Implementation of Efficient Stopping Criteria for Turbo Decoding

Ruaha Mahdi1, Ahmed A. Hamad2.
1Department of Electrical Engineering, College of Engineering, University of Babylon, Hilla, Iraq
2Department of Electrical Engineering, College of Engineering, University of Babylon, Hilla, Iraq

E-mail: ruaa.alkaldi94@gmail.com
E-mail: eng.ahmed.ak@uobabylon.edu.iq

Abstract. The cellular systems with 4th generation, Long Term Evolution (LTE) standard have been transmitted the data at higher rates than the 3G, and 2G systems, in an ever-crowded frequency spectrum. This transmission needs more accuracy, high reliability, and throughput with a low area and power consumption. Therefore, it requires an efficient error control coding. Turbo code is used for LTE system for its good error correction ability at low signal to noise power ratio which can approach Shannon limit performance for large frame lengths. This paper presents a simple and efficient modification that can be applied to all present stopping criterion (SC) to achieve the quality of service mentioned previously. It’s proved that using a trimmed sequence of log-likelihood values (LLR) instead of the full length in the algorithms of stopping the iterative decoding has a significant impact on the utilized silicon area and throughput without a significant sacrifice in performance. It also presents a comparison of designing the Soft-Output Viterbi (SOVA) decoder using different arbitrary-precision fixed data types that offers by Vivado high-level synthesis (HLS) instead of the costlier float representation to reduce processing time and consumed area. Due to its high flexibility in designing and implementing prototype systems, the FPGA device (Kintex-7, Xilinx part number XC7K325T-2FFG900C) was utilized with different parallelism and loop pipelining directives to ensure acquiring the targeted initiation interval and silicon area.

Keywords. Soft-Output Viterbi (SOVA), long term evolution (LTE), Field-Programmable Gate Array (FPGA), High-Level Synthesis (HLS).

1. Introduction
These days, efficient data transmission is due to the vast development of wireless communication systems. The channel coding technique in these systems becomes an important part of their design. It helps in alleviating the effects of noise that contaminated the transmitted signal. This has made it an indispensable unit in all the recent communication standards. Turbo codes are a group of efficient forward error-correcting codes that assist in achieving reliable data transmission over noisy channels [1,2]. It was pioneers by Berrou, Glavieux, and Thitimajshima in 1993 [3]. It becomes commonly
applied in many wireline and wireless communication standards because of its performance that approaches the Shannon limit. The Maximum A-Posteriori Probability (MAP) [4], Logarithmic MAP (Log-MAP) [5], Maximum Log-MAP (Max Log-MAP) [6], and the Soft-Output Viterbi Decoder (SOVA) [7] are prominent algorithms that have been used for the Turbo decoding. The extrinsic information produced by the two component decoders comprises the Turbo decoder is passed to each other iteratively. This external data which circulated between these decoders are the reason for its eminent performance [8]. In most cases, the performance is improved as the number of iterations increases until a certain point where no significant improvement can be achieved. However, this will increase the processing time as well as the consumed power. To solve this problem, a design trade-off is needed, in which, the decoder terminates the iteration once the traded LLR information converges even before reaching the preset maximum number of iterations (Imax). Numerous SC have been introduced since the invention of Turbo code aim to terminate the iteration in the decoding process and hence minimize the latency, hardware resources, computational complexity, and power consumption.

The rest of this paper is organized as follows. Section 2 provides an overview of the basic concepts of Turbo encoding and decoding. The stopping criteria rule is formulated in Section 3. The proposed system is described in Section 4. The overall process for simulation and hardware of Genie, CE, SCR, and HDA stopping criteria is discussed in Section 5. Finally, we conclude in Section 6.

2. Turbo Codes

The authors of the article [3], introduced a scheme called Turbo Code in which they adopted a concatenation scheme of two parallel convolutional code (PCCC) joined by one interleaver. The two constituent encoders are fed by the same information bits in different orders. Typically, the lower encoder is fed by a permuted version of the bit sequence (by the interleaver) of those entering the upper encoder. Usually, the first encoder is generally terminated (i.e., return to all zero states) by adding dummy bits, usually equal to the constraint length of the encoder, at the end of the information sequence.

