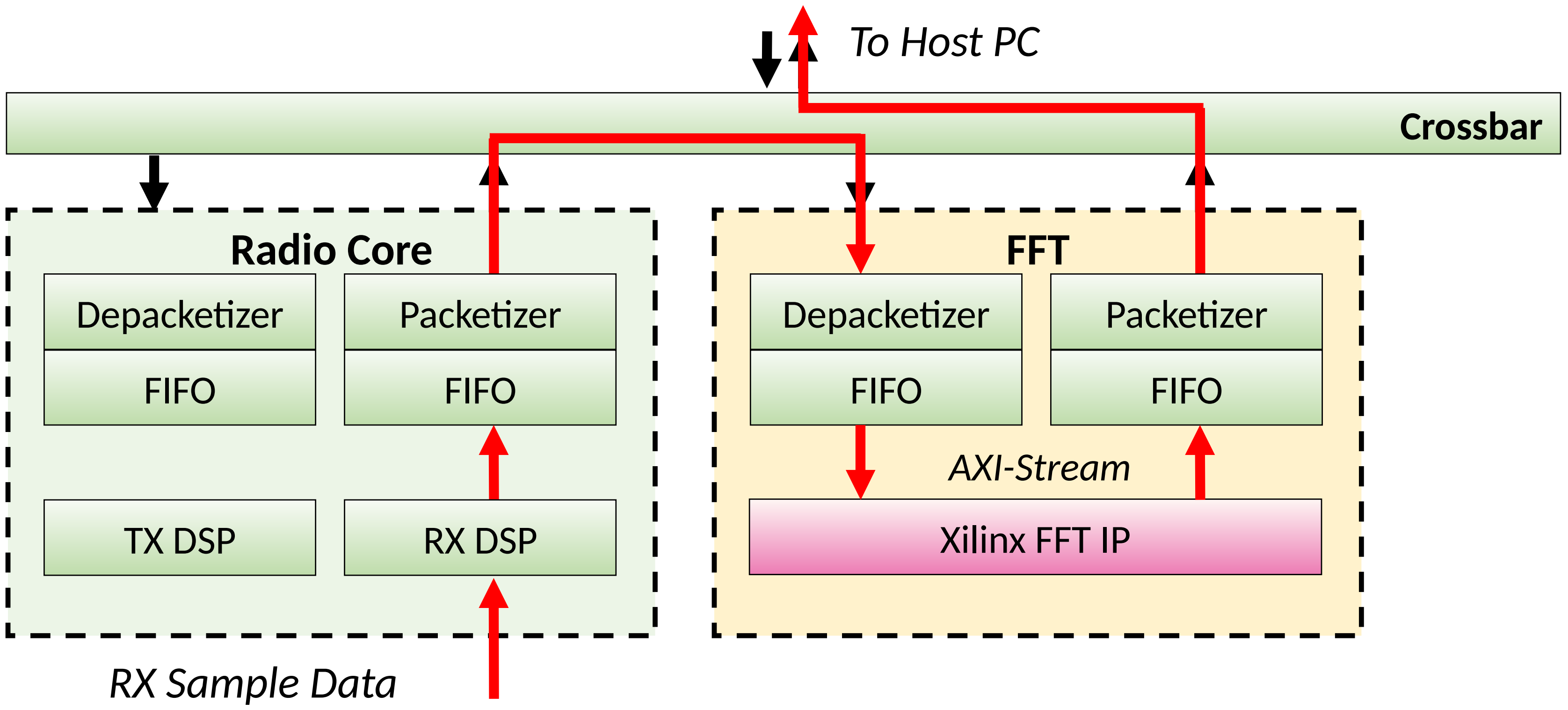
RFNoC Architecture



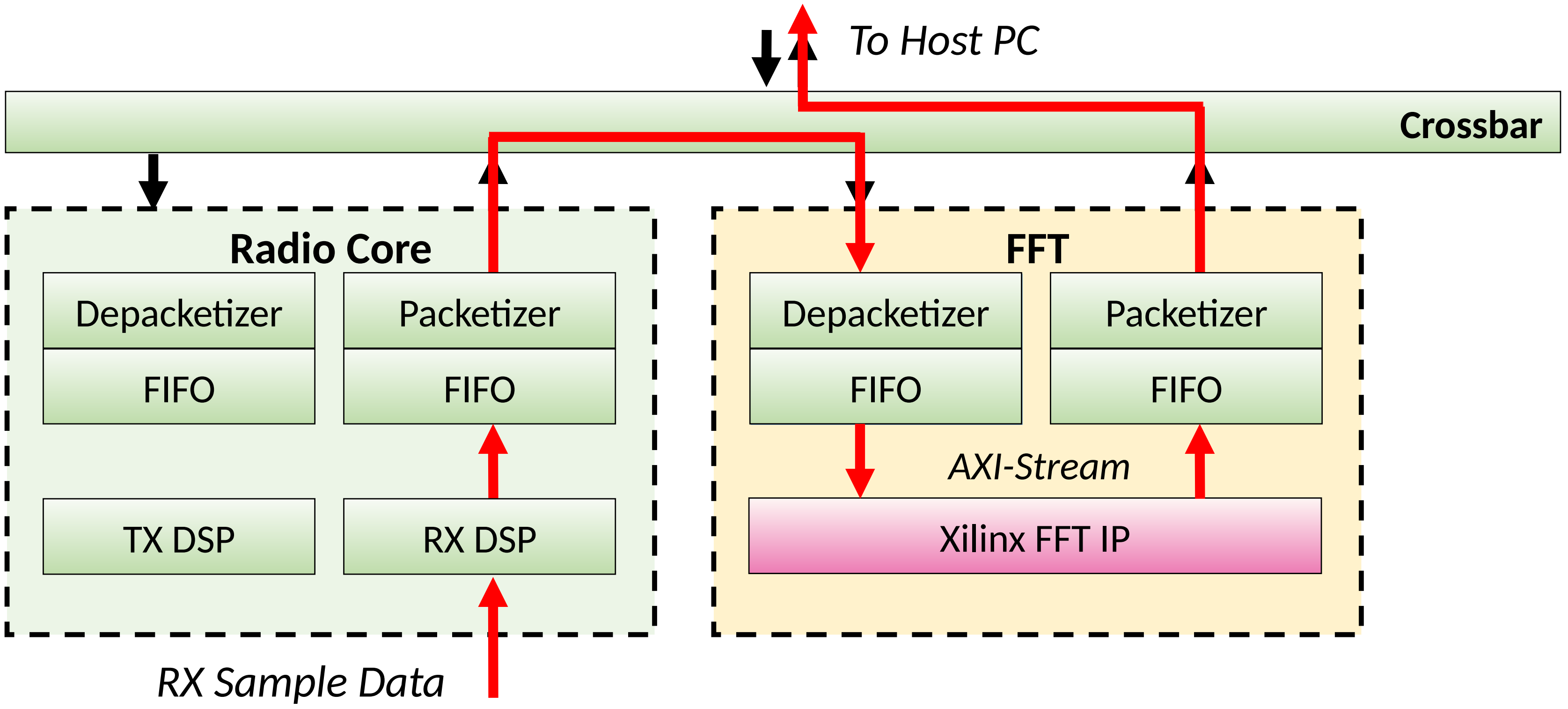
RFNoC Architecture



Computation Engine

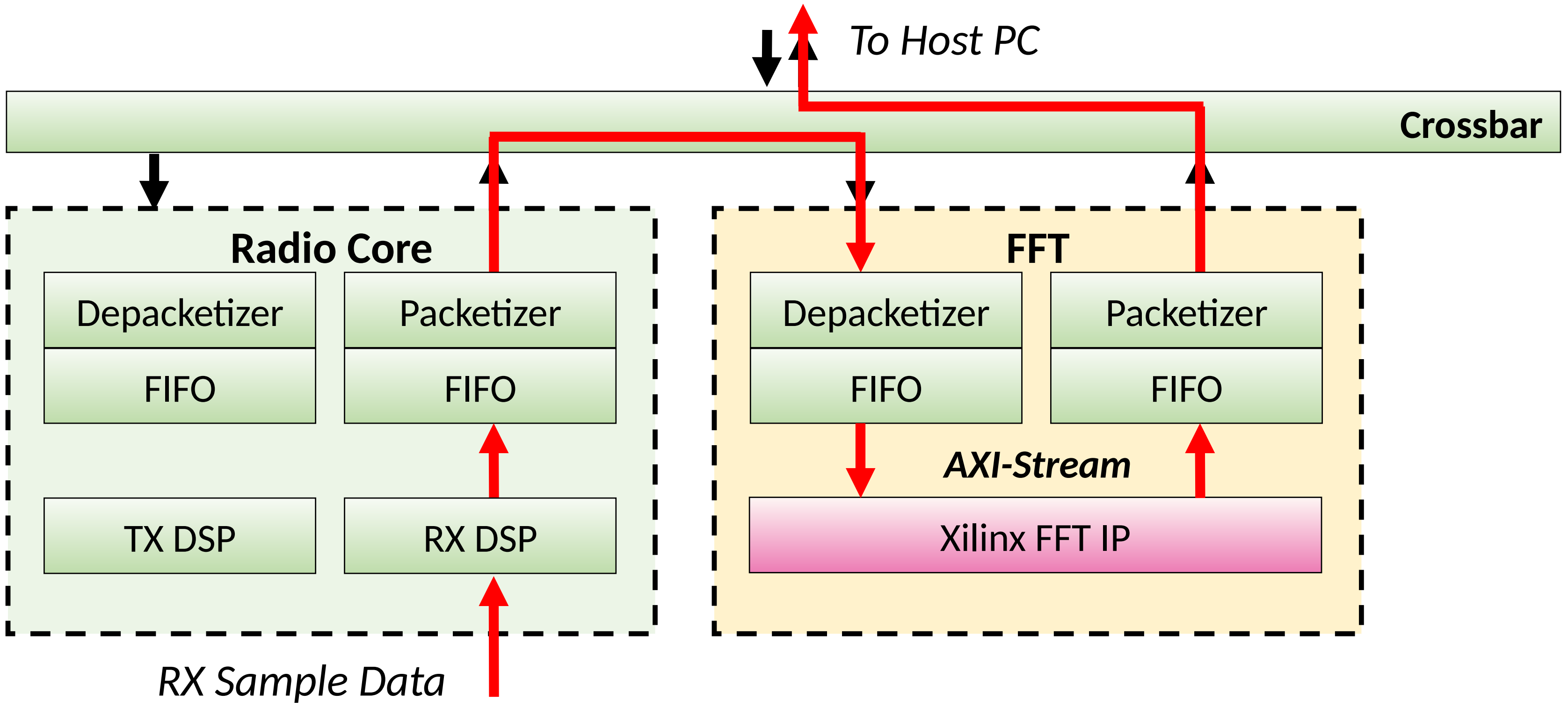


