이중 입력 터보 코드를 위한 저지연 부호화 알고리즘

박숙민, 곽재영, 이귀로
삼성전자, 한국과학기술원

Low Latency Encoding Algorithm for Duo-Binary Turbo Codes with Tail Biting Trellises

Sook Min Park, Jaeyoung Kwak, Kwyro Lee
Samsung Electronics, KAIST

Abstract - The low latency encoder for high data rate duo-binary turbo codes with tail biting trellises is considered. Encoder hardware architecture is proposed using inherent encoding property of duo-binary turbo codes. And we showed that half of execution time as well as the energy can be reduced with the proposed architecture.

1. Introduction

We focus on the duo-binary turbo encoder structure with tail biting trellises. The general structure of duo-binary turbo codes is same as binary one that two recursive systematic convolutional(RSC) component codes are parallel concatenated via an interleaver. For terminating each component RSC code, all the codewords are encoded with the trellis starting at state zero. After N information bits have been encoded, at the end of the trellis, m termination bits have to be appended to the codeword to terminate the trellis to state zero. Here, m is the number of the shift registers of the component RSC code. To make each shift registers zero state, recursive feedback of RSC code goes directly through the encoder. Therefore, the termination bits do not have any information and a rate loss occurs by the termination. To avoid this rate loss, tail biting RSC codes is proposed instead of terminated RSC codes [1][2]. In tail biting RSC code, the trellis does not start and end at zero state. However, it is required that the initial state of a codeword is the same as the ending state. So, the code rate is higher than a terminated RSC code by eliminating the additional tail bits for termination. This scheme is accepted in the applications for high data rate transmission such as mobile WiMAX (IEEE 802.16e [3]). Although tail biting method prevents data rate loss for additional termination bits, it makes encoding time twice to find an initial state which is same as the last state. To improve encoding delay, we focus on the calculation to meet the tail biting boundary condition, S_0=S_N, and propose the low latency encoder algorithm for high data rate applications.

In the following section the principles of duo-binary turbo codes with tail biting trellises are described. Then we propose low late ncy encoder algorithm for the duo-binary turbo codes in the appl ication of mobile WiMAX(IEEE 802.16e). And low latency encode r algorithm is also represented to an efficient hardware architectu re in the aspect of latency and power.

2. Low latency encoding algorithm for 802.16e duo-binary turbo codes

In this section we propose the low latency encoding algorithm u sing the inherent characteristic of 802.16e encoder structure. First, we examine the inherent characteristic of 802.16e encoder structu re. Then the low latency encoding algorithm and its hardware ar chitecture will be described.

2.1 Characteristics of 802.16e duo-binary encoder with tail biting trellises

We will consider the duo-binary RSC encoder of Figure 1. The sequence of $\underline{u}=(u_1, u_2, \cdots, u_{N-1})$ denotes information symbol sequence and each information symbol consists of bit couple of $u_n=(u_n^0, u_n^1)$. Hence, the information sequence can be represented as follows:

$$\underline{u}=(u_0^0,u_0^1,u_1^0,u_1^1,\cdots,u_n^0,u_n^1,\cdots,u_{N-1}^0,u_{N-1}^1) \tag{1}$$

Here N is the number of symbols in one frame, which is fed to each RSC encoder. The state space representation of the encoder is represented as follows:

$$S_{n+1}=AS_n+Bu_n^T \tag{2}$$

$$v_n^T=CS_n+Du_n^T \tag{3}$$

The complete solution of can be calculated by the superposition of the zero input solution $S_n^{[zi]}=A^nS_0$ and the zero state solution $S_n^{[zs]}=\sum_{i=0}^{n-1}A^{(n-1-i)}Bu_i^T$:

$$S_n=S_n^{[zi]}+S_n^{[zs]}=A^nS_0+\sum_{i=0}^{n-1}A^{(n-1-i)}Bu_i^T \tag{4}$$

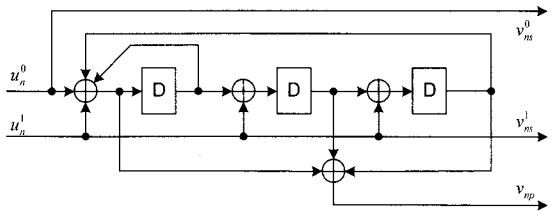
The right part in (4) consists of the initial state calculation and the zero-state calculation. By calculating S_N=S_0 in (4) we can calculate the initial state which is the same as the last state. So, (4) becomes our final calculation to meet the basic tail biting encoding property.

