TSG-RAN Working Group 1(Radio) meeting #4
Yokohama, Japan 19-20, April 1999

TSGR1#4(99) 338

Source: **LGIC**

Title: **Puncturing Algorithm for Turbo Code**

---

1. Introduction

The conventional puncturing algorithm was designed on the basis of "uniform puncturing of all coded bits"[1]. This concept is appropriate for convolutional code and confirms optimal performance in the processing of rate matching for the transport channel multiplexing. But, the conventional puncturing algorithm is not optimal for turbo code and a more efficient design of puncturing algorithm could be possible.

The well known design principles of puncturing for turbo code are as below.

- minimize puncturing of systematic bits
- provide approximately equal puncturing of parity bits of the two encoders

We present the novel puncturing algorithm for turbo code which has a good performance gain over the conventional algorithm.

2. Design of Puncturing Algorithm for Turbo Code

The design rules imposed on the puncturing algorithm for turbo code are as below.

Code symbol(word) based puncturing process
In the puncturing algorithm for convolutional code, every code bit has equal importance in sense of BER performance and uniform puncturing over all code bits is appropriate. So a puncturing algorithm performed by code bit-based processing is reasonable. But, because code bits of turbo code have unequal importance depending on whether it is a systematic or parity bit, bit based puncturing is no more suitable and it is more efficient to use a symbol based approach.

Preventing puncturing of systematic bits
Systematic bits of turbo code are more important than parity bits which means that puncturing of one systematic bit results in more performance degradation than a parity bit. In the proposed algorithm, puncturing of systematic bits is completely excluded and thus puncturing is only done for parity bits

Preventing puncturing of termination bits
Termination bits are used to ensure that the ending states of the encoder and the decoder are zero state. Termination bits of turbo code are more important than other encoded bits because non termination of turbo code may result in large performance loss. So, in the proposed algorithm, we do no puncturing of termination bits.

Alternative puncturing of parity bits of two encoders
In order to maximize the BER performance of turbo code, the coding strength of each RSC code must be balanced. Balanced puncturing of parity bits between the two encoders means balanced puncuring of each RSC code, thus alternative puncturing is a good method to acquire this.

First puncturing of parity bits of the second encoder
When alternative puncturing starts, we can choose a first puncturing point in the first parity encoder bits or the second parity encoder bits. If initial offset is appropriately selected, first puncturing of parity bits of the second encoder and

alternative processing provide the same puncturing pattern as the optimal pattern when code rate is 1/2. So, in the proposed algorithm, first puncturing is done in the parity bits of the second encoder.

3. Proposed Puncturing Algorithm for Turbo Code

The proposed puncturing algorithm for turbo code is as below.

Let's denote:

- N = information bit block size(systematic code block size)

N is also the size of code symbol(word) block size. The puncturing algorithm is performed based on the "while" loop of the size N.

- $S_N = \{N_1, N_2, \ldots, N_L\}$ = ordered set (in ascending order from left to right) of allowed number of bits per block
- N_C = number of bits per matching block = $3N$
- $S_0 = \{d_1, d_2, \ldots, d_{N_C}\}$ = set of N_C data bits

We can define $I_1, I_2, \ldots, I_N$ as systematic bits, $C_{1,1}, C_{1,2}, \ldots, C_{1,N}$ as parity bits 1, and $C_{2,1}, C_{2,2}, \ldots, C_{2,N}$ as parity bits 2. This notation may be understood by figure 1.

It is noted that $S_0 = \{d_1, d_2, \ldots, d_{N_C}\} = \{I_1, C_{1,1}, C_{2,1}, I_2, C_{1,2}, C_{2,2}, \ldots, I_N, C_{1,N}, C_{2,N}\}$.

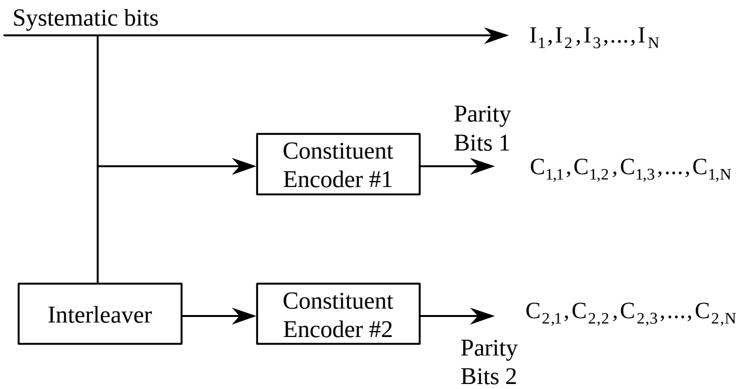

Figure 1. The notation of turbo encoded bits

The puncturing rule is as follows:

find N_i and N_{i+1} so that $N_i \le N_C < N_{i+1}$

$if\ (\frac{1}{3} < \frac{N_i}{N_C} < \frac{2}{3})$

$y = 2*N-N_i$

$e = N$

$m = 1$

```
count = 0
do while m <= N
        e = e – 2 * y
        if e <= 0 then
                puncture bit C_{1,m}, C_{2,m} from set S0
                e = e + 2*NC
        else if m%2==0
                puncture bit C_{1,m} from set S0
        else if m%2==1
                puncture bit C_{2,m} from set S0
        end if
        m = m + 1
end do
```

$if\ (\frac{2}{3} \le \frac{N_i}{N_C} < 1)$

```
    y = 3*N-Ni
    e = N
    m = 1
    count = 0
    do while m <= N
        e = e – 2 * y
        if e <= 0 then
            if count%2 ==0
                puncture bit C_{2,m} from set S0
            endif
            if count%2 ==1
                puncture bit C_{1,m} from set S0
            endif
            e = e + 2*NC
            count = count + 1
        end if
        m = m + 1
    end do
endif
```

The above algorithm follows the turbo code puncturing rules imposed.

In the above algorithm, $N_i / N_C = 2/3$ means that the code rate is 1/2 and the optimal puncturing pattern of code rate 1/2 is as shown figure 2. The proposed puncturing algorithm provides the same puncturing pattern as shown in figure 2. when $N_i / N_C = 2/3$ and so is transparent to optimal puncturing pattern of 1/2.

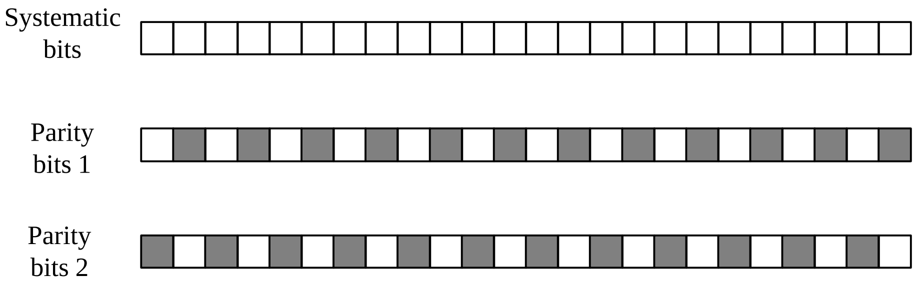

Figure 2. puncturing pattern of 1/2 turbo code

4. Simulation Results

Simulation conditions are as below.

- Interleaver depth is 640
- Internal interleaver is CDI.
- Constraint length of the constituent code is 4.
- Conventional termination method is applied.
- A full MAP with floating point implementation is used for the decoding of the constituent encoders
- Iteration number is 4.
- At a BER of 10^{-5}, at least 100 frame errors have to be counted
- Simulations are carried out in an AWGN channel
- Conventional puncturing algorithm parameters for comparison purpose are performed

Case 1 : 128 puncturing from 1/2 turbo code

$$\text{Effective code rate} = \frac{640}{640*2+12-128} = \frac{640}{1164} = 0.55$$

Case 2 : 384 puncturing from 1/3 turbo code

$$\text{Effective code rate} = \frac{640}{640*3+12-384} = \frac{640}{1548} = 0.41$$

The simulation results shows that the proposed puncturing algorithm has a coding gain of about 0.04 dB at a BER of 10^{-5} and 0.015 dB at a FER of 10^{-3} in Case 1. In Case 2, the proposed puncturing algorithm has a coding gain of about 0.1 dB at a BER of 10^{-5} and 0.07 dB at a FER of 10^{-3}.

