Proceedings of the 5th WSEAS Int. Conf. on Instrumentation, Measurement, Circuits and Systems, Hangzhou, China, April 16-18, 2006 (pp218-223)

A Memory Efficient FPGA Implementation of Quasi-Cyclic LDPC Decoder

JIN SHA, MINGLUN GAO, ZHONGJIN ZHANG,LI LI

Institute of VLSI design
Key Laboratory of Advanced Photonic and Electronic Materials
Nanjing University
Nanjing, China, 210093

ZHONGFENG WANG

School of EECS
Oregon State University
Corvallis, OR 97331-5501,USA

Abstract - Low-Density Parity-Check (LDPC) code is one kind of prominent error correcting codes (ECC) being considered in next generation industry standards. The decoder implementation complexity has been the bottleneck of its application. This paper presents an implementation of Quasi-Cyclic Low-Density Parity-Check decoder by using FPGA. Modified Min-Sum decoding algorithm is applied to reduce the memory size needed for information storage. Partially parallel structures, memory management and pipelining schemes are discussed in this paper.

Key-Words: - Low-density parity-check (LDPC) codes, VLSI architecture, decoder, Quasi-Cyclic, memory

1 Introduction

A basic communication system is composed of three parts: a transmitter, channel, and receiver. Transmitted information is usually corrupted due to noise and channel distortion. To correct these errors, redundancy coding is intentionally introduced, and the receiver employs a decoder to make corrections based on the redundancy. Turbo codes and LDPC codes [1] are the two most popular error-correcting control codes (ECC) which showing results very close to the Shannon limit. Turbo code is overwhelmed by LDPC code in some aspects such as lower error floor, less computation requirement and fully parallel decoding schemes.

However, the implementation of LDPC decoder is not trivial. Implementing it directly in its inherent parallel manner may get the highest decoding throughput，while the complex interconnection will take up more than half of the chip area to avoid routing conflict for large codeword length (bigger than 1K). A rate-1/2, 1024-bit LDPC decoder [2] implemented in 0.16μm technology occupies an area of 7mm×7mm, where logic density is only 50%. Serial VLSI architecture or partly parallel architecture is well studied nowadays [3] [4] [5] [6], which uses RAM to store messages transferring between variable nodes (VN) and check nodes (CN). Therefore, a huge memory requirement for message storage will be a problem.

As for the decoding algorithm, the minimum-sum algorithm [7] [8] is an approximation of the Sum-Product algorithm (SPA). From the perspective of implementation, the Min-Sum algorithm requires less computation and the noise power estimation is unnecessary for an additive white Gaussian noise (AWGN) channel. Furthermore, the Min-Sum algorithm can help reducing the message storage requirement because the messages transmitted from a check node to adjacent variable nodes have only two possible magnitudes per iteration. However, this advantage is not easy to take in hardware implementation. In [5], the authors implemented an LDPC decoder utilizing the Min-Sum algorithm. The memory requirement reduction is achieved, but the hardware is complex, and the decoding latency is extremely high. The efficient use of Min-Sum algorithm in decoding LDPC codes still remains unresolved.

In this paper, the VLSI design issues of a memory efficient VLSI implementation for Quasi-Cyclic LDPC (QC-LDPC) codes are discussed. In Section 2, a brief review for the modified Min-Sum decoding algorithm and QC-LDPC codes is given. Section 3 gives the VLSI architecture of our decoder. Synthesis

results based on FPGA are given in Section 4. Section 5 concludes the paper.

2 Min-sum decoding algorithm and QC-LDPC codes

LDPC codes can be typically defined by an M × N parity check matrix H. The symbol N represents the length of the block (i.e. the number of bits in the codeword), while the symbol M represents the number of parity checks in the code. The rate of such a code is thus (N-M)/N. The LDPC code can be represented by a bipartite graph (Tanner graph) of variable nodes and check nodes as shown in Fig.1.

