FLEXIBLE NON-BINARY LDPC DECODING ON FPGAS

Joao Andrade[*]*, Gabriel Falcao, Vitor Silva*

Instituto de Telecomunicações
Dept. of Electrical and Computer Eng.
University of Coimbra, Portugal

Kenta Kasai[†]

Dept. of Comm. and Comp. Eng.
Graduate School of Science and Eng.
Tokyo Institute of Technology, Japan

ABSTRACT

Despite their ability to reach within the channel capacity in shorter codeblock lengths, non-binary LDPC codes have a higher decoding complexity that poses non-trivial barriers to their generalized adoption at algorithmic and compute-intensive levels. In this work, we propose a programmable FFT-SPA decoder that delivers high decoding throughput at low power consumptions, while retaining a design flexibility at the system level which surpasses typical VLSI descriptions, guaranteeing quick retargeting and prototyping of variants of this family of signal processing algorithms with effective decoding throughputs of up to 1 Mbit/s and potential throughputs of dozens of Mbit/s.

Index Terms— Non-binary LDPC codes, GF(q), Communications, FPGA, OpenCL

1. INTRODUCTION

Among the different types of Error–Correcting Codes (ECCs) that offer low Bit Error Rate (BER) at high noise levels we find Turbo codes and Low–Density Parity–Check (LDPC) codes. In fact, both offer capacity-approaching characteristics, and in the particular case of LDPC codes, it has been shown that they arbitrarily approach the Shannon limit [1, 2]. However, the adoption of LDPC codes has been accompanied by the development of powerful decoding architectures able to match the decoding rates and latency with nowadays' requirements within a sensitive level of energy consumption [3].

A fundamental reason why non-binary LDPC codes have not been as widely standardized as their binary relatives is due to the high numerical complexity involved in the soft-decoding algorithms. Notable exceptions on the adoption of non-binary LDPC codes are: *i)* Quantum–Key Distribution (QKD) encryption protocols [4]; *ii)* long-haul optical communications [5]; and *iii)* RaptorQ codes for the erasure channel [6]. To ameliorate the numerical complexity, several reduced complexity algorithms have been proposed, amongst them the Fast Fourier Transform Sum–Product Algorithm (FFT–SPA) which is explored in this work for low order binary extension fields (GF(2^m)). Moreover, given the particular nature of the end decoding platforms, domain-specific knowledge of the hardware description language is typically required for the Register Transfer Level (RTL) description of the Very Large Scale Integration (VLSI) decoding system in addition to the design space exploration involved in a non-binary LDPC decoder project. Addressing these two issues at once is yet to be tackled by the scientific community, with the due exception of Software–Defined Radio (SDR) systems where multicore technology has been successfully deployed [7, 8, 9].

This challenge may be addressed by the use of High–Level Synthesis (HLS) tools that generate dedicated accelerators for Field–Programmable Gate Arrays (FPGAs) while maintaining the development effort at an affordable level. This way, the focus can be given to the algorithmic details of the accelerator rather than on low-level details of the underlying processor platform. In particular, we explore the OpenCL programming model and the HLS tool from Altera to generate a wide-pipeline accelerator for LDPC decoders over Galois Fields of dimension q (GF(q)).

We summarize the paper contributions as follows: *i)* a programmable approach based on OpenCL and the HLS compiler tool from Altera that automatically generates RTL descriptions from OpenCL kernels, allowing rapid test and prototyping of algorithmic changes with high throughput; *ii)* exploiting architectural extensions to accommodate higher levels of parallelism that mask memory accesses and bandwidth limitations.

2. DECODING NON-BINARY LDPC CODES

Non-binary LDPC codes are defined by sparse parity-check matrices ($\mathbf{H}$) whose elements are defined over GF(q). The matrix $\mathbf{H}$ is composed of M rows and N columns, corresponding to the number of parity-check equations and the codeblock length, respectively. $\mathbf{H}$ is also the adjacency matrix of the Tanner graph LDPC code representation [10]. In graph notation, columns in $\mathbf{H}$ correspond to Variable Nodes (VNs)

[*]Support by Instituto de Telecomunicações, Fundação para a Ciência e Tecnologia grants PEst-OE/EEI/LA0008/2013 and SFRH/BD/78238/2011.

