Extrinsic Information Memory Reduced Architecture for Non-Binary Turbo Decoder Implementation

Sook Min Park, Jaeyoung Kwak*
Samsung Electronics
Suwon, South Korea

Kwyro Lee
School of Electrical Engineering & Computer Science
KAIST
Daejeon, South Korea

Abstract—Two methods are presented that can substantially reduce the memory requirements of non-binary turbo decoders by efficient representation of the extrinsic information. In the case of the duo-binary turbo decoder employed by the IEEE 802.16e standard, the extrinsic information can be reduced by about 43%, which decreases the total decoder complexity by 18%. We also show that the proposed algorithm can be implemented by simple hardware architecture.

Keywords—Non-binary, duo-binary, turbo codes, extrinsic information, extrinsic memory

I. INTRODUCTION

A turbo code is said to be non-binary if each component encoder accepts multiple, say \( k \geq 2 \) input bits each time \([1][2]\). Non-binary turbo codes have many advantages over binary turbo codes such as higher throughput and excellent decoding convergence characteristics and hence are found in various practical applications. For example, duo-binary \((k=2)\) turbo codes have been selected as options in mobile WiMAX (IEEE 802.16e [3]) and European satellite network standards (DVB-RCS, ETSI EN 301 790 [4]). However, \( k \)-input non-binary turbo decoders require \((2^k-1)\) extrinsic informations and hence the extrinsic information memory size is \((2^k-1)/k\) times that of a binary turbo decoder [5].

Table 1 illustrates the chip area occupation rates based on the turbo decoder architecture introduced in [8] for a “constraint length \( K=4\), ‘rate \( R=1/2\), ‘packet length \( N=960\)” duo-binary Max-Log-MAP decoder employed by mobile WiMAX. We see that 43% of the total area is consumed for the extrinsic information. Moreover, the storage of extrinsic information into the memory is responsible for the largest portion of the total turbo decoder energy consumption. Consequently, the reduction of the extrinsic information memory requirement is of particular importance in non-binary turbo decoding.

In [9], an algorithm was introduced requiring only \( k \) extrinsic informations rather than \((2^k-1)\). However, this algorithm degrades the bit error rate (BER) performance by about 1dB. Another algorithm was introduced in [10] that can reduce unnecessary extrinsic information exchange by measuring each symbol’s contribution to the convergence of iterative process. This algorithm achieves about 20% extrinsic memory reduction without noticeable performance degradation. However, two sub-decoders are necessary in its decoder architecture.

In this paper, we propose two methods which are more efficient than these algorithms in reducing the extrinsic information memory size for non-binary turbo decoders. First we propose a bitwise approximation method for extrinsic information which approximates \((2^k-1)\) LLRs to bitwise \( k \) LLRs. The other proposed algorithm decreases the necessary bit description width of the extrinsic information by employing a pseudo floating point representation. The bit description width of the significand is reduced by taking only the most informative several bits. Moreover, we show it is possible to further reduce the overall bit width by using a single common exponent for the \((2^k-1)/k \) extrinsic informations without noticeable performance degradation. As a consequence, the memory reduction of the proposed algorithm tends to become more substantial with increasing \( k \). We demonstrate that the proposed algorithm reduces the extrinsic information memory by about 43% in the duo-binary case. Also we provide a simple hardware architecture with which we can implement the algorithm.

The rest of this paper is organized as follows. In Section 2, we describe very briefly the duo-binary turbo decoding algorithm considered throughout this paper. In Section 3, we propose the method that can substantially reduce the necessary bit description width for the extrinsic information. Then, in Section 4, we demonstrate the proposed algorithms can significantly reduce the memory requirement without appreciable performance degradation over AWGN channels. Finally, we draw conclusions in Section 5.

