Employing GNU Radio for Robust Testing of the Novel DASH SoC

Drake Silbernagel, Prithvi Hemanth, Saquib Siddiqui, Wylie Standage-Beier, Alex Chiriyath, Daniel W. Bliss

Arizona State University - WISCA/Bliss Labs

Arizona State University

Bliss Laboratory of Information, Signals, and Systems

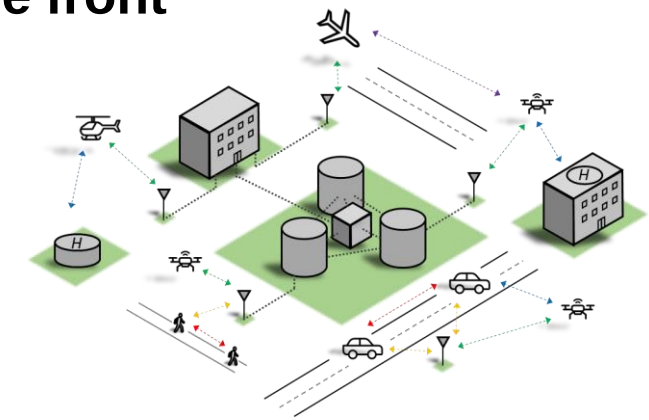
WISCA

Topics

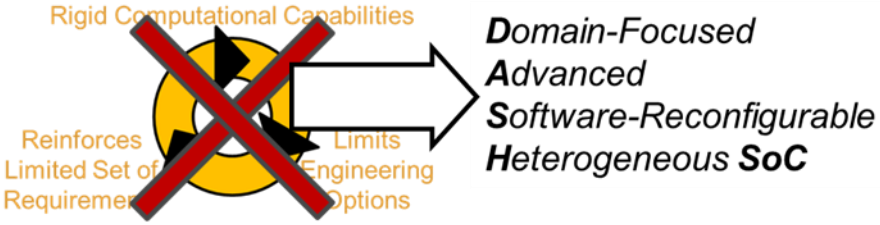
- Introduction
- Motivation
- The DASH SoC
- LDPC Encoding and Decoding
- GNU Radio Setup
- Results
- Conclusion

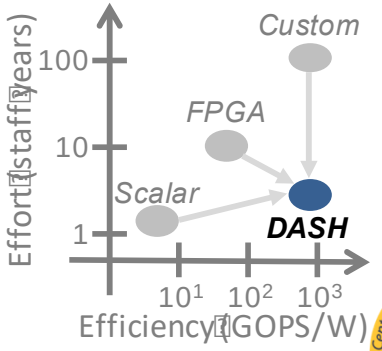
Introduction

- **Next generation wireless systems require cutting-edge front and back-end technology**
  - Next gen systems must provide more functionality
    - Develop systems to do multiple tasks simultaneously
    - Reuse RF/mmWave signals and receivers
  - Traditionally limited by available technologies
    - Inflexible frontends
    - Inflexible computational capabilities



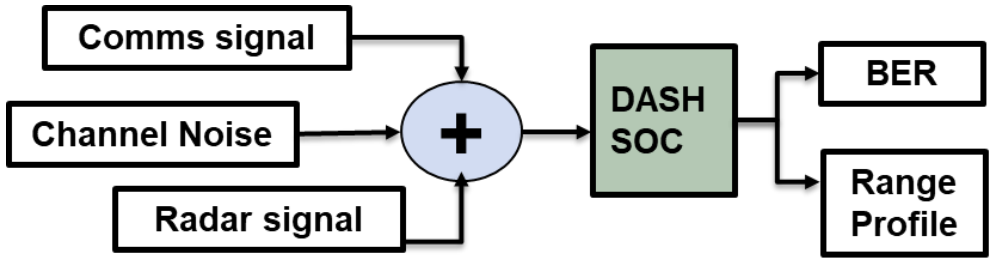
- **DASH SoC & framework enable flexible hardware support**
  - Escape conventional computational system trap
  - Enables the next generation of flexible RF employment
  - Assure non-expert programmability





