Double-Binary Circular Turbo Decoding Based on Border Metric Encoding

Ji-Hoon Kim, *Student Member, IEEE*, and In-Cheol Park, *Senior Member, IEEE*

***Abstract*—This brief presents an energy-efficient soft-input soft-output (SISO) decoder based on border metric encoding, which is especially suitable for nonbinary 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 infrequent accesses to the border memory and its small size, the energy consumed for SISO decoding is reduced by 26.2%. Based on the proposed SISO decoder and the dedicated hardware interleaver, a double-binary tail-biting turbo decoder is designed for the WiMAX standard using a 0.18-μm CMOS process, which can support 24.26 Mbps at 200 MHz.**

***Index Terms*—Low power design, maximum *a posteriori* (MAP) algorithm, nonbinary turbo codes, turbo decoding.**

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 0.5 dB of Shannon's limit at BER of 10^{-5} [1]. Having this remarkable performance, the turbo codes have been accepted in many standardized mobile radio systems. Recently, nonbinary turbo codes have received a great attention and adopted in several mobile radio systems such as DVB-RCS and IEEE 802.16 standard (WiMAX) [2], as they can offer many advantages over the classical single-binary turbo codes [3]. To avoid spectrum waste caused by the tail bits, the circular coding technique called tail-biting is also employed in the turbo codes.

Effective schemes have been suggested for the soft-input soft-output (SISO) decoding of double-binary turbo codes [4], [5]. However, there has been little research dedicated to the hardware implementation of the nonbinary turbo codes, although the previous works on the classical single-binary turbo codes can be applied to the nonbinary turbo codes. Compared with the classical single-binary turbo codes, nonbinary turbo codes are much more complex in hardware implementation. In addition, the initial state determination incurred by the tail-biting property makes the hardware complexity increase.

This brief presents an energy-efficient SISO decoder suitable for the nonbinary circular turbo decoding. Based on the proposed architecture, a double-binary circular turbo decoder is implemented for the WiMAX with a dedicated hardware interleaver.

Manuscript received October 27, 2006; revised February 14, 2007 and June 18, 2007. This work was supported by the Institute of Information Technology Assessment through the ITRC and by IC Design Education Center (IDEC). This paper was recommended by Associate Editor M. Maggio.

The authors are with the School of Electrical Engineering and Computer Science, Korea Advanced Institute of Science and Technology (KAIST), Daejeon 305-701, Korea (e-mail: sckid@eeinfo.kaist.ac.kr; icpark@ee.kaist.ac.kr).

Digital Object Identifier 10.1109/TCSII.2007.907803

II. Max–Log–MAP Algorithm

A typical turbo decoder consists of two SISO decoders serially concatenated via an interleaver. Based on the maximum *a posteriori* (MAP) algorithm algorithm [1], how to decode the single-binary turbo codes is well described in [6], [7]. In the double-binary turbo codes, the three log-likelihood ratio outputs of the kth symbol are expressed 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 belongs to $\phi = \{01, 10, 11\}$, s_k is the state of an encoder at time k, and $\tilde{\alpha}$, $\tilde{\beta}$ and $\tilde{\gamma}$ are the forward, backward, and branch metrics, respectively. The metrics are calculated as expressed in (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

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

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

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

where z belongs to $\varphi = \{00, 01, 10, 11\}$, u_k is the input symbol consisting of two bits, $P(u_k)$ is *a priori* probability of u_k, and $\mathbf{x}_k$ and $\mathbf{y}_k$ are transmitted and received codewords associated with u_k, respectively. The 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 assumed to be transmitted through an AWGN channel with a noise variance σ^2. Since the Max–log–MAP decoding algorithm is independent of the signal-to-noise ratio (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 nonbinary turbo codes is higher than that of the single-binary turbo codes. For the double-binary turbo codes, the number of branches connected to each trellis state is increased from two to four as shown in

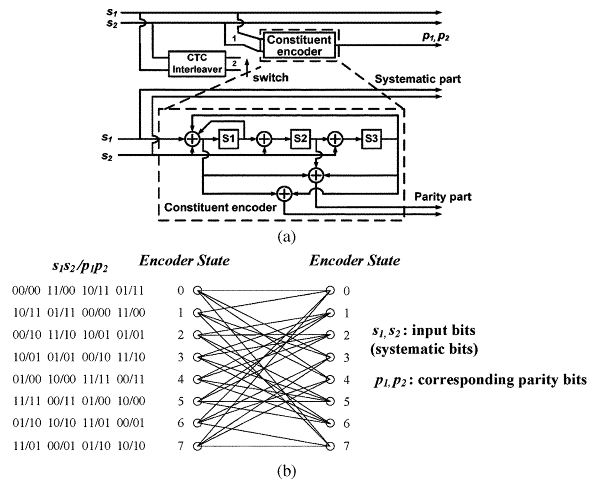

Fig. 1. Double-binary circular turbo code for the WiMAX. (a) Turbo encoder. (b) Trellis diagram.

