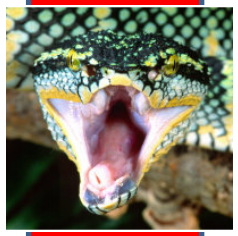
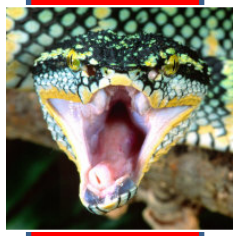


OpenCPI

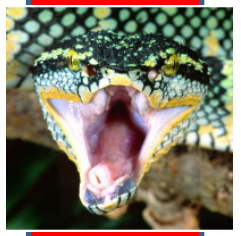
Introduction & Overview

Overview



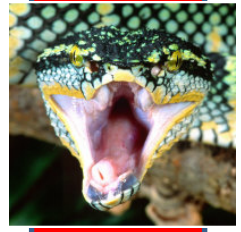
- What is OpenCPI?
- Comparison to other SDR frameworks

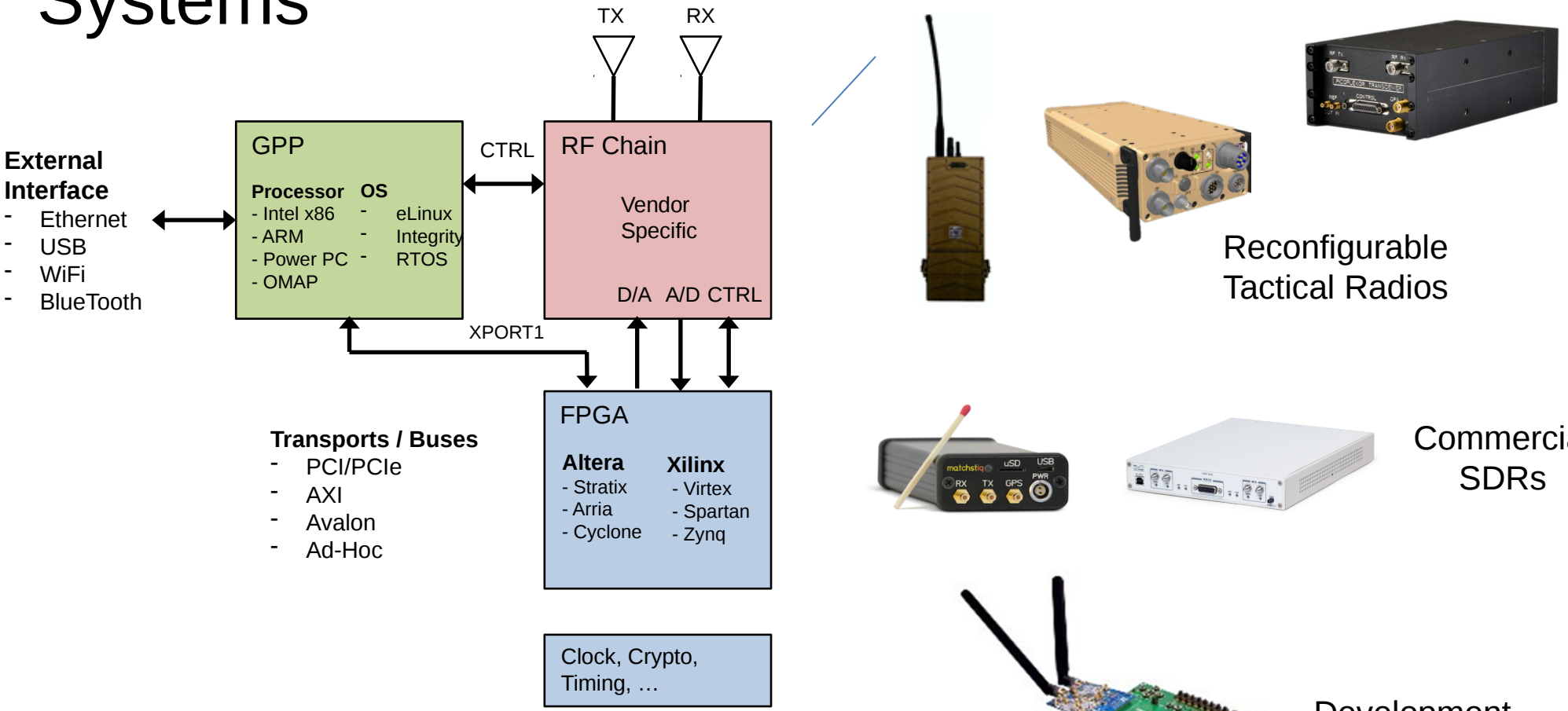
What is OpenCPI?



