

# A Generic Architecture of CCSDS Low Density Parity Check Decoder for Near-Earth Applications

Fabien Demangel, Nicolas Fau, Nicolas Drabik  
R-interface  
Marseille Innovation BP 20038  
13302 Marseille Cedex 03, France  
demangel,fau,drabik@r-interface.com

François Charot, Christophe Wolinski  
Irisa, Inria  
University of Rennes 1, Campus de Beaulieu  
35042 Rennes Cedex, France  
charot,wolinski@irisa.fr

## Abstract

*Low Density Parity Check (LDPC) codes have recently been chosen in the CCSDS standard for uses in near-earth applications. The specified code belongs to the class of Quasi-Cyclic LDPC codes which provide very high data rates and high reliability. Even if these codes are suited to high data rate, the complexity of LDPC decoding is a real challenge for hardware engineers. This paper presents a generic architecture for a CCSDS LDPC decoder. This architecture uses the regularity and the parallelism of the code and a genericity based on an optimized storage of the data. Two FPGA implementations are proposed: the first one is low-cost oriented and the second one targets high-speed decoder.*

## 1. Introduction

The Consultative Committee for Space Data Systems (CCSDS) was formed in 1982 by the major space agencies of the world to provide a forum for discussion of common problems in the development and operation of space data systems. It has been actively developing Recommendations for data- and information-systems standards that represent the most performing algorithms within the bounds of the current technology. Concerning the error correcting codes that are key elements of the quality of communication systems, the CCSDS has introduced some new recommendations for the near-earth applications, also known as CCSDS C2 standard [2, 1].

Since their discovery by Gallager in the 60's [6], and their rediscovery in the mid 90's [7], the Low Density Parity Check (LDPC) codes have been the focus of many researches that have resulted in the emergence of a new class of powerful error correcting codes. A recent development of these researches has resulted in the appearance of the Quasi-

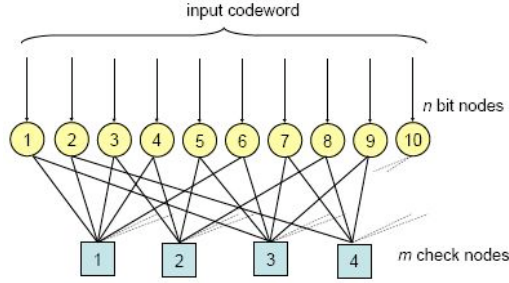
Cyclic LDPC codes, a subclass of LDPC codes with a very low error floor achieved with a very fast iterative convergence. These qualities make these codes good candidates for near-earth applications where very high data rates and high reliability are the driving requirements. Until now the implementations of the LDPC codes were restricted by their output rates, and the FPGA logics and memories they need.

In this paper, after presenting the theoretical aspect of the CCSDS LDPC code, we propose optimizations for a generic architecture, and implementation of two differently oriented decoders. Then, the FPGA implementations of the architectures and their performances are detailed and discussed.

## 2. Low Density Parity Check Codes

A  $(n, k)$  Low Density Parity Check Code is a linear block code, whose  $n$ -bits codewords must satisfy  $m = n - k$  parity check equations. These  $n - k$  equations can be depicted as a Tanner graph (a bipartite graph) that is composed of two kinds of nodes. The nodes drawn as circles in the Figure 1 represent the bits of the codeword, they are called bit nodes (BN). The nodes drawn as squares represent the parity check equations, called check nodes (CN). When the  $i^{th}$  parity check equation involves the  $j^{th}$  codeword bit, there is an edge in the graph between the  $j^{th}$  bit node and the  $i^{th}$  check node. The number of edges connected to a node is called its degree. The connectivity between these two types of nodes is specified in the standard.

The codes can also be represented by a sparse parity check matrix  $H$  of size  $m$ -by- $n$ , where  $n$  is the length of the code and  $m$  is the number of parity-check bits in the code, specifying the parity-check constraints of the bits in the codewords. Each row of  $H$  corresponds to a parity check. If an edge of the bipartite graph links check node  $i$  and bit node  $j$ , element  $h_{i,j}$  of  $H$  is equal to 1, otherwise element  $h_{i,j}$  is equal to 0. In the rest of this paper this rep-



**Figure 1. Tanner Graph of a LDPC code**

resentation will be preferred.

