BBC Communications in GNURadio

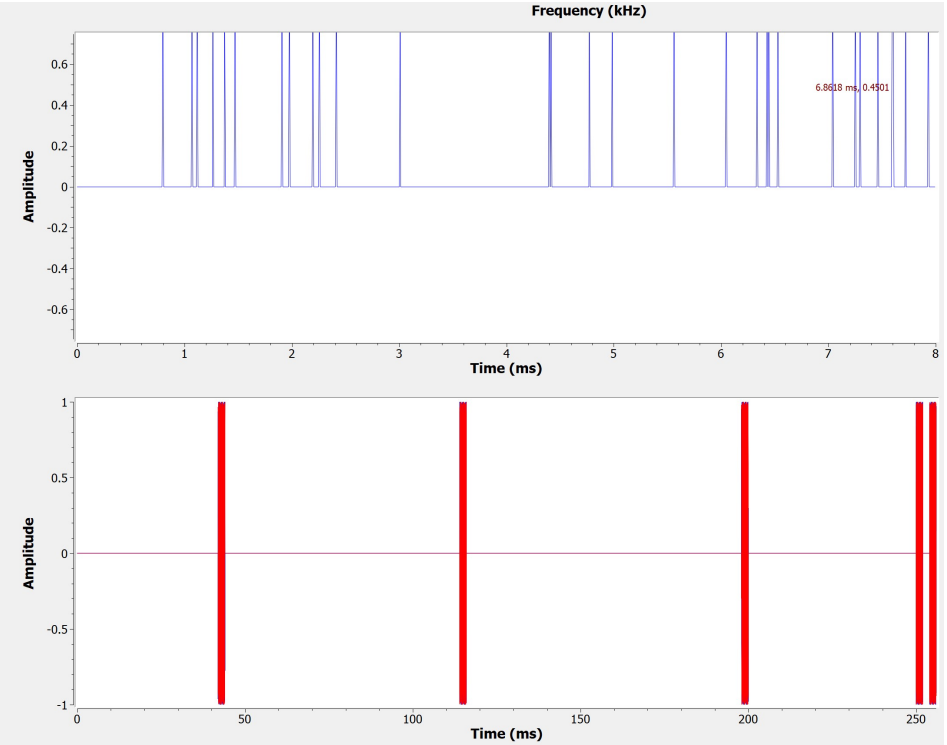
N. Rogers and J. Morrison

United States Air Force Academy
Department of Electrical and
Computer Engineering

September 2022

Overview

- BBC Overview
- GNURadio implementation
- Lessons learned
- Future Development



What is BBC?

UNITED STATES AIR FORCE ACADEMY

- Keyless
- Concurrent-codes
- Spread spectrum
- Jam-resistant (not LPI)
- Invertable hash algorithm encoding
- Limitation: Low throughput
- Limitation: modulation type
- Limit: Mark Density

No pre-shared key

What is BBC?

- Keyless
- Concurrent-codes
- Spread spectrum
- Jam-resistant (not LPI)
- Invertable hash algorithm encoding
- Limitation: Low throughput
- Limitation: modulation type
- Limit: Mark Density

Can encode multiple messages in a single codeword

What is BBC?

- Keyless
- Concurrent-codes
- Spread spectrum
- Jam-resistant (not LPI)
- Invertable hash algorithm encoding
- Limitation: Low throughput
- Limitation: modulation type
- Limit: Mark Density

Transmitted signal size greater than message

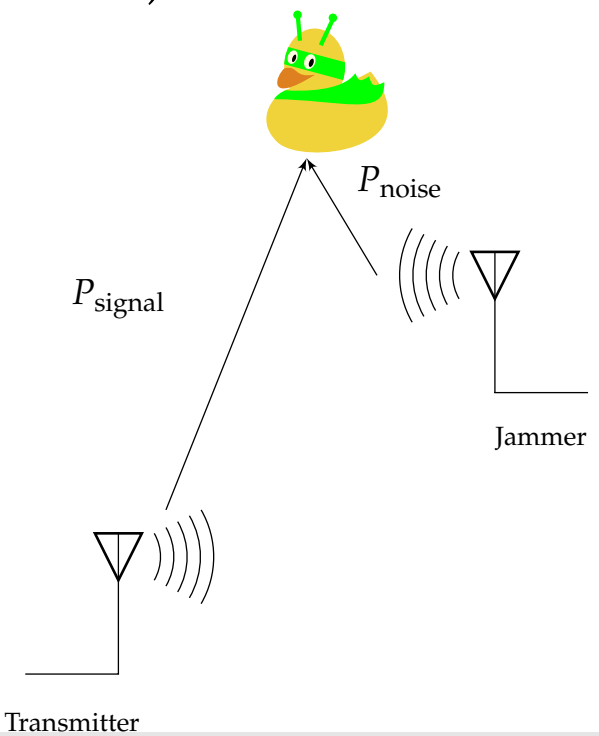
What is BBC?

- Keyless
- Concurrent-codes
- Spread spectrum
- Jam-resistant (not LPI)
- Invertable hash algorithm encoding
- Limitation: Low throughput
- Limitation: modulation type
- Limit: Mark Density

UNITED STATES AIR FORCE ACADEMY

$\longrightarrow$ Cannot remove information that exists

$\longrightarrow$ Additional information in codeword can be ignored (to an extent)



What is BBC?

- Keyless
- Concurrent-codes
- Spread spectrum
- Jam-resistant (not LPI)
- Invertable hash algorithm encoding
- Limitation: Low throughput
- Limitation: modulation type
- Limit: Mark Density

What is BBC?

- Keyless
- Concurrent-codes
- Spread spectrum
- Jam-resistant (not LPI)
- Invertable hash algorithm encoding
- Limitation: Low throughput
- Limitation: modulation type
- Limit: Mark Density

What is BBC?

- Keyless
- Concurrent-codes
- Spread spectrum
- Jam-resistant (not LPI)
- Invertable hash algorithm encoding
- Limitation: Low throughput
- Limitation: modulation type
- Limit: Mark Density

