Reduced-Latency and Area-Efficient Architecture for FPGA-Based Stochastic LDPC Decoders

Ghania Zerari, Abderrezak Guessoum
"LATSI" Laboratory, Department of Electronics,
University of Blida,
Blida, Algeria

***Abstract*—This paper introduces a new field programmable gate array (FPGA) based stochastic low-density parity-check (LDPC) decoding process, to implement fully parallel LDPC-decoders. The proposed technique is designed to optimize the FPGA logic utilisation and to decrease the decoding latency. In order to reduce the complexity, the variable node (VN) output saturated-counter is removed and each VN internal memory is mapped only in one slice distributed RAM. Furthermore, an efficient VN initialization, using the channel input probability, is performed to improve the decoder convergence, without requiring additional resources. The Xilinx FPGA implementation shows that the proposed decoding approach reaches high performance along with reduction of logic utilisation, even for short codes. As a result, for a (200, 100) regular codes, a 57% reduction of the average decoding cycles is attained with an important bit error rate improvement, at Eb/N0 = 5.5dB. Additionally, a significant hardware reduction is achieved.**

Keywords—Stochastic decoding; low-density parity-check (LDPC) decoder; field programmable gate array (FPGA)

I. Introduction

The need for increasing the throughput of modern communication systems capacity, for optical and wireless networks, requires high performance error correcting code. In 1962, Gallager presented the first version of low-density parity-check (LDPC) codes [1], offering an excellent process for repairing transmission errors, added by the channel effects. The close Shannon capacity decoding performances, of the LDPC codes [2], justify their exploitation by various digital communication standards. WiMAX (IEEE 802.16e), DVB-S2, WiFi (IEEE 802.11) and 10GBASE-T (IEEE 802.3an) standards attest this great performance.

A variety of LDPC decoding implementations have been explored to accomplish high throughput results [3]-[5]. It has been evidently shown that the higher throughput is achieved by the fully parallel decoding solutions; nevertheless they enlarged the hardware complexity. To overcome this drawback, several reduced-complexity and stochastic LDPC decoding algorithms are developed [6]-[8]. The current stochastic decoding algorithm confirmed their adaptability for fully parallel decoding approach [9]-[18]. Moreover and for additional silicon area reduction, diverse LDPC stochastic based decoding architectures and strategies are proposed.

However, an area-efficient architecture for ASIC-Based stochastic LDPC decoder can't systematically produces an efficient FPGA logic utilisation. It is straightforward that the ASIC implementation of six bits counter requires less silicon area compared to 32 bits memory. Nevertheless an inverse result is obtained with the FPGA implementation. A Xilinx FPGA 32 bits memory implementation can be routed using only one LUT, in contrast with the counter logic utilisation. This paper introduces a new and powerful field programmable gate array (FPGA) based stochastic Low-Density Parity-Check (LDPC) decoding process, to implement fully parallel LDPC-decoders. The proposed technique is designed to optimize the FPGA logic utilization and to decrease the decoding latency in addition to improve the convergence, even for short codes. To validate the advantage of the proposed approach, an FPGA is implemented using Xilinx Virtex-6 VLX240T. The paper is organized as follows. In Section II, an overview of the LDPC stochastic decoding is provided. In Section III, the architecture of the new proposed stochastic LDPC decoding is introduced. Results of FPGA implementation and performance are presented in Section IV, and finally a conclusion is given in Section V.

II. LDPC Stochastic Decoders

The design of an LDPC decoder is based on the $\boldsymbol{M \times N}$ parity check matrix H. N defines the number of variable nodes (VNs) while M defines the number of CNs. To encode k information bits, an ***(N, K)*** LDPC code uses $\boldsymbol{N}$ encoded bits, where $\boldsymbol{N > K}$. LDPC decoder can be represented by a factor graph which uses $\boldsymbol{N}$ VNs and ***(N-K)*** CNs. A ***dv*** degree VN has ***(dv+1)*** ports, one of which gets the channel probability and the other ***dv*** are connected to different CNs, by bidirectional ports. In the same way, the ***dc*** degree CN has dc bidirectional ports, which are connected to different VNs, and one parity-check output port. Conventional LDPC fully parallel decoder uses fixed-point operands to represent the probabilities, exchanged between the VNs and the CNs of the factor graph. Stochastic LDPC decoders function by a bit-serial iterative process. In this architecture, the received probabilities $\boldsymbol{P_{ch}}$ from the channel are converted to Bernoulli sequences as random bits sequences. Different encoded stochastic sequences can be generated for the same probability. In a ***{ai}*** Bernoulli sequence of $\boldsymbol{m}$ bits, in which $\boldsymbol{a_i \in \{0, 1\}}$, the estimated probability value is computed as:

