

Agenda Item: 6.4
Source: Broadcom and ZTE
Title: Tail-biting offers coding gain with the warm-up using the existing state metric processor
Document for: Discussion/Decision

1 Summary

Tail-biting encoding has been discussed as an option for 3GPP LTE turbo coding during the past years (see reference [1-7]). Moreover, this technique has already been used in several standards [8-11]. New Interleaver structures of QPP based turbo coding will support parallel decoding operations where a circular structure of the tail-biting encoder will benefit decoder operation. However, several questions have been raised in 3GPP LTE discussions, namely,

- a) Is there a performance gain of tail-biting;
- b) Is tail-biting encoding latency a problem; and
- c) Is there a practical “genie” for decoding with tail-biting

In this document, we try to answer these questions.

We show performance for block sizes where tail-biting dramatically improves performance. It may also be the case that turbo coding over block sizes that may required for VOIP operations may increase performance and sensitivity of the overall VOIP operation. A contribution addressing this aspect is being prepared for consideration.

The incurred latency, and impact to the encoder in terms of power and gate count are shown to be minimal.

We show the circular structure of the tail-biting coupled with the required warm-up decoding operation necessary for slide window decoding technique and parallel decoding of high rate data transmission make tail-biting an attractive alternative for LTE.

Also in this document methods of tail-biting encoding for 3GPP LTE turbo codes using QPP interleaves with and without limitation on multiple 7 are discussed.

2 Trellis termination methods - old and new

Trellis termination that makes the end state of the encoder equal to the initial state is widely used for turbo coding. In 3GPP Rel.6 [12], the termination is performed by taking the 3 tail bits from the shift register feedback after all information bits are encoded. This method causes several problems, namely, 1) rate loss (or energy increase) and 2) non-uniform protection since the terminate bits are not protected by

interleave, 3) complicated rate matching [7], 4) small minimum distance [7 and 13], and etc. Tail-biting technique [14-16] is introduced to prevent these problems. Fig. 1 shows the tail-biting performance gain in these aspects. The performance in Fig.1 (a) is without the rate loss normalization. It shows the gain by uniform protection. The performance in Fig.1 (b) is with the rate loss normalization, which shows more than 0.5dB gain from tail-biting.

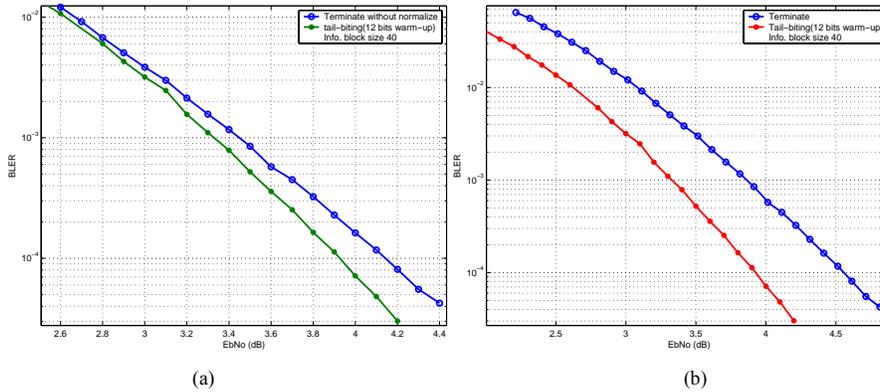


Figure 1 (a) Tail-biting performance gain by uniform protection only; (b) Tail-biting performance gain by both no rate loss and uniform protection

Short block codes are frequently used in transporting voice for 3GPP. In fact, in [17] it shows that most used block size for voice is 39,159 and 244. Therefore, with the none-negligible coding gain tail-biting will benefit the turbo coding for 3GPP LTE.

3 Answers to the three questions

3.1 Tail-biting (circular) encoding

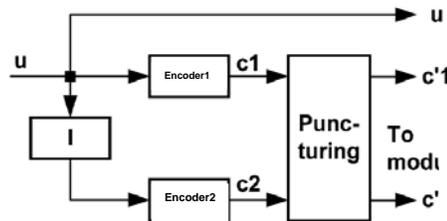


Figure 2, Turbo code.

Definition Let u_0, u_1, \dots, u_{k-1} be the information sequence sent to one of the convolutional encoders of a turbo code such that S_0, S_1, \dots, S_k be the corresponded states sequence. The encoding is tail-biting (or circular) for this given information sequence if and only is $S_0=S_k$.

3.2 Encoding latency

Different to tail-terminating turbo coding in [12], tail-biting needs encoding information bits twice. This will cause extra one frame latency. However, since the encoder is a shift register the following example show that adding extra latency is acceptable or we can modify the encoder to make it up.

