Energy-Efficient Double-Binary Tail-Biting Turbo Decoder Based on Border Metric Encoding

Ji-Hoon Kim and In-Cheol Park
Department of EE, Korea Advanced Institute of Science and Technology (KAIST)
373-1 Guseong-dong, Yuseong-gu, Daejeon 305-701, Republic of Korea
{jhkim, icpark}@ics.kaist.ac.kr

***Abstract*—This paper presents an energy-efficient turbo decoder based on border metric encoding, which is especially suitable for non-binary circular turbo codes. In the proposed method, the size of the branch memory is reduced to half and the dummy calculation is removed at the cost of a small-sized memory that holds encoded border metrics. Due to the small size and infrequent access to the border memory, power consumption for soft-input soft-output (SISO) decoding is reduced by 26.0%. Based on the proposed SISO decoder and the dedicated hardware interleaver, a double-binary tail-biting turbo decoder is designed for WiMAX standard using a 0.18 μm CMOS process and it can support 12.14Mbps at operating frequency of 100MHz.**

I. INTRODUCTION

The turbo code introduced in 1993 is one of the most powerful forward error correction channel codes, and provides near optimal bit error rates (BERs), that is, within 0.5dB of Shannon's limit at BER of 10^{-5} [1]. Having this remarkable performance, the turbo codes were accepted in many standardized mobile radio systems. Recently, non-binary turbo codes have been adopted in several mobile radio systems such as DVB-RCS and IEEE 802.16 standard (WiMAX) [2][3], as they can offer many advantages over the classical single binary turbo codes [4]. To avoid spectrum waste caused by the tail bits, circular coding technique, namely tail-biting, is also adopted in recent turbo codes.

Recently, several papers have suggested effective schemes for the soft-input soft-output (SISO) decoding of double-binary turbo codes [5]-[7]. However, there is little research yet on the hardware implementation of the non-binary turbo codes, although the previous works on the classical single binary turbo codes can be applied to the non-binary turbo codes. Compared with the classical single binary turbo codes, non-binary turbo codes are much more complex in hardware implementation. In addition, the initial state determination incurred by the tail-biting property leads to increased hardware complexity.

This paper presents an energy-efficient SISO decoder suitable for the hardware implementation of the non-binary circular turbo decoding. Based on the proposed architecture, a double-binary tail-biting turbo decoder is implemented for WiMAX with a dedicated hardware interleaver.

II. MAX-LOG-MAP ALGORITHM

A typical turbo decoder consists of two SISO decoders serially concatenated via an interleaver. Based on the MAP algorithm [1], how to SISO decode the single-binary turbo codes is well described in [8]. In the double-binary turbo codes, the output of the k-th symbol is expressed in LLR form and three LLR values are required as follows.

$$\begin{aligned}\Lambda_k^{(z)} \cong & \max_{(s_k \to s_{k+1}, z)}\left[\tilde{\alpha}_k(s_k) + \tilde{\gamma}_{k+1}(s_k \to s_{k+1}) + \tilde{\beta}_{k+1}(s_{k+1})\right] \\ & - \max_{(s_k \to s_{k+1}, 00)}\left[\tilde{\alpha}_k(s_k) + \tilde{\gamma}_{k+1}(s_k \to s_{k+1}) + \tilde{\beta}_{k+1}(s_{k+1})\right]\end{aligned} \quad (1)$$

where $z \in \phi = \{01,10,11\}$, s_k is the state of an encoder at time k, and α, β and γ are the forward, backward, and branch metrics, respectively. The metrics are calculated as expressed in equations (2), (3) and (4), where A is the set of states at time k-1 connected to state s_k, and B is the set of states at time k+1 connected to state s_k.

$$\tilde{\alpha}_k(s_k) \cong \max_{s_{k-1} \in A}\left[\tilde{\alpha}_{k-1}(s_{k-1}) + \tilde{\gamma}_k(s_{k-1} \to s_k)\right] \quad (2)$$

$$\tilde{\beta}_k(s_k) \cong \max_{s_{k+1} \in B}\left[\tilde{\beta}_{k+1}(s_{k+1}) + \tilde{\gamma}_{k+1}(s_k \to s_{k+1})\right] \quad (3)$$

$$\begin{aligned}\tilde{\gamma}_k(s_k \to s_{k+1}) &= \ln\left[P(\mathbf{y}_k \mid \mathbf{x}_k) \cdot P(u_k = z)\right] \\ &= \frac{L_c}{2}(x_k^{s_1} y_k^{s_1} + x_k^{s_2} y_k^{s_2} + x_k^{p_1} y_k^{p_1} + x_k^{p_2} y_k^{p_2}) + L_{e,IN}^{(z)}\end{aligned} \quad (4)$$

