Indonesian Journal of Electrical Engineering and Computer Science
Vol. 24, No. 2, November 2021, pp. 896∼903
ISSN: 2502-4752, DOI: 10.11591/ijeecs.v24.i2.pp896-903

Combining serial and parallel decoding for turbo codes

Mohammed AlMahamdy, Naser Al-Falahy
Department of Electrical Engineering, University of Anbar, Ramadi, Iraq

Article Info

Article history:

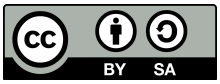
Received Apr 9, 2021
Revised Aug 26, 2021
Accepted Aug 31, 2021

Keywords:

Channel coding
Error correcting codes
Parallel decoding
Turbo codes

ABSTRACT

Reducing the decoding latency of the turbo codes is important to real-time applications. Conventionally, the decoding of the turbo codes (TC) runs in serial fashion, which means only one of the constituent soft decoders runs at a time. Parallel decoding (PD) refers to running the soft decoders in parallel. Although it delivers the output faster (compared to the serial decoding (SD)), it affects the bit- and frame-error rates. This paper proposes a decoding procedure that combines both PD and SD. It bridges the two decoding modes to determine the best combination scheme to achieve the required level of performance at an acceptable decoding latency. Presented results show how this procedure can mitigate the performance degradation at a slight increase in the decoding latency.



Corresponding Author:

Mohammed AlMahamdy
Department of Electrical Engineering
University of Anbar, Ramadi, Iraq
Email: mohammed.almahamdy@uoanbar.edu.iq

1. INTRODUCTION

Because of their outstanding performance, turbo codes (TC) [1] have been adopted in many modern communication standards, like 3G and 4G [2]. Improving their performance and optimizing their operation have been the subjects of many fields within the research community. Since the inception of the turbo code, many advances have been made in terms of the performance, and the decoding latency. A brief explanation of the turbo code is needed before we discuss the proposal.

The conventional encoder of the turbo code [1], illustrated in Figure 1 (a), comprises two half-rate recursive systematic convolutional encoders (RSC), and an interleaver (π). The input to the first encoder is the binary data (M). The input to the other one is $\tilde{M}$, which is the permutation of M. The output turbo code is the multiplexing of these three segments: M,$U^{(1)}$, and ,$U^{(2)}$ (where $U^{(i)}$ is the parity bits from the encoder i). The parity bits can be punctured to reduce the transmission rate. Termination bits can be added to the transmission. So, if the length of M is n bits, the transmission rate of the punctured turbo codes becomes $r = n/(2n + \tau)$, where τ is the number of the bits associated for termination.

Turbo codes are well recognized for their outstanding performance. The performance is mainly derived from the iterative progression of decoding virtually uncorrelated signals that are collaborated to produce an output. Conventional turbo decoders comprise of two soft-input-soft-output (SISO) decoders, Figure 1 (b). The received turbo code is restructured into the original components: the systematic message bits $\mathbf{Y}$, and the parity bits from the first and second RSC, $\mathbf{Q}^{(1)}$ and $\mathbf{Q}^{(2)}$ respectively.

The SISO decoder runs the Bahl-Cocke-Jelinek-Raviv (BCJR) algorithm, which generates a log-likelihood ratio (LLR) L_j for the information bit m_j (the j^{th} bit in $\mathbf{M}$) as in (1), where $\mathbf{w}_j$ is the received codeword for m_j. This Maximum A Posteriori algorithm maximizes either of the probabilities $P(m_j|\mathbf{w}_j)$ of

(1). As a result, the values of the LLRs range from $-\infty$ to ∞. The reliability of these LLRs is directly related to the magnitude of their absolute value.

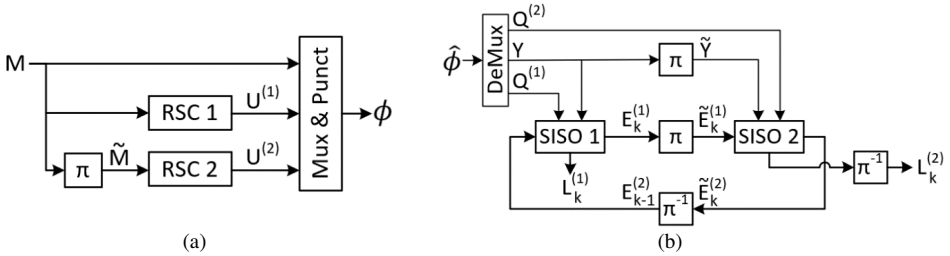


Figure 1. The conventional schemes of: (a) the turbo encoder, and (b) the turbo decoder (serial mode)

$$L_j = \log\left(\frac{P(m_j = +1|\mathbf{w}_j)}{P(m_j = -1|\mathbf{w}_j)}\right) \tag{1}$$