- OpenCPI is a framework for developing and deploying portable component-based applications
- Component-based development
- Targeting heterogeneous systems (GPP + FPGA)
  - Built-in transport between technologies
  - Automatic testbench code generation
- Vendor and technology neutral
- No required data format standards

OpenCPI Target Platforms: Heterogeneous Systems



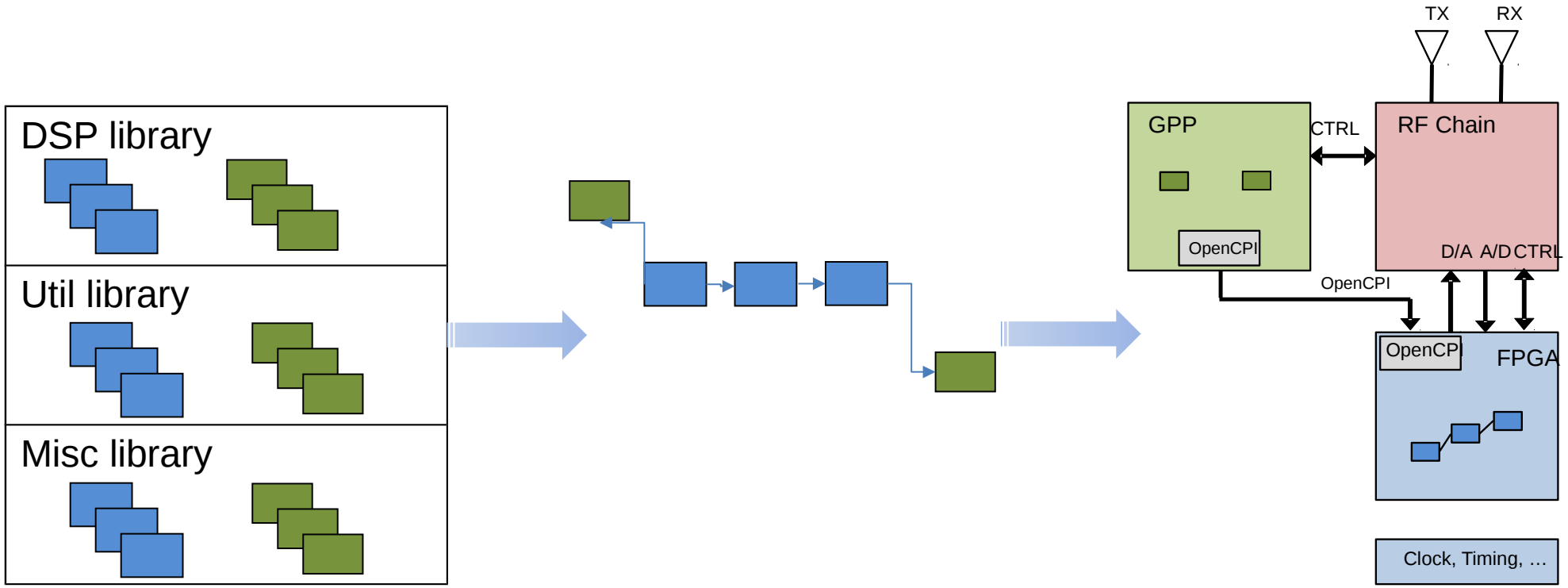


Reconfigurable Tactical Radios

Commercial SDRs

Development Platforms

OpenCPI Concept of Operation



Portable, reusable components are developed in VHDL/C++ independent of intended application or target platform

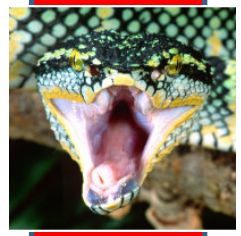
Using IDE, applications are constructed from existing libraries and built