Turbo Encoder

In this paper, the scheme of turbo encoder that is employed is a PCCC with two 4-state constituent encoders and one interleaver of length $N$ [9]. In the turbo code scheme, interleaving is the process of rearranging the ordering of an information sequence, which enters directly to the first convolutional encoder, in a one-to-one deterministic way before the application of the second component code [10]. The inverse of this process is called deinterleaving which restores the received sequence to its original order. The quadratic permutation polynomials (QPP) were selected as interleavers for LTE turbo codes [11]. The QPP interleavers are given by the following quadratic polynomial.

\[
\pi(x) = (q_1x + q_2x^2) \mod N
\]

(1)

The structure of the turbo encoder is shown in figure 1. The constituent systematic convolutional (RSC) code has the following transfer function:

\[
G(D) = \left[1, g_1(D)/g_0(D)\right]
\]

(2)

Where we define

\[
g_1(D) = 1 + D + D^2 \\
g_0(D) = 1 + D^2
\]

To encode the input bits, the initial value of the shift registers for each component encoder should be reset to all zeros when starting.
Figure 1. Schematic diagram of turbo encoder.

Turbo Decoder
The iterative turbo decoder has a typical structure as shown in Figure 2. Two-essential SOVA decoders are linked by interleaver in a structure analog to that of the encoder [12].

Figure 2. Schematic diagram of the SOVA turbo decoder.

The soft outputs are typically represented in terms of the so-called Log-Likelihood Ratios (LLRs). The concept of LLRs was introduced in [13, 12, 14] to facilitate the passing of information from one component decoder to the other in the iterative decoding of turbo codes. $L(u_k)$ is the LLR of a data bit $u_k$ and is defined as the log of the ratio of the probabilities of the bit taking its two typical values, i.e.,
In some applications, like channel coding theory it is interested in LLRs based on conditional probabilities, i.e., the probability that \( u_k = \pm 1 \) based, on the receiver’s matched filter output \( y_k \).

Therefore, the conditional LLR \( L(u_k \mid y_k) \) can be written as

\[
L(u_k \mid y_k) = L_y y_k + L(u_k)
\]

Where

\[
L_y = \frac{E_b}{2\sigma^2} 4a
\]

\( L_y \) is defined as the channel reliability value, \( E_b \) is the broadcast energy per bit, \( a \) is the fading amplitude (\( a=1 \) for non-fading AWGN channel), and \( \sigma^2 \) is the noise variance.

In a practical implementation of turbo decoding, two fitting decoders are utilized, namely SOVA algorithm and the MAP algorithm. The SOVA is less complex but suffers degradation of about 0.7 dB at bit-error-rate (BER) of \( 10^{-4} \) compared to the MAP in general [13]. To improve their decoding performance SOVA decoder receives the reliability values as a-prior information.

**Implementation of SOVA Algorithm**

The SOVA algorithm can be performed as follows:

1. Compute the metric for two paths meeting into the state using Equation (6)

\[
M(s_k^i) = M(s_{k-1}^i) + \frac{1}{2} u_k L(u_k) + \frac{L_y}{2} \sum_{l=1}^{n} y_{kl}x_{kl}.
\]

Let \( (S_k^i \) is the state sequence) \( M(s_k^i) \) denotes the survivor path metric and \( M(s_k^d) \) denotes the discarded (competing) path metric.

2. Select the path with the highest metric \( M(s_k^d) \).

3. Store \( M(s_k^d) \) and its associated survivor bit with an indicator to the previous state along the surviving path.

4. Compute the difference \( \Delta^i_k \) using Equation (7)

\[
\Delta^i_k = M(s_k^i) - M(s_k^d) \geq 0
\]

and store its value, jointly with a binary vector containing \( \delta + 1 \) bits, which indicate whether or not the discarded path would have given the same sequence of bits \( u_t \) for \( l = k \) back to \( l = k - \delta \) as the surviving path does. This sequence of bits is called the update sequence [15].

5. At the end of \( \delta + 1 \) transitions, the SOVA decoder identifies the ML path, and with aid of the stored update sequence, metric differences and the values of \( L(u_k \mid y_k) \) are calculated using Equation 7 and

\[
L(u_k \mid y_k) \approx u_k \min_{\Delta^i_k} \Delta^i
\]
The SOVA decoder has the facility to accept a-priori information $L(u_k)$ as well as producing $L(u_k \mid y_i)$ for each input bit in the received sequence. This makes SOVA a suitable candidate for iterative turbo decoder.

