An area efficient LDPC decoder using a reduced complexity min-sum algorithm

Vikram Arkalgud Chandrasetty*, Syed Mahfuzul Aziz

School of Electrical and Information Engineering, University of South Australia, Mawson Lakes, SA 5095, Australia

Abstract

Hardware implementation of Low-Density Parity-Check (LDPC) decoders using conventional algorithms such as Sum-Product or Min-Sum requires large amount of hardware resources. A rather simplistic way to reduce hardware resources is to reduce the intrinsic message quantization. However this adversely affects the bit error rate (BER) performance significantly. In this paper, a resource efficient LDPC decoder based on a reduced complexity Min-Sum algorithm is presented. It reduces the inter-connect complexity by restricting the extrinsic message length to 2 bits and also simplifies the check node operation. Simulation at the algorithmic level shows that the proposed decoder achieves BER performance better than that of a 3-bit Min-Sum decoder, and therefore addresses the problem of massive BER performance degradation of a 2-bit Min-Sum decoder. The reduction in algorithmic complexity and further hardware optimization of the variable node leads to significant savings in hardware resources compared to 3-bit Min-Sum. An LDPC decoder with a code length of 1152 bits has been implemented on a Xilinx FPGA using the proposed algorithmic and hardware enhancements. With a 0.1 dB BER performance gain to that of 3-bit Min-Sum decoder, the proposed decoder saves about 18% of FPGA slices and provides a higher throughput.

© 2011 Elsevier B.V. All rights reserved.

1. Introduction

Low-Density Parity-Check (LDPC) [1] codes have become one of the most attractive error correction codes due to its excellent performance [2] and suitability in high data rate applications, such as WiMax and DVB-S2 [3]. The inherent structure of the LDPC code makes the decoder achieve high degree of parallelism in practical implementation [4]. LDPC decoding algorithms are primarily iterative [5] and are based on belief propagation message passing algorithm [6]. The complexity of the decoding algorithm is highly critical for the overall performance of the LDPC decoder. Various algorithms have been proposed in the past to achieve tradeoffs between complexity and performance [7,8]. The Sum-Product Algorithm (SPA) [9], a soft-decision based message passing algorithm can achieve the best performance, but with high decoding complexity. Whereas, the Bit-Flip algorithm is a hard-decision based algorithm with least decoding complexity, but suffers from poor performance [8]. Min-Sum Algorithm (MSA) is the simplified version of SPA that has reduced implementation complexity with a slight degradation in performance [9].

Simple arithmetic and logical operations required by MSA renders it suitable for hardware implementation. But the performance of the algorithm is significantly impacted by the quantization of the soft-input messages [10]. The soft messages are often quantized with higher precision to achieve higher bit error rate (BER) performance. However, with higher level of quantization the hardware resource requirement grows. Reducing the quantization leads to reduction in hardware resources, but this comes with degradation in decoding performance, such as BER and average iterations. To date, the performance and alternative hardware implementations of LDPC algorithms that use low level quantization have not been rigorously studied.

Majority of the LDPC hardware architectures reported to date [11–15] use high precision quantized soft-input messages that require large amount of resources. Some of these are ASIC implementation [14,15]. In Ref. [16], dynamic scaling of intrinsic/extrinsic message is proposed and Ref. [17] presents adaptive quantization of intrinsic messages. However, the dynamic scaling and adaptive quantization techniques of messages introduce substantial complexities in practical implementation and hardware resources overhead of the decoder. A number of other techniques are proposed to reduce routing congestion and inter-connect complexity in the LDPC decoders [18–20]. The pulse-width message encoding technique [18] reduces the routing congestion by exchanging extrinsic messages serially between the nodes. The split-row algorithm [19]...
and circular-shift network technique [20] require custom LDPC matrices to alleviate the inter-connect complexity of the decoder.