Example 1 Suppose the frame size = 5,000 bits, 10 Mbps (0.5 msec. Frame). Let us take the clock = 150 MHz . The encoder can process 1bit/cycle. Then we have

$$time/bit=1/(150e+6)=6.7e-9$$

Tail-terminating encoding: $time/frame = 5000*6.7e-9=33e-6$ seconds

Tail-biting encoding: $time/frame = 67e-6$ seconds

If one wants the tail-biting encoder has the same throughput as terminating encoder, we can do the following:

A) Let encoding process 2 bit/cycle for tail-biting. Then $time/frame = 33e-6$ seconds. The increase of hardware is very small compared to the encoder with 1 bit/cycle

B) Faster clock to 300 MHz since the encoding has bitwise operation only.

3.3 There is a practical pseudo-genie in turbo decoder

It is obvious that if one does not take the advantage of the tail-biting property on decoding, the performance of tail-biting turbo coding will be worse than that of tail-terminating turbo coding. In the following we will explain how to apply tail-biting property to the decoding without extra hardware and latency cost.

3.3.1 Serial decoding with slide window technique

In order to save the decoder area, it is well known in the industry that slide window technique [18] need to be used. In the slide-window decoding, the turbo block is divided into several windows, see Figure 3. To decode tail-terminating turbo code, the start values for computing the forward state metric α are given by previous windows, except the first window. Since the initial state is known, the start values (which equivalent to the probability of states) of the first window can be consider given. Thus, the computation starts from window 1. However, the starting value for backward state metric β is never known. To solve this problem, a *warm-up* (or cold start) procedure (blue color in Fig.3) has to be added so that the start value can be estimated. One can consider this warm-up as a pseudo-genie. This estimate use the bit metric of the next window, except the one for the last window (light blue in Fig. 3). For the last window, the warm up is start from the first window since the final state is equal to the initial state. This procedure is based on “*the fact that the VA can start cold in any state at any time; initially, the state metrics generated are nearly worthless, but after a few constraint lengths, the set of state metrics are as reliable as if the process had been started at the initial (or final) node* [18].” In fact this will not cost any additional hardware since the warm-up hardware is the same hardware used in the window and there is no need to store the values (except the last one which is the start value of the current window) obtained by warm-up.

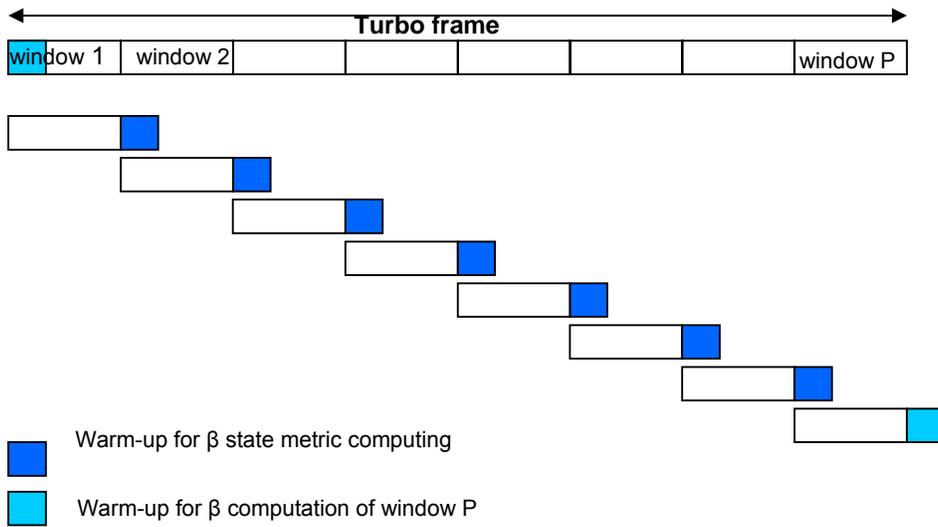


Figure 3 Tail-Terminating decoding with slide windows

To decoding tail-biting turbo code (see Fig.4), the only difference is on the first window which need cold start from the last window since the last state is equal to the first state given by tail-biting. This one additional procedure uses the same hardware for α computation, and it uses the same time as cold state of β . Therefore, there is no extra hardware and extra latency. Our simulation results in the following sections are obtained by using this technique.