Application deploys to target platform with components executing on disparate parts of platform

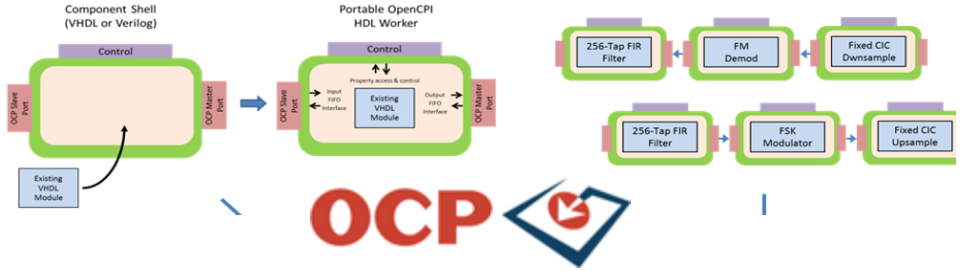
Open Source

github.com/opencpi

- OpenCPI (FOSS)
  - Component shell and structural code generation (VHDL, C/C++) to simplify component and application design
  - Supplies the IP and infrastructure for component interconnection, setup, control and data transfer
  - A robust vendor-neutral build environment that links to FPGA vendor tool chains to generate FPGA bit files
- OpenCPI has been developed in conjunction with the ANGRYVIPER Team for 5+ years
  - Framework history dates back 15+ years
- OpenCPI has features not yet explored by ANGRYVIPER Team
  - GPU support



Standards Based Code Generation



Portable Infrastructure for Control, DMA

PCI EXPRESS

I2C BUS

AMBA® Interconnect Standards from ARM

Build Engine Integrated with Vendor Tools

ModelSim

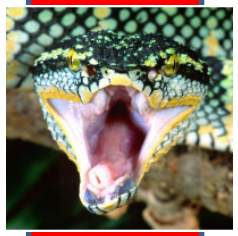
ALTERA®

ARM

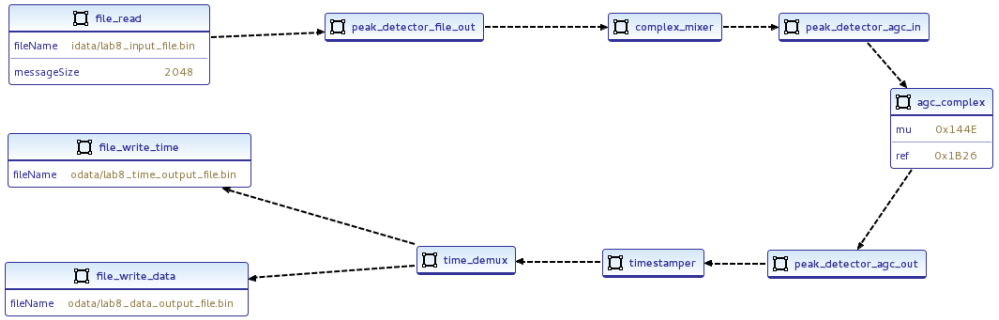
intel

XILINX ALL PROGRAMMABLE

ANGRYVIPER Team's Added Value



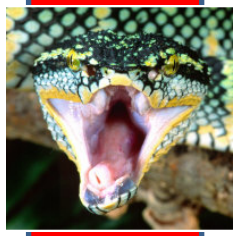
- IDE interface
  - Waveform development and Asset generation
  - Build and Test
- Develop a standard of API interfaces for radio controls
- Standardized Installation
  - RPM-based install for developers
- Updated Documentation and Training
- Component library including working transceiver examples
- Continuous Integration build and test for HDL and software



