FPGA Acceleration in GNU Radio: A Block-Centric Approach

By Troy Bates

Troy Bates - About Me

- B.S Clarkson University – Computer Engineering, '17
- M.S. University of Maryland – Embedded Systems, Exp Dec '25
  - Origin of this talk - 3-credit semester long project
- Research Scientist at Peraton Labs
  - Focus on Software Define Radio Development
  - Embedded systems FPGA and Microprocessor background
  - Extensive Experience with Xilinx System on a Chip

Field-Programmable Gate Array (FPGA) Overview

- **Field-Programmable Gate Array:** A semiconductor device built around a matrix of configurable logic blocks (CLBs).
    - **"Blank Slate" Hardware:** Unlike a CPU, FPGAs have no fixed functionality. Their internal hardware can be reprogrammed to perform specific tasks.
    - **Massively Parallel Architecture:** Consists of thousands to millions of independent, interconnected logic elements, allowing for true parallel processing.
- **Core Components**
    - **Configurable Logic Blocks (CLBs):** The fundamental building blocks, containing Look-Up Tables (LUTs) for combinational logic and Flip-Flops for sequential logic.
    - **Embedded Memory (Block RAMs):** High-speed, on-chip memory for storing data and lookup tables.
    - **DSP Slices:** Dedicated hardware for high-performance arithmetic operations (multiplication, accumulation).
    - **Programmable Interconnect:** A routing fabric that connects all the blocks together, allowing the user to define custom hardware data paths.
    - **High-Speed I/O:** Interfaces for communication with other devices (e.g., PCIe, Ethernet, SerDes).
- **Key Advantages**
    - **Performance:** Can achieve massive parallelism, leading to orders of magnitude performance increase over CPUs for specific tasks (e.g., signal processing, machine learning inference).
    - **Flexibility & Reprogrammability:** Can be reconfigured to adapt to new algorithms or standards without requiring a new chip.
    - **Low Latency:** Data moves through a dedicated hardware path, eliminating the overhead and latency of an operating system and software stack.
    - **Power Efficiency:** Hardware is designed specifically for a task, often leading to better power efficiency than a general-purpose CPU.
- **Common Applications:** Data Centers, Telecommunications, Automotive, Aerospace & Defense, Medical Imaging

Code Name Kirby

Reprogrammability:

- The core functionality of an FPGA tends to stay the same
  - Connection to I/O, memory, or processors are static resources
- The FPGA can do many different things in between I/O
- Traditionally FPGA systems are expensive and relatively complicated, users must understand at least at some level where their "stuff" needs to go

One may make an analogy to beloved Video Game Character Kirby, who can change their abilities based on the situation they're in

Xilinx Artix 7 Platform

- Artix 7 Released 2012
  - Designed for "cost-sensitive" and "power-limited" applications that require high-end features.
  - **No-Cost Xilinx License**
  - Supported at least through 2040
- Nitefury II (used in this project)
  - **PCIe Gen2 x4 = 2GB/s data transfers**
  - M.2 Form Factor
  - **$249MSRP (currently out of stock)**

NiteFury-II FPGA

Part Number	Logic Cells	DSP Slices	Memory (Kb)
XC7A12T	12,800	40	720
XC7A25T	23,360	80	1,620
XC7A50T	52,160	120	2,700
XC7A100T	101,440	240	4,860
XC7A200T	215,360	740	13,140

