PERFORMANCE ANALYSIS AND IMPLEMENTATION FOR NONBINARY QUASI-CYCLIC LDPC DECODER ARCHITECTURE

Tony Tsang[1]

[1]Department of Computer Engineering, La Trobe University, Melbourne, Australia.

ABSTRACT

Non-binary low-density parity check (NB-LDPC) codes are an extension of binary LDPC codes with significantly better performance. Although various kinds of low-complexity iterative decoding algorithms have been proposed, there is a big challenge for VLSI implementation of NBLDPC decoders due to its high complexity and long latency. In this brief, highly efficient check node processing scheme, which the processing delay greatly reduced, including Min-Max decoding algorithm and check node unit are proposed. Compare with previous works, less than 52% could be reduced for the latency of check node unit. In addition, the efficiency of the presented techniques is design to demonstrate for the (620, 310) NB-QC-LDPC decoder.

KEYWORDS

Wireless Communications; NB-QC-LDPC; VHDL; Performance Analysis

1. INTRODUCTION

Channel coding plays key role in providing a reliable communication method that can overcome signal degradation in practical channels. A new field of study into non-algebraic codes based on linear transformations using generator and parity check matrices are led off the breakthrough of convolutional codes [1]. Using a finite-state process is encoded the Convolutional codes, which generates them a linear order encoding scheme. Afterward, convolutional codes led to the discovery of a class of codes called Turbo codes [2], which are the class of concatenated convolutional codes and randomize the order of some of the bits by using an inter-leaver. The first known capacity approaching error correction code is Turbo code, which provides a powerful error correction capability when decoded by an iterative decoding algorithm. The rediscovery of low density parity check (LDPC) code, which was originally proposed by Gallager [3] and was later generalized as MacKay-Neal [4] code puts back Turbo coding as the forward error correction (FEC) technique. LDPC codes were neglected for a long time since their computational complexity for the hardware technology was high. LDPC codes have acquired considerable attention due to its near capacity error execution and powerful channel coding technique with an adequately long code-word length. There are several advantages LDPC codes over Turbo codes. In the decoding of Turbo codes it is difficult to apply parallelism due to the sequential nature of the decoding algorithm, while in LDPC decoding can be accomplished with a high degree of parallelism to attain a very high decoding throughput. Since, turbo codes usually cause a large delay, but LDPC codes do not need a long inter-leaver. LDPC codes can be constructed directly for a desired code rate. In case of turbo codes, which are based on convolutional codes, require other methods such as puncturing to acquire the desired rate.

The codes are classified into two major categories, explicitly, block codes and convolutional codes. Hamming codes, Bose-Chaudhuri-Hocquenghem (BCH) codes, Reed-Solomon (RS) codes and newly rediscovered LDPC codes are the example of block codes. Block codes like Hamming, BCH and RS codes have structures but with limited code length. A bounded-distance decoding algorithm is usually employed in decoding block codes except LDPC codes, in general it is hard to use soft decision decoding for block codes.

Advances in error correcting codes have revealed that, using the message passing decoding algorithm, irregular LDPC codes can accomplish consistent communication at SNR very close to the Shannon limit on the AWGN channel, outperforming turbo codes of the same block size and code rate. Hou et al [5] examined the numerical analysis method for calculating the threshold of the LDPC codes. For the AWGN channel; the proposed method is implemented to the uncorrelated flat Rayleigh fading channel. Additionally, using the nonlinear optimization technique of differential evolution, the degree distribution pairs are optimized for the uncorrelated Rayleigh fading channel and observe that their threshold values are very close to the capacity of this channel for moderate block size with excellent performance. Zhang et al [6] proposed the two adaptive coded modulation schemes employing LDPC codes for Rayleigh fading channels. The proposed schemes have made good use of the time-varying nature of Rayleigh fading channel. It is also observed that the proposed schemes perform better by employing LDPC with large code length.

