List of 10 selected publications:

Variable-Node-Shift Based Architecture for Probabilistic Gradient Descent Bit Flipping on QC-LDPC Codes

Khoa Le, David Declercq, *Senior Member, IEEE*, Fakhreddine Ghaffari, *Member, IEEE*, Lounis Kessal, *Member, IEEE*, Oana Boncalo, *Member, IEEE*, and Valentin Savin, *Member, IEEE*

***Abstract*— Probabilistic gradient descent bit-flipping (PGDBF) is a hard-decision decoder for low-density parity-check (LDPC) codes, which offers a significant improvement in error correction, approaching the performance of soft-information decoders on the binary symmetric channel. However, this outstanding performance is known to come with an augmentation of the decoder complexity, compared to the non-probabilistic gradient descent bit flipping (GDBF), becoming a drawback of this decoder. This paper presents a new approach to implementing PGDBF decoding for quasi-cyclic LDPC (QC-LDPC) codes, based on the so-called variable-node-shift architecture (VNSA). In VNSA-based PGDBF implementations, the regularity of QC-LDPC connection networks is used to cyclically shift the memory of the decoder, leading to the fact that, a variable node (VN) is processed by different computing units during the decoding process. With this modification, the probabilistic effects in VN operations can be produced by implementing different types of processing units, without requirement of a probabilistic signal generator. The VNSA is shown to further improve the decoding performance of the PGDBF, with respect to other hardware implementations reported in the literature, while reducing the complexity below that of the GDBF. The efficiency of the VNSA is proven by ASIC synthesis results and by decoding simulations.**

***Index Terms*— Low-density parity-check, quasi-cyclic, probabilistic gradient descent bit flipping, random generation, low complexity implementation, high throughput decoder.**

I. Introduction

LOW-DENSITY Parity-Check (LDPC) codes have been increased research interest over the last years, due to their near-capacity error correction performance and their manageable complexity implementations [1]. Hence, they have been used in many practical applications, such as storage (hard drives, flash memories, *etc.*) [2]–[4], or wireless transmission standards such as IEEE 802.3an 10GBASE-T, IEEE 802.11n, 802.16a [5]–[7]. LDPC codes can be effectively decoded by either soft decision decoding algorithms, such as Belief Propagation (BP) or Min Sum (MS), providing very good error correction capability but the computation complexity is very high, or the hard-decision decoding algorithms, such as Bit Flipping (BF) or Gallager-A/B, which are very low in complexity but a weak performance is observed [1]. LDPC decoding algorithms follow an iterative manner where the messages are iteratively passed between two groups of nodes: Variable Nodes (VNs) and Check Nodes (CNs). The low complexity of hard-decision decoders comes from the fact that, only binary messages are iteratively passed between the VNs and CNs during the decoding process, which simplifies the connection network of the decoders. In addition, the computation units to process these binary messages, are very simple. This paper is concerned with the Bit Flipping (BF) hard decision decoder where, contrary to message passing hard decision decoders, both extrinsic and intrinsic information are passed between the computation nodes.

The conventional BF decoder, introduced by Gallager in 1963 [8], is a very low complexity decoder but being weak in performance, compared to soft-decision decoders. Many modifications have been introduced to fill this performance gap, such as Weighted BF (WBF) [9], Modified WBF (MWBF) [10], Improved MWBF (IMWBF) [11], especially the class of the Gradient Descent Bit Flipping (GDBF) [12] and its variants such as Probabilistic Gradient Descent Bit Flipping (PGDBF) [13], Noisy Gradient Descent Bit Flipping (NGDBF) [14]. The differences between these BF decoders can be summarized by the difference from the computation of the so-called *inversion function* (or *energy function*) and from the mechanism of choosing the VNs to flip at each iteration. In the conventional BF, at each iteration, the energy value of every VN is given by the number of unsatisfied neighboring CNs, and the VNs that have energy lower than a pre-defined threshold will be flipped. By this energy computation, the conventional BF considers the contribution of neighbor CNs to VN energy value equally. Several modifications such as Weighted BF (WBF) [9], Modified WBF (MWBF) [10], Improved MWBF (IMWBF) [11] *etc.*, have put

Manuscript received August 28, 2017; revised November 7, 2017; accepted November 19, 2017. Date of publication December 18, 2017; date of current version May 29, 2018. This work was supported in part by French ANR through the NAND Project under Grant ANR-15-CE25-0006-01 and in part by the Franco-Romanian (ANR-UEFISCDI) Joint Research Program (DIAMOND Project). This paper was recommended by Associate Editor I. Kale. *(Corresponding author: Khoa Le.)*

K. Le, D. Declercq, F. Ghaffari, and L. Kessal are with ETIS, UMR8051, Université Paris Seine, Université de Cergy-Pontoise, ENSEA, CNRS, 95014 Cergy-Pontoise, France (e-mail: khoa.letrung@ensea.fr; declercq@ensea.fr; fakhreddine.ghaffari@ensea.fr; kessal@ensea.fr).

O. Boncalo is with the Computer Engineering Department, University Politehnica Timisoara, Timisoara 300006 Romania (e-mail: oana.boncalo@cs.upt.ro).

