High Throughput and Resource Efficient Pipelined Decoder Designs for Projective Geometry LDPC Codes

Ved Mitra¹*, Mahesh C. Govil², Girdhari Singh¹, Sanjeev Agrawal³

¹ Department of Computer Science and Engineering, Malaviya National Institute of Technology, Jawaharlal Nehru Marg, Jaipur, Rajasthan – 302017, India
² National Institute of Technology Sikkim, Ravangla, South Sikkim, Sikkim – 737139, India
³ Department of Electronics and Communication Engineering, Malaviya National Institute of Technology, Jawaharlal Nehru Marg, Jaipur, Rajasthan – 302017, India
* Corresponding author, e-mail: vedrig111@gmail.com

Received: 08 August 2019, Accepted: 29 September 2019, Published online: 07 December 2019

Abstract
Projective geometry (PG) based low-density parity-check (LDPC) decoder design using iterative sum-product decoding algorithm (SPA) is a big challenge due to higher interconnection and computational complexity, and larger memory requirement caused by relatively higher node degrees. PG-LDPC codes using SPA exhibits the best error performance and faster convergence. This paper presents an efficient novel decoding method, modified SPA (MSPA) that not only shortens the critical-path delay but also improves the hardware utilization and throughput of the decoder while maintaining the error performance of SPA. Three fully-parallel LDPC decoder designs based on PG structure, PG(2,GF(2¹⁰)) of LDPC codes are introduced. These designs differ in their bit-node (BN) and check-node (CN) architectures. Fixed-point, 9-bit quantization scheme is used to achieve better error performance. Another significant contribution of this work is the pipelining of the proposed decoder architectures to further enhance the overall throughput. These parallel and pipelined designs are implemented for 73-bit (rate 0.616) and 1057-bit (rate 0.769) regular-structured PG-LDPC codes, on Xilinx Virtex-6 LX760 FPGA and on 0.18 μm CMOS technology for ASIC. Synthesis and simulation results have shown the better performance, throughput and effectiveness of the proposed designs.

Keywords
low-density parity-check (LDPC) codes, sum-product decoding algorithm (SPA), projective geometry (PG), Galois fields (GF), FPGA, ASIC

1 Introduction
Error-control coding has a great significance in the present digital communication systems for error detection and correction over received data streams. Low-density parity-check (LDPC) codes represent a special class of linear block codes, primarily introduced by Gallager [1] in 1962; rediscovered by MacKay and Neal [2] in 1995, are the focus of intense research since late 1990s. They are strong contenders to turbo codes [3] for error control in digital data storage and communication systems. However, turbo codes have limited acceptance as compared to LDPC codes due to their low coding gain, high decoding latency and requirement of much complex computations. LDPC codes play a prominent role in modern communication systems that demand not only the superior error performance close to the Shannon limit over conventional channels [4, 5] but also very-high throughput services. These two features promote LDPC codes as one of the most propitious candidates for communication standards – Digital Video Broadcasting (DVB-S2) [6, 7]; IEEE 802.11n (WLAN) [8], IEEE 802.16e (WiMAX) [9] and Magnetic storage systems [10, 11]. However, LDPC codes with higher code rates are mainly useful for low noise high throughput standards like – Orthogonal frequency-division multiplexing (OFDM) [12], IEEE 10GBase-T [13] and Fiber-optic communication systems [14].

Among various classes of LDPC codes, projective geometry (PG) LDPC codes with sum-product decoding algorithm (SPA) have large minimum distance that exhibits the best error performance and faster convergence. However, high-performance high-throughput pipelined PG-LDPC decoder design using SPA is still a big challenge due to higher interconnection and computational
complexity, and larger memory requirement caused by relatively higher node degrees. For example, the degree of a check-node (CN) in an OFDM system with code rate 0.875 proposed by Yang et al. [12] is 24. In contrast, by using simplified decoding methods; for example, bit-flipping [15] and turbo-decoding message passing (TDMP) [16, 17] or approximations to original SPA; for example, min-sum (MS) algorithm and its variants [18–21] complexities can be reduced and throughput can be enhanced but these results in far worse error performance and much slower decoding convergence than the SPA. Bit-flipping is a hard-decision based algorithm that exhibits significant error performance loss due to the lack in quantization precision. MS algorithm and its variants, replaces complex computations of check-nodes (CNs) in SPA with simple addition and comparison operations, but it causes up to 1 dB performance loss compared to SPA for higher code-word lengths, code rates and node degrees [22].

