A 170 Mbps (8176, 7156) Quasi-Cyclic LDPC Decoder Implementation with FPGA

Zhiqiang Cui and Zhongfeng Wang
School of Electrical Engineering and Computer Science
Oregon State University, Corvallis, OR 97333, USA
{cuizh, zwang}@eecs.oregonstate.edu

***Abstract*—This paper presents a low-complexity, high-speed VLSI decoder architecture and its FPGA implementation for Euclidian Geometry (EG) based Quasi-Cyclic (QC) Low-Density Parity-Check (LDPC) codes. In the design, various optimizations are employed to increase the clock speed. More parallelism is enabled for the partially parallel decoding architecture through the introduction of small hardware overhead. An efficient non-uniform quantization scheme is proposed to reduce the size of soft message memories without sacrificing the decoding performance. Synthesis results show that the proposed decoder for a (8176, 7156) EG-LDPC code can achieve a maximum (information) decoding throughput over 170 Mbps on Xilinx Virtex II FPGA when performing 15 iterations.**

I. Introduction

Recently, several classes of Quasi-Cyclic (QC) Low-Density Parity-Check (LDPC) codes [1]-[4], have been proposed that can achieve comparable performance with that of random codes. Among of them, a class of Euclidean Geometries (EG) based QC-LDPC codes have shown excellent error correction performance, which is below the error floor of equivalent computer generated random LDPC codes [4]. QC-LDPC codes are well suited for hardware implementation. The encoders of QC-LDPC codes can be built with shift-registers [5], while random codes usually require complex encoding circuitry to perform matrix and vector multiplications [6] [7]. In addition, QC-LDPC codes also facilitate efficient high-speed decoding because of regularity in their parity check matrices.

A memory-based partially parallel decoding architecture was presented [8] to obtain a good trade-off between hardware complexity and decoding speed. Recently, Chen [9] proposed a rate of 1/2, 8088-bit irregular QC-LDPC code decoder. The information decoding throughput is 40 Mbps at 25 iterations. Karkooti [11] implemented a decoder for (3, 6) 1536-bit QC-LDPC code. This proposed decoder can obtain 63.5Mbps information decoding throughput. However, to facilitate the target level of parallelism, the author added one additional constraint on each circulant matrix that the cyclic shifting value must be a multiple of the desired parallelism level. The added constraint unavoidably limits the performance of the LDPC codes.

In this paper, a low complexity, very high speed decoder architecture and its FPGA implementation for the (8176, 7156) EG-based QC-LDPC code is proposed. A worst-case source information decoding throughput (at 15 iterations) over 170Mbps is achieved. Optimizations at various levels are employed to increase the decoding throughput. An efficient non-uniform quantization scheme is proposed to reduce the size of soft message memories without sacrificing the decoding performance. The decoder architectures presented in this paper are suited for other QC-LDPC codes as well.

The structure of this paper is as follows. In section II, the (8176, 7156) EG-based QC-LDPC code and an algorithmic transformation are briefly introduced. Section III presents the decoder architectures for the LDPC code. Section IV proposes the non-uniform quantization and corresponding datapath architectures. Section V discusses the FPGA implementation results. Finally, conclusions are drawn in section VI.

II. Decoding of LDPC Codes

A. The (8176, 7156) EG-based QC-LDPC Code

The EG-based QC-LDPC codes are constructed based on the decomposition of finite Euclidean geometries. The (8176, 7156) code (originally designed for NASA) is a regular QC-LDPC code with a column weight of 4 and a row weight of 32 [4]. The parity-check matrix **H** is a 2×16 array of thirty-two 511×511 submatrices as shown in the following expression:

$$\mathbf{H} = \begin{bmatrix} \mathbf{H}_{1,1} & \mathbf{H}_{1,2} & \cdots & \mathbf{H}_{1,16} \\ \mathbf{H}_{2,1} & \mathbf{H}_{2,2} & \cdots & \mathbf{H}_{2,16} \end{bmatrix}$$

Each submatrix is a circulant matrix with both column and row weight of 2.

