Memory-Reduced Turbo Decoding Architecture Using NII Metric Compression

Youngjoo Lee, *Member, IEEE*, Meng Li, and Liesbet Van der Perre

***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.**

***Index 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 codewords, 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.

Manuscript received February 27, 2015; revised June 26, 2015; accepted September 5, 2015. Date of publication September 28, 2015; date of current version January 28, 2016. The presented research has been conducted by the research grant of Kwangwoon University in 2015. This brief was recommended by Associate Editor M. S.-W. Chen.

Y. Lee is with the Department of Electronic Engineering, Kwangwoon University, Seoul 01897, Korea (e-mail: yjlee@kw.ac.kr).

M. Li and L. Van der Perre are with the Interuniversity Microelectronic Center (IMEC) vzw, 3001 Leuven, Belgium (e-mail: meng.li@imec.be).

Digital Object Identifier 10.1109/TCSII.2015.2483361

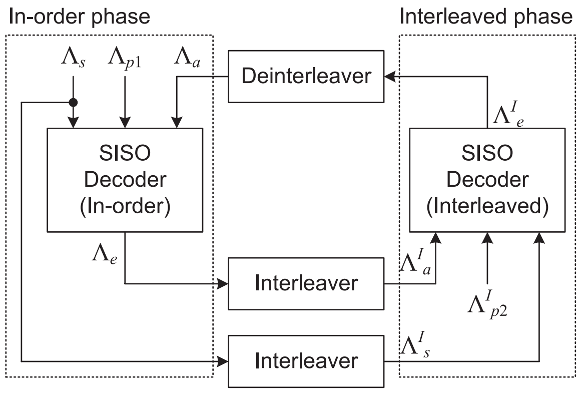


Fig. 1. Generalized turbo decoding architecture.

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.

The rest of this brief is organized as follows. Section II describes the previous works, and Section III explains the proposed NII metric compression technique. The simulation results are shown and compared to the previous algorithms in Section IV. Finally, the conclusion is in Section V.

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. In the figure, the input log-likelihood ratio (LLR) sequences of the systematic bits and parity bits are denoted as Λ_s (or Λ_s^I) and Λ_{p1} (or Λ_{p2}^I), respectively, where superscript I denotes the sequences related to the interleaved phase. Based on the input LLRs and *a priori*

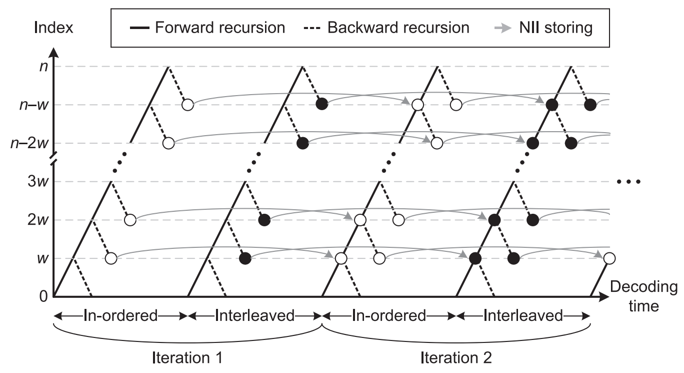


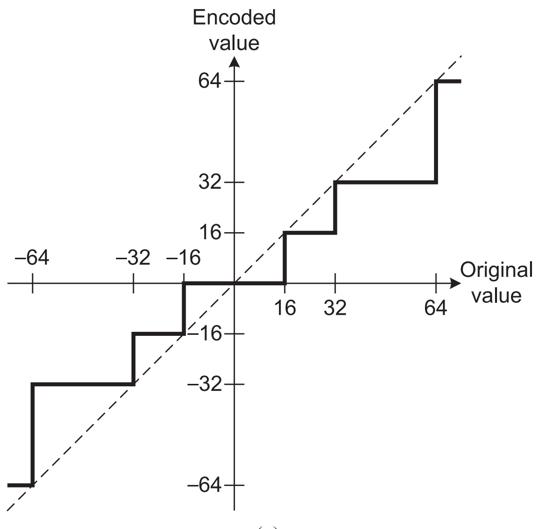
Fig. 2. Sliding-window-based turbo decoding with NII technique.

information Λ_a (or Λ_a^I), a SISO decoder generates *a posteriori* information, i.e., the extrinsic information Λ_e (or Λ_e^I), 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.