V. Savin is with CEA-LETI, MINATEC, 38054 Grenoble, France (e-mail: valentin.savin@cea.fr).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSI.2017.2777802

Layered LDPC Decoders With Efficient Memory Access Scheduling and Mapping and Built-In Support for Pipeline Hazards Mitigation

Oana Boncalo, Gyorgy Kolumban-Antal, Alexandru Amaricai, Valentin Savin, *Member, IEEE*, and David Declercq, *Senior Member, IEEE*

***Abstract*— This paper proposes a holistic approach that addresses both the message mapping in memory banks and the pipeline-related data hazards in low-density parity-check (LDPC) decoders. We consider a layered hardware architecture using single read/single write port memory banks. The throughput of such an architecture is limited by memory access conflicts, due to improper message mapping in the memory banks, and by pipeline data hazards, due to delayed update effect. We solve these issues by: 1) a residue-based layered scheduling that reduces the pipeline related hazards and 2) off-line algorithms for optimizing the message mapping in memory banks and the message read access scheduling. Our estimates for different LDPC codes indicate that the hardware usage efficiency of our layered decoder is improved by 3%–49% when only the off-line algorithms are employed and by 16%–57% when both the residue-based layered architecture and the off-line algorithms are used.**

***Index Terms*— Bank allocation, graph coloring, LDPC QC-LDPC, in-place memory mapping, pipeline conflicts.**

I. Introduction

DUE to their error correction performance close to the Shannon limits, Low-Density Parity-Check (LDPC) codes have been adopted in multiple wireless communication standards (such as WiFi [1], WPAN [2], or WiMAX [3]), digital video broadcasting (DVB-S2-eXtended [4]), or in flash memory applications [5]. A typical LDPC decoder architecture consists of memory modules, interconnection networks, as well as parallel processing units. A key property of LDPC decodes is represented by their capability to accommodate different degrees of parallelism yielding different throughput-cost trade-offs. In this paper, we study off-line algorithms to optimize the memory access, combined with built in architecture support for mitigating pipeline related hazards, targeting Quasy-Cyclic Low-Density Parity-Check (QC-LDPC) decoders using layered scheduling [6].

Increasing the throughput in layered LDPC architectures can be achieved by increasing the level of parallelism at the processing unit level, which implies a larger number of simultaneous memory accesses. Furthermore, message access pattern is code dependent, and a message memory mapping is needed, such that the simultaneous memory accesses are conflict free. A key design choice is the number of ports used for the memory building blocks. Single port memories are the low cost option independent of the technology choice. In addition to this, for Field Programmable Gate Array (FPGA) devices, the cost-efficient storage resource – the Block RAM (BRAM) – does not allow more than two ports. If multi-port memories are employed, mapping is straightforward. If single port memory banks are used, then access conflicts may appear as a result of multiple messages that need to be accessed simultaneously from the same bank. The layered decoder architecture considered in this work has a multiple single port memory bank organization. Therefore, we propose message mapping off-line algorithms that try to minimize the stall clock cycles needed for message read access serialization, due to access conflicts, at the memory bank interface.

Another speedup knob is increasing the working frequency by pipelining the design. However, this leads to delayed message write-backs in memories, and thus, to pipeline related hazards, such as Read After Write (RAW). Solving RAW hazards requires pipeline stalls that introduce stall clock cycles for all memory read interfaces. As a result, the iteration processing latency increases, decoder throughput decreases, while the hardware resource usage efficiency degrades.

Reduction of stall cycles can be obtained by employing off-line strategies, for both message mapping in memory banks (for reducing memory access conflicts), and message read scheduling (for reducing pipeline related hazards). Approaches in [7]–[10] have dealt with the message mapping problem by translating it into an equivalent graph coloring formulation. The aforementioned papers do not address issues related to pipeline. Approaches in [11]–[14] proposed message read re-ordering strategies in order to mitigate the stall clock cycles introduced for avoiding hazards. These works only consider the case when parallelism at the processing unit level is 1 (*i.e.* one message is inputted per clock cycle to the processing unit), which requires a single memory bank. Hence, there is no message mapping issue.

Manuscript received July 1, 2018; revised October 11, 2018 and November 19, 2018; accepted November 25, 2018. Date of publication December 17, 2018; date of current version March 15, 2019. This work was supported by the Franco-Romanian (ANR-UEFISCDI) Research Programme Project—DIAMOND. This paper was recommended by Associate Editor M. Martina. *(Corresponding author: Oana Boncalo.)*

O. Boncalo, G. Kolumban-Antal, and A. Amaricai are with University Politehnica Timişoara, 300006 Timisoara, Romania (e-mail: oana.boncalo@cs.upt.ro; alexandru.amaricai@cs.upt.ro).

V. Savin is with the CEA-LETI, MINATEC Campus, 38054 Grenoble, France (e-mail: valentin.savin@cea.fr).

D. Declercq is with ETIS, UMR8051, Universite Paris Seine, Universite de Cergy-Pontoise, ENSEA, CNRS, 95014 Cergy-Pontoise, France (e-mail: declercq@ensea.fr).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSI.2018.2884252

Analysis and Design of Cost-Effective, High-Throughput LDPC Decoders

Thien Truong Nguyen-Ly, Valentin Savin, Khoa Le, David Declercq, *Senior Member, IEEE*, Fakhreddine Ghaffari, and Oana Boncalo

***Abstract*— This paper introduces a new approach to cost-effective, high-throughput hardware designs for low-density parity-check (LDPC) decoders. The proposed approach, called nonsurjective finite alphabet iterative decoders (NS-FAIDs), exploits the robustness of message-passing LDPC decoders to inaccuracies in the calculation of exchanged messages, and it is shown to provide a unified framework for several designs previously proposed in the literature. NS-FAIDs are optimized by density evolution for regular and irregular LDPC codes, and are shown to provide different tradeoffs between hardware complexity and decoding performance. Two hardware architectures targeting high-throughput applications are also proposed, integrating both Min-Sum (MS) and NS-FAID decoding kernels. ASIC post synthesis implementation results on 65-nm CMOS technology show that NS-FAIDs yield significant improvements in the throughput to area ratio, by up to 58.75% with respect to the MS decoder, with even better or only slightly degraded error correction performance.**

***Index Terms*— Error correction, high throughput, low-density parity-check (LDPC) codes, low cost, nonsurjective finite alphabet iterative decoder (NS-FAID).**

I. Introduction

THE increasing demand of massive data rates in wireless communication systems will require significantly higher processing speed of the baseband signal, as compared with conventional solutions. This is especially challenging for forward error correction (FEC) mechanisms, since FEC decoding is one of the most computationally intensive baseband processing tasks, consuming a large amount of hardware resources and energy. The use of very large bandwidths will also result in stringent, application-specific, requirements in terms of both throughput and latency. The conventional approach to increase throughput is to use massively parallel architectures. In this context, low-density parity-check (LDPC) codes are recognized as the foremost solution, due to the intrinsic capacity of their decoders to accommodate various degrees of parallelism. They have found extensive applications in modern communication systems, due to their excellent decoding performance, high-throughput capabilities [1]–[4], and power efficiency [5], [6], and have been adopted in several recent communication standards.

