Memory-Reduced Turbo Decoding Architecture Using NII Metric Compression

Syed kareem saheb,
Research scholar, Dept. of ECE,
ANU, GUNTUR, A.P, INDIA.

K. Ramamohan Reddy
Asst. professor, Dept. of ECE, ASIT,
GUDUR, A.P, INDIA.

Abstract—This brief proposes a new compression technique of next-iteration initialization metrics for relaxing the storage demands of turbo decoders. The proposed scheme stores only the range of state metrics as well as two indexes of the maximum and minimum values, while the previous compression methods have to store all of the state metrics for initializing the following iteration. We also present a hardware-friendly recovery strategy, which can be implemented by simple multiplexing networks. Compared to the previous work, as a result, the proposed compression method reduces the required storage bits by 30% while providing the acceptable error-correcting performance in practice.

Key Terms—Channel codes, communication systems, error-correction codes, memory compression, very-large-scale integration (VLSI) designs.

I. INTRODUCTION

The turbo code is one of the most attractive forward error correction codes, which can provide near-optimal bit error rates (BERs) of Shannon’s limit [1]. Due to the fascinating error-correcting performance, the turbo codes have been applied to various wireless communication systems [2]–[5]. For achieving a high decoding throughput, an aggressive puncturing on long turbo codes is normally defined at the recent wireless standards. The extreme case of 3GPP LTE-advanced specification, for example, necessitates a code length of 6144 bits and a code rate of 0.95 [2].

To minimize the performance loss in decoding of high-rate code words, the next-iteration initialization (NII) scheme is widely accepted for the initialization of backward recursions instead of the traditional dummy calculation method [6]–[9]. However, the conventional NII technique requires additional memories for storing all of the final backward states of the current iteration, which denote the starting confidence levels of the following iteration. If the sliding-window technique is used for practical realization, moreover, the number of NII metrics to be stored increases drastically according to the number of window boundaries [8]. To mitigate the memory overheads, the static compression scheme in [8] introduces a dedicated transfer function to encode NII metrics into 3 or 4 bits. More recent research presents a dynamic scaling factor for encoding of NII metrics [9]. However, the previous schemes still require a large amount of storage bits as all of the state metrics have to be collected after an individual compressing process.

To reduce the number of storage bits further, this brief presents a new algorithm for encoding and decoding of NII metrics. Without storing the individually compressed values of each state metric, the proposed algorithm generates much compact information related to the minimum and maximum values among state metrics. In addition, the novel recovery scheme minimizes the performance degradation caused by the proposed aggressive compression with negligible hardware costs. While providing an attractive error-correcting capability, as a result, the proposed technique reduces the NII memory by 83% and 33%, compared to the conventional NII scheme and the previous static encoding method, respectively.

II. PREVIOUS WORKS

A. Conventional Turbo Decoding Architecture

Fig. 1 describes the generalized turbo decoding architecture based on the soft-input soft-output (SISO) decoders [5]. The turbo decoder alternatively processes two decoding phases, i.e., in-order and interleaved phases.

---

Fig. 1. Generalized turbo decoding architecture.
In the figure, the input log-likelihood ratio (LLR) sequences of the systematic bits and parity bits are denoted as $\Lambda_s$ (or $\Lambda'_s$) and $\Lambda_p$ (or $\Lambda'_p$), respectively, where superscript $I$ denotes the sequences related to the interleaved phase. Based on the input LLRs and a priori

![Image](https://example.com/image.png)

Fig. 2. Sliding-window-based turbo decoding with NII technique.

information $\Lambda_a$ (or $\Lambda'_a$), a SISO decoder generates a posteriori information, i.e., the extrinsic information $\Lambda_e$ (or $\Lambda'_e$), which will be a priori information of the opposite phase after passing through an interleaver (or deinterleaver). As the two different phases are exclusive in time, only one SISO decoder is normally adopted in practice for realizing the time-interleaved process.