Fig. 1(b). Since a $\max$ operation with four operands can be implemented by using three $\max$ operations with two operands as shown in (5), the hardware complexity is almost three times higher than that of the classical single-binary turbo codes if the four-operand $\max$ operation is computed in a cycle

$$\max(a,b,c,d) = \max\left(\max(a,b), \max(c,d)\right). \tag{5}$$

It is possible to compute the four-operand $\max$ operation serially using a two-operand $\max$ operator, but this structure requires more than one cycles and additional buffers to hold the intermediate values. Moreover, the serial $\max$ computation results in severe throughput degradation, as the forward and backward metrics are recursively defined using the previously calculated metrics. Compared to the single-binary SISO decoders [6], [7], the wordlength of internal metrics should be increased in hardware implementation, as the number of terms to be added in the branch metric calculation is increased from three to five as expressed in (4).

A. Initial Values for Circular Turbo Codes

Due to the tail-biting property, the initial values of the forward metric and backward metric are not explicitly specified, that is, the initial state values, namely the circular state values, are 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 [4], and the other way is to use the metric values of the previous iteration [4], [8]. It has been reported that using the information of the previous iteration shows better performance and lower computational complexity than the pre-computing method [4]. Therefore, in this brief, the initial forward and backward metrics of a frame are determined as follows if the frame size is N in pairs

$$\tilde{\alpha}_0(s) = \begin{cases} 0, & \text{for the first iteration} \\ \tilde{\alpha}'_N(s), & \text{otherwise} \end{cases}$$
$$\tilde{\beta}_N(s) = \begin{cases} 0, & \text{for the first iteration} \\ \tilde{\beta}'_0(s), & \text{otherwise} \end{cases} \tag{6}$$

where s denotes the encoder state, and $\tilde{\alpha}'_N(s)$ and $\tilde{\beta}'_0(s)$ are the metric values of the previous iteration.

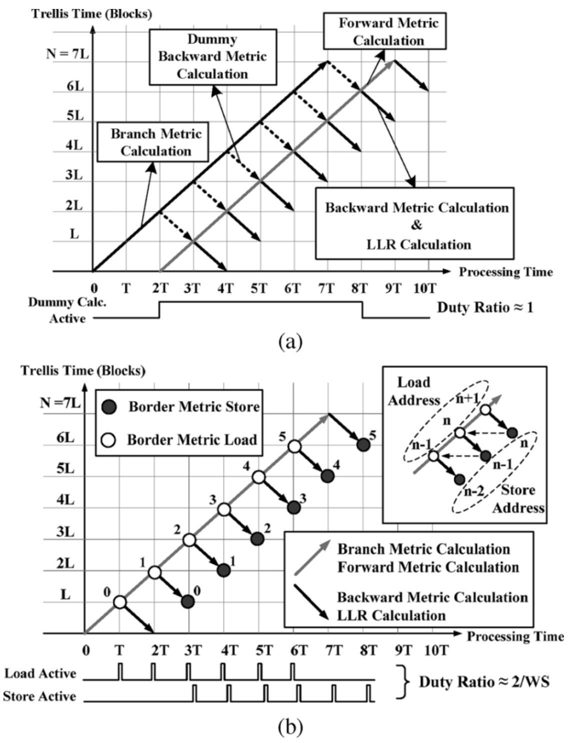

Fig. 2. Sliding window diagram: (a) with dummy calculation and (b) with border memory.

III. Sliding Window for Nonbinary Turbo Codes

The sliding window technique is effective in reducing the memory size required to store metric values. A large frame is split into a number of small windows and the MAP decoding is applied to each window independently [9]. Fig. 2(a) shows the conventional sliding window diagram where forward metrics are calculated prior to backward metrics [6]. In the sliding window technique, however, the initial values at the border of each window are also required. To obtain the reliable initial values of each window, the dummy calculation is performed for the backward metrics as shown in Fig. 2(a). If the window size is sufficiently long, the initial values obtained by the dummy calculation do not degrade performance.

