

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



- 1 Motivation
- 2 LDPC Codes
- 3 Encoding
- 4 Decoding
- 5 Performance
- 6 Results
- 7 Conclusion

- 1 Motivation
- 2 LDPC Codes
- 3 Encoding
- 4 Decoding
- 5 Performance
- 6 Results
- 7 Conclusion

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

- 1 Motivation
- 2 LDPC Codes**
- 3 Encoding
- 4 Decoding
- 5 Performance
- 6 Results
- 7 Conclusion

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

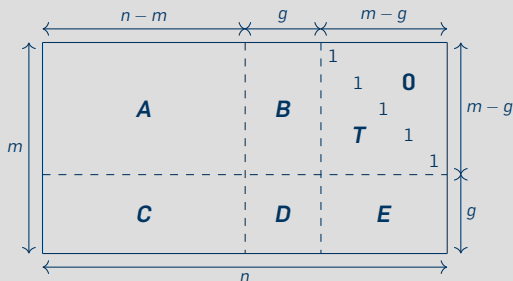
$$x_1 \oplus x_3 \oplus x_4 \oplus x_7 = 0$$

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

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

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

- 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 desirable form[QiGo07]



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

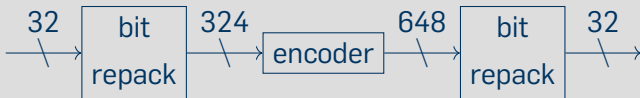
- 1 Motivation
- 2 LDPC Codes
- 3 Encoding**
- 4 Decoding
- 5 Performance
- 6 Results
- 7 Conclusion

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

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

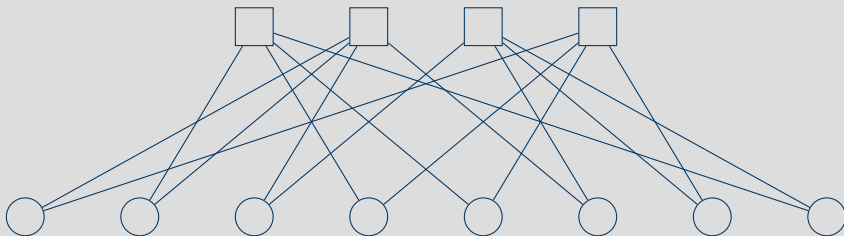
- Implemented as combinatorial logic
- Connects to the other modules with an axi stream bus
- Repacking is needed as bit counts dont evenly divide
- Encoder is generated from the QC PCM with Python scripts

For wifi ldpc with rate 0.5 it looks like this



- 1 Motivation
- 2 LDPC Codes
- 3 Encoding
- 4 Decoding**
- 5 Performance
- 6 Results
- 7 Conclusion

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



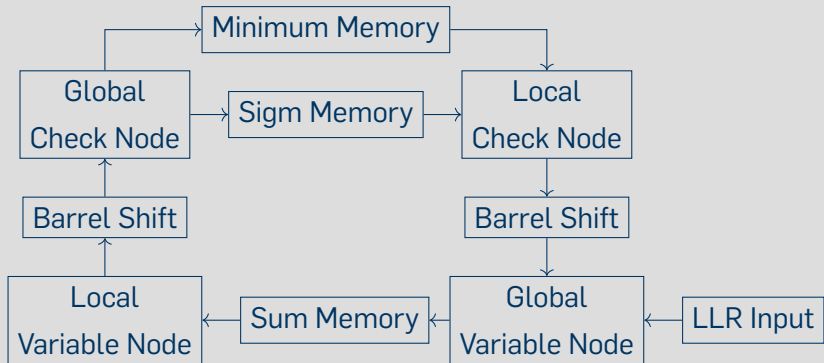
$$r_{mn} = \left(\prod_{n' \in M(m) \setminus n} \text{sign}(q_{n'm}) \right) \min_{n' \in M(m) \setminus n} (|q_{n'm}|)$$

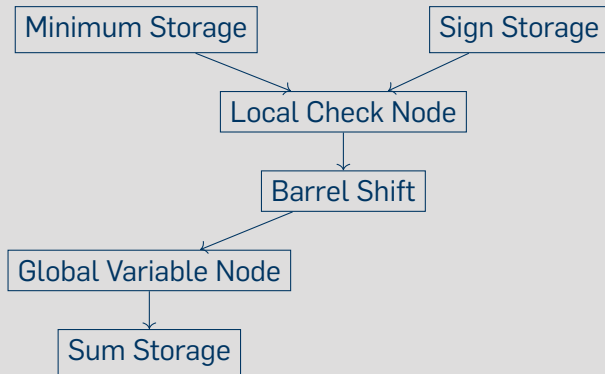
$$q_{nm} = y_n + \sum_{m' \in N(n) \setminus m} r_{m'n}$$

