Near-LSPA Performance at MSA Complexity

Joao Andrade, Gabriel Falcao, Vitor Silva
Instituto de Telecomunicações
Dept. of Electrical & Computer Engineering
University of Coimbra, Portugal
Email: {jandrade,gff,vitor}@co.it.pt

Joao P. Barreto, Nuno Goncalves
Institute of Systems and Robotics
Dept. of Electrical & Computer Engineering
University of Coimbra, Portugal
Email: {jpbar,nunogon}@isr.uc.pt

Valentin Savin
CEA-LETI
MINATEC Campus
38054 Grenoble, France
Email: valentin.savin@cea.fr

***Abstract*—The tradeoff between error-correcting performance and numerical complexity of LDPC decoding algorithms is a well-known problem. In this paper we depict the unseen error-floor performance of the Self-Corrected Min-Sum algorithm for long length DVB-S2 codes. We developed a massively parallel simulation using GPUs which allowed a comprehensive BER characterization either in the waterfall or in the error-floor region. We show that the self-correction technique increases the BER performance by 0.5 and 0.2 dB, in the waterfall and error-floor region, when compared to the Min-Sum algorithm. Furthermore, it reaches within 0.2 dB to the Logarithmic Sum-Product BER performance and it also outperforms the Normalized Min-Sum at high SNR, a low complexity decoding algorithm which yields good BER performance.**

***Index Terms*—LDPC codes; Self-Corrected Min-Sum; BER Performance; DVB-S2**

I. Introduction

Low-Density Parity-Check (LDPC) codes were introduced in 1962 [1] and recaptured the attention of the scientific community in 1996 [2], providing a very powerful alternative to Turbo codes due to their capacity-approaching characteristics [3]. They have since been adopted in several digital communication standards. However, even by today's standards, the problem of LDPC decoding still poses a very complex numerical problem. Such complexity combined with bandwidth and QoS requirements compelled engineers and the information theory community to investigate more efficient decoding solutions, such as new LDPC code structures and decoding algorithms capable of achieving very fast decoding times. The project and development of real-time field decoders using Application-Specific Integrated Circuit (ASIC) and Field-Programmable Gate Array (FPGA) technologies provides limited hardware resources. Therefore, strategies need to be adopted that balance a trade-off between decoding complexity and error-correcting capabilities. Typically, sub-optimal decoding, and thus lower error-correcting capabilities, is tolerated if faster decoding is achieved, around a trade-off that is application specific.

The Min-Sum Algorithm (MSA) [4], [5] is a very popular soft-decoding algorithm that presents a fair tradeoff between sub-optimal decoding and numerical complexity. This latter characteristic can be dealt with by addressing the sub-optimality through offset and normalization corrections [4], [5]. However, another promising strategy can be adopted, namely through the self-correction technique, which addresses the MSA sub-optimality by introducing erasure messages into the decoding algorithm [6].

In this paper, we analyse how self-correction can outperform the MSA and reach within a negligible gap of the benchmark decoding algorithm in terms of error-correction for long length irregular LDPC codes. We analyse the numerical complexity of the Self-Corrected Min-Sum Algorithm (SCMSA) against the Logarithmic Sum-Product Algorithm (LSPA), the MSA and the Normalized Min-Sum Algorithm (NMSA). Moreover, we have conducted an error-floor characterization of the SCMSA with a coherent statistical significance.

II. LDPC Codes

LDPC codes are linear block codes characterized by sparse parity-check matrices, which verify the following condition:

$$\mathbf{c} \times \mathbf{H}^{\mathrm{T}} = \mathbf{0}, \ \mathbf{c} \in \mathbf{C}, \tag{1}$$

where $\mathbf{c}$ is a codeword from the set $\mathbf{C}$ of possible codewords. In (1), the codeword set $\mathbf{C}$ is defined as the null-space of $\mathbf{H}$.

$\mathbf{H}$ also defines a bipartite graph, as shown in Figure 1, where rows and columns constitute a node type and the non-null elements, i.e. the bits or Bit Nodes (BNs) participating in a parity-check equation or Check Node (CN), constitute the graph's edges. The following notation will be used throughout the paper regarding LDPC codes:

- N is the coded block length;
- K represents information bits length;
- M=$N - K$ is the number of parity-check equations;
- $R = K/N$ is the coding rate;
- a BN corresponds to a bit in the codeword;
- a CN corresponds to a parity-check equation;
- d_v and d_b are the degree of CNs and BNs respectively, or alternatively the number of non-null elements CNs and BNs possess in the parity-check matrix $\mathbf{H}$;
- edges are associated with non-null elements in $\mathbf{H}$ and connect BNs to CNs;
- γ_n *a-priori* reliability (LLR) BN n;
- α_{nm} message sent by BN n to CN m;
- β_{mn} message sent by CN m to BN n;
- $\hat{\alpha}_n$ *a-posteriori* LLR of BN n.

