Scientific
Research

Reliable Wireless Communication for Medical Devices Using Turbo Convolution Code

Debasish Bera[1], Tapas Chakravarty[1], Saswat Chakrabarti[2]
[1]*Tata Consultancy Services Ltd., Kolkata, India*
[2]*Indian Institute of Technology Kharagpur, Kharagpur, India*
E-mail:{*debasish.bera, tapas.chakravarty*}*@tcs.com, saswat@ece.iitkgp.ernet.in*

Abstract

Wireless technology is now being used to bring significantly innovative products and services in the healthcare sector, enabling new medical sensors, and treatment methods. In this paper, a new and improved communication system for communicating signals among different medical devices and a console is presented. Considering the need for very high degree of functional reliability of communication link in RF challenged environment (indoor), a novel algorithm called bi-directional "Soft Output Viterbi Algorithm (SOVA)" decoding for double-binary Circular Recursive Systematic Convolutional (CRSC) turbo codes is presented. The bi-directional SOVA is considered in view of its better performance and implementation complexity trade-off. The basic SOVA has been described for binary turbo code. We have extended it for double binary case, which is useful for high data rate healthcare applications using real time streaming data. Necessary changes in basic message passing equations for double-binary case have been introduced. Coding gain can be used to increase the robustness and immunity to interference. Decoding of CRSC codes requires a prologue decoder, prior to the actual trellis decoding, to estimate the initial state. Efficient determination of circulation state through prologue decoding has helped in achieving impressive error performance for CRSC codes. The issues related to digital implementation of turbo encoder/decoder and their effects on error performance have also been discussed. Adequate simulation results have been included.

Keywords: CRSC, FPGA, SOVA, Turbo Code, Reliable Communication

1. Introduction

The medical-healthcare industry is constantly evolving, so medical device manufacturers must keep pace with industry trends. Healthcare costs have been also of extreme interest over the last several years. Healthcare providers have been addressing cost issues to attempt to become more competitive. A common requirement is to add wireless connectivity to previously isolated serial devices. To reduce costs and improve care, providers *must* reengineer their acute care settings using information technology. The availability of mass market wireless technology that supports these devices is necessary. That technology is maturing in the form of Bluetooth, ZigBee, Wi-Fi, and Ultra Wide Band (UWB). Data reliability is an important issue in all these standards, which can be taken care by efficient error correction method.

Unlike transporting general data over wireless networks, however, medical data, video applications can't tolerate bandwidth fluctuations, so the challenge is significantly more rigorous.

A number of critical factors contribute to satisfactory performance. These include bandwidth, latency, and breadth of coverage, reliability and quality of service (QoS). Reliable operating range is difficult to predict due to the lack of knowledge of the special propagation characteristics in indoor scenario. Reflected signals by floors, ceilings, walls, various furniture and people are present near transmitters and receivers. That is the signals are travelling over multi-paths. In most indoor cases, there is no direct line-of-site path, and all signals are the result of reflection, diffraction and scattering. Higher throughput improves immunity to interferences and excess bandwidth can be traded for longer reach and better power efficiency. In this article we present turbo convolution coding technology for reliable communication. Its inbuilt interleaving technology exploits time diversity, which helps in combating multi-path effect. It is also useful in high throughput scenario.

Turbo coding is a powerful forward error correction

(FEC) method that has gained considerable attention in recent past [1]. The fundamental turbo encoder is built with two identical Recursive Systematic Convolutional (RSC) codes, which are parallelly concatenated [1] through a random interleaver. A RSC encoder is typically of rate-1/2 and termed as component encoder. The interleaver is a device that permutes the data sequence in some predetermined manner. Only one of the systematic outputs from the two component encoders is used. Double-binary Circular Recursive Systematic Convolutional (CRSC) elementary codes for iterative decoding have been introduced by Berrou *et al.* [2]. It provides significant error-correcting performance while not suffering from the well-known Bit Error Rate (BER) floor of classical binary turbo codes. Double-binary (in general *m*-binary) codes have certain advantages [3] over binary codes, like: better convergence of the iterative process, larger minimum distance, less sensitivity to puncturing patterns, higher throughput and reduced latency, robustness of the decoder etc. Instead of recursive systematic convolution code, a parallel concatenation of CRSC codes [4] makes convolution turbo codes efficient for coding of data in blocks.

The double-binary CRSC codes specified for various block sizes and wide range of code rates have been adopted in the Digital Video Broadcasting – Return Channel Satellite (DVB-RCS) standard [5]. In this paper, we present the performance of DVB-RCS compliant rate-1/3, constraint length 4 (*i.e.*, memory $v = 3$) CRSC code with bi-directional Soft Output Viterbi Algorithm (SOVA) [6] as the component decoding algorithm. It have been considered, A, B are the systematic bits and W, Y are the parity or redundant bits generated from the encoder.