B. Sliding-Window Technique With NII Metric Compressions

The sliding-window technique is widely accepted for the recent turbo decoders to reduce the size of internal buffers [6]. Fig. 2 illustrates the decoding procedure for the $n$-bit codeword associated with sliding windows of $w$ bits. Based on the maximum a posteriori (MAP) decoding algorithm, each sliding window first computes state metrics recursively in forward direction by using the corresponding branch metrics. For the sake of simplicity, $k$ forward state metrics of the $i$th trellis step are defined as $a_i(0), a_i(1), \ldots, a_i(k-1)$. Then, as shown in Fig. 2, the backward recursion computes the extrinsic information of each trellis step as well as the next state metrics in backward direction. Similar to the forward state metrics, $k$ backward state metrics of the $i$th trellis step are represented as $\beta_i(0), \beta_i(1), \ldots, \beta_i(k-1)$. Before starting the backward recursion, it is important to properly initialize the starting confidence levels of each backward state. In the NII technique, as shown in Fig. 2, the final backward states of each window boundary are stored to be used for the starting points at the next backward recursion of the corresponding phase. Assuming that all of the state metrics are normalized by the zeroth state, i.e., $a_i(0) = \beta_i(0) = 0$, which is popularly applied to the turbo decoder for reducing the internal computing resolution, the conventional NII scheme requires the dedicated storage of $2 \times (k - 1) \times d \times n/w$ bits, where $d$ stands for the bit-width of a state metric. For the practical turbo decoder of LTE-advanced systems ($k = 8$), for example, a total of 32 256 bits have to be stored for realizing the conventional NII scheme, where $d$ and $w$ are assumed to be 12 and 32, respectively [8].

To reduce the storage demands caused by the NII scheme, the static encoding method is widely used in the recent turbo decoders [8].
III. PROPOSED NII METRIC COMPRESSION

A. Memory-Reduced NII Metric Compression

In the contemporary turbo decoder, the max-log-MAP decoding algorithm is widely adopted due to the simple max operations instead of complicated max-star operations in the MAP algorithm [7]. As the max-log-MAP decoding only focuses on the trellis path having the maximum reliability, it is necessary to determine the most reliable state at the initializing process of each sliding window. Unlike the previous works preserving each state metric value as much as possible, the proposed NII metric compression considers the range of state metrics denoted as $\alpha$, i.e., the difference between the maximum and minimum state values among the $w$th backward state metrics, $\beta_{\text{wd}}()$. It is possible to make the bit-width of $\alpha$ smaller than $d$, the bit-width of each state metric, as the saturation of ranges exceeding a certain value is acceptable without degrading the BER performance [9]. Based on the numerous simulations, only 8 bits are enough to represent $\alpha$, while each original state metric requires at least more than 12 bits to prevent overflows in the LTE-advanced systems [9]. To give a hint for the confidence levels of each state at the recovery process, in addition, we store the indexes of the maximum and minimum states, represented as $I^\text{MAX}$ and $I^\text{MIN}$, respectively. Conceptually, the proposed compression method tries to offer the precise information of the peak differences by sacrificing the accuracy of each state metrics, whereas the previous works only consider the approximations of each state metric. Simulation results, which are described in Section IV, show that the BER of the proposed work using accurate $\alpha$ is comparable to those of the previous NII metric encoding schemes. Hence, the proposed algorithm reduces the number of storage bits in effect without degrading the error-correcting capability. When the size of a sliding window is set to 32, for the case of 6144-bit turbo codes, our compression scheme uses only 5376 bits for NII information, which is 6 times less than the conventional algorithm.

