A 18-Mbp/s, 8-State, High-Speed Turbo Decoder

Ji-Won Jung · Min-Hyuk Kim · Jin-Hee Jeong

Abstract

In this paper, we propose and present implementation results of a high-speed turbo decoding algorithm. The latency caused by (de) interleaving and iterative decoding in a conventional maximum a posteriori(MAP) turbo decoder can be dramatically reduced with the proposed design. The source of the latency reduction is come from the combination of the radix-4, dual-path processing, parallel decoding, and early-stop algorithms. This reduced latency enables the use of the turbo decoder as a forward error correction scheme in real-time wireless communication services. The proposed scheme results in a slight degradation in bit-error rate(BER) performance for large block sizes because the effective interleaver size in a radix-4 implementation is reduced to half, relative to the conventional method. Fixed on the parameters of N=212, iteration=3, 8-states, 3 iterations, and QPSK modulation scheme, we designed the adaptive high-speed turbo decoder using the Xilinx chip (VIRTEX2P (XC2VP30-5FG676)) with the speed of 17.78 Mb/s. From the results, we confirmed that the decoding speed of the proposed decoder is faster than conventional algorithms by 8 times.

Key words : Turbo Code, Radix-4, Dual-Path Processing, Parallel Decoding, Early-Stop, FPGA.

Ⅰ. Introduction

For wireless communications, channel coding is an important tool for improving communications reliability. In satellite communications and 3rd generation mobile communication system, turbo codes have been adopted for high-speed data transmissions. Turbo codes, introduced in [1] and [2], have been shown to perform near the capacity limit in additive white Gaussian noise (AWGN) channels. Such a powerful coding technique offers great promise for improving the reliability of com- munication over wireless channels. Wireless communication systems are moving today from the conventional narrow-band voice service to the wide-band multimedia service requiring high-speed turbo decoders. Important issues in high-speed applications of turbo decoders are decoding delay and computational complexity. To solve the latency problem, in this paper four decoding algorithms are presented and combined into one decoder architecture. The first algorithm is the radix-4 decoding algorithm, where the previous state at $t=k-2$ goes forward to the current state at $t=k$, and the reverse state at $t=k+2$ goes backwards from the current one such that the time interval from $t=k-2$ to $t=k$ is merged into $t=k$. Therefore, we can decode two source data bits at the same time without any performance degradation while reducing the block size buffered in memory. In order to apply the radix-4 algorithm, we derived the branch metric(BM), forwardstate metric(FSM), α, backward state metric(BSM), β, and log likelihood ratio(LLR), λ.

The second algorithm is the dual-path processing. In a conventional scheme, the decoder must wait for finishing the BSM (or FSM) calculations before calculating the LLR. The dual-path processing doesn't need to wait. The decoder calculate the FSM (left to right) and BSM (right to left), simultaneously. When the FSM and BSM reach the same point, then the decoder begins to calculate the LLR.

The third algorithm is the full parallel decoding algorithm. Different from the original turbo decoder consisting of two decoders concatenated in a serial fashion, we propose a parallel decoder structure using the parallel sum, where the two decoders operate in parallel and update each other immediately and simultaneously after each one has completed its decoding. In decoding the estimated data, we use the sum of the LLR outputs of the parallel decoders to reduce the latency to half while maintaining the same performance level.

The fourth algorithm is the hard-decision-aided(HDA) algorithm implementing the early-stop algorithm. It compares each decision generated by the two decoders, and when the two sets of decisions match, it stops decoding the current block and outputs the hard decision bits.

These schemes are desirable in designing a high-speed turbo decoder because they provide significant reductions in decoder memory requirements as well as allowing increased parallelism. To verify the high-speed feature of the decoder, we implemented the proposed

Manuscript received March 29, 2006 ; revised September 15, 2006. (ID No. 20060329-008J)
Department of Radio Science & Engineering, Korea Maritime University, Busan, Korea.

combined schemes Xillinx FPGA chip (VIRTEX2P (XC2 VP30-5FG676)) and compared its decoding speed to a conventional design. The proposed architecture is demonstrated for a 8-state, R=1/2 turbo maximum a posteriori(MAP) decoder.

The paper consists of the following sections. In section II, we present and analyze the four algorithms. Section III describes computer simulation results of the new high-speed turbo decoder based on combining these algorithms. In section IV, we present the optimal parameters and design of the FPGA-based high-speed turbo decoder. Section V concludes the paper.