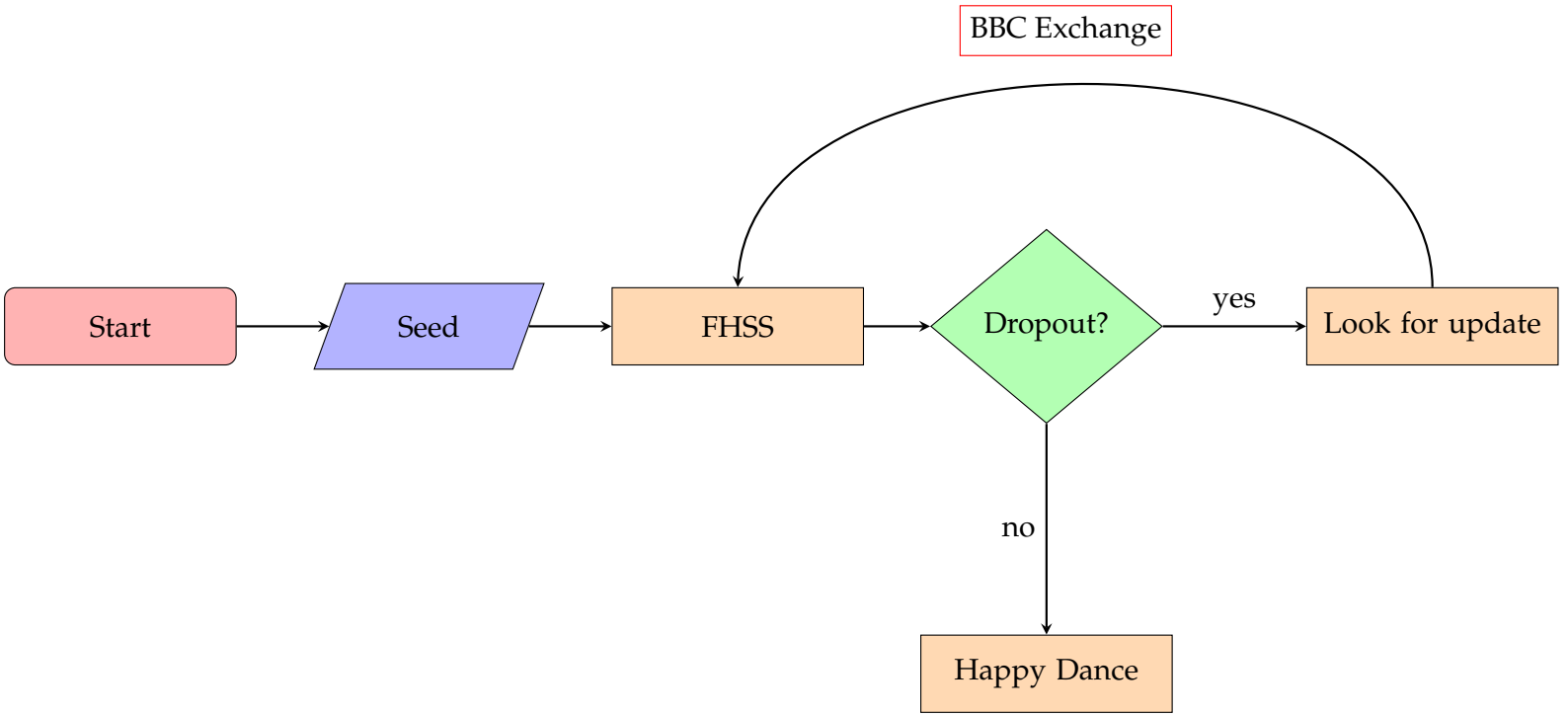
UNITED STATES AIR FORCE ACADEMY

Mark Density

$$\mu = \frac{n_{\text{bits, message}}}{n_{\text{bits, codeword}}}$$

$$\text{Limit} \longrightarrow 0.5$$

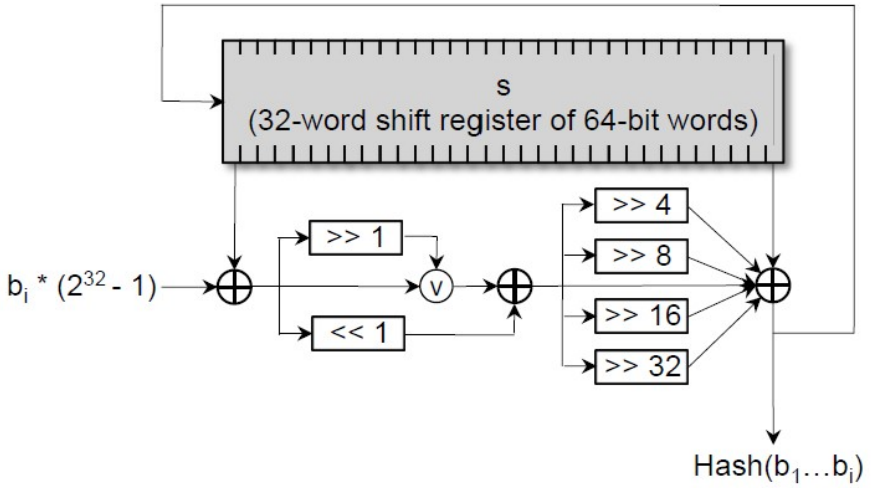
Use Case - FHSS Sequence



Encoding – the Glowworm Hash

UNITED STATES AIR FORCE ACADEMY

- Input is length of codeword
- Output is "sparse" codeword
- Shift register produces mark locations in codeword
- Shift register values depend on current substring
- Example ⟶
  Message = 0011 0010
  Codeword = 16bits

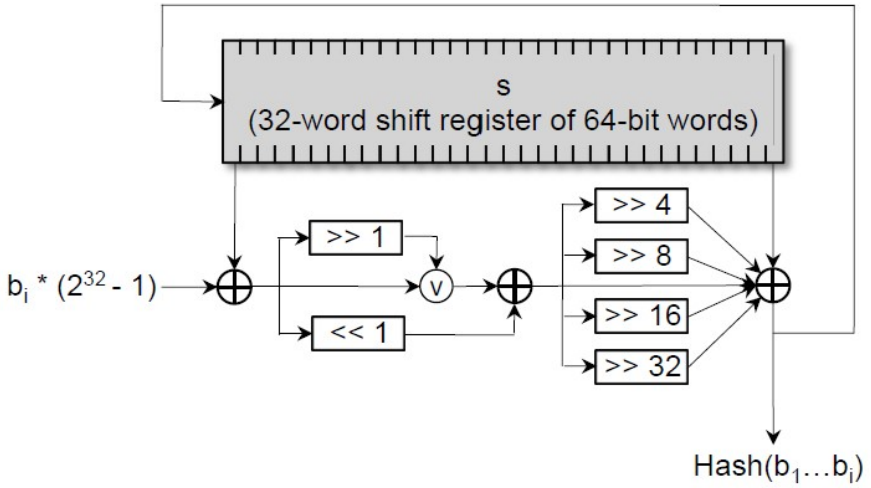


[1] Baird, Carlisle, Bahn. The Glowworm Hash: Increased Speed and Security for BBC Unkeyed Jam Resistance. 2012.