Motivation

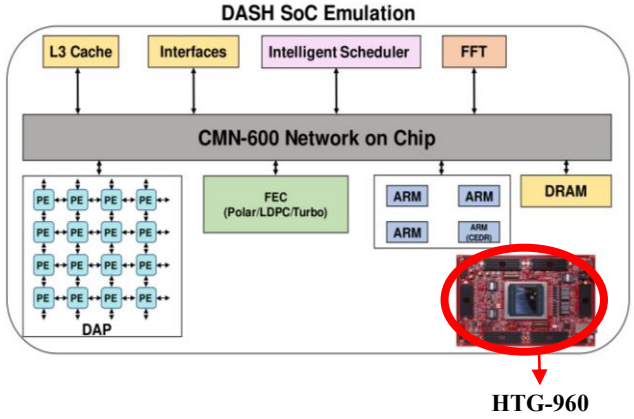
- **What are we doing?**
  - Use GNU Radio to rapidly test DASH SoC w/ OTA data in target domain
    - DASH SoC targets communications & remote sensing domains
- **Why is it important?**
  - Develop robust platform to test novel hardware capabilities
  - Implement end-to-end systems to inspect hardware performance
    - Testing and validating individual hardware components can obfuscate errors
    - End-to-end system tests provide clear picture of functional correctness
  - Evaluate power and latency performance of hardware
- **Looking ahead: Tighter integration w/ GNU Radio, GR-CEDR [1]**
  - Allow GNU Radio to schedule and execute kernels directly on DASH SoC
  - Allows real-time implementation of kernels on cutting-edge hardware



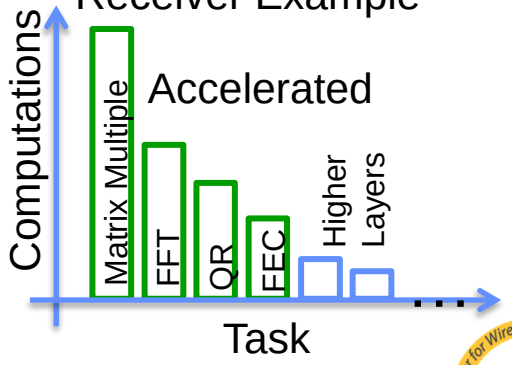
[1] Mack, Joshua, et al. "GNU Radio and CEDR: Runtime Scheduling to Heterogeneous Accelerators." *Proceedings of the GNU Radio Conference*. Vol. 7. No. 1. 2022.

The DASH SoC

- **DASH SoC: Domain-specific coarse scale heterogeneous processor**
  - Compute efficiency & performance near full-custom ASIC
  - Implement domain-specific kernels on mix of custom, flexible accelerators
- **DASH SoC Emulation: Enable rapid prototyping**
  - Platform for extensive testing and performance
- **CEDR : Compiler-integrated extensible DSSoC runtime**
  - Deploy applications on heterogenous hardware automatically & dynamically (Ex: DASH SoC)
- **GR CEDR : Break limitation of accelerator selection at design time**
  - Execute GNU radio blocks on heterogenous accelerators
  - GNU radio is an application within CEDR profiling



Adaptive Multiple-Antenna Receiver Example



LDPC Encoding

- **Systematic block code with sparse parity check matrix (PCM)**
- **Derive PCM & generator matrix (G) from smaller prototype matrix (A)**
  - **Form quasi cyclic PCM (H) from A**
    - **Prototype matrix contains permutations for lifting matrix Q**
    - **$A_{i,j}$ = # column permutations of identity matrix of size Z**
  - **Use Gaussian elimination to form equivalent PCM (H') & G**

Building PCM (Z=3)

$$A = \begin{bmatrix} A_{1,1} & \cdots & A_{1,n_b} \\ \vdots & \ddots & \vdots \\ A_{m_b,1} & \cdots & A_{m_b,n_b} \end{bmatrix};$$

$$Q(1) = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix};$$

$$H = \begin{bmatrix} Q(A_{1,1}) & \cdots & Q(A_{1,n_b}) \\ \vdots & \ddots & \vdots \\ Q(A_{m_b,1}) & \cdots & Q(A_{m_b,n_b}) \end{bmatrix}$$

