FPGA Design of High Throughput LDPC Decoder based on Imprecise Offset Min-Sum Decoding

Truong Nguyen-Ly*†, Khoa Le†, F. Ghaffari†, A. Amaricai‡, O. Boncalo‡, V. Savin* and D. Declercq†
*CEA-LETI, MINATEC Campus, Grenoble, France, {thientruong.nguyen-ly, valentin.savin}@cea.fr
†ETIS, ENSEA / CNRS UMR-8051 / University of Cergy-Pontoise, France, {le.khoa, ghaffari, declercq}@ensea.fr
‡University Politehnica Timisoara, Computer Engineering Department, Timisoara, Romania, {boncalo, amaricai}@cs.upt.ro

***Abstract*—This paper first proposes two new LDPC decoding algorithms that may be seen as imprecise versions of the Offset Min-Sum (OMS) decoding: the Partially OMS, which performs only partially the offset correction, and the Imprecise Partially OMS, which introduces a further level of impreciseness in the check-node processing unit. We show that they allow significant reduction in the memory (25% with respect to the baseline) and interconnect, and we further propose a cost-efficient check-node unit architecture, yielding a cost reduction of 56% with respect to the baseline. We further implement FPGA-based layered decoder architectures using the proposed algorithms as decoding kernels, for a $(3, 6)$-regular Quasi-Cyclic LDPC code of length 1296 bits, and evaluate them in terms of cost, throughput and decoding performance. Implementation results on Xilinx Virtex 6 FPGA device show that they can achieve a throughput between 1.95 and 2.41 Gbps for 20 decoding iterations (48% to 83% increase with respect to OMS), while providing decoding performance close to the OMS decoder, despite the impreciseness introduced in the processing units.**

I. Introduction

Low-Density Parity-Check (LDPC) codes have found extensive applications in modern communication systems due to their excellent decoding performance, high throughput capabilities [1], [2], [3], [4], and power efficiency [5], [6]. They have been adopted in several recent communication standards, such as 802.11n (Wi-Fi), 802.16e (WiMax), 802.15.3c (WPAN), and several DVB standards.

Most hardware implementations of LDPC decoders are based on the Min-Sum (MS) algorithm [7] or enhanced versions of it, such as Normalized MS (NMS) and Offset MS (OMS) [8]. Further enhancements have been proposed in the literature, as follows. In [9], the authors proposed a modification of the OMS, in which the offset factor is adjusted iteratively, by using information from previous decoding steps. Similarly, adaptive NMS or OMS decoding algorithms have been proposed in [10], [11], where the normalization/offset factor is adjusted at each iteration according to information gained from the check-node processing step. In [10] the offset factor is determined by the magnitude of the minimum output of the check-node, while in [11] the normalization/offset factor is adjusted according to the check-node state (verified or not). It should be noticed that all the above-mentioned papers focus on improving the error correction performance of the NMS/OMS decoders, especially in case of irregular codes. This is achieved at the price of an increased computational complexity, required by the adaptation mechanism, which translates into cost and throughput penalties in case of hardware implementation.

This paper proposes two algorithms, with two levels of impreciseness to OMS, aimed at improving cost efficiency and increasing throughput, while keeping degradation of the decoder's error correction performance as small as possible. The first level of impreciseness concerns the offset factor: rather than subtracting a constant offset factor from check-node messages, we simply "erase" the value of the least significant bit (LSB). We refer to this decoding algorithm as Partially OMS (POMS) and show that it has the following advantages in terms of hardware implementation: (i) significant reduction of interconnect and memory requirements, and (ii) simple architecture for the check-node processing unit (CNU), which avoids the use of comparator trees. Besides, the error-correction performance of the proposed POMS is very close to that of the OMS, both outperforming the conventional MS decoding. Moreover, we introduce a second level of impreciseness in the CNU, by suppressing some of the signals computed by the POMS decoder. The corresponding decoder is referred to as Imprecise-POMS (I-POMS) and we show that it yields further improvements in hardware cost and throughput.

The rest of the paper is organized as follows. Section II introduces the MS, OMS as well as the proposed POMS decoding algorithms. The hardware architecture of the MS, OMS, POMS decoders for (3,6)-regular Quasi-Cyclic (QC) LDPC codes is discussed in Section III. Section IV presents the I-POMS decoder. Implementation results are presented in Section V, and Section VI concludes the paper.