Another way to obtain reliable border metric values is to use those values of the previous iteration, which has been adopted for the classical single-binary turbo codes [10]. In this brief, this approach is employed with modifying it for the double-binary turbo codes. For each window, the last backward metric is stored in a memory called the border memory. The stored border metrics are loaded in the next iteration to regard them as the initial backward metric values at the borders as illustrated in Fig. 2(b). Since there is no stored value in the first iteration, all states at the borders are assumed to be equiprobable in the first iteration. Compared to the conventional method based on the dummy calculation, this approach results in slight performance degradation for the earlier iterations, but the performance degradation disappears after a few iterations. By using the metrics stored in the previous iteration, we can completely avoid the dummy backward metric calculation. Additionally, the size of the branch metric memory is reduced to half since the number of processes in which the branch metrics are participated is changed from four to two.

IV. Border Metric Encoding

As described in the previous section, an additional memory is needed to hold the border metric values of the previous iteration. Although the sliding window with the border memory can eliminate the need of the dummy calculation, the border memory size is considerable. To achieve more area-efficient and energy-efficient turbo decoding, the border memory should be minimized. If the maximum frame size is $N_{\max}$, the number of states in trellis is K, and state metric values are represented in P bits, then the border memory size (BM) is defined as follows.

$$\mathrm{BM} = \left(\left\lceil \frac{N_{\max}}{WS} \right\rceil - 1 \right) \times K \times P \tag{7}$$

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 wordlength of state metrics. To reduce the border memory size, we can either increase the window size or decrease the wordlength of state metrics. Increasing the window size, however, increases the sizes of the memories storing the forward and branch metrics, and the window size is usually set to 32 for 8-state trellis. Therefore, we should decrease P to reduce the overall border memory size. Otherwise the sliding window associated with the border memory may not be suitable for the hardware implementation because a large border memory is indispensable for the W-CDMA whose $N_{\max}$ is 5114 and for the WiMAX whose $N_{\max}$ is 2400.

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 metric value to the closest power-of-two number. The experimental environment for the 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 final quantization schemes shown in Table I are determined by performing several simulations and referring to [7] and [11]. The encoding function for the proposed 3-bit encoding is depicted in Fig. 3. The encoding function for the 4-bit encoding can be similarly defined. Possible values at the border are listed in Table II. As the range of the original border metrics is $[-512, +511]$ which can be represented with 10 bits, the proposed border metric encoding can be obtained by limiting the value into $[-256, +256]$ for the 4-bit encoding and $[-64, +64]$ for the 3-bit encoding and by allowing only power-of-two values. In Fig. 4, the BER performance of the proposed encoding is compared with those of various methods. The schemes in which the border metric is initialized with the value of the previous iteration degrade the performance by about 0.02 dB in the water fall region. If the SNR is higher than 1 dB, however, the proposed 4-bit encoding shows about 0.1 dB better performance than the classical method which uses the dummy calculation. It is well-known that we can obtain better performance for the Max–log–MAP algorithm by scaling the extrinsic information [12]. Since the floor function reduces the border metrics, its effect is similar to the extrinsic information scaling. When the SNR is high, the performance of the 3-bit encoding is degraded by about 0.1 dB compared to the other schemes as the values are restricted to a relatively small region. The size of the border memory can be reduced significantly by using the proposed encoding. As the iteration proceeds, the BER degradation resulting from the proposed encoding scheme becomes negligible as shown in Fig. 5.

TABLE I
Simulation Environment

SISO Algorithm	Max-log-MAP
Window Size	32
Quantization	Received input: (6, 2) Branch Metric & State Metrics : (10, 2) Extrinsic Information : (8, 2) LLR value : (11, 2)

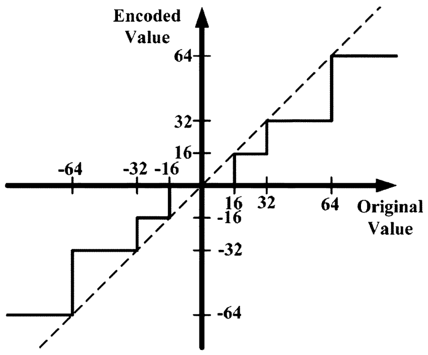

Fig. 3. 3-bit border metric encoding function.

