Experiences with using GNU Radio for Real-time Wireless Signal Classification

Christopher Becker (University of Utah)
Aniqua Baset (University of Utah)
Sneha Kasera (University of Utah)
Kurt Derr (Idaho National Laboratory)
Samuel Ramirez (Idaho National Laboratory)

Idaho National Laboratory

Outline

- Introduction
- System Overview
- Challenges and Approaches
- Setup
- Evaluation
- Conclusion

Outline

- Introduction
- System Overview
- Challenges and Approaches
- Setup
- Evaluation
- Conclusion

Real-time Wireless Monitoring in a Variety of Environments

- High-security/control system
  - Power plants
  - Military bases
  - Water treatment plants
- Shared spectrum
  - 3.5 GHz band

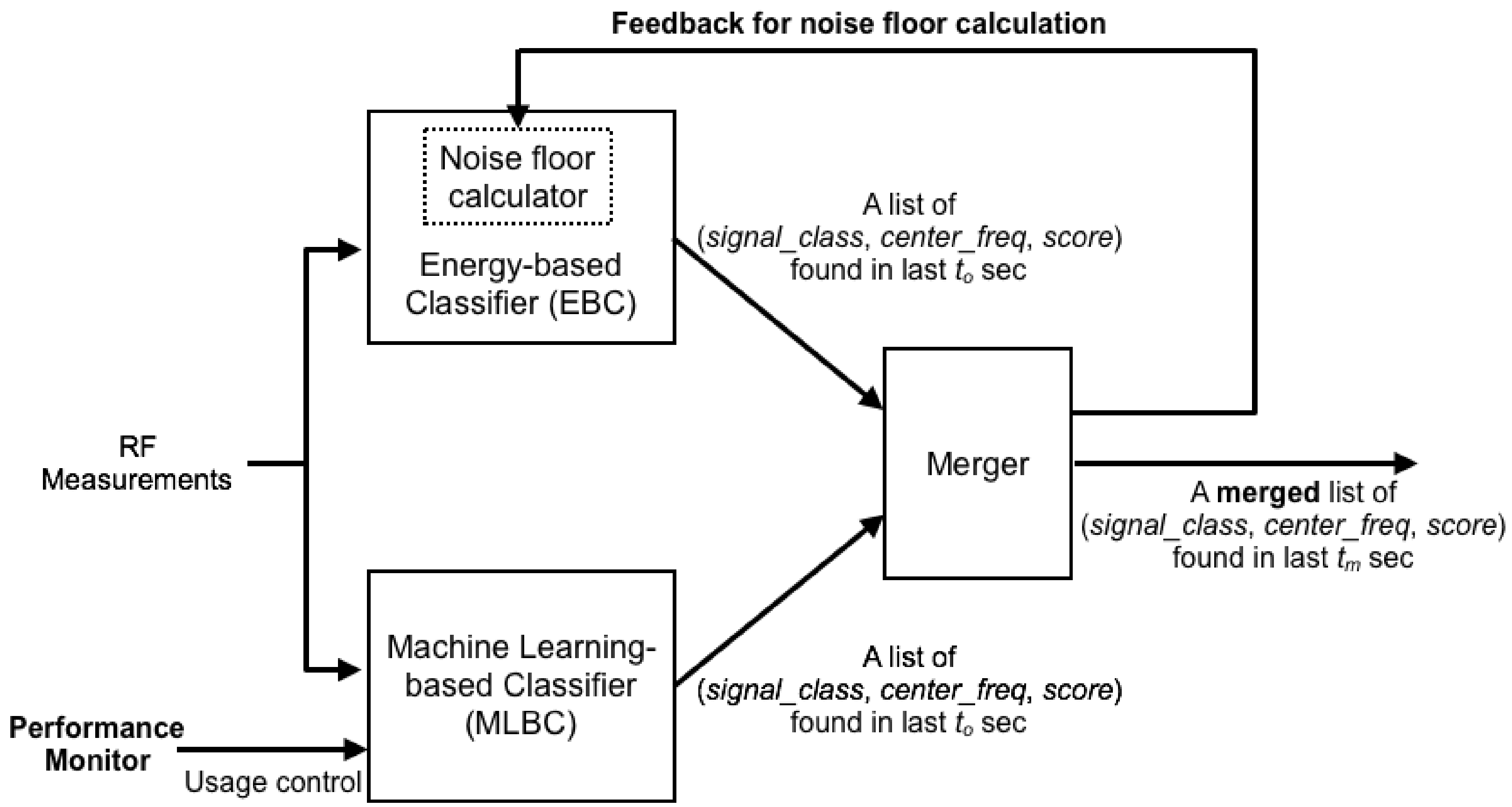
Motivation

Safety	Enforcement
Security	Interference Identification and Mitigation

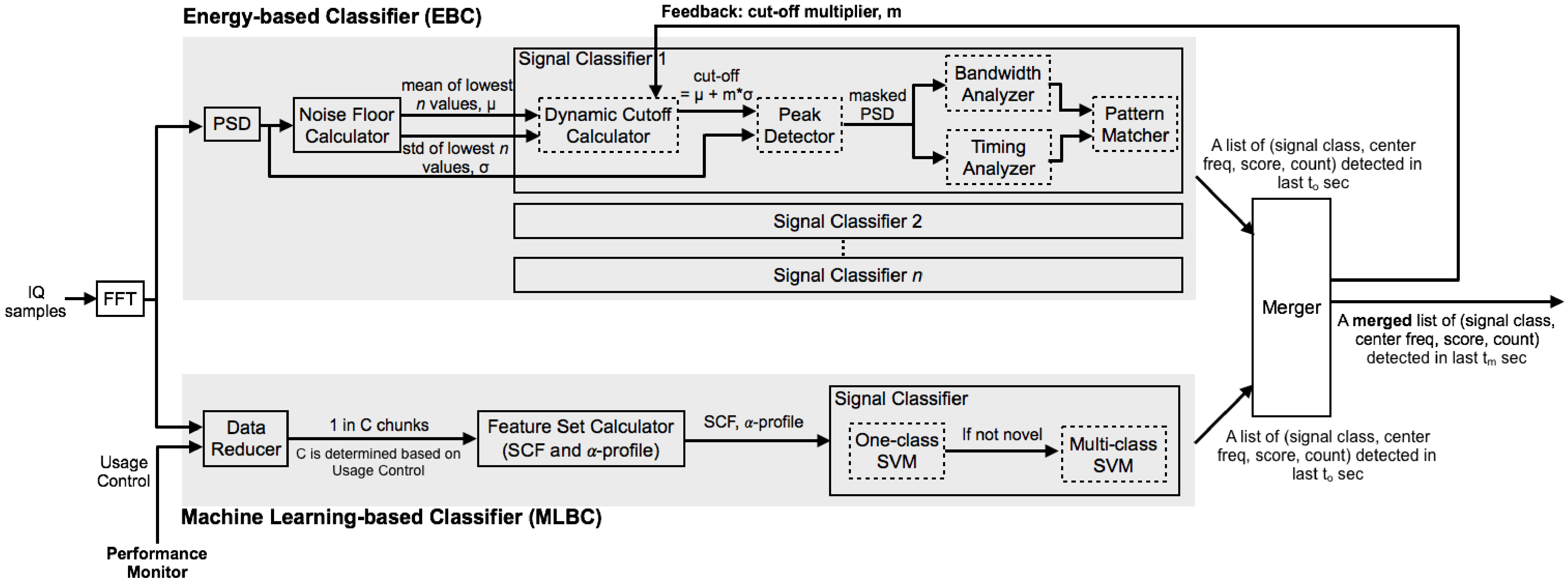
Outline

- Introduction
- System Overview
- Challenges and Approaches
- Setup
- Evaluation
- Conclusion

System Architecture


Feedback for noise floor calculation
Noise floor calculator
Energy-based Classifier (EBC)
A list of (*signal_class*, *center_freq*, *score*) found in last t_o sec
Merger
RF Measurements
A **merged** list of (*signal_class*, *center_freq*, *score*) found in last t_m sec
Machine Learning-based Classifier (MLBC)
A list of (*signal_class*, *center_freq*, *score*) found in last t_o sec
Performance Monitor
Usage control

