A Parallel and Modular Architecture for 802.16e LDPC Codes

François Charot, Christophe Wolinski, Nicolas Fau, François Hamon

A Parallel and Modular Architecture for 802.16e LDPC Codes

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

Nicolas Fau, François Hamon
R-Interface
Marseille Innovation BP 20038
Pôle Média Belle de Mai
13302 Marseille Cedex 03
{fau,hamon}@r-interface.com

Abstract

We propose a parallel and modular architecture well suited to 802.16e WiMax LDPC code decoding. The proposed design is fully compliant with all the code classes defined by the WiMax standard. It has been validated through an implementation on a Xilinx Virtex5 FPGA component. A four or six-module FPGA design yields a throughput ranging from 10 to 30 Mbit/s by means of 20 iterations at a clock frequency of 160 MHz which mostly satisfies communication throughput in the case of the WiMax Mobile communication.

1 Introduction

Low density parity-check (LDPC) are linear block codes. They have recently been included as error correcting codes in several new communication standards. A codeword of an (n, k) LDPC code must satisfy $m = n - k$ parity check equations on its n codeword bits. The whole set of $(n - k)$ equations can be depicted by means of a bipartite graph (see figure 1), composed of two kinds of nodes: *bit nodes* (*BN*), representing the bits of the codeword and *check nodes* (*CN*), representing the parity check equations. It 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.

The hardware realization of an LDPC decoder is determined by many strongly interrelated parameters, leading to a large design space and various implementations [1, 4, 7, 3]. For a fully parallel hardware realization, each node is instantiated and the connections between them are hardwired. Even for relatively short block length, severe routing congestion problems occur. Therefore partly parallel architectures (where the nodes of only one subset at a time are simultaneously processed) become mandatory. The sizing of such architectures with regards to the constraints to be satisfied (throughput performance, amount of hardware resources) requires an architectural exploration phase based on methodologies and tools to solve complex optimization problems.

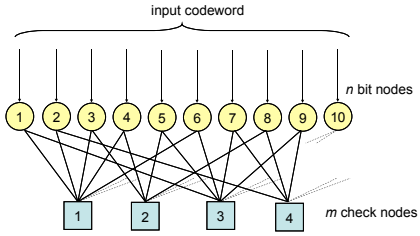

Figure 1. Bipartite graph of a LDPC code.

The parallel and modular architecture well suited to LDPC code decoding presented in this paper is made up of several processing modules communicating through an optimized interconnection structure. Each processing module includes two processing units (called bit node and check node), and a set of memory banks. The size of the architecture – number of modules, number of interconnection buses, size and number of memory banks – is both communication standard and throughput dependent. The size of the architecture in the case of a given standard and a given throughput can be established during the space exploration process thanks to our optimization system based on a constraints programming approach.

The paper is structured as follows. Our decoding LDPC algorithm and its performance are summarized in section 2. Our architecture is outlined in section 3. Implementation results of a WiMax LDPC decoder are presented in section 4. Finally some conclusions are given in section 5.

2 LDPC Code Decoding algorithm

The LDPC decoder achieves good performance with the so called BP or SP based algorithms [5]. We consider BPSK *(Binary Phase Shift Keying)* modulation under the AWGN *(Additive White Gaussian Noise)* channel. The channel

978-0-7695-3277-6/08 $25.00 © 2008 IEEE
DOI 10.1109/DSD.2008.81

model is:

$$r_n = s_n + v_n$$

where r_n is the received noisy symbol, s_n = +/-1, the transmitted BPSK symbol and v_n is an additive white Gaussian noise with variance $\sigma^2 = N_O/2$. The log-likelihood ratio of bit n is:

$$u_n = (2/\sigma^2).r_n$$

The BP-based algorithm operates as follows. Let $v_{n\to m}$ denote the message sent from by the bit node n to the check node m and let $w_{m\to n}$ denote the message sent from the check node m to the bit node n.

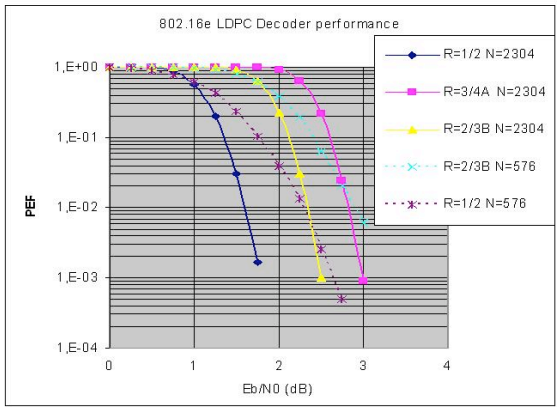

Figure 2. 802.16e LDPC decoder performance.

