International Journal of Computer Applications (0975 - 8887)
Volume 61 - No. 19, January 2013

A New PWL Approximation for the "Self-Adjustable Offset Min-Sum" Decoding with a highly Reduced-Complexity

Abdessalam Ait Madi
LSSC Sidi Mohammed
Ben Abdellah University
Faculty of Sciences and Technolgy
B.P, 2202, Fez, V.N 30000 Morocco

Anas Mansouri
LSSC Sidi Mohammed
Ben Abdellah University
National School of Applied Sciences of Fez

Ali Ahaitouf
LSSC Sidi Mohammed
Ben Abdellah University
Faculty of Sciences and Technolgy
B.P, 2202, Fez, V.N 30000 Morocco

ABSTRACT

A new PieceWiseLinear (PWL) function is proposed for the decoding of the Low-Density-Parity-Check (LDPC) code with the Self Adjustable Offset Min-Sum (SAOMS) algorithm.
Avoiding the use of the non-linear function $f(x) = \ln\left(1 + e^{-|x|}\right)$ in the adjustable offset factor, this linear approximation greatly simplifies the hardware implementation with a little BER performance loss.
The proposed solution is new, useful and is successfully tested on decoding regulars (504, 252) and (8000, 4000) LDPC codes. The corresponding Check Node Processing Unit (CNPU) have been designed, described and simulated using Very High Speed integrated circuits Hardware Description language(VHDL). The synthesis results were obtained using an Altera Quartus II software with cyclone II EP2CF896C6 as the target FPGA device.
The designed CNPU is fully parallel and flexible to be used for different block length when a regulars (3, 6) LDPC codes are required.

Keywords:

LDPC , BER, FPGA, BP, PWL, VHDLifx

1. INTRODUCTION

Low-density parity-check (LDPC) code is an Error Correcting Code (ECC) which is used in highthrouput applications such the Digital Video Broadcasting for the second generation digital satellite and terrestrial television broadcasting system (DVB-S2)[1], (DVB-T2)[2].
The LDPC codes [3]can be efficiently decoded with the original belief-propagation (BP) algorithm or the sum-product algorithm (SPA). The BP algorithm is recognized by its good error correction performance [4] but needed many multiplications to update the check and variable nodes which make it difficult to use for efficient hardware implementation.
Several modified versions of the BP decoding algorithm have been introduced such log-likelihood ratio belief propagation (LLR-BP) in which the multiplications are transformed in additions. In spite of this considerable simplification, the LLR-BP still shows some difficulties for digital circuit's implementation. In fact it need the Look-Up Table (LUT) to implement respectively the hyperbolic tangent and the involution transform function $-\ln(tanh(x))$ in the *tanh-rule* and the Gallager's approach LLR-BP based algorithm.
However the use of the LUTs introduces quantization effects leading to some performance degradation. Some non LUT-based approach has been developed. Among them, the Min-Sum (MS) algorithm [5, 6] which greatly reduces both the computational and the hardware implementation complexity, is the main approximated version. Despite these achievements in the implementation improvement, this simplification introduces non-negligible performance degradation due to the overestimation in the outgoing message from the check to variable node. Ever since, many approaches have been proposed leading to the development of many reduced-complexity variants of the BP algorithm to trade off between the BER performance and hardware complexity. The Normalized Min-Sum (NMS) and the Offset Min-Sum (OMS) [6] are two kinds of schemes used to correct the overestimation by using a fixed multiplying factor or a fixed additive offset factor. The optimal factor can be estimated by simulation or by the so-called Density Evolution (DE) [7].
In order to reduce more the performance degradation, several other approaches have been proposed, such the Degree Matched Min-Sum (DMMS)[8] in which the degree of the check node and offset factor are associated and the Adaptive Offset Min-Sum (AOMS)[9] which adapts the offset factor according to the minimum magnitude output from the check node. Among the other works, the Self-Adjustable Offset Min-Sum algorithm (SAOMS) have been proposed for the Integrated Service Digital Broadcasting via Satellite-Second Generation (ISDB-S2) LDPC decoder [10]. This works shows that the offset value is mainly dependent on the difference between the second and the first minimum magnitude of the incoming messages to the check node, thus the offset value is made adjustable during the iterative decoding process. A performances comparison of the BP, NMS, OMS, DMMS, AOMS and SAOMS algorithms shows that the latter is an efficient candidate for the ISDB-S2[10]. Then, it was proposed that the non-linear function $f(x) = \ln\left(1 + e^{-|x|}\right)$ should be computed to update the check nodes. As a result both the computational complexity and the hardware resources were increased when compared to the those required for the OMS and the NMS algorithms [10].This non-linear function is then approximated in hardware by using the LUTs or sometimes split into regions in which the behavior can be considered as linear.
Motivated by this challenge we propose a new PWL function to transform and replace $f(x)$.This non-linear function is then split on five pieces with a negligible performances loss. This novel approximation provides high BER performance with almost the same area and the computational complexity when compared to the MS. In addition, it can reduce computational complexity and hardware cost design when compared to LLR-BP algorithm.
The PWL transformation approximates the continuous function linearly, using line segments to split the non-linear function in the required range. The objective is to estimate a section of $f(x)$ and to approximate it by a straight line. We divide the function in a number of segments knowing that a smaller number of segments will be simple to implement in hardware with eventually some loss in performance. Some previous PWL approximation

