GNU Radio and RFNoC in Space: How Hawkeye 360 uses GNU Radio on Small-Satellites

EJ Kreinar

Dan CaJacob

©2018 HawkEye 360

HawkEye 360

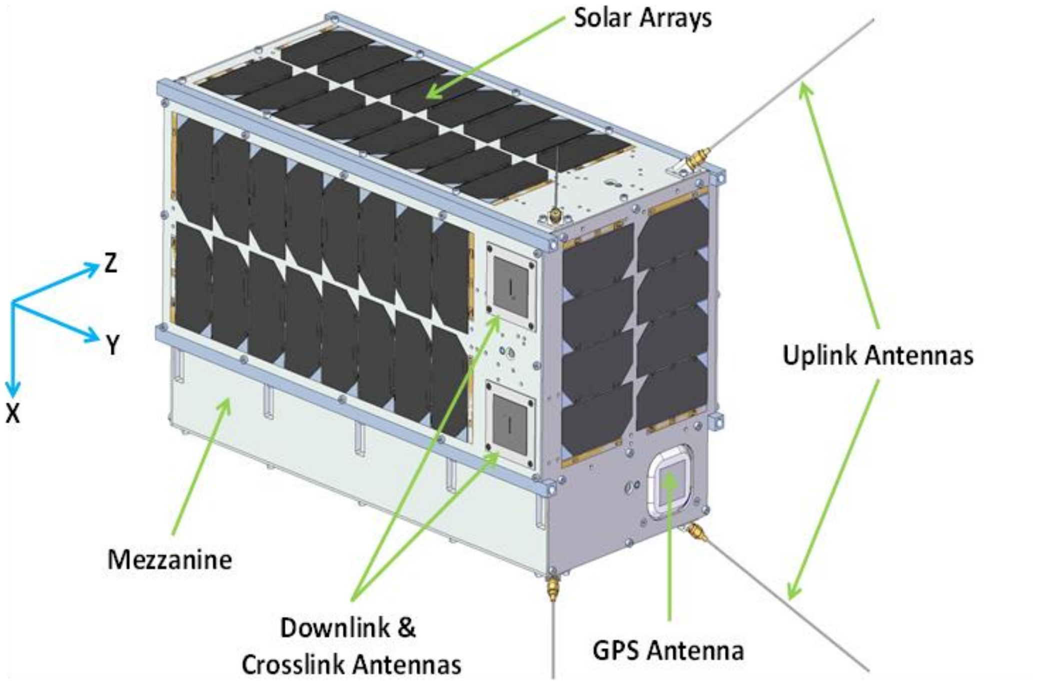
Topics

- Hawkeye 360 Overview
- Gnuradio + RFNoC: Prototype and Production
- **Use-Case Example**: Custom OQPSK Communications System
- **Use-Case Example**: Full-rate data captures on Zynq

©2018 HawkEye 360

HawkEye 360

- Venture-backed startup in Herndon, VA
- Technical Mission:
  - Launch a cluster of small satellites in Fall 2018
  - 3 satellites per cluster, flying in formation
  - Satellites in LEO (low earth orbit) in a polar orbit
  - Satellites share a common ground footprint and provide geometric diversity
  - Passively receive RF signals
  - Independently geolocate emitters from 100 MHz to 15 GHz ("DC to Daylight") using TDOA and FDOA measurements



©2018 HawkEye 360

Hawkeye 360 Development Concept

- Use open source tools to prototype, develop, and deploy RF applications
- Software-defined radio:
  - Dynamic reconfigurability
  - Regular software updates, improvements, bug fixes
  - Rapid iteration cycle
- Gnuradio + RFNoC:
  - Software and FPGA development across a variety of open source/ low cost devices
  - Industry-standard for RF processing applications

Gnuradio and RFNoC in space...

... looks pretty similar in the lab

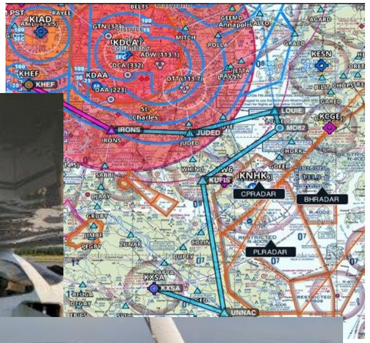
©2018 HawkEye 360

Hawkeye 360 Payload and Flight Kit

Hawkeye "Flight Kit"

