International Journal of Research
Available at https://edupediapublications.org/journals

e-ISSN: 2348-6848
p-ISSN: 2348-795X
Volume 05 Issue 16
June 2018

Memory-Reduced Turbo Decoding Architecture Using NII Metric Compression

Dr.Ch.Ravi Kumar & C.Siva Kumar

[1]Prof & HOD
Department of ECE
Prakasam Engineering College
Kandukur (m), Prakasam(dt),A.P,India.
kumarece0@gmail.com

[2] M.Tech scholar
Department of ECE
Prakasam Engineering College
Kandukur (m), Prakasam(dt),A.P,India
sivakumarchalla123@gmail.com

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 while providing the acceptable error-correcting performance in practice.*

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. Due to the fascinating error-correcting performance, the turbo codes have been applied to various wireless communication systems. 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. 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 .

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. To mitigate the memory overheads, the static compression scheme 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. 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.

II EXISTING SYSTEM

A. Conventional Turbo Decoding Architecture

Fig. 1 describes the generalized turbo decoding architecture based on the soft-input soft-output (SISO) decoders. 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 Λs (or ΛI s) and Λp1 (or ΛI p2), respectively, where superscript I denotes the sequences related to the interleaved phase. Based on the input LLRs and a priori information Λa (or ΛI a), a SISO decoder generates a posteriori information, i.e., the extrinsic information Λe (or ΛI e), which will be a priori information of the opposite phase after passing through an inter leaver (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.

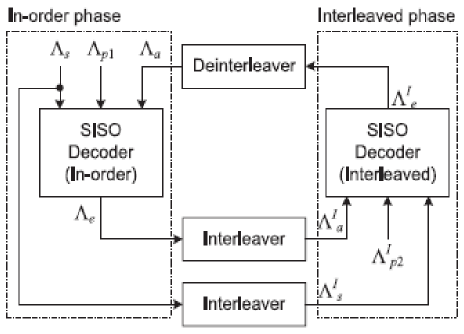


Fig. 1. Generalized turbo decoding architecture

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. 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 αi(0), αi(1), . . . , αi(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 βi(0), βi(1), . . . , βi(k − 1). Before starting the backward recursion, it is important to properly initialize the starting confidence levels of each backward state.

To reduce the storage demands caused by the NII scheme, the static encoding method is widely used in the recent turbo decoders. 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.

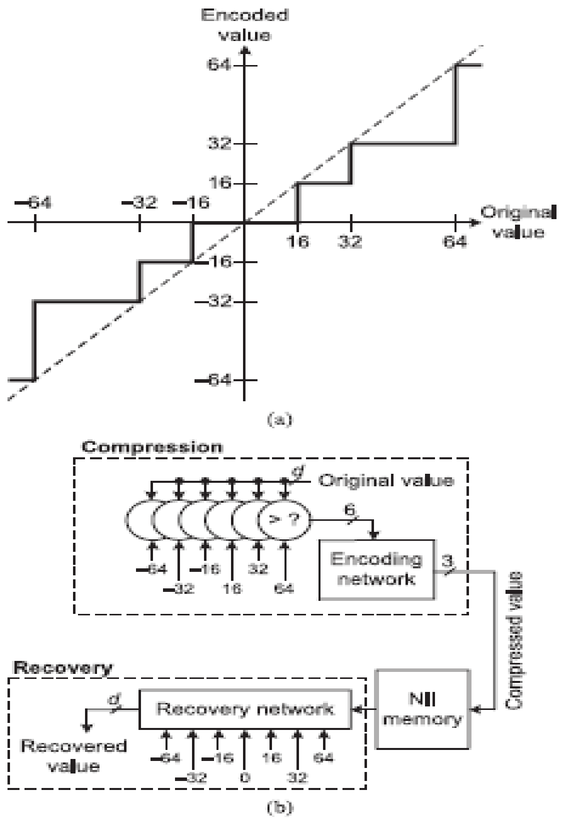


Fig. 3. (a) Transfer function of the previous 3-bit static NII metric compression (b) Encoding and decoding process.

Compared to the conventional NII scheme, the previous 3-bit static compression reduces the number of storage bits to 6 × (k − 1) × n/w. As each NII metric has to be compared with the reference values, however, the work of 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, the recent work in 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. 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 SYSTEM

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. 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 Δx, i.e., the difference between the maximum and minimum state values among the wxth backward state metrics, βwx(·). It is possible to make the bit-width of Δx 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 .

