Flexible FPGA implementation of Min-Sum Decoding Algorithm for regular LDPC codes

Ahmed M. Sadek
Faculty of Computers & Information
Fayoum University
Fayoum, Egypt
ams13@fayoum.edu.eg

Aziza I. Hussein
Computer & Systems Eng. Dept
Minia University, Minia, Egypt
Electrical & Computer Eng. Dept
Effat Univesity, Jeddah KSA
azibrahim@effatuniversity.edu.sa

***Abstract*— Because of their excellent error correction performance, Low-Density Parity Check Codes (LDPC) have become the most widely used technique for forward error correction in almost all modern communications applications. This paper introduces an FPGA implementation of a partial parallel, flexible LDPC decoder based on the Min-Sum decoding algorithm. The suggested architecture uses a combination of unicast and multicast communications between its processing elements in order to reduce the intercommunication overhead and at the same time, keep the processing elements simple.**

The use of the less complex Min-Sum algorithm on the suggested architecture produces a very compact and resource efficient design which allows the instantiation of many processing elements to deliver high processing rate

To allow flexibility to support different codes, a non-blocking interconnection network is used to pass messages between processing elements. It is a modified version of the multi stage network called Arbitrary Size Benes.

The decoder architecture was implemented on Virtex5 FPGA using VHDL in Xilinx ISE environment. Results show efficient resources utilization compared to other implementations. The decoder achieves up to 3.49 Gbps per iteration for a code length of 2640 with BER of 10-5 at 4.4dB.

Keywords*— FEC; LDPC; iterative decoding; minimum sum algorithm; partial parallel decoder; FPGA implementation; permutation networks Introduction

I. INTRODUCTION

Low Density Parity check codes were introduced more than 50 years ago by Gallager [1]. Today they are considered the most attractive forward error correction codes because of their excellent performance approaching the channel capacity defined by Shannon [2]. They are found today in many applications and standards like IEEE 802.11n (Wi-Fi), IEEE 802.16e (WiMax), DVB-S2 and the wired communication standard IEEE 802.3 (10 GBase-T) [3]-[5].

LDPC codes are block codes where a *k-bits* message is replaced with *n-bits* codeword introducing *n-k* redundant bits. The constraints equations that validate the codewords are grouped in a matrix called parity check matrix. The parity check matrix is represented graphically with a Tanner graph[4]. A Tanner graph is a bipartite graph consisting of two sets of nodes named variable nodes (VN) and check nodes (CN).

LDPC codes are decoded iteratively with the passing of messages between VNs and CNs according to the connections of the Tanner graph.

In partial parallel decoders [7-12], some of the VNs and CNs are implemented along with memory units and that provides a good tradeoff between complexity and performance compared to full parallel decoders that implement every VN and CN with a fixed interconnection. Full parallel decoders suffer the most complexity and resource usage [13-19].

Various message passing decoding algorithms are available to decode LDPC. The sum product achieves the best bit error rate BER performance at the expense of complex hardware architectures [7],[11]. Other algorithms like the min-sum algorithm reduce the hardware complexity by approximating the complex operations of the sum-product algorithm at the cost of acceptable BER performance degradation[12, 14, 20-24].

The interconnection mechanism is the heart of flexibility of the partial parallel LDPC decoder. It is necessary to use a non-blocking interconnection mechanism between the multiple processing elements PEs of the decoder that work in parallel. Multiple processors compete for memory access and that results in memory collisions on a blocking interconnection. Some methods were suggested to reduce the collision effects instead of queuing the PEs to memory accesses [11].

This paper suggests an FPGA implementation of a partial parallel LDPC decoder where the PEs are interconnected with a modified version of the Arbitrary Size Benes network [32] to solve the memory contention issue and preserve the flexibility of the decoder to support different LDPC codes. The architecture implements the reduced complexity min-sum algorithm in order to produce a compact design that allows the implementation of an increased number of PEs.

Results show a very good resource utilization and a very good BER performance with an excellent processing capacity

The rest of this paper is organized as follows: Section II reviews the min-sum LDPC decoding algorithm. Section III introduces the suggested decoder architecture. Section IV discusses the implementation and simulation results.