II. High-Speed Turbo Decoding Algorithm

Since convolutional turbo codes are very flexible codes, easily adaptable to a large rate of block sizes and coding rates, they have been adopted in the DVB standard for return channel via satellite(DVB-RCS). The use of RCST(RCS Terminal) includes individual and collective installation(e.g. SMATV) in domestic environment. However, the applications of turbo codes are limited to specific data such as low data-rate services because of their limit of decoding speed. Therefore, it is highly required to develop the high-speed turbo decoder. To solve the problem with latency of turbo decoder, four kinds of algorithms are introduced. The first algorithm is radix-4 algorithm and the second algorithm is the dual-path processing algorithm. The third algorithm is the full parallel decoding algorithm. The fourth algorithm is the early-stop algorithm based on hard-decision-aided(HDA) scheme. The decoding iteration processes until a certain stopping condition is satisfied then hard decisions are made based on the reliability measures of the decoded symbol at the last decoding iteration.

2-1 Scheme 1: Radix-4 Algorithm

The first algorithm is the radix-4 decoding algorithm, where the previous state at $t=k-2$ goes forward to the current state at $t=k$, and the reverse state at $t=k+2$ goes backwards from the current one such that the time interval from $t=k-2$ to $t=k$ is merged into $t=k$. Therefore, we can decode two source data bits at the same time without any performance degradation while reducing the block size buffered in memory. Using the unified approach to state metrics, a $2^{\nu-1}$-state trellis can be iterated from time index $n-k$ to n by decomposing the trellis into $2^{\nu-k}$ sub-trellises, each consisting of k iterations of a 2^k-state trellis. Each 2^k-state's sub-trellis can be collapsed into an equivalent one-stage radix-2^k trellis by applying k levels of look-ahead to the recursive update. Collapsing the trellis does not affect the decoder performance since there is a one-to-one mapping between the collapsed trellis and radix-2 trellis. An example of the decomposition of a 4-state radix-2 into an equivalent radix-4 trellis using one stage of look-ahead is shown in Fig. 1, where $\nu=3$, $g_1=(7)_{octal}$, $g_2=(5)_{octal}$ with ν denoting the constraint length.

For radix-4 trellis iterations time k to $k-2$ time backward state for calculating the α_k^m and from time $k+2$ to time k forward state for calculating the β_k^m can be expressed

$$m_{k-2}(d_{k-1}, d_k, m_k) = (d_{k-1} \oplus d_k \oplus m_{0,k}) \| (d_k \oplus m_{0,k} \oplus m_{1,k})$$
$$m_{k+2}(d_{k+1}, d_{k+2}, m_k) = (d_{k+1} \oplus d_{k+2} \oplus m_{1,k}) \| (d_{k+1} \oplus m_{0,k} \oplus m_{1,k}) \quad (1)$$

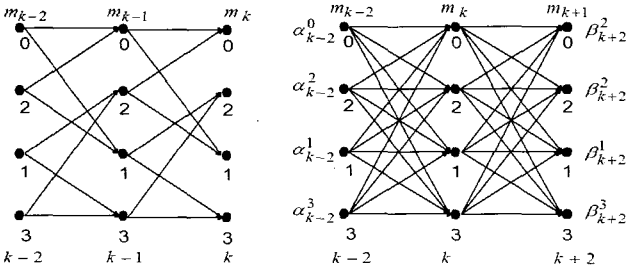
where d_k and m_k mean uncoded data bit state values at time of k, and $(a\|b)$ means concatenate the a and b. We have that $m_k=\{m_{0,k},m_{1,k},...,m_{\nu-1,k}\}$ corresponding to the encoder state. m_{k-2} is used for calculating the $\alpha_k^{m_k}$ and m_{k+2} is used for calculating the $\beta_k^{m_k}$ at k times.

A. Derivation of $\delta_k^m \alpha_k^m \beta_k^m$ and λ_k^m

We consider an AWGN channel with zero mean and variance σ^2. In radix-4 MAP decoder, branch metric $\delta_k^{p,m}$ for received signal, at time k, can be expressed;

$$\begin{aligned}
\delta_k^{p,m} &= P_r(D_k=p, S_k=m_k, R_k) \\
&= P_r(D_k=i_{k-1}\|i_k, S_k=m_k, R_k) \\
&= P_r(d_{k-1}=i_{k-1}, S_k=m_{k-1}, R_{k-1}) P_r(d_k=i_k, S_k=m_k, R_k) \\
&= P_r(x_{k-1} \mid d_{k-1}=i_{k-1}, S_{k-1}=m_{k-1})\, P_r(y_{k-1} \mid d_{k-1}=i_{k-1}, S_{k-1}=m_{k-1}) \varsigma_{k-1}^{i_{k-1}} / 2^{\nu} \\
&\quad \cdot P_r(x_k \mid d_k=i_k, S_k=m_k)\, P_r(y_k \mid d_k=i_k, S_k=m_k) \varsigma_k^{i_k} / 2^{\nu} \\
&= K_k \exp\left(\frac{2}{\sigma^2}(x_{k-1}d_{k-1} + y_{k-1}Y_{k-1} + x_k d_k + y_k Y_k)\right)
\end{aligned} \quad (2)$$

