Parallel Architecture for Decoding LDPC Codes on High Speed Communication Systems

Damián A. Morero, Graciela Corral-Briones, and Mario R. Hueda

Digital Communications Research Laboratory - National University of Cordoba - CONICET
Av. Vélez Sarsfield 1611 - Córdoba (X5016GCA) - Argentina
Emails: dmoreno, gcoral, mhueda@com.uncor.edu

Abstract—This paper presents a novel parallel architecture for decoding LDPC codes. The proposed architecture has low memory and interconnection requirements, becoming attractive for high speed applications such as fiber optic communications and high density magnetic recording. As an example, the implementation on an FPGA of a TPC/SPC code using the proposed architecture will also be described.

I. INTRODUCTION

The use of iterative decoders based on Low Density Parity Check (LDPC) codes has allowed to reach information rates close to Shannon’s channel capacity [1]. SISO (Soft Input/Soft Output) decoding of LDPC codes using the Sum-Product Algorithm (SPA) requires approximately one order of magnitude less calculations than equivalent Turbo Codes [2]. Nevertheless, the implementation complexity of SISO LDPC detectors is one of the main obstacles that conditions its viability on commercial integrated circuits. High interconnection complexity and important amounts of memory are required. This issue becomes more important on high-speed applications where parallel processing schemes are needed.

Different low-complexity architectures for implementing LDPC codes have been proposed. [3] describes one that has as drawback a degradation on the decoder performance due to a simplification on the SPA algorithm. Architectures described in [2] y [4] have high requirements of memory and interconnection. [5] y [6] show architectures focused on implementing LDPC codes with particular structural properties. The present paper introduces a new architecture for implementing a wide family of LDPC codes. It operates in a parallel fashion without requiring approximations on the SPA algorithm, and requires a reduced amount of memory and interconnection complexity. These features make the architecture desirable for implementing on integrated circuits.

The rest of the paper is organized as follows. Section II gives a brief introduction to LDPC codes. Section III explains the decoding algorithm. Section IV describes the proposed novel architecture that implements the algorithm explained on the previous section. Section V shows the results after implementing on an FPGA the architecture proposed on Section IV. Finally, section VI presents the conclusions.

II. LOW-DENSITY PARITY CHECK CODES

An LDPC code is a linear block code defined by a sparse parity check matrix \( H \), of dimensions \( (m \times n) \). This matrix defines a code of length \( n \) and dimension \( k \), where \( k = n - \text{rank}(H) \). A parity check matrix \( H \) has a bipartite graph associated with it, called Tanner graph (TG), which fully characterizes the code. The Tanner graph is composed of two types of nodes: variable nodes \( v_i \) and check nodes \( c_j \). The nodes \( v_i \) and \( c_j \) represent the \( i \)-th coded bit or column of \( H \), and the \( j \)-th check equation or row of \( H \) respectively. In a TG, connections exist only between a variable node and a check node. \( v_i \) is connected to \( c_j \) if and only if \( H_{i,j} \neq 0 \), where the indices \( j \) and \( i \) mean \( H \)'s row and column respectively. The degree of a TG’s node is determined by the number of connections the node has; i.e. the degree of \( v_i \) and \( c_j \) represent the number of non-zero elements on the \( i \)-th column and on the \( j \)-th row of \( H \) respectively. If all \( v_i \) nodes have the same degree \( \gamma \) and all \( c_j \) nodes the same degree \( \rho \), then the code is said to be regular. Let \( V \) be a set of variable nodes and \( C \) a set of check nodes. If \( v_i \in V \) and \( c_j \in C \), the set of neighbors of \( v_i \) is denoted as \( g(v_i) \in C \) and \( g(c_j) \in V \) respectively; i.e. \( g(v_i) = \{ c_j \in C : H_{i,j} \neq 0 \} \) and \( g(c_j) = \{ v_i \in V : H_{i,j} \neq 0 \} \).

