Group 3:
LDPC Codes in Computer Memory

Tom Wang
Natalie Balashov

April 11th, 2024

Overview

1. Applications of Codes in Memory
2. Introduction to LDPC Codes
3. Implementation of LDPC Codes in HDL
4. Implementation of Hamming Codes in HDL
5. Comparison of LDPC and Hamming Codes

Application Codes in Computer Memory

- Error-correcting codes in memory systems are crucial to maintaining data integrity.
- With increasing memory densities and memory chip sizes, bit errors occur more frequently.
- Special technologies like spin-torque transfer random access memory (STT RAM) experience asymmetric bit errors.
- Niche applications like space missions require more robust error-correction strategies due to overly noisy channels created by radiation.
- In addition to performance and error-correcting capacity, hardware resources are a design constraint.

Low-Density Parity Check (LDPC) Codes

- LDPC codes are linear block codes characterized by sparse parity check matrices.
- LDPC codes have been shown to achieve near Shannon capacity performance.
- First proposed by Gallager in 1960, as part of his Ph.D. thesis.
- Two main types: regular and irregular.
- A LDPC code with parameters (n, j, i) has a block length n, j ones in each column of H and i ones in each row of H.

LDPC Parity Check Matrix

We examined the LDPC code with parameters $n = 72, j = 2$ and $i = 18$.

This particular matrix was generated with the Python library `pyldpc`, which uses the typical algorithm.

Figure: 8×72 Parity check matrix for LDPC code.

LDPC Generator Matrix

To generate the generator matrix, we used the following algorithm:

1. Re-organize the codeword $\mathbf{c}$ of length n so that it is of the form $\mathbf{c} = [\mathbf{b}, \mathbf{m}]$, where $\mathbf{b}$ denotes the parity bits vector and $\mathbf{m}$ denotes the message bits vector.
2. Sub-divide the parity matrix H into two sub-matrices such that $H = [H_1, H_2]^T$ where H_1 is a square matrix of size $n - k$.
3. Since $\mathbf{b}$ is of length $n - k$ as well, we can break up the matrix multiplication into $[\mathbf{b}, \mathbf{m}][H_1, H_2]^T = \mathbf{b}H_1^T + \mathbf{m}H_2^T = 0$.
4. Since we are dealing with a code defined on a binary field, $\mathbf{b}H_1^T = \mathbf{m}H_2^T$ implies that $\mathbf{b} = \mathbf{m}H_2^T(H_1^T)^{-1}$.
5. With $A = H_2^T(H_1^T)^{-1}$, the generator matrix G is $[A, I_{n-k}]$.

LDPC Decoding Algorithm

Algorithm 1 Bit-Flipping Decoding Algorithm

```
while parity check is not satisfied and maximum iteration
is not reached do
  for each check node do
    if the check node is not satisfied then
      vote for flipping the bit
    else
      vote for not flipping the bit
    end if
  end for
  each bit vote for not flipping the bit
  flip the bit with the majority vote
end while
if parity check is satisfied then
  return the decoded message
else
  return failure
end if
```

LDPC Performance

On DE1-SOC, Quartus analysis shows that the highest possible clock frequency is **325.73 MHz**.

- Combinational ALUT usage for logic: 114
  - 7 input functions: 0
  - 6 input functions: 4
  - 5 input functions: 10
  - 4 input functions: 28
  - $\leq$ 3 input functions: 72
- Dedicated logic registers: 72

Hamming Implementation

- Typical Hamming code implementation uses 64-bit to 72-bit encoding schemes.
- This typical implementation is called Single Error Correction, Double Error Detection (SECDED) Hamming code, where the parity bits indicate the index of the bit error.

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

Figure: Parity bits and data bits in a Hamming code.

FPGA Implementation

Logic can be easily achieved with a combination of AND gates and XOR gates.
According to Quartus timing analysis, the highest possible clock frequency is **387.3 MHz**.

- Combinational ALUT usage for logic: 154
  - 7 input functions: 0
  - 6 input functions: 121
  - 5 input functions: 10
  - 4 input functions: 13
  - $\leq$ 3 input functions: 10
- Dedicated logic registers: 69

Comparison

Table: Comparison of Decoding Algorithms

Parameter	**Hamming**	**LDPC**
Highest Frequency (MHz)	387.3	325.73
ALUT Usage for Logic	154	114
7 input functions	0	0
6 input functions	121	4
5 input functions	10	10
4 input functions	13	28
$\leq$ 3 input functions	10	72
Dedicated Logic Registers	69	72

Conclusion

- Hamming code can operate at a higher clock frequency.
- LDPC code has a lower ALUT usage and more dedicated logic registers.
- LDPC code uses significantly fewer 6-input function blocks.
- This means that the LDPC code is more efficient in terms of hardware usage.
- The frequency difference is not significant. On ASIC designs, the performance comparison might be different.

Appendix

1. Hamming Code Excel Demo
2. Hamming Code 64-in RTL
3. Hamming Code 72-out RTL
4. LDPC Decoding RTL
5. LDPC Parity Check Matrix