[†]This work is also supported by the Storage Research Consortium.

and rows to Check Nodes (CNs) that are connected whenever a non-null element exists in $\mathbf{H}$. Furthermore, the set of edges connected to VN_v is denoted by $C(v)$ and those connected to CN_c are denoted by $V(c)$, with cardinality d_v and d_c.

The Sum–Product Algorithm (SPA), also designated as Belief Propagation (BP), can be extended to GF(q) [11]. However, the numerical complexity involved in SPA decoding has lead to the introduction of several sub-optimal variants [11] and also to Fourier representations [12, 7], which lower the decoding complexity. In particular, the FFT–SPA, used in this work, is defined as:

$$\mathbf{p}_v(x) = \mathbf{m}_{vc}^{(0)}(x) = p(v_v = x|y_v) \tag{1}$$

$$\mathbf{m}_{cv}^{(i)}(z) = \prod_{v' \in V_c \setminus v} \text{FWHT}\{\mathbf{m}_{v'c}^{(i-1)}(x)\}$$

$$\mathbf{m}_{cv}^{(i)}(x) = \text{FWHT}\left\{\mathbf{m}_{cv}^{(i)}(z)/\mathbf{m}_{cv}^{(i)}(z=0)\right\} \tag{2}$$

$$\mathbf{m}_{vc}^{(i)}(x) = \mathbf{p}_v(x) \prod_{c' \in C(v) \setminus c} \mathbf{m}_{c'v}^{(i)}(x) \tag{3}$$

$$\mathbf{m}_{v}^{*(i)}(x) = \mathbf{p}_v(x) \prod_{c' \in C(v)} \mathbf{m}_{c'v}^{(i)}(x), \tag{4}$$

and consists of three processing phases: *a)* the channel demodulator feeds the VNs with their corresponding *probability mass function* (*pmf*) $\mathbf{p}_v(x)$, i.e. the vector of probabilities for each symbol in GF(q), and initializes all $\mathbf{m}_{vc}(x)^{(0)} = \mathbf{p}_v(x)$ (1); *b)* (CN processing) CNs process the $\mathbf{m}_{vc}(x)^{(i-1)}$ to compute $\mathbf{m}_{cv}(x)^{(i)}$ (2), with (i) the i-th iteration; *c)* (VN processing) VNs receive their adjacent $\mathbf{m}_{cv}(x)^{(i)}$ and compute $\mathbf{m}_{vc}(x)^{(i)}$ (3), and also compute the *a-posteriori* reliability of VN_v, $\mathbf{m}_v^{*(i)}(x)$ (4), from which the symbol of VN_v may be retrieved [12]. When the decoded word has converged to a valid codeword or a maximum number of iterations has been reached, the decoder iterates between the CN and the VN processing. Due to the definition of parity-check coefficients over GF(q), a permutation of the *pmf*s is required, but for contention was omitted from (1)-(4) [13].

3. FINE-GRAINED PARALLEL NON-BINARY LDPC DECODERS ON FPGAS

Herein, the FFT–SPA mapping on the OpenCL model is detailed. Despite its cross-platform characteristics, the OpenCL programming model shows both limitations and advantages for the particular case of the wide-pipeline accelerators generated by the Altera OpenCL tool that are also discussed next. An overview of the OpenCL parallel programming model can be found in [9].

3.1. Wide-Pipeline LDPC Decoder

The Altera OpenCL model generates wide-pipeline accelerators from the fine-grainded parallel kernels defined by the developer. These accelerators offer high throughput of the work-item processing rate through two features: to a greater extent through the highly-pipelined execution of the work-items; and also through the parallel execution of work-items by different Compute Units (CUs) or Single Instruction Multiple Data (SIMD) processing. The processing rate of the work-items in the pipeline is the inverse of the Initiation Interval (II), where II is defined as:

Algorithm 1 GF(2^m) FFT–SPA wide-pipeline decoder on FPGA. Routines are preceded with [H]ost and [F]PGA.

```
[H] Mem Copy: (Host to FPGA) Copy LUTs and Tanner graph
– – – – – – – – – – (Start simulation) – – – – – – – – – –
[H] Generate codeword and transmit it through AWGN
[H] Mem copy: (Host to FPGA) Copy transmit codeword
– – – – – – – (Start FFT-SPA Decoding) – – – – – – –
while c · H ≠ 0 or maximum iterations reached do
    [F] (cnProc) Execute CN kernel on the Fourier-domain
    [F] (fwht) Execute the FWHT for m_cv messages
    [F] (vnProc) Depermute the m_cv messages, execute the VN kernel and permute m_vc messages
    [F] (fwht) Execute the FWHT for m_vc messages
end while
– – – – – – – – (End FFT-SPA Decoding) – – – – – – – –
[H] Mem copy: (FPGA to Host) Copy decoded word
```

$$\frac{1}{\text{II}} = k_{\text{SIMD}} \times k_{\text{CUs}} \text{ [work-items/clock cycles]}, \tag{5}$$

where $k_{\text{SIMD}} \in \{1, 2, 4, 8, 16\}$ denotes the number of work-items that are vector processed and $k_{\text{CUs}} \in \mathbb{N}$ the number of CUs that compose the accelerator, and II defines the elapsed time, in clock cycles, between two consecutive work-items.

OpenCL also defines a streaming model where data is streamed from one kernel to another as required by the algorithm. Thus, we have defined three kernels for the OpenCL FFT–SPA: the CN (2) and VN (3) processing and also for the Fast Walsh–Hadamard Transform (FWHT) (2). Data is initialized from the host system and sent to the FPGA device where the FFT–SPA decoding occurs, after which it is sent back to the host system as formalized in Algorithm 1, and also pictured in the upper box of Figure 1. In the lower box of Figure 1, the wide-pipeline diagram of the FFT–SPA LDPC decoder is presented showing: the input and output FIFOs for the $\mathbf{p}_v(x)$ and the $\mathbf{m}_v^{*(i)}(x)$ messages; in the upper right corner the initiation of work-items in the pipeline with II$= 1$; and in the bottom left the memory hierarchy of the decoder coupled to the ***cnProc***, ***vnProc*** and ***fwht*** processing units.

A key challenge of OpenCL LDPC decoding is the definition of a suitable variable-grain parallelism that fits both the udnerlying architecture and the decoding algorithm. However, in the particular case of FPGA programming many Single Instruction Multiple Thread (SIMT)-related restric-

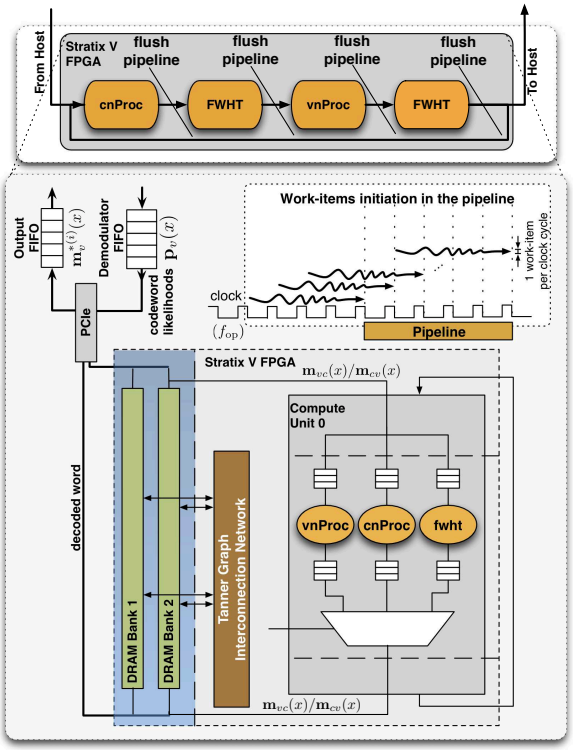

Fig. 1. Data is streamed from the host system to the FPGA, where it is then streamed from each kernel to the next. In the absence of streaming extensions, it is given to its corresponding kernel in a data producer-consumer logic.

tions, that are transversal to Graphics Processing Unit (GPU) programming [9], do not hold. Namely, the performance achieved is no longer dependent on whether the workgroup dimensions are multiple of the warp size [9]. Furthermore, *constant* memory and *local* memory no longer possess hard limits as imposed by a fixed memory hierarchy, but can be adjusted as required by the algorithm expression in OpenCL, limited only to the logic resources of the FPGA. As a consequence, we are able to load all Lookup Tables (LUTs) regarding the Tanner graph nodes' connections in the *constant* memory, regardless of their size.

