GNU Radio FPGA Acceleration with Xilinx Zynq

Jonathon Pendlum
jon.pendlum@gmail.com
Northeastern University

Mentor
Phillip Balister

Advisor
Professor Miriam Leeser

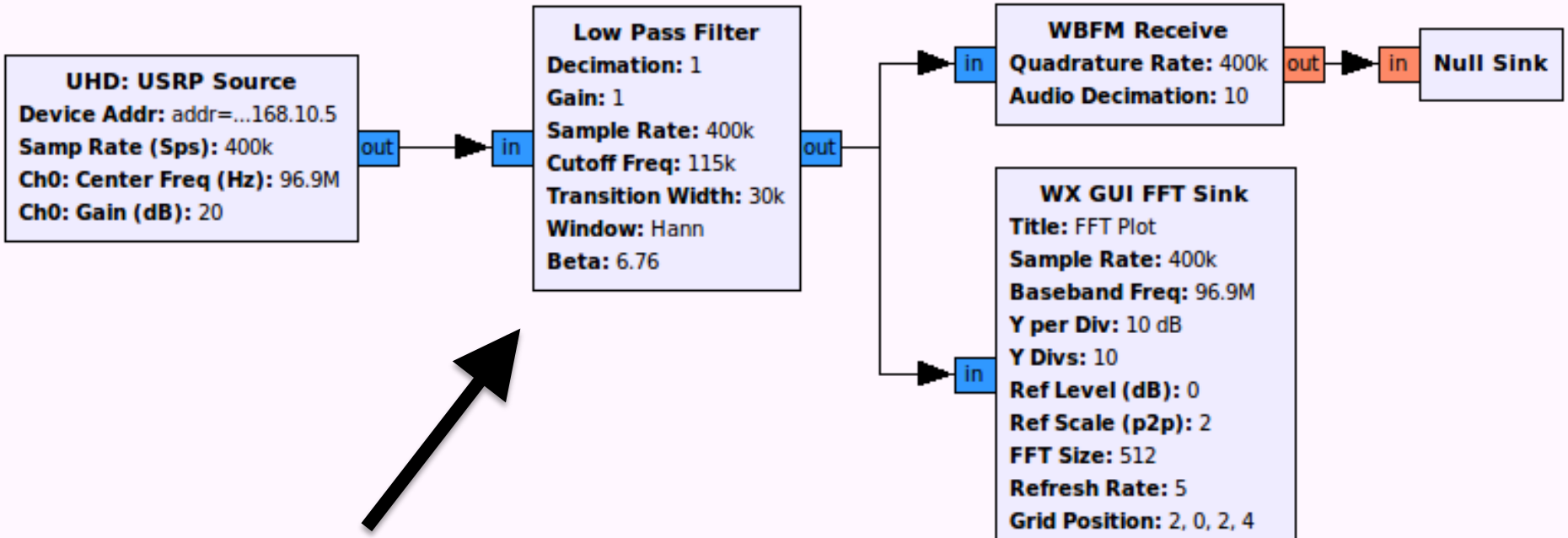
Recent Developments

- Why FPGA Acceleration in GNU Radio?
  - Faster performance for some algorithms
  - Frees processor to perform other tasks
  - Low latency, deterministic response time
- Xilinx Zynq – ARM + FPGA
  - Dual Core Cortex A9
  - Plentiful FPGA Resources
  - Tightly coupled via high speed buses
  - Zedboard, inexpensive development kit

Google Summer of Code

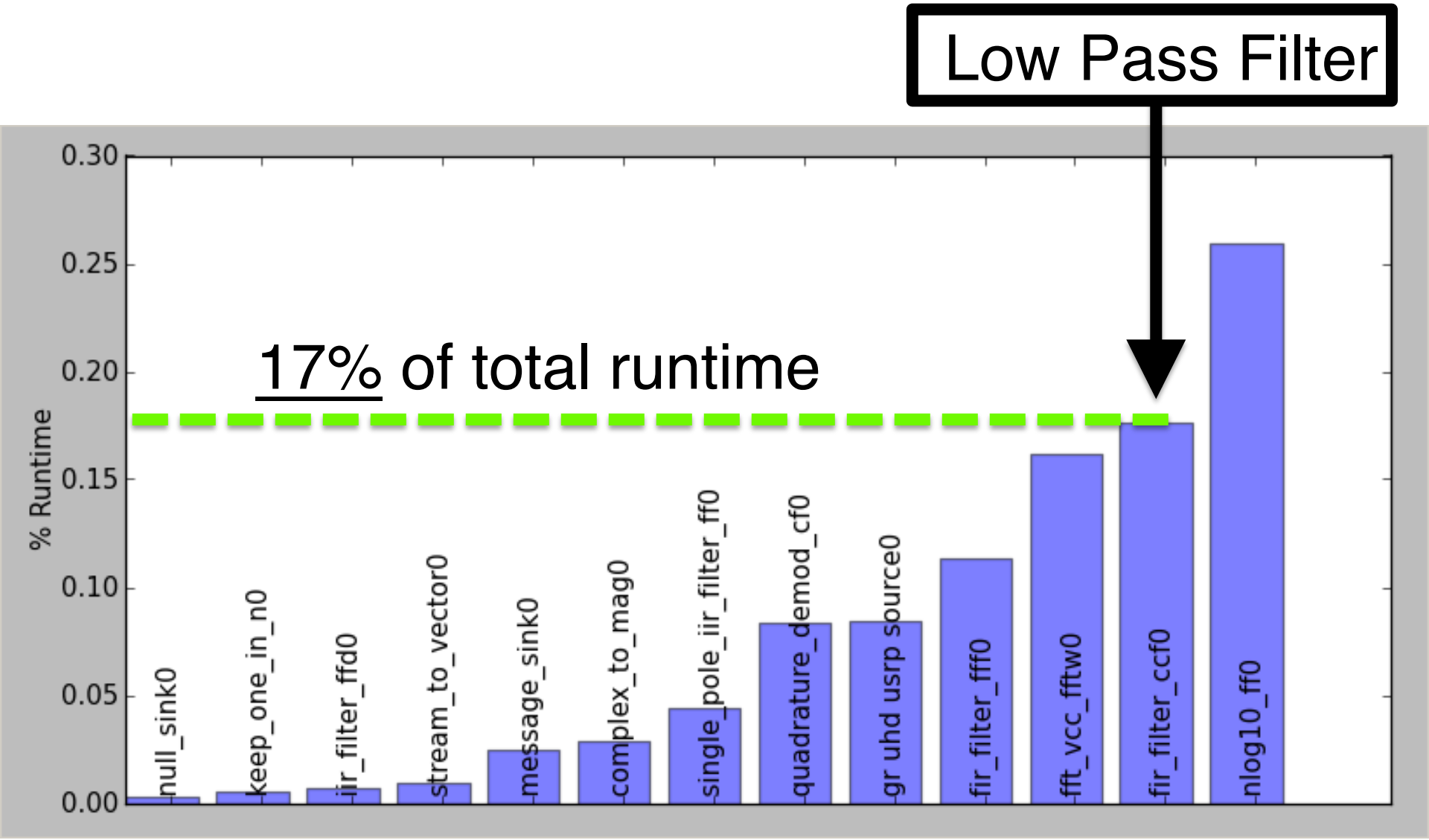
- Project Goals:
  - Run GNU Radio on the Zynq's ARM processors
  - Create a FPGA acceleration infrastructure
  - Demonstrate FPGA acceleration in GNU Radio
  - Provide comprehensive documentation

