REAL-TIME DVB-S2 LDPC DECODING ON MANY-CORE GPU ACCELERATORS

Gabriel Falcao[⋆]*, Joao Andrade*[⋆]*, Vitor Silva*[⋆] *and Leonel Sousa*[†]

[⋆]Instituto de Telecomunicações, Department of Electrical and Computer Engineering
University of Coimbra, P-3030-290 Coimbra, Portugal
[†]INESC-ID, Department of Electrical and Computer Engineering
IST/Technical University of Lisbon, Portugal

ABSTRACT

It is well known that LDPC decoding is computationally demanding and one of the hardest signal operations to parallelize. Beyond data dependencies that restrict the decoding of a single word, it requires a large number of memory accesses. In this paper we propose parallel algorithms for performing in GPUs the most demanding case of irregular and long length LDPC codes adopted in the Digital Video Broadcasting – Satellite 2 (DVB-S2) standard used in data communications. By performing simultaneous multicodeword decoding and adopting special data structures, experimental results show that throughputs superior to 90 Mbps can be achieved when LDPC decoders for the DVB-S2 are implemented in the current GPUs.

Index Terms— DVB-S2, LDPC, Communications, GPU, CUDA

1. INTRODUCTION

When Gallager invented Low-Density Parity-Check (LDPC) codes in 1962 [1], the computational power required to decode large codewords in real-time was not available. Nearly thirty years later (in 1996) LDPC codes recaptured the attention of academia and industry [2] and Very Large Scale Integration (VLSI) solutions were proposed [3, 4, 5]. These became the approach to computing this type of intensive signal processing applications used in modern communications.

Until recently, signal processing algorithms were developed with a single processor in mind, but because of the change of paradigm in computer architectures, which has increased the number of cores in a system rather than its frequency of operation, they have to be parallelized and accelerated in order to take advantage of multi-core processing systems [6]. Although LDPC decoding solutions have recently been proposed for multi-core platforms [7, 8, 9, 10, 11, 12], they mainly address short and regular codes. In this paper we propose for the first time LDPC decoders based on Graphics Processing Units (GPU) for the computationally demanding case of irregular LDPC codes adopted in the Digital Video Broadcasting – Satellite 2 (DVB-S2) standard [13], which use very large codewords up to $N = 64800$ bit long. The irregular nature of these LDPC codes can impose memory access constraints and this, associated with large code size, creates challenges which are difficult to overcome. Also, the scheduling mechanism imposes important restrictions on the attempt to parallelize the algorithm. Thread-level and data-level parallelism can be conveniently exploited, together with the use of fast local memories, to harness the computational efficiency of these GPU-based signal processing algorithms. The algorithms developed support multicodeword decoding [10, 12] and are scalable to future GPU generations, which are expected to have a higher number of cores. We show that it is possible to achieve real-time DVB-S2 LDPC decoding with throughputs above 90 Mbps on ubiquitous GPU computing platforms. We conclude that such programmable devices can compete with dedicated VLSI hardware [3, 4, 5] in throughput and BER performance, and that they represent an alternative to typical VLSI solutions which have limitations imposed by quantization effects and the extensive use of fixed-point arithmetic, and high non-recurring engineering (NRE) costs.

This paper is structured as follows. Section 2 addresses the main properties of LDPC codes used in the DVB-S2 standard. Section 3 introduces the parallel processing structures provided by the many-core GPUs and presents the proposed parallel algorithm for LDPC decoding. Section 4 shows experimental results, and section 5 concludes the paper and indicates future directions for research.

2. DVB-S2 LDPC CODES AND DECODING ALGORITHM

LDPC codes are (N, K) linear block codes defined by sparse binary parity-check $\mathbf{H}$ matrices of codeword length N, with K information nodes and $N - K$ parity-check equations. They are usually represented by bipartite Tanner graphs [14], connecting Bit Nodes (BN) and Check Nodes (CN). The in-

This work has been partially supported by the Portuguese Foundation for Science and Technology (FCT) under grant SFRH/BD/37495/2007.

formation received from the channel is propagated, processed and exchanged between neighboring nodes of the graph, as depicted by the arrows in figure 1. If at the end of an iteration the codeword does not verify all parity-check equations, a new iteration is launched until the maximum number of allowed iterations occurs or a valid codeword is reached.

