GPU Accelerated Gigabit Level BCH and LDPC Concatenated Coding System

Selcuk Keskin[1] and Taskin Kocak[2]
Department of Computer Engineering, Bahcesehir University, Istanbul 34353, Turkey
Email: {selcuk.keskin[1],taskin.kocak[2]}@eng.bau.edu.tr

Abstract—**Increasing data traffic and multimedia services in recent years have paved the way for the development of optical transmission methods to be used in high bandwidth communications systems. In order to meet the very high throughput requirements, dedicated application specific integrated circuit and field programmable gate array solutions for low-density parity-check decoding are proposed in recent years. Conversely, software solutions are less expensive, scalable, and flexible and have shorter development cycle. A natural solution to lower the error floor is to concatenate the LDPC code with an algebraic outer code to clean up the residual errors. In this paper, we present the design and parallel software implementation of a major computation algorithm for LDPC decoding on general purpose graphics processing units as inner code and BCH decoding algorithm as outer code to achieve excellent error-correcting performance. The experimental results show that the proposed GPU-based concatenated decoder achieves the maximum decoding throughput of 1.82Gbps at 10 iterations with low bit-error rate (BER).**

I. Introduction

Low-density parity-check (LDPC) codes were originally proposed by Robert Gallager in 1962 [1] and re-discovered by Mackay and Neal in 1996 [2]. With the advent of technology, the interests in LDPC codes have been dramatically increased because their excellent error-correcting performance is much closer to the Shannon limit [2]. They have been used in recent digital communication systems, such as DVB-S2, WiMAX(802.16e), WiFi(802.11n), etc. Tanner introduced an effective graphical representation for LDPC Tanner codes [3]. LDPC can be decoded with iterative soft-decision decoding algorithms called message-passing algorithms [4].

Error correcting has started to become a challenge because of complex computations [5]. The belief propagation algorithm can be simplified using the "min-sum" approximation, which greatly reduces the implementation complexity, but incurs a degradation in decoding performance [6]. LDPC can be concatenated with Bose-Chaudhuri-Hochquenghem (BCH) codes to meet higher data rates with closer Shannon limit performance due to the increasing demand for high-speed communication [7]. Concatenated codes have since then evolved as a standard for those applications where very high coding gains are needed, such as deep-space applications, satellite communication, etc [8]. To design LDPC decoders that are suitable to ASIC/FPGA implementation, a fundamental basis is to introduce some reasonable constraints in the code design and then exploit regularities in the structure of the parity check matrix [9]. In this work, we examine that whether Graphics Processing Units (GPUs) can be used for the decoding process.

GPUs have only been used for 3D graphics rendering in the first years of their evolution. Later GPUs have started to offer high performance general purpose processing by executing thousands of threads simultaneously. A GPU provides a parallel architecture, which combines raw computation power with programmability [10]. GPU provides extremely high computational throughput by employing many cores working on a large set of data in parallel. Though, the parallelization algorithms is a challenging process. The number of threads and the amount of memories must be decided carefully, also the limited number of resources should be utilized fully. The design must be realized with a minimum latency because of frequent communication between the threads. In addition, attention should be paid to the synchronization to get correct results.

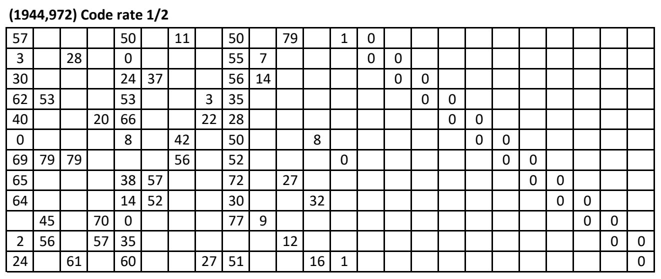
II. Concatenated LDPC and BCH decoder

Low-density parity-check (LDPC) codes are a class of linear block codes. The name comes from the characteristic of their parity-check matrix **H** which contains only a few *1*'s in comparison to the number of *0*'s. Their main advantage is that they provide performance which is very close to the capacity for a lot of different channels and linear time complex algorithms for decoding.