$$P_{ch}(m) = \frac{\sum_{i=1}^{m} a_i}{m} \quad (1)$$

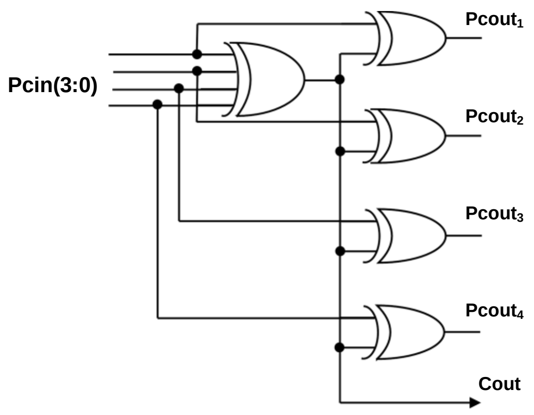

Fig. 1. Structure of a degree-4 parity-check node.

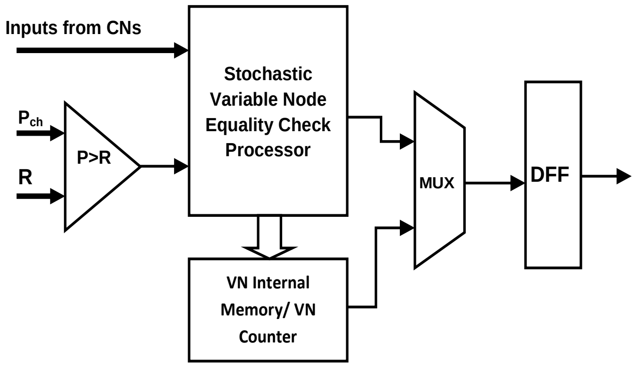
Inputs from CNs
Pch
R
P>R
Stochastic Variable Node Equality Check Processor
VN Internal Memory/ VN Counter
MUX
DFF

Fig. 2. Structure of recent stochastic variable node.

Let ***Cout***, ***Pcout$_i$*** and $Pcin_i$ be the parity-check output, the CN outputs and the CN probabilities inputs respectively, where ***Pcin$_i$ = Pr(cin$_i$=1)*** is the probability of each CN inputs, in which ***i ∊ {1, 2, ..., dc}*** and ***dc*** is CN degree. The output probability ***Pcout$_i$*** can be computed as:

$$\left.\begin{array}{l} Pcout_1 = Pcin_2 \oplus \; \ldots\ldots\ldots\oplus\; Pcin_{dc} \\ Pcout_l = Pcin_1 \oplus..\oplus\; Pcin_{l-1} \oplus Pcin_{l+1} \oplus..\oplus Pcin_{dc} \\ Pcout_{dc} = Pcin_1 \oplus \ldots\ldots\ldots.\oplus\; Pcin_{dc-1} \end{array}\right\} (2)$$

The parity-check output ***Cout***, of dc degree CN, can be computed according to (3). Fig. 1 illustrates the structure of a ***dc=4*** CN used in the conventional stochastic decoder.

$$Cout = Pcin_1 \oplus . . .\oplus\; Pcin_{dc} \qquad (3)$$

Let ***Pvin$_1$*** and ***Pvin$_2$*** be the probability of two input bits in ***dv=2*** VN. The variable node output probability *Pvout* can be computed as:

$$Pvout = \frac{Pvin_1.Pvin_2}{Pvin_1.Pvin_2 + (1 - Pvin_1).(1 - Pvin_2)} \qquad (4)$$

If the VN inputs are same, this state is named the agreement state, one of the input bits will be transmitted to the output. When the inputs are not identical, the variable node requires an advanced method to generate the output bit. This state is named the hold state or disagreement state. One of the advanced stochastic method bit generation (ASMBG) can be used.

$$Pvout(N) = \begin{cases} Bit\ using\ (4) & if\ Pa = Pb \\ Bit\ using\ \text{ASMBG} & otherwis \end{cases} \quad (5)$$

Fig. 2 shows the recent stochastic variable node principal structure. The stochastic LDPC decoding algorithm can be summarised as follows:

