Design and Implementation of an LDPC-based FEC encoder/decoder suitable for Storage devices

15. March 2018

Ruhr University of Bochum

Chair for Embedded Systems for Information Technology

Henry Bathke

Outline

Outline

Motivation

- Build a Hardware encoder and decoder suitable for storage devices
- Create a system for testing the encoder and decoder

Outline

Low Density Parity Check (LDPC) Codes

- Multiple parity check equations
- Each bit is contained in multiple equations
- With these equations we can correct errors

Low Density Parity Check (LDPC) Codes

- Multiple parity check equations
- Each bit is contained in multiple equations
- With these equations we can correct errors
- We construct the check matrix with the check equations

$$x_1 \oplus x_3 \oplus x_4 \oplus x_7 = 0$$

$$\boldsymbol{H} = \begin{bmatrix} 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \end{bmatrix}$$

Quasi Cyclic Low Density Parity Check (LDPC)

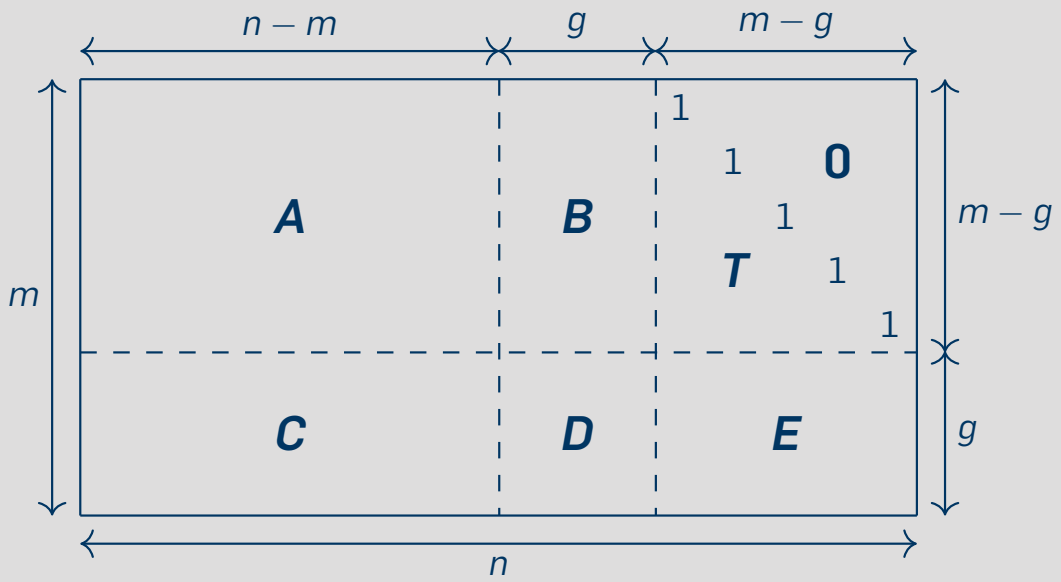
- High complexity of LDPC codes
- Reduce complexity by adding structure to the PCM
- Split PCM into submatrices
- Only allow shifted version of a submatrix

$$\boldsymbol{B} = \begin{bmatrix} -1 & 0 & 2 & -1 \\ 0 & 1 & -1 & -1 \\ -1 & 1 & 0 & 2 \end{bmatrix}$$

Encoding

- Is usually done with generator matrix
- The generator matrix is dense due to the inversion
- With long codes the dense matrix multiplication is large
- Use transforms on the PCM to convert it into a more desireable form[**QiGo07**]

Encoding



- Reach minimum gap g by doing only row and column permutations
- Only need an inverted matrix of size $g \times g$

Outline

Encoding

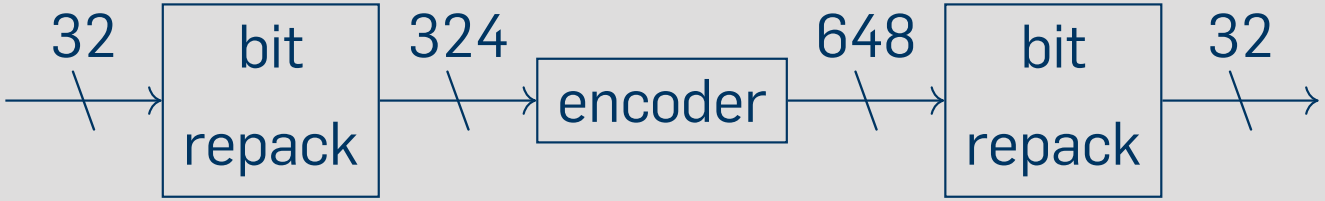
- Only large sparse matrix multiplication and back substitution
- One small dense matrix multiplication

Operation	Type
$\boldsymbol{A}s^T$	sparse multiplication
$\boldsymbol{T}^{-1}\boldsymbol{A}s^T$	sparse back substitution
$-\boldsymbol{E}\boldsymbol{T}^{-1}\boldsymbol{A}s^T$	sparse multiplication
$\boldsymbol{C}s^T$	sparse multiplication
$\left(-\boldsymbol{E}\boldsymbol{T}^{-1}\boldsymbol{A}s^T\right) + \left(\boldsymbol{C}s^T\right)$	vector addition
$\phi^{-1}\left(-\boldsymbol{E}\boldsymbol{T}^{-1}\boldsymbol{A}s^T + \boldsymbol{C}s^T\right)$	dense $g \times g$ multiplication

Encoding Hardware

- Implemented as combinatorial logic
- Connects to the other modules with an axi stream bus
- Repacking is needed as bit counts dont evenly divide
- Encoder is generated from the QC PCM with Python scripts