II. DUO-BINARY TURBO DECODING

Throughout this paper, we only consider duo-binary turbo decoders for AWGN channels. In particular, we shall consider the \( K=4, R=1/2, N=960 \), “iteration number \( I=8\), duo binary turbo code employed by the IEEE 802.16e for performance evaluations. However, the proposed algorithm can easily be
applied to any non-binary turbo decoders for more general types of channels with appropriate modifications. As described in [5] and [6], the basic structure and algorithm of duo-binary turbo decoders are the same as those of binary turbo decoders that are detailed in various places [1]. The major difference of duo-binary turbo decoding is in that a set of three log-likelihood ratios (LLRs), instead of a single LLR, must be exchanged between the component decoders. In the following, we shall briefly deal with the definitions and algorithms used in this paper following the notational conventions of [6] and [7].

First, the set \( L_i(u_n) = \{ L_i^A(u_n), L_i^B(u_n), L_i^C(u_n) \} \) of LLRs to be passed between component decoders are defined as follows:

\[
L_i^A(u_n) = \log \left( \frac{p(u_n = (-1, -1) | y)}{p(u_n = (-1, 1) | y)} \right) \tag{1}
\]

\[
L_i^B(u_n) = \log \left( \frac{p(u_n = (-1, -1) | y)}{p(u_n = (1, -1) | y)} \right) \tag{2}
\]

\[
L_i^C(u_n) = \log \left( \frac{p(u_n = (-1, -1) | y)}{p(u_n = (1, 1) | y)} \right) \tag{3}
\]

where \( y_n = (u_i^{(1)} n, u_i^{(2)} n) \) denotes the duo-binary input bit couples at time \( n \) and \( y \) the received symbol sequence after the channel.

Second, the branch metric \( \gamma_n(s', s) \) corresponding to the branch at time instant \( n \) between states \( s' \) and \( s \) can be represented as follows:

\[
\gamma_n(s', s) = C' p(u_n) \exp \left( -\frac{1}{2\sigma^2} \| y_n - c_n \|^2 \right) \tag{4}
\]

where \( C' \) is an irrelevant constant, \( \sigma \) the additive white Gaussian noise variance, \( c_n = (c_i^{(1)} n, c_i^{(2)} n, \ldots) \) the turbo encoder output as a result of the state transition from \( s' \) to \( s \), \( y_n = (y_i^{(1)} n, y_i^{(2)} n, \ldots) \) the corresponding output at the demodulator and \( p(u_n) \) denotes the a priori probability of the input bit couple \( u_n \), which is one of the four binary couple \((-1, -1), (1, -1), (-1, 1), (1, 1)\). The first two elements of \( c_n \) correspond to the systematic bits, namely, \( u_i^{(1)} n \) and \( u_i^{(2)} n \). With this branch metric, the forward and backward state metrics \( \alpha \) and \( \beta \) are calculated as usual in the following manner:

\[
\alpha_n(s) = \sum_{s'} \alpha_n(s') \gamma_n(s', s) \tag{5}
\]

\[
\beta_n(s') = \sum_s \beta_n(s) \gamma_n(s', s) \tag{6}
\]

The LLRs are then updated in a similar manner as in the usual binary turbo decoders, although it is necessary to update three LLRs instead of one. For example, \( L_i^A(u_n) \) defined in (1) can be computed from

\[
L_i^A(u_n) = \log \left( \frac{\sum_{s = (-1, -1)} \alpha_n(s') \gamma_n(s', s) \beta_n(s)}{\sum_{s = (-1, -1)} \alpha_n(s') \gamma_n(s', s) \beta_n(s)} \right) \tag{7}
\]

and \( L_i^A(u_n) \) can be computed, in sequence, from

\[
L_i^A(u_n) = L_i^B(u_n) - L_i^C(u_n) - \frac{2(y_n^{(1)})^2}{\sigma^2} \tag{8}
\]

where \( L_i^B(u_n) \) is the extrinsic information passed from the previous decoding module. The other two extrinsic informations \( L_i^B(u_n), L_i^C(u_n) \) can be obtained similarly from

\[
L_i^B(u_n) = L_i^B(u_n) - L_i^C(u_n) - \frac{2(y_n^{(1)})^2 + (y_n^{(2)})^2}{\sigma^2} \tag{10}
\]

We note that the extrinsic information, namely, the \( L_i \)'s need to be saved in memory for the next decoder iteration. This extrinsic memory size is determined by the block size, \( k \) and the bit description width of the extrinsic information. We note that duo-binary turbo codes need three extrinsic informations per 2 information bits, which increases extrinsic memory by 1.5 times compared to the binary case.