Both Maximum A Posteriori probability (MAP) and SOVA are *Maximum Likelihood* (ML) algorithms. MAP algorithm finds the most probable information symbol that was transmitted, while the SOVA finds the most probable information sequence that was transmitted for a given code sequence. That means MAP minimizes the bit or symbol error probability, where as SOVA minimizes the word error probability. Symbol-by-symbol MAP algorithm [7] for estimating the states or outputs of a Markov process observed in white noise is optimal. However MAP algorithm is not practically implementable due to its numerical complexities. Approximate versions of the MAP algorithm, such as Max-Log-MAP [8], [9], Log-MAP [10], have been derived. Those are less complex and can be used instead of MAP. Log-MAP algorithm avoids the approximations in the Max-Log-MAP algorithm using a simple correction function at each max operation and its performance is near to MAP. SOVA and Max-Log-MAP are sub-optimal approaches at low signal-to-noise ratio (SNR).

The most attractive feature of the SOVA is its simplicity, as far as hardware implementation is concerned, with a little degradation in performance [6]. Battail [11] and Hagenauer *et al* [12] first proposed the SOVA algorithm. Berrou *et al* [13] also proposed an algorithm, which is fundamentally based on [11,12]. Berrou *et al* [13] considered all 2^v (where v = memory order of the constituent encoder) parallel trace back operations, starting from each node at the end of the trellis. The values, which have been memorized during the forward traversal on the 2^v survivors, are updated by taking into account the present values. The size is roughly twice the size of classical Viterbi decoder. Vucetic *et al*. [6] proposed a *Bi-directional SOVA* (BSOVA) by considering the backward path metric calculation. It is computationally more complex than Hagenauer's method but gives better BER performance. That is why we have selected SOVA as our decoding algorithm.

The architectural designs of encoder and decoder for digital implementation on Field Programmable Gate Array (FPGA) have been discussed. This paper is organized as follows; Different SOVA algorithms have been studied and bi-directional SOVA decoding mechanism with necessary modifications in message passing equations for using them in double-binary codes have been described in Section 2. Section 3 explains the architecture of the codec to implement on FPGA. Section 4 presents the simulation results and its analysis. Section 5 concludes the paper.

2. SOVA for Double-Binary Code

In binary case, only two types of transmitted symbols are possible, *i.e.*, $d = 0$ or 1. But in double binary case four types of transmitted symbols are possible, *i.e.*, $d = 00, 01, 10$ or 11. The corresponding likelihood ratios are as follows:

$$\frac{P(d=00|x)}{P(d=00|x)} = \frac{p(x|d=00) \cdot P(d=00)}{p(x|d=00) \cdot P(d=00)}$$

$$\text{for } d = 00 \qquad (1)$$

$$\frac{P(d=01|x)}{P(d=00|x)} = \frac{p(x|d=01) \cdot P(d=01)}{p(x|d=00) \cdot P(d=00)}$$

$$\text{for } d = 01 \qquad (2)$$

$$\frac{P(d=10|x)}{P(d=00|x)} = \frac{p(x|d=10) \cdot P(d=10)}{p(x|d=00) \cdot P(d=00)}$$

$$\text{for } d = 10 \qquad (3)$$

$$\frac{P(d=11|x)}{P(d=00|x)} = \frac{p(x|d=11) \cdot P(d=11)}{p(x|d=00) \cdot P(d=00)}$$

$$\text{for } d = 11 \qquad (4)$$

where, d = transmitted symbol and x = received continuous valued noisy symbol.

The basic message passing equation of an iterative (turbo) decoder in terms of log-likelihood ratio (LLR) is as follows,

$$\Rightarrow L(d|x) = L_c(x|d) + L(d) \tag{5}$$

where, $L(d|x)$ = soft value in terms of LLR, out of the demodulator *i.e.*, demodulator a posteriori LLR value. $L_c(x|d)$ = LLR of the channel measurements of x under the alternate condition, that $d = +1$ or $d = -1$, *i.e.*, this is the result of a channel measurement at the demodulator. $L(d)$ = A priori LLR of the data symbol d.

After introducing decoder, for a systematic code:

$$L_{soft}(d|x) = L(d|x) + L_e(d) \tag{6}$$

where, $L_{soft}(d|x)$ = Soft output of a data symbol out of the decoder.

$L(d|x)$ = LLR of the data symbol out of the demodulator *i.e.* input to the decoder.

$L_e(d)$ = Extrinsic LLR, represents extra knowledge that comes from the decoder.

Therefore it can be written as,

$$L_{soft}(d|x) = L_e(x|d) + L(d) + L_e(d) \tag{7}$$

where, $L_e(x|d)$ = Systematic Information, $L(d)$ = A priori information, and $L_e(d)$ = Extrinsic information.

