
ANDRO

# Software Defined Radio Implementation of A DVB-S Transceiver

Ashwin Amanna, James Bohl, Zachary Goldsmith – *ANDRO Computational Solutions, Rome NY*

Michael Gudaitis, Benjamin Kraines, Robert DiMeo, William Lipe, Richard Butler II – *Air Force Research Lab*

Originator Reference Number: RIT-17-435
Public Approval Case Number: 88ABW-2017-4648

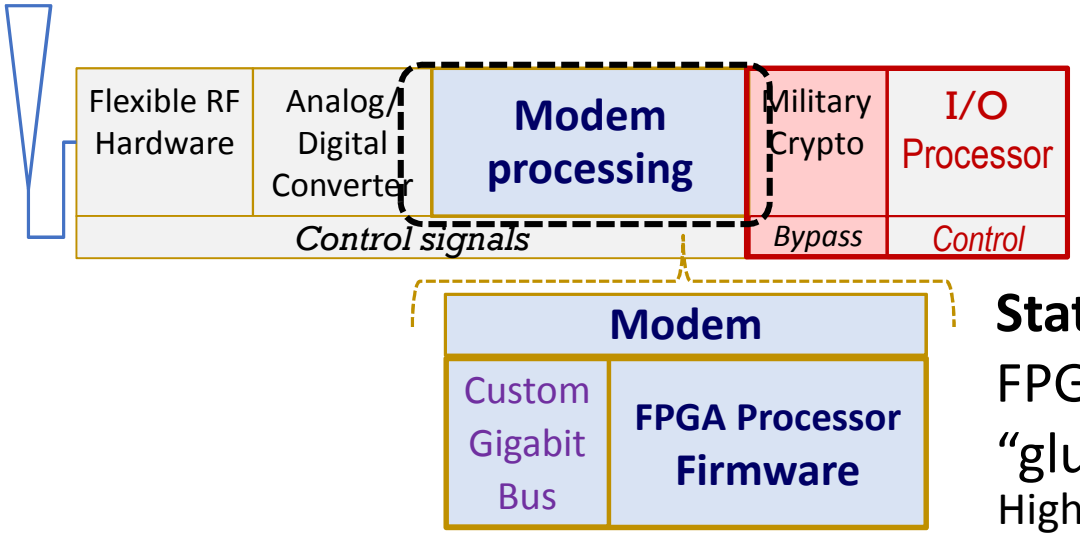
# Outline

- Problem / Motivation
- Review of existing software implementations of DVB-S
- Contributions
- Implementation
- Results
- Limitations
- Implications

# Problem

- Early promises of Software Defined Radios (SDR) included:
  - Accelerated development times
  - Greater accessibility to the waveforms (not a black box)
  - Ease of modification and adaptation
- The reality in the 1990's was general purpose processors (GPP) could not meet the demands of complex waveforms
- This led to firmware emphasis
  - Less portability
  - Less accessibility
  - Less adaptable
- Today's modern GPP and advancement in low-cost SDRs has created an opportunity to go **Back to the Future** with SDR waveforms

# Motivation for an Agile SW/HW Ecosystem




**SDR Block Diagram**

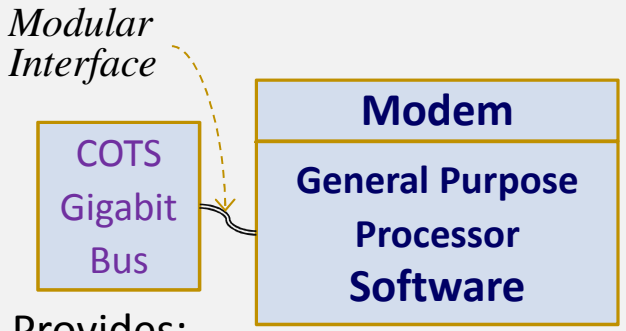
*Focus on the Modem*

**State of the Practice**

FPGA-based approach

"glued" to custom gigabit bus

Highest theoretical performance, firmware-based but limited flexibility, long development, higher cost




**Software-based** GPP approach with Modular Open System Bus for gigabit data transfers across multiple HW instantiations