Extrinsic information $\mathbf{E}_k^{(i)}$ for the i^{th} SISO at the k^{th} iteration is derived from these LLRs as in (2) and (3). A SISO decoder uses these values as a priori to decode the channel input. Each SISO runs once per an iteration. The iterative decoding loop ends when reaching: either the maximum number of iterations, or a certain level of reliability for the output. The latter is detected by a rule to facilitate the decoding by skipping "unnecessary" iterations. Finally, the sign of the generated LLRs is used as the hard decision to deliver the binary bits as output.

$$\mathbf{E}_k^{(1)} = \mathbf{L}_k^{(1)} - \mathbf{Y} - \mathbf{E}_{k-1}^{(2)} \tag{2}$$

$$\mathbf{E}_k^{(2)} = \mathbf{L}_k^{(2)} - \mathbf{Y} - \mathbf{E}_k^{(1)} \tag{3}$$

2. REDUCING THE DECODING LATENCY

Conventional Turbo decoding is performed through running the BCJR algorithm twice an iteration, and for several iterations. The BCJR algorithm itself requires extensive number of computations. Hence, the decoding latency represents a major concern for the operation in practical systems. Latency reduction has been the subject of many studies in the literature. It can generally be manifested in many ways including: reducing and/or simplifying the number of computations per iteration, skipping the less-likely computations per iteration, early-terminating the iterating loop once a reliability level is reached, and deploying parallel decoding of the constituent sub-decoders. These methods can be summarized as follows.

2.1. Reduction and/or simplification of the computations

Although the BCJR algorithm involves numerous types of algebraic computations, the most time-consuming expression is the exponential function (e^x). Replacing it by a more time-efficient function can significantly reduce the delay. One important proposal for this simplification is the Max Log-MAP [3], which uses the maximum function to approximate the exponential. The slight performance degradation can be tolerated in many applications. This approximation is further improved in the Log-MAP [4] (4) algorithm, where a look-up table is used to reduce the error in the approximation. The computations within the BCJR algorithm can also be reduced by eliminating the less-likely state transitions, and/or skipping those states whose values less than a threshold [5].

2.2. Early-terminating the iterating loop

In addition to the extensive computations within the SISO decoders, the latency in turbo decoding is also related to the iterative behavior of the decoding. Along the iterations, the two constitutive SISO decoders collaborate to result in gradual growth of the output LLRs. The speed of this growth depends on many factors, including the signal-to-noise ratio (SNR) and the locations of the corrupted bits. There are three possible behaviors for this evolution to occur: fast-, slow-, and the non- converging LLRs. The converging blocks are

decodable. For this type, allowing the decoder to perform more iterations per a block increases the chance to achieve the convergence. However, especially as the SNR increases, the decoder is more likely to reach the convergence very early. In these cases, the decoding practically requires fewer iterations; hence, an early stopping rule is used to terminate the loop. Thus, reducing the number of iterations is important to reduce the latency in decoding. Numerous works in the literature propose early stopping (ES) rules to terminate the iterations early. These methods determine the instance when there is no further performance improvement expected. A simple example for such rule is [6], which terminates the loop once the hard outputs from both SISOs match. This significantly reduces the average number of iterations as the SNR increases. For lower SNRs: the decoder is more likely to receive "undecodable" blocks, which means it continues iterating in the hope of reaching the convergence. Several methods, like in [7], define rules to terminate the iteration on undecodable blocks. The convergence behavior of the extrinsic information transfer (EXIT) charts is analyzed in [8] to be used as an early stopping technique.

2.3. Deploying parallel decoding

The classical approach in turbo decoding functions in serial mode as in Figure 2 (a). The constituent decoders take turns computing new LLRs for the received signal based on the last updated extrinsic information gathered from the other decoder as a priori values. This means at any given time, only one of the SISOs operates. The main disadvantage of this mode of operation is the high decoding latency.