978-1-5090-3266-2/16/$31.00 ©2016 IEEE

II. LDPC DECODING ALGORITHM

A. Representation of LDPC codes

An LDPC parity check matrix is represented graphically by a Tanner graph. A parity-check matrix is called (w_c, w_r)-$regular$ if each code bit is contained in a fixed number (w_c or column weight) of parity check equations and each parity-check equation contains a fixed number (w_r or row weight) of code bits [25].

Equation (1) shows a regular parity-check matrix with $w_c = 2$ and $w_r = 3$, and *Fig. 1* shows its Tanner graph representation.

$$H = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix} \quad (1)$$

B. Min-sum decoding algorithm

Decoding of LDPC codes is performed iteratively using the Tanner graph. Two half iterations are repeated until a codeword is corrected or until a maximum number of iterations is reached. VNs and CNs exchange probabilities representing a level of belief about the value of each codeword bit and it is often to represent those probabilities as log likelihood ratio (LLR).

In the first half iteration, each check node *j* calculates an extrinsic message to every variable node *i* connected by an edge in the Tanner graph. That message is called $E_{i,j}$ and it represents the LLR of the probability that bit *i* causes parity check equation *j* to be satisfied.

$$E_{j,i} = \prod_{\alpha \in V[j], \alpha \neq i} sgn(M_{j,\alpha}) \min_{\alpha \in V[j], \alpha \neq i} |M_{j,\alpha}| \quad (2)$$

Where *V[j]* is the set of VNs connected to check node *j*, and $M_{j,\alpha}$ are their LLR values.

Each VN has an initial LLR value (a priori probability) and receives LLRs from every connected CN. The total LLR of each codeword bit is the sum of these LLRs:

$$L_i = r_i + \sum_{\alpha \in C[j]} E_{\alpha,i} \quad (3)$$

Where *C[j]* is the set of CNs connected to variable node *i* and r_i is its initial LLR.

The messages sent from VNs to CNs are the full LLR without the component $E_{j,I}$, which was just received from check node *j*, and hence, the message sent from each VN to all connected CNs is:

$$M_{j,i} = \sum_{\alpha \in C[j], \alpha \neq j} E_{\alpha,j} + r_i \quad (4)$$

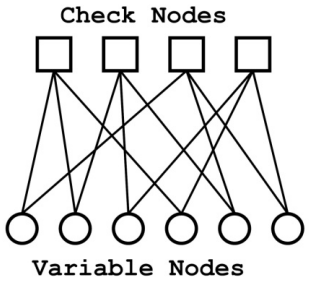

Fig. 1 Tanner graph representation

The CN processing element calculates (2) which is an approximation of the original sum-product algorithm that uses a much more complex non-linear function.

Obviously, the hardware implementation of the CN and VN nodes based on the min-sum algorithm is simple. That simplicity allows the instantiation of many PEs working in parallel to produce higher bitrates at the output of the decoder.

Those approximations do not come without cost. The sum-product algorithm offers better BER performance compared to the min-sum. Some variations were introduced to narrow the gap between the performances of the two algorithms. Normalized min-sum multiplies a normalizing factor (ν) where $0 < \nu < 1$ in the CN to VN half iteration achieving a better error performance[24]. Adaptive offset min-sum adaptively adjusts the normalization factor according to the state of check nodes in each iteration[26]. Probabilistic min-sum calculates a probabilistic minimum value when calculating the min term in (2) to reduce the complexity of the CN processor [19].

III. LDPC DECODER IMPLEMENTATIONS

Implementation of partial parallel decoders uses memory units along with the CN and VN processing elements. Typically, each CN calculates (2) and generates multiple messages that are sent as unicasts to multiple VNs, and every VN calculates (4) and also generates multiple unicast messages to be sent to all connected CNs. Using unicasts is the most common implementation method as in [7-10], [13-19].

Another implementation was introduced in [11] and suggested the following: a modified variable node processor named (MVNP) that delivers one output message calculated as defined by (3), and in the same way, a modified check node processor named (MCNP) that generates a single output message calculated according to a modified version of (2). That algorithm was named *Low Traffic Belief Propagation (LTBP)* with the goal of reducing the number of messages to be exchanged between VN and CN processors. That architecture needs to regenerate the older messages (from the previous iteration), $E_{j,i}$ and $M_{j,i}$ locally at each node processor so that it can correctly calculate for the next iteration.