$H_{(n-k) \times n} \leftrightarrow H'_{(n-k) \times n} \rightarrow H' = [P_{(n-k) \times k} \; I_{(n-k) \times (n-k)}]$, $G_{k \times n} = [I_{k \times k} \; P^T_{k \times (n-k)}]$

H = Parity check matrix, P = Parity Matrix, I = Identity Matrix
n = number of encoded bits, k = number of message bits

- **Function signature of LDPC encoder**
  - **Obtain the generator matrix (G) from the prototype matrix (A)**

```
void DASH_FEC_LDPC_GEN_MAT_ENC(size_t n, size_t k, size_t Z, int8_t **A, uint8_t **H, uint8_t **G);
```

  - **Use the generator matrix to encode information**

```
void DASH_FEC_LDPC_ENCODE_ENC(size_t n, size_t k, uint8_t **G, uint8_t **message, uint8_t **encoded);
```

LDPC Decoding

- **LDPC Decoder API Call**

```
void DASH_FEC_LDPC_DEC(uint8_t n, uint8_t k, uint8_t z, std::valarray<int8_t> P, std::valarray<int8_t> LLR, fec_result_t* output);
```

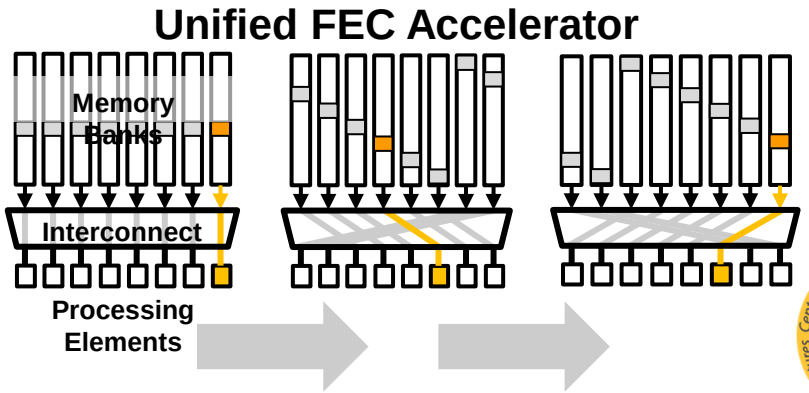
- **LDPC Decoding Algorithm: Min – Sum Algorithm**

Iterative process computing LLRs from Vnode to Cnode

$$\eta_{mn} = \prod_{k,\mathbf{H}(m,k)=1,k\neq m} sign(\lambda_{km}) \min_{k}(\lambda_{km}) \qquad \lambda_{mn} = \sum_{k,\mathbf{H}(k,n)=1,k\neq n} \eta_{kn} + \eta_{0n}$$

- **Capable of quickly switching to each code type**
- **Reconfigurable to adapt to changing standards**