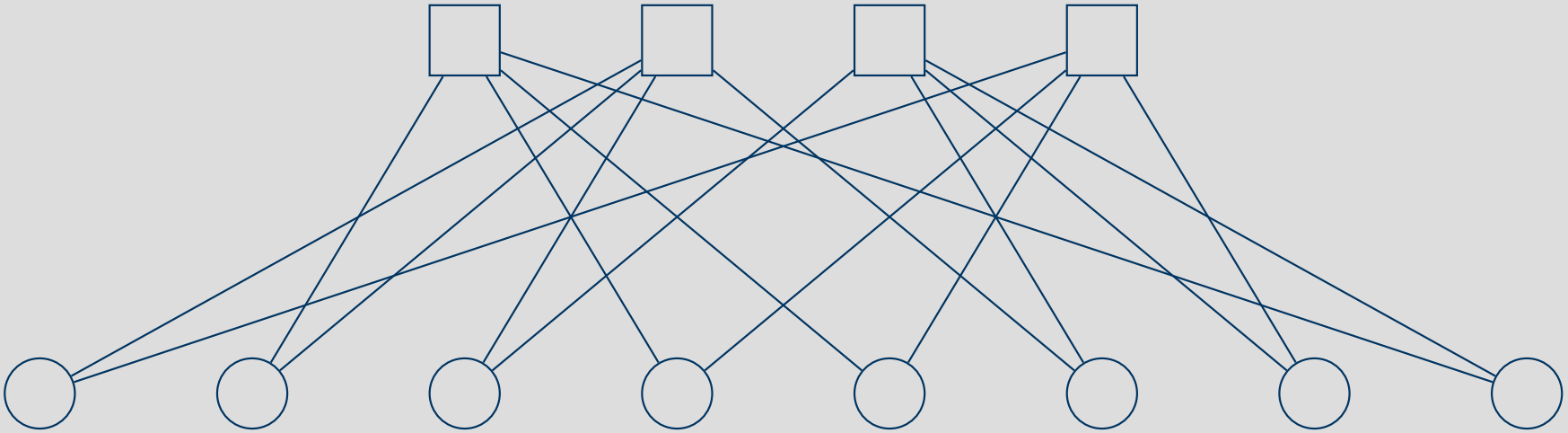
For wifi ldpc with rate 0.5 it looks like this



Outline

Decoding

- I implemented a message passing decoder
- Messages are passed along the edges on the tanner graph[**Ta81**]
- Computations are done on the nodes



Decoding

$$r_{mn} = \left(\prod_{n' \in M(m) \setminus n} \operatorname{sign}(q_{n'm}) \right) \min_{n' \in M(m) \setminus n} (|q_{n'm}|)$$

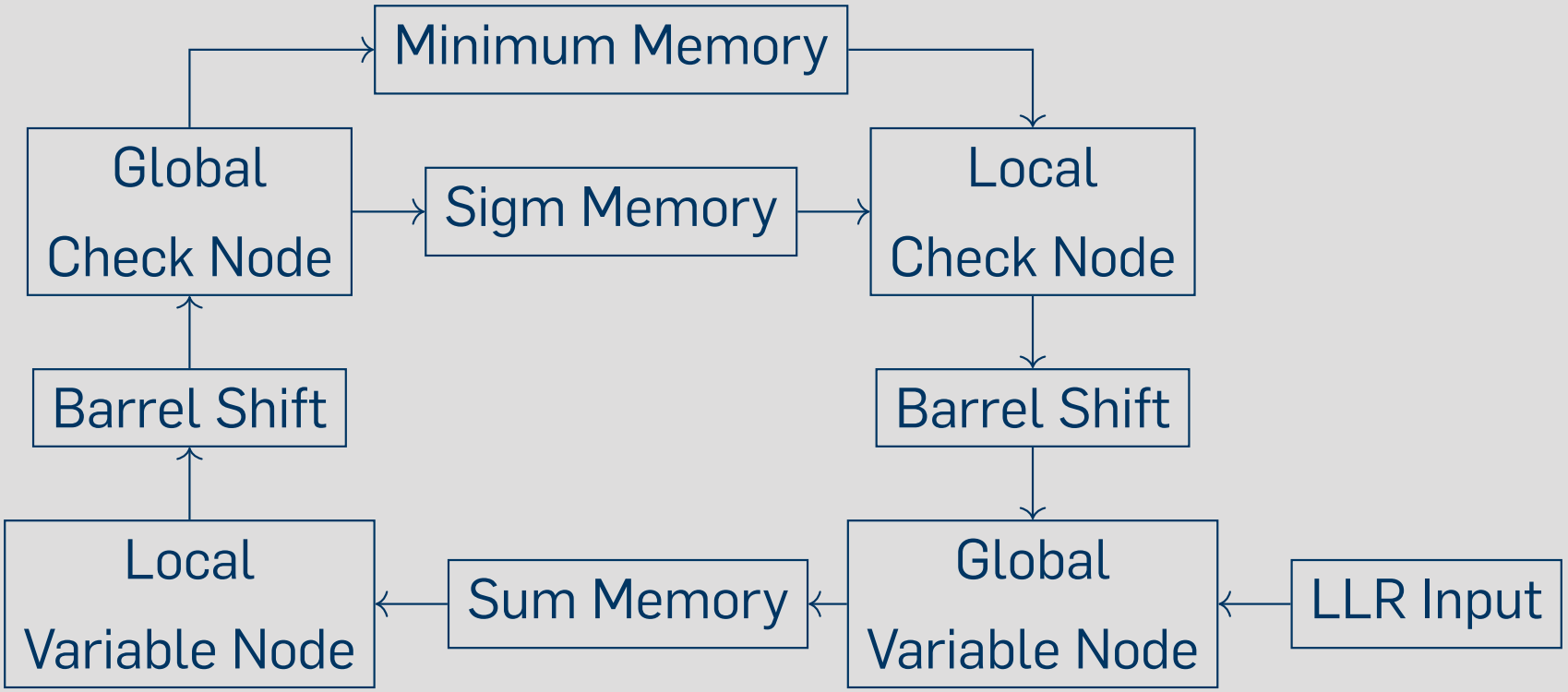
$$q_{nm} = y_n + \sum_{m' \in N(n) \setminus m} r_{m'n}$$