The performance of irregular LDPC codes is investigated in [7] with three BP based decoding algorithms, specifically the uniformly most powerful (UMP) BP- based algorithm, the normalized BP-based algorithm, and the offset BP-based algorithm on a fast Rayleigh fading channel by employing density evolution. It is observed from the study of proposed method that the performance and decoding complexity of irregular LDPC codes with the offset BP-based algorithm can be very close to that with the BP algorithm on the fast Rayleigh fading channel. Ohhashi and Ohtsuki [7] provide successful evolution of irregular LDPC codes, and then analyze the performance of regular LDPC codes with the normalized BP-based algorithms on the fast Rayleigh fading channel. Formulas for short and long regular LDPC codes are derived based on the probability density function (PDF) of the initial likelihood information and DE for the normalized BP-based algorithm on the fast Rayleigh fading channel. Performance of the long regular LDPC codes with the normalized BP-based algorithm in the proposed method outperforms the BP algorithm and the UMP BP-based algorithm on fast Rayleigh fading channel. In this paper new Nonbinary LDPC codes have been developed and implement the newly designed codes on FPGA platform. Simulation results demonstrate that the proposed Nonbinary LDPC codes achieve a 0.8 dB coding gain over randomly constructed codes and perform 1 dB from the Shannon-limit at a BER of 10−6 with a code rate of 0.89 for block length of 620.

2. NONBINARY QC-LDPC CODES AND MIN-MAX DECODING ALGORITHM

2.1. Non-binary QC-LDPC Codes

StandardLet GF (q) be a finite field with q elements. A q -ary LDPC code C is given by the null space of a sparse parity-check matrix H = [hi;j] over GF(q) . If each column has constant weight (the number of nonzero entries in a column) and each row has constant weight γ (the number of nonzero entries in a row), the code C is referred to as a (γ,ρ) -regular LDPC code. If the columns and/or rows of the parity-check matrix have multiple weights, the null space of H gives an irregular LDPC code.

Consider the construction field GF (q) with α as a primitive element. Then $\alpha^{-\infty} = 0, \alpha^0 = 1, \alpha, \ldots \alpha^{q-2}$

give all the elements of GF(q) and $\alpha^{q-1} = 1$. We define α -multiplied circulate permutation matrix (CPM) as a (q −1)×(q −1) matrix over GF (q) . Each row of the α -multiplied CPM is the right cyclic-shift of the row above it multiplied by α ; the first row is the right cyclic-shift of the last row multiplied by α . Thus, each α -multiplied CPM is characterized by its offset, which denotes the position of the nonzero entry in the first row of the matrix. Generally, for 0 ≤ l < q −1 , the nonzero entry in the l th column of the α -multiplied CPM is α^l . For example, an α -multiplied CPM over GF(8) with an offset of 3 is shown as

$$\begin{bmatrix} 0 & 0 & 0 & \alpha^3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \alpha^4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \alpha^5 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \alpha^6 \\ \alpha^0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \alpha & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \alpha^2 & 0 & 0 & 0 & 0 \end{bmatrix}$$

If $\mathbf{H}$ is an array of α -multiplied CPMs and all-zero matrices, then the null space of gives a non-binary quasi-cyclic (QC)- LDPC codes. The non-binary QC-LDPC codes are usually constructed based on algebraic methods [8]. QC-LDPC codes are advantageous over other codes in terms of encoding complexity.

2.2. B. Min-Max Decoding Algorithm

A bipartite graph called Tanner graph, represented graphically the non-binary LDPC code C, which consists of two disjoint sets of nodes. Nodes in one set, called variable nodes (VNs), represent the code symbols; nodes in the other set, called check nodes (CNs), represent the check-sums that the code symbols must satisfy. For a code with a J × n parity-check matrix, label the VNs from 0 to n − 1 and the CNs from 0 to J − 1 . The i th CN is connected to the j th VN by an edge if and only if hi;j □= 0 . The VNs connected to the i th CN simply correspond to the code symbols that are contained in the i th check-sum. The number of these VNs is referred to as the CN degree of thei th CN. The CNs connected to the j th VN simply correspond to the check sums that contain the j th VN. The VN degree of the j th VN referred to the number of these CNs.

Message passing algorithms can iteratively decode the Non-binary LDPC codes. Instead of a single message (as for the binary codes), a vector of sub-messages are passed through each edge of the Tanner graph. The QSPA is approximated the min-max decoding algorithm [9] with reduced complexity and 0.1 V 0.2 dB of performance degradation, and thus is widely adopted for decoder implementation [10, 11].