Note: For Reference Only; part of a paper under submission



Energy-based Classifier (EBC)

Feedback: cut-off multiplier, m

IQ samples

FFT

PSD

Noise Floor Calculator

mean of lowest *n* values, μ

std of lowest *n* values, σ

Signal Classifier 1

Dynamic Cutoff Calculator

cut-off = μ + m*σ

Peak Detector

masked PSD

Bandwidth Analyzer

Timing Analyzer

Pattern Matcher

Signal Classifier 2

Signal Classifier *n*

A list of (signal class, center freq, score, count) detected in last t_o sec

Merger

A **merged** list of (signal class, center freq, score, count) detected in last t_m sec

Data Reducer

1 in C chunks

C is determined based on Usage Control

Feature Set Calculator (SCF and α-profile)

SCF, α-profile

Signal Classifier

One-class SVM

If not novel

Multi-class SVM

A list of (signal class, center freq, score, count) detected in last t_o sec

Usage Control

Machine Learning-based Classifier (MLBC)

Performance Monitor

Note: For Reference Only; part of a paper under submission

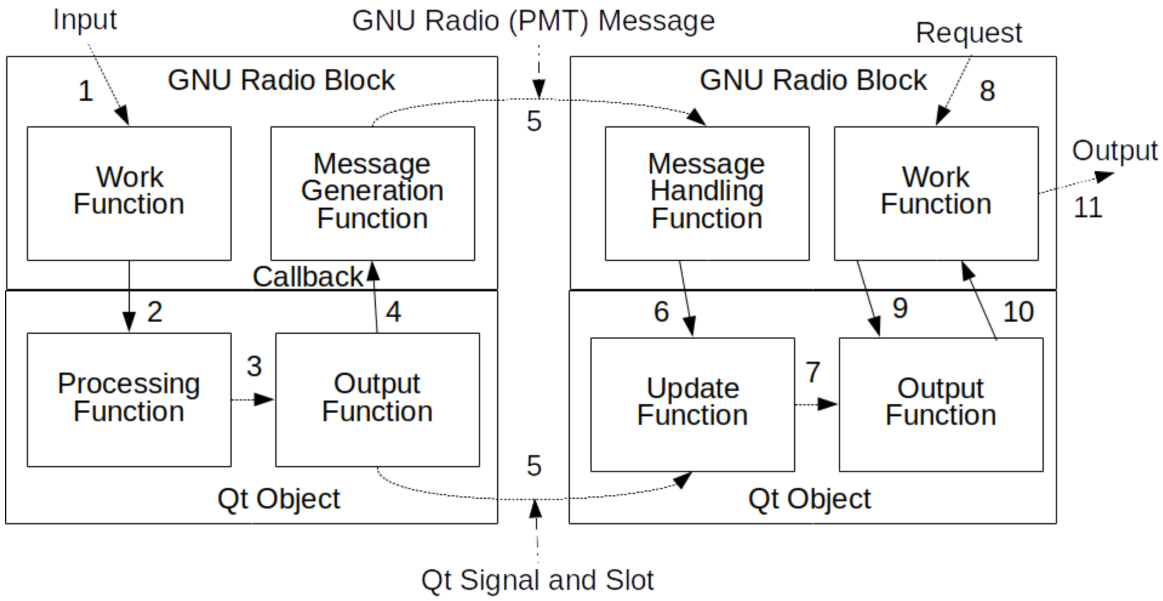
Outline

- Introduction
- System Overview
- Challenges and Approaches
- Setup
- Evaluation
- Conclusion

Message Passing

- Challenges
  - Limited message rate
  - Data encapsulation/unencapsulation expensive
- Approaches
  - Fixed-size formatted data strings
  - Qt Signals and Slots/Message Passing Wrappers