have been proposed with a performances loss cost,as in the case of two linear regions [11], and with a hardware cost for the case of eight regions proposed by Hu et al. [12].
To validate the proposed approach, a new VHDL CNPU design is proposed and simulated. Besides, both of the LUT-based approach and the PWL with two regions CNPU are also designed in the aim of comparison. The comparison of the synthesis results were obtained using an Altera Quartus II software with cyclone II EP2CF896C6 as the target FPGA device.
This paper is organized in the following manner. Section 2 presents the proposed approximation and its computation complexity. In the section 3 the simulation results are shown. The CNPU architecture and its FPGA implementation is presented in section 5 and 6. Finally in the section 7 concludes the paper.

2. PROPOSED APPROXIMATION

The check node update equation, in SAOMS algorithm [10] is given by

$$L_m = \prod_{n' \in N(m) \setminus n} sign(Z_{n'}) \max\left(\min_{n' \in N(m) \setminus n} (|Z_{n'}|) - \beta_o, 0 \right) \quad (1)$$

Where

- $N(m) \setminus n$, is the set of indexes "n'" corresponding to the other Variable Nodes (VNs) connected to the Check Node (CN) "m" with $n' \neq n$.
- L_m is the message coming from the CN, "m", to VNs, "n".
- $Z_{n'}$ the message coming from the VNs, "n' ", to the CN, "m"
- β_o is the adjustable offset factor given by

$$\beta_o = \gamma' f(Z_{min2} - Z_{min1}) \quad (2)$$

where γ' is the normalization factor, chosen equal to 1.25 [10] and Z_{min1} and Z_{min2} are the first and second minimum magnitude of the incoming messages given by

$$Z_{min1} = \min_{n' \in N(m)} (|Z_{n'}|) \quad (3)$$

$$Z_{min1} = \min_{n' \in N(m) \setminus min1} (|Z_{n'}|) \quad (4)$$

and $f(Z_{min2} - Z_{min1}) = \ln\left(1 + e^{-|Z_{min2} - Z_{min1}|}\right)$. This function is respectivelly implemented with LookUp Table (LUTs) with quantization effects or directly with a high hardware cost.
In our approach, five intervals have been limited and in each one the function has been replaced by a linear function $a_i x + b_i$. The different regions and their corresponding functions are grouped in table 1.
In this table, the linear function with the power of two of each a_i are given. These power of two representations, avoid the use of multiplications and uses only shifts and additions instead, which highly reduces the computation and hardware complexity. Note that all the numbers are represented in fixed-point two's complement with 5 bits for the fractionnal part. Figure 1 shows respectively the plots of the function and the proposed PWL approximation. One can clearly notice the good agreement between the two functions.
To validate our approach, we first proceed by a computation complexity comparison. For that we use the optimal normalization factor $\gamma' = 1.25 = 2^0 + 2^{-2}$ chosen to optimize both the BER performance and hardware implementation [10] . To update a CN node one proceeds as follows: For the MS algorithm

