www.ijecs.in
International Journal Of Engineering And Computer Science ISSN:2319-7242
Volume 7 Issue 3 March 2018, Page No. 23781-23784
Index Copernicus Value (2015): 58.10, 76.25 (2016) DOI: 10.18535/ijecs/v7i3.21

Design of Quasi-Cyclic Low Density Parity Check Decoder Using Optimized Min-Sum Algorithm

Rajarshini Mishra
GGITS, Jabalpur

Abstract
Low-density parity-check (LDPC) have been shown to have good error correcting performance approaching Shannon's limit. Good error correcting performance enables efficient and reliable communication. However, a LDPC code decoding algorithm needs to be executed efficiently to meet cost , time, power and bandwidth requirements of target applications.
Quasi-cyclic low-density parity-check (QC-LDPC) codes are an important subclass of LDPC codes that are known as one of the most effective error controlling methods. Quasi cyclic codes are known to possess some degree of regularity. Many important communication standards such as DVB-S2 and 802.16e use these codes.
The proposed Optimized Min-Sum decoding algorithm performs very close to the Sum-Product decoding while preserving the main features of the Min-Sum decoding, that is low complexity and independence with respect to noise variance estimation errors.Proposed decoder is well matched for VLSI implementation and will be implemented on Xilinx FPGA family.

Introduction

Hamming Code

Hamming codes are a family of linear error-correcting codes which were invented by Richard Hamming . These codes can detect up to two-bit errors or correct one-bit errors without detection of uncorrected errors. Hamming codes are perfect codes, that is, they achieve the highest possible rate for codes with their block length and minimum distance of three. **The key to the Hamming Code is the use of extra parity bits to allow the identification of a single error.**

The number of parity bits required depends on the number of bits in the data transmission, and is calculated by the Hamming rule:

$$n + k + 1 <= 2^k \quad (1)$$

Where *n* is the number of data bits and *k* is the number of parity bits. The total of the two is called the Hamming code word, which is generated by multiplying the data bits by a generator matrix. In the Hamming code, k parity bits are added to an n-bit data word, forming a new word of n + k bits. The bit positions are numbered in sequence from 1 to n+k. Those positions numbered with powers of two are reserved for the parity bits. The remaining bits are the data bits. The code can be used with words of any length.

Consider, for example, the 8-bit data word 11000100. We include four parity bits with this word and arrange the 12 bits as follows:

Bit position	1	2	3	4	5	6	7	8	9	10	11	12
	P1	P2	1	P4	1	0	0	P8	0	1	0	0

The 4 parity bits P1 through P8 are in positions 1, 2, 4, and 8, respectively. The 8 bits of the data word are in the remaining positions. Each parity bit is calculated as follows:

P1 XOR of bits (3, 5, 7, 9, 11)

P2 XOR of bits (3, 6, 7, 10, 11)

P4 XOR of bits (5, 6, 7, 12)

P8 XOR of bits (9, 10, 11, 12)

each parity bit is set so that the total number of 1's in the checked positions, including the parity bit, is always even. The 8-bit data word is written into the memory together with the 4 parity bits as a 12-bit composite word. Substituting the 4 parity bits in their proper positions, we obtain the 12-bit composite word written into memory:

Bit position 1 2 3 4 5 6 7 8 9 10 11 12

0 0 1 1 1 0 0 1 0 1 0 0

When the 12 bits are read from memory, they are checked again for errors. The parity of the word is checked over the same groups of bits, including their parity bits. The four check bits are evaluated as follows:

C1 = XOR of bits (1, 3, 5, 7, 9, 11)

C2 = XOR of bits (2, 3, 6, 7, 10,11)

C4= XOR of bits (4, 5, 6, 7, 12)

C8= XOR of bits (8, 9, 10, 11, 12)

A 0 check bit designates an even parity over the checked bits, and a 1 designates an odd parity. Since the bits were written with even parity, the result, C C8C4C2C1 0000, indicates that no error has occurred. However, if , the 4-bit binary number formed by the check bits gives the position of the erroneous bit if only a single bit is in error.