Algorithm 1 Stochastic LDPC decoding

```
Initialization
 1. Load the LLRs corresponding probabilities $P_{ch}$ for each variable node (one DC) and transform $P_{ch}$ to Bernouli sequence $a_i$ (each DC).
 2. Initialize the variable nodes internal memories (16 to 32 DCs for 32 bits memory [10]) or the internal saturated counter (one DC [16]).

Iterations
 3. Variable to check node: At each decoding cycle, the variable node computes there inputs bits using (5) and sends there outputs bits to the corresponding check nodes.
 4. Check to variable node: At each decoding cycle, the check node computes there inputs bits using (2) and sends there outputs results to the corresponding variable nodes. Simultaneously, the check nodes send their outputs states using (3) to the syndrome checker.
 5. If $xH^T = 0$ or the maximum of DCs is reached, terminate the decoding process. Otherwise go to Step 3.
```

III. Proposed LDPC Stochastic Structure

As mentioned in the introduction section, an efficient ASIC-based architecture algorithm can't systematically provide the best approach for an efficient FPGA implementation. In this section we present the new LDPC stochastic decoding method which aims to improve the decoder performance and to reduce the FPGA resource utilization.

It has been shown that the LDPC stochastic decoder, which there VNs use the latest output bits as code bits, provide similar BER performance to the version with saturating up/down counters as a VN output decision mechanism [17]-[18]. Furthermore, it has been demonstrated that the initialization of the first VN output bit, transmitted to the hard decision unit according to received probability channel, helps to improve the stochastic decoder convergence [15]. The proposed VN exploits the two referenced characteristics, in addition to adopt an internal memory-based approach, similar to DS and EM versions. The converted Bernoulli sequences are used as a variable node input. All output variable nodes are initialized by one bit coded probability channel, during the loading of the channel Log Likelihood Ratio (LLR) corresponding probabilities. Each VN internal memory is mapped in one FPGA LUT RAM.

The output variable node probability will be computed as:

$$Pvout(t) = \begin{cases} Pst & if\ t = 1 \\ Pa\ or\ Pb & if\ t \neq 1\ \&\ Pa = Pb \\ Bit\ from\ FPGA\ LUT\ RAM & otherwis \end{cases} \quad (6)$$

where $$Pst = \begin{cases} 1 & if\ Pch \geq 0.5 \\ 0 & otherwise \end{cases}$$

Pvout(1) is the first iteration VN output probability. During the disagreement state ($t \neq 1$) , the proposed architecture generate a new bit based on a random bit selection from the VN internal memory (IM). The IM length can be increased up to FPGA LUT RAM size. Based on (6), the hardware implementation of the new improved decoding approach does not require extra hardware complexity, for FPGA devices. Moreover, the projected technique computes the received probability without any additional decoding cycle.

The $P_{ch}(k\text{-}1)$ signal is transmitted to the VN output DFF during the first process cycle. After the first iteration and until the last one, the multiplexer sends the variable node processor output bit to the VN output DFF. In this way and identically to the CSS process, the majority of variable nodes outputs start with a right bit and detour the random stochastic initialization. Fig. 3 represents the main structure of the proposed variable node. Similar to the EM and DS design, the decoding cycle (DC) matches to one of the iteration for the proposed LDPC stochastic decoding.

The proposed CN possess two inputs signals categories and two outputs. The inputs signals are the $\boldsymbol{CNin_i}$ signals and $\boldsymbol{State_i}$ signals, in which $\boldsymbol{i \in \{1, 2\ldots dc\}}$. These two signals are provided by VNs outputs signals. The first outputs signals are the $\boldsymbol{CNout_i}$ signals, which are sent to VN inputs. The second output is the parity-check output state ***CNstate***.

The new CN uses a computing process similar to the DS approach. All ***CNstate*** outputs are connected to the syndrome checker unit.

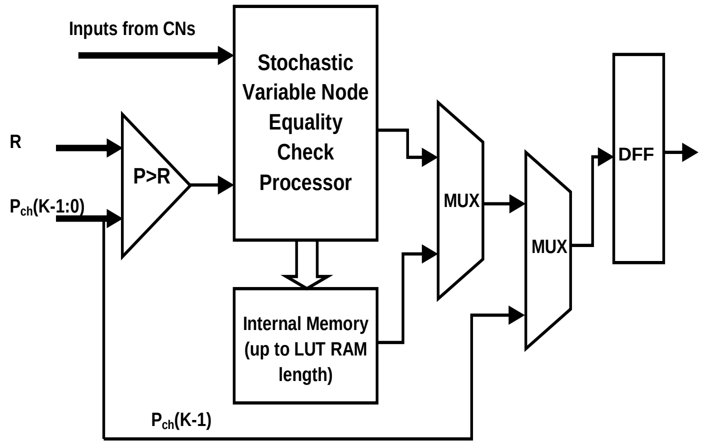
Inputs from CNs
R
P_{ch}(K-1:0)
P>R
Stochastic Variable Node Equality Check Processor
Internal Memory (up to LUT RAM length)
MUX
MUX
DFF
P_{ch}(K-1)

Fig. 3. The structure of the proposed stochastic variable node.