II. Min-Sum Based Decoding

We consider an LDPC code defined by a parity-check matrix H of size $M \times N$. Using the conventional bipartite graph representation, the N columns of H correspond to variable-nodes $n \in \{1, \cdots, N\}$, the M rows of H correspond to check-nodes $m \in \{1, \cdots, M\}$, and the non-zero entries of H correspond to edges connecting variable and check-nodes. We further denote by $H(n)$ (or $H(m)$) the set of neighbor nodes of a variable-node n (or check-node m). We shall consider that H is composed of L horizontal layers, with each layer consisting of M/L consecutive rows, such that each column has at most one non-zero entry in any horizontal layer. We denote by M_l the set of the rows belonging to the l^{th} layer.

We consider a codeword $(x_1, \cdots, x_N)$ that is sent over a binary input channel, and denote by $(y_1, \cdots, y_N)$ the received word. The following notation for message-passing decoders will be used throughout the paper:

- γ_n is the log-likelihood ratio (LLR) value of x_n according to the received y_n value; it is also referred to as the a priori LLR of variable node n;
- $\tilde{\gamma}_n$ is the a posteriori (AP) LLR of variable node n;
- $\alpha_{m,n}$ is the message sent from variable-node n to check-node m;
- $\beta_{m,n}$ is the message sent from check-node m to variable-node n;
- $\text{LSB}[x]$ denotes the least significant bit of x.

MS, OMS, and POMS decoding algorithms are described in Algorithm 1. While our description assumes layered decoding [12] and finite quantization (due to hardware limitations, see next section for detailed discussion), it can be easily extended to more general settings. We assume that input LLRs and exchanged messages are quantized on q bits, while AP-LLR values are quantized on $\tilde{q}$ bits, with $q < \tilde{q}$. Subtractions and additions used in the Variable Node Unit (VNU) and AP-LLR update steps are implemented through the use of $\tilde{q}$-bit saturated adders. $\alpha_{m,n}$ messages are saturated to q bits just

978-1-4799-8893-8/15/$31.00

Algorithm 1 MS / OMS / POMS decoding

[Initialization]

for all $n = 1, \ldots, N$ **do** $\tilde{\gamma}_n = \gamma_n = \log \frac{\Pr(x_n=0|y_n)}{\Pr(x_n=1|y_n)}$;

for all $n = 1, \ldots, N$ and $m \in H(n)$ **do** $\beta_{m,n} = 0$;

[Decoding Iterations]

for all $l = 1, \ldots, L$ **do** ▷ Loop over horizontal layers

  for all $m \in M_l$ and $n \in H(m)$ **do** ▷ **VNU**

    $\alpha_{m,n} = \tilde{\gamma}_n - \beta_{m,n}$;

  for all $m \in M_l$ and $n \in H(m)$ **do** ▷ **CNU**

    $\beta_{m,n} = \left(\prod_{n' \in H(m)\setminus n} \mathrm{sgn}(\alpha_{m,n'})\right) \cdot |\beta|_{m,n}$;

    // where $|\beta|_{m,n}$ is defined in the algorithm footnote below

  for all $m \in M_l$ and $n \in H(m)$ **do** ▷ **AP-LLR**

    $\tilde{\gamma}_n = \alpha_{m,n} + \beta_{m,n}$;

end (horizontal layers loop)

Let $\alpha_{m,n}^{\text{SAT}}$ be the value of $\alpha_{m,n}$ (output of the VNU) saturated to a lower quantization level. In the CNU, the message amplitude $|\beta|_{m,n}$ is computed as follows, according to the decoding algorithm:

MS : $|\beta|_{m,n} = \min_{n' \in H(m)\setminus n} \left(|\alpha_{m,n'}^{\text{SAT}}|\right)$

OMS: $|\beta|_{m,n} = \max\left(\min_{n' \in H(m)\setminus n} \left(|\alpha_{m,n'}^{\text{SAT}}|\right) - \lambda,\ 0\right)$

// where $\lambda > 0$ is the offset factor

POMS: $|\beta|_{m,n} = \min_{n' \in H(m)\setminus n} \left(|\alpha_{m,n'}^{\text{SAT}}|_{[\text{LSB}\leftarrow 0]}\right)$

// $x_{[\text{LSB}\leftarrow 0]}$ means that the LSB of x is set to 0

before entering the Check Node Unit (CNU). Hence, the amplitude values $|\alpha_{m,n}^{\text{SAT}}|$ used in the CNU are $(q-1)$-bit values.