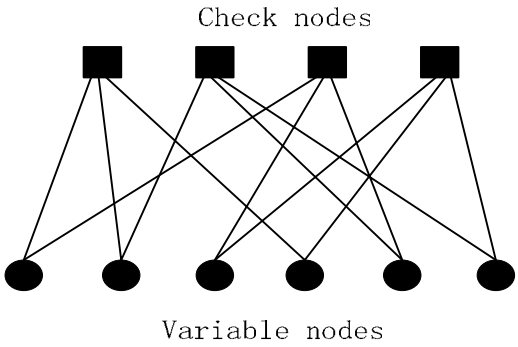

Figure.1 Tanner graph for a regular (2,3) LDPC code

The typical LDPC decoding algorithm is the sum-product algorithm which has two phases. In the first phase, the variable nodes compute updated information which is sent to adjacent check nodes. In the second phase, the check nodes compute updated information based on the new massages from the variable nodes. This update information is then sent back to adjacent variable nodes and the process is repeated over and over again.

The modified min-sum decoding algorithm is similar to the sum-product algorithm, with an approximation of check node process. It has some advantages in implementation against the sum-product algorithm, such as less computation and estimation of noise power is unnecessary for an AWGN channel.

In the modified min-sum decoding algorithm, the check node processors compute the check-to-variable messages R_{cv} through the following formulation:

$$R_{cv} = \alpha \times \prod_{n \in N(c)\backslash v} sign(L_{cv}) \times \min_{n \in N(c)\backslash v} |L_{cv}| \quad (1)$$

where α is a scaling factor. L_{cv} is the variable-to-check messages. $N(c)$ denotes the set of variable nodes that participate in cth check node.

The variable processor computes the variable-to-check messages L_{cv} through the following formulation:

$$L_{cv} = \sum_{m \in M(v)\backslash c} R_{mv} + I_v \quad (2)$$

Where, $M(v)\backslash c$ denotes the set of check nodes connected to the variable node v excluding the variable node c. I_v denotes the intrinsic message of variable node v.

Quasi-Cyclic LDPC code is a kind of highly structured LDPC code which can achieve comparable performance to random codes. QC-LDPC codes are well suited for hardware implementation, especially in that its encoder can be easily built with shift-registers [9]. In addition, the structure in the parity check matrix of QC-LDPC codes can facilitate efficient partially parallel [4] [6] decoding architecture. A normally regular *(c, t)* QC-LDPC code is defined as its parity check matrix H ($M \times N$) consisting of a $c \times t$ array of $P \times P$ sub-matrices which is a circulant of identity matrix $\boldsymbol{I}$.

$$H = \begin{bmatrix} H_{11} & H_{12} & \cdots & H_{1t} \\ H_{21} & H_{22} & \cdots & H_{2t} \\ \cdots & \cdots & \ddots & \cdots \\ H_{c1} & H_{c2} & \cdots & H_{ct} \end{bmatrix} \quad (3)$$

Each sub-matrix $\boldsymbol{H_{ij}}$ is a circulant matrix of identity matrix.

Here gives a $\boldsymbol{H_{ij}}$ example with P=6 and offset=3:

$$H_{ij} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \quad (4)$$

3 Decoder architecture

This section gives the VLSI architecture of our decoder.

3.1 The partially parallel decoder architecture

Several papers have addressed partially parallel decoding architecture for QC-LDPC codes such as [4] and [6]. This kind of architecture generally achieves a good trade-off between hardware complexity and decoding throughput. Considering a regular LDPC code with column weight c and row weight t, t variable node processors (VNP1,VNP2, ... VNPt) and c check node processors (CNP1, CNP2, ... , CNPc) are instantiated, and totally $t \times c$ memory banks are used for storing the messages exchanging between VNPs and CNPs. In each clock cycle, each sub-block (with dimension P $\times$ P) provides a message data for processing and gets back the new message calculated. The architecture presented in this paper is based on the architecture described in [4].

In [7], it is stated that the implementation of Min-Sum is more robust against quantization error, comparing with SPA. So, 4 bits uniform quantization method is applied. Our simulation shows that this quantized modified min-sum algorithm performs very close to the ideal belief-propagation algorithm.

Some modifications are further applied to reduce the message memory.

3.2 The method to save message memory

Aiming at QC-LDPC code structure, a more memory efficient decoding schedule can be applied by using the Min-Sum decoding algorithm.