3.2. Variable-Grain Parallelism

Work-item-per-node granularities in binary LDPC decoding allow obtaining high decoding throughputs [9], while in the non-binary case, the field dimension can be incorporated as an extra level of algorithmic parallelism [7, 8]. Hence, we define a work-item for each node field element and a workgroup for each node. On the FPGA, this strategy is flexible enough to accomodate as few as 4 work-items per workgroup (GF(2^2)) or as high as 256 work-items (GF(2^8)). Altough we are able to define the ***cnProc***, ***vnProc***, and the ***fwht*** kernels' execution grid to launch workgroups composed of 2^m work-items, the ***fwht*** OpenCL kernel can benefit from optimizations carried out for the FFT acceleration on FPGAs [14]. Consequently, we defined two distinct FWHT for each GF(2^m) order tested. A radix-2 factorization whose work-item-granularity is tightly coupled to the LDPC decoding context and another with radix-n tuned to the field's order for maximum throughput.

Furthemore, we employed single floating-point data representation, both scalar and vectorial [7], of the soft-decoding *pmf* messages exchanged. This overcomes the need to explicitly handle fixed-point multiplication, since the OpenCL standard does not offer native fixed-point arithmetic. Notwithstanding, given the nature of the arithmetic operations of the FFT–SPA and a high level of DSP units incorporated in modern FPGAs, floating-point operation is not forbiddingly expensive for a Stratix V FPGA.

3.3. Hardware Resource Optimizations

Extracting the maximum performance out of the available computing resources of the FPGA given certain kernel definitions, involves tuning wide-pipeline concepts for each kernel: the number of CUs defined; whether compiler-driven SIMD processing is defined or manual SIMD is included by using vector data types; compiler-driven logic resource sharing that reuses logic functions in the pipeline, at the cost of reducing the throughput but in the prospect of freeing up resources that can push the number of CUs and SIMD processing to higher levels. We allowed the compiler to search for suitable combinations of the latter parameters aiming for less than 85% logic resource utilization.

4. EXPERIMENTAL RESULTS

The decoder was profiled for a $(2, d_c)$, N=768 symbols LDPC code [7] on a Nallatech PCIe385N D5 FPGA board using the Altera OpenCL Quartus 13sp1 release, for GF(2^m), with m=$\{2, 3, 4\}$. The throughput was computed for the full potential of the wide-pipeline accelerator, i.e. assuming that between kernels there was no pipeline flushing and is denoted by T_{pot} (6), and was also profiled in runtime using OpenCL events denoted by T. T_w is another throughput metric that is given by the OpenCL compiler expressed as the number of processed work-items per second.

$$T_{\text{potential}} = \frac{N \times m}{N_{iter} \times f_{op}} \times \frac{\sum_j W_j}{\sum_j W_j / T_w(j)} \; [bit/s] \qquad (6)$$

In (6), N_{iter} represents the number of decoding iterations execution, W_i is the number of work-items executed by kernel i, with $W_{\textbf{\textit{cnProc}}}$=$W_{\textbf{\textit{vnProc}}}$=$W_{\textbf{\textit{fwht}}}$=$2^m \times N \times d_v$, and f_{op} is the frequency of operation. Also, the second product term is the weighed harmonic mean of the kernels' T_w.

Table 1. Logic resource utilization: rows labelled with * decode 1 codeword and those with † decode 4 codewords at a time.