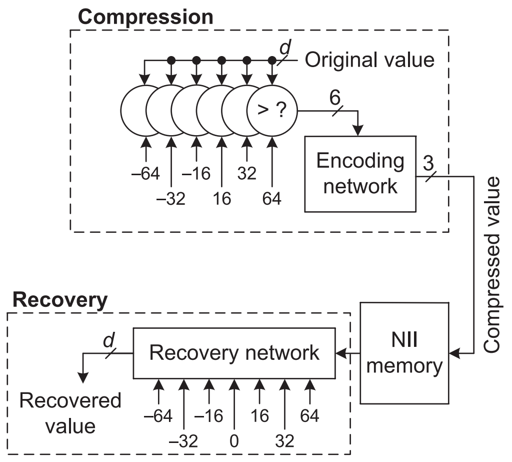
B. Sliding-Window Technique With NII Metric Compressions

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 ith trellis step are defined as $\alpha_i(0), \alpha_i(1), \ldots, \alpha_i(k-1)$. 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 ith 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., $\alpha_i(0) = \beta_i(0) = 0$, 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]. In the algorithm, the dedicated transfer function is introduced to restrict NII metrics to the power of twos. Fig. 3(a) shows the 3-bit encoding transfer function, which maps each input NII metric onto one of the seven encoded values [8].



(a)



(b)

Fig. 3. (a) Transfer function of the previous 3-bit static NII metric compression [8]. (b) Encoding and decoding process.

Compared to the conventional NII scheme, the previous 3-bit static compression reduces the number of storage bits to $6 \times (k-1) \times n/w$. As each NII metric has to be compared with the reference values, however, the work of [8] additionally requires numerous comparisons. As shown in Fig. 3(b), more precisely, each state metric is compared with six reference values, and the comparison results are used for generating the 3-bit compressed metric at the encoding network. The 3-bit saved NII metrics are read from the NII memory and fed into the recovery network to reconstruct the starting values of backward recursions. Similar to [8], the recent work in [9] presents a dynamic scaling algorithm for each NII metric; however, it requires more storage bits as well as the hardware resources to provide an acceptable BER performance, which consumes more power than [8].

Note that the previous works are based on the independent compression of each state metric. As all of the $k-1$ original state metrics have to be encoded and stored to the NII memory, the reduction in memory bits is limited by nature. The proposed algorithm, which is explained in the following section, considers the range of NII metric values to make a more compact format of storing data.

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 Δ_x, i.e., the difference between the maximum and minimum state values among the wxth backward state metrics, $\beta_{wx}(\cdot)$. It is possible to make the bit-width of Δ_x 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 Δ_x, 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_x^{MAX} and I_x^{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 Δ_x 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 $\beta_{wx}(\cdot)$, the number of comparisons can be greatly reduced by sharing the intermediate results. To find Δ_x 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 $\text{MIN}(A, B)$ and $\text{MAX}(A, B)$, respectively. The SUB/CLIP unit computes the final output Δ_x with the reduced bit-width d'. It is well known that I_x^{MAX} and I_x^{MIN} can be easily generated by collecting 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 Δ_x. 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].

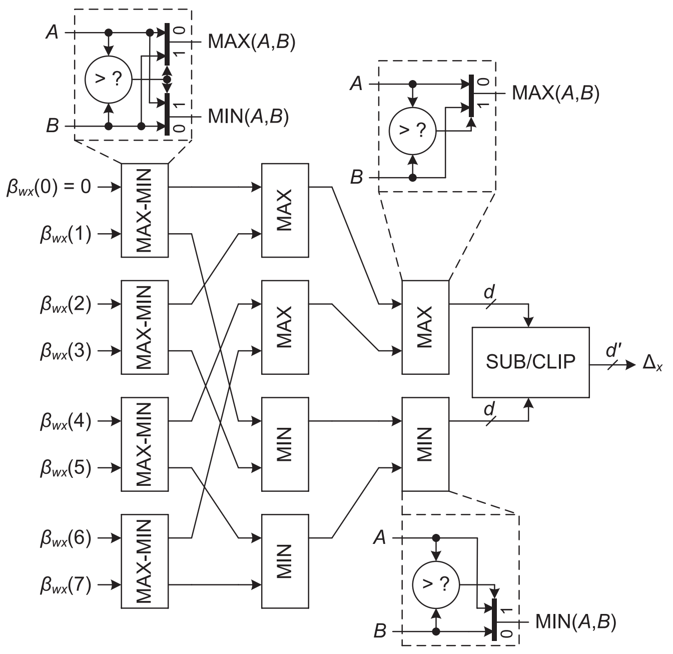


Fig. 4. Cost-effective compressing process of eight state metrics based on the proposed NII metric compression.

