<table>
<thead>
<tr>
<th>Title</th>
<th>A general decoding framework for high-rate LDPC codes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Author(s)</td>
<td>Hosseini, S. M. Ehsan; Goh, Wang Ling; Chan, Kheong Sann</td>
</tr>
<tr>
<td>Date</td>
<td>2009</td>
</tr>
<tr>
<td>URL</td>
<td><a href="http://hdl.handle.net/10220/6376">http://hdl.handle.net/10220/6376</a></td>
</tr>
</tbody>
</table>

© 2009 IEEE. Personal use of this material is permitted. However, permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the IEEE. This material is presented to ensure timely dissemination of scholarly and technical work. Copyright and all rights therein are retained by authors or by other copyright holders. All persons copying this information are expected to adhere to the terms and constraints invoked by each author's copyright. In most cases, these works may not be reposted without the explicit permission of the copyright holder.
A General Decoding Framework for High-Rate LDPC Codes

S. M. Ehsan Hosseini†, Wang Ling Goh
School of Electrical and Electronic Engineering,
Nanyang Technological University,
Singapore
†hosseini@pmail.ntu.edu.sg

Kheong Sann Chan
Data Storage Institute,
A*STAR (Agency for Science, Technology and Research),
Singapore

Abstract—This paper presents a hardware solution to the design of general low-density parity-check (LDPC) decoders, which simplifies the delivery network required by the message passing algorithm. While many designs of LDPC decoders for specific classes of codes exist in the literature, the design of a general LDPC decoder capable of supporting random LDPC codes is still challenging. The method proposed in this paper tries to pack different check node (CN) and variable node (VN) messages in the Tanner graph representation of the LDPC code, and is therefore called message packing. This method takes advantage of the fact that for high-rate LDPC’s the CN’s degree is much larger than the VN’s, and two distinct methods for delivering the messages to the CNs and VNs are proposed. Using the proposed interconnection network (IN) results in lower complexity decoding of LDPC codes when compared to other designs.

Index Terms—low-density parity-check (LDPC), decoder, interconnection networks, VLSI architectures.

I. INTRODUCTION

Low-density parity-check (LDPC) codes [1] are a class of error-correction codes specified by a sparse parity-check matrix proposed by Gallager in the 1960’s. Due to the high complexity of decoding LDPC codes, they were forgotten for several decades until Mackay and Neal [2] reported their superior performance in the mid 1990s.

In order to employ LDPC codes in everyday applications, significant research work has been carried out on the design of LDPC decoders in the past few years [3]. In these architectures, three main factors have been considered as the design goals. Firstly to maximize the throughput by taking advantage of inherent parallelism in the decoding algorithm, while trying to minimize the consumed resources such as silicon area, memory and routing. Thirdly, LDPC decoders usually need to have a high degree of flexibility, to allow quick and easy modification to the code being tested.

Although quite a few published works have considered flexibility in their architectures, they are generally restricted to a class of LDPC codes such as the quasi-cyclic codes [4]. Nonetheless, LDPC decoders that are capable of decoding arbitrarily constructed LDPC codes have been introduced in recent years [5], [6]. These decoders, are based on the same reformulation of the decoding algorithm, making a trade-off between the flexibility and throughput/area. The performance of LDPC codes are studied via simulations. Hardware simulations are significantly faster [7] than the software counterparts, but typically suffer from inflexible implementation. Hence, it is useful to find flexible ways of implementing LDPC codes in hardware, where the performance of different codes can be quickly and easily compared.

In this work, a new solution, referred to as message packing, that combines the outputs of several VNs or CNs in a pack, is proposed. A method of finding the messages that can be combined in a pack is also proposed, together with an architecture aware IN that permits the exchange of messages between the CNs and VNs. The proposed network is simpler than other similar designs [5], [6].

The remainder of this paper is organized as follows. Section 2 describes the LDPC codes and the decoding algorithm. In Section 3, the message packing method is presented. Section 4 compares the message delivery network with other similar designs. Finally, we conclude our work in section 5.

II. LDPC CODES

LDPC codes are a type of linear block codes for which the $M \times N$ parity-check matrix, $H$, is sparse, (has a small number of ones in each row and column). LDPC codes, may be graphically represented by a Tanner graph. This is a bipartite graph with two types of nodes: variable nodes (VNs) and check nodes (CNs). There are $M$ CNs and $N$ VNs corresponding to the number of rows and columns in $H$. An edge exists between CN $m$ and VN $n$ if and only if $H_{m,n} = 1$. The VNs also represent the $N$ bits of each codeword that should satisfy $M$ parity check equations. A parity-check matrix and its Tanner graph are depicted in Fig. 1.

