IEICE TRANS. FUNDAMENTALS, VOL.E89–A, NO.3 MARCH 2006

PAPER

Low-Power Hybrid Turbo Decoding Based on Reverse Calculation

Hye-Mi CHOI[†], **Ji-Hoon KIM**[†], *and* **In-Cheol PARK**[†a)], *Nonmembers*

SUMMARY As turbo decoding is a highly memory-intensive algorithm consuming large power, a major issue to be solved in practical implementation is to reduce power consumption. This paper presents an efficient reverse calculation method to lower the power consumption by reducing the number of memory accesses required in turbo decoding. The reverse calculation method is proposed for the Max-log-MAP algorithm, and it is combined with a scaling technique to achieve a new decoding algorithm, called hybrid log-MAP, that results in a similar BER performance to the log-MAP algorithm. For the W-CDMA standard, experimental results show that 80% of memory accesses are reduced through the proposed reverse calculation method. A hybrid log-MAP turbo decoder based on the proposed reverse calculation reduces power consumption and memory size by 34.4% and 39.2%, respectively.

key words: *log-MAP algorithm, low-power design, Max-log-MAP algorithm, reverse calculation, turbo codes, turbo coding, turbo decoding*

1. Introduction

Turbo coding introduced in 1993 [1] is one of the most powerful forward error correction (FEC) channel codes. It has been reported that the turbo code can approach the Shannon's limit within a few tenths dB with moderate complexity. Having this remarkable performance, the turbo code has found its way into many standardized systems. For example, it has attracted much interest in magnetic recording systems, and the third-generation mobile radio systems such as CDMA2000 and W-CDMA have employed it for high-speed data transmission.

As turbo decoding is a highly memory-intensive algorithm, however, a significant amount of power is consumed for memory accesses, resulting in a power bottleneck that limits the incorporation of turbo decoders in commercial hand-held systems. It has been reported that the memory access power accounts for more than 50% of the entire power consumption [2].

To overcome this problem, several low-power techniques have been proposed [2]–[10]. One of the efficient methods is the reverse calculation that replaces memory accesses with additional computations. Y. Wu et al. first proposed the reverse calculation of state metrics [8], but did not describe implementation issues because their reverse calculation requires impractically large lookup tables. In [9], two reverse calculation methods were proposed, a loser storing method for the Max-log-MAP algorithm and a periodic storing method for the log-MAP algorithm. In the loser storing method, only the losers are stored into a memory while the winners are recovered by applying the reverse calculation. The periodic storing method is a variation of the sliding window scheme. A practical reverse calculation method was proposed for the log-MAP algorithm along with its implementation in [10]. However, the implementation is relatively complex and requires a large logic overhead.

This paper proposes an efficient reverse calculation method to reduce not only memory power consumption but also memory size. The proposed reverse calculation method is based on the Max-log-MAP algorithm, and it is combined with a scaling technique to achieve a new decoding algorithm called hybrid log-MAP. Compared to the loser storing method that always accesses the memory to decide the position and the value of the loser stored during forward recursion, the proposed reverse calculation method stores and accesses the loser selectively only when the loser cannot be recovered by the reverse calculation. Therefore, the proposed hybrid log-MAP algorithm can further reduce power consumption by reducing the number of memory accesses as well as the memory size.

The rest of this paper is organized as follows. In Sect. 2, conventional turbo decoding algorithms are described. A new reverse calculation method is proposed for Max-log-MAP algorithm in Sect. 3. A hybrid log-MAP algorithm that combines Max-log-MAP and log-MAP algorithms is proposed in Sect. 4. Experimental results are presented in Sect. 5.

2. Decoding Algorithms for Turbo Codes

The fundamental background on the turbo code is presented in this section. A conventional architecture for turbo encoding and decoding is described first, and two decoding algorithms are briefly explained.

2.1 Fundamentals of the Turbo Codes

Figure 1 shows a conventional turbo encoding architecture that consists of three parts, i.e., two recursive systematic convolutional (RSC) encoders and an interleaver. Let $\boldsymbol{u} = (u_1, u_2, \cdots, u_N)$ be a set of binary variables representing information bits, where N denotes the frame

Manuscript received July 6, 2005.
Manuscript revised October 10, 2005.
Final manuscript received November 21, 2005.

[†]The authors are with the Division of Electrical Engineering, the Department of Electrical Engineering and Computer Science, Korea Advanced Institute of Science and Technology, v 373-1, Guseong-dong, Yuseong-gu, Daejeon, Korea.
a) E-mail: icpark@ee.kaist.ac.kr

DOI: 10.1093/ietfec/e89–a.3.782

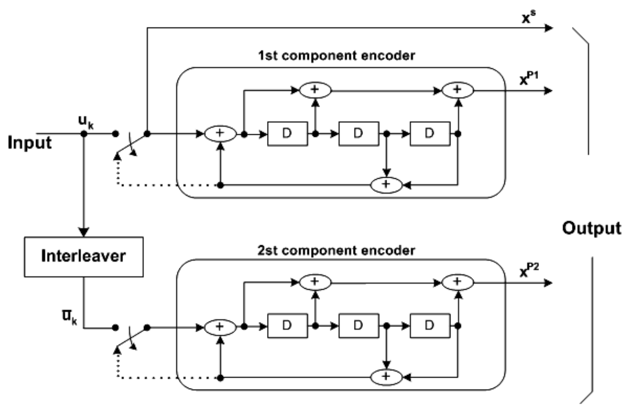

Fig. 1 Turbo encoder for W-CDMA.