$$L_n = y_n + \sum_{m \in N(n)} r_{m'n}$$

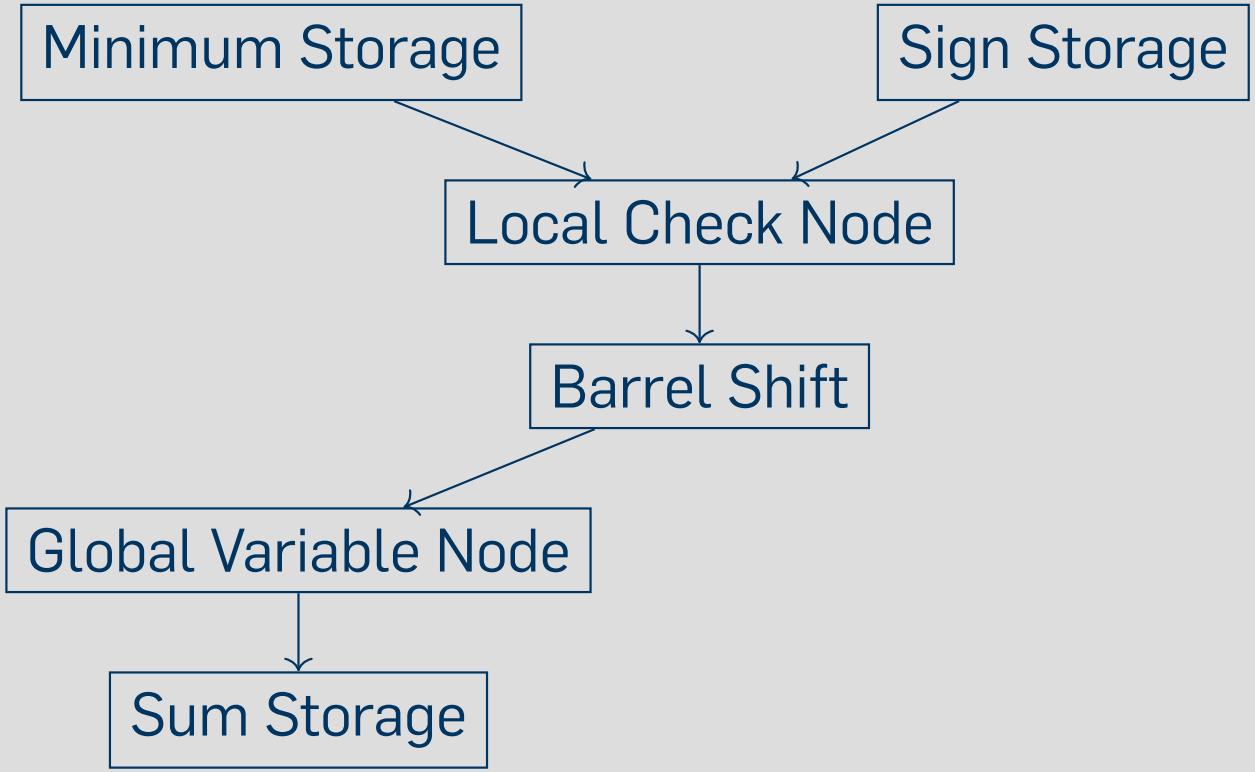
- We have to implement these equations
- Can exploit similarities

Hardware Decoding

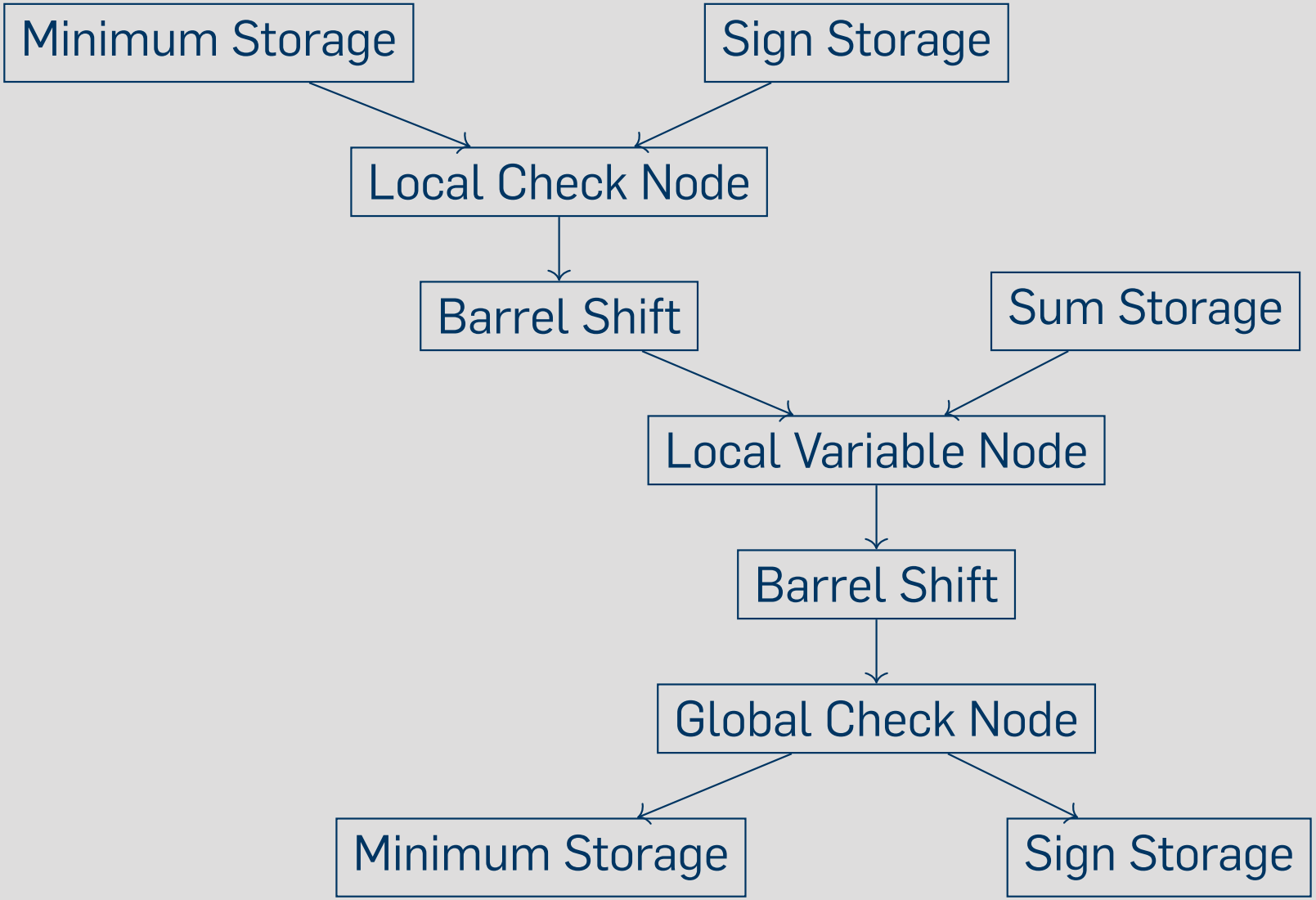
- two pass algorithm
- intermediate values stored im memory



