High-Performance Programmable SISO Decoder VLSI Implementation for Decoding Turbo Codes

Toshiyuki Miyauchi, Kouhei Yamamoto, Takashi Yokokawa
Makiko Kan, Yuichi Mizutani, Masayuki Hattori
Sony Corporation, Japan
Phone: +81-3-5435-3541, Fax: +81-3-5435-3553
e-mail: miyauchi@av.crl.sony.co.jp

Abstract—We developed a high-performance programmable SISO decoder LSI for decoding Turbo codes based on the SW-Log-BCJR algorithm. This LSI is based on the original architecture and memory management method, which guarantees the order of the soft-output to be the same as soft-input without attaching LIFO memory. Moreover, we propose new accurate implementation on 4-input Log-Sum operations used in the recursion of $\alpha$ and $\beta$ for codes of rate $2/3$ and $3/3$. This technique enables high operating frequency and low coding loss simultaneously. The interleaver and inevitable delay lines required to realize the Turbo decoders are embedded on the chip so the most Turbo code applications, including PCCC, SCCC, TTCM and SCTCM, can easily be implemented scalably by cascading this LSI. The LSI is fully programmable for code types, code polynomials and interleaver structures and applicable for BPSK, QPSK and 8-PSK with arbitrary signal constellation. Operating frequency of $100$[MHz] is achieved using CMOS $0.25$[μm] process whereas the coding loss according to implementation is kept within $0.03$[dB].

I. INTRODUCTION

Turbo codes have been applied to various kinds of communication systems and storage systems since their first appearance [1]. Recent low-speed communication systems, including the Consultative Committee for Space Data Systems (CCSDS) telemetry channel coding [14] and the 3rd Generation Partnership Project (3GPP) [15], have already standardized Turbo codes as their channel coding systems. As these applications emerge, there has been a strong requirement for the efficient implantation of Soft-In Soft-Out (SISO) decoders, which are the key components for decoding Turbo codes. Moreover, for recent broadband applications, these decoders must be high-speed, cascadable [2], and programmable, i.e., capable of programming variable code polynomials as well as interleaver structures and patterns. In addition, the inevitable coding losses due to implementations and quantizations should be minimized.

There have been many proposals [2], [3], [4], [5], [6], [7] for implementing SISO decoders. Most of these are targeted for applications on PLDs or DSPs. Some of them have already been available on products [6], [7]. However, in order to realize Turbo decoding systems for broadband applications, SISO decoders should be high-speed and high-throughput without significant coding loss. Moreover, such high-speed Turbo decoding systems should be scalable, i.e., the performance of the decoder can easily be changed by the number of cascaded SISO decoders (processing elements) as in systoric array structures. But little research has been done for implementing such cascadable SISO decoders in ASICs to realize high-speed Turbo decoding systems since [2], which introduced sub-optimal Soft-Output Viterbi Algorithm in exchange of inevitable performance degradation.

In this paper, we apply the Sliding-Window Log-BCJR (also known as SW-Log-MAP) algorithm [9], [10], [11] to avoid the performance degradation, moreover we modify this algorithm and propose an original SISO decoder architecture and memory management method, so that the LIFO memories required in previous research can be omitted. We also propose a new implementation for fast and accurate arithmetic operations for $\alpha$ and $\beta$ [8] log-likelihoods, especially for codes with 2 or more input bits, e.g. codes of rate $2/3$ and $3/3$. Note that patents have been filed by Sony Corp. for these proposed methods.

Based on these techniques, we developed a very high-performance SISO decoder LSI, which can be applied to Parallel Concatenated Convolutional Codes (PCCC) [1], Serial Concatenated Convolutional Codes (SCCC) [16], Turbo Trellis Coded Modulation (TTCM) [17] and Serial Concatenated Trellis Coded Modulation (SCTCM) [18]. This LSI is programmable, cascadable, high operating frequency ($100$[MHz]) and has very low coding losses (within $0.03$[dB]) for all Turbo code applications.

We begin by introducing the main characteristics and a block diagram of the designed decoder LSI, and describing decodable code polynomials. Next, we describe our method to compute soft-output efficiently and accurately. Finally, we show several applications and decoding performance curves for AWGN channels.

II. MAIN CHARACTERISTICS

- programmable code polynomials
- on chip interleavers/deinterleavers (maximum 16K)
- capable of 1–3 symbol-wise interleave
- arbitrary interleaver patterns
- adapt for 8PSK-TCM applications
- arbitrary signal constellation configurations
- symbol-wise soft-output (code type B and C)
III. LSI ARCHITECTURE

The block diagram for the LSI is shown in Fig. 1. Fig. 2 illustrates in detail the SISO decoder block in Fig. 1.