A. DVB-S2 LDPC Codes

The Digital Video Broadcasting - Satellite 2 (DVB-S2) digital communication standard is one of numerous protocols relying on LDPC codes for their Forward Error Correcting (FEC) systems. These codes define particularly long block

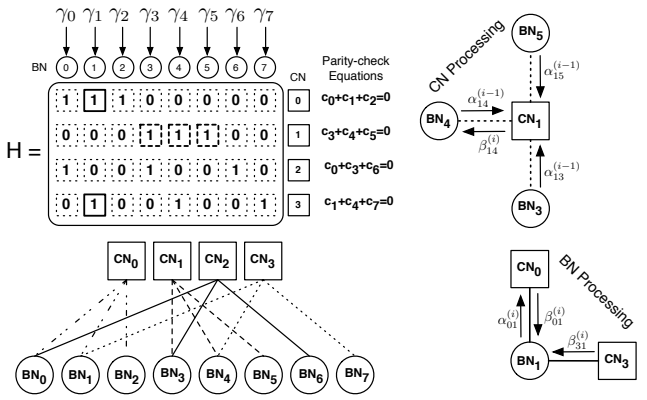

Fig. 1. Example LDPC code, defined by the depicted parity-check restrictions, its corresponding Tanner graph and the message-passing procedure for CN_1 and BN_1.

lengths, with 16200 and 64800 bits for short and normal frame modes, which elects them as suitable candidates for long length case studies. Moreover, they belong to the LDPC Irregular Repeat and Accumulate (LDPC-IRA) class which is prone to fast encoding, and their parity-check matrices exhibit periodical properties which can be exploited to efficiently map the BNs and CNs adjacency to memory [7], [8].

III. Soft-Decoding Algorithms

Soft-decoding algorithms are the elected methods to decode LDPC codes [1], [2], [4], [5] and follow graph-based message-passing algorithms between BNs and CN of the Tanner graph, as illustrated in Figure 1. The Sum-Product Algorithm (SPA) [1], [2] has practical implementation issues related to quantization errors due to the massive use of multiplications, and is too complex for Very Large Scale Integration (VLSI) technology, the prime target of LDPC decoders. Hence, variations of the SPA have been proposed that ameliorate the decoding complexity and ease the decoder project.

A. Logarithmic Sum-Product

The LSPA operates on the log-likelihood domain, since multiplications become additions, and its formalization is presented in Algorithm 1. Approaches *i-iii)* [1], [9], [10] are variations which present different decoding complexities. All of them show dependencies on transcendental functions as seen in Algorithm 1. Approach *iii)* allows the introduction of forward-backward techniques which minimize the number of instructions issued [10] and a sub-optimal approximation of the $\boxplus$ function leads to the MSA.

B. Min-Sum Algorithm

The MSA formalization *iv)*, dismisses the transcendental component of the $\boxplus$ operator in (2), thereby introducing a sub-optimal approximation to the CN processing. This leads to an overestimation of the β messages, due to the $\log$ term in (2) being strictly positive.

$$u \boxplus v = \operatorname{sign}(uv) \times \min\{|u|, |v|\}) - \log\left[\frac{1+e^{-|u+v|}}{1+e^{-|u-v|}}\right] \quad (2)$$

Algorithm 1 Variants of the LSPA and MSA CN processing and BN processing.