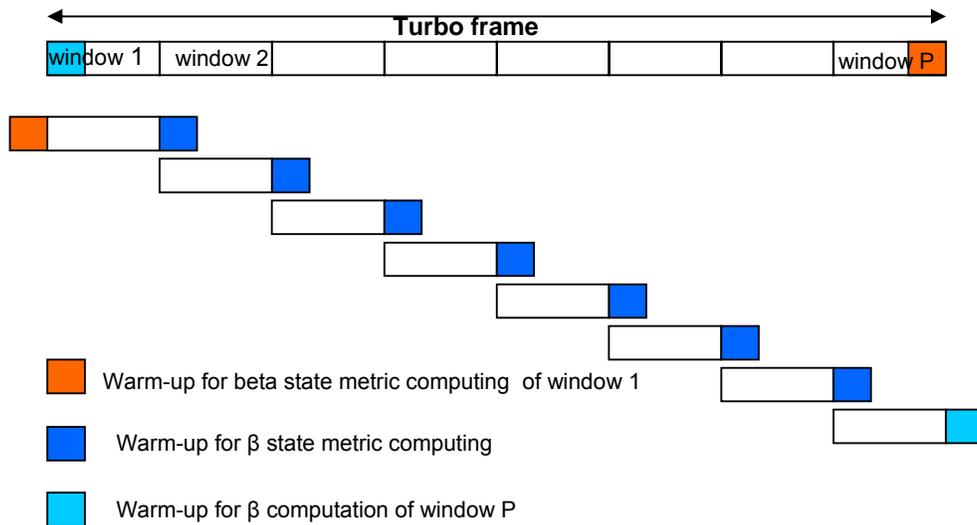


Figure 4 Tail-biting decoding with slide windows

3.3.2 Parallel decoding

The difference of parallel decoding to the slide-window serial decoding is that all α and β computation need warm up (see Fig.5 and Fig.6) since all P processors starts at the same time.

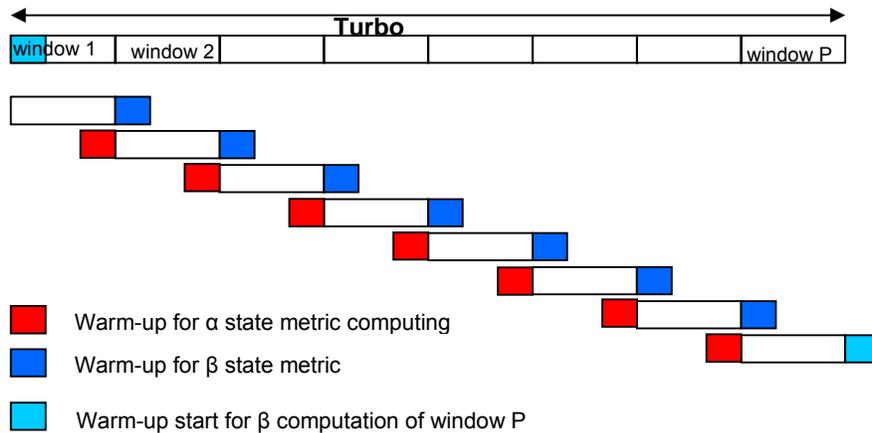


Figure 5 Tail-terminating parallel decoding

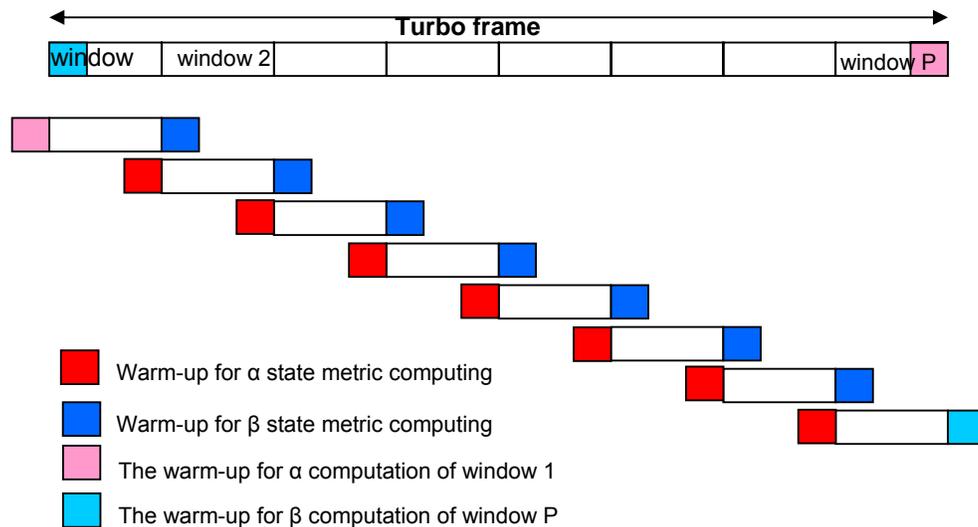


Figure 6 Tail-biting parallel decoding

The only difference between tail-biting and tail-terminating parallel decoding is the warm-up for α of the first window, which can be done from the last window (pink color in Fig.6) thanks to the tail-biting. Obviously, there is no need to add extra hardware and latency.