Many of the existing hardware implementations for LDPC codes have used memory-shared, serial or partial parallel architectures [21, 23, 24]; where chip area and hardware cost is of more concern. For example, a memory-shared architecture introduced by Chandrasetty and Aziz [23] – for 2304-bit, 1/2-rate LDPC code were used 232 memory blocks of fixed size memory primitives. These port-limited architectures suffer from limited memory bandwidth, poor memory utilization and interconnection complexity which may be even worse in the presence of multiplexers. For collision free memory blocks access, separate address generation units are required that will increase chip-area and complexity. Again, these implementations are difficult to pipeline, and also not portable and synthesizable on ASIC tools. Further, achievable throughput is of the order of only hundreds of megabits-per-second (Mbps). So, to address these problems, in this paper we propose high-performance, high-throughput, resource-efficient fully-parallel and pipelined PG-LDPC decoder designs those are capable of providing throughput in the range of gigabits-per-second (Gbps) at moderate block lengths. An efficient novel decoding method, called modified SPA (MSPA), a variation to the original SPA algorithm is introduced for this purpose to decode PG-LDPC codes that not only shortens the critical-path delay, but also improves the hardware utilization and throughput of the decoder while maintaining the error performance of SPA. Three different fully-parallel LDPC decoder designs are implemented based on PG structure, PG(2,GF(2^w)) [25] of LDPC codes with fixed-point, 9-bit quantization scheme. These decoder designs differ in their bit-node (BN) and check-node (CN) architectures, and are further pipelined in order to enhance the overall throughput. The proposed parallel and pipelined designs are implemented for 73-bit (rate 0.616) and 1057-bit (rate 0.769) regular-structured PG-LDPC codes, on Xilinx Virtex-6 LX760 FPGA [26] and on 0.18μm CMOS technology for ASIC that provides a throughput of 6.5 Gbps comparable with existing IEEE 802.11 ac/ad/ax WLAN [8] and IEEE 10GBase-T [13] standards.

The organization of the paper is as follows – Section 2 introduces the structured property of LDPC codes, the SPA algorithm, PG structure of LDPC codes and the message quantization. Section 3 presents the hardware architecture of various functional units used in the decoder designs. Section 4 introduces the efficient novel MSPA decoding method and its hardware implementation. It also describes the architectures of fully-parallel and pipelined SPA and MSPA decoders. Section 5 presents the hardware implementation results targeted to both FPGA and ASIC. Section 6 concludes the paper.

### 2 Structured PG based LDPC codes

LDPC code is fully described by an $M \times N$ sparse parity-check matrix $H$, where $M$ rows represent the parity-check constraints and $N$ columns each corresponds to a specific codeword bit. A codeword of length $N$ bits has $K$ message bits and $M$ check bits. The code rate $R$ is

$$ R = K/N = 1 - M/N. \tag{1} $$

In a regular LDPC code, matrix $H$ contains exactly $w_c$ 1’s in every column (column-weight) and exactly $w_r$ 1’s in every row (row-weight); otherwise, it is said to be an irregular code. For example, Fig. 1 (a) shows LDPC matrix $H$, for a $(2, 3)$ regular code with $w_c = 2$ and $w_r = 3$ of length 6-bit.

#### 2.1 Graphical representation of LDPC codes

A bipartite Tanner graph can be used to graphically represent the LDPC codes [27]. The graph consists of two types of nodes – bit-nodes (BNs) and check-nodes (CNs), and two nodes of different type can only connect to each other through an edge. The edges of a Tanner graph can be represented by non-zero entries “1” in matrix $H$. There are $N$ BNs, one for every codeword bit $c_i$ and $M$ CNs, one for every set-of parity-check constraints. Tanner graph corresponding to matrix $H$ in Fig. 1 (a) is illustrated in Fig. 1 (b), where $N$ BNs are represented by circles and $M$ CNs by squares.
2.2 Sum-product decoding algorithm (SPA)

The aim of SPA is to evaluate the a posteriori probability (APP) $P_c$ of every codeword bit $c_i \in C = [c_1, c_2, c_3 \ldots c_n]$; given the received word $Y = [y_1, y_2, y_3 \ldots y_n]$.

$$P_c = P_c(c_i = 1 | y)$$

In log-likelihood-ratio (LLR) form, Eq. (2) can be represented as

$$L(c_i) = \log \left( \frac{P_c(c_i = 0 | y)}{P_c(c_i = 1 | y)} \right).$$

The SPA computes an approximation of the APP value for every codeword bit iteratively, based on the code’s Tanner graph as follows:

- **BN Update Stage** – In the first half of the iteration, every BN processes its input messages (intrinsic channel information $u^*_n$) plus extrinsic messages received from all its neighbor CNs except $CN m$ and computes the resulting bit-to-check message for the desired neighbor $CN m$. BN updates can be represented in LLR form as

  $$u^*_n = u^*_n + \sum_{m' \in \mu(m) \cap m} v_{m',n},$$

  where $u^*_n$ as the message passed from $BN n$ to $CN m$ and $m' \in \mu(m) \cap m$ as the set of all CNs connected to $BN n$ except $CN m$ itself. It has to be noted that initial message passed by $BN n$ to respective $CN m$, in the first iteration is the intrinsic probability only.

- **CN Update Stage** – In the other half iteration, based on the messages received from the BNs, each CN processes its input messages as (here the sign and the magnitude part has been separated to aid the understanding of its hardware realization) follows:

  $$v_{m,n} = \left( \prod_{x \in \nu(m) \cap n} \text{sgn}(u^*_x) * (-1)^{x(n)} \right) \phi^{-1} \left( \sum_{x \in \nu(m) \cap n} \phi(\{u^*_x\}) \right),$$

  where $\text{sgn}(x) = |x|/x$ and $\phi(x) = \phi^{-1}(x) = -\log \tanh(|x|/2)$, $v_{m,n}$ as the message passed from $CN m$ to $BN n$ and $m' \in \sigma(m) - n$, as the set of all BNs connected to $CN m$ except $BN n$ itself. It has to be noted that in the first iteration, message passed by every $CN m$ to respective $BN n$, should be zero.

- **Parity Check** – For the correctness of the codeword, an estimate is performed by guessing a value for every bit at the BN as follows:

  $$u^*_n = u^*_n + \sum_{m' \in \mu(m) \cap n} v_{m',n}.$$ (6)

  This is the accumulated sum (total-sum) obtained from the accumulation-scan during the bit-update process. A hard-estimate about the bit value is made using the following conditions:

  $$c_n = 1, \text{ if } u^*_n \leq 0 \text{ or } c_n = 0, \text{ otherwise.}$$ (7)

The algorithm terminates, if $H \cdot C^T = 0$ or if the number of permissible iterations are completed; otherwise, it proceeds to the next iteration starting from Eq. (4).

2.3 Message quantization of LDPC code over $PG(2,2^s)$

The Tanner graph for our work is same as the point-line incidence graph of a dimension-$m$ projective plane over $PG(m,GF(2^s))$ [28]; where, $m = 2$ and $s = 3$. Here, BNs/CNs represents the points/lines of the geometry respectively, and correspondingly the columns/rows of the parity-check matrix $H$. A codeword of LDPC code over $GF(2^s)$ contain symbols from the Galois field $GF(p = 2) = \{0,1\}$; where constraints are defined over modulo-$2$ arithmetic [29] and $p$ denotes a prime number.

Message quantization choices affect not only on the complexity and performance of the design but also on the throughput. However, it depends on the resources available for storage and computation on the FPGA [26]. We are considering 9-bit(9-5) quantization scheme in fixed point sign-magnitude (SM) format for better performance, where the most-significant bit (MSB) represents the sign and the rest of 8-bits, the magnitude. In the magnitude part, the most significant 3-bits represent the integer and the remaining 5-bits, the fractional part. In order to accommodate additional bits for sign-extensions and overflows due to accumulation, the internal datapath is made 13-bit wide for BNs and 12-bit wide for CNs.
3 Functional units of proposed LDPC decoder

The LDPC decoder has three fundamental components—the Processing Elements (BNs/CNs) comprising the datapath, the Memory Modules to store bit/check-updates during iterations, and the Interconnection Network for routing of updates between different kind of nodes.

For the proposed designs, we are considering the 73-bit (rate 0.616) and 1057-bit (rate 0.769) regular-structured LDPC codes based on PG(2,GF(2^15)) [25]. The main computational blocks in BNs and CNs are multi-input multi-bit adders, subtractors and multipliers/LUTs. Bit/check-updates are computed using total-sum-first method [30]. The total-sum-first calculations (for BN/CN updates) are implemented using unfolded parallel architecture [31] for accumulation-scan rather than folded one; as it offers higher degree of parallelism and thus, suited for high throughput applications. The two types of memories involved in the decoder designs are – bit-memories (BMs) and check-memories (CMs). BMs and CMs will be discussed in detail in Subsection 3.3.