where $z \in \varphi = \{00,01,10,11\}$, u_k is the input symbol consisting of two bits, $P(u_k)$ is *a priori* probability of u_k, $\mathbf{x}_k$ and $\mathbf{y}_k$ are transmitted and received codewords associated with u_k, respectively. Superscripts p and s denote the parity bits and systematic bits, respectively. In (4), $L_{e,IN}^{(z)}$ is the extrinsic information received from the other SISO decoder and the code is transmitted through an AWGN channel with a noise variance σ^2. Since turbo decoding based on the Max-log-MAP algorithm is independent of SNR, $L_c = 2/\sigma^2$ is usually set to a constant value, although it can be obtained from channel estimation. As expressed above, the metric calculation complexity of the non-binary turbo codes is higher than that of the single-binary turbo codes since the number of branches connected with an each trellis state is increased from two to four and the complexity of the branch metric calculation is also increased.

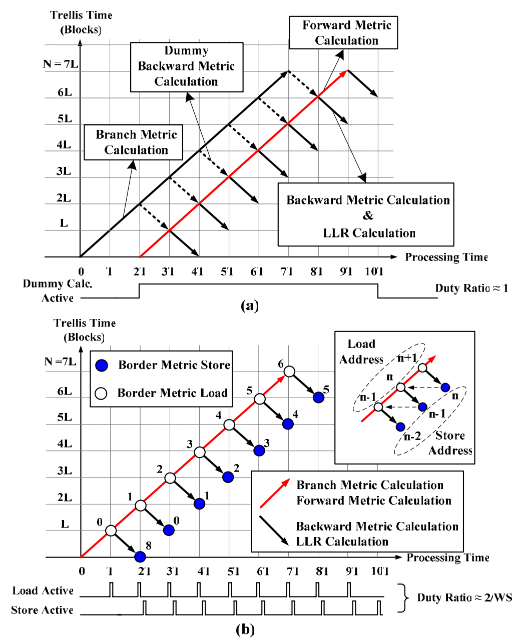
Trellis Time (Blocks)
Forward Metric Calculation
Dummy Backward Metric Calculation
Branch Metric Calculation
Backward Metric Calculation & LLR Calculation
Processing Time
Dummy Calc. Active
Duty Ratio ≈ 1
(a)
Border Metric Store
Border Metric Load
Load Address
Store Address
Branch Metric Calculation Forward Metric Calculation
Backward Metric Calculation LLR Calculation
Load Active
Store Active
Duty Ratio ≈ 2/WS
(b)

Figure 1. Sliding window diagram (a) with dummy calculation and (b) with border memory

Due to the tail-biting property, the initial values of the forward metric and backward metric are not explicitly specified, because the initial state, namely the circular state, is not delivered to the decoder. To determine the initial values, two methods are generally preferred. The first one is to pre-compute several trellis stages to get reliable initial values [6]. Another way is to use the metric values of the prior iteration [5][6]. It has been reported that using the information of the prior iteration shows better performance and lower computational complexity than the pre-computing method [6]. Therefore, in this paper, the initial values of the forward and backward metrics can be determined as follows.

$$\tilde{\alpha}_0(s)=\begin{cases}0 & \text{for the 1}^{\text{st}}\text{ iteration}\\ \tilde{\alpha}'_N(s) & \text{otherwise}\end{cases}$$
$$\tilde{\beta}_N(s)=\begin{cases}0 & \text{for the 1}^{\text{st}}\text{ iteration}\\ \tilde{\beta}'_0(s) & \text{otherwise}\end{cases} \tag{5}$$

where N is the frame size in pairs, s denotes the encoder state and $\tilde{\alpha}'_N(s)$ and $\tilde{\beta}'_0(s)$ is the final metrics of the prior iteration.

III. Sliding Window for Non-Binary Turbo Codes

The sliding window technique is effective in reducing the memory size required to store metric values in hardware implementation. In the sliding window technique, a large frame is split into several windows and the MAP decoding is applied to each window independently [11]. Figure 1(a) shows the conventional sliding window diagram where forward metrics are calculated prior to backward metrics [7]. As shown in Fig. 1(a), the dummy calculation is performed for the backward metrics to obtain the reliable initial values at the border of the window. We can obtain the reliable values that do not degrade performance if the window size is sufficiently long.