3.4 Performance gain on tail-biting

In the following, the performance comparisons are given between 12 bits tail-terminating technique and tail-biting technique for rate 1/3 turbo code using QPP

interleave [19] on AWGN channel. The plots are normalized with 12-bits tail-terminating. Same as in [20-21], the normalized plot effectively scales the Eb/No by $10\log_{10}[(3L+12)/(3L)]$ dB. We note that all the simulation results are obtained using the practical state metric warm-up. A 12 bits warm-up period is used for the block size less than or equal to 128. Block size larger than 128 will have more benefits by using 20 bits warm-up period.

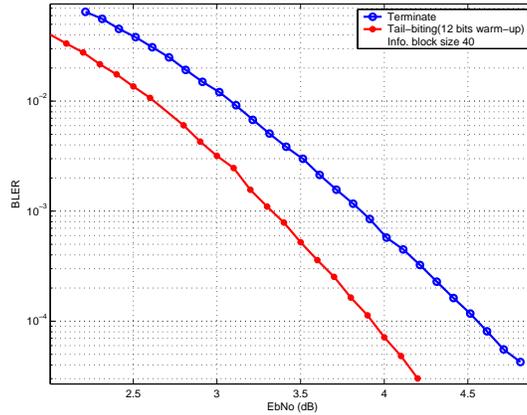


Figure 7 Information block size 40 (0.5dB gain)

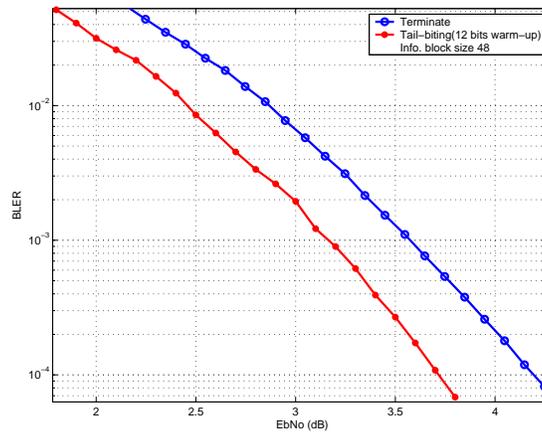


Figure 8 Information block size 48 (0.5dB gain)

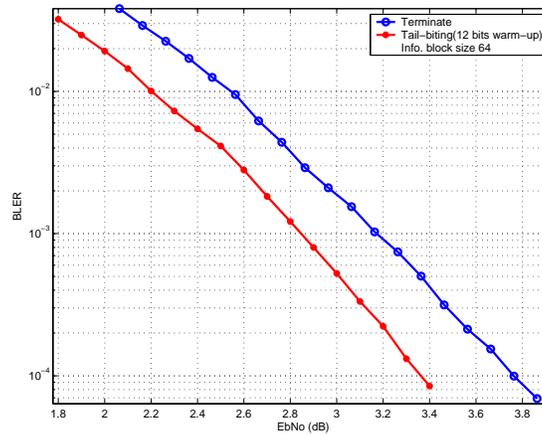


Figure 9 Information block size 64 (0.4dB gain)

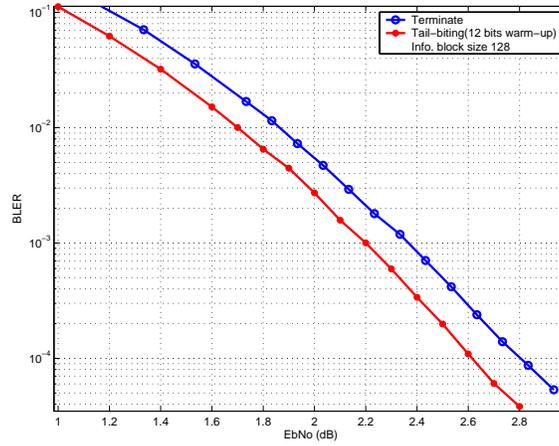


Figure 10 Information block size 128 (0.2 dB gain)