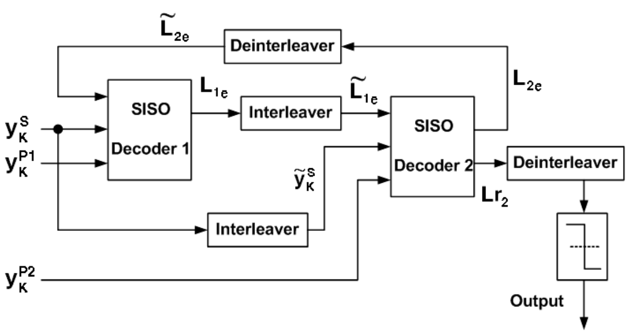

Fig. 2 Turbo decoder.

size. In the systematic encoders, one of the three outputs, $\boldsymbol{x}^S = (x_1^s, x_2^s, \cdots, x_N^s)$, is identical to the information sequence $\boldsymbol{u}$ and the others are the parity sequences, $\boldsymbol{x^{P1}} = (x_1^{p1}, x_2^{p1}, \cdots, x_N^{p1})$ and $\boldsymbol{x^{P2}} = (x_1^{p2}, x_2^{p2}, \cdots, x_N^{p2})$. The noisy outputs are represented by $\boldsymbol{y}^s = (y_1^s, y_2^s, \cdots, y_N^s)$, $\boldsymbol{y^{p1}} = (y_1^{p1}, y_2^{p1}, \ldots, y_N^{p1})$ and $\boldsymbol{y^{p2}} = (y_1^{p2}, y_2^{p2}, \ldots, y_N^{p2})$. The first RSC encoder takes the information sequence $\boldsymbol{u}$ in the original order, while the second encoder uses the interleaved information sequence. The two parity sequences and the information sequence itself are multiplexed to generate the turbo code sequence.

A typical turbo decoder consists of two component decoders serially concatenated via an interleaver that is identical to the one used in the encoder, as shown in Fig. 2. The first SISO decoder takes as input the information sequence $\boldsymbol{y}^s$ and the parity sequence generated by the first encoder $\boldsymbol{y^{p1}}$. The decoder then produces extrinsic information, denoted as L_{1e}, which is interleaved and used to produce an improved estimate of a priori probabilities of the information sequence in the second decoder. The additional two inputs of the second SISO decoder are the interleaved information sequence $\tilde{\boldsymbol{y}}^s$ and the parity sequence produced by the second encoder $\boldsymbol{y^{p2}}$. The second SISO decoder also produces extrinsic information L_{2e} to be used to improve the estimate of the a priori probabilities of the information sequence in the first SISO decoder. The decoder performance can be improved by these iterative operations. After a certain number of iterations, the soft outputs of both SISO decoders are not improved further in performance. Then the last stage makes hard decisions by examining the log-likelihood ratio (LLR).

Turbo coding requires SISO decoders to generate extrinsic information and LLR. Either the maximum a posteriori (MAP) algorithm [11] or the soft output Viterbi algorithm (SOVA) [12] can be used for SISO decoding. As MAP-based turbo decoders provide much better performance than SOVA-based turbo decoders, we focus on the MAP-based algorithm in this work. In practical implementations, Max-log-MAP and log-MAP algorithms are commonly employed due to the high complexity of the MAP algorithm [13], [14].

2.2 Max-log-MAP Algorithm

The Max-log-MAP decoder decides whether an information bit $u_k = 0$ or 1 depending on the log-likelihood ratio $L_R(u_k)$ which is defined below.

$$\begin{aligned} L_R(u_k) = &\max_{U^1}[\alpha_{k-1}(s_{k-1}) + \gamma_k(s_{k-1} \to s_k) + \beta_k(s_k)] \\ &- \max_{U^0}[\alpha_{k-1}(s_{k-1}) + \gamma_k(s_{k-1} \to s_k) + \beta_k(s_k)] \end{aligned} \tag{1}$$

where s_k is an encoder state at time k, U^1 is the set of all possible state transitions $(s_{k-1} \to s_k)$ corresponding to the case of $u_k = 1$, and U^0 is similarly defined. If $L_R(u_k)$ is greater than or equal to zero, u_k is decided to 1. Otherwise, u_k is decided to 0. In Eq. (1), $\alpha_{k-1}(s_{k-1})$, $\beta_k(s_k)$, and $\gamma_k(s_{k-1} \to s_k)$ are the forward, backward, and branch metrics, respectively. The metrics are calculated as

$$\alpha_k(s_k) = \max_{S_{k-1}}[\alpha_{k-1}(s_{k-1}) + \gamma_k(s_{k-1} \to s_k)] \tag{2}$$

$$\beta_{k-1}(s_{k-1}) = \max_{S_k}[\beta_k(s_k) + \gamma_k(s_{k-1} \to s_k)] \tag{3}$$

$$\gamma_k(s_{k-1} \to s_k) = \frac{1}{2}x_k^s(L_e(u_k) + L_C y_k^s) + \frac{L_C}{2}x_k^p y_k^p \tag{4}$$

Eq. (4) is derived with assuming that the code is transmitted through an AWGN channel with a noise variance of σ^2, where x_k and y_k represent the transmitted codeword associated with this transition, and the codeword received from the channel, respectively, $L_e(u_k)$ is the extrinsic information received from the other SISO decoder, and $L_C = 2/\sigma^2$.

