GPU-based DVB-S2 LDPC decoder with high throughput and fast error floor detection

G. Falcao, J. Andrade, V. Silva and L. Sousa

A new strategy is proposed for implementing computationally intensive high-throughput decoders based on the long length irregular LDPC codes adopted in the DVB-S2 standard. It is supported on manycore graphics processing unit (GPU) architectures, for performing parallel multi-threaded decoding of multiple codewords with reduced accesses to global memory. This novel approach is flexible and scalable, and achieves throughputs superior to the 90 Mbit/s required by the DVB-S2 standard, while at the same time it improves error-correcting performances such as BER and error floors regarding conventional VLSI-based decoders.

Introduction: The second generation of the digital video broadcasting standard for satellite communications (DVB-S2) [1] uses binary and irregular low-density parity-check (LDPC) codes with frame lengths up to $n = 64\,800$-bit long. The length and irregularity of these codes make it harder to achieve real-time LDPC decoding with throughputs above 90 Mbit/s, as required by the DVB-S2 standard [1]. LDPCs are (n, k) linear block codes with length n and k information bits and they can be defined by sparse binary parity-check $n \times (n - k)$ $\boldsymbol{H}$ matrices. They are also represented as bipartite or Tanner graphs [2] having two types of node processors: check nodes (CNs) and bit nodes (BNs), connected by bidirectional edges.

The information received from the channel is propagated through neighbouring nodes of the graph (belief propagation), iteration after iteration, in order to infer a valid codeword $\hat{\boldsymbol{c}}$ that verifies all parity-check equations [2]. The intensive nature of such computation is imposed by thousands of messages being processed and communicated between adjacent nodes of the graph (several messages per node and iteration).

Although major solutions typically have to be developed by using VLSI-based dedicated architectures [3, 4], with 5 to 6-bit fixed-point precision arithmetic and corresponding BER and error floor limitations, recently more flexible and programmable approaches have been proposed [5–7]. However, they target only regular or short-length LDPC codes. The novel approach proposed here is based on ubiquitous, low-cost and homogeneous manycore GPU architectures [8] that nowadays exist in conventional personal computers. The GPU device communicates with the host CPU by using the PCIe bus and processes data by exploiting a high level of parallelism based on hundreds of cores and thousands of threads launched simultaneously [8]. The proposed solution decodes irregular and very long length ($n = 64\,800$) LDPC codes. Moreover, it is programmable, which introduces flexibility and allows using higher precision to represent data, therefore increasing coding gains as Fig. 1 shows. These properties represent significant advantages when compared with typical VLSI decoders [3, 4].

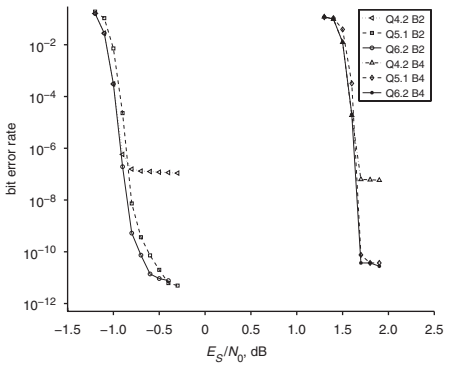

Fig. 1 *Min-sum BER results for codes B2 and B4 using fixed-point arithmetic, with messages in QX.Y format (X-integer bits, Y-decimal bits)*

Min-sum decoding algorithm: Although the min-sum algorithm (MSA) consists of a simplification of the sum–product algorithm (SPA) [2] based on the processing of log-likelihood ratios (LLR), it still requires intensive processing. As step 1 in Algorithm 1 illustrates, before the first iteration executes, all input Lp_n data received from the channel (*a priori* LLRs) are used to initialise Lq_{nm} elements. Then, the MSA processes kernels 1 and 2 (steps 3 to 6) on an iterative basis until the stop conditions occur [2].