Now, $L_{soft}(d|x)$ is a real number whose sign denotes the hard decision and magnitude denotes the reliability of that decision. *i.e.*, if $L_{soft}(d|x)$ is more + ve then it is more reliable to be + 1 and if more – ve then it is more reliable to be – 1.

$$\therefore L_i(d) = \ln\left[\frac{P(d=i)}{P(d=0)}\right], i = 1, 2, 3 \tag{8}$$

In iterative case to compute symbol probabilities for the next decoder from previous decoder,

$$L_e^i(d) = L_i(d) = \ln\left[\frac{P(d=i)}{P(d=0)}\right] \tag{9}$$

Now for double binary code, it can be realized as follows,

$$L_e^{00}(d) = \ln\left[\frac{P(d=00)}{P(d=00)}\right] \tag{10}$$

$$L_e^{01}(d) = \ln\left[\frac{P(d=01)}{P(d=00)}\right]$$

$$\Rightarrow P(d=01) = e^{L_e^{01}(d)}P(d=00) \tag{11}$$

$$L_e^{10}(d) = \ln\left[\frac{P(d=10)}{P(d=00)}\right]$$

$$\Rightarrow P(d=10) = e^{L_e^{10}(d)}P(d=00) \tag{12}$$

$$L_e^{11}(d) = \ln\left[\frac{P(d=11)}{P(d=00)}\right]$$

$$\Rightarrow P(d=11) = e^{L_e^{11}(d)}P(d=00) \tag{13}$$

In general, it can be extended for *m*-binary code.

2.1. For Binary Turbo Code

SOVA estimates the soft output information for each transmitted binary symbol (d_t) in the form of LLR and which can be simplified as follows [6]:

$$\Lambda(d_t) = \ln\left[\frac{P(d_t=+1|x)}{P(d_t=-1|x)}\right],$$

x = received noisy symbol,

$$= (-1)^\alpha (M_{T,\min} - M_{t,c}) \tag{14}$$

$M_{T,min}$ = Minimum path metric of ML-path selected in the same as in Viterbi algorithm (VA).

$M_{t,c}$ = min$\{$ $M_{t-1,f}$ (k') + B$_t$(k',k) + $M_{t,b}$(k)$\}$ = Best competitor of the ML symbol at time instant t.

$M_{t-1,f}$ (k') = Path metric of the forward survivor path at time $t - 1$ and node k'; B$_t$(k', k) = Branch metric at time instant t for a complement symbol c from node k' to k; $M_{t,b}$(k) = Backward survivor path metric at time t and node k.

For binary case the Equation (14) can further be simplified to:

$$\Lambda(d_t) = (M_{t,0} - M_{t,1}) \tag{15}$$

It is basically path difference between ML paths when ML symbol is "0" and "1" respectively. Equation for extrinsic information [6] in iterative process should be as follows:

For decoder-1 in *r*-th iteration:

$$\Lambda_{1e}^{(r)}(d_t) = \Lambda_1^{(r)}(d_t) - 4d_{t,0} - \tilde{\Lambda}_{2e}^{(r-1)}(d_t) \tag{16}$$

Similarly for decoder-2 in *r*-th iteration:

$$\Lambda_{2e}^{(r)}(d_t) = \Lambda_2^{(r)}(d_t) - 4\tilde{d}_{t,0} - \tilde{\Lambda}_{1e}^{(r)}(d_t) \tag{17}$$

2.2. For Double-Binary Turbo Code

In double-binary (or duo-binary) code, soft output information for each transmitted quaternary symbol (d_t) in the form of LLR is:

$$\Lambda(d_t) = \ln\left[\frac{P(d_t=i|x)}{P(d_t=0|x)}\right], i = 0,1,2,3$$

$$= M_{t,c} - M_{T,\min} \tag{18}$$

$M_{T,min}$ = Minimum path metric of ML-path selected in the same way as in Viterbi Algorithm.

$M_{t,c}$ = min $\{M_{t-1,f}(k') + B_t(k', k) + M_{t,b}(k)\}$ = Best competitor of the ML symbol at time instant t.

$M_{t-1,f}(k')$ = Path metric of the forward survivor path at time $t-1$ and node k'; $B_t(k', k)$ = Branch metric at time instant t for a complement symbol c from node k' to k; $M_{t,b}(k)$ = Backward survivor path metric at node k and time t.

In iterative process of double-binary code, there will be three extrinsic information for each input symbol (01, 10, and 11 w.r.t. 00) to be passed from one decoder to other:

For 1st decoder, in r-th iteration,

$$\Lambda_{1e}^{01(r)}(d_t) = \Lambda_1^{01(r)}(d_t) - 2d_{t,b} - \tilde{\Lambda}_{2e}^{01(r-1)}(d_t) \quad (19)$$

$$\Lambda_{1e}^{10(r)}(d_t) = \Lambda_1^{10(r)}(d_t) - 2d_{t,a} - \tilde{\Lambda}_{2e}^{10(r-1)}(d_t) \quad (20)$$

$$\Lambda_{1e}^{11(r)}(d_t) = \Lambda_1^{11(r)}(d_t) - 2(d_{t,b} + d_{t,a}) - \tilde{\Lambda}_{2e}^{11(r-1)}(d_t) \quad (21)$$

Similarly, for 2nd decoder, in r-th iteration,

$$\Lambda_{2e}^{01(r)}(d_t) = \Lambda_2^{01(r)}(d_t) - 2d_{t,b} - \tilde{\Lambda}_{1e}^{01(r)}(d_t) \quad (22)$$

$$\Lambda_{2e}^{10(r)}(d_t) = \Lambda_2^{10(r)}(d_t) - 2d_{t,a} - \tilde{\Lambda}_{1e}^{10(r)}(d_t) \quad (23)$$

$$\Lambda_{2e}^{11(r)}(d_t) = \Lambda_2^{11(r)}(d_t) - 2(d_{t,b} + d_{t,a}) - \tilde{\Lambda}_{1e}^{11(r)}(d_t) \quad (24)$$

where, I = 01, 10, 11 and j = 1, 2. $d_{t,a}$ and $d_{t,b}$ construct a symbol i.