It can be noticed that the three decoders only differ in the computation of the amplitude of check-node messages, denoted by $|\beta|_{m,n}$. For the POMS decoding, the LSB of $|\alpha_{m,n}^{\text{SAT}}|$ is set to zero prior to the minimum computation. It can be easily seen that this is equivalent to first computing the minimum and then setting the LSB of the result to zero. Moreover, assuming that the offset factor for the OMS decoding is $\lambda = 1$, we have (here below, superscripts are used to denote the corresponding algorithm):

$$|\beta|_{m,n}^{(\text{POMS})} = \begin{cases} |\beta|_{m,n}^{(\text{MS})} & \text{if } \text{LSB}\left[|\beta|_{m,n}^{(\text{MS})}\right] = 0 \\ |\beta|_{m,n}^{(\text{OMS})} & \text{otherwise } \ (\text{for } \lambda = 1) \end{cases} \quad (1)$$

The proposed POMS decoding operates the same as the OMS decoding only if $\text{LSB}\left[|\beta|_{m,n}^{(\text{MS})}\right] = 1$, and thus the offset correction (with $\lambda = 1$) is only partially achieved. Consequently, the decoding performance of POMS is expected to be a trade-off between MS and OMS. Note that $\lambda = 1$ is the optimal offset factor in case that exchanged messages are quantized on a relatively small number of bits (e.g. $q = 4$). For such a quantization scheme, since the LSBs of $\beta_{m,n}^{(\text{POMS})}$ messages are always zero and need not be stored, POMS may lead to significant savings in memory and interconnects.

III. Hardware Architecture for QC-LDPC Decoders with Layered Scheduling

A. Hardware Architecture for Min-Sum Based Decoders

QC-LDPC codes are known to achieve error correction performance comparable to that of random codes, while facilitating the hardware implementation of the decoder, thanks to the specific structure of their parity check matrix [2]. In this paper, we consider a QC-LDPC code, defined by a base-matrix B of size 12×24. The parity check matrix is defined by an expansion factor $Z = 54$,

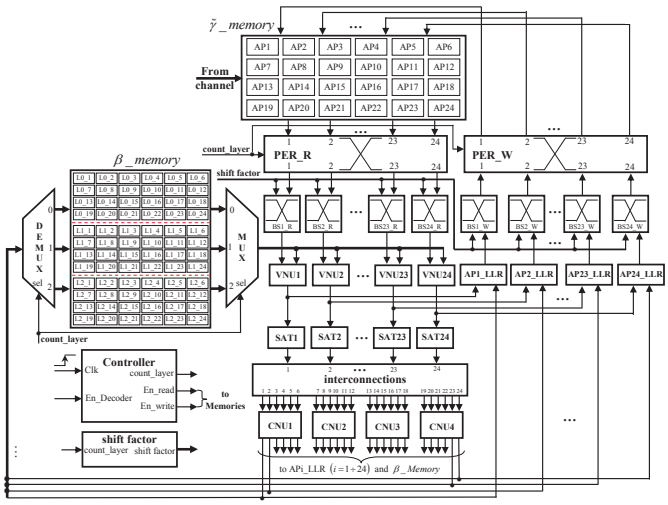

Fig. 1. Block diagram for $(3, 6)$-regular QC-LDPC decoder

meaning that each non-negative entry of B is expanded to a cyclic permutation matrix of size $Z \times Z$. This construction yields a parity check-matrix H of size $(M = 648) \times (N = 1296)$. The code is $(3, 6)$-regular, meaning that H has 3 non-zero entries per column, and 6 non-zero entries per row. Moreover, B is designed such that H can be divided in $L = 3$ horizontal layers, with each layer corresponding to 4 consecutive rows of B. Each column of H has exactly one non-zero entry in each layer. Figure 1 illustrates the block diagram of MS, OMS and POMS decoders, and can be summarized by the following main blocks:

Memory blocks. Two memory blocks are used, one for the $\tilde{\gamma}_n$ values ($\tilde{\gamma}$*_memory*) and one for the $\beta_{m,n}$ messages (β*_memory*). $\tilde{\gamma}_n$ values are quantized on $\tilde{q} = 6$ bits, and $\beta_{m,n}$ values on $q = 4$ bits. Memory requirements for the $\beta_{m,n}$ values are further reduced for the POMS decoder, as will be discussed in the next section. All data is read and processed at during one clock cycle, then written during the consecutive clock cycle, and so on. Therefore, the decoder takes 2 clock cycles for each layer processing.