This paper targets the design of cost-effective, high-throughput LDPC decoders. One important characteristic of LDPC decoders is that the memory and interconnect blocks dominate the overall area/delay/power performance of the hardware design [7]. To address this issue, we build upon the concept of finite alphabet iterative decoders (FAIDs) introduced in [8]–[10]. While FAIDs have been previously investigated for variable-node (VN) regular LDPC codes over the binary symmetric channel, this paper extends their use to any channel model and to both regular and irregular LDPC codes.

The approach considered in this paper, referred to as nonsurjective finite FAIDs (NS-FAIDs), is to allow storing the exchanged messages using a lower precision (a smaller number of bits) than that used by the processing units. The basic idea is to reduce the size of the exchanged messages, once they have been updated by the processing units. Hence, to some extent, the proposed approach is akin to the use of imprecise storage, which is seen as an enabler for cost and throughput optimizations. Moreover, NS-FAIDs are shown to provide a unified framework for several designs previously proposed in the literature, including the normalized and offset Min-Sum (OMS) decoders [11], [12], the partially OMS (POMS) decoder [13], the MS-based decoders proposed in [14] and [15], or the recently introduced dual-quantization domain MS decoder [16].

This paper refines and extends some of the concepts we previously introduced in [17] and [18]. In particular, the definition of NS-FAIDs [17] is extended such as to cover a larger class of decoders, which is shown to significantly improve the decoding performance in case that the exchanged messages are quantized on a small number of bits (e.g., 2 bits per exchanged message). We show that NS-FAIDs can be optimized by using the density evolution (DE) technique, so as to obtain

Manuscript received April 10, 2017; revised July 13, 2017 and October 2, 2017; accepted November 4, 2017. Date of publication December 14, 2017; date of current version February 22, 2018. This work was supported in part by the European H2020 Work Program under Project Flex5Gware and in part by the Franco-Romanian (ANR-UEFISCDI) Joint Research Program Blanc-2013 under Project DIAMOND. *(Corresponding author: Valentin Savin.)*

T. T. Nguyen-Ly is with CEA-LETI, MINATEC Campus, 38054 Grenoble, France, and also with ETIS ENSEA/UCP/CNRS UMR-8051, 95014 Cergy-Pontoise, France (e-mail: thientruong.nguyen-ly@cea.fr).

V. Savin is with CEA-LETI, MINATEC Campus, 38054 Grenoble, France (e-mail: valentin.savin@cea.fr).

K. Le, D. Declercq, and F. Ghaffari are with ETIS ENSEA/UCP/CNRS UMR-8051, 95014 Cergy-Pontoise, France (e-mail: khoa.letrung@ensea.fr; declercq@ensea.fr; ghaffari@ensea.fr).

O. Boncalo is with the Computers and Information Technology Department, University Politehnica Timisoara, 300006 Timisoara, Romania (e-mail: boncalo@cs.upt.ro).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TVLSI.2017.2776561

Microprocessors and Microsystems 63 (2018) 216–225

Contents lists available at ScienceDirect

Microprocessors and Microsystems

journal homepage: www.elsevier.com/locate/micpro

Code-design for efficient pipelined layered LDPC decoders with bank memory organization

Oana Boncalo[a,*], Gyorgy Kolumban-Antal[a], David Declercq[b], Valentin Savin[c]

[a] Politehnica University of Timisoara, Timisoara, Romania
[b] ENSEA, Cergy-Pontoise, France
[c] CEA-LETI, MINATEC Campus, Grenoble, France

ARTICLE INFO

Article history:
Received 19 February 2018
Revised 20 July 2018
Accepted 21 September 2018
Available online 25 September 2018

Keywords:
Parity check codes
Decoding
Channel coding
Progressive edge growth
Layered
Graph coloring
Architectures

ABSTRACT

This paper presents an architecture-aware Progressive Edge Growth (PEG)-based construction method for Low-Density Parity-Check (LDPC) codes. We target optimization through code construction for layered architectures with pipelined processing and memory organized in single-port banks. For a given layered Quasy-Cyclic Low-Density Parity-Check (QC-LDPC) decoder architecture configuration, the code constraints need to maximize hardware usage efficiency. Implementation results for Field-Programmable Gate Array (FPGA) technology suggest that the codes obtained using the proposed algorithm have a throughput increase of 39% up to 110%, due to the increase in working frequency obtained by using pipeline.

1. Introduction

A key objective in the development of LDPC decoding architecture is to obtain the maximum throughput, given a fixed resource budget. This goal can be supported by employing architecture aware code construction techniques. In this paper, we analyse the input parameter constraints needed for custom code design, such that the maximum theoretical Hardware Usage Efficiency (HUE) of layered QC-LDPC decoding architectures is possible. QC-LDPC codes are a class of structured LDPC codes that are typically employed in standards [1]. We target LDPC decoder architectures employing layered scheduling, due to their increased convergence, and improved memory footprint of their hardware implementation [2].

We consider a layered scheduling decoder having memory organized in multiple single read/single write port banks, and pipelined processing units. Multiple bank organization is motivated by the cost efficiency of the single port Static Random Access Memorys (SRAM-s) with respect to the multi-port one, for both FPGA and Application-Specific Integrated Circuit (ASIC) technologies. Employing this technique allows higher working frequencies, due to the aditional pipeline registers. Therefore, throughput is also increased.

The decoder architecture at hand is affected by two types of data hazards:

- In a multiple bank based memory organization setting, conflicts appear when at least two messages have to be read/written in the same bank during the same clock cycle. In this case, the memory access has to be serialized by introducing stall cycles.
- The pipeline technique can cause Read After Write (RAW) data hazards that also require the introduction of stall cycles [3].