This paper investigates innovative ways to reduce the hardware resources while still achieving acceptable level of decoding performance. Unlike the concise papers [21,22], this paper first analyzes the performance and hardware implementation complexity associated with the Min-Sum Algorithm (MSA). Enhancements are then proposed in the algorithm to reduce the complexity and improve its overall decoding performance. Simulation results reveal that the proposed Modified Min-Sum (MMS) algorithm achieves significant improvement in decoding performance compared to 2-bit and 3-bit MSA. This paper also proposes hardware optimization of the variable node for FPGA implementation of the decoder. This optimization leads to a massive saving of about 64% of slices for a Xilinx FPGA. With a comparable BER performance to that of 3-bit MSA, FPGA implementation of proposed MMS can save up to 18% of slices and lead to 23% improvement in maximum operating frequency of the implementation. The performance compared to other 3-bit MSA, FPGA leads to a massive saving of about 64% of slices for a Xilinx FPGA.

This paper is organized as follows: Section 2 provides a brief explanation of the original Min-Sum algorithm, followed by detailed illustration of the proposed MMS algorithm in Section 3. An exhaustive performance analysis of the algorithms is discussed in Section 4. Finally, the hardware design and implementation results are provided in Section 5.

2. Min-Sum algorithm

The Min-Sum algorithm (MSA) [9] is the modified version of the Sum-Product algorithm (SPA) [9], where the check node operation is simplified to significantly reduce the complexity of the algorithm. In MSA, the quantized intrinsic message, also known as log-likelihood ratio (LLR) and the extrinsic messages (between variable and check nodes) are of equal length. Hence, the hardware implementation complexity, especially the decoding node inter-connects [24] for MSA drastically increases as the quantized message length increases [10]. The variable node (V) and check node (C) operations are as shown in Eqs. (1) and (2), respectively.

MSA variable node operation:

\[ V_i = LLR_n + \sum_{j \neq i} C_j \]  

where \( n = 1, 2, \ldots, N \) (variable nodes) and \( i = j = 1, 2, \ldots, d_v \) (degree of variable node 'n').

MSA check node operation:

\[ C_k = \prod_{j \neq k} \text{sign}(V_i) \times \min_{j \neq k} |V_i| \]  

where \( l, k = 1, 2, \ldots, d_c \) (degree of check node).

3. Proposed modified Min-Sum algorithm

Although the simplified check node operation in MSA has reduced complexity compared to SPA, the former still requires high precision messages to be exchanged between the decoding nodes in the decoder. This is important to achieve comparable decoding performance to that of SPA, with minimal performance degradation. The level of quantization used in the LLR and extrinsic messages of MSA directly impacts the decoding performance. As the quantization length of the message decreases, the performance and complexity of the algorithm reduces [25]. Studies have shown that there is slight performance loss in going from 5-bit to 4-bit or even 3-bit [10]. Using 2-bit quantized messages in MSA leads to massive reduction in implementation complexity but suffers from significant loss in decoder performance compared to 3-bit MSA. The performance of 2-bit MSA has been improved through optimization reported in [26]. The performance is further improved by the Modified Min-Sum (MMS) algorithm proposed in this paper. The MMS uses higher precision LLR messages while exchanging lower precision (2-bit) extrinsic messages. The check node and variable node operations of the MMS algorithm are described as follows:

3.1. Variable node operation

The variable node operation is similar to that of the original Min-Sum algorithm [9]. The difference in the proposed algorithm is that the variable node (\( V_i \)) performs higher precision quantized LLR operations (\( LLR_n \)), but maps the computed result to 2-bit message to be passed to the check nodes, as in Eq. (3). The 2-bit message consists of a sign bit and a magnitude bit representing the computed LLR sum. The mapping is based on a threshold (\( T_m \)) obtained from simulations. Depending on the message received from the check nodes (\( C_j \)), the 2-bit information is again mapped to constant values (+W or -W) to perform the LLR sum operation in the variable node. These constant values for mapping are also obtained from simulations. The functions for mapping the 2-bit messages are shown in Eqs. (4) and (5)

\[ V_i = g \left( LLR_n + \sum_{j \neq i} f(C_j) \right) \]  

where \( n = 1, 2, \ldots, N \) (variable nodes) and \( i = j = 1, 2, \ldots, d_v \) (degree of variable node 'n').

\[ g(y) = \begin{cases} 
01 & \text{if, } y > T_m \\
00 & \text{if, } 0 \leq y \leq T_m \\
10 & \text{if, } 0 > x \geq -T_m \\
11 & \text{if, } x < -T_m 
\end{cases} \]  

\[ f(x) = \begin{cases} 
+W & \text{if, } x = 01 \\
+w & \text{if, } x = 00 \\
-w & \text{if, } x = 10 \\
-W & \text{if, } x = 11 
\end{cases} \]  

where \( T_m \) is the optimized threshold for mapping obtained from simulations; \( W \) is the optimized higher integer constant obtained from simulations; \( w \) is the optimized lower integer constant obtained from simulations.