where i_k is the value of information bit at time k, and $p=i_{k-1}\|i_k$. Therefore we have that $i=\{0, 1\}$, and $p=\{00, 01, 10, 11\}$. K_k is a constant and x_k and y_k are received in-phase signal and quadrature signal. Y_k is coded bit as a function of the input bit d_k and encoder state m_k. Forward state metric at time k and m state α_k^m can be ex-



(a) 4-state radix-2 trellis (b) 4-state radix-4 trellis

Fig. 1. Four-state radix-2 to radix-4 trellis.

pressed;

$$\begin{aligned}\alpha_k^m &= P_r(R_1^{k-2} \mid D_k=p, S_k=m_k, R_k^N) = P_r(R_1^{k-2} \mid S_k=m_k)\\ &= \sum_{m'}\sum_{p=0}^{3} P_r(D_{k-2}=p, S_{k-2}=m', R_1^{k-2} \mid S_k=m_k)\\ &= \sum_{m'}\sum_{p=0}^{3} P_r(D_{k-1}=p, S_{k-2}=m', (R_1^{k-4}, R_{k-2}) \mid S_k=m_k)\\ &= \sum_{p=0}^{3} \alpha_{k-2}^{b(p,m_k)} \delta_{k-2}^{p,b(p,m_k)}\end{aligned} \tag{3}$$

where $R_k^n = \{R_k, R_k+1, \cdots, R_n\}$.

$b(p,m_k)$ is previous $k-2$ time stage given an input p and state m_k

$$b(p,\ m_k)=m_{k-2}(d_{k-1},\ d_k,\ m_k) \tag{4}$$

In similar way, after the whole block of data is received, we can recursively calculate the probability β_k^m from the probability β_{k+2}^m.

$$\begin{aligned}\beta_k^m &= P_r(R_k^N \mid S_k=m_k)\\ &= \sum_{m'}\sum_{p=0}^{3} P_r(D_k=p, S_{k+2}=m', R_k^N \mid S_k=m_k)\\ &= \sum_{m'}\sum_{p=0}^{3} P_r(D_k=p, S_{k+2}=m', R_{k+2}^N, R_k \mid S_k=m_k)\\ &= \sum_{p=0}^{3} \beta_{k+2}^{f(p,m_k)} \delta_k^{p,m_k}\end{aligned} \tag{5}$$

where $f(p,\ m_k)$ is next $k+2$ time stage given an input p and state m_k. With the Equation (1), $f(p,\ m_k)$ is expressed;

$$f(p,\ m_k)=m_k+2(d_{k+1},\ d_{k+2},\ m_k) \tag{6}$$

Taking into account the fact that events after time k are not influenced by that part of observation up to time k, that is R_1^k and R_{k+2}^N are independent. Finally, with Equations (2), (5) and (6), the LLR of radix-4 algorithm can be written as;

$$\begin{aligned}\lambda_k^{p,m} &= P_r(D_k=p, S_k=m \mid R_1^N)\\ &= P_r(D_k=p, S_k=m,\ R_1^N)/P_r(R_1^N)\\ &= P_r(D_k=p, S_k=m, R_1^{k-2}, R_k^N)/P_r(R_1^N)\\ &= P_r(R_1^{k-2} \mid D_k=p, S_k=m,\ R_k^N) P_r(R_{k+2}^N \mid D_k=p, S_k=m,\ R_k)\\ &\quad \cdot P_r(D_k=p, S_k=m,\ R_k)/P_r(R_1^N)\\ &= \alpha_k^m \beta_{k+2}^{f(p,m)} \delta_k^{p,m}/P_r(R_1^N)\end{aligned} \tag{7}$$

In order to decode the two bits $D_k=d_{k-1}||d_k$, equation (8) can be used.

$$D_k = \max\left\{\sum_m \lambda_k^{00}(m), \sum_m \lambda_k^{01}(m), \sum_m \lambda_k^{10}(m), \sum_m \lambda_k^{11}(m)\right\} \tag{8}$$

Where $\lambda(d_k^{00})$ denotes corresponds to LLR value for two input bits '00'.

2-2 Scheme 2: Dual-Path Processing

In a conventional scheme, the decoder must wait for finishing the backward state metric(BSM) (or forward state metric(FSM)) calculations before calculating the extrinsic information. The dual-path processing method doesn't need to wait. The decoder calculates the FSM (left to right) and BSM (right to left), simultaneously. When the FSM and BSM reach the same point, then the decoder begins to calculate the extrinsic information. Fig. 2 shows the operation of dual-path processing.