TABLE II
Encoded Values for Border Metrics

Encoding Scheme	Encoded values for border metric
4-bit encoding	±256, ±128, ±64, ±32, ±16, ±8, ±4, 0
3-bit encoding	±64, ±32, ±16, 0

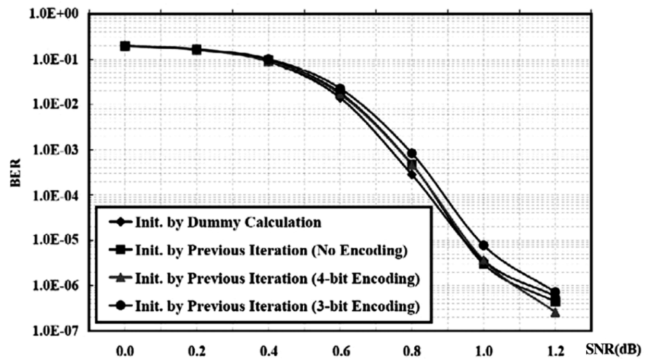

Fig. 4. BER performance comparison with 8 iterations for 4800-bit frame.

It has been reported that we can achieve higher bandwidth efficiency for triple-binary turbo codes [13] and obtain better performance if the number of states in the trellis increases [3]. Since these two factors increase the complexity of the dummy backward metric calculation, the sliding window associated with the proposed border metric encoding can be more effective for the future turbo codes.

V. Double-Binary Turbo Decoder Implementation

A. Hardware Interleaver Design for the WiMAX

In turbo codes, the interleaver is involved in both encoding and decoding. The most straightforward way to implement

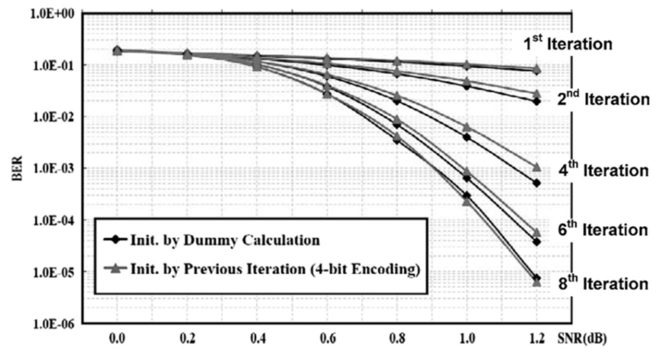

Fig. 5. BER performance of 1920-bit frame according to the number of iterations.

```
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 : $P(i) = (P_0 \cdot i + 1 + N/2 + P_3)_{mod\,N}$
```

Fig. 6. Interleaving procedure for the WiMAX.

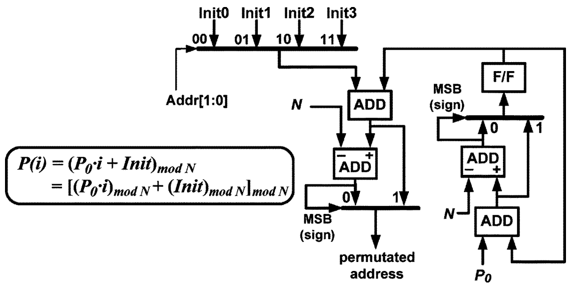

Fig. 7. Interleaver structure based on the incremental calculation.

the address interleaving is to store interleaved addresses in a memory. In case of the WiMAX, the memory size required to store all the interleaving patterns is about 90 K bits since every frame size is associated with a different interleaving pattern. This large-sized memory leads to significant area occupation and power consumption. For 3G wireless systems, a dedicated hardware interleaver that generates interleaved addresses on-the-fly has been proposed to achieve small area [14]. Such a dedicated interleaver is also effective in reducing power consumption as there is no need to include a large-sized memory. Due to the property of the interleaver adopted in the WiMAX, the dedicated hardware interleaver can be implemented efficiently. Fig. 6 describes how to calculate the interleaved addresses on-the-fly for the WiMAX, where P_0, P_1, P_2 and P_3 are determined according to the frame length, N [2]. The first step can be simply accomplished by switching the values according to the least significant bit (LSB) of the address. Fig. 7 illustrates the hardware structure for the second step, that is, inter-symbol permutation. Since the input address increases sequentially, accumulating P_0 and adding it to an initial value selected by the two LSBs can generate the permutated address.

