A Turbo Coding System for High Speed Communications

Yan-Xiu Zheng, *Student Member, IEEE,* and Yu T. Su, *Senior Member, IEEE*

***Abstract*— This paper presents a new turbo coding scheme for high data rate applications. It uses a special interleaver structure that is naturally suited for parallel processing and a multiple-round early stopping test involving both sign check and a CRC code. A memory (storage) management mechanism is included as a critical part of the decoder. The proposed coding scheme offers new design options and tradeoffs that are not available to conventional convolutional turbo codes (CTCs). In particular, it becomes possible for the decoder to employ an efficient inter-block collaborative decoding algorithm, passing the information obtained from stopping test proved blocks to other unproved blocks. It also becomes important to have a proper decoding schedule. The combined effect is improved performance and reduction in the average decoding latency. We show that the memory manager has a modular-like effect in that additional memory units render enhanced performance due not only to less forced early stopping but to possible increases of the interleaving depth. It also provides additional design tradeoff amongst performance, speed and required memory size. Depending on the decoding schedule, the degree of parallelism and other decoding resources available, the proposed scheme admits a variety of decoder architectures that meet a large range of speed and performance demands.**

***Index Terms*— Turbo codes, interleaver, inter-block permutation, parallel decoding, early-stopping.**

I. Introduction

CONVENTIONAL convolutional turbo codes (CTCs) usually employ a block-oriented interleaving so that the message-passing process associated with an iterative decoder is confined to proceed within a block. The performance of such CTCs improves as the block size increases. This is in part due to the fact that the range (interleaving size) of the extrinsic information collected for decoding increases accordingly. But the interleaving size along with the number of iterations are the dominant factors that determine the decoding latency and complexity which, in turn, are often the main concerns that preclude the adoptability of such codes in high rate communication or storage applications.

A technique to overcome the dilemma between increasing the range of message exchange and extrinsic information collection and limiting the interleaving size is the recently proposed inter-block permutation interleaver (IBPI) [1] [2]. For a turbo code (TC) using an IBPI, the encoder partitions the incoming data sequence into L-bit blocks upon which the IBPI performs intra-block and then inter-block permutations. For example, the IBPI may move contents of a block either to coordinates within the same block or to its $2S$ immediate neighboring blocks so that the IBP-interleaved contents of a block are spread over a range of $2S+1$ blocks centered at the original block. Such an IBPI is said to have the (left or right) IBP span S.

An in-depth study on the properties and design of IBPI and IBP-interleaved turbo code (IBPTC) is presented in [3]. An example is given in subsection 2.2 to demonstrate the fact that, unlike a conventional CTC decoder that has a fixed range of message (extrinsic information) passing, the range of message passing for an IBPTC decoder increases as it proceeds with more decoding iterations. Moreover, as explained in Section 3, the corresponding average decoding latency can be kept at least the same as that of a conventional CTC with the same block size and the same number of a posteriori probability (APP) decoders. It suffices to say that, using a fixed block size L, a well-designed IBPI with a proper decoding schedule not only increases the minimum distance of the corresponding IBPTC but also enables an iterative decoder to collect extrinsic information from a range much wider than L while maintaining fixed "local" interleaving size and average decoding delay. Note that an IBPI can be built on any existing block-wise interleavers. Using one of them for intra-block permutation, an IBPI has only to add an extra IBP step.

Besides having an expanding interleaving size, the IBPTC enjoys the advantage of being parallel decodable (see Section 3). To further accelerate an iterative decoder's decoding speed, one can introduce an early stopping mechanism, which also offers the extra benefit of lower the computing power needed for achieving a given performance. The issue of (decoders') stopping criteria has been widely discussed [4]–[6]. These criteria can be classified into four categories: (i) cross entropy (CE) stopping criteria, (ii) sign check (SC) stopping criteria, (iii) soft value (SV) stopping criteria and (iv) cyclic redundancy check (CRC) stopping criteria. The last one guarantees the correctness of decoded bits with a high probability while the others only promise the convergence of the decoded bit sequence. The SC and the CRC stopping criteria use the bit operations only while the remaining two categories operate over the floating-point domain. Moreover, CE and SV stopping criteria have to optimize threshold for different channel conditions whence is less robust. On the other hand, CRC codes have been widely used in the data

Manuscript received March 17, 2006; revised August 26, 2006 and November 19, 2006; accepted January 12, 2007. The associate editor coordinating the review of this paper and approving it for publication was G. Vitetta. This work is supported by the National Science Council of Taiwan under Contract 92-2213-E-009-050. Part of this paper was presented at the 15th IEEE PIMRC, Barcelona, Spain, Sep. 2004.

The authors are with the Department of Communications Engineering, National Chiao Tung University, Hsinchu, Taiwan (email: ytsu@mail.nctu.edu.tw, non2000.cm88g@nctu.edu.tw).

Digital Object Identifier 10.1109/TWC.2007.060086.

link or higher layer as part of the error-control mechanism and is an indispensable component of a packet-oriented data communication system. Using CRC codes as a part of the stopping criterion thus causes little or no extra complexity.

Since an IBPI permutes bits in a block to neighboring blocks within its span, a block-by-block early stopping scheme will inevitably result in stopping time variation over different blocks. On the other hand, the special structure of IBPIs implies that bits in neighboring blocks are algebraically related, hence the information about bits in terminated blocks can be used to help decoding bits in unterminated blocks, i.e., one can actually take advantage of the stopping time variation if proper statistical information can be extracted from terminated blocks.

This paper presents a new novel dynamic codec structure that allows collaborative decoding among different blocks in the above-mentioned sense. When used in conjunction with a highly reliable multiple round stopping test using both SC and a CRC code, the proposed coding scheme yields low latency (small average decoding iteration number) while achieving very impressive performance.

We start our presentation in the next section with a description of the proposed codec system model and related iterative decoding procedure. Issues related to latency, parallel decoding and some implementation aspects are addressed in Section 3. In Section 4, we present the structure of a variable termination (stopping[1]) time IBPTC decoder and propose some multiple-round stopping criteria and tests. The next section addresses the issue of storage requirement and suggests a general dynamic memory management algorithm. As an application example, two re-transmission protocols based on our coding scheme are presented. Section 6 provides some numerical examples that validate the superiority of the proposed coding scheme and finally, in Section 7 we summarize our main results.

In addition to the abbreviations mentioned above, we also use the following acronyms. DR stands for decoding round, ADU for an APP decoding unit, and SRID for single-round interleaving delay. Moreover, we have ST (stopping test), SC (sign check), MRT (multiple-round test), MRST (multiple-round stopping test), HST (hybrid stopping test), VTT (variable termination time), MU (memory unit), ESD (extended stopping decision) and some combinations of the above acronyms, e.g., SCST SVST, CRCST, MR-HST, etc.

II. System Model and IBP Behavior

A. System model

Shown in Fig. 1 is a generic block diagram for a communication system using an IBPTC. The input data sequence $\mathbf{X}$ is partitioned into blocks of the same length, $\{\mathbf{X}_1, \mathbf{X}_2, \cdots\}$, where $\mathbf{X}_i$ is a row vector of length $L-K_{CRC}$ representing the ith block. They are CRC-encoded into $\mathbf{W} = \{\mathbf{W}_1, \mathbf{W}_2, \cdots\}$, where $\mathbf{W}_i = \{w_{i0}, w_{i1}, \cdots, w_{i(L-1)}\}$ is a row vector with length L. $\mathbf{W}$ is formed by padding at the end of each data block parity bits that are the coefficients of the remainder polynomial $r(x)$ obtained by dividing a data polynomial associated with a data block by a binary generator polynomial $g(x)$ of order K_{CRC}. Of course, the degree of $r(x)$ is less than K_{CRC}. The corresponding probability of undetectable error is roughly equal to $2^{-K_{CRC}}$. In other words, longer CRC codes possess better error detection capability.

[1]We will use stopping and termination interchangeably throughout the paper.

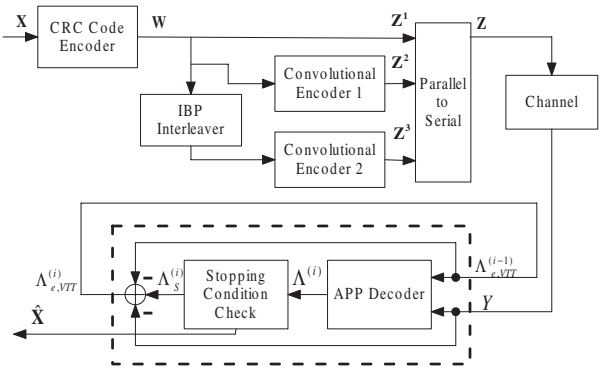

Fig. 1. A block diagram for the proposed coding system in which the encoder uses an IBP interleaver. More than one VTT-APP decoding unit can be used; the notations denote various extrinsic information for the ith decoding unit.