$\Lambda_{je}^{i(r)}(d_t)$ = *Extrinsic Information for i-th data symbol in r-th iteration from j-th decoder at time t.*

$\Lambda_{je}^{i(r)}(d_t)$ = *Soft output for i-th data symbol in r-th iteration from j-th decoder at time t.*

$\tilde{\Lambda}_{je}^{i(r)}(d_t)$ = *Interleaved extrinsic Info for i-th data symbol in r-th iteration from j-th decoder at time t.*

2.3. Relevance of Prologue Decoding

In circular coding of convolution codes, an encoder goes back to its initial state at the end of the encoding operation. The decoding trellis can therefore be seen as a circle and decoding may be started anywhere on this circle. This avoids forceful termination of a trellis to all-zero state, which is well known as tailbiting [14]. Gianchristofaro *et al*. [15] have suggested a search for initial state using the last part of the frame and exploiting the trellis closure property. For circular encoding of the double-binary code [5] and associated prologue decoding have been explained in [16,17].

3. Architecture of Turbo Code

This section deals with digital design and FPGA implementation of turbo decoder. System and design specifications are given in [5]. It is targeted on Xilinx Virtex-II FPGA [18]. Interleavers are implemented by storing the pre-calculated interleaved addresses in ROM [19]. As discussed in [5] the interleaver follows some algebraic rule.

3.1. Design of Decoder

The incoming in-phase and quadrature-phase symbols (quantized in 5-bit sign-magnitude) from the demodulator are sampled @ 3 Msps and then demultiplexed into six branches: systematic A, B, parity from first encoder $Y1, W1$ and parity from second encoder $Y2, W2$. The functional block diagram of turbo decoder data-path is shown in **Figure 1**.
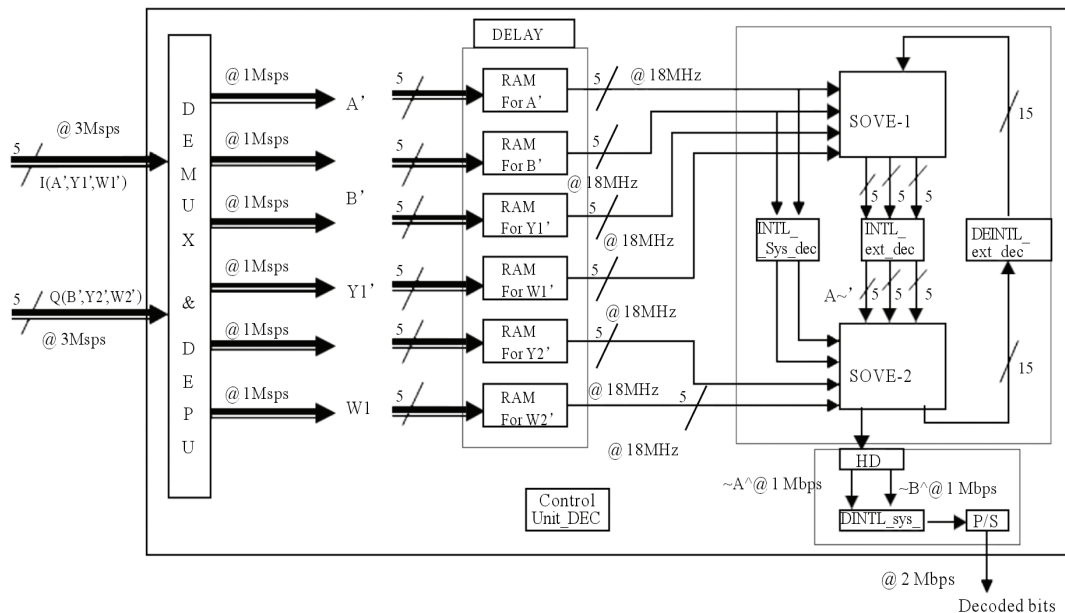

Figure 1. Decoder data path.

After de-multiplexing and de-puncturing, each of $A', B', Y1', W1', Y2', W2'$ are written into RAM (part of the delay unit) @ 1Msps. Iterative nature of the decoder is implemented by reading data from these RAMs at higher rate (e.g. @18 MHz for 4 iteration). This data reading rate would depend on the SOVA architecture and number of iterations. The samples are then passed through iterative SOVA architecture. Decisions about the decoded samples are taken depending on their sign bit. This decoded sequence first deinterleaved, then passed through P/S converter and get the actual decoded sequence.