Other FPGAs

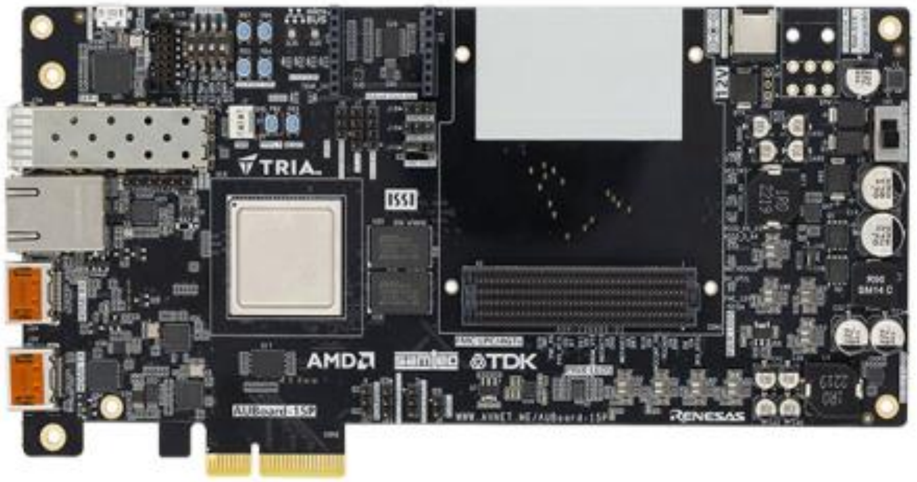
Captain DMA

- MSRP $210
- XC7A100T
  - LUTs: 101,440
  - DSP: 240
  - Memory: 4,860kB

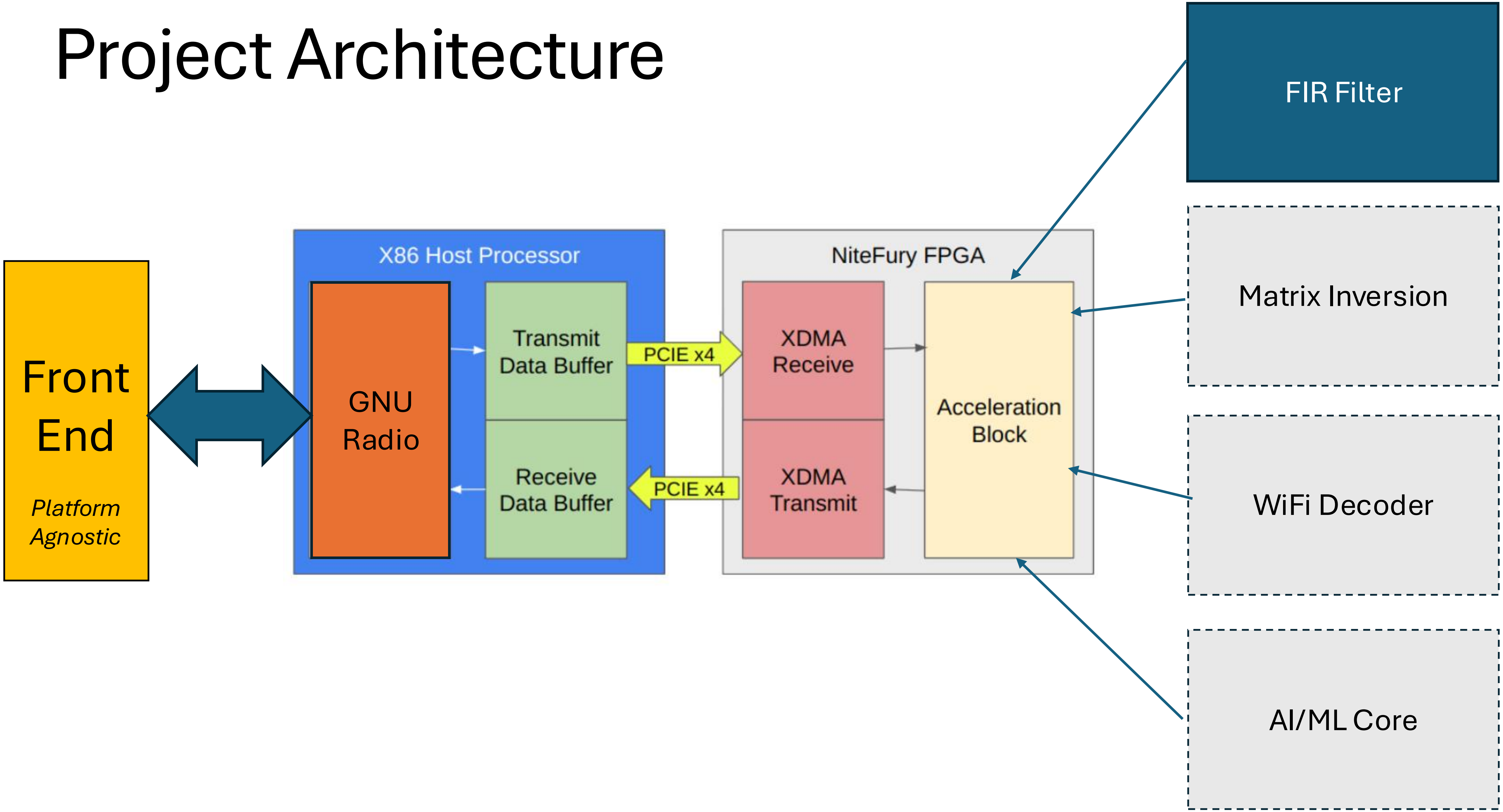
(Not a product recommendation but may provide an inexpensive platform)
https://captaindma.com/product/captain-dma-100t-7th/