Therefore, the throughput for a layered LDPC decoder can be significantly reduced, due to the stall clock cycles required for solving the two problems. Reduction of these clock cycles can be obtained by employing bank mapping algorithms – for memory conflicts –, and message access scheduling strategies – for pipelined hazards. However, due to the LDPC code structure, these strategies may not completely remove the clock cycles.

In this paper, we propose a novel LDPC code construction algorithm, based on PEG method [4], called Architecture-aware Layered Progressive Edge Growth (AL-PEG), that allow efficient memory mapping and message access scheduling for layered based decoding architectures. Code construction that addresses solely the pipeline RAW hazards in LDPC decoders has been proposed in [5]. However, in this work, we address also address the memory message mapping in several single-port memory banks.

The main contributions of this paper are:

* Corresponding author.
E-mail address: oana.boncalo@cs.upt.ro (O. Boncalo).

https://doi.org/10.1016/j.micpro.2018.09.011
0141-9331/

2017 IEEE Computer Society Annual Symposium on VLSI

Ultra High Throughput Unrolled Layered Architecture for QC-LDPC Decoders

Oana Boncalo
University Politehnica Timisoara
Timisoara, Romania
Email: oana.boncalo@cs.upt.ro

Alexandru Amaricai
University Politehnica Timisoara
Timisoara, Romania
Email: amaricai@cs.upt.ro

***Abstract*—This paper proposes a layered decoder architecture for array QC-LDPC codes which targets tens of Gbps data rates. It relies on layer unrolling with pipeline stages in between layers, allowing simultaneous decoding of multiple layers. The most important features of the proposed decoder are: (i) fully parallel processing units within each layer (ii) hardwired layer interconnect that allows the removal of high cost variable shift units, (iii) A posteriori log-likelihood ratio (AP-LLR) message memory type of storage is replaced by in-between layers pipeline registers, as the messages are being forwarded from one layer to the next. Data dependencies is this masively parallel structure is resolved by decoding multi-codewords at time. Hence, the proposed architecture allows optimum throughput/cost ratio. FPGA based implementation results indicate that for a 1296 bits LDPC code with 3 layers, throughput of up to 62 Gbps for an average of 4 iterations is by obtained using the proposed architecture.**

***Keywords*-FPGA; LDPC decoder; Layered scheduling;**

I. Introduction

Future wireless communication systems will require increased throughput, with data rates in the order of tens or hundreds of giga-bits per second (Gbps). Due to their high parallelism degree, Low Density Parity Check (LDPC) codes are suitable candidates for ensuring the forward error correction (FEC) of future communication standards. Another emerging application domain of LDPC codes is modern NAND based Flash storage drives. A more than 10x increased lifespan of these devices [1] has been reported due to these FEC scheme. Both multi-Gbps wireless communication and solid state drives FEC systems share two important requirements: high coding rates and high data throughput.

LDPC codes are a class of linear algebraic codes, with channel approaching capability, when large codewords are used (sizes of several thousands or ten of thousands of bits for a codeword) [2]. The architecture space area allows a variety of choices of the number of processing nodes and of the interconnection routes, leading to intricate architectures that offer different trade-offs between the area and the throughput outcomes.

Quasi Cyclic LDPC (QC-LDPC) codes are a sub-class of structured LDPC codes, suitable for hardware implementations [3]. For this sub-class, two main types of scheduling exist: layered and flooding. The layered scheduling has two important advantages with respect to the flooding: increased convergence – less iterations for the decoding process –, and less memory requirements. [4].

This work proposes the development of ultra high throughput layered LDPC architectures, by applying two simple, but effective techniques: loop unrolling and pipelining. Unrolling is applied at layer level – layer unrolling– , while pipeline stages are inserted between layers. We optimize the interconnect for each layer, by hard wiring it to the required shift amount; thus, we remove the variable length shifting networks such as the typical barrel shifters. Regarding the memory organization, it is built out of shift-registers for the check node messages, and from flip-flops, that make out the in between-layers pipeline registers for the AP-LLR messages. Due to pipelining, multiple codewords can be processed simultaneously. Hence, the throughput is increased by of factor equal to the number of layers. This architecture is appropriate for codes with small number of layers, which have zero or a very small number of non-zero elements on each row of the code base matrix. This is the case for the majority of the high rate standard codes. This requirement ensures efficient usage of in-between layer pipeline registers, as the majority of the buffered AP-LLR messages will be used by the processing units associated with the corresponding layer.

The remaining of this paper is organized as follows: Section II offers a short overview of QC-LDPC layered decoding principle, Section III discuses the main layered decoding architecture trade-offs with respect to prior art, Section IV presents the proposed decoder, and results, and Section V provides some closing remarks.

II. Layered decoding: notations and overview

QC-LDPC codes are a class of LDPC codes obtained by expanding the base matrix B by an expansion factor z [3]. The LDPC code has a length of $N = z \times n_{col}$, with n_{col} denoting the number of columns of B. The parity check matrix H can be obtained in the following way: every element of B is replaced by a square matrix that is either the zero matrix for -1 entries of B, or the identity shifted by the non-negative entry value of B. An equivalent view of the H matrix is the graph representation referred to as the Tanner graph. It is a bipartite graphs, with two types of nodes, interconnected by edges. Lines from H, denoted

2159-3477/17 $31.00 © 2017 IEEE
DOI 10.1109/ISVLSI.2017.47

Unrolled Layered Architectures For Non-Surjective Finite Alphabet Iterative Decoders

Oana Boncalo
University Politehnica Timisoara
Timisoara, Romania
oana.boncalo@cs.upt.ro

Valentin Savin
CEA-LETI, MINATEC Campus
Grenoble, France
valentin.savin@cea.fr

Alexandru Amaricai
University Politehnica Timisoara
Timisoara, Romania
oana.boncalo@cs.upt.ro