The procedure of the dual-path processing is as follows.

Step 1: Initialize the forward state metric and backward state metric.

$$\begin{aligned}\alpha_0^i(s_0^i(m)) &= 1 \text{ for } m=0\\ &= 0, \ else\\ \beta_N^i(s_b^i(m)) &= 1 \text{ for } m=0\\ &= 0, \ else\end{aligned} \tag{9}$$

$\alpha^i{}_k(m)$ and $\beta^i{}_k(m)$ are FSM and BSM at time of k, information bit of i, and state of m.

Step 2: After receiving the whole set of received symbols of N, FSMs (left to right) and BSMs (right to left) are calculated simultaneously.

$$\hat{\alpha}_k{}^i(m) = \exp\left(\frac{2}{\sigma^2}(x_k i + y_k Y_k(i,m)) \sum_{j=0}^{1} \hat{\alpha}^j{}_{k-1}(S_b^i(m))\right) \tag{10}$$

$$\hat{\beta}_k{}^i(m) = \sum_{j=0}^{1} \hat{\beta}^j{}_{k+1}(m) \exp\left(\frac{2}{\sigma^2}(x_{k+1} j + y_{k+1}(j, S_f^i(m)))\right) \tag{11}$$

Step 3: At the middle point, begin to calculate the log likelihood ratios(LLR).

$$L(\vec{d_k}) = \log \frac{\sum_m \alpha_k^1(m)\beta_k^1(m)}{\sum_m \alpha_k^0(m)\beta_k^0(m)} \ (k=(N/2), \cdots, (N-1)) \tag{12}$$

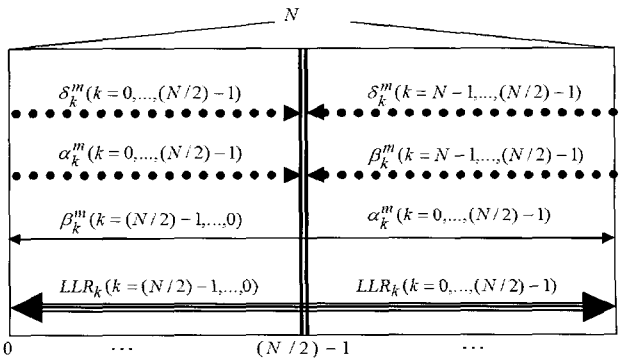


Fig. 2. Dual-path processing.

$$\overleftarrow{L(d_k)} = \log \frac{\sum_m \alpha_k^1(m)\beta_k^1(m)}{\sum_m \alpha_k^0(m)\beta_k^0(m)} (k=(N/2)-1,\cdots,0) \quad (13)$$

$\overrightarrow{L(d_k)}$ means LLR outputs in the direction of right to left and $\overleftarrow{L(d_k)}$ means LLR outputs in the direction of left to right.

2-3 Scheme 3: Parallel Decoding Algorithm[4]

Different from the original turbo decoder consisting of two decoders concatenated in a serial fashion, we present a parallel decoder structure using the parallel sum, where the two decoders operate in parallel and update each other immediately and simultaneously after each one has completed its decoding. In decoding the estimated data, we use the sum of the LLR outputs of the parallel decoders to reduce the latency to half while maintaining the same performance level.

2-4 Scheme 4: Early Stop Algorithm

The decoding iteration processes until a certain stopping condition is satisfied, hard decisions are made based on the reliability measures of the decoded symbol at the last decoding iteration. HAD algorithm is used as an early-stop algorithm. It compares each decision generated by the two decoders, and when the two sets of decisions match, it stops decoding on the current block and outputs the hard decision bits. As shown in Table 1, at an E_b/N_0 of 2 dB, it requires about 2.8 iterations relative to 8 for a given performance. This means that the decoding speed is improved or the power consumption (cost) is reduced by 64.6 %.

Ⅲ. Computer Simulation Results

The bit-error rate(BER) performance of the new high-speed turbo decoder architecture combining the four schemes is analyzed in this section.