The check node update for each iteration of the algorithm is defined by:

$$w_{m\to n_i} = g(v_{n_0\to m}, v_{n_1\to m}, ..., v_{n_{d_c-1}\to m})$$

where dc is the degree of the check node, it is to say the number of bit nodes connected to the considered check node.

$$G(x,y) = sign(x).sign(y)Min(\mid x \mid, \mid y \mid) + ln(1+e^{-|x+y|}) - ln(1+e^{-|x-y|})$$

In [2] the following simplification is proposed:

$$g(x,y) = sign(x).sign(y)Min(\mid x \mid, \mid y \mid)/\alpha \qquad (1)$$

Where α is a normalization factor greater than one.

The bit node update for each iteration of the algorithm is defined by:

$$v_{n\to m_i} = u_n + \sum_{m_j \in M(n)} w_{m_j\to n} - w_{m_i\to n} \qquad (2)$$

The iterations are completed applying the following rule on the codeword:

$$\text{hard decision} \begin{cases} 0, & \text{if } (v_{n\to m_i} + w_{m_i\to n}) \geq 0 \\ 1, & \text{if } (v_{n\to m_i} + w_{m_i\to n}) < 0 \end{cases}$$

The figure 2 illustrates the decoder performance in terms of packet error rate (PER) when applied to the 802.16e LDPC codes. The results are given for 6-bit LLR quantization and 20 decoding iterations. Two code lengths are considered: n=576 bits and n= 2384 bits with four coding rates: r=1/2, 2/3, and 3/4 and 5/6.

3 Proposed Decoding Architecture

In order to ensure at the same time flexibility and the demanded throughput, partly parallel architectures are mandatory. Instead of exploiting the inherent parallelism in the sub-matrices of the H parity-check matrix as in [6], we propose to have a global approach to the problem by performing the calculation of independent CN or BN nodes in parallel. In this section, the exploited data decomposition is first explained. We then focus on the computation distribution and the memory organization and show how bit node and check node computations are performed.

Our approach consists in clustering nodes of the whole set of check nodes (CN nodes) (respectively bit nodes, BN nodes) into independent sets called CS_j (respectively BS_j) according to the features of the parity check matrix H.

$$CS_j = \{CN_i \mid 24.j \leq i \leq 24.(j+1) - 1\} \quad 0 \leq j < 11$$
$$BS_j = \{BN_i \mid 24.j \leq i \leq 24.(j+1) - 1\} \quad 0 \leq j < 23$$

The left part of the figure 3 indicates the bit nodes required to compute two sets of check nodes: CS_0 (from CN_0 to CN_{23}) and CS_1 (CN_{24} to CN_{47}). It is shown that in order to compute CN_0 belonging to set CS_0, a message comes from nodes BN_{47}, BN_{66}, BN_{205}, BN_{236}, BN_{289} and BN_{312}. These six bit nodes belong respectively to sets BS_1, BS_2, BS_8, BS_9, BS_{12} and BS_{13}. The right part of the figure shows that BN_{47} belonging to set BS_1 also has an edge with check node CN_{41} belonging to set CS_1.

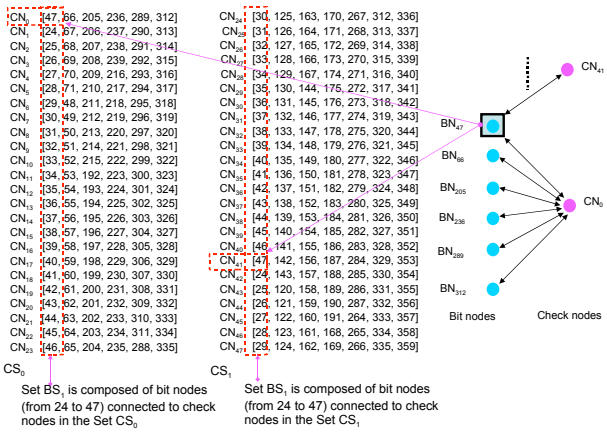

Figure 3. Dependency relations between bit nodes and check nodes.

All nodes of a given CS or BS set are processed in the same computation unit called a *module*. All the nodes of the set are then processed sequentially. According to the degree of parallelism of the architecture (number of modules), nodes of different CS sets (respectively BS) can be processed in parallel on different *modules*.