- Ettus B200
- Ettus E310
- Battery
- Odroid
- Reconfigurable RF frontends

Runs the same software and FPGA as payload

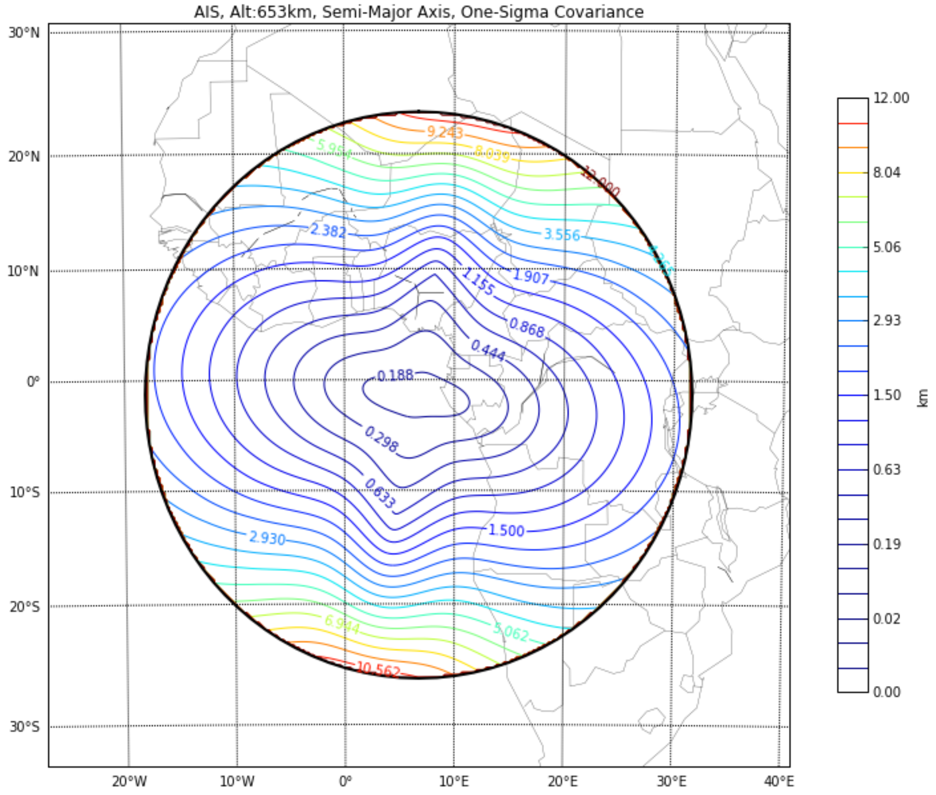


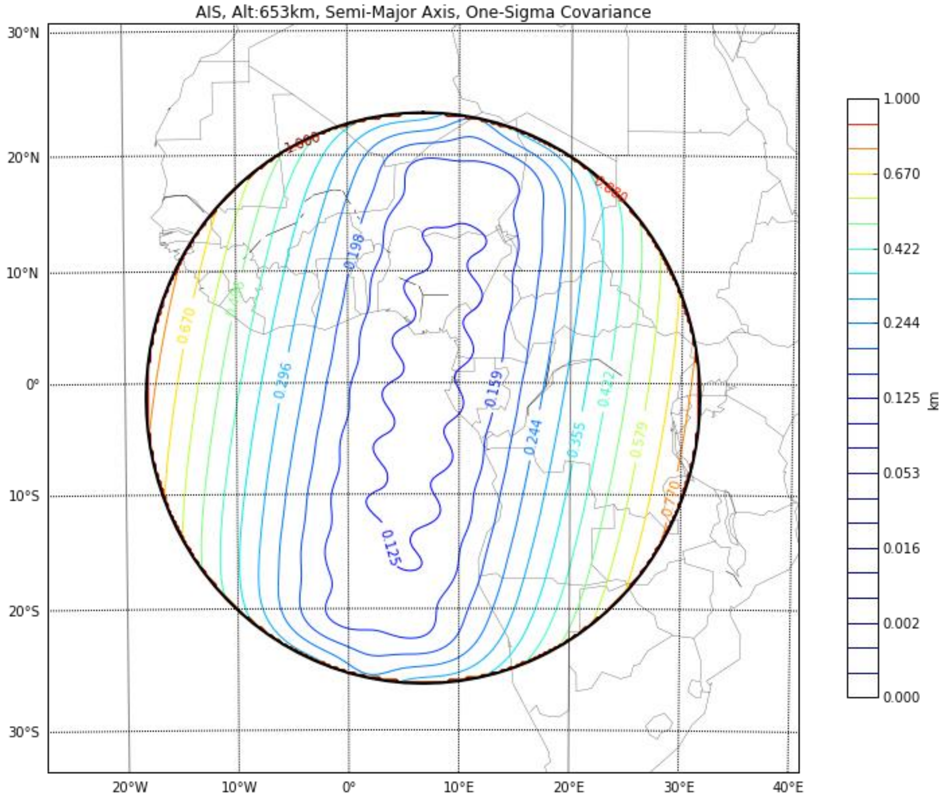
Hawkeye Payload

- Zynq 7045
- 3x AD9361s
- RFNoC
- Openembedded Linux
- Gnuradio

©2018 HawkEye 360

- CRLB maps show the effect of formation geometry and system design on theoretical geolocation accuracy for a signal of interest
- Used to predict performance





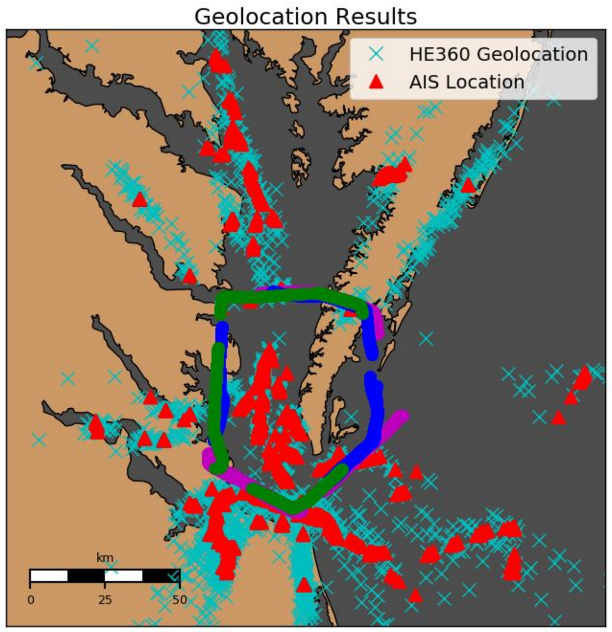
©2018 HawkEye 360

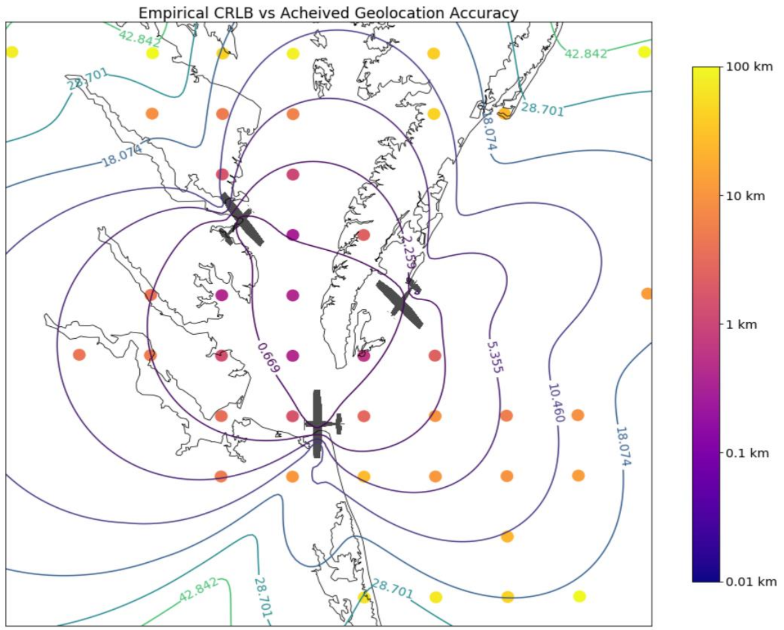
HawkEye 360

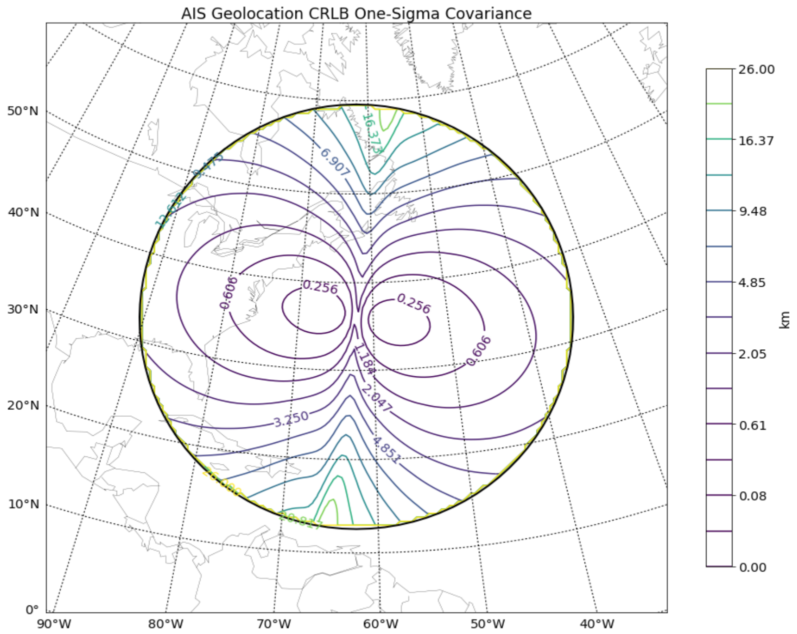
Achieved Airborne Performance

- Provide a basis for validating time of arrival and frequency of arrival measurements, critical to geolocation accuracy and performance
- Compare results vs CRLB simulations for both airplanes and overhead performance

Geolocation Results vs Theoretical (airborne and space)







©2018 HawkEye 360

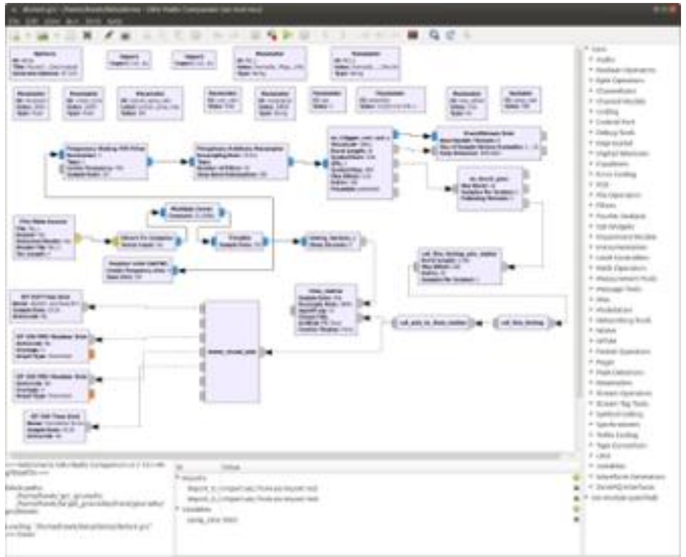
HawkEye 360

Gnuradio and RFNoC: Prototype and Production

HawkEye 360

©2018 HawkEye 360

- Use Gnuradio as framework for signal processing development
  - C++ for compute-intensive tasks
  - Python for ease-of-development and scripting
  - Gnuradio-companion GUI for applications