Wideband FM Example

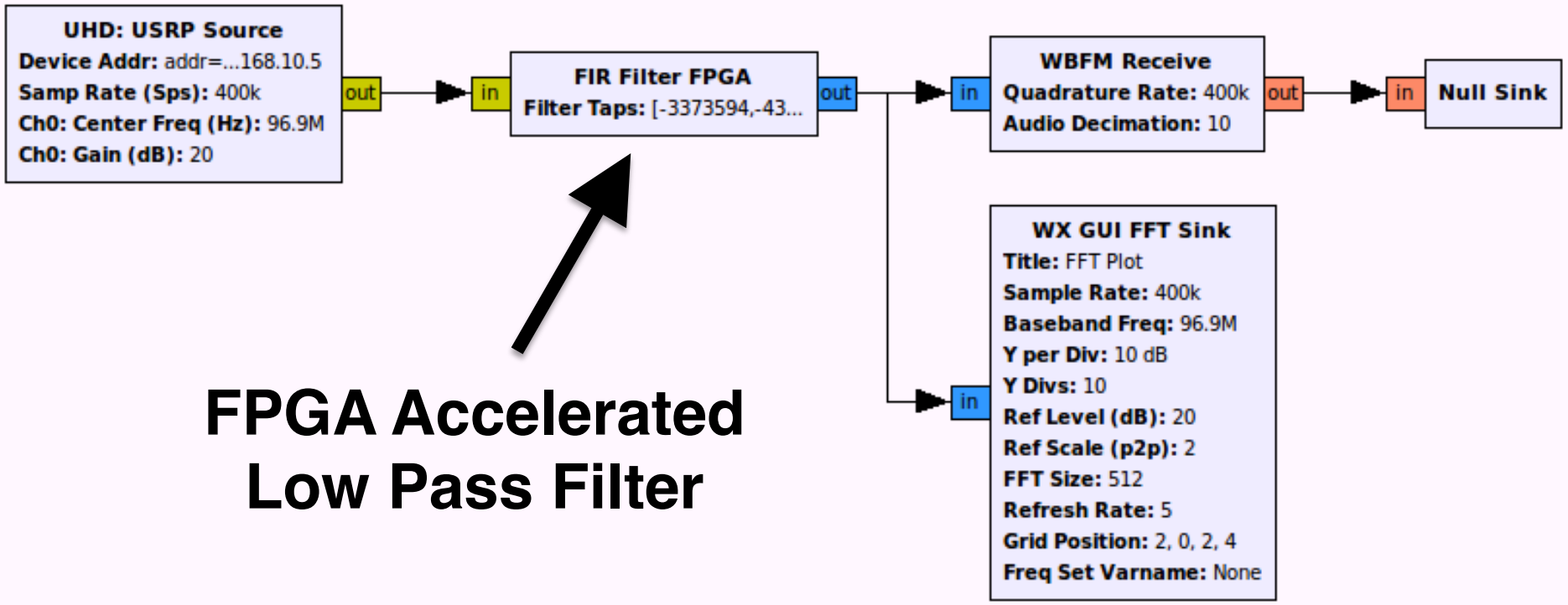


Low Pass Filter

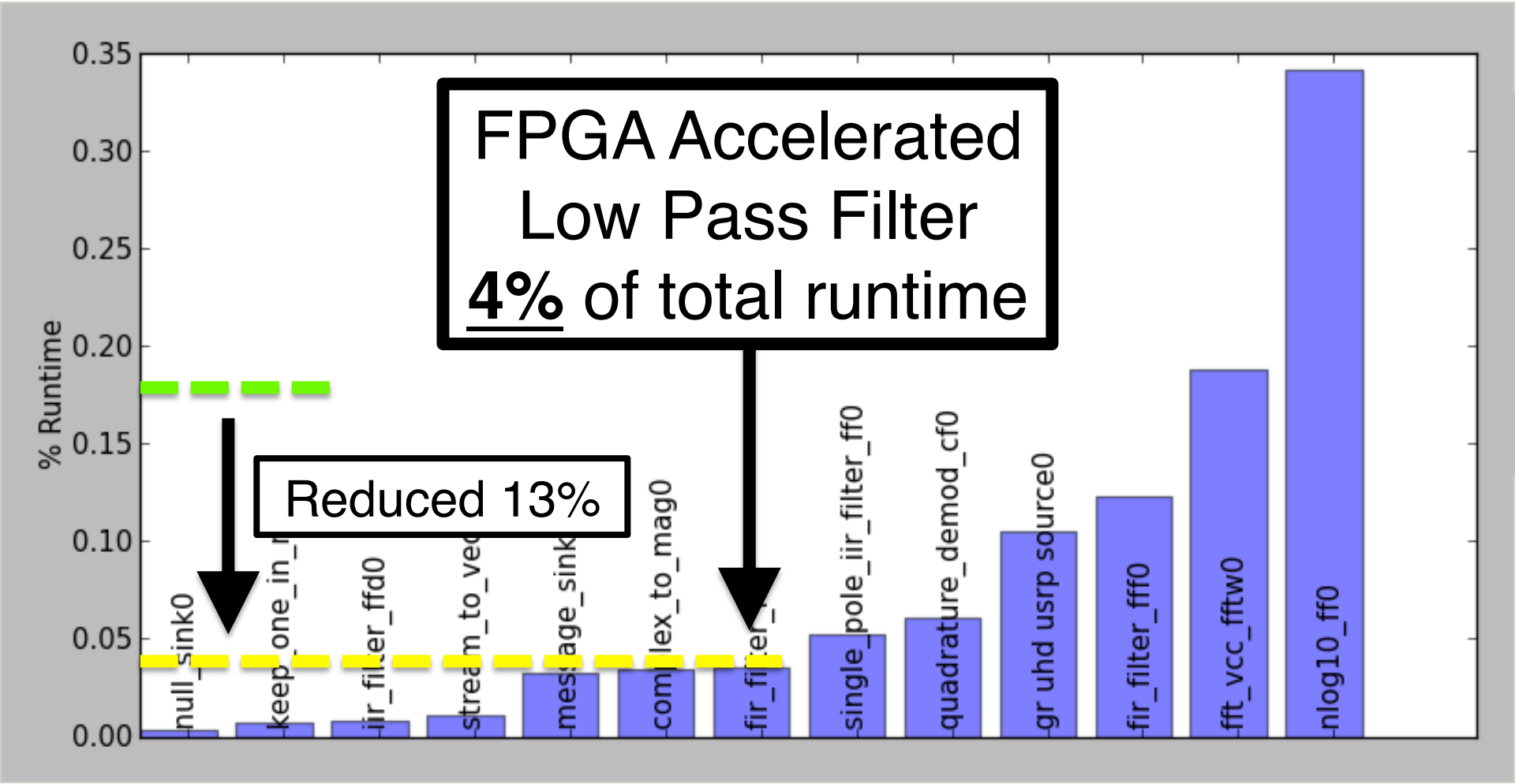
Runtime Performance on ARM



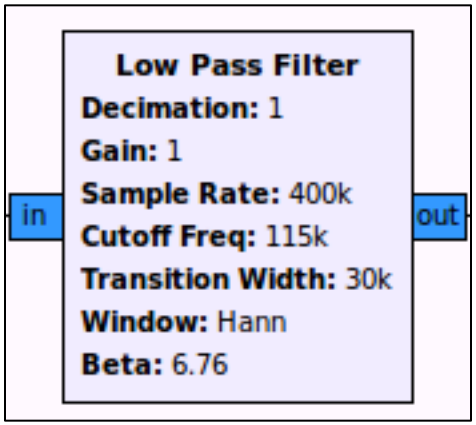
FPGA Accelerated Filter



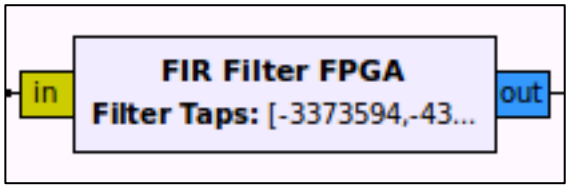
Runtime Performance Improvement



Performance Comparison

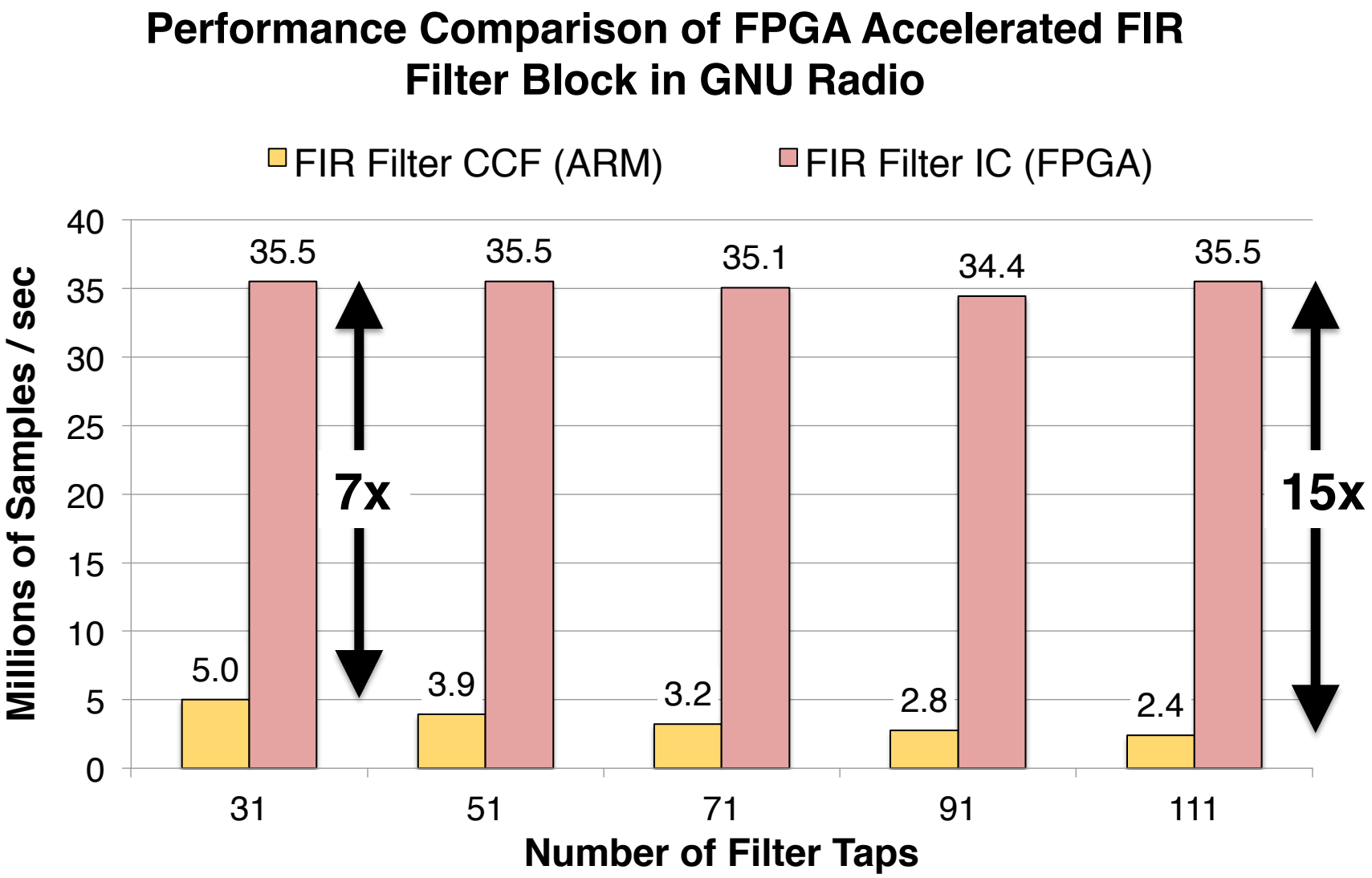


vs



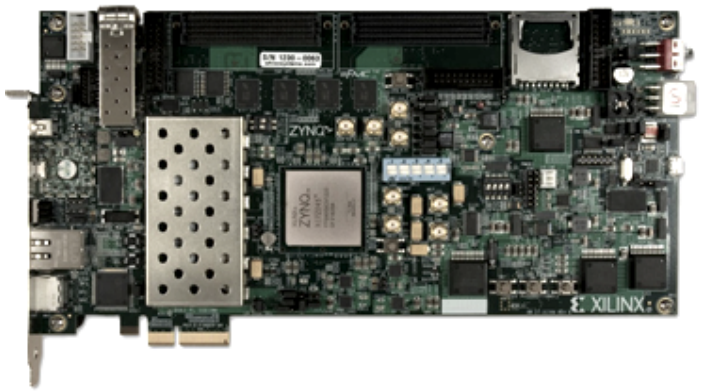
- Isolate block performance from GNU Radio
- Quantify effect of filter length on performance
- Wrote simple C++ program to measure each block's sample processing performance
  - gettimeofday() on work() method