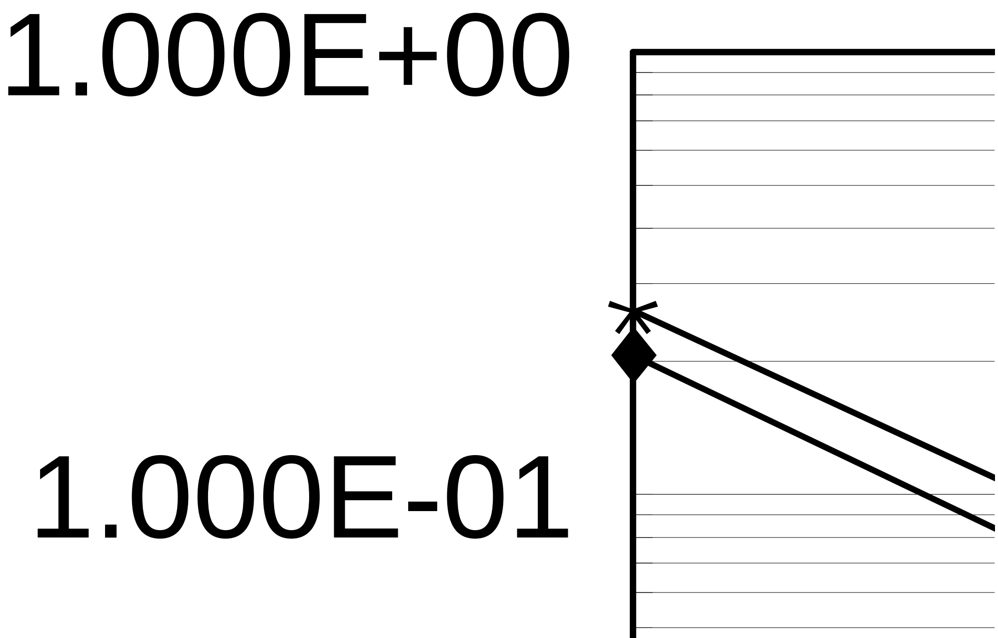

Figure 3. Frame and bit error rate comparison of 128 punctured 1/2 turbo code (Case 1)

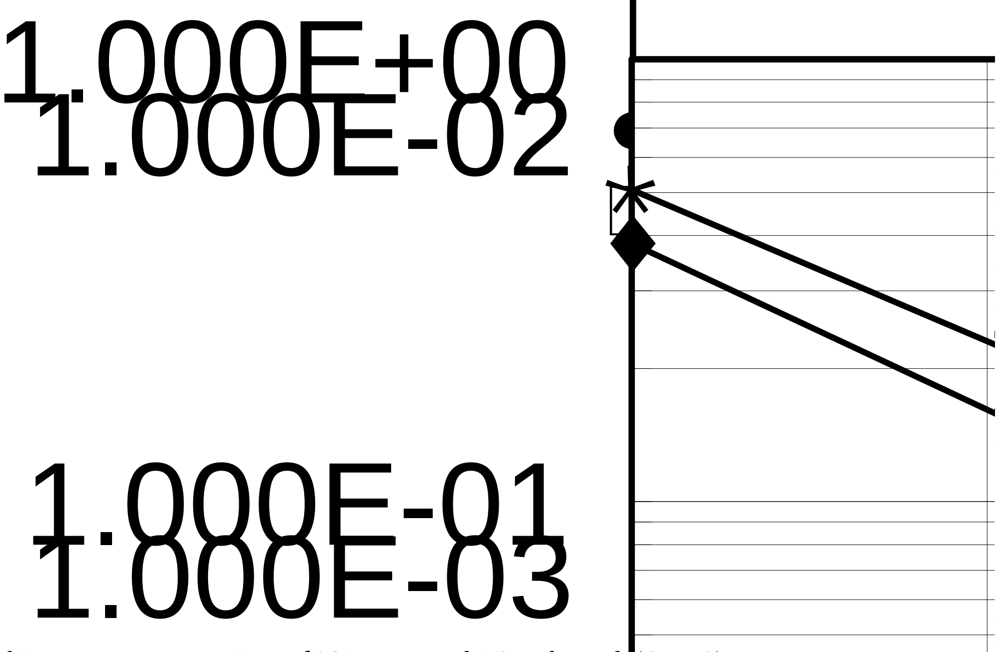

Figure 4. Frame and bit error rate comparison of 384 punctured 1/3 turbo code (Case 2)

5. Conclusion

To enhance the performance of the conventional puncturing algorithm for turbo code, a novel puncturing algorithm is proposed in which puncturing is only done alternatively between two encoder parity bits. The proposed puncturing algorithm is simple and has superior performance over the conventional algorithm.

6. Reference

1. 3GPP TSG RAN WG1 Multiplexing and channel coding (FDD) S.12 v1.0.1(1999.03)