Based on the numerous simulations, only 8 bits are enough to represent Δx, while each original state metric requires at least more than 12 bits to prevent overflows in the LTE-advanced systems. 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 IMAX x and IMIN x , 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. show that the BER of the proposed work using accurate Δx 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 βwx(·), the number of comparisons can be greatly reduced by sharing the intermediate results. To find Δx 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 Δx with the reduced bit-width d. It is well known that IMAX x and IMIN x can be easily generated by collecting the prior comparison results . 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. 1. In case of arbitrary turbo codes,

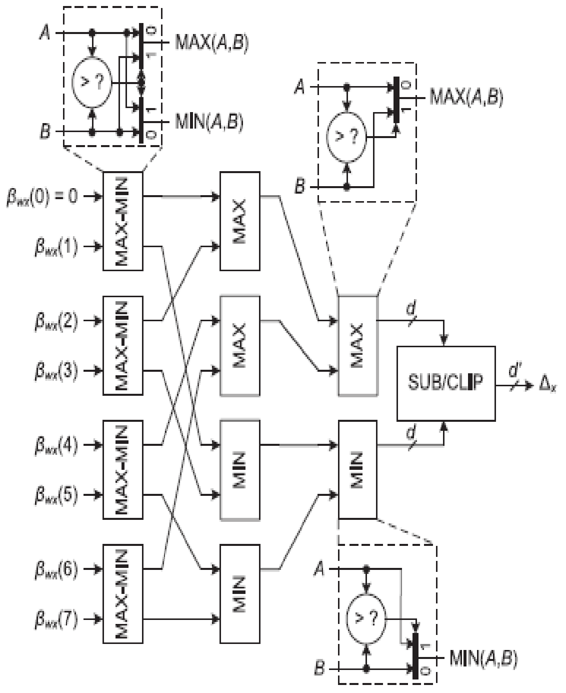
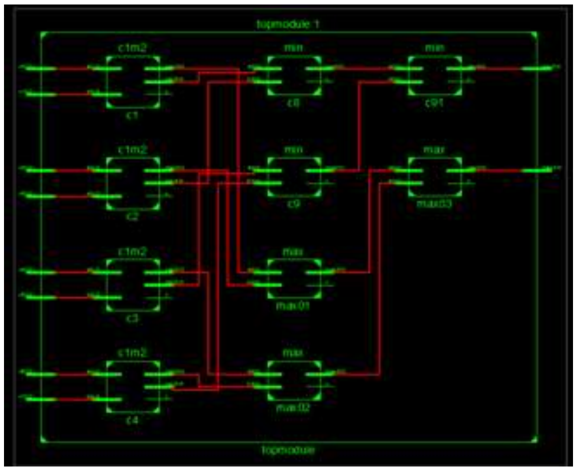


Fig. 1. Cost-effective compressing process of eight state metrics based on the proposed NII metric compression.

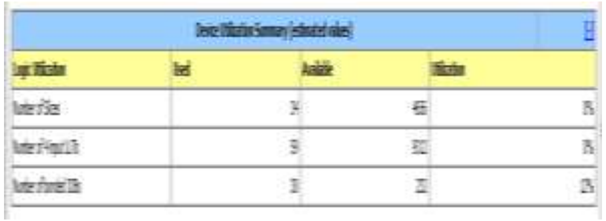
IV RESULTS

RTL schematic:



Technology schematics:

Design summary:

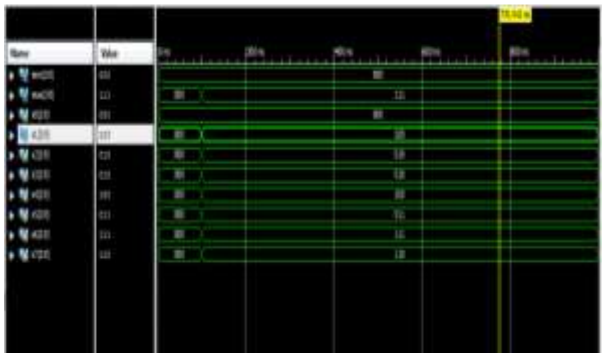


Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of Slices	[illegible]	[illegible]	[illegible]
Number of 4 input LUTs	[illegible]	[illegible]	[illegible]
Number of bonded IOBs	[illegible]	[illegible]	[illegible]

Timing report:

```
LUT4:I0->O       1  0.704  0.499  c8/c1_SW0 (N22)
LUT4:I1->O       2  0.704  0.622  c8/c1 (c8/c)
LUT3:I0->O       3  0.704  0.706  c8/m1/y<1>1 (min11<1>)
LUT4:I0->O       1  0.704  0.595  c91/m1/y<0>51_SW0 (N36)
LUT4:I0->O       1  0.704  0.455  c91/m1/y<0>51 (c91/m1/y<0>51)
LUT4:I2->O       1  0.704  0.420  c91/m1/y<0>84 (min_0_OBUF)
OBUF:I->O           3.272         min_0_OBUF (min<0>)
----------------------------------------
Total              16.668ns (10.826ns logic, 5.842ns route)
                          (65.0% logic, 35.0% route)
```

Simulation results:



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 lowcomplexity 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 turbodecoding for high-throughput wireless systems," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 61, no. 6, pp. 1824–1835, Jun. 2014.