Documentation
- Data Sheets
- User Guides
- Installation Guides

Jenkins

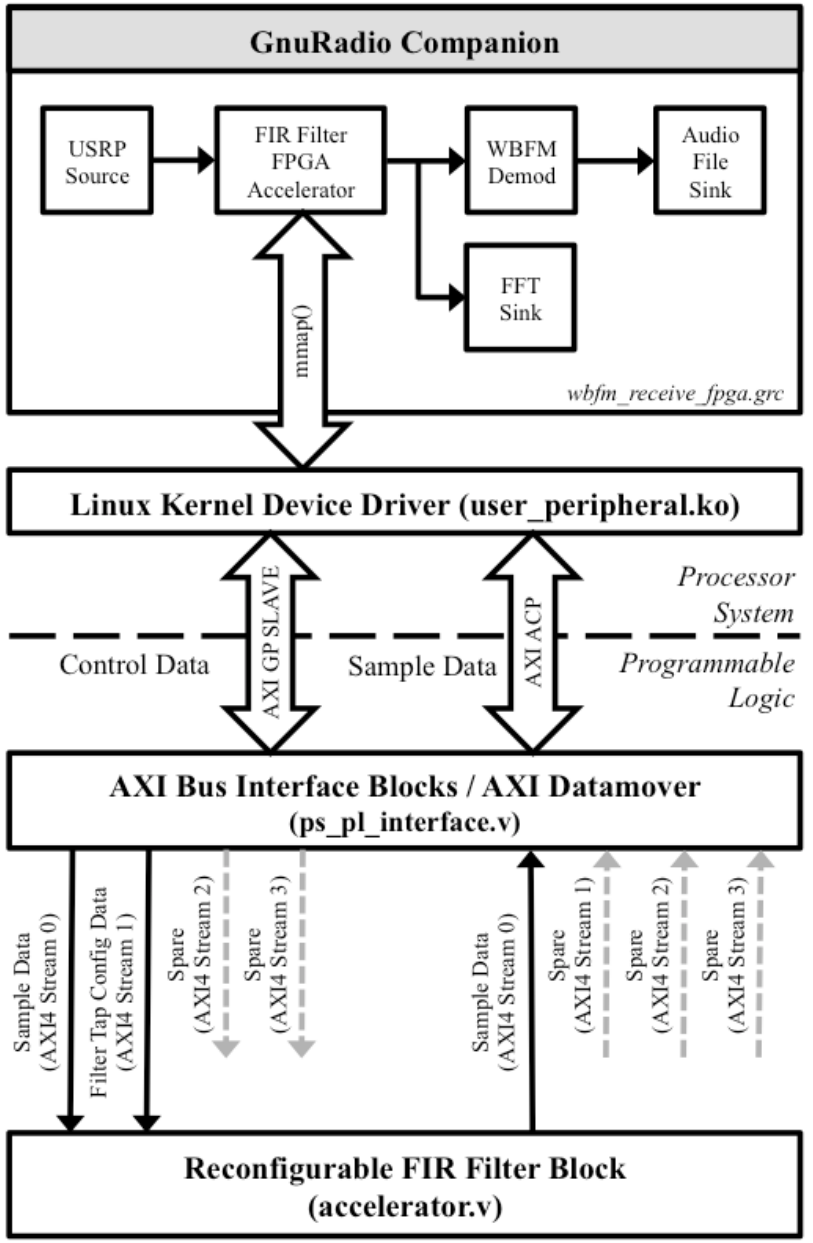
Comparison to Other Frameworks



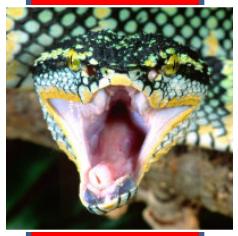
- GNURadio
- Ettus RFNoC
- REDHAWK
- OpenCL

GNU Radio
THE FREE & OPEN SOFTWARE RADIO ECOSYSTEM

- Highly popular FOSS Software Radio Framework
  - Large number of users and existing components
  - Mature IDE for developing applications and components
- Hardware is strictly not part of GNU Radio
  - Quote directly from their wiki
- No unified method for supporting FPGA and data exchange with host processors
  - Custom solutions requiring vendor IP and custom drivers
- Buffer management for compute engines requires memory copying
  - More inefficient than the shared memory approach of OpenCPI