Similar to the previous works, the proposed NII metric compression also requires additional computations for the encoding and decoding processes. To compensate the computational complexity overheads, it is important to reduce the number of comparisons, which are much more time-consuming than the encoding networks in Fig. 3(b). In the proposed compression, as exemplified in Fig. 4 dealing with eight backward states denoted as $\beta_{\text{wd}}()$, the number of comparisons can be greatly reduced by sharing the intermediate results. To find $\alpha$ efficiently, three basic modules are utilized in the proposed architecture: the MAX–MIN, MIN, and MAX modules. As detailed in Fig. 4, the MAX–MIN module finds both the minimum and maximum values between two inputs by utilizing one comparator and two multiplexors. Note that the simplified modules MIN and MAX, associated with one comparator and one multiplexor, are also introduced for performing MIN(A, B) and MAX(A, B), respectively. The SUB/CLIP unit computes the final output $\alpha$ with the reduced bit-width $d$. It is well known that $I^\text{MAX}$ and $I^\text{MIN}$ can be easily generated by collect-ing the prior comparison results [10]. Note that it is impossible to reduce the number of comparisons at the previous algorithm as all of the compressing processes are independent of each other. Since the maximum and minimum values are generated at the same time by taking into account all of the states, on the other hand, we can reduce the number of comparisons by utilizing MIN–MAX modules to share the comparators of each processing as shown in Fig. 4. In case of arbitrary turbo codes, Table I compares the proposed method to the previous works in terms of the required storage bits and the number of comparisons for generating compressed NII values. Our NII metric compression stores only $d+2\times\log_2 k$ bits per window boundary, where $d$ is the reduced bit-width of $\alpha$. Note that the storage demands of the proposed algorithm are proportional to $\log_2 k$, whereas the previous solutions are proportional to $k$, requiring a much larger memory size. As shown in Table I, moreover, the additional comparisons of the proposed work are also remarkably relaxed by more than 50%, compared to those of the static compressing method [8].

![Fig. 4. Cost-effective compressing process of eight state metrics based on the proposed NII metric compression.](image)

**TABLE I** COMPARISONS OF NII METRIC COMPRESSIONS

<table>
<thead>
<tr>
<th></th>
<th>Proposed</th>
<th>Conventional</th>
<th>3-bit encoding [8]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Storing bits per boundary of windows</td>
<td>$d^*+2\times\log_2 k$</td>
<td>$d^*+(k-1)$</td>
<td>$3\times(k-1)$</td>
</tr>
<tr>
<td>Total storing bits</td>
<td>$2\times(d^*+2\times\log_2 k)/(k/w)$</td>
<td>$2\times d^*+(k-1)\times n/w$</td>
<td>$6\times(k-1)\times n/w$</td>
</tr>
<tr>
<td>Number of comparisons</td>
<td>$\frac{3}{2} k-2$</td>
<td>0</td>
<td>$6\times(k-1)$</td>
</tr>
</tbody>
</table>

B. Hardware-Friendly Recovery Process

To recover the compressed NII information, the proposed algorithm necessitates a predefined reference. For the sake of simplicity, we set this reference value to zero by considering the normalization used in this work. Note that the reference
value can be any number as only the relative amounts between state metrics play an important role in the proposed algorithm. According to the reading-out range of state metrics $\Delta$, the proposed recovery scheme first generates a set of five confidence levels, which is denoted as $S_{c}(\Delta) = \{+\Delta_{c}, +\Delta_{c}/2, 0, -\Delta_{c}/2, -\Delta_{c}\}$. Including reference value, the three subsets of $S_{c}$, i.e., $S_{c}(1) = \{+\Delta_{c}, +\Delta_{c}/2, 0, -\Delta_{c}/2, -\Delta_{c}\}$ and $S_{c}(3) = \{+\Delta_{c}, +\Delta_{c}/2, 0, -\Delta_{c}/2, -\Delta_{c}\}$, are used in the recovery process as described in Fig. 5. Note that the stored $I_{M\text{AX}}$ and $I_{M\text{IN}}$ are considered for selecting the dedicated subset in order to fix $\beta_{c}(0)$ to zero. When neither of $I_{M\text{AX}}$ and $I_{M\text{IN}}$ is zero, as described in Fig. 5(a), all of the backward states have their initial confidence levels from the subset $S_{c}(1)$. More precisely, $\beta_{c}(I_{M\text{AX}})$ and $\beta_{c}(I_{M\text{IN}})$ are set to $+\Delta_{c}/2$ or $-\Delta_{c}/2$, respectively, while the rests of the backward states are initialized by zero. If $I_{M\text{AX}}$ is zero, which means that $\beta_{c}(0)$ has the maximum level of confidence, the initial val-ues are chosen from the second subset $S_{c}(2)$ as illustrated in Fig. 5(b). In this case, $\beta_{c}(I_{M\text{IN}})$ becomes $-\Delta$, where others are initialized by the middle point of $S_{c}(2), -\Delta_{c}/2$. Similarly, as shown in Fig. 5(c), the set $S_{c}(3)$ is used for the recovery process when $\beta_{c}(0)$ has the minimum confidence $=0$. Based on the proposed recovery scheme, the backward states can be initialized by offering an accurate range of state metrics, providing a reasonable starting condition to the backward recursion.