To speed-up the decoding, simple parallel decoding is proposed in [9], as in Figure 2 (b). In this scheme, all the constituent decoders run simultaneously. Each one decodes the channel input using the a priori gathered form the other decoder(s) from the previous iteration. In [10], [11], the two decoders operate in synchronous manner: when the first decoder is working on its decoding, it considers the instantaneously generated extrinsic values form the other one, bit by bit. A method is proposed in [12] to use twice the number of decoders: as two SISO decoders per a convolutional code. Each one of the twin decoders run in the opposite direction to deliver the LLRs faster to the other decoders.

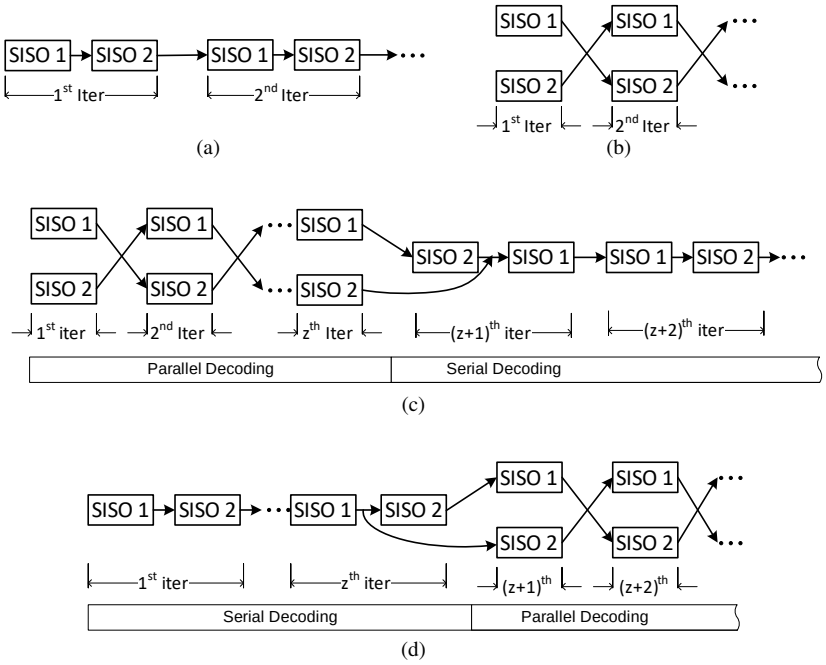


Figure 2. Schemes for the extrinsic information flow for the turbo decoding; (a) Serial decoding, (b) Parallel decoding, (c) Parallel-then-serial decoding, and (d) Serial-then-parallel decoding

Parallel decodable turbo codes are presented in [13]-[17]. The information bits here are divided into multiple groups, each of which is encoded separately and then multiplexed. These groups are decoded in parallel at the receiver. A method is proposed in [18], to speed up the decoding by dividing the received turbo code into sub-blocks without termination, each of which is decoded in parallel separately. Long frames can be

split into smaller sub-blocks and they can be decoded in parallel separately. Each sub-block can be terminated [19] for simpler parallel decoding. Otherwise, the reception is sub-divided directly as in [20], [21].

On the BCJR level, many researches in the literature have proposed methods to deploy parallel processing to speed up the computations of the algorithm's metrics. For example, methods are proposed in [18], [22]-[26] to compute the forward and backward metrics separately in parallel starting from both ends of the whole data block or from a sliding window. Fully-Parallel Turbo Decoding algorithms [27], [28] process the decoding in parallel fashion in terms of the computations of the BCJR parameters per individual bit.

2.4. The proposed methods

Although the proposals of many methods in the literature reduce the decoding latency, they increase the system complexity and/or affect the bit error rate (BER). The extra complexity includes: the requirement for additional hardware (like the processors and the memory units), and the collision-free interleaving. On the other hand, the simple parallel decoding (PD) of the SISOs' Figure 2 (b) [9] can be considered a relatively simple and a good alternative to the conventional serial decoding (SD) Figure 2 (a).