The parity-check output state ***CNstate***, of ***dc*** degree CN, is computed in the same way of (3) and can be written as follows:

$$CNstate = \sum_{i=1}^{dc} \oplus CNin_i \quad (8)$$

Where, $\sum \oplus$ is the bitwise XOR operation and ***dc*** is the parity-check node degree.

Each CN outputs signals $\boldsymbol{CNout_i}$ uses the $\boldsymbol{CNin_i}$ signals and $\boldsymbol{State_i}$ signals, to produce the $\boldsymbol{CNout_i}$ signals according to (9). The ***CNstate*** result given by (8) can be exploited.

$$CNout_i = \left((CNstate) \wedge \overline{\left(\bigvee_{i=1}^{dc} state_i \right)} \right) \oplus CNin_i \quad (9)$$

Where, $\overline{\bigvee}$ is the bitwise NOR operation and $\wedge$ is the bitwise AND operation.

The new stochastic LDPC decoding algorithm can be summarised as follows:

Algorithm 2 The proposed Stochastic LDPC decoding

```
Initialization
 1. Load the LLRs corresponding probabilities $P_{ch}$ simultaneously with initializing the variable node output (one DC) and transform $P_{ch}$ to Bernouli sequence $a_i$ (each DC).
Iterations
 2. and 3. Similar to Step 3 and Step 4 of Algorithm 1.
 4. If $xH^T = 0$ or the maximum of DCs is reached, terminate the decoding process. Otherwise go to Step 2.
```

IV. Implementation Results and Performance

It has been demonstrated that the enlargement of the VN internal memories size increases the LDPC stochastic decoding converges [10]-[11]. However and mainly, adding additional memory capacity implicates an extra hardware complexity and resources. The FPGA organization and implementation need special considerations. In addition to slices resources, memory can be mapped using Block RAMs or using Distributed RAMs (LUT RAM).

The main target of the proposed structure is to improve the FPGA-based LDPC decoding performance, without supplementary FPGA resources. To confirm the improvement of the new design, a medium (1024, 512) and short (200, 100) LDPC codes are implemented on Xilinx Virtex-6 VLX240T field programmable gate array (FPGA) device, with various methods.

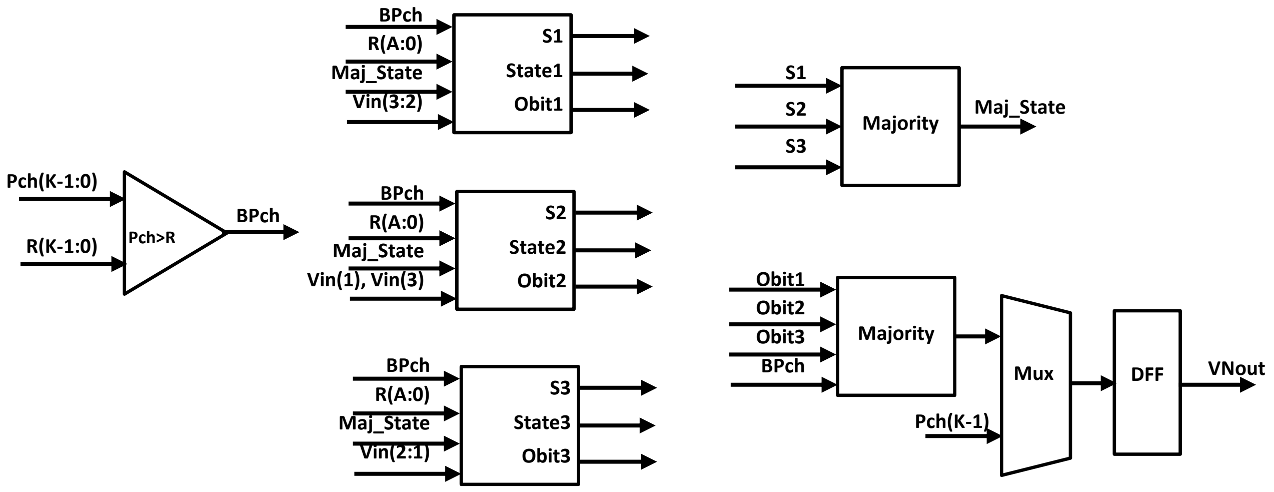

Fig. 4. The block diagram of the proposed degree-3 Variable node.