For 0 ≤ i < J and 0 ≤ j < n, we define $N_i = \{j : 0 \le j < n, h_{i,j} \ne 0\}$, and $J_j = \{i : 0 \le i < J, h_{i,j} \ne 0\}$. Let Kmax be the maximum number of iterations to be performed. Let Lj be the a priori information of the j th code symbol from channel. Let $\mathbf{z} = (z_0, z_1, \dots, z_{n-1})$ be the hard decision symbols for code symbols generated either when initialized or during the VN processing. For 0 ≤ j < n , each $\mathbf{L}_j = (L_{j,0}, L_{j,1}, \dots, L_{\alpha^{q-2}})$ consists q of log likelihood ratios (LLRs) $\mathbf{L}_{j,\alpha^l} = log(Prob(z_j = \beta)) / log(Prob(z_j = \alpha^l))$

where 0 ≤ l < q − 2 , or l = −∞ , and is the most likely symbol for zj (i.e., $Prob(z_j = \beta)$ is the largest among all q probabilities). From this definition, all LLRs are non-negative. Also, the smaller the $\mathbf{L}_{j,\alpha^l}$ is, the most likely that the code symbol zj is α^l . Let $\mathbf{Q}_j = (Q_{j,0}, Q_{j,1}, \dots, Q_{j,\alpha^{q-2}})$ be the a posteriori information of the j th code symbol. Let $\mathbf{Q}_{j\to i} = (Q_{j\to i,0}, Q_{j\to i,1}, \dots, Q_{j\to,\alpha^{q-2}})$ and $\mathbf{R}_{j\leftarrow i} = (R_{j\leftarrow i,0}, R_{j\leftarrow i,1}, \dots, R_{j\leftarrow,\alpha^{q-2}})$ be the VN-to-CN and CN-to-VN message vectors passed between the j th VN and i th CN, respectively. The VN-to-CN and CN-to-VN messages are also referred to as extrinsic messages. Let be the sequence of finite field element assignments of $z_{j'}(j' \in N_i \setminus j, z_{j'} \in GF(q))$ such that $\sum_{j' \in N_i \setminus j} h_{i,j'} z_j' = h_{i,j} \alpha^l$

Let be the iteration counter. The min-max decoding is as follows.

Initialization : Set k = 0 . For all 0 ≤ j < n , set Qj = Lj , and zj = arg max_l (Qj ;_l) . For all i ; j , set Qj→i = Lj .

Step 1) Parity check: Compute the syndrome $\mathbf{zH}^\mathrm{T}$ of z. If $\mathbf{zH}^\mathrm{T} = 0$, stop decoding and output as the decoded code word; otherwise go to Step 2.

Step 2) If k = Kmax , stop decoding and declare a decoding failure; otherwise, go to Step 3.

Step 3) CN processing: Compute the CN-to-VN message
$R_{j\leftarrow i,\alpha^l} = min_{z_{j'} \in L_i, z_j = \alpha^l} (max_{j' \in N_i \setminus j} Q_{j' \to i, z_{j'}})$ and pass messages from CNs to VNs.

Step 4) VN processing: k ← k + 1 . Compute the VN-to-CN message in two steps. First compute the primitive message by

$$Q_{j\to i,\alpha^l} = L_{j.\alpha^L} + \sum_{i' \in Jj \setminus i} R_{j \leftarrow i',\alpha^l} .$$

Then we compute

$$Q_{j\to i}^{min} = min_{\alpha^L \in GF(q)} Q_{j\to i,\alpha^l}$$

After that, the VN-to-CN message is normalized

$$Q_{j\to i,\alpha^l} = Q_{j\to i,\alpha^l} - Q_{j\to i}^{min}$$

and passed from VN to CN. Also, we update the reliability of each code symbol by

$$Q_{j\to i,\alpha^l} = L_{j.\alpha^L} + \sum_{i' \in Jj} R_{j\leftarrow i,\alpha^l} .$$

The code symbol is determined as $max_{\alpha^l}(Q_{j,\alpha^l})$. Go to Step 1.