The architecture is illustrated in figure 4. It is made up of several processing modules communicating through an optimized interconnection structure. Each processing module includes two processing units (called bit node and check node), and a set of memory banks. The size of the architecture– number of modules, number of interconnection buses, size and number of memory banks– is both communication standard and throughput dependent.

Each *module* has its own local memory used for storing the exchanged messages between check nodes and bit nodes during the computation process (corresponding to the edges of the bipartite graph). Each local memory is composed of several memory banks allowing parallel accesses to be performed. The basic idea is that all components of a message can be accessed simultaneously.

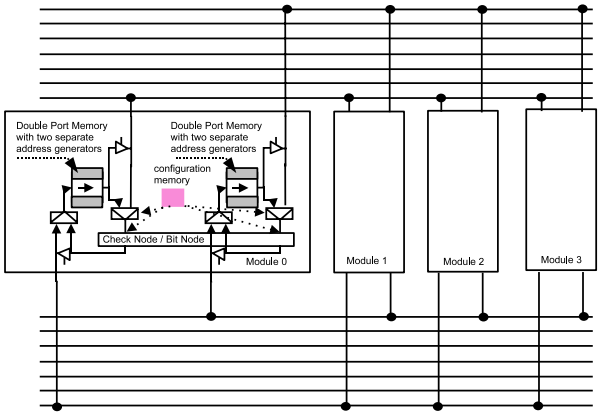

Figure 4. Organization of the parallel architecture.

Each module is able to process check nodes as well as bit nodes. In the case of check node processing, the data comes from their local memories. The local memories of the module contain all the messages coming from the related BN sets. After the processing accomplished in the module, results are stored in the same local memories. In the case of bit node processing, the data can come from local or non-local memories according to the corresponding parity matrix. In the case of non-local memory, a data transfer through the interconnection structure is performed.

The interconnection structure is composed of several buses (B_i). B_i allows the data reception from any memory bank $MB_{i,k}$ of any module M_k, or the data sending to any $MB_{i,k}$.

The number of buses, the data distribution over the memory banks and the buses selection to a data transfer are optimized in order to speedup the application execution. This is done thanks to our optimization system based on a constraints programming approach. The optimization system is not described in the paper. However some results obtained with the system are presented in the next section.

Figure 5 shows the memory organization in the case of a four-module architecture (case $r = 1/2$). The 12 check nodes sets that have to be calculated are distributed between the 4 modules at the rate of 3 sets per module. Six computation steps (called scenarii) allow all bit node sets to be processed.

4 Case Study

Table 1 summarizes the results of the architecture exploration phase which allows different configurations of the architecture to be compared. For each code ratio of the WiMax LDPC standard, the table gives, for a given number of modules, an optimal solution in terms of CN set quantity per module, bus quantity of the interconnection structure, scenario quantity required for the computation of the BN sets and memory size required to store the messages during the decoding.

Table 1 shows that a parallel architecture composed of 4 modules connected to a 20-bus interconnection structure is able to support all the WiMax code. Each module is in charge of at most three CS sets. It includes a local memory organized into 20 banks of 3×96 words.

Code	Modules	Sets per module	Buses	Scenarii	Memory size (6-bit word)
1/2	3	4	8	8	2592
	4	3	10	6	2880
	6	2	22	4	6336
2/3A	4	2	10	6	1920
2/3B	4	2	11	6	2112
3/4	3	2	15	8	2160
5/6	2	4	20	12	1920
	4	1	20	6	1920

Table 1. Optimal solutions for the different WiMAx ratios.

PM	FFs	Memory 18 kb block	Slice LUT	Max Frequency MHz
4	10 K (14%)	92	19K (27%)	192

Table 2. Implementation results of the IP on a Xilinx Virtex5 110LXT.

A synthesizable generic VHDL IP core, fully compliant with the 802.16e standard, has been developed. It covers all the modes and ratios defined in the standard. Our generic IP core allows to switch from one ratio to another on-the-fly. The core has been synthesized with Xilinx XST on a Virtex5 LX110T target. Table 2 gives the implementation results of the IP for a four processing module (PM) configuration.