CN processing:

i) Gallager approach (ϕ-LSPA):

$$\beta_{mn}^{(i)} = \phi\left(\sum_{n' \in N(m)\setminus n} \phi\left(\alpha_{n'm}^{(i-1)}\right)\right) \times \prod_{n' \in N(m)\setminus n} \operatorname{sign}\left(\alpha_{n'm}^{(i-1)}\right),$$
$$\phi(x) = \log\left(\frac{\exp|x|+1}{\exp|x|-1}\right) \quad (3)$$

ii) tanh approach (tanh-LSPA):

$$\beta_{mn}^{(i)} = 2\,\operatorname{atanh}\left(\prod_{n' \in N(m)\setminus n} \tanh\left(\frac{\alpha_{n'm}^{(i-1)}}{2}\right)\right) \quad (4)$$

iii) Jacobian approach ($\boxplus$-LSPA):

$$\beta_{mn}^{(i)} = \boxplus_{n' \in N(m)\setminus n} \alpha_{n'm}^{(i)} \quad (5)$$

iv) (MSA):

$$\beta_{mn}^{(i)} = \prod_{n' \in N(m)\setminus n} \operatorname{sign}\left(\alpha_{n'm}^{(i-1)}\right) \times \min_{n' \in N(m)\setminus n} |\alpha_{n'm}^{(i-1)}| \quad (6)$$

v) (NMSA):

$$\beta_{mn}^{(i)} = \prod_{n' \in N(m)\setminus n} \operatorname{sign}\left(\alpha_{n'm}^{(i-1)}\right) \times \min_{n' \in N(m)\setminus n} |\alpha_{n'm}^{(i-1)}| \times \Theta \quad (7)$$

BN processing:

i-v):

$$\alpha_{nm}^{(i)} = \gamma_n + \sum_{m' \in M(n)\setminus m} \beta_{m'n}^{(i)} \quad (8)$$

This problem has been previously addressed by offset and normalization corrections [4].

C. Normalized Min-Sum Algorithm

The NMSA addresses the sub-optimality introduced by the negligence of the second term in (2), by multiplying the minimum term with a normalization factor. The optimal normalization factor is Signal to Noise Ratio (SNR) dependent [11], although a considerable Bit Error Rate (BER) performance gain can be achieved by fixing a constant Θ [4] or alternatively by fixing $\Theta_1 = 0.75$ for the absolute minimum and $\Theta_2 = 0.875$ for the second minimum [12]. Alternatively, a strategy that also aims towards the correction of such overestimation can be achieved by self-correction [6].

D. Self-Corrected Min-Sum Algorithm

The SCMSA [6] addresses the β messages overestimation at the BN processing side of the algorithm. Whereas β messages are computed through (8) in the LSPA and MSA-variants, the SCMSA BN processing is defined by (9). The rationale behind the SCMSA is that the overestimation of the β messages

TABLE I
NUMERICAL COMPLEXITY OF THE LSPA-VARIANTS, MSA AND SCMSA FOR CN AND BN PROCESSING.

CN Processing

Algorithm	Mult.	Additions	XOR	Transcendental
ϕ-LSPA in (3)	-	$2d_v$	d_v	$2d_v\phi$
tanh-LSPA (4)	$2d_v$	-	-	$d_v\{\tanh, \text{atanh}\}$
$\boxplus$-LSPA (5)	-	$d_v - 1$	-	$3(d_v-2)\boxplus$
MSA (6)	-	$d_v + \log d_v - 2$	$d_v - 1$	-
NMSA (7)	d_v	$d_v + \log d_v - 2$	$d_v - 1$	-
SCMSA (6)	-	$d_v + \log d_v - 2$	$d_v - 1$	-

BN Processing

Algorithm	Mult.	Additions	XOR	Transcendental
SCMSA (9)	-	d_b	d_b	-
all others	-	d_b	-	-

is not critical unless any given α, through the natural MSA decoding process, goes through a signal change, which in the log-likelihood ratio domain corresponds to a bit state change. Thus, in the SCMSA, an α message signal change is followed by the introduction of an erasure (9).

$$\tilde{\alpha}_{mn}^{(i)} = \gamma_n + \sum_{m' \in M(n)\setminus m} \beta_{m'n}^{(i)}$$

$$\alpha_{mn}^{(i)} = \begin{cases} 0, & \text{sign}\{\tilde{\alpha}_{mn}^{(i)}\} \times \text{sign}\{\alpha_{mn}^{(i-1)}\} < 0 \wedge \alpha_{mn}^{(i-1)} \neq 0 \\ \tilde{\alpha}_{mn}^{(i)}, & \text{otherwise.} \end{cases} \quad (9)$$

An erasure in the log-likelihood ratio domain is expressed by a zero message, i.e. bit states 0 and 1 are equiprobable, and due to the CN processing nature, which will update erasures to adjacent BNs, several erased messages will be given to BNs adjacent to the CN that received the erasure. Upon receiving an erased message, the SCMSA acts similarly to the MSA, i.e. messages erased do not remain erased for more than one iteration as the SCMSA processes β messages differently according to their magnitude, as formalized in (9).