Encoding – the Glowworm Hash

UNITED STATES AIR FORCE ACADEMY

- Input is length of codeword
- Output is "sparse" codeword
- Shift register produces mark locations in codeword
- Shift register values depend on current substring
- Example ⟶
  Message = 0011 0010
  Codeword = 16bits

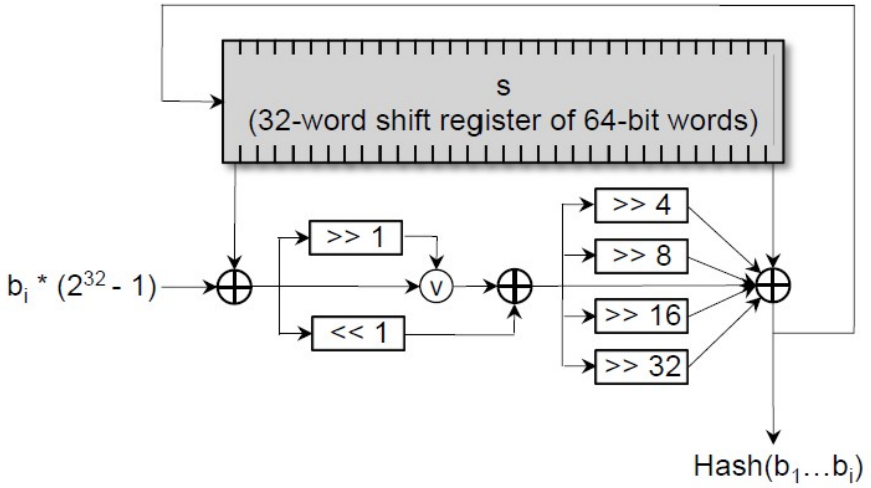


[1] Baird, Carlisle, Bahn. The Glowworm Hash: Increased Speed and Security for BBC Unkeyed Jam Resistance. 2012.

Encoding – the Glowworm Hash

UNITED STATES AIR FORCE ACADEMY

- Input is length of codeword
- Output is "sparse" codeword
- Shift register produces mark locations in codeword
- Shift register values depend on current substring
- Example ⟶
  Message = 0011 0010
  Codeword = 16bits

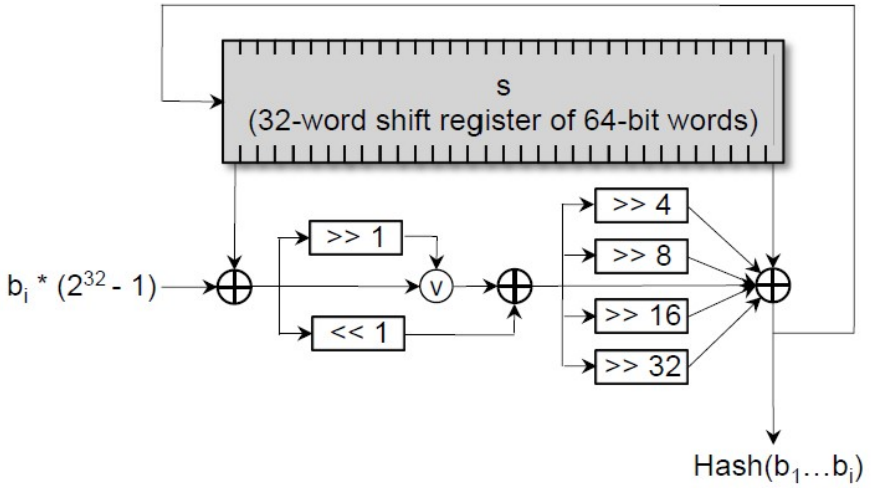


[1] Baird, Carlisle, Bahn. The Glowworm Hash: Increased Speed and Security for BBC Unkeyed Jam Resistance. 2012.

Encoding – the Glowworm Hash

UNITED STATES AIR FORCE ACADEMY

- Input is length of codeword
- Output is "sparse" codeword
- Shift register produces mark locations in codeword
- Shift register values depend on current substring
- Example ⟶
  Message = 0011 0010
  Codeword = 16bits



[1] Baird, Carlisle, Bahn. The Glowworm Hash: Increased Speed and Security for BBC Unkeyed Jam Resistance. 2012.

Encoding – the Glowworm Hash

UNITED STATES AIR FORCE ACADEMY

- Input is length of codeword
- Output is "sparse" codeword
- Shift register produces mark locations in codeword
- Shift register values depend on current substring
- Example ⟶
  Message = 0011 0010
  Codeword = 16bits

MESSAGE:

0	0	1	1	0	0	0	0
0	1	2	3	4	5	6	7

GLOWWORM HASH:

CODEWORD:

0	0	0	0	0	0	0	0
0	1	2	3	4	5	6	7 . . .

0	0	0	0	0	0	0	0
8	9	1	11	12	13	14	15

Encoding – the Glowworm Hash

UNITED STATES AIR FORCE ACADEMY

- Input is length of codeword
- Output is "sparse" codeword
- Shift register produces mark locations in codeword
- Shift register values depend on current substring
- Example ⟶
  Message = 0011 0010
  Codeword = 16bits

MESSAGE:

0	0	1	1	0	0	0	0
0	1	2	3	4	5	6	7

GLOWWORM HASH: 5

CODEWORD:

0	0	0	0	1	0	0	0
0	1	2	3	4	5	6	7 . . .

0	0	0	0	0	0	0	0
8	9	1	11	12	13	14	15

Encoding – the Glowworm Hash

- Input is length of codeword
- Output is "sparse" codeword
- Shift register produces mark locations in codeword
- Shift register values depend on current substring
- Example ⟶
  Message = 0011 0010
  Codeword = 16bits

MESSAGE:

0	0	1	1	0	0	0	0
0	1	2	3	4	5	6	7

GLOWWORM HASH: 0

CODEWORD:

1	0	0	0	1	0	0	0
0	1	2	3	4	5	6	7 . . .
0	0	0	0	0	0	0	0
8	9	1	11	12	13	14	15

Encoding – the Glowworm Hash

UNITED STATES AIR FORCE ACADEMY

- Input is length of codeword
- Output is "sparse" codeword
- Shift register produces mark locations in codeword
- Shift register values depend on current substring
- Example ⟶
  Message = 0011 0010
  Codeword = 16bits

MESSAGE:

0	0	1	1	0	0	0	0
0	1	2	3	4	5	6	7

GLOWWORM HASH: 14

CODEWORD:

1	0	0	0	1	0	0	0
0	1	2	3	4	5	6	7 . . .

0	0	0	0	0	0	1	0
8	9	1	11	12	13	14	15

Encoding – the Glowworm Hash

UNITED STATES AIR FORCE ACADEMY

- Input is length of codeword
- Output is "sparse" codeword
- Shift register produces mark locations in codeword
- Shift register values depend on current substring
- Example ⟶
  Message = 0011 0010
  Codeword = 16bits

MESSAGE:

0	0	1	1	0	0	0	0
0	1	2	3	4	5	6	7

GLOWWORM HASH: 2

CODEWORD:

1	0	1	0	1	0	0	0
0	1	2	3	4	5	6	7 . . .