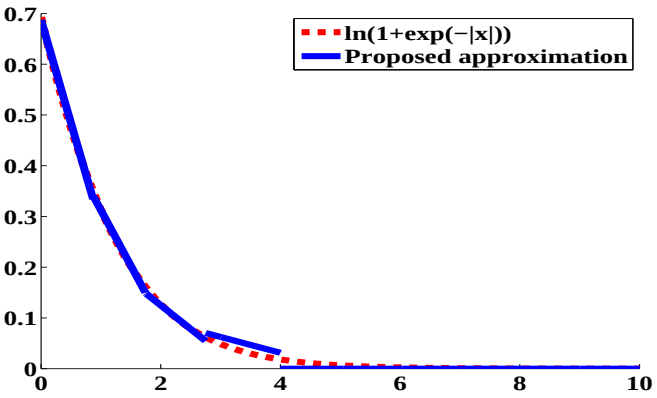

Fig. 1. The non-linear function $f(x)$ and the corresponding five PWL function curves.

$(d_c - 1)$ and $(d_c - 2)$ items are compared serially to seek for the first and the second minimum values Z_{min1} and Z_{min2}. Besides another comparison is added to choose which minimum value to be used in equation (1) instead of $\min_{n' \in N(m) \setminus n}(|Z_{n'}|)$. Note that d_c is the degree of a given CN node. For the SAOMS algorithm with the PWL approximation (only two regions) proposed in [10], one need $2(d_c - 1) + 1$ comparisons, 5 additions, 4 rights shift. The first $2(d_c - 1)$ comparisons are performed in the same way as in the MS algorithm and the last one is used to select which of 0 or $\left(-\frac{1}{4}x + \frac{5}{8}\right)$ has to be used instead of $f(x)$ calculation. One addition is used to calculate the $x = Z_{min2} - Z_{min1}$ and two other to subtract the offset factor from Z_{min1} and Z_{min2}. Another addition and the two shifts are used to perform the product $\gamma' f(x) = 1.25 f(x) = (2^0 + 2^{-2}) f(x)$.The last addition and the two remaining shifts are required to calculate $-\frac{1}{4}x + \frac{5}{8} = -2^{-2}x + 0.625$ used to perform $f(x)$. For the SAOMS algorithm with a LUT approximation, one need for the same number of comparisons, as in MS algorithm, 4 additions, 2 rights shift and an access to the LUT table. In such case, the same operations are performed in the same order as in SAOMS except that the addition and the two shifts to perform $f(x)$ are replaced by a LUT. The access to the LUT is used to read the value of the non-linear function $f(x)$ for $x = Z_{min2} - Z_{min1}$. In the proposed algorithm, to update the same CN node, we need $2(d_c - 1) + 4$ comparisons in the worst case ($x > 4$); however we don't need more than 3 additions and 5 shifts (in the worst case) to determine the value of $f(x)$ rather than the use of the LUT. This computation complexity comparison, for the four algorithms, is summarized in the table 2.

3. SIMULATION RESULTS

A regulars $(504, 252)$ and $(8000, 4000)$ LDPC code check matrices H downloaded from Mackay's online database [14] are used in the software simulation. The fixed degree node of each VN and CN are three and six, respectively. Encoded bits are binary-phase-shift-keying (BPSK) modulated and transmitted over the simulated Additive White Gaussian Noise (AWGN) channel. The number of maximum iterations is set to 16 at each Signal to Noise Ratio (SNR) Eb/N0 value. The normalization factor γ' for both SAOMS and the proposed algorithm is fixed to 1.25. For each SNR value, codeword of length 504 and 8000 have been transmitted respectively, until we have 20000 errors or 2000 codeword transmitted. The BER is the ratio between the number of occurred errors and the number of the transmitted bits. The simulation program halts when the decoded word is valid or if the maximum number of iterations is reached.In those simulations we use the classical flooding schedule to update the check nodes and variable nodes.