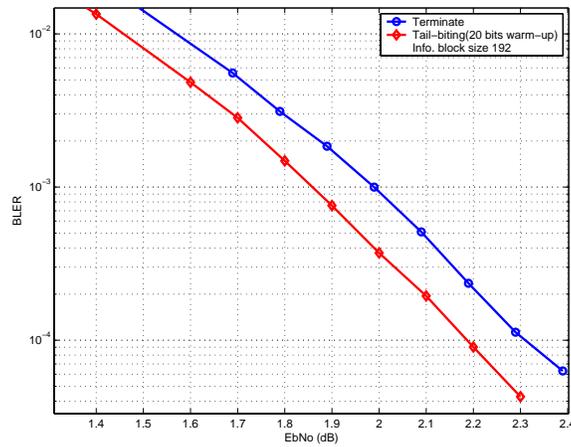


Figure 11 Information block size 192 (>0.1dB gain)

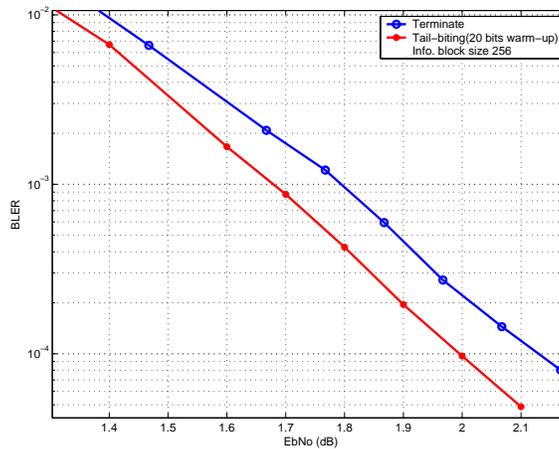


Figure 12 Information block size 256 (>0.1dB gain)

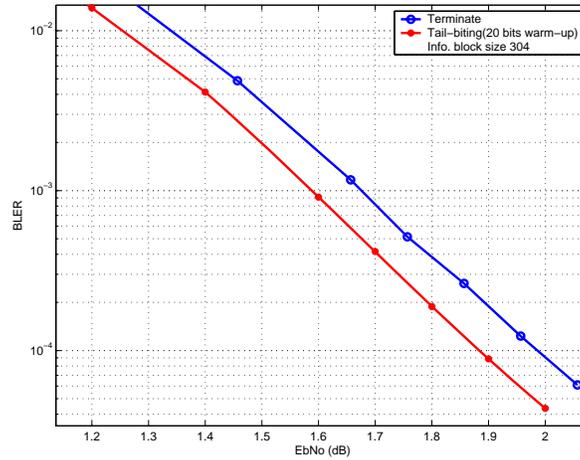


Figure 13 Information block size 304 (0.1dB gain)

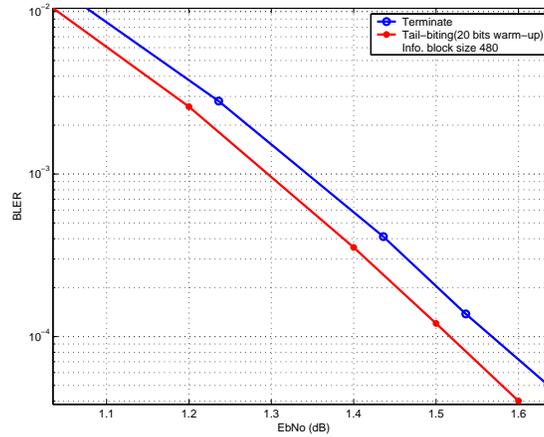


Figure 14 Information block size 480 (>0.05dB gain)

4 Applying tail-biting on 3GPP LTE turbo codes

It is shown in [6] that Rel.6 turbo encoder is not tail-biting encoder for 1/7 of all possible information sequences. On the other hand, the channel coding system of 3GPP LTE has to support arbitrary number of information bits ranged from 40 to 6144 (or 8192). In [7], a necessary and sufficient condition is given for a tail-biting turbo code. Using this condition, possible approaches of tail-biting system for 3GPP LTE are discussed in this section. With these methods, both QPP interleave lists, i.e. Table 2 and Table 3 in [19] can be used for tail-biting.

4.1 Necessary and sufficient condition on tail-biting encoding

In [15], a sufficient condition is given for an encoder being tail-biting for any information sequence with a given block size. In [22], it is proved that this condition is also necessary for an encoder with minimal degree (i.e. the number of states cannot be reduced). To state this necessary and sufficient condition, the definition of state-space realization of convolutional encoder from [15 and 23-25] is needed.