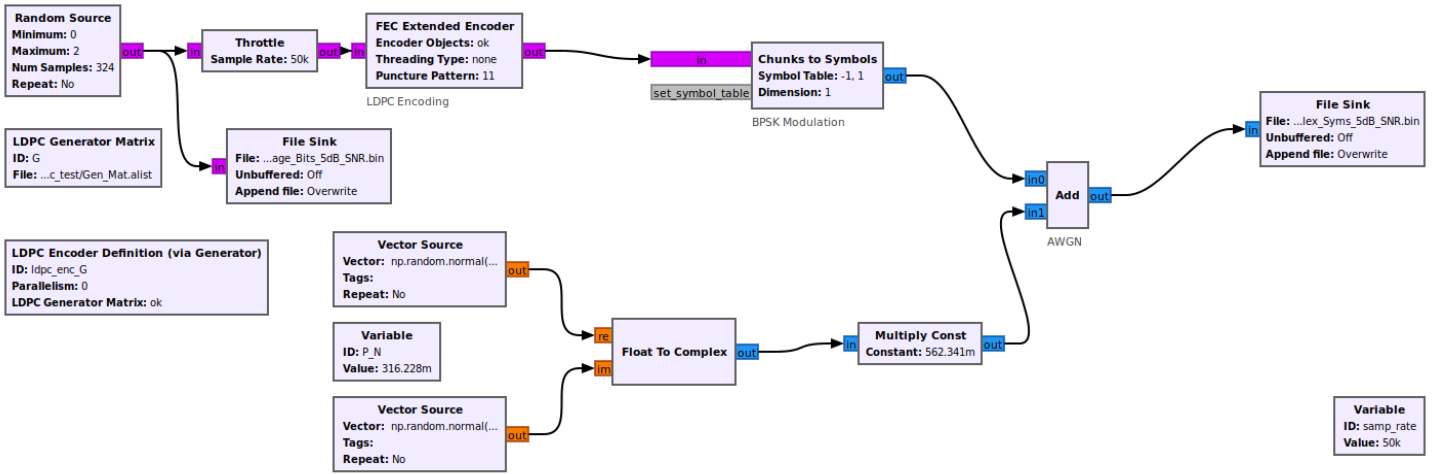
Turbo		LDPC		Polar	
Param.	Range	Param.	Range	Param.	Range
N	$\leq$3072	N	$\leq$1944	N	$\leq$4096
R	1/3	R	arbitrary	R	arbitrary
k	3$\sim$7	Z	$\leq$ 81	F	arbitrary
P	arbitrary	**H**	QC, arbitrary pattern		
I	arbitrary				



Validation Simulation Setup

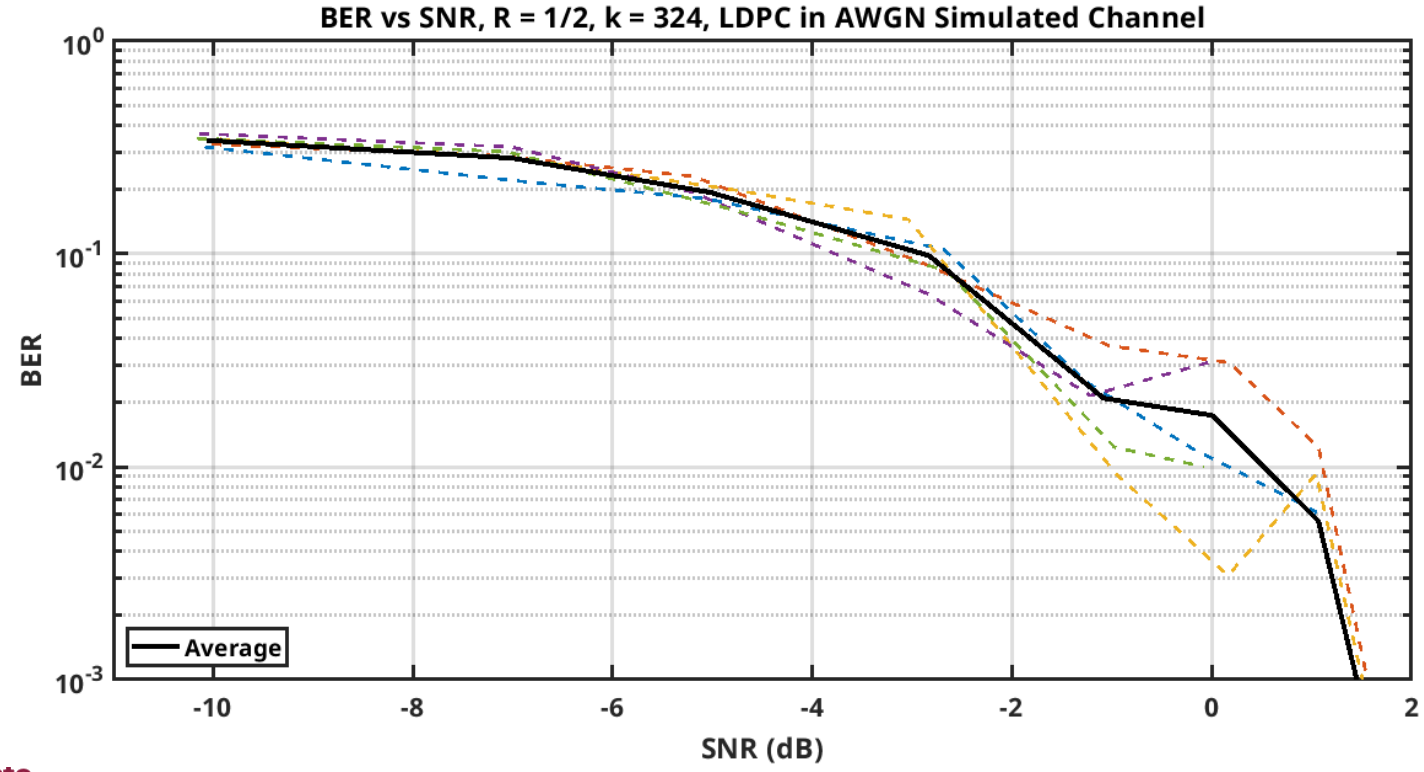
- **LDPC Parameters:**
  - Rate ½ code, k = 324, N = 648
  - Generation Matrix created from DASH SoC