3. ARCHITECTURAL IMPLEMENTATION OF ALGORITHM

In this brief, the proposed scheme is designed a check node unit (CNU), which perform consists of two major parts: a sorter that sorts out the 1.5nm v-to-c messages with the smallest nonzero LLRs, and a path constructor that generates the c-to-v messages from the sorting results. In the following, the architectures for these two parts are presented.

3.1. Sorter Architecture

The right-shift cells [12] can realized with the sorter. Totally processing elements (PEs) are required. Each PE composes of a right-shift cell and a compare cell. The right-shift cell one is responsible for the date storage and right-shift operation. The right-shift cell executes the comparison and generates the control signals for the right-shift cell. Here, ri denotes the right-shift enable signal, pi presents the pre-sorted data, and is the result of comparison. Finally, the sorter will output nm intrinsic messages with the most significant magnitudes.

As a result, the decode processing algorithm of CNU is implemented, and two sub-blocks are needed. In total, the compare sub-block consists of | L | (dc −1) –input comparators and one (| L | −1) -input comparator, where | L | is the cardinality of the set L(c | av = _i). The inputs data can be read off from the LUT of the finite solution sequence generator, which simple control unit achieved with selecting. The output to the sorter provides the final result, which will pick up the nm most significant ones, realign them in decreasing order, and output them to the v-th VNU for further operations. The corresponding hardware structure of the CNU block is presented in Fig. 2(a) as above. Moreover, the proposed architecture can be further simplified as shown in Fig. 2(b), where only one (dc − 1) -input comparator is employed. For both structures, given all | L | finite solution sequences are generated by the module illustrated in Fig. 1. In Fig. 2(a), firstly all possible solutions are processed in parallel to select the required one. In the outer loop, indexed with i, only nm intrinsic messages with the most significant magnitudes are chosen by data sorter which is shown in Fig. 1. In Fig. 2(b), the switch is left open during the first step. All solution sequences are input in serial. A 2-input comparator with one delay element is employed to generate LLR value of $R_{cv}^{k,t}(\alpha^i)$. In the outer loop, the switch is closed and the sorter will pick up the nm most significant intrinsic messages.

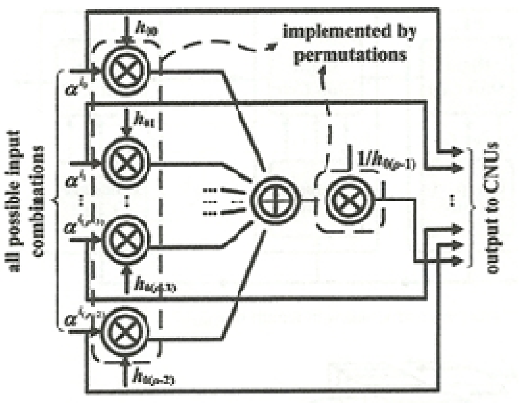

Fig. 1. Internal structure of generator for possible solution sequences

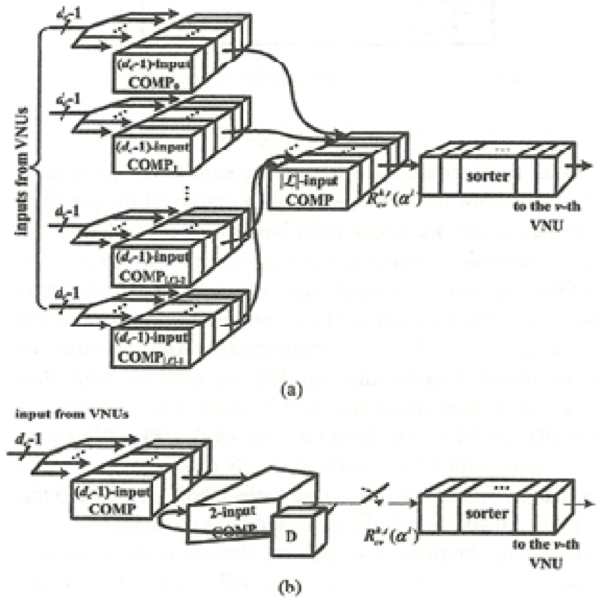

Fig. 2. Propoased CNU block architecture employing data sorter

The Sorter is defined in VHDL in the following:
entity Sorter is
port (dc – 1; input : in std logic vector;
input : in std logic vector(31downto0);
ri; pi; ci; nm : out std logic vector(31downto0);
architecture eq1 of Sorter is
component Right – Shifter the data of
inputs to detect the largest value • • • ;
– – Smallest values is right shift with the same
number of difference between the two input;
component Swap – –Swap the two input
from initial state to finalstate;);
• • • ;
component Compare the data of
inputs to detect the largest value • • • ;
– – largest values is sorter with the output $R_{cv}^{k,t}(\alpha^l)$