Message Passing Wrapper



1. Receive Input
2. Process the Input
3. Prepare the Output Data
4. Generate Message (if needed)
5. Forward the Data
6. Convert to Internal Format (if needed)
7. Update Internal Data
8. Request for Data
9. Forward Request
10. Prepare and Return Data
11. Send Output

Overflow and Dropped Packet Detection

- Challenges
  - Run-time performance tuning
  - Timestamps set for recovery from Overflow/Dropped Packet
  - UHD driver sends information to stderror output (not accessible by GNU Radio API)
- Approach
  - Separate program to monitor performance
    - Monitor stderror for overflow/dropped packet indicators
    - Collect information from operating system
    - Send adjustment information to rest of our system

Non-optimized Block Implementations

- Challenge
  - Some of the standard blocks not always optimized correctly (at least for our use cases)
    - Examples: add_cc, Log Power FFT, analog_const_source
- Approaches
  - add_cc and Log Power FFT: use Vector Optimized Library of Kernels (VOLK) to improve speed
  - Log Power FFT and analog_const_source: change interface/internal code to add capabilities and/or reduce overhead for our specific use case

Dynamic Decimation

- Challenge:
  - Dynamically change decimation rates at run-time
    - Pausing and reconfiguring our flowgraph causes errors due to buffer size issues
- Approach
  - Delete and re-initialize the immediately preceding blocks to have buffers re-initialize correctly

Scheduler

- Challenge
  - GNU Radio scheduler thread can become a bottleneck for performance when a large number of blocks (threads) are used
  - Still an open problem
- Current Approach
  - Limit the number of threads
    - Combine functionality of multiple blocks into new blocks
    - Allow more inputs to a single block

GPU Acceleration

- Challenges
  - Limited data buffer sizes
    - Does not lend itself to GPU Acceleration due to memory copying overhead
    - See also: Hitefield & Clancy GRCon 2016, Piscopo GRCon 2017
- Approach
  - Use VOLK whenever possible
  - Use other processing-optimization techniques
    - e.g., heaps and lookup tables

Outline

- Introduction
- System Overview
- Challenges and Approaches
- Setup
- Evaluation
- Conclusion

Laptop Configurations

	Development and Testing	Testing and Evaluation
Operating System	Ubuntu 14.04.1	Ubuntu 18.04
UHD Version	3.11.0	3.11.1
GNU Radio Version	3.7.11	3.7.12
VOLK Version	1.3.0	1.4.0

- Dell M4800 Laptops with increased RAM
  - 2.8 GHz 4-core 64-bit Intel i7-4810MQ processors
  - 32 GB of RAM
- Build GNU Radio script (slightly modified for Ubuntu 18.04) used for installing software

X310 Configurations

	Development and Testing	Testing and Evaluation
Hardware Revision	6	8
Firmware Version	5.1	6
FPGA Version	33	35

- 1 Gbps Ethernet Link
  - 25 MHz sample rate

Outline

- Introduction
- System Overview
- Challenges and Approaches
- Setup
- Evaluation
- Conclusion

Message Passing

- Used wrapped sender and receiver blocks
- 4096 (long) integers via either GNU Radio messages or Qt Signals and Slots
- Ran 100 times