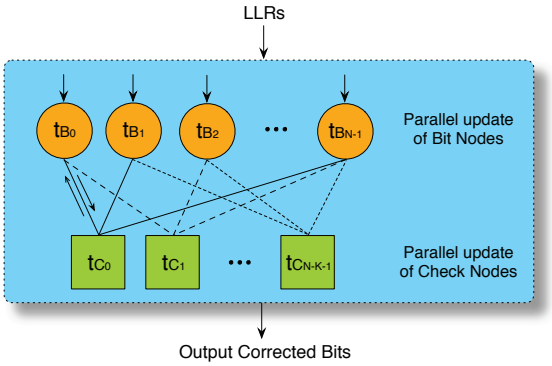

Fig. 1. Generic Tanner graph and iterative decoding using a thread-per-node parallel approach.

The LDPC codes adopted in DVB-S2 have a periodic nature, which allows the exploitation of suitable representations of data structures for attenuating their computational requirements. The properties of DVB-S2 codes are exploited for the GPU parallel architectures in this paper.

2.1. DVB-S2 LDPC IRA Codes

The FEC system of the recent DVB-S2 standard [13] incorporates a special class of systematic LDPC codes based on Irregular Repeat and Accumulate (IRA) codes [13, 14]. The parity-check matrix $\mathbf{H}$ has the form:

$$\mathbf{H}_{(N-K)\times N} = \left[\, \mathbf{A}_{(N-K)\times K} \,\middle|\, \mathbf{B}_{(N-K)\times(N-K)} \,\right] = \left[\begin{array}{ccc|cccccc} a_{0,0} & \cdots & a_{0,K-1} & 1 & 0 & \cdots & \cdots & \cdots & 0 \\ a_{1,0} & \cdots & a_{1,K-1} & 1 & 1 & 0 & & & \vdots \\ \vdots & & \vdots & 0 & 1 & 1 & \ddots & & \vdots \\ \vdots & \ddots & \vdots & \vdots & \ddots & \ddots & \ddots & 0 & \vdots \\ a_{N-K-2,0} & \cdots & a_{N-K-2,K-1} & \vdots & & \ddots & 1 & 1 & 0 \\ a_{N-K-1,0} & \cdots & a_{N-K-1,K-1} & 0 & \cdots & \cdots & 0 & 1 & 1 \end{array}\right], \quad (1)$$

where $\mathbf{A}$ is sparse and $\mathbf{B}$ is a staircase lower triangular matrix [13]. The periodicity constraints imposed on the pseudo-random generation of $\mathbf{A}$ allow a significant reduction in the storage requirements without code performance loss [13].

Moreover, the LDPC codes adopted in the DVB-S2 standard support two different frame lengths, one for short ($N = 16200$ bit) and the other for normal frames ($N = 64800$ bit). The short frame mode supports 10 distinct code rates, while the normal one supports 11 rates [13]. The column and row weights are variable and depend on the rate.

2.2. Min-Sum Algorithm

The Min-Sum algorithm was adopted in this work to perform the decoding of computationally intensive long LDPC codes because it is less complex than the well-known Sum-Product algorithm [14]. As figure 1 indicates, the inputs of the decoder are log-likelihood ratios (LLRs), which represent the logarithm of the ratio of two complementary probabilities ($LLR(x) = ln(p(x = 0)/p(x = 1))$) at the input of the decoder [14].

For each node pair (BN_n, CN_m) we initialize Lq_{nm} with the *a priori* LLR information received from the channel, LP_n. Then, we proceed to the iterative body of the algorithm by performing steps 4 and 5 described in Algorithm 1. At the

Algorithm 1

1: {Initialization}
2: **while** ($\mathbf{H}\hat{\mathbf{c}}^T \neq \mathbf{0} \wedge$ i $<$ I) {c-decoded word; I-Max. n. of iterations.} **do**
3: {For all node pairs (BN_n, CN_m), corresponding to $\mathbf{H_{mn}} = \mathbf{1}$ in the parity check matrix $\mathbf{H}$ of the code **do**:}
4: {Compute the LLR of messages sent from CN_m to BN_n:}
(Kernel 1 – Check Node processing)

$$Lr_{mn} = \prod_{\mathrm{n'}\in\mathcal{N}(\mathrm{m})\backslash\mathrm{n}} sign\left(Lq_{n'm}\right) \min_{\mathrm{n'}\in\mathcal{N}(\mathrm{m})\backslash\mathrm{n}} \left|Lq_{n'm}\right|, \quad (2)$$