TABLE III
SINGLE-PORT SRAM SIZE REQUIRED FOR A SISO DECODER

	With Dummy Calculation	With Border Memory (No Encoding)	With Border Memory (4-bit Encoding)
Forward Memory	2 banks, 32*(10*7) bits/bank	2 banks, 32*(10*7) bits/bank	2 banks, 32*(10*7) bits/bank
	4480 bits	4480 bits	4480 bits
Branch Memory	4 banks, 32*(10*8) bits/bank	2 banks, 32*(10*8) bits/bank	2 banks, 32*(10*8) bits/bank
	10240 bits	5120 bits	5120 bits
Border Memory	N.A.	[(2400/32)-1] *(10*7) bits = 5180 bits	[(2400/32)-1] *(4*7) bits = 2072 bits
Total	14720 bits (100%)	14780 bits (100.4 %)	11672 bits (79.3 %)

To replace the complicated *modulo* operation, subtractions are performed in Fig. 7 when the intermediate values are not less than N. The initial values are pre-calculated and maintained in a small table.

B. Turbo Decoder for the WiMAX

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 were annotated for gate-level power estimation. The window size is set to 32 and the 4-bit border metric encoding is employed. In the hardware implementation, the forward metrics and backward metrics are normalized by subtracting the value of state 0, $\tilde{\alpha}_k(s_0)$ 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 the metric value of state 0. Since the SISO decoder takes two systematic bits and two parity bits as inputs, 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 distinguishable and sufficient to derive the others. Although the number of branch metrics to be stored is reduced by half, the branch memory size is still considerable if the conventional sliding window with the dummy calculation is adopted as indicated in Table III. Even in the case that the sliding window is associated with the border memory, the total memory size is increased because of the border memory requirement. By applying the proposed border metric encoding method, the total memory size needed in the SISO decoder is reduced by 20.7% as summarized in Table III. As the size of extrinsic information is not related to the window size and the number of SISO decoders included, it is separately summarized in Table IV, where the 5-bit encoding technique proposed in [15] is used to reduce the total size.

The proposed double-binary tail-biting turbo decoder is based on the time-multiplex architecture consisting of one SISO decoder, an extrinsic information memory and a dedicated interleaver as shown in Fig. 8. All the components are shared for both of the two SISO decodings of an iteration. The total gate count of

TABLE IV
MEMORY CONFIGURATION FOR THE EXTRINSIC INFORMATION

	$L_e^{(01)}$	$L_e^{(10)}$	$L_e^{(10)}$
Extrinsic Info.	2 banks, 5*2400bits/bank	2 banks, 5*2400bits/bank	2 banks, 5*2400bits/bank
Memory	24000 bits	24000 bits	24000 bits
Total	72000 bits		

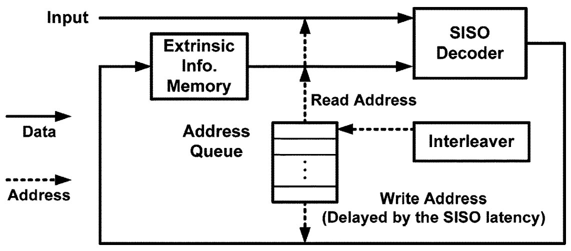

Fig. 8. A block diagram of the time-multiplex turbo decoder.

TABLE V
ENERGY CONSUMPTIONS OF SISO DECODERS

	With Dummy Calculation	With Border Memory (4-bit Encoding)
SISO Logic	2347.9 pJ/bit/iter	1876.9 pJ/bit/iter
Interleaver	55.1 pJ/bit/iter	3.5 pJ/bit/iter
Branch Memory	1288.4 pJ/bit/iter	649.9 pJ/bit/iter
Forward Memory	559.1 pJ/bit/iter	559.1 pJ/bit/iter
Border Memory	N.A.	49.4 pJ/bit/iter
Total	4250.5 pJ/bit/iter (100%)	3138.8 pJ/bit/iter (73.8 %)

the proposed turbo decoder is 50,866 excluding memories and the critical path delay is 4.4 ns. The buffers are implemented using single-port SRAMs, and small-sized ROMs are replaced with logic circuitry. To process a 2400-pair (4800-bit) frame, the proposed turbo decoder takes 39,585 cycles for eight iterations. As a result, the data rate that the proposed turbo decoder operating at 200 MHz can process is 24.26 Mbps. To achieve the higher throughput, several SISO decoders can be employed for parallel turbo decoding. In this case, the number of SISO decoders should be chosen carefully to keep the *collision-free* property in the interleaver.