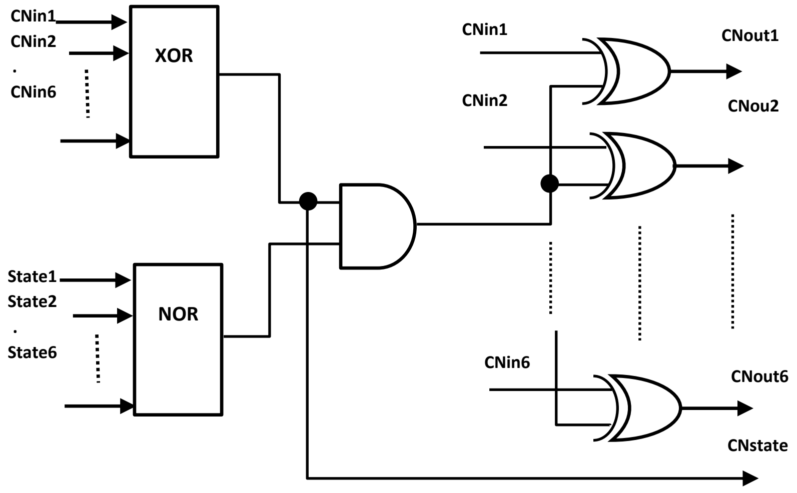

Fig. 5. The structure of the proposed degree-6 CN.

Fig. 4 presents the block diagram of the new degree-3 Variable node of the proposed (1024, 512) and (200, 100) LDPC codes. The degree-3 Variable Node is composed by 3 degree-3 sub-node. The majority state and the random address signals are connected to all sub-nodes. One of the 3 degree-3 sub-node input is connected to probability signal by a comparator. The other two sub-node inputs are connected to the check node output. The three ***S*** signals are combined to produce the majority state signal. The FPGA implementation of VN Internal Memory is achieved by using the (LUT) Slice FPGA distributed RAMs.

The implemented CNs use similar structure to CNs adopted by the DS and the CSS decoders. Fig. 5 gives the main structure of degree-6 CN. The results of the FPGA implementation of the (200, 100) and (1024, 512) LDPC Regular Codes, with one-step initialized counter-based VN [16], DS and the proposed approach, are shown in Table 1.