Table 1. Piecewise linear function approximation for $f(x) = \ln(1 + e^{-|x|})$

Region	PWL function
$[0, 0.875]$	$-(2^{-2} + 2^{-3} + 2^{-5})x + 0.6875$
$[0.875, 1.75]$	$-(2^{-3} + 2^{-4} + 2^{-5})x + 0.53125$
$[1.75, 2.75]$	$-(2^{-4} + 2^{-5})x + 0.3125$
$[2.75, 4]$	$-2^{-5}x + 0.15625$
$[4, +\infty\ [$	0

Table 2. Results of the comparison of the computation complexity

Algorithm	Comparisons	Additions	Shifts	Others
MS	$2(d_c - 1)$	–	–	–
SAOMS with PWL [10]	$2(d_c - 1) + 1$	5	4	–
SAOMS with LUT	$2(d_c - 1)$	4	2	$1LUT$
Proposed	$2(d_c - 1) + 4$	7	7	–

Figure 2 shows the comparison between the proposed PWL approximation and the original SAOMS algorithm. It is clearly seen that there is no more performance degradation for both the regulars $(504, 252)$ and $(8000, 4000)$ LDPC codes.
Figure 3 shows the comparison performance between the SAOMS with the three approximations: the proposed PWL , PWL with two regions and LUT.
For the (8000,4000) LDPC code decoding, One can see clearly that, in high SNR, the SAOMS algorithm with the proposed approximation have the better performance than one simulated with the two others approximations.
As a result the proposed PWL is better and have nearly exactly the same performances as the original algorithm. The proposed solution can be easily implemented in hardware when compared to the original algorithm.

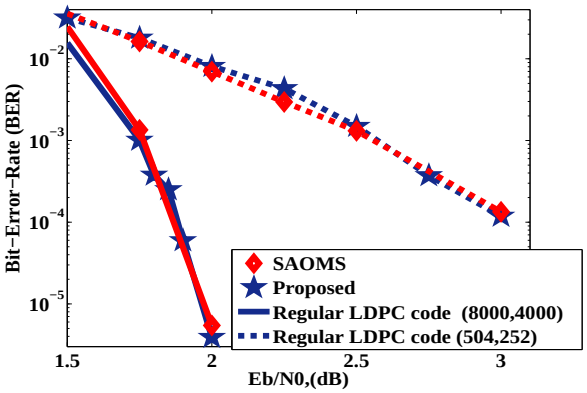

Fig. 2. Performances of a regular $(504, 252)$ and $(8000, 4000)$ LDPC codes with the use of the proposed PWL, a normalization factor γ' is set to 1.25.

4. ARCHITECTURE FOR THE CN PROCESSING UNIT FOR A REGULAR (3,6) LDPC CODE

This module receives 6 inputs Z_i (i=1 to 6) and 6 outputs L_i (i=1 to 6). To compute the message L_i (i=1 to 6) , given by equation(1) in hardware, we separate the operation into three main steps, sign calculation of the outgoing message L_i (i=1 to 6) and the two minimum magnitudes calculation of the incoming messages according to the equations (3) and (4) and finally, the determination of the Adjustable Offset Factor (AOF) according to the equation(2).
To construct a modular CNPU, the set of blocks will be instantiated in parallel manner to carry out the 6 messages L_i (i=1