GF(2^m)	m=2					m=3					m=4				
Logic (%)	LEs+FFs	RAMs	DSPs	P	T_w(M)	LEs+FFs	RAMs	DSPs	P	T_w(M)	LEs+FFs	RAMs	DSPs	P	T_w(M)
cnProc*	10.38	9.09	2.14	(2,1)	483.05	19.26	18.07	4.15	(4,1)	965.52	10.37	9.33	2.14	(2,1)	483.05
FWHT*	13.76	13.26	1.26	(2,1)	500.00	18.39	18.67	1.32	(2,1)	500.00	23.45	24.03	1.38	(2,1)	464.00
vnProc*	31.89	20.66	4.03	(2,2)	966.00	16.13	10.32	2.01	(2,1)	483.00	16.12	10.32	2.01	(2,1)	483.00
Total*	85.23	62.26	7.42	–	587.61	82.86	66.34	7.48	–	587.53	80.33	62.96	5.53	–	476.51
Clock (MHz)*	163.07					188.96					193.16				
T/T_{pot}(Mbit/s)*	1.08/9.79					0.82/7.34					0.68/3.97				
cnProc†	13.77	7.51	12.71	(2,1)	348.00	13,72	13.11	6.67	(2,1)	348.00	10.37	9.33	2.14	(2,1)	483.05
FWHT†	18.65	11.97	0.25	(1,1)	170.45	13.10	9.68	1.38	(1,1)	154.67	27.51	20.26	0.19	(1,1)	108.95
vnProc†	18.84	12.86	4.65	(2,1)	348.00	18.83	12.86	4.65	(2,1)	348.00	16.12	10.33	2.01	(2,1)	483.00
Total†	76.00	56.80	11.57	–	311.90	73.36	54.92	12.70	–	245.65	77.13	57.70	12.96	–	225.24
Clock (MHz)†	206.52					216.07					203.5				
T/T_{pot}(Mbit/s)†	3.36/27.72					1.73/12.28					0.98/7.51				
T (Mbit/s) [7]†	17.44					17.51					11.97				
T (Mbit/s) [15]*	∼60 (est.)														

4.1. Wide-pipeline Acceleration

The logic resource utilization by each kernel and the decoder system, as well as the obtained throughputs, are shown in Table 1. In the top rows, correspoding to the single codeword decoder, the clock frequency of operation grows with the order of GF(2^m). However, this trend is offset by lowering T_w, due to the lack of logic resources to keep a high SIMD level, or a number of CUs greater than 1, which contributes to a lower T_{pot} and an effectively lower T. Namely, for m=$\{2, 3, 4\}$ we obtain, respectively, 1.08, 0.82 and 0.68 Mbit/s out of 9.79, 7.34 and 3.97 Mbit/s potentially achievable. Increasing the parallelism level to 4 codewords decoded at a time results in lower logic availability to keep SIMD and CUs levels high. However, the effective throughput T, and also the potential T_{pot}, increased due to the decrease of T_w at lower rates than the increase of the number of codewords decoded simultaneously. Hence, the decoding throughput obtained for the multi-codeword decoder increases to 3.36, 1.73 and 0.98 Mbit/s, for m=$\{2, 3, 4\}$, while having potential throughputs of 27.72, 12.28 and 7.51 Mbit/s.

4.2. Discussion

Notwithstanding the increase in decoding throughput experienced when moving from single- to multi-codeword systems, the relative efficiency of the wide-pipeline accelerator—the fraction of potential throughput actually delivered—is of 11∼17% in the single-codeword case, and of ∼11% in the multi-codeword case. In other words, while absolute increases in decoding throughput can be accomplished by increasing the data-parallelism levels through vector type operations and data structures, the efficiency of the accelerator is left unchanged. Nonetheless, the potential throughput, albeit non-trivial to realize, as it would likely involve LDPC codes specifically designed for this type of wide-pipeline accelerators and avoiding memory hazards could still add non-negligble overheads, shows that the methodology proposed is inline with both FPGA and proposed GPU solutions.

5. RELATED WORK

The state-of-the-art shows the need for low powered LDPC decoding architectures [16, 17], based on FPGAs, developed at a programming effort close to multicore approaches [7, 8, 15, 18]. In [16] and in [17] the authors focused on minimization of the logic resources utilization by avoiding memory hazards and reducing the footprint of the memory transactions by using the EMS algorithm. In [18], a sequential decoder on GPU that exploits the field's dimension for extracting parallelism within the Extended Min-Sum (EMS) was developed, while in [8] a highly parallel expression of the Min-Max algorithm, similar to the FFT–SPA decoders developed in [7, 15] was explored. Considering that this work stands on the reconfigurable hardware field and in the flexible high-level programming of signal processing algorithms, the design space of LDPC decoders may be thoroughly explored with the proposed methodology.