LDPC decoding requires the propagation of messages between connected nodes, as indicated by the Tanner graph. Tanner graphs are bipartite graphs. That means that the nodes of the graph are separated into two distinctive sets and edges are only connecting nodes of two different types. The two types of nodes in a Tanner graph are called variable nodes (bit node - BN) and check node (CN). Tanner graphs have proved to be an efficient algorithm for inference calculation, and it is used in numerous applications, including LDPC codes.

In Fig. 1 the parity check matrix **H** for length 1944 and code rate 1/2 is given. The parity check matrix **H** is represented by cyclic-permutation matrices that are obtained from a 81×81 identity matrix by cyclically shifting the columns to the left. The parity check matrix **H** consists of 6966 connection points.

A concatenated LDPC and BCH decoding algorithm has a LDPC code as the inner code and a BCH code as the outer code. In the other words, the data is encoded by an outer BCH code encoder and then each segment is encoded by an inner LDPC code encoder. In the receiver part, all codewords



(1944,972) Code rate 1/2

57				50		11		50		79		1	0										
3		28		0				55	7				0	0									
30				24	37			56	14					0	0								
62	53			53			3	35							0	0							
40			20	66			22	28								0	0						
0				8		42		50			8						0	0					
69	79	79				56		52				0						0	0				
65				38	57			72		27									0	0			
64				14	52			30			32									0	0		
	45		70	0				77	9												0	0	
2	56		57	35						12												0	0
24		61		60			27	51			16	1											0

Fig. 1: Permutation indices of parity check H matrix for FEC rate-1/2 LDPC code in 802.11n standard

are first decoded by the LDPC decoder and then the BCH outer decoder is executed to obtain better bit error rate (BER) performance. The overview of components for concatenated coding system is illustrated in Fig. 2.

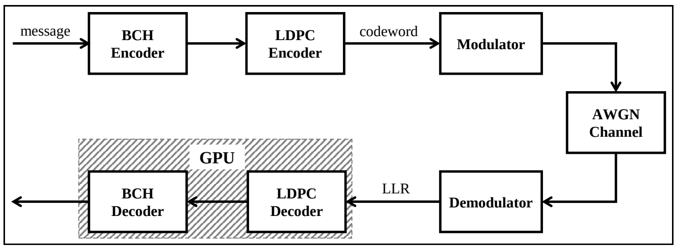


Fig. 2: Concatenated coding system

The decoding of the BCH codes requires three steps. First, the syndrome $S(x)$ is generated by generator polynomial of the received words $R(x)$; second, if the syndrome has a non-zero value then a key equation σ is generated using the Berlekamp-Massey algorithm [11]; finally, the Chein search algorithm [12] is employed to find the error locator $E(x)$ and then correct words are got by

$$C(x) = E(x) + R(x) \tag{1}$$

Let n be the code length and t is the error correcting capability of BCH codes. The syndrome values can be calculated by

$$S_{j-1} = \sum_{i=0}^{n} R_i \alpha^{ij}, \quad j = 1, 2, ..., 2t \tag{2}$$

where α^{ij} represent the primitive elements of GF(2^m) when m is the power of BCH codes. Suppose there are e error bits, then the error locator polynomial is defined as

$$\sigma(x) = \sigma_0 + \sigma_1 x + \sigma_2 x^2 + \cdots + \sigma_e x^e \tag{3}$$

III. GPU Based Concatenated Decoder

The decoding algorithm is implemented on TITAN X which has 12 billion transistors, 3584 Compute Unified Device Architecture (CUDA) cores, and 12GB of GDDR5 memory. GPU provides extremely high computational throughput by employing many cores working on a large set of data in parallel. However, the parallelization of algorithms is a challenging process. The number of threads and the amount of memories must be decided carefully, also the limited number of resources should be utilized fully. The NVIDIA GPU architectures consist of multiple stream multiprocessors (SM) which consist of pipelined cores and instruction dispatch units. All cores of TITAN X have grouped to 28 SMs. The SM processing core architecture of TITAN X can be seen in Fig. 3. The six SMs are grouped into a Graphics Processing Cluster (GPC) which has one Raster Engine.