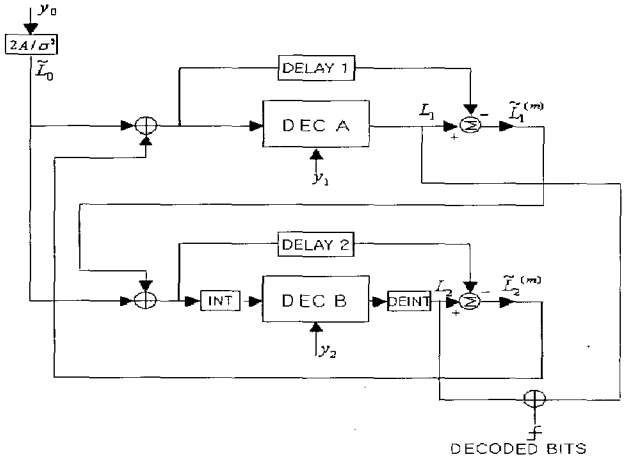


Fig. 3. Parallel decoder structure.

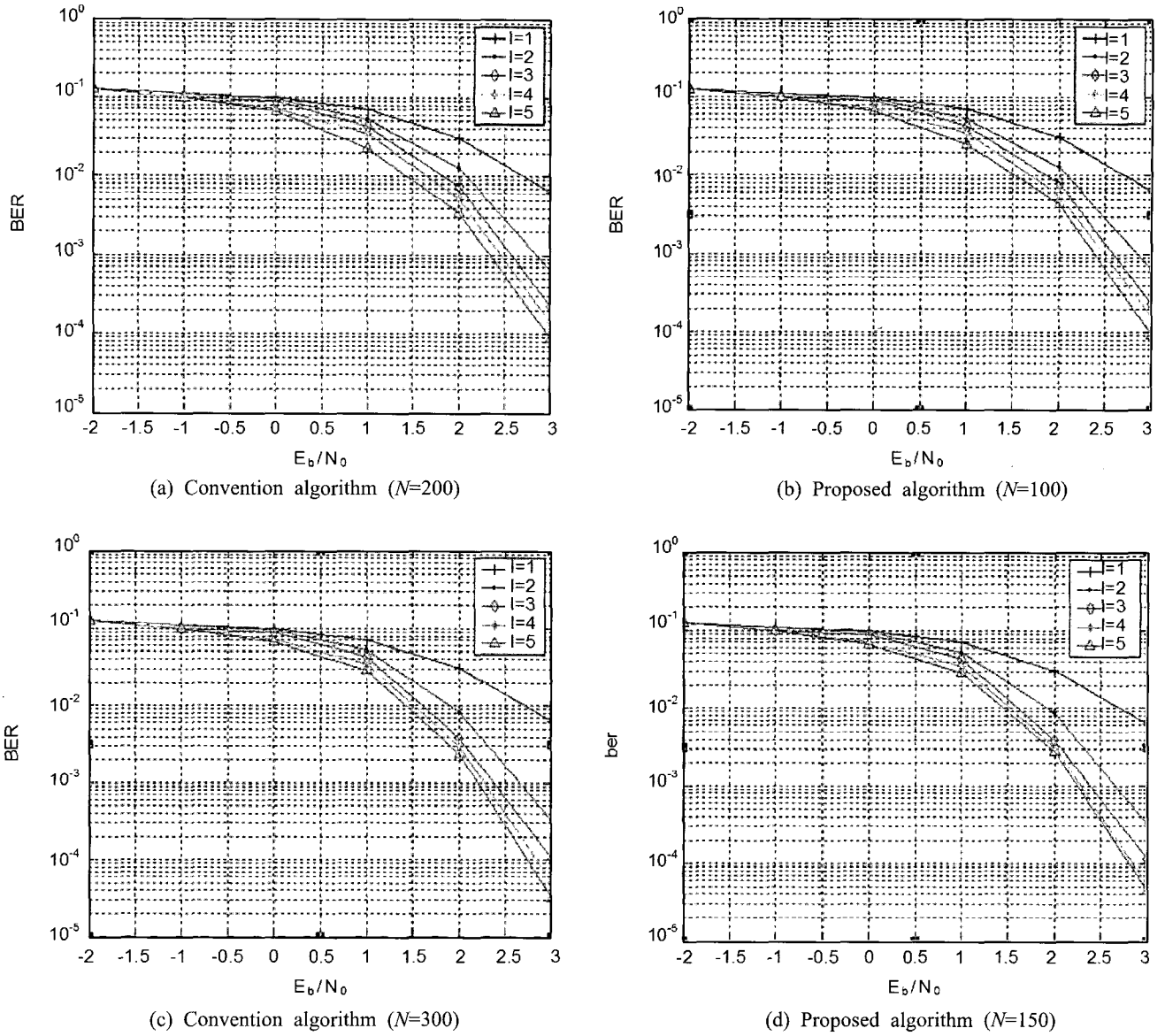
Table 1. The average number of iterations according to E_b/N_0 (the predetermined number of iterations is 8).