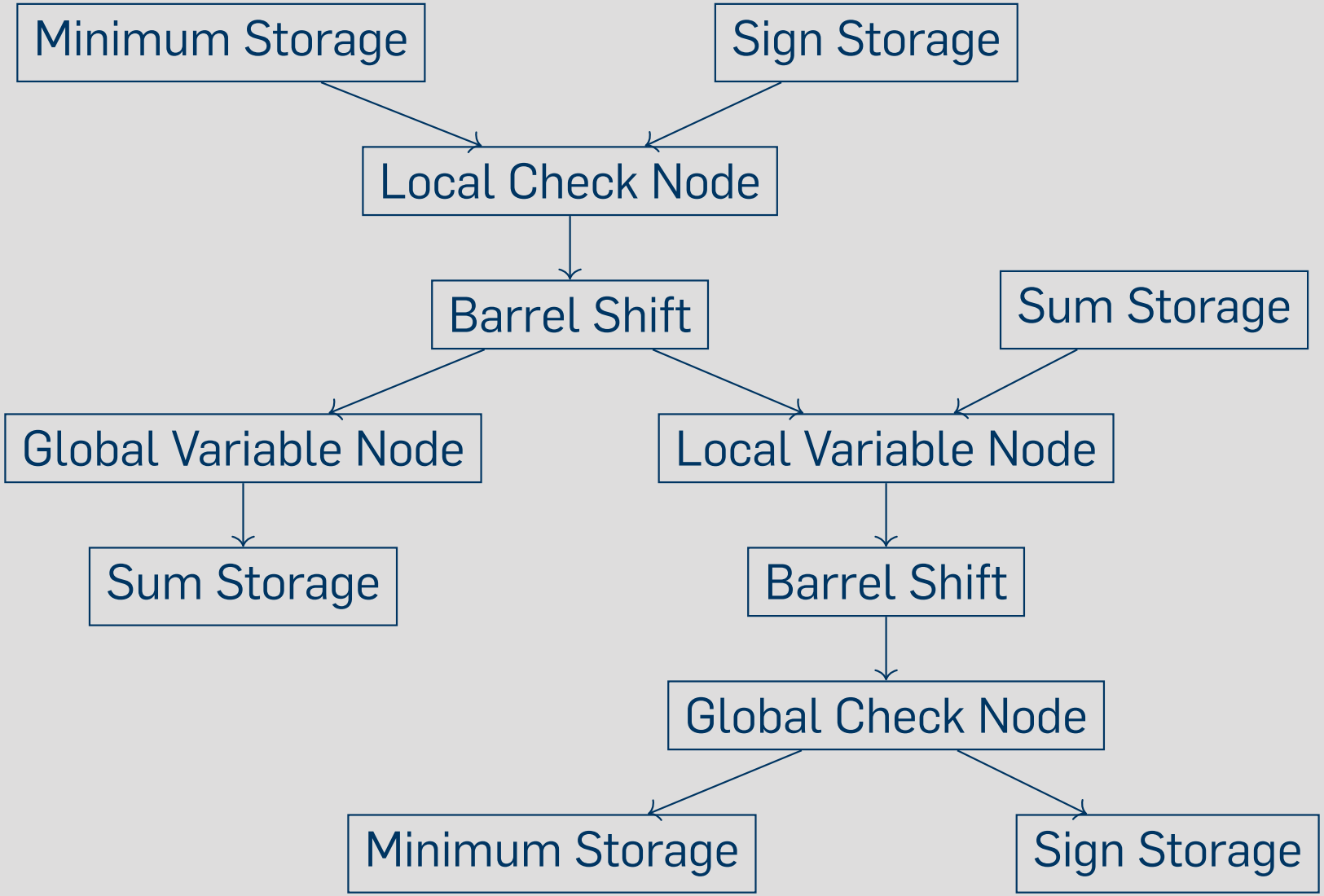
Hardware Decoding



Hardware Decoding



Hardware Decoding



Hardware Decoding

Running minimum

0	0	0	1	0	0	0	0	1	0	0	0	∞
0	0	0	0	1	0	1	0	0	0	0	0	∞
0	0	0	0	0	1	0	1	0	0	0	0	∞
1	0	0	0	1	0	0	0	0	0	0	0	∞
0	1	0	0	0	1	0	0	0	0	0	0	∞
0	0	1	1	0	0	0	0	0	0	0	0	∞
0	0	0	0	1	0	1	0	0	0	0	1	∞
0	0	0	0	0	1	0	1	0	1	0	0	∞
0	0	0	1	0	0	0	0	1	0	1	0	∞
4	1	6	3	8	7	5	9	2	4	2	8	

Hardware Decoding

												Running minimum
0	0	0	1	0	0	0	0	1	0	0	0	3
0	0	0	0	1	0	1	0	0	0	0	0	8
0	0	0	0	0	1	0	1	0	0	0	0	7
1	0	0	0	1	0	0	0	0	0	0	0	∞
0	1	0	0	0	1	0	0	0	0	0	0	∞
0	0	1	1	0	0	0	0	0	0	0	0	∞
0	0	0	0	1	0	1	0	0	0	0	1	∞
0	0	0	0	0	1	0	1	0	1	0	0	∞
0	0	0	1	0	0	0	0	1	0	1	0	∞
4	1	6	3	8	7	5	9	2	4	2	8	

Hardware Decoding

												Running minimum
0	0	0	1	0	0	0	0	1	0	0	0	2
0	0	0	0	1	0	1	0	0	0	0	0	5
0	0	0	0	0	1	0	1	0	0	0	0	7
1	0	0	0	1	0	0	0	0	0	0	0	∞
0	1	0	0	0	1	0	0	0	0	0	0	∞
0	0	1	1	0	0	0	0	0	0	0	0	∞
0	0	0	0	1	0	1	0	0	0	0	1	∞
0	0	0	0	0	1	0	1	0	1	0	0	∞
0	0	0	1	0	0	0	0	1	0	1	0	∞
4	1	6	3	8	7	5	9	2	4	2	8	