IV. NEAR-OPTIMAL BER PERFORMANCE AT LOW NUMERICAL COMPLEXITY

The BER performance improvement achieved by the SCMSA is very promising regarding its implementation effort.

A. LDPC Decoding Numerical Complexity

The different numerical complexities of the CN and BN processing of the LSPA-variants and the MSA-variants are highlighted in Table I [5]. The MSA is very appealing, due to the non-dependence on transcendental functions. However, the degrading BER performance is severe enough to justify the use of offset or normalized variations of the MSA [5]. These variants would imply d_v subtractions or d_v multiplications per CN. On the other hand, the proposed SCMSA solution requires only an additional d_b exclusive-or operations and a comparator to properly introduce the erasure as seen in Figure 2.

B. Practical Implementation

Figure 2 depicts a schematic that illustrates the differences between the MSA and the SCMSA datapaths. The SCMSA

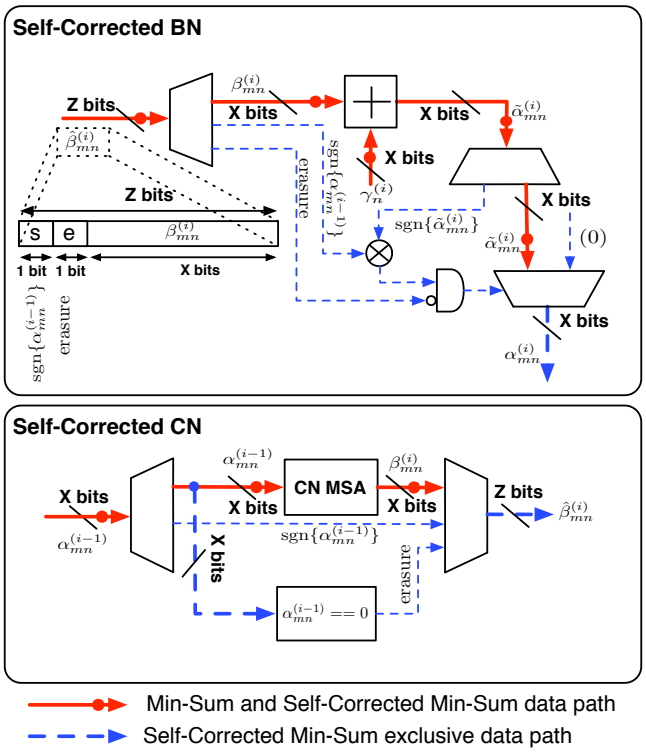

Fig. 2. Memory-efficient computation of the self-corrected CN and BN processing for a possible hardware implementation. The datapaths of both the MSA and the SCMSA are presented. In the top figure the BN processing is depicted, where the self-correction takes place. As shown, only $Z = X + 2$ bits per computed $\alpha_{mn}^{(i)}$ have to be loaded, since the $\hat{\beta}_{mn}^{(i)}$ message contains both the relevant $\beta_{mn}^{(i)}$ message, $\text{sign}\{\alpha_{mn}^{(i-1)}\}$ and the information regarding whether an erasure has been introduced in the previous iteration. This is possible because the self-corrected CN processing variant has packed both $\text{sign}\{\alpha_{mn}^{(i-1)}\}$ and the erasure bit e into $\hat{\beta}_{mn}^{(i)}$. This allows the SCMSA to be implemented for the same memory requirements of the MSA, if the former employs Y bits fixed-precision for the α and β messages and the latter $(Y-1)$ bits for α messages and $(Y+1)$ bits for β messages.

dependency of the $\text{sign}\{\alpha_{mn}^{(i-1)}\}$ in (9) can be dealt with in different ways for software or hardware decoders. The former, targeted at simulation and prototyping, are less constrained by memory impediments as opposed to hardware-based decoders. Thus, equation (9) can be evaluated by loading the actual $\alpha_{mn}^{(i-1)}$ message from memory. In the hardware, memory bottlenecks seriously compromise the decoding execution time, and thus a packed message $\hat{\beta}_{mn}^{(i)}$ is proposed, which stores $\text{sign}\{\alpha_{mn}^{(i-1)}\}$, $\beta_{mn}^{(i)}$ and an erasure bit e together in the same message. The latter is used in the BN processing to inform whether an erasure has been introduced or not, thus avoiding the loading of $\alpha_{mn}^{(i-1)}$ messages.