Fig.2 shows the start points of variable process per iteration. In the shadowed place, the t variable-to-check messages generated contemporarily are in the same row. Hence, the CNP can process these messages immediately after the variable process. It gives an opportunity to avoid storing the variable to check messages of the shadowed part. For 1/ c part of the matrix, the check node process can be done immediately after the variable node process. And, the check to variable messages of the shadowed part can be saved in compressed form just like what [5] mentioned, which is composed of four elements, i.e., 1) the smallest magnitude, 2) the second smallest magnitude, 3) the index of the smallest magnitude, and 4) the signs of all messages.

Here a quantitative comparison of the needed memory between the proposed method and the conventional designs is given [4]. Using a commonly used (4,32) regular QC-LDPC code mentioned in [10], with message quantized to four bits, this architecture can store the messages for each row with 43 bits (32 for signs, 5 for index and 6 for two magnitudes) instead of 128 bits (4 x 32). This saves about 17% message memory in total. In addition, the memory access operation is reduced from 30 times to 1 time (96 percent reduced) for the shadowed place, which is significantly power saving.

The decoding steps are given below:

a) Initialization: Read the values from channel and store them in t memories.

b) Iteration: Compute the message from Variable nodes to Check nodes and save them in message memory MEMij except the upper P rows. For the upper P rows, do the check node processing and save the returned message in MEM1 with compressed form.

c) Check node process. Compute the check to variable messages of the lower 2P rows and save back in MEMij. Do iteration until all the check equations are satisfied or the maximum iteration number reached.

d) Output the decoded codeword.

3.2.1 Memory Banks

Fig.2 shows the memory management of our decoder. A (3,5) code is used for clarity.

As indicated in Fig.2, MEM1 is the message memory bank for the shadowed place in the parity check matrix. It saves only the check to variable messages, in compressed form. MEMij (i=2,3. j=1~5) saves the messages of subblock (i,j) of the parity check matrix. For each memory bank, there is an address generator to control the memory access. Because of the special structure of QC-LDPC codes, the address generator for each memory bank can be built with a simple counter. The VNP and CNP will get the input from appropriate MEMs and save back the computed messages at the same address. Besides this, five memory banks are instantiated to save the channel information, each for every P variable nodes.

3.2.2 Memory Partition

Utilizing the partially parallel decoding architecture, for a single decoding step, two access operations need to be done for each memory bank, one time read and one time write. To save area, single-port memory is employed, instead of the dual-port memory. In order to enable the read and write process of one memory bank

in one clock cycle, the memory partition method [8] is adopted. Every memory bank is partitioned into two blocks, one for the even row numbers and the other for the odd row numbers. They can be distinguished from the last bit of the address generated. Table.1 illustrates the address generation method and the memory access procedure for MEMij. Table.2 illustrates the memory access procedure of MEM1. They are different because a delay with two extra clock cycle is induced by the pipeline in CNP1. It will be illustrated more clearly later. Message package [4] or more memory partition schemes [8] can be used to increase the parallel level and linearly increase the decoding speed.