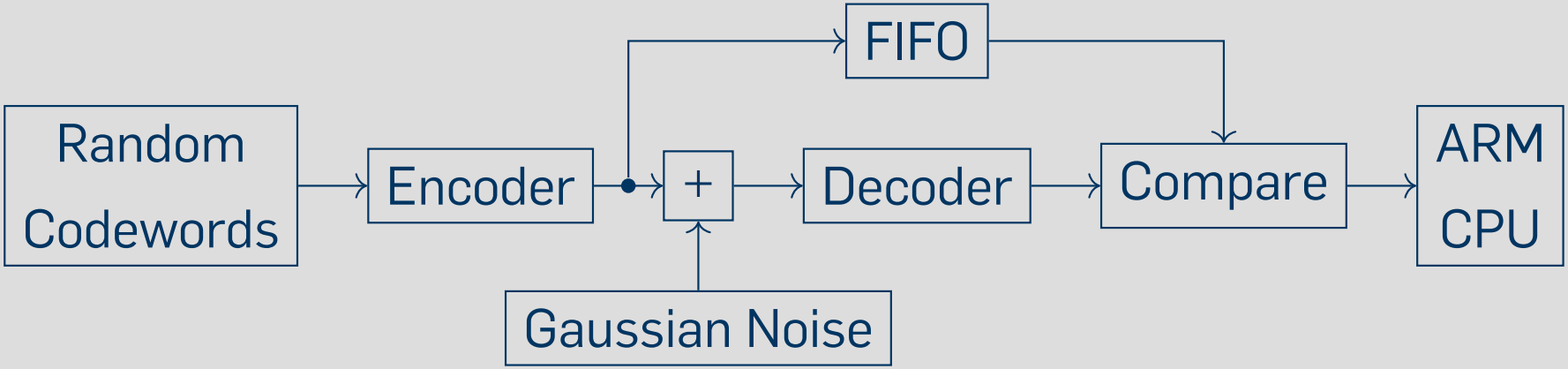
Hardware Decoding

												Running minimum
0	0	0	1	0	0	0	0	1	0	0	0	2
0	0	0	0	1	0	1	0	0	0	0	0	5
0	0	0	0	0	1	0	1	0	0	0	0	7
1	0	0	0	1	0	0	0	0	0	0	0	∞
0	1	0	0	0	1	0	0	0	0	0	0	∞
0	0	1	1	0	0	0	0	0	0	0	0	∞
0	0	0	0	1	0	1	0	0	0	0	1	∞
0	0	0	0	0	1	0	1	0	1	0	0	∞
0	0	0	1	0	0	0	0	1	0	1	0	∞
4	1	6	3	8	7	5	9	2	4	2	8	

Outline

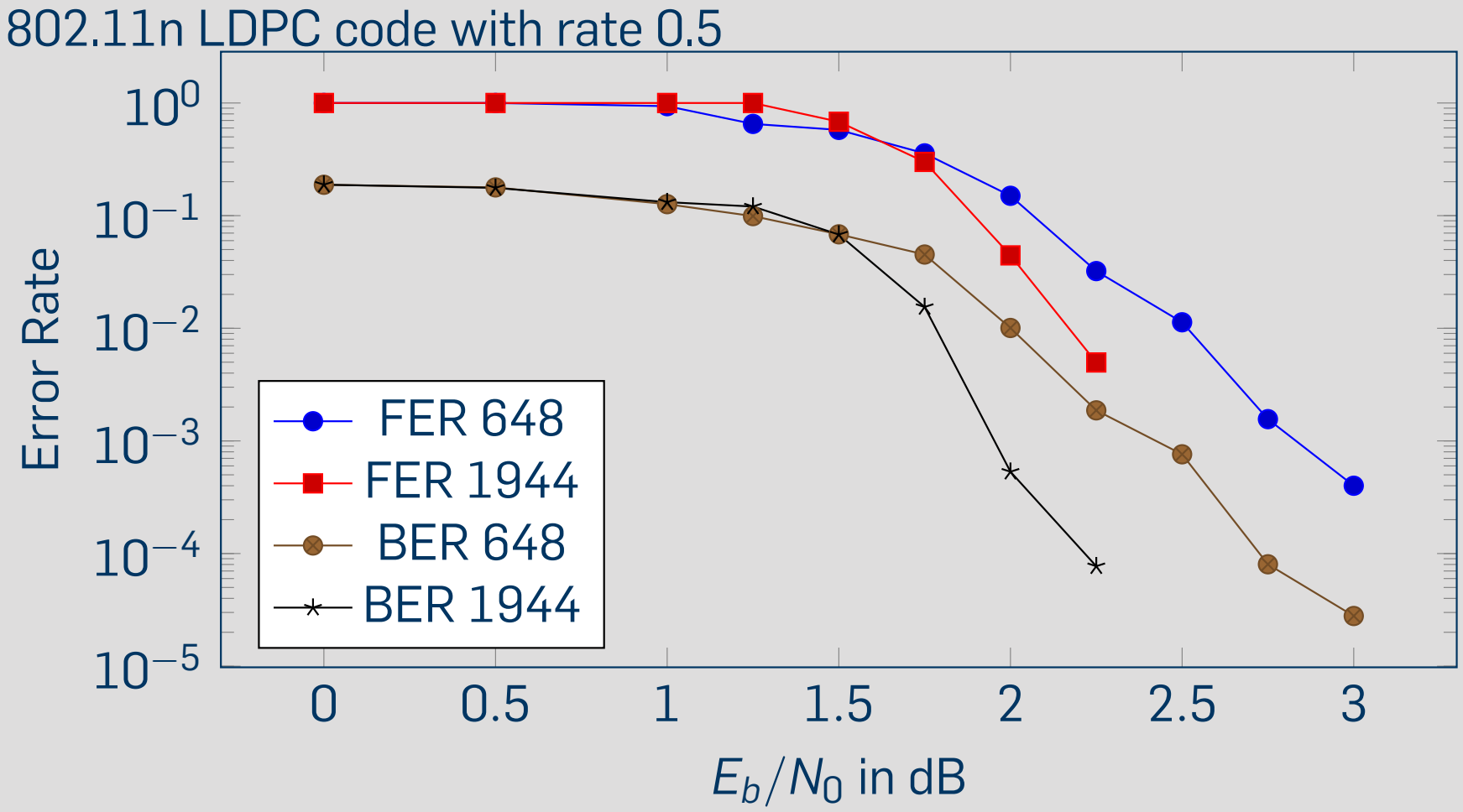
Performance Testing

- Used a ZedBoard
- Testing requires an encoder, channel, decoder setup
- AWGN channel
- Whole performance test is implemented on Hardware
- ARM cpu calculates test parameters and reads the error counts