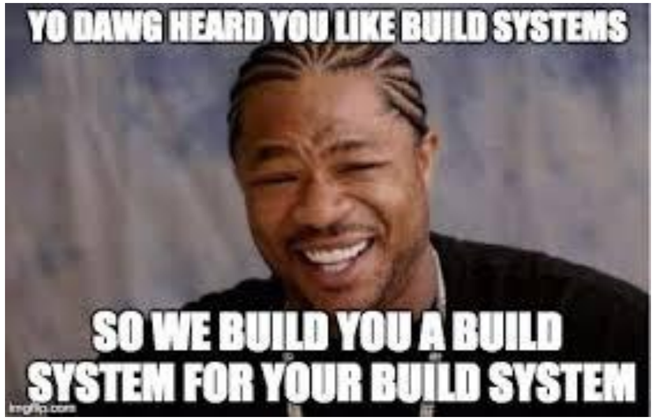
- All signal processing applications
  - Why reinvent scheduling?
  - Why reinvent item tags?
  - Why reinvent Gnuradio companion?
  - … etc

Engineers learn Gnuradio (or start having prior experience) and gain expertise through documentation, tutorials, and community involvement

HawkEye 360

©2018 HawkEye 360

- Goal: Deliver reliable prebuilt software and FPGA images – when software and FPGAs are changing daily
- Requirements:
  - Multi-repo builds (>> 2 repos)
  - Cross-compile software for 3+ embedded targets
  - Version all output artifacts
  - Ability to immediately rollback changes
  - Small installation size
  - Nightly builds
  - Nightly tests: unit tests per repo, and full integration tests on embedded devices

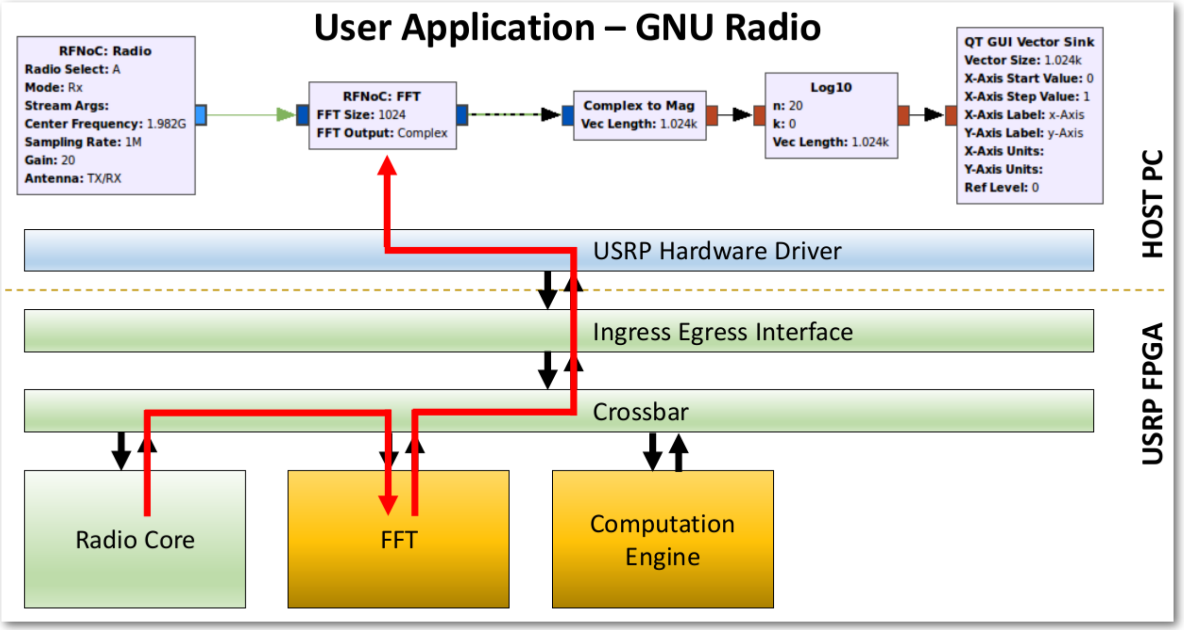


Some things to consider

©2018 HawkEye 360

RFNoC Applications

- Baseband processing supplied with UHD
  - Upsampling/downsampling
  - FFTs
  - Gnuradio flowgraphs via gr-ettus
- More processing applications...
  - Complex ambiguity function
  - Signal detection (squelch, other algorithms)
  - Polyphase channelizer
  - OQPSK modem
  - Full-rate data transfer to processor
  - ... machine learning?



Use the FPGA to intelligently downsample higher bandwidths than the Zynq's ARM can process in software (> 2 MHz or so)

HawkEye 360

©2018 HawkEye 360

RFNoC in Production

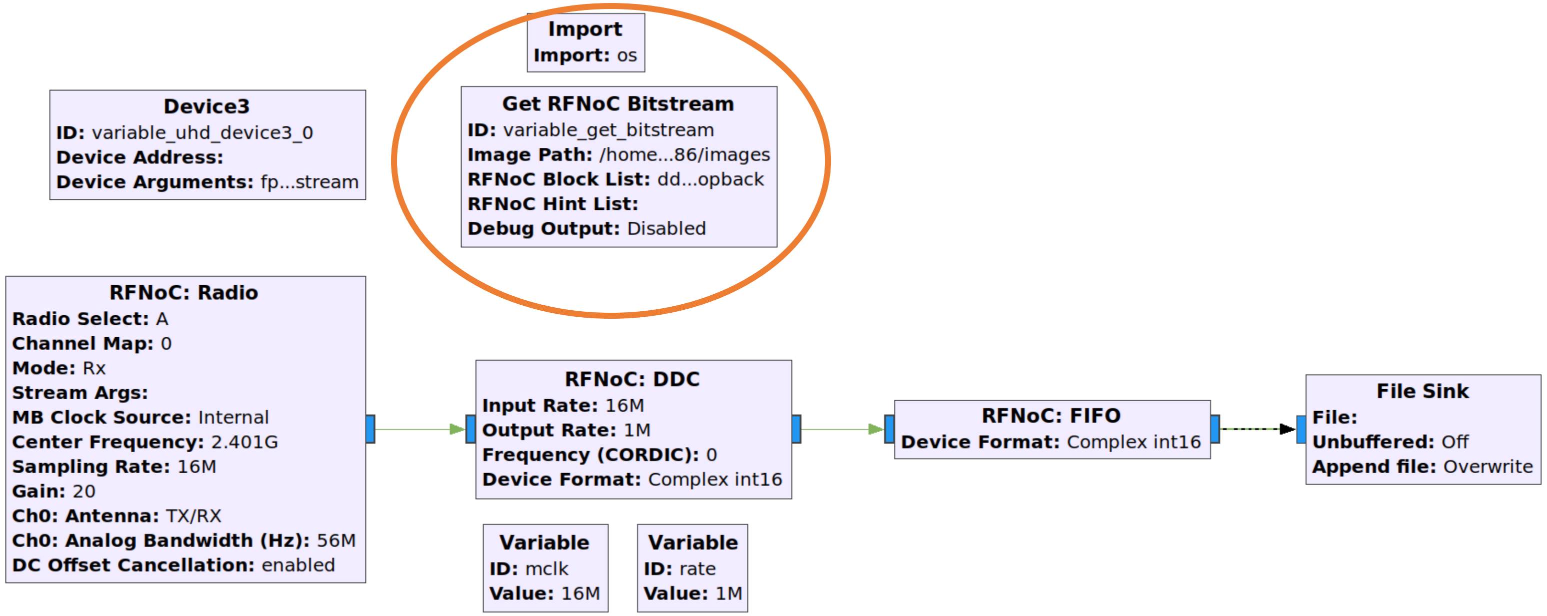
- Goal: Reliable, repeatable FPGA builds for multiple targets with many varieties of RFNoC compute engines
- Build system
  - Use yml files to define images; Iterate over many images (10+); build and export all images
- Packaging & Run-Time
  - The gnuradio build maintains FPGA image dependencies and deploys images with software
  - When running Gnuradio applications, dynamically parse YML files to select a compatible image
- Other helpful RFNoC updates
  - Read user registers when debugging
  - Transfer gnuradio PDUs directly into and out of RFNoC (no intermediate streaming data required)