3.1 Bit-node architecture

Fig. 2 shows the architecture of a fully-parallel BN. The BNs compute bit-to-check messages (bit-updates) according to Eq. (4). BNs read check-to-bit messages from the CMs and intrinsic data from Intrinsic-memory, perform SM to 2’s complement (SM–2’s) transformation on the received inputs; perform accumulation-scan using multibit adder tree along with individual input messages are stored separately on corresponding L-Regs; perform output-scan (residue calculation) by subtracting the individual inputs from the accumulated sum using multibit full subtractors and finally the outputs (bit-updates) are saturated to 9-bit SM format and written back into BMs.

3.2 Check-node architecture

The CNs compute check-to-bit messages in the same way as their bit counterparts, but with two significant differences – The CN computations are done in the logarithmic and hyperbolic tangent domain as stated in Eq. (5). Further, the magnitude and sign part of the updates are computed through different data-subpaths. In the magnitude subpath, the 9-bit SM inputs received from BMs undergo through ϕ(x) transformation, before the magnitude’s residues are calculated using the same total-sum-first approach as used in bit-updates. Finally, the residues are reconvered back by applying the inverse function ϕ⁻¹(x) (ϕ(x) is self-inverse) on them. A piece-wise linear approximation method can be used as suggested by Masera et al. [32] to implement ϕ(x). The one of the direct way to implement ϕ(x) is by using LUTs [16, 33]. As the function ϕ(x) is highly non-linear, a large performance loss will be induced by its quantization. Therefore, to achieve proper decoding performance direct implementation using LUTs would require much large amount of memory, specifically for codes having higher node degrees and quantization.
bits. Again, for fully-parallel design, each CN will require its own LUT in order to speed up the operation and to avoid memory access conflicts. This in turn makes the design comparatively expensive and hence, we have not used this method in our proposed designs.

Some other implementations as stated in [24, 34]; used DSP slices available on FPGA to compute \( \phi(x) \). The main disadvantages in using DSP slices are – First, the resources are limited on FPGA. For fully-parallel and pipelined LDPC decoder designs, we need at least 657 (73*9) DSP slices for 73-bit code. If the number of nodes is increased, we require more DSP slices which makes it difficult to accommodate on latest FPGAs [26]. Secondly, this increases the hardware complexity, area and cost for the design, comparatively. Finally, DSP slices are FPGA specific macros, these slices are not portable and synthesizable on ASIC tools for ASIC design. So, to overcome these problems and for the implementation of proposed decoder designs on both FPGA and ASIC, we are used our own VHDL constructs array-multipliers and adders as suggested in [31] to compute \( \phi(x) \) function. For simplicity and clarity, codeword testing part is not shown here. Again, two different approaches are discussed here for the fully-parallel CN design:

- The first design (CN_A) consists of 2\( w \) MAC units as shown in Fig. 3. These are termed as – PH-MACs and IN-MACs. The \( w \) PH-MACs are used to compute the function \( \phi(x) \), whereas other \( w \) IN-MACs are used to compute the function \( \phi^{-1}(x) \). The outputs from PH-MACs are scaled down into 12-bits for magnitude calculation in total-sum-first block in the similar way as in BN computation, using unfolded parallel architecture. After magnitude’s \( \phi^{-1}(x) \) transformation – outputs are saturated, combined with their sign counterpart and finally stored in CMs. The sign logic is implemented by using an XOR-gate tree that works concurrently with magnitude processing.

- The second design (CN_B) is similar to first – except MAC and saturation units are reused through a feedback path vide Fig. 4. However, one multiplexer (MUX) at the input to every MAC unit in order to select between \( \phi(x) \) and \( \phi^{-1}(x) \) operations; and one de-multiplexer (De-MUX) at the output to every saturation unit are introduced in the design.

3.3 Memory organization

Memory is used for preserving the bit-to-check and check-to-bit updates during every iteration in the Sum-Product decoding. As stated earlier, there are two types of memories used in the decoder designs – BMs and CMs. For writing back results, every BN (CN) is associated with a memory of its own type. The memories of other kind that each node reads data from, are determined by the projective geometry space PG(2,2^s) [25]. Data stored in memories are in 9-bit sign-magnitude form.