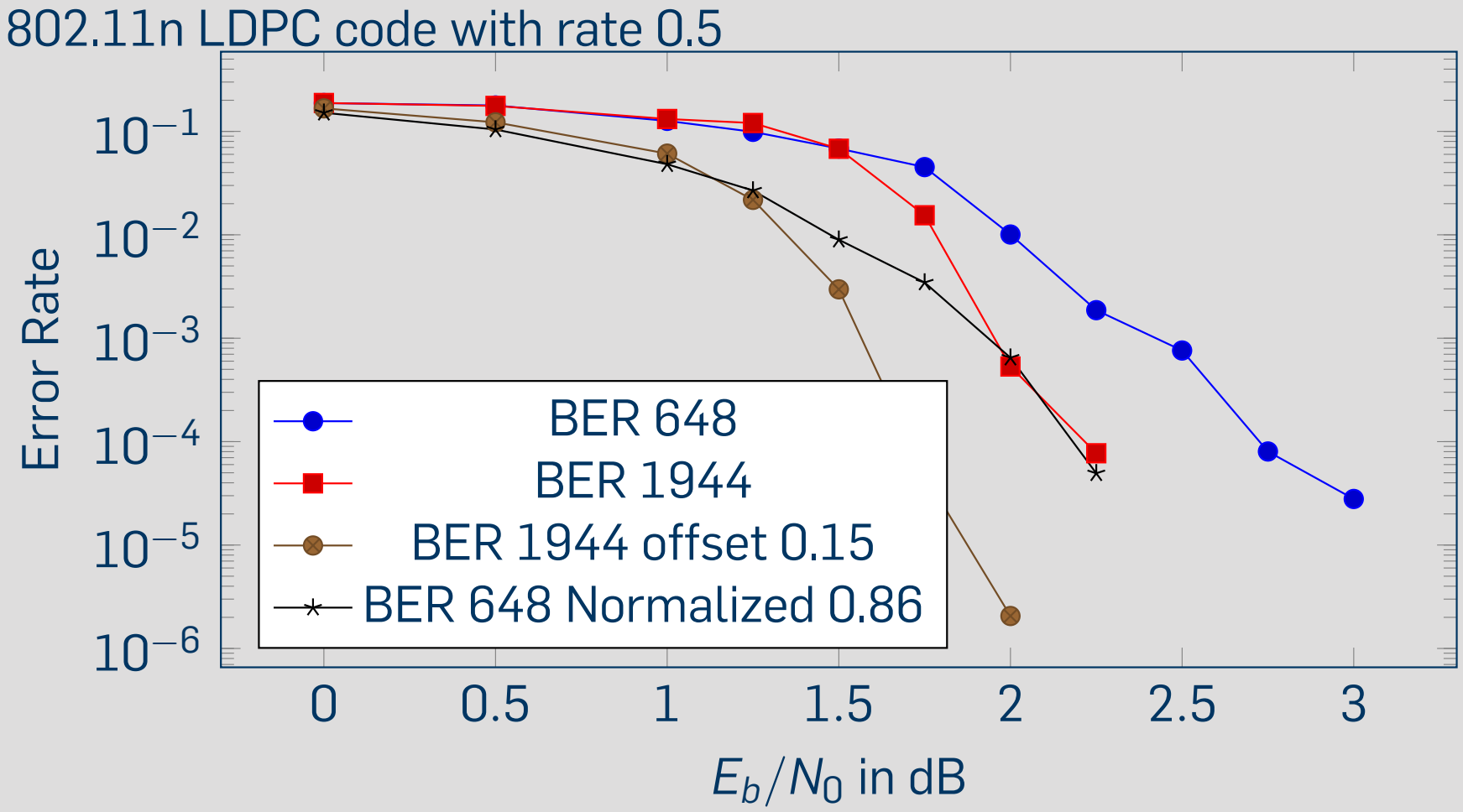


Outline

Simulation Results

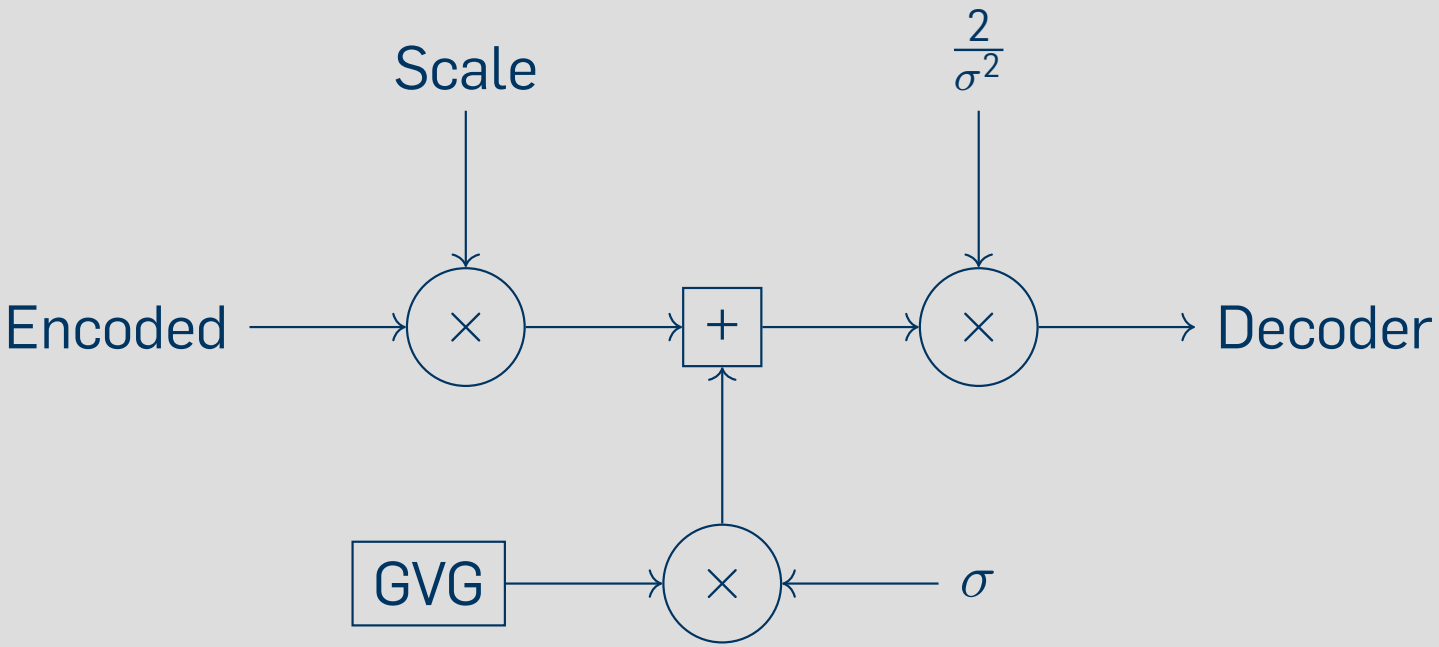


Simulation Results



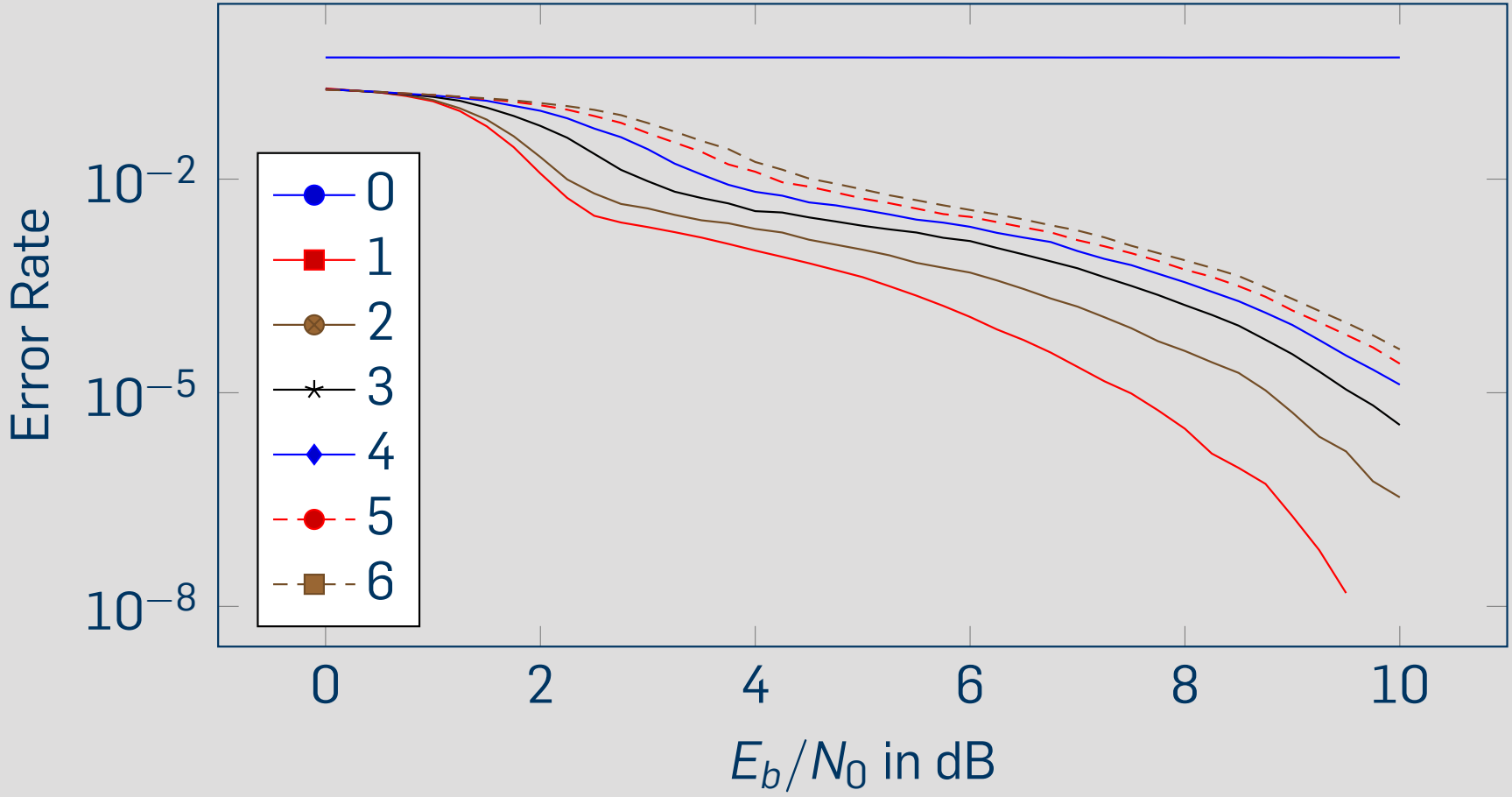
Hardware Results

- Decoder expects LLR