The extrinsic information to be passed to the companion decoder, $L_{e,OUT}(u_k)$ is calculated as

$$L_{e,OUT}(u_k) = L_R(u_k) - L_C y_k^s - L_{e,IN}(u_k), \tag{5}$$

where $L_{e,IN}(u_k)$ denotes the extrinsic information received from the companion decoder. In the case of the second SISO decoder in Fig. 2, $L_{e,OUT}(u_k) = L_{2e}$ and $L_{e,IN}(u_k) = \tilde{L}_{1e}$.

2.3 log-MAP Algorithm

As some approximations are used in the max functions of Eqs. (1), (2), and (3), there is performance degradation in the Max-log-MAP algorithm compared to the MAP algorithm. It can be improved by using the max* function [15], [16] which is defined as

$$\max *(x, y) = \max(x, y) + \ln(1 + e^{-|x-y|}) \tag{6}$$

Given the max* function, we can rewrite Eqs. (1), (2), and (3) as

$$\alpha_k(s_k) = \max_{S_{k-1}}{}^*[\alpha_{k-1}(s_{k-1}) + \gamma_k(s_{k-1} \to s_k)] \tag{7}$$

$$\beta_{k-1}(s_{k-1}) = \max_{S}{}^*[\beta_k(s_k) + \gamma_k(s_{k-1} \to s_k)] \tag{8}$$

$$L_R(u_k) = \max_{U^1}{}^*[\alpha_{k-1}(s_{k-1}) + \gamma_k(s_{k-1} \to s_k) + \beta_k(s_k)] - \max_{U^0}{}^*[\alpha_{k-1}(s_{k-1}) + \gamma_k(s_{k-1} \to s_k) + \beta_k(s_k)] \tag{9}$$

The performance of the log-MAP algorithm is identical to that of the MAP algorithm. As computing $\ln(\cdot)$ in Eq. (6) is complex, a look-up table is usually used to simplify the computation.

3. Proposed Reverse Calculation for Max-log-MAP Decoding

As shown in the previous section, the calculation direction of the forward metrics is different from that of the backward metrics. The forward metrics are initialized at the starting point and calculated in the forward direction. On the other hand, the backward metrics initialized at the ending point progress in the backward direction. Since the calculation directions of the two metrics are different, it is impossible to calculate both metrics simultaneously. All the forward (backward) metrics have to be calculated first and saved in a memory. Then, the saved forward (backward) metrics are loaded from the memory in order to compute $L_R(u_k)$ with the calculated backward (forward) metrics. Due to the memory accesses, a turbo decoder suffers from large power consumption. If we can calculate the forward (or backward) metrics in the reverse direction, there is no need to access the memory any more, saving power consumption. Assuming that forward metrics are calculated prior to backward metrics, we will present in this paper an efficient method to reversely calculate the forward metrics.

In a binary system, two branch transitions appear as a butterfly pair if the first and the last shift register are connected both in the feedback and feed-forward polynomials, and good RSC encoders always satisfy this condition [8]. Since the W-CDMA turbo encoder has such a butterfly structure as shown in Fig. 1, the trellis diagram can be grouped into butterfly pairs. Figure 3(a) shows a trellis section, where the dotted line means the transition by a systematic bit of 0, and the solid line represents the transition induced by a systematic bit of 1. A re-arranged version of the trellis section is drawn in Fig. 3(b) to clearly show four butterfly pairs.

Let us consider one butterfly pair shown in Fig. 4, $(\alpha_k^0, \alpha_k^4, \alpha_{k+1}^0, \alpha_{k+1}^1)$. Even though there are four branches in a butterfly pair, there exist only two distinguishable branch metric values for a butterfly pair [8]. The branch metric values, γ_p and γ_c, are always different except the case of

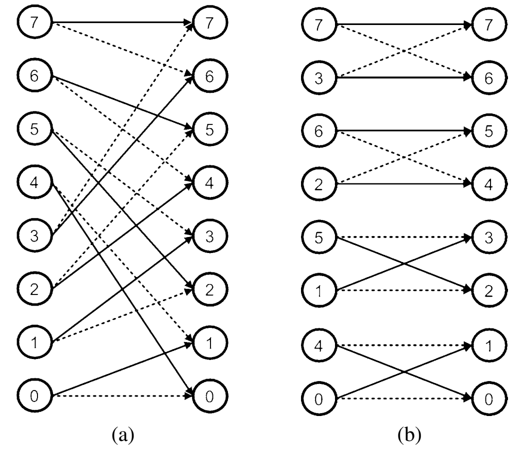

Fig. 3 Trellis section for W-CDMA turbo encoder. (a) Trellis section. (b) Re-arranged trellis section.

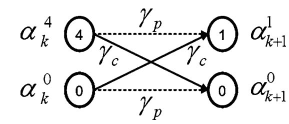

Fig. 4 One butterfly pair.

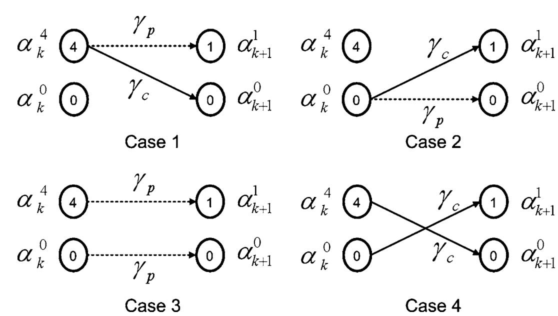

Fig. 5 Four cases of forward metric calculation.