The sum-product algorithm (SPA), also known as the message passing algorithm (MPA), is an efficient iterative method to decode LDPC codes [8]. The MPA can be expressed on the Tanner graph in which messages are passed from CN (VN) to VN (CN) along the edges of the graph. The SPA is computationaly complex. The Offset Min-Sum algorithm, that has one extra addition, but improves the performance significantly when compared with the original Min-Sum approximation, has been proposed in [9]. During the CN update of the Offset Min-Sum algorithm, the CNs receive and update information from the VN's.
A generic prototype of such a decoder consists of two sets of compared to the direct implementation of the Tanner graph. or check node processor (CNP), significantly reduces the area (CNs) is updated by a single variable node processor (VNP) in equations (1) and (2), which also enables us to deliver the This rearrangement has been applied to the Offset Min-Sum that it reduces the memory requirements of the algorithm [3].

The offset parameter \( \beta \) is optimized via density evolution. \( \lambda_i \) is the log-likelihood ratio (LLR) message from VN \( i \) and \( u_{m,n} \) is the message to be passed from CN \( m \) to VN \( n \). The offset parameter \( \beta \) is optimized via density evolution. The VN update equation is given as:

\[
    \lambda_n^{(l)} = \frac{2}{\sigma^2} r_n + \sum_{i \in M(n)} u_{i,n}^{(l)}
\]

where \( r_n \) is the \( n \)th bit of the received codeword, \( \sigma^2 \) is the variance of the channel noise, (assumed to be additive white Gaussian Noise). Equations (1) and (2) are iterated until a valid codeword is detected or a maximum number of iterations has been reached.

The original Sum-Product algorithm can be rearranged so that it reduces the memory requirements of the algorithm [3]. This rearrangement has been applied to the Offset Min-Sum in equations (1) and (2), which also enables us to deliver the same value from a VN to all connected CNs.

III. MESSAGE PACKING DECODING ARCHITECTURE

Partially-parallel LDPC decoders, in which a subset of VNs (CNs) is updated by a single variable node processor (VNP) or check node processor (CNP), significantly reduces the area compared to the direct implementation of the Tanner graph. A generic prototype of such a decoder consists of two sets of VNPs and CNPs, which are connected to each other through a bidirectional IN. Hence, a minimum of \( [E/p] \) cycles are required for a half-iteration, if there are \( p \) processors of each type and the Tanner graph has \( E \) edges.

There are two types of deliveries in the MPA, the check-to-variable (CTV) and variable-to-check (VTC) deliveries. In each type of delivery, it is desired that all processors at the output of the network receive a message at any time instance. Note that the delivery and updating of nodes can be overlapped, i.e. the CNs are updated during the VTC delivery. We take advantage of the aforementioned property of high-rate codes (number of VNs >> number of CNs) to simplify the delivery network.

A. Message packing for variable-to-check delivery

Assuming \( N \) is the codeword length of the LDPC code, \( N \) VNs need to be distributed amongst \( p \) VNP groups. We assume that every VNP handles \( \lfloor N/p \rfloor \) VNs, except for the last one which is handles the remaining \( N - (p - 1) \lfloor N/p \rfloor \) VNs. Moreover, the simple assignment of the first \( \lfloor N/p \rfloor \) VNs being assigned to the first VNP, etc... can be used. The same applies to the \( M \) CNPs with \( p \) CNPs.

According to equation (2), a VN sends the same value to all connected CNs. This allows one to deliver VN messages in \( N \) rather than \( E \) cycles. Assume that the VNs have a maximum degree of \( j \), i.e. a VN’s message is delivered to at most \( j \) CNPs. If \( p > j \), it might be possible to deliver more than one VN each cycle. \( q = \lfloor p/j \rfloor \) VN messages can be sent over the network, if the values do not conflict at the output of the network. We call these \( q \) VNs a pack and the delivery method, message packing. This would further reduce the VTC delivery time from \( N \) to \( N/q \), equal to the maximum number of packs. As, \( j \) is small for high-rate codes, it can result in a significant reduction in the delivery time. For instance, if \( p = 16 \) and \( j = 4 \), there will ideally be a four times reduction in the delivery time. Therefore, the main issue is how to find the packs (each containing four VNs) so that a maximum number of VNs will be covered. Additionally, we are interested in employing less expensive INs rather than non-blocking multicast crossbar networks.