Avenet: AES-AUB-15P-DK-G

- MSRP $699
- AMD Artix™ UltraScale+™ 15P FPGA
  - 170K logic cells
  - 7.6 Mb on-chip memory
  - 576 DSP slices
  - SSI 2GB DDR4 memory
  - ISSI 512Mb QSPI Flash memory (configuration and boot)
  - PetaLinux BSP available for download
  - SFP+ 10Gb Ethernet
  - PCIe Gen. 4 x4 end point interface
  - https://www.avnet.com/americas/product/avnet-engineering-services/aes-aub-15p-dk-g/evolve-66431652/



Project Architecture



Reference project

- Reference project leveraged Memory Mapped XDMA core
  - Utilization of XDMA Kernel Module (provided by Xilinx)
  - The goal of this reference project is to write host data to the DDR memory located on the FPGA
  - TX and RX rates of 1 GB/s
    - ½ theoretical throughput

      https://github.com/RHSResearchLLC/NiteFury-and-LiteFury/tree/master

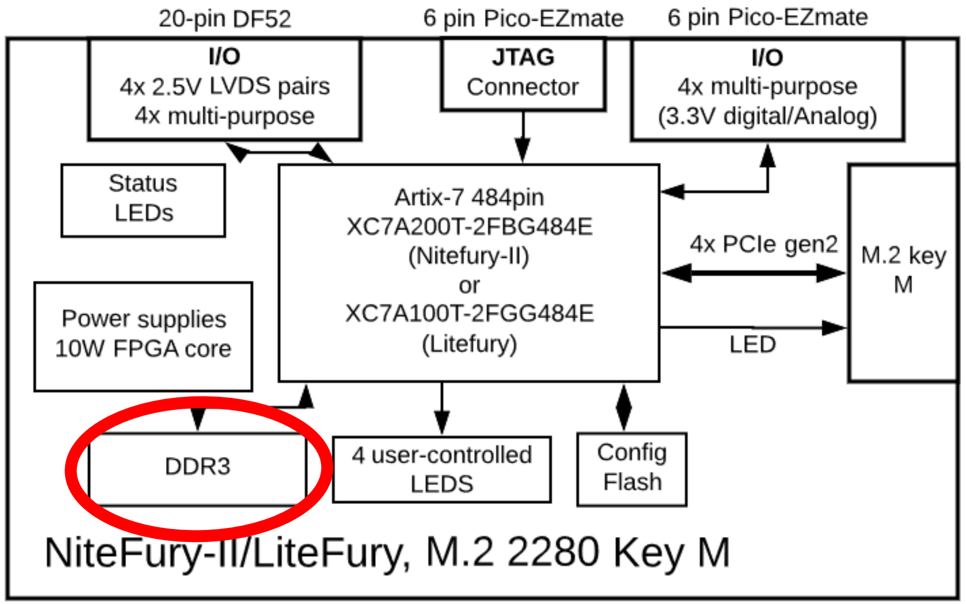
```
0a:00.4 Audio device: Advanced Micro Devices, Inc. [AMD] Starship/Matisse HD Audio Controller
tbates@tbates-linux-desktop:~/install/dma_ip_drivers/XDMA/linux-kernel/tests$ lspci | grep Xilinx
05:00.0 Processing accelerators: Xilinx Corporation 7-Series FPGA Hard PCIe block (AXI/debug)
tbates@tbates-linux-desktop:~/install/dma_ip_drivers/XDMA/linux-kernel/tests$
```

Lspci results with recognized FPGA

```
tbates@tbates-linux-desktop:~/install/dma_ip_drivers/XDMA/linux-kernel/tests$ sudo ./load_driver.sh
interrupt_selection .
Loading driver...insmod xdma.ko interrupt_mode=2 ...

The Kernel module installed correctly and the xmda devices were recognized.
DONE
tbates@tbates-linux-desktop:~/install/dma_ip_drivers/XDMA/linux-kernel/tests$
```