Works extremely well

©2018 HawkEye 360

HawkEye 360

- Gnuradio block "Get RFNoC Bitstream" dynamically chooses the right bitstream based on the required RFNoC blocks



1. User does not have to manually identify a compatible bitstream
2. Run the exact same Gnuradio flowgraph on different devices

©2018 HawkEye 360

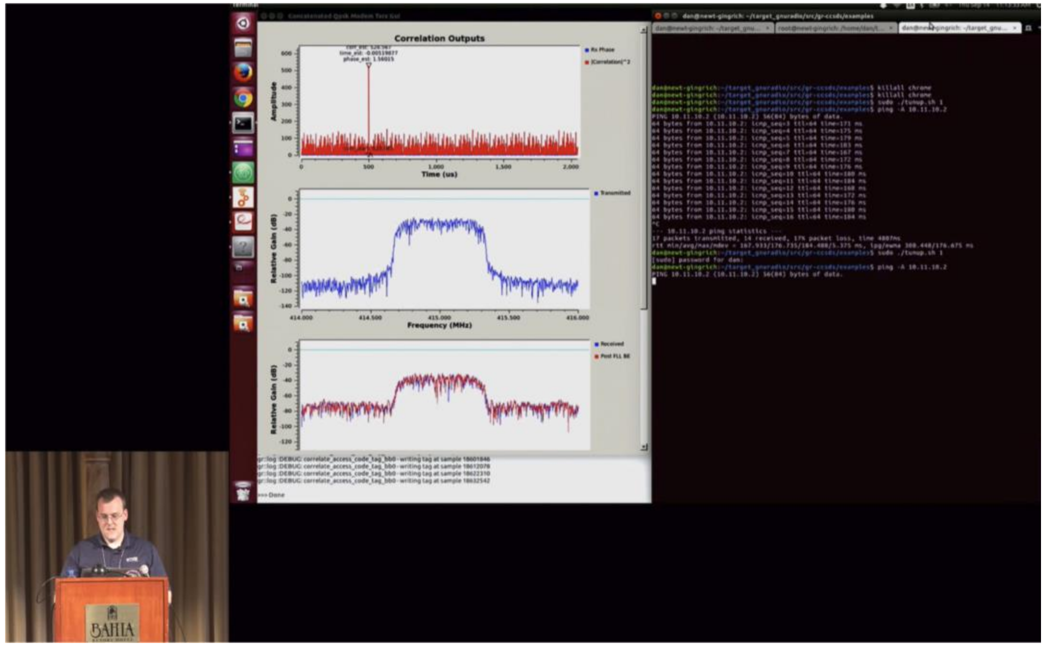
Use-Case: OQPSK Communications System

HawkEye 360

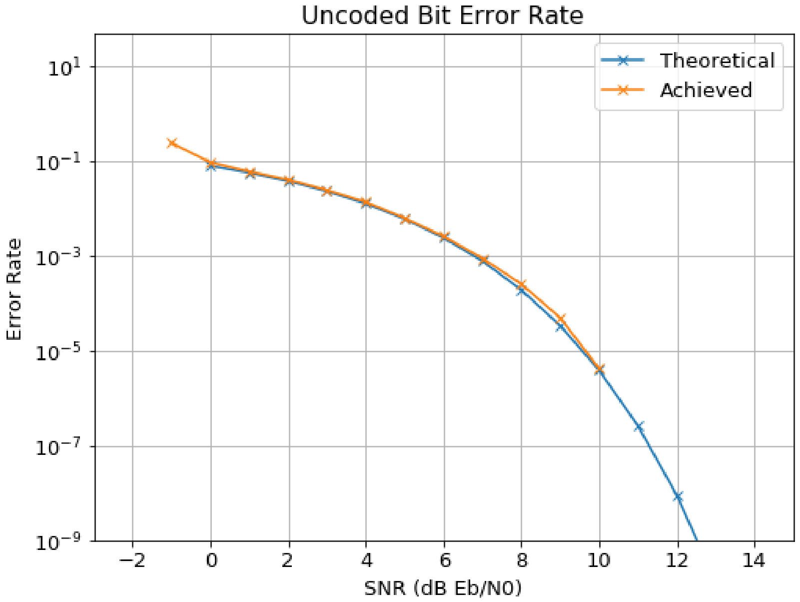
©2018 HawkEye 360

- Payload provides S-band uplink receiver (demodulator + decoder)
- Payload provides comms downlink coder

Status in 2017:
Dan has a prototype using Gnuradio blocks
Demo at GRCon 2017
No coder/decoder

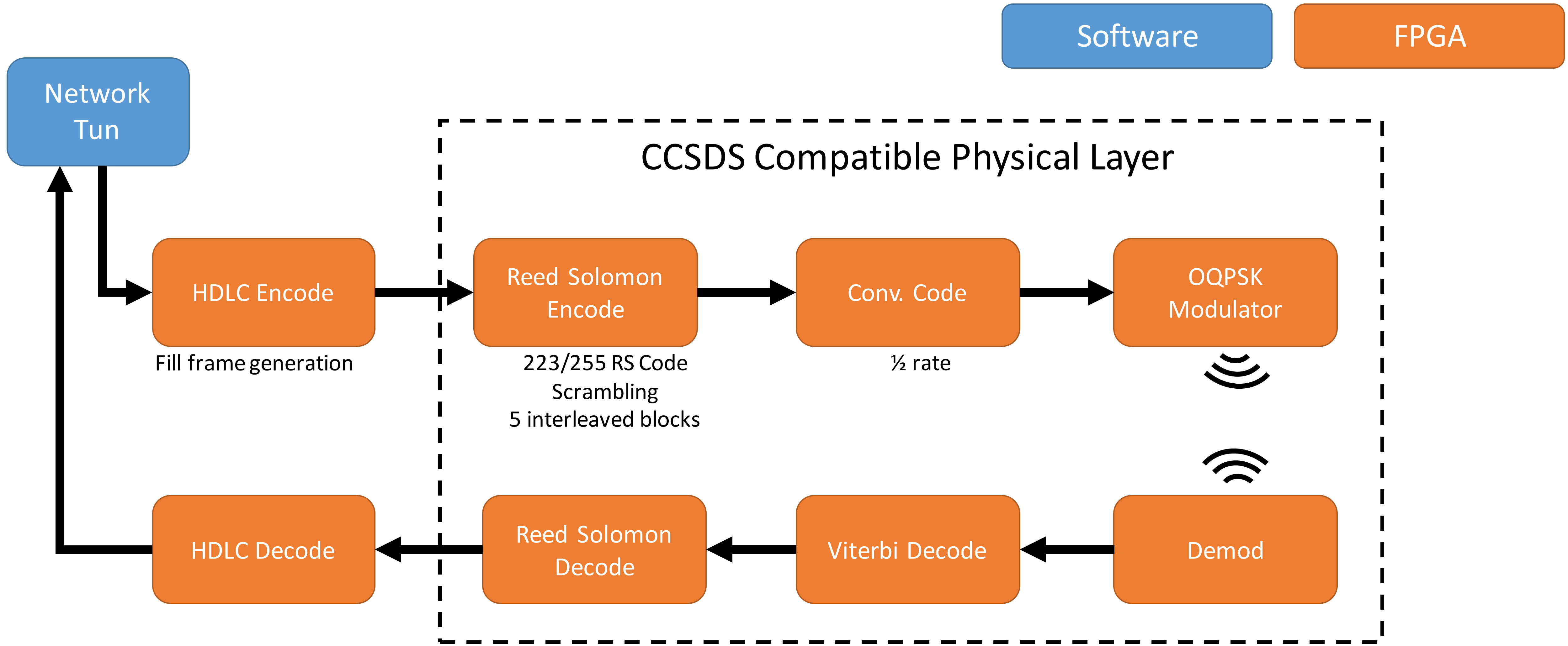


Status in 2018:
Implemented and deployed using RFNoC
Full codec and modem
Compatible with industry standards
2 Mbps uplink/50 Mbps downlink