There is another way to obtain the reliable border metric values of the window. For each window, the final backward metric is stored in the border metric memory. The stored border metric is loaded in the next iteration to regard it as the initial backward metric value at the border of the window as illustrated in Fig. 1(b). Since there is no stored value in the first iteration, all states at the border of the window are assumed to be equiprobable at the first time. Although there is slight performance degradation compared to the method based on the dummy calculation, the performance degradation disappears after a few iterations. By using the metrics stored in the prior iteration, we can avoid the dummy backward metric calculation. It has been reported that we can achieve higher bandwidth efficiency for triple-binary turbo codes [9] and obtain better performance if the number of states in the trellis increases [4]. Since these two factors increase the computational complexity of the dummy backward metric calculation, the sliding window associated with no dummy calculation can be more effective for the future turbo codes as well as the double-binary turbo codes. Additionally, the memory size required for the branch metric is reduced to half since the number of processes where the branch metrics are participated is changed from four to two.

IV. Border Metric Encoding

As described, a memory that holds the border metric values of the prior iteration is needed. To achieve more energy-efficient turbo decoding, the size of the border memory should be minimized. If the maximum frame size supported in the turbo codes is N_{max}, the number of states in trellis is K, and state metric values are represented in P bits, the size of the border memory (BM) is defined as follows.

$$BM=\left\lceil\frac{N_{\max}}{WS}\right\rceil\times K\times P \tag{6}$$

where WS is the window size. Since N_{max} and K are fixed for a standard, the border memory size depends only on the window size and the size of the state metric. Since the window size is usually set to 32 for the 8-state trellis, we should decrease the value of P in order to reduce the overall border memory size.

The reduction of the border memory can be realized by allowing a few values to represent the border metrics. Though the reliability of the border metric is slightly decreased due to the loss of accuracy, this can be totally recovered after a few trellis stages. A simple encoding with low hardware complexity is to floor the original border metrics to the closest power of two numbers. The experimental environment for the double-binary turbo decoder for WiMAX is indicated in Table I, where (q, f) denotes a quantization scheme that uses q bits in total and f bits to represent the fractional part. The quantization schemes are determined by performing several simulations. Possible values at the border are denoted in Table II. BER results with

TABLE I. SIMULATION ENVIRONMENT

SISO Algorithm	Max-log-MAP
Window Size	32
Quantization	Received input: (6, 2) Branch Metric : (10, 2) State Metric : (10, 2) Extrinsic Information : (11, 2) LLR values : (12, 2)

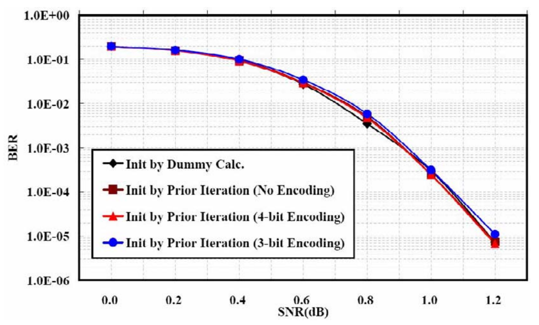

Figure 2. BER performance (1920 bit frame size and 8 iterations)

various methods are shown in Fig. 2. From Fig. 2, it is noticed that two methods described in Fig. 1(a) and (b) shows almost equal performance. Also, 4-bit border metric encoding schemes shows that there is almost no performance degradation and the method with 3-bit border metric encoding works well with about 0.03dB performance degradation compared to the non-encoding method. The performance in 3-bit border metric encoding is degraded mainly at the high SNR since the values are limited at the relatively small value. The size of the border metric memory can be reduced significantly by using the proposed encoding.

V. HARDWARE INTERLEAVER DESIGN

In turbo codes, the interleaver is involved in both encoding and decoding. The most straightforward method of implementing the address interleaving is to store interleaved addresses in a memory after generating all the possible addresses. In case of WiMAX, the memory size should be large enough to cover 2400 by 12bits, leading to significant area occupation and power consumption. To achieve low power consumption, a dedicated hardware interleaver is designed to generate the address on-the-fly [10]. Fig. 3 describers how to calculate the interleaved addresses on-the-fly for WiMAX, where P_0, P_1, P_2 and P_3 are determined according to the frame length, N [3]. The first step can be simply accomplished by switching the values according to the least significant bit (LSB) of the address. Figure 4 illustrates the hardware structure for the second step, that is, inter-symbol permutation. Since the input address increases sequentially, we can generate the permutated address by accumulating P_0 to four different initial values and selecting one of four values according to two LSBs. For *modulo* operation, the accumulated values are always maintained to be less than N as shown in Fig. 4. Initial values of the accumulators are pre-calculated and maintained in a small table.