E_b/N_0[dB]	Average number of iterations	Decoding speed improvement(%)
1	6.21	22.4
1.5	4.31	46.1
2	2.83	64.6

For a comparison purpose, Fig. 4 shows the performance of the new decoder and a conventional one using K=3 turbo codes with generator polynomials $g_1=(7)_{octal}$, $g_2=(5)_{octal}$ as a function of interleaver size N and as a function of the number of iterations I. In the radix-4 method, the symbol interleaver, takes an information stream of length Ns composed of 2-bit words and feeds it to a random interleaver. From the figure, it can be verified that the performances of the proposed decoder architecture is very close to the conventional decoder for small block sizes (less than 300 bits). For large interleaver sizes (more than 300 bits), the performance of the new decoder is slightly degraded relative to the conventional one because the randomness of the symbol interleaver is reduced. Therefore, it can be concluded that the proposed decoder reduces the decoding latency while maintaining almost the same performance of a conventional algorithm.

Ⅳ. Design of High-Speed Turbo Decoder

A schematic diagram of the high-speed turbo decoder for implementation is shown in Fig. 5(a). A detailed signal flow of internal MAP based on high-speed algorithms is shown in Fig. 5(b). In Fig. 5(a), MAP 1 and MAP 2 decoders are operated in parallel, with their outputs being log-likelihood ratios of input bits as shown in Fig. 5(b). For instance, $\overrightarrow{LLR}0_{N/2\sim N}$ denotes log-likelihood ratios of input bits "00", $N/2$~N denotes from the time of $N/2$ to N, and the arrow $\rightarrow$ denotes the direction of the LLR calculation, that is, left-to-right. The ALU(arithmetic logic unit) calculates the extrinsic information using LLR outputs, the received symbol and the previous extrinsic information. To add the extrinsic information exactly in the next iteration, the decoder needs the dual port RAM(128×36) buffered ALU block. As shown in Fig. 5(b), the decoder consists of six major units, the Radix-4 forward branch metric unit (R4FBMu), Radix-4 backward branch metric unit (R4BBMu),



(a) Convention algorithm (N=200)

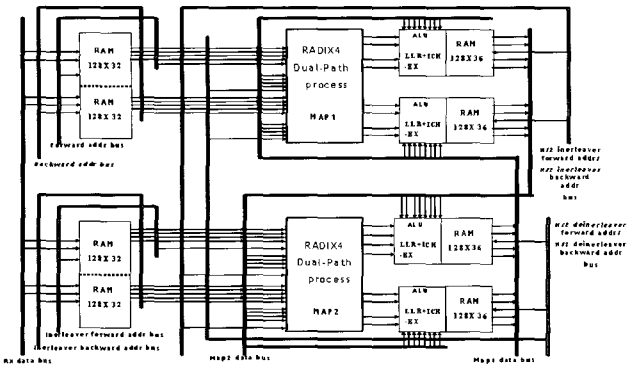
(b) Proposed algorithm (N=100)

(c) Convention algorithm (N=300)

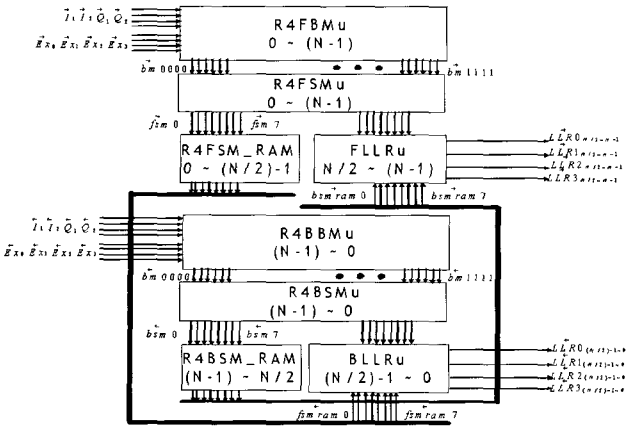
(d) Proposed algorithm (N=150)

Fig. 4. Performance of the proposed decoder over an AWGN channel compared with that of a conventional algorithm as a function of interleaver size and number of iterations.