©2018 HawkEye 360

Payload Communications



System Architecture

©2018 HawkEye 360

Implemented with feed-forward stages for ease of development and testing

5x oversampled → **Timing Recovery** → 2x oversampled → **Frequency Recovery** → 2x oversampled → **OQPSK Phase Locked Loop** → 1 sample per symbol

Timing Recovery

- Maximize sample alignment
- Independent of frequency offsets
- Accurate timing recovery improves downstream freq & phase recovery

Frequency Recovery

- Calculate FFT
- Find FFT peak
- Adjust input signal by FFT peak frequency

OQPSK Phase Locked Loop

- Maximize real[n]*imag[n-1]
- Results in several ambiguity points
- "Synchronization Techniques for Digital Receivers", Mengali and D'Andrea

Non-Data-Aided Demodulator

©2018 HawkEye 360

HawkEye 360

1. Prototype reference algorithms in **python** (1-3 weeks)

2. Implement algorithms in **C/C++** using Vivado HLS

(1-2 months)

3. **Verilog** FPGA wrappers. Synthesize and test using RFNoC testbenches

(1-2 months)

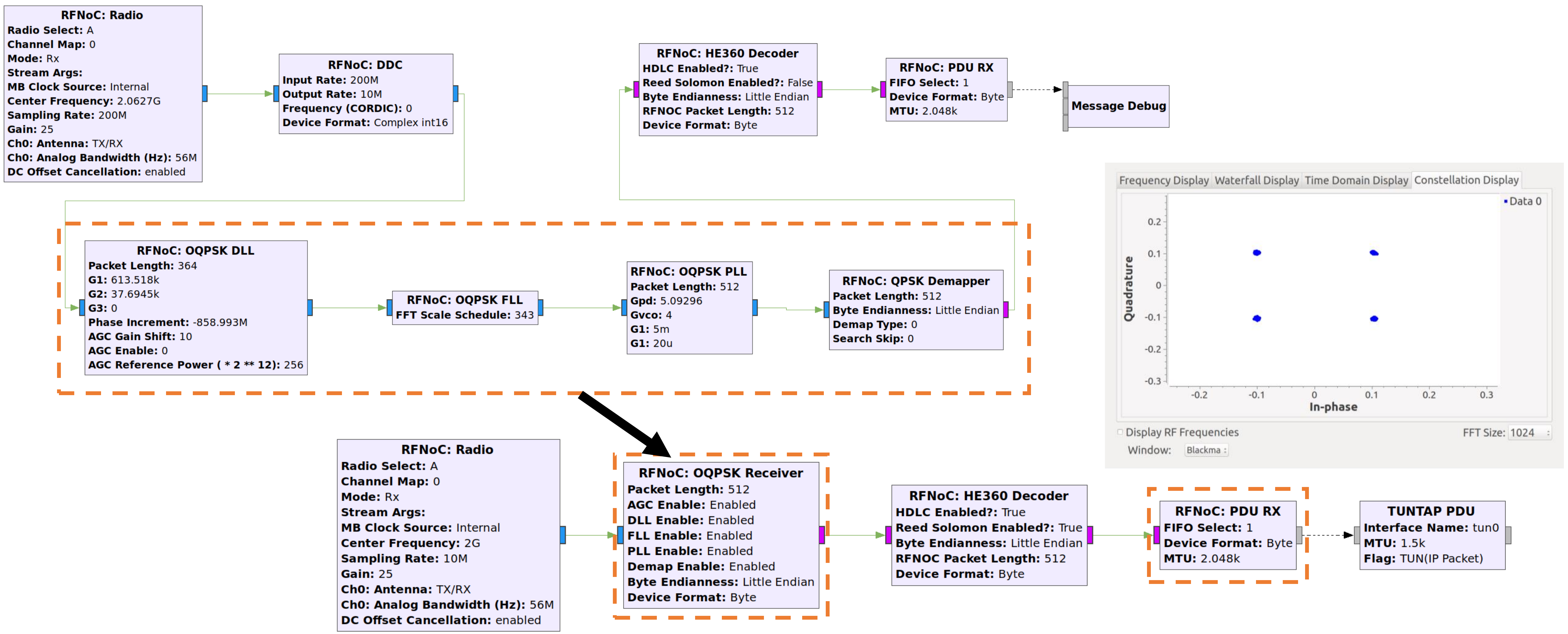
4. Test components on X310

5. Optimize and deploy to E310 (2 weeks)

Development Process

©2018 HawkEye 360

HawkEye 360



Testing on X310 provides component-level validation—

Then we condense and integrate onto E310

©2018 HawkEye 360

HawkEye 360

►1. RFNoC Register Logger

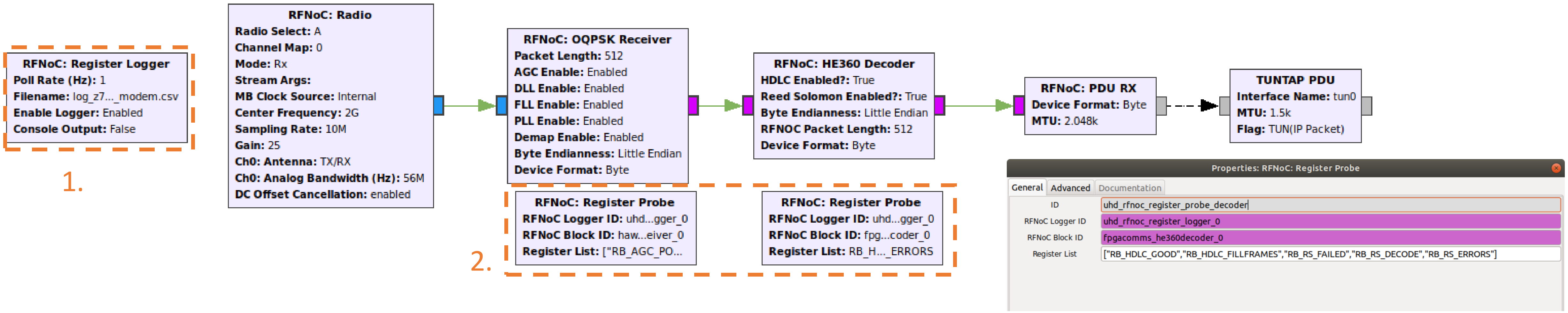
- Spawns a thread that queries RFNoC registers
- Write to console and/or CSV file

►2. RFNoC Register Probe

- Indicates which registers to read for each RFNoC block

```
Querying RFNoC blocks...
0/syrlinks_0/RB_TX_HDLC_PACKETS: 1314
0/syrlinks_0/RB_TX_HDLC_BYTES: 2693514
0/syrlinks_0/RB_TX_HDLC_FILLFRAMES: 32049
0/syrlinks_0/RB_TX_RS_BLOCKS: 12248
0/syrlinks_0/RB_TX_RS_BYTES: 3124179

Querying RFNoC blocks...
0/syrlinks_0/RB_TX_HDLC_PACKETS: 2646
0/syrlinks_0/RB_TX_HDLC_BYTES: 5421139
0/syrlinks_0/RB_TX_HDLC_FILLFRAMES: 32049
0/syrlinks_0/RB_TX_RS_BLOCKS: 24501
0/syrlinks_0/RB_TX_RS_BYTES: 6248614
```