Computation Engine



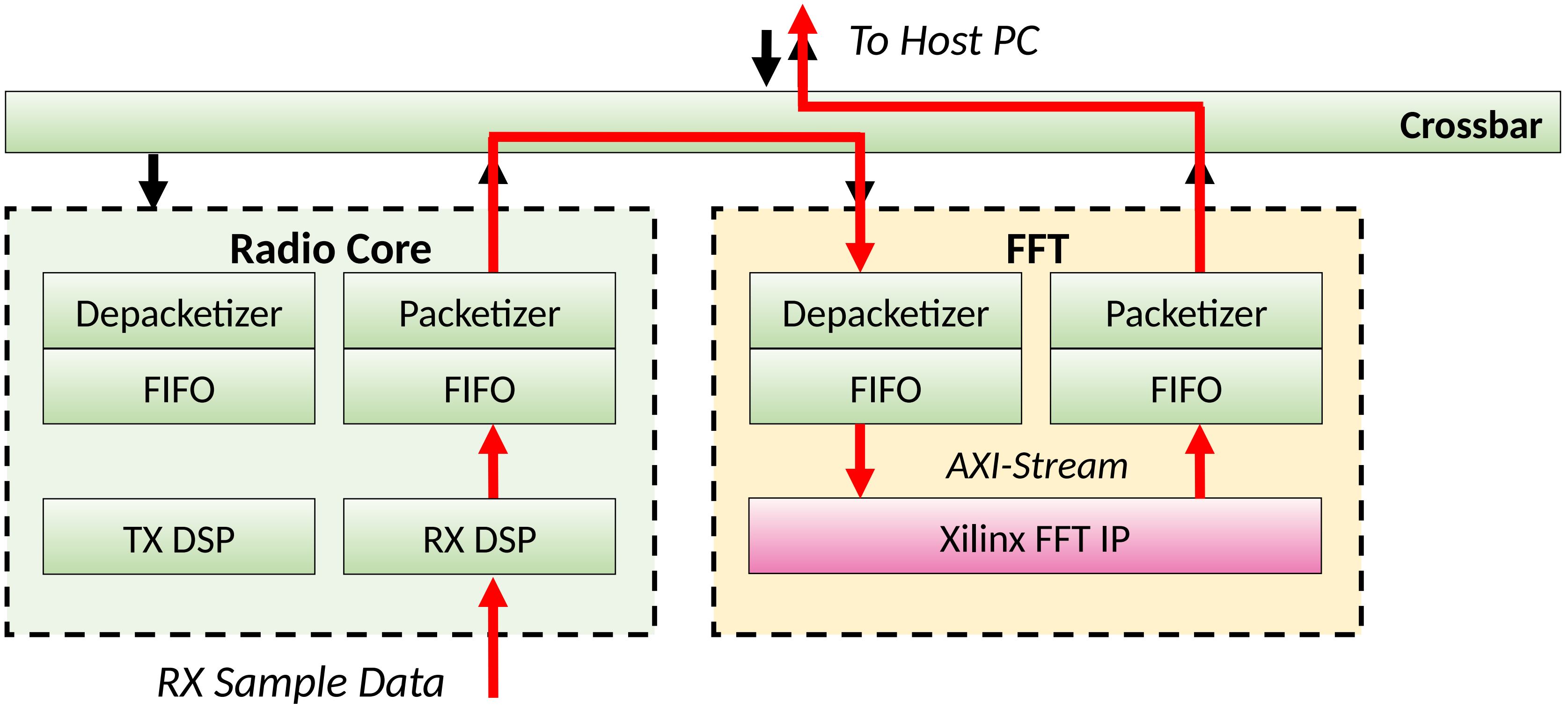
- FIFO to FIFO, packetization, flow control
- Provided by RFNoC infrastructure

Computation Engine