$\gamma_p = \gamma_c = 0$. In the Max-log-MAP decoding, α_{k+1}^0 and α_{k+1}^1 are calculated from α_k^0 and α_k^4 as

$$\begin{aligned} \alpha_{k+1}^1 &= \max(\alpha_k^4 + \gamma_p, \alpha_k^0 + \gamma_c) \\ \alpha_{k+1}^0 &= \max(\alpha_k^4 + \gamma_c, \alpha_k^0 + \gamma_p) \end{aligned} \tag{10}$$

After calculating α_{k+1}^0 and α_{k+1}^1, α_k^0 and α_k^4 are stored into the memory. To calculate $L_R(u_k)$, they are loaded later from the memory. If α_k^0 and α_k^4 can be calculated from α_{k+1}^0 and α_{k+1}^1 in the same direction as backward metrics, the memory access can be substituted by calculations. The rest of this section describes how to calculate the forward metrics in the backward direction.

From Fig. 4 and Eq. (10), the decision of α_{k+1}^0 and α_{k+1}^1 can be categorized into 4 cases as shown in Fig. 5.

1) Case 1
The values of α_{k+1}^0 and α_{k+1}^1 are determined by α_k^4 as follows.

$$\begin{aligned} \alpha_{k+1}^1 &= \alpha_k^4 + \gamma_p \\ \alpha_{k+1}^0 &= \alpha_k^4 + \gamma_c \end{aligned} \tag{11}$$

Equation (11) can be rewritten as

$$\alpha_k^4 = \alpha_{k+1}^1 - \gamma_p$$
$$\alpha_k^4 = \alpha_{k+1}^0 - \gamma_c \tag{12}$$

From Eq. (12), we can see that if α_{k+1}^0 and α_{k+1}^1 come from α_k^4, $\alpha_{k+1}^1 - \gamma_p$ and $\alpha_{k+1}^0 - \gamma_c$ are equal to the value of α_k^4, as shown below.

$$\alpha_{k+1}^1 - \gamma_p = \alpha_{k+1}^0 - \gamma_c = \alpha_k^4 \tag{13}$$

2) Case 2
The values of α_{k+1}^0 and α_{k+1}^1 are determined by α_k^0. This case is very similar to case 1, and we can see that if α_{k+1}^0 and α_{k+1}^1 come from α_k^0, $\alpha_{k+1}^1 - \gamma_c$ and $\alpha_{k+1}^0 - \gamma_p$ are equal to the value of α_k^0.

$$\alpha_{k+1}^1 - \gamma_c = \alpha_{k+1}^0 - \gamma_p = \alpha_k^0 \tag{14}$$

3) Case 3
The values of α_{k+1}^0 and α_{k+1}^1 come from α_k^0 and α_k^4 as follows.

$$\alpha_{k+1}^1 = \alpha_k^4 + \gamma_p$$
$$\alpha_{k+1}^0 = \alpha_k^0 + \gamma_p \tag{15}$$

Equation (15) can be rewritten as

$$\alpha_k^4 = \alpha_{k+1}^1 - \gamma_p$$
$$\alpha_k^0 = \alpha_{k+1}^0 - \gamma_p \tag{16}$$

From Eqs. (10) and (15), the followings are derived.

$$\alpha_{k+1}^1 > \alpha_k^0 + \gamma_c$$
$$\alpha_{k+1}^0 > \alpha_k^4 + \gamma_c \tag{17}$$

From Eqs. (16) and (17), the following expressions come out.

$$\alpha_{k+1}^1 - \gamma_c > \alpha_{k+1}^0 - \gamma_p = \alpha_k^0$$
$$\alpha_{k+1}^0 - \gamma_c > \alpha_{k+1}^1 - \gamma_p = \alpha_k^4 \tag{18}$$

From Eq. (18), we can see that between two paths connected to each α_k, the smaller one is the value of α_k.

4) Case 4
The values of α_{k+1}^0 and α_{k+1}^1 come from α_k^4 and α_k^0. This case is analogous to case 3, and we can derive the following expressions.

$$\alpha_{k+1}^1 - \gamma_p > \alpha_{k+1}^0 - \gamma_c = \alpha_k^4$$
$$\alpha_{k+1}^0 - \gamma_p > \alpha_{k+1}^1 - \gamma_c = \alpha_k^0 \tag{19}$$

From Eq. (19), the smaller $\alpha_{k+1} - \gamma$ is the value of α_k as in case 3.

3.1 Strategy for Reverse Calculation

Among the four cases, case 1 and case 2 can be satisfied simultaneously if and only if $\gamma_p = \gamma_c = 0$. To avoid such a situation, although the situation is extremely rare, a zero-valued branch metric is replaced by a value of minimal non-zero magnitude. The replacement makes the four cases unique and experimental results show that the performance influence induced by the replacement is ignorable. Looking at the above four cases, we can calculate the forward metrics in the reverse direction, as is summarized below.

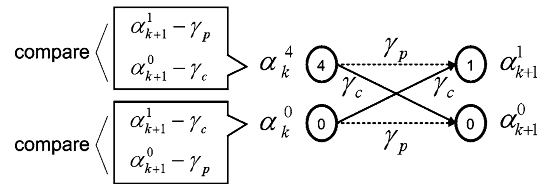

Fig. 6 The strategy for reverse calculation.