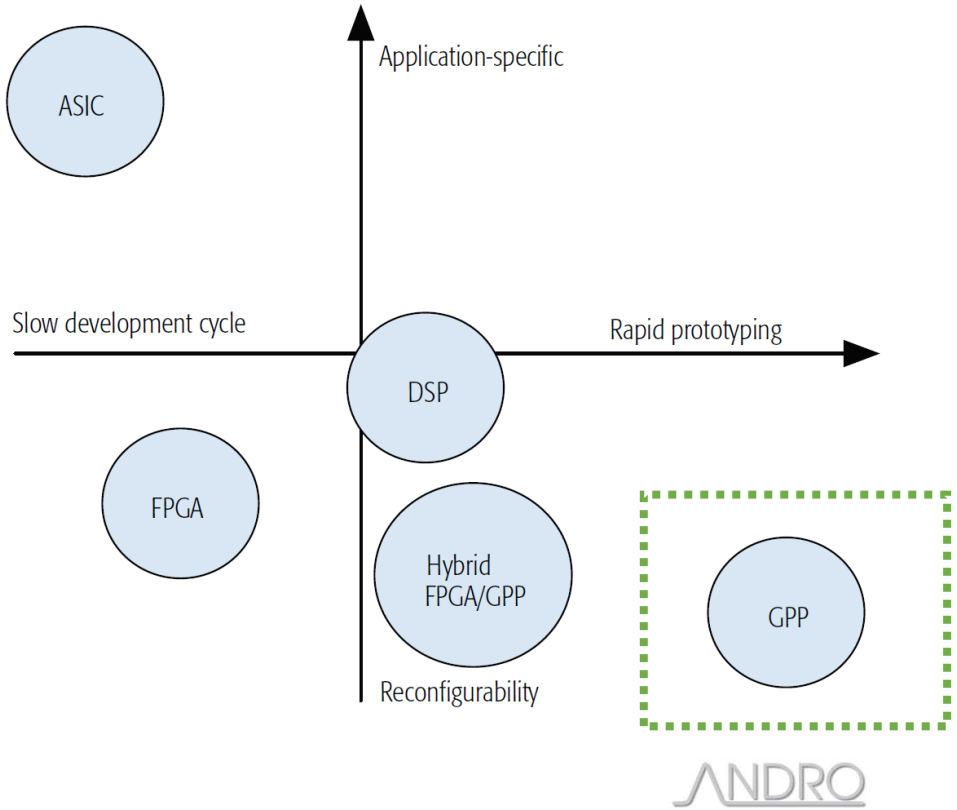
Provides:

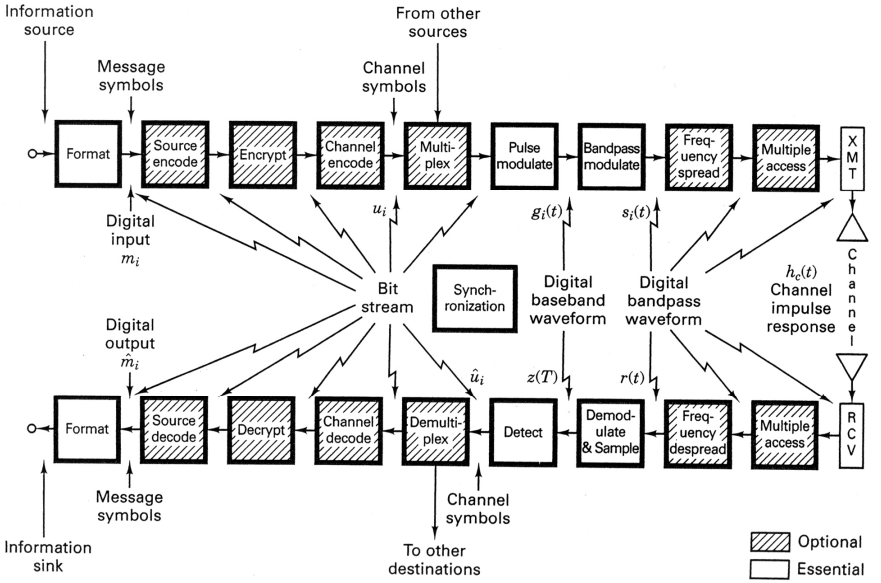
High theoretical performance, innovative **software-based Modem**.

Best flexibility, shortest development, lower cost, **agnostic hardware**

# Our Approach

- Sklar functional waveform model
  - Progressively build functional block pairs
- 'Wind Tunnel'
  - Scaled versions, functionally similar, controlled conditions
  - Test early test often (TETO)




Source: https://fypfpga.files.wordpress.com/2011/07/communication-sys-block-diag.jpg


Source: http://evworld.com/press/genovation_glenmartinwindtunnel.jpg

Source: G. Sklivanitis, A. Gannon, S. N. Batalama, D. Pados, "Addressing Next-Generation Wireless Challenges with Commercial Software-Defined Radio Platforms," IEEE Communications Magazine, Jan 2016.