3.2. Path Constructor

The architecture of the path construction module is illustrated in Fig. 3. The 1.5nm sorted v-to-c messages are read from the RAM device one at a cycle, starting from the smallest nonzero LLR. During the path construction for c-to-v message Rm;n; e(i) , which is the index of the variable node that the message belongs to, is passed to the decoder to generate a binary vector with dc bits, in which only the e(i) th bit is “1” . The bit test block in Fig. 4 takes this vector and Pj . It outputs “1” when Pk (e(i)) □= 1 , as the enable signal to other blocks. If it outputs “0” , the path has been constructed before and thus should not be included. The f in the proposed algorithm can be computed by two GF adders. The multiplexer is added to enable the algorithm initialization. If the enable signal is “1”, a new path will be constructed. In addition, the new path vector and the finite symbol will be stored into a path track block and a symbol track block, respectively. In addition, the message LLR, the computed path symbol, and the path signal will be sent to select units (SU), which is used to compute Rm;n for each variable node. In an SU, the path symbol is added to the z (n) , which is the GF symbol of the zero-LLR node in column n . The computed symbol fn is copied to the GF comparator and the first-in-first-out (FIFO) buffer consisting of serially concatenated registers. In this way, each symbol in FIFO can be simultaneously compared with the newly computed fn to test if fn ∈ fLn . The GF comparator outputs "1" when fn equals none of the symbols in the FIFO. When the GF comparator and the path signal both output "1" ,h the LLR and GF symbol fn will be loaded into corresponding memory devices, and a new entry of Rm;n is computed. Otherwise, the message is skipped. The path construction will take 4 nm cycles in total, according to the proposed algorithm. Using the proposed CNU architecture, only 1.5 nm sorted v-to-c messages need to be stored for each check node. Compared with sorting dc nm intermediate messages for each of the check node process using the FB scheme, the memory requirement has been substantially reduced. Once the sorted messages are available, the path constructor can start to derive the c-to-v messages for the current check node. The total cycles needed to finish the check node processing are around (2dc + 5:5nm). The architecture can compute all dc c-to-v messages at a time. The total cycles for a check node process are much smaller compared with the original path construction architecture in [13].