- When we calculate α_{k+1} from α_k, the only α_k value which never wins in the max competitions of Eq. (10) is saved into the memory. If two values for a max function are equal, the loser in the other max function is saved in the memory. Later when we need α_k, we calculate α_k from α_{k+1} as described below.
- First, we calculate two $\alpha_{k+1} - \gamma$ values on the paths connected to a state of α_k, as shown in Fig. 6.
- If these values are equal, then they are the value of α_k as indicated in Eqs. (13) and (14). And the value of α_k of the other state in the butterfly pair has to be loaded from the memory. The latter α_k must be in the memory because it was the loser in the max competition.
- If these values are different in both the states of a butterfly pair, each state α_k takes the smaller value of two $\alpha_{k+1} - \gamma$ values, as shown in Eqs. (18) and (19). Since in this case the two α_k's can be reversely calculated from the two α_{k+1}'s belonging to the butterfly pair, there is no need to store any α_k's into the memory.

This strategy stores, in the worst case, only one α_k in the memory for a butterfly pair, instead of two α_k's required in the conventional turbo decoding. Therefore the required memory size is reduced to a half of the original memory size, and the number of memory accesses is reduced drastically. Compared to the loser storing method [9] which always accesses the memory to read the position and the value of the loser, the proposed strategy requires memory accesses only when the loser cannot be recovered by the reverse calculation, reducing the number of memory accesses significantly than the loser storing method.

4. Hybrid log-MAP Algorithm

The hardware complexity of the Max-log-MAP algorithm is lower than that of the log-MAP algorithm, and the former algorithm is more tolerant to the channel estimation error. However, if the channel noise is properly estimated, the log-MAP algorithm outperforms the Max-log-MAP algorithm. The log-MAP algorithm gives exactly the same performance as the MAP algorithm, but the Max-log-MAP algorithm gives a slight degradation in performance compared to the MAP algorithm. If we can have reliable channel estimation and do not take into account the higher complexity, the log-MAP algorithm will be the best choice. In the log-MAP algorithm, all the max functions used in the Max-log-MAP algorithm are replaced with max* functions defined in Eq. (6).

Due to the correction factor of the max* function, the reverse calculation of state metrics in the log-MAP algorithm is much more difficult than the Max-log-MAP algorithm. In [10], a reverse calculation method is applied to the log-MAP algorithm, but the logic complexity is increased by 40%.

In this section, we propose a new decoding algorithm called hybrid log-MAP, which gives almost the same performance as the conventional log-MAP algorithm. The reverse calculation method proposed for the Max-log-MAP decoding is also applied to the hybrid log-MAP algorithm.

The reverse calculation method for the Max-log-MAP decoding can be applied to the log-MAP decoding if all the max* functions used for the forward metric calculation in the log-MAP algorithm are replaced with the max functions. This replacement makes the forward metric calculation equal in both of the log-MAP and Max-log-MAP algorithms. The replaced max function leads to performance degradation in the log-MAP algorithm. Since the backward metric calculation and $L_R(u_k)$ calculation are still based on the max* function even if the forward metric calculation is replaced with the max function, the performance of the hybrid log-MAP decoding will be placed between the conventional log-MAP and Max-log-MAP decodings.

In addition, more performance improvement is expected in the proposed hybrid log-MAP decoding if it is associated with the scaling technique [17], [18]. Recent works proved that scaling the extrinsic information improves the performance of Max-log-MAP algorithm by more than 0.2 dB. As the hybrid log-MAP decoding shares some characteristics with the Max-log-MAP decoding because of the replaced max function in the forward metric calculation, the extrinsic information scaling can be employed to improve the performance of the hybrid log-MAP decoding.

5. Experimental Results

Based on the proposed hybrid log-MAP decoding, a turbo decoder of which specification is presented in Table 1 was described in Verilog-HDL and synthesized using a 0.18 μm standard-cell library and compiled SRAM memories. In Table 1, (q,f) denotes a quantization scheme that uses q bits in total and f bits to represent the fractional part. As the fractional part of the branch metric is represented by 2 bits, zero-valued branch metrics are replaced with 0.25, the value of minimal non-zero magnitude, in the proposed reverse calculation.

The proposed decoder employs the sliding window technique [19] to reduce the size of memory required. The timing diagram for the sliding window technique is shown in Fig. 7. The window size is set to 32, which is 8 times of

Table 1 Specification of the proposed hybrid log-MAP decoder.

Application	W-CDMA
SISO algorithm	Sliding window Hybrid log-MAP
Block size	1024
Sliding Window size	32
The number of Iteration	8 (Fixed)
Scaling factor	0.875
Quantization	Received input: (4,2) Extrinsic Information : (6,2) Branch metric : (7,2) State metric : (9,2)
Normalization	No
Process	0.18 μm process

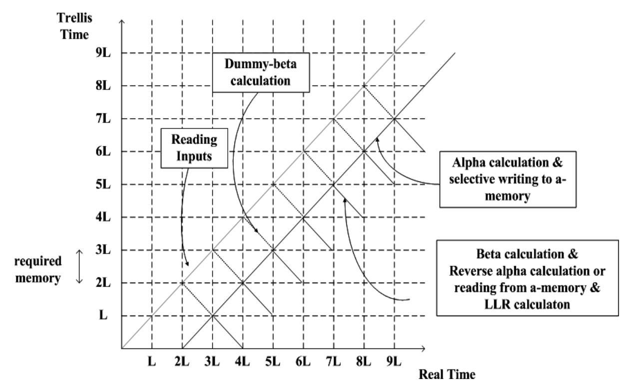

Fig. 7 Timing diagram of the proposed decoder associated with the sliding window technique.

the constraint length of W-CDMA turbo code. The window size of 32 gives almost the same performance as the ideal case associated with an infinite window.