## III. EXTRINSIC INFORMATION MEMORY REDUCED ARCHITECTURE FOR NON-BINARY TURBO DECODER IMPLEMENTATION

Before we describe the proposed methods, we recall (8) – (10) which shows that \( L_i(u_n) \) is computed from \( L_i(u_n), L_i(u_n), \) and \( y_n \). Here, \( L_i(u_n) \) is the extrinsic information passed from the previous stage and hence is described in the same manner as \( L_i(u_n) \). In the case of \( L_i(u_n) \), the value is used once and doesn’t have to be stored. Consequently, we describe \( L_i(u_n) \) by a 10-bit fixed point number, which provides enough precision.

Like \( L_i(u_n) \), the received data \( y_n \) has to be stored for use throughout the iterative decoding procedure and hence needs to be described efficiently. To determine the required number of quantization levels for \( y_n \), we performed numerical simulations over the additive white Gaussian noise (AWGN) channel to obtain the results summarized in Figure 1-(a). We assumed that the receiver uses fixed point representations for both \( L_i(u_n) \) and \( y_n \) and that 4 more bits are used to describe \( L_i(u_n) \) in order to reduce the effect of \( L_i(u_n) \) precision. We note that the performance improvement is more or less saturated after 4-bit quantization for \( y_n \). Hence, we use 4-bit quantization for \( y_n \) in this paper. Next, with 4-bit quantization for \( y_n \) we performed numerical analysis with various bit description widths for \( L_i(u_n) \), which we summarize in Figure 1-(b). We note from the figure that the performance improvement is negligible when we increase the bit width to more than 7 bits. From this analysis, we choose 4-bit and 7-bit quantization for \( y_n \) and \( L_i(u_n) \) as the baseline representation.

![Graph](image-url)
A. Bitwise approximation for Extrinsic Information

Efficient without appreciable loss in system performance.

The probability of the input bit couple is the probability of bitwise 1 and the probability of bitwise 0 where a combination of the first two input bit couple is the don’t care value which can be either +1 or -1. Now, \( p(u) \) can be represented as follows:

\[
p(u_n = -1, X)) = p(u_n^0 = -1) \\
p(u_n = 1, X)) = p(u_n^0 = 1) \\
p(u_n = X, -1)) = p(u_n^1 = -1) \\
p(u_n = X, 1)) = p(u_n^1 = 1)
\]

where a combination of first two input bit couple is the probability of bitwise \( u_n^0 \) and a combination of the other two input bit couple is the probability of bitwise \( u_n^1 \). So the set \( L(u) = \{ L^A(u), L^B(u) \} \) of LLRs to be passed between component decoders can be defined as follows:

\[
L^A(u) = \log \left( \frac{p(u_n^0 = 1 | y)}{p(u_n^0 = -1 | y)} \right) \\
L^B(u) = \log \left( \frac{p(u_n^1 = 1 | y)}{p(u_n^1 = -1 | y)} \right)
\]

\( L^A(u) \) defined in (12) can be computed from

\[
L^A(u) = \log \left( \sum_{s=-1}^{s=1} \alpha_{s+1}(s') \gamma_{s}(s', s) \beta_{s}(s) \right)
\]

\( L^B(u) \) can be computed, in sequence, from

\[
L^B(u) = L^A(u) - \frac{2 \gamma_{s+1}}{\sigma^2}
\]

where \( L^B(u) \) is the extrinsic information passed from the previous decoding module. The other extrinsic information \( L^A(u) \) can be obtained similarly from

\[
L^A(u) = L^B(u) - \frac{2 \gamma_{s+1}}{\sigma^2}
\]

B. Pseudo Floating Point Representation for Extrinsic Information

To reduce the memory size for extrinsic information of a turbo decoder, we first investigate the characteristics of the extrinsic information in the decoding process. In general, the extrinsic information \( L_e \) starts from 0 and then tends to go away from 0, converging to some number as the iteration progresses. Moreover, its exact value does not affect the decoder performance significantly once its magnitude becomes large. From this observation, we propose to use a pseudo floating point representation for extrinsic information to greatly reduce the memory requirement.