0	0	0	0	0	0	1	0
8	9	1	11	12	13	14	15

Encoding – the Glowworm Hash

UNITED STATES AIR FORCE ACADEMY

- Input is length of codeword
- Output is "sparse" codeword
- Shift register produces mark locations in codeword
- Shift register values depend on current substring
- Example ⟶
  Message = 0011 0010
  Codeword = 16bits

MESSAGE:

0	0	1	1	0	0	0	0
0	1	2	3	4	5	6	7

GLOWWORM HASH: 0

CODEWORD:

1	0	1	0	1	0	0	0
0	1	2	3	4	5	6	7 . . .

0	0	0	0	0	0	1	0
8	9	1	11	12	13	14	15

Encoding – the Glowworm Hash

UNITED STATES AIR FORCE ACADEMY

- Input is length of codeword
- Output is "sparse" codeword
- Shift register produces mark locations in codeword
- Shift register values depend on current substring
- Example ⟶
  Message = 0011 0010
  Codeword = 16bits

MESSAGE:

0	0	1	1	0	0	0	0
0	1	2	3	4	5	6	7

GLOWWORM HASH: 0

CODEWORD:

1	0	1	0	1	0	0	0
0	1	2	3	4	5	6	7 . . .

0	0	0	0	0	0	1	0
8	9	1	11	12	13	14	15

Encoding – the Glowworm Hash

- Input is length of codeword
- Output is "sparse" codeword
- Shift register produces mark locations in codeword
- Shift register values depend on current substring
- Example ⟶
  Message = 0011 0010
  Codeword = 16bits

MESSAGE:

0	0	1	1	0	0	0	0
0	1	2	3	4	5	6	7

GLOWWORM HASH: 3

CODEWORD:

1	0	1	1	1	0	0	0
0	1	2	3	4	5	6	7 . . .

0	0	0	0	0	0	1	0
8	9	1	11	12	13	14	15

Encoding – the Glowworm Hash

UNITED STATES AIR FORCE ACADEMY

- Input is length of codeword
- Output is "sparse" codeword
- Shift register produces mark locations in codeword
- Shift register values depend on current substring
- Example ⟶
  Message = 0011 0010
  Codeword = 16bits

MESSAGE:

0	0	1	1	0	0	0	0
0	1	2	3	4	5	6	7

GLOWWORM HASH: 12

CODEWORD:

1	0	1	1	1	0	0	0
0	1	2	3	4	5	6	7 . . .

0	0	0	0	1	0	1	0
8	9	1	11	12	13	14	15

Encoding – the Glowworm Hash

- Input is length of codeword
- Output is "sparse" codeword
- Shift register produces mark locations in codeword
- Shift register values depend on current substring
- Example ⟶
  Message = 0011 0010
  Codeword = 16bits
- $\mu = \frac{b_{\text{message}}}{b_{\text{codeword}}} = \frac{6}{16} = 0.375$

MESSAGE:

0	0	1	1	0	0	0	0
0	1	2	3	4	5	6	7

GLOWWORM HASH:

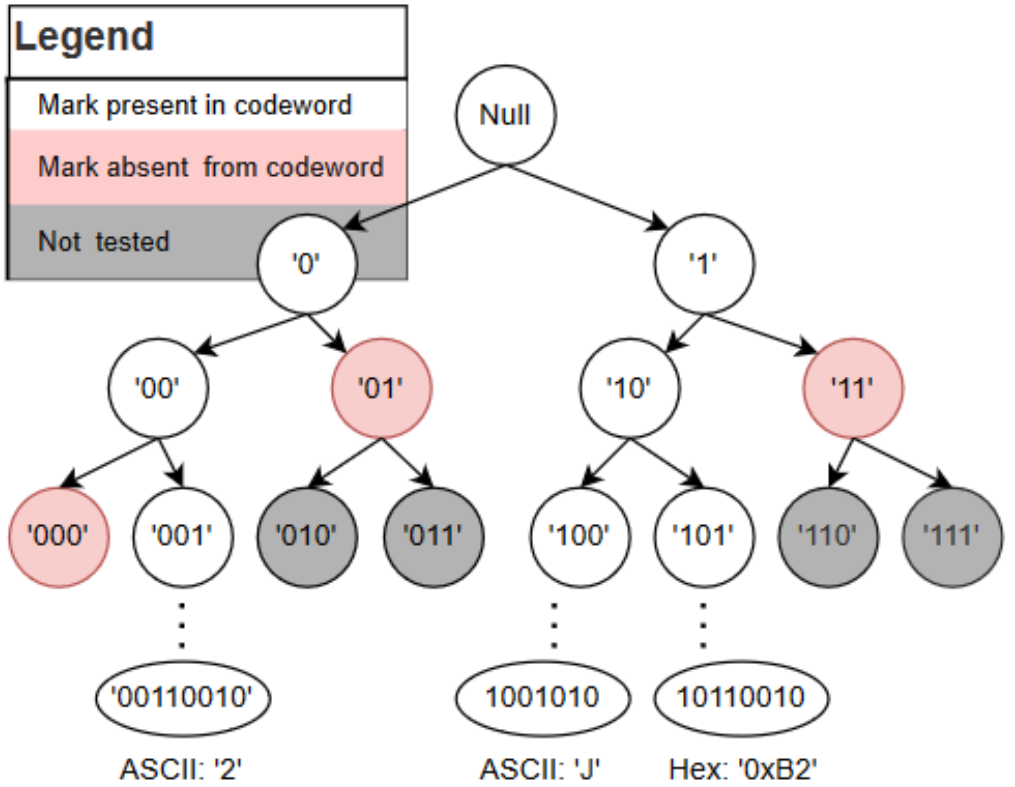
CODEWORD:

1	0	1	1	1	0	0	0
0	1	2	3	4	5	6	7 . . .

0	0	0	0	1	0	1	0
8	9	1	11	12	13	14	15

Decoding

- Depth-First Search:
  - Inuitive to implement,
  - Stack depth
  - Inefficient use of memory
- Breadth-First Search
  - More difficult to implement
  - Memory / CPU efficient