Permutation Networks for Reading and Writing (PER_R, PER_W). PER_R permutation is used to rearrange the data read from $\tilde{\gamma}$*_memory*, according to the processed layer, so as to ensure processing by the proper VNU/CNU. PER_W block operates oppositely to PER_R.

Barrel Shifters for Reading and Writing (BS_R, BS_W). Barrel shifters are used to implement the cyclic (shift) permutations corresponding to the non-negative entries of the base matrix B. In our design, we use 24 BS_R and 24 BS_W blocks, corresponding to the number of columns of B. Each of them has 54 inputs and 54 outputs (for expansion factor $Z = 54$).

Variable Node Units (VNUs). These processing units compute the $\alpha_{m,n}$ messages. The inputs of the VNUs are read from $\tilde{\gamma}$*_memory* and β*_memory*. Each VNUi block ($i = 1, \ldots, 24$) in Figure 1 consists of 54 6-bit saturated subtractors for the parallel execution of 54 variable-nodes (one column of B).

Saturators (SATs). Prior to CNU processing, $\alpha_{m,n}$ values are saturated to $q = 4$ bits. After saturation, $\alpha_{m,n}^{\text{SAT}}$ values are within the range $\{-7, \ldots, +7\}$.

Check Node Units (CNUs). These processing units compute the $\beta_{m,n}$ messages. For simplicity, in Figure 1 we represented four CNUi blocks ($i = 1 \ldots, 4$), corresponding to the 4 rows of the base matrix within one layer. Each CNUi block includes 54 computing units

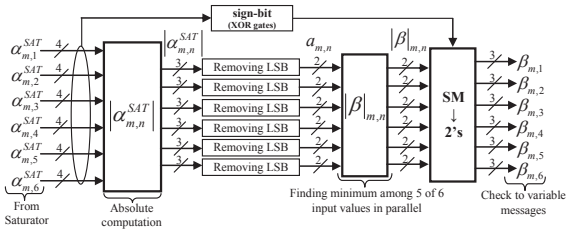

Fig. 2. Proposed CNU architecture for POMS ($d_c = 6$)

CNUij (equal to the expansion factor of the base matrix). A total of 216 CNUij blocks are used to process in parallel all the check nodes within one layer. Further details regarding the CNU architecture are given in Section III-B, where the proposed CNU architecture for the POMS decoder is also be presented.

AP-LLR Units. These units compute the $\tilde{\gamma}_n$ values. Each APi-LLR ($i = 1, \ldots, 24$) in Figure 1 consists of 54 6-bit saturated adders, for the parallel execution of 54 variable-nodes (corresponding to one column of B).

Controller. This block generates control signals such as *count_layer* for indicating which layer is being processed, *En_read* and *En_write* for reading and writing data, etc. It also controls the synchronous execution of the other blocks.

B. Hardware Architecture for Proposed POMS Decoder

To ensure efficient implementation of the proposed POMS decoder, the CNU, VNU and AP-LLR blocks are modified as discussed below.

CNU Architecture. Many CNU architectures compute only the first minimum, the second minimum and index of the first minimum of the incoming $|\alpha^{\text{SAT}}_{m,n}|$ messages (based on the observation that the amplitudes of the outgoing check-node message are equal to the first minimum, except for the corresponding index for which the amplitude of the check-node message is given by the second minimum). Yet, as indicated in [13], this method is suitable for ASIC implementation, but for FPGA implementation it is three times more complex than parallel computation of $|\beta|_{m,n}$ by comparator trees. In this work, the CNU architecture proposed in [13] has been implemented, both for the MS and the OMS decoders.