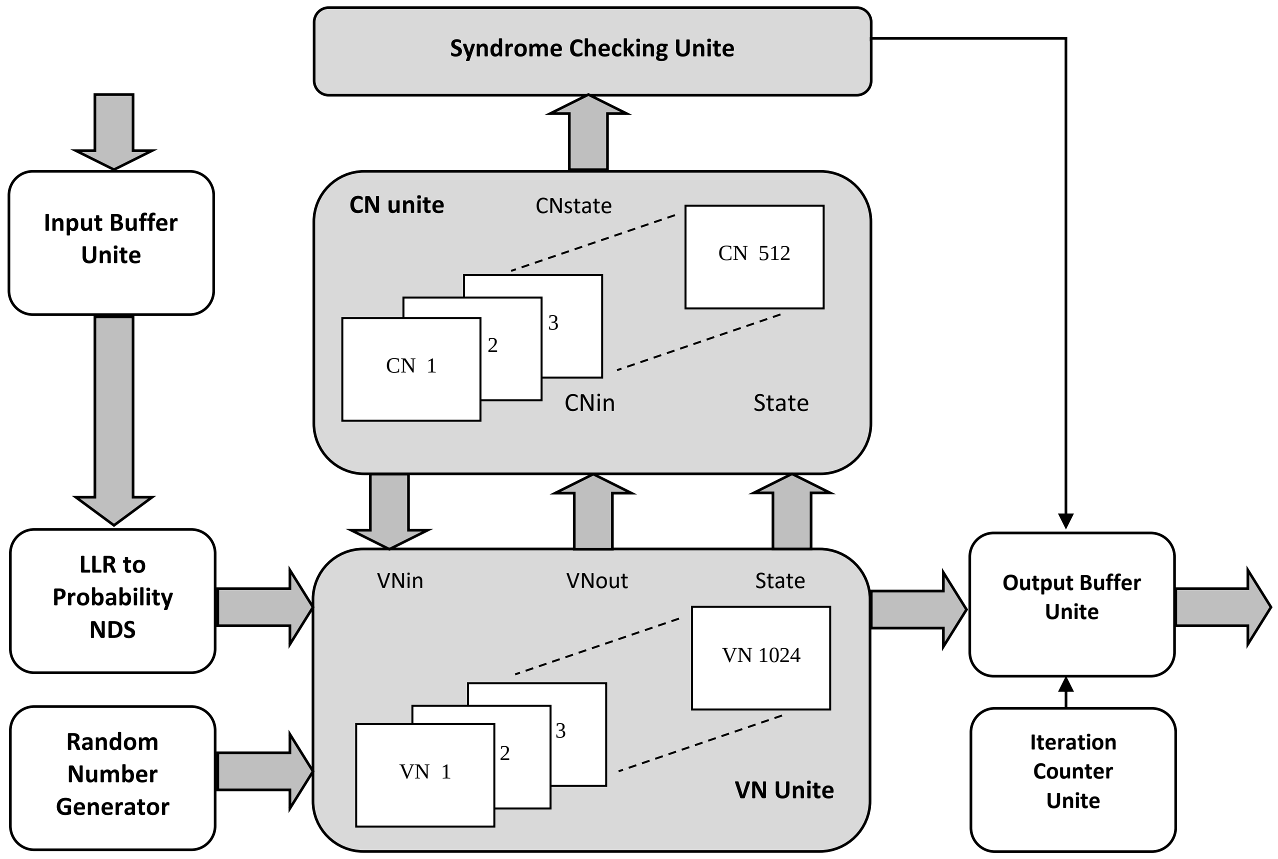

Fig. 6. The block diagram of the proposed (1024, 512) LDPC decoder.

The EM version gives an FPGA implementation result close to the DS version. The implementation of 2×1 bits up to 64×1 bits uses only one LUT in Virtex-6 Xilinx FPGA. Therefore, the proposed LDPC decoder version can be implemented using up to 64-bit VN internal memory without requiring additional FPGA resources. As we can see, the FPGA implementation of the one-step initialized counter based decoder need additional resources, compared to the EM and the DS versions. This disadvantage is caused by the utilisation of initializing counters instead of the VN internal memories used in DS. The additional reduction of FPGA logic utilisation seen for the new proposed decoder is principally obtained as a result of the unemployment of VN output saturated counter.

Fig. 6 presents the block diagram of the proposed (1024, 512) LDPC stochastic decoder. The main units are the variable nodes unite, the parity-check nodes unite, and the syndrome checking unite. The VN unite and CN unite exchange the stochastic information until reaching a correct code or the maximum number of iteration. The correct code is detected by the syndrome checking unite and the maximum of iterations is pointed by the iteration counter unite. The outputs of the Random Number Generator are employed with the VN comparators to generate the Bernouli sequences. Furthermore, they are directly used to drive the addresses buses of the FPGA distributed RAM, used as VN internal memory.

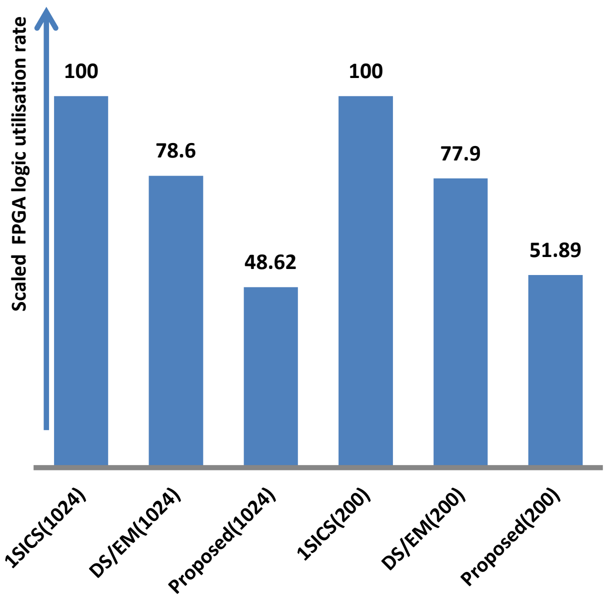

Fig. 7. FPGA logic utilisation reduction rate.