The implementation introduced in [27] suggested the combination of unicast and multicast: VN processors which calculate (4) use multicasts as the operation to be performed is a simple addition. Using multicast will not pose any increase in complexity but will reduce the number of messages greatly.

978-1-5090-3266-2/16/$31.00 ©2016 IEEE

On the other hand, CN processors which calculate (2) will work as usual with unicasts as their operation is already complex.

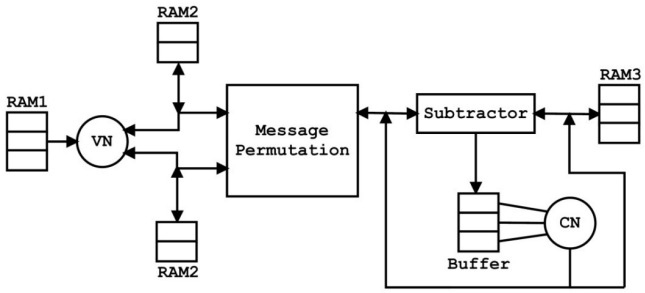

Fig. 2 Architecture of LDPC decoder

A. Proposed LDPC decoder

The decoder proposed here is an extension to the one we presented in [27] with a solution to a neglected memory contention problem. Fig. 2 shows an overall block diagram of the suggested architecture which supports multicasting at the VN processor and unicasting at the CN processor.

Each VN processor is connected to a set of RAM units. One unit holds the initial LLRs to be processed by this processor (RAM1). Other units hold the values of messages communicated over the connected edges to this processor; two (RAM2) units for two connected edges. The capacity of these RAM units determines the maximum code length that can be supported by the decoder.

On the other hand, each CN processor also has a set of RAM units. One unit holds the previous iteration values of the edges connected to this processor (RAM3). Another unit works as a buffer that holds the difference between the multicast message calculated by the VN processors and the previous iteration value. The capacity of these RAM units is related to the row weight of the code or in other words, the number of edges connected to the CN processor.

B. Message permutation and Memory contention problem

In the CN half iteration, all CN processors need to load and store values of their connected edges from and to the RAM units through the permutation network. Many types of permutation networks are available and discussed in [28]. Most implementations use Benes network for message permutation as it offers a good compromise between scalability and flexibility. The Benes network is a non-blocking network that consists of $(2Log_2N - 1)$ number of stages, where N is the number of inputs or outputs as it is a symmetric network[29].

The architecture we presented in [27] used the permutation network shown in Fig. 3. Each VN processor has three RAM units for 3 Tanner edges so that it can support regular codes with a column weight of 3. Multiplexers were used on the VN side to select one of the connected RAM units. With that arrangement, only one of the RAM units connected to a multiplexer can be accessed at any given time. When two CN processors try to simultaneously access contents through the same multiplexer, one of them has to stall to the next cycle and that means wasted time and reduced performance.

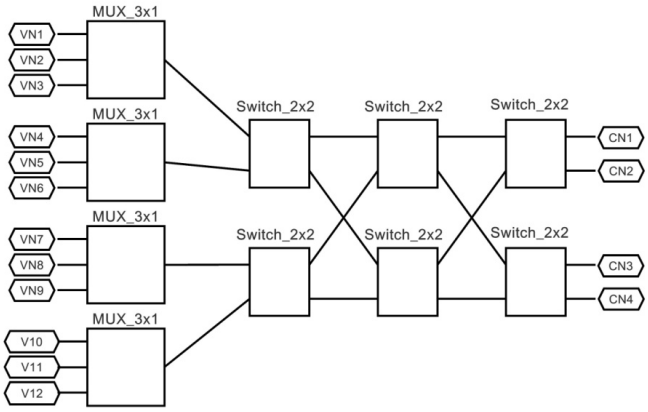

Fig. 3 12x4 permutation network[27]

C. Arbitrary Size Benes

Ordinary Benes network is a symmetric multi-stage network [29]. The r-dimensional Benes network with 2^r inputs/outputs has $2r - 1$ levels of switches, with 2^{r-1} switches in each level. The Benes network topology must have the number of inputs or outputs to be a power of 2. That restriction is not suitable for many situations including the suggested architecture.