B. An Efficient Decoding Scheme

Among various LDPC codes decoding algorithms, the Sum-Product Algorithm (SPA) has the best decoding performance. The conventional SPA algorithm has unbalanced computation complexity between the variable-to-check and check-to-variable message updating phases. This leads to unbalanced datapaths between the processing units for the two message updating phases. To balance the computation load and reduce the critical path, a modified version based on algorithmic transformation was proposed in [10]. The reformulated algorithm for check node processing, variable node processing, and tentative decoding is expressed as follows:

$$R_{cv} = \prod_{n\in N(c)\setminus v} sign(L_{cn}) \sum_{n\in N(c)\setminus v} \Psi(L_{cn}), \quad (1)$$

$$L_{cv} = \sum_{m\in M(v)\setminus c} (sign(R_{mv})\Psi(R_{mv})) - \frac{2r_v}{\sigma^2}, \quad (2)$$

$$L_v = \sum_{m\in M(v)} (sign(R_{mv})\Psi(R_{mv})) - \frac{2r_v}{\sigma^2}, \quad (3)$$

where, $\Psi(x) = -log(tanh(|x/2|))$, R_{cv} and L_{cv} stand for the check-to-variable message and the variable-to-check message, respectively.

III. Partially Parallel Decoder Architecture

A. The Check Node and Variable Node Processing Units

Fig. 1 shows the architecture of a Check node Processing Unit (CPU), which performs R_{cv} message computation. The *LUT-A* is introduced to perform the function $\Psi(x)$. The magnitude of the output is the sum of 31 magnitudes out of 32 input data. The sign bit of the output is the product of 31 out of 32 sign bits of the input data. To reduce the addition delay in the last addition stage, each word of the two addends is separated into higher and lower parts. Thus, two partial additions are performed in parallel. To reduce the critical path in the CPU, pipeline latches are inserted as indicated by the dashed lines. The data representations for the inputs of CPU, the output of *LUT-A*, and the final outputs of CPU are in the forms of two's complement, unsigned, and sign-magnitude, respectively.

The architecture of a Variable node Processing Unit (VPU) is illustrated in Fig. 2, which performs L_{cv} message computation. Z and C stand for the intrinsic message and the tentative decoding bit, respectively. The *LUT-B* performs the function $sign(R_{mv})\Psi(R_{mv})$. It is convenient to use the two's complement format in VPU computations. Thus, the intrinsic message Z, the output of *LUT-B*, and the outputs of VPU are all in two's complement format. The input of *LUT-B* is in sign-magnitude format.

B. Enhanced Partially Paralle Decoder Architecture

Conventionally, 1 CPU performs check-to-variable messages updating for 1 row of matrix **H** per clock cycle and is assigned for each block row of matrix **H**. Similarly, 1 VPU performs variable-to-check messages updating for 1 column of matrix **H** per clock cycle and is assigned for each block column of matrix **H**. To increase the parallelism, we propose an enhanced architecture that enables processing multiple rows/columns corresponding to each submatrix of **H** at the same time. In this design, only double parallelism is considered (i.e., the number of VPUs and CPUs are $2\times16=32$ and $2\times2=4$, respectively), though the proposed architecture can be easily extended to higher parallelism cases. The key issue for this enhancement is how to access 4 soft messages corresponding to each submatrix at each clock cycle. For the considered EG-LDPC code, each submatrix consists of 2 superimposed cyclically shifted identity matrices. Thus, two memory modules are allocated to store soft messages corresponding to each submatrix. Because the cyclic shifting offsets are generally not multiples of the targeted parallelism level, to facilitate multiple data accesses for each cyclically shifted matrix in both row and column updating phases, memory partitioning and data switching schemes are developed. Apparently, the proposed architecture is also suited for decoders using Min-Sum algorithm [12]. On the other hand, multiple adjacent soft messages can be stored at one memory entry to increase the parallelism, where extra buffers and data switching networks are needed [14].