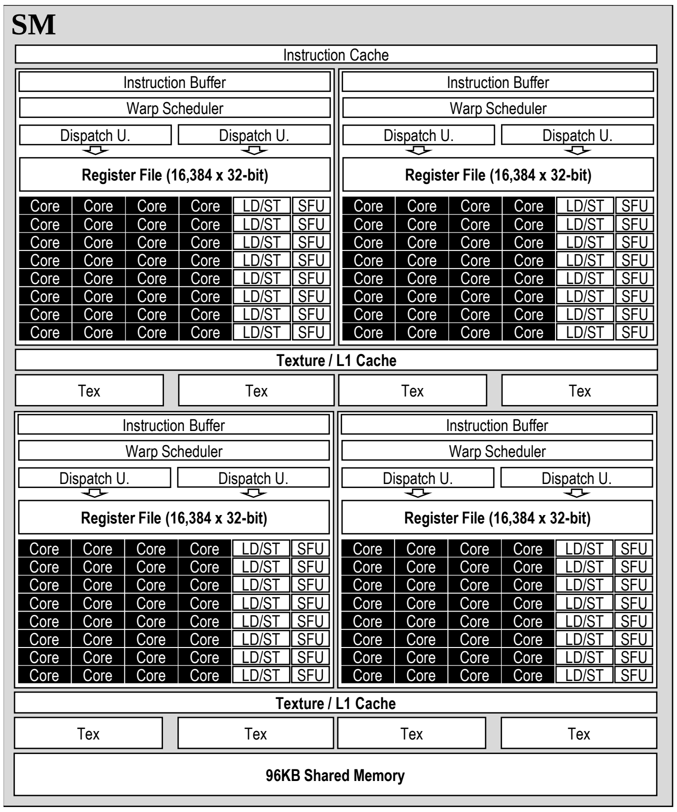


Fig. 3: Pascal GP102 Streaming Multiprocessor

Compared to previous chip architectures Kepler and Maxwell, the SMs memory hierarchy has also changed. Rather than implementing a combined shared memory/L1 cache block as in Kepler SMX, Pascal SM units feature a 96 KB dedicated shared memory, while the L1 caching function has been moved to be shared with the texture caching function. Global memory of GPU is an off-chip memory. Whole SM can access the global memory, but access time is the slowest.

Instead of calculating a single codeword, a group of codewords are computed. Because of the kernel initialization, codewords as much as possible must be transferred to the global memory of GPU. Also, we observed that some threads access the same location in the array which is defined on GPU global memory. The shared memory (SMEM) on the GPU is faster than the GPU global memory, which is off-chip. We move the inputs to the shared memory in a memory coalesced fashion. Reading from the same location on the shared memory is not as problematic as the GPU global memory. The transfer is illustrated in Fig. 4.