![Fig. 3 Fully-parallel CN architecture without Feedback (CN_A)](image-url)
Most of the existing memory-shared partial-parallel decoder designs reported in literature have used true dual-port Block-RAM (BRAM) blocks with fixed sized memory primitives. For example, Gajare [24] used 146 dual-port BRAM blocks, one for each type of node with $2k \times 9$, fixed memory primitives. Another architecture introduced by Chandrasetti and Aziz [23] – for 2304-bit, 1/2-rate, (3,6)-regular LDPC code were used 232 such memory blocks. However, Chen et al. [21] pointed out that for an 8-bit message quantization, in this way upto 78% of available memory bandwidth will not be used. The main disadvantages in these decoder designs are:

- For the proposed 73-bit (rate 0.616) regular LDPC decoder design, nine-coefficients (inputs) have to be accessed in parallel per cycle per node. This figure is much higher for codes having relatively higher node degrees and block lengths. However, due to the port limiting architecture, BRAM blocks are able to access only two-coefficients concurrently per cycle. This in turn increase the number of cycles required and hence lowers the overall throughput significantly.
- As stated above, each BRAM block was configured in a fixed size memory primitive (like $2k \times 9$) as per the need of message quantization. Hence, a significant amount of every BRAM block locations were left unused which results in poor utilization of available memory bandwidth. This also increases the complexity and chip area when implementing physical-layout floorplan.
- Large memory bandwidth also results in significant power dissipation.
- For collision free dual-port BRAM blocks access, separate address generation units are required that will further increase the chip-area and complexity.
- BRAM blocks are FPGA specific macros, these are not portable and synthesizable on ASIC tools for ASIC design.

Hence, in order to eliminate these problems and for the implementation of proposed decoder designs on both FPGA and ASIC, we are using distributed memory elements as suggested in [31] to store bit/check updates instead of BRAM blocks. These are named as B-Regs and C-Regs respectively.

### 3.4 Interconnect architecture

Interconnects consume most of the floor space in the layout of a design [35]. Therefore, efficient implementation of interconnects is required for the decoder design to reduce the complexity and hence routing congestion.
As PG is a point-to-point interconnect with high node degrees, Bus Architecture is not suitable for our implementation. This is because in a particular cycle, large number of nodes will try to access the bus simultaneously, leading to widespread congestion in interconnect. Further, some of the designs as stated in [9, 17, 36-38] used routing (permutation) networks for the implementation of interconnects. Routing networks are suitable for designs having relatively small node degrees. For PG-LDPC codes with relatively higher degrees, hardware and routing complexity of such networks will be much high. For such designs, dedicated wiring reduces hardware complexity comparatively and increases the flexibility for both regular and irregular parity check matrices. Hence, we have adopted direct fixed network of wires between the nodes and memories for our implementations.

For the interconnection between C-Regs and the BNs, and the other one between B-Regs and the CNs global wiring is used, as per geometry of Tanner graph. Dedicated wiring is used to implement these connections. The wires between the nodes (BN/CN) and the memory-units (B-Regs/C-Regs) of the same type are expected to be local, due to the proximity in their placement. However, broadcasting technique [35] can be used to reduce the number of interconnect wires and connection lengths between BNs and CNs, further by more than 40%.

4 Decoder architecture

In Section 4, first we discuss about the architecture of fully-parallel LDPC decoder and then introduce the MSPA decoding, a novel variation of the original SPA algorithm, that not only shorten the processing latency but also improve the hardware utilization and throughput of the decoder. Finally, we discuss pipelined decoder architecture to further enhance the overall throughput.

4.1 Parallel decoder architecture

Two different versions of fully-parallel LDPC decoder designs are discussed here. These designs differ in their CN architectures, and are implemented using the original iterative SPA algorithm in LLR form for 73-bit (rate 0.616) and 1057-bit (rate 0.769) PG-LDPC codes, separately. These decoders are termed as decoder-1 (comprising of BN_A and CN_A architectures) and decoder-2 (comprising of BN_B and CN_B architectures), respectively.

A parallel decoder can be implemented by using p copies of BNs(CNs) and the corresponding interconnects in parallel as a group; where p represents parallel factor. This permits to update all the p BNs(CNs) concurrently with in a group where each BN(CN) group requires one clock cycle to complete its computation. There are a total of \(2 \lceil N/p \rceil\) such groups. Hence, total \(2 \lceil N/p \rceil\) cycles are needed to complete all BNs and CNs computations in a single iteration. It is obvious that the throughput, hardware complexity and power consumption increases as p increases where as time required for completing a single iteration decreases.