3.2. Check node operation

In MSA, the check node is expected to determine the product of the sign of incoming messages and also find the minimum of the magnitude of the input messages [9]. In the proposed MMS, the product of the sign of incoming messages are computed using XOR operation (\( S_k \)) and the minimums are determined using AND operation (\( M_k \)), as in Eqs. (6) and (7), respectively. The check node output message (\( C_j \)) is obtained simply by concatenating the sign bit and the magnitude bit, as in Eq. (8). The message passing between the nodes continues till the parity check is satisfied or maximum iteration is reached.

\[ S_k = \oplus_{i \neq k} V_i^{(m)} \]  

\[ M_k = \land_{i \neq k} V_i^{(m)} \]  

\[ C_k = \{ S_k, M_k \} \]
where \( l, k = 1, 2, \ldots, d_c \) (degree of check node), \( S \) the sign bit of check node message, \( M \) the magnitude bit of check node message, \( V_l^{(s)} \) the sign bit of the message 'l' from variable node and \( V_l^{(m)} \) the magnitude bit of the message 'l' from variable node.

The message mapping in the variable node described above is similar to that presented in Refs. [26, 27]. However, the difference in the proposed MMS algorithm is that, it

a) uses higher precision intrinsic messages (LLR) to improve the decoding performance;
b) requires higher precision addition operation (instead of 2-bit) in the variable node due to use of higher precision intrinsic messages;
c) simplifies the variable node operation by eliminating the scaling factor;
d) incorporates simple 'AND' logic instead of comparators for check node operation.

All these modifications lead to simplified check node and reduced inter-connect complexity. Since the check node operation in (7) requires comparing single-bit inputs to determine the minimum, a simple 'AND' operation is sufficient to perform the comparison. A slight increase in complexity of variable node is introduced due to use of higher precision LLRs. However, the impact of this complexity is insignificant when compared to the advantages gained in improvement of overall performance and complexity of the decoder. The decoding performance and hardware implementation complexity of the MMS algorithm is discussed in the following section.

4. Performance of the proposed algorithm

To ascertain the performance gain from the Modified Min-Sum (MMS) algorithm compared to the original Min-Sum Algorithm (MSA), software simulation models of both have been developed using the C programming language in the Matlab environment. The models represent ½ rate (3, 6) regular LDPC decoders [28] with parameterized code length and are bit-true simulation model to handle quantization, over-flow and saturation conditions that occur in fixed-point operations in a typical hardware. The LDPC codes used in the simulations were generated using Progressive Edge Growth (PEG) algorithm [29]. Simulations were carried out assuming that the code words were modulated using Binary Phase Shift Keying (BPSK) [30] and passed over an Additive White Gaussian Noise (AWGN) channel [30].

The models were executed for 1200-bit code length to analyze the performance of MMS and MSA at a maximum iteration of 10. For the sake of comparison, the performance of MSA is simulated for 3-bit and 2-bit quantization. Also, decoders from [12, 13] are included for comparison. The LLR quantization used in the simulation of MMS is 4-bit. The Monte Carlo simulations were carried out with different values of \( T_m \), \( W \) and \( w \) that resulted in an optimum decoding performance. Extraction of these parameters at \( E_b/N_0 = 4 \) dB is shown in Fig. 1. Some of the configuration of parameters leading to optimum BER performance are selected and tabulated in Table 1.

Note that the proposed MMS algorithm is based on simplification of extrinsic messages, which leads to modification of variable/check nodes irrespective of the modulation scheme. The impact on algorithmic performance due to change in modulation scheme (e.g. QPSK, 16QAM) will have little or no impact on the design parameters (\( T_m, W \) and \( w \)). The parameters are expected to vary based on the quantization of intrinsic and extrinsic messages. So the technique presented here for optimization of \( T_m, W \) and \( w \) should be applicable to other modulation schemes.