# Why DVB-S

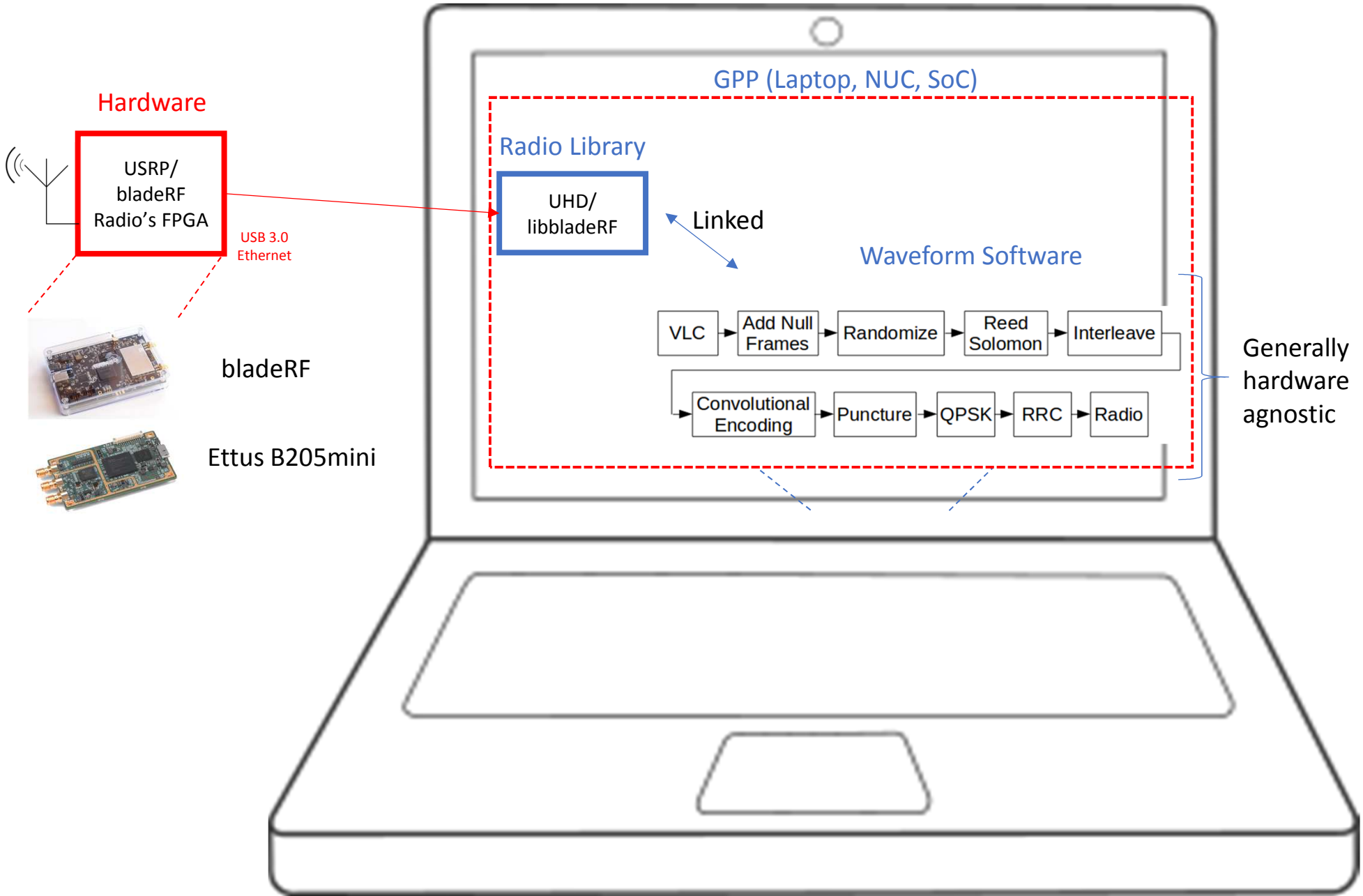
- Data rates, modulation, encoding, interleaving, randomization
  - Similar complexity to tactical waveforms
- DVB-S is an open standard allowing us to share/discuss
- COTS devices exist to demonstrate limited interoperability
- Serve as a teaching waveform and reference implementation for the ecosystem

# Our Contributions

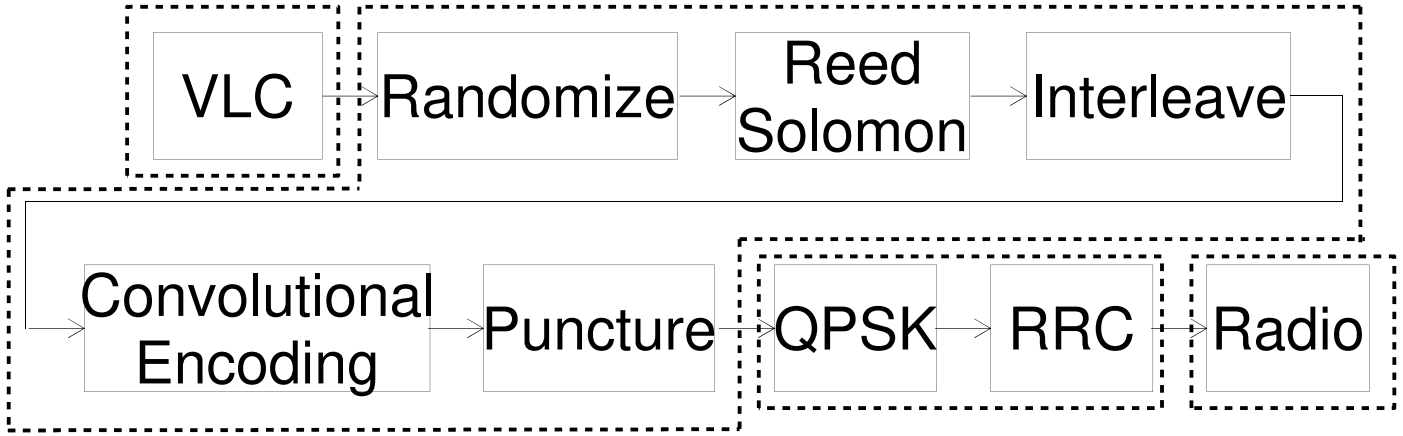
- Complete real-time DVB-S transceiver in software
  - 2 man-week rapid development timeline
- Validated transmitter with COTS DVB-S Satellite Receiver
- Leverage with multi-core parallelization
  - Separate threads that pass data between threads
  - G. Baruffa et al., 2014 [1] utilize thread pool with main thread directing individual threads
- Selective use of Advanced Vector Extensions (AVX) to improve computational load
- Performance assessments of processor usage and latency