3.2. Design of Timing Unit

The turbo encoder and decoder require different clocks viz., Outclk_encinternal (1 MHz), Outclk_indata (2MHz), Outclk_SOVA (4.5 MHz, 9 MHz, 18 MHz) and Outclk_ACS (54 MHz, 108 MHz, 216 MHz) for the iterative processing (up to four iterations) of a frame. A timing unit derives the clocks from an input master clock of 55.296 MHz. The input master clock is divided (CLKDV) or multiplied (CLKFX, CLK2X) with different set of values to generate the different clocks. There

are three main synchronization clock signals in the architecture. One clock signal controls the time sequence of the decoder that is equal to the data symbol time. Each symbol time interval is divided into (2 × number of iterations + number of internal latches of SOVA) iteration sequence. Each of the iteration intervals is divided into (number of states + Add-Compare-Select (ACS) internal delay) to perform the reading and writing metrics memory within ACS block. Thus, the highest clock signal (for ACS) required by the architecture for four iterations is 216 MHz. Architecture of timing unit has been realized using four inbuilt Digital Clock Manager (DCM) blocks of FPGA [18].

3.3. Component SOVA Decoder

The functional block diagram of SOVA data-path is shown in **Figure 2**. It is implemented with micro-level pipelining. Each component SOVA decoder consists of several functional blocks. Branch Metric Calculation Unit (BMCU) takes four type of input and generates all sixteen possible (from ABYW to ~ A~ B ~ Y ~ W) branch metrics.
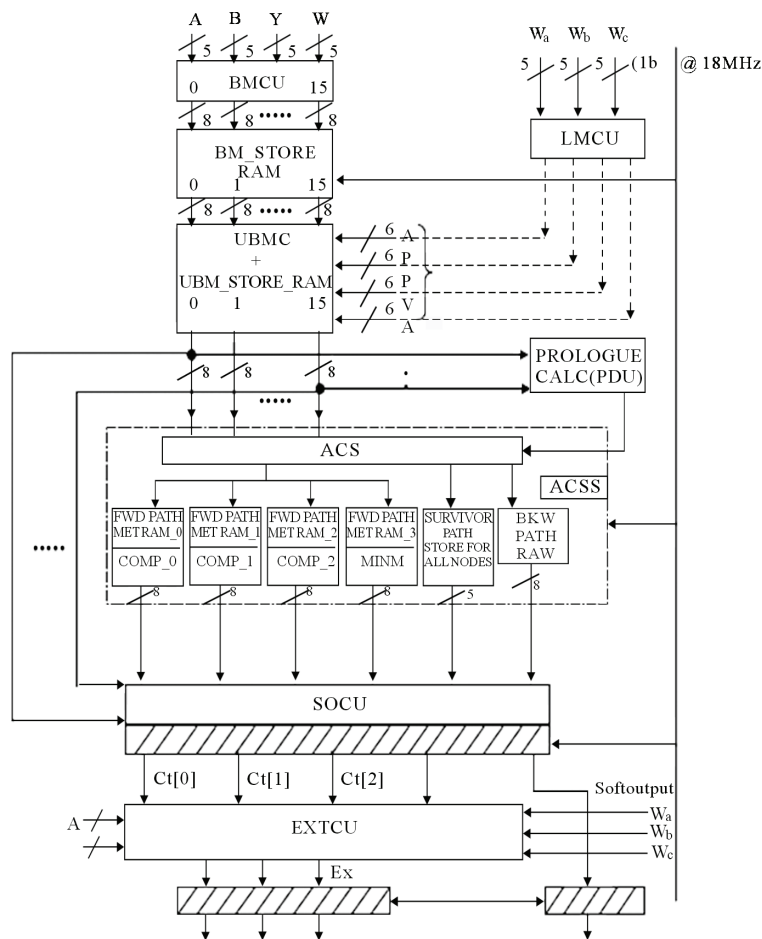

Figure 2. SOVA data path.

IJCNS

The metrics generated by BMCU is stored in the Branch Metric Storage (BMS) and simultaneously routed to Updated Branch Metric Calculation Unit (UMBCU) and Prologue Decoder Unit (PDU). The circular state (S_c) generated by PDU is routed to Add Compare Select Storage Unit (ACSSU) with the updated branch metrics generated from UBMCU. ACSSU first calculates forward path metrics and survivor path (for ML-symbol) and then backward path metrics for all nodes. After processing, its metrics are stored in forward path memory and backward path memory respectively. After calculation of both forward and backward path metrics, Log Likelihood Ratio (LLR) *i.e.*, soft output calculation unit (SOCU) and extrinsic information calculation unit (EXTCU) generates weighted data. The iterative nature has been realized with serial and micro level pipe-lining architecture.