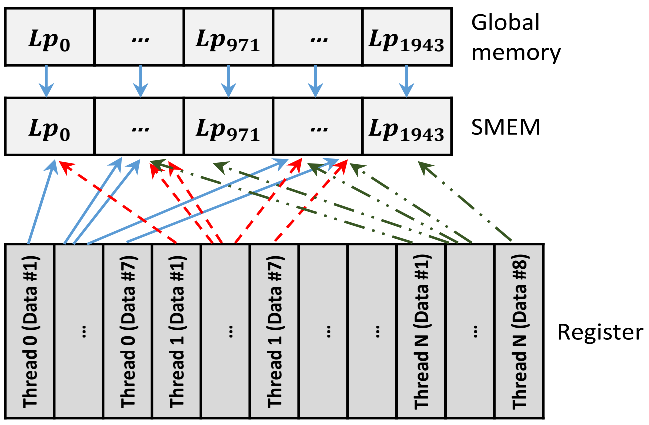


Fig. 4: Copying initial values from global memory to register

The inner LDPC decoding can be split into three blocks: horizontal processing, vertical processing and hard decision. In our implementation, hard decision is performed only one time when the maximum iteration is reached. One thread block calculates one LDPC codeword. So, data transfer from one thread to another one is needed between horizontal and vertical processing. When threads calculate their rows during horizontal processing or columns during vertical processing, the values are written to SMEM because reading for the next process from shared memory will be faster than global memory. Memory usage between processes is shown in Fig. 5.

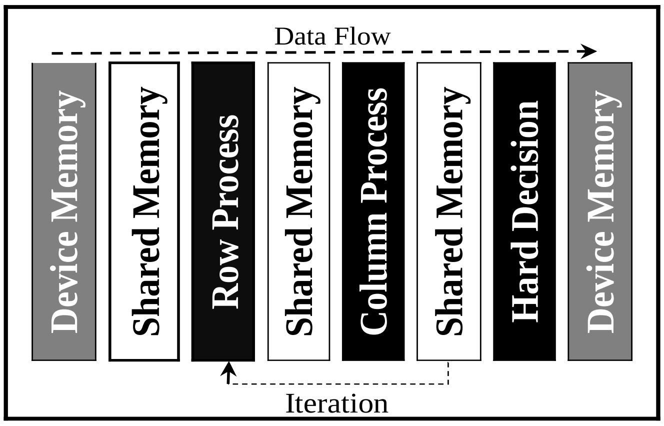


Fig. 5: Shared and global memory usage

The parity check matrix in Fig. 1 has 972 rows and 1944 columns when the code rate is 1/2. During the row processing, all data in the same row are dependent each other. This also applies to the column processing. So, one row or column can be calculated by one kernel thread. Accordingly, 1944 threads are needed to calculate whole columns in H matrix during the vertical processing. On the other hand, 972 threads are enough to calculate whole rows during the horizontal processing.

Unfortunately, 1944 is not a suitable number as thread count for the LDPC decoding on GPU because each thread uses 63 registers to perform the decoding process. So, total number of thread count is $1944 \times 63 = 122472$ per LDPC process while 65536 registers per multiprocessor is a physical limit for GPU. The resource usage according to different count of threads is summarized in Table I. The count of threads are set as 243 for the inner LDPC decoding which gives better SM occupancy performance with more active blocks to reduce the computation time of LDPC decoding.

TABLE I: The resource usage of GPU vs Threads per block

# of threads :	162	**243**	486	648	972	1944
Active Threads per SM	576	**768**	1024	672	992	N/A
Active Warps per SM	18	**24**	32	21	31	N/A
Active Thread Blocks per SM	3	**3**	2	1	1	N/A
Occupancy of each SM	28%	**38%**	50%	33%	48%	N/A

After the thread count optimization, each thread calculates 4 rows and 8 columns in each iteration for FEC rate 1/2. If the number of iterations is reached, each thread calculates 4 columns to perform hard decision. During the outer BCH decoding, each coefcient of polynomial is computed by one thread and the thread calculates the information bit as an output of concatenated decoder. The input of BCH decoding is already on GPU memory because it is the output of LDPC decoder. Each thread calculates the syndrome polynomial, determines the error locator and finally makes error corrections. As final step of concatenated decoder system, the decoded blocks are transferred from the device global memory to the host memory in CPU. Algorithmic flow can be seen in Fig. 6.

