Adaptive Computation Control of Variable Complexity Fano Decoders

W. David Pan, *Senior Member, IEEE*, and Antonio Ortega, *Fellow, IEEE*

***Abstract*—We propose a novel computation control method for variable-complexity Fano decoders with buffers. Our method substantially lowers the rates of data block loss associated with conventional Fano decoders. For reasonably large buffer sizes, our method outperforms Layland's buffer management scheme with block loss rates close to the theoretical lower bound.**

***Index Terms*—Buffers, computational complexity, convolutional codes, Fano decoder.**

I. INTRODUCTION

VARIABLE-COMPLEXITY (VC) algorithms may be beneficial in mobile communications systems, since their computational complexity can be varied (usually as a tradeoff with certain performance metrics) according to the changing needs, thereby enabling variable power consumption of the system. In this work, we investigate the Fano decoder, which belongs to a family of variable-complexity (VC) channel decoders called *sequential* decoders [1][2]. Refer to [3] (pp. 620) for a detailed description of the Fano algorithm. The variability of the complexity of the Fano algorithm as a function of the channel SNR has been exploited in the past, for example, in hybrid ARQ schemes [4]. In a typical Fano decoding system, however, buffer is often required to accommodate the variable (random) decoding delays of the variable-complexity decoder. The decoder may need to spend more time decoding noisier data, thereby causing the buffer to fill up quickly. If we can force the decoder to operate in a "fast" mode, then we can avoid data being dropped due to buffer overflow. We refer to the practice of regulating the level of computation efforts according to the changing needs as *Adaptive Computation Control* (ACC). We seek to design an adaptive computation control (ACC) policy for the Fano decoder with buffers, such that if the buffer occupancy is high, the ACC will instruct the decoder to run sufficiently fast in order to avoid buffer overflow; otherwise, it will slow down the decoder so that finer decoding can be achieved (Fig. 1).