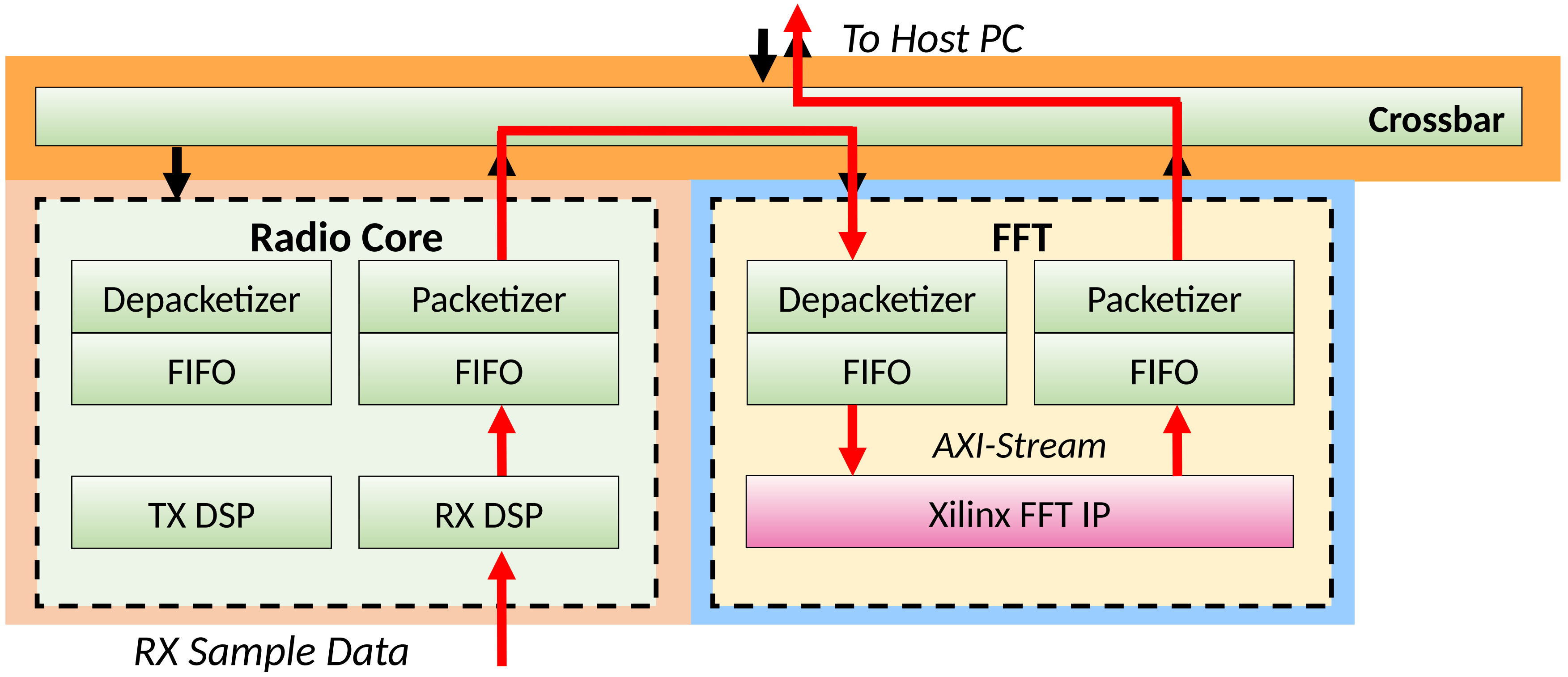
- User interfaces to RFNoC via AXI-Stream
  - Industry standard (ARM), easy to use
  - Large library of existing IP cores