![Fig. 1. BER performance for varied values of \( T_m \), \( W \) and \( w \): (a) extraction of \( W \), \( w \) values at \( T_m = 1 \); (b) extraction of \( W \), \( w \) values at \( T_m = 2 \) and (c) extraction of \( W \), \( w \) values at \( T_m = 3 \).](image-url)
over 3-bit MSA. MMS (C4) and 2-bit MSA exhibits error floor at BER of $10^{-6}$ onwards. MMS (C2 and C3) has BER performance close to each other with a slight degradation of about 0.1 dB compared to C1. Note that Refs. [12] and [13] are stochastic based decoders and achieves better performance compared to Min-Sum based decoders. The complexity of such decoders is briefly discussed later in Section 5.2. The decoding iterations required by these algorithms are shown in Fig. 3. A significant improvement of average decoding iterations for MMS compared to 2-bit MSA can be observed. MMS (C1 and C3) has average iterations close to that of 3-bit MSA. But MMS (C2 and C4) requires slightly higher iterations compared to that of 3-bit MSA. Note that MMS (C3) provides optimum performance with a gain of about 0.1 dB at BER of $10^{-6}$ and a negligible loss in average iterations compared to 3-bit MSA. It also uses smaller parameter values ($T_m$ and $W$) compared to other configurations with comparable performance. That means smaller logic units will be used in the architecture of the decoder leading to efficient utilization of hardware resources.

### Table 1

Selected configuration of parameters leading to optimum BER performance.

<table>
<thead>
<tr>
<th>Configurations</th>
<th>BER</th>
<th>$T_m$</th>
<th>$W$</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>6.8 x $10^{-6}$</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>C2</td>
<td>1.8 x $10^{-7}$</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>C3</td>
<td>2.4 x $10^{-7}$</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>C4</td>
<td>4.8 x $10^{-6}$</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>C5</td>
<td>8.2 x $10^{-6}$</td>
<td>1</td>
<td>3</td>
</tr>
</tbody>
</table>

At $w=1$ and $E_b/N_0=4$ dB.

5. Hardware design, optimization and implementation

The design, hardware optimizations and FPGA implementation of fully-parallel LDPC decoders based on the reduced complexity Modified Min-Sum (MMS) algorithm are presented in this section. Although fully-parallel implementation of LDPC decoders requires large amount of hardware resources compared to partially-parallel architectures, the primary motivation for fully-parallel implementation is to ascertain the performance as well as the savings in hardware resources from the reduced complexity algorithm and hardware optimizations presented in this paper. This information can be used to develop suitably partitioned partially-parallel architectures to achieve further savings in hardware resources while providing acceptable level of BER performance [31]. Nonetheless there are many applications which can still accommodate a fully-parallel decoder and benefit from the high throughput [32,33].

The top level block diagram of the proposed decoder depicting the hardware interfaces is shown in Fig. 4. The decoder consists of a global ‘Clock’ and synchronous ‘Reset’ inputs. The maximum permissible number of iterations is determined by the value supplied at the ‘MaxIter’ input. This can be set at a value in the range 0–15. When the ‘Configure’ input is high, the ‘MaxIter’ value is read. The LLRs are fed into the decoder using the ‘Load’ control signal. The decoding process is initiated by the ‘Start’ signal. After the decoding is completed, the ‘Decoded Data’ can be obtained when indicated by the ‘DataOut Ready’ signal. The receipt of data can be acknowledged on ‘DataOut Ack’ to receive the next decoded bit. The number of iterations used for decoding can be obtained from ‘Used Iter’ port. The ‘Decoder Status’ port indicates the progress (Active/Idle) of the decoder.

Note that the LLRs are loaded serially one at a time to the decoder. Similarly, the ‘Decoded Data’ is latched bit by bit serially. This technique is used because of the limited number of Input/Output ports available in the FPGA. It also provides flexibility for implementing LDPC decoders with variable code length without modifying the port configuration.