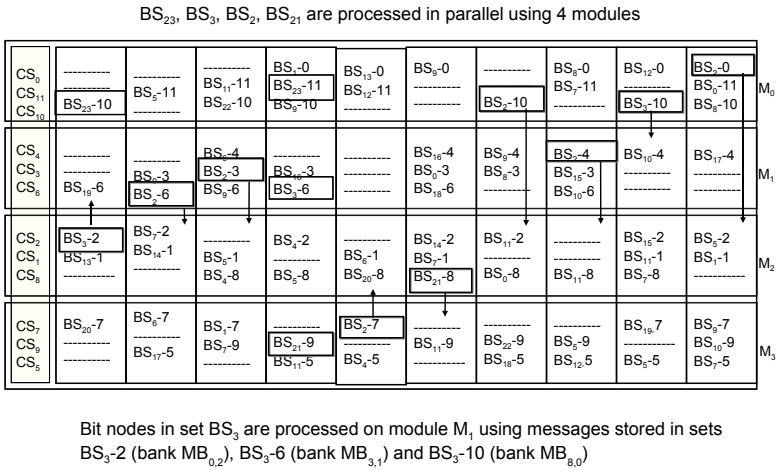

Figure 5. Data memory organization and scenario example for bit node and check node processing.

Memory usage can be drastically reduced when switching to the ASIC implementation because in the case of the FPGA implementation, the number of memories is greater than really needed. Currently 80 memories are used for message passing in this design but less than 10% of the total 18 Kbits of each memory is used. The whole design is synchronous and is running at 192 MHz. In order to increase the overall clock frequency, the IP core memories have a two-cycle latency.

PM		ratio 1/2	ratio 2/3A	ratio 2/3B	ratio 3/4A	ratio 3/4B	ratio 5/6
1	cycles/block	884	788	788	740	740	692
	Throughput	2.6	3.94	3.94	4.74	4.74	5.27
4	cycle/block	236	212	212	212	212	188
	Throughput	10.4	15.8	15.8	19.2	19.2	21.6
6	cycle/block	164	164	164	140	140	140
	Throughput	14.84	19.78	19.78	26.34	26.34	29.26

Table 3. Performance results of the LDPC decoder.

Table 3 gives some performance results obtained in the case of different configurations: 1, 4 and 6 processing modules. The system performances are expressed by: the number of cycles required for the execution of one algorithm iteration and the throughput of the IP in Mbit/sec (output payload stream rate after redundancy suppression) for 20 iterations and a 160 MHz system clock.

5 Conclusions

We have presented a parallel and modular LDPC architecture. An IP core fully compliant with all the modes of the 802.16e WiMax standard has been developed and synthesized on a Xilinx Virtex5. A four or six-module FPGA design yields a throughput ranging from 10 to 30 Mbit/s at a clock frequency of 160 MHz. This frequency range is compatible with the throughput requirements of the WiMax mobile standard.

Future work will consist in applying the principles of this scalable generic parallel architecture to other communication standards such as the 802.11n standard. It will also include further optimizations.

References

[1] A. Blanksby and C. Howland. A 690-mW 1-Gb/s 1024-b, Rate-1/2 Low-Density Parity-Check Code Decoder. *IEEE Journal of Solid-State Circuits*, 37(3):404–412, March 2002.

[2] J. Chen and M. P. C. Fossorier. Density Evolution for BP-Based Decoding Algorithms of LDPC Codes and their Quantized Versions. In *Proceedings of the IEEE Global Telecommunications Conference, (GLOBECOM 2002)*, pages 1378–1382, Nov. 2002.

[3] M. Cocco, J. Dielissen, M. Heijligers, A. Hekstra, and J. Huisken. A Scalable Architecture for LDPC Decoding. In *Proceedings of the conference on Design, automation and test in Europe (DATE 2004)*, pages 88–93, 2004.

[4] F. Kienle, T. Brack, and N. Wehn. A Synthesizable IP Core for DVB-S2 LDPC Code Decoding. In *Proceedings of the conference on Design, Automation and Test in Europe (DATE 2005)*, pages 100–105, 2005.

[5] D. J. C. MacKay. Good Error-Correcting Codes based on Very Sparse Matrices. *IEEE Trans. Inform. Theory*, 45:399–431, Mar. 1999.

[6] M. Mansour and N. Shanbhag. Architecture-Aware Low-Density Parity-Check Codes. In *Proceedings of the International Symposium on Circuits and Systems (ISCAS 2003)*, pages 57–60, May 2003.

[7] P. Urard, L. Paumier, P. Georgelin, T. Michel, V. Lebars, E. Yeo, and B. Gupta. A 135Mbps DVB-S2 Compliant Codec based on 64800-bit LDPC and BCH Codes. In *Proceedings of the Design Automation Conference, (DAC 2005)*, pages 547–548, June 2005.