Computation Engine



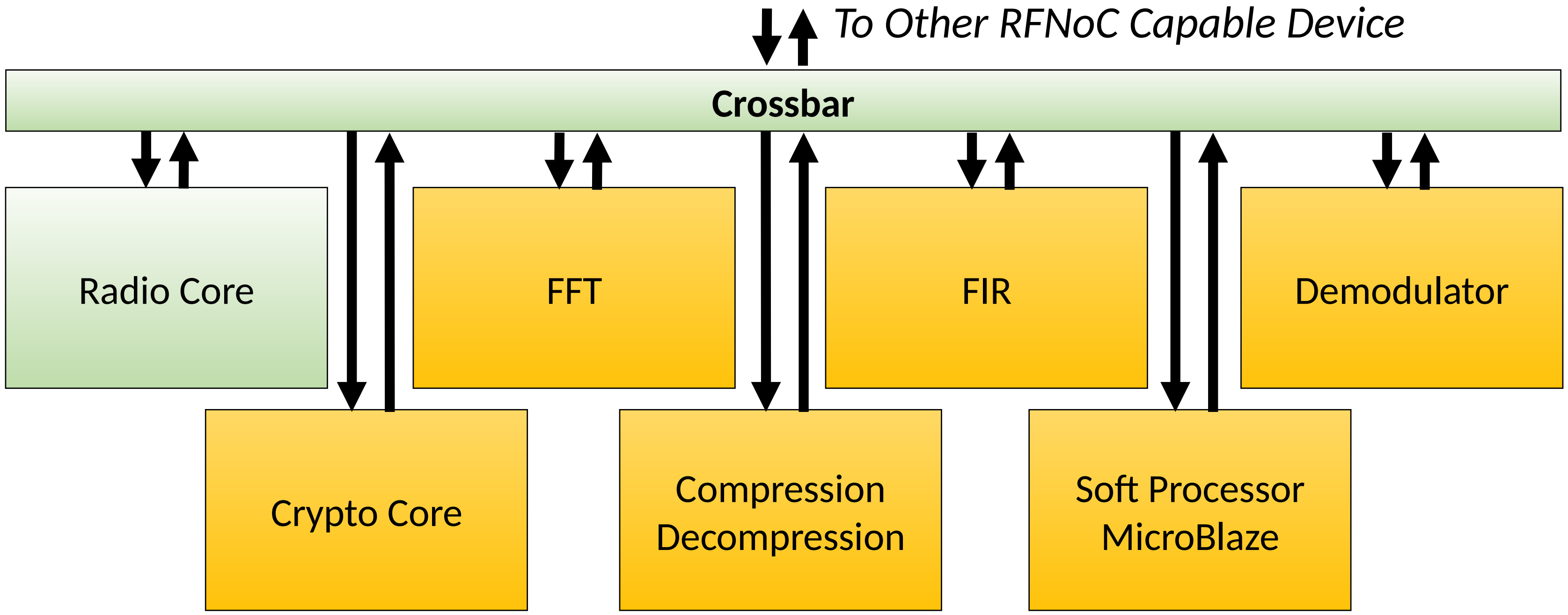
- User writes their own HDL or drops in IP
  - Multiple AXI-Streams, Control / Status registers

Computation Engine



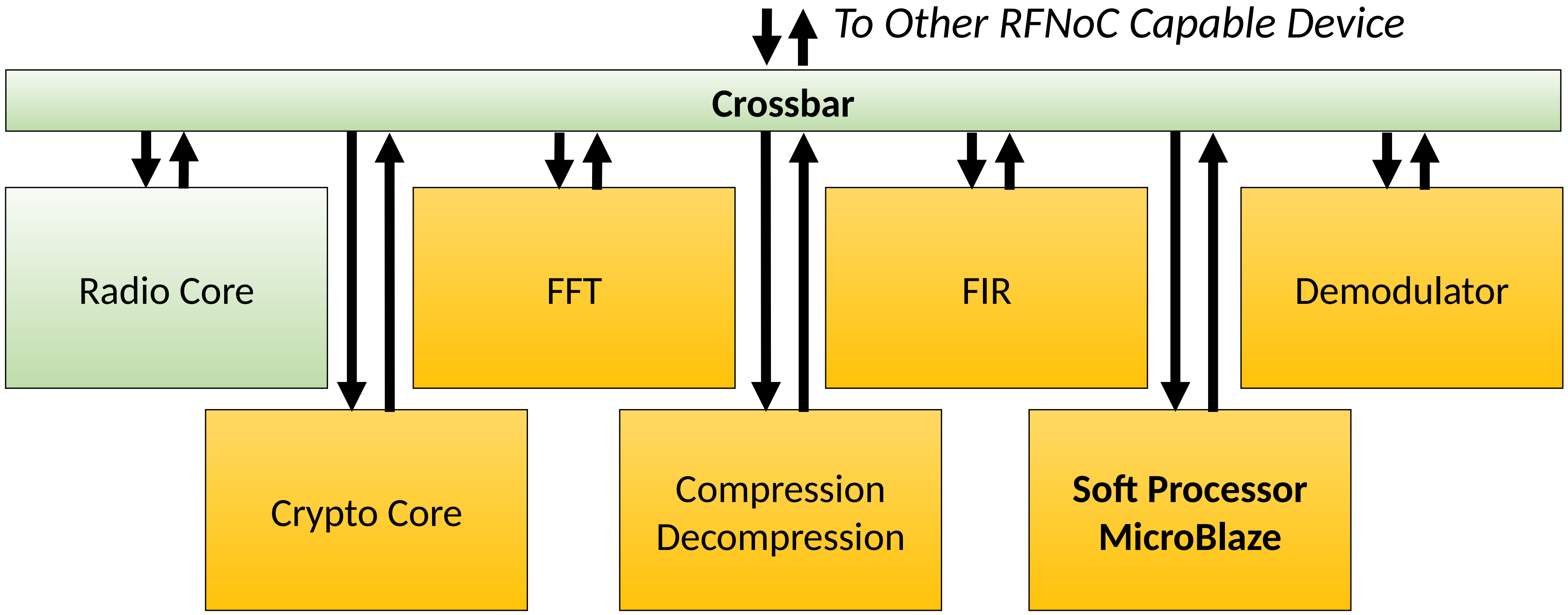
- Each block is in their own clock domain
  - Improve block throughput, timing
  - Interface to Crossbar has clock crossing FIFOs