At higher SNRs and with the existence of the ES [6], our simulation shows the difference in the average number of iterations between SD and PD is mostly less than 1 iteration. However, PD considerably affect the BER and the frame error rates (FER) performances at these SNRs. Fast converging blocks require few iterations and they are mostly decoded correctly. The slow and non converging blocks are those which affect the average number of iterations, the BER, and the FER. Despite the large number of iterations required by the slow converging blocks, the results are not always immune from errors. The decoding of such type of blocks performs better through the SD compared to the PD, as the LLRs are gradually evolve through more steps. This could explain the degradation in both the BER and FER using the PD. This work studies the combination of both PD and SD for decoding a received block.

There are two possible ways for the simple combination for these two decoding schemes:

1. The process starts in PD mode for z iterations. Then the turbo decoder continues in SD mode for the rest of the remaining iterations. We call this method *Parallel-then-Serial decoding* $(P|S)$, as illustrated in Figure 2 (c).
2. The process starts in SD mode for z iterations. Then the turbo decoder continues in PD mode for the rest of the remaining iterations. We call this method *Serial-then-Parallel* decoding $(S|P)$, as illustrated in Figure 2 (d).

To compute the average decoding latency, we need to define the units of measure for the different decoding schemes. A single iteration completes when both SISOs deliver their LLRs. However, as shown in Figures 2 (a) and (b), the latency is different per iteration according to the decoding mode. To unify the measures, we set the definition of an iteration for the SD as the standard. This means the iterations in PD is half of those for SD. The number of iterations is used in this paper to refer to the decoding latency. For simplifying the comparison and for the sake of presentation, we strict the latency calculations to the number of iterations for the decoding.

Let μ be the maximum number of decoding iterations, and z be an integer whose value $0 \leq z \leq \mu$. To define a decoding scheme according to the value of z, we can use the notations: $P_z|S_{\mu-z}$ and $S_z|P_{\mu-z}$. The latter, for example, means the constituent decoders run in series for z iterations then they run in parallel for $\mu - z$ iterations. For all values of z, the number of iterations (without ES) for $P|S$ and $S|P$ respectively are expressed in the (4) and (5).

It is apparent that setting $z = \mu$ result in a simple parallel and serial decoding for $P|S$ and $S|P$ respectively, while setting z to 0 results the other way around. In this work, we examine different values for z, for the two presented decoding schemes.

$$\delta_{P|S} = \frac{z}{2} + (\mu - z) = \mu - \frac{z}{2}. \tag{4}$$

$$\delta_{S|P} = z + \frac{\mu - z}{2} = \frac{z + \mu}{2}. \tag{5}$$

3. SIMULATION

The long-term evolution (LTE) specifications of the turbo code [29] are considered. Two 8-state constituent encoders (the generators of the feed-forward and the feedback are 15 and 13, in octal, respectively). The

Quadratic Permutation Polynomial [30] is used for generating the permutation sequence for the interleavers. In the receiver, the used SISO decoders are the Max* Log-MAP algorithm [3]. μ is 12 iterations. The ES [6] is used to terminate the decoding loop once convergence is reached. The size of the message blockes is 512 bits. The channel is assumed to be an ISI-free AWGN with a variance of σ^2. The termination bits of only one encoder are included in the transmission. Puncturing is deployed to reduce the transmission rate. Therefore, the transmission rate becomes 0.4971.

4. DISCUSSION

As we discussed earlier, the reception is more likely to be fast-converging blocks as the SNR increases, while the reduction in the SNR increases the probability of receiving slow- and non-converging blocks. Slow converging blocks perform better through SD as there are more steps to evolve. This is always true, as indicated by comparing the BER and FER curves of SD and PD in Figure 3 (a) (also in Figure 3 (c)). The performance curves for the proposed mixed decoding schemes are also shown in these figures. We limit the results for the cases where $z = 0, 2, 6, 10$ and 12, as listed in Table 1, just for the sake of the clarity of the presentation. The listed $\delta^{\max}$ in this table are for the number of the full-iterations reached without an ES. The schemes $P_2|S_{10}$ and $S_{10}|P_2$, for example, both run for the same number of iterations, but in the opposite order. With the deployment of the ES [6], the actual average number of iterations (ρ^{AVG}) decreases as the SNR increases, as shown in Figure 3 (a) and Figure 3 (b). The reduction in ρ^{AVG} is a result for the increased percentage of receiving fast converging blocks, which are detected by the deployed ES.

