Resource Aware LDPC Decoder Algorithm on ARM and GPU of Mobile Devices

Roohollah Amiri
Department of Electrical and Computer Engineering
Boise State University
Email: roohollahamiri@boisestate.edu

Hani Mehrpouyan
Department of Electrical and Computer Engineering
Boise State University
Email: hanimehrpouyan@boisestate.edu

Inanc Senocak
Department of Mechanical Engineering
Boise State University
Email: senocak@boisestate.edu

***Abstract*—Low Density Parity Check(LDPC) code is an efficient way of communication and is beeing largely used in mobile communication. With the wide usage and having computational intensivity of LDPC decoders, there has been a lot of effort to reduce decoder's complexity through algorithm optimization and parallel implementation. Recent improvements in mobile processors's architectures has made it exclusively reachable to have a real-time decoder based on a software solution. Knowing this capability, low profile GPU based decoders has been introduced that are capable of reaching high throughput by low latency. On the other hand recently there has been some work that has used ARM NEON SIMD unit with promising throughput and latency. What this works miss is that a mobile processor that is used in a smart phone should support a lot of task and we can not allocate all resources to decoding prcesses. In this paper we propose a heterogeneous LDPC decoder that uses both ARM and GPU Processors of a mobile device to reach real-time effieciency. The different stages of decoder processes has been allocated to ARM and GPU based on an optimization solution.**

I. Introduction

Originally proposed by Robert Gallager in 1962 [1] and rediscovered by MacKay and Neal in 1996 [2] Low Density Parity Check (LDPC) codes have been adopted by a wide range of applications including many communication system standards such as WiFi(IEEE 802.11n), 10 Gbit Ethernet (IEEE 802.3an), WiMAX (IEEE 802.16e), and DVB-S2. Recently, Chung and Richardson [3] showed that the LDPC code can approach the Shannon limit to within 0.0045 dB. However, the drawback of high correcting efficiency comes from its decoding computation complexity [4] and to date there exist no known mathematical tools to accurately evaluate their performance. Thus, a resort is typically made to simulations using computers or dedicated hardware [5].

LDPC decoding algorithms are compute-intensive and need powerful computer architecture to convey low latency and high decoding rate which caused to be initially implemented using application-specific integrated circuits(ASIC) and field-programmable gate array(FPGA) circuits [6]. However, their high speed often comes at a price of high developement cost and low programming flexibility [7] and it is very challenging to design decoder hardware that supports various standards and multiple data rates [8]. On the other hand, iterative LDPC decoding schemes based on the sum-product algorithm (SPA) can fully be parallelized, leading to high-speed decoding [3]. For these reasons, designers have recently focused on software implementations of LDPC decoders on multi/many-core devicdes [9] to achieve requirements through Software Defined Radio (SDR) Systems.

As in terms of multicore architectures, researchers have used CPUs [10], [11], GPUs [5], [9], [12]and ARM [11], [13] architectures to develop high throughput, low latency SDR systems.

In microarchitectures, increasing clock frequencies to obtain peorformance has reached a limit, so to hold this increase, other techniques based on parallel processing is being investigated [4]. Todays' multicore architectures support SIMD (Single Instruction Multiple Data), SPMD(Single Programm Multiple Data) and SIMT(Single Instruction Multiple Threads). The general purpose multicore processors replicate a single core in a homogeneous way, typically with a x86 instruction set, and provide shared memory hardware mechanisms [9]. They can be programmed at a high level by using different software technologies [14]. OpenMP [15] provides an effective and relatively straightforward approach for programming general-purpose multicores. On the other hand newer microarchitectures are trying to provide larger SIMD units for vector processing like SSE, AVX and AVX2 [16] on Intel Architectures. In [4], the authors have used Intel SSE/AVX2 SIMD Units to effieciently implement a high throughput LDPC decoder. In [8], OpenMp is used to generate address patterns with parity check H-matrix.