### 2.1. LDPC Codes Decoding Algorithm

There exists many ways to decode LDPC codes [5], the difference between them is the choice of the trade-off between computation time and reliability. All of them are based on the principle of message passing algorithm which means that the data processed are the reliability of the received bits, expressed as Log likelihood ratios. The more commons of these algorithms are the belief propagation (BP) algorithm and the sum-product (SP) algorithm. In a classical decoding process, one iteration of the algorithm is composed of four steps. In the first, all messages are sent from all  $BN$  nodes over corresponding edges of the bipartite graph to all  $CN$  nodes. In the second step all  $CN$  nodes process the data. In the next step all  $CN$  nodes send back the messages over the same corresponding edges of the bipartite graph to the  $BN$  nodes. The iteration is completed by the processing performed by all  $BN$  nodes. The iterations are repeated a given number of times which corresponds to the decoding period. The LDPC decoder achieves good performance with the so called BP or SP based algorithms. The BP-based algorithm operates as follows. Let  $bc_{n \rightarrow m}$  denote the message sent from by the  $BN$  node  $n$  to the  $CN$  node  $m$  and let  $cb_{m \rightarrow n}$  denote the message sent from the  $CN$  node  $m$  to the  $BN$  node  $n$ . The check node update for each iteration of the algorithm, using the simplification proposed in [5] is defined by :

$$cb_{m \rightarrow n_i} = g(bc_{n_0 \rightarrow m}, bc_{n_1 \rightarrow m}, \dots, bc_{n_{dc-1} \rightarrow m}) \quad (1)$$

$$g(x, y) = \frac{\text{sign}(x) \cdot \text{sign}(y) \text{Min}(|x|, |y|)}{\alpha} \quad (2)$$

where  $dc$  is the degree of the check node, it is to say the number of bit nodes connected to the considered check node and  $\alpha$  is a normalization factor greater than one.

The bit node update for each iteration of the algorithm ( $M(n)$  representing the edges linked to the  $n^{\text{th}}$  BN and  $u_n$

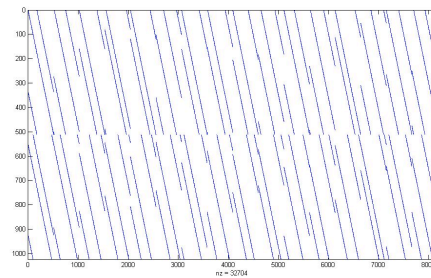
is the incoming LLR) is defined by:

$$bc_{n \rightarrow m_i} = u_n + \sum_{m_j \in M(n)} cb_{m_j \rightarrow n_i} - cb_{m_i \rightarrow n_i} \quad (3)$$

For further details concerning LDPC codes refer to the bibliography.

### 2.2. CCSDS Parity Check Matrix for Near-Earth Applications

The LDPC code considered in the CCSDS specification is a member of a class of codes called Quasi-Cyclic (QC) codes. The construction of these codes involves juxtaposing smaller circulants (or cyclic) submatrices to form a larger parity check or base matrix. The entire circulant matrix is uniquely determined and specified by its first row or column. The CCSDS LDPC code is a shortened code based



**Figure 2. Parity Check Matrix**

on a (8176, 7156) LDPC code formed by using a  $2 \times 16$  array of  $511 \times 511$  square circulants. This creates a parity check matrix of dimension  $1022 \times 8176$ . The row weight of each of the 32 circulants is two; i.e., there are two '1's in each row. The total row weight of the parity check matrix is  $2 \times 16$ , or 32. The column weight of each circulant is also two. The total column weight of the parity check matrix is four. A scatter chart of the parity check matrix is shown in Figure 2 where every '1' bit in the matrix is represented by a point.

The circulant property due to this construction of the parity check matrix produces two positive features. The first is that it reduces the encoder complexity which is linear to the number of parity bits. The second is that it reduces both the encoder and decoder routing complexities in the interconnections of integrated circuits.

The rows of the parity matrix represent the check node updates and the columns the bit node updates. Even if the regularity of the code simplifies a part of the implementation, during the decoding, more than 32k messages (corresponding to the '1's values in the parity matrix) are updated at each iteration. Consequently an optimized scheduling for the message passing and a good storing of data are needed.