For the 73-bit (rate 0.616) PG-LDPC decoder designs, we have used \(p = 73\). Hence, these designs are fully-parallel, having single group for BNs(CNs) and total \(2 \lceil N/p \rceil = 2\) cycles are needed to complete all BNs and CNs updates in an iteration. For the 1057-bit (rate 0.769) PG-LDPC decoder designs using SPA, it is impractical to use \(p = 1057\) because of the severity of interconnects complexity, routing congestion and power consumption. Hence, in order to maintain the design feasibility and high throughput, in the proposed designs we have used two different parallel factors \(-p = 73\) and \(p = 151\). Besides of using different parallel factors, these designs are also made flexible in terms of different node degrees, quantization and codeword lengths.

4.2 Modified SPA (MSPA) decoder architecture

The two SPA based parallel decoder designs as stated above, have unbalanced computation complexities and unbalanced datapaths between BNs and CNs. This in turn effects on the critical-path delay and number of cycles required per iteration. These effects can be minimized by using the MSPA decoding method that modifies the BN/CN update stages of SPA to achieve the hardware balancing between BNs and CNs. The MSPA decoding method is described in the following steps:

- **BN Update Stage** – For the MSPA decoding method, Eq. (4) of BN computations can be modified as follows:
  \[
  u_{n,m} = u_n^+ + \sum_{m'\in\text{nbr}(m)} \phi^{-1}(v_{n,m'}) \tag{8}
  \]
  where all the incoming messages to BN \(n\) from all its neighbor CNs are first \(\phi^{-1}(\cdot)\) transformed before accumulation-scan.

- **CN Update Stage** – Equation (5) of CN computations can be modified as follows:
  \[
  v_{n,m} = \prod_{n' \in \text{nbr}(n)} \text{sgn}(u_{n',m}) \ast (-1)^{b_{n'} - 1} \ast \left( \sum_{m' \in \text{nbr}(m)} \phi(|v_{n',m'}|) \right) \tag{9}
  \]
Let $v_{n,n} = P \times S$; where $P$ is the magnitude part and $S$ is the sign part. The magnitude part $P$ can be re-written as:

$$P = \left( \sum_{n=0}^{\text{cnot count}} \phi \left( \left| V_n \right| \right) \right) - \phi \left( \left| V_{n,0} \right| \right) \right].$$

In the MSPA decoder design, termed it as decoder-3, we are introducing hardware balancing between BNs and CNs for slack minimization using Eqs. (8) and (9) respectively. For its hardware implementation, consider the Fig. 3 of a fully-parallel CN without feedback ($CN_A$). If all the output side IN-MACs ($\phi^{-1}(x)$) and saturate units of this CN are removed and placed to input side of corresponding BN, then we obtain optimized BN ($BN_C$) and optimized CN ($CN_C$) as shown in Figs. 5 and 6, respectively. This indeed provides us hardware balancing among BNs and CNs without affecting the performance of original SPA decoder. The MSPA decoding has the following advantages:

- As per the largest latency in a pipeline, critical-path can be determined that will limit the overall throughput. However, in decoder-3 design, after critical-path balancing, the optimized BNs and CNs will have similar path delay. This in turn reduces the slack time, improves the clock frequency and hence the throughput.

- In $CN_A$ design, output side IN-MACs ($\phi^{-1}(x)$) and saturate units remain idle till the last time-slot of CN computation; so moving these units into a BN will definitely provide us effective hardware utilization without affecting on the decoder performance.

- For high-rate codes, where $w >> w$, MSPA decoding will be more beneficial in terms of hardware reduction (e.g., number of IN-MACs, level of adder tree).

A complete throughput and timing statistics for these decoder designs are shown in implementation results.

### 4.3 Pipelined decoder architecture

In the Subsection 4.1, it is observed that during the first half iteration of BNs processing, CNs remain idle and vice versa, in the next half. This in turn effects on hardware utilization and increases (almost doubles) the overall time required to complete single iteration; assuming that BNs and CNs processing takes same amount of time to complete. Fig. 7 (a) shows the above non-pipelined sequential timing structure for decoder-1/decoder-3 implementations between BNs and CNs update stages.