Consider a rate k_0/n_0 convolutional encoder of degree m , let the input sequence be $\mathbf{u} = (u_0, \dots, u_{N-1})$ where $u_i = (u_{i,k_0-1}, \dots, u_{i,0})$ and the output sequence be $\mathbf{x} = (x_0, \dots, x_{N-1})$ where $x_i = (x_{i,n_0-1}, \dots, x_{i,0})$. Moreover, let $\mathbf{S}_t = (s_{m-1}^{(t)}, \dots, s_0^{(t)})$ be the encoding state at time t . Then there exists an m by m matrix A , m by k_0 matrix B , n_0 by m matrix C and k_0 by n_0 matrix D , which is called *state-space realization* of the encoder, such that

$$\mathbf{S}_t^T = (s_{m-1}^{(t)}, \dots, s_0^{(t)})^T = A(s_{m-1}^{(t-1)}, \dots, s_0^{(t-1)})^T + B\mathbf{u}_t^T = A\mathbf{S}_{t-1}^T + B\mathbf{u}_t^T$$

and $x_t^T = C\mathbf{S}_{t-1}^T + D\mathbf{u}_t^T$. The generate matrix of this convolutional encoder is

$$C(A, B, C, D) = G(x) = D + C(x^{-1}I_m - A)^{-1}B$$

Consider the convolutional encoder in Rel.6 turbo code depicted in Fig.15. The encoder has minimal degree 3 and $s_0^t = s_2^{t-1} + s_1^{t-1} + u_t$, $s_1^t = s_0^{t-1}$, $s_2^t = s_1^{t-1}$. Its 4 state matrices are,

$$A_{\text{Rel.6}} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}, B_{\text{Rel.6}} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, C_{\text{Rel.6}} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}, D_{\text{Rel.6}} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad (*)$$

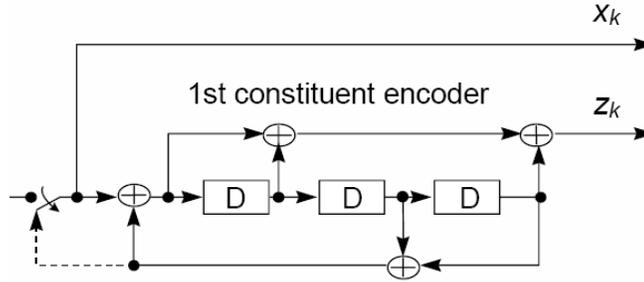


Figure 15 Constituent convolutional encoder of Rel.6

Theorem 1[22] Let the matrices (A,B,C,D) be the state-space realization of a convolutional encoder with minimal degree m . This encoder is tail-biting for any information sequence of block size $N \geq m$ if and only if $A^N + I_m$ is invertible.

It was stated in [4] that turbo encoder in Rel.6 may not offer tail-biting for information sequence of size multiple of 7. In fact, we can prove that

Theorem 2 [22] For any recursive convolution encoder of minimal degree m there exists a positive integer P such that this encoder gives no tail-biting termination for some information sequences of size tP ($t > 0$).

Proposition 1 [22] Let $A = A_{\text{Rel.6}}$ be the matrix for the constituent encoder of the turbo code in Rel.6 (see (*)). Then for any positive integer $n=7q+i$, $0 \leq i \leq 6$,

$$A^n + I_m = A^i + I_m = \begin{cases} 0 & i = 0 \\ \text{invertible} & i \neq 0 \end{cases}$$

Thus, by Theorem 1, any information sequence of size not a multiple of 7 can be tail-biting encoded by the turbo encoder in Rel.6.

4.2 Tail-biting for 3GPP LTE turbo code with QPP interleaves

Let (A,B,C,D) in (*) be the state space realization of the 8 states convolutional encoder used in turbo code of Rel.6. Then

I) Pre-compute the followings states for $i=1,2,3,4,5,6$

$$S_{i,1} = (A^i + I_3)^{-1} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, S_{i,2} = (A^i + I_3)^{-1} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, S_{i,3} = (A^i + I_3)^{-1} \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix},$$

$$S_{i,4} = (A^i + I_3)^{-1} \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, S_{i,5} = (A^i + I_3)^{-1} \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}, S_{i,6} = (A^i + I_3)^{-1} \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}, S_{i,7} = (A^i + I_3)^{-1} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

II: Pre-store the above 42 index-state pairs as a look-up-table $LT(i, b_{(2)}) = S_{i,b}$, where $b=1,2,3,4,5,6,7$ and $b_{(2)}$ is the 3 bits binary representation of b . Moreover, let $LT(i,0)=0$ state.

To handle any information block sizes from 40 to 6144 using the 160 (or 188) QPP interleaves listed in Table 2 and Table 3 of [19], two major methods are available, namely pruning and puncturing. In the following the tail-biting applications are given for these two cases.