![Fig. 1. LSI block diagram](image)

![Fig. 2. SISO decoder block diagram](image)

IV. DECODABLE CODE POLYNOMIALS

The designed LSI is capable of decoding the following four types of codes (A–D) and their arbitrary punctured codes. $G$ is a generator matrix of the code and $R$ stands for the rate of the code. Maximum memory lengths of the codes are different for each type. Note that these four types of codes cover most of the component codes for the Turbo code family. Code polynomials are configured by setting the entries of generator matrices through configuration bus.

(A) controller canonical form: $R = 1/1, 1/2, 1/3, 1/4$

$$G = \begin{bmatrix} G_0 & G_1(D) & G_2(D) & G_3(D) \\ G_B(D) & G_B(D) & G_B(D) & G_B(D) \end{bmatrix}$$

$G_0 : 0$ or $1$

$\deg(G_1(D)), \deg(G_2(D)), \deg(G_3(D)) \leq 4$

$\deg(G_B(D)) \leq 4$. The constant term must be $1$.

(B) controller canonical form: $R = 2/3$

$$G = \begin{bmatrix} G_{11}(D) & G_{21}(D) & G_{31}(D) \\ G_{12}(D) & G_{22}(D) & G_{32}(D) \end{bmatrix}$$

$\deg(G_{11}(D)), \deg(G_{21}(D)), \deg(G_{31}(D)) \leq 2$

$\deg(G_{12}(D)), \deg(G_{22}(D)), \deg(G_{32}(D)) \leq 1$

(C) observer canonical form: $R = 2/3$

$$G = \begin{bmatrix} 1 & 0 & G_1(D)/G_B(D) \\ 0 & 1 & G_2(D)/G_B(D) \end{bmatrix}$$

$\deg(G_1(D)), \deg(G_2(D)) \leq 3$

$\deg(G_B(D)) \leq 3$. The constant term must be $1$.

(D) observer canonical form: $R = 3/3$

$$G = \begin{bmatrix} 1 & 0 & G_1(D)/G_B(D) \\ 0 & 1 & G_2(D)/G_B(D) \\ 0 & 0 & G_3(D)/G_B(D) \end{bmatrix}$$

$\deg(G_1(D)), \deg(G_2(D)), \deg(G_3(D)) \leq 2$

$\deg(G_B(D)) \leq 2$. The constant term must be $1$.

V. EFFICIENT MEMORY MANAGEMENT METHOD

It is well-known that the BCJR algorithm [8] is the optimal algorithm for obtaining exact soft-output values. Here, we use the same notation$^1$ of $\alpha$, $\beta$ and $\gamma$ as [8]. For LSI implementation, we apply the so-called Log-BCJR algorithm (e.g., [13]), which is equivalent to the BCJR algorithm. But instead of treating probabilities directly, the Log-BCJR algorithm uses log-likelihoods and operates in logarithm domain. This converts the probability multiplications to log-likelihood additions, so that the computational complexity is reduced significantly. However, in order to implement this into an LSI, we need efficient scheduling and data storage management so that all the arithmetic logic units are fully occupied, the size of the storage is minimized and the delay time for decoding is minimized.

In order to solve this essential problem, there have been several implementations based on the sliding-window technique [10], [11], by using banks of RAMs and trace-back technique, which is popular for realizing long path-memories for Viterbi decoders [12]. Note that these implementations require Last-In-First-Out memory (LIFO) at the final stage of the SISO decoders. Some of these also require dual-port RAMs.

Here, we propose a new implementation and memory management method as shown in Fig. 3 and 4. Let $L(=128)$ be the truncation length. $\gamma$ is computed from the soft-input

$^1$These notations express probabilities in the original paper but here we use these notations as log-likelihood.
stored in five banks of RAMs. As shown in Fig. 3, β is computed in parallel by using two banks of RAMs for realizing a sliding-window. Starting time of the two parallel β computations is shifted as L time slots. Each of these computations corresponds to backward recursion for β and updates for 2L time slots in reverse order. This technique itself is the same as in [10], [11], but we do not compute soft-output from obtained β. Rather, we “store” β in RAMs only for the latter half of the 2L time slots.

Simultaneously, forward recursion for computing α is executed based on the 4L-time-slot delayed version of γ. The final soft-output is computed from these α, stored β, γ and delayed soft-input. The scheduling of this memory management method is illustrated in Fig. 4. Note that the soft-output can be obtained in normal order (same as input order) with our structure and scheduling.

\[
\begin{align*}
X_{i0} & \overset{\text{def}}{=} \max\{X_0, X_1, X_2, X_3\} \\
\Delta & \overset{\text{def}}{=} \min\{X_{i0} - X_i\},
\end{align*}
\]

and approximate as follows.

\[
\ln(e^{X_0}, e^{X_1}, e^{X_2}, e^{X_3}) \approx X_{i0} + \ln(1 + e^{-\Delta})
\]

From our analyses and simulations, the coding loss of this implementation is negligible. For all of the Turbo decoders that can be realized by our LSI, the coding loss due to the approximation of the Log-Sum operation is less than 0.01(dB). As a result, high operating frequency and low coding loss are realized simultaneously.