5.1. Variable node optimization

The proposed MMS algorithm improves the decoding performance compared to the original 2-bit Min-Sum. However the...
variable node requires Look-Up Tables (LUT) for mapping messages to/from check nodes. The mapping is to be performed for converting 4-bit messages to 2-bit values and vice versa as shown in Fig. 5(a). The Look-Up Tables required for message mapping are shown in a typical variable node structure depicted in Fig. 5(b). This is a variable node of degree 3, i.e. it takes three inputs and generates three outputs. Clearly the additional LUTs lead to some hardware overhead. It is possible to reduce this overhead by merging the LUTs required for message mapping and the addition/subtraction block that are normally required by the algorithm. Unless this merging is done at the design level, the synthesis tool uses separate LUT resources for the message mapping and addition/subtraction. The resulting optimized variable node is shown in Fig. 5(c).

The optimized and un-optimized variable nodes for MMS (C3) have been synthesized to estimate the difference in hardware resource requirement. The synthesis results are shown in Table 2. It is clear that the variable node optimization presented above leads to significant reduction (64%) in FPGA slices required.

However, the operating frequency is reduced by 14% in the optimized design due to longer chain of large LUTs in the logic.

5.2. Hardware modeling and analysis

A parameterized hardware model of the LDPC decoder based on the MMS algorithm was developed using the Verilog Hardware Description Language (HDL). As stated earlier, MMS algorithm with configuration C3 is used to design hardware resource efficient decoder architecture. The decoder also incorporates the variable node optimization presented above. To facilitate comparison with the 1200-bit LDPC decoder presented in Ref. [28] the above model was used to synthesize, place and route a 1200-bit LDPC decoder for the Xilinx Virtex 4 (XC4VLX200) FPGA. Behavioral and post synthesis simulations were carried out using ModelSim.

The maximum achievable operating clock frequency for the decoder, as obtained after post-synthesis placement and routing is 123 MHz. The throughput (T) of the decoder is calculated using the formula shown in Eq. (9) [28]. This calculation excludes the serial load time of individual LLRs (before starting the decoding process) and latch time of decoded data (after decoding is complete)

$$T = \frac{r f_{max}}{N_y \theta}$$

where ‘r’ is the rate and ‘C’ is the length of the LDPC code, ‘f_{max}’ is the maximum operating frequency, ‘N_y’ is the number of decoding iterations and ‘\theta’ is the number of clock cycles required to complete one iteration (\theta=1 for fully-parallel architecture implementation of the proposed decoder).

Average throughput of the decoder is computed using the F_{max} obtained from post-synthesis placement and routing, and the average decoding iteration at a given E_b/N_o. At an average decoding iteration of 7 at 3.75 dB E_b/N_o (see Fig. 3) the proposed decoder can achieve an average throughput of \sim 10.5 Gbps.

The routing complexity of the algorithms is assessed by calculating the number of node inter-connects (I_c) in the decoder using the formula shown in Eq. (10)

$$I_c = 2C_E V_d$$

where C_c is the code length, E_t is the extrinsic message length and V_d is the degree of the variable node. The calculation includes the inter-connections from variable node to check node and vice versa, by multiplying the equation by factor 2’ in the formula.

The hardware model of the proposed MMS (C3) decoder was synthesized and simulated to determine the hardware resource requirements and key performance parameters. The results are presented in Table 3 along with the synthesis and simulation results for 2-bit and 3-bit Min-Sum decoders as well as other decoders reported in the literature [28,12,13]. Examination of the number of inter-connects, slices, LUTs and registers in Table 3 provides indication about the amount of hardware resources required by various decoders. Clearly the proposed MMS decoder significantly reduces the hardware resources required compared to 3-bit Min-Sum [28], achieves a higher throughput and has a small gain of 0.1 dB at a BER of 10^{-6}. Compared to 2-bit Min-Sum, the MMS decoder has the same throughput (~7.4 Gbps at maximum iterations, i.e. 10), but

Table 2 Synthesis results of variable nodes.