Thus, an efficient implementation, for both software or hardware platforms, of the SCMSA requires only two extra bits when compared to other LSPA- or MSA-variants.

V. EXPERIMENTAL RESULTS

A. Apparatus

The experiments conducted simulated an Additive White Gaussian Noise (AWGN) channel under Quadrature Phase-

Shift Keying (QPSK) modulation for the DVB-S2 normal frame code with $N = 64800$ bits and $R = 1/2$. The MSA, NMSA and SCMSA simulations were carried out on Nvidia Tesla M2050 Graphics Processing Unit (GPU) platforms using the Compute Unified Device Architecture (CUDA) programming model [13]. The LSPA decoder was simulated using the Simulink Communication Blockset. This allowed not only to provide the BER performance benchmark, but also to provide a benchmark simulation time to illustrate the potential of using GPU engines in the LDPC BER simulation.

B. Simulation Environment

Simulating on the GPU represents a tremendous improvement in simulation time. Namely, the SCMSA GPU single precision floating-point decoder can compute the BER for a given SNR in about 5 hours for 10^4 simulated codewords, whereas the equivalent LSPA Simulink decoder takes 22 hours to complete on a ubiquitous Intel Pentium M T2330 in double precision floating-point. We have employed 7.2 and 8.2 fixed-point data representations ($x.y$ fixed-point data representation should be read as $(x - y)$ bits allocated for signal and magnitude representation and y bits for the decimal part). The considerable decoding throughputs (120 to 180 MBit/s) are obtained due to the high levels of data-parallelism and task-parallelism that the massively multithreaded GPU-decoder exploits. Data-parallelism has been exploited by the use of 128-bit vector data types which optimize the GPU achieved memory bandwidth and allows to decode 16 codewords at once. Furthermore, we have employed a Two-Phased Message Passing (TPMP) decoding schedule which allows to express a fine-grained level of parallelism, where each thread is responsible to process a BN or a CN.

Naturally, for the error-floor region, 10^4 codewords cannot provide statistical significance and in many cases it is not deep enough to observe any errors at all. Thus, the error-floor region was tested for 5×10^8 codewords, which is equivalent to 3.2×10^{13} bits of data. In order to simulate such large datasets we have employed 30 GPU engines to collaboratively and concurrently speed up the decoding time. Each data point is simulated by a group of 10 GPUs. The same random sequence can be sampled starting at any given k-th sample due to the employed Random Number Generator (RNG) XORWOW, available in the CUDA CURAND library [14], and thus each GPU can be allocated to process the corresponding portion of the sequence. Furthermore, the XORWOW RNG presents a period of $2^{190} \approx 10^{57}$. This allowed the simulation time to be reduced from a colossal one and a half month to just over one and a half day per data point in the error-floor region.

C. Waterfall BER Performance

The decoding performance, expressed in BER, on the waterfall region is shown in Figure 3 for several decoding algorithms and data representations. The LSPA obtains the best decoding performance, since it is an "optimal" decoding algorithm in the sense that no sub-optimality is introduced, besides the one inherent to the cycle effect, and it is used as a benchmark to the remaining algorithms. Namely, the SCMSA yielded a BER which approaches that of the LSPA by ~ 0.2 dB. The MSA presents a performance degradation of ~ 0.8dB, which is the result of the overestimation introduced in (2). Consequently, the SCMSA is ~ 0.6dB better than the MSA and worse by ~ 0.2dB than the LSPA. It should be noted that there is no significant penalty in BER performance on the waterfall region between the 7- and 8- bit precision fixed-point data representation for the SCMSA. Hence, the two bits saved in the 7-bit representation, one per each α_{mn} and β_{mn} message when compared to the 8-bit fixed-point, can be used for the erasure e and signal $\text{sign}\{\alpha_{mn}^{(i-1)}\}$ representation, meaning that the actual memory requirements of the 8-bit MSA and the 7-bit SCMSA are the same. Another expected result is the ability of the SCMSA decoder in floating-point data representation to outperform by 0.05 dB the 7- and 8-bit fixed-point decoders, and reach within a 0.15 dB of the LSPA BER performance. In fact, arguably 0.15 dB is a negligibly small penalty to incur when numerical complexity is tremendously lowered and decoding times reduced by a considerable factor. However, the ability to capitalize on this result is at most limited as VLSI technology for LDPC decoding is better targeted for fixed-point data representations. Furthermore, the SCMSA floating-point decoder has a similar BER performance than the NMSA, albeit the latter performing 0.05 dB better than the fixed-point SCMSA decoders.