The arbitrary size (AS) Benes network provides a solution to the restricted number of inputs/outputs by suggesting a 3×3 permutation network as shown in fig.4. A simple wire is considered a simple network that realizes a 1×1 permutation [30].

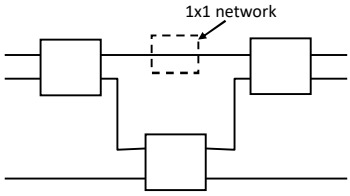

Fig. 4 3×3permutation network

978-1-5090-3266-2/16/$31.00 ©2016 IEEE

An AS-Benes of size n is constructed recursively from an AS-Benes of size $\left\lfloor \frac{n}{2} \right\rfloor$ and an AS-Benes of size $\left\lceil \frac{n}{2} \right\rceil$.

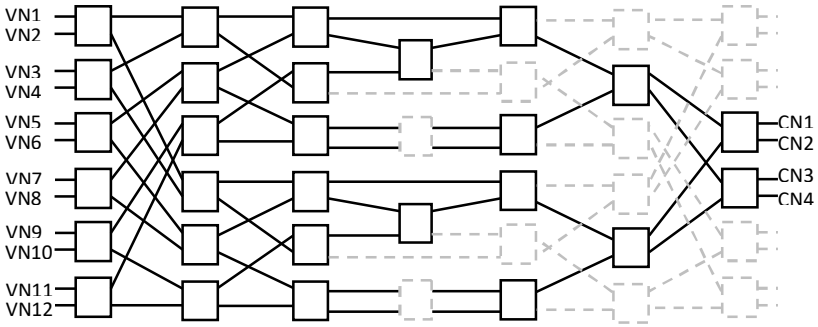

Fig. 5 12×4 Modified AS-Benes

When n is even, the construction is similar to that of the Benes network where the n inputs are connected to n switches and each switch is connected to two AS-Benes networks of size n. Similarly, the n outputs are connected to n switches and each switch is connected to the two n AS-Benes networks [32].

To construct an AS-Benes of an odd size, the first $n-1$ inputs are connected to $\left\lfloor \frac{n}{2} \right\rfloor$ switches and each switch is connected to the AS-Benes of size $\left\lfloor \frac{n}{2} \right\rfloor$ and the AS-Benes of size $\left\lceil \frac{n}{2} \right\rceil$. The same procedure applies for the first $n-1$ outputs. The last input and the last output are connected directly to the $\left\lceil \frac{n}{2} \right\rceil$ AS-Benes.

In the suggested decoder architecture, the number of VN and CN processors is not the same. Therefore, the AS-Benes is modified to suit the architecture requirements using the following steps:

1) building a network with $n = Number\ of\ VNs$.

2) align the CN processors on the center of the available switches' terminals

3) eliminate the unconnected switches in each stage

Figure 5 shows an example of a modified AS-Benes connecting 12 VNs to 4 CNs

To configure the Modified AS-Benes network, a C++ program was developed to model the network and generate the state of each switch. It accepts a parity check matrix file in the same format used for the matrices available on MacKay's website [31].

The configuration of the permutation network is stored in memory units. The decoder's control unit is responsible for invoking the permutation memory to setup the permutation network to connect the CN processors to the appropriate RAM unit. The contents of the permutation memory depend on the parity check matrix of the code. Hence, the decoder can support more than one code by loading the proper permutation configuration to the memory of the network.

D. Message quantization

An important aspect that has a direct influence on both the performance and the complexity of the decoder is the message quantization scheme. The methods introduced in [16] and [32] are similar in suggesting the use of two different quantization schemes for the a priori LLR of each variable node and the extrinsic LLRs between variable and check nodes. Either it is called hybrid or dual quantization, it uses the representation $(n_1, m_1) - (n_2, m_2)$ to indicate that the initial LLR composed of n_1 bits where m_1 of them are fraction bits, and the messages to and from the check nodes are composed of n_2 bits where m_2 of them are fraction bits.