To illustrate the proposed method of representation, consider, as an example, the case in which \( L^A_e = 25, L^B_e = -12, \) and \( L^C_e = 3 \). In the baseline system, the \( L_e \)’s are represented by 0011001, 1110100, and 0000011, respectively. We note that the \( L_e \)’s are represented by 7-bit strings and the negative number is represented by the 2’s complement. However, we note that only the most significant few digits and the magnitude of the extrinsic information play important roles in the decoding procedure. Consequently, for example, we may attempt to represent \( L^A_e, L^B_e, \) and \( L^C_e \) by (0110,01), (1010,10), and (0111,11) employing floating point representation concept. To illustrate how we recover the original 7-bit strings from these, consider (0110,01). We first note that 0110 starts with ‘0’. The 2-bit string 01 called the shift index (SI) shall indicate the number of 0’s to prefix, which will be used to determine the amount of shift. Now, we prefix “(= 01)” 0 to 0110 to make the 5-bit string 00110 and then append two 0’s to obtain a 7-bit string 0011000. Similarly, for (1010,10), we prefix two (=10) 1’s to 1010 and then append one 0 to obtain 1110100. We note that the recovered numbers may not be exactly the same as the original 7-bit strings.

We denote the significand of \( L^A_e \) by \( \hat{L}^X_e \) and the number of bits to be prefixed by \( SI_X \) for \( X = A, B \) or \( C \). Consequently, in the example, \( L^A_e = 0110 \) and \( SL_A = 01 \). We shall denote by \( W_{le} \) the bit width to describe the significand including the sign bit and by \( W_{SI} \) the bit width used to describe \( SI_e \). Consequently, in this example, \( W_{le} = 4, W_{SI} = 2 \). Here, we note that the representation is slightly different from ordinary floating point representations in the case of \( 3^0000011 \). With ordinary floating point representation, this number might be represented by (0110,100) which however is represented by (0011,11) to reduce \( W_{SI} \). At this point, to further reduce the required number of total bit description width, we choose the minimum \( SL_{min} \) of \( SL_A, SL_B, \) and \( SL_C \) and store only this number instead of the three. For this reason, the proposed algorithm becomes more efficient for larger \( k \) values. As the final step, we redefine \( \hat{L}^X_e \) by choosing \( SI_X = SL_{min} \).
The proposed algorithm can be implemented by a very simple architecture shown in Figure 2. At the top-most part of the hardware shown in Figure 2-(a), the shift index \( S_{L_e} \) is determined from the 7-bit string \( (L^A_0, [6], ..., L^A_6, [0]) \) by simple Ex-OR and else-if logics. We note that the Ex-OR and else-if determines the first position (from the most significant bit) of the digit that has a different value from \( L^A_6 \), which yields \( S_{L_e} \). Then, \( S_{\text{trans}} \) is determined to find the shift index of the barrel shifter. Finally, the barrel shifter shifts \( L^A_0, L^B_1, L^C_2 \) by \( (7-W_{Le}, S_{\text{trans}}) \), and generates \( L^A_0, L^B_1, L^C_2 \). The detransformer that recovers the 7-bit strings from \( L^A_0, L^B_1, L^C_2 \) and \( S_{\text{trans}} \) is described in Figure 2-(b). We note that the detransformer is much simpler than the transformer, requiring only the barrel shifter by \( (7-W_{Le}, S_{\text{trans}}) \). The synthesized gate counts for the transformer and the detransformer are 391 and 240, respectively, which are negligible compared to the total turbo decoder gate counts.