	Median Time (s)
GNU Radio Messages	0.26392602
Qt Signals and Slots	0.00337398

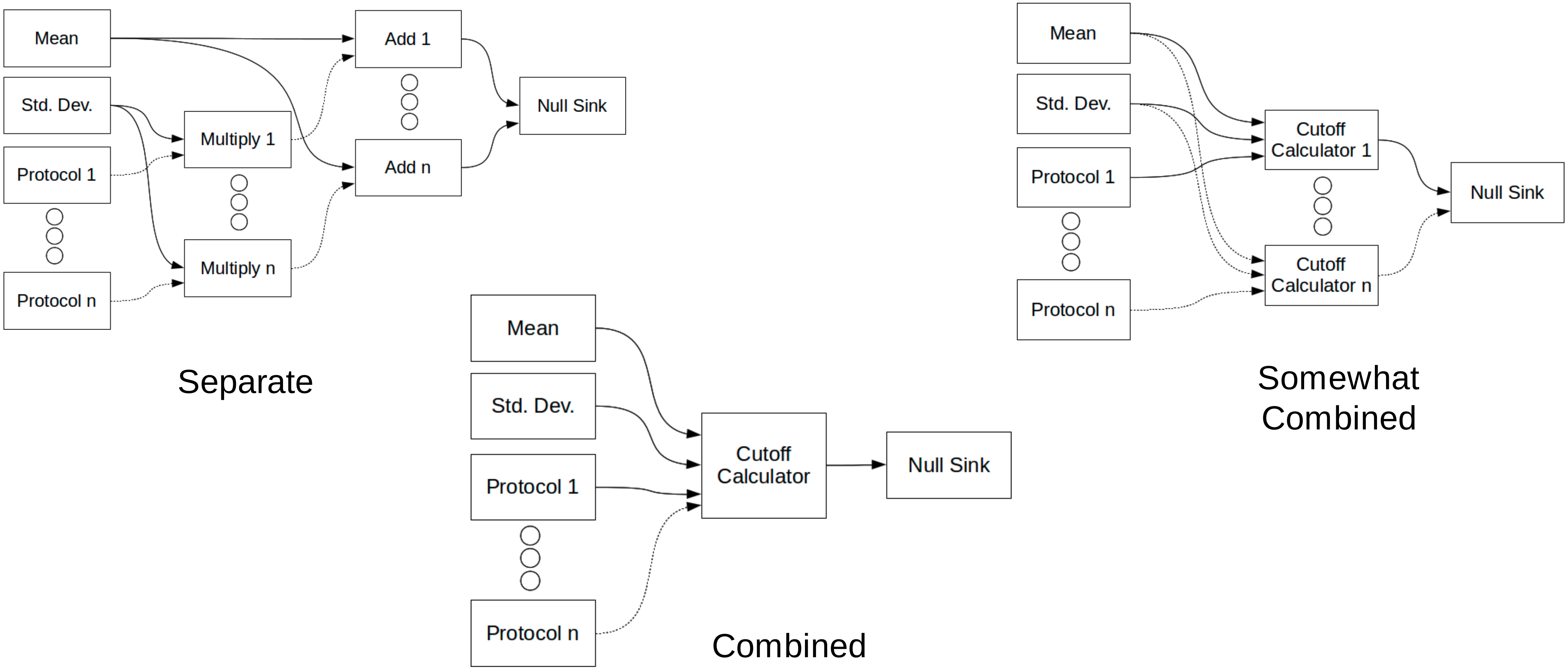
> 78x performance improvement

Non-optimized Implementations

- 4096 Samples
- Ran 100 tests

	Median Time (s)
Standard Complex Add	0.00143647
VOLK Complex Add	0.00141096
Log Power FFT	0.01590848
PSD	0.01587605

Combining Blocks/Adding Inputs (Setup 1)