We discuss now the CNU architecture for the POMS decoder. As before, $\alpha_{m,n}$ messages are saturated to $q = 4$ bits by the SAT block. After saturation, POMS requires that the LSB of the absolute value $|\alpha^{\text{SAT}}_{m,n}|$ is zeroed. Let $a_{m,n}$ denote the 2-bit value obtained from $|\alpha^{\text{SAT}}_{m,n}|$ by removing the LSB, as illustrated in Figure 2. Let d_c be the number of incoming messages into the CNU (in our case $d_c = 6$). The computation of $|\beta|^{(\text{POMS})}_{m,n}$ requires finding the minimum among $(d_c - 1)$ 2-bit values (precisely the $a_{m,n'}$ values, for $n' \in H(m) \setminus n$). Let us define:

$$\textit{AndMsb} = \textbf{AND}\,(a_{m,n'}[1])\,;\text{// AND of MSB of } (d_c-1) \text{ inputs} \quad (2)$$

$$\textit{AndLsb} = \textbf{AND}\,(a_{m,n'}[0])\,;\text{// AND of LSB of } (d_c-1) \text{ inputs} \quad (3)$$

$$\textit{Detect\_0} = \begin{cases} 0, & \text{if at least one input equals to } 0 \\ 1, & \text{otherwise} \end{cases} \quad (4)$$

// signal that detects if at least one input equals to 0

With the above notation, the $|\beta|^{(\text{POMS})}_{m,n}$ 2-bit value (omitting LSB that is equal to 0) can be computed by:

$$|\beta|^{(\text{POMS})}_{m,n}\ (2 \text{ bits}) = \begin{cases} 01, & \text{if } [\textit{Detect\_0}, \textit{AndMsb}, \textit{AndLsb}] = 100 \\ [\textit{AndMsb}, \textit{AndLsb}], & \text{otherwise} \end{cases} \quad (5)$$

(in binary format)

where $[x, y]$ denotes concatenation operator.

VNU and AP-LLR Architecture. According to the above discussion,

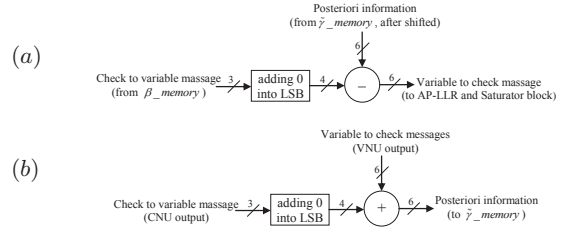

Fig. 3. VNU (a) and AP-LLR (b) architectures for POMS decoder

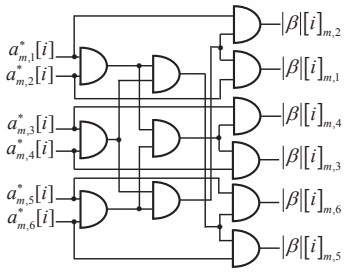

Fig. 4. Diagram circuit for computing $|\beta|_{m,n}$ messages in parallel ($d_c = 6$) for I-POMS decoder (one such circuit is used for each of LSB and MSB)

only 3 bits are required for each $\beta^{(\text{POMS})}_{m,n}$ message (2 bits for $|\beta|^{(\text{POMS})}_{m,n}$ and 1 bit for the sign), representing a reduction by 25% of the β*_memory* size, with respect to conventional MS and OMS decoders. To accommodate this change, in the VNU and AP-LLR blocks the $\beta^{(\text{POMS})}_{m,n}$ message is extended to 4 bits, by appending a zero LSB, as illustrated in Figure 3.

IV. Imprecise Partially Offset Min-Sum Decoder

This section describes the I-POMS decoder, obtained by introducing a second level of impreciseness in the CNU processing. As discussed in the above section, POMS's CNU comprises a *Detect*_0 signal, which is used to obtain the correct minimum value in case that both *AndMsb* and *AndLsb* signals are 0. In case that *AndMsb* = *AndLsb* = 0, the *Detect*_0 signal allows to distinguish between the following two possible cases: (i) at least one input is equal to 0 (hence the minimum is equal to 0), or (ii) no input is equal to zero, in which case there must be at least one input equal to 1 and another one equal to 2 (hence the minimum is equal to 1). However, compared to the *AndMsb* and *AndLsb* signals, the *Detect*_0 signal requires much more logic to be implemented. To further simplify the CNU architecture, we propose an imprecise CNU by suppressing the *Detect*_0 signal. Using the notation from the previous section, let us further define:

$$a^*_{m,n} = \begin{cases} a_{m,n}, & \text{if } a_{m,n} = 0, 1, 3 \\ 1, & \text{if } a_{m,n} = 2 \end{cases} \quad (6)$$

The I-POMS decoder computes $|\beta|^{(\text{I-POMS})}_{m,n} = [\textit{AndMsb}^*, \textit{AndLsb}^*]$, with *AndMsb** and *AndLsb** signals computed from $a^*_{m,n}$ values. Note that $|\beta|^{(\text{I-POMS})}_{m,n} \neq |\beta|^{(\text{POMS})}_{m,n}$ only if $a_{m,n'} \geq 2, \forall n'$ and there exist at least one $a_{m,n'}$ value equal to 2. Figure 4 illustrates the implementation of the CNU for the I-POMS decoder. It computes six $|\beta|^{(\text{I-POMS})}_{m,n}$ messages in parallel and can be implemented by only 24 AND gates.