### 3. Generic Architecture for LDPC Decoder

We designed a generic parallel architecture (see [3]) optimized for FPGA components which is the basis of two kinds of LDPC decoders. The first decoder is a low cost one assuring 130 Mbps data throughput while the second decoder is a high-speed version enabling 1040 Mbps data throughput (table 1). The design of the second architecture was possible thanks to the flexibility of the proposed architecture.

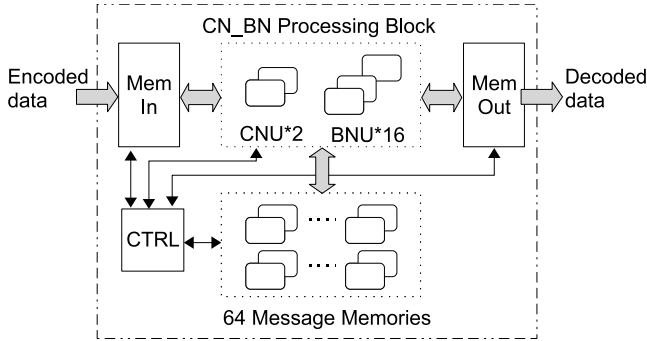


Figure 3. Base Parallel Architecture

As shown in Figure 3, the base architecture of the decoder consists of a controller, input/output memories, multi-block message memories (in the BP-based algorithm messages are stored in message memories) and a processing block containing many instances of the CN node and BN node processing units. As described in Section 2, the challenge of decoding CCSDS C2 LDPC codes comes from the high number of messages to deal with. In our low cost solution, we process 16 BN (/2 CN) concurrently thanks to the regularity and the parallelism of the QC LDPC code. The generic design consists in the use of several processing blocks and memory blocks with a larger word size (the messages corresponding to the different input frames are stored in the same memory word and are accessed concurrently) within the base architecture depicted in Figure 3.

### 4. Implementation

We have developed a generic synthesizable VHDL IP that is fully compliant with the CCSDS C2 standard. The generic architecture described in Section 3 can implement a low-cost and a high-speed version of a LDPC decoder. The IP has been mapped on two different Altera FPGA targets as mentioned below.

Even if the decoders have different data rates, the performances of the architecture in terms of errors correction are maintained. It depends mainly on the numbers of iterations required to converge. This parameter is programmable and its relative performances are detailed in Section 5. It has a

Number of iterations	Low-Cost Decoder Output Throughput	High-Speed Decoder Output Throughput
10	130 Mbps	1040 Mbps
18	70 Mbps	560 Mbps
50	25 Mbps	200 Mbps

Table 1. Number of iterations influence on the output data rate of LDPC decoders with a clock frequency of 200 MHz

direct impact on the throughput of the decoder. The Table 1 gives the results for different numbers of iterations and a system clock frequency of 200 MHz for both decoder versions. We can see that eighteen iterations is a good trade-off between error correction and output throughput (see the performances results Figure 4).

#### 4.1. Low-Cost LDPC Decoder

We have implemented the base architecture on an Altera low-cost FPGA, a Cyclone II EP2C50F. It will be a good target for low-cost decoders because of its limited resources compared to high-performance FPGA.

The Table 2 summarizes the implementation results. We can see that the optimization of the architecture impacts on the logic cells requirement (less than 10k ALUTs and registers). We note that only 50% of the total memory space is necessary.

ALUTs	Registers	Total Memory Bits
8k(16%)	6k(12%)	290k(50%)

Table 2. Implementation results of the low-cost decoder on a Altera Cyclone II EP2C50F

With eighteen iterations and a system clock frequency of 200 MHz, the decoding output throughput is 70 Mbps. Compared to other known implementations of CCSDS C2 LDPC decoder (as in [2] [1]), this decoder is lowest-cost with better performances at the same data rate (Section 5). The small amount of resources required (Table 2) for decoding LDPC codes with a 70 Mbps output data rate also justifies the “low-cost decoder” term.

#### 4.2. High-Speed LDPC Decoder

A high-speed decoder can be designed with the generic architecture. In terms of data rate, this decoder is eight times more efficient than the low-cost version. Because of the resources required, this high-speed version has to be implemented on a more powerful FPGA than the Cyclone II, for example an Altera Stratix II EP2S180. The Table 3 summarizes the results of this implementation.