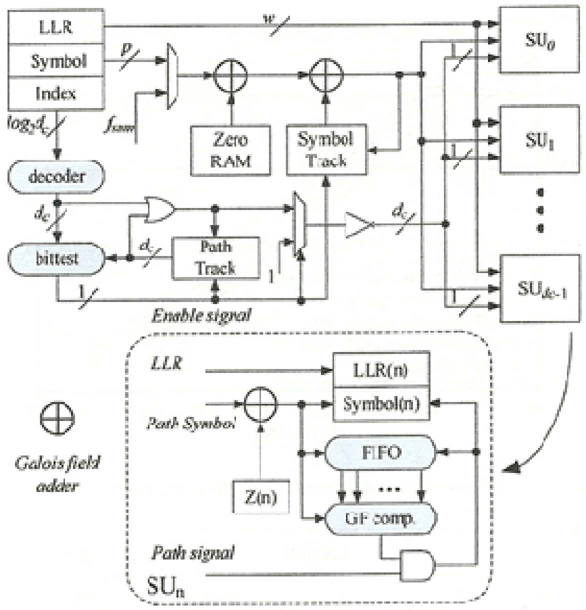

Fig. 3. Architecture of the path constructor

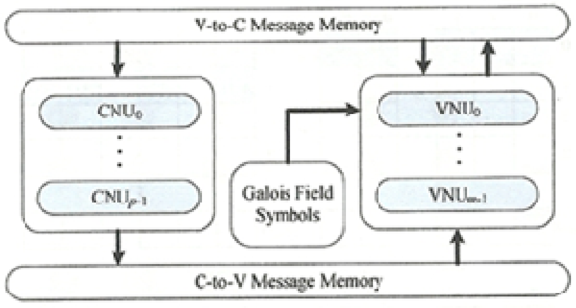

Fig. 4. Top-level architecture of the min-max NB-LDPC decoder

The path constructor is defined in VHDL in the following:

```
entity Path Constructor is
port Rm:n; e(i); V to CMessage : in std logic vector;(31 down to 0);
Clk : in std logic;
dc; C to V Message : out std logic vector(31downto0);
end Path Constructor;
architecture Decoder of Path Constructor is
signal e(i) : std logic vector(31downto0);
component Decoder the Decocder in Path Constructor
port e(i) : in std logic vector(31downto0);
port dc : out std logic vector(31downto0);
end component;
component GF adder the Decoder in Path Constructor • • • ;
component Multiplexer
– – signal initializerfor real multiplier
port P; fsum : instd logic vector(31downto0);
Clk : in std logic;
port (Zero RAM; Symbol Track : instd logic vector(31downto0);
data; dc : out std logic vector (31downto0);
end component;
• • • ;
component Select Unit the SU in Path Constructor
port LLR; Path Symbol; Path Signal : in std logic vector;
(31 down to 0);
port fn : out std logic vector (31downto0);
end component;
component GF Comparator the SU in Path Constructor • • • ;
component GF adder the SU in Path Constructor • • • ;
component FIFO the SU in Path Constructor • • • ;
end component;
• • • ;
```

3.3. Min-Max Decoder Architecture

Section In our design, layered decoding is adopted to reduce the memory requirement and to increase the decoding convergence speed. The H matrix is divided into several layers. The c-to-v messages derived from one layer are used right away to update the v-to-c messages of the next layer.

The CNU mainly consists of the proposed sorter and path constructor. The variable node unit (VNU) is an extension of that used in binary LDPC decoders and can directly be implemented with an adder, a subtractor, and a parallel sorter. Since only nm messages are kept for each vector, it is possible that, for a message in one vector, there is no message with the same GF element in the other vector. Taking this into account, the variable node elementary processing is mainly composed of two loops of nm cycles each to skim through all the values of the two input vectors. The details of the processing and VNU architecture can be referred to [14].

The top-level architecture of the min V max decoder is shown in Fig. 4. Here, we assume that there are p CNUs and m VNUs in the proposed decoder architecture.

The parameter p is determined by the row number of one layer, and m is equal to the number of columns with nonzero GF elements in one layer. For QCNB-LDPC codes, the H matrix can be divided into several submatrices of dimension $s \times s$, and each column of H has at most one nonzero entry in each layer. Hence, the parameters are p = s; m = s×dc. During the check node processing, all the rows in one layer are processed in parallel. During the decoding iterations, the p CNUs will read channel messages from the v-to-c message memory and fill the c-to-v message memory with updated c-to-v messages. The VNUs will compute updated v-to-c messages once check node processing has been done. Denote the v-to-c LLRs of layer l in the j th decoding iteration by $Q_l^{(j)}$. Represent the c-to-v LLRs computed from layer l in the j th iteration by $R_l^{(j)}$. It can be derived that $Q_{l+1}^{(j)} = (Q_l^{(j)} + R_l^{(j)}) - R_{l+1}^{(j-1)}$.

After v-to-c messages have been updated to the v-to-c message memory, a new round of check node processing will begin. In this brief, based on the proposed architecture, a decoder for a (620, 310), (6, 3) NB-LDPC code over GF(32) is designed. The base matrix size is 31×31 . The check node degree and variable node degree are 6 and 3, respectively. There are 31 CNUs and 186 VNUs as well as two message memories.