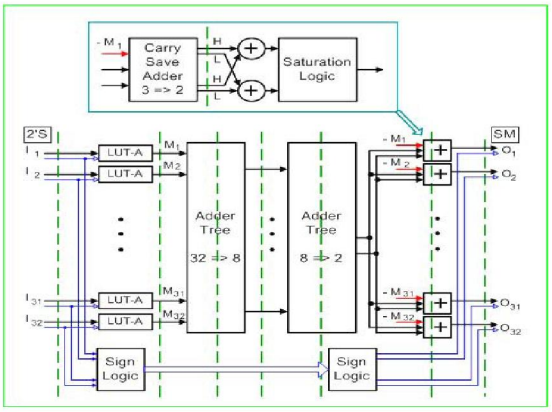

Figure 1. Check node processing unit architecture.

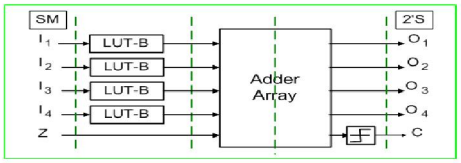

Figure 2. Variable node processing unit architecture.

The block diagram of the proposed architecture is shown in Fig. 3. Each memory block $M_{i,j}$, which consists of two memory modules, corresponds to a circulant matrix $H_{i,j}$ of the **H** matrix. They are used to store the extrinsic soft message conveyed at the both decoding phases. The memory modules Z_i and C_i are used to store the intrinsic soft messages and the estimated codeword bits, respectively. Each memory module for a cyclically shifted identity matrix is partitioned into an even-addressed bank containing soft messages corresponding to 1-components in the even rows of the submatrix and an odd-addressed bank containing data corresponding to the odd rows. Thus, to ensure 2 data in both row and column updating phases can be accessed without confliction and be sent to the right processing units, three different data switching schemes are needed, which correspond to the cases of odd, even, and zero shifting offsets that range from 0 to 510. Due to the limited space, only the data switching scheme for an odd shifting offset is illustrated. The other two schemes can be developed in a similar way.

Fig. 4 shows an example of the memory partitioning and data switching schemes applied to a 15×15 cyclically shifted identity matrix with an odd shifting offset of 5. A soft message $p(i,j)$ saved in a memory sub-bank corresponds to a 1-component located at row i and column j of a cyclically shifted identity matrix. In the check-to-variable message updating phase, the two data located in the even memory sub-bank MEM_E and the odd memory sub-bank MEM_O with the same index are sent to CPU_E and CUP_O in parallel, respectively, which are CPU components for even row and odd row message updating. In the variable-to-check message updating phase, the two data connected by an arrow are sent to VPU_0 and VPU_1 in the same clock cycle, respectively. Here, VPU_0 and VPU_1 are VPU components for even column and odd column data computation.

It can be seen that the data located in the even columns are distributed in both even and odd memory sub-banks. Similar case exists for the data located in the odd columns. Therefore, switching units are needed to route data between memories and VPUs in the

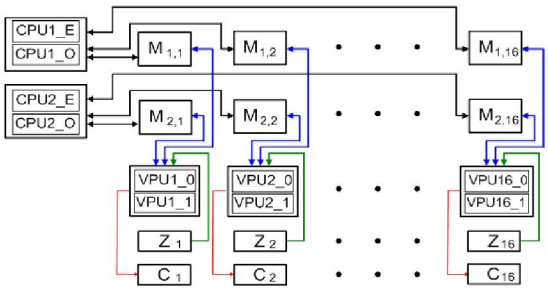

Figure 3. Enhanced partially parallel decoder architecture.

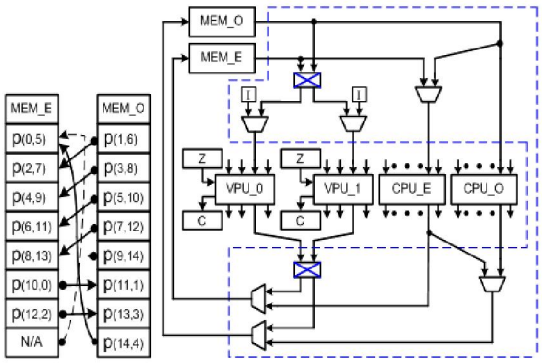

Figure 4. Memory partitioning and data switching scheme.