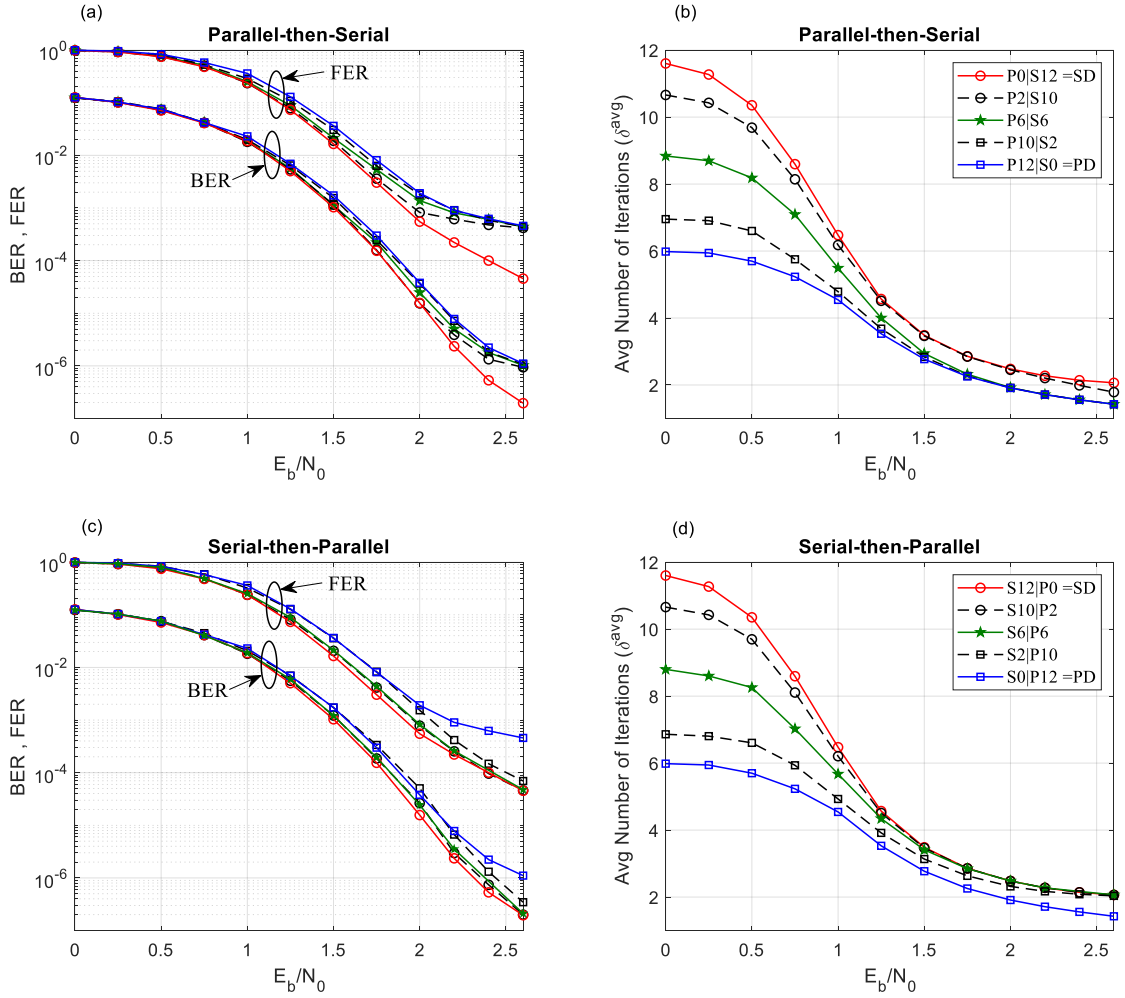


Figure 3. The curves at block size=512 bits of: (a) BER and FER Parallel-then-serial, (b) ρ^{AVG} of Parallel-then-serial, (c) BER and FER of Serial-then-parallel, and (d) ρ^{AVG} of Serial-then-parallel

It is apparent in Figure 3 that the simulation results fill the gaps between SD and PD. The overall characteristics shifts toward SD, as the number of instances of running SISO decoders serially increases. The opposite is also true: they shift toward PD as the parallel instances outnumbers the serial decoding instances.