The CRC encoder output $\mathbf{W}$ and its IBPI-permuted version $\mathbf{W}' = \{\mathbf{W}'_1, \mathbf{W}'_2, \cdots, \}$ are then encoded to form the coded sequence $\mathbf{Z} = \{\mathbf{Z}^1, \mathbf{Z}^2, \mathbf{Z}^3\}$, where $\mathbf{Z}^i = \{\mathbf{Z}^i_1, \mathbf{Z}^i_2, \cdots\}$ and the superscript i is used to denote the systematic part ($i = 1$), the first encoder's output (parity) sequence ($i = 2$) and the second encoder's output sequence ($i = 3$). As can be seen, the only difference between a CTC and an IBPTC encoders is the interleaver used.

The receiver uses one or multiple ADUs like that shown in lower part of Fig. 1 to decode the corresponding received baseband sequence $\mathbf{Y} = \{\mathbf{Y}^1_1, \mathbf{Y}^2_1, \mathbf{Y}^3_1, \mathbf{Y}^1_2, \mathbf{Y}^2_2, \mathbf{Y}^3_2, \cdots\}$, where $\mathbf{Y}^i_j$ is the subsequence corresponding to $\mathbf{Z}^i_j$; other notations are defined in the next section. An ADU consists of an APP decoder and a stopping condition checker. It also performs the corresponding interleaving or de-interleaving and other related operations but for simplicity we do not show these operations in the figure.

The stopping condition checker applies CRC check and/or other forms of STs to verify if the APP decoder output satisfies the stopping criterion. An affirmative answer leads to the decision to stop (terminate) decoding the block in question and this is the only possible early-stopping opportunity for CTCs. Besides such a regular early-stopping, however, there are two other early-stopping opportunities for IBPTCs since no matter whether the decoder output passes the ST, the corresponding soft output is interleaved or de-interleaved to the neighboring blocks within the IBP span. The ADU will then examine each block within the span to see if a block's content has been filled with stop-decoding decisions. If such a block is found the ADU will issue a termination decision accordingly. The ADU can also run STs on these neighboring blocks and make a termination decision. We refer to the latter two early-stopping possibilities as extended (or pre-decoding) early-stoppings.

Note that a decoding iteration consists of two DRs that are respectively responsible for decoding the pre-permuted (non-interleaved) $\mathbf{Z}^2_j$ and post-permuted (interleaved) blocks $\mathbf{Z}^3_j$ and CRC check is feasible for pre-permuted blocks only. Hence

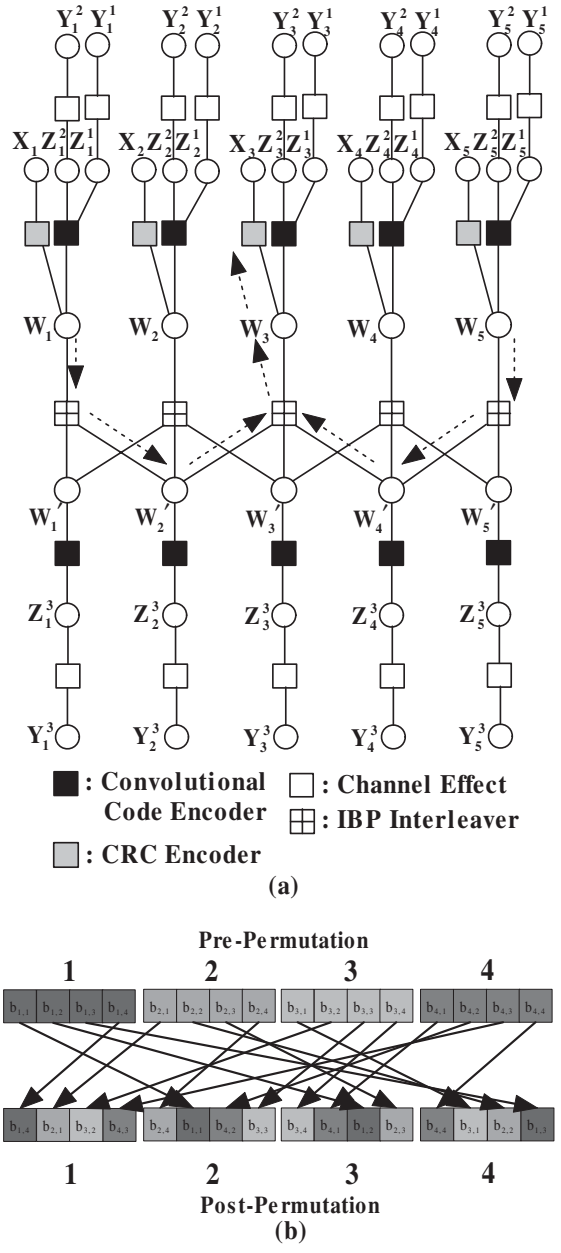

Fig. 2. (a) A graph representation for a CRC and IBPTC encoded system with interleaving span $S = 1$, (b) An inherent IBP structure can be found in most practical interleavers.

in the first DR one can perform both regular and extended early-stopping tests, but in the second DR, only extended early-stopping is viable unless the ST does not involve a CRC check. Examples are given in Section 5 to further elaborate this property of IBPTCs.

B. Graphical representation of an IBPTC

Fig. 2 (a) is a graphical representation for the system of Fig. 1 with a symmetric IBPI of interleaving span $S = 1$ and an input data sequence $\mathbf{X}$ of five-block duration. The dark, gray, crossed and blank squares represent respectively the functions of convolutional encoder, CRC encoder, IBPI and the channel effect. This extended graph is used to describe the iterated

CTC/IBPTC

APP Decoding Round \ Block Index	1	2	3	4	5	6	7
1	1/1	5/2	9/4	13/7	17/11	21/15	25/19
2	2/3	6/5	10/8	14/12	18/16	22/20	26/23
3	3/6	7/9	11/13	15/17	19/21	23/24	27/26
4	4/10	8/14	12/18	16/22	20/25	24/27	28/28

Fig. 3. A comparison of exemplary decoding schedules for CTC and IBPTC when decoding 7 blocks with 2 iterations (four DRs). The numbers a/b in the constituent squares represent the order the APP decoder performs decoding for CTC/IBPTC. Hence the first block of the CTC is decoded by the first 4 DRs (the left upper numbers in the second leftmost column) but that of the IBPTC is decoded by the first, third, sixth and tenth DRs (the right lower numbers in the same column); see subsection 2.3 for detailed discussion.

decoding behavior. As one can see, the content of a given block, say $\mathbf{W}_2$, is interleaved to parts of itself $\mathbf{W}'_2$ and the two neighboring blocks $\mathbf{W}'_1$ and $\mathbf{W}'_3$ to its immediate left and right.

At the first DR, the decoder uses $\mathbf{Y}_i^1$, $\mathbf{Y}_i^2$ to decode block $\mathbf{W}_i$. The extrinsic information of, say $\mathbf{W}_1$ and $\mathbf{W}_5$, is interleaved for use in decoding $\mathbf{W}'_1$, $\mathbf{W}'_2$ and $\mathbf{W}'_4$, $\mathbf{W}'_5$ in the second DR. It is easy to see that, in decoding the third block $\mathbf{W}_3$ at the beginning of the second iteration, the decoder can use as a priori information some message passed from all five neighboring blocks. In general, an IBPTC decoder can exploit information collected from $4SI+1$ adjacent blocks in I iterations while, as will be shown in the ensuing subsection, the average decoding delay between two output blocks is kept fixed. This message passing range expansion capability implies that an IBPI can have an unbounded equivalent interleaving depth (size) that is constrained only by the numbers of turbo decoding iterations and the data blocks involved in decoding while keeping the interleaving delay per iteration bounded by its local interleaving depth.

III. Latency, Parallelism, and Decoding Schedule

Latency is perhaps the most important issue in high speed codec design. We first analyze the encoding and decoding delays when only one APP decoder is used. The corresponding decoding delay for a parallel decoder is minimized by using a proper decoding schedule. But even if only one APP decoder is used, as we will see shortly, the decoding schedule still plays a pivotal role in minimizing the decoding delay of an IBPTC.

A. Encoding and decoding latency

Although the interleaving process of an IBPI is defined by the composition of the intra- and inter-block permutations, it can be implemented by a single step. The encoder knows to which position each bit (or sample) in a given block should be moved and can do so immediately after it receives each incoming bit. But to encode a given, say the ith, interleaved

block $\mathbf{U}_i'$ into $\mathbf{X}_i^2$, it has to wait until the complete $(i+S)$th block is received. The time elapsed between the instant the encoder receives the first bit of the ith block and the moment when it receives the last bit of the $(i+S)$th block and outputs its first encoded bit of $\mathbf{X}_i^2$ is simply $(1+S)L$-bit durations. By contrast, a CTC with a block size of L bits has an encoding delay of approximately L bits.