$$(A^n+I_m)S_0=\sum_{i=0}^{n-1}A^{(n-1-i)}Bu_i^T \tag{5}$$

where I_m denotes the (mxm) identity matrix. In the equations of (2) and (3), we can extract A,B,C and D matrices from 802.16e encoder structure of Figure 1 as follows:,

$$A=\begin{pmatrix}1&0&1\\1&0&0\\0&1&0\end{pmatrix}\ B=\begin{pmatrix}1&1\\0&1\\0&1\end{pmatrix},\ C=\begin{pmatrix}0&0&0\\0&0&0\\1&1&0\end{pmatrix},\ D=\begin{pmatrix}1&0\\0&1\\1&1\end{pmatrix} \tag{6}$$

We can find an important clue in the characteristic of A matrix to reduce the encoding latency for tail biting method. Multiplicati on of matrix A repeats every 7 times, since $A^7=I$. Here I is th e identical matrix. Using above property, the low latency algorith m can be devised.



<Figure 1> Constituent RSC Encoder for 802.16e Duo-Binary Turbo Codes

2.2 Low latency encoding algorithm for 802.16e duo-bi nary turbo codes

To find initial state which satisfy the tail biting boundary condition, i.e. s_N=s_0, we should find zero state solution of right part in Equation (5). Zero state solution for given data sequence is described in the Equation (7)

$$S_N^{[zs]}=\sum_{i=0}^{N-1}A^{(N-1-i)}Bu_i^T$$
$$=A^{N-1}Bu_0+A^{N-2}Bu_1+\cdots+ABu_{N-2}+Bu_{N-1} \tag{7}$$

In this formula, we can see that matrix calculations should be pe rformed per every given data at each time. Once $S_N^{[zs]}$ is compute

d from $A^7 = I$. Equation (7), we can easily find initial state in E quation (5) by using lookup table. Since the conventional tail biti ng method accumulates the input sequence serially, it executes m atrix calculation every cycle. Additionally it consumes cycle time as much as input sequence frame length. We utilize the character istic of A matrix to reduce the latency and power consumption. Equation (7) can be rewritten as (8).

$$\begin{aligned} S_N^{[zs]} &= A^{(N-1)\%7}B(u_0 + u_7 + u_{14} + \cdots) \\ &+ A^{(N-2)\%7}B(u_1 + u_8 + u_{15} + \cdots) \\ &+ \cdots \\ &+ A^{(N-6)\%7}B(u_5 + u_{12} + u_{19} + \cdots) \\ &+ A^{(N-7)\%7}B(u_6 + u_{13} + u_{21} + \cdots) \end{aligned} \quad (8)$$

In Equation (8), input sequence, $\underline{u}$ is classified into 7 groups, which have same remainder value of n modulo 7. Each group is multiplied by $A^{(N-n\%7-1)\%7}$. Here $A^{(N-n\%7-1)\%7}$ can be described in one o f 7 combinations, i.e. B, AB, A^2B, A^3B, A^4B, A^5B, and A^6B usin g the property of $A^7=I$. We carry out the input accumulation first followed by matrix calculation. In conventional tail biting procedu re, every input symbol needs matrix calculation of $A^{(N-1-n)}B$. This architecture not only saves the unnecessary matrix calculation for every receiving data but also reduce the calculation time if we u se parallel input. Equation (8) can be implemented as a simple ha rdware. It can be separated into two independent parts. One is th e accumulation part of input sequence, and the other is the matri x calculation part with A, B matrices and accumulated input dat a. At first, the input sequences are accumulated in accumulation part. We considered parallel calculation that 16 or 32 data bits pa ss to encoder. We denote this bit width as W. This input sequen ce generates 7 accumulated data for next matrix calculation. The matrix calculation part executes matrix calculation with these 7 a ccumulated data and already calculated matrix, which are B, AB, A^2B, A^3B, A^4B, A^5B, and A^6B. Since we already know A and B matrices from encoder structure, these 7 values can be calculated prior to encoding process and also implemented in simple hardwa re. Finally the output becomes the final state. In the following se ction, we describe the low latency hardware architecture for com puting zero state solution in Equation (8).