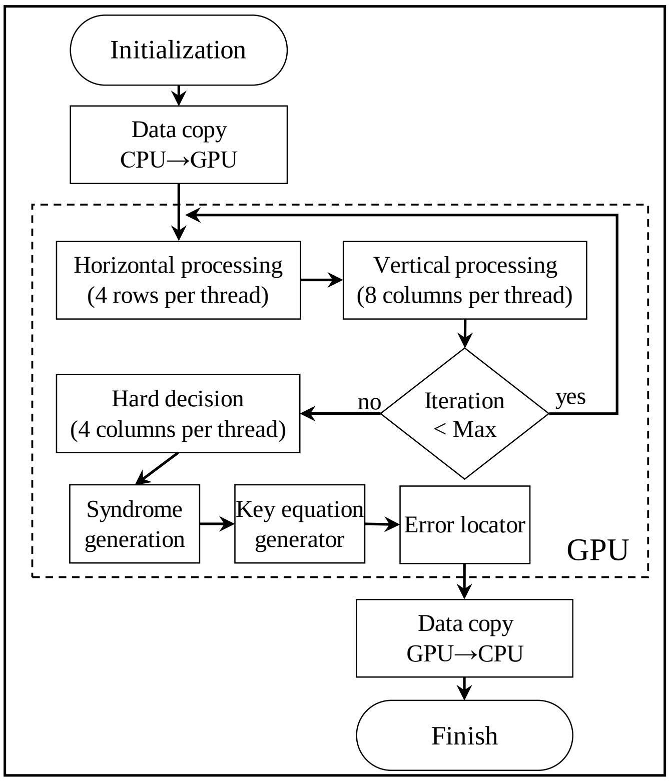


Fig. 6: Decoding one codeword on concatenated decoder

IV. Experimental Results

The concatenated decoder algorithm for GPGPU was implemented on CUDA 8.0 environment and executed on NVIDIA's TITAN X board which has GP102 architecture. All the experiments were conducted on a 3.5 GHz Intel Core i7-4770K processor with 12 GB DDR3 of memory.

In our experiments, there are 243 threads for the computation of one codeword. 32768 concurrent codewords (32768×243 threads) are decoded to achieve high decoding throughput. To calculate bit error rate (BER), we designed a simulator platform with Matlab Communications System Toolbox, that provides algorithms for designing, simulating, and analyzing communications systems.

The measured throughput of the GPU based BCH-LDPC concatenated decoder implementation are presented in Table II. As seen from the results, the proposed GPU implementation is able to meet the constraints of IEEE 802.11n standard. As the throughput increases with a decreasing maximum number of iterations, we can derive that about 10 iterations should give us the required maximum bitrate of the IEEE 802.11n standard.

TABLE II: GPU decoder average throughput in Gbps

Rate	5 iterations	10 iterations	15 iterations
1/2	2.43	1.66	1.26
3/4	2.66	1.82	1.38
5/6	2.58	1.76	1.34

Fig. 7 shows simulation results for a 16-QAM conguration at the code rates 1/2. The simulations were performed on energy per bit to noise power spectral density ratio (E_b/N_0) levels 0.5 dB apart under the Additive White Gaussian Noise (AWGN) channel.