Performance Comparison

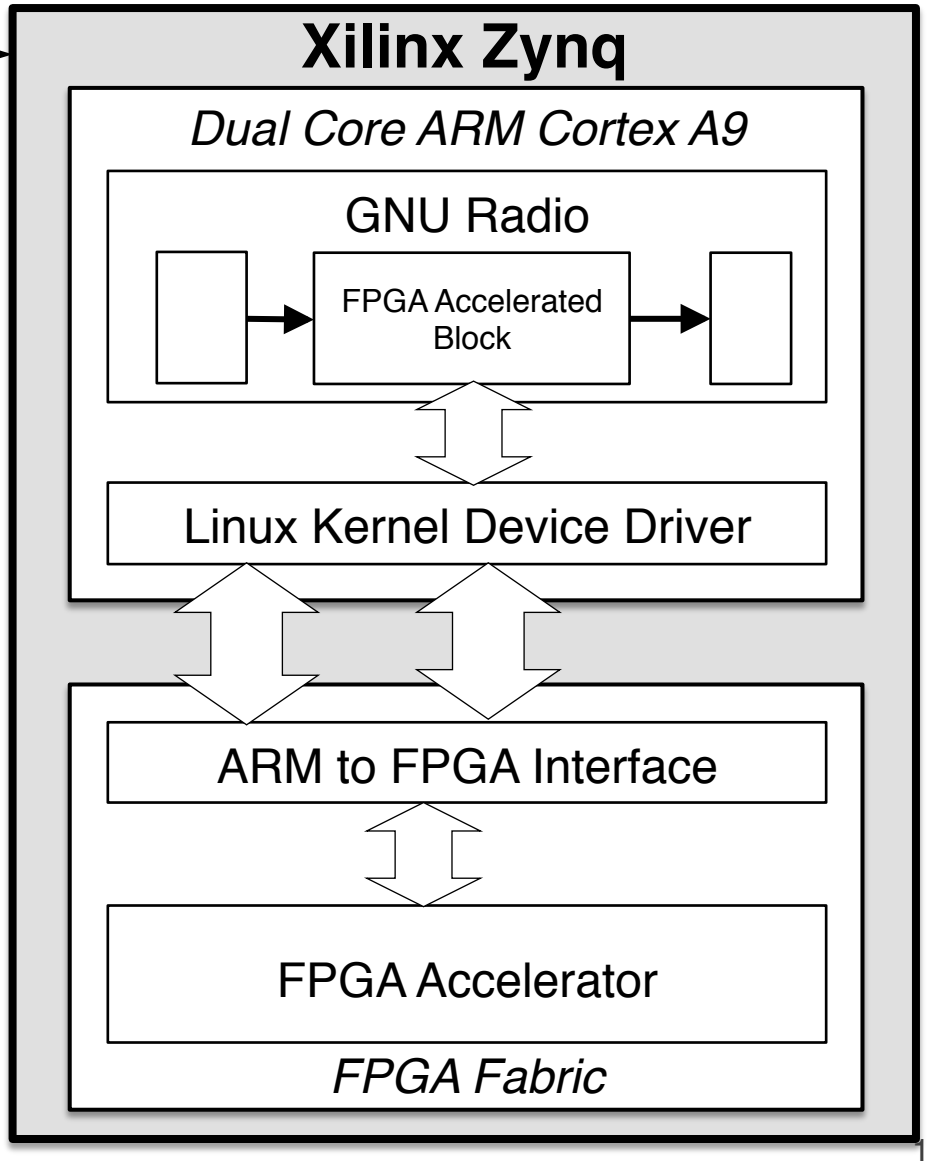


FPGA Accelerator Infrastructure

Hardware:
ZC706 Development Board



Others have used Zedboard & ZC702



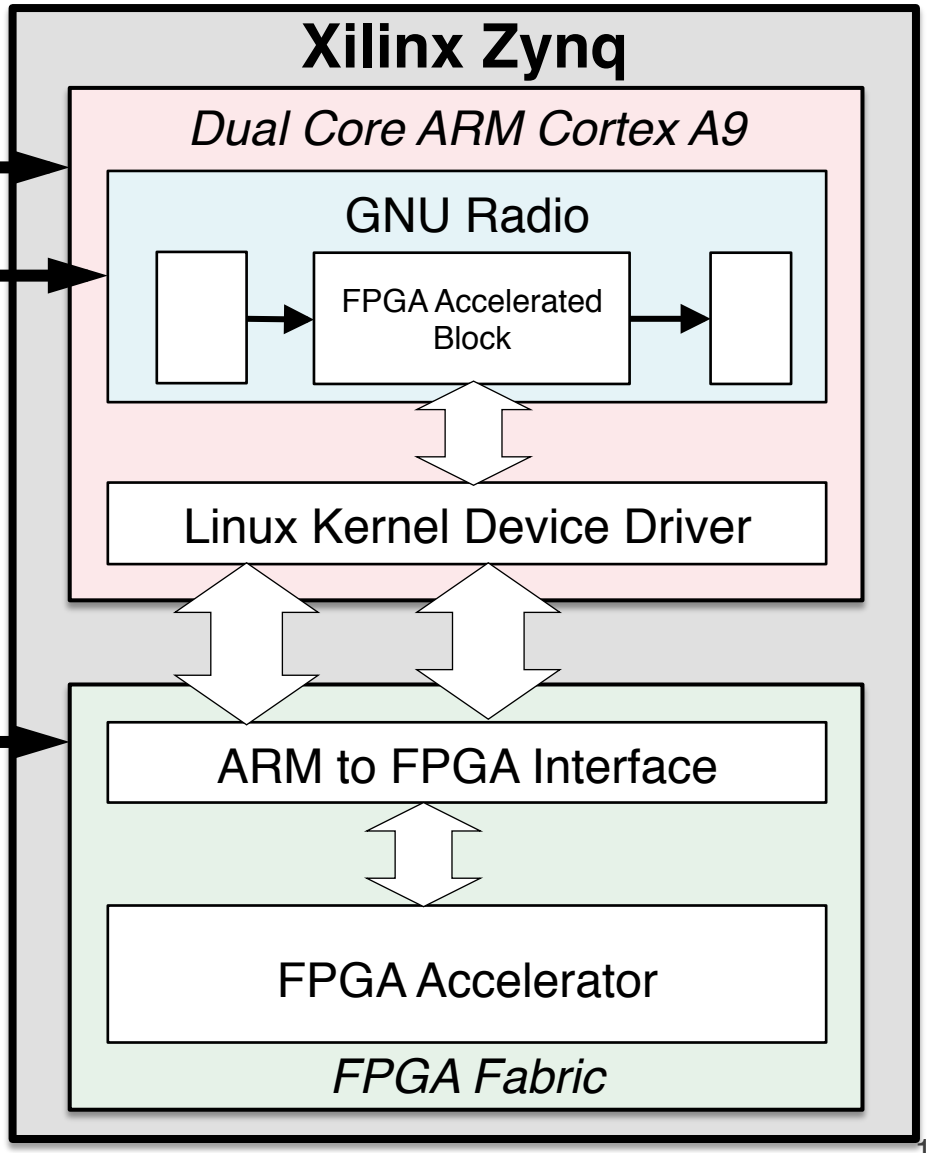
FPGA Accelerator Infrastructure

Linux 3.9, Ubuntu 13.04

GNU Radio 3.7.1

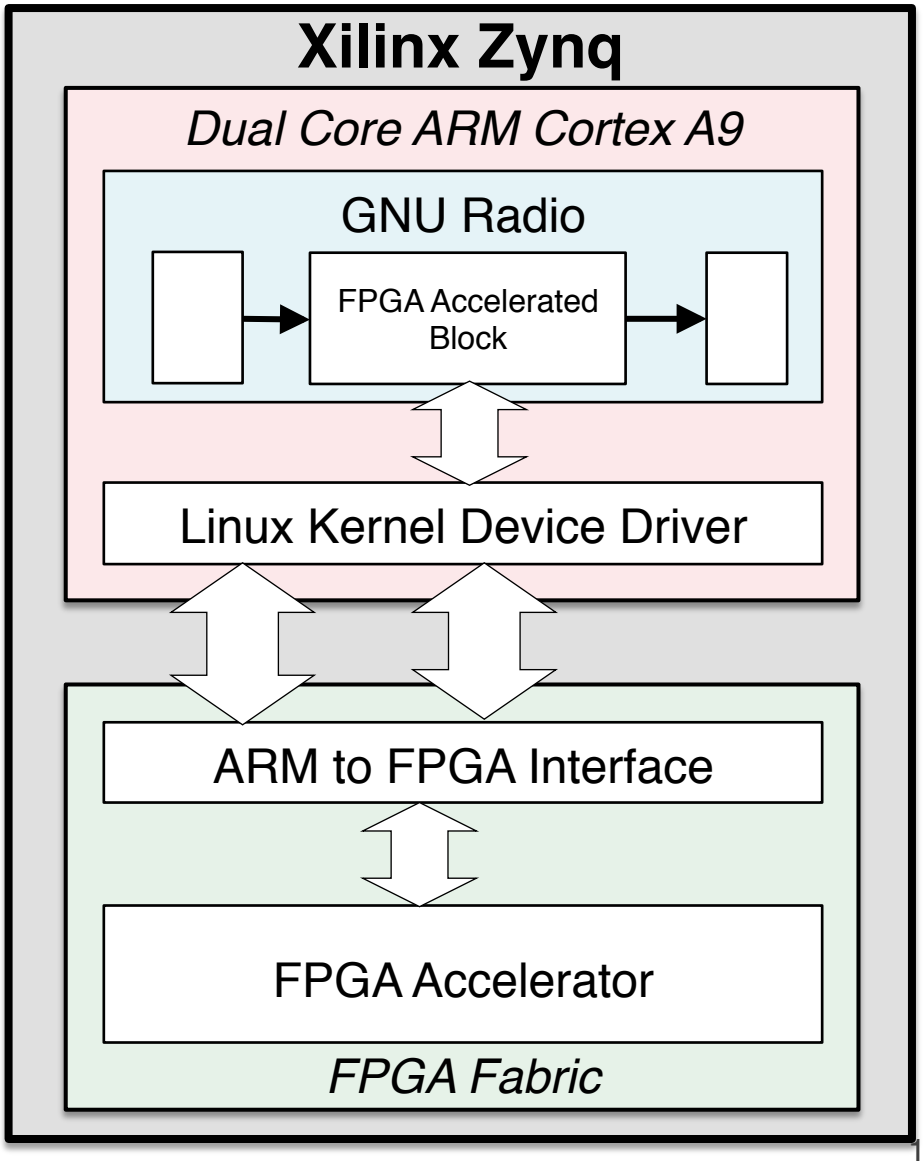
Xilinx ISE Design Suite 14.6

Setup information available on GNU Radio Zynq Wiki:
http://gnuradio.org/redmine/projects/gnuradio/wiki/Zynq



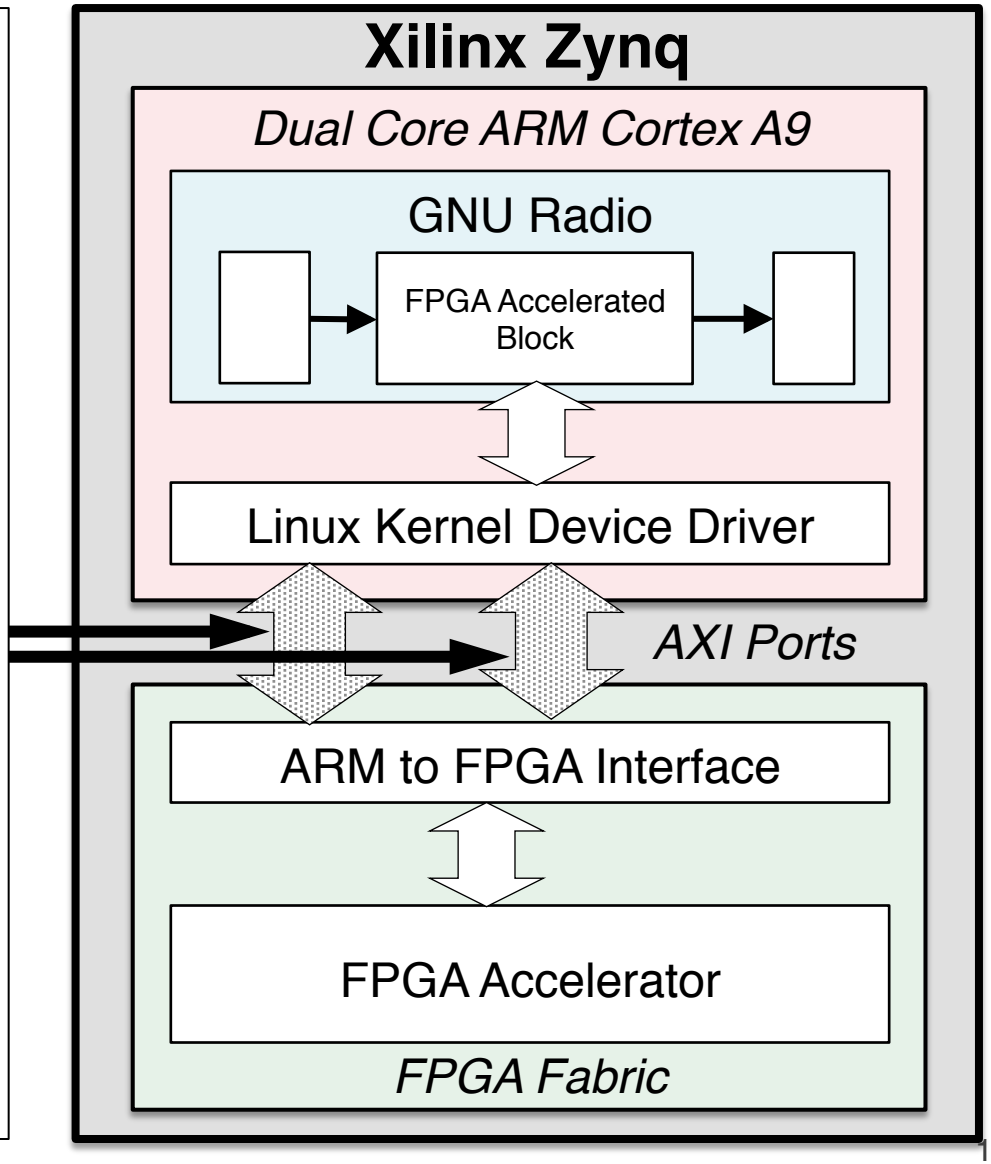
FPGA Accelerator Infrastructure

- **Goal**: Offload GNU Radio blocks to the FPGA
- **Requires**: Moving GNU Radio sample & control data between ARM / FPGA
- **Implement**:
  - Shared memory between ARM & FPGA
  - FPGA control interface
  - Accessible by GNU Radio Blocks