At low SNRs, where the reception is more likely to by non-converging blocks, all the BER and FER curves are virtually the same. However, the ρ^{AVG} curves of the presented schemes gradually shift from SD to PD as δ decreases (see Figure 3 (b) and Figure 3 (d)). The clear variance in ρ^{AVG} due to the fact each scheme has a different maximum number of iterations, as in Table 1.

Table 1. Number of the full-iterations (without ES) per the selected decoding schemes

$P_z\|S_{\mu-z}$	$S_z\|P_{\mu-z}$	Number of iterations (ρ_{max})
$P_0\|S_{12}$	$S_{12}\|P_0$	$12 \equiv$ SD
$P_2\|S_{10}$	$S_{10}\|P_2$	11
$P_6\|S_6$	$S_6\|P_6$	9
$P_{10}\|S_2$	$S_2\|P_{10}$	7
$P_{12}\|S_0$	$S_0\|P_{12}$	$6 \equiv$ PD

As the SNR increases, the probability of convergence also increases. The performance of BER and FER is directly affected by the decoding scheme. SD achieves the best performance as it runs in a fashion that requires the highest decoding latency. Although PD, on the other hand, requires the least latency, it compromises the performance. The selected combined schemes lie between the two. However, simulation results show that even at small values of z, the $S|P$ method shows noticeable improvement in terms of the FER and BER performance compared to the $P|S$, even at the same ρ^{max}. The improvement is more effective in terms of the FER as the SNR increases. This can be credited to the improvement in the decoding of the slow-converging blocks. The $S|P$ method allows the slow-converging blocks to develop at their early stages through more gradual steps. The effectiveness of this method throughout different block sizes depends on the value of z, as these figures show.

Regarding the comparison through ρ^{AVG}, it is clear from the curves that the proposed methods bridge the gap at the scalable compromise between the decoding latency and the BER/FER performance. The value of z can be selected to satisfy the required level of performance and/or the tolerable level of the decoding latency. However, at high SNRs, the value of z becomes ineffective on the average decoding latency. This means the decoding loop already terminates at a very early stage. At very high SNR, simulation shows that ρ^{AVG} when using the decoding scheme $P_z|S_{\mu-z}$ converges to the value of PD, and regardless of z. The same is true for $S_z|P_{\mu-z}$, the ρ^{AVG} converges to the value of SD. This can be a result to the used ES, which successfully terminates most of the reception. Yet, $S_z|P_{\mu-z}$ provide a significant gain in BER and FER by this additional slight latency.

5. CONCLUSION

Although the parallel turbo decoding provides significant latency reduction, it affects the FER and BER performance. Running the SISOs in serial achieves better performance at the price of higher decoding latency. In this paper, we propose two methods that combine both decoding modes. The first one starts the decoding by running the constitutive SISOs in serial for few iterations and then in parallel for the rest of the remaining iterations. The second one runs constitutive SISOs is in the other way around. The simulation shows improvements in BER and FER in both methods. However, the improvement is more significant using the former method. The proposed method can be further studied in the future on different encoder and interleaver configurations to examine its effectiveness.

REFERENCES

[1] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo-codes. 1," in *Proceedings of ICC '93 - IEEE International Conference on Communications*, 1993, pp. 1064-1070 vol.2, doi: 10.1109/ICC.1993.397441.

[2] S. Shao *et al.*, "Survey of Turbo, LDPC, and Polar Decoder ASIC Implementations," in *IEEE Communications Surveys & Tutorials*, vol. 21, no. 3, pp. 2309-2333, thirdquarter 2019, doi: 10.1109/COMST.2019.2893851.

[3] P. Robertson, E. Villebrun, and P. Hoeher, "A comparison of optimal and sub-optimal MAP decoding algorithms operating in the log domain," in *Proceedings IEEE International Conference on Communications ICC '95*, 1995, pp. 1009-1013 vol.2, doi: 10.1109/ICC.1995.524253.

[4] P. Robertson, P. Hoeher, and E. Villebrun, "Optimal and Sub-Optimal Maximum A Posteriori Algorithms Suitable for Turbo Decoding," *European Transactions on Telecommunications*, vol. 8, no. 2, pp. 119-125, 2008, doi: 10.1002/ett.4460080202.

[5] V. Franz and J. B. Anderson, "Concatenated decoding with a reduced-search BCJR algorithm," in *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 2, pp. 186-195, Feb. 1998, doi: 10.1109/49.661107.