4. Result and Discussion

In this section we present our results on prologue decoding algorithm and BER performance of the DVB-RCS codec with SOVA decoder. Some of the relevant code parameters are indicated below:

Variable information frame length: 12 bytes (48 double-bits), 110 bytes (440 double-bits), 188 bytes (752 double-bits), 212 bytes (848 double-bits); Code rate: 1/2, 1/3; Length of Circular Trellis (IL): information frame length in double-bits; Number of double-bits in a codeword: information frame length in double-bits × (1/Code rate); Interleaver: Type-I and Type-II [5]; Variable number of iterations performed:1, 2, 4, 5 and 8.

Figure 3 presents BER performances of turbo decoder when the prologue decoding is applied in every iteration of turbo decoding and the same is carried out in every alternate iteration. It may be seen that a performance degradation of about 0.1 dB (at BER of 10^{-4}) may be expected if prologue decoding is carried out in alternate iteration. Therefore by performing prologue decoding in alternate iteration, power dissipation of the chip can be reduced with graceful degradation in performance. It shows, proposed method performs better than Saouter's method.

Abbreviations mentioned in the **Figure 3** are explained as: *uncoded signal*: Channel input BER; *Passing actual circular state*: Decoder performance with complete knowledge of the circulation states; *Prologue at each iteration*: Decoder performance when prologue decoding is performed at the start of each iteration; *Prologue at alternate iteration*: Decoder performance when prologue decoding is performed only for even iterations; *Coded as Saouter*: Performance reported in [20] for a rate-1/2 code with constraint length 4 and eight iterations.
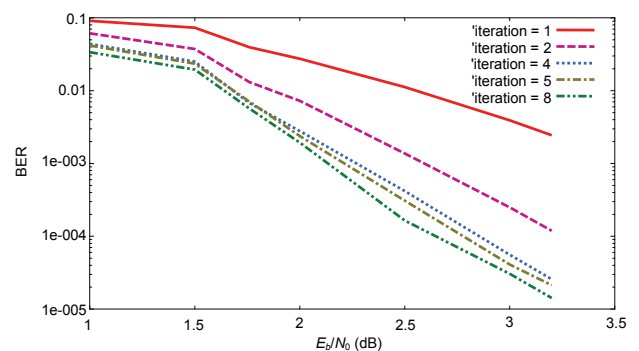
Figure 4 explains BER performance of bi-directional SOVA decoding with variable number of iteration for

double-binary CRSC code. BER of 10^{-4} can be obtained at 2.75dB after 5th iteration. It is also observed that, an improvement in performance between 1st and 2nd iteration is greater than the improvements in performance between 5th and 8th iteration. It signifies that, the rate of improvements in performance decreases with increase of iteration numbers, which is a well known property of turbo code. In this case prologue decoding is performed in every iteration.

Figure 5 explains how the BER performance of bi-directional SOVA varies with variation of interleaver length. A BER of 10^{-5} can be obtained at 1.5 dB with interleaver length of 212 bytes. Whereas the same BER is obtained at 1.75 dB with interleaver length of 110 bytes. It proves that the BER performance of a turbo code is greatly influenced by the length of the interleaver used. Larger length means larger time dispersion (time diversity) *i.e.*, larger the gain obtained. Performance improves as the frame length increases.
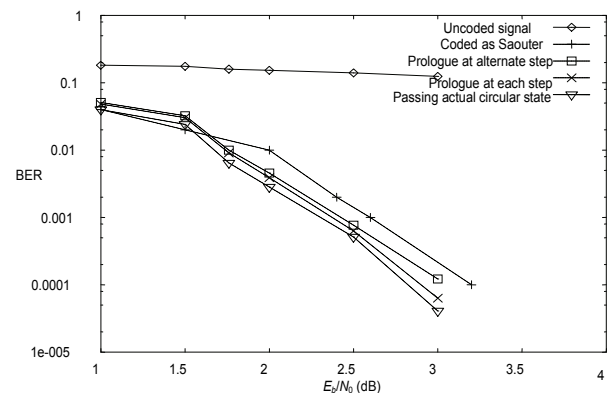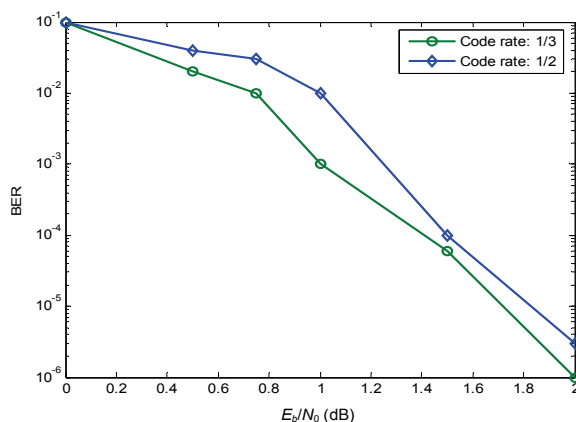

Figure 3. BER performance of bi-directional SOVA and Saouter's decoding for duo-binary CRSC code according to DVB-RCS standard (frame length = 12 bytes, No. of iteration = 5). Performance increases as the prediction of S_c improves.