Many Types of CEs


To Other RFNoC Capable Device
Crossbar
Radio Core
FFT
FIR
Demodulator
Crypto Core
Compression Decompression
Soft Processor MicroBlaze

- Many computation engines
- Not limited to one crossbar, one device
  - Scales across devices for massive distributed processing

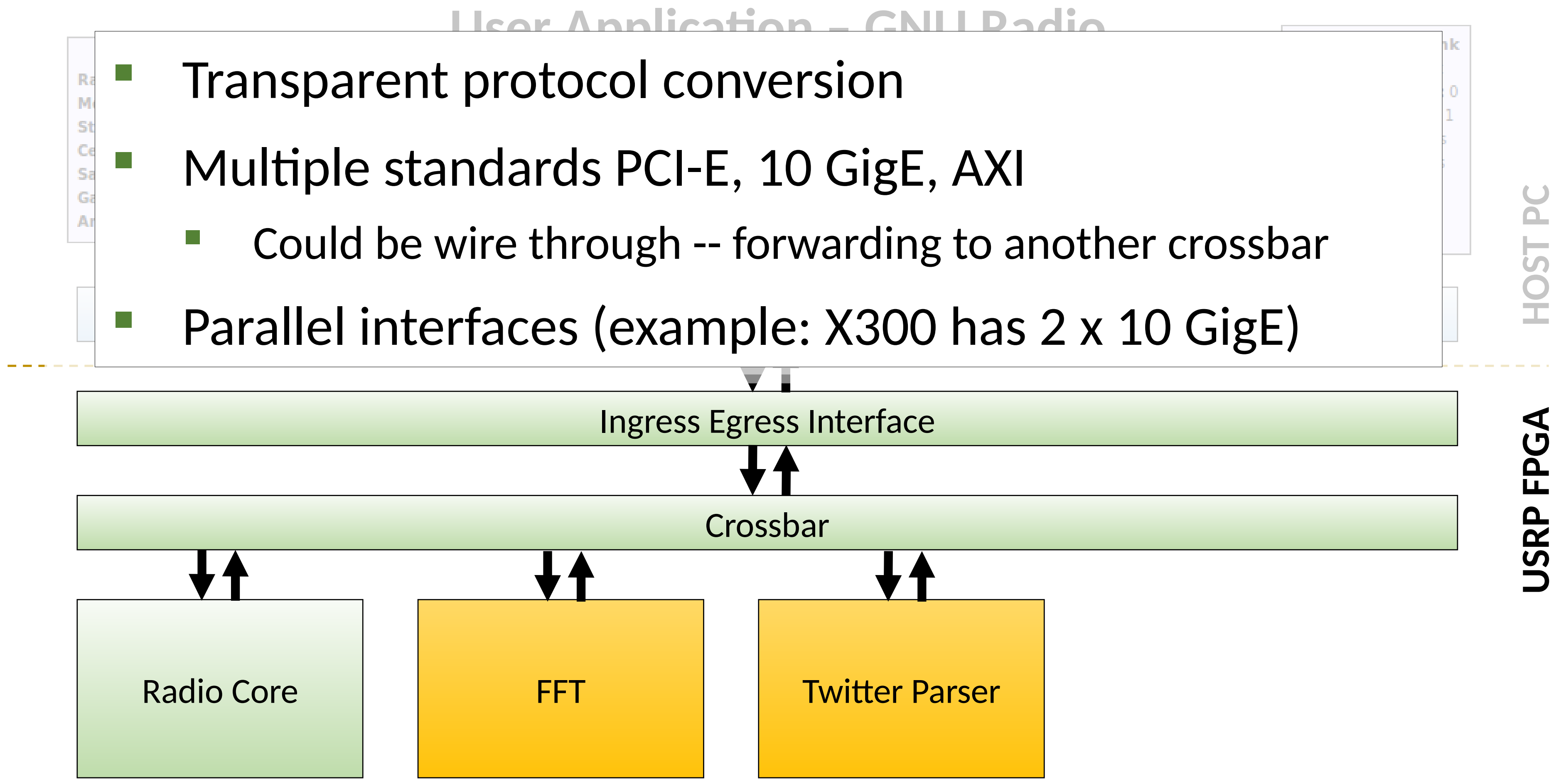
Many Types of Blocks


To Other RFNoC Capable Device
Crossbar
Radio Core
FFT
FIR
Demodulator
Crypto Core
Compression Decompression
Soft Processor MicroBlaze