- **Simple Simulation GRC Flow Graph**



Results – Validation Testing with GNU Radio

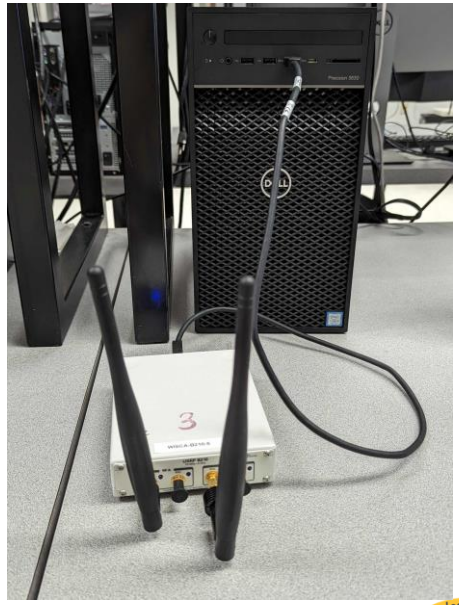
- **BER vs SNR under a Simple AWGN Channel Model.**

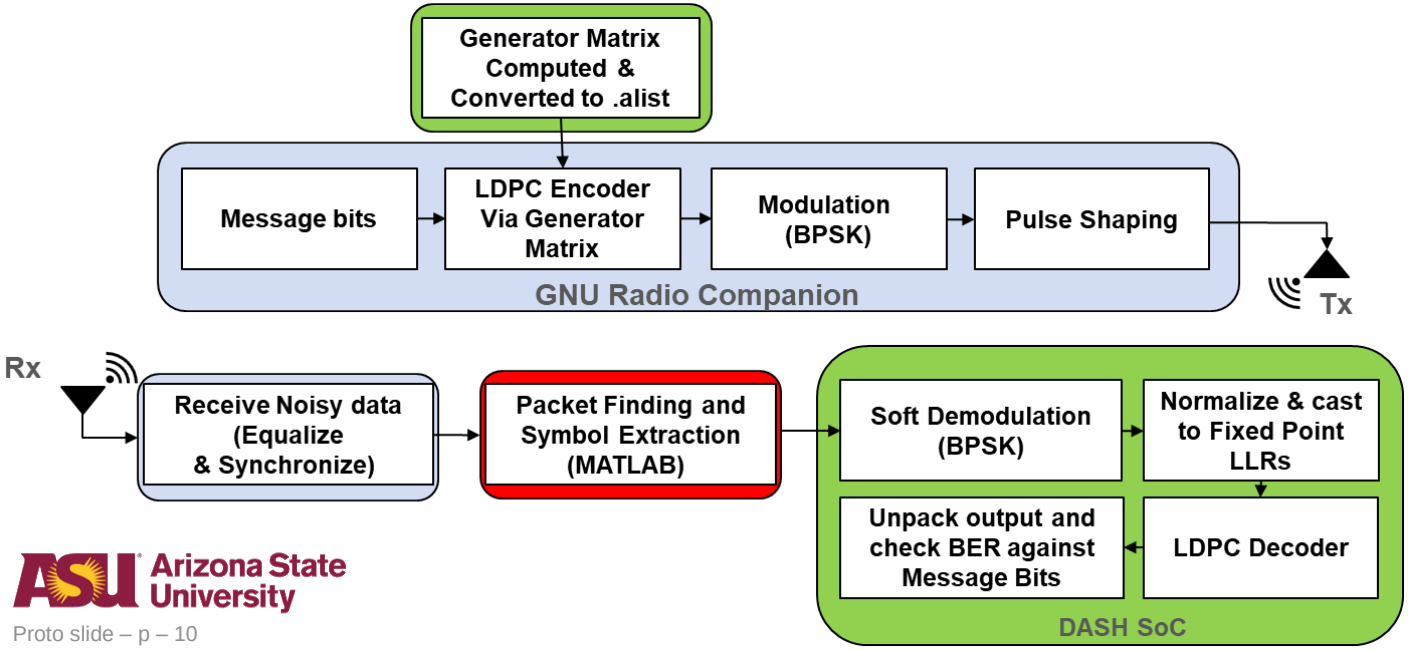


Over-The-Air Experimental Setup

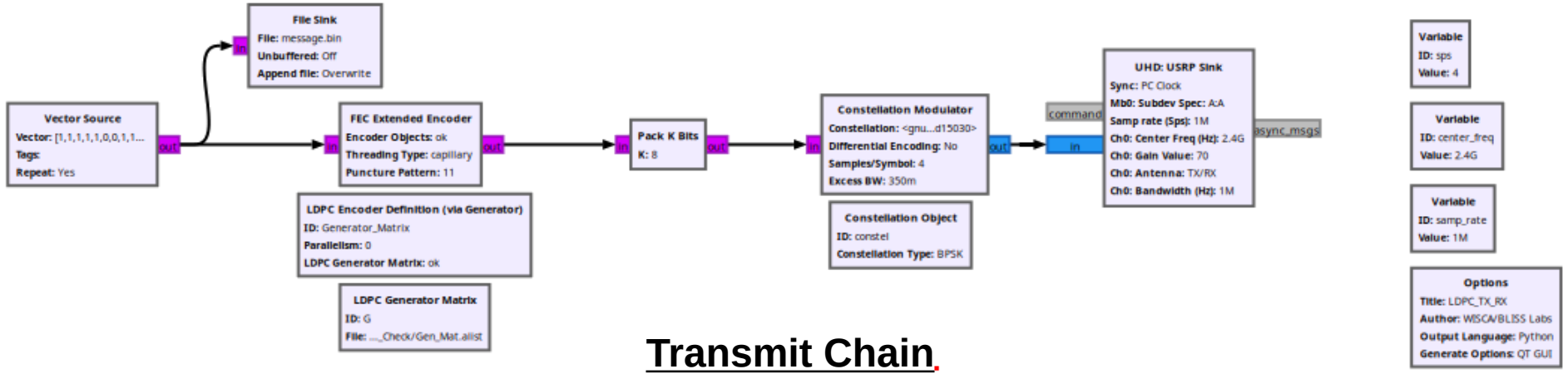
- **Experiment Parameters**
  - Single USRP B210 → Tx from A to B
  - 3dBi Omni-directional antennas
  - Center Frequency: 2.4GHz
  - Bandwidth/Sampling Rate: 1 MHz
  - 70 dB Tx gain, 50 dB Rx gain, 30 dB atten.

- **LDPC Parameters:**
  - Rate ½ code, k = 324, N = 648
  - Generation Matrix created from DASH SoC