Since at each updating step branch metrics are added to the current metrics and the largest value is selected as a new metric for each state, the state metric values increase as their calculations are progressed according to the calculation direction. To avoid arithmetic overflow, a normalization scheme is required, leading to additional hardware and latency. However, the normalization can be avoided by using the 2's complement arithmetic and setting its bit width to k to take into account the upper bound of metric difference [19].

$$k = \lceil \log_2 \Delta_{\mathrm{MAX}} \rceil + 1 \tag{20}$$

In Eq. (20), Δ_{MAX} is the upper bound of metric difference between two path metrics of a butterfly. The value of Δ_{MAX} for W-CDMA turbo code is given to 256 in [20]. Hence, the bit width of $k = 9$ is sufficient for the internal SISO operation.

Table 2 shows the memory specification of the proposed decoder, comparing with the conventional decoder and the loser storing method [9]. Since $L_e(u_k)$ and $L_c y_k^s$ are always used together as shown in Eqs. (4) and (5), we

Table 2 Memory size comparison of a SISO decoder for single-port SRAM.

		Conventional log-MAP decoder	Loser Storing Method [9]	Proposed Hybrid log-MAP decoder
Branch memory		4 banks, 32*10 bits/ bank	4 banks, 32*10 bits/ bank	4 banks, 32*10 bits/ bank
		1280 bits	1280 bits	1280 bits
Alpha state memory		2 banks, 32*(9*8) bits/bank	8 banks, 32*9 bits/ bank	8 banks, 32*9 bits/bank
		4608 bits	2304 bits	2304 bits
Flag memory		None	4 banks, 32bits/bank	None
			128 bits	
Total		5888 bits	3712 bits	3584 bits
		100%	63.0%	60.8%

save $(L_{e,k} + L_c y_k^s)/2$ and $L_c y_k^p/2$ in the branch memory, instead of separate instances, $L_{e,k}$ and $L_c y_k^s$. Hence, the word size of branch memory is determined to 10 bits: 6 bits for $(L_{e,k} + L_c y_k^s)/2$ and 4 bits for $L_c y_k^p/2$. Since there are 4 processes as shown in Fig. 7, the whole branch memory is partitioned into 4 memory banks to make every process access data at the same time without causing conflicts, one for storing the inputs coming from the channel and three for dummy beta, beta, and alpha calculations. As usual, the alpha memory comprises 2 banks, one for storing the value during the alpha calculation and the other for reading the value for the $L_R(u_k)$ calculation, as shown in Fig. 7. In this case, the word size of each bank is set to 72 bits to store 8 state metrics. To allow each butterfly pair individual access to a memory, we use a separate bank for each butterfly pair. Totally, the alpha memory comprises 8 banks of word size 9 bits in the proposed SISO decoder. Employing the proposed reverse calculation scheme reduces the state memory size by 50%. Therefore, the total memory size is reduced by 39.2% as shown in Table 2.

Figure 8 shows the overall architecture of the proposed SISO decoder. The branch metric input stream is fed into several units for parallel processing as shown in Fig. 7. As beta metrics are calculated in the opposite direction, the dummy beta unit is needed to decide the initial values of beta metrics, and its architecture is identical to the beta unit.

Figure 9 compares the numbers of alpha metrics loaded from the memory in the proposed hybrid log-MAP decoding and in the loser storing method [9], where the number is normalized by the number of alpha metrics loaded in the conventional log-MAP decoding. The loser storing method reduces the memory accesses by 50% because only one of two metrics in a butterfly is loaded, while the proposed reverse calculation scheme reduces the memory accesses by about 80% because there is no need to access the memory if both the metrics in a butterfly can be reversely calculated. As the memory access is tightly related to the power consumption, the more reduction of the memory accesses leads to the less power consumption.

The power consumption of the proposed SISO decoder

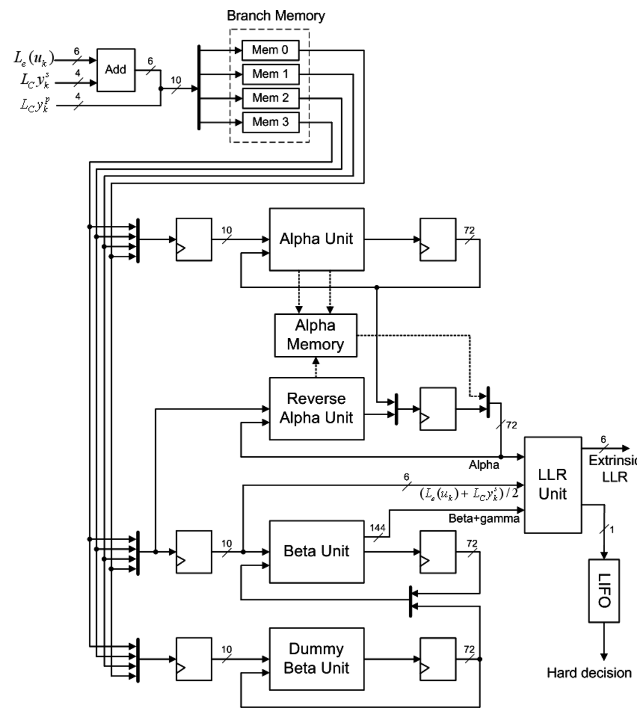

Fig. 8 Entire architecture of the proposed SISO decoder.