4.2.1 Tail-biting encoding using pruning technique for interleave

Pruning[12]: According to the pruning technique in Rel.6, if the information block size L is not in the QPP list, than take the smallest size N in the list such that $N > L$. The dummy bits are padded for $k=L, L+1, \dots, N-1$. Using size N QPP to interleave the information sequence with the padded. After the interleaving, dummy bits are pruned away from the output of the interleaved bits.

With pruning, both convolutional encoders of the turbo code encode the information bits without padded dummy bits. In this case, one may use the following tail-biting method. Consider information block u_0, u_1, \dots, u_{n-1} .

1. Compute $m = n \bmod 7$. If $m=0$, pad one more symbol $u_k=0$ and let $L=n+1$ and $M=1$, otherwise let $L=n$ and $M=m$. Then the padded sequence becomes

$$u_0, u_1, \dots, u_{n-1}, u_{L-1}$$

2. With 0 state encoding information symbols u_0, u_1, \dots, u_{L-1} (or interleaved symbols $u_{\pi(0)}, \dots, u_{\pi(L-1)}$) to find the final state S_{final} (or S'_{final} (do not store the encoded symbols). Then use look-up table to find the initial state $S_0 = LT(M, S_{\text{final}})$ (or $S'_0 = LT(M, S'_{\text{final}})$)

3. Use S_0 (or S'_0) as initial state to encode u_0, u_1, \dots, u_{L-1} (or $u_{\pi(0)}, \dots, u_{\pi(L-1)}$)

□

4.2.2 Tail-biting encoding using shortening and puncturing technique for interleave

Shortening and Puncturing [26 and 27]: According to the methods proposed in [26-27], if the information block size L is not in the QPP list, take the smallest size N in the list such that $N > L$ and pad $N-L$ 0 bit at the beginning of the information sequence. Then, turbo encoder will encode the padded information sequence and its QPP-interleaved sequence. After that the padded 0 systematic bits will be shortened. For the check bits corresponded to the shortened bits, there are two methods proposed, namely a) [26] puncture only the check bits (which are also 0) obtained from none interleaved information sequence. The check bits of the padded bits from interleaved information sequence will not be punctured. This will cause some rate loss but not the performance loss; b) [26-27] puncture check bits obtained from both none interleaved and interleaved information sequence. This will not cause rate loss but it will cause a performance loss.

4.2.2.1 Tail-biting encoding without adding an extra bits outside the interleave

Since the sizes of the interleaves in Table 2 of the QPP list in [19] are not multiple of 7, one may use the following tail-biting method. Consider information block u_0, u_1, \dots, u_{n-1} .

1. Find the smallest L in the list such that $L \geq n$ and pad $L-n$ 0 bit at the beginning of the information sequence. Then the padded sequence becomes

$$u_0, u_1, \dots, u_{n-1}, u_n, \dots, u_{L-1}$$

2. With 0 state encoding information symbols u_0, u_1, \dots, u_{L-1} (or interleaved symbols $u_{\pi(0)}, \dots, u_{\pi(L-1)}$) to find the final state S_{final} (or S'_{final} (do not store the encoded symbols). Then use look-up table to find the initial state $S_0 = \text{LT}(M, S_{\text{final}})$ (or $S'_0 = \text{LT}(M, S'_{\text{final}})$)
3. Use S_0 (or S'_0) as initial state to encode u_0, u_1, \dots, u_{L-1} (or $u_{\pi(0)}, \dots, u_{\pi(L-1)}$)

□

4.2.2.2 Tail-biting encoding with adding an extra bit outside the interleave

If the QPP list in Table 3 of [19] is used i.e. allows multiple of 7 interleave size, then the following tail-biting method can be applied.

1. Find the smallest L' in the list such that $L' \geq n$. and pad $L'-n$ 0 bits at the beginning of the information sequence. Then the padded sequence becomes

$$u_0, u_1, \dots, u_{n-1}, u_n, \dots, u_{L'-1}$$

2. Compute $m = L' \bmod 7$. If $m=0$, pad one 0 bit to both none interleaved and interleaved information sequence and let $L=L'+1$ and $M=1$, otherwise, no padding and let $L=L'$ and $M=m$.

3. With 0 state encoding information symbols u_0, u_1, \dots, u_{L-1} (or interleaved symbols $u_{\pi(0)}, \dots, u_{\pi(L-1)}$) to find the final state S_{final} (or S'_{final} (do not store the encoded symbols). Then use look-up table to find the initial state $S_0 = \text{LT}(M, S_{\text{final}})$ (or $S'_0 = \text{LT}(M, S'_{\text{final}})$).
4. Use S_0 (or S'_0) as initial state to encode u_0, u_1, \dots, u_{L-1} (or $u_{\pi(0)}, \dots, u_{\pi(L-1)}$).