VI. THE α AND β RECURSIONS

It is essential in the Log-BCJR algorithm to implement what we call “Log-Sum” operations accurately for updates of α and β recursions. This impacts the performance of overall Turbo codes drastically. For codes of type A, we realize Log-Sum function with MAX operation and correction function as shown in Fig. 5. The correction function is realized by a look-up table.

\[
\max(X,Y) + \ln(1 + \exp(-|X-Y|))
\]

In the trellis diagrams for codes of type B and type C, four branches are merged to one state as shown in Fig. 6(a). Therefore, we need 4-input Log-Sum functions for these types of codes for computing α and β.

\[
\begin{align*}
X_{i0} & \overset{\text{def}}{=} \max\{X_0, X_1, X_2, X_3\} \\
\Delta & \overset{\text{def}}{=} \min\{X_{i0} - X_i\},
\end{align*}
\]

and approximate as follows.

\[
\ln(e^{X_0}, e^{X_1}, e^{X_2}, e^{X_3}) \approx X_{i0} + \ln(1 + e^{-\Delta})
\]

From our analyses and simulations, the coding loss of this implementation is negligible. For all of the Turbo decoders that can be realized by our LSI, the coding loss due to the approximation of the Log-Sum operation is less than 0.01(dB). As a result, high operating frequency and low coding loss are realized simultaneously.
Fig. 7. The $\alpha$ recursion for 4-input Log-Sum

For codes of type D, eight branches merge at each state. However, since we restrict the length of the memory to less than two in this type, the maximum number of states is only four. Hence, there must be a parallel path for this case as shown in Fig 6(b). Therefore, by adding the likelihood corresponding to this parallel path to $\gamma$ in advance, we can use the same 4-input Log-Sum functions.

VII. Turbo Decoder Implementation

A. Applications

The designed LSI consists of an interleaver, which is capable of programming any interleave patterns, as well as a delay line sufficient to compensate for the delay of the SISO decoder and interleaver. Each symbol of soft-inputs and soft-outputs is quantized in 8[bit]. The interleaver is capable of interleaving 3[symbol] simultaneously according to the Turbo code constructions. We briefly list the examples of applications that can be realized by the LSI.

1. PCCC: CCSDS standard PCCC [14](Fig.8)

Other PCCC, e.g., 3GPP standard [15] or Berrou Code [1] can also be realized. These applications consist of two type A codes. Maximum interleaver size is 16K×1[symbol].

2. SCCC: Benedetto SCCC [16](Fig.9)

The application consists of type A outer code and type C inner code. Maximum interleaver size is 8K×2[symbol].

3. TTCM: Robertson TCM [17](Fig.10)

The application consists of two type C codes. Maximum interleaver size is 16K×3[symbol](pairwise).

4. SCTCM: Divsalar SCTCM [18](Fig.11)

The application consists of type B or C outer code and type D inner code. Maximum interleaver size is 4K×3[symbol].

B. Decoder Structure & Performance

The decoders corresponding to the applications given in section VII. A can be implemented by just cascading the LSI enough to achieve the required system performance. All these decoders can be implemented without modifying connections between LSIs (Fig. 12).

Fig. 8. CCSDS encoder

Fig. 9. SCCC encoder

Fig. 10. TTCM encoder

Fig. 11. SCTCM encoder

Fig. 12. Turbo decoding system
The performance curves of CCSDS PCCC ($N=8920$, $R = 1/3$,$-1/6$, 20 iterations) over the AWGN channel is illustrated in Fig. 13. Fig. 14 shows the performance for Robertson TTCM ($N=8192$, 20 iterations). We can see that the coding losses are less than 0.03[dB] compared to floating-point simulations (f.p. in the figures).

VIII. Conclusion

We designed a high-speed programmable SISO decoder LSI for decoding Turbo codes. The SW-Log-BCJR algorithm is implemented based on the original memory architecture and clever memory management so that the LIFO memory is omitted. High operating frequency and negligible coding loss are satisfied simultaneously even for codes of $R = 2/3$ and $3/3$ by introducing new 4-input Log-Sum implementation for $\alpha$ and $\beta$ recursion. Thus, the operating frequency of 100[MHz] is achieved with moderate hardware cost. The decoders for most Turbo codes, including PCCC, SCC, TTCM and SCTCM, can easily be realized by cascading the LSI scalably. The coding loss for all these applications is less than 0.03[dB]. This LSI is implemented in a 304 pin QFP package using 0.25[$\mu$m] CMOS process.

REFERENCES