Fig. 7 shows the scaled FPGA logic utilisation rate. The proposed decoder achieves an average reduction about of 50% compared to the one-step initialized counter and an average reduction about of 35% compared to DS and EM decoders.

TABLE I. RESOURCE UTILIZATION OF STOCHASTIC LDPC DECODERS IMPLEMENTATION IN XILINX VIRTEX-6 VLX240T FPGA DEVICE

LDPC Architecture	Code	Implementation	Decoder Logic utilization (LUT)	VNs Logic utilization (LUT)	VN counter / Memory length (bits)	VNs rate utilization (%)	VNs Reduction rate compared to DS (%)	Reduction rate compared to DS (%)
Initialized Counter-based VN [16]	(1024, 512)	Stochastic fully parallel	60 947	51 200	06	84,01	- 35,13	- 27,95
	(200, 100)		11 767	10 000	06	84,98	- 35,13	- 28.36
DS [14]	(1024, 512)	Stochastic fully parallel	47 635	37 888	02	79,54	0	0
	(200, 100)		9 167	7 400	02	80,72	0	0
Proposed	(1024, 512)	Stochastic fully parallel	29 633	21 504	16	72,57	+ 43,24	+ 37,79
	(200, 100)		6 106	4 200	16	68,78	+ 43,24	+ 33,39

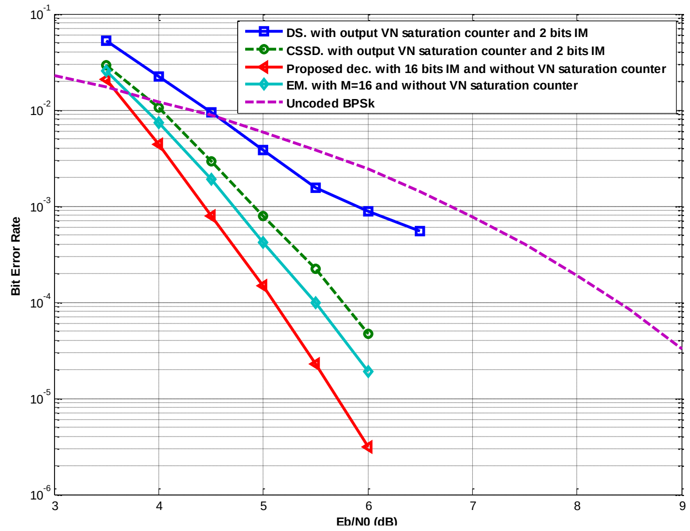

Fig. 8. BER performance of the (200,100).

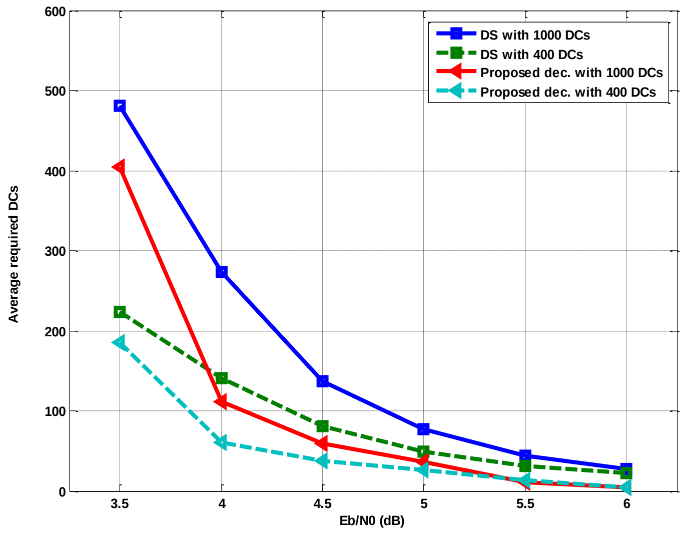

Fig. 9. Average decoding cycles.

In addition to the logic utilisation reduction, the proposed architecture offers additional performance even for short codes. Fig. 8 displays the BER performance of the (200,100) FPGA-based LDPC decoding with DS, CSS, EM and the proposed methods. The maximum decoding latency is set to 1000 DCs.

The BER of the new approach demonstrate the significant improvement. The proposed decoder with 16 bits internal memory outperforms the EM decoder by 0,5 dB at BER of 10^{-5}. The average required DCs of the DS and the proposed architecture, with 1000 and 400 maximum decoding latency are presented in Fig. 9. A reduction of 85,3% and 81,1%, in the ADC compared to the DS decoder at an SNR of 6 dB, is observed for 1000 and 400 Max-DCs, respectively.