The single-round interleaving (or de-interleaving) delay is proportional to the encoding delay. But the total decoding delay is a much more complicated issue. For a decoder that uses a single ADU, the decoding delay depends mainly on three variables: the single-round interleaving delay (SRID), the single-round APP decoding delay, and the number of decoding iterations. As the SR APP decoding delay (speed) is usually much less than the SRID, we ignore the APP decoding delay in the subsequent discussion.

For the first decoding of each incoming block, there can be zero waiting time, but for later DRs the corresponding delays depend on, among other things, the decoding schedule used. With the same block size, the decoding delay of the first received block for the CTC is definitely shorter than that for the IBPTC. But if one considers a period that consists of multiple blocks (otherwise one will not have enough blocks to perform inter-block permutation) and takes the decoding schedule into account, then the average decoding latency difference can be completely eliminated. This is because the APP decoder (including the interleaver and deinterleaver) will not stay idle until all blocks within the span of a given block are received. Instead, the APP decoder will perform decoding-interleaving or deinterleaving operations for other blocks according to a predetermined decoding schedule before it can do so for the given block (and the given DR).

If we define the total decoding delay as the time span between the instant a decoder receives the first input sample (from the input buffer) and the moment it outputs its last decision then it is possible that both the IBP and the classic approaches yield the same total decoding delay even if only one APP decoder is used. We use the following example and Fig. 3 to support our claim; its generalization is straightforward.

Suppose we receive a total of 7 blocks of samples (in a packet, say) and want to finish decoding in 2 iterations (4 DRs). One can easily see from Fig. 3 that a CTC decoder would output the first decoded block in 4 DT cycles, where DT is the number of cycles needed to perform a single-block APP decoding plus SRID. The IBPTC decoder, on the other hand, needs 10 DT cycles to output its first decoded block. However, if one further examines the decoding delays associated with the remaining blocks, then one finds they are 8, 12, 16, 20, 24, and 28 DT cycles for the CTC decoder while those for the IBPTC decoder are 14, 18, 22, 25, 27 and 28 DT cycles, respectively. So in the end, both approaches reach the final decision at the same time.

It can be shown that, for a decoder with $2N$ DRs and $S = 1$, both decoders result in a constant delay of $2N$ DT cycles between two adjacent output blocks, except for the first block and the last $2N-1$ blocks. For an $S = 1$ IBPTC, the decoder requires a first-block decoding delay of $N(1+2N)$ DT cycles while that for the CTC is only $2N$

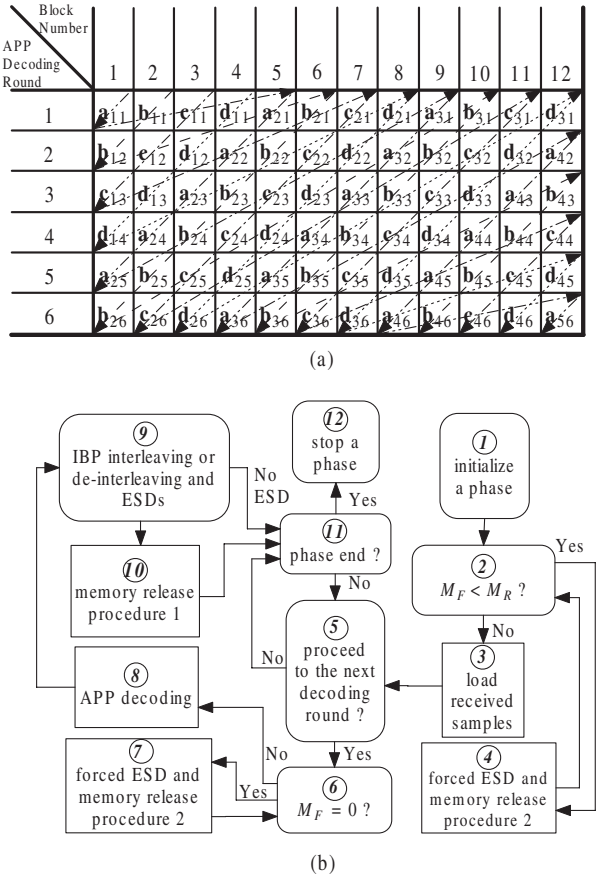

Fig. 4. (a) A multiple zigzag decoding schedule for IBPTCs with an interleaving span $S = 1$; (b) A joint memory management and IBPTC decoding procedure.

DT cycles. The inter-block decoding delays, i.e., decoding latency between two consecutive output blocks, for the last $2N-1$ output blocks of an IBPTC decoder using a decoding schedule similar to that shown in Fig. 3 (e.g., the one shown in Fig. 4 (a)) form a monotonic decreasing arithmetic sequence $\{2N-1, 2N-2, \cdots, 1\}$ (in DT cycles). The inter-block decoding delay of a CTC decoder remains a constant $2N$ DT cycles. On the average, both codes give the same inter-block decoding delay.

B. Memory contention and decoding schedule for multiple ADUs

The above assessment on the encoding/decoding delay is made under the assumptions that both codes use the same block size L, no early stopping mechanism is applied, and a single ADU is used. The delay will be shortened if the latter two assumptions are removed. In particular, the decoding delay can be reduced significantly by using multiple ADUs for parallel decoding[2]. When iterative APP decoding is performed by multiple ADUs, these ADUs have to access memory via interleaver (or deinterleaver) for extrinsic information update

[2]The decoding delay can also be reduced by using a faster APP decoder. But as mentioned in previous subsection, the APP decoder delay is much less than SRID hence speeding up APP decoder does not help much in reducing the total decoding delay.

and exchange. To have the maximum delay reduction, the interleaver should also have a parallel structure to avoid memory access collision. It can be shown that the structure of IBP interleavers allows flexible degrees of parallelism and highly parallel memory access. In fact, *Theorem 1* of [2] implies that a good IBPI should possess the local-invariant property that preserves the relative position within a block during the IBP process. This property promises contention-free across the span (parallel-decodable blocks) of the IBPI. Furthermore, like the contention-free interleaver design presented in [7], closed-form contention-free IBP rules are available and, more importantly, as were shown in [2] [3], they guarantee some good distance properties for the associated IBPTC.

Just the same as the single ADU case, the decoding schedule for multiple ADUs is a critical design concern. An example of decoding an IBPTC with multiple APP decoders is given in Section 5; see Fig. 4 (a). Although both IBPTCs and CTCs can use multiple decoders for parallel decoding and apply an early stopping method to shorten the decoding latency, we will explain in the next section and prove numerically in Section 6 that the former class does derive much more benefit in block error rate (BER) performance.

C. Block-oriented consideration

Although we have assumed a stream-oriented scenario so far, our arguments are valid for the conventional block-oriented consideration as well. It is thus of paramount importance that we recapture the IBP concept from the block-oriented viewpoint before returning to the main discourse.

For a block-oriented CTC that uses a reasonable good interleaver of size N, partitioning each N-bit group into $L = \lfloor N/W \rfloor$-bit blocks immediately transforms the interleaving rule into an IBP structure like that shown in Fig. 2(b). Such a structure can also be found in other codes such as product codes. Consequently, all conventional CTCs and product codes can be regarded as subclasses of IBPTCs. There are, however, two major distinctions between CTCs and most other subclasses of IBPTCs.

Firstly, for a CTC with an interleaving size of W blocks, encoding within each disjoint group of W consecutive blocks is continuous across blocks while a product code encodes each row (column) separately (discontinuously). More specifically, the product code encoder divides information stream into multiple blocks and independently encodes each block. In general, the class of IBPTCs can encode each block either separately or continuously. Secondly, an interleaver used in a CTC, after the above virtual regular partition, usually yields a non-regular local interleaving structure, i.e., the interleaving relation between a block and other blocks in the same group does not follow the same permutation rule. In contrast, product codes and many IBPTCs have much more regular local interleaving structures. An appropriate regular local interleaving (and deinterleaving) structure makes implementation easier and, as mentioned before, provides properties that are useful for parallel decoding, e.g., (memory access) contention-free and simple routing requirement. Moreover, with or without parallel decoding, as the following example shows, it also results in reduced decoding latency.

Consider the example illustrated in Fig. 3. For a CTC with an interleaving (block) size of $7L$ bits, the first-block decoding delay for a 2-iteration single-ADU decoder is 28 DT cycles. But if one divides this $7L$-bit block into 7 subblocks and uses a special block-oriented interleaver which performs successive intra-subblock and inter-subblock permutations on these subblocks, the corresponding (2-iteration single-ADU) decoding delays in DT cycles for these subblocks are 14, 18, 22, 25, 27 and 28, respectively. Therefore, although both code structures result in identical total decoding delay the IBPTC structure is able to supply partial decoded outputs much earlier. This feature, when combined with proper intra-(sub)block and inter-(sub)block interleaving rules, multiple ADUs, optimized decoding schedule and implementation resource management, become very beneficial for high speed applications. More importantly, it can been shown by computer simulations that a turbo code with such an interleaver does not yield performance inferior to that of a CTC with a block-oriented interleaver (e.g., 3GPP interleavers) of the same size.