Algorithm 1 min-sum (MSA)

1. Initialisation: $Lq_{nm} = Lp_n$
2. **while** $\hat{\boldsymbol{c}}\boldsymbol{H}^T \neq 0 \wedge i < I$ **do**
3. $Lr_{mn} = \left(\prod_{n' \in \mathcal{N}(m)\backslash n} sign(Lq_{n'm})\right) \cdot \min_{n' \in \mathcal{N}(m)\backslash n} |L_{qn'm}|$ //kernel 1 – horizontal processing
4. $Lq_{nm} = Lp_n + \sum_{m' \in \mathcal{M}(n)\backslash m} Lr_{m'n}$ //kernel 2 – vertical processing
5. $LQ_n = Lp_n + \sum_{m' \in \mathcal{M}(n)} Lr_{m'n}$ //*aosteriori* LLRs calculation
6. $\hat{\boldsymbol{c}}_n = (LQ_n > 0 \ ? \ 0 \ : \ 1)$ //hard decoding
7. **end while**

Exploiting parallelism on CUDA-based GPUs: The GPUs from NVIDIA, based on the compute unified device architecture (CUDA), consist of several multiprocessors, each composed of stream processors that compute thousands of threads simultaneously following a single-instruction multiple-thread (SIMT) approach.

The host CPU transfers data and controls the kernels' execution on the device. During the execution of each iteration of Algorithm 1, the host launches two main kernels on the device for processing the horizontal and vertical steps, respectively 3 and 4 (as Fig. 2 illustrates). This parallel approach adopts a thread-per-node perspective, where each thread processes a complete row (horizontal processing) or column (vertical processing) of $\boldsymbol{H}$. At the end, data is returned from device back to host.

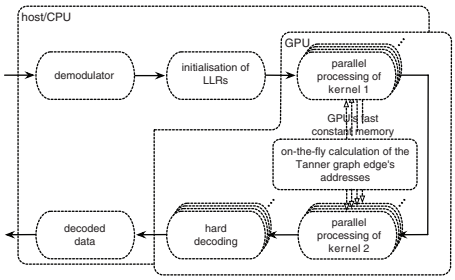

Fig. 2 *Parallelisation strategy for parallel LDPC decoder running MSA on GPU manycore architecture with CUDA*

The GPU DVB-S2 LDPC decoder processes 16 codewords in parallel. The fact that data is represented with 8-bit precision allows each 128-bit memory access to the GPU's global memory to read/write 16 elements of data, which favours parallelism. The Lr_{mn} and Lq_{nm} 8-bit data elements are processed independently for each of the 16 codewords. Furthermore, concerning the addresses of the Tanner graph edges, we specifically developed for this GPU-based approach a data memory organisation similar to the one used by Kienle *et al.* in ASICs [3]. Consequently, it is possible to calculate these addresses on-the-fly using data from the GPU's fast constant memory. This avoids the need of accessing the GPU's slow global memory to read the edge addresses, which saves considerable computation time. Therefore, significantly higher throughputs are obtained, compared with other previously reported approaches [5–7].

Experimental results: The experimental results were achieved by decoding a subset of DVB-S2 [1] LDPC codes using a C2050 Fermi GPU programmed with the C++ language compiled with GCC-4.4 and CUDA 3.2 [8]. The host platform uses a GNU/Linux kernel 2.6.31-22 x86_64.

The selected GPU has 14 multiprocessors with 32 stream processors each. It was programmed by launching 128 threads per block and a number of blocks that depends on the DVB-S2 code and kernel type (1 or 2).

Throughputs reported in Table 1 show that more than 90 Mbit/s are achieved for all rates under test running 20 iterations for DVB-S2 codes B2 (64 800, 21 600) and B4 (64 800, 32 400), which have rates 1/3 and 1/2, respectively. They compare well with state-of-the-art dedicated VLSI approaches [3, 4]. Fig. 1 shows that the use of 8-bit arithmetic leads to superior BER performance as opposed to the use of 6-bit, typical in VLSI solutions [3, 4]. Comparing fixed-point data representations, respectively with Q5.1 (6-bit, with 5-bit dedicated to the integer part, and 1-bit for decimal representation) and Q6.2 (8-bit), the

Figure clearly shows that coding gains exist in the waterfall region (data transmission is simulated over an additive white Gaussian noise (AWGN) channel using QPSK modulation). Furthermore, the GPU accelerates processing allowing the detection of error floors for normal frame length DVB-S2 codes within hours, instead of weeks of computation [9] (we tested 10^8 codewords per each E_s/N_0 value shown in Fig. 1, for a maximum number of iterations $I = 50$). Fig. 1 shows that the use of a Q6.2 (8-bit) representation produces error floors with performance more than three orders of magnitude superior compared with Q4.2 (6-bit) arithmetic for both B2 and B4 DVB-S2 codes.