Different hybrid quantization schemes were simulated on Matlab to evaluate the BER performance of each scheme and to help in selecting the one that will be used in the suggested decoder.

The simulation results are shown in fig.6 for an LDPC code with n=2640 bits with BPSK modulation in AWGN channel. In order to get a BER of 10^{-5}, the hybrid *(3,1)-(2,1)* needs additional 0.8 dB of SNR compared to the hybrid *(4,1)-(2,1)*.

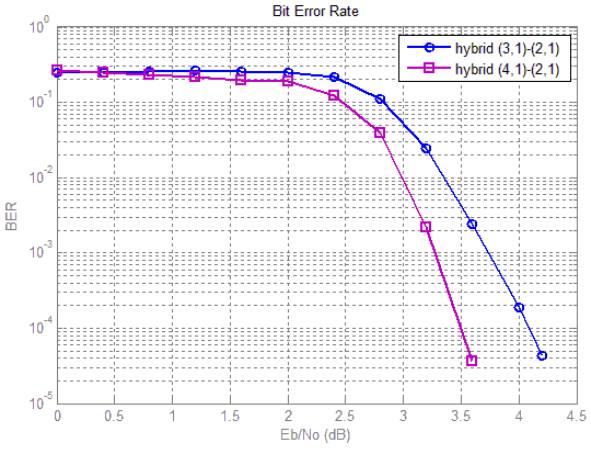

Fig. 6 BER performance of various quantization schemes

We select the hybrid *(3,1)-(2,1)* because a single bit reduction on the VN side of the decoder will reduce the design

978-1-5090-3266-2/16/$31.00 ©2016 IEEE

complexity and the resources requirements, especially when designing a decoder with a large number of processing elements.

For the selected quantization scheme, the initial LLRs range from (binary 101 to 011) or (decimal -1.5 to +1.5), while the messages to and from the CN processors range from (binary 11 to 01) or (decimal -0.5 to +0.5). The 3-bit representation of (-2) and the 2-bit representation of (-1) are omitted to make the range symmetrical and unbiased toward negative values.

Using two bits for the messages to and from the check nodes will greatly lower the implementation complexity and the resources requirements for the permutation network and the check node processors.

E. Check Node Processor

The core of the check node processor is supposed to calculate (2) which is quite simple. The sign multiplication of the inputs is computed with an XOR gate on the most significant bit (MSB) of the inputs. If the number of inputs with MSB = '1' is odd, the output of the XOR would be '1' indicating a negative result. Similarly, if the number of inputs with MSB = '0' is even, the XOR will output '0' indicating a positive result.

The minimum of the absolute values of the inputs is easily computed with an AND gate on the least significant bit (LSB) of the inputs. The AND gate produces the minimum among inputs composing of a single bit.

Figure7 shows a block diagram of the CN processor core.

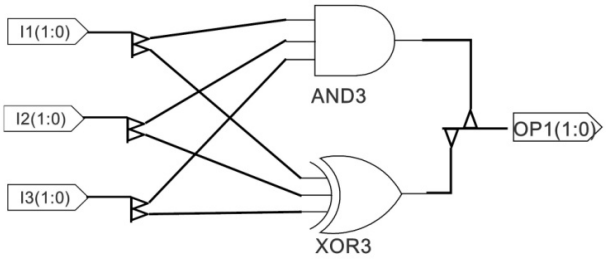

Fig. 7 Block diagram of CN Core

The overall CN processor is shown in Fig. 8 and it is based on the one represented in [27]. The CN processor works on the correct edges' values stored in its buffer and calculates (2). The result is latched so that it can be written to the local RAM unit, and to the edges RAM units through the permutation network.

IV. RESULTS

To verify the throughput performance of the suggested architecture, it was implemented using Xilinx ISE 14.2 with a combination of VHDL and schematics tools on Xilinx Virtex5 (XC5VLX50T) development board. The performance analysis of the architecture implemented in [27] indicated that the number of clock cycles is dominated by the CN half iteration. This is because the CN half iteration is performed using unicast messages, not like the VN.