Radix-4 forward state metric unit (R4FSMu), Radix-4 backward state metric unit(R4BSMu), forward LLR unit(FLLRu), backward LLR unit (BLLRu). After receiving the whole set of received symbols, the quantized I and Q samples are fed to the R4FBMu and R4BBMu at the same time. The branch metrics between the branch codeword "0000" and the received symbols is denoted by "bm0000". The R4FBMu and R4BBMu calculate the branch metrics for four samples of received data in the direction of left to right and right to left simultaneously. As shown in Fig. 5(b), $I_n(n=1,2)$ and $Q_n(n=1,2)$ are fed to R4FBMu to calculate forward branch metric and $I_n(n=1,2)$ and $Q_n(n=1,2)$ are also fed to R4BBMu to calculate backward branch metric. From the second iteration, R4FBMu and R4BBMu need extrinsic information, Ex_n, that is LLR of input bit "i_1, i_2". The index n of Ex_n means $2\times i_2+i_1$. R4FSMu calculates forward state metrics in the direction left-to-right and R4BSMu calculates backward state metrics in the direction right-to-left. The data calculated from R4-FSMu and R4BSMu are buffered in dual-port RAM with size of 64×72. Therefore we need the two dual-port RAM that is R4FSM_RAM and R4BSM_RAM. When the R4FSMu and R4BSMu reach the same point, then FLLRu and BLLRu begin to calculate the LLR of information n using the data read from R4FSM_RAM and R4BSM_RAM respectively. ALU block shown in Fig. 5(a) begins to calculate the extrinsic information



(a) High speed turbo decoder architecture



(b) Signal flow of internal MAP

Fig. 5. The block diagram of proposed high-speed turbo decoder.

and store extrinsic information in dual-port RAM with size of 128×36 in order to input to next iteration.

In order to implement optimally the high-speed decoder, we determined the optimum quantized bits of each block shown in Fig. 5, r_q bits of received in-phase and quadrature signals, b_q bits of R4FBMu and R4BBMu outputs, s_q bits of R4FSMu and R4BSMu outputs, and l_q bits of FLLR and BLLR. By fixed-point computer simulation, the output of the demodulator is quantized to 8 bits. The internal parameters of the turbo decoder were always saturated to 9 bits, and the optimum quantization bits of the turbo decoder derived from the fixed-point simulations are listed in Table 2.

We design the adaptive turbo decoder using VHDL (Very High-speed hardware Description Language) and verified its operation by RTL simulation. During the verification, some errors were added in C-description. Then we confirmed that the errors were corrected while being processed in RTL simulation. VHDL and C language process communicate with each other through program language interface. The decoder designed by VHDL was synthesized using the Xillinx FPGA chip

Table 2. The optimum quantized bits of the adaptive turbo decoder(R=1/2, 8-states, N=212 bits, 3 iterations, QPSK).

	Numver of optimized quantization bits
r_q	8 bits
b_q	9 bits
s_q	9 bits
l_q	9 bits

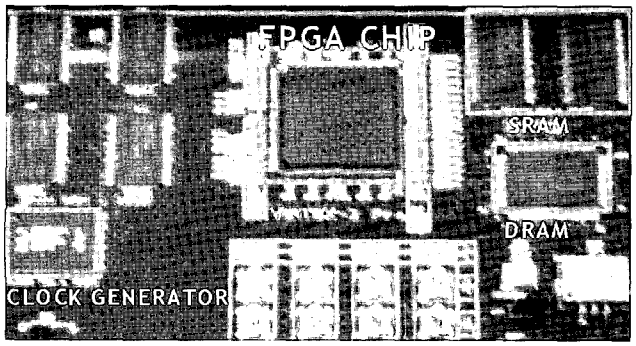


Fig. 6. The implement chip-set of the adapted high-speed turbo decoder.

(VIRTEX2P(XC2VP30-5FG676)) as shown in Fig. 6, The received data file generated by C language process at E_b/N_0 of 2 [dB] is fed into internal memory of Xillinx FPGA chip.

We designed the high-speed adaptive turbo decoder with 8-state, N=212, R=1/2, and QPSK modulation scheme. In order to compare the decoding speed of the conventional serial turbo decoder based on radix-2 trellis structure. To compare the decoding speed between conventional and high-speed adaptive turbo decoder, we implemented the conventional decoder using the same procedures. Based on Table 1, since the required iteration number is 3 in the case of E_b/N_0=2 [dB], we fixed the iteration number to 3. The maximum operating clock speed of the conventional and proposed decoders is 18 [ns] Table 3 shows comparison of decoding speed between conventional and high-speed decoder. In the case of combining the radix-4 and parallel and dual-path process, the proposed high-speed decoder is faster than conventional one by 8 times.

V. Conclusion

To extend the application area of turbo decoding to real time services, it is important to reduce the latency in the decoding process of turbo decoders. In this paper, we proposed a new high-speed turbo decoding algorithm

Table 3. Comparison of decoding speed between a conventional method and the proposed method (N =212, iteration=3, 8-state, 8PSK, main clock speed =18 *ns*).