3. Stopping Criteria (SC)

In iterative decoding, $I_{\text{max}}$ (fixed number of iterations) is the maximum number of iterations that are required by the worst corrupted frame to be decoded, although, most frames include highly reliable decoded bits that require a small number of iterations to terminate especially in high signal to noise power ratio (SNR) [16]. To avoid excess decoding, the iteration process for these frames should be stopped before $I_{\text{max}}$ by an efficient SC [17]. It would reduce the average decoder iteration with or without a shallow degradation in performance [18]. Hence, it is important to reduce unnecessary computations and decoding delay and to develop an SC for early iteration termination [17, 19]. A short review of some SC are as follows.

A. Genie algorithm

The “Genie” instance, where the information bits are known and after the frame is correctly decoded the iteration is stopped immediately. This scheme is practically infeasible as the information bits are not available at the decoder side and it is considered as the limit of all possible stopping schemes.

B. Cross-Entropy (CE)

The CE stopping criterion calculates the approximate cross-entropy between the distribution estimates of the decoder's output at each iteration.

So, the cross-entropy $H(l)$ is approximated as follows [20-22]

$$H(l) \approx \sum_k \frac{\Delta L e_x^{(l)}(u_k)}{e^{[L^{(l)}(u_k)|y_k]}}$$

where,

$$\Delta L e_x^{(l)}(\tilde{u}_t) = L e_x^{(l)}(\tilde{u}_t) - L e_x^{(l)}(\tilde{u}_t) = L e_x^{(l)}(\tilde{u}_t) - L e_x^{(l-1)}(\tilde{u}_t)$$

and $L e_x^{(l)}(\tilde{u}_t)$ is the extrinsic information estimated in the $l$ iteration at decoder $c$. The met of the following condition will stop the iteration.

$$H(l) < (10^{-2} \sim 10^{-4}) H(1)$$

C. Hard-Decision aided (HDA)

The HDA stopping criterion is an effective method derived from the concept of CE [22]. It inspects the signs of the data bits, $L_x(u_k \mid y_i)$ between two successive iterations. The iteration is ceased when the signs of decoded bits are typical to those of the previous iteration.

$$\text{Sign of } (L_x(u_k \mid y_i)^{(l-1)}) = \text{Sign of } (L_x(u_k \mid y_i)^{(l)}) \text{ for } k = 0 \text{ to } N - 1$$

D. Sign Change Ratio (SCR)

The SCR scheme can concede the complex CE computation through estimating the change of sign in the decoder’s soft output as well as comparing the output of the hard decision decoder [23]. The sign changes of $L e_x(\tilde{u})$ from iteration $(l-1)$ to $l$ iteration Known as $C(l)$ [20][22]. At each iteration, when $C(l) \leq (0.005 \sim 0.03)N$, SCR algorithm stop the iterations [24][25]. This method needs storage to hold the sign values calculated from the previous iteration.
4. Proposed System
To reduce the time consumed by the SC it is proposed to process only a partial length of the LLR sequence that is used in the SC instead of the full length. It is interesting to study the effects of trimming the processed LLR sequences on the bit-error performance, consumed silicon area, the average number of iterations, and average decoding time.

Different data types like float and arbitrary precision fixed (ap-fixed) provided by HLS have been employed to present their effects on the quality of the tested systems. Table (1) shows the parameters of turbo codes that simulated and implemented using Vivado HLS and SDK platform.

Table 1. Turbo encoder and decoder parameters.

| Component encoder | Two Identical Recursive Systematic Convolutional Codes (RSC) |
|-------------------|-------------------------------------------------------------|
| Interleaver       | Quadratic Permutation Polynomial Interleaver (QPP)          |
| Maximum number of iteration (I_{max}) | 6                                                          |
| Interleaver Length | 64,256                                                     |
| percentages of process length | %100, %50, %25, %10                                       |
| Decoding Technique | SOVA                                                        |
| Coding Rate       | R=1/3                                                       |
| Generator Polynomials | g[7,5]                                                    |
| Channel           | AWGN                                                        |
| Stopping Criteria | Fixed, Genie, CE, SCR, HDA                                 |
| Fixed-point data types | ap\_fixed<16,10,AP\_TRN,AP\_SAT>                     |
|                   | AP\_TRN :Truncation to minus infinity (default)            |
|                   | AP\_SAT: Saturation                                        |