{where $\mathcal{N}(m)\backslash n$ represents BN's connected to CN_m excluding BN_n.}
5: {Compute the LLR of messages sent from BN_n to CN_m:}
(Kernel 2 – Bit Node processing)

$$Lq_{nm} = Lp_n + \sum_{m'\in\mathcal{M}(n)\backslash m} Lr_{m'n}. \quad (3)$$

{where $\mathcal{M}(n)\backslash m$ represents the set of CN's connected to BN_n excluding CN_m.}
6: {Finally, we compute the *a posteriori* $LLRs$:}

$$LQ_n = Lp_n + \sum_{m'\in\mathcal{M}(n)} Lr_{m'n}. \quad (4)$$

7: {And perform hard decoding.}
8: **end while**

end, the decoded bits are obtained at the output of the system in codeword $\hat{\mathbf{c}}$.

3. PROPOSED PARALLEL ALGORITHM FOR MANY-CORE GPU

A computing system with a GPU consists of a host, typically a Central Processing Unit (CPU) that is used for programming and controlling the operation of the GPU. The GPU is a massively parallel processing engine [15] that can speed up processing by simultaneously performing the same operation on distinct data distributed by many arithmetic processing units. GPUs are programmable and one of the most widely used programming models is the NVIDIA Compute Unified Device Architecture (CUDA). It exploits data-level parallelism

while guaranteeing the coherent use of the different levels of memory hierarchy. The execution of a kernel on a GPU is

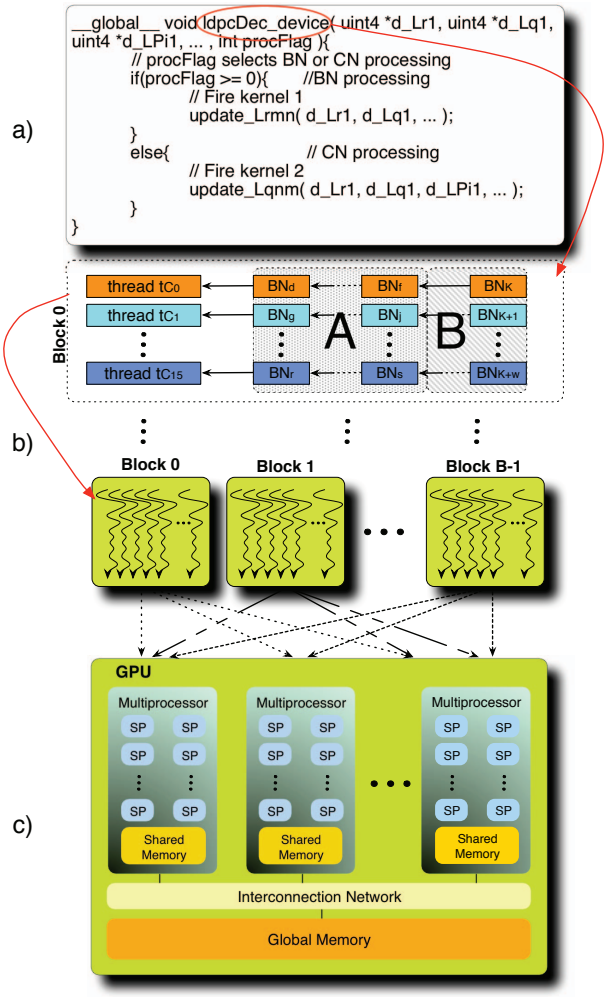

Fig. 2. Parallel multithreaded LDPC decoder processing a) kernels 1 and 2 on the GPU using one thread per node of the Tanner graph where, for example b), $BN_d, BN_f, ..., BN_K$ are BNs connected to CN_0, and threads are grouped and processed in B blocks on the c) GPU many-core architecture.

distributed across a grid of thread blocks with adjustable size. As figure 2 shows, each multiprocessor has several cores that can control more than one block of threads. Each block is composed of a predefined number of threads that execute the kernel synchronously. They are organized in groups of 32 threads, where each multiprocessor time-slices the threads in a group among its stream processors (SP in the figure).