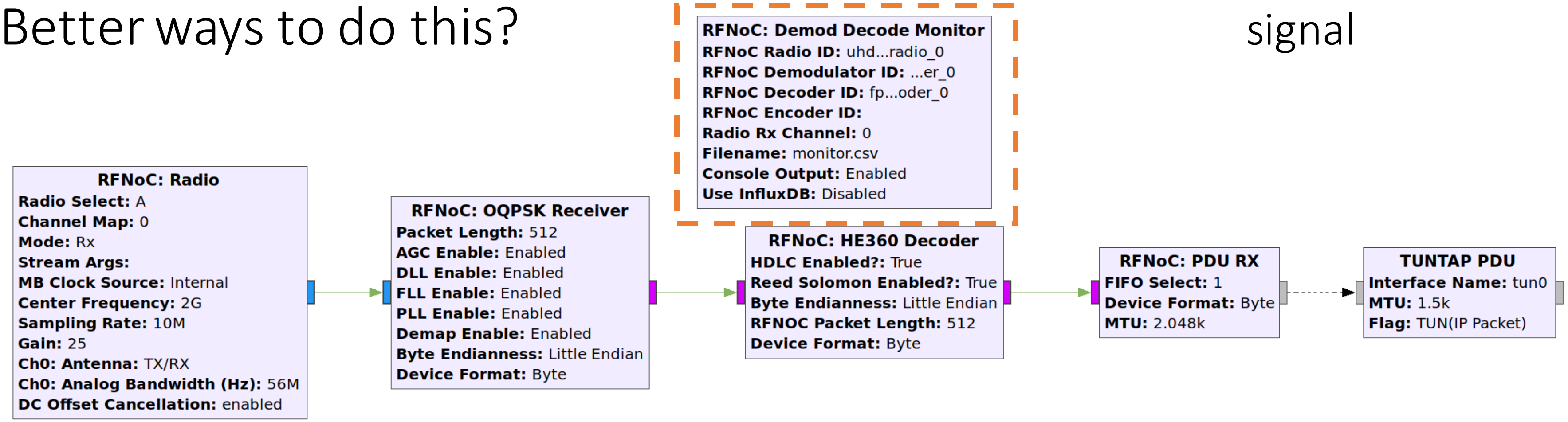
RFNoC "Register Probe" blocks provide a convenient way to access FPGA registers in real time

©2018 HawkEye 360

- State machine has pointers to RFNoC blocks
- Register read/writes to coordinate behavior:
  - Set frontend AGC based on estimated received power
  - Reset demodulator components
  - Confirm decoder locks
- Inquiry: Better ways to do this?

AGC Algorithm:

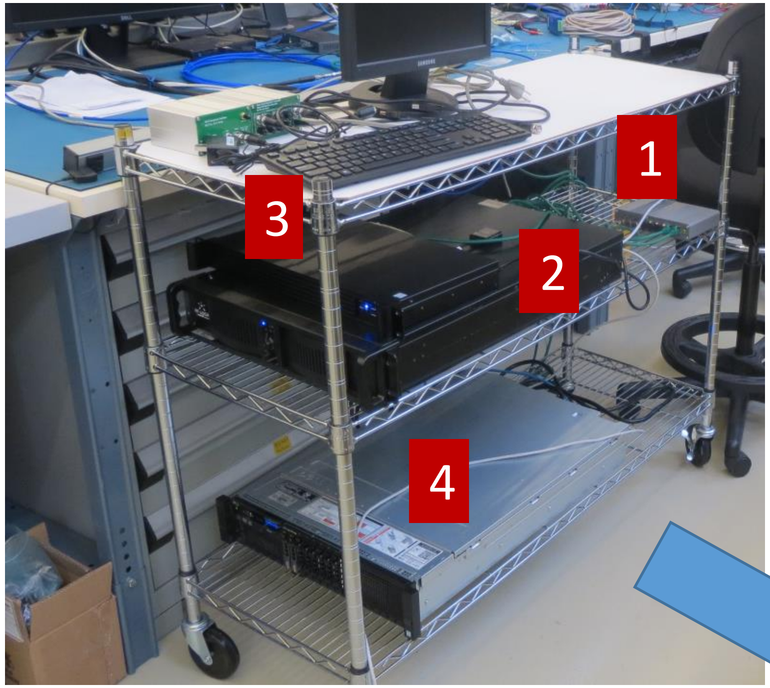
1. **While power is saturated high or low**: Binary search
2. **After power observed**: Lock onto signal
3. **If at max gain**: Declare success, but be ready to re-lock if we see a signal



Python-based state machine performs hardware AGC

©2018 HawkEye 360

Payload Communications



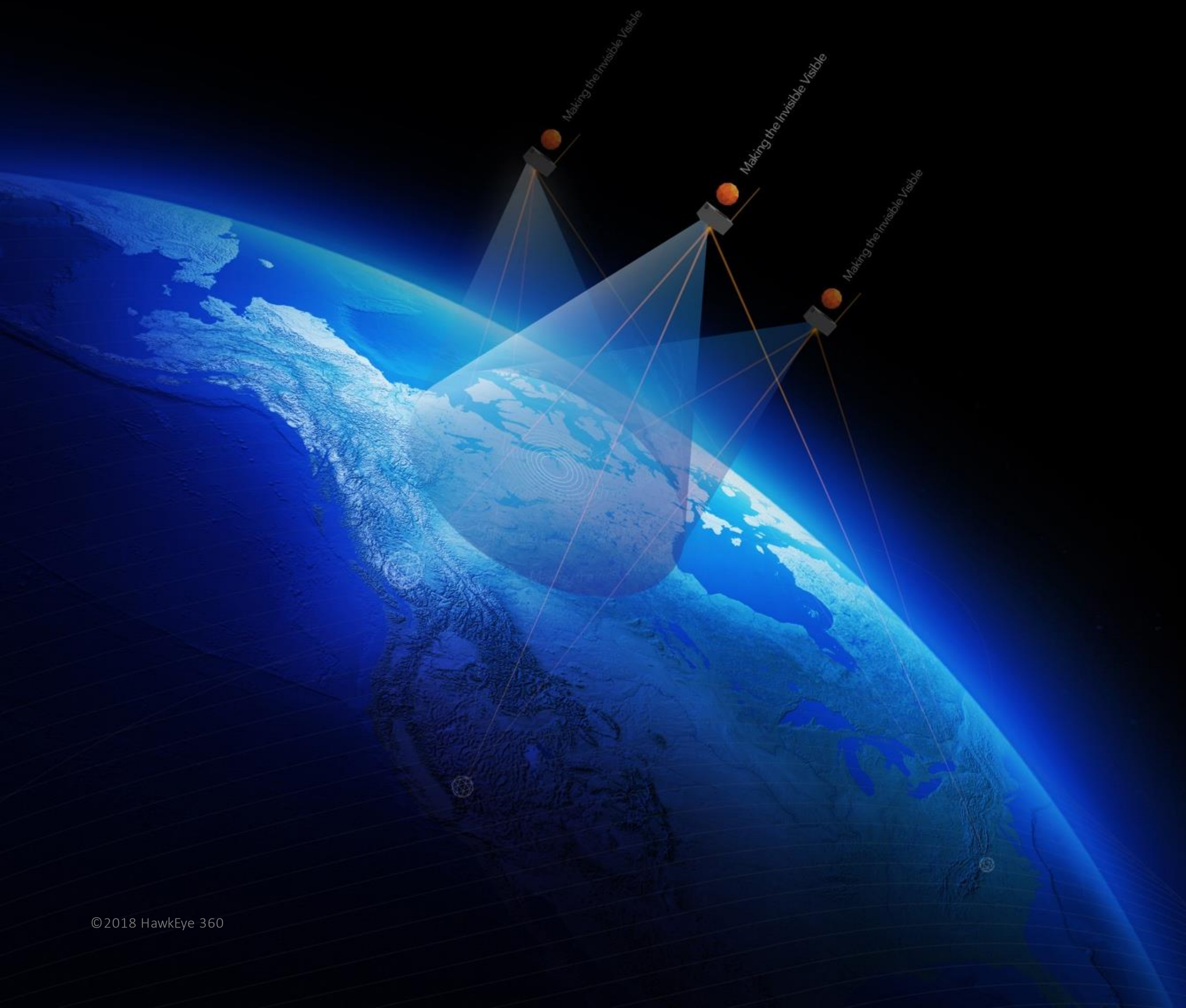

```
------------------------------------------------------------
Server listening on 5201
------------------------------------------------------------
Accepted connection from 192.168.200.1, port 42274
[  5] local 192.168.200.2 port 5201 connected to 192.168.200.1 port 47037
[ ID] Interval           Transfer     Bandwidth       Jitter    Lost/Total Datagrams
[  5]   0.00-1.00   sec  4.67 MBytes  39.2 Mbits/sec  0.322 ms  0/3380 (0%)
...
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Jitter    Lost/Total Datagrams
[  5]   0.00-30.97  sec   150 MBytes  40.7 Mbits/sec  0.281 ms  0/108769 (0%)
```