TABLE I
Comparisons of NII Metric Compressions

	Proposed	Conventional	3-bit encoding [8]
Storing bits per boundary of windows	d'+2×log$_2k$	d×(k–1)	3×(k–1)
Total storing bits	2×(d'+2×log$_2k$)×n/w	2×d×(k–1)×n/w	6×(k–1)×n/w
Number of comparisons	$\frac{3}{2}k-2$	0	6×(k–1)

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

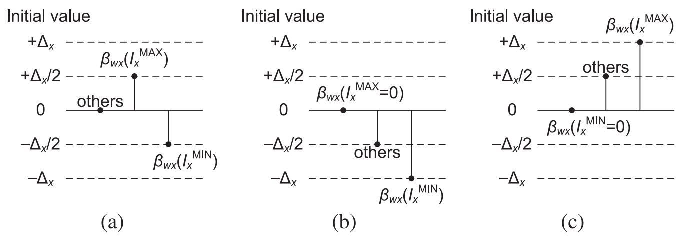

Initial value
$+\Delta_x$
$+\Delta_x/2$
0
$-\Delta_x/2$
$-\Delta_x$
$\beta_{wx}(I_x^{\mathrm{MAX}})$
others
$\beta_{wx}(I_x^{\mathrm{MIN}})$
$\beta_{wx}(I_x^{\mathrm{MAX}}=0)$
$\beta_{wx}(I_x^{\mathrm{MIN}}=0)$
(a)
(b)
(c)

Fig. 5. Proposed recovery strategy using the subsets (a) $S_x(1)$, (b) $S_x(2)$, and (c) $S_x(3)$.

TABLE II
IMPLEMENTATION RESULTS OF NII METRIC COMPRESSION STRUCTURES

		Proposed	[8]	[9]
Encoding	Gate count	3119	7002	5607
Encoding	Latency	786 ps	396 ps	1209 ps
Encoding	Energy consumption	2.45 pJ	5.73 pJ	4.55 pJ
Decoding	Gate count	832	5294	1159
Decoding	Latency	236 ps	319 ps	267 ps
Decoding	Energy consumption	0.65 pJ	4.12 pJ	0.92 pJ
Gate count for NII memory (*w*=32)		27955	41932	59904

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 Δ_x, the proposed recovery scheme first generates a set of five confidence levels, which is denoted as $S_x = \{+\Delta_x, +\Delta_x/2, 0, -\Delta_x/2, -\Delta_x\}$. Including reference value 0, three subsets of S_x, i.e., $S_x(1) = \{+\Delta_x/2, 0, -\Delta_x/2\}$, $S_x(2) = \{0, -\Delta_x/2, -\Delta_x\}$, and $S_x(3) = \{+\Delta_x, +\Delta_x/2, 0\}$, are used in the recovery process as described in Fig. 5. Note that the stored I_x^{MAX} and I_x^{MIN} are considered for selecting the dedicated subset in order to fix $\beta_{wx}(0)$ to zero. When neither of I_x^{MAX} and I_x^{MIN} is zero, as described in Fig. 5(a), all of the backward states have their initial confidence levels from the subset $S_x(1)$. More precisely, $\beta_{wx}(I_x^{\mathrm{MAX}})$ and $\beta_{wx}(I_x^{\mathrm{MIN}})$ are set to $+\Delta_x/2$ or $-\Delta_x/2$, respectively, while the rests of the backward states are initialized by zero. If I_x^{MAX} is zero, which means that $\beta_{wx}(0)$ has the maximum level of confidence, the initial values are chosen from the second subset $S_x(2)$ as illustrated in Fig. 5(b). In this case, $\beta_{wx}(I_x^{\mathrm{MIN}})$ becomes $-\Delta_x$, where others are initialized by the middle point of $S_x(2), -\Delta_x/2$. Similarly, as shown in Fig. 5(c), the set $S_x(3)$ is used for the recovery process when $\beta_{wx}(0)$ has the minimum confidence level ($I_x^{\mathrm{MIN}} = 0$). 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, the modified set $M_x = \{+\Delta_x, \lfloor +\Delta_x/2 \rfloor, 0, \lceil -\Delta_x/2 \rceil - 1, -\Delta_x - 1\}$ is proposed and applied to the simulation instead of S_x, which suffers from additional additions for sign conversions in the two's complement arithmetic. Note that three subsets are also accordingly changed to $M_x(1)$, $M_x(2)$, and $M_x(3)$. By introducing the modified set M_x, the additional additions are totally removed due to the simple inverting conversion similar to the ones' complement arithmetic. The recovered range of state metrics may be increased by one for the case of $I_x^{\mathrm{MAX}} = 0$; however, the effects on the BER performance are negligible in our simulation. Hence, the proposed recovery scheme based on the new set M_x can be realized by utilizing only a few inverters and multiplexors, which becomes a cost-effective solution compared to the previous works requiring numerous multiplexing networks [8] or multiple scaling units [9] for the recovery process.