<table>
<thead>
<tr>
<th>Resources</th>
<th>Un-optimized</th>
<th>Optimized</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Slices</td>
<td>22</td>
<td>8</td>
<td>64%</td>
</tr>
<tr>
<td>LUTs</td>
<td>49</td>
<td>26</td>
<td>47%</td>
</tr>
<tr>
<td>Registers</td>
<td>11</td>
<td>11</td>
<td>–</td>
</tr>
<tr>
<td>Clock</td>
<td>456 MHz</td>
<td>392 MHz</td>
<td>–14%</td>
</tr>
<tr>
<td>FPGA</td>
<td>Xilinx Virtex5 (XC5VLX110T-3FF1136)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Fig. 5. Block diagram of the designed variable node structure: (a) variable node structure for MMS algorithm; (b) un-optimized variable node structure and (c) optimized variable node structure.
significantly improves the BER performance (2.4 dB gain at a BER of $10^{-6}$).

Some recent papers have reported synthesis of LDPC decoders using stochastic algorithms [12,13]. In stochastic based decoders the extrinsic messages are exchanged in a bit-serial fashion between the decoding nodes [34]. One iteration at the bit-level is called a decoding cycle. The termination of the algorithm is dependent on the maximum number of bit-level decoding cycles [35]. This is why the stochastic decoders presented in [12,13] require a very large number of decoding cycles, as shown in Table 3, compared to a maximum iteration count of only 10 used by the proposed decoder. Table 3 also shows that the proposed decoder can achieve a much higher throughput compared to those in [12,13]. With comparable FPGA resource requirement, the proposed decoder suffers a BER degradation of only about 0.6 dB at a BER of $10^{-6}$ compared to [12,13]. The BER performances of the stochastic decoders [12,13] are better due to the very large number of decoding cycles, however the latter adversely affects the throughput as stated above. Another issue with the stochastic decoders is the overhead associated with buffering long extrinsic serial messages.

5.3. Implementation and testing

A fully-parallel LDPC decoder has been implemented on a Xilinx Virtex 5 FPGA (xc5vlx110t-3ff1136) using the proposed MMS (C3) algorithm and variable node optimization. Note that the large throughputs required by certain applications are only achievable by fully-parallel architectures [32,33]. As stated previously, the fully-parallel implementation results can also be used to facilitate the design of suitably partitioned new partially-parallel architectures, providing further hardware savings at acceptable BERs. Alternatively, the proposed MMS decoder architecture can be incorporated into some of the existing partially-parallel architectures [11,36–40] to reduce their hardware resource requirements further.

The decoder implemented is a 1/2 rate (3, 6) regular 1152-bit LDPC code that is compliant with the WiMax standard [23]. A comprehensive testing environment was developed to test the decoder at various levels of abstraction, i.e. from a high-level bit-true simulation model all the way down to the decoder implemented on FPGA [31]. The setup used to test the LDPC decoder is shown in Fig. 6.

An RS232 transceiver module was embedded on the FPGA along with the LDPC decoder module to interface with the computer’s RS232 serial port. MatLab was used to communicate with the FPGA using the serial port. LLRs were generated and sent to the FPGA with appropriate control signals for decoding. The decoded data received via the serial port was used to analyze the performance of the decoder. The same sets of LLRs were used in the software model of the decoder in MatLab. This technique was used to compare and verify the BER performance of the hardware decoder on FPGA incorporating LUT optimization (see Section 5.1) to that of the software model.

The results obtained from testing the LDPC decoder on FPGA are summarized in Table 4. The optimized decoder (where the variable node is optimized) requires 37% less FPGA Slices and saves up to 22% of total power compared to the un-optimized version. However, there is approximately 7% reduction in maximum operating frequency compared to the un-optimized version for reasons stated in Section 5.1. This penalty is negligible when compared to the significant savings achieved in the hardware resources and power.

Various performance results obtained from tests conducted on the MMS decoder implemented on FPGA are shown in Figs. 7–9. For comparison, these figures also report the performance parameters obtained from the software simulation models of the MMS decoder as well as 2-bit and 3-bit Min-Sum. Fig. 7 illustrates that the BER performance of the proposed decoder on FPGA closely follows that of its software simulation model (refer to Section 4) as well as that of 3-bit Min-Sum decoder. The BER improvement compared to 2-bit Min-Sum is quite significant. Fig. 8 shows that at $E_b/N_0=4$ dB the MMS decoder implemented on FPGA has an average iteration count of 7.3. Note that the average iteration count is increased by 0.5 compared to its software model (which