	Radix -2+ Serial	Radix -4+ Serial	Radix -4+ Parallel	Radix 4+Parallel+ Dual-path Processing
Main clock	18 *ns*	18 *ns*	18 *ns*	18 *ns*
Execution time of LLR output	95163 *ns*	47418 *ns*	23643 *ns*	11919 *ns*
Decoding speed	2.27 M	4.47 M	8.96 M	17.78 M

and architecture that provides a solution. Two new algorithms are presented, radix-4 and dual-path processing that are combined to two other techniques to reduced latency. The different decoding algorithms are incorporated into a decoder design architecture to show how much the decoding latency can be reduced. To extend the application area of turbo decoding to real time services, it is important to reduce the latency in the decoding process of turbo decoders. In this paper, we proposed a new high-speed turbo decoding algorithm and implementation architecture. Two new low latency version of decoder are presented, radix-4 algorithm and dual-path processing that are combined to parallel mode and early-stop algorithm. Fixed on the parameters of N=212, iteration=3, 8-states, 3 iterations, and QPSK modulation scheme, we designed the adaptive high-speed turbo decoder using the Xilinx chip (VIRTEX2P (XC2 VP30-5FG676)) with the speed of 17.78 Mb/s. From the results, we confirmed that the decoding speed of the proposed decoder is faster than conventional algorithms by 8 times.

References

[1] C. Berrou, A. Glavieus, and P. Thitimajshima, "Near shanon limit error-correcting coding and decoding: Turbo-codes", *Proc. ICC93*, pp. 1064-1070, May 1993.

[2] P. Robertson, E. Villebrun, and P. Hoeher, "A comparison of optimal and sub-optimal MAP decoding algorithms operating in the log domain", *ICC*, pp. 1009-1013, 1995.

[3] D. Divsalar, F. Pollara, "Serial and hybrid concatenated codes with applications", *Proc. of the Int'l Symp. on Turbo Codes & Related Topics*, pp. 80-87, Sep. 1997.

[4] S. Benedetto *et al.*, "Soft output decoding algorithm in iterative decoding of codes", *TDA Progress Rep. 42-124*, Jet Propulsion Lab., Pasadena, CA, pp. 63-87, Feb. 1996.

[5] P. Hoeher, "New iterative (turbo) decoding algorithms", *Proc. of the Int'l Symp. on Turbo Codes & Related Topics*, pp. 63-70, Sep. 1997.

[6] S. S. Pietrobon, "Implementation and performance of a serial MAP decoder for use in an iterative turbo decoder", *Proc. IEEE Int. Symp. on Information Theory*, pp. 471-480, 1995.

[7] S. S. Pietrobon, "Implementation and performance of a turbo/MAP decoder", *Int'l J. of Satellite Comm.*, vol. 16, pp. 23-46, 1998.

[8] Bernard Sklar, "A primer on turbo code concepts", *IEEE Comm. Magazine*, Dec. 1997.

[9] D. Divsalar, F. Pollara, "Multiple turbo codes for deep-spacemunications", *TDA Progress Rep. 42-141*, Jet Propulsion Lab., Paradena, CA, pp. 66-77, 1995.

[10] S. Benedetto, G. Montorsi, "Unveiling turbo dodes: Some results on parallel concatenated coding schmes", *IEEE Trans. on Information*, vol. 42, no. 2, pp. 409-429, Mar. 1996.

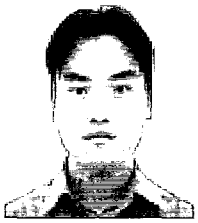
[11] Digital Video Broadcasting(DVB) Interaction Channel for Satellite Distribution System, *ETSI Reference EN 301 799*, v1. 2.2, Dec. 2000.

[12] C. Douillard *et al.*, The Turbo Code Standard for DVB-RCS, *Proc. of the 2nd Int Symp. on Turbo Codes*, Brest, Fran.

Ji-Won Jung

received his B.S. in 1989, M.S. in 1991, and Ph.D. in 1995 from Sungkyunkwan University, Seoul, Korea, all in electronics engineering. From Nov. 1990 to Feb. 1992, he was with LG research center, Korea. From Sep. 1995 to Aug. 1996, he was with Korea Telecom(KT). From Aug. 2001 to July 2002, he was an invited researcher at the communication research center Canada supported by NSERC. Since 1996, he joined the department of radio science and engineering, Korea Maritime University, Busan, Korea. His research interests are channel coding, digital modem, FPGA design technology, and digital broadcasting system.

Min-Hyuk Kim



received his B.S. in 2006, is studying as M.S. in Korea Maritime University. His research interests are channel coding, digital modem, FPGA design technology, and digital broadcasting system.

Jin-Hee Jeong

received her B.S. in 2005, is studying as M.S. in Korea Maritime University. Her research interests are channel coding, digital modem, FPGA design technology, and digital broadcasting system.