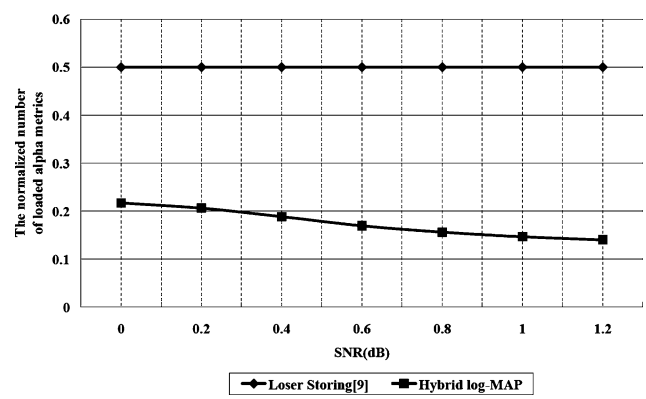

Fig. 9 The normalized number of loaded alpha metrics.

is presented in Table 3, compared to the conventional decoder and the loser storing method. The power consumption of the state metric memory is reduced by 75%, and the total power consumption is reduced by 34.4%. With a higher SNR and a larger number of iterations, the proposed decoder will consume less power because more alpha metrics can be calculated reversely, while the conventional decoder consumes a fixed power. In the case of the loser storing method, more power consumption in the state memory is required due to more frequent memory accesses as shown in Fig. 9. As the SISO logic complexity of the hybrid log-MAP decoder is comparable to that of the loser storing method, the less total power consumption is expected due to the less power consumption in the state memory.

Table 4 shows that the proposed decoder is designed with 21820 gates, while the conventional decoder requires

Table 3 Power comparison of SISO decoders in case of 1.2 dB SNR.

	Conventional log-MAP decoder	Loser Storing Method [9]	Proposed hybrid log-MAP decoder
Branch memory	274.74 μW/MHz	274.74 μW/MHz	274.74 μW/MHz
Alpha State memory	573.34 μW/MHz	261.80 μW/MHz	146.59 μW/MHz
SISO logic	348.47 μW/MHz	N.A.	364.43 μW/MHz
Total	1196.55 μW/MHz	N.A.	785.76 μW/MHz
	100%	N.A.	65.60%

Table 4 Comparison of area and critical path delay.

	Conventional log-MAP decoder	Proposed hybrid log-MAP decoder
Logic Gates	18004	21820
Critical path delay	4.74 ns	4.74 ns

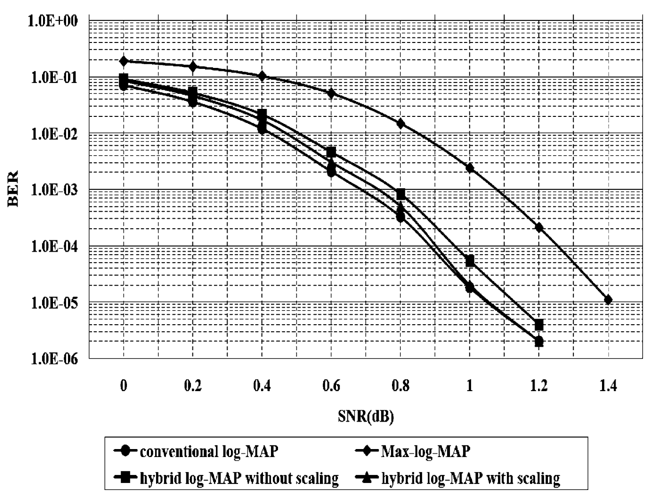

Fig. 10 Performance comparison of four algorithms.

18004 gates when they are synthesized under the constraint of the same operation frequency of 200 MHz. The logic complexity of the proposed decoder is increased by 21.2%. The critical path delay is 4.74 ns. As a result, the proposed decoder can operate at approximately 200 MHz, which is enough to meet the W-CDMA standard specification of 2 Mbps.

The BER performances of the Max-log-MAP, conventional log-MAP and hybrid log-MAP algorithms are presented in Fig. 10. The extrinsic information scaling improves the performance of the hybrid log-MAP decoding. Simulating on various scaling factors, we found that the hybrid log-MAP algorithm gives the best performance when the scaling factor is close to 0.9. Considering finite precision, we set the scaling factor to 0.875 as presented in Table 1. The hybrid log-MAP with the scaling factor shows almost the same performance as that of the conventional log-MAP algorithm. In the worst case, the performance of the proposed decoder associated with scaling is only 0.05 dB lower than the conventional log-MAP decoder. Without scaling, the performance of the hybrid log-MAP algorithm is 0.08 dB worse than the conventional log-MAP algorithm and 0.27 dB better than the Max-log-MAP algorithm for a BER of 10^{-5}.

6. Conclusion

This paper has presented a new efficient reverse calculation method to reduce power consumption by minimizing memory accesses required in turbo decoding. As the turbo code has butterfly structures in its trellis diagram, the reverse calculation is possible and simple especially for the Max-log-MAP decoding. First, we presented an efficient reverse calculation method for the Max-log-MAP decoding. Then, to apply the same reverse calculation to the log-MAP algorithm, we modified the log-MAP algorithm by replacing the max* operations needed in the forward metric calculation with the max operation and by scaling the extrinsic information. The hybrid log-MAP algorithm shows almost the same performance as the log-MAP algorithm.

Experimental results show that about 80% forward metrics can be reversely calculated in the W-CDMA standard. By employing the reverse calculation method in the hybrid log-MAP turbo decoder, power consumption and memory size are reduced by approximately 35% and 40%, respectively, compared to the conventional log-MAP turbo decoder. Although the proposed method is simulated and implemented for the W-CDMA standard, it can also be applied to other applications.