V. Implementation Results

FPGA synthesis results reported in this section are after place and route and have been obtained by using the Xilinx tool ISE 14.6.

TABLE I
CNU HARDWARE RESOURCES FOR MS, OMS ($\lambda = 1$), POMS, AND I-POMS DECODERS ($d_c = 6$)

Decoder	MS	OMS	POMS	I-POMS
Device	Xilinx Virtex 6 (XC6VLX240T-1FF1156) – ISE 14.6			
No. slice LUTs/CNU unit (post PAR)	54	69	42	30
Cost reduction/CNU unit (compared to OMS)	21%	0%	39%	56%

TABLE II
IMPLEMENTATION RESULTS FOR MS, OMS ($\lambda = 1$), POMS, AND I-POMS DECODERS

Decoder	MS	OMS	POMS	I-POMS
Device	Xilinx Virtex 6 (XC6VLX240T-1FF1156) – ISE 14.6			
LDPC code	QC-Regular (3,6), code rate: 0.5, codeword length: 1296			
No. slice registers	23352	23352	23352	23352
No. slice LUTs	95188	97604	93202	90872
No. RAMB36E1	144	144	96	96
No. RAMB18E1	0	0	24	24
Max freq (MHz)	164	122	181	223
Throughput (Gbps)	1.77	1.32	1.95	2.41

Table I shows the hardware (HW) resources required to implement one CNU block for the four decoders under investigation. It can be seen that the CNU architectures proposed for the POMS and I-POMS decoders achieve significant HW resources reduction compared to the conventional CNU architecture used for MS and OMS (*e.g.*, the number of LUTs is reduced by 56%). Full implementation results for the four decoders are shown in Table II. It can be seen that the proposed POMS not only saves 25% memory resources on FPGA, but also increases the throughput by 10% and 48%, compared with conventional MS and OMS, respectively. Since all data is processed in full parallel at each layer, the number of BRAMs may appear to be large, but only 3 among 512 words are used for each BRAM.

We use two different signals to control memory reading and writing in consecutive clock cycles. This significantly improves the maximum operating frequency, compared to using 1 signal for both reading and writing, but also leads to an increased number of registers. Depending on the application, one may trade-off between throughput and area. Decoding throughput is computed by [2]:

$$\text{Throughput} = (\text{CodeLength} \times f_{\text{max}}) \,/\, (N_{\text{iter}} \times N_{\text{cyc}}) \qquad (7)$$

where f_{max} is the maximum operating frequency post place and route, N_{iter} is the maximum number of decoding iterations, and N_{cyc} is the number of clock cycles needed to complete one iteration. The throughput reported in Table II corresponds to $N_{\text{iter}} = 20$, with $N_{\text{cyc}} = 6$ (2 clock cycles per layer).

Figure 5 shows the Bit Error Rate (BER) and Frame Error Rate (FER) performance of the proposed POMS and I-POMS decoders, compared to that of conventional MS and OMS ($\lambda = 1$) decoders. It can be seen that the proposed POMS decoder operates very close to the OMS decoder. For a BER of 10^{-5}, the POMS is at only 0.07 dB from the OMS, while showing a gain of 0.16 dB with respect to the MS decoder. I-POMS performance is also comparable to that of the MS decoder (gap of 0.08 dB), in spite of the impreciseness introduced in the CNU, while the throughput is considerably increased (36%).