OR

OR

Rapid development, deployment, reconfiguration = Success

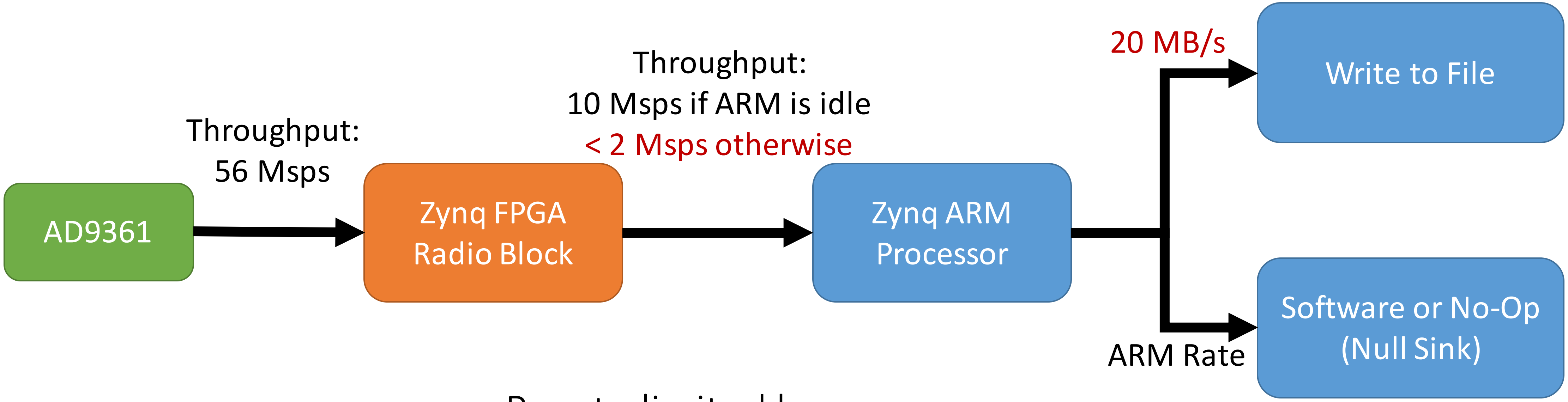
©2018 HawkEye 360



Use-Case: Full Rate Data Captures on Zynq

©2018 HawkEye 360

- E310 has trouble recording > 2ish Msps with RFNoC
- AD9361 can digitize 56 Msps
- Analog devices IIO gets full rate (56 Msps)??



Rx rate limited by:

1. Hard drive (i.e. SD card) write speed
2. FPGA-to-Processor transfer speed (DMA rate)

©2018 HawkEye 360

HawkEye 360

- Possible solutions:
  - Use E310 FPGA DRAM as a FIFO
  - **Faster FPGA to Processor transfers**
- How to increase transfer speed?
  - Dedicated DMA
  - Bigger data packets
  - Less processor activity

Zynq can theoretically support 600 MB/s!

Table 22-2: **Theoretical Bandwidth of PS-PL and PS Memory Interfaces**

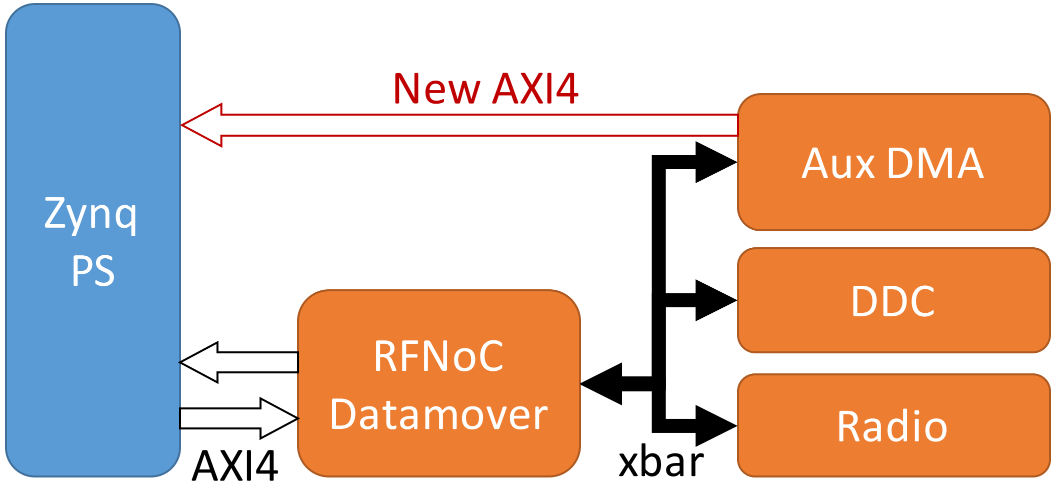
Interface	Type	Bus Width (bits)	IF Clock (MHz)	Read Bandwidth (MB/s)	Write Bandwidth (MB/s)	R+W Bandwidth (MB/s)	Number of Interfaces	Total Bandwidth (MB/s)
General Purpose AXI	PS Slave	32	150	600	600	1,200	2	2,400
General Purpose AXI	PS Master	32	150	600	600	1,200	2	2,400
High Performance (AFI) AXI_HP	PS Slave	64	150	1,200	1,200	2,400	4	9,600
AXI _ACP	PS Slave	64	150	1,200	1,200	2,400	1	2,400
DDR	External Memory	32	1,066	4,264	4,264	4,264	1	4,264
OCM	Internal Memory	64	222	1,779	1,779	3,557	1	3,557

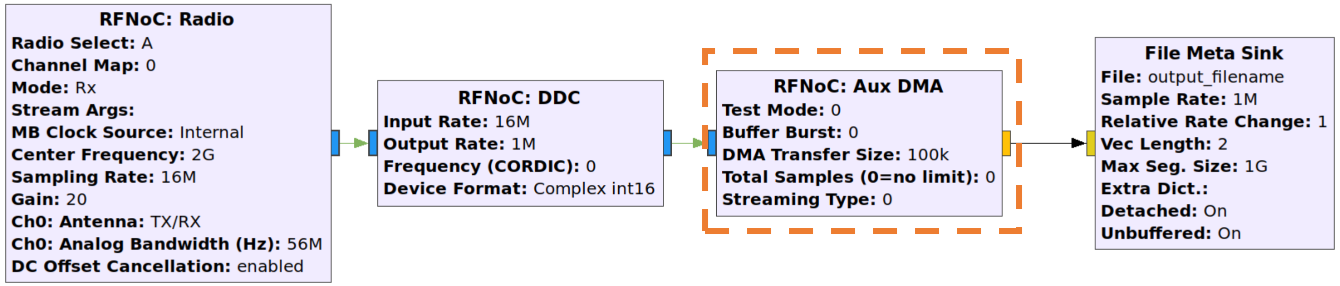
We created a new DMA operation that provides an alternate entrypoint into Gnuradio