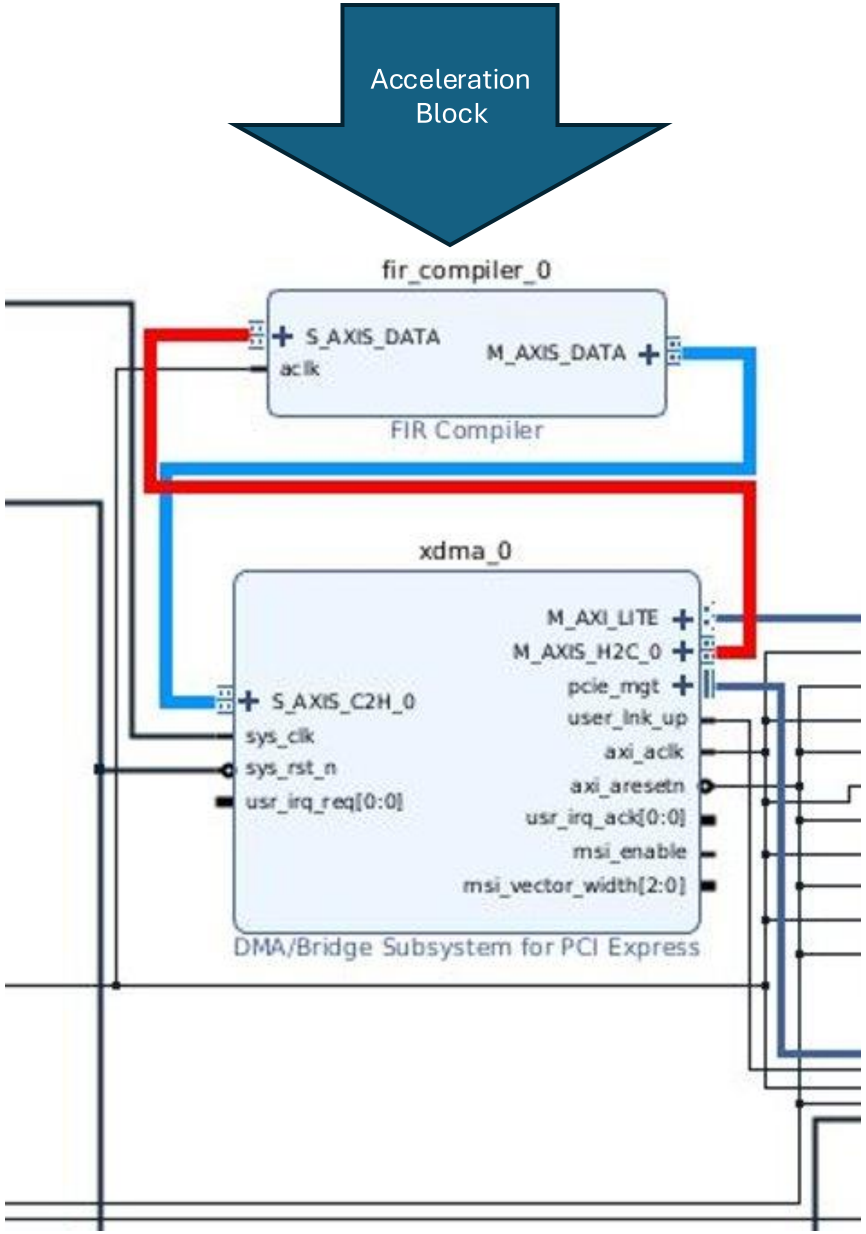
Inserting Kernel Module



NiteFury II HW layout

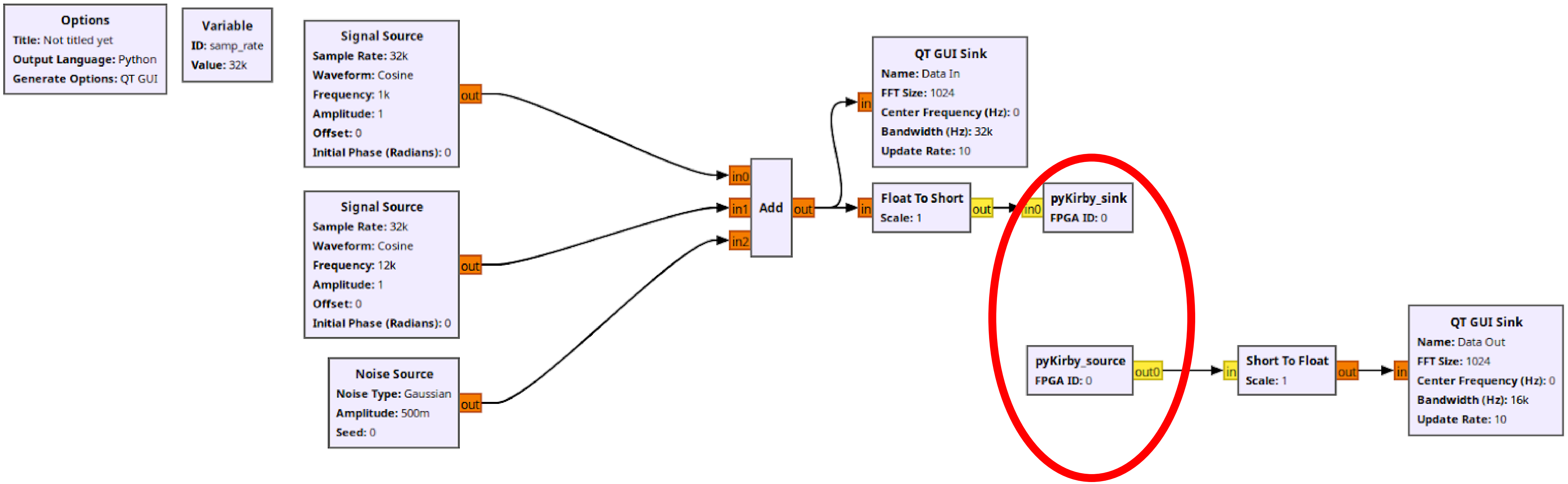
Kirby Design

- AXI Overview
  - AXI4: memory-mapped, supports random access reads/writes with addresses.
  - AXI4-Stream: lightweight, unidirectional, continuous streaming of data (no addresses, just flow control).
- The updates to this block simply stream data in and out of the XDMA shown in Blue and Red