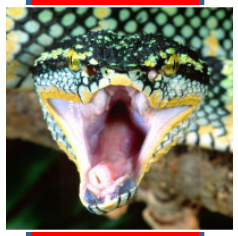


https://wiki.gnuradio.org/index.php/Zynq



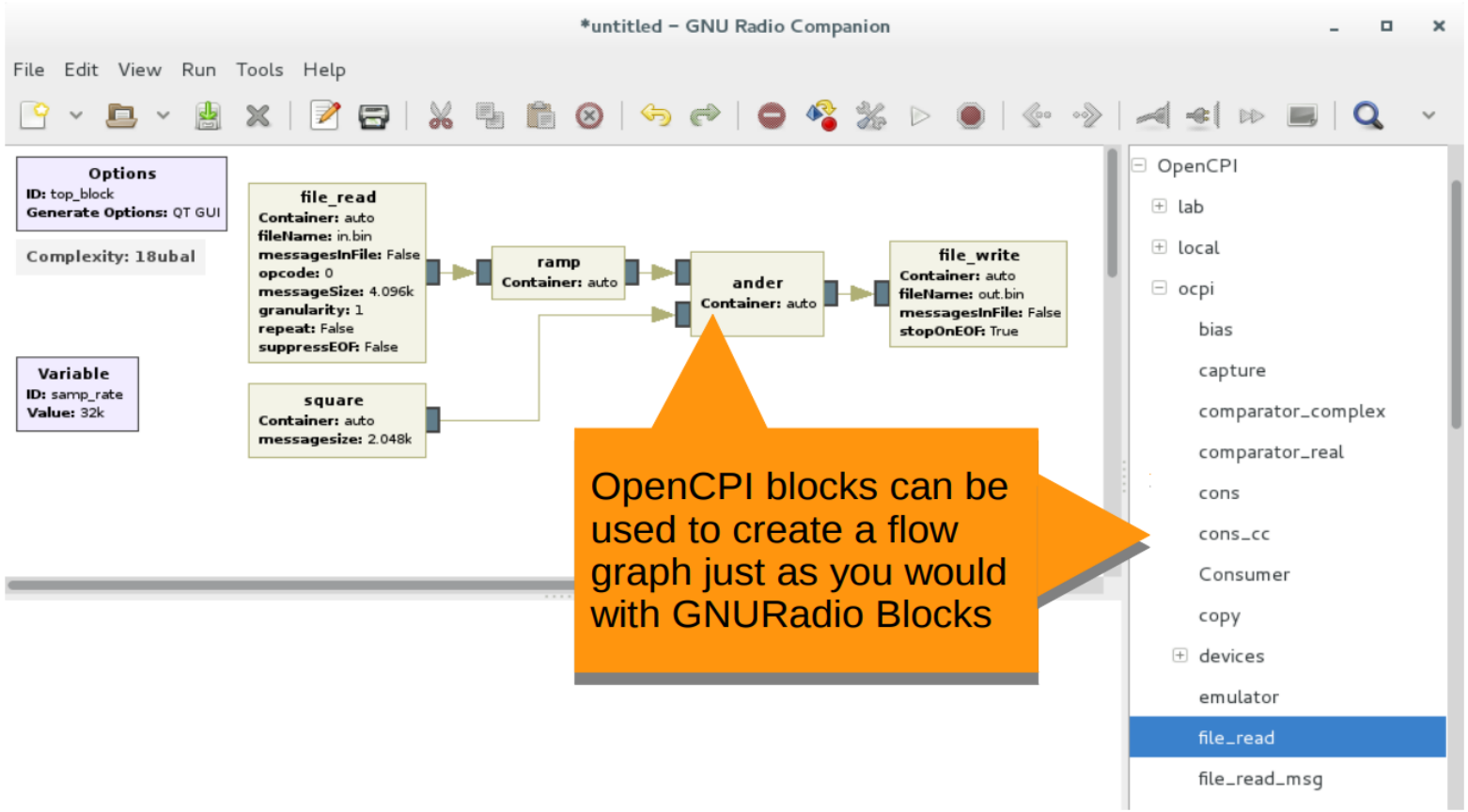
GNURadio
THE FREE & OPEN SOFTWARE RADIO ECOSYSTEM