In Table V, the energy consumption of the proposed SISO decoder is compared with that of the conventional decoder, which is measured for 1.2 dB SNR and eight iterations at the operating frequency of 200 MHz. Due to the increased computational complexity of the double-binary turbo codes, the energy consumption of the SISO logic is also increased compared to the classical single-binary turbo codes [6], [7]. As shown in Table V, the energy consumption of the SISO logic is reduced by eliminating the dummy calculation. Also, as shown in Fig. 2(b), the energy consumption of the border memory is very low because the memory is small and infrequently accessed. 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 dummy calculation, however, the dummy calculation logic should operate almost all the time as indicated in Fig. 2(a). Employing the dedicated hardware interleaver, the area and the power consumption are significantly reduced compared to the table-based interleaver. Therefore, the proposed SISO decoder can reduce the energy consumption by 26.2% compared to the conventional SISO decoder based on the dummy calculation and the table-based interleaver.

VI. CONCLUSION

We have presented an energy-efficient SISO decoding method based on the border metric encoding, which is especially suitable for the nonbinary circular turbo codes. By applying the proposed method, the branch memory size is reduced by half and the dummy calculation is completely removed at the cost of a small memory that holds encoded border metrics. In addition, a dedicated hardware interleaver compatible with the WiMAX is employed to reduce power and area further. Due to the proposed SISO decoder and interleaver, the memory size is reduced by 20.7% and energy consumption is decreased by 26.2%. The double-binary circular turbo decoder implemented for the WiMAX supports 24.26 Mbps at 200 MHz.

REFERENCES

[1] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near shannon limit error correcting coding and decoding: Turbo codes," in *Proc. Int. Conf. Commun.*, May 1993, pp. 1064–1070.
[2] *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*, IEEE Std 802.16e/D5-2004, Nov. 2004.
[3] C. Berrou, M. Jezequel, C. Douillard, and S. Kerouedan, "The advantages of nonbinary turbo codes," in *Proc. IEEE Inf. Theory Workshop*, Sep. 2001, pp. 61–63.
[4] C. Zhan, T. Arslan, A. T. Erdogan, and S. MacDougall, "An efficient decoder scheme for double binary circular turbo codes," in *Proc. IEEE ICASSP'06*, May 2006, pp. IV-229–IV-232.
[5] S. Papaharalabos, P. Sweeny, and B. G. Evans, "Constant log-MAP decoding algorithm for duo-binary turbo codes," *Electron. Lett.*, vol. 42, no. 12, pp. 709–710, Jun. 2006.
[6] H. M. Choi, J. H. Kim, and I. C. Park, "Low-power hybrid turbo decoding based on reverse calculation," in *Proc. IEEE Int. Symp. Circuits Syst.*, 2006, pp. 2053–2056.
[7] D. S. Lee and I. C. Park, "Low-power log-MAP decoding based on reduced metric memory access," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 53, no. 6, pp. 1244–1253, Jun. 2006.
[8] J. B. Anderson and S. M. Hladik, "Tailbiting MAP decoders," *IEEE J. Sel., Areas Commun*, vol. 16, no. 2, pp. 297–302, Feb. 1998.
[9] 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.
[10] A. Abbasfar and K. Yao, "An efficient and practical architecture for high speed turbo decoders," in *Proc. IEEE Vehicular Technol. Conf.*, Oct. 2003, pp. 337–341.
[11] H. Michel and N. When, "Turbo-decoder quantization for UMTS," *IEEE Commun. Lett.*, vol. 5, no. 2, pp. 55–57, Feb. 2001.
[12] J. Vogt and A. Finger, "Improving the Max–log–MAP turbo decoder," *Electron. Lett.*, vol. 36, no. 23, pp. 1937–1939, Jun. 2000.
[13] Y. Gao and M. R. Soleymani, "Triple-binary circular recursive systematic convolutional turbo codes," in *Proc. 5th Int. Symp. Wireless Personal Multimedia Commun.*, 2002, vol. 3, pp. 951–955.
[14] M. C. Shin and I. C. Park, "Processor-based turbo interleaver for multiple third-generation wireless standards," *IEEE Commun. Lett.*, vol. 7, no. 5, pp. 210–212, May 2003.
[15] D. Garrett, B. Xu, and C. Nicol, "Energy efficient turbo decoding for 3G mobile," in *Proc. ISLPED'001*, Aug. 2001, pp. 328–333.