Ldpc Code

In information theory, a low-density parity-check (LDPC) code is a linear error correcting code, a method of transmitting a message over a noisy transmission channel. An LDPC is constructed using a sparse bipartite graph. LDPC codes are capacity-approaching codes, which means that practical constructions exist that allow the noise threshold to be set very close (or even *arbitrarily* close on the binary erasure channel) to the theoretical maximum (the Shannon limit) for a symmetric memoryless channel.

LDPC codes are finding increasing use in applications requiring reliable and highly efficient information transfer over bandwidth or return channel-constrained links in the presence of corrupting noise. LDPC codes functionally are defined by a sparse parity-check matrix.

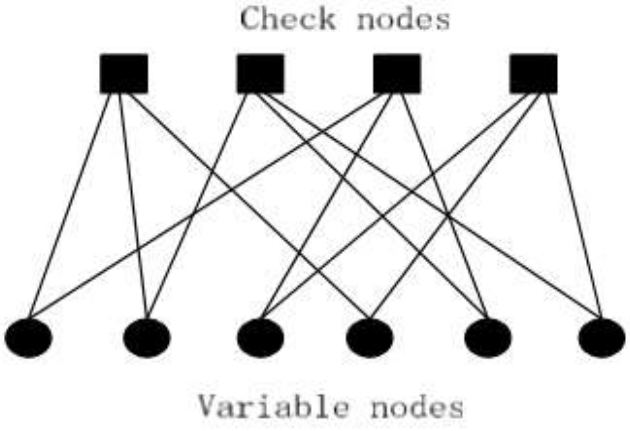

Figure.1 Tanner graph for a regular LDPC code

In this graph, n variable nodes at the bottom of the graph are connected to $(n-k)$ constraint nodes in the top of the graph. This is a popular way of graphically representing an (n, k) LDPC code. The bits of a valid message, when placed on the **T's** at the top of the graph, satisfy the graphical constraints.

Modulo-2 Operation

Modular arithmetic is an extremely flexible problem solving tool. In computing, the modulo operation finds the remainder after division of one number by another (sometimes called *modulus*).

Given two positive numbers, a (the dividend) and n (the divisor), a modulo n (abbreviated as a mod n) is the remainder of the Euclidean division of a by n. For example, the expression "5 mod 2" would evaluate to 1 because 5 divided by

2 leaves a quotient of 2 and a remainder of 1, while "9 mod 3" would evaluate to 0 because the division of 9 by 3 has a quotient of 3 and leaves a remainder of 0.

In nearly all computing systems, the quotient q and the remainder r of a divided by n satisfy

$q \in z$

$a = nq+r$

$|r|<|n|$

The range of numbers for an integer modulo of n is 0 to $n − 1$.

Many programming languages have a mod operator, typically represented with the % symbol.

Literature Survey

Hossein Gharaee et al [1] presented their work entitled "A High-Throughput FPGA Implementation of Quasi-Cyclic LDPC Decoder" In this paper, an FPGA implementation of a partial-parallel QC-LDPC decoder is proposed based on the sum-product algorithm. Here, a modified version of TPMP1 algorithm to improve the number of clock cycles, resource usage, and power consumption. The decoder is implemented for a code length of 672 with code rate of 3/4. This implementation is achieved to maximum throughput of 3.3 Gbps with frequency of 280 MHz and its power consumption is less than 150mW. To create an acceptable trade-off between parallelism level of check node units, variable node units, and maximum throughput, a scheduling with semi-parallel structure is proposed.

The VLSI implementation results show that the proposed decoder occupies an area of 3.4mm^2 and achieves maximum decoding throughput of 3360 Mbps with maximum 10 iterations. The estimated power consumption is 150 mW in frequency of 280 MHz. The results show that this decoder presents high Throughput with less power consumption and area by implementing sum-product algorithm in proposed time scheduling.

