High-Speed Conflict-Free Layered LDPC Decoder for the DVB-S2, -T2 and -C2 Standards

C. Marchand, L. Conde-Canencia and E. Boutillon

Université Européenne de Bretagne, Lab-STICC, UBS, UMR 6285, 56100 Lorient, France.
Email: cedric.marchand@univ-ubs.fr

Abstract
Layered decoding is known to provide efficient and high-throughput implementation of LDPC decoders. However, the implementation of layered architecture is not always straightforward because of memory update conflicts in the a posteriori information memory. In this paper, we focus our attention on a particular type of conflict that is due to multiple-diagonal sub-matrices in the DVB-S2, -T2 and -C2 parity-check matrices. We propose an original solution that combines repetition of the concerned layers and the write disable of the a posteriori information memory. The implementation of this solution on an FPGA-based LDPC decoder led to an average air throughput of 200 Mbit/s with a parallelism of 45 and a clock frequency of 300 MHz. Increasing the parallelism to 120 led to an average air throughput of 720 Mbit/s with a clock frequency of 400 MHz on CMOS technology.

Index Terms—DVB-S2, layered decoder, Low-Density Parity-Check (LDPC) code, memory conflict, VLSI implementation.

1. Introduction
Low Density Parity-Check (LDPC) codes [1] have gained a lot of attention due to their remarkable error correcting capabilities. The design of structured codes [2] allows practical hardware implementation of LDPC decoders. This family of codes has been adopted in several standards such as the 2nd Generation Satellite Digital Video Broadcast (DVB-S2) [3] ratified in 2005. More recently, the DVB-T2 and DVB-C2 standards have adopted the same family code as the DVB-S2 standard. In the following, DVB-X2 stands for DVB-S2, -T2 and -C2.

Even if the DVB-X2 standards define structured parity-check matrices, they are not perfectly structured for layered decoding because of the overlapped sub-matrices or Multiple-Diagonal Sub-Matrices (MDSM). These MDSMs lead to memory update conflicts in the a posteriori or Soft Output (SO) memories. The main idea of this paper is to solve the problem by processing twice the layers containing MDSMs and generating the appropriate memory control to cancel the MDSM effects. Layers with MDSM are repeated in such a way that the extra layers contribute also to the convergence of the code. The obtained scheduling can be viewed as a combination of the horizontal scheduling and the replica shuffle scheduling proposed by Zhang and Fossorier [4].

This paper is organized as follows: Section II is dedicated to describe the memory update conflicts and their resolution in the state-of-the-art. Section III describes the repeat process and its implementation. Finally, Section IV presents BER performance and synthesis results.

2. Memory Update Conflicts Due to the DVB-X2 Matrix Structure
After a short review on the layered decoding algorithm, the memory update problem is explained and the existing work on this subject is presented. For further information on layered decoders, readers are invited to read [2] and [5].

2.1. Layered decoding algorithm
The layered algorithm divides each iteration in sub-iterations or layers and uses the intermediate Soft Output (SO) for each variable node \( v = 1, \ldots, N \). First, the \( SO_v \) are initialized with the channel LLR and messages from check node to variable nodes \( M_{c \rightarrow v} \) are initialized to zero. For each sub-iteration:

\[
M_{v \rightarrow c}^{old} = SO_v^{old} - M_{c \rightarrow v}^{old},
\]

\[
M_{c \rightarrow v}^{new} = 2 \tanh^{-1} \left( \prod_{v_c/v} \tanh \left( \frac{M_{c \rightarrow v}^{old}}{2} \right) \right),
\]

\[
SO_v^{new} = M_{v \rightarrow c}^{old} + M_{c \rightarrow v}^{new},
\]