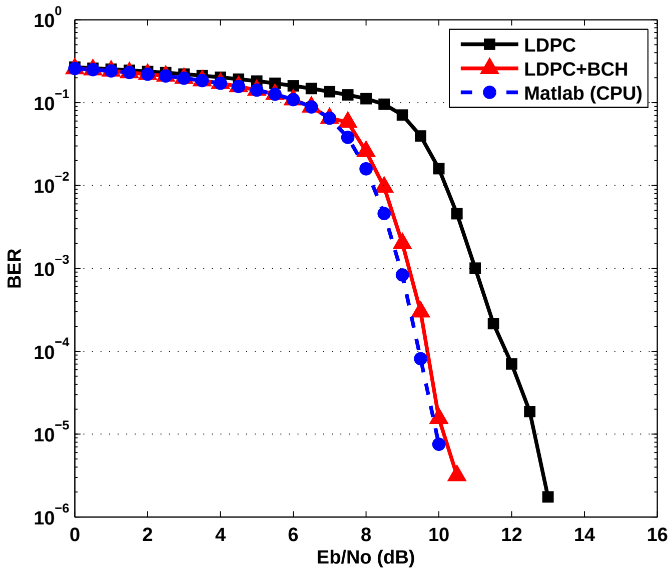


Fig. 7: BER comparison of algorithms with 10 iterations, for FEC rate = 1/2

GPU based MSA decoder has better throughput performance than Matlab standard LDPC decoder. However, as we observed from Fig. 7, BER performance of MSA is worse than Matlab. On the other hand, concatenated GPU decoder catches up the Matlab's performance. Both Matlab and concatenated GPU decoder take the available performance close to the Shannon limit.

V. Conclusions

This paper presents the design, methodology and implementation of concatenated LDPC and BCH decoders on Pascal based GPU architecture. GPUs are resourceful and efficient for accelerating DSP-based algorithms; however, the challenge is to map the algorithms to the computational resources in GPUs. To achieve high decoding throughput with better BER performance, thread based optimization and memory access optimizations are used as enhancements on the basic form of the algorithm. The results exhibit that our concatenated decoder system achieves up to 1.82 Gbps peak throughput for 10 iterations. It is possible that GPU based concatenated LDPC and BCH codes may show the greatest potential for practical implementation in advanced systems like high speed optical transmission, deep-space applications, etc.

Acknowledgment

As a member of CUDA Research Center, we also acknowledge the support of NVIDIA Corporation with the donation of the GPU used for this research.

References

[1] R. 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," *IEEE Electronics Letters*, vol. 32, no. 18, pp. 1645–1646, 1996.

[3] R. Tanner, "A recursive approach to low complexity codes," vol. 27, no. 5, pp. 533–547, 1981.

[4] K.-B. Png, X. Peng, and F. Chin, "Performance studies of a multi-band OFDM system using a simplified LDPC code," in *Proc. International Workshop on Ultra Wideband Systems*, May 2004, pp. 376–380.

[5] L. Zhang, Z. Wang, Q. Hu, and J. Zhang, "High Speed Concatenated Code Codec for Optical Communication Systems," in *2009 Symposium on Photonics and Optoelectronics*, Aug 2009, pp. 1–4.

[6] J. Chen, A. Dholakia, E. Eleftheriou, M. Fossorier, and X. Hu, "Reduced-complexity decoding of LDPC codes," vol. 53, no. 8, pp. 1288–1299, 2005.

[7] P. H. Chen, J. J. Weng, C. H. Wang, and P. N. Chen, "BCH Code Selection and Iterative Decoding for BCH and LDPC Concatenated Coding System," *IEEE Communications Letters*, vol. 17, no. 5, pp. 980–983, 2013.

[8] D. Divsalar and F. Pollarai, "Hybrid concatenated codes and iterative decoding," California Institute of Technology, Tech. Rep., 1997.

[9] C. Chang, M. Huang, and Y. Chang, "Design of GPU-based platform for LDPC decoder," in *IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, July 2011, pp. 3429–3432.

[10] J. B. Srivastava, R. Pandey, and J. Jain, "Implementation of Digital Signal Processing Algorithm in General Purpose Graphics Processing Unit (GPGPU)," *International Journal of Innovative Research in Computer and Communication Engineering*, vol. 1, no. 4, pp. 1006–1012, 2013.

[11] J. Massey, "Shift-register synthesis and BCH decoding," *IEEE Transactions on Information Theory*, vol. 15, no. 1, pp. 122–127, 1969.

[12] A. K. Subbiah and T. Ogunfunmi, "Efficient implementation of BCH decoders on GPU for flash memory devices using iBMA," in *2016 IEEE International Conference on Consumer Electronics (ICCE)*, Jan 2016, pp. 275–278.