FPGA Accelerator Infrastructure

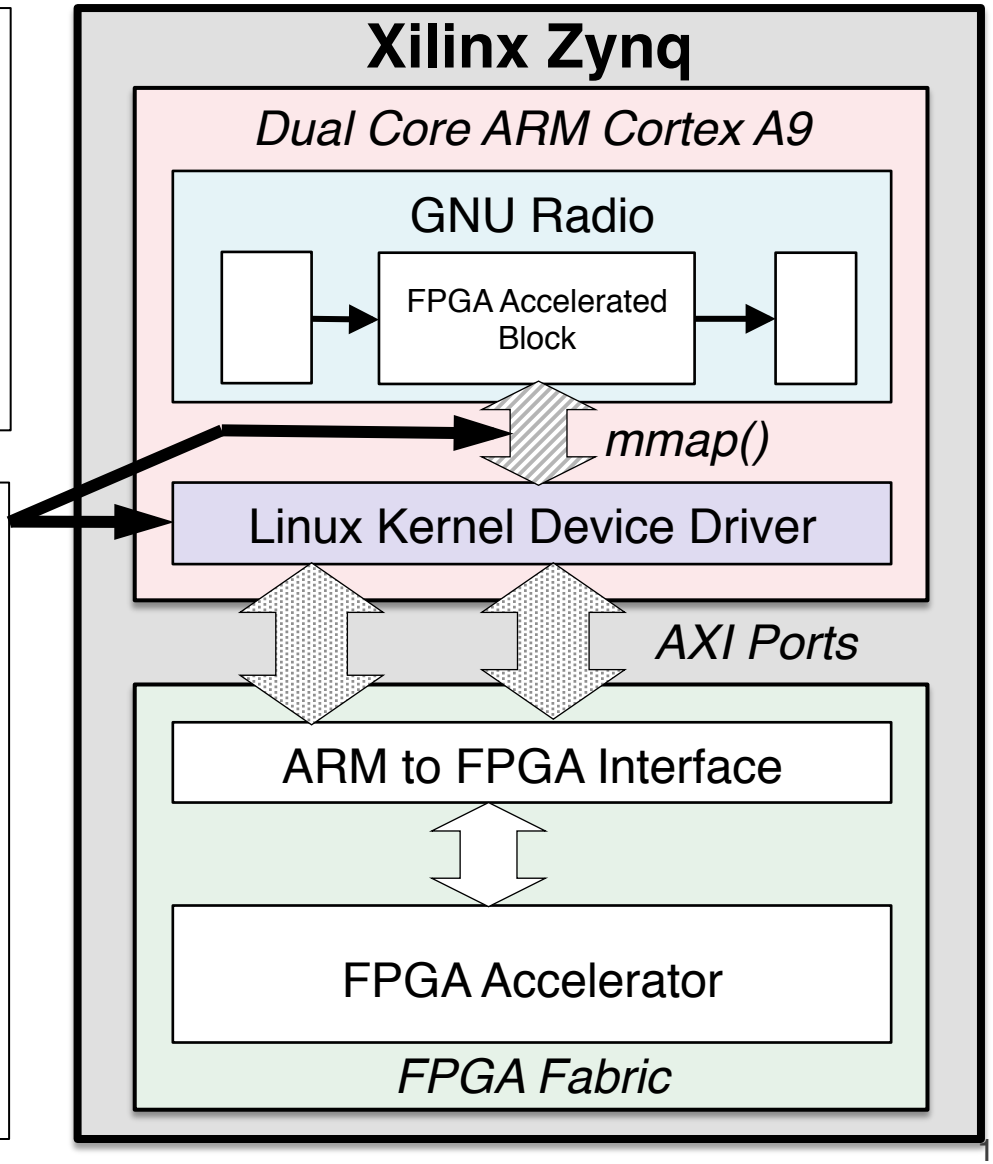
- ARM & FPGA communicate over AMBA AXI4 interconnect
  - ARM standardized bus
  - Connects ARM cores, RAM, & FPGA
  - FPGA uses AXI ports to access the interconnect
  - Simplified diagram to show only AXI ports
- Use two AXI ports
  - Read / write Sample & Control data in RAM
  - FPGA Control Interface



FPGA Accelerator Infrastructure

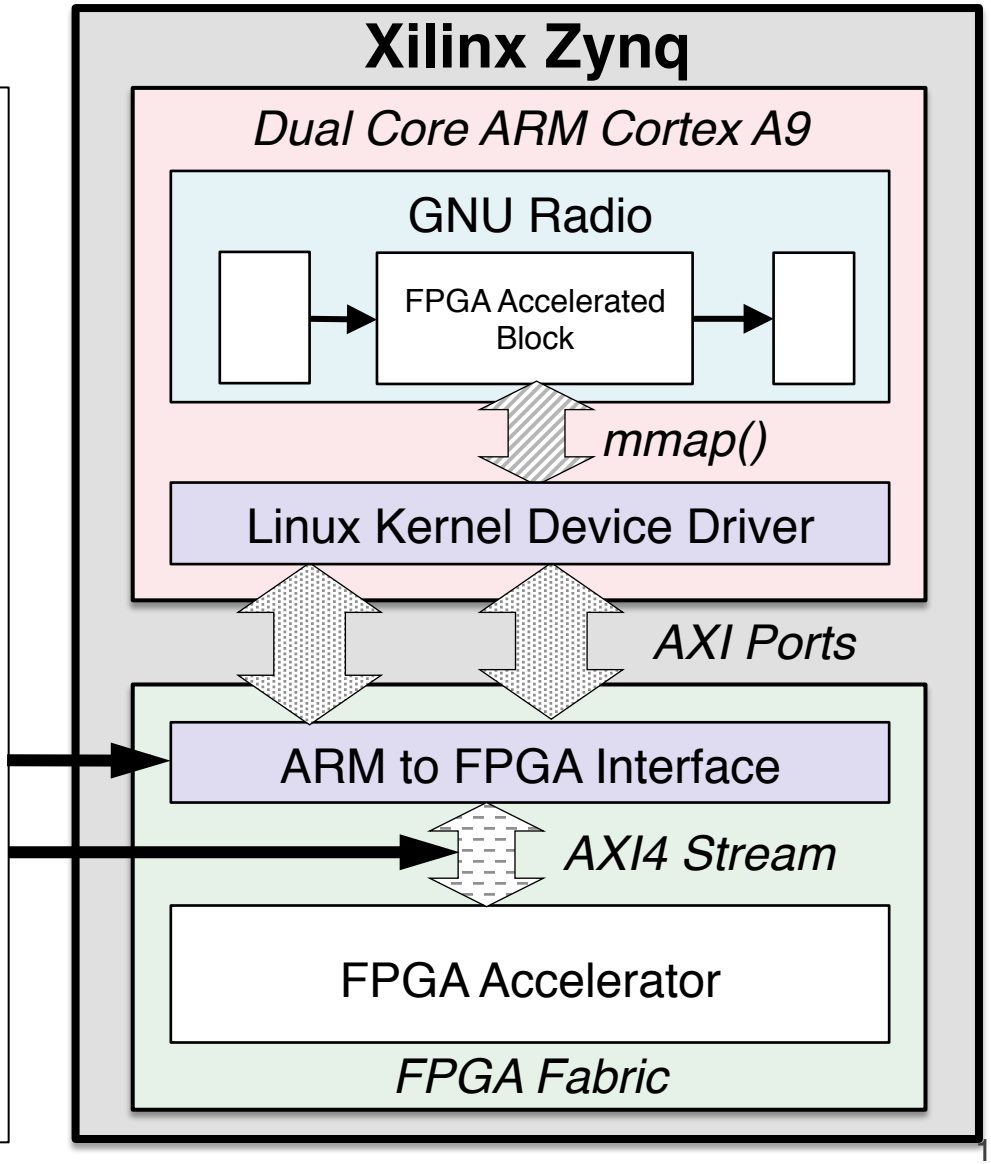
- ARM & FPGA pass control & sample data through RAM
  - Knowledge of physical memory addresses

- Device driver
  - Handles memory allocation & resolves physical addresses
  - Provides interface (mmap) to access shared memory & AXI port for FPGA control