[6] T. M. N. Ngatched and F. Takawira, "Simple stopping criterion for turbo decoding," *Electronics Letters*, vol. 37, no. 22, pp. 1350-1351, Oct. 2001, doi: 10.1049/el:20010896.

[7] M. AlMahamdy and J. Dill, "Early Termination of Turbo Decoding by Identification of Undecodable Blocks," in *2017 IEEE Wireless Communications and Networking Conference (WCNC)*, 2017, pp. 1-6, doi: 10.1109/WCNC.2017.7925644.

[8] Y. Liu, L. Xiang, R. G. Maunder, L.-L. Yang, and L. Hanzo, "Hybrid Iterative Detection and Decoding of Near-Instantaneously Adaptive Turbo-Coded Sparse Code Multiple Access," in *IEEE Transactions on Vehicular Technology*, vol. 70, no. 5, pp. 4682-4692, May 2021, doi: 10.1109/TVT.2021.3071808.

[9] D. Divsalar and F. Pollara, "Multiple turbo codes," in *Proceedings of MILCOM '95*, 1995, pp. 279-285 vol. 1, doi: 10.1109/MILCOM.1995.483313.

[10] Y.-C. Lu and E.-H. Lu, "A Parallel Decoder Design for Low Latency Turbo Decoding," in *Second International Conference on Innovative Computing, Informatio and Control (ICICIC 2007)*, 2007, pp. 386-386, doi: 10.1109/ICICIC.2007.73.

[11] J. Zhang and M. P. C. Fossorier, "Shuffled iterative decoding," in *IEEE Transactions on Communications*, vol. 53, no. 2, pp. 209-213, Feb. 2005, doi: 10.1109/TCOMM.2004.841982.

[12] Y. Wang, J. Zhang, M. Fossorier, and J. S. Yedidia, "Reduced latency turbo decoding," in *IEEE 6th Workshop on Signal Processing Advances in Wireless Communications, 2005.*, 2005, pp. 930-934, doi: 10.1109/SPAWC.2005.1506276.

[13] L. Xiang, M. F. Brejza, R. G. Maunder, B. M. Al-Hashimi, and L. Hanzo, "Arbitrarily Parallel Turbo Decoding for Ultra-Reliable Low Latency Communication in 3GPP LTE," in *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 4, pp. 826-838, April 2019, doi: 10.1109/JSAC.2019.2898654.

[14] O. Gazi and A. O. Yilmaz, "Fast Decodable Turbo Codes," in *IEEE Communications Letters*, vol. 11, no. 2, pp. 173-175, Feb. 2007, doi: 10.1109/LCOMM.2007.061528.

[15] O. Gazi, "Analysis of parallel decodable turbo codes," *IEICE Transactions on Communications*, vol. E95.B, no. 5, pp. 1584-1591, May 2012, doi: 10.1587/transcom.E95.B.1584.

[16] M. Taskaldiran, R. C. S. Morling, and I. Kale, "Parallel decoding of turbo codes using multi-point trellis termination and collision-free interleavers," in *2009 Wireless Telecommunications Symposium*, 2009, pp. 1-5, doi: 10.1109/WTS.2009.5068947.

[17] B. Le Gal and C. Jego, "Low-latency and high-throughput software turbo decoders on multi-core architectures," *Annals of Telecommunications*, vol. 75, no. 1, pp. 27-42, Feb. 2020, doi: 10.1007/s12243-019-00727-5.

[18] C. Schurgers, F. Catthoor, and M. Engels, "Optimized MAP turbo decoder," in *2000 IEEE Workshop on SiGNAL PROCESSING SYSTEMS. SiPS 2000. Design and Implementation (Cat. No.00TH8528)*, 2000, pp. 245-254, doi: 10.1109/SIPS.2000.886722.

[19] K. Wan, Q. Chen and P. Fan, "A novel parallel turbo coding technique based on frame split and trellis terminating," in *Proceedings of the Fourth International Conference on Parallel and Distributed Computing, Applications and Technologies*, 2003, pp. 927-930, doi: 10.1109/PDCAT.2003.1236452.

[20] J.-M. Hsu and C.-L. Wang, "A parallel decoding scheme for turbo codes," *ISCAS '98. Proceedings of the 1998 IEEE International Symposium on Circuits and Systems (Cat. No.98CH36187)*, 1998, pp. 445-448 vol. 4, doi: 10.1109/ISCAS.1998.698923.