This paper focuses on regular LDPC codes that hold the following structural property. That is: \( P_C = \{ V_q : V_q \subseteq V \land V_q \cap V_{\neq q} = \emptyset \land \cup_q V_q = V \} \) is a partition of \( V \) and with the following properties:

- \( P_1: \forall V_q \in P_C \) it is true that \( g(V_q) = C \)
- \( P_2: \forall V_q \in P_C \) let \( v_i, v_j \in V_q \) where \( i \neq j \); therefor it is true that \( g(v_i) \cap g(v_j) = \emptyset \).

A partition \( P_C \) fulfilling the properties mentioned above is called a Variable-node Partition with Full Check-node Connectivity (VPFCC); and a code in which it is possible to create a partition of this nature is said to be a VPFCC-code. The architecture proposed in this paper to implement

1A sparse matrix is a matrix populated primarily with zeros.
an LDPC decoder is grounded on these specific properties of a partition $P_V$.

There are several LDPC codes that satisfy the characteristic properties of VPFCC-codes. A set of these are the Turbo Product Codes based on Single Parity Check (TPC/SPC). [7], [8] y [9] suggest that these codes are suitable for iterative detection schemes for magnetic recording. In fact, the implementation of a TPC/SPC code is described in this paper to illustrate the proposed decoding architecture. The utility the VPFCC property of an LDPC code has on the new architecture will be analyzed in Section IV. Next, the main characteristics of TPC/SPC codes will be briefly described.

A. TPC/SPC Codes

A TPC/SPC code is built through a multi-dimensional array of code words derived from other previously defined code or codes. The codes used for building the multi-dimensional array are called base codes. Typical base codes are: Hamming, BCH, and simple parity check codes. Within TPC codes, those based on bi-dimensional arrays (denoted as 2D-TPC) are of special interest. A 2D-TPC code consisting of two component codes $C_1$ and $C_2$ with parameters $(n_1, k_1, d_1, G_1)$ and $(n_2, k_2, d_2, G_2)$ respectively, has parameters $(n_1n_2, k_1k_2, d_1d_2, G_1 \otimes G_2)$; where $n$, $k$, $d$, $G$ are the length, dimension, minimum distance, and generator matrix of the code respectively, and $\otimes$ represents the Kronecker product. Within the 2D-TPC codes, the attention is focused on those based on single parity check codes (SPC) of length $N$ and dimension $N-1$, denoted as 2D-TPC/SPC($N,N-1$)² [7], [8] y [9]. Fig. 1 shows the TG and its structure for a 2D-TPC/SPC($4,3$)² code.

2D-TPC/SPC codes are all VPFCC-codes. This can be seen through the 2D-TPC/SPC($4,3$)² example code of Fig. 1, from which the partition $P_V = \{ V_1, V_2, V_3, V_4 \}$ is built, where:

$$
\begin{align*}
V_1 &= \{ B_1, B_5, B_9, P_4 \} \\
V_2 &= \{ B_2, B_6, P_7, P_1 \} \\
V_3 &= \{ B_3, P_6, B_7, P_2 \} \\
V_4 &= \{ P_5, B_4, B_8, P_3 \}.
\end{align*}
$$

It can be verified that this partition holds properties (P1) and (P2), making 2D-TPC/SPC($4,3$)² a VPFCC-code. This result can be generalized to a 2D-TPC/SPC($N,N-1$)² code by creating a partition $P_V = \{ V_i : i = 1, \ldots, N \}$, where $V_i$ is the $i$-th modular-diagonal from the square array of $N \times N$ variable nodes. The modular-diagonal of a $N \times N$ square array of elements $x_{i,j}$ is defined as the sets

$$
D_k = \{ x_{i,j} : i = 1, \ldots, N \wedge j = (i + k - 1) \mod(N) \}. \quad (1)
$$