- Convert channel output to LLR



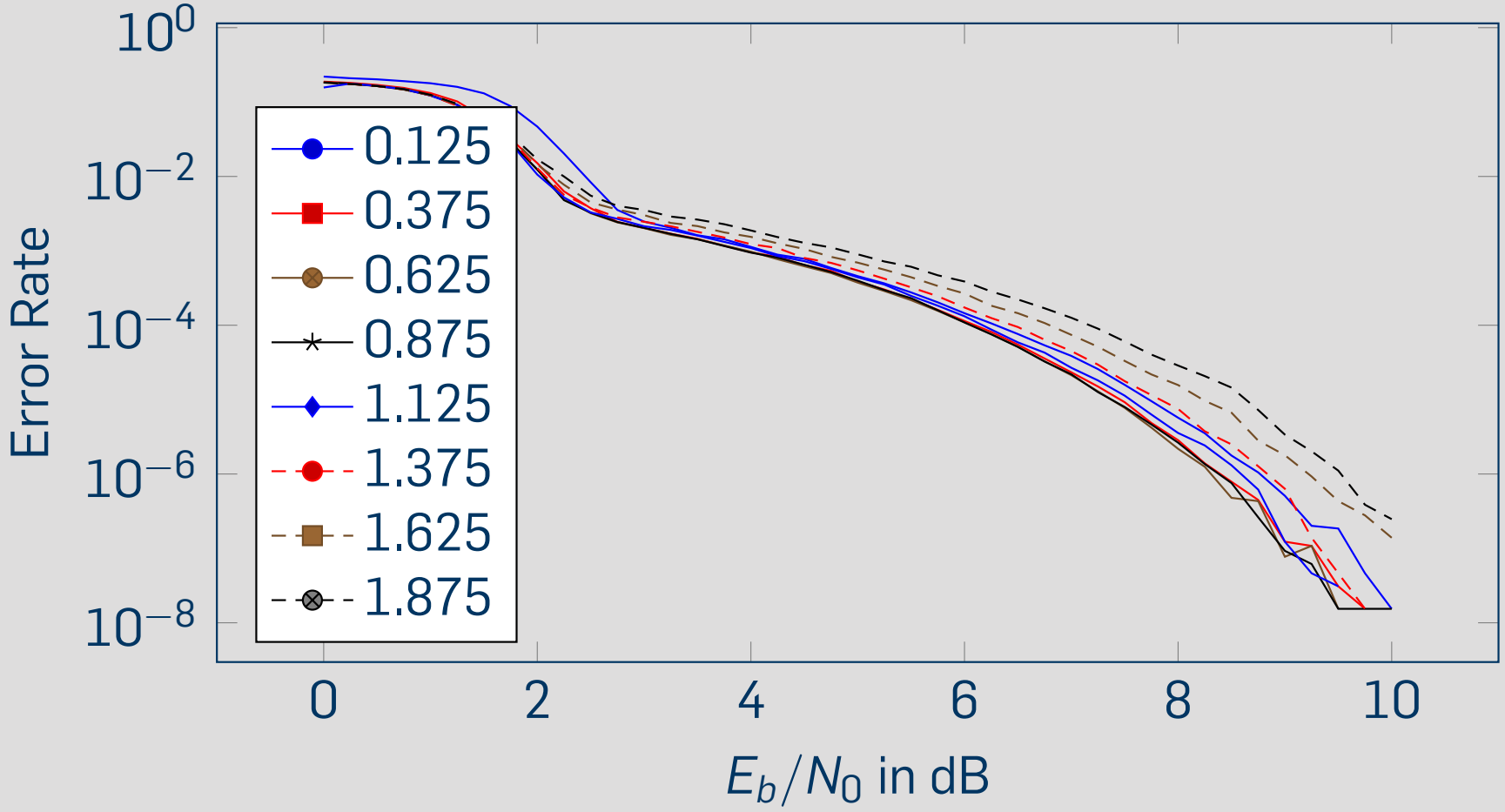
Hardware Results

Optimize the input scaling



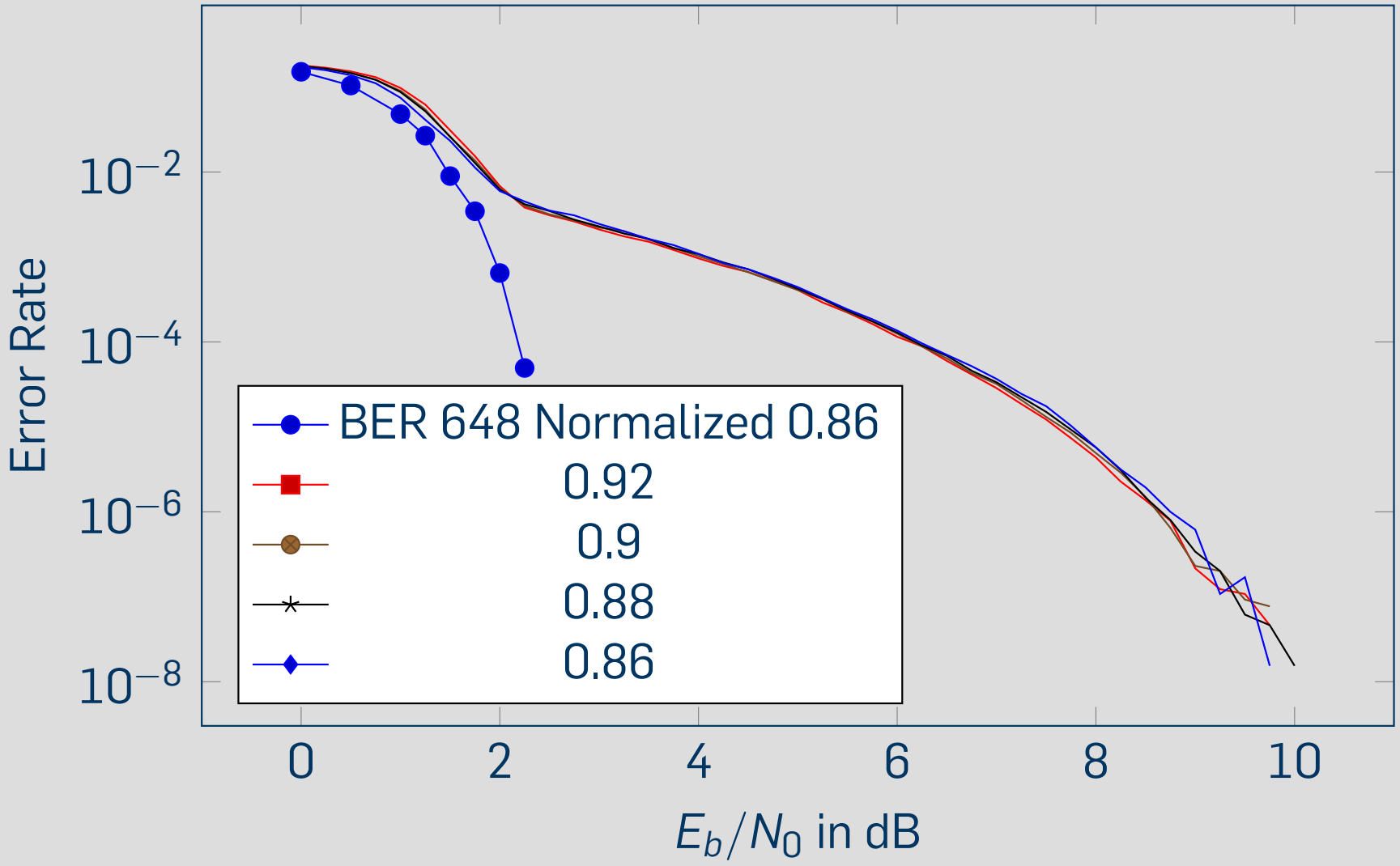
Hardware Results

I chose a scaling of 0.75



Hardware Results

Normalized Min Sum



Outline

Conclusion and Outlook

Conclusion

- LDPC codes can be implemented on hardware
-

Outlook

- Look at power consumption
- Optimize area utilization

Questions?

Questions?

References I