Figure 4. BER performance of bi-directional SOVA decoding with variable number of iteration for double binary CRSC code (interleaver length = 12 bytes, code rate = 1/3). Prologue decoding is performed in every iteration. Performance increases as number of iteration increases. *Itr_i.dat*: BER curve for *i*-th iteration. E_b/N_0 is in dB.
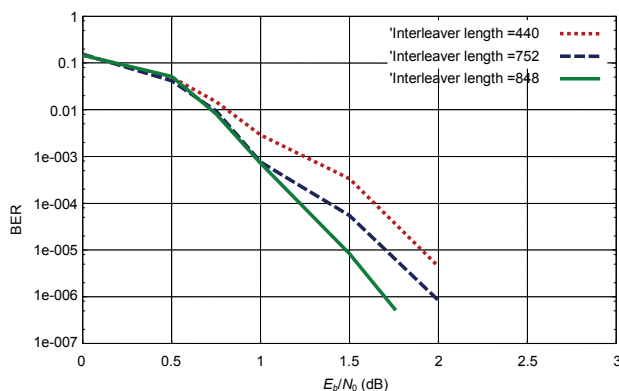
Figure 5. BER performance of bi-directional SOVA with code rate = 1/3, number of iteration is 5 and frame lengths 110 bytes (Interleaver length = 440 double-bits), 188 bytes (Interleaver length = 752 double-bits) and 212 bytes (Interleaver length = 848 double-bits). Prologue decoding is performed in alternate iteration.

Figure 6. BER performance comparison of DVB-RCS standard compliant turbo code with bi-directional SOVA decoder. Code rate = 1/3 and 1/2 for interleaver length 752 (188 bytes) with 5 iterations.
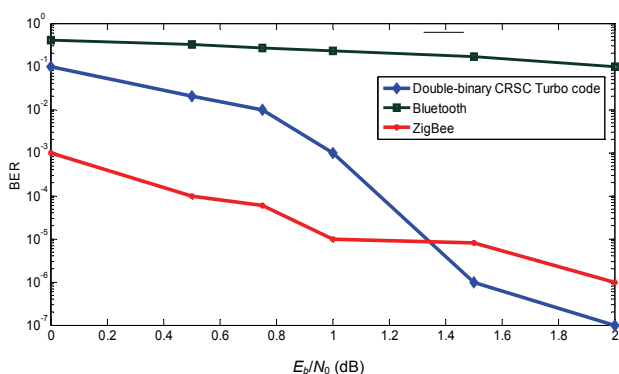
Figure 7. BER performance comparison of DVB-RCS standard compliant turbo code with bluetooth and ZigBee standard. Code rate = 1/3 for interleaver length 752 (188 bytes) with 5 iterations. Turbo code outperforms bluetooth in low and medium SNR and ZigBee in medium SNR region.

Figure 6 explains how the BER vs SNR performance of bi-directional SOVA varies with variation of code rate. It shows that BER of 10^{-3} is obtained at 1.0 dB for code rate 1/3. But with code rate ½ we get BER of 10^{-2} at 1.0 dB. In both cases same code, interleaver length (188 bytes) and iteration number have been used. It can also observe that at high SNR (2.0 dB) BER performance does not improve considerably, though we decrease code rate from 1/2 to 1/3. It can be concluded that code rate of turbo code affects significantly at low SNR and the performance decreases as the code rate increases.

Figure 7 compares the performance of proposed decoding algorithm for turbo code with Bluetooth and ZigBee standard. These two standards are widely used as short range communication for interconnectivity of medical devices. It shows that proposed algorithm completely out performs Bluetooth in both low and medium SNR region. On the other hand, though ZigBee performs better at low SNR, but in medium SNR (1.4 to 2 dB) proposed algorithm over performs it. It happens because CRSC code does not suffer from conventional error floor region.

5. Conclusions

The healthcare market is just starting to rely on information technology in acute care. From hospital wide cellular networks, providing continuous reliable monitoring of patients, to personal medical devices used in the home, wireless connectivity provides benefits to patients, and medical professionals. FEC schemes extends the reach that's possible at any given data rate. In the present work, double-binary turbo coding and its bi-directional SOVA based decoding for reliable communication in healthcare have been discussed in details. Instead of general recursive systematic convolutional code, a special type of component encoder called circular recursive convolutional code has been considered. We have proposed a prologue decoding for double-binary CRSC code. Double-binary code ensures the higher throughput, which is very essential for wireless indoor scenario.

The basic bi-directional SOVA is described for binary turbo code. We have extended it for double-binary case. Double binary turbo code encodes and decodes two bits at a time which leads to certain advantages compare to single bit in binary case. The necessary message passing equations have been derived. It reduces the well known error-floor of classical turbo codes. Thus ensures the data reliability in medium SNR and multi-path scenario of indoor application. The article also presents the concept of prologue decoding. The decoding complexity can be reduced by not performing *prologue* decoding on some iteration, with a minimal loss of performance. At the cost of *prologue* decoding in CRSC, one can avoid the overhead of tail-biting codes. Other interesting future works can be: extension to *m*-binary codes (where $m = 2^2, 2^3$ etc) to get higher throughput; looking for a satisfactory ex-

planation of the quasi-equivalence of the MAP and Max-Log-MAP decoding algorithm; connecting tail-biting codes with CRSC and their prologue decoding; searching for better architecture (both from power and size) which is more suitable to use in portable medical devices.