TABLE III
COMPLEXITY COMPARISONS OF 4-PARALLEL RADIX-4 SISO DECODERS HAVING DIFFERENT BACKWARD INITIALIZATION SCHEMES

	Proposed work	Conventional NII metric storing	Conventional dummy calculation
Total gates (ratio)	376.4 K (1.00)	499.4 K (1.33)	371.1 K (0.99)
Forward recursion	39.8 K	39.8 K	39.8 K
Backward recursion	39 K	39 K	39 K
Branch metric calculation	8.2 K	8.2 K	8.2 K
Extrinsic data calculation	125.4 K	125.4 K	125.4 K
Dummy recursion	N. A.	N. A.	38.7 K
NII metric compression	16 K	N. A.	N. A.
NII Memory	28 K	167 K	N. A.
Other Memories	120 K	120 K	120 K

IV. IMPLEMENTATION RESULTS AND ANALYSIS

Table II describes the implementation results of the NII metric encoding and decoding circuitries 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 Δ_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

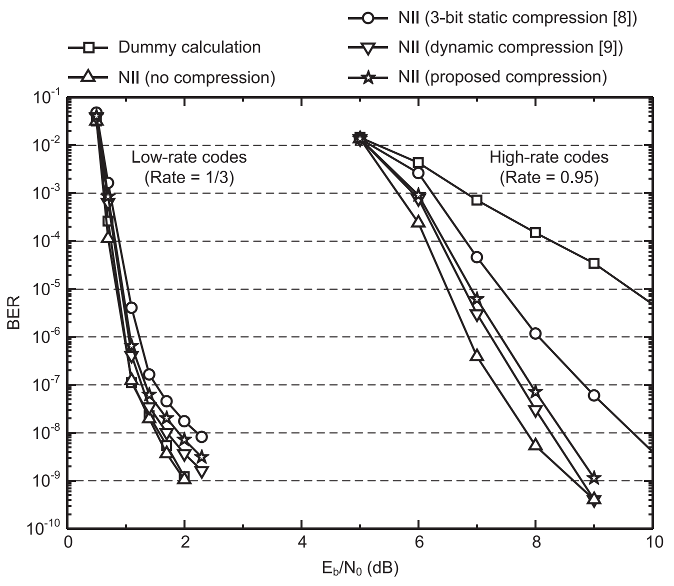


Fig. 6. BER curves of various backward initializing algorithms for different 6144-bit codes.

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 the high-rate codeword, 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

A new NII metric storing scheme has been proposed for reducing the memory demands of turbo decoders. 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

[1] C. Berrou, A. Glavieux, and P. Thitimajshima, “Near Shannon limit error correcting coding and decoding: Turbo codes,” in *Proc. IEEE Int. Conf. Commun.*, 1993, pp. 1064–1070.

[2] *Multiplexing and Channel Coding (Release 11)*, 3GPP TS 36.212 v11.3.0, Jun. 2013.

[3] G. Wang *et al.*, “Parallel interleaver design for a high throughput HSPA+/LTE multi-standard turbo decoder,” *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 61, no. 5, pp. 1376–1389, May 2014.

[4] R. Shrestha and R. P. Paily, “High-throughput turbo decoder with parallel architecture for LTE wireless communication standards,” *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 61, no. 9, pp. 2699–2710, Sep. 2014.

[5] L. Li, R. G. Maunder, B. M. Al-Hashimi, and L. Hanzo, “A low-complexity turbo decoder architecture for energy-efficient wireless sensor networks,” *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vo1. 21, no. 1, pp. 14–22, Jan. 2013.

[6] C. Benkeser, C. Roth, and Q. Huang, “Turbo decoder design for high code rates,” in *Proc. IEEE Int. Conf. VLSI-SoC*, 2012, pp. 71–75.

[7] C. Roth, S. Belfanti, C. Benkeser, and Q. Huang, “Efficient parallel turbo-decoding for high-throughput wireless systems,” *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 61, no. 6, pp. 1824–1835, Jun. 2014.

[8] J.-H. Kim and I.-C. Park, “Double-binary circular turbo decoding based on border metric encoding,” *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 55, no. 1, pp. 79–83, Jan. 2008.

[9] I. Yoo, B. Kim, and I.-C. Park, “Memory-optimized hybrid decoding method for multi-rate turbo codes,” in *Proc. IEEE Veh. Technol. Conf. Spring*, 2013, pp. 1–5.

[10] Y. Lee, B. Kim, J. Jung, and I.-C. Park, “Low-complexity tree architecture for finding the first two minima,” *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 62, no. 1, pp. 61–64, Jan. 2015.