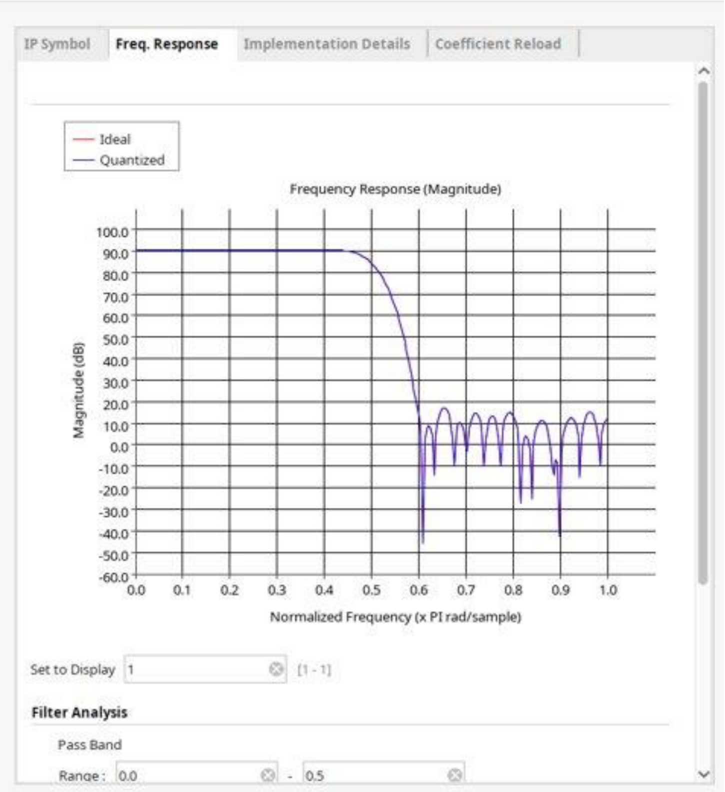
Kirby HDL Design Changes

Current Design

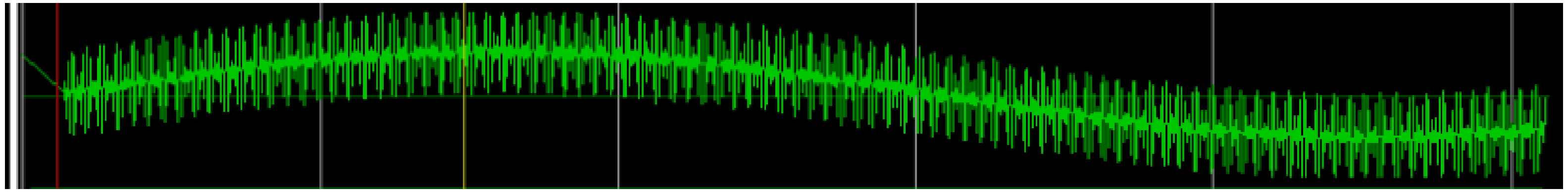


- Python Blocks (Quick approach to integration)
  - **PyKirby Sink**
  - **PyKirby Source**
  - PyKirby Sync <- Not Implemented