IV. RESULTS

A. Bitwise approximation for Extrinsic Information

As described in Section 3, the bitwise approximation method approximates \( 2^{k-1} \) LLRs to bitwise \( k \) LLRs. Comparing with the original duo-binary turbo decoder, we only save two extrinsic informations for next iteration. So, we can save the extrinsic memory by about 33% compared with the baseline system. The simulation results for the proposed method are depicted in Fig. 3., which shows about 0.5dB degradation at BER \( 10^{-4} \) by bitwise approximation. However 14.3% of the original duo-binary turbo decoder size can be reduced by the proposed method.

B. Pseudo Floating Point Representation for Extrinsic Information

In Section 3, we have chosen 4 and 2 for the values of \( W_{Le} \) and \( W_{SI} \) respectively, for illustration. However, these values may be changed for better memory efficiency. It is the purpose of this section to choose the optimum pair. We first note that the values of \( W_{Le} \) and \( W_{SI} \) cannot be chosen independently. The possible pairs are listed in Table 2 as well as the total bit width \( W_{Le} \cdot (2^k-1) + W_{SI} \) for two information bits.

In Section 3, we have chosen \( 4 \) and \( 2 \) for the values of \( W_{Le} \) and \( W_{SI} \) respectively, for illustration. However, these values may be changed for better memory efficiency. It is the purpose of this section to choose the optimum pair. We first note that the values of \( W_{Le} \) and \( W_{SI} \) cannot be chosen independently. The possible pairs are listed in Table 2 as well as the total bit width \( W_{Le} \cdot (2^k-1) + W_{SI} \) for two information bits. To determine the optimum pair among the pairs in Table 2, we simulated the proposed method over the AWGN channel. The simulation results obtained for different \( (W_{Le},W_{SI}) \) pairs are depicted in Fig. 4. There is negligible performance degradation until we decrease \( W_{Le} \) from 7 to 3, while the performance is appreciably worse for the \( W_{Le}=2 \) case. Since \( (W_{Le},W_{SI}) = (3,3) \) gives the smallest total bit description width without appreciable performance degradation, we propose to choose this pair which saves the extrinsic memory by 43% compared with the baseline system.

Even though we exclusively considered AWGN channels for our experiments.

In the case of \( k=3 \), the number of extrinsic information bits per symbol for the original architecture in [5] is 49 bits. This can be reduced to 24 bits with our \( L_e \) transformation method, decreasing the extrinsic memory size by 51%. In the case of \( k=4 \), the extrinsic memory is reduced by 54%. We can find our \( L_e \) transformation method is more efficient as \( k \) increases. If we adopt the proposed method, we can save 18% of the original duo-binary turbo decoder size.

V. CONCLUSIONS

In this paper, two methods and a hardware architecture are presented that can greatly reduce the extrinsic memory of a non-binary turbo decoder. In the duo-binary case over AWGN channels, we can reduce the necessary bit width of extrinsic information by about 43%, without significant performance degradation. The overall duo-binary turbo decoder complexity can thus be decreased by about 18% in the case of the IEEE 802.16e turbo decoder considered. The proposed methods are more effective when adopted for non-binary turbo codes with \( k>2 \). Even though we exclusively considered AWGN channels.
only, we expect similar levels of memory reduction with the proposed algorithm for a reasonably large variety of channels.

### Table 2. Required bit width for extrinsic information and \(SI\)

<table>
<thead>
<tr>
<th>Bit width per each Extrinsic info. ((W_{Le}))</th>
<th>Bit width of (SI_{min}(W_{Si}))</th>
<th>Total bit width for (L_e)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>3</td>
<td>9</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>12</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>14</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>17</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>19</td>
</tr>
<tr>
<td>7</td>
<td>0</td>
<td>21</td>
</tr>
</tbody>
</table>

Figure 4 BER with various \(W_{Le}\) for the proposed method

### REFERENCES