©2018 HawkEye 360

Major Surgery:

- Aux DMA: noc_block_auxdma connects directly to the E310 processor via AXI4 DMA
- New kernel driver services DMA interrupts
- Reuse typical RFNoC register reads/writes and graph infrastructure
- Aux DMA overrides "general_work" function of rfnoc_block_impl (in gr-ettus)



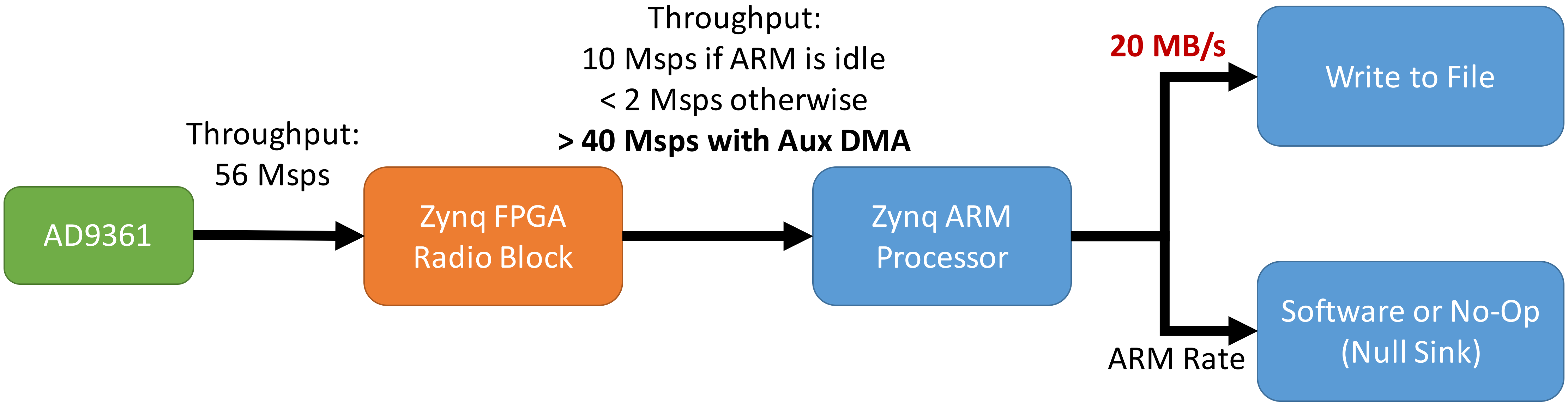


"Aux DMA" send data directly into shared processor DRAM using large packet sizes

©2018 HawkEye 360

Full Rate Data Transfers

- Throughput can sustain 40 Msps transfers to processor
- Requires more user interaction (knowledge of packet sizes, etc)
- Limiting factor is now hard drive write speed!



E310 and N310 series devices *can* transfer data between processor and FPGA at full rates – Some assembly required!

©2018 HawkEye 360

HawkEye 360

Summary

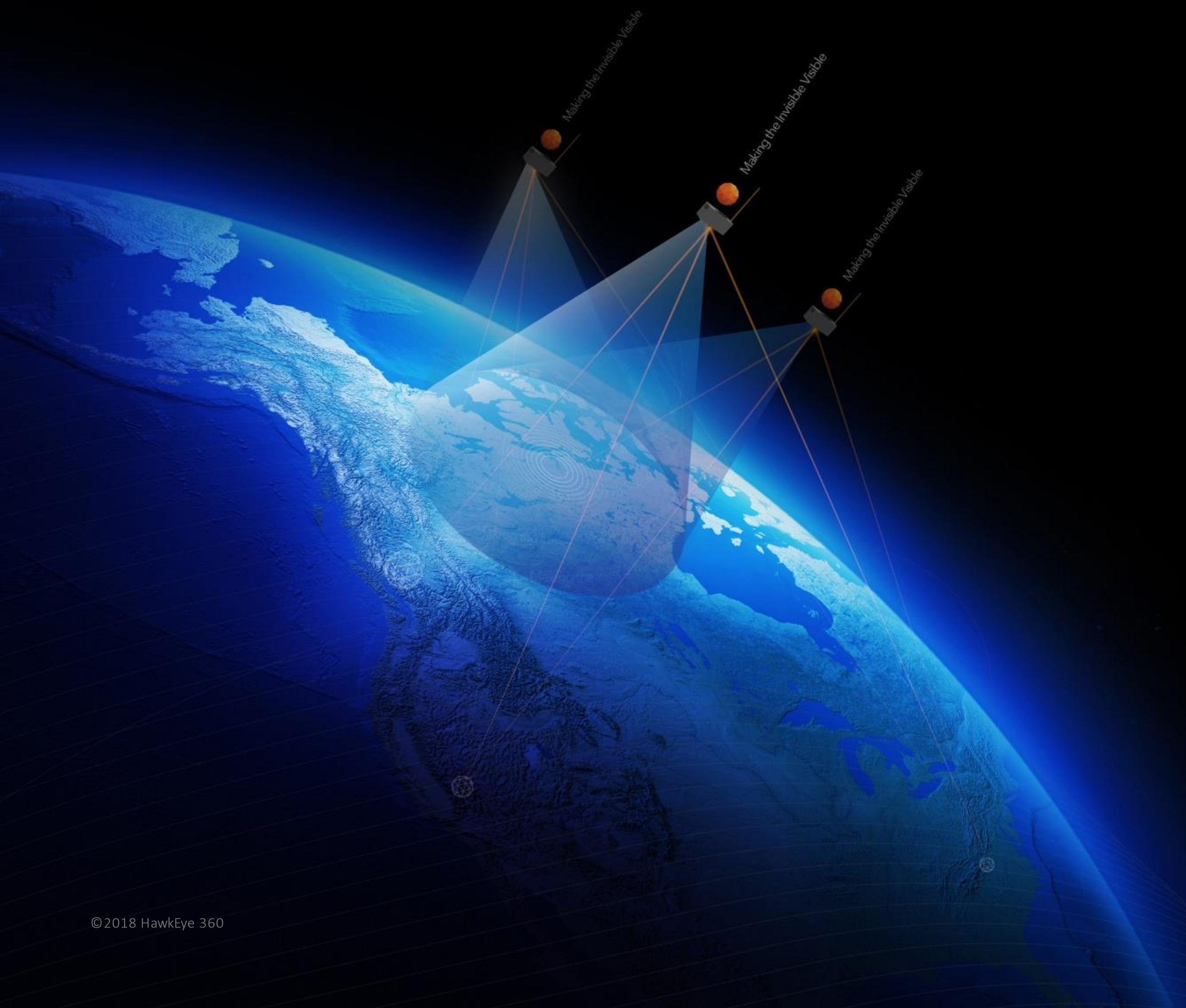
- Gnuradio + RFNoC improve development time and provide standardized development frameworks
- Some modifications to use Gnuradio and RFNoC in production
- With a working comms system, full-rate recordings, and more launch-ready applications, Hawkeye 360 will begin operations this year running SDR in LEO
- Happy to contribute to and help the community where we can

Hawkeye 360 is excited for our (literal) product launch.

Watch for updates in 2019!

HawkEye 360

©2018 HawkEye 360



Thank You

… and we're hiring!

EJ Kreinar

ej@he360.com

HawkEye 360

©2018 HawkEye 360