Swapnil Mhaske et al [2] presented their paper entitled "High-Throughput FPGA-based QC-LDPC Decoder Architecture", This paper presents a high-throughput FPGA-based architecture for a binary Quasi-Cyclic Low-Density Parity-Check (QC-LDPC) code based on min-sum algorithm. A novel representation of the parity-check matrix (PCM) providing a multi-fold throughput gain is proposed here. Splitting of the node processing algorithm enables to achieve pipelining of blocks. The main advantage is that outcoming message from every variable node can be computed by finding only two lowest values of reliability in the check. To validate the architecture, decoder is implemented on the Xilinx Kintex-7 FPGA with the help of the FPGA IP compiler. It offers an automated and systematic compilation flow where an optimized hardware implementation from the LDPC algorithm was generated, achieving an overall throughput of 608Mb/s (at 260MHz).

Problem Statement

Among a variety of decoding algorithms, the well-known Sum Product (SP) algorithm achieves a good decoding performance but requires a large hardware complexity. The decoding of LDPC codes based on the belief propagation algorithm, known as Sum-Product algorithm (SPA), needs complex number calculations. There are alternative methods such as several kinds of Min-Sum (MS) algorithms which can significantly reduce the hardware complexity of SP at the cost of acceptable performance degradation where complex computations at the check nodes are approximated by using simple comparison and summation operations.

The optimal iterative decoding is performed by the Sum- Product algorithm at the price of an increased complexity, computation instability, and dependence on thermal noise estimation errors. The Min-Sum algorithm performs a suboptimal iterative decoding, less complex than the Sum-Product decoding. The sub-optimality of the Min-Sum decoding comes from the overestimation of check-node messages, which leads to performance loss with respect to the Sum-Product decoding.

Several correction methods were proposed in the literatures in order to recover the performance loss of the Min-Sum decoding with respect to the Sum-Product decoding which are called quasi optimal algorithms.

The decoder using semi-parallel architecture takes full advantage of the structure of the code and the hardware resources present in an FPGA but its throughput is less. the hardware implementation of this algorithm involves making a number of design choices that have a tradeoff between the BER performance and the complexity.

Throughput, power consumption, hardware complexity, error performance are the various factors which are to be considered for decoder design.

References

[1] Hossein Gharaee, Mahdie Kiaee "A High-Throughput FPGA Implementation of Quasi-Cyclic LDPC Decoder", International Journal of Computer Science and Network Security, VOL.17 No.3, March 2017

[2] Jin Sha, Minglun Gao, Zhongjin Zhang,Li La " Memory Efficient FPGA Implementation of Quasi-Cyclic LDPC Decoder" Int. Conf. on Instrumentation, Measurement, Circuits and Systems, Hangzhou, China, April 16-18, 2016 (pp218-223)

[3] N. Wiberg. Codes and decoding on general graphs. PhD thesis, Likoping University, 2015 Sweden.

[4] M.P.C.Fossorier, M. Mihaljevic, and H.lmai, Reduced Complexity Iterative Decoding of Low-Density Parity Check Codes Based on Belief Propagation, IEEE Trans. on Comm. May 2017,

[5] Xiang B, Shen R, Pan A, Bao D, Zeng X. An area-efficient and low-power multirate decoder quasi-cyclic low-density parity-check codes. IEEE Transactions on Very Large Scale Integration Systems. 2016 Oct; 18(10):1447–60.

[6] R. Gallager and L.-D. P.-C. Codes, "MIT press, 1963," Low Density Parity-Check Codes.

[7] T. Zhang. "Efficient VLSI Architectures for Error Correcting Coding." Ph.D. Thesis. 2012.

[8] S. Lin, and D.J. Costello, Error Control Coding: Fundamentals and Applications, Prentice-Hall, N.J, 2013

[9] Han, Wei; Huang, Jianguo; Fangfei Wu; , "A modified Min-Sum algorithm for low-density parity-check codes," Wireless Communications, Networking and Information Security (WCNIS), 2014 IEEE International Conference., pp.449-451

[10] Xiaofu Wu; Yue Song; Long Cui; Ming Jiang; Chunming Zhao; , "Adaptive-normalized min-sum algorithm," Future Computer and Communication (ICFCC), 2010 2nd International Conference on , vol.2, no., pp.V2-661-V2-663, 21-24 May 2010