When the information block size is multiple of 7, this method will transmit 4 extra bits outside the interleaved block.

5 References

- [1] France Telecom, GET, Enhancement of Rel. 6 Turbo Code, 3GPP TSG RAN WG1#43 R1-051310.
- [2] Motorola, "Convolution Code Structure for E-AGCH," 3GPP TSG-RAN2 #46 R1-050164.
- [3] Motorola, France Telecom, GET, Orange, "EUTRA FEC Enhancements," 3GPP TSG RAN WG1#44bis, R1-061050.
- [4] HighDimension Ltd, "System impact of Rel'6 turbo coding tail-bits removal," R1-062157.
- [5] Motorola, "E-UTRA DL Coding Performance for Control Channel," 3GPP TSG RAN1 #47-bis, R1-70066.
- [6] Broadcom, "Tail-biting encoding for 3GPP LTE turbo code of arbitrary number of information bits," 3GPP TSG RAN WG1 #47 R1-063242.
- [7] Motorola, "Eliminating tail bits in LTE channel coding," 3GPP TSG RAN WG1 #47bis R1-070058.
- [8] DVB-RCS and DVB-RCT.
- [9] IEEE 802.16-2004, "IEEE Standard for Local and Metropolitan Area Networks Part 16: Air Interface for Fixed Broadband Wireless Access Networks".
- [10] DVB-RCS, "Interaction Channel for Satellite Distribution Systems", ETSI EN 301 790, v1.2.2, pp. 21-24, December 2000.
- [11] DVB-RCT, "Interaction Channel for Digital Terrestrial Television", ETSI EN 301 958, v1.1.1, pp 28-30, August 2001.
- [12] 3GPP TS 25.212 V6.8.0 (2006-06)
- [13] E. Rosnes and O. Ytrehus, "Improved algorithms for the determination of turbo-code weight distributions," *IEEE Transactions on Communications*, Vol 53, No. 1, pp. 20 – 26, Jan. 2005.
- [14] J. B. Anderson and S. M. Hladik, "Tailbiting MAP Decoders," *IEEE JSAC*, VOL. 16, NO. 2, Feb. 1998

- [15] C. Weiß, C. Bettstetter and S. Riedel, “Code Construction and Decoding of Parallel Concatenated Tail-Biting Codes,” *IEEE Trans on Information Theory*, Vol.47, No.1, pp.366-386, Jan. 2001
 - [16] C. Berrou, Y. Saouter, C. Douillard, S. Kerouédan, and M. Jézéquel, “Designing good permutations for turbo codes: towards a single model,” 2004 *IEEE International Conference on Communications (ICC)*, Vol.: 1, pp: 341-345, 20-24 June 2004.
 - [17] 3GPP TS 25.101 V6.14.0 (2006-12), A.4
 - [18] A. J. Viterbi, “An Intuitive Justification and a Simplified Implementation of the MAP Decoder for Convolutional Codes,” *IEEE JSAC*, VOL. 16, NO. 2, pp. 260-264 FEBRUARY 1998.
 - [19] Ericsson and Motorola, “QPP interleaver parameters,” 3GPP TSG RAN WG1 #47bis R1- 070484.
 - [20] Texas Instruments, “Turbo Code Tail Methods,” 3GPP TSG RAN WG1 #47 bis R1-070253
 - [21] Nokia, “Performance of tail-biting turbo codes,” 3GPP TSG RAN WG1 #47bis R1-070376
 - [22] Broadcom, “Tail-biting encoding for 3GPP LTE turbo codes of arbitrary number of information bits and performance comparison,” 3GPP TSG RAN WG1 #47bis R1-070544
 - [23] J. L. Massey and M. K. Sain, “Codes, automata, and continuous systems: Explicit interconnections,” *IEEE Trans. Automat. Contr.*, vol. AC-12, no. 6, pp. 644–650, 1967.
 - [24] R.J. McEliece. The Algebraic Theory of Convolutional Codes. In *Handbook of Coding Theory*, R. Brualdi, W.C. Human and V. Pless (eds.). Elsevier Science Publishers, Amsterdam, The Netherlands, 1998
 - [25] J. Rosenthal, and E. V. York, “BCH Convolutional Codes,” *IEEE Trans. on Inform. Theory*, Vol.45, No.6, PP.1833-1842, Sept. 1999
 - [26] Motorola, Contention-free Interleaver designs for LTE Turbo Codes, 3GPP TSG RAN WG1 #47bis R1-070054
 - [27] Ericsson, Performance Comparison of LTE Turbo Internal Interleaver Proposals, 3GPP TSG-RAN WG1 #47bis R1-070464
-