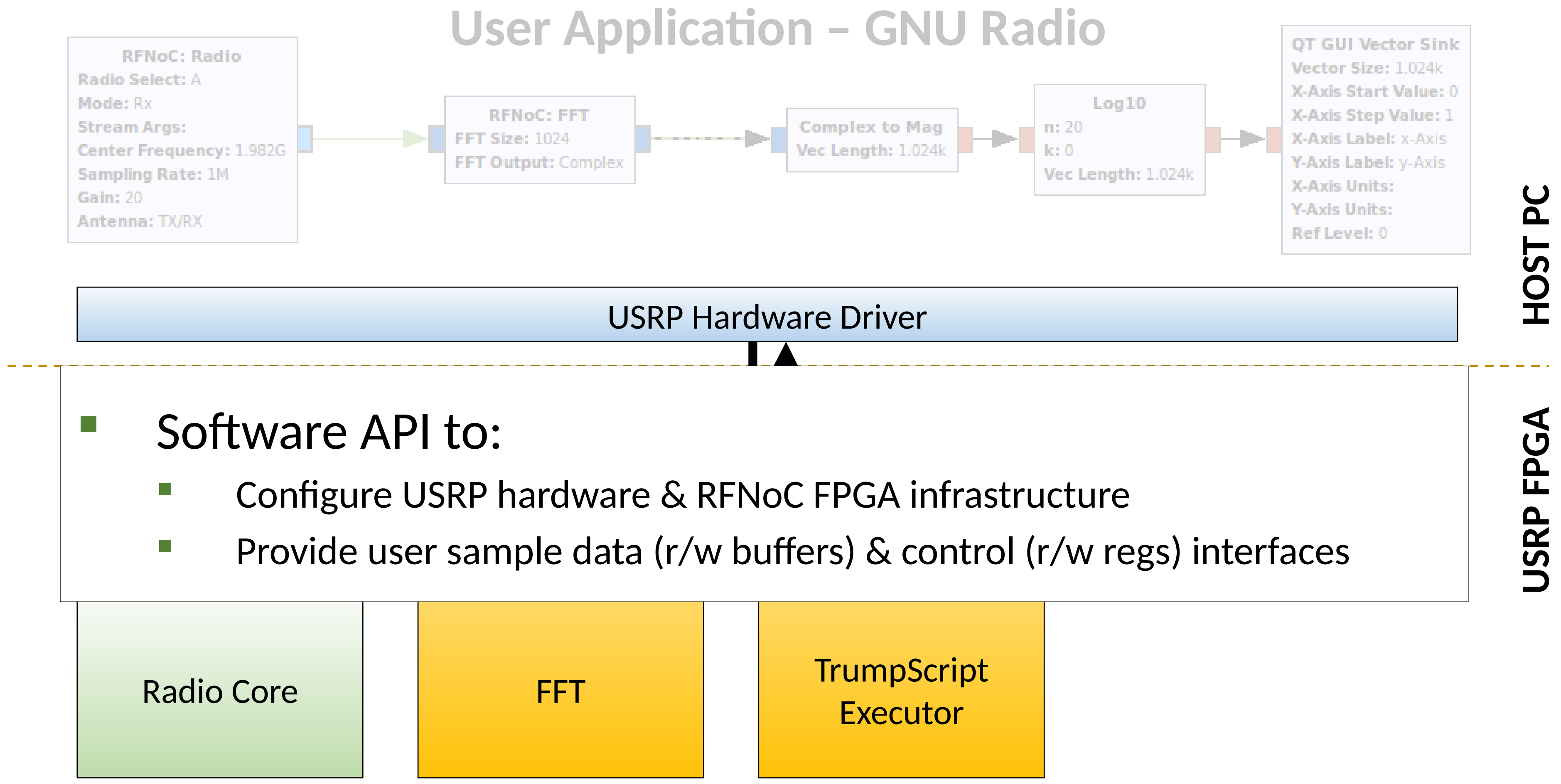
- Low latency protocol processing in FPGA

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)

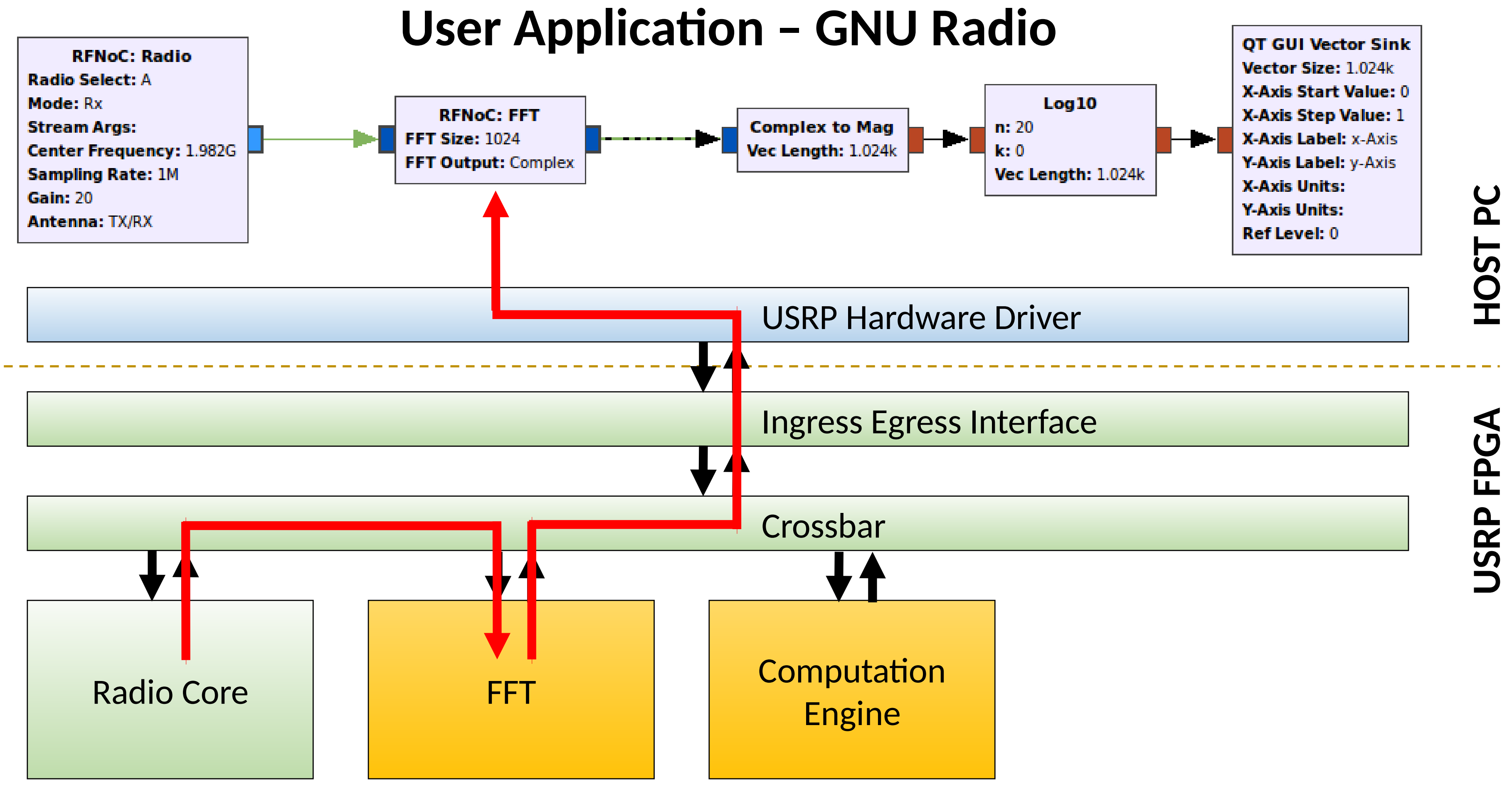


RFNoC Architecture



- Software API to:
  - Configure USRP hardware & RFNoC FPGA infrastructure
  - Provide user sample data (r/w buffers) & control (r/w regs) interfaces