The Min-Max Decoder is defined in VHDL in the following:

```
entity Min – Max Decoder is
port (VtoC Message : in std logic vector(31downto0);
CtoV Message : out std logic vector(31downto0); );
end Min – Max Decoder;
architecture Decoder of Min – Max Decoder is
signal Data : std logic vector(31downto0);
signal nm : std logic vector(31downto0);
signal pi : std logic vector(31downto0);
signal mi : std logic vector(31downto0);
component H matix – –Parameterizable • • • ;
component GF Symbol – –Parameterizable • • • ;
component CNU – –Sorter & Path constructor • • • ;
component VNU – –Adder & Subtractor & Parallel sorter • • • ;
```

3. PERFORMANCE SIMULATION RESULT

The study of how changes in performance depend on changes in parameter mode values is known as sensitivity analysis. We can vary some parameter's value a little, and see its influence degree to the model's performance, for example, the throughput or response time. Throughput is an action-related metric showing the rate at which an action is performed at steady-state. In other words, the throughput represents the average number of the activities completed by the system during one unit time.

From Figure 5, it can be observed that the impact of the number of devices on the throughput of transmit is more sensitive than the FB-Min-Max, nm = 32, floating and nm = 16 , floating modulation for nm = 16 . If we could make some efforts to optimize the cache, and raise the Min-Max, nm = 32 , floating modulation form 0.6 to 0.85 or even more high value, the throughput of transmit could greatly improved. In Fig. 6, some simulation results of the frame error rate of the minVmax algorithm for an (620, 310) NB-LDPC code over GF(32) using layered decoding are shown. Here, we use w = 5 bits to represent each LLR, and nm = 16 most reliable entries are kept in each message. The maximum number of iterations is set to 15. It can be observed that under 5-bit quantization, the proposed trellis path scheme has the same performance as the FB scheme and only has about 0.02-dB performance loss compared with the floating point minVmax decoding. For the purpose of comparison, the FB minVmax algorithm with nm = 32 and the fast fourier transform (FFT)-BP algorithm are also included.

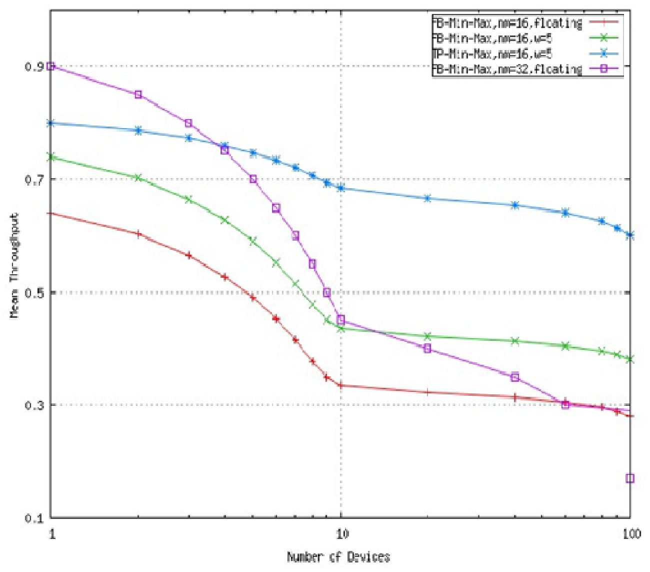

Fig. 5. Throughput versus Number of Devices

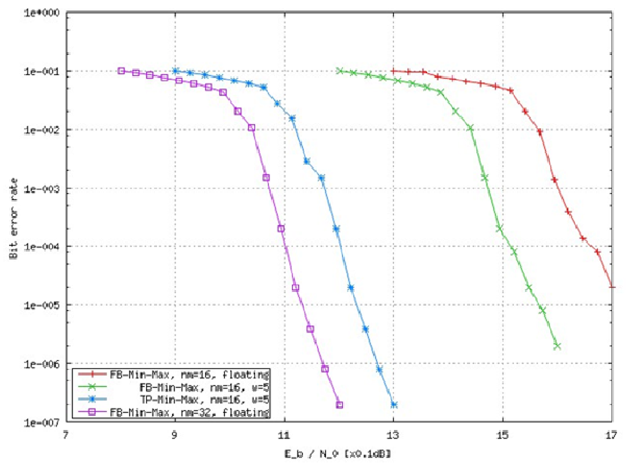

Fig. 6. Performance simulation for a (620,310) NB-LDPC code

4. Conclusions

Channel coding plays key role in providing a reliable communication method that can overcome signal degradation in practical channels. In this paper new Non-binary LDPC codes have been developed and implement the newly designed codes on FPGA platform. Prototype architecture of the LDPC codes has been implemented by writing Hardware Description Language (VHDL) code and targeted to VLSI chip. Simulation results demonstrate that the proposed Non-binary QC-LDPC codes achieve a 0.8 dB coding gain over randomly constructed codes and perform 1 dB from the Shannon-limit at a BER of 10–6 with a code rate of 0.89 for block length of 620.