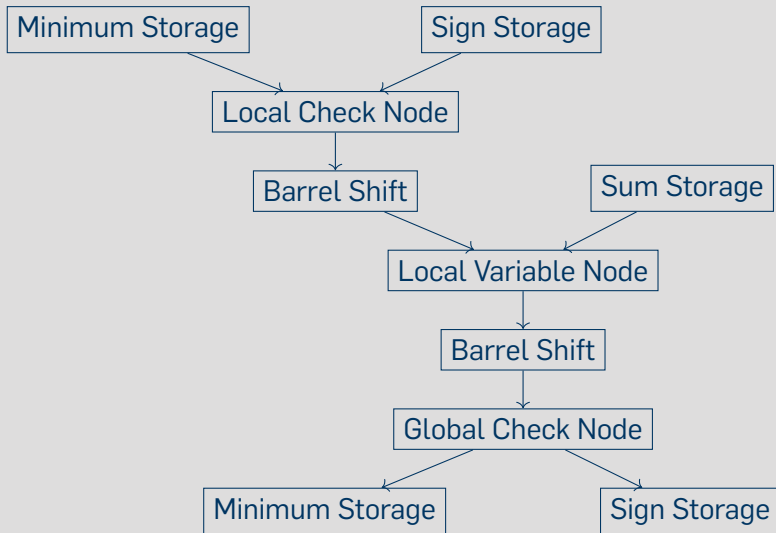
$$L_n = y_n + \sum_{m \in N(n)} r_{m'n}$$

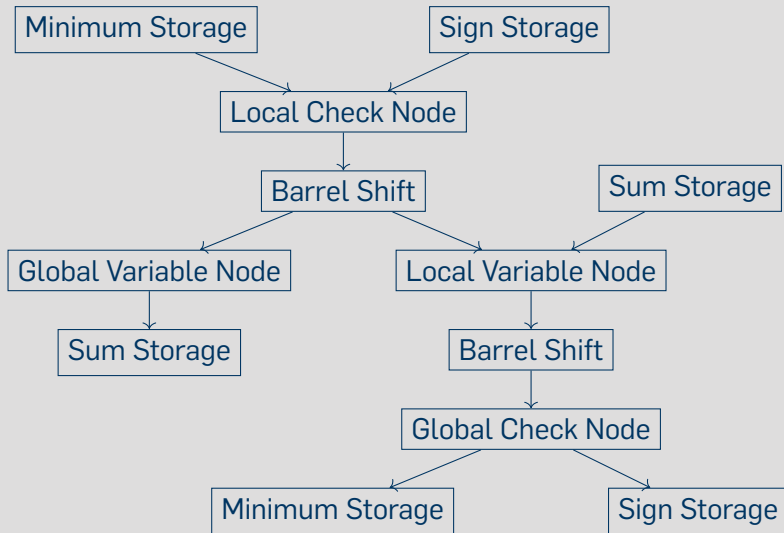
- We have to implement these equations
- Can exploit similarities

- two pass algorithm
- intermediate values stored in memory





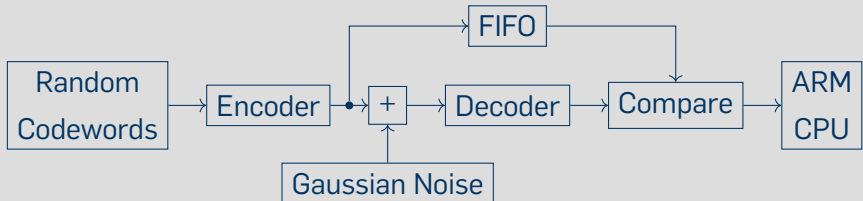




- 1 Motivation
- 2 LDPC Codes
- 3 Encoding
- 4 Decoding
- 5 Performance**
- 6 Results
- 7 Conclusion

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



- 1 Motivation
- 2 LDPC Codes
- 3 Encoding
- 4 Decoding
- 5 Performance
- 6 Results**
- 7 Conclusion

Simulation Results

- 1 Motivation
- 2 LDPC Codes
- 3 Encoding
- 4 Decoding
- 5 Performance
- 6 Results
- 7 Conclusion**

- Do a conclusion and maybe a part on future works?

Questions?