For the hardware-friendly recovery operation, moreover, $I_{M\text{IN}}$ the modified set $M_{c} = \{+\Delta_{c}, -\Delta_{c}/2, 0, -\Delta_{c}/2, -\Delta_{c}\}$ is applied and proposed to the simulation instead of $S_{c}$, which suffers from additional additions for sign conversions in the two’s complement arithmetic. Note that

| TABLE II |
| Implementation Results of NII Metric Compression Structures |

<table>
<thead>
<tr>
<th>Encoding</th>
<th>Gate count (n.n)</th>
<th>Latency (ps)</th>
<th>Energy consumption (pJ)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>3119</td>
<td>7602</td>
<td>2.45</td>
</tr>
<tr>
<td></td>
<td>786</td>
<td>396</td>
<td>5.73</td>
</tr>
<tr>
<td></td>
<td>832</td>
<td>5294</td>
<td>4.55</td>
</tr>
<tr>
<td></td>
<td>236</td>
<td>319</td>
<td>267</td>
</tr>
<tr>
<td></td>
<td>0.65</td>
<td>4.12</td>
<td>0.92</td>
</tr>
</tbody>
</table>

| Gate count for NII metrics (n.n) | 27955 | 41932 | 59904 |

| TABLE III |
| Complexity Comparisons of 4-Parallel Radix-4 SISO Decoders Having Different Backward Initialization Schemes |

<table>
<thead>
<tr>
<th></th>
<th>Proposed work</th>
<th>Conventional NII metric storing</th>
<th>Conventional dummy calculation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total gates (ratio)</td>
<td>376.4 K (1.00)</td>
<td>499.4 K (1.33)</td>
<td>371.1 K (0.99)</td>
</tr>
<tr>
<td>Forward recursion</td>
<td>39.8 K</td>
<td>39.8 K</td>
<td>39.8 K</td>
</tr>
<tr>
<td>Backward recursion</td>
<td>39 K</td>
<td>39 K</td>
<td>39 K</td>
</tr>
<tr>
<td>Extrinsic data calculation</td>
<td>8.2 K</td>
<td>8.2 K</td>
<td>8.2 K</td>
</tr>
<tr>
<td>NII metric compression</td>
<td>N. A.</td>
<td>N. A.</td>
<td>38.7 K</td>
</tr>
<tr>
<td>NII Memory</td>
<td>16 K</td>
<td>N. A.</td>
<td>N. A.</td>
</tr>
<tr>
<td>Other Memories</td>
<td>120 K</td>
<td>120 K</td>
<td>120 K</td>
</tr>
</tbody>
</table>