variable-to-check message updating phase. If the memory is partitioned in a straightforward way, the last data in the even row should be stored in the last location of memory sub-bank MEM_E. Consequently, data access confliction indicated by the dashed arrow occurs when the two data from column 4 and 5 are retrieved from the memory sub-bank MEM_E in the same clock cycle. In our design, the last data in the even row is stored in the odd memory sub-bank to eliminate the data access confliction. In this way, a pair of multiplexers are needed to steer the displaced data between the MEM_O and the CPU_E. In this figure, symbol *I* represents a fixed data value for initialization procedure.

C. *Architecture of the Controller*

The controller, which generates the control signals of the data switch networks and memory addresses, is composed of a two-level finite state machine. The first level includes 5 states, i.e., idle, initialization, row processing, anti-overlap, and column processing. In the initialization state, the intrinsic soft messages stored in the memory *Z* are transferred into the memory *M*. The anti-overlap state is introduced to avoid the data access confliction between the two decoding phases caused by pipelining stages.

For high speed design, controller could also be a bottleneck for decoder implementation. In this design, the clock speeds of pipelined CPU and VPU exceed 200MHz. Therefore, extensive optimization is needed for controller implementation. The block diagram of the controller is shown in Fig. 5. The memory write addresses and the data switching control signals for writing are the delayed versions of the memory read addresses and the control signals for reading, respectively. To reduce the critical path of the controller, retiming technique is employed. By introducing one delay element, the critical path can be significantly reduced, though one clock cycle latency is introduced as well.

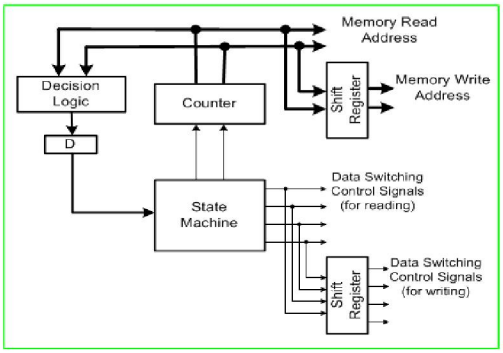

Figure 5. Block diagram of controller.

IV. Fixed-point Implementation

The word length of the soft messages determines the memory size, the computation unit size and the decoding performance of a LDPC codes decoder. The overall hardware of LDPC decoder is predominantly determined by the size of the memories holding intrinsic and extrinsic soft messages. Therefore, an efficient quantization schemes is very important for LDPC decoder implementation.

A. *Uniform and Non-uniform Quantization Schemes*

Let $q:f$ denote the uniform quantization scheme in which the finite word length is q bits, of which f bits are used for the fractional part of the value. A non-uniform quantization scheme which generally out-performs the uniform quantization under the same word length was proposed [13]. However, in this method, a q-bit non-uniform quantization scheme generally performs worse than the uniform quantization case with (q+1)-bit word length since the precision is significantly reduced when $x \geq 1$. This paper presents an improved non-uniform quantization scheme that can achieve a decoding performance almost identical to that of the uniform quantization case with 1-bit longer word length.

In the both decoding phases, the extrinsic soft messages are sent to look-up tables, which perform the nonlinear function $\Psi(x)$. A close study of the 7:4 uniform quantization results of $\Psi(x)$ reveals that no quantized values of $\Psi(x)$ are lost if 6:4 and 6:3 uniform quantization are applied to x when $0 \leq x < 3$ and $3 \leq x \leq 4.875$, respectively. The values of $\Psi(x)$ are all 0 when $x \geq 4.875$. Therefore, we can use 6 bits to represent these 64 distinct cases for x. In the new non-uniform quantization scheme, the soft messages in uniform quantization computed from equations (1) and (2) are converted into 6-bit non-uniform quantization to reduce the word length and are stored into memories for extrinsic messages. To avoid increasing the complexity of computation units, the values of $\Psi(x)$, which are stored in lookup table, are in uniform quantization. The proposed 2-step non-uniform quantization also ensures simple data format conversion. Fig. 7 shows that using the 6-bit non-uniform quantization scheme can achieve almost identical decoding performance to that using the 7:4 uniform quantization scheme.