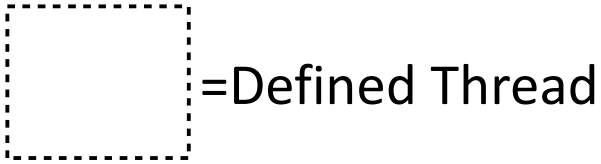
[1] G. Baruffa, L. Rugini and P. Banelli, "Design and Validation of a Software Defined Radio Testbed for DVB-T Transmissions," Radioengineering, vol. 23, pp. 387-398, 2014.

# General System Model

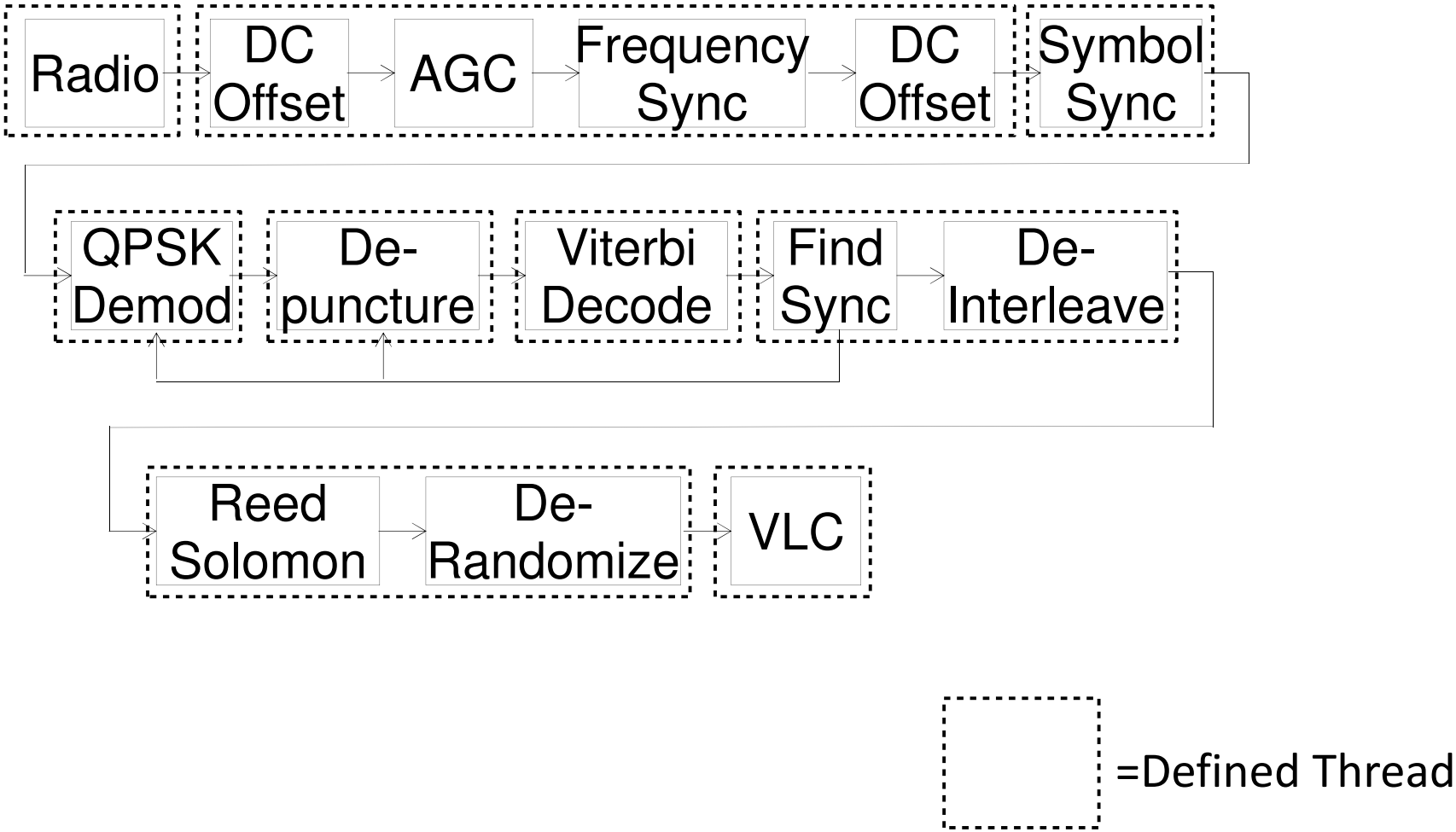


# Implementation - Transmitter



# Implementation - Receiver

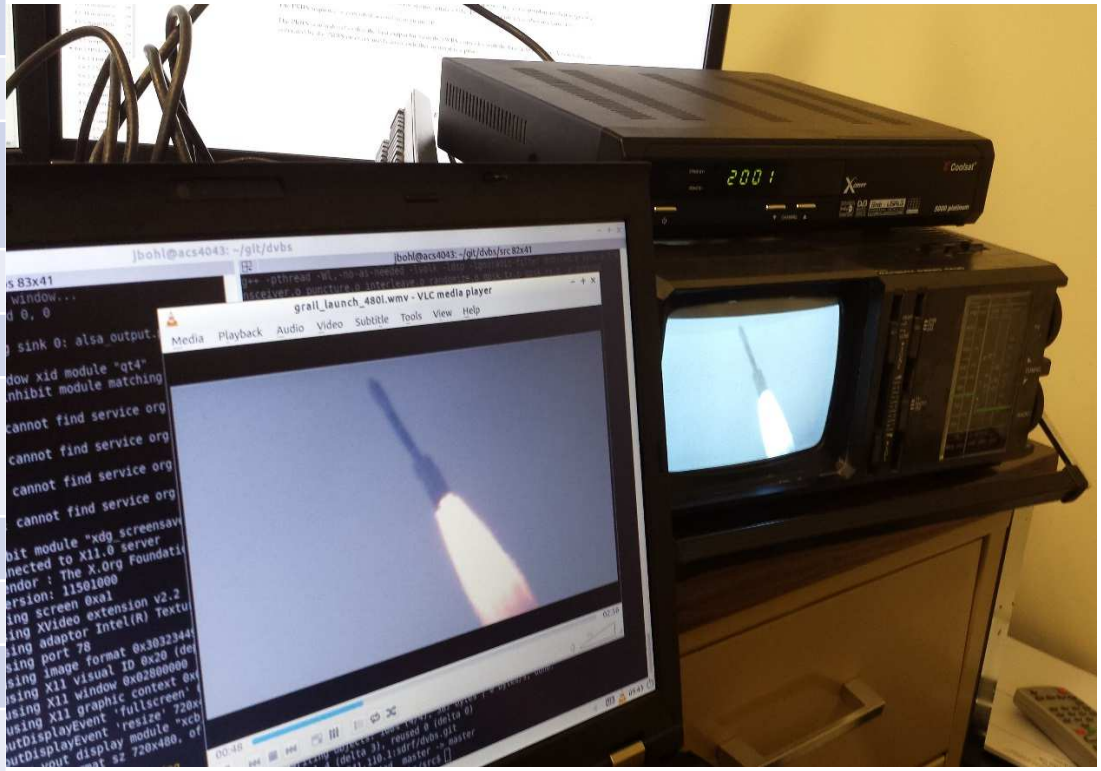


# Functions implemented with SIMD Intrinsics

- DC Offset Correction
  - Calculate DC offset (accumulate samples in block)
  - Subtract DC offset (subtract scalar from vector)
- AGC
  - Calculate RMS amplitude
  - Divide each sample by RMS amplitude
- Frequency Sync
  - Correct sample phase (complex multiply two vectors)
  - Calculate error (dot product)
  - FFT
- Symbol Sync
  - Calculate timing error (dot product)
  - Interpolation filter (dot product)
- Viterbi Decoder
  - Calculate branch metrics
    - In-line assembly code within the function not intrinsic

# Demonstration System Model

| Item | Description |
|---|---|
| DVBS receiver | Coolsat 5000 Platinum (IF 950-2150MHz) |
| DVBS specification | EN 300 421 V1.1.2 (1997-08) |
| SDR platforms | Tested with bladeRF and USRP B205MINI |
| Linux distribution | Ubuntu 14.04.3 |
| Linux kernel version | 3.13.0 |
| Software dependencies | • VLC 2.1.6-0-gea01d28<br>• libbladerf<br>• libuhd |
| Antenna | OmniLOG 70600 Antenna |
| Frequency | 1GHz |
| Receiver gain | Between 20dB and 40dB |
| Transmitter gain | Between 20dB and 35dB |
| Samples/symbol | 2.25M |
| Sample rate | 33.75M |
| Frame size | 188 Bytes |