IV. IMPLEMENTATION RESULTS AND ANALYSIS

Table II describes the implementation results of the NII metric encoding and decoding circuits for LTE systems [2]. For fair comparisons, all of the architectures are based on the 12-bit resolution of NII metrics and equally synthesized at the speed of 500 MHz in a 28-nm CMOS process. Note that we store only 14 bits for each window boundary by limiting the bit-width of $x$ to 8 in the proposed algorithm. Compared to the previous works, the proposed work remarkably relaxes the hardware complexity as well as the energy consumption by reducing the number of comparisons. Due to the aggressive compression, moreover, the proposed scheme stores much fewer bits for NII metrics than other works, resulting to area-efficient realization. The latency of the proposed scheme is slightly longer than that of the static encoding method [8]; however, it is still acceptable as the processes are only activated at the window boundaries, which can be isolated from the critical path by using the additional cycle [9]. Hence, the proposed work provides a memory-reduced SISO decoder while maintaining the data rates.
Based on the existing turbo decoders in [3], [7], and [8], we also developed 4-parallel radix-4 SISO decoders [7] using three different initializing algorithms for backward recursions: the conventional dummy operation, the conventional NII metric storing, and the proposed NII metric compression. Each SISO decoder deals with 4-bit LLR inputs, 12-bit state metrics, and 10-bit extrinsic values. For the longest 6144-bit codes, the window size and the maximum number of iterations are equally set to 32 and 6, respectively. To improve the BER performance, we apply the scaled max-log-MAP algorithm with an extrinsic scaling factor of 0.75. Table III compares the number of gate counts for realizing different SISO decoders, which are synthesized at the speed of 500 MHz in a 28-nm CMOS process.

By reducing both the storage demands and compression complexity, as shown in Table III, the overall complexity of the proposed SISO decoder is relaxed by 33% compared to that of the conventional SISO decoder storing the raw NII metric information. The SISO decoder based on the dummy recursion requires comparable hardware complexity by eliminating the storage demands of NII information; however, the dummy-recursion-based architecture requires a much large window size than the NII metric storing architecture to obtain the same error-correcting performance, especially for the high-rate codewords [6]. As the complexity of buffer memories drastically increases according to the window size, the usage of compressed NII information is more useful for the cost-effective turbo decoder associated with the practical window size [7].

For the 6144-bit code in the LTE system [2], Fig. 6 illustrates the BER performances of various NII metric compressions associated with the 4-parallel SISO decoder based on the sliding windows of 32 bits. For the low code rate of 1/3, as shown in the figure, the proposed algorithm degrades the performance by only 0.06 dB compared to the conventional NII technique for achieving the BER of $10^{-6}$. In case of high-rate code word, the proposed NII metric compression still achieves an attractive BER performance, whereas the previous 3-bit static encoding [9] and the dummy calculation cannot provide the acceptable performances.

Note that the proposed algorithm also reaches the similar error-floor levels to the conventional NII metric storing method without any compression. The dynamic scaling scheme [9] may provide slightly better BER performances than the proposed work. However, it uses considerable resources by utilizing more storage bits as described in Table II. Hence, the proposed algorithm allows the area-efficient decoder while providing the acceptable error-correcting capability.

V. CONCLUSION

By storing the precise ranges rather than the individually compressed metrics, the proposed algorithm remarkably reduces the size of NII metric memory while achieving an attractive error-correcting capability. Compared to the previous algorithms, moreover, the proposed compressing technique allows a low computational complexity in encoding and decoding of NII metrics, leading to the cost-effective turbo decoder architecture.

REFERENCES

BIOGRAPHY

Mr. Syed Kareem Saheb, Associate Professor, HOD, ECE
Dept of Audisankara Institute of Technology, Gudur. He is a
Research Scholar in Dept of Communication and
Engineering at ANU College of Engineering and
Technology, Guntur, A.p

K. Ramamohan Reddy, Asst. Professor, Dept of ECE,
Audisankara Institute of Technology, Gudur.

K. Nagendra Kumar, Asst. Professor,
Dept of ECE, Audisankara Institute of Technology, Gudur.

A. Srihitha, she did his bachelor of Technology in ECE
at brahmaiah college, Nellore, Andhra pradesh, India
and doing Master of Technology in VLSI Design at
Audisankara institute of Technology, Gudur,
Andhra pradesh, India