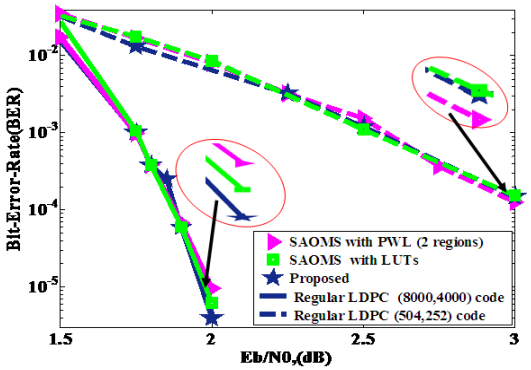

Fig. 3. Performances of a regular $(504, 252)$ and $(8000, 4000)$ LDPC codes decoded by the SAOMS algorithm with the use of the proposed PWL, PWL with two regions and LUTs approximations, a normalization factor γ' is set to 1.25.

to 6) at the same time. Figure 4 shows a global block diagram of the CNPU unit, which implements the check node equation according to the equation (1). The *comp_mag* computes in a parallel manner the minimum magnitudes (Z_{min1} or Z_{min2}) corresponding to the 6 inputs Z_i (i=1 to 6). From the sixth inputs of this CNPU, the sign block computes the signs of the 6 outputs L_i. The two minimum magnitudes Z_{min1} and Z_{min2} are carried out from the *min_search* block.
The difference between Z_{min2} and Z_{min1} is performed by using one adder and one unary minus operators. The final adjustable offset factor (AOF) is computed by using the *shift_adder* block. This block uses shifts and additions operations to perform the multiplication between the *comp_nlfx* output and the normalization factor $\gamma' = 1.25$. The AOF is thereafter subtracted from the *comp_mag* outputs by using *six_adders* block to perform $\min_{n' \in N(m) \setminus n}(|Z_{n'}|) - \beta_o$ given in equation (1). If the result is negative the *max_in_zero* yield the null output, otherwise it yield the computed result. The outputs L_i (i=1 to 6), with the appropriate sign, are selected from *max_in_zero* or from *six_unary* by using the *six_mux* taking into account the results of *sign* block.
The used blocks in this design works in fixed-point two's complement format with a fixed word length set to 9 bits: The MSB bit corresponds to the sign, the next three MSBs bits represent the integer part and the five remaining bits represent the fractional part. Thus, all the data will be delimited in the [min, max] interval where min=-8 and max=7.96875 in 9 bits fixed point format. Consequently, if a value computed by the block is outside this interval, it will be immediately replaced by the mim (respectively max) value.