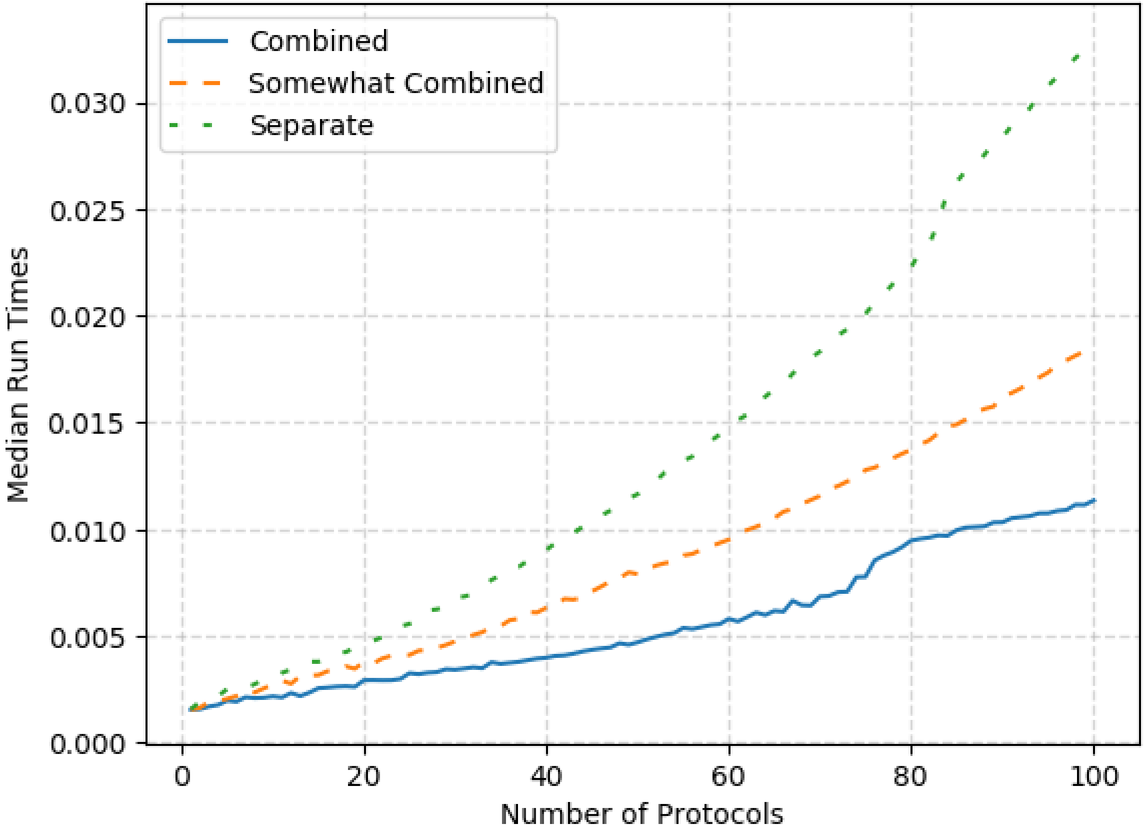


Separate

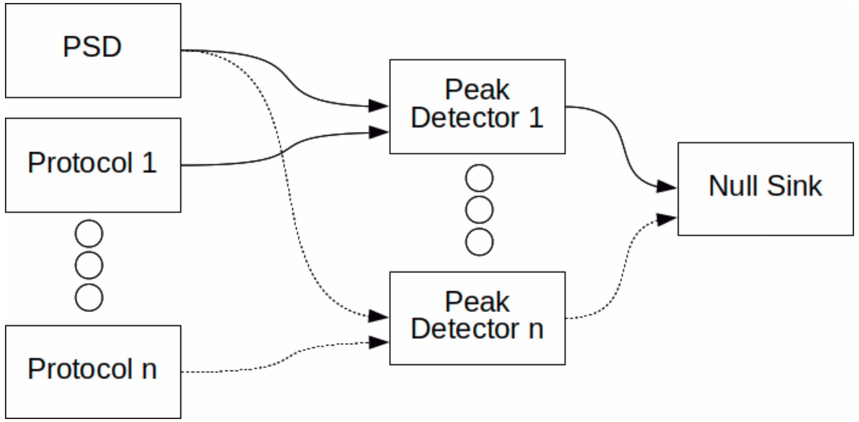
Combined

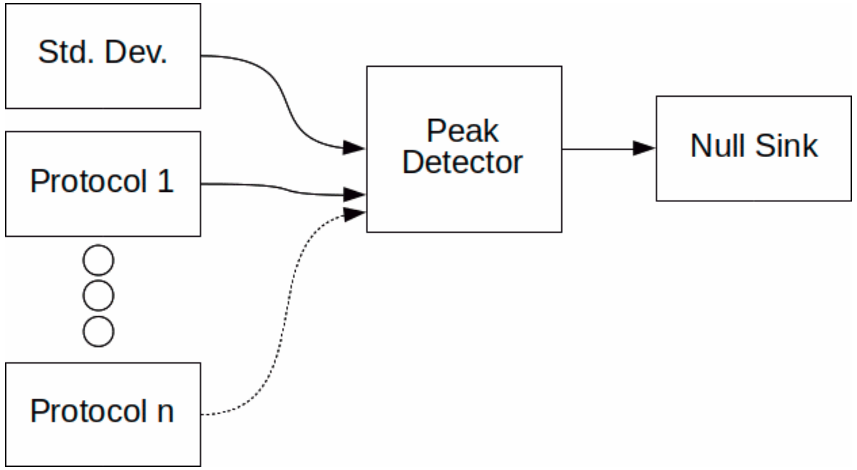
Somewhat Combined

Combining Blocks/Adding Inputs (Results 1)