III. THE SUM-PRODUCT ALGORITHM

Let $b_i$ be the $i$-th bit of the code word. The Sum-Product Algorithm (SPA) [10] takes as input the a priori log-likelihood ratio of each bit:

$$
AprLLR_i = \ln \left( \frac{P(b_i = 1)}{P(b_i = 0)} \right). \quad (2)
$$

It then iterates on the computation of the a posteriori log-likelihood ratio:

$$
ApoLLR_i = \ln \left( \frac{P_C(b_i = 1)}{P_C(b_i = 0)} \right), \quad (3)
$$

where

$$
P_C(b_i) = \sum_{b_j : j \neq i} P(b_1, \ldots, b_n | AprLLR_1, \ldots, AprLLR_n),$$

being $P(\cdot | \cdot)$ the joint probability function of the code word given the a priori information. One iteration of the SPA algorithm consists of two steps. In the first one (A), messages are calculated and sent from the variable nodes $v_i$ to the check nodes $c_j$, $M(v_i \to c_j)$. In the second step (B), messages are calculated and sent from the check nodes $c_j$ to the variable nodes $v_i$, $M(c_j \to v_i)$. Steps (A) and (B) are repeated as many times as required iterations. Finally, in a third step (C), the $ApoLLR$ is computed. Summarizing, the calculations performed on each step are the followings:

A. Messages from variable to check nodes:

$$
M(v_i \to c_j) = AprLLR_i + \sum_{c_k \in g(v_i) \cap c_j} M(c_k \to v_i) \quad (4)
$$

B. Messages from check to variable nodes:

$$
M(c_j \to v_i) = \phi^{-1} \left\{ \sum_{v_k \in g(c_j) \cap v_i} \phi \left[ M(v_k \to c_j) \right] \right\} \quad (5)
$$

where $\phi(x) = \ln \left[ \tanh \left( x/2 \right) \right]$, $\phi^{-1}(x) = 2 \tanh^{-1}(e^x)$.
messages can be rewritten in the following way:

\[ M(c_j \rightarrow v_i) = \sum_{c_k \in \mathcal{G}(v_i)} M(c_k \rightarrow v_i) \]  

This version is based on the following approximation:

\[ \psi^{-1}(\phi(a) + \phi(b)) = 2 \tanh^{-1} \left( \frac{\tanh \left( \frac{a}{2} \right) \cdot \tanh \left( \frac{b}{2} \right)}{1 - e^{-|a| - |b|}} \right) \]

As the BCV block, which computes messages \( M(c_j \rightarrow v_i) \), consumes most of the computational requirements of the SPA algorithm, a modified version is usually implemented. This version is based on the following approximation:

\[ \psi^{-1}(\phi(a) + \phi(b)) = 2 \tanh^{-1} \left( \frac{\tanh \left( \frac{a}{2} \right) \cdot \tanh \left( \frac{b}{2} \right)}{1 - e^{-|a| - |b|}} \right) \]

In this way, the computation of the check to variable messages can be rewritten in the following way:

\[ M^*(c_j \rightarrow v_i) = \min_{v_k \in \mathcal{G}(c_j) \setminus v_i} \left\{ |M(v_k \rightarrow c_j)| \right\} \]

\[ \cdot \prod_{v_k \in \mathcal{G}(c_j) \setminus v_i} \text{sign}(M(v_k \rightarrow c_j)) \]

C. A posteriori log-likelihood ratio (ApoLLR) computation:

\[ \text{ApoLLR}_i = \text{AprLLR}_i + \sum M(c_k \rightarrow v_i) \]  

The computation in steps (A), (B), and (C) is performed in independent blocks named BVC, BCV, and BApoLLR respectively. As the BCV block, which computes messages \( M(c_j \rightarrow v_i) \), consumes most of the computational requirements of the SPA algorithm, a modified version is usually implemented. This version is based on the following approximation:

\[ \psi^{-1}(\phi(a) + \phi(b)) = 2 \tanh^{-1} \left( \frac{\tanh \left( \frac{a}{2} \right) \cdot \tanh \left( \frac{b}{2} \right)}{1 - e^{-|a| - |b|}} \right) \]