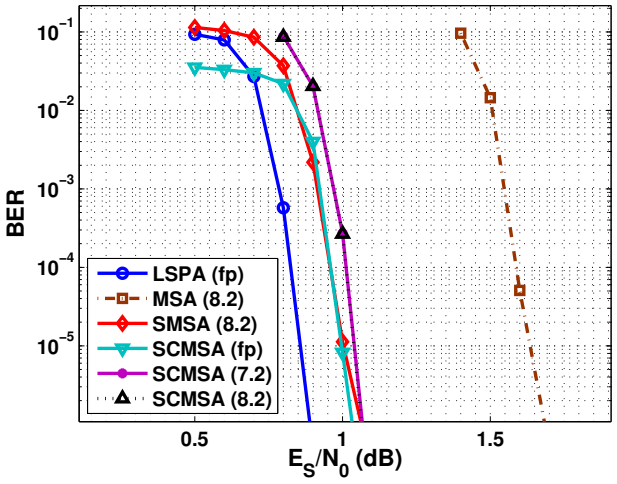

Fig. 3. LSPA, MSA, NMSA and SCMSA BER performance on the waterfall region. The self-correction technique used by the SCMSA allows an approach of just ~ 0.2 and ~ 0.05 dB away from LSPA and the NMSA BER performance, respectively. It should be noted that losing one bit in the precision does not carry any performance penalty on the BER, as seen with the tested 7.2 and 8.2 fixed-point implementation of the SCMSA, where $x.y$ designates $(x - y)$ bits for signal and integer representation and y bits for decimal representation. Therefore, the curves for 7.2 and 8.2 bits are hardly distinguishable from one another.

D. Error-floor BER Performance

The error-floor region is depicted in Figure 4. Given the constraints on the GPU available timeslots, we were able to

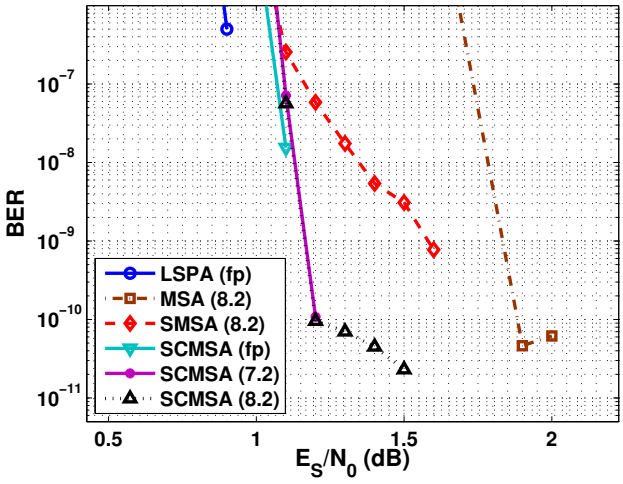

Fig. 4. Captured MSA and SCMSA error-floor BER. Both the decoding algorithms converge to a BER of 10^{-11}, with the SCMSA reaching its error-floor 0.5 dB earlier than the MSA. The NMSA starts degrading its decoding performance at 1.2 dB and is outperformed by the fixed-point SCMSA.

find at least 3 error-floor points for the SCMSA and 2 for the MSA algorithm, each using 8-bit data representation. We have shown that the SCMSA lowers to its error-floor 0.5 dB earlier than the non-self-corrected MSA. Such a feature is particularly interesting as it indicates that the SCMSA does not degrade its performance along the waterfall and maintains the same capacity up to the error-floor region. Regarding the NMSA BER curve, it is observed that the fixed-point SCMSA is able to outperform it when the BER falls beyond 10^{-7} and while the NMSA worsens its decoding performance at a SNR of 1.2 dB, the SCMSA enters its error-floor at a BER of 10^{-11}. Despite employing Θ values that were optimized for this particular LDPC code [12], the NMSA correcting performance degrades for a SNR above 1.2 dB. The alternative of employing Θ values depending on the SNR would further add to the complexity of the FEC system, since in the log-likelihood domain the decoding algorithms are independent of the channel SNR.