JAM-RESISTANCE

UNITED STATES AIR FORCE ACADEMY

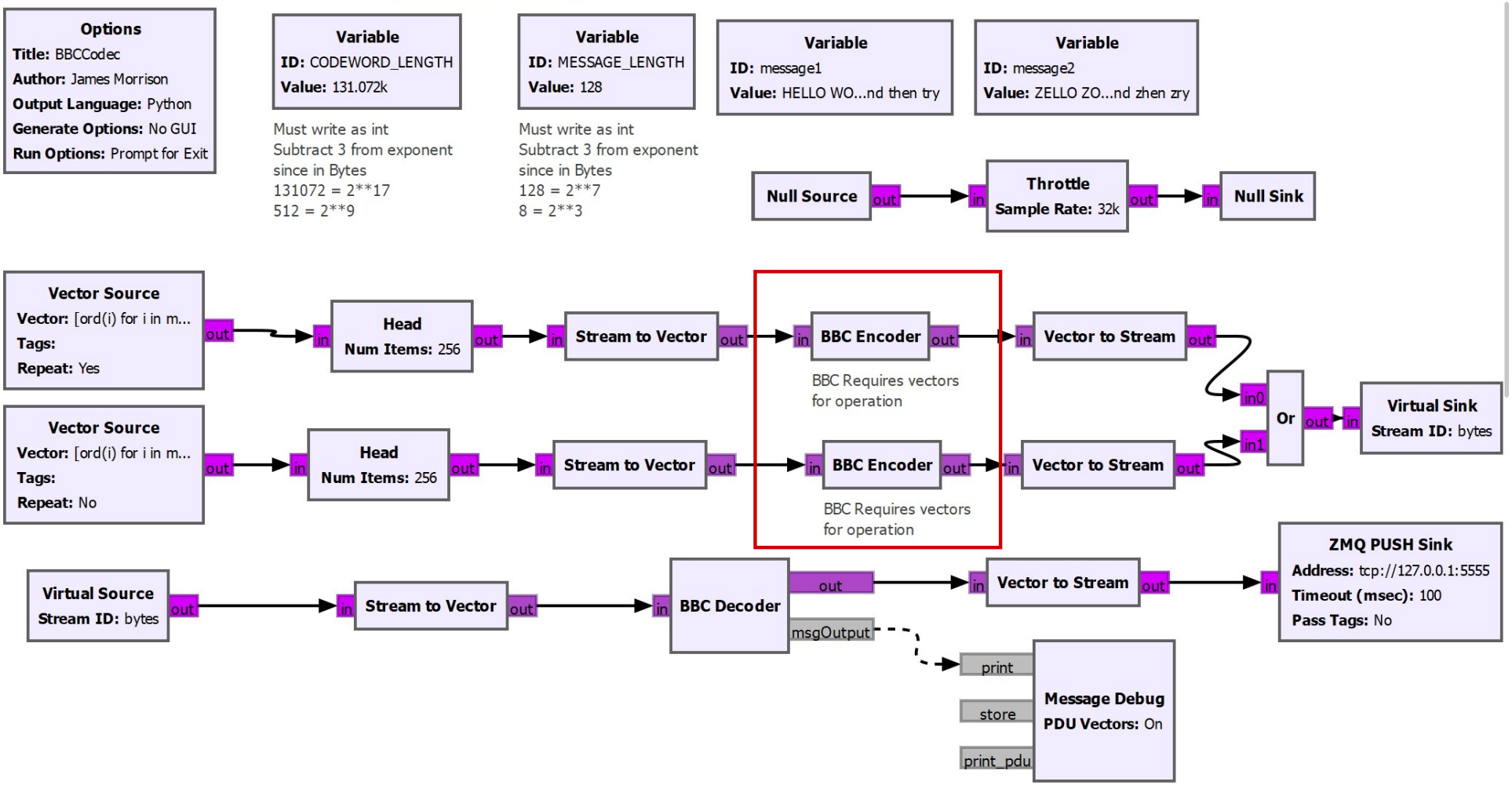
Jamming acts as a bit-wise "OR"

Origin	ASCII Character	BBC Codeword
Original message (TX)	2	10011100 00001010
Jamming signal	!	01000001 10001101
Channel packet (OR)	**N/A**	**11011101 10001111**

- Broadband noise attack $\longrightarrow$ enough energy wins (always)
- Random marks add additional messages, don't affect existing
- Checksum bits protect against hallucinations

BBC In GNURadio - Encode/Decode

UNITED STATES AIR FORCE ACADEMY



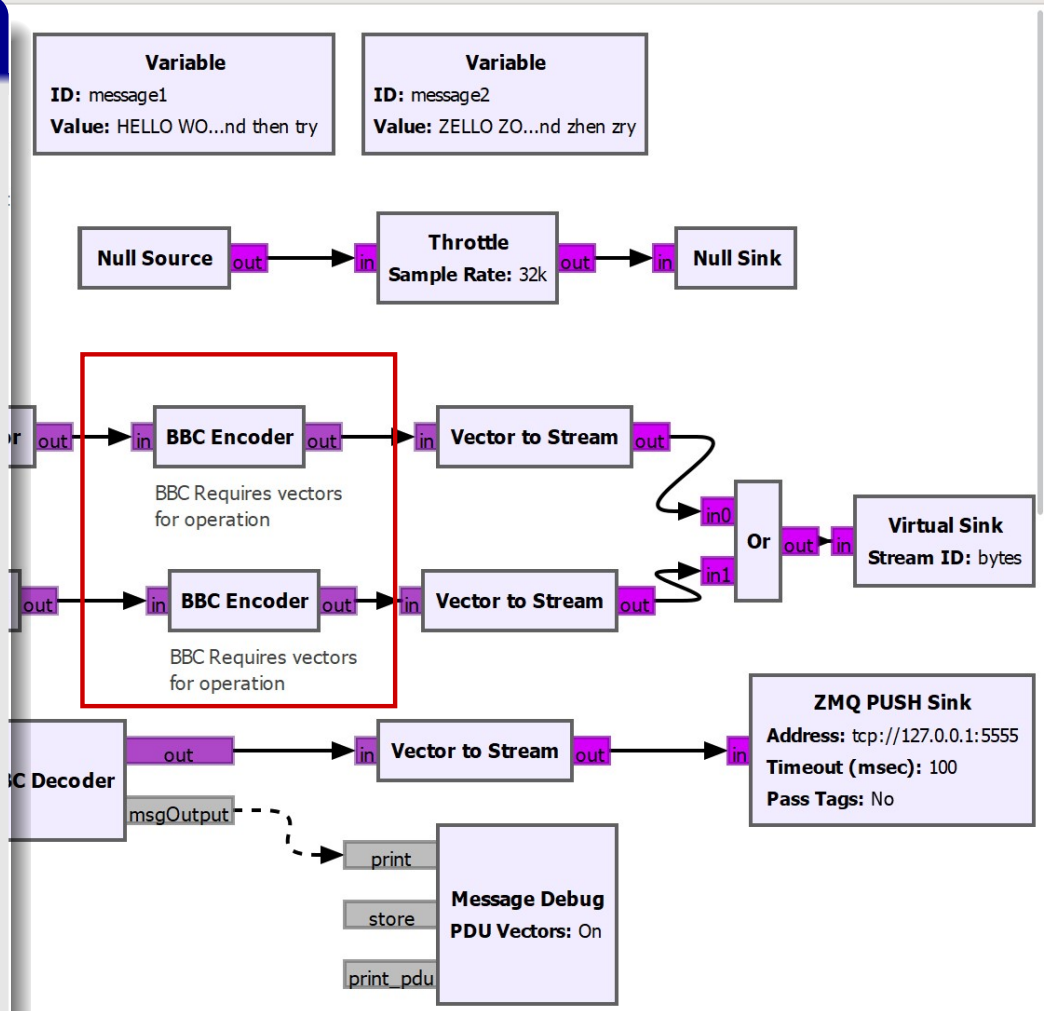
HELLO WORLD! Welcome to BBC in GNURadio. This is a jam-resistant codec, and we are sending messages, encoding them, and then try
ZELLO ZORLD! Zelcome zo ZBC zn ZNURadio. Zhis zs z zam-zesistant zodec, znd ze zre zending zessages, zncoding zhem, znd zhen zry