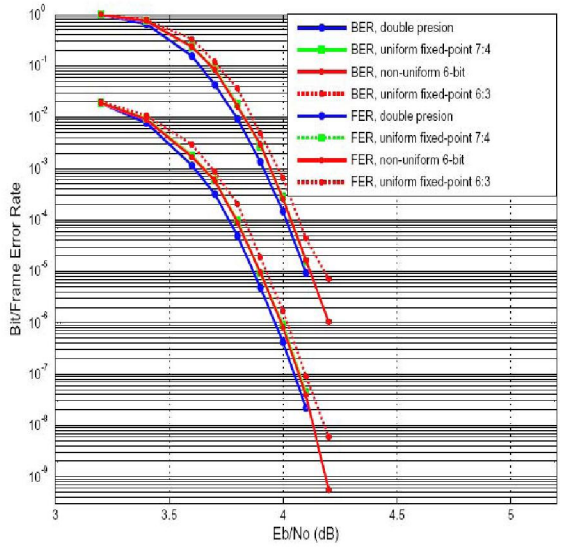

Figure 6. Comparison of fixed-point vs. double precision simulation.

B. *Check Node and Variable Node Processing Units with Non-uniform Quantization*

The new architectures for CPU and VPU with the non-uniform quantization scheme are shown in Fig. 7 and Fig. 8, respectively. The uniform to non-uniform quantization converter, *U2NU*, is implemented with simple combinational logic. The look-up table, i.e., *LUT*, for both CPU and VPU are the same.

V. FPGA Implementation

Based on the architectures described above, the (8176, 7156) EG-based LDPC code decoder was modeled in VHDL, simulated and synthesized targeting on the Xilinx Virtex II-6000. After place and route, the Xilinx TRACE report shows that the maximum clock frequencies of the uniform and non-uniform quantization implementation are 193.4MHz and 192.6MHz, respectively. Table I shows the FPGA utilization statistics of both implementations.

The maximum iteration number is set as 15. It takes half of an iteration to transfer the intrinsic soft messages from memory Z into memory M of the decoder. It takes $2\times 256+5=517$ clock cycles to perform one iteration in which 5 clock cycles are allotted to the anti-overlap state. Thus, the implementation with non-uniform quantization can achieve a maximum information decoding throughput of $(7156\times 192.6)/(15\times 517+256)\approx 172$ Mbps. Because the computation part of CPU and VPU with 6-bit non-uniform quantization implementation is the same as that with 7-bit uniform quantization implementation and our synthesis optimization goal is speed, the 6-bit non-uniform quantization implementation needs more logic resource than 6-bit uniform quantization implementation. However, the size of memory and the data width of global routing for extrinsic message in the two implementations are the same.

VI. Conclusion

In this paper, the design and implementation of a low complexity, very high speed decoder for EG-based QC-LDPC codes is presented. Various techniques were incorporated to increase the clock speed. An enhanced partly parallel decoder that can linearly increase the throughput of traditional ones is proposed. An efficient non-uniform quantization scheme is proposed to reduce memory size of the decoder without performance loss. The FPGA implementation with Xilinx Virtex II 6000 achieves a maximum decoding throughout of over 170 Mbps at 15 iterations, which is more than twice faster (in terms of Mbps per iteration) than other published LDPC codec implementations based on similar platforms (*e.g.*, [8][9][11]).

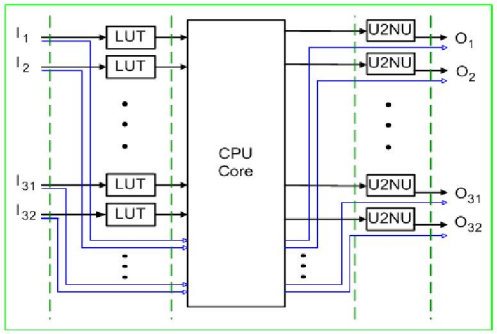

Figure 7. Check node processing unit with non-uniform quantization.