To boost the performance of the decoder, an asymmetric implementation was considered where the number of the CN processors is twice the VN processors.

As the CN and VN processing elements for the min-sum algorithm are very simple to implement. It encouraged the instantiation of many processing elements. Therefore, we implemented the decoder with 64 VN processors and 128 CN processors.

The interconnection mechanism is required to connect 192 RAM units of 64 VN processors to 128 CN processors. The modified AS-Benes satisfies that requirement with 15 stages of elementary switches in the same way described in section III.C

Table (1) shows the device utilization report for the implementation

Table 1 Device utilization of 64VN/ 128CN PEs

Logic Utilization	Used	Available
Number of slice registers	720	28800
Number of slice LUTs	913	28800
Number of fully used LUT-FF pairs	336	1297
Max clock frequency	319 MHz	

To demonstrate the efficiency of the suggested design, the device utilization of the decoder presented in [17] is shown in table (2).

Table 2 Device utilization of decoder presented in [11].

Logic Utilization	Used
Number of slice registers	23352
Number of slice LUTs	95188

It was predicted that the resource usage of the full parallel decoder implemented with 1296 VNs in [17] to be greater than the suggested decoder. But considering the ratio between the number of VNs in the two decoders (1296/64), the suggested one is very efficient.

To measure the processing rate of the suggested decoder, two special counters were implemented inside the control unit. The first one counts the clock cycles of the VN half iteration, and the second one counts the clock cycles of the CN half iteration.

The suggested decoder achieves 3.4 Gbps at 319 MHz per iteration for the LDPC code of length (2640-1320)

The full parallel architecture in [17] processes 1.77 Gbps after 20 iterations with 1296 VNs. That throughput is about 10 times the throughput of the suggested decoder despite that the number of VNs in [11] is about 20 times the number of VNs in the suggested decoder

978-1-5090-3266-2/16/$31.00 ©2016 IEEE

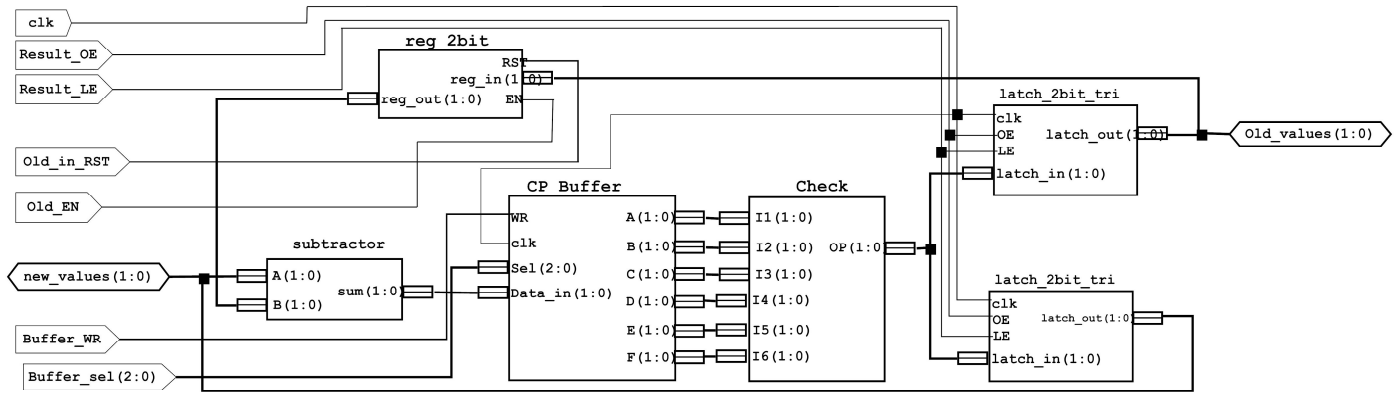

Fig. 8 Block diagram of a CN processor

V. CONCLUSION

As modern communications technologies advance, the need for high bandwidth communication increases. Forward error correction is one of the primary processes of any communication system and its performance has great impact over the whole communication system. LDPC codes are becoming more and more attractive for modern communication technologies as it can deliver the demands of high bitrates. Decoding of LDPC code was investigated in many directions including various algorithm implementations and variety of processing architectures.