OpenCPI's 2017 GNURadio Conference Presentation
https://youtu.be/CriLitW5vtM

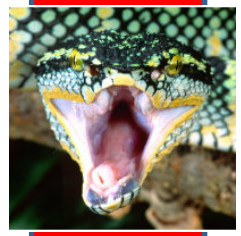


OpenCPI

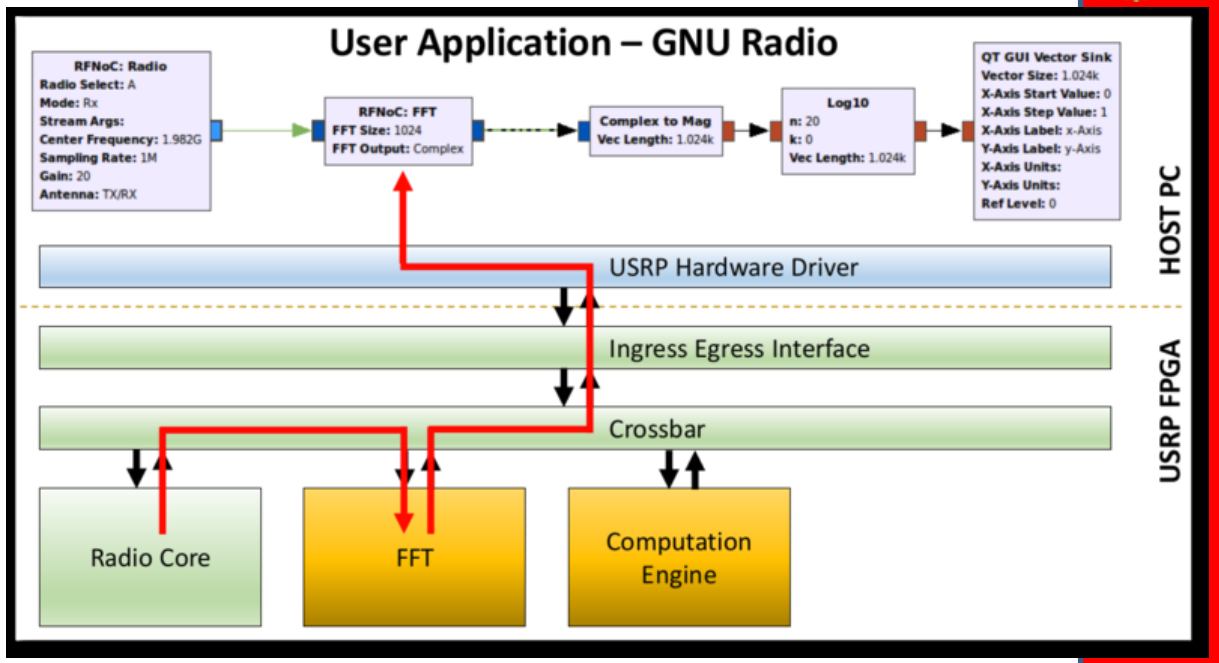
- **Preliminary integration with OpenCPI and GNURadio Companion**
  - Tool developed to translate OpenCPI XML into GRC block XML
  - GRC Flowgraph can be generated and run using the standard GRC buttons