- Original video is an Open source 480i.wmv file from NASA:
- http://s3.amazonaws.com/akamai.netstorage/HD_downloads/grail_launch_480i.wmv

# Limitations

- Synchronization loss occurs intermittently with data transfers exceeding approximately 900,000 MPEG2 frames.
- Limited hardware automatic gain control (AGC) on low-cost platforms
- Calibration of transmitter/receiver gains not stable requiring re-calibration periodically
- Implemented a subset of specification necessary to demonstrate limited interoperability with a COTS device
  - Not fully compliant with specification
- Validated interoperability of Tx with COTS Receiver. Have not tested interoperability of Rx

# Results: Bit Error Rate

- Over-the-air, uncontrolled office environment
- BER measured on fully synchronized mock MPEG2 frames at the receiver
- 188 Byte frames transmitted in 2ms bursts with 10 frames/burst.
  - 13 repetitions of 90,000 bursts (900,000 total frames)
- Under stable synchronization→ no measured BER
- Observed periods of synchronization instability after 90,000 bursts

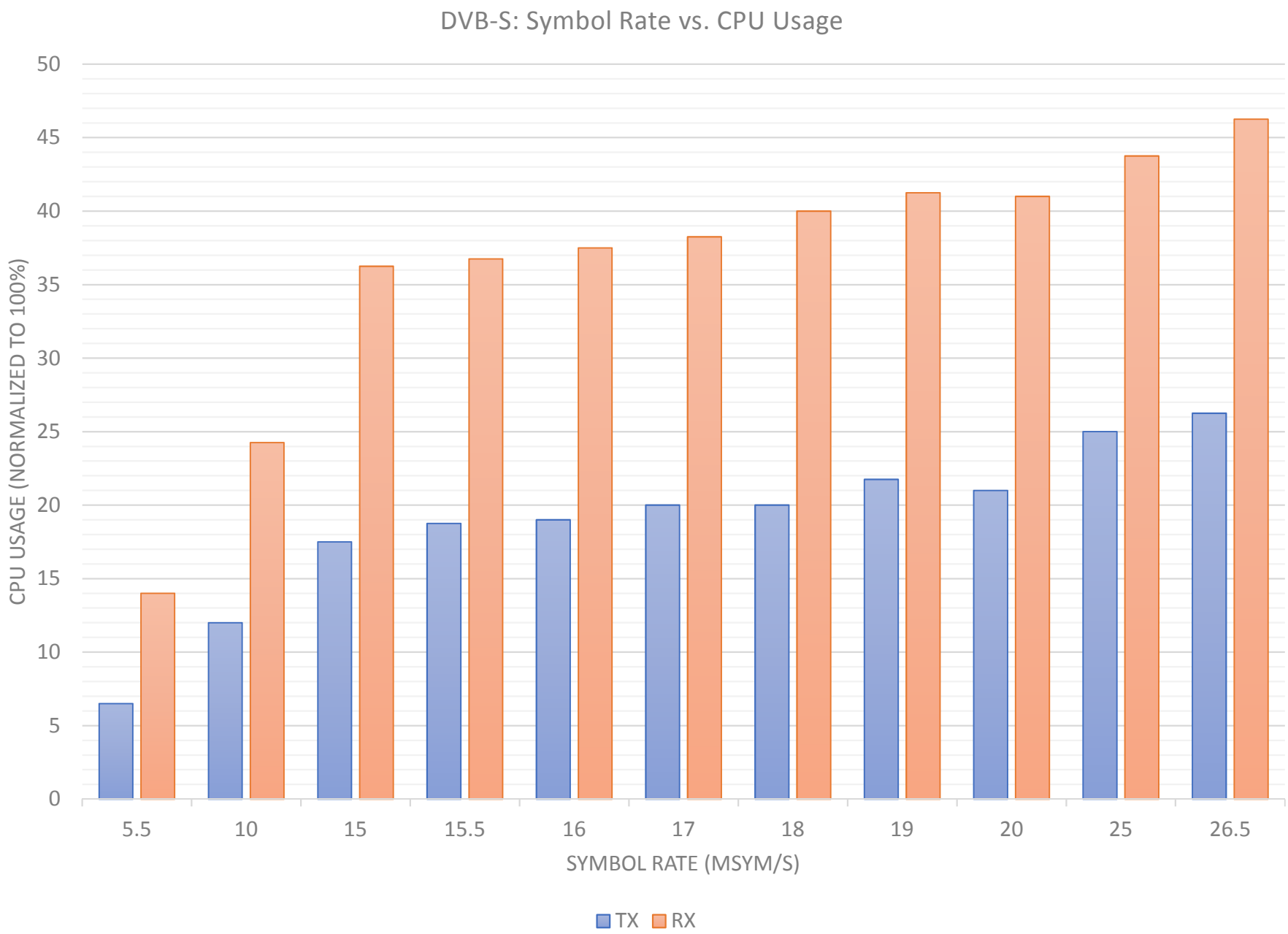
# Results – CPU Usage Intel i7, i3

- *htop*, was used to monitor CPU usage of the transmitter software alone, the receiver alone, and both the transmitter and receiver running simultaneously.