FPGA Accelerator Infrastructure

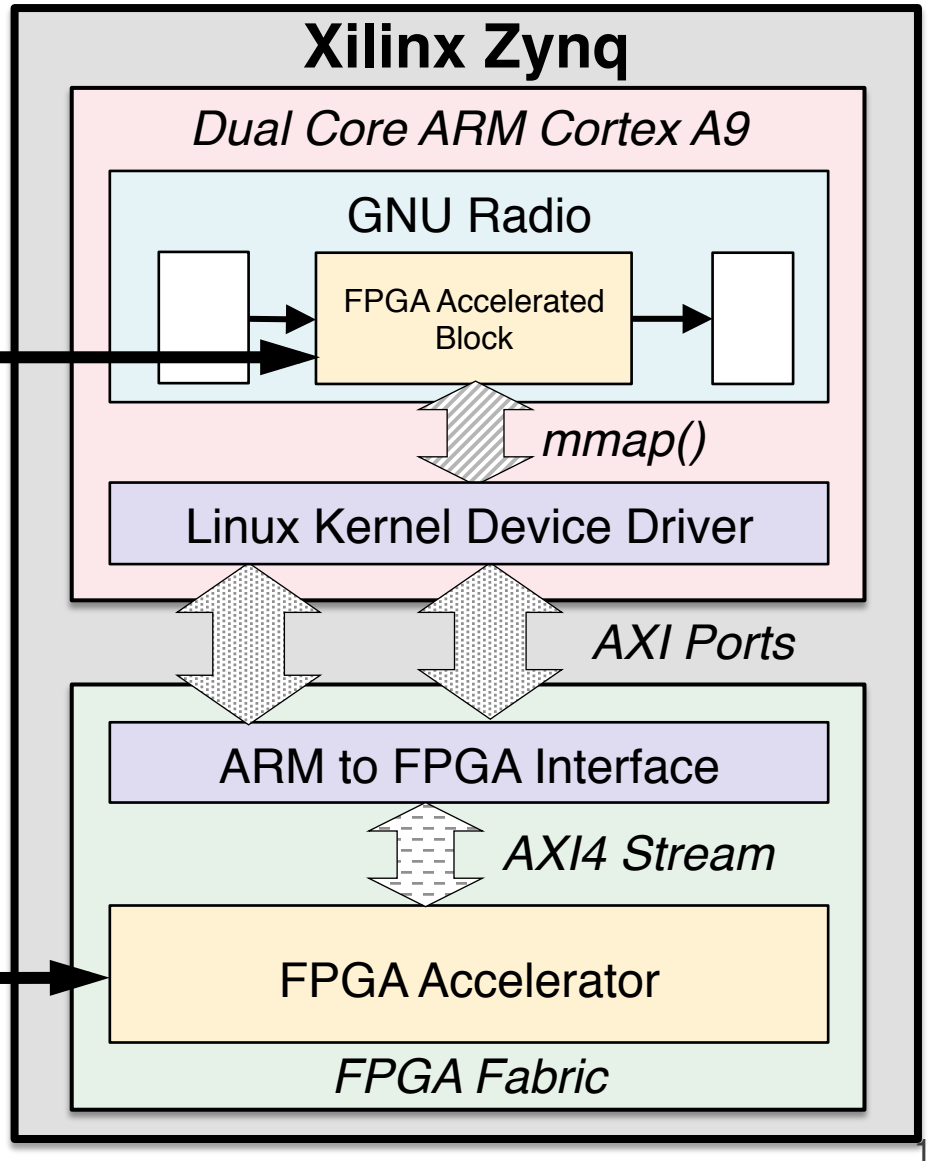
- ARM to FPGA interface
  - Uses Xilinx IP to read / write sample & control data from AXI port for RAM access
  - Receives read / write commands from ARM via AXI port for FPGA control
  - Output sample & control data on a simple bus for the FPGA accelerator
    - AXI4 Stream Bus



FPGA Accelerator Infrastructure

- Interface with Device Driver
- Code to copy GNU Radio sample & control data to shared memory
  - memcopy
- Methods to control custom FPGA accelerator

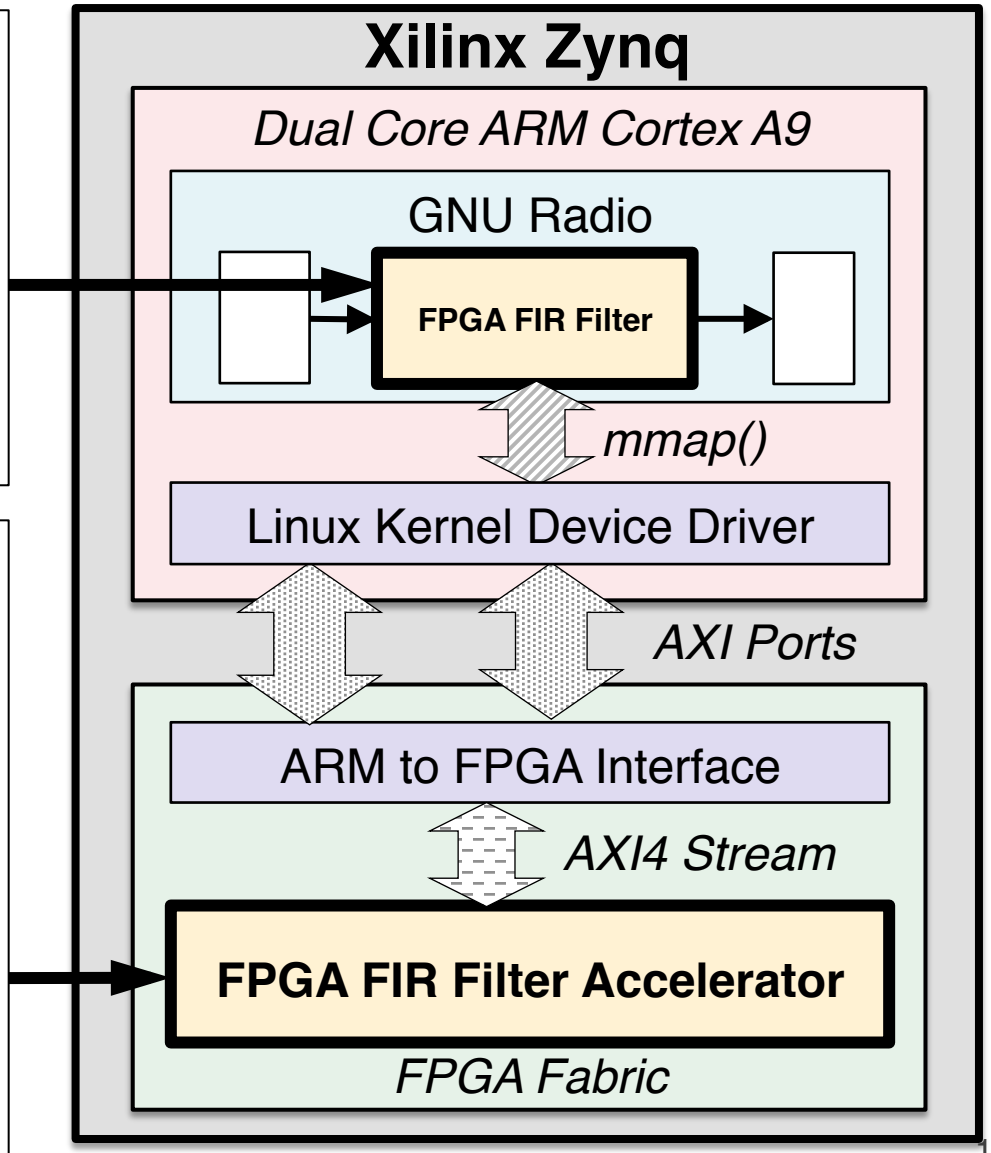
- Drop in custom FPGA accelerator(s)
- Compatible with Xilinx IP library
  - Advantage of AXI4 Stream
  - Example FIR Filter