RF NoC

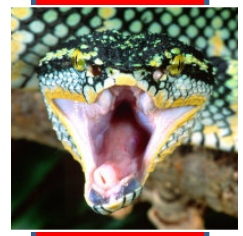


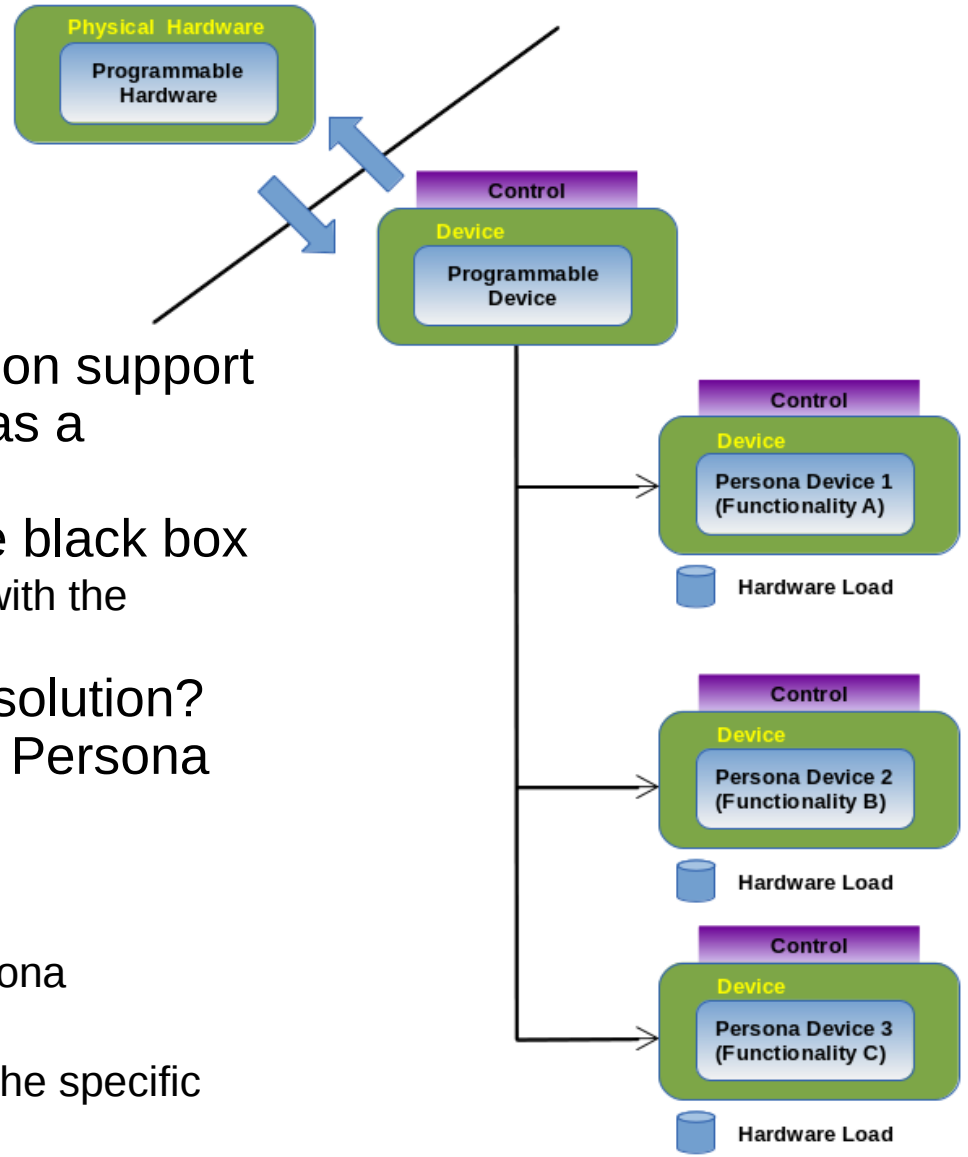
- Tool for enabling GPP-based and FPGA-based processing in an application
- Targets Ettus lines of radios
  - X and E series
- Re-configurable connections between components are made through a crossbar switch
  - Reliant on Xilinx IP cores (limits portability)
- Integrated with GNU Radio