IV. Iterative Decoder with Variable Termination Time

A conventional iterative decoder is composed of one or more APP decoders that will not stop decoding until a fixed number of decoding iterations have been performed. With an early-stopping mechanism in place, as shown in Fig. 1, the decoding procedure can stop (terminate) at the end of an iteration (two DRs) or at the end of a DR. We refer to such a decoder as a variable termination time APP (VTT-APP) decoder or simply a VTT decoder. When a ST is included in the turbo decoding process, the test results in either a stop- or a continue-decoding decision. Given the decision, which is very useful side information, our computation of the extrinsic information and soft output should be modified accordingly. The first part of this section presents our modifications while the second part suggests several highly reliable STs.

Note that all STs, whether they are used in CTCs or IBPTCs, incur additional computational complexity which is usually more than compensated for by the reduced average DRs brought about by the use of a ST.

A. Extrinsic output after a stopping test

Let $\Lambda(w) = \log \frac{p(w=0)}{p(w=1)}$ be the log-likelihood ratio of the random variable w where $p(\cdot)$ denotes the probability density function of w. If w_{jk} represents the kth bit of the jth block and $\Lambda^{(i)}(w_{jk})$, $\Lambda_e^{(i)}(w_{jk})$ denote the corresponding estimated log-likelihood ratio and the extrinsic information obtained at the end of the jth block's ith DR, we have [4]

$$\Lambda_e^{(i)}(w_{jk}) = \Lambda^{(i)}(w_{jk}) - \Lambda_e^{(i-1)}(w_{jk}) - L_c \cdot y_{jk}^1, \qquad (1)$$

We assume that $\Lambda_e^{(-1)}(w_{jk}) = 0, \ \forall \ j,k$. $L_c = 4aE_s/N_0$ represents the channel reliability, where a is the signal amplitude which is usually normalized to 1 for additive white Gaussian noise (AWGN) channel, E_s being the signal energy per symbol while N_0 is the noise power spectral density.

For the ith DR of the jth data block, the VTT-APP decoder in charge uses baseband vectors $\mathbf{Y}_j^1, \mathbf{Y}_j^2$ or $\mathbf{Y}_j^3$ and the

a priori information $\{\Lambda_e^{(i-1)}(w_{jk})\}_{k=0}^{k=L-1}$ as its input and outputs $\{\Lambda_e^{(i)}(w_{jk})\}_{k=0}^{k=L-1}$ for use in the next DR as a priori information until $i = D_{max}$, where D_{max} is the maximum allowed APP DRs; see Fig. 1.

A tentative decision $\hat{w}_{jk}^i$ on the kth bit of the jth block at the end of the ith APP DR can be obtained by

$$\hat{w}_{jk}^i = \begin{cases} 0 & , \ \Lambda^{(i)}(w_{jk}) \geq 0, \\ 1 & , \ \Lambda^{(i)}(w_{jk}) < 0. \end{cases} . \tag{2}$$

Let $Q(\widehat{\mathbf{W}}_j^i)$ be the stopping indicator for the tentative decision vector of the jth block at the ith DR, $\widehat{\mathbf{W}}_j^i = (\hat{w}_{j0}^i, \hat{w}_{j1}^i, \cdots, \hat{w}_{j(L-1)}^i)$, $0 < i \leq D_{max}$, i.e.,

$$Q(\widehat{\mathbf{W}}_j^i) = \begin{cases} 1, & \text{if } \widehat{\mathbf{W}}_j^i \text{ satisfies the ST} \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

Given the ST result, the conditional soft value $\Lambda_S^{(i)}(w_{jk})$ and the extrinsic information $\Lambda_{e,S}^{(i)}(w_{jk})$ are given by

$$\Lambda_S^{(i)}(w_{jk}) = \begin{cases} \log \frac{P\left[w_{jk}=0 \middle| Q\left(\widehat{\mathbf{W}}_j^i\right)=1\right]}{P\left[w_{jk}=1 \middle| Q\left(\widehat{\mathbf{W}}_j^i\right)=1\right]}, & Q(\widehat{\mathbf{W}}_j^i) = 1 \\ \Lambda^{(i)}(w_{jk}), & Q(\widehat{\mathbf{W}}_j^i) = 0 \end{cases} \tag{4}$$

and

$$\Lambda_{e,S}^{(i)}(w_{jk}) = \Lambda_S^{(i)}(w_{jk}) - \Lambda^{(i)}(w_{jk}). \tag{5}$$

The extrinsic information $\Lambda_{e,VTT}^{(i)}(w_{jk})$ of an APP decoder then becomes

$$\begin{aligned} & \Lambda_{e,VTT}^{(i)}(w_{jk}) = \Lambda_{e,S}^{(i)}(w_{jk}) + \Lambda_e^{(i)}(w_{jk}) \\ = \ & \Lambda_S^{(i)}(w_{jk}) - \Lambda_{e,VTT}^{(i-1)}(w_{jk}) - L_c y_{jk}. \end{aligned} \tag{6}$$

The resulting VTT-APP decoder is shown in Fig. 1.

To ease the burden of computing the conditional log-likelihood function that appears in (4), we make the idealized assumption that the stopping test is perfect, i.e.,

$$P(\hat{w}_{jk}^i \text{ is correct} | Q(\widehat{W}_j^i) = 1) = 1, \ \forall \ k$$

With this perfect stopping decision assumption, (4) becomes

$$\Lambda_S^{(i)}(w_{jk}) = \begin{cases} \Lambda^{(i)}(w_{jk}) \cdot \infty, & Q(\widehat{\mathbf{W}}_j^i) = 1 \\ \Lambda^{(i)}(w_{jk}), & Q(\widehat{\mathbf{W}}_j^i) = 0 \end{cases} , \tag{7}$$

and (6) is modified accordingly.

The perfect stopping assumption actually makes the computation of extrinsic information or soft output easier as when the tentative decision vector $\widehat{W}_j^i$ meets the stopping condition, then $\Lambda_{e,VTT}^{(i)}(w_{jk})$ has only two values $\pm\infty$. When the perfect stopping assumption is approximately true (say, the false stopping probability is less than 10^{-5}), a practical approximation is to assign a fixed large number to $\Lambda_{e,VTT}^{(i)}(w_{jk})$. However, it should be noted that, after interleaving or de-interleaving, the large metric value will be passed to neighboring blocks and then to the corresponding partial path metric computers, eliminating other branches which are not associated with these bits. Hence the passing of the extrinsic information of these perfect detected bits to neighboring blocks further reduce the complexity of the associated APP decoder. Moreover, as the APP decoder selects survivor branches based on the relative magnitudes of the partial path metrics only, the actual value assigned to $\Lambda_{e,VTT}^{(i)}(w_{jk})$ is immaterial. In fact, it can be as simple as a binary sign telling the APP decoder which branches should be eliminated.

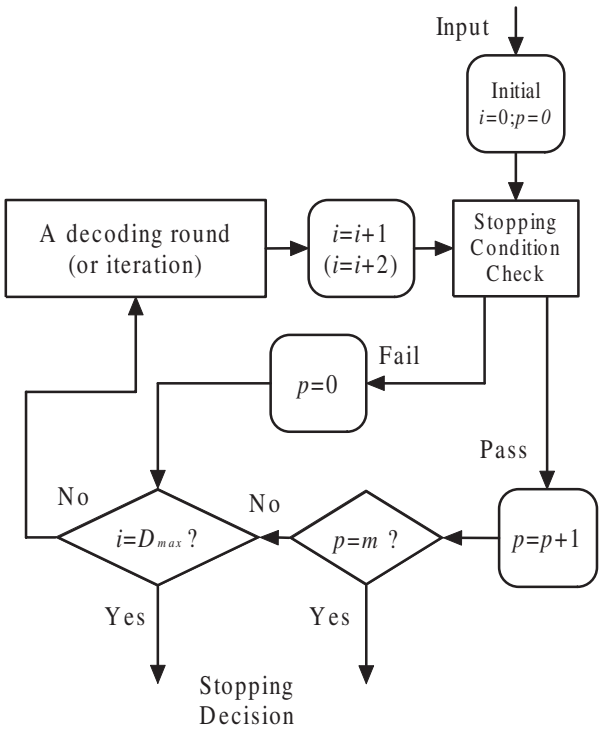

Fig. 5. Flow chart of a general m-round stopping test.

All these nice features depend, besides the IBP design, on the availability of a highly reliable block ST such that the perfect stopping assumption holds with a probability close to 1, which is the subject of the following subsection. Note that although there is no perfect ST and the probability that a ST gives a wrong block stopping decision is nonzero, the influence of these wrong indications results in no catastrophic failure as our numerical results will demonstrate later.