VI. Conclusions

We have shown that a practical and memory efficient implementation of the SCMSA is feasible and achieves near-LSPA performance, 0.1 dB away for floating-point precision and 0.2 dB away for 7- and 8-bit fixed-point representation. Furthermore, the SCMSA also significantly outperforms the NMSA at high SNRs. The proposed implementation requires only two extra bits per $\hat{\beta}_{mn}$ message. Moreover, in order to demonstrate the promising SCMSA BER performance, we have developed a massively parallel channel simulator and decoder for the $N = 64800$ bits, $R = 1/2$ DVB-S2 LDPC code. The developed SCMSA fixed-point decoding solution was able to simulate the error-floor region of long length DVB-S2 LDPC codes in an acceptable timespan, guaranteeing statistical significance for the obtained results. Aided by a higher-level of data-parallelism provided by several GPU engines, we were able to execute a one and a half month simulation time in just over one day. The developed simulators are available to the scientific community at http://montecristo.co.it.pt/decoder.

In the near future, we intend to port the developed simulators to a generic parallel programming model, which would allow these simulators to be executed on a wide range of Central Processing Units (CPUs), GPUs and other accelerators, as it has been shown with good performance for the particular case of regular LDPC decoding [15].

Acknowledgment

This work was supported by Instituto de Telecomunicações and by Fundação para a Ciência e Tecnologia grants PEst-OE/EEI/LA0008/2011 and SFRH/BD/78238/2011 under the European structural programmes POPH-QREN, FSE and national MEC funding.

References

[1] R. G. Gallager, "Low-Density Parity-Check Codes," *IRE Transactions on Information Theory*, vol. 8, no. 1, pp. 21–28, jan 1962.

[2] D. MacKay and R. Neal, "Near Shannon limit performance of low density parity check codes," *Electronics Letters*, vol. 32, pp. 1645–1646, 1996.

[3] 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.

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

[5] J. Chen, A. Dholakia, E. Eleftheriou, M. P. C. Fossorier, and X.-Y. Hu, "Reduced-Complexity Decoding of LDPC Codes," *Communications, IEEE Transactions on*, vol. 53, no. 7, p. 1232, july 2005.

[6] V. Savin, "Self-corrected Min-Sum decoding of LDPC codes," in *Information Theory. ISIT 2008, IEEE International Symposium on*, july 2008, pp. 146 –150.

[7] F. Kienle, T. Brack, and N. Wehn, "A Synthesizable IP Core for DVB-S2 LDPC Code Decoding," in *DATE '05: Proceedings of the conference on Design, Automation and Test in Europe*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 100–105.

[8] G. Falcao, J. Andrade, V. Silva, and L. Sousa, "GPU-based DVB-S2 LDPC decoder with high throughput and fast error floor detection," *Electronics Letters*, vol. 47, no. 9, pp. 542–543, April 2011.

[9] W. Leung, W. Lee, A. Wu, and L. Ping, "Efficient implementation technique of LDPC decoder," *Electronics Letters*, vol. 37, no. 20, pp. 1231 –1232, sep 2001.

[10] X.-Y. Hu, E. Eleftheriou, D.-M. Arnold, and A. Dholakia, "Efficient implementations of the sum-product algorithm for decoding LDPC codes," in *Global Telecommunications Conference, 2001. GLOBECOM '01. IEEE*, vol. 2, 2001, pp. 1036 –1036E vol.2.

[11] R. Lucas, M. Fossorier, Y. Kou, and S. Lin, "Iterative decoding of one-step majority logic deductible codes based on belief propagation," *IEEE Transactions on Communications*, vol. 48, no. 6, pp. 931–937, 2000.

[12] C.-j. Tsai and M.-c. Chen, "Efficient LDPC decoder implementation for DVB-S2 system," *Proceedings of 2010 International Symposium on VLSI Design, Automation and Test*, pp. 37–40, Apr. 2010.

[13] NVIDIA, *CUDA C Programming Guide 5.0*. NVIDIA, 2012.

[14] NVIDIA, *CUDA Toolkit 4.2 CURAND Guide*. NVIDIA, 2012.

[15] G. Falcao, V. Silva, L. Sousa, and J. Andrade, "Portable LDPC Decoding on Multicores Using OpenCL [Applications Corner]," *IEEE Signal Processing Magazine*, vol. 29, no. 4, pp. 81–109, 2012.