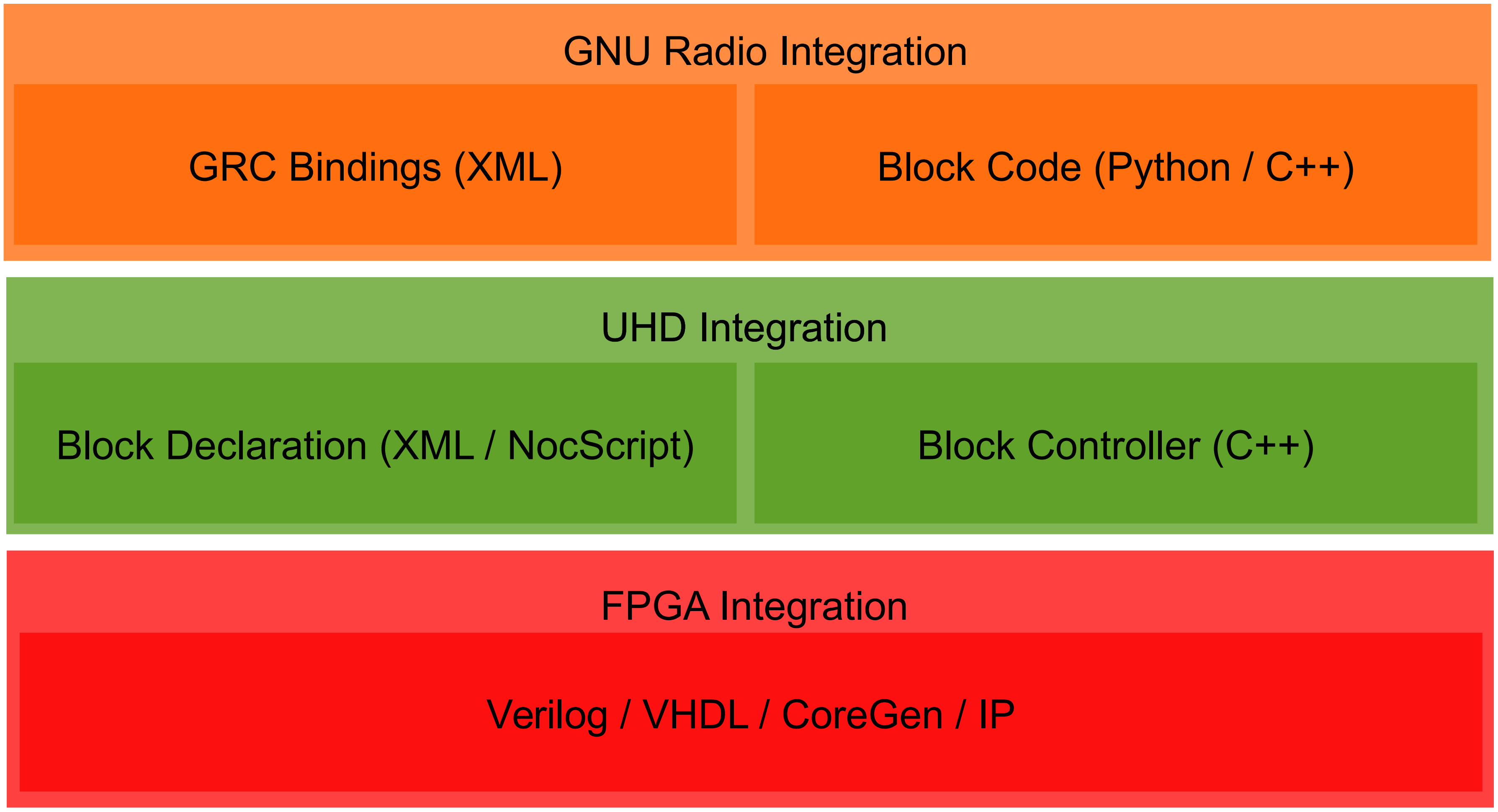
RFNoC Architecture



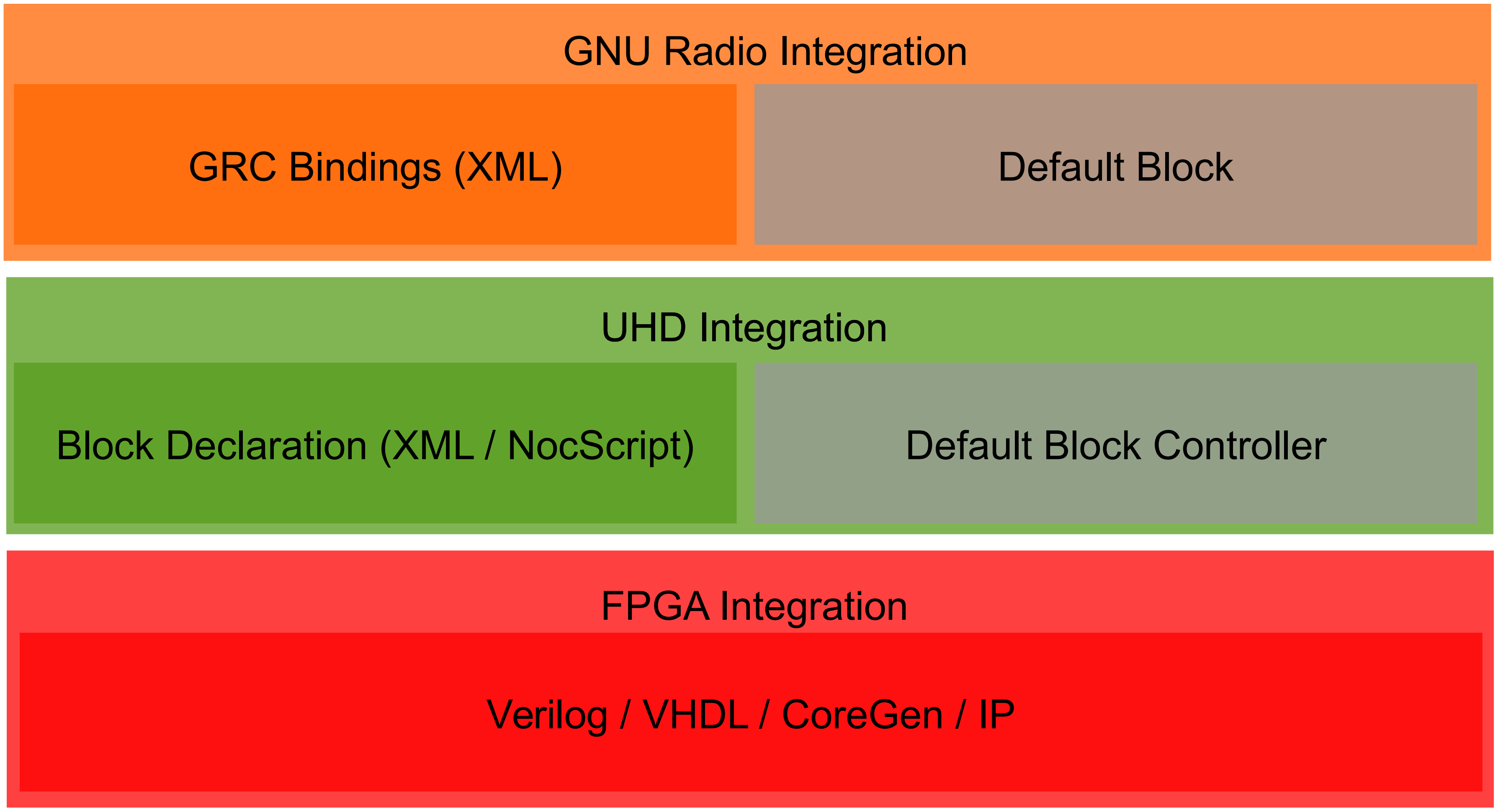
Ettus Research™
A National Instruments Company