https://kb.ettus.com/RFNoC

REDHAWK

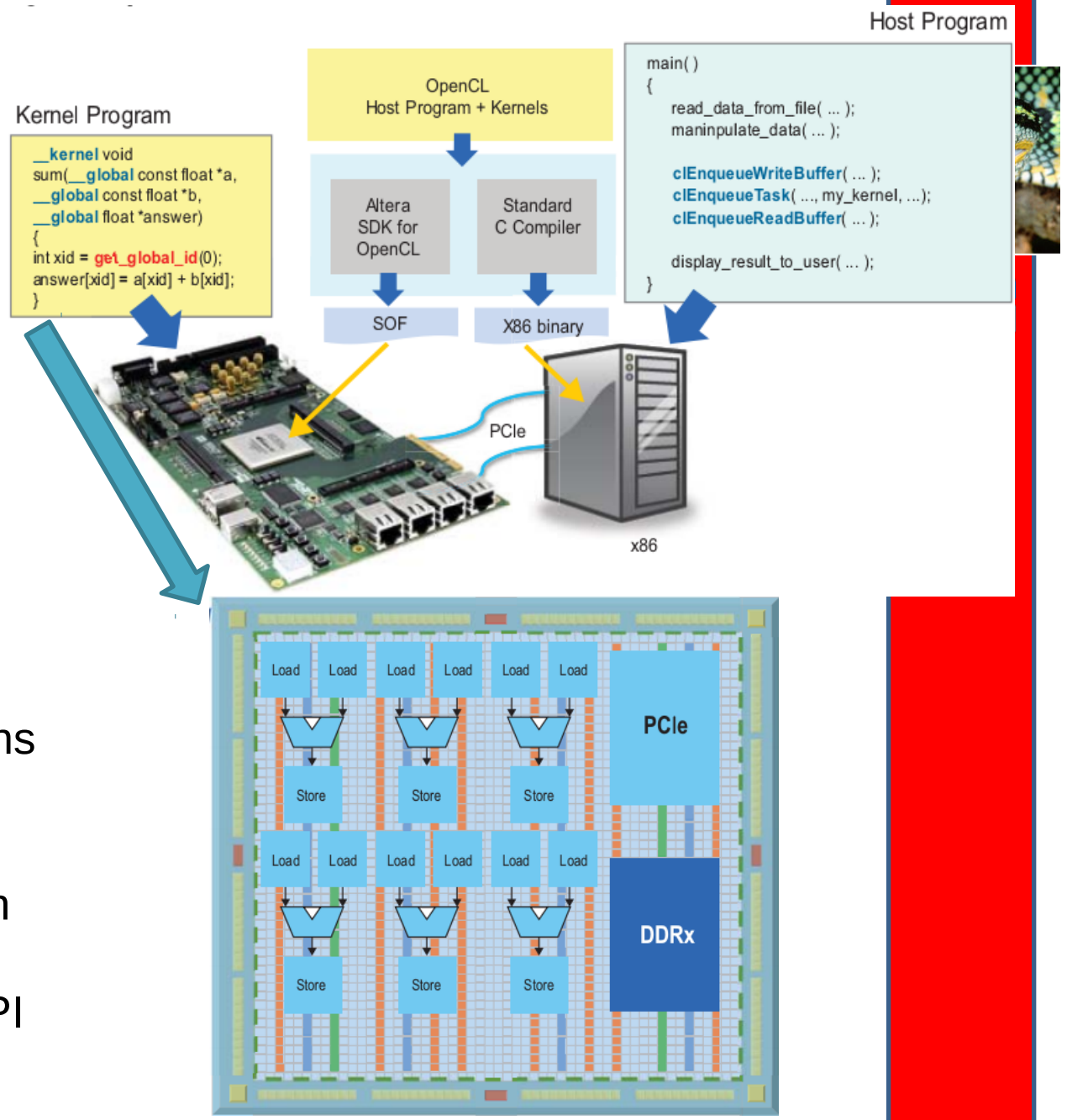




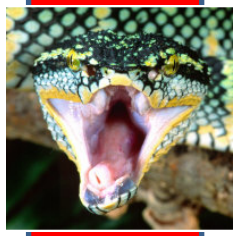
- Homogenous processing system for SDR
- Very SIGINT-focused with limited transmission support
- Hardware acceleration/offload were added as a secondary capability
- Does not dictate what is designed inside the black box
  - Only describes how the interface should interact with the framework
- What are the mechanics of the REDHAWK solution?
- Two part solution - Programmable Role and Persona
- Programmable role is the gatekeeper
  - Controls what Persona is loaded
  - Blocks subsequent requests until unloaded
  - Contains generic functionality common to all Persona
- Persona
  - Defines the load and any interfaces pertaining to the specific load
    - Bit file for FPGAs
    - Kernels for GPUs

OpenCL

- A framework for writing programs that execute across heterogeneous platforms
  - GPPs
  - GPUs
  - FPGAs
- Xilinx and Altera supported
  - Each require vendor-specific extensions to OpenCL
- Limited FPGA platform support
  - Mainly targeting hardware acceleration
  - No ADC/DAC design examples
- Can have OpenCL workers *within* OpenCPI



Comparison Summary



- Advantages
  - Extremely light weight
    - Minimal dependencies
  - Hardware/software interaction standard
    - No need for specialized drivers
  - Easy integration with other existing solutions
    - REDHAWK programmable device concept
    - GNURadio component model
    - OpenCL component Model
  - Low overhead execution model
    - Containers = process
    - Direct memory
    - No CORBA
  - FPGA vendor neutral
  - Existing ADC/DAC examples
- Disadvantages
  - Limited platform support
  - Smaller user base