B. Multiple-Round Stopping Tests

[6] summarized various STs for turbo decoders using sign check, soft values and CRC checks. The sign check stopping test (SCST) compares the tentative decoded bits from two successive rounds. A tentative decoded block passes the test if most or all of them are consistent. The soft value stopping test (SVST) compares the soft value(s) with a threshold; the soft values can be the reliability of tentative decoded soft bits, the average soft value of a block, the extrinsic value of the least reliable bit etc. The CRC stopping test (CRCST) uses the CRC result to decide if further decoding of a block is needed. SCST and CRCST operate over bit level but SVST operates over the real domain. The performance of SVST is subject to the choice of the threshold which, in turn, is a function of the channel condition and code structure. Moreover, the convergence rate of soft bit values also depends on the above two factors [6], [8]. In short, the classes of CRCST, SCST or their variations have the complexity and robustness advantages over the class of SVST.

1) A general algorithm: All early stopping tests are sequential in nature. They either compare or manipulate some values corresponding to two consecutive DRs, or just check a single DR output to make a stop-or-continue decoding decision. In contrast, our proposed tests make a stop-decoding decision based on multiple observations and are thus referred to as multiple-round stopping tests (MRSTs)

It is well known that a statistical decision based on a single observation is inferior to that based on multiple observations which, however, require a longer observation time (or equivalently, larger sample size). The MRST has the distinct capability of balancing performance (reliability of the test) and cost (time or sample size needed to make a termination decision). A dismissal on a decoder output is issued as soon as it fails a single test but a decision to stop decoding a block has to wait until the same block is verified by several rounds of test. Therefore, incorrect tentative decoder outputs are quickly discarded while any final decision on a block is prudently made. Since most of the DRs do not lead to the final decision of a block, an MRST that consists of a series of simple, short-duration tests spends much less time and overhead on these intermediate DRs than those required by a single long-duration test. While the first round of an MRST provides an initial tentative decision, the additional verification test rounds greatly reduce the probability of false stopping and give more robust and reliable decision, avoiding spreading incorrect information to neighboring blocks. An m-round test using a short CRC-k code gives a false detection probability similar to that of using a long CRC-mk code but with only $1/m$ overhead bits. Moreover, since a correct stopping on a certain block helps bringing earlier stoppings to its adjacent blocks, the average decoding latency is shortened as well.

A flow chart of the general MRST is shown in Fig. 5. In this figure, i is used to denote the ith DR, p represents the number of times a block has passed a ST and can be regarded as a quality indicator, m is the required quality condition and D_{max} is the maximum number of DRs allowed. Either $p = m$ or $i = D_{max}$ will force the decoding process to be terminated. As discussed in subsections 2.1 and 2.2, an ST is performed at the end of an iteration (even DRs) or the beginning of an odd DR. For the latter case, an ST means checking if all pre-permuted blocks within its span have satisfied the stopping condition. A special case of MRST is the multiple-round SCST of [6]. It was found that the block error rate (BER) performance improves as the number of test rounds increases.

As mentioned before, we shall not consider the class of SVSTs. Multiple-round CRCST, SCST and a hybrid CRC-SC ST are briefly defined in the following.

2) T1.m: the m-round CRCST: This scheme is based on an m-round CRC test. A block is said to pass the m-round CRCST if all m consecutive tentative decision vectors $\widehat{\mathbf{W}}_j^{i-m+1}, \widehat{\mathbf{W}}_j^{i-m+2}, \cdots, \widehat{\mathbf{W}}_j^{i}$ succeed in passing the same CRC test, i.e., $I_{CRC}(\widehat{\mathbf{W}}_j^{l}) = 1,\ l = i-m+1, i-m+2, \cdots, i$ and $i \le D_{max}$, where

$$I_{CRC}(\widehat{\mathbf{W}}) = \begin{cases} 1, & \widehat{\mathbf{W}} \text{ passes CRC condition} \\ 0, & \text{otherwise} \end{cases}. \tag{8}$$

As the error detection capability of a CRC code is an increasing function of the code length, one can trade the order m for the code length.

3) T2.m: the m-round SCST: This ST [6] compares tentative decoded bits in m ($m \ge 2$) consecutive DRs or iterations. The decoder stops when the nth tentative decision vector, $i \le D_{max}$, are the same with the previous $m-1$ tentative decision vectors, i.e.,

$$\hat{w}_{jk}^{i-m+1} = \hat{w}_{jk}^{i-m+2} = \cdots = \hat{w}_{jk}^{i}, \quad \forall\ k,\ 0 \le k < L. \tag{9}$$

Note that MR-SCST checks the convergence of tentative decisions, it does not guarantee the convergence to the correct decisions.

4) T3.m: the m-round hybrid stopping test (MR-HST): Unlike CTCs, errors in STs for IBPTC will propagate to different blocks and might lead to a catastrophic consequence. A highly reliable ST can be obtained by increasing m or it can be obtained by incorporating multiple criteria in a single round. A block that passes both CRC and SC tests is more reliable than one that passes only a single test.

Hence, we suggest the hybrid stopping criterion

$$I_{CRC}(\widehat{W}_j^{l}) = 1,\ \forall\ l, i-m < l \le i,\ i \le D_{max}, \tag{10}$$

and

$$\hat{w}_{jk}^{i-m+1} = \hat{w}_{jk}^{i-m+2} = \cdots = \hat{w}_{jk}^{i}, \quad \forall\ k, 0 \le k < L,\ i \le D_{max}. \tag{11}$$

If the CRC-8 is used, the undetect error probability is approximately 2^{-8} only. The probability that the sign check does not match the CRC result is of the order 2^{-16} or 2×10^{-5}. Using a longer CRC code increases the reliability of a CRC test but it also implies an increase in the overhead. Additional sign consistency check is the price we paid for using a short CRC code to cut down the CRC overhead.

5) Genie stopping test: Genie ST is a hypothetic ideal test that is capable of verifying the tentative decision vector without error. The performance of this ideal test is used as the ultimate bound for reference purpose.

At the first glance, we might expect the hybrid test or higher-order (larger m) tests to take more DRs since a received block is less likely to pass both SC and CRC or a higher-order requirement. But the fact is that a correct block decision, through the IBP interleaving, will help other blocks to meet the stopping condition sooner while an incorrect one tends to has an adverse effect. Our numerical experiment indicates that the hybrid test not only gives better performance but also requires less DRs. This is another advantage of IBPTCs that is not shared by CTCs.

V. VTT Decoder with Finite Memory Space

A. Decoding schedule and early stopping

In Section 3, we have demonstrated the importance of the decoding schedule in minimizing the decoding delay. Parallel decoding is a popular design option to shorten the latency. Fig. 4 (a) shows a multiple expanding-window zigzag scheduling table for decoding an IBPTC with IBP span $S = 1$ and four ADUs, denoted respectively by **a**, **b**, **c** and **d**. Data blocks processed in the odd rows are in the original (pre-permutation) order while those processed in the even rows

are in the interleaved (post-permutation) order. Each dashed or dotted zigzag curve represents the schedule for an ADU. The symbol $\mathbf{x}_{mn}$ denotes the nth DR of the mth phase in the ADU $\mathbf{x}$'s schedule, where a DR represents the APP decoding of a pre- or post-permuted block and the associated interleaving or de-interleaving and the mth phase refers to the mth parallel line associated with an ADU's decoding schedule. Obviously, the mth decoding phase of $\mathbf{x}$ is followed by the $(m+1)$th decoding phase to its right.

Taking the decoding schedule of ADU $\mathbf{c}$ as an example, its first DR of the first phase $\mathbf{c}_{11}$ corresponds to the first DR of Block 3 while the first phase' second DR $\mathbf{c}_{12}$ corresponds to the second DR of Block 2. $\mathbf{c}_{12}$ can be performed, as the scheduling table shows, after Blocks 1,2,3 have been decoded once and the corresponding extrinsic information output has been inter-block interleaved so that the post-permuted Block 2 has all a priori information needed for a new DR. $\mathbf{c}$ finishes its first phase after $\mathbf{c}_{13}$ is done. It then proceeds with the first DR of the next phase $\mathbf{c}_{21}$, i.e., the first DR of Block 7. An ADU can not start a new DR until the DR on its top is completed, e.g., $\mathbf{a}_{2k},\ k>2$ cannot start unless the DR corresponding to $\mathbf{d}_{12}$ is finished.