[21] S. Yoon and Y. B.-Ness, "A parallel MAP algorithm for low latency turbo decoding," in *IEEE Communications Letters*, vol. 6, no. 7, pp. 288-290, July 2002, doi: 10.1109/LCOMM.2002.801310.

[22] J.-W. Jung *et al.*, "Design and Architecture of Low-Latency High-Speed Turbo Decoders," *Etri Journal*, vol. 27, no. 5, pp. 525-532, Oct. 2005, doi: 10.4218/etrij.05.0905.0033.

[23] M. Taskaldiran, R. C. S. Morling, and I. Kale, "Increasing the speed of parallel decoding of turbo codes," in *2009 Ph.D. Research in Microelectronics and Electronics*, 2009, pp. 304-307, doi: 10.1109/RME.2009.5201330.

[24] Y. Zhang and K. K. Parhi, "Parallel Turbo decoding," in *2004 IEEE International Symposium on Circuits and Systems (IEEE Cat. No.04CH37512)*, 2004, pp. II-509, doi: 10.1109/ISCAS.2004.1329320.

[25] M. May, T. Ilnseher, N. Wehn, and W. Raab, "A 150Mbit/s 3GPP LTE Turbo code decoder," in *2010 Design, Automation & Test in Europe Conference & Exhibition (DATE 2010)*, 2010, pp. 1420-1425, doi: 10.1109/DATE.2010.5457035.

[26] T. Ilnseher, F. Kienle, C. Weis, and N. Wehn, "A 2.15GBit/s turbo code decoder for LTE advanced base station applications," in *2012 7th International Symposium on Turbo Codes and Iterative Information Processing (ISTC)*, 2012, pp. 21-25, doi: 10.1109/ISTC.2012.6325191.
[27] R. G. Maunder, "A Fully-Parallel Turbo Decoding Algorithm," in *IEEE Transactions on Communications*, vol. 63, no. 8, pp. 2762-2775, Aug. 2015, doi: 10.1109/TCOMM.2015.2450208.
[28] A. Li, L. Xiang, T. Chen, R. G. Maunder, B. M. Al-Hashimi, and L. Hanzo, "VLSI Implementation of Fully Parallel LTE Turbo Decoders," *IEEE Access*, vol. 4, pp. 323-346, 2016, doi: 10.1109/ACCESS.2016.2515719.
[29] ETSI, "Technical Specification; LTE; Evolved Universal Terrestrial Radio Access (E-UTRA); Multiplexing and channel coding (3GPP TS 36.212 version 16.2.0 Release 16)," *Tech. Rep.*, 2020. [Online] Available: https://www.etsi.org/deliver/etsi_ts/136200_136299/136212/16.02.00_60/ts_136212v160200p.pdf
[30] J. Sun and O. Y. Takeshita, "Interleavers for turbo codes using permutation polynomials over integer rings," in *IEEE Transactions on Information Theory*, vol. 51, no. 1, pp. 101-119, Jan. 2005, doi: 10.1109/TIT.2004.839478.

BIOGRAPHIES OF AUTHORS

Mohammed AlMahamdy has been a faculty member at the department of Electrical Engineering in University of Anbar (Iraq) since 2006. He received: the Ph.D. degree in Communication Engineering from Ohio University (USA) in 2017; the Master degree in Communication Engineering from AlNahrain University (Iraq) in 2003; and the Bachelor degree in Electronic Engineering from the University of Technology (Iraq) in 2000. From 2005-2006, he worked for Asiacell, a GSM operator in Iraq. The research interests are: Communication Engineering; Error Correcting Codes; Turbo Codes.
Further info on his homepage: https://www.uoanbar.edu.iq/English/staff-page.php?ID=665

Naser Al-Falahy received his Ph.D. degree in wireless telecommunications from University of Salford, Manchester, UK in 2018, and his BSc and MSc degrees in electronics and communications from Al-Nahrain University, Baghdad, Iraq in 2001 and 2005, respectively. From 2005 to 2010, he worked with Motorola® for mobile networks provision and network optimisation. His research interests include mobile communications, radio network planning and optimisation, and millimetre wave communications.
Further info on his homepage: https://www.uoanbar.edu.iq/English/staff-page.php?ID=678