We tackle the problem by dividing \( p \) VNP groups into \( j \) groups, each of which contains an equal number of VNs. The value of \( j \) should be a factor of \( p \), which also defines the maximum VN degree supported by this decoder. We work with the case when \( p = 16 \) and \( j = 4 \), which makes every four VNP a packing group, however it is noted that this algorithm is true in general. We form four VNP groups in our solution: \( G_1 = \{ \text{VNP}_1, \ldots, \text{VNP}_4 \} \), \( G_2 = \{ \text{VNP}_5, \ldots, \text{VNP}_8 \} \), \( G_3 = \{ \text{VNP}_9, \ldots, \text{VNP}_{12} \} \) and \( G_4 = \{ \text{VNP}_{13}, \ldots, \text{VNP}_{16} \} \). We propose two stage delivery network shown in Fig. 2. The first stage switches between packing groups (\( G_1 \) to \( G_4 \)) and the second one generates the fanout. The network is built using \( p/j \) :1 multplexers and \( p \lfloor j/1 \rfloor \) multplexers.

In order to find the packs, we work on each of the four groups (\( G_1 \), \( G_2 \), \( G_3 \) and \( G_4 \)) individually so that a pack is formed by the VNs all from one of the groups’ VNP. Additionally, the VNs are selected such that they do not conflict at their destination CNPs. For this, we propose a

![Parity-check matrix](image)

Fig. 1: The parity-check matrix and its graphical representation.
new algorithm based on the Backtracking algorithm. Here, we have four levels, corresponding to four VNPs that belong to a packing group. The algorithm searches each level (VNP) to find a valid VN such that it does not conflict with the previously selected VNs. Since there are four levels, an answer consists of four VNs, which form a pack. If the algorithm is unable to find a valid node at any level, it backtracks to the previous level and finds another valid node. The search stops whenever all the VNs are assigned to a pack, or else it is not possible to expand the non-packed nodes of the first level to the last level according to the constraints.

The proposed algorithm applies a heuristic to the original Backtracking algorithm to quickly obtain a solution that is close to optimum (i.e. maximizes the number of packs). At every level, the heuristic performs a forward checking on each candidate node, determining the number of children it can have at the next level called the degrees of freedom. The heuristic will choose a node with minimum degrees of freedom. After a pack is found, all of its nodes are removed from the search tree. To measure the efficiency of the heuristic, we define the packing efficiency $\eta_p$ as the ratio of the number of packs derived from the algorithm to the maximum number of possible packs.

The packing efficiency has a direct impact on the delivery time. If the number of VNs assigned to a VNP is $S_{vnp}$ and the number of packing groups is $G$ then the delivery time can be written as

$$T = G \cdot S_{vnp} \cdot \eta_p + (1 - \eta_p) \cdot S_{vnp} \cdot p \text{ cycles} \quad (3)$$

In the general case, $S_{vnp} \approx N/p$ and the number of packing groups will be $G = \left\lceil \frac{p}{j} \right\rceil \approx j$. Substituting these values into (4) results in the following approximation:

$$T \approx \left[ 1 - (1 - \frac{j}{p}) \eta_p \right] N \text{ cycles} \quad (4)$$

It can be observed that the delivery time depends on three factors: the amount of parallelism or $p$, the maximum VN degree, $j$, and the algorithm efficiency $\eta_p$.

**B. Message packing for check-to-variable delivery**

Since the CN’s degree is much larger, and CNs send different messages to their corresponding VNs, a different approach is used for the CTV deliveries. In this section, we propose a new method of CTV delivery in which all messages of a single CN are packed together and delivered in one cycle.

In the Min-Sum decoding algorithm (and its variants), the message sent from a CN can only have two magnitudes independent of its degree. In [3], the authors introduce a way to store messages originated from a CN in a compact form comprised of four parts as shown in Fig. 3. These parts include: the minimum value, the second minimum, the minimum index (position) and the sign pattern, which contains the sign of all messages being sent to the VNs. Here, we take advantage of this method not only for storing values but also for the delivery of CN messages. We call each of these words a row pack.

The row packs, stored in the CNPs, are fetched and sent to all VNPs using the architecture in Fig. 4. Since the CN degree is typically higher than the number of VNPs in high-rate codes, it is very likely that there exists at least a VN in each VNP which should be updated by the CN message. However, there is a possibility that the received message by a VNP is engaged in the update of more than one VN. Hence, a row pack has to be held at the input of VNPs for a few cycles...
leading to a large delay in the CTV delivery and VN update. In order to avoid this, a FIFO is located at the input of every VNP. The unbalanced distribution of different rows compensates each other, i.e. a VNP may process a row in one cycle while it processes the next one in three cycles. The delivery is then halted as soon as a FIFO becomes full. Our experiments on different LDPC codes have shown that a FIFO depth of sixteen is adequate.