The genericity of the proposed architecture allows to increase the output throughput of the decoder by a factor

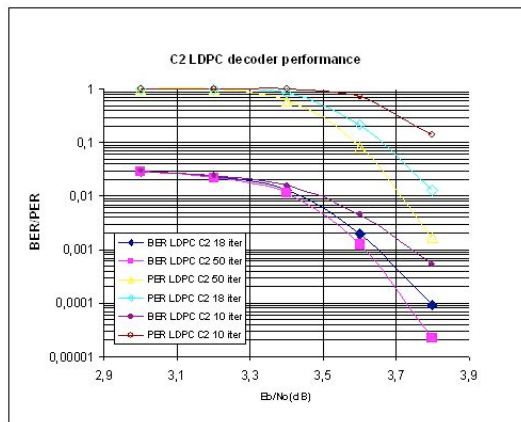
of eight while only increasing the amount of resources by about four. Especially for the memories which are more optimized and more filled. As we can see in the Table 1 for eighteen iterations and a system clock frequency of 200 MHz, the decoding output throughput is 560 Mbps. This high data-rate makes this decoder very compliant with the CCSDS LDPC specification for near-earth applications. The performances of the error correction, which are the same for the two decoders, are described in the following section.

ALUTs	Registers	Total Memory Bits
38k(27%)	30k(20%)	1300kb(20%)

**Table 3. Implementation results of the high-speed decoder on a Altera Stratix II EP2S180**

## 5. Performances

Our implementation of the C2 LDPC code, in addition to be fast, achieves good bit and packet error rates (Figure 4) without facing any high error floor issues. Actually our decoder achieves BER and PER which are 0.05dB better than the CCSDS FPGA tests results in [2]. In terms of implementation it means that we can achieve the same performances, with 18 iterations instead of 50. The speed and the reliability of the decoder being mainly related to the number of iterations, the Table 1 and the Figure 4 highlight the fact that 18 iterations represent the best trade-off.



**Figure 4. Bit and Packet error rate LDPC decoder performance**

These good results have been obtained thanks to the use of a fine scaled correction factor [4]. The role of this factor is to reduce the degradation of the information carried by the LLR, introduced by the sign min simplification. Thus

the key idea is to find the factor which minimizes the difference between the means of the messages passed in the BP algorithm and the sign-min algorithm.

## 6. Conclusion

In this paper, we have presented a generic architecture that can be useful to implement two kinds of decoder for CCSDS C2 LDPC codes. We have designed the base architecture in order to realize a low-cost LDPC decoder with an output throughput of 70 Mbps which minimizes resources. We have also implemented a generic version of this architecture targeting a high-speed LDPC decoder which has a high output data-rate of 560 Mbps. We have seen that these LDPC decoders have good error correction. Both of these decoders are fully compliant with the CCSDS C2 LDPC standard but the high-speed version is more adapted to the high data-rates required in near-earth applications.

Future work will consist in applying the principles of this generic parallel architecture to other CCSDS recommendation such as the several rates AR4JA LDPC codes for deep-space applications.

## References

- [1] *Progress on LDPC Codes at JPL*. CCSDS Spring Meeting - NASA Data Standards Working Group, Mar. 10-14 2008.
- [2] *Orange Book, Experimental specifications, Low Density Parity Check Codes For Use In Near-Earth And Deep Space Applications CCSDS 131.1-O-2*. CCSDS, Sept. 2007.
- [3] F. Charot, C. Wolinski, N. Fau, and F. Hamon. A New Powerful Scalable Generic Multi-Standard LDPC Decoder Architecture. In *Proceedings of the 16th Annual IEEE Symposium FCCM 2008*, Stanford, Palo Alto, CA, April 2008.
- [4] J. Chen and M. Fossorier. Density evolution for two improved bp-based decoding algorithms of ldpc codes. *IEEE Comm. Letters*, Vol.6 : 208–210, May 2002.
- [5] J. Chen and M. Fossorier. Near optimum universal belief propagation based decoding of low-density parity check codes. *IEEE Trans. Commun.*, Vol.50 : 406–414, Mars 2002.
- [6] R. G.Gallager. Low density parity check codes. *Cambridge, MA, M.I.T. Press*, 1963.
- [7] D. MacKay. Good error-correcting codes based on very sparse matrices. *IEEE Trans. Inform. Theory*, Vol.45 : 399–431, Mars 1999.