The paper introduces a partially parallel decoder architecture implementing the min-sum algorithm. The architecture is described using VHDL to implement the suggested algorithm along with a new permutation network. The implementation results showed that the decoder was able to deliver high bitrates with linear scalability to support even higher bitrates with more processing elements. The min-sum decoder shows a low degradation in BER performance compared to the sum-product decoder but it offers much more processing rate that makes it suitable for situations with better communications conditions such as low to medium range communication

REFERENCES

[1] R. Gallager, “Low-density parity-check codes,” Information Theory, IRE Transactions on, vol. 8, no. 1, pp. 21 –28, 1962

[2] S.-Y. Chung, J. Forney, G.D., T. Richardson, and R. Urbanke, “On the design of low-density parity-check codes within 0.0045 db of the Shannon limit," Communications Letters, IEEE, vol. 5, no. 2, pp. 58-60, Feb 2001.

[3] J. Lorincz and D. Begusic, “Physical layer analysis of emerging IEEE 802.11n WLAN standard,” in Advanced Communication Technology 2006. ICACT 2006. The 8th International Conference, vol. 1, pp. 6 pp. –194.

[4] M. Khan and S. Ghauri, “The WiMAX 802.16e physical layer model,”in Wireless, Mobile and Multimedia Networks, 2008. IET International Conference on, 2008, pp. 117 –120.

[5] A. Morello and V. Mignone, “DVB-S2: The second generation standard for satellite broad-band services,” Proceedings of the IEEE, vol. 94, no. 1, pp. 210 –227, 2006

[6] D.J.C. MacKay and R.M. Neal, "Near Shannon limit performance of low density parity check codes," Electronics Letters, vol. 33, no. 6, pp. 457-458, 13 March 1997

[7] Hayes, "FPGA implementation of a Flexible LDPC decoder",PSc thesis, University of Newcastle, Australia 2008

[8] T. Zhang; K. K. Parhi, An FPGA implementation of (3,6)-regular low-density parity-check code decoder ,Eurasip Journal on Applied Signal Processing.?2003(6):530-542.

[9] Jong-Yeol and H.-J. Ryu, A 1-gb/s flexible LDPC decoder supporting multiple code rates and block lengths," Consumer Electronics, IEEE Transactions on, vol. 54, pp. 417{424, May 2008.

[10] S.M. Aziz and M.D. Pham, "Implementation of Low Density Parity Check Decoders using a New High Level Design Methodology," Journal of Computers, Academy Publisher, vol. 5, no. 1, pp. 81-90, January 2010.

[11] Guido Masera, Federico Quaglio, and Fabrizio Vacca, “Implementation of a Flexible LDPC Decoder” , IEEE transactions on circuits and systems—ii: express briefs, vol. 54, no. 6, June 2007

[12] Condo and G. Masera, A Flexible LDPC code decoder with a Network on Chip as underlying interconnect architecture. In Proceedings of CoRR. 2011

[13] A. J. Blanksby and C. J. Howland,“A 690-mW 1-Gb/s 1024-b, rate ½ low-density parity-check code decoder ” IEEE J. Solid-State Circuits, vol. 37, no. 3, pp. 404–412, Mar. 2002

[14] V. A. Chandrasetty, S. M. Aziz, "FPGA Implementation of a LDPC Decoder using a Reduced Complexity Message Passing Algorithm" Journal of Networks, Vol 6, No 1 (2011), 36-45, Jan 2011

[15] V. A. Chandrasetty, S. M.Aziz, "An area efficient LDPC decoder using a reduced complexity min-sum algorithm