3.1. Proposed parallel algorithm

The algorithm developed attempts to exploit two major capabilities of GPUs: the massive use of thread and data parallelism and the minimization of memory accesses, which often degrade performance in multi-core systems [15].

Multithread-based processing: In order to extract the essence of full thread-level parallelism from the GPU, the proposed DVB-S2 LDPC decoder exploits a thread-per-node approach (thread per row and thread per column based processing). Figure 2 illustrates this strategy with 16 threads per block (here represented by $t_{C_0}, ..., t_{C_{15}}$) being processed in parallel inside block 0 for the Check Node processing indicated in kernel 1 from Algorithm 1. A similar approach is applied to the remaining threads $t_{C_{16}}, ..., t_{C_{N-K-1}}$ of kernel 1 (retrieved from figure 1), which are grouped and executed in other blocks of the grid. Also, in kernel 2 threads $t_{B_0}, ..., t_{B_{N-1}}$ perform the equivalent parallel Bit Node processing. The efficiency of this parallelism is achieved by adopting a flooding schedule strategy that eliminates data dependencies in the exchange of messages between BNs and CNs [14]. Additionally, to fully exploit the massive processing power of the GPU, the algorithm performs multicodeword decoding by decoding 16 codewords in parallel. Moreover, this solution uses 8 bit to represent data, which compares favorably [10] with existing state-of-the-art VLSI DVB-S2 LDPC decoders that typically use 5 or 6 bit to represent data [3, 4, 5].

Coalesced accesses to data structures: In a GPU, parallel accesses to the slow global memory may kill performance and should, whenever possible, be avoided. To optimize this type of operation, data is contiguously aligned in memory, which favors coalescence to take effect and allows several threads to access corresponding data in simultaneous [10], as depicted in figure 3. Nevertheless, modern GPU hardware can be more

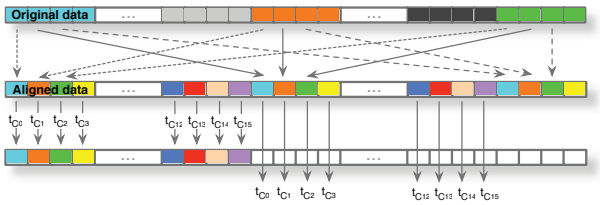

Fig. 3. Threads t_{C0} to t_{C15} performing parallel coalesced accesses to the GPU global memory.

efficient at dealing with out-of-order memory accesses and related issues [15].

4. EXPERIMENTAL RESULTS

This section presents experimental results of decoding all DVB-S2 codes B1 to B11; they represent the complete set of rates used in the standard [13] for normal frame lengths.

4.1. Experimental setup

The application was developed using CUDA 3.0 and the C/C++ programming language compiled with GCC-4.3. The

host is based on an Asus P6T7 Workstation running the GNU/Linux kernel 2.6.31-22 x86_64. The device consists of a Fermi C2050 GPU with 14 multiprocessors and 32 stream processors per multiprocessor, in a total of 448 cores.

4.2. LDPC DVB-S2 decoders on Fermi GPUs with CUDA

Table 1 shows the throughputs obtained from decoding different numbers of iterations for all codes and rates of the DVB-S2 normal frame standard. It can be seen that throughputs superior to 90 Mbps can be achieved for all codes. Also, the

Table 1. Throughput of the DVB-S2 LDPC decoder (Mbps)

Code	GPU				GPU-optimized			
	# of iterations							
	1	3	5	10	1	3	5	10
B1	239	163	126	79	284	191	143	87
B2	246	156	116	69	253	167	121	74
B3	227	143	103	61	234	148	109	65
B4	221	133	95	55	233	148	109	65
B5	177	102	72	41	180	104	74	43
B6	225	133	94	55	229	137	99	57
B7	204	116	82	47	210	121	85	49
B8	187	102	70	40	191	105	72	41
B9	175	92	63	35	177	94	64	36
B10	192	97	65	36	194	99	66	36
B11	190	95	64	35	192	97	65	36

right-hand side of table 1 shows an optimized version of the algorithm that minimizes memory accesses to the GPU's slow global memory. In this case, all memory addresses that represent the edges of the Tanner graph for the case of matrix **B** in (1) have a regular staircase profile and can be computed on-the-fly automatically on the GPU side, thereby reducing accesses to global memory. This saves processing time and throughputs are superior to those in the left-hand side of table 1. Because kernel 1 performs divergent operations and since codes B7 to B11 have high row weights (they range from 14 to 30), it penalizes the performance of the decoder. As matrix **B** is small for such rates, there are no significant improvements seen in the comparison of both versions. In spite of that, throughputs compare well with state-of-the-art VLSI solutions for DVB-S2 [3, 4, 5].