DEMO

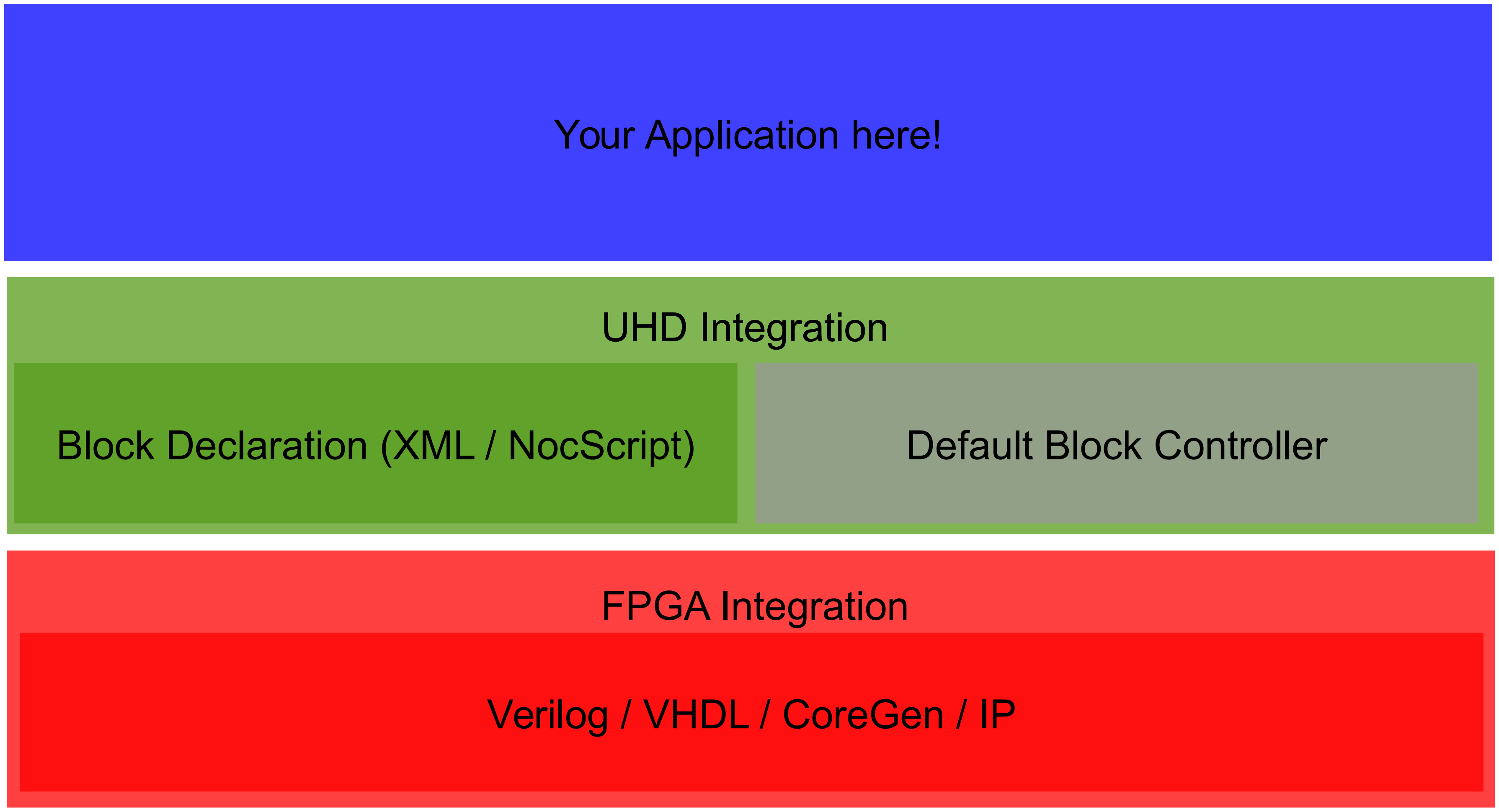
RFNoC Stack


GNU Radio Integration
GRC Bindings (XML)
Block Code (Python / C++)
UHD Integration
Block Declaration (XML / NocScript)
Block Controller (C++)
FPGA Integration
Verilog / VHDL / CoreGen / IP

RFNoC Stack (Simple)



RFNoC Stack (Even Simpler)



Summary

Ettus Research™ A National Instruments Company

- 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