As mentioned at the end of subsection 2.1, an ADU can make both regular and extended stopping decisions (ESDs) in odd rows' DRs only, unless a non-CRC-based ST is used. For DRs in even rows, however, ESDs are still feasible. For example, in $\mathbf{a}_{23}$ ($\mathbf{c}_{25}$) we check if block 3 passes the ST and early stopping on this block becomes effective if affirmative. ADU $\mathbf{a}$ ($\mathbf{c}$) then go on to examine whether $\mathbf{a}_{24}$ ($\mathbf{c}_{26}$) is necessary by checking whether both $\mathbf{c}_{13}$ and $\mathbf{d}_{13}$ ($\mathbf{a}_{25}$ and $\mathbf{b}_{25}$) pass the ST as well. When this condition is satisfied, decoding of block 2 is terminated. On the other hand, in $\mathbf{b}_{24}$ no ST is performed but after de-interleaving its output we run a ST on the content of $\mathbf{b}_{25}$, which contains de-interleaved outputs from $\mathbf{d}_{14}$ and $\mathbf{a}_{24}$. We stop decoding block 2 and $\mathbf{b}_{25}$ is no longer needed (because of our schedule and the IBP structure, $\mathbf{c}_{25}$ and $\mathbf{d}_{25}$ can not yet be verified although extrinsic information from $\mathbf{b}_{24}$ will be passed on to them) if the ST result is positive.

B. Memory management

From the above discussion, it is clear that decoding many blocks at the same time requires no small storage area for ASIC or DSP implementation. One should therefore try to make the most of the memory space available. The decoder needs space to store (I) received samples undergoing decoding, (II) extrinsic information, (III) decoded bits to be forwarded to a higher layer for further processing, and (IV) received samples awaiting decoding. The management of the last category, assuming no buffer overflow, requires only an indicator signal to forward a new block of received samples to the part of the storage area designated for category (I) that was just released due to a stopping decision.

Category (III) is needed because of the stopping time variation across blocks. Its management is straightforward and, besides, it requires much less storage space. As mentioned in Section 3.1, assigning the extrinsic values for ST-approved bits a constant large value is equivalent to using a (special) binary-valued bit to indicate which partial paths should survive in the APP decoding process. Hence the decoded bits serve the dual purposes of representing the decoder decisions and bookkeeping the survivor paths. The management of categories (I) and (II), however, needs more efforts and careful considerations.

As long as the probability of termination-defying blocks exists, practical latency consideration will force us to set an upper limit D_{max} on the number of DRs. It can be shown that an unterminated block prevents the decoder from discarding $\mathbf{Y}^3$ associated with those terminated blocks within its span. When the number of blocks that terminate at or around the D_{max}th DR is large so will be the memory required. Hardware constraint thus imposes another threshold M_{max}, the maximum affordable (allowable) memory units (MU) where an MU refers to the space for storing categories (I) and (II) associated with a block of data in the decoder. As our sole purpose is to demonstrate the critical role a memory manager plays in the VTT-APP decoder, we assume, for simplicity, that the same number of bits is used to represent the extrinsic information of a bit and the corresponding received sample. An MU is thus assumed to contain KL bits, where a K-bit word is used to store either the extrinsic information or received baseband sample associated with a transmitted bit.

Because of the stopping time variation nature of our decoder, a memory manager has to take into account both thresholds, D_{max} and M_{max} so as to optimize the performance. When a block has failed to pass the ST for D_{max} times, it will automatically be discarded and the MUs storing the corresponding categories (I) and (II) information are released accordingly. Chances are more than one block that reach the threshold D_{max} simultaneously and it is even more likely that the decoder runs out of MUs before a block reaches the threshold D_{max}. For both cases, one should then give up decoding one or some of the unterminated blocks. It is both reasonable and intuitively-appearing to terminate the most ancient block, i.e., the one which has failed the ST most often. To distinguish from the regular and ESDs described in subsection 2.1, we refer to these memory shortage induced stopping as forced early stopping.

Fig. 4 (b) shows a finite-memory IBPTC decoding procedure for one phase of an ADU. The procedure involves APP decoding, interleaving, deinterleaving, regular and ESDs, and memory check and release. The last two operations are collectively called the memory management scheme which is responsible for verifying if there is enough memory during the decoding process and make a proper memory-release decision if there is not enough storage space. As there is no computation involved at all, the complexity is moderate at most.

Denote by M_F, M_d and M_R, the numbers of free (unused) MUs, ADUs, and the required MUs for storing one received block. It follows that $M_R = x$ for a rate $R = 1/x$ turbo code. The decoder is initialized with $M_F = M_{max}$. An ADU begins a phase by checking if $M_F < M_R$ (Box 2) where the additional MU is for storing extrinsic information. If M_F meets the condition, the memory manager determines which block is to be discarded, makes a forced ESDs, and releases the related storage space (Box 4). Otherwise, the decoder moves the received samples of the new block from where they were saved (in the buffer area) to the corresponding category (I)

MUs (Box 3).

Deciding which block is to be given up is simple and clear since our decoding schedule allows only a single most ancient block in its left-most active column at any time. When a forced ESD is made the ADU makes hard decisions on the block to be discarded and releases the related categories (I) and (II) MUs. As the discarded block is always the most ancient block and our decoding schedule is such that all blocks to its left must have been terminated for one reason or another, we are left with the problem of dealing with the S adjacent blocks to its right if they have not been terminated. At least two alternatives exist for solving this problem. The first solution, which leads to better performance at the cost of higher complexity, is to interleave or de-interleave the extrinsic values for use in decoding the S blocks to its right without further updates. The second one is to make hard-decisions (stop any further decoding) on all S blocks within its (right) span, releasing their category (I) MUs while keeping their category (II) MUs for use in decoding other related blocks.

At beginning of each DR, we ask the decoder whether the scheduled DR is needed (Box 5). Unless the decoder has been notified to by-pass the ensuing DR, we still have to ask if the space for storing the extrinsic information of the coming DR is available. When such a space is not available ($M_F = 0$) the decoder has to find room for the next DR by discarding the most ancient unterminated block and following the memory release procedure described above (Box 7). The operations in Box 9 include those described in the last paragraph of subsection 2.1. When a regular or extended stopping decision is made, the memory manager releases the corresponding category (II) and parts of category (I) memory (Box 10), memory release procedure 1, and notifies the decoder that further decodings on these blocks are no longer necessary.

C. Re-transmission Protocols

Another advantage of the proposed IBPTC is that it renders flexible and efficient ARQ options. It can be used for both Type-I and Type-II ARQ protocols [9]. We do not intend to pursue detailed investigation on the ARQ applications of IBPTCs here. The sole purpose of this subsection is to describe concrete candidate application examples of the proposed coding scheme.

Note that our coding scheme requires the partition of the data stream into blocks of equal length with each block CRC-encoded and then IBPTC-encoded. A data packet can also be formatted in a similar manner so that it consists of many blocks. At the receiving end, the receiver employs a VTT-APP decoder with a given D_{max}. If after D_{max} DRs (or if the decoder runs out of storage space), some blocks still fail to pass the ST, the receiver then sends a re-transmission request to the transmitter. The decoding of these failed blocks can be continued by Chase-combining the re-transmitted blocks with the original blocks, and with the aid of the neighboring ST-proven blocks (which is not feasible for a similar CTC-coded protocol). Such an ARQ protocol is similar to the so-called type-II hybrid ARQ that requires retransmission of incremental redundancy only. We can also limit the retransmission part to be those less reliable bits within those failed blocks (which will increase the feedback channel bandwidth requirement). Another candidate protocol is to insert the failed blocks in a new packet and re-encode these blocks and their post-IBP-interleaved blocks so that they are decoded, along with other new blocks, by the receiver.

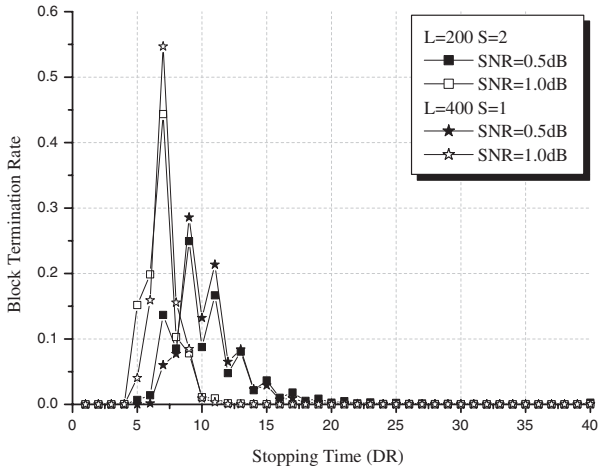

Fig. 6. Block termination rate distribution of the stopping test T3.3.

Note that both protocols can also be incorporated with a CTC. But, as has been explained in Section 2, the interleaving size of the CTC-coded system is equivalent to the block length, which is only a fraction of the packet length, will be much smaller than that of its IBPTC counterpart. The second protocol is not a conventional ARQ protocol, for, with IBP, it has the flavor of Luby's LT codes [10]. By contrast, without IBP, using a CTC in such a protocol is the same as a conventional ARQ that retransmits the failed block(s).