VI. CONCLUSIONS

This paper proposed the use of imprecise processing units for the OMS decoder, in order to improve the cost and the throughput of the hardware implementation. Impreciseness has been introduced at two levels: (i) within the offset correction, which is only partially achieved, and (ii) within the minimum computation required by the CNU. We further demonstrated the merits of the proposed

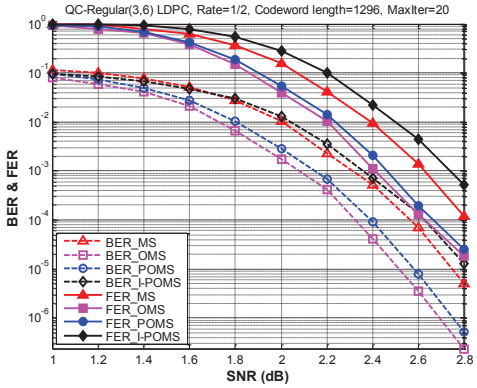

Fig. 5. Decoding performance for proposed algorithms (AWGN channel)

POMS and I-POMS decoders in terms of hardware implementation. Hardware architectures have been proposed for $(3, 6)$-regular QC-LDPC codes, which highlighted the following advantages of our proposal: (i) significant memory and interconnect savings, (ii) simple CNU architecture, requiring only a very small number of AND gates, (iii) increased throughput, up to 2.41 Gbps and (iv) provides good to very good decoding performance, despite the imprecise processing units.

ACKNOWLEDGMENT

This work has been supported by the Franco-Romanian (ANR-UEFISCDI) Joint Research Programme "Blanc-2013", project DIAMOND.

REFERENCES

[1] M. Karkooti and J. R. Cavallaro, "Semi-parallel reconfigurable architectures for real-time LDPC decoding," in *Proc. of Int. Conf. on Inf. Technology: Coding and Computing (ITCC)*, vol. 1, 2004, pp. 579–585.

[2] K. Zhang, X. Huang, and Z. Wang, "High-throughput layered decoder implementation for quasi-cyclic LDPC codes," *IEEE Journal on Selected Areas in Communications*, vol. 27, no. 6, pp. 985–994, 2009.

[3] ——, "A high-throughput LDPC decoder architecture with rate compatibility," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 58, no. 4, pp. 839–847, 2011.

[4] N. Jiang, K. Peng, J. Song, C. Pan, and Z. Yang, "High-throughput QC-LDPC decoders," *IEEE Transactions on Broadcasting*, vol. 55, no. 2, pp. 251–259, 2009.

[5] X. Peng, Z. Chen, X. Zhao, D. Zhou, and S. Goto, "A 115mW 1Gbps QC-LDPC decoder ASIC for WiMAX in 65nm CMOS," in *IEEE Asian Solid State Circuits Conference (A-SSCC)*, 2011, pp. 317–320.

[6] B. Xiang and X. Zeng, "A 4.84 mm^2 847–955 Mb/s 397 mW dual-path fully-overlapped QC-LDPC decoder for the WiMAX system in 0.13 μm CMOS," in *IEEE Symp. on VLSI Circuits (VLSIC)*, 2010, pp. 211–212.

[7] M. Fossorier, M. Mihaljevic, and H. Imai, "Reduced complexity iterative decoding of low-density parity check codes based on belief propagation," *IEEE Trans. on Communications*, vol. 47, no. 5, pp. 673–680, 1999.

[8] J. Chen and M. P. Fossorier, "Near optimum universal belief propagation based decoding of low density parity check codes," *IEEE Trans. on Communications*, vol. 50, no. 3, pp. 406–414, 2002.

[9] M. Xu, J. Wu, and M. Zhang, "A modified offset min-sum decoding algorithm for LDPC codes," in *IEEE Int. Conf. on Computer Science and Information Technology (ICCSIT)*, vol. 3, 2010, pp. 19–22.

[10] M. Jiang, C. Zhao, L. Zhang, and E. Xu, "Adaptive offset min-sum algorithm for low-density parity check codes," *IEEE Communications Letters*, vol. 10, no. 6, pp. 483–485, 2006.

[11] X. Wu, Y. Song, M. Jiang, and C. Zhao, "Adaptive-normalized/offset min-sum algorithm," *IEEE Communications Letters*, vol. 14, no. 7, pp. 667–669, 2010.

[12] D. Hocevar, "A reduced complexity decoder architecture via layered decoding of LDPC codes," in *IEEE Workshop on Signal Processing Systems (SIPS)*, 2004, pp. 107–112.

[13] R. Zarubica, S. G. Wilson, and E. K. Hall, "Multi-Gbps FPGA-based low density parity check (LDPC) decoder design." in *IEEE Global Communications Conference (GLOBECOM)*, 2007, pp. 548–552.