As stated earlier, our designs are based on PG(2,GF(2^2)) [25]. The structured property of PG-LDPC codes and the use of distributed memory elements to store BN(CN) updates allow BN and CN groups to operate in pipelined manner. Two pipelined LDPC decoder designs are discussed in Subsection 4.3. These designs are obtained by overlapping the BN and CN update stages for decoder-1/decoder-3 implementations in order to enhance the overall throughput. Fig. 7 (b) shows, pipeline timing structure for decoder-1p based on decoder-1 architecture with unbalanced computation complexities between BNs and CNs. Here, dashed portion in BN update stages shows the slack period. Fig. 7 (c) shows, pipeline timing for decoder-3p based on decoder-3 architecture with balanced data paths between BNs and CNs.

---

**Fig. 5 Optimized BN architecture using MSPA decoding**
For the above two cases illustrated in Figs. 7 (b) and 7 (c), the first BN group completes its computation in the first clock cycle and in the subsequent \( N_p - 1 \) clock cycles of the same iteration, the remaining BN groups are processed, consecutively. The CN groups start their computations just after first clock cycle of BNs computation, in consecutive manner. In this way, the last CN group will finish its computation, one cycle after all the BNs computation ends. Hence, clock latency between BN/CN update stages is now reduced to one clock only and therefore, total \( \lceil N/p \rceil + 1 \) clock cycles are needed for one decoding iteration. Here, computations of BNs and CNs are overlapped for \( \lceil N/p \rceil - 1 \) cycles. In the next iteration BNs can start their computations just after CNs ends their computations in the current iteration. Hence, the throughput gain is \[ \text{Gain} = 2\frac{\lceil N/p \rceil}{\lceil N/p \rceil + 1} \approx 2. \]

A complete throughput and timing statistics are shown in implementation results.

### 5 Implementation results

The proposed fully-parallel and pipelined LDPC decoder designs have been implemented and targeted on the Xilinx Virtex-6 LX760 FPGA and on 0.18 \( \mu \)m CMOS technology for ASIC. These designs are based on PG(2,GF(2^s)) [25].

PG codes converge very fast under SPA decoding [25]. Faster convergence is one of the important factors to achieve higher throughput. It has been observed that the proposed designs can be able to decode errors...
on an average in less than eight iterations at practical SNRs (> 2). The entire simulation was carried out assuming AWGN channel with BPSK modulation scheme. As all the three designs (that belong to same codeword length) use the same quantization scheme and based on SPA algorithm; their performance measures in terms of bit-error rate (BER) versus signal-to-noise-ratio (SNR) were found to be more likely. Fig. 8 presents BER versus SNR performance for the two distinct codeword lengths. It is clear from Fig. 8 that the BER performance improves significantly with larger codeword lengths.

5.1 FPGA Synthesis results
The Synplify Pro and Xilinx synthesis tools have been used for Synthesis. For the 73-bit (rate 0.616) regular-structured PG-LDPC code; Table 1 shows the comparative analysis between the three parallel designs as per the synthesis report in terms of utilization of various resources, post placement route timing analysis and throughput. Similarly, for the 1057-bit (rate 0.769) regular-structured PG-LDPC code with parallel factor –73; Table 2 shows comparative analysis for proposed parallel and pipelined designs using the same parameters discussed as above. As decoder-1p and decoder-3p are the pipelined versions of decoder-1 and decoder-3, respectively; they have the same resource utilization as decoder-1 and decoder-3, respectively. Table 3 shows the comparative analysis for the 1057-bit PG-LDPC code with parallel factor –151.

5.2 ASIC Synthesis results
The functional units of the proposed decoders have been synthesized using high-speed standard cell library on 0.18 µm CMOS technology with operating conditions 1.8 V and 85 °C. Table 4 shows performance analysis for the three BNs and CNs designs in terms of area, power and cell used.

5.3 Comparison with other LDPC decoders
Table 5 shows the comparison between our 73-bit (rate 0.616)/1057-bit (rate 0.769) PG-based optimized, pipelined MSPA LDPC decoder and other state-of-the-art decoders. Our work shows better error performance, even with the 1057-bit regular-structured PG-LDPC code as compared to the other well-known structured LDPC code decoders listed in Table 5 having larger codeword lengths. The error performance is measured at a BER of $10^{-5}$. The proposed decoders are implemented for both FPGA and ASIC. The achievable throughput is 6.5 Gbps much higher than decoders listed in Table 5. A shift-LDPC decoder with 8192-bit codeword length, based on min-sum (MS) decoding was proposed in [20], which can achieve a comparable throughput of 5.1 Gbps. However, the MS algorithm is an approximation to SPA that has lower computational complexities (for CNs) but exhibits much worse error performance than the proposed MSPA. The performance loss of shift-LDPC decoder [20] is about 0.8 dB as compared with the proposed 1057-bit optimized MSPA decoder. Again, in [20], the codeword length is approximately 8 times larger than our proposed designs. Further, analysis shows that for the similar codeword lengths, proposed designs especially decoder-3p would provide much higher throughput that crosses the requirement of IEEE 10GBase-T standard.