TABLE II. POSSIBLE VALUES FOR BORDER METRIC

Encoding Scheme	Possible values for border metric
4-bit encoding	-256, -128, -64, -32, -16, -8, -4, 0, 4, 8, 16, 32, 64, 128, 256 (15 values)
3-bit encoding	-64, -32, -16, 0, 16, 32, 64 (7 values)

VI. IMPLEMENTATION RESULTS

With the quantization indicated in Table I, a Max-log-MAP decoder based on the proposed border metric encoding was described in Verilog-HDL and synthesized with a 0.18 µm 4-Metal CMOS standard-cell library and compiled SRAM memories. Design Compiler and Power Compiler of Synopsys were used for the synthesis and power estimation, respectively. Switching activities resulting from gate-level simulation are annotated for gate-level power estimation. The window size is set to 32 and 4-bit border metric encoding has been employed. In the hardware implementation of the SISO decoder, forward metrics and backward metrics are normalized by subtracting a value of the zero state, $\tilde{\alpha}_k(s_0)$ and $\tilde{\beta}_k(s_0)$, from other metrics at the same trellis stage in order to avoid overflow in state metrics, which also eliminates the need to store metric values at the zero state. Since the SISO decoder takes two systematic bits and two parity bits at a time, the number of possible branch metrics is 16 while the number of possible branch metrics is 4 in the classical single binary turbo codes. Among the 16 possible branch metrics, only 8 branch metrics are unique for each trellis stage and others can be calculated by these unique branch metrics. Although the number of branch

```
Step 1 : Switch alternate couples
for i = 0 ... N-1
if (i mod 2 == 1) let (A_i, B_i) → (B_i, A_i) (i.e., switch the couple)

Step 2 : Inter-symbol permutation
for i = 0 ... N-1
switch i mod 4 :
case 0 : $P(i) = (P_0 \cdot i + 1)_{mod N}$
case 1 : $P(i) = (P_0 \cdot i + 1 + N/2 + P_1)_{mod N}$
case 2 : $P(i) = (P_0 \cdot i + 1 + P_2)_{mod N}$
case 3 : $P(i) = (P_0 \cdot i + 1 + N/2 + P3)_{mod N}$
```

Figure 3. Pseudocode of the interleaver for WiMAX

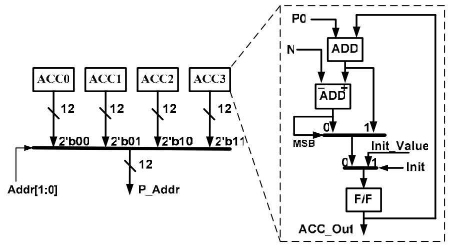

Figure 4. Interleaver structure with four accumulators

TABLE III. MEMORY SIZE COMPARISON FOR SINGLE-PORT SRAM

	Conventional Sliding Window	**Proposed (4-bit Encoding)**
Forward Metric Memory	2 banks, 32*(10*7) bits/bank	2 banks, 32*(10*7) bits/bank
	4480 bits	4480 bits
Branch Memory	4 banks, 32*(10*8) bits/bank	2 banks, 32*(10*8) bits/bank
	10240 bits	5120 bits
Border Memory	N.A.	(2400/32)*(4*7) bits → 2100 bits
Total	14720 bits (100%)	11700 bits (79.5 %)

TABLE IV. ENERGY CONSUMPTION COMPARISON OF SISO DECODER AND INTERLEAVER

	Conventional Sliding Window	**Proposed (4-bit Encoding)**
SISO logic	2381.0 pJ/bit/iter	1901.8 pJ/bit/iter
Interleaver	55.0 pJ/bit/iter	10.8 pJ/bit/iter
Branch Memory	1286.8 pJ/bit/iter	649.1 pJ/bit/iter
Forward Memory	559.1 pJ/bit/iter	559.1 pJ/bit/iter
Border Memory	N.A.	50.2 pJ/bit/iter
Total	4281.9 pJ/bit/iter (100%)	3171.0 pJ/bit/iter (74.0 %)

metrics to be stored is reduced by half, the size of branch memory is considerable if the conventional sliding window is adopted as indicated in Table III. The total memory size in the SISO decoder is reduced by 20.5% by applying the proposed border metric encoding method, as indicated in Table III.