***Abstract*—This paper proposes cost efficient very high throughput layered decoding architecture for array quasi-cyclic Low-Density Parity-Check (QC-LDPC) codes, targeting tens of Gbps data rates. The targeted throughput is achieved by employing layer unrolling, with pipeline stages inserted in between layers. In order to obtain improved hardware efficiency for the decoder, multiple codewords are processed simultaneously. This leads to increased memory overhead. In order to reduce the associated cost, approximate message storage using Non Surjective-Finite Alphabet Iterative Decoding (NS-FAID) compression tables is employed. Cost efficiency is achieved by hardwired interconnects, compressed message storage using the NS-FAID, as well the A Posteriori Log-Likelihood Ratio (AP-LLR) message memory removal. Several compression tables are evaluated using the Throughput to Area Ratio (TAR) metric.**

***Index Terms*—Forward Error Correction, LDPC Decoding, Layered Scheduling, NS-FAID**

I. Introduction

Forward Error Correction (FEC) represents an important component in both wireless communication systems and NAND based Flash storage devices. With ever increasing data rate requirements, novel approaches on both algorithmic and architectural level are required to accommodate this trend. Low Density Parity Check (LDPC) codes are a class of linear algebraic codes, with decoding performance close to the channel capacity and increased potential for highly parallel decoding architectures [1]. Therefore, improved error correction capability at very high throughputs can be obtained for these codes. LDPC decoding is performed using message passing algorithms, such as Min-Sum (MS) and its variants [2]. These algorithms rely on passing soft messages (messages quantized on a number of bits) along a bipartite graph - Tanner graph [3] - between two types of nodes - variable nodes(VN) and check node(CN) - in several iterations. The hardware architectures implementing message passing LDPC decoding are composed of processing units, interconnection networks and memory blocks. The number and the size of these components are dependent on the desired area and parallelism, as well as on the given code.

Decoding can be performed using two types of scheduling strategies: flooded and layered. The latter has two important advantages: (i) increased convergence, which results in reduced number of iterations [4], and (ii) reduced memory requirements [5]. The increased convergence is due to the fact that AP-LLR message updates are performed after each layer; thus, updating of these message is performed several times during one iteration. In order to increase the throughput of layered decoding architectures, layer unrolling has been proposed in [6]. Combined with multiple codeword processing, this technique has shown that throughputs of tens of Gbps can be obtained. Unrolled layered architectures are suitable for very high rate codes, as well as array LDPC codes.

In this paper, we aim to reduce the memory requirements for unrolled layered decoders by employing Non Surjective Finite Alphabet Iterative Decoding. NS-FAID represents a method that builds on the classical MS, while performing a tight fitting on message storage [7]. The basic idea is to have two working domains in terms of quantization during the iterative processing. The crossing between the two domains is made by means of the framing and de-framing functions. From the memory storage perspective, these are in fact compression/decompression tables.

This work evaluates the impact of using NS-FAID inside a high throughput layer unrolled decoder architecture implementation of a (3,12) regular array code. First, we need to derive the NS-FAID tables for this code, using the methodology proposed in [8]. Next, we assess the NS-FAID efficiency by implementing the corresponding FPGA unrolled layered architecture using the framing/deframing tables obtained earlier. Synthesis results for Xilinx Virtex-7 devices indicate improvements of up to 30% in logic resource usage with respect to MS based decoders, and an increase of TAR between 25% and 125%.

This paper is organized as follows: the theoretical background regarding layered decoding of LDPC codes using NS-FAID is described in Section II; the proposed unrolled layered architecture using NS-FAID based processing units is presented in Section III; Section IV is dedicated to FPGA synthesis estimates.

II. Theoretical background

A. Layered decoding of QC-LDPC codes

LDPC codes are a class of linear algebraic codes, defined by the parity check matrix H. Quasi-cyclic LDPC (QC-LDPC) are a sub-class of LDPC codes [9], for which the parity check matrix is obtained by expanding each element of the base matrix B with a square matrix of size $z \times z$ – with z

978-1-5386-2844-7/17/$31.00 2017 IEEE

Template-Based QC-LDPC Decoder Architecture Generation

Oana Boncalo, Petru Florin Mihancea, Alexandru Amaricai
University Politehnica Timisoara
Timisoara, Romania
{oana.boncalo, petru.mihancea, alexandru.amaricai} @cs.upt.ro

***Abstract*— This paper presents an automated template-based approach for layered QC-LDPC architectures. The underlying idea is to be able to generate any number of fairly optimized hardware designs with only a minimum effort required by tuning high level parameters (parity check matrix, number of AP-LLR messages being processed at a time, etc.). Having chosen a partially parallel architecture, template design effort has been concentrated in two directions: distributed control across the LDPC decoder for modularity and versatility, and datapath design in order to support any given parallelism at processing node level. All user inputs are propagated by the proposed automated flow for all design phases: (i) Verilog HDL LDPC architecture (ii) test-benches and verification environments (iii) evaluation step (BER/FER, average iterations, cost, throughput) of the LDPC decoder architecture. We present the results for the architectures generated for $dv = 3, dv = 4$ regular QC-LDPC code, as well as for the WiMAX (1152,2304) irregular code, with different levels of parallelism at processing node level.**

Keywords—template; layered QC-LDPC; design exploration;

I. INTRODUCTION

Time, space and energy optimization of LDPC decoders is of main concern for today hardware industry [5]. However, this kind of investigation requires the usage of a wide range of distinct tools for hardware description and architectural variation specification, testing, simulation, synthesis, etc. [1][2][3][4][5]. In this paper, we present an environment to support design parameter exploration for LDPC decoders. It includes a tool for automatic generation of customized decoder architectures, as well as the integration of well-known industrial tools for verification and implementation (Modelsim, Xilinx ISE). As pressure is ever-increasing for design productivity, people are reluctant to try-out new languages/methodologies. We have opted for a template based design with some smart tags for specifying information/tuning architecture parameters that would shorten the learning curve in an already established process, but, on the other hand, would support the extra flexibility needed. Hence, the template approach tries to act as extension of the existing Verilog HDL constructions for modeling variability.