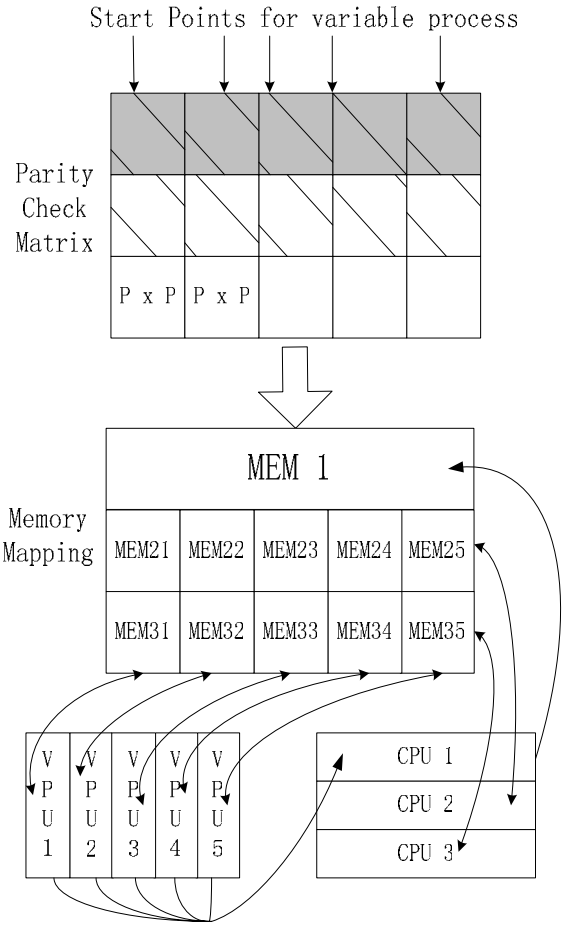

Figure.2 The mapping of the parity check matrix to message memorys and the decoding schedule

TABLE I. MEMORY ACCESS OF MEMIJ

Address Generator	Memory	
	Bank A	***Bank B***
00 0	Read ADDR 0	/
00 1	Write ADDR 0	Read ADDR 0
01 0	Read ADDR 1	Write ADDR 0
01 1	Write ADDR 1	Read ADDR 1
10 0	Read ADDR 2	Write ADDR 1
...	...	...

TABLE II. MEMORY ACCESS OF MEM1

Address Generator	Memory	
	Bank A	***Bank B***
00 0	Read ADDR 0	/
00 1	/	Read ADDR 0
01 0	Read ADDR 1	/
01 1	Write ADDR 0	Read ADDR 1
10 0	Read ADDR 2	Write ADDR 0
10 1	Write ADDR 1	Read ADDR 2
11 0	Read ADDR 3	Write ADDR 1
...	...	...

3.3 CNP architecture

Fig.3 gives the CNP architecture of the shadow place. It finds the smallest two inputs and the index of the minimum one. Function of the sub-module MIN is to record the minimum, 2^{nd}-Min and the index of the minimum.

This check node process can be time consuming for a big row weight matrix. It has $log_2(t)$ levels of comparators. Plus the time needed in the variable node process, the critical path is long. To increase the clock speed, the data paths are cut by two level pipelining. It can be seen in Table.2 that the output of CNP1 is delayed with two more clock cycles. Hence, the pipelining can increase the clock speed without inducing memory access conflict.

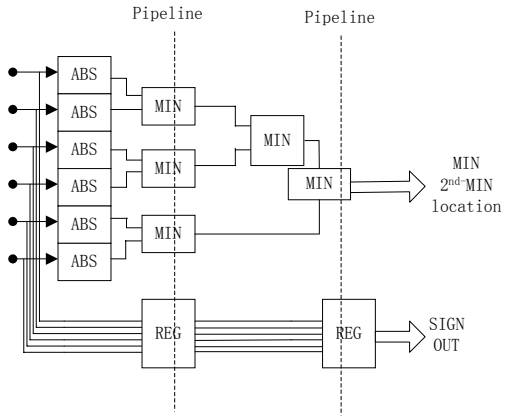

Figure.3 Architecture of CNP1

3.3 VNP architecture

Fig.4 shows the architecture of variable node processor. The input messages are firstly transferred to two's complement format and then do the add operation. Finally they are transferred back to sign and magnitude format. The scale module is shown in Fig.5.

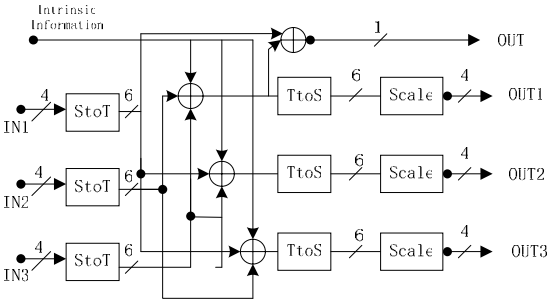

Figure.4 Architecture of VNP

There are some methods to improve the performance of Min-Sum product algorithm [10]: to scale the variable to check node message or to minus an offset. The algorithm is developed to suit for hardware implementation. Here gives the pseudo code:

if input >= 8 output = 3'b111;

else if input >= 4 output = input-1;

else output = input (unchanged);

Fig.5 shows the circuit. According to performance simulation, it is quite clear that this quantized scaling method is accuracy and performs well with QC-LDPC codes.