A hybrid SBF (Soft bit-flipping) LDPC decoder was introduced in [15] that can acquire a throughput of 1.05 Gbps at 16 iterations. As it is based on bit-flipping decoding, the performance loss is more and convergence is very slow as compared with proposed MSPA decoding. We want to point out that, our work demonstrates the feasibility of LDPC decoder designs using large number of quantization bits (i.e., nine) and node degrees, for better

---

**Table 1** FPGA implementation results for 73-bit (Rate 0.616) code

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Decoder-1</th>
<th>Decoder-2</th>
<th>Decoder-3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flip-Flops/Latches</td>
<td>72851 (7.7 %)</td>
<td>70828 (7.5 %)</td>
<td>73294 (7.73 %)</td>
</tr>
<tr>
<td>6-input LUTs</td>
<td>66585 (14 %)</td>
<td>61283 (13 %)</td>
<td>66585 (14 %)</td>
</tr>
<tr>
<td>Parallel Factor</td>
<td>73</td>
<td>73</td>
<td>73</td>
</tr>
<tr>
<td>Clock Freq. (MHz)</td>
<td>151.55</td>
<td>145.8</td>
<td>170.3</td>
</tr>
<tr>
<td>No. of Cycles</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Throughput (Gbps)</td>
<td>1.84</td>
<td>1.77</td>
<td>2</td>
</tr>
</tbody>
</table>
error performance while other listed decoders have used small number of quantization bits (≤ 6) for limiting the chip size, hardware complexity and improving the throughput but this results in significant performance loss. Finally, in most of the implementations, the time required for I/O operations (storing the decoded messages/updates into memories/registers and fetching the inputs from memories/registers) has not been incorporated while evaluating the decoder throughput. Hence the practical throughput is much less than the evaluated one. In our designs, we have considered the I/O time while computing the throughput.
6 Conclusion
In this paper, we have presented an efficient novel decoding method, the MSPA, to decode PG-LDPC codes that not only shortens the critical path delay, optimizes the decoder functional units but also improves the throughput of the decoder. Parallel LDPC decoder designs have been implemented for 73-bit and 1057-bit regular-structured PG-LDPC codes using the traditional SPA and the proposed MSPA decoding, separately. From these designs, we analyzed that the MSPA decoding minimizes the effects of unbalanced computation complexities between BNs and CNs that exists in the SPA decoder by introducing the hardware balancing. The proposed designs are further pipelined by overlapping the BN and CN update stages in order to achieve near-optimal throughput and effective hardware utilization. These optimized, pipelined decoder designs on average saves 45% of the number of cycles per iteration.

With 9-bit quantization using MSPA decoding method and pipelining the maximum achievable throughput is 6.5 Gbps which is two times larger than when compared to traditional SPA decoding, and also comparable with existing IEEE 802.11 ac/ad/ax WLAN and IEEE 10GBase-T standards. Our implementations also outperform in terms of processing latency, and error performance at a BER of $10^{-5}$ as compared to the other state-of-the-art decoders. The proposed designs are also flexible in terms of quantization, node degree, parallel factor and codeword length.

References
https://doi.org/10.1109/TIT.1962.1057683
https://doi.org/10.1007/3-540-60693-9_13
http://doi.org/10.1109/ICC.1993.379441
http://doi.org/10.1049/el:19961141
https://doi.org/10.1109/4234.905935
https://doi.org/10.1109/MWSCAS.2013.6674853
https://doi.org/10.1109/TMAG.2006.878652
https://doi.org/10.1109/JPROC.2008.2004310
https://doi.org/10.1109/TCSI.2005.862074
https://doi.org/10.1109/MWSCAS.2013.6674853
https://doi.org/10.1109/TCSI.2010.2047743
https://doi.org/10.1109/TVLSSI.2003.817545


[26] Xilinx "Virtex-6 Family Overview Data Sheet - DS150 (v2.5)", [online] Available at: https://www.xilinx.com/support/documentation/data_sheets/ds150.pdf [Accessed: 01 July 2019]