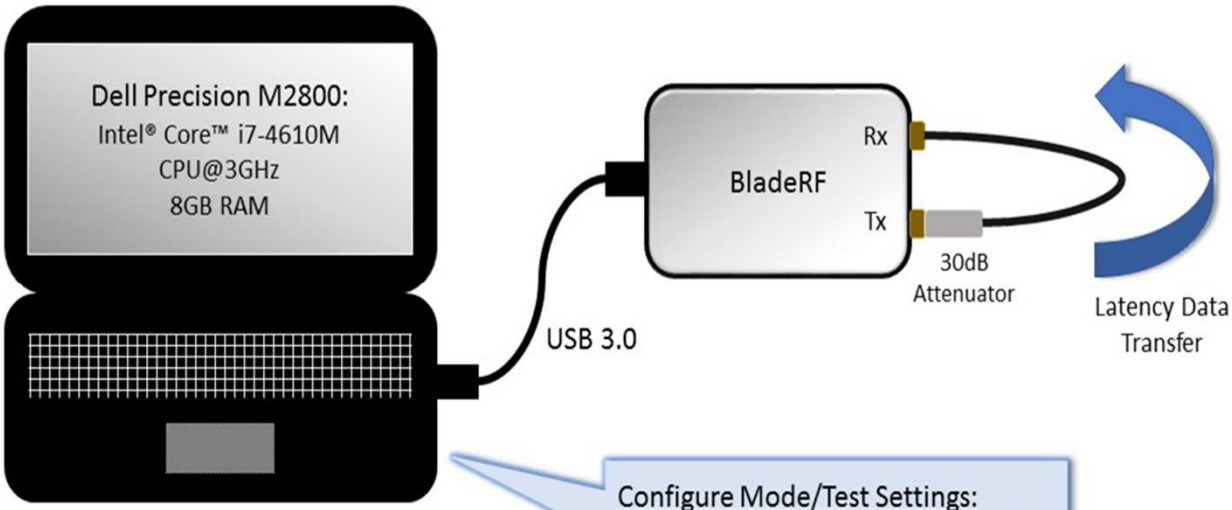
| Laptop | Features | HTop Reported CPU Usage out of 100% | | |
|---|---|---|---|---|
| | | Tx/Rx | Tx | Rx |
| **Dell Precision M2800** | Intel® Core™ i7-4610M CPU@3GHz 8GB RAM, 4CPUs | 22% | 8% | 16% |
| **Lenovo L530** | Intel® Core™ i3-2348M CPU@2.3GHz 4GB RAM, 4CPUs | 50% | 24% | 35% |

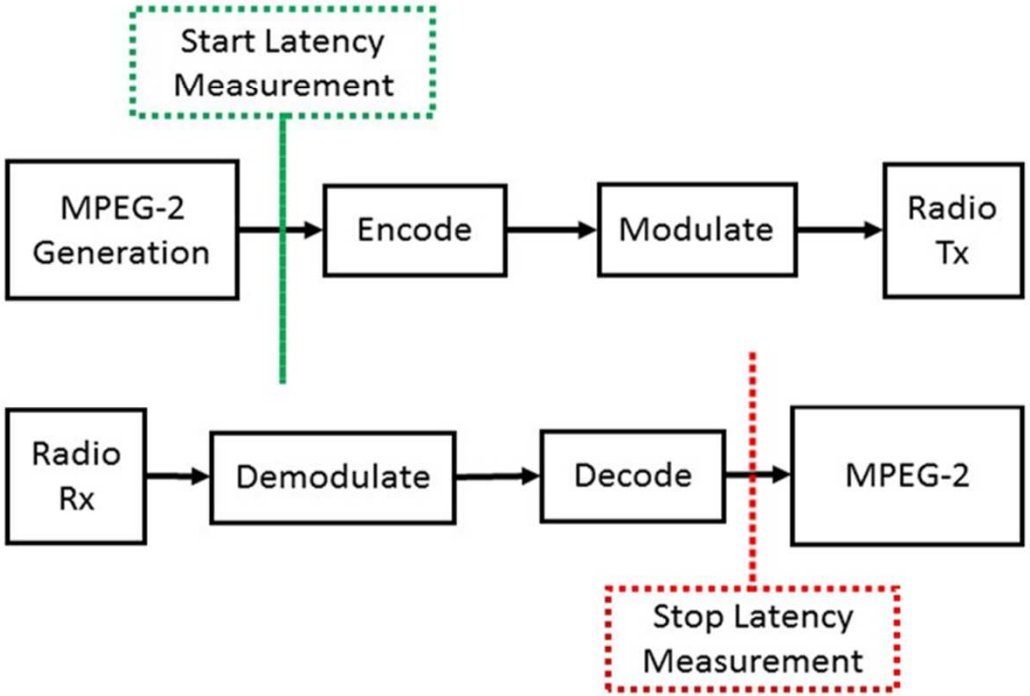
Note that for quad-core processors, *htop* typically reports results out of a maximum of 200% to 400% depending on # of threads/core. Here, results are normalized to 100% maximum