### IV. DISCUSSION AND RESULTS

The message packing algorithm was implemented in C to investigate its performance on some high-rate LDPC codes. This C code emulates the VTC and CTV deliveries based on the proposed algorithm and derives the according cycles for every input LDPC code. Additionally, a set of high-rate LDPC codes are chosen to examine the performance of the message packing algorithm. Codes I to IV are unstructured LDPC codes taken from Mackay’s website [10]. Code V is a structured code constructed using the method in [4]. Code VI is the LDPC code defined in IEEE802.3an standard. The CTV and VTC cycles are derived via simulations and the overhead (normalized to $E/p$ cycles per half iteration) of each phase is provided in Table I.

The VTC overhead ranges from 14.2% to 70.8% for our codes. The minimum overhead is for codes with VN degree of four since the architecture is optimized for this value. Code V is an example of structured code and the CTV overhead (in the last column of Table I) shows 0% overhead since the graph edges, originating from each CN, are equally assigned to all VNP’s (since the CN degree is equal to the number of VNPs). The CTV overhead increases once the CN degree becomes either smaller or greater than the number of VNPs.

In addition to the algorithm performance, it is desired to study the complexity of delivery networks by comparison with other similar designs. Two similar flexible LDPC implementations are presented in [6] and [5], which are based on a similar reformulation of the Sum-Product algorithm. The decoder in [5] employs a Beneˇs network for message passing while the message delivery in [6] is done through a complex network of multiplexers and registers that allows all VNP’s and CNP’s to access every single node on the Tanner graph. Although the latter network results in a higher throughput, it necessitates a vast amount of resources that would not be implementable for the case of large codes with high node degrees. The proposed delivery network is Synthesized on Xilinx Virtex4 FPGA and is compared to architectures of [6] and [5] in Table II. A rate 0.9 LDPC code with length of 4000 and CN degree of 40 is considered for the comparisons since the architecture of [6] relies on the code. Comparison results shows that the proposed delivery network in this work requires fewer resources than other designs.

### V. CONCLUSION

In this paper, we have proposed an efficient interconnection framework for the design of general LDPC decoders. The proposed architecture is based on a new method for delivery of several messages in the message passing algorithm, named as message packing. This solution is optimized for high-rate LDPC codes, and consists of two distinct schemes for delivery to CNs and VNs. The proposed method has been examined for some high-rate codes. The results reveal the ability to decode an arbitrary designed LDPC code. This high degree of flexibility comes with the expense of an overhead in decoding time ranging from 9.6% to 68.7%. Additionally, the IN is synthesized and compared to other flexible decoders, showing a simpler network in terms of area and control.

### REFERENCES


### TABLE I: VTC and CTV delivery performance for different high-rate LDPC codes.

<table>
<thead>
<tr>
<th>Code</th>
<th>Edges</th>
<th>$p$</th>
<th>IPT (cycles)</th>
<th>VTC cycles</th>
<th>overhead</th>
<th>CTV cycles</th>
<th>overhead</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>(4095,3358)</td>
<td>12285</td>
<td>16</td>
<td>768</td>
<td>1080</td>
<td>40.6%</td>
<td>1046</td>
</tr>
<tr>
<td>II</td>
<td>(4095,3359)</td>
<td>16380</td>
<td>16</td>
<td>1024</td>
<td>1170</td>
<td>14.2%</td>
<td>1223</td>
</tr>
<tr>
<td>III</td>
<td>(4376,4094)</td>
<td>17504</td>
<td>16</td>
<td>1094</td>
<td>1286</td>
<td>17.5%</td>
<td>1114</td>
</tr>
<tr>
<td>IV</td>
<td>(1057,244)</td>
<td>3171</td>
<td>16</td>
<td>199</td>
<td>340</td>
<td>70.8%</td>
<td>312</td>
</tr>
<tr>
<td>V</td>
<td>(2048,1536)</td>
<td>8192</td>
<td>16</td>
<td>512</td>
<td>646</td>
<td>26.2%</td>
<td>512</td>
</tr>
<tr>
<td>VI</td>
<td>(2048,1664)</td>
<td>12288</td>
<td>32</td>
<td>384</td>
<td>644</td>
<td>67.7%</td>
<td>652</td>
</tr>
</tbody>
</table>

### TABLE II: Complexity of delivery networks.

<table>
<thead>
<tr>
<th>$p$</th>
<th>Slices</th>
<th>Control bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>5.24M</td>
<td>448</td>
</tr>
<tr>
<td>32</td>
<td>10.48M</td>
<td>1152</td>
</tr>
</tbody>
</table>