6. References

[1] C. Berrou, A. Glavieux and P. Thitimajshima, "Near Shannon Limit Error-Control Coding and Decoding: Turbo Codes," *Proceedings of IEEE International Conference on Communications*, Geneva, 23-26 May 1993, pp. 1064-1070.

[2] C. Berrou and M. Jezequel, "Non-Binary Convolutional Codes for Turbo Coding," *Electronics Letters*, Vol. 35, No. 1, 1999, pp. 39-40.

[3] C. Douillard and C. Berrou, "Turbo Codes With Rate-m/(m + 1) Constituent Convolutional Codes," *IEEE Transactions on Communications*, Vol. 53, No. 10, 2005, pp. 1630-1638.

[4] C. Berrou, C. Douillard and M. Jezequel, "Multiple Parallel Concatenation of Circular Recursive Systematic Convolutional (CRSC) Codes," *Annals of Telecommunications*, Vol. 54, No. 3-4, 1999, pp. 166-172.

[5] Digital Video Broadcasting, "Interaction Channel for Satellite Distribution Systems," *Guidelines for the User of EN* 301 790, Version 1.3.1, 2003, pp. 23-27.

[6] B. Vucetic and J. Yuan "Turbo Codes: Principles and Applications," Kluwer Academic Publishers, Boston, 2000.

[7] L. Bhal, J. Cocke, F. Jelinck and J. Raviv, "Optimal Decoding of Linear Codes for Minimizing Symbol Error Rate," *IEEE Transactions on Information Theory*, Vol. 20, No. 2, 1974, pp. 284-287.

[8] J. A. Erfanian, S. Pasupathy and G. Gulak, "Reduced Complexity Symbol Detectors with Parallel Structures for ISI Channels," *IEEE Transactions on Communications*, Vol. 42, No. 234, 1994, pp. 1661-1671.

[9] W. Koch and A. Baier, "Optimum and Sub-Optimum Detection of Coded Data Disturbed by Time-Varying Intersymbol Interference," *Proceedings of IEEE Global Telecommunications Conference*, San Diego, Vol. 3, 2-5 December 1990, pp. 1679-1684.

[10] P. Robertson, P. Hoher and E. Villebrun. "A Comparison of Optimal and Sub-Optimal MAP Decoding Algorithms Operating in the Log Domain," *Proceedings of IEEE*

International Conference on Communications, Seattle, 18-22 June 1995, pp. 1009-1013.

[11] G. Battail, "Ponderation des Symbols Decodes par l'Algorithme de Viterbi," *Annals of Telecommunications*, Vol. 42, No. 1-2, 1987, pp. 31-38.

[12] J. Hagenauer and P. Hoeher, "A Viterbi Algorithm with Soft-Decision Outputs and its Applications," *Proceedings of IEEE Global Telecommunications Conference*, Dallas, 27-30 November 1989, pp. 1680-1686.

[13] C. Berrou, P. Adde, E. Angui and S. Faudeil, "A Low Complexity Soft-Output Viterbi Decoder Architecture," *Proceedings of IEEE International Conference on Communications*, Geneva, Vol. 2, 23-26 May 1993, pp. 737-740.

[14] J. B. Anderson and S. M. Hladik, "Tailbiting MAP Decoders," *IEEE Journal on Selected Areas in Communications*, Vol. 16, No. 2, 1998, pp. 297-302.

[15] D. Gianchristofaro and A. Bartolazzi, "Novel DVB-RCS Standard Turbo Code: Details and Performances of a Decoding Algorithm," *Proceedings of 7th International Workshop on Digital Signal Processing Techniques for Space Communications*, Sesimbra, 1-3 October 2001.

[16] D. Bera and J. Sen, "SOVA Based Decoding of Double-Binary Turbo Convolutional Code," *Proceedings of 1st IEEE International Conference on Wireless Communication*, *Vehicular Technology*, *Information Theory and Aerospace* & *Electronic Systems Technology*, Aalborg, 17-20 May 2009, pp. 757-761.

[17] D. Bera, "Design of Duo-Binary CRSC Turbo Convolution Code," *Proceedings of 4th IEEE International Conference on Computers and Devices for Communication*, Kolkata, 14-16 December 2009, pp. 40-43.

[18] http://www.xilinx.com

[19] G. Masera, M. Mazza, G. Piccinini, F. Viglione and M. Zamboni, "Architectural Strategies for Low-Power VLSI Turbo Decoders," *IEEE Transactions on Very Large Scale Integration Systems*, Vol. 10, No. 3, 2002, pp. 279-285.

[20] Y. Saouter, "Decoding M-Binary Turbo Codes by the Dual Method," *Proceedings of IEEE Information Theory Workshop*, Paris, 31 March-4 April 2003, pp. 74-77.