Table 1: Throughputs achieved for DVB-S2 LDPC codes B2 and B4 (values are presented in megabits/second)

DVB-S2 code	Number of edges	5 iter.	10 iter.	15 iter.	20 iter.	25 iter.	30 iter.
B2	216000	297.3	190.6	140.2	110.9	91.7	78.1
B4	226800	287.7	185.2	136.5	107.8	89.0	75.8

Conclusions: We propose a novel GPU-based LDPC decoding solution for the DVBS2 standard, adopted in satellite communications. This programmable parallel decoder exploits massive data-parallelismwell suited for the GPU and uses a reduced number of accesses to the device's slow global memory, owing to the computation on-the-fly of the edge addresses, which allows the acceleration of processing and high throughputs to be obtained. This approach is scalable to future generations of GPUs that are expected to have more cores, which should improve performance, either in throughput or in BER, namely by increasing the level of multicodeword parallelism or by using higher precision to represent data, respectively. It compares fairly well with non-scalable and non-reusable VLSI DVB-S2 decoders and presents throughputs above the required 90 Mbit/s.

21 January 2011
doi: 10.1049/el.2011.0201

G. Falcao, J. Andrade and V. Silva (*Department of Electrical and Computer Engineering, University of Coimbra, and Instituto de Telecomunicações, Polo II – Universidade de Coimbra, Coimbra 3030-290, Portugal*)

E-mail: gff@co.it.pt

L. Sousa (*Department of Electrical and Computer Engineering, Technical University of Lisbon, Instituto Superior Técnico, and INESC-ID, R. Alves Redol, 9, Lisboa 1000-029, Portugal*)

References

1 EN 302 307 V1. 1.1, European Telecommunications Standards Institute (ETSI). Digital video broadcasting (DVB); second generation framing structure, channel coding and modulation systems for broadcasting, interactive services, news gathering and other broad-band satellite applications
2 Lin, S., and Costello, D.J.: 'Error control coding' (Prentice Hall, 2004)
3 Kienle, F., Brack, T., and Wehn, N.: 'A synthesizable IP core for DVB-S2 LDPC code decoding'. Proc. Design, Automation and Test in Europe (DATE), Munich, Germany, March 2005, Vol. 3, pp. 100–105
4 Dielissen, J., Hekstra, A., and Berg, V.: 'Low cost LDPC decoder for DVBS2'. Proc. Design, Automation and Test in Europe (DATE), Munich, Germany, March 2006, Vol. 2, pp. 1–6
5 Falcao, G., Sousa, L., and Silva, V.: 'Massively LDPC decoding on multicore architectures', *IEEE Trans. Parallel Distrib. Syst.*, 2011, **22**, (2), pp. 309–322
6 Wang, S., Cheng, S., and Wu, Q.: 'A parallel decoding algorithm of LDPC codes using CUDA'. Proc. 42nd Asilomar Conf. on Signals, Systems and Computers, Pacific Grove, CA, USA, October 2008, pp. 171–175
7 Falcao, G., Silva, V., Sousa, L., and Marinho, J.: 'High coded data rate and multicodeword WiMAX LDPC decoding on Cell/BE', *Electron. Lett.*, 2008, **44**, (24), pp. 1415–1416
8 Kirk, D., and Hwu, W.-M.: 'Programming massively parallel processors: a hands-on approach' (Morgan Kaufmann, Burlington, MA, 2010)
9 Cai, Y., Jeon, S., Mai, K., and Kumar, B.V.K.V.: 'Highly parallel FPGA emulation for LDPC error floor characterization in perpendicular magnetic recording channel', *IEEE Trans. Magn.*, 2009, **45**, (10), pp. 3761–3764