5. Simulation and Hardware Results
The work was divided into two stages; The first is modeling the proposed system using Vivado HLS platform using C++ language and applying different directives to make a tradeoff between silicon area and throughput. The second is the hardware implementation based on the Xilinx Kintex-7 FPGA device. The schematic diagram for the different IPs (intellectual property) that represents the tested system is shown in Figure 3. It consists of two types of IP blocks; the ones that are generated by HLS (data source, AWGN channel, turbo code encoder, and decoder) and the supporting IPs provided by Vivado to facilitate the hardware process (Microblaze, timer, AXI interface, DDR memory, and clock manager, etc.).

![Figure 3. The schematic diagram of the turbo coded system](image)

Directives such as array partitioning, pipelining, Data-flow, and unrolling processes are used in Vivado HLS which impactful the hardware designs, and thus greatly affecting performance and resource utilization. Table 2 shows details on latency (in terms of the number of clock cycles) and different FPGA
resources (LUT, FF, BRAM, and DSP processor) for two syntheses solutions with different optimization directives. Solution 1 uses the default HLS directive which is arranged to achieve a compromise between throughput and resources. On the other hand, in solution 2 different parallelism directives are applied such as partitioning all arrays and unrolling all looping iterations. The calculations are based on using different data types like float and fixed-point data types. The results present the tradeoff between latency and resource utilization. The achieved gain in latency in solution 2 is offset by an increment in area. Applying the arbitrary-precision data types like ap_fixed<W, I, Q, O>, which characterizes a fixed-point number has a considerable effect on latency and resource utilization. Here, W represents the hole width in a bit of the signed word and I is the number of fractional bits. The letters Q and O represent the Quantization and Overflow Modes respectively which can take different forms as illustrated in [26]. Figure 4 shows a BER performance comparison for the same system, 60 bits information length, HDA SC, with float and arbitrary precision fixed data types. It should be accepted of about 0.2 dB scarification for the migration from the float to the fixed format as a result of limited precision.

Table 2. Latency and utilization estimate for floating-point and arbitrary-precision fixed types for the SOVA decoder with g(7,5), N=64, and Imax=6.

| Float type | Latency (CLK) | BRAM_18K | DSP48E | FF | LUT |
|------------|--------------|----------|--------|----|-----|
| Solution1 | 14774         | 14       | 16     | 4713| 7222 |
| Solution2 | 6017          | 25       | 26     | 11551| 34429|

Fig. 5 shows the BER performance for different SC with a maximum number of iterations equal Imax =8 with 64 and 256 bits information length. The HDA algorithm reveals a closer performance to the benchmark genie.

![Figure 4. BER performance of 60 bits turbo coded systems using float and ap-fixed data types.](image-url)
To illustrate the effect of trimming the length of LLR sequence that is utilized in the stopping algorithm, simulation tests are carried out for different percentages (%100, %50, %25, %10) of the original length N (64 bits) and applying SCR algorithm are shown in Figure 6. Trimming causes a significant reduction in the consumed silicon area, the average number of iterations, and delay time on the account of a shallow degradation in the bit-error performance. A maximum of about 0.2 dB for 10% trimming is lost at BER of $10^{-4}$ as shown in Figure 6.
Table 3 illustrate the latency, resource utilization of the SOVA turbo decoder for various trimming ratios taken \(N=256\). A gain is achieved in latency, the number of used FF’s, and LUT’s as a result of the reduction in processing and memory requirements for the trimmed systems.

**Table 3.** The latency, resource utilization for different trimming ratios.

|            | 100% | 50%  | 25%  | 10%  |
|------------|------|------|------|------|
| Latency (CLK) | 129510 | 129446 | 129414 | 129395 |
| BRAM_18K   | 24   | 24   | 24   | 24   |
| DSP48E     | 21   | 21   | 21   | 21   |
| FF         | 16246 | 11191 | 8663 | 6893 |
| LUT        | 54736 | 33302 | 21860 | 14498 |

To illustrate the effect of trimmed-length SC on the average number of iterations (AvgItr) and average decoding time (AvgDecT), different hardware tests are carried out for turbo coded system of length \(N=64\) information bits and HDA stopping algorithm. Figure 7 and 8 depict the average number of iterations and the average decoding time against \(E_b/N_o\) respectively. The systems with high trimmed length have less AvgItr and AvgDecT.