6. CONCLUSIONS

In this paper we propose a wide-pipeline FFT–SPA decoder for non-binary LDPC codes on FPGA. Namely, an efficient fine-grained algorithm expression has been developed, decoding one or four codewords at a time, that is suitable to the characteristics of a wide-pipeline accelerator. While the effective throughput is at par with the state-of-the-art decoders, which lie in the Mbit/s range, the potential throughput of this kind of architecture surpasses it and promises greater decoding throughputs to be reached in the near future, while keeping the development effort of this class of signal processing algorithms low.

7. REFERENCES

[1] R. G. Gallager, *Low-Density Parity-Check Codes*, MIT Press, Cambridge, 1963.

[2] T. J. Richardson, M. A. Shokrollahi, and R. L. Urbanke, "Design of Capacity-Approaching Irregular Low-Density Parity-

Check Codes," *IEEE Trans. Inform. Theory*, vol. 47, no. 2, pp. 619 –637, feb 2001.

[3] T. Richardson and R. Urbanke, "The Renaissance of Gallager's Low-Density Parity-Check Codes," *IEEE Commun. Mag.*, vol. 41, no. 8, pp. 126–131, 2003.

[4] K. Kasai, R. Matsumoto, and K. Sakaniwa, "Information reconciliation for QKD with rate-compatible non-binary LDPC codes," in *IEEE Symp. on Inf. Theory*, 2010, pp. 922–927.

[5] Ivan B Djordjevic and Bane Vasic, "Nonbinary LDPC codes for optical communication systems," *IEEE Photon. Technol. Lett.*, vol. 17, no. 10, pp. 2224–2226, 2005.

[6] Amin Shokrollahi, "Raptor codes," *IEEE Trans. Inform. Theory*, vol. 52, no. 6, pp. 2551–2567, 2006.

[7] J. Andrade, G. Falcao, V. Silva, and Kenta Kasai, "FFT-SPA non-binary LDPC decoding on GPU," in *Proc. IEEE ICASSP*, 2013, pp. 5099–5103.

[8] Guohui Wang, Hao Shen, Bei Yin, Michael Wu, Yang Sun, and Joseph Cavallaro, "Parallel Nonbinary Decoding on GPU," in *Proc. IEEE ASILOMAR*, Nov. 2012.

[9] G. Falcao, V. Silva, L. Sousa, and J. Andrade, "Portable LDPC Decoding on Multicores Using OpenCL," *IEEE Signal Processing Mag.*, vol. 29, no. 4, pp. 81–109, 2012.

[10] Michael Tanner, "A Recursive Approach to Low Complexity Codes," *IEEE Trans. Inform. Theory*, 1981.

[11] D. Declercq and M. Fossorier, "Decoding Algorithms for Nonbinary LDPC Codes Over GF ," *IEEE Trans. Commun.*, vol. 55, no. 4, pp. 633 –643, april 2007.

[12] Valentin Savin, "Fourier Domain Representation of Nonbinary LDPC Codes," *IEEE Symp. on Inf. Theory*, pp. 2541–2545, July 2012.

[13] Rolando Antonio Carrasco and Martin Johnston, *Non-Binary Error Control Coding for Wireless Communication and Data Storage*, Wiley, Chichester, 2008.

[14] Joao Andrade, Vitor Silva, and Gabriel Falcao, "From OpenCL to gates: The FFT," in *Proc. IEEE GlobalSIP*, Dec 2013, pp. 1238–1241.

[15] Moritz Beermann, Enrique Monz, Laurent Schmalen, and Peter Varyx, "High Speed Decoding of Non-Binary Irregular LDPC Codes Using GPUs," in *Proc. IEEE SiPS*, 2013.

[16] Yaoyu Tao, Youn Sung Park, and Zhengya Zhang, "High-throughput Architecture and Implementation of Regular (2,d_c) Nonbinary LDPC Decoders," in *Proc. IEEE ISCAS*, 2012, pp. 2625–2628.

[17] Emmanuel Boutillon, Laura Conde-Canencia, and Ali Al Ghouwayel, "Design of a gf (64)-ldpc decoder based on the ems algorithm," *IEEE Trans. Circuits Syst.*, 2013.

[18] D.L. Romero and N.B. Chang, "Sequential decoding of non-binary ldpc codes on graphics processing units," in *Proc. IEEE ASILOMAR*, 2012, pp. 1267–1271.