Shown in Fig. 6 is the block termination rate (i.e., the percentage of time a block passes the ST at a given DR) distribution as a function of the stopping time for different SNRs and block sizes if T3.3 is used as the ST. We find that, at SNR = 1 dB and 0.5 dB, the cumulated block decoding failure rate (i.e., the probability a block fails to pass the T3.3 test after D_{max} DRs) for $D_{max} = 16$ (8 iterations), $L = 400, S = 1$ is about 4.5×10^{-4} and 2.1×10^{-2}, respectively. This failure rate can be used to estimate the retransmission probability and is related to the throughput and latency computation of the proposed IBPTC-based ARQ protocols. Nevertheless, a detailed treatment on the throughput/latency performance and fair comparison with CTC-based protocols are beyond the scope of this paper.

VI. Numerical Examples

The simulation results reported in this section is based on the following assumptions and parameters. The component code of the rate=1/3 TC, $G(D) = [1, 1+D^2+D^3/1+D+D^3]$, and the CRC-8(= "110011011") code used are the same as those specified in the 3GPP standard [11] except that the component code is tail-biting [12] encoded. The APP decoder uses the Log-MAP algorithm and the IBP algorithm of [2]

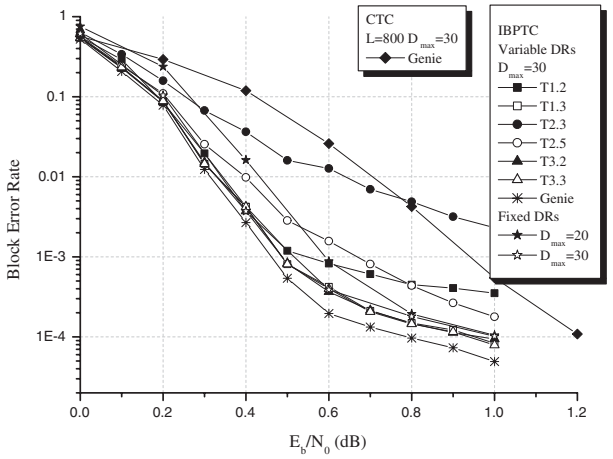

Fig. 7. Bit error rate performance of various stopping tests; no memory constraint; $D_{max} = 30$ DRs.

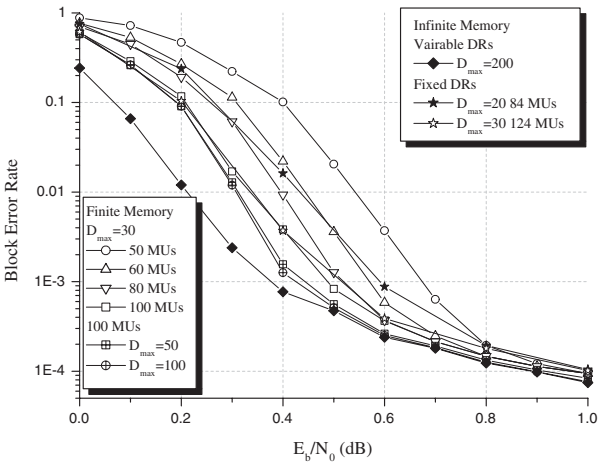

Fig. 9. The effect of memory constraint and management on the bit error rate performance. Curves labelled with infinite memory are obtained by assuming no memory constraint; "fixed DRs" implies that no early-stopping test is involved.

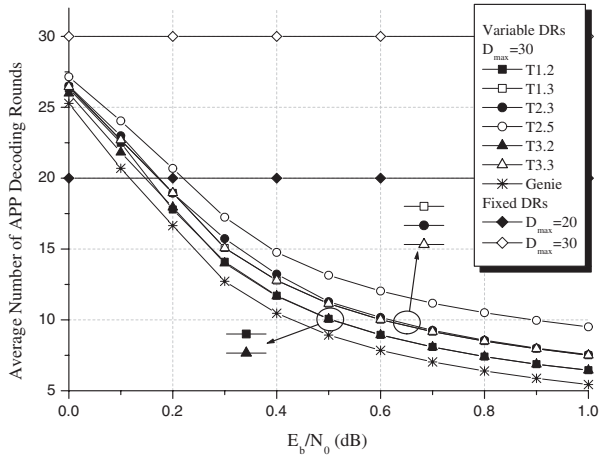

Fig. 8. Average APP DR performance of various stopping tests; $D_{max} = 30$ DRs, no memory constraint.

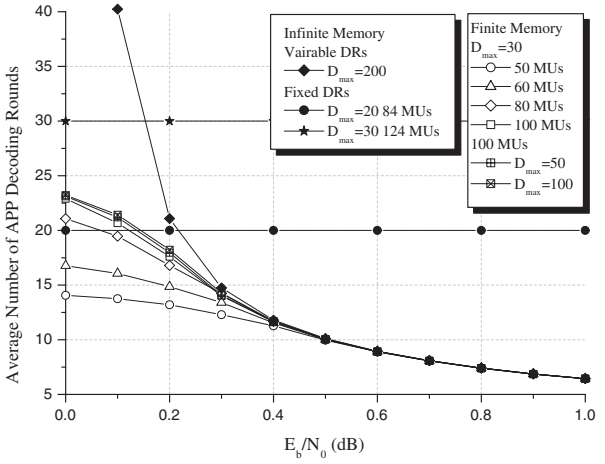

Fig. 10. Average APP DR performance for various decoding schemes and conditions. Curves labelled with infinite memory are obtained by assuming no memory constraint; "fixed DRs" means no early-stopping condition is imposed.

while the interleaving length and IBP span are left as variables; $M_R = 3$ MUs and $N = 1000$ per computer run are assumed. Except for the Genie ST, our simulations do not assume perfect block termination.

The effects of various STs on the IBPTC VTT-APP decoder performance for the system with $S = 1$, $L = 400$, $D_{max} = 30$ and tail-biting encoding are shown in Figs. 7 $\sim$ 8. Multiple-round CRCST, SCST and HST are considered. For comparison, we include performance curves of the decoder using the genie ST, that with fixed 20 and 30 DRs (10 and 15 iterations) and, for reference purpose, that of the CTC with block length $L = 800$ using the genie ST with $D_{max} = 30$.

Block error rate performance improves as the number of test rounds m increases no matter which ST is used. Fig. 7 shows that T1.3 outperforms T1.2 for E_b/N_0 greater than 0.3 dB. Tests using sign-check alone, T2.3 and T2.5, are inferior to other termination criteria since, as mentioned before, the class of sign-check tests check if decoded bits converge but can not guarantee the quality of the tentative decoded vectors. Incorrect stopping decisions will spread false information to the neighboring blocks through interleaving and result in degraded performance. T1.3, T3.2, T3.3 and the one with fixed 30 DRs yield the best performance and they are almost as good as the genie ST. Using T3.2 for early stopping, the IBPTC has $0.4 \sim 0.6$ dB gain against the CTC for BER=$10^{-3} \sim 10^{-4}$ although the average decoding delay per DR for both codes are about the same.

Fig. 8 shows the average DR performance of various STs. Except for the two sign-check tests, all STs require less than 20 or 10 APP DRs (10 or 5 iterations) when E_b/N_0 is greater

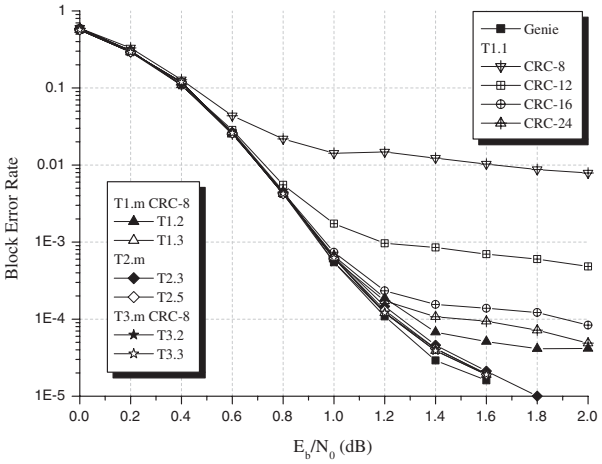

Fig. 11. Bit error rate performance of a CTC using various STs; $L = 800$ bits and $D_{max} = 30$ DRs.