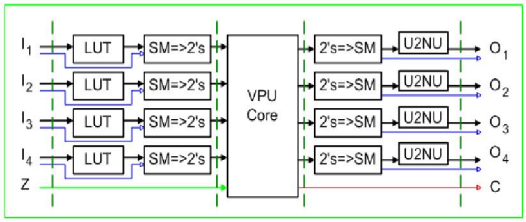

Figure 8. Variable node processing unit with non-uniform quantization.

Table I. Xilinx VirtexII-6000 FPGA Utilization Statistics

Resource	6-bit uniform quantization		6-bit non-uniform quantization	
	Used	Utilization ratio	Used	Utilization ratio
Slices	23052	68%	27,460	81%
Slice Flip Flops	26926	39%	38,266	56%
4-input LUTs	28229	41%	36848	54%
Block RAMs	128	88%	128	88%

References

[1] R. M. Tanner, D. Sridhara, A. Sridharan, T.E. Fuja, and D. J. Costello, "LDPC block and convolutional codes based on circulant matrices," *IEEE Trans. on Inform. Theory*, vol. 50, pp. 2966-2984, Dec. 2004.

[2] M. P. C. Fossorier, "Quasi-cyclic low-density parity-check codes from circulant permutation matrices," *IEEE Trans. on Info. Theory*, vol. 50, pp. 1788 – 1793, Aug. 2004.

[3] Z. Li and B. V. K. V. Kumar, "A class of good quasi-cyclic low-density parity check codes based on progressive edge growth graph," *Thirty-Eighth Asilomar Conference on Signals, Systems and Computers,* vol. 2, pp. 1990-1994, 2004.

[4] L. Chen, J. Xun, I. Djurdjevic and S. Lin, "Near shannon limit quasi-cyclic low-density parity-check codes," *IEEE Trans. on Communications*, vol 52, pp. 1038-1042, July 2004.

[5] Z. Li, L. Chen, S. Lin, W. Fong and P. Yeh, "Efficient encoding of quasi-cyclic low-density parity-check codes," to appear in IEEE trans. on communications.

[6] T. Richardson and R. Urbanke, "Efficient encoding of low-density parity-check codes," *IEEE Trans. on Inform. Theory*, vol 47, pp. 638–656, Feb. 2001.

[7] D. –U. Lee, W. Luk, C. Wang, C. Jones, "A flexible hardware encoder for low-density parity-check codes," *IEEE Symp. on FCCM'04*, pp. 101-111.

[8] Tong Zhang and Keshab Parhi, "A 54 Mbps (3,6)-regular FPGA LDPC decoder," *IEEE SiPS'2002*, pp 127-132, 2002.

[9] Y. Chen and D. Hocevar, "A FPGA and ASIC implementation of rate 1/2, 8088-b irregular low density parity check decoder," *IEEE GLOBECOM '03*, vol. 1, pp. 113-117, Dec. 2003.

[10] Z. Wang, Y Chen and K Parhi, "Area-efficient decoding of quasi-cyclic low density parity check codes," *ICASSP 2004*, vol. 5, pp.49-52, May 2004.

[11] M. Karkooti and J. R. Cavallaro, "Semi-parallel reconfigurable architectures for real-time LDPC decoding," *ITCC'2004*, vol. 1, pp. 579 – 585, Apr. 2004.

[12] J. Chen and M. Fossorier, "Near optimum universal belief propagation based decoding of low-density parity check codes," *IEEE Trans. Commun.*, vol. 50, pp. 406–414, Mar. 2002.

[13] T. Zhang, Z. Wang, K.K. Parhi, "On finite precision implementation of low density parity check codes decoder," *The 2001 IEEE International Symposium on Circuits and Systems*, vol.4, pp. 202-205, May 2001.

[14] Z. Wang and Q. Jia, "Low complexity, high speed decoder architecture for quasi-cyclic LDPC codes," *The 2005 IEEE International Symposium on Circuits and Systems,* pp. 5786-5789, May 2005.