Mainly due to the demands for visualization technology in the games industry, the performance of graphics processing units (GPUs) has undergone increasing performances over the last decade. With many cores driven by a considerable memory bandwidth, recent GPUs are targeted for computationally intensive, multithreaded, highly parallel computation, and researchers in high-performance computing fields are applying GPUs to general-purpose applications (GPGPU) [5], [8], [12], [17]–[19]. They have used Compute Unified Device Architecture (CUDA) from NVIDIA [20] and Open Computing Language (OpenCL) platforms to develop LDPC Decoders.

Due to large computing capacity of multicore devices, software LDPC decoders have met the required throughputs of communication standards, although power consumption of x86 and GPU devices is incompatible with most of the embedded

systems [13]. To solve this issue, ARM-based SDR systems have been prposed in recent years [6], [11], [13] with goal of a SDR LDPC decoder that provides high through, low latency on a low-power embedded system. The authors in [13] have used ARM Processors's NEON SIMD and SIMT programming models to implement to implement a horizontal layered-based decoder that is based on parallel decoding of a low set of frames. This approach allows reaching high throughput while maintaining low-latency. Due to restrictions in an embedded system, using all resources of the system is a crucial task. Recent works in SDR LDPC embedded systems are missing the fact that todays mobile devices have powerful CUDA enabled GPUs. This paper has prposed a new algorithm that exploits ARM NEON SIMD Units and GPU together to reach a high throughput, low latency LDCP decoder. The main specification of the algorithm is that is devides processing task between system's resources.

II. LDPC codes and their Decoding Processes

Many works as in [6], [9], [11], [17] focused on mapping LDPC decoders on multicore architectures. Most of these works are based on the standard Two-Phase Message Passing (TPMP) schedule described in [9]. This algorithm works in two phases. In the first phase, all the variable nodes send messages to their neighboring parity check nodes, and in the second phase the parity check nodes send messages to their neighboring variable nodes. Due to transcendental operations and relying of Sum-Product algorithm to the estimation of noise standard deviation, in practice Min-Sum (MS) variants are prefered by designers [13]. More efficient layered schedules, such horizontal layered-based decoding algorithm, allow updated imformation to be utilized more quickly in the algorithm thus speeding up the decoding [**?**], [19]. In fact, the parity check matrix can be viewed as a layered graph decoded sequentially. Thw work in [17] has applied a form of layered belief propogation to irregular LDPC codes to reach 2x faster convergence in a given error rate.By using this methos they have reduced memory bits usage by 45-50%. The major limitation of layered algorithm is its irregular memory access although it is composed of a single loop kernel composed to two sequential kernels in standard algorithms. To solve the irregular memory access a data interleaving/deinterleaving process is bein used before and after the decoding process [13], [17].

In this paper the interleaving/deinterleaving process is done ny using ARM Vector processing units and frame decoding is being done in GPU of a mobile device.

III. Parallel Frame Processing

The porposed LDPC decoder is implemented on Jetson TK1 SoCs which contains 4 Cortex-A15 processors. Each core includes a NEON SIMD unit. To achieve high throughput performance on such low-power embedded processors, the following programming model is exploited in the proposed LDPC decoder.

IV. Experimental Results

The experiments were carried out by decoding LDPC codes using NVIDIA Tegra K1 Socs. The programs compiled with GCC-4.8 and CUDA 6.5. The TK1 is composed of 4 cortex-A15 ARM processors and one NVIDIA Kepler "GK20a" GPU with 192 SM3.2 CUDA cores. The host platform uses a GNU/Linux kernel 3.10.40-gdacac96.

V. Conclusion

The conclusion goes here.

REFERENCES

[1] R. Gallager, "Low-density parity-check codes," *IRE Transactions on Information Theory*, vol. 8, no. 1, pp. 21–28, January 1962.

[2] D. J. C. MacKay and R. M. Neal, "Near shannon limit performance of low density parity check codes," *Electronics Letters*, vol. 33, no. 6, pp. 457–458, Mar 1997.