<table>
<thead>
<tr>
<th>Table 3</th>
<th>Comparison of fully-parallel LDPC decoders.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Decoder</td>
<td>Proposed</td>
</tr>
<tr>
<td>LDPC code</td>
<td>(1200, 600)</td>
</tr>
<tr>
<td>Regularity</td>
<td>(3, 6) Regular</td>
</tr>
<tr>
<td>Algorithm</td>
<td>MMS (C3)</td>
</tr>
<tr>
<td>Intrinsic message</td>
<td>4-bit</td>
</tr>
<tr>
<td>Extrinsic message</td>
<td>2-bit</td>
</tr>
<tr>
<td>No. of inter-connects</td>
<td>14,400</td>
</tr>
<tr>
<td>Slices</td>
<td>33,345</td>
</tr>
<tr>
<td>LUTs</td>
<td>58,053</td>
</tr>
<tr>
<td>Registers</td>
<td>15,691</td>
</tr>
<tr>
<td>Clock</td>
<td>123 MHz</td>
</tr>
<tr>
<td>Maximum decoding iterations or cycles</td>
<td>10 decoding iterations</td>
</tr>
<tr>
<td>$E_b/N_0$ at BER of $10^{-6}$</td>
<td>3.75 dB</td>
</tr>
<tr>
<td>Avg. Throughput</td>
<td>~10.5 Gbps at BER of 10^{-6}</td>
</tr>
<tr>
<td>Throughput at maximum iterations</td>
<td>~14.2 Gbps</td>
</tr>
<tr>
<td>FPGA device</td>
<td>Xilinx Virtex 5 (xc5vlx110t)</td>
</tr>
</tbody>
</table>

Fig. 6. Block diagram of FPGA test setup for LDPC decoder.
This difference is due to the approximations made in the values stored in the reorganized LUTs for hardware optimization of the variable nodes (see Section 5.1). More number of bits is in error due to these reorganized LUTs and hence requires additional decoding iterations in the FPGA implemented decoder compared to software simulation model.

Using the average iteration count of 6.8 at $E_b/N_0 = 4$ dB (from Fig. 8) and a clock frequency of 138 MHz. (from Table 4) in Eq. (9), the average throughput of the FPGA implemented decoder can be estimated to be $\sim 11.7$ Gbps. Fig. 9 shows the estimated throughput of the implemented decoder as a function of $E_b/N_0$.

6. Conclusion

This paper has presented an innovative way to reduce the extrinsic message length in LDPC decoders without incurring the BER degradation normally associated with reduction in intrinsic message quantization. As an example, this paper has demonstrated that the proposed MMS decoder with 2-bit extrinsic messages and 4-bit intrinsic messages achieves BER performance slightly better than that of a 3-bit Min-Sum decoder. It also provides higher throughput with much reduced hardware resource requirement. The simulation and FPGA implementation results clearly validate these claims. The results also demonstrate that the proposed decoder addresses the problem of massive degradation in BER performance resulting from a decoder based on straightforward realization of 2-bit Min-Sum algorithm. The savings in hardware resources in the proposed MMS decoder is primarily from the reduced number of inter-connects between the decoding nodes, simplified check node operation and hardware optimization of the variable node. Maintaining a higher quantization at the intrinsic message level helps to provide a higher BER performance. The proposed scheme can be adapted to appropriate combinations of intrinsic and extrinsic message quantization to suit particular applications. The 1152-bit decoder that has been implemented on FPGA can be used for WiMax applications. If desired, the BER performance can be improved further by using a different combination of intrinsic and extrinsic message quantization. This of course will require higher amount of hardware resources, but the resource requirement will still be much less than that of a straightforward implementation of Min-Sum.

References


[14] R. Zarubica, R. Hinton, S.G. Wilson, E.K. Hall, Efficient quantization schemes for hardware implementation. His research is mainly focused on implementing low complexity algorithms for decoding LDPC codes and investigating efficient architectures for hardware implementation. His research is mainly focused on implementing high performance LDPC decoders on reconfigurable devices.


[31] FRAZER flow


[38] IEEE Standard 802.16e, Air interface for fixed and mobile broadband wireless access systems. Amendment 2: Physical and medium access control layers for combined fixed and mobile operation in licensed bands, IEEE, December 2005.