The energy consumption of the proposed SISO decoder and the interleaver in 1.2dB SNR with 8 iterations is presented in Table IV. As indicated in Table IV, the power consumption of the SISO logic is reduced by eliminating the dummy calculation logic. Also, as shown in Fig. 1(b), the power consumption of the border metric memory accesses is very low because of its small size and infrequent accesses. While processing a window, we need to access the border memory only two times – one for load and the other for store. For the case of the dummy calculation, however, the dummy calculation logic should operate almost all the time as described in Fig. 1(a). Also, by using the dedicated hardware interleaver rather than the table-based interleaver, the power consumption in the interleaver is hugely reduced. Therefore, the proposed turbo decoder can reduce the power consumption by 26.0% compared to the conventional turbo decoder where the dummy calculation and table-based interleaver are adopted. The proposed double-binary tail-biting turbo decoder is based on the time-multiplex architecture consisting of only one SISO decoder and one interleaver like [10]. All the components are shared for both the first and the second SISO decoding of an iteration. The gate count of the proposed turbo decoder is 46,338 excluding memories. The buffers are implemented using single-port SRAMs, and small-sized ROM memories are replaced with logic circuitry. For processing a 2400-pair (4800-bit) frame, the proposed turbo decoder takes 39,537 cycles with eight iterations. Operating at 100 MHz, therefore, the data rate of the proposed turbo decoder is 12.14 Mbps.

VII. CONCLUSION

We have proposed an energy-efficient turbo decoding method based on border metric encoding, which is especially suitable for the non-binary circular turbo codes. By applying the proposed method, the size of branch memory is reduced by half and the dummy calculation that is complex in non-binary turbo codes is completely removed at the cost of a small memory that holds encoded border metrics. Also, a dedicated hardware interleaver compatible with WiMAX standard is employed to reduce power and area further. Due to the proposed SISO decoder and interleaver, the memory size is reduced by 20.5% and power consumption by 26.0%. The double-binary tail-biting turbo decoder implemented for WiMAX supports 12.14 Mbps when operated at 100MHz.

ACKNOWLEDGMENT

This work was supported by Institute of Information Technology Assessment through the ITRC and by IC Design Education Center (IDEC).

REFERENCES

[1] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error correcting coding and decoding : turbo codes," in Proc. Int. Conf. Commun., pp. 1064–1070, May 1993.

[2] C. Douillard, M. Jezequel, C. Berrou, N. Brengarth, J. Tousch, and N. Pham, "The Turbo Code Standard for DVB-RCS," in Proc. 2nd Int. Symp. Turbo Codes and Related Topics, Brest, France, Sept. 2000.

[3] IEEE Std 802.16e/D5-2004, Part 16: Air Interface for Fixed and Mobile Broadband Wireless Access Systems – Amendment for Physical and Medium Access Control Layers for Combined Fixed and Mobile Operation in Licensed Bands, Nov. 2004.

[4] C. Berrou, M. Jezequel, C. Douillard, and S. Kerouedan, "The advantages of non-binary Turbo codes," Proc. IEEE Information Theroy Workshop, pp. 61–63, Sept. 2001.

[5] John B. Anderson and Stephen M. Hladik, "Tailbiting MAP decoders," IEEE Journal on Selected Areas in Communications, vol. 16, no. 2, pp. 297–302, Feb. 1998.

[6] C. Zhan, T. Arslan, A. T. Erdogan, and S. MacDougall, "An Efficient Decoder Scheme for Double Binary Circular Turbo Codes," IEEE International Conference on Accoustics, Speech and Signal Processing, pp. IV-229–IV-232, May 2006.

[7] S. Papaharalabos, P. Sweeny, and B. G. Evans, "Constant log-MAP decoding algorithm for duo-binary turbo codes," Electronics Letters, vol. 42, no. 12, pp. 709–710, June 2006.

[8] H. M. Choi, J. H. Kim, and I. C. Park, "Low-Power Hybrid Turbo Decoding Based on Reverse Calculation," in Proc. of IEEE Int. Symp. On Circuits and Systems, pp. 2053–2056, May 2006.

[9] Yingzi Gao and M. R. Soleymani, "Triple-binary Circular Recursive Systematic Convolutional Turbo Codes," the 5th international Symposium on Wireless Personal Multimedia Communications, vol. 3, pp. 951–955, Oct. 2002.

[10] M. C. Shin and I. C. Park, "A programmable turbo decoder for multiple third-generation wireless standards," IEEE Int. Solid-State Circuits Conf. (ISSCC), pp. 154–155, Feb. 2003.

[11] A. J. Viterbi, "An intuitive justification and a simplified implementation of the MAP decoder for convolutional codes," IEEE J. Sel. Areas Commun., vol. 16, no. 2, pp. 260–264, Feb. 1998.