In this way, the computation of the check to variable messages can be rewritten in the following way:

\[ M^*(c_j \rightarrow v_i) = \min_{v_k \in \mathcal{G}(c_j) \setminus v_i} \left\{ |M(v_k \rightarrow c_j)| \right\} \]

\[ \cdot \prod_{v_k \in \mathcal{G}(c_j) \setminus v_i} \text{sign}(M(v_k \rightarrow c_j)) \]

IV. PROPOSED ARCHITECTURE

The proposed implementation architecture is based on the order that messages \( M(v_i \rightarrow c_j) \) and \( M(c_j \rightarrow v_i) \) are computed and evaluated. Fig. 2 shows the BVC and BCV block architectures used to compute the messages \( M(v_i \rightarrow c_j) \) and \( M(c_j \rightarrow v_i) \), respectively. The BVC block works in a parallel way while the BCV blocks do it recursively, minimizing thus their complexity. Fig. 4 shows the complete architecture of a SPA decoder with two iterations for a 2D-TPC/SPC(4,3) code. The system parallelism is 4, i.e., on each clock cycle 4 values of AprLLR and ApoLLR input and output respectively. The input and output sequence corresponds with the partition elements \( P \mathcal{V} = \{ V_i : i = 1, \ldots, 4 \} \) described in Section II. Fig. 3 shows how messages of the SPA algorithm are computed when the partition \( P \mathcal{V} \) is used. Property (P2) of \( P \mathcal{V} \) prevents a conflict between \( M(v_i \rightarrow c_j) \) messages, i.e., on each clock cycle there will not be two \( M(v_i \rightarrow c_j) \) messages arriving at the same check node. Property (P1) ensures that, on each clock cycle, all check nodes will have input information available. As Fig.3 shows, only \( N = 4 \) variables nodes are updated on each clock cycle. Due to this property, only 4 BVC blocks need to be implemented instead of the \( N^2 = 16 \) that a full parallel architecture would need. Multiplexers shown in Fig. 4 interconnect the 4 BVC blocks with the corresponding BCV blocks of each set \( V_i \) of the partition \( P \mathcal{V} \).

For the case of the 2D-TPC/SPC(4,3) family of codes, the proposed architecture presents the following char-
Figure 4. SISO decoder architecture with two SPA iterations for the 2D-TPC/SPC(4,3) code.

characteristics:
- The partition of the computation flow in the SPA algorithm saves the need to store the \( M(v_i \rightarrow e_j) \) and \( M(e_j \rightarrow v_i) \) messages, which are computed and used on demand.
- The possibility to use a recursive architecture for the BCV block while maintaining a parallel processing of the whole decoder reduces the implementation complexity.
- Only \( N \) BCV blocks are implemented, instead of \( N^2 \).

The architecture of Fig. 4 can be easily generalized to any other LDPC code fulfilling the VPFCC property. Besides, the proposed technique can be implemented recursively without major modifications. This allows to do several SPA iterations by only instantiating the hardware that corresponds to an iteration, e.g. the first half of the architecture shown in Fig. 4.

V. FPGA IMPLEMENTATION

The synthesis of the proposed architecture was performed on an FPGA Virtex 5 (XC5VLX330) from Xilinx. The decoder was designed for the 2D-TPC/SPC(32,31) code with the following parameters: \( (n = 1024, k = 961, d = 4) \) and rate \( R = 0.9384 \). All the signals present in the decoder were digitized using 8 bits precision variables. The resources used were: 16952 registers out of 297369 (8.18%) and 14402 lookup tables (LUTs) out of 207360 (6.95%).

VI. CONCLUSIONS

This paper proposed a novel parallel architecture for decoding VPFCC-type LDPC codes. This architecture is suitable for working on high speed applications, requiring low memory and interconnection complexity. These features make the architecture attractive for implementing iterative decoders on integrated circuits.

REFERENCES