BBC In GNURadio - Encode/Decode

UNITED STATES AIR FORCE ACADEMY

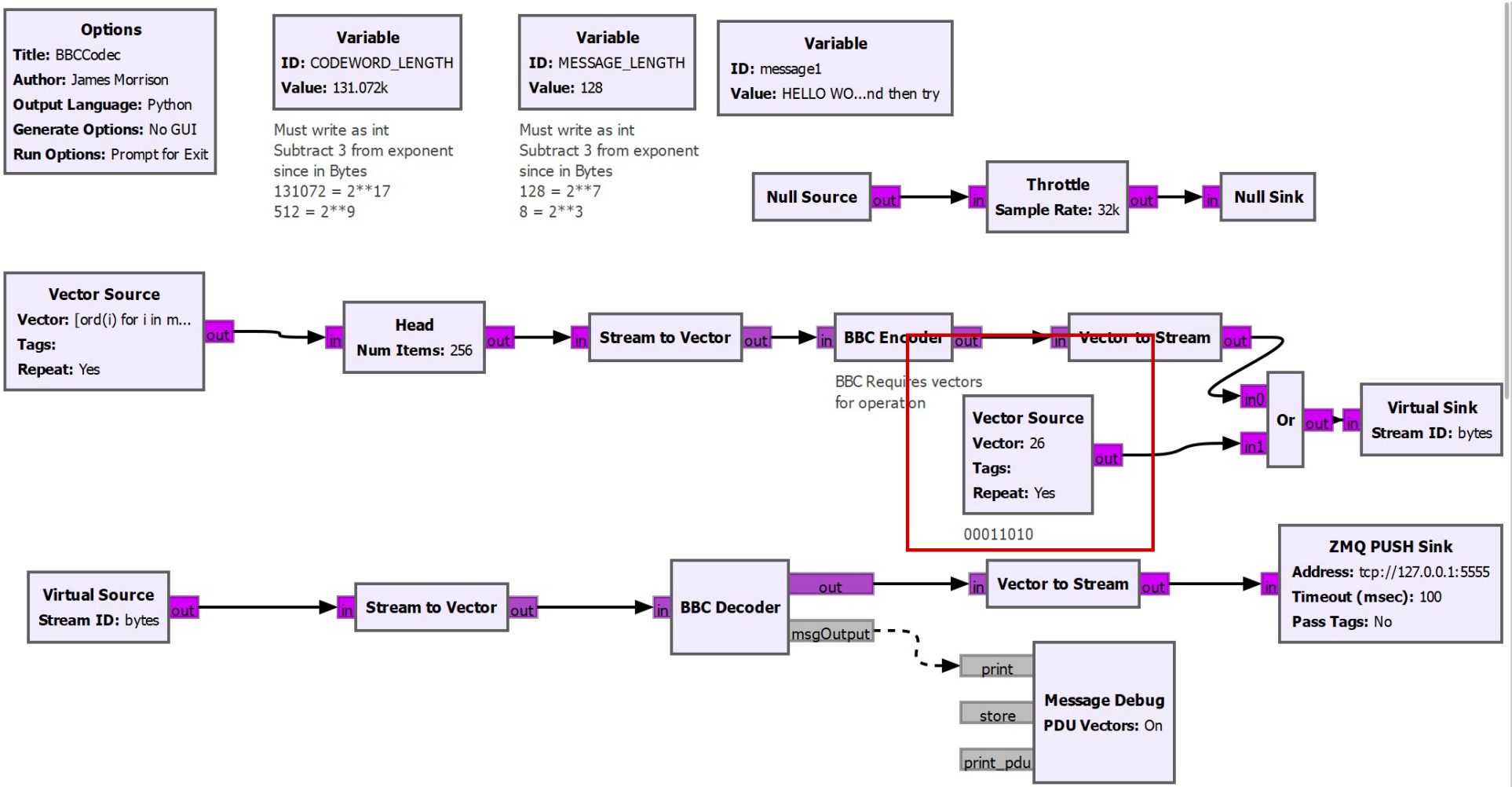
Details

- Encoder
  - Input = bytes vectors of length MESSAGE LENGTH
  - Output = bytes vectors of length CODEWORD LENGTH
- Decoder
  - Input =bytes vectors of length CODEWORD LENGTH
  - Output = bytes vectors of length MESSAGE LENGTH



HELLO WORLD! Welcome to BBC in GNURadio. This is a jam-resistant codec, and we are sending messages, encoding them, and then try
ZELLO ZORLD! Zelcome zo ZBC zn ZNURadio. Zhis zs z zam-zesistant zodec, znd ze zre zending zessages, zncoding zhem, znd zhen zry