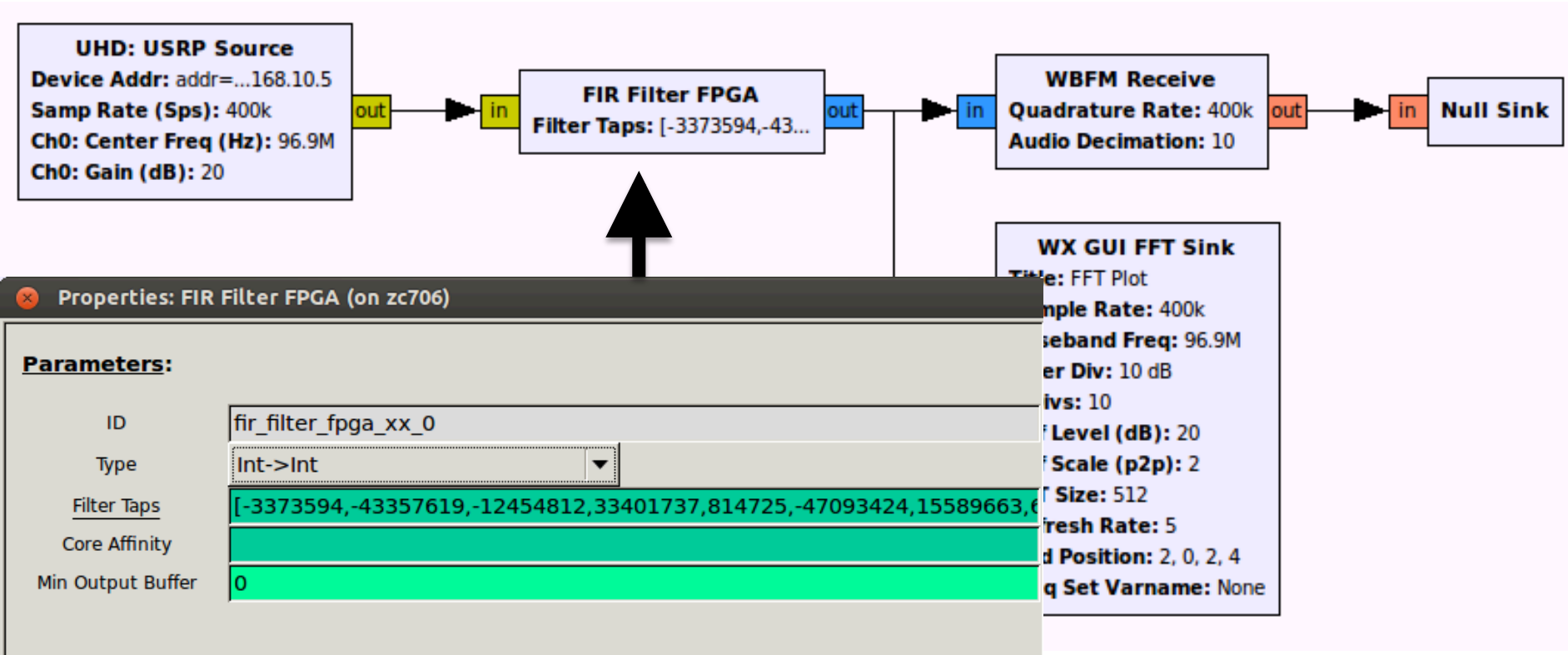
Example FIR Filter Implementation

- Versions supporting integer, complex float, & complex short int
- Set coefficients with set_taps() method

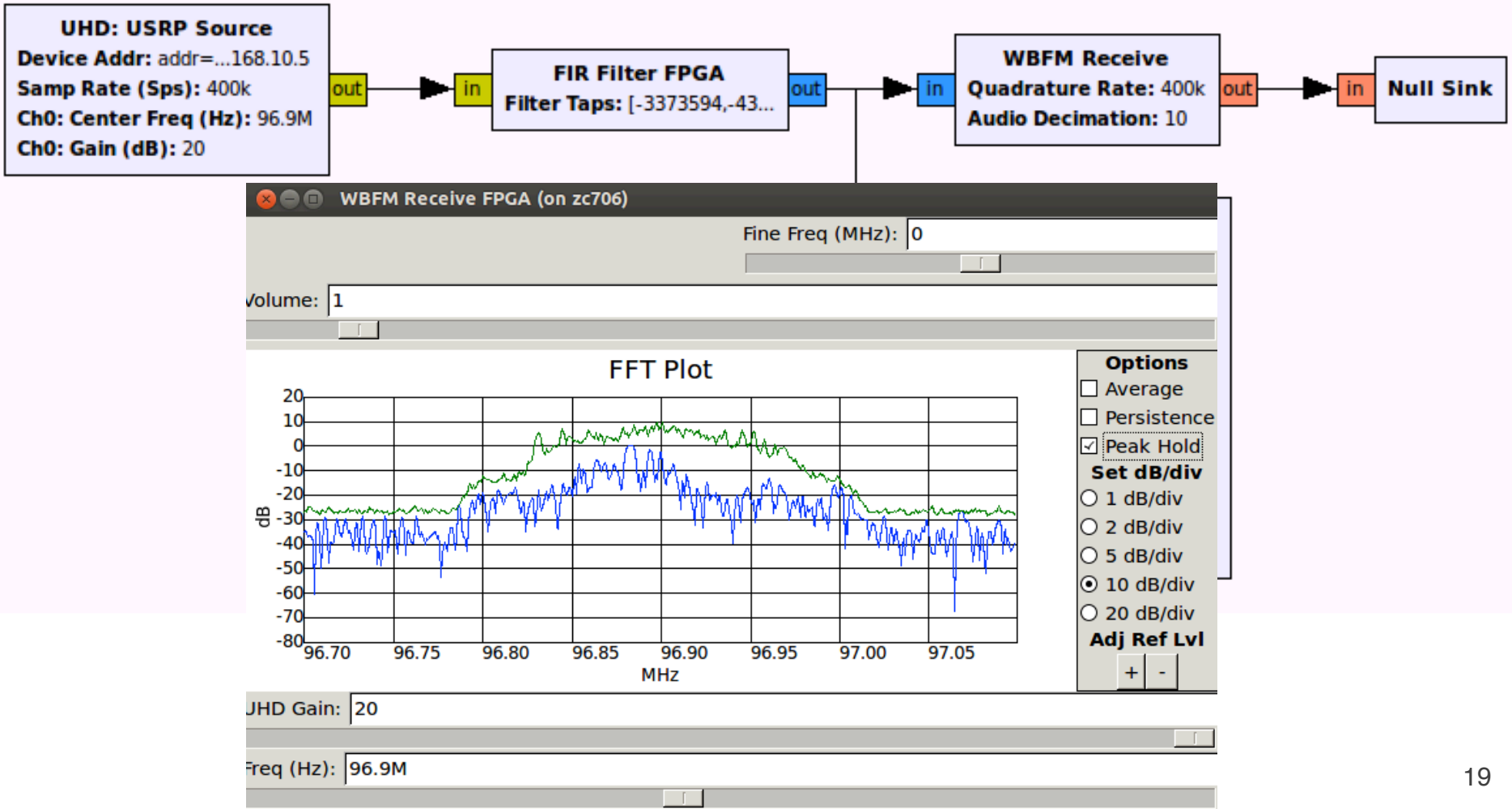
- Xilinx Coregen FIR Filter
- Reloadable coefficients
- 32-bit fixed point
  - Floating point in future
- Dual channel for complex samples
- Tested up to 111 taps



Example FIR Filter Implementation



Example FIR Filter Implementation



What else can we do with it?

- Accelerate GNU Radio signal processing
  - Filters, FFT, Error Correction / Viterbi Decoder
- High sample rate processing in FPGA
  - Process USRP raw ADC / DAC data (100 Msps) with very low latency
    - Port previous project, CRUSH, to Zynq
- Implement agile algorithms
  - Spectrum sensing and channel occupancy
  - Split MAC architecture
- Heterogeneous software defined radio
  - ARM implements control
  - FPGA offloads heavy signal processing

Results

- Completed GSoC Goals:
  - ✓ Ran GNU Radio on the Zynq's ARM processors
  - ✓ Created a FPGA acceleration infrastructure
  - ✓ Demonstrated FPGA acceleration in GNU Radio
    - Example FPGA accelerated FIR Filter
    - 7 – 15x performance increase (FIR Filter on ARM versus on FPGA)
  - ✓ Wrote comprehensive documentation available on the GNU Radio Wiki