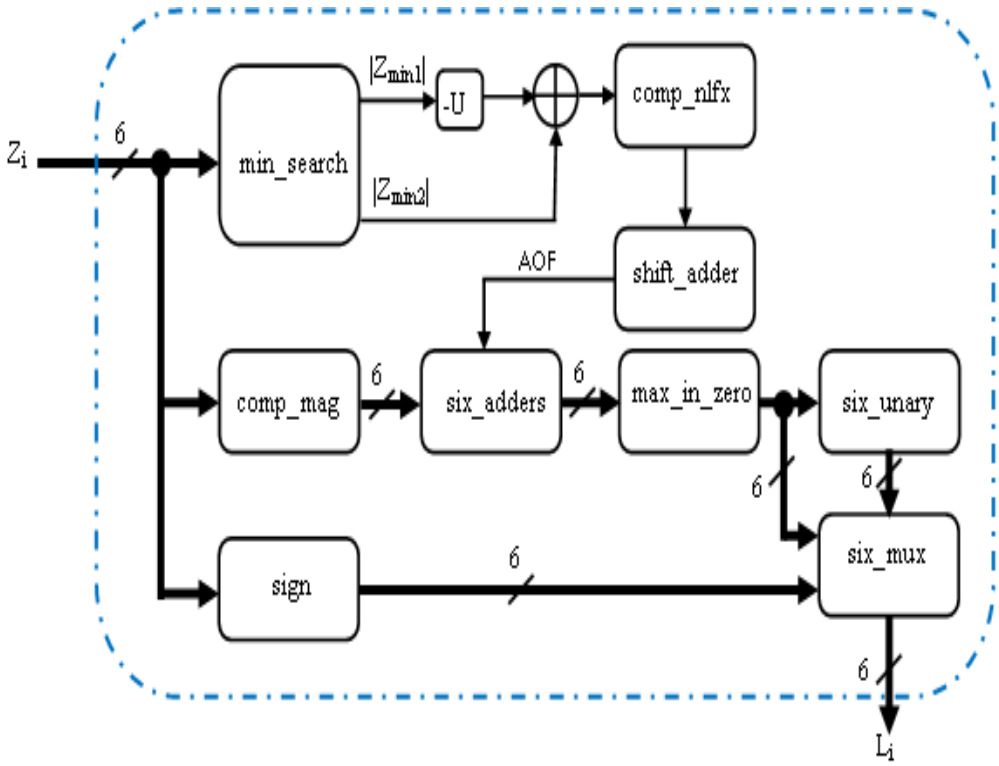

Fig. 4. Bloc diagram of the proposed CNPU in parallel configuration.

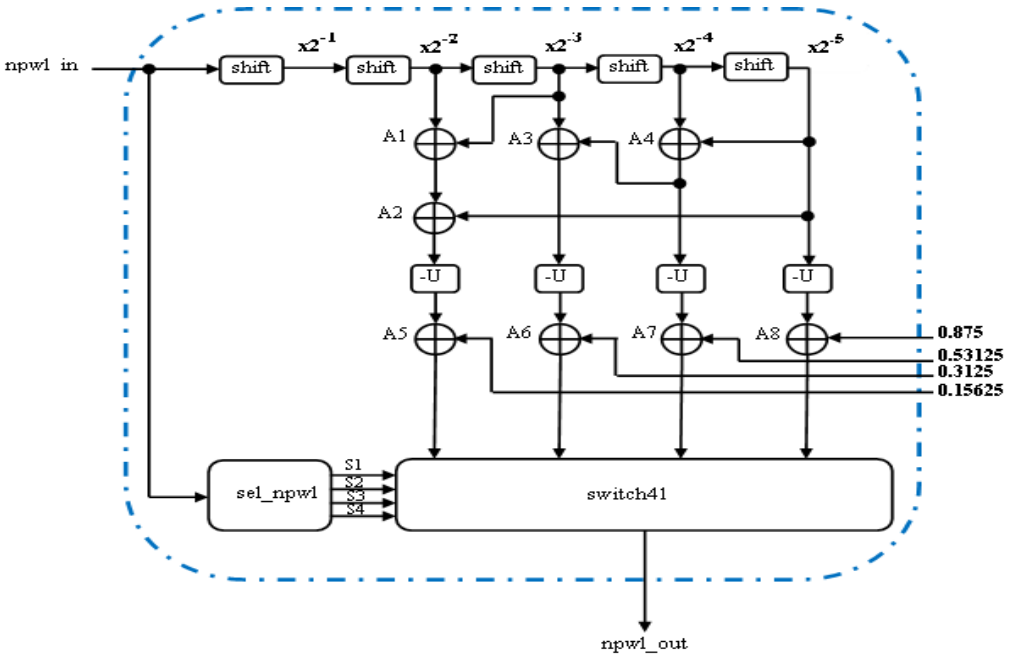

Fig. 5. Bloc diagram of the proposed approximation to replace $f(x)$ with a PWL .