Basic Testing – Packet Density Limit

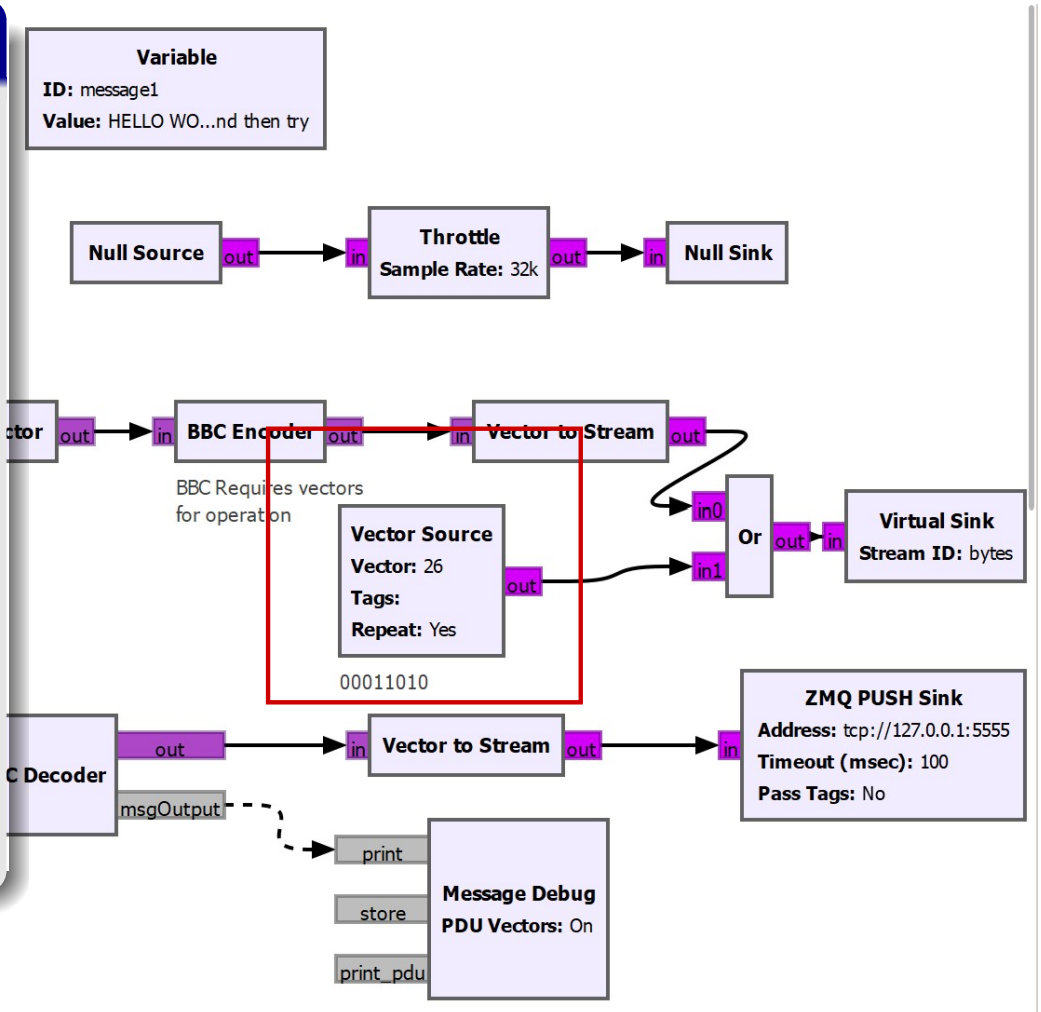


Basic Testing – Packet Density Limit

UNITED STATES AIR FORCE ACADEMY

Details

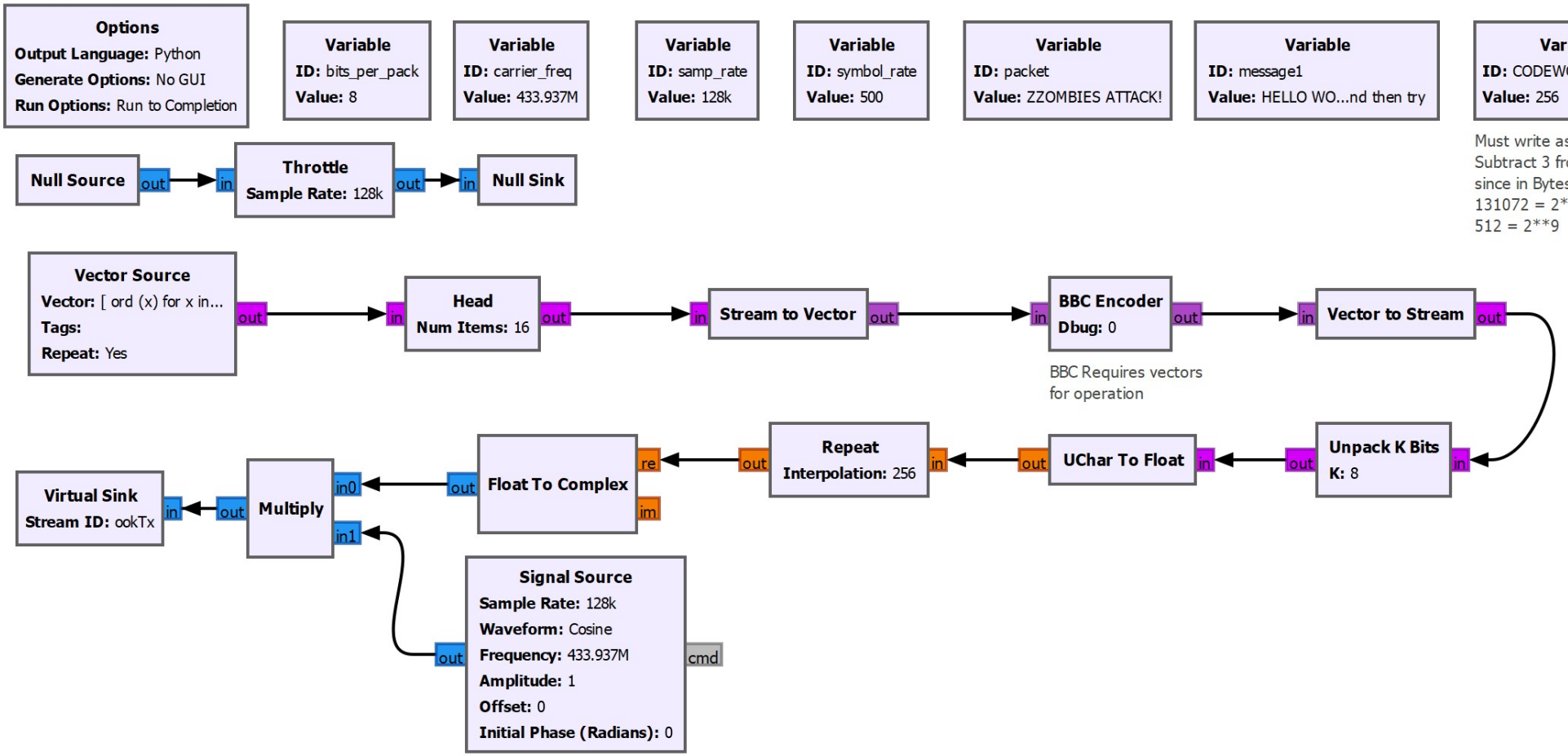
- Replaced message2 with vector source
- Idea is to test μ =50% mark density
- $26_{10} = 00011010_2$ (hovering around limit)
  PASS
- $27_{10} = 00011011_2$ (over limit)
  FAIL