5. CONCLUSIONS

In this paper we proposed and developed efficient parallel algorithms to perform the massive decoding of DVB-S2 LDPC codes on GPUs. We have discussed the main challenges to implement efficient LDPC decoders on typical GPU architectures supported on CUDA. We have shown that high throughputs can be achieved for real-time applications, with values surpassing the 90 Mbps. They accommodate throughput requirements of the DVB-S2 standard and also compete well in BER performance with existing state-of-the-art VLSI dedicated LDPC decoders.

We intend to pursue this work by investigating the possibility of efficiently performing the calculation of all memory addresses of the Tanner graph on-the-fly, on the GPU side of the system. The goal is to reduce expensive accesses to the slow global memory of the GPU to accelerate the processing and improve throughput even more.

6. REFERENCES

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

[2] D. Mackay and R. Neal, "Near Shannon Limit Performance of Low Density Parity Check Codes," *IEE Electronics Letters*, vol. 32, no. 18, pp. 1645–1646, 1996.

[3] F. Kienle, T. Brack, and N. Wehn, "A Synthesizable IP Core for DVB-S2 LDPC Code Decoding," in *Proceedings of Design, Automation and Test in Europe, 2005 (DATE'05)*, March 2005, pp. 1–6.

[4] J. Dielissen, A. Hekstra, and V. Berg, "Low cost LDPC decoder for DVB-S2," in *Proceedings of Design, Automation and Test in Europe, 2006 (DATE'06)*, March 2006, pp. 1–6.

[5] S. Muller, M. Schreger, M. Kabutz, M. Alles, F. Kienle, and N. Wehn, "A novel LDPC decoder for DVB-S2 IP," in *Proceedings of Design, Automation and Test in Europe, 2009 (DATE'09)*, April 2009, pp. 1308–1313.

[6] T. P. Chen and Yen-Kuang Chen, "Challenges and opportunities of obtaining performance from multi-core CPUs and many-core GPUs ," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP'09)*, April 2009, pp. 613–616.

[7] M. Wu, S. Gupta, Yang Sun, and J. R. Cavallaro, "A GPU implementation of a real-time MIMO detector," in *Proceedings of the IEEE Workshop on Signal Processing Systems (SiPS'09)*, October 2009, pp. 303–308.

[8] Hyunwoo Ji, Junho Cho, and Wonyong Sung, "Massively parallel implementation of cyclic LDPC codes on a general purpose graphics processing unit," in *Proceedings of the IEEE Workshop on Signal Processing Systems (SiPS'09)*, October 2009, pp. 285–290.

[9] H. Ji, J. Cho, and W. Sung, "Memory Access Optimized Implementation of Cyclic and Quasi-Cyclic LDPC Codes on a GPGPU," *Journal of Signal Processing Systems*, DOI 10.1007/s11265-010-0547-9, 2010.

[10] G. Falcao, V. Silva, and L. Sousa, "How GPUs can outperform ASICs for fast LDPC decoding," in *Proceedings of the 23rd ACM International Conference on Supercomputing (ICS'09)*, June 2009, pp. 390–399.

[11] G. Falcao, V. Silva, and L. Sousa, *GPU Computing Gems*, chapter Parallel LDPC Decoding, ed. by Wen-mei Hwu, vol. 1, NVIDIA, Morgan Kaufmann, 2011.

[12] G. Falcao, L. Sousa, and V. Silva, "Massively LDPC Decoding on Multicore Architectures," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 2, pp. 309–322, 2011.

[13] 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, 2005.

[14] T. K. Moon, *Error Correction Coding – Mathematical Methods and Algorithms*, John Wiley & Sons, Inc., 2005.

[15] David Kirk and Wen-mei Hwu, *Programming Massively Parallel Processors: A Hands-on Approach*, Morgan Kaufmann, 2010.