978-1-5090-3266-2/16/$31.00 ©2016 IEEE

",Integration, the VLSI Journal, Volume 45 Issue 2, pp. 141-148, March 2012.
[16] Abu-Surra, S.; Pisek, E.; Henige, T.; Rajagopal, S., "Low-power dual quantization-domain decoding for LDPC codes," in Global Communications Conference (GLOBECOM), 2014 IEEE , vol., no., pp.3151-3156, 8-12 Dec. 2014
[17] Nguyen-Ly, T.; Khoa Le; Ghaffari, F.; Amaricai, A.; Boncalo, O.; Savin, V.; Declercq, D., "FPGA design of high throughput LDPC decoder based on imprecise Offset Min-Sum decoding," in New Circuits and Systems Conference (NEWCAS), 2015 IEEE 13th International , vol., no., pp.1-4, 7-10 June 2015
[18] Kumawat, S.; Shrestha, R.; Daga, N.; Paily, R., "High-Throughput LDPC-Decoder Architecture Using Efficient Comparison Techniques & Dynamic Multi-Frame Processing Schedule," in Circuits and Systems I: Regular Papers, IEEE Transactions on , vol.62, no.5, pp.1421-1430, May 2015
[19] Chung-Chao Cheng; Jeng-Da Yang; Huang-Chang Lee; Chia-Hsiang Yang; Yeong-Luh Ueng, "A Fully Parallel LDPC Decoder Architecture Using Probabilistic Min-Sum Algorithm for High-Throughput Applications," in Circuits and Systems I: Regular Papers, IEEE Transactions on , vol.61, no.9, pp.2738-2746, Sept. 2014
[20] J.H. Han and M.H. Sunwoo, "Simplified sum-product algorithm using piecewise linear function approximation for low complexity LDPC decoding," Proceedings of the 3rd International Conference on Ubiquitous Information Management and Communication, Suwon, Korea, pp. 302-308, 2009.
[21] S. Papaharalabos, et al., "Modified sum-product algorithms for decoding low-density parity-check codes," IET Communications, vol. 1, no. 3, pp. 294-300, June 2007.
[22] J. Chen, A. Dholakia, E. Eleftheriou, M.P.C. Fossorier, and X. Hu, "Reduced-complexity decoding of LDPC codes," IEEE Trans. Comms., vol. 53, no. 8, pp. 1288-1299, Aug. 2005.
[23] J. Zhao, F. Zarkeshvari, and A.H. Banihashemi, "On implementation of min-sum algorithm and its modifications for decoding low-density Parity-check (LDPC) codes," IEEE Trans. Comms., vol. 53, no. 4, pp. 549-554, 2005.
[24] Mohammad Rakibul Islam, Dewan Siam Shafiullah, Muhammad Mostafa Amir Faisal, Imran Rahman, "Optimized Min-Sum Decoding Algorithm for Low Density Parity Check Codes",(IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 2, No. 12, 2011
[25] S.J. Johnson, "Introducing Low-density Parity-check codes", Published Internal Technical Report , Department of Electrical and Computer Engineering, University of Newcastle, Australia
[26] Xiaofu Wu; Yue Song; Ming Jiang; Chunming Zhao, "Adaptive-Normalized/Offset Min-Sum Algorithm," in Communications Letters, IEEE , vol.14, no.7, pp.667-669, July 2010
[27] Sadek, A.M., Hussein, A.I., "C1. Flexible LDPC decoder architecture for (3–6) regular codes," in Radio Science Conference (NRSC), 2015 32nd National , vol., no., pp.100-107, 24-26 March 2015
[28] Mark Woh, "ARCHITECTURE AND ANALYSIS FOR NEXT GENERATION MOBILE SIGNAL PROCESSING", Ph.D. thesis, dep. of Electrical Engineering, The University of Michigan, 2011
[29] V. E. Benes, \Permutation groups, complexes and rearrangeable connecting network, "Bell System Technical Journal, vol. 43, no. 4, pp. 1619{1640, 1964.
[30] Chihming Chang and Rami Melhem, "Arbitrary Size Benes Networks", Parallel Processing Letters. 07, 279 (1997)
[31] Encyclopedia of Sparse Graph Codes: http://www.inference.phy.cam.ac.uk/mackay/codes/data.html
[32] Balatsoukas-Stimming, A.; Dollas, A., "FPGA-based design and implementation of a multi-GBPS LDPC decoder," in Field Programmable Logic and Applications (FPL), 2012 22nd International Conference on , vol., no., pp.262-269, 29-31 Aug. 2012

978-1-5090-3266-2/16/$31.00 ©2016 IEEE