OOK Implementation

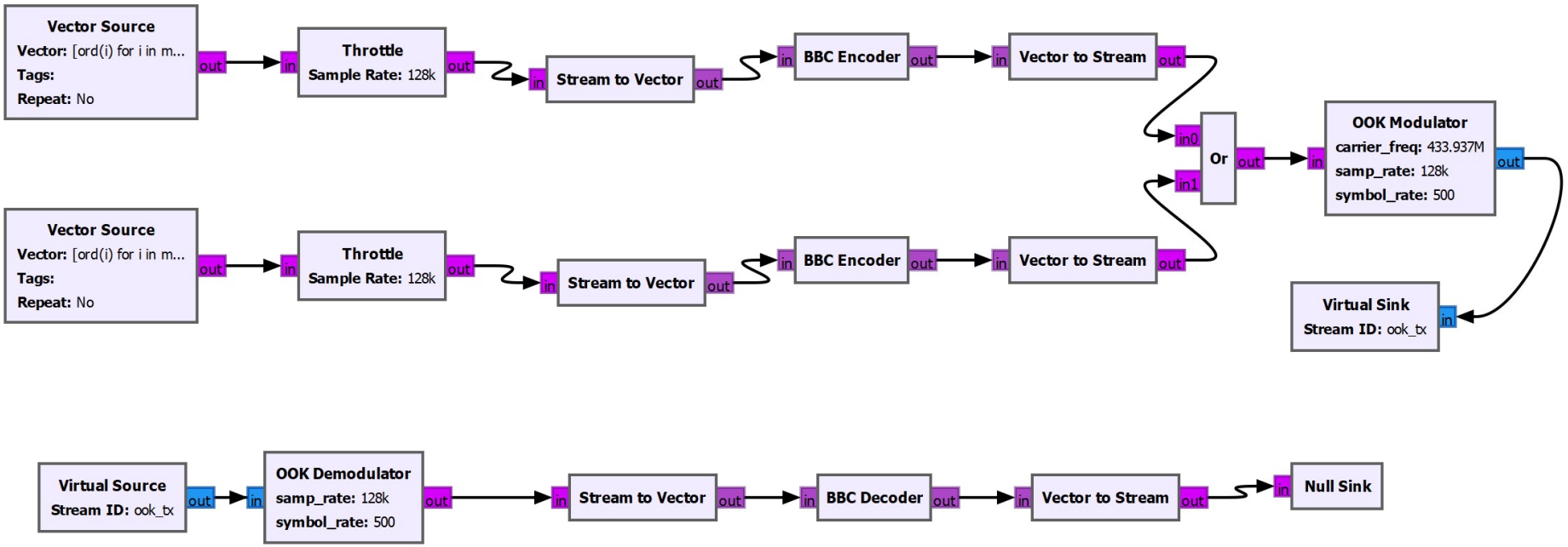
UNITED STATES AIR FORCE ACADEMY

- Chose OOK for simplicity
- Complements jam-resistance of BBC

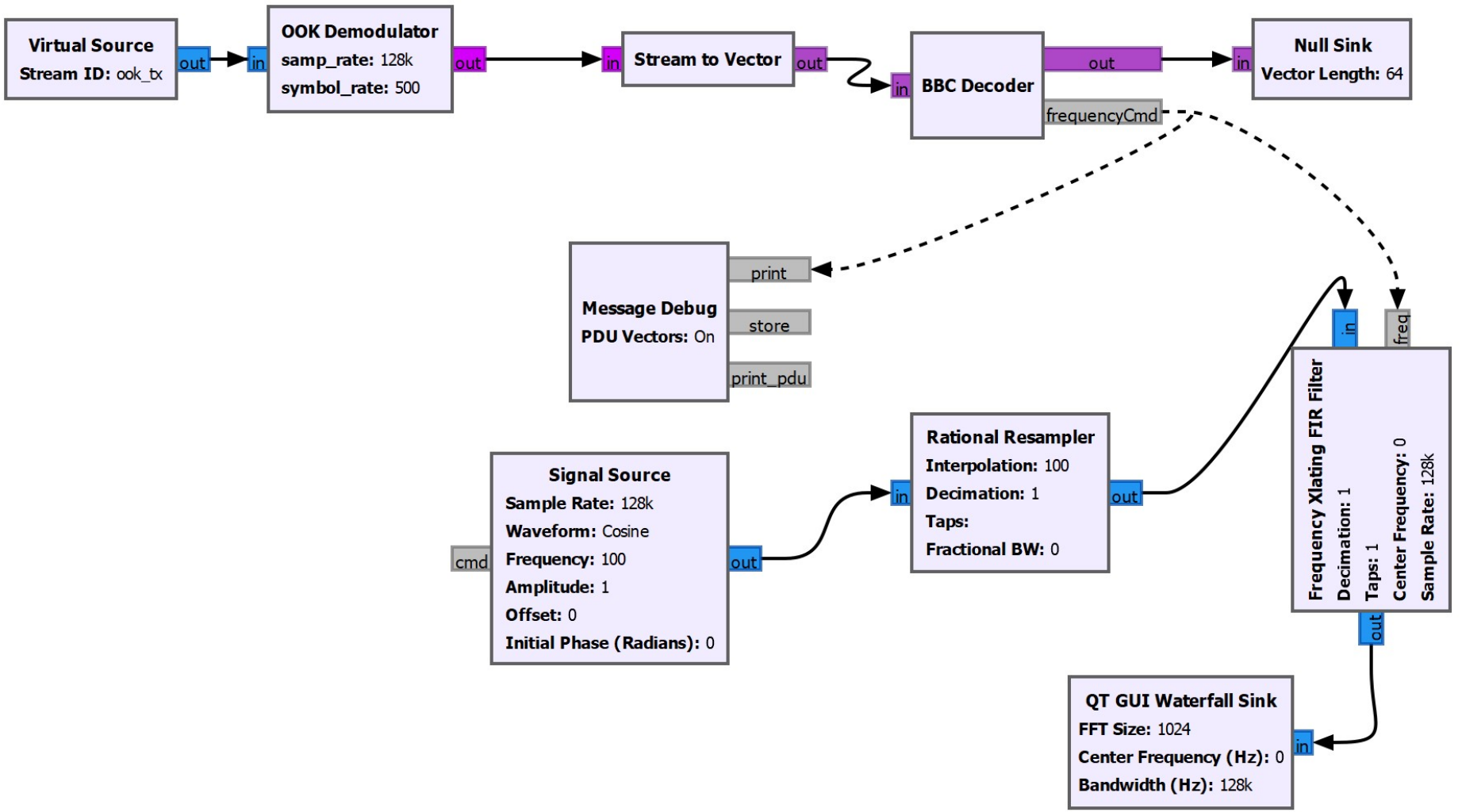


OOK Implementation

- OOK Hier blocks
- Sends codeword as bytes over message port as PMT.intern()
- BBC Decoder modified to receive bytes/re-assemble codewords
- Outputs messages



USE CASE - FHSS



Lessons Learned

UNITED STATES AIR FORCE ACADEMY

- out_sig can be a challenge
- Dynamic sizing of in/out vectors
- Sometimes, you just need a PMT
- Block development documentation difficult to follow (but is getting better!)
- Future goal is to output bytes to out_sig

Summary & What's Next?

UNITED STATES AIR FORCE ACADEMY

- Fix bytes output
- Package for CGAN
- Performance benchmark testing
- Hardware testing/implementation
- Active statistical thresholding
- Multimark BBC
- Codeword detection using ring buffer