Acknowledgments

This work was supported by Institute of Information Technology Assessment through the ITRC and by the Korea Science and Engineering Foundation through MICROS center 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," Proc. Int. Conf. Commun., pp.1064–1070, May 1993.
[2] C. Schurgers, F. Catthoor, and M. Engels, "Memory optimization of MAP turbo decoder algorithms," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol.9, no.2, pp.305–312, April 2001.
[3] Y. Wu, B.D. Woerner, and W.J. Ebel, "A simple stopping criterion for turbo decoding," IEEE Commun. Lett., vol.4, pp.258–260, Aug. 2000.
[4] D. Garrett, B. Xu, and C. Nicol, "Energy efficient turbo decoding for 3G mobile," Proc. IEEE Int. Symp. Low Power Electronics Design (ISLPED), pp.328–333, Aug. 2001.
[5] S.J. Lee, N.R. Shanbhan, and A.C. Singer, "A low-power VLSI architecture for turbo decoding," Proc. IEEE Int. Symp. Low Power Electronics Design (ISLPED), pp.366–371, Aug. 2003.

[6] 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.
[7] G. Maser, M. Mazza, G. Piccinini, F. Viglione, and M. Zamboni, "Architectural strategies for low-power VLSI turbo decoders," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol.10, no.3, pp.279–285, June 2002.
[8] Y. Wu, W.J. Ebel, and B.D. Woerner, "Forward computation of backward path metrics for MAP decoders," Proc. VTC, pp.2257–2261, 2000.
[9] J. Kwak, S.M. Park, and K. Lee, "Reverse tracing of forward state metric in log-MAP and Max-log-MAP decoders with fixed point precision," IEICE Trans. Commun., vol.E86-B, no.1, pp.451–455, Jan. 2003.
[10] D.S. Lee and I.C. Park, "Low-power log-MAP turbo decoding based on reduced metric memory access," Proc. IEEE Int. Symp. On Circuits and Systems, pp.3167–3170, May 2005.
[11] P. McAdam, L. Welch, and C. Weber, "MAP bit decoding of convolutional codes," Int. Symp. on Inform. Theory, p.91, 1972.
[12] J. Hagenauer and P. Hober, "A Viterbi algorithm with soft-decision outputs and its applications," IEEE GLOBECOM, pp.47.1.1–47.1.7, Nov. 1989.
[13] W. Koch and A. Baier, "Optimum and sub-optimum detection of coded data disturbed by time-varying inter-symbol interference," IEEE GLOBECOM, pp.1679–1684, Dec. 1990.
[14] J.A. Erfanian, S. Pasupathy, and G. Gulak, "Reduced complexity symbol detectors with parallel structures for ISI Channels," IEEE Trans. Commun., vol.42, no.2/3/4, pp.1661–1671, 1994.
[15] S.S. Pietrobon and A.S. Barbulescu, "A simplification of the modified Bahl algorithm for systematic convolutional codes," Proc. ISITA, pp.1073–1077, Nov. 1994.
[16] P. Robertson, E. Villebrun, and P. Hoeher, "A comparison of optimal and sub-optimal MAP decoding algorithms operating in the log domain," Proc. ICC, pp.1009–1013, June 1995.
[17] J. Vogt and A. Finger, "Improving the max-log-MAP turbo decoder," Electron. Lett., vol.36, no.23, pp.1937–1939, Nov. 2000.
[18] H. Claussen, H.R. Karimi, and B. Mulgrew, "Improved max-log-MAP turbo decoding using maximum mutual information combining," IEEE Int. Symp. on Personal, Indoor and Mobile Radio Comm. (PIMRC), vol.1, pp.424–428, 2003.
[19] G. Masera, G. Piccinini, M.R. Roch, and M. Zamboni, "VLSI architectures for turbo codes," IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol.7, no.3, pp.369–379, Sept. 1999.
[20] G. Montorsi and S. Benedetto, "Design of fixed-point iterative decoders for concatenated codes with interleavers," IEEE J. Sel. Areas Commun., vol.19, no.5, pp.871–882, May 2001.

Hye-Mi Choi received the B.S. and M.S. degrees in electronics engineering from Korea Advanced Institute of Science and Technology (KAIST) in 2003 and in 2005, respectively. Her research interests include VLSI design for communication and SoC.

Ji-Hoon Kim received the B.S. degree in electronics engineering from Korea Advanced Institute of Science and Technology (KAIST) in 2004. Currently, he is a M.S. student in the Department of Electrical Engineering and Computer Science at KAIST. His research interests include VLSI design for communication and low power circuit design.

In-Cheol Park received the B.S. degree in electronics engineering from Seoul National University, in 1986, and the M.S. and Ph.D. degrees in electrical engineering from the Korea Advanced Institute of Science and Technology (KAIST) in 1988, 1992, respectively. Since June 1996, he has been an Assistant Professor and now a Professor in the Department of Electrical Engineering and Computer Science at KAIST. Prior to joining KAIST, he was with IBM T.J. Watson Research Center, Yorktown, NY, from May 1995 to May 1996, where he researched high-speed circuit design. His current research interests include computer-aided design algorithm for high-level synthesis and very large scale integration architectures for general-purpose microprocessors. Dr. Park received the best paper award at the ICCD in 1999 and the best design award at the ASP-DAC in 1997.