Thank You

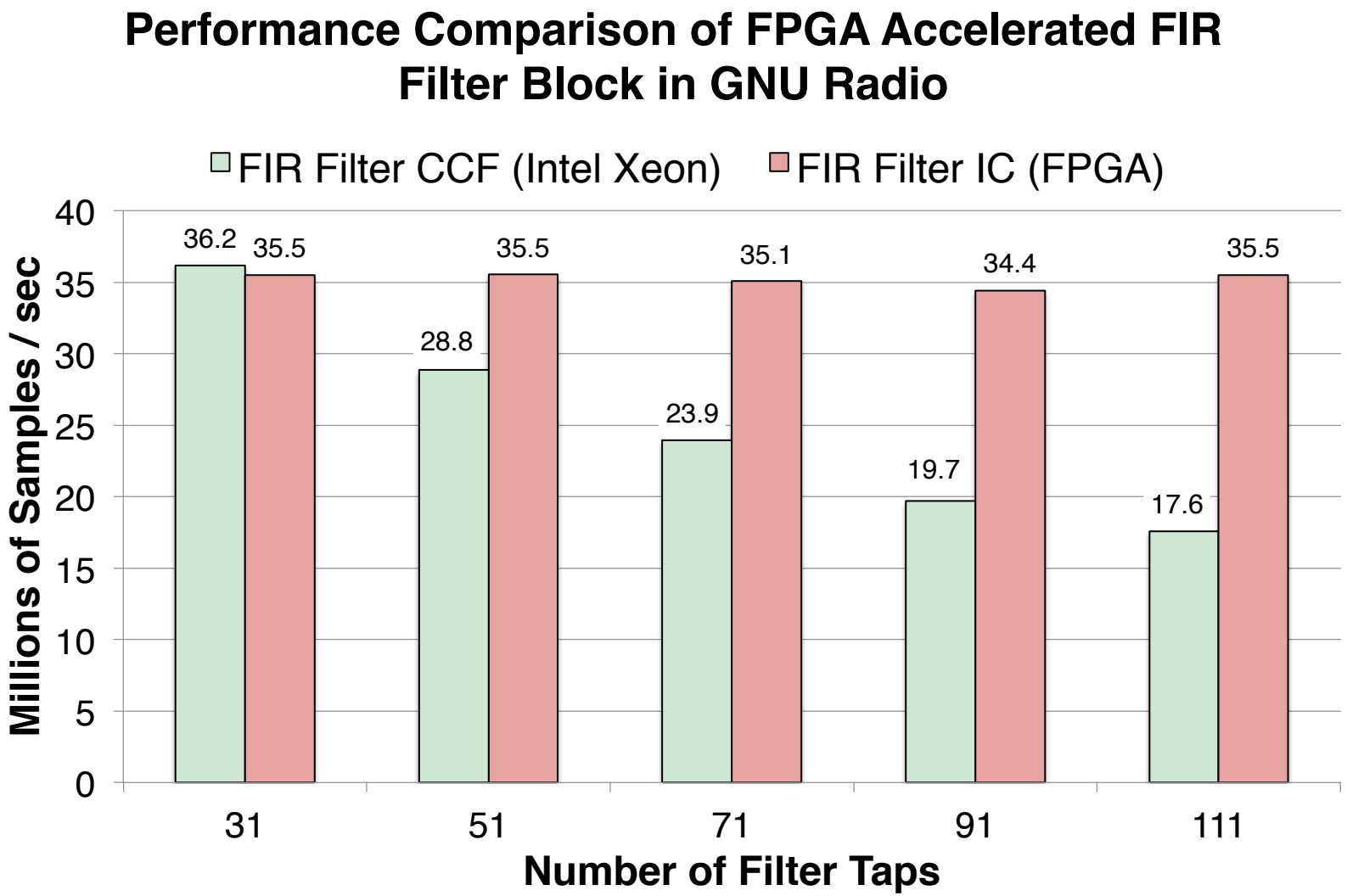
- Mentor Phillip Balister
- Advisor Professor Miriam Leeser
- Moritz Fischer
- Tom Rondeau, Martin Braun
- GNU Radio Community

Jonathon Pendlum (jon.pendlum@gmail.com)

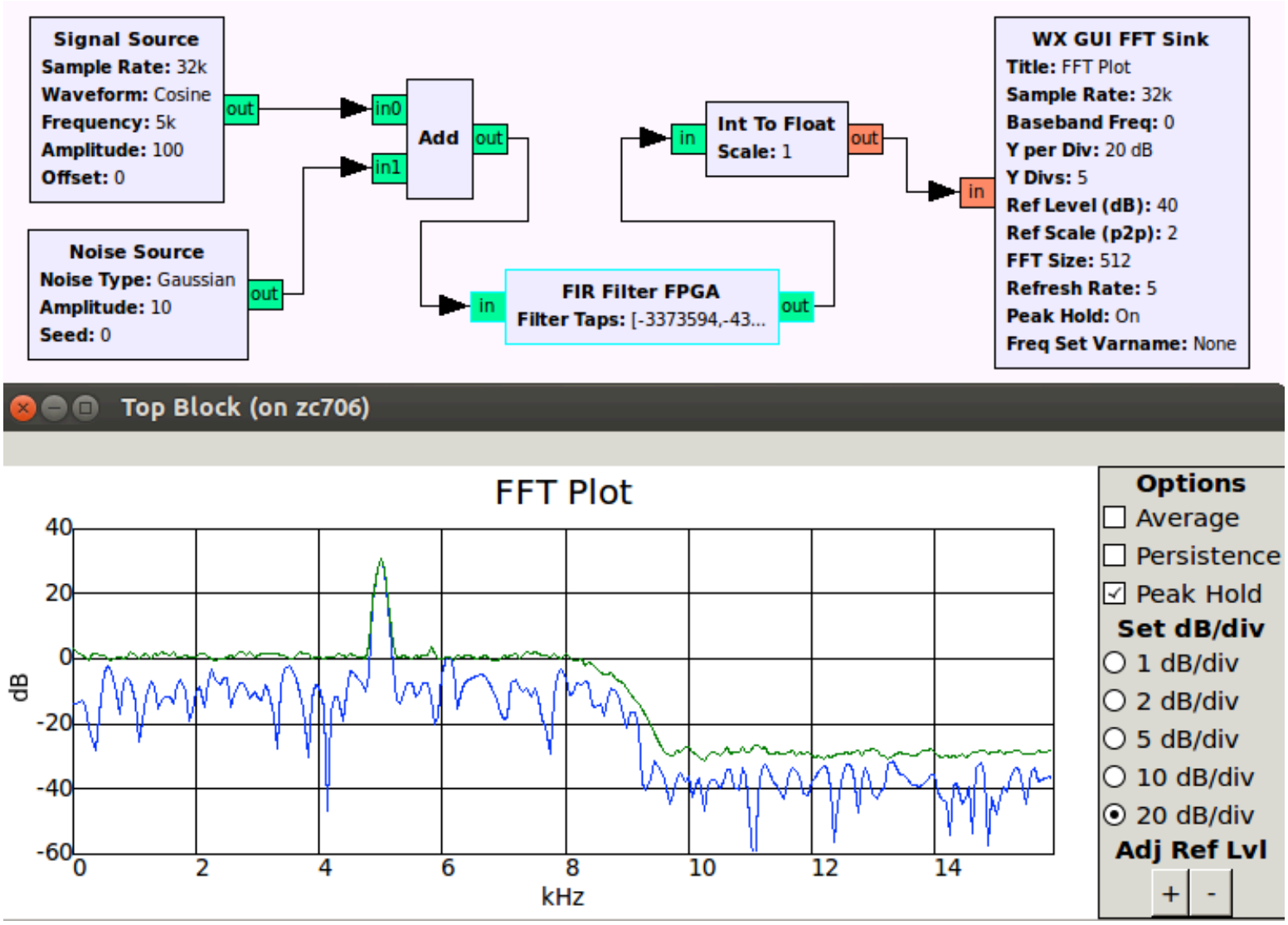
GNU Radio Zynq Wiki Page (installation instructions):
http://gnuradio.org/redmine/projects/gnuradio/wiki/Zynq

Northeastern Reconfigurable Computing Laboratory
http://coe.neu.edu/Research/rcl/

Appendix


Performance Comparison of FPGA Accelerated FIR Filter Block in GNU Radio
FIR Filter CCF (Intel Xeon)
FIR Filter IC (FPGA)
Millions of Samples / sec
0
5
10
15
20
25
30
35
40
36.2
35.5
28.8
35.5
23.9
35.1
19.7
34.4
17.6
35.5
31
51
71
91
111
Number of Filter Taps

Appendix


Signal Source
Sample Rate: 32k
Waveform: Cosine
Frequency: 5k
Amplitude: 100
Offset: 0
out
in0
in1
Add
out
Noise Source
Noise Type: Gaussian
Amplitude: 10
Seed: 0
out
in
FIR Filter FPGA
Filter Taps: [-3373594,-43...
out
in
Int To Float
Scale: 1
out
in
WX GUI FFT Sink
Title: FFT Plot
Sample Rate: 32k
Baseband Freq: 0
Y per Div: 20 dB
Y Divs: 5
Ref Level (dB): 40
Ref Scale (p2p): 2
FFT Size: 512
Refresh Rate: 5
Peak Hold: On
Freq Set Varname: None
Top Block (on zc706)
FFT Plot
40
20
0
-20
-40
-60
dB
0
2
4
6
8
10
12
14
kHz
Options
Average
Persistence
Peak Hold
Set dB/div
1 dB/div
2 dB/div
5 dB/div
10 dB/div
20 dB/div
Adj Ref Lvl
+
-