![Figure 7. Average number of iterations for various trimming ratios.](image-url)
Figure 8. Average decoding time for various trimming ratios.

It is interesting to compare the performance of hardware and simulation results. Figures 9, 10, and 11 show the BER, AvgDecT, and AvgItr comparison of the hardware and simulation results of the turbo coded system of length 64, HDA SC (50 and 100% trimming ratio), and ap-fixed <16,10, AP_TRN, AP_SAT> LLR data type. From figure 8, it is shown that the simulated system outperform the hardware performance by about 0.3dB at BER of $10^{-4}$, this is due to the hardware limitation of the FPGA device. On the other hand, the trimmed systems (hardware, and software) reveals a shallow degradation in BER performance compared to the full-length systems. Figure 9 reveals that the simulated and hardware systems that apply the trimming scheme present a reduction in the average number of iterations of about 10% compared to the original systems. Finally, figure 10 depicts a comparison of the same systems in terms of AvgDecT in which a reduction of more than 50% in decoding time is achieved.

Figure 9. BER performance of hardware and simulation, frame size=64, ap-fixed<16,10,AP_TRN,AP_SAT>, and HDA (50% and %100 SC).
Figure 10. AvgTimedec of HW and simulation, frame size=64, ap-fixed<16,10,AP_TRN,AP_SAT>, and HDA (50% and %100 SC).

Figure 11. AvgDecT for hardware and simulation, frame size=64, ap-fixed<16,10, AP_TRN, AP_SAT>, and HDA (50% and %100 SC).

6. Conclusions
In this paper, we suggested a simple method to reduce the latency of turbo decoding by trimming the LLR sequence that is employed in various stopping criteria. Although all SC assume that the passed LLR information between the two parallel turbo decoders should be long enough and has a Gaussian distribution, it is proved that using a shortening sequence in SC, even in the codes that have short lengths, has no significant effects on its performance. At the same time, a reduction in the average number of
iterations and average turbo decoding time is acquired. Different hardware tests are carried out using Kintex 7 FPGA and HLS platform to confirm the simulation results of various turbo coded systems. It is revealed that a significant reduction in latency and utilization resources are obtained. A significant reduction in latency and silicon area is achieved when fixed-point data type is used instead of the costlier float type to represent the LLR information processed by the two constituent SOVA decoders. It also present improvement in average number of required iterations, and average decoding time at the expense of a degradation in BER performance.