REFERENCES

[1] Charles H. L., Error-Control Convolutional Coding, 1st edition, Artech House, Inc. Norwood, MA, USA, 1997.

[2] Berrou C., Glavieux A. and Thitimajshima P., “Near Shannon limit error correcting coding and decoding: turbo-codes”, IEEE ICC’93, Geneva, Switzerland, pp. 1064-1070, 1993.

[3] Gallager R. G., Low-Density Parity-Check Code. Cambridge, MA: MIT Press, 1963.

[4] Mackay D. and Neal R., “Near Shannon Limit Performance of Low Density Parity Check Codes”, Electronic Letters, Vol. 32, No.18, Pages 1645-1646, 1996.

[5] Huo G. and Alouini S., “Another Look at the BER Performance of FFH/BFSK with Product Combining Over Partial-Band Jammed Rayleigh-Fading Channels”, IEEE Transaction on Vehicular Technology, vol. 50(5),Pages 1203-1215, 2001.

[6] Zhang H. and Moura J. M., “The design of structured regular lpdc codes with large girth”, GLOBECOM, Pages 4022V4027, 2003.

[7] Ohhashi A. and Ohtsuki T., “Performance analysis of BP-based algorithms for irregular low-density parity-check codes on fast Rayleigh fading channel”, IEEE 60th Vehicular Technology Conference, 2004. VTC2004-Fall. vol. 4,pp.2530-2534, 2004.

[8] Farid Ghani, Abid Yahya and Abdul Kader, ”New Qc-LDPC Codes Implementation on FPGA platform in Rayleigh Fading Environment”, IEEE Symposium on Computer & Information, 2011.

[9] Savin V., “Min-Max decoding for non-binary LDPC codes”, in Procedure IEEE International Symposium on Information Theory, ISIT, Pages 960 - 964, 2008.

[10] Lin Jun, Sha Jin, Wang Zhongfeng, Li Li, “An Efficient VLSI Architecture for Nonbinary LDPC Decoders”, IEEE Transactions on Circuits and Systems II: Express Briefs, Volume: 57 , Issue: 1, Pages 51 - 55, Jan. 2010.

[11] Lin Jun, Sha Jin, Li Li, “Efficient decoder design for nonbinary quasi-cyclic LDPC codes”, IEEE Transition on Circuits System I, Regular Papers, Volume 57, No. 5, Pages 1071-1082, May 2010.

[12] X. Zhang and F. Cai, ``Efficient partial -parallel decoder architecture for Quasi-cyclic nonbinary LDPC codes”, IEEE Transactions on Circuits and Systems I, Regular Papers, Volume 58, No.2, Pages 402-414, Feb 2011.

[13] Zhang X. and Cai F., “Reduced-complexity decoder architecture for nonbinary LDPC codes”, IEEE Transition of Very Large Scale Integr. (VLSI) System, volumn 19, no. 7, Pages 1229-1238, July 2011.

[14] He Kai, Sha Jin, Wang, Zhongfeng, “Nonbinary LDPC Code Decoder Architecture With Efficient Check Node Processing”, IEEE Transactions on Circuits and Systems II: ExpressLee, S.hyun. & Kim Mi Na, (2008) “This is my paper”, ABC *Transactions on ECE*, Vol. 10, No. 5, pp120-122.

Tony Tsang

received the BEng degree in Electronics & Electrical Engineering with First Class Honours in U.K., in 1992. He received the Ph.D from the La Trobe University (Australia) in 2000. He was awarded the La Trobe University Post graduation Scholarship in 1998. He is a Lecturer at the Hong Kong Polytechnic University. Prior to joining the Hong Kong Polytechnic University, Dr. Tsang earned several years of teaching and researching experience in the Department of Computer Scienc and Computer Engineering, La Trobe University. His research interests include mobile computing, networking, protocol engineering and formal methods. Dr. Tsang is a member of the ACM and the IEEE.