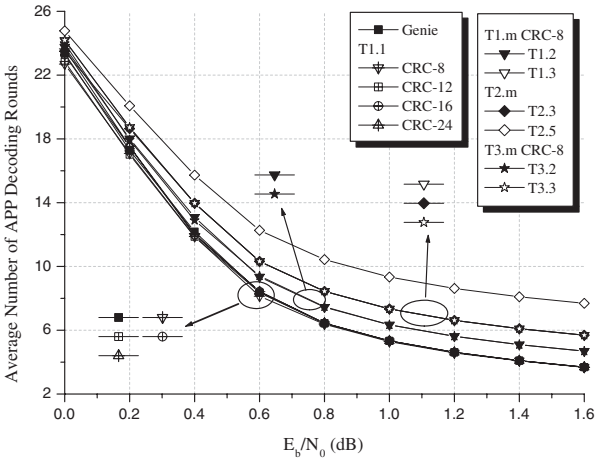

Fig. 12. The effect of various STs on the average APP DR performance of a CTC with $L = 800$ and $D_{max} = 30$ DRs.

than 0.2 or 0.6 dB . Considering both block error rate and average latency performance, we conclude that, among the STs we have examined, T3.2 is the best choice for ESDs.

The numerical results presented so far assume no memory constraint. Figs. 9 and 10 reveal the impact of finite memory size for the system that employs a T3.2-aided VTT-APP decoder and the memory management algorithm of the previous section with block length $L = 400$, interleaving span $S = 1$ and $M_d = 1$. Fig. 9 shows block error rate performance for different memory constraints. For convenience of comparison, we also present three cases without memory constraint, one with $D_{max} = 200$, the other two with fixed DRs. It is reasonable to find that larger memory sizes give better performance. At higher $E_b/N_0 (> 0.8$ dB), all performance curves converge to the same one since all VTT-APP decoders finish decoding after only a few DRs (see also Fig. 10) and memory size is no longer a problem. The fact that the cases $D_{max} = 100$ with 100 MUs, and $D_{max} = 30$ with 100 MUs give almost identical performance indicates that increasing D_{max} beyond a certain number (30 in this case) can not improve block error rate performance and the memory size becomes the dominant factor. Performance for the decoder with $D_{max} = 200$ and no memory constraint (it can be shown that 804 MUs is sufficient for this case, which is at least eight times larger than that required by other decoders) is clearly better than the other decoders when $E_b/N_0 < 0.6$ dB but this edge is gradually diminished after 0.6 dB.

The average DR performance is given in Fig. 10. For $E_b/N_0 \geq 0.5$ dB, all VTT-APP decoders need less than or equal to 10 DRs (5 iterations). But when $E_b/N_0 < 0.3$ dB, the performance curves are distinctly different–if we do not impose a memory constraint, the average DR will increase significantly as E_b/N_0 decreases. Most of the computation effort will be wasted, so is the memory. In other words, at the low E_b/N_0 region, ST can not offer early stopping decision. Imposing a memory constraint and invoking a proper memory management algorithm provide a solution that forces early stoppings, saving computing power and memory at the cost of a small performance loss. Finally, we find that, comparing with our proposed schemes, the two decoders with fixed DRs (20 and 30) usually need much more memory and DRs.

The effectiveness of various STs on the performance of a CTC with $L = 800$ are shown in Fig. 11 and Fig. 12 where $D_{max} = 30$ DRs and tail-biting encoding are assumed. The performance of T1.1 with CRC-24 is worse than those of T1.2 and T3.2 with CRC-8. Using CRC-8, T2.3 provides error rate performance similar to that of T3.2 but at the cost of one more DR. Both tests yield performance very close to that of the genie ST. In summary, these two figures show that (i) the proposed MRSTs can also be used in CTC-coded systems and (ii) using a proper MRST has the benefits of reduced CRC overhead and DRs (decoding latency) without compromising the performance. The latter conclusion implies that a multiple-round test with a short CRC code is better than a single-round test with a much longer CRC code. Of course, the same advantages are shared by IBPTC-coded systems as well.

VII. Conclusion

We present a powerful IBPTC-based coding scheme and propose the associated decoder architecture and algorithms for high speed communications. Using an MRST and a dynamic memory management scheme, our decoder yields performance achievable by a conventional turbo coded system with higher E_b/N_0 and much larger average decoding latency. The highly reliable STs require only a short CRC code and binary sign checks. The memory management scheme makes efficient use of the storage space while maintaining low average decoding latency even at low SNR region. The decoder structure is such that expanding the decoder memory size increases the dynamic range of the memory manager or the interleaving size. Both lead to improved performance.

The new coding scheme offers a variety of design options that are not available to CTCs. It also provides increased degrees of freedom for the same design option. For the same

number of ADUs, much more flexible decoding schedules are available. Its decoding is amenable for highly dynamic decoding schedules that are both distributive and cooperative: sharing all modularized decoding resources-the ADUs, interleavers/deinterleavers, memory-while passing information amongst component decoders.

System performance can be improved by using a proper decoding schedule, increasing the block size, the IBP period, the number of decoding iterations, the memory space, and the number of blocks involved in decoding. The design allows tradeoffs amongst performance, latency, computing and hardware complexities, e.g., the block size can be traded for other parameters without performance loss and lower memory requirement and higher degrees of parallelism (including memory access) are feasible by shortening the block size and using more flexible decoding resource management scheme. Multiple data sequences can be decoded in parallel and throughput is limited only by the degree of parallelism bestowed in the design. Finally, we want to remark that, since any existing block-wise interleaver can be regarded as an IBPI with $S = 1$, the associated APP decoders are reusable and, in a sense, our proposal is backward compatible, which makes the evolution from existing standard CTCs easy and natural.

REFERENCES

[1] Y.-X. Zheng and Y. T. Su, "A new interleaver design and its application to turbo codes," in *Proc. VTC2002Fall*, vol. 3, pp. 1437-1441, Sept. 2002.

[2] Y.-X. Zheng and Y. T. Su, "On inter-block permutation and turbo codes," in *Proc. International Symp. Turbo Codes and Related Topics*, Sept. 2003.

[3] Y.-X. Zheng and Y. T. Su, "On inter-block permuted turbo codes," http://arxiv.org/abs/cs.IT/0602020.

[4] J. Hagenauer, E. Offer, and L. Papke, "Iterative decoding of binary block and convolutional codes," *IEEE Trans. Inf. Theory*, vol. 42, no. 2, pp. 429-445, Mar. 1996.

[5] R. Y. Shao, S. Lin, and M. P. C. Fossorier, "Two simple stopping criteria for turbo decoding," *IEEE Trans. Commun.*, vol. 47, no. 8, pp. 1117-1120, Aug. 1999.

[6] A. Matache, S. Dolinar, and F. Pollara, "Stopping rules for turbo decoders," *TMO Progress Report 42-142*, 15 Aug. 2000.

[7] O. Y. Takeshita, "On maximum contention-free interleavers and permutation polynomials over integer rings," *IEEE Trans. Inf. Theory*, vol. 52, no. 3, pp. 1249-1253, Mar. 2006.

[8] D. Agrawal and A. Vardy, "The turbo decoding algorithm and its phase trajectories," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 699-722, Feb. 2001.

[9] S. B. Wicker, *Error Control Systems for Disgital Communication and Storage 2nd.* Prentice Hall International, Inc., 1995.

[10] M. Luby, "LT codes," in *Proc. 43rd Annual IEEE Symp. Foundations of Computer Science*, pp. 271-280, 2002.

[11] *TS 25.222 V3.1.1 multiplexing and channel coding (TDD),* 3GPP TSG RAN WG1, Dec. 1999.

[12] C. Weiss, C. Bettstetter, and S. Riedel, "Code construction and decoding of parallel concatenated tail-biting codes," *IEEE Trans. Inf. Theory*, vol. 47, no. 1, pp. 366-386, Jan. 2001.

Yan-Xiu Zheng was born in Taipei, Taiwan. He received B.S. degree in Electrical Engineering from National Tsing Hua University, Hsinchu, Taiwan in 1997 and the M.S. and Ph.D. degrees in Communications Engineering from the Department of Communications Engineering, National Chiao Tung University, Hsinchu, Taiwan, in 1999 and 2007, respectively. He was a visiting student at the Institute for Communications Engineering (LNT), TU München, Germany, during 2002-2003. In January 2006, he joined the Integrated B3G Project of the Information and Communications Laboratories, Industrial Technology Research Institute (ITRI), Chutung, Taiwan. His research interests include communication theory and coding theory.

Yu T. Su received the B.S. and Ph.D. degrees in electrical engineering from Tatung Institute of Technology, Taipei, Taiwan and the University of Southern California, Los Angeles, USA, in 1974 and 1983, respectively. From 1983 to 1989, he was with LinCom Corporation, Los Angeles, USA, where he was a Corporate Scientist involved in the design of various measurement and digital satellite communication systems. Since September 1989, he has been with the National Chiao Tung University, Hsinchu, Taiwan, where he is Associative Dean of the College of Electrical and Computer Engineering and was the Head of the Communications Engineering Department from 2001 to 2003. He is also affiliated with the Microelectronic and Information Systems Research Center of the same university and served as a Deputy Director from 1997 to 2000. In 2005, he was appointed as the Area Coordinator of National Science Council's Telecommunications Programme. His main research interests include communication theory and statistical signal processing.