V. CONCLUSION

In this work, we investigated the complexity and performance of FPGA-based implemented LDPC stochastic decoders. Therefore, a new fully parallel stochastic LDPC decoding approach was presented, which can outperform all state of the arte versions. The improvement was accomplished by introducing an efficient stochastic variable node.

A reduction of decoding latency and complexity, in addition of BER amelioration, are achieved even for short codes. A Xilinx Virtex-6 VLX 240T FPGA implementation results validated the advancement of the new method. An average reduction of 35% and 85% of logic utilisation and average decoding cycles respectively, compared to DS method. The BER performance of the new (200, 100) decoder exceed DS, CSS and EM versions. A gain gap of 0,5 dB, compared to EM, is observed at BER of 10^{-5}.

REFERENCES

[1] R. G. Gallager, “Low density parity check codes,” IRE Trans. Inf. Theory, vol. 8, no. , pp. 21–28, Jan. 1962.

[2] D. J. C. MacKay and R. M. Neal, “Near Shannon limit performance of low density parity check codes,” Electron. Lett., vol. 32, no. 18, pp. 1645–1646, 1996.

[3] The IEEE 802.16 Working Group [Online]. Available: http://www.ieee802.org/16/

[4] The IEEE 802.3an 10GBASE-T Task Force [Online]. Available: www.ieee802.org/3/an

[5] IEEE Working Group for 40 Gb/s and 100 Gb/s Operation Std. [Online]. Available: http://www.ieee802.org/3/

[6] The Digital Video Broadcasting Standard [Online]. Available: www.dvb.org

[7] The IEEE 802.11n Working Group Std. [Online]. Available: http://www.ieee802.org/11/

[8] V. Gaudet and A. Rapley, “Iterative decoding using stochastic computation,” Electron. Lett., vol. 39, no. 3, pp. 299–301, Feb. 2003.

[9] S. S. Tehrani, W. Gross, and S.Mannor, “Stochastic decoding of LDPC codes,” IEEE Commun. Lett., vol. 10, no. 10, pp. 716–718, Oct. 2006.

[10] S. S. Tehrani, S. Mannor, and W. Gross, “Fully parallel stochastic LDPC decoders,” IEEE Trans. Signal Process., vol. 56, no. 11, pp. 5692–5703, Nov. 2008.

[11] S. Tehrani, A. Naderi, G.-A. Kamendje, S. Mannor, and W. Gross, “Tracking forecast memories in stochastic decoders,” in IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP), Apr. 2009, pp. 561–564.

[12] S. S. Tehrani, A. Naderi, G. A. Kamendje, S. Mannor, and W. Gross, “Tracking forecast memories for stochastic decoding” J. Signal Process. Syst. pp. 1–11, 2010 [Online]. Available: http://dx.doi.org/10.1007/s11265-009-0441-5

[13] S. S. Tehrani, A. Naderi, G.-A. Kamendje, S. Hemati, S. Mannor, and W. Gross, “Majority-based tracking forecast memories for stochastic LDPC decoding,” IEEE Trans. Signal Process., vol. 58, no. 9, pp. 4883–4896, Sep. 2010.

[14] Ali Naderi, Shie Mannor, Mohamad Sawan and Warren J. Gross, “Delayed Stochastic Decoding of LDPC Codes” IEEE Trans. Signal Process., vol. 59, no. 11, pp. 5617–5626, Nov. 2011.

[15] Mountassar Maamoun, Rafik Bradai, Ali Naderi, Rachid Beguenane and Mohamad Sawan, “Controlled Start-Up Stochastic Decoding of LDPC Codes” in IEEE Int. NEWCAS Conference, Paris, France, June. 2013.

[16] Di Wu, Yun Chen, Qichen Zhang, Yeong-luh Ueng and Xiaoyang Zeng “Strategies for Reducing Decoding Cycles in Stochastic LDPC Decoders” IEEE Trans. Circuits and Systems II, vol. 63, no.91, pp. 873 - 877, Sept. 2016.

[17] Kuo-Lun Huang, Vincent Gaudet and Masoud Salehi, “Output Decisions for Stochastic LDPC Decoders” in 48th Annual Conference on Information Sciences and Systems, (CISS), *Princeton, USA,* March. 2014.

[18] Kuo-Lun Huang, “Efficient Algorithms for Stochastic Decoding of LDPC Codes” Doctor of Philosophy, Northeastern University, Boston, USA, May. 2016.