where \( v_c/v \) is the set variable node indices connected to check \( c \) excluding \( v \). Combining (1) and (3), we obtain:
tal layer decoder in [9] and with a vertical layered decoder efficiently applied to the Min-Sum algorithm with an horizon-
cation. In [7] and [8] the contribution of the SO values. This solution requires an additional access to the
when there are DDSM. The selective problem is usually called a “cutting edge”. The existance of
SO are updated serially in the layered architec-
Many papers in the literature consider this problem [6, 5, 7, 8, 9, 10, 11, 12, 13, 14] and their solutions are briefly
This problem is usually called a “cutting edge”. The existence of “cutting edge” leads to performance degradation.
Fig. 1 shows a layer (L) with one Double-Diagonal Sub-Matrix (DDSM). Each square with one diagonal line represents a shifted Identity Matrix (IM). The first square has two diagonals D₁ and D₂ corresponds to a DDSM. Let us consider two check nodes (c₁ and c₂) which are connected through D₁ and D₂ to the same variable node denoted v. During the processing of this layer, the SO value should benefits from c₁ according to
\[
SO_{v}^{new} = SO_{v}^{old} + \Delta M_{c_{1}\rightarrow v},
\]
where \(\Delta M_{c_{1}\rightarrow v}\) is equal to \(M_{c_{1}\rightarrow v}^{new} - M_{c_{1}\rightarrow v}^{old}\).

2.2. Memory update conflicts problem

In [11] the conflicts due to DDSM are efficiently solved by a parallel computation of the horizontal layers. However, this implementation is only possible with the Chinese Mobile Multimedia Broadcasting (CMMB) standard which provides matrices with a constant variable node degree of connection. In [12], an architecture supporting MDSM is designed for CMMB, however, when dealing with a single diagonal sub-matrix, the architecture is underused.

In [13, 14] the parallelism is first reduced to eliminate most of the DDSM, then layers with DDSM are modified with specific scheduling. In [13] the presented design is efficient but limited to a parallelism of \(P = 40\). In [14] the authors describe a design without modification of the layered decoder architecture and using only one barrel shifter. However, simulations showed some degradation for a parallelism higher than \(P = 45\). For this reason, we propose a new technique that successfully solves the matrix-structure conflicts, even when keeping the maximum level of parallelism (i.e., \(P = 360\)).

The main objective of this work is to keep the data-path of the decoder as simple as possible (a single barrel shifter, a simple layered node processor) in order to obtain both high speed clock frequency and low area design. Then, starting from this constraint, we solved the DDSM memory conflicts using an appropriate control.

3. RESOLUTION OF THE MEMORY UPDATE CONFLICTS

In this section the repeat process dedicated to solve the DDSM problem is explained and an efficient architecture is deduced.

3.1. Principle of the repeat process

The idea is to perform twice the layers with one DDSM to allow update of the two diagonals serially. Let us consider the layer in Fig. 1 with two consecutive processing. \(L₁\) will refer to the first processing of the layer and \(L₂\) the second processing. The result of \(L₁\) on \(SO_{v}\) is defined as \(SO_{v}^{L₁}\). The repeat process principle is exposed hereafter by focusing on the evolution of the \(SO_{v}\) value, then control signals required for the architecture are deduced.

During the first occurrence \(L₁\), the conflict occurs as described in Section 2.2: \(SO_{v}^{old}\) is replaced first by \(SO_{v}^{old} (5)\), then overwritten by \(SO_{v}^{old} (6)\). Thus \(SO_{v}^{L₁}\) is updated according to
\[
SO_{v}^{L₁} = SO_{v}^{old} + \Delta M_{c_{2}\rightarrow v}.
\]

During \(L₂\), we consider a specific control process that enables the \(SO_{v}^{old}\) update of equation (6). As a consequence,
SO^{L_2}_{c} is updated by applying equation (5) to SO^{L_1}_{c} which gives:

\[ SO^{L_2}_{c} = SO^{L_1}_{c} + \Delta M_{c_{1} \rightarrow \emptyset}. \quad (9) \]

The result of the layer repeat process is given by combining the result of \( L_1 \) and \( L_2 \) respectively (8) and (9):

\[ SO^{L_{2,2}}_{c} = SO^{old}_{c} + \Delta M_{c_{1} \rightarrow \emptyset} + \Delta M_{c_{2} \rightarrow \emptyset}. \quad (10) \]

From (10), one can conclude that the repeat process allows the contribution of both diagonals \( D_1 \) and \( D_2 \) to \( SO_{c} \) during a decoding iteration.