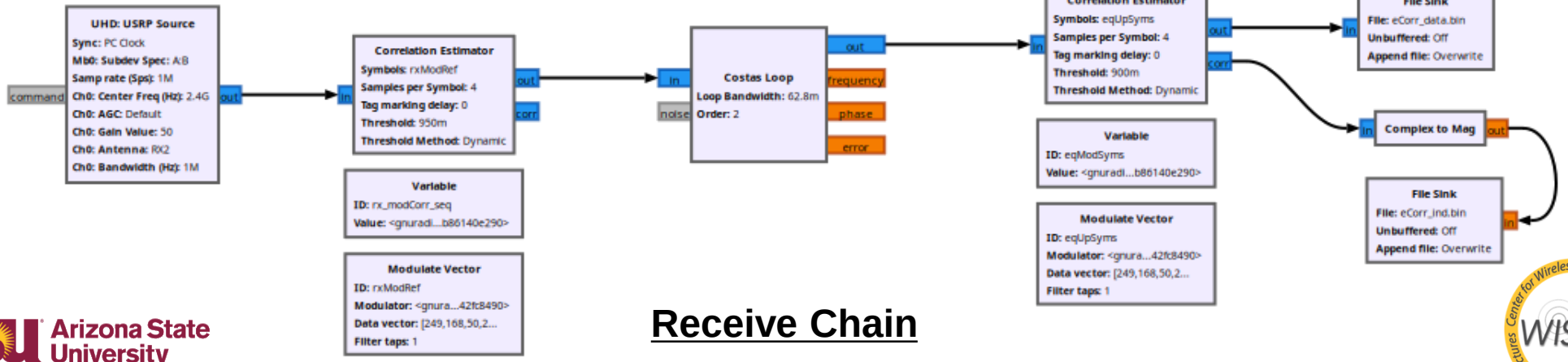




GNU Radio Over-The-Air Flowgraph



Transmit Chain



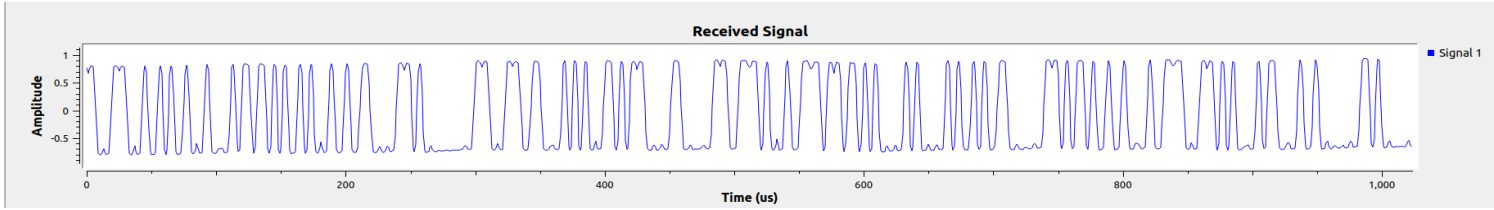
Receive Chain

Arizona State University

Center for Wireless Information Systems and Computational Architectures WISCA

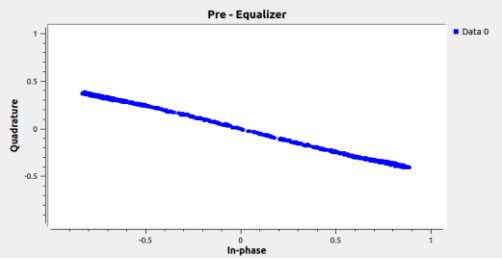
Results: Over-the-Air Experiments

Received Waveform

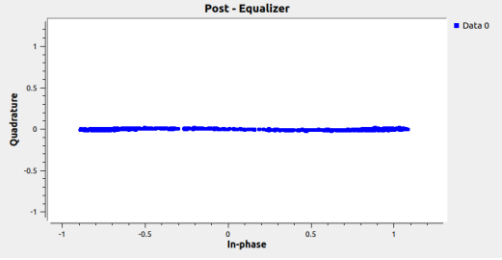


Received pulse shaped waveform at 4 samples per symbol

Constellation pre-eq



Constellation post-eq




```
Decoded Bits
1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1,
Bit Error Rate: 0.000000
```

LDPC decoded bits with a bit error rate of 0

Conclusion

- **GNU Radio, specifically GNU Radio Companion, provides a platform of rapid testing for novice and knowledgeable users alike. Both through simulating various test cases (waveforms, encoding type, etc) and practical applications of testing over-the-air.**

- **In combination with last year's GR-CEDR presentation, the joint findings demonstrate the potential integration of GNU Radio with the DASH SoC to provide a platform of robust testing.**