Table 3. The selected PWL according to *sel_npwl* block functionality

npwl_in	S1	S2	S3	S4	Selected PWL
≤ 0.875	1	0	0	0	$-(2^{-2}+2^{-3}+2^{-5})x+0.6875$
$0.875 >$ et ≤ 1.75	0	1	0	0	$-(2^{-3}+2^{-4}+2^{-5})x+0.53125$
$1.75 >$ et ≤ 2.75	0	0	1	0	$-(2^{-4}+2^{-5})x+0.3125$
$2.75 >$ et ≤ 4	0	0	0	1	$-(2^{-4}+2^{-5})x+0.3125$
> 4	0	0	0	0	0

In our case, the block *comp_nlfx* in figure 4 is replaced by another block using the proposed PWL. Its corresponding block diagram is presented in figure 5. The single input *npwl_in* of this block is the difference between the second and the first minimum magnitudes Z_{min2} and Z_{min1}. The fifth shifters block are combined with four adders *(A1, A2, A3 et A4)* and four unary minus operators to perform multiplications by the fourth slopes coefficients used in the first fourth PWL functions shown in the table 1. To add the y-intercepts 0.875, 0.53125, 0.3125 and 0.15625 to the results, four additional adders *(A5, A6, A7 et A8)* are used. The appropriate PWL is selected via a *switch41* block controlled by the *sel_npwl* block according to the table 3. If the input *npwl_in* is greater than 4, the outputs of *sel_npwl* are set to zero and thereby the *switch41* output drive zero. If not, one of *sel_npwl* output is set to one to select the appropriate linear function, to be used according to the input value.

5. VHDL IMPLEMENTATION OF THE PROPOSED CNPU

5.1 implementation

The work consists of transforming the two proposed designs into computerized representation by using the VHDL code. The blocks used in these designs are described and tested individually following a modular approach. After checking the eventual errors and warnings in each module, all the elements were wired to produce the required top design. All the used blocks were instantiated in a parallel manner to implement the CNPU top design given in Figure 4.

5.2 synthesis

The synthesis results for the proposed design is obtained using an Altera Quartus II software with Cyclone II EP2C70F896C6 as the FPGA target device. The SAOMS with LUT and PWL(with only two regions) uses 907 and 945 FPGA's total logic elements respectively when the proposed algorithm with a new PWL uses 1081. This small increase of the area cost can easily justified by the performances amelioration in the proposed solution. The maximum frequency, after timing simulation, close 178 MHz by using one pipelined processing stage.

6. FPGA IMPLEMENTATION VALIDATION

In this hardware implemention functionality validation, the decoder prototype designed in this paper is based on the regular matrix (10,5). The decoder have 10 VNs and 5 CNs. The corresponding Variabe Node Processing Units (VNPUs) and CNPUs, tested and validated individualy in ModelSim, are wired to copy exactly the Tanner graph [13].

6.1 Decoder performance within FPGA device

The Log LikeLihood Ratios (LLRs) values generated by a Matlab program and stored in hexadecimal file are loaded into input Megafunction Altera RAM:1 port. This LLRs has been applied to the decoder. The last one itertes until finding valid codeword or a maximum iteration is reached. The decoded words are written into output Megafunction Altera RAM:1 port and thereafter written into hexadecimal file. The BER is obtained, in Matlab, by comparing the code words, generated by the Matlab coding program, and the FPGA decoded words.
The performance results for both implemented decoders in Matlab and VHDL codes are presented in figure 6. At SNRs 6 dB below, the hardware decoder and Matlab floating point are in complete agreement. At higher SNR's (6.5 dB and above) the Matlab floating point outperforms the FPGA implementation due to superior precision in a floating point representation.
The performance comparison prove that the design has been correctly implemented and run onto FPGA device.

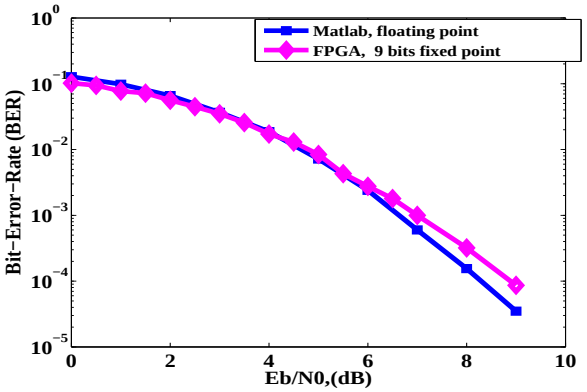

Fig. 6. Performances of a regular $(10, 5)$ LDPC codes decoded by the SAOMS algorithm with the use of the proposed PWL, implemented in Matlab and FPGA device. a normalization factor γ' is set to 1.25 and the maximum of iterations is equal to 16.