[3] S.-Y. Chung, G. D. Forney, T. J. Richardson, and R. Urbanke, "On the design of low-density parity-check codes within 0.0045 db of the shannon limit," *IEEE Communications Letters*, vol. 5, no. 2, pp. 58–60, Feb 2001.

[4] B. L. Gal and C. Jego, "High-throughput multi-core ldpc decoders based on x86 processor," *IEEE Transactions on Parallel and Distributed Systems*, vol. PP, no. 99, pp. 1–1, 2015.

[5] S. Kang and J. Moon, "Parallel ldpc decoder implementation on gpu based on unbalanced memory coalescing," in *Communications (ICC), 2012 IEEE International Conference on*, June 2012, pp. 3692–3697.

[6] J. Andrade, G. Falcao, and V. Silva, "Flexible design of wide-pipeline-based wimax qc-ldpc decoder architectures on fpgas using high-level synthesis," *Electronics Letters*, vol. 50, no. 11, pp. 839–840, May 2014.

[7] Y. Hou, R. Liu, H. Peng, and L. Zhao, "High throughput pipeline decoder for ldpc convolutional codes on gpu," *IEEE Communications Letters*, vol. 19, no. 12, pp. 2066–2069, Dec 2015.

[8] J.-Y. Park and K.-S. Chung, "Parallel ldpc decoding using cuda and openmp," *EURASIP Journal on Wireless Communications and Networking*, vol. 2011, no. 1, pp. 1–8, 2011. [Online]. Available: http://dx.doi.org/10.1186/1687-1499-2011-172

[9] 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, Feb 2011.

[10] S. Grönroos, K. Nybom, and J. Björkqvist, "Efficient gpu and cpu-based ldpc decoders for long codewords," *Analog Integrated Circuits and Signal Processing*, vol. 73, no. 2, pp. 583–595, 2012. [Online]. Available: http://dx.doi.org/10.1007/s10470-012-9895-7

[11] S. Grnroos and J. Bjrkqvist, "Performance evaluation of ldpc decoding on a general purpose mobile cpu," in *Global Conference on Signal and Information Processing (GlobalSIP), 2013 IEEE*, Dec 2013, pp. 1278–1281.

[12] G. Wang, M. Wu, B. Yin, and J. R. Cavallaro, "High throughput low latency ldpc decoding on gpu for sdr systems," in *Global Conference on Signal and Information Processing (GlobalSIP), 2013 IEEE*, Dec 2013, pp. 1258–1261.

[13] B. L. Gal and C. Jego, "High-throughput ldpc decoder on low-power embedded processors," *IEEE Communications Letters*, vol. 19, no. 11, pp. 1861–1864, Nov 2015.

[14] H. Kim and R. Bond, "Multicore software technologies," *IEEE Signal Processing Magazine*, vol. 26, no. 6, pp. 80–89, November 2009.

[15] B. Chapman, G. Jost, and R. v. d. Pas, *Using OpenMP: Portable Shared Memory Parallel Programming (Scientific and Engineering Computation)*. The MIT Press, 2007.

[16] M. Deilmann, "A guide to auto-vectorization with intel c++ compilers," *Intel Corporation*, April 2012.

[17] B. L. Gal, C. Jego, and J. Crenne, "A high throughput efficient approach for decoding ldpc codes onto gpu devices," *IEEE Embedded Systems Letters*, vol. 6, no. 2, pp. 29–32, June 2014.

[18] G. Falcao, V. Silva, L. Sousa, and J. Andrade, "Portable ldpc decoding on multicores using opencl [applications corner]," *IEEE Signal Processing Magazine*, vol. 29, no. 4, pp. 81–109, July 2012.

[19] B. L. Gal and C. Jego, "Gpu-like on-chip system for decoding ldpc codes," *ACM Trans. Embed. Comput. Syst.*, vol. 13, no. 4, pp. 95:1–95:19, Mar. 2014. [Online]. Available: http://doi.acm.org/10.1145/2538668

[20] Cuda homepage. [Online]. Available: http://www.nvidia.com/object/cuda_home_new.html