The emphasis when building the QC-LDPC template architecture has been: *(i)* ***IP reuse by versatile building blocks*** – design has been intended as modular as possible in a bid to extend and/or reuse some of the building blocks for new templates; furthermore, the goal is to have fairly optimized and parametrized sub-design modules that target good throughput and high hardware (HW) utilization efficiency (HUE), while taking into account the characteristics of FPGA technology; *(ii)* ***verification environment*** reuse - a key ingredient that drives the acceptance of the overall environment is the System Verilog verification environment that co-simulates C++ models of decoders with their HDL architectures; *(iii)* ***automatized flow*** that integrates both academic and commercial tools covering all the steps from generation to actually running the processes (e.g. simulation, implementation).

The contributions of this paper are: *(i) automated flow with centralized parameter tuning* (one configuration file) that propagates user options throughout the entire process; *(ii) modular architecture with distributed control*; a key ingredient is datapath parallelism *p*, which states the number of parallel AP-LLR messages processed at one time; (iii) *efficient memory utilization* (multi-codeword approach, packed AP-LLR messages, serialization of check-node message accesses). Although the generated architecture is fairly optimized, the focus of this work is to provide an efficient way to generate HW designs, which can be validated through co-simulation with a higher level model for LDPC codes. By using the proposed methodology, weeks of design and verification time can be saved by not manually re-doing the work needed to customize: the Verilog HDL design files, the testbench files, the scripts for simulation, synthesis, and implementation, the higher level model inputs.

This paper is organized as follows. Section II summarizes related approaches. Section III offers a brief introduction of QC-LDPC codes. It then presents the proposed decoder template architecture with its sources of variability. Section IV discusses the build flow for the design exploration environment, the process and the customization tool. Results for both regular and irregular codes are presented in Section V, while Section VI is dedicated to concluding remarks.

II. RELATED WORK

Several approaches for generating QC-LDPC decoders have been proposed. These rely either on HDL synthesis performed by commercial tools, or in-house tools to derive the HDL from an untimed, typically C descriptions [3][5]. Both approaches consider layered scheduling; the approach in [3] uses Vivado HLS, while [5] relies on an in-house tool called PICO. The

Check node unit for LDPC decoders based on one-hot data representation of messages

O. Boncalo✉, A. Amaricai, V. Savin, D. Declercq and F. Ghaffari

A novel check node unit architecture for low-density parity check (LDPC) decoders, which avoids the usage of carry-based comparators for the computation of the required first and second minimum values, is presented. It relies on a one-hot representation of the input messages' magnitude, obtained by q-to-2^q decoders. The two minimums are computed using an OR tree and a modified leading zero counter. The proposed architecture is imprecise, as the second minimum is not computed correctly when it is equal to the first one. The implementation results and the analysis of the error correction capability show that the proposed imprecise unit is highly suited for high rate LDPC codes; it presents up to 30% better hardware cost, a higher working frequency, while the loss of the decoding capability is negligible with respect to standard implementations.

Introduction: low-density parity check (LDPC) codes have been widely adopted in many communication and storage systems, because of their excellent performance under iterative message passing algorithms, such as the belief propagation [1] or the min-sum (MS) decoders [2]. For wired transmissions or storage systems, high-throughput decoders for high rate LDPC codes are often used, and the target throughput requirements (from 10 to 30 Gbit/s [3]) must be achieved with low-complexity decoders.

Among the low-complexity decoding algorithms, MS is probably the most efficient solution, since it provides a good trade-off between coding gain and complexity. The check node unit (CNU) update equation of the MS algorithm requires the search of the two smallest values (first and second minimum) among the check node degree, d_c, input messages [4, 5] and the first minimum index. For high rate applications, where the d_c is very large, this leads to large hardware complexity (interconnects and processing units), especially when fully parallel solutions are used. A typical CNU implementation uses two modules: a sorting module, which arranges the input pairs, and a compare-select tree, based on a compare-select module [4, 5]. The latter block selects two minimum out of the incoming inputs. Both the sorting and the compare-select module use carry propagate comparators and multiplexers. To further improve the implementations of MS decoders, several methods to approximate the second minimum with reduced hardware resources at the check node have been recently proposed [6–9]. Most of these techniques rely on the estimation of the second minimum value by a modified version of the first minimum value [6, 7] and/or on the partial hardware reuse of the circuit used to compute the first minimum [8, 9].

In this Letter, we propose a novel CNU implementation for computing the first and second minimum, which always computes accurately the first minimum value, but computes the second minimum in an imprecise manner. The main novelty is represented by the fact that we employ 'a one-hot representation of the magnitude values that reduces the comparators tree to an OR tree'. The two minimums are obtained using a bitwise OR operation between the new representation of the input messages and by employing a modified leading zero counter (LZC). Hence, the usage of carry propagate comparators is no longer required. With respect to [6–9], our approach is different, as we do not rely on the first minimum value to infer the value of the second minimum.

Proposed CNU: The proposed CNU (as depicted in Fig. 1) uses the following building blocks: q-to-2^q decoders, the OR reduction tree block, modified LZC and first minimum index computation.

This solution relies on an alternative representation of the magnitude values: an array of bits with only one non-zero bit position corresponding to the input value (e.g. the 3-bit value 2 is represented by the 8-bit vector 00000100). q-to-2^q decoders are used to obtain the new representation of the input messages. To compute the two minimum values, a bitwise OR operation is performed between these representations (e.g. a bitwise OR between 1, 2, 4 and 5 will result in the vector 00110110). The least significant one in the vector obtained after the OR operation represents the value of the first minimum, while the second least significant one represents the value of the second minimum. To extract these values, a modified LZC is used; the conventional LZC computes only the position of the least significant one. The proposed CNU cannot compute correctly the second minimum when its value is equal to the first minimum (e.g. for the following inputs 1, 2, 1, 5, 4 the computed first minimum is 1, while the result for the second minimum is 2 instead of 1). Hence, the CNU operation is imprecise.