7. CONCLUSION

In this paper, we propose to transform and replace $f(x)$ by a new PieceWise Linear function to improve the decoding with the previous Self Adjustable Offset Min-Sum (SAOMS).
Avoiding the use of the non-linear function $f(x) = \ln\left(1 + e^{-|x|}\right)$ in the adjustable offset factor, this linear approximation achieve a good balance between the Bit-Error-Rate (BER) performances and the hardware implementation. The simulations results validate the effectiveness of the proposed approximation. The last one is useful and it was successfully tested on decoding regulars (504, 252) and (8000, 4000) LDPC codes.
The designed CNPU is fully parallel and flexible to be used for different block length when a regulars (3, 6) LDPC codes are required.

8. REFERENCES

[1] Digital Video Broadcasting (DVB).2004. Second generation framing structure, channel coding and modulation systems for Broadcasting, Interactive Services, News Gathering and other broadband satellite applications. Draft ETSI EN 302 307 V1.1, European Standard (Telecommunications series).

[2] Digital Video Broadcasting (DVB).2009. frame structure, channel coding and modulation systems for second generation digital terrestriel television broadcasting system (DVB-T2). Draft ETSI EN 302 755 V1.1.1 , European Standard (Telecommunications series).

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

[4] D.J.C.Mackay. Good Error-Correcting Codes Based on very Sparse Matrices. IEEE Transaction on information Theory. Res. 45(Jan. 1999), 399-431.

[5] M.P.C.Fossorier,M.Mihaljevic,H.Imai. Reduced Complexity Iterative Decoding of Low Density Parity Check codes based on Belief Propagation. IEEE Transaction on Communications. 47(May. 1999), 673-680.

[6] J.Chen,A.Dholakia,E.Eleftheriou,M.Fossorier,X.Y.Hu. Reduced-Complexity Decoding of LDPC codes. IEEE Transaction On Communications. 53(Aug. 2005), 1288-1299.

[7] J.Chen, M.P.C.Fossorier. 2003. Density evolution for BP-based decoding algorithms of LDPC codes and their quantized versions. In Proceedings of the IEEE GlOBCOM'02.

[8] S. L. Howard, C. Schlegel, and V. C. Gaudet. Degree-matched check node decoding for regular and irregular LDPCs. IEEE Transaction on Circuits and Systems II: Express Briefs. 53(Oct. 2006), 1054-1058.

[9] M. Jiang, C. Zhao, L. Zhang, and E. Xu. Adaptive offset min-sum algorithm for low-density parity check codes. IEEE Communications Letters, 10(2006), 483-485.

[10] W.Ji,M.Hamaminto,H.Nakayam,S.Goto. Self-Adjustable Offset min-sum Algorithm for ISDB-S2 LDPC Decoder. IEICE Electronis Express. 7(Aug. 2010), 1283-1289

[11] M.M. Mansour,N.R. Shanbhag. High-Throughput LDPC Decoders. IEEE Transaction on very large integration systems. 11(Dec. 2003), 976-996.

[12] X.Yu.Hu, E.Eleftheriou, D.M.Arnold, A.Dholakia. 2002. Efficient Implementation of the Sum-Product Algorithm for Decoding LDPC Codes. In Proceedings of the IEEE GlOBCOM'01.

[13] R.M.Tanner. A recursive Approach to Low Complexity Codes. IEEE Transaction on information Theory. IT-27(Sep. 1981), 533-547

[14] D.J.C. MacKay, Online database of low-density parity-check codes,http:// www.inference.phy.cam.ac.uk/mackay/ CodesFiles.html