Combining Blocks/Adding Inputs (Setup 2)

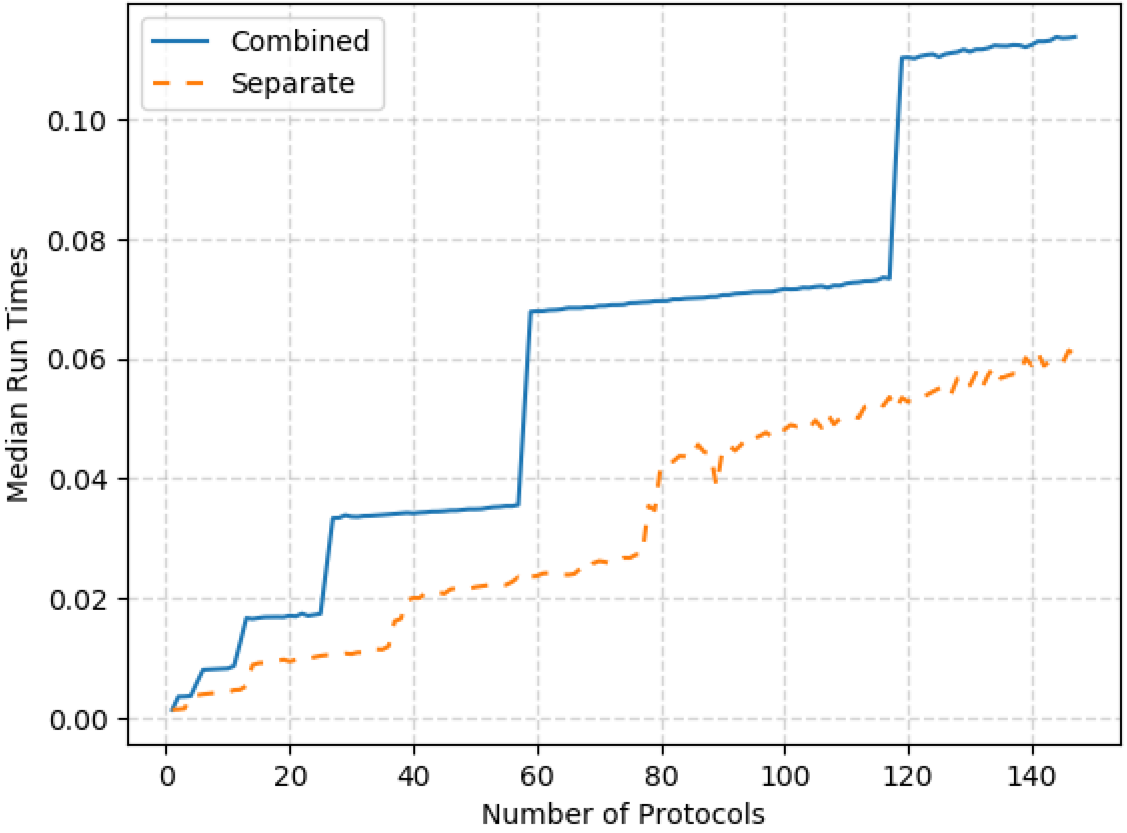




Separate

Combined

Combining Blocks/Adding Inputs (Results 2)



Outline

- Introduction
- System Overview
- Challenges and Approaches
- Setup
- Evaluation
- Conclusion

Conclusion

- Many challenges for using GNU Radio in real-time spectrum monitoring applications
- Many ways to approach these challenges
  - Some provide large performance improvements
  - No one-size fits all solution, however

INL

Idaho National Laboratory