2.3 Hardware Architecture

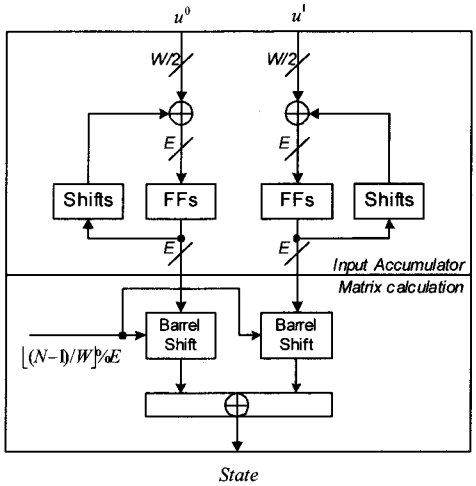
We focus on the calculation of last state which takes time as much as real encoding step. The task of this step is to find last state after all the input data are encoded. The designed hardware is partitioned into two parts, input accumulation and matrix calculation in the (8). Whenever the data are ready input accumulator works to accumulate all the input. After the last input is accumulated, matrix calculation is executed only once. Figure 2 shows the detailed architecture. Here W is the width of parallel input data, and we assumed W=16 in this case. In many case, input data are saved to then memory and these go to the encoder. Many system use the bus width as 16 or 32, so we set 16 as our example. We have 7 matrices combinations(B, AB, A^2B, A^3B, A^4B, A^5B, and A^6B) and 2 bits per input symbol (u_i^0, u_i^1) per each matrix, so we need 14 accumulators. In the case of W=16, first 14 bits are passed to the accumulator and remaining two bits should be EXORed with first two bits, after that these are accumulated. At the next cycle, the latched outputs are shifted one bit and then passed through EXOR block with next 16 incoming input data. As shown in Figure 2, we prepare 14 accumulators but the input bits are 16. We separate and , so 8 input bits go to 7 accumulators each. One input bit is overlapped each, and to align the bit position, accumulated data need to be shifted one bit for next accumulation. These are continued until the end of input data. In this example, we do 1 bit shift, however, the general shift value is $(W/2 - E)$. When the last input data block is accumulated, those values are re-arranged by the barrel shift in the matrix calculation block. As input bit width $W/2$=8 and number of accumulator E=7 are not matching, shift was executed every cycle. This also generates the shifted accumulated output. So we need to reverse back this output. In general, shift value is $(N-1)/W\%E$, where is the biggest integer. Finally, these data are added by final EXOR to make the last state. This EXOR function is pre-defined by the B, AB, A^2B, A^3B, A^4B, A^5B, and A^6B easily. By implementing low latency encoder in real hardware, we can get the benefit of latency and power consumption. Table 1 shows the implementation results. W=16 and W=32 denote number of bits which are calculated at the same time. Implemented hardware can be partitioned into two parts, input accumulation and matrix calculation. Whenever the data are ready, input accumulation works to accumulate all the input. After the last input is accumulated, matrix calculation is executed only once. So, this effect is shown in activation column. The parentheses of fourth and sixth column are normalized performance to original data. When W=16 is used gate count increased 9.2 times. However, the speed is 8 times faster and 51% of energy is enough in comparison to the original case. When W=32, 10.2 times more hardware, but 16 times faster and only 31% of energy is enough to complete the task. The actual hardware for this task is so small comparing to other hardware, such as turbo decoder, this demerit is not so critical compared to its benefit in latency.

<Table 1> Hardware Complexity, Latency and Activation Comparison Table

	Gates (input accu.)	Gates (matrix Cal.)	Gates	latency	Activation
Original			37	2	$37*2*N$
16 bits	148	193	378	1+1/8	$37*N+148(N/8)+193$
32 bits	180	193	410	1+1/16	$37*N+180(N/16)+193$

3. Conclusions

We proposed the low latency turbo encoder architecture for high data rate applications. We can get the benefit of throughput and power efficiency. We found that the encoding time for zero state response was 8 times faster and 16 times faster in case of 16 bits and 32 bits memory output, respectively. The low latency encoder for high data rate duo-binary turbo codes with tail biting trellises is considered. Encoder hardware architecture is proposed using inherent encoding property of duo-binary turbo codes. And we showed that half of execution time as well as the energy can be reduced with the proposed architecture.



<Figure 2> Hardware Architecture for Zero State Solution

[References]

[1] Critian Weiss, Cristian Bettstetter, Sven Riedel and Daniel J. Costello," Turbo decoding with tail-biting trellises," *1998. ISSSE 98. 1998 URSI International Symposium on Signals, Systems, an d Electronics*,pp. 343-348 Oct. 1998.

[2] Christian Weifi, Christian Bettstetter, "Code Construction and Decoding of Parallel Concatenated," *IEEE Transactions on Inform ation Theory*, pp 366-386, Jan. 2001.

[3] IEEE P802.16e/D3-2004 Draft Amendment to IEEE Standard f or Local and metropolitan area networks Part 16: Air Interface a nd Fixed and Mobile Broadband Wireless Access Systrems.