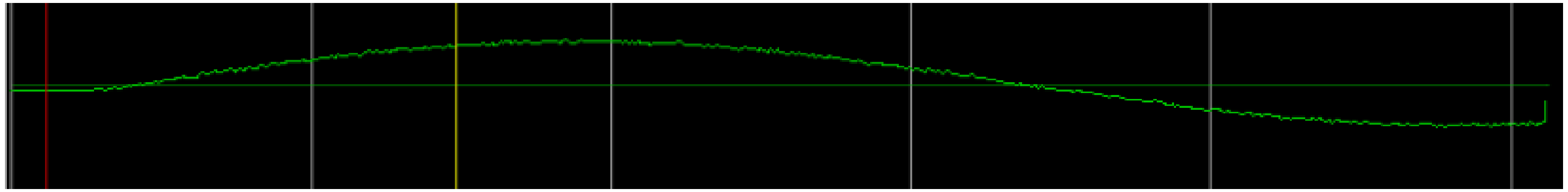
Vivado FIR Filter



FIR Frequency Response



Data In ILA Output



Filtered ILA Output

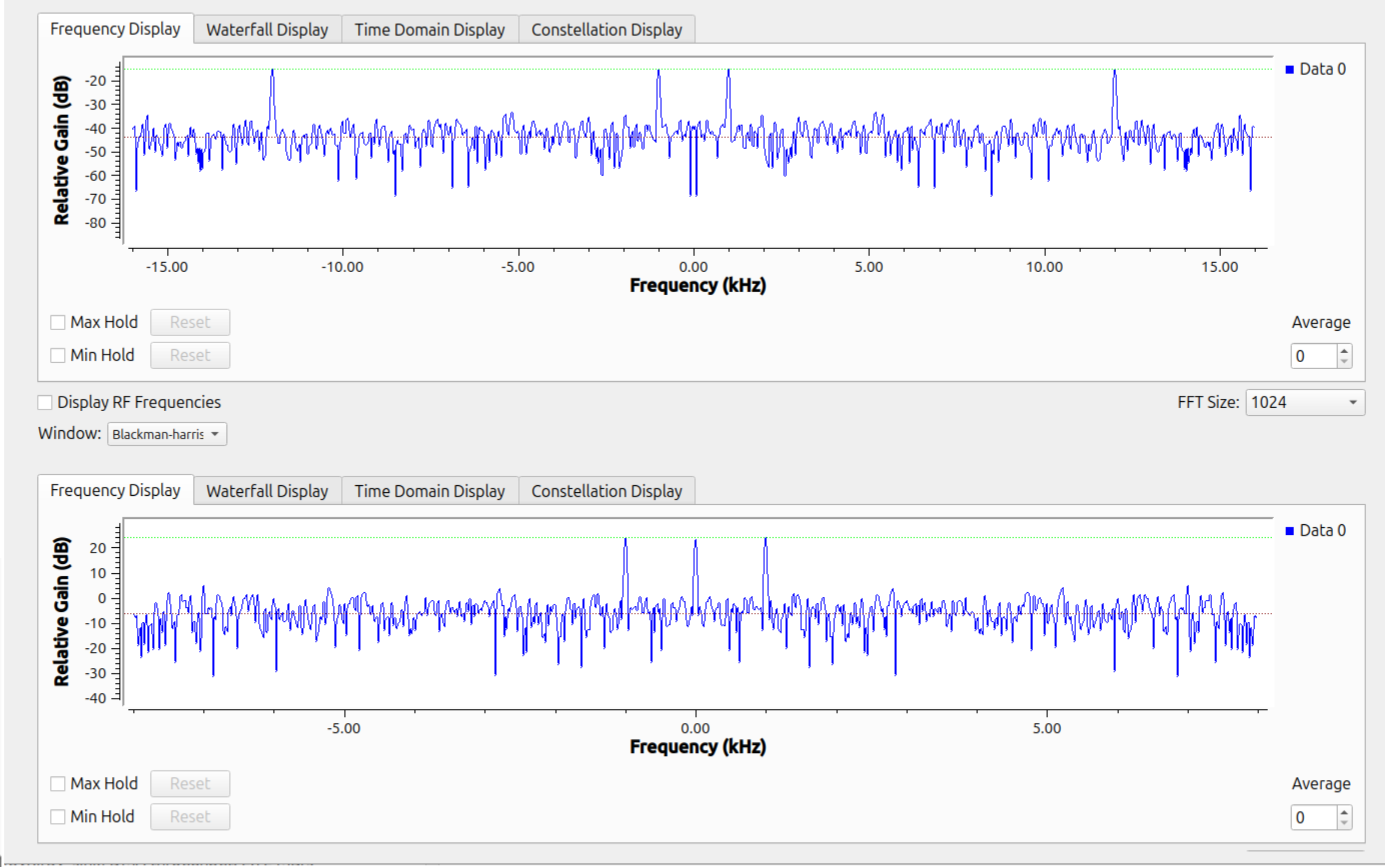
Performance and plots

2:1 Decimation Filter

- Supports 8 samples per clock cycle
- Sink supports **~90MB/s**
- Source **~40MB/s** (Expected ½ Input)

```
pyKirby_sink] DMA throughput: 97.97 MB/s
pyKirby_source] Source DMA throughput: 28.22 MB/s
pyKirby_sink] DMA throughput: 103.03 MB/s
pyKirby_source] Source DMA throughput: 36.68 MB/s
pyKirby_sink] DMA throughput: 94.82 MB/s
pyKirby_source] Source DMA throughput: 36.68 MB/s
pyKirby_sink] DMA throughput: 103.43 MB/s
pyKirby_source] Source DMA throughput: 38.98 MB/s
```

Py Kirby Performance Metric



GNU Radio Output Plots

A Lot of Work to Do

- Reloading FPGAs is very clunky
  - Updating build requires unloading and reloading kernel module
  - Permissions required for xdma channel
- Artix 7 platform is old
  - Technology has greatly progressed
  - PCIe gen 2 is data (though is probably good enough)

Metadata - Work Continued

Problem – GNU Radio has no knowledge of the FPGA capabilities

- Defined Memory capabilities across acceleration blocks
  - Metadata can define blocks in a library as well as on the board
  - Libraries could exist as a folder/installed package in gnu radio
  - Must track part compatibility

Field	Offset	Data
Build ID Number	0x0000	32-bit unique ID
Version Number	0x0004	Major, Minor, Patch (Packed)
Part Identification	0x0008	32-bit unique ID
Board Capabilities	…	To Be Defined
Stream 0	0x0100	
Stream Capabilities	…	To Be Defined
Stream 1	0x0200	
Stream Capabilities	…	To Be Defined

Example Memory Layout

Questions?

Email: Troy.B.Bates@gmail.com

Git Project:

https://github.com/too9le/gr-pyKirby.git

https://github.com/too9le/NiteFury-and-LiteFury

References and Links

Reference Git project: https://github.com/RHSResearchLLC/NiteFury-and-LiteFury

XDMA Driver: https://github.com/Xilinx/dma_ip_drivers/tree/master/XDMA/linux-kernel

NiteFury II Sales Page: https://rhsresearch.com/collections/rhs-public/products/nitefury-xilinx-artix-fpga-kit-in-nvme-ssd-form-factor-2280-key-m