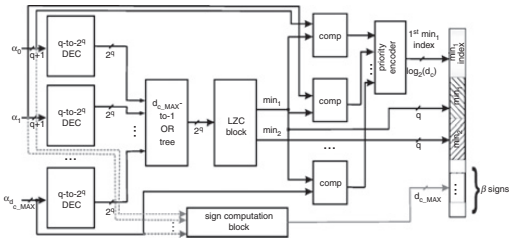

Fig. 1 *Proposed CNU architecture*

The index of the first minimum is computed in two stages: first compare the first minimum with the inputs, and then use a priority encoder during the second stage.

Implementation results: Tables 1 and 2 present the synthesis estimations (cost and performance) for the proposed check node unit architecture, as well as for the comparator-based standard architecture used in [4, 5] LDPC decoders. The results have been obtained using the Xilinx ISE 14.7 synthesis tool for Xilinx Virtex-7 FPGA (speed grade 2) for a 4-bit quantisation of the check node message (1-bit sign and 3-bits of magnitude). The proposed architecture uses 3-to-8 decoders. We analyse the effect of varying the number of inputs on cost and frequency. These check node degrees d_c correspond to different codes and code rates. Both the proposed architecture and the standard check node unit have a single pipeline stage (latched inputs and outputs).

Table 1: Cost estimates for the baseline [4, 5] and the proposed CNU for Virtex-7 FPGA (in look-up tables – flip-flop pairs)

d_c	10	15	20	32	40	64	72
Baseline	141	195	274	430	591	873	976
Proposed	113	155	204	332	383	612	685

Table 2: Frequency estimates for the baseline [4, 5] and the proposed CNU for Virtex-7 FPGA (in MHz)

d_c	10	15	20	32	40	64	72
Baseline	272	246	209	197	182	166	153
Proposed	254	216	197	185	**197**	**176**	**175**

Table 1 suggests that the proposed implementation of the CNU presents a 9 to 30% better cost with respect to the baseline implementation. Significant cost improvements can be obtained if we further increase the number of CNU inputs. Regarding the performance estimates, for smaller d_c, the proposed implementation presents a lower frequency compared with the baseline. However, for a higher number of inputs (such as 40), the proposed CNU equals and surpasses the performance of the baseline CNU.

Error correction performance: In this Section, we evaluate the decoding performance degradation because of the introduced impreciseness in our proposed CNU. The evaluation is performed for MS and self-corrected MS (SCMS) [10] based LDPC decoders. Regular LDPC codes with $d_v = 3$ and $d_c = 6$, 9, 12, 18, 30, corresponding to coding rates $R = 1/2$, 2/3, 3/4, 5/6, 9/10 have been considered for analysis. The bit error rate (BER) performance over the additive white Gaussian noise channel with quadrature phase-shift keying modulation is shown in Fig. 2. For both decoders, exchanged messages are quantised on 4 bits (hence CNU operations are performed on 3-bit values). BER curves have been derived for each coding rate. MS is denoted with black and SCMS with red. Solid curves correspond to the exact CNU and dashed curves to the imprecise CNU proposed in

An Imprecise Stopping Criterion Based on In-Between Layers Partial Syndromes

D. Declercq, V. Savin, O. Boncalo, and F. Ghaffari

***Abstract*— In this letter, we address the issue of early stopping criterion for layered LDPC decoders, aiming at more safeness with low hardware cost and minimum latency. We introduce a new on-the-fly measure in the decoder, called in-between layers partial syndrome, and define a family of stopping criteria, with different tradeoffs among complexity, latency, and performance. Numerical results show that our stopping criteria surpass existing solutions, and can be as safe as the full-syndrome detection, down to frame error rates (FERs) as low as FER $= 10^{-8}$.**

***Index Terms*— Layered LDPC decoder, early termination, in-between layers partial syndrome.**

I. Introduction

LOW-DENSITY Parity-Check (LDPC) codes have attracted much attention in the past several years [1] due to their excellent performance under iterative decoding. Iterative decoding of LDPC codes suffer from a high latency issue, since the decoding iterations need to be performed sequentially. In order to solve this latency issue, it is important to limit the number of decoding iterations needed to correct the channel errors. Since the average number of iterations is typically much smaller than the maximum number of iterations, stopping the decoder as early as possible may lead to significant improvement in terms of both average throughput and energy consumption [2], [3]. The classical approach to declare convergence to a codeword is to compute the syndrome of the LDPC code and to verify that it is equal to zero. However, this straightforward implementation of the stopping criterion impacts an increase of the hardware (HW) cost, or a degradation of the decoding latency.

For the class of structured quasi-cyclic LDPC codes (QC-LDPC) [4], specific stopping criteria have been proposed, benefiting from the organization of the decoding flow of the layered architecture [5]. Of special interest are the low cost early stopping criteria which can be easily integrated in the layered decoding flow, *i.e.* without halting the decoding process, referred to hereafter as *on-the-fly* stopping criteria.

Multiple strategies have been employed for performing early termination for QC-LDPC codes, which can be classified in two main types: *(i)* parity-check (PC) computation based [6], [7], and *(ii)* a posteriori log-likelihood ratio (AP-LLR) based [2], [3], [8], [9]. The lowest complexity approaches usually come at the price of some performance degradation in error correction, in which case the stopping criterion is said *unsafe* or *imprecise* [7]. Unsafe criteria lead to residual errors on the hard-decision bits, which are qualified as *undetected errors*. The safe solutions usually suffer from the drawback of a slower convergence rate, leading to a loss in average throughput and energy consumption.

In this letter, we propose a stopping criterion based on a new measure, called in-between layers partial syndrome (IBL-PS), which consists of checking the PCs of two consecutive layers of QC-LDPC codes. This measure has not been discussed previously in the literature, and is especially interesting for layered decoding, since it can be computed on-the-fly. We propose a family of imprecise stopping criteria, based on the computation and use of one to several IBL-PSs, with different tradeoffs between latency and performance.