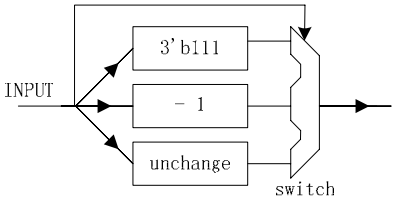

Figure.5 Scale Module Architecture

4 Implementation Result

The LDPC decoder is implemented as a synthesizable Verilog HDL model. Results are obtained with the Quartus synthesis tool.

It is worth noting that the RTL code is fully parameterized. The parameter P and shift value for each sub-matrix can be easily reconfigured. Here we choose a (3,6) code with P = 256, and get the codeword length 1536, code rate 1/2. Table Ⅲ gives the mapping result of our decoder on the Altera Cyclone EP1C12Q240C7 device. The maximum clock frequency is 90MHz. The throughput is 9Mbps (with 15 iteration times). As stated above, by instantiating more processing units and applying massage pack or more memory partition, the decoder throughput can be linearly increased.

TABLE III. ALTERA FPGA MAPPING RESULT

Resource	Used
Logic elements	1536
Memory bits	22272
Memory blocks	32

5 Conclusions

A memory efficient partially parallel architecture for decoding QC-LDPC codes has been designed and implemented on FPGA. The special structure of the parity check matrix not only makes the memory addressing simply, but also enables the use of modified Min-Sum decoding algorithm to save the message memory. This architecture can be easily configured for different code rates, block sizes, and parallelism factors.

6 Acknowledgments

The work presented in this paper was supported by the Foundation of High-Tech of Jiangsu Province of China under Grant No.BG2005030; the National Nature Science Foundation of China under Grant No.90307011.

References:

[1] R.G.Gallager. “Low density parity check codes.” IRE Trans. Info. Theory, vol. IT-8,pp.21-28,1962.

[2] A.J.Blanksby and C.J.Howland, “A 690-mW 1-Gbps 1024-b, Rate-1/2 Low-Density Parity-Check Code Decoder” IEEE J.Solid-State Circuits, vol.37, pp.404-412,2002.

[3] T.Zhang. “Efficient VLSI Architectures for Error-Correcting Coding.” Ph.D. Thesis. 2002.

[4] Marjan Karbooti “Semi-Parallel Reconfiguable Architectures for Real-Time LDPC Decoding” Proceedings of the International Conference on Information Technololy: Coding and Compution(ITCC’04)

[5] Mauro Cocco “A Scalable Architecture for LDPC Decoding” Proceedings of the Design, Automation and Test in Europe Conference and Exhibition Designers’ Forum(DATE’04)

[6] Mohammad M.Mansour “Low-Power VLSI Decoder Architectures for LDPC Codes” ISLPED’02

[7] Jianguang Zhao, “On Implementation of Min-Sum Algorithm and Its Modifications for Decoding Low-Density Parity-Check (LDPC) Codes” IEEE Transactions on Communications VOL.53, NO.4 2005

[8] Jinghu Chen, “Reduced-Complexity Decoding of LDPC Codes” IEEE Transactions on Communications VOL.53, NO.8 2005

[9] Z.Li, L.Chen and S.Lin, W Fong and P Yeh, “Efficient Encoding of Quasi-Cyclic Low Density Parity-Check Codes,” IEEE Transactions on Communications

[10] L.Chen, J.Xun, I.Djurdjevic, and S.Lin, “Near Shannon Limit Quasi-Cyclic Low Density Parity Check Codes” IEEE Transactions on Communications

[11] Z.Wang, “Low Complexity, High Speed Decoder Architecture for Quasi-Cyclic LDPC Codes” ISCAS 2005, pp.5786-5789.

[12] J. Heo. “Analysis of Scaling Soft Information on Low Density Parity Check Codes” Elect. Letters, 39(2):219−221, Jan 2003.