References
[1] S. Lin and D. J. Costello “Error Control Coding, Second Edition” Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 2004.
[2] S. Tripathi, R. Mathur, and J. Arya, “Unified 3gpp and 3gpp2 turbo encoder FPGA implementation using run-time partial reconfiguration,” in Wireless Telecommunications Symposium (WTS), 2010. IEEE, 2010, pp. 1–8.
[3] C. Berrou and A. Glavieux and P. Thitimajshima, “Near Shannon Limit Error-Correcting Coding and Decoding: Turbo Codes,” in Proceedings of the International Conference on Communications, (Geneva, Switzerland), pp. 1064–1070, May 1993.
[4] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, “Optimal Decoding of Linear Codes for Minimizing Symbol Error Rate,” IEEE Trans. Inform. Theory, vol. IT-20, pp. 284-287, 1974.
[5] P. Sharma, L. Chawla and D. Vats, "BER Analysis Using Log-Map Decoding Algorithm for Turbo Codes Channel", International Journal of Enhanced Research in Science, Technology & Engineering, Vol. 4 Issue 8, August 2015.
[6] E. M. A. Abdel-Azim, "Design and Implementation for a multistandard Turbo Decoder using a reconfigurable ASIP", MSc. thesis, Cairo University, 2013.
[7] J. Hagenauer and P. Hoeher, “A Viterbi Algorithm with Soft-Decision Outputs and its Applications,” in IEEE Globecom, pp. 1680–1686, 1989.
[8] Y. Beeharry, T. P. Fowdur, and K. M. S. Soyjaudah, “Performance of Bit-Level Decoding Algorithms for Binary LTE Turbo Codes with Early Stopping,” vol. 17, no. 2, pp. 3399–3415, 2017.
[9] 3GPP Technical Specifications 36.212, “Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA); Multiplexing and channel coding (Release 15)”, 2019.
[10] R. Soleymani, Y. Gao, and U Vilaipornsawai, “Turbo Coding for Satellite and Wireless Communications,” Kluwer Academic Publishers, 2002.
[11] Ching-Lung Chi, Chih-Hsiao Kuo, “Quadratic Permutation Polynomial Interleaver for LTE Turbo Coding” IEEE, 2012.
[12] L. Hanzo, and P. Woodard, “Comparative Study of Turbo Decoding Techniques: An Overview,” IEEE Transactions On Vehicular Technology, Vol. 49, NO. 6, November 2000.
[13] P. Robertson, E. Villebrun, and P. H’Oher, “A Comparison of Optimal and Sub-Optimal MAP Decoding Algorithms Operating in the Log Domain,” in Proceedings of the International Conference on Communications, (Seattle, United States), pp. 1009–1013, June 1995.
[14] P. Robertson, “Illuminating the Structure of Code and Decoder of Parallel Concatenated Recursive Systematic (turbo) Codes,” IEEE Globecom, pp. 1298–1303, 1994.
[15] J. Hagenauer, “Source-controlled Channel Decoding,” IEEE Transactions on Communications, vol. 43, pp. 2449-2457, September 1995.
[16] H. Chen, et al., “A Survey and Tutorial on Low-Complexity Turbo Coding Techniques and a Holistic Hybrid ARQ Design Example,” Communications Surveys & Tutorials, IEEE, vol. PP, pp. 1-21, 2013.
[17] W. Shao and L. Brackenbury, "Early stopping turbo decoders: a high-throughput, low-energy bit-level approach and implementation," Communications, IET, vol. 4, pp. 2115-2124, 2010.
[18] H. Lei, et al., "Information Theoretic Criterion for Stopping Turbo Iteration," IEEE Transactions on Signal Processing, vol. 59, pp. 848-853, 2011.
[19] E. A. Obiedat and L. Cao, "Turbo Decoder for Low-Power Ultrawideband Communication Systems," International Journal of Digital Multimedia Broadcasting, vol. 2008, 2008.

[20] Z. C. Zhang and Y. Li, “Research and improvement on stopping criterion of Turbo iterative decoding,” 2010 6th Int. Conf. Wirel. Commun. Netw. Mob. Comput. WiCOM 2010, no. 4, 2010.

[21] R. Mohamad, H. Harun, M. Mokhtar, W. A. W. Adnan, and K. Dimyati, “Performance analysis of stopping turbo decoder iteration criteria,” Proc. - 2014 IEEE 10th Int. Colloq. Signal Process. Its Appl. CSPA 2014, no. 1, pp. 5–9, 2014.

[22] W. Shao and L. Brackenbury, “Early stopping turbo decoders: a high throughput, low-energy bit-level approach, and implementation,” Commun. IET, vol. 4, no. 17, pp. 2115–2124, 2010.

[23] R. Mohamad, H. Harun, M. Mokhtar, W. A. W. Adnan, and K. Dimyati, “Threshold-Based Bit Error Rate for Stopping Iterative Turbo Decoding in a Varying SNR Environment,” Frequenz, vol. 71, no. 1–2, pp. 83–94, 2017.

[24] R. Mohamad and M. Kassim, “Robustness Analysis of the Improved Minimum Descriptive Length Stopping Criterion,” 2016 IEEE Conf. on Sys., Pro. and Cont. (ICSPC), no. December, pp. 86–90, 2016.

[25] R. Mohamad, H. Harun, M. Mokhtar, W. A. W. Adnan, and K. Dimyati, “On the robustness of measurement of reliability stopping criterion in turbo iterative decoding,” 2015 IEEE/ACIS 16th Int. Conf. Softw. Eng. Artif. Intell. Netw. Parallel/Distributed Comput. SNPD 2015 - Proc., pp. 1-5, 2015.

[26] "Vivado Design Suite User Guide: High-Level Synthesis", UG902 (v2018.3) December 20, 2018.