# Results: Normalized CPU Usage vs Symbol Rate

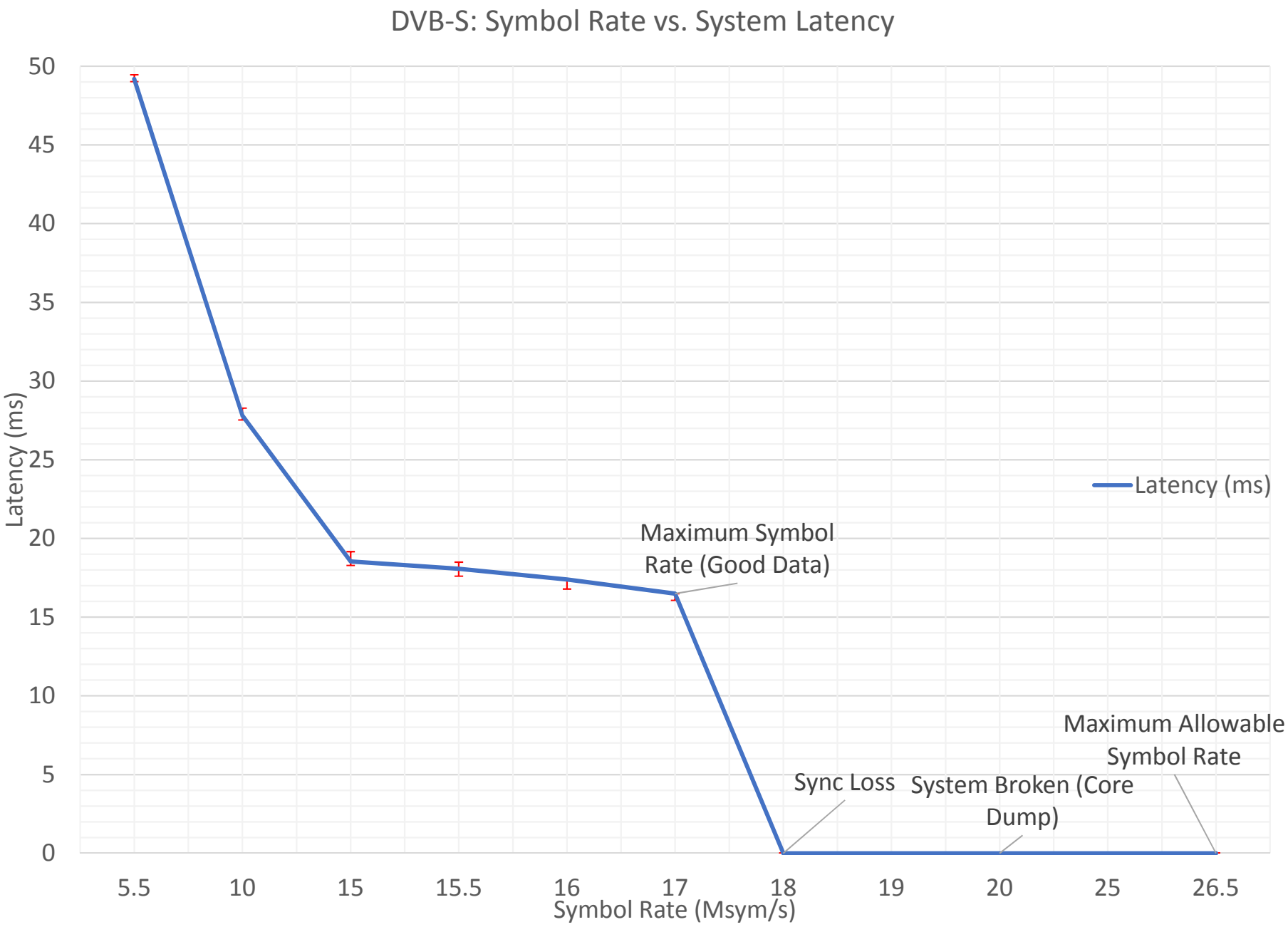


# Latency Measurement System Model



# Latency Results



# Conclusions

- Implemented real-time DVB-S transceiver where all I/Q processing is performed in GPP
- Leveraged multi-core thread management and AVX instructions to reduce computational load
- Agile-based, iterative implementation approach yields operational waveforms at a faster rate than traditional requirements-based approaches
- Opens doors for low-cost platforms capable of supporting complex waveforms
- Enables reference government implementations with increased flexibility