The specific control process that disables \( SO^{c}_{c} \) update during \( L_2 \) can be easily implemented by an on time Write Disable (WD) signal on the SO memory. To avoid inconsistency at the next iteration, during \( L_2 \) and update of \( M_{c_{2} \rightarrow \emptyset} \) in the \( M_{c_{1} \rightarrow \emptyset} \) memory, the \( M_{c_{1} \rightarrow \emptyset} \) memory must also be Write Disabled to keep \( M_{c_{2} \rightarrow \emptyset} \) unchanged. The same important remark is valid during the \( L_1 \) with \( M_{c_{1} \rightarrow \emptyset} \). Note that equation (8) can also be obtained by a WD of the SO memory during \( D_1 \) update. To summarize, a Write Disable (WD) of the SO and \( M_{c_{1} \rightarrow \emptyset} \) memories must be active during the processing of the diagonal \( D_1 \) of \( L_1 \) and during the processing of diagonal \( D_2 \) of \( L_2 \).

Note that during a decoding iteration, the repeat process implies that all the variables connected to a layer containing a DDSM are updated two times. If the repeated layer are separated by several layers, then the second processing will use more updated SO and thus, contribute more efficiently to the convergence of the decoder. With a well chosen layer scheduling, an iteration with \( x \%) of layers repetition will perform almost equivalent with \( 1 + x \) conventional iterations.

3.2. Conflict free decoder

The resolution of memory update conflicts due to pipeline [15] can be efficiently combined with the repeat process to design a conflict free decoder. This solution is performed in three steps: the first step applies the split process [15] up to the minimum level of parallelism needed to fulfill the speed requirement of the application; the second step solves the problem of the remaining DDSMs by repeating the concerned layers. Finally, the third step applies an efficient layer scheduling (including the duplicated layers) to solve conflicts due to pipeline. In case of remaining pipeline conflicts, an optional fourth step consisting in scheduling the variable group inside the layers [16] can be considered.

The complexity added to the control process to drive the WD signals does not impact the layered architecture. This allows the implementation of a high speed and high parallelism layered decoder such as the one presented in [17].

3.3. Architecture

In Fig. 2, the layered decoder architecture is presented. The Node Processor (NP) computes (1), (2) and (3). Only one barrel shifter (II) is implemented thanks to the computation of the shift variation \( \Delta Shift \) as described in [14]. The modifications required to make the layered decoder architecture compliant with a WD architecture are described hereafter. In a layered decoder, the \( M_{c_{1} \rightarrow \emptyset} \) memory can be made of a FIFO memory as a \( M_{c_{1} \rightarrow \emptyset} \) update occurs only once during one iteration. In case of repeated layers, the \( M_{c_{1} \rightarrow \emptyset} \) are updated more than once and the \( M_{c_{1} \rightarrow \emptyset} \) read and write should be at the same address as during the first call. This can be implemented using a RAM and an ad hoc address generator. When a layer is repeated, an address identical to the first occurrence should be provided.

The WD signal connected to the SO RAM and the \( M_{c_{1} \rightarrow \emptyset} \) RAM can be generated by detecting DDSM and layer repeat occurrence. In the proposed implementation, when two consecutive IM are connected to a common variable group then a DDSM is detected. A DDSM counter is then incremented to access in a specific ROM. The output of the ROM is a single bit that indicates if the current layer of this DDSM is the \( L_1 \) or the \( L_2 \) layer. The DDSM counter is set to zero at each new decoding iteration.

The presented modifications are included in the control block to generate appropriate control signals. These signals are processed in parallel to the layered decoder and do not increase the latency or the critical path of the layered decoder. Thus, the modifications retain the layered decoder efficiency. The splitting process [14], the layer scheduling [15] and the repeat process can be combined and give the designer the freedom to choose the check node update algorithm. Nevertheless, if the \( M_{c_{1} \rightarrow \emptyset} \) are stored in a compressed way, as can be the case for the Min-Sum algorithm and its variations, then a special adaptation of the architecture is required.

3.4. Conflict resolution applied to compressed memories

If the \( M_{c_{1} \rightarrow \emptyset} \) messages are compressed in the \( M_{c_{1} \rightarrow \emptyset}^{MEM} \) memory, there are no more direct accesses to the \( M_{c_{1} \rightarrow \emptyset} \) and \( M_{c_{2} \rightarrow \emptyset} \) in the \( M_{c_{1} \rightarrow \emptyset} \) memory to apply the WD signal.
The proposed solution is to store in a compressed way all the $M_{c\rightarrow v}$ messages that do not belong to a DDSM, and to store individually the messages that belong to a DDSM. The modifications required to the $M_{c\rightarrow v}$ memory are presented in Fig. 3. Two additional memories, $D_1$ FIFO and $D_2$ FIFO are responsible to memorize respectively the $M_{c_1\rightarrow v}$ and $M_{c_2\rightarrow v}$ messages. $D_1$ FIFO stores $M_{c_1\rightarrow v}^{old}$ during the process of layer $L_2$ (no overwrite of $D_1$) with signal $WE_{D_1}$ while $D_2$ FIFO stores $M_{c_2\rightarrow v}^{new}$ during the process of layer $L_1$ with signal $WE_{D_2}$. The read process in the $M_{c\rightarrow v}^{MEM}$ implies three kinds of memory accesses:

1. **Normal access**: generation of the $M_{c\rightarrow v}^{old}$ message based on the compressed information and the current index of the message.
2. **D1 access**: the $M_{c_1\rightarrow v}^{old}$ message is read in $D_1$ FIFO.
3. **D2 access**: the $M_{c_2\rightarrow v}^{old}$ message is read in $D_2$ FIFO.

The over cost due to added memory depends on the number of DDSM. A decoder with $P = 360$ and compatible with all DVB-S2 code rates will require a maximum of 35 DDSM (code rate=5/6). The number of $M_{c\rightarrow v}$ messages ($\#M_{c\rightarrow v}$) which are required to be stored is given by $\#M_{c\rightarrow v} = 35 \times 360 \times 2 = 25200$ (over 285120 edges). With $P = 120$, this number is reduced to $\#M_{c\rightarrow v} = 16 \times 120 \times 2 = 3840$ and with $P = 45$, $\#M_{c\rightarrow v} = 4 \times 45 \times 2 = 360$.

### 4. APPLICATION CASE

The WD architecture was simulated and implemented on an FPGA platform for validation purpose and synthesized on CMOS technology for comparison with state of art.

#### 4.1. Simulation results

Fig. 4 illustrates fixed-point simulation results for a rate-2/3, standard frame (i.e. long frame) with the Normalized Min-

![Fig. 3. Modified $M_{c\rightarrow v}$ memory](image)

![Fig. 4. BER for R=2/3 long frame DVB-T2 LDPC code for a decoding throughput equivalent to 30 decoding iterations without layer repetition](image)
Table 1. Layered decoder CMOS implementation comparison

<table>
<thead>
<tr>
<th></th>
<th>[8]</th>
<th>[19]</th>
<th>This</th>
</tr>
</thead>
<tbody>
<tr>
<td>Parallelism</td>
<td>180</td>
<td>180</td>
<td>120</td>
</tr>
<tr>
<td>Algorithm</td>
<td>3-min</td>
<td>?</td>
<td>NMS</td>
</tr>
<tr>
<td>Throughput[Mbit/s]</td>
<td>180</td>
<td>135</td>
<td>720</td>
</tr>
<tr>
<td>Frequency[MHz]</td>
<td>270</td>
<td>174</td>
<td>400</td>
</tr>
<tr>
<td>Extrinsic [bits]</td>
<td>6</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>Channel [bits]</td>
<td>6</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>Buffer</td>
<td>?</td>
<td>?</td>
<td>yes</td>
</tr>
<tr>
<td>Capacity[Mbits]</td>
<td>2.68</td>
<td>3.18</td>
<td>2.2</td>
</tr>
<tr>
<td>BCH</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>Technologie[mm²]</td>
<td>65</td>
<td>65</td>
<td>65</td>
</tr>
<tr>
<td>Total area[mm²]</td>
<td>6.03</td>
<td>6.07</td>
<td>5.89</td>
</tr>
</tbody>
</table>

The second design targeted a CMOS 65 nm technology with a parallelism of \( P = 120 \). The preliminary results gave a total area of 5.89 mm² and a clock frequency of 400 MHz, which corresponds to a decoding throughput of 720 Mbit/s. For the maximum parallelism of \( P = 360 \), the repeat layer technique can solve the DDSM conflicts at a cost of a 0.05 dB when no throughput degradation is accepted.

6. REFERENCES