The rest of the letter is organized as follows. Section II introduces the concept of IBL-PS, and describes briefly its advantages. In section III, we give a characterization of the decoding situations for which the IBL-PS stopping criterion is unsafe. Section IV shows the performance results of our approach compared with other stopping criteria, and the conclusion is drawn in section V.

II. Concept of In-Between Layers Partial Syndrome

A. Definitions

Quasi-Cyclic LDPC codes are build from blocks of circulant matrices [4]. A QC-LDPC code is defined by a base matrix B, with integer entries $b_{i,j} \geq 0$, $\forall i = 1 \ldots M_b$, $\forall j = 1 \ldots N_b$. The matrix B is also referred to as *protograph* [11]. The parity-check matrix H of the LDPC code is obtained by expanding the base matrix by a factor L. Each nonzero entry of B is replaced by $b_{i,j}$ circulant matrices of size $L \times L$. In this letter, we discuss only the case of protograph LDPC of type-I, *i.e.* when $b_{i,j} \in \{0, 1\}$. The parity-check matrix of a QC-LDPC code, of size $(M, N) = (M_b\, L, N_b\, L)$, is organized in *layers*. A layer is defined by the set of L consecutive parity-checks, corresponding to one row of circulants.

LDPC codes are decoded by message-passing (MP) decoders that exchange messages between parity-checks and codeword bits. Accordingly, an LDPC decoder comprises two types of processing units, namely check node units (CNUs) and variable node units (VNUs). The specific structure of QC-LDPC codes makes them suitable for HW implementations, when they are decoded using layered scheduling [5]. In layered decoders, parity-checks within one layer

Manuscript received May 19, 2017; accepted June 16, 2017. Date of publication June 22, 2017; date of current version January 8, 2018. This work has been supported by the Franco-Romanian (ANR-UEFISCDI) Joint Research Programme Blanc-2013, project DIAMOND. The associate editor coordinating the review of this letter and approving it for publication was Z. Ma. *(Corresponding author: D. Declercq.)*

D. Declercq and F. Ghaffari are with ETIS, ENSEA/University of Cergy-Pontoise/CNRS, 95014 Cergy-Pontoise, France (e-mail: declercq@ensea.fr).

O. Boncalo is with the Computer Engineering Department, University Politehnica Timisoara, 300223 Timisoara, Romania (e-mail: boncalo@cs.upt.ro).

V. Savin is with the CEA-LETI, MINATEC Campus, 38054 Grenoble, France, (e-mail: valentin.savin@cea.fr).

Digital Object Identifier 10.1109/LCOMM.2017.2718523

Analog Integr Circ Sig Process (2016) 87:169–180
DOI 10.1007/s10470-015-0639-3

Memory trade-offs in layered self-corrected min-sum LDPC decoders

Oana Boncalo[1] · Alexandru Amaricai[1] · Petru Florin Mihancea[1] · Valentin Savin[2]

Received: 1 May 2015 / Accepted: 29 September 2015 / Published online: 4 November 2015

Abstract This paper proposes two variants which aim at reducing the memory requirements of the self-corrected min-sum (SCMS) with respect to min-sum (MS). The first improvement—SCMS-V1—eliminates the need for check node messages' signs storage. The second improvement—SCMS-V2—is based on a novel imprecise self-correction rule, which allows the reduction of the erasure bits. We analyze the decoding performance and the hardware cost for the Min-Sum and three variants of the SCMS decoder for two types of regular and irregular LDPC codes, four coding rates, and two message quantizations. The decoding performance analysis indicates that the SCMS-V2 introduces almost no degradation in the error correction capability with respect to the conventional SCMS, while it has an almost 0.5 dB better performance compared to MS. Regarding the implementation analysis, we use an LDPC customizer tool developed by us in order to generate FPGA based LDPC layered architectures with serial AP-LLR processing and multi-codeword decoding support. The synthesis results for Virtex-7 FPGA devices indicate that the SCMS-V2 has similar cost in both BRAM blocks and slices with the Min-Sum decoder and up to 33 % less than SCMS-V1 decoder and up to 40 % less than SCMS decoder.

Keywords LDPC decoding · Min-sum · Self-corrected min-sum · FPGA

✉ Oana Boncalo
oana.boncalo@cs.upt.ro

[1] University Politehnica Timisoara, Timisoara, Romania

[2] CEA-LETI, MINATEC Campus, Grenoble, France

1 Introduction

Due to their increased error correction capability when using message passing iterative decoding algorithms [1, 2], LDPC codes are widely used in modern communication standards, such as IEEE 802.16 WiMAX [3], or DVB S2 standards [4]. The belief-propagation (BP) decoding is known to be optimal for LDPC codes defined by cycle-free bipartite graphs, in the sense that it outputs the maximum a posteriori (MAP) estimates of the coded bits. Even if in practical cases BP deviates from the MAP—because practical LDPC codes are defined by graphs with cycles—it is still referred to as the "optimal iterative decoder", due to its excellent decoding performance. However, from the hardware implementation perspective, BP decoding is disadvantaged by (i) its computational complexity, (ii) the severe degradation of its error-correction performance for low quantization levels, and (iii) the fact that it requires an accurate estimation of the signal-to-noise ratio (SNR), which may be imprecisely estimated in practical situations. The min-sum (MS) decoding algorithm is aimed at reducing the computational complexity of the BP, by using max-log approximations of the check node messages. The only computations required by the MS decoding are additions and comparisons, which makes it suitable for hardware implementations [1]. The performance of the MS decoding is also known to be independent of the knowledge of the signal-to-noise ratio, for most of the usual channel models. However, it presents sub-optimal decoding performance, due to the overestimation of the check node messages. Different MS variants, such as normalized MS (NMS) and offset MS (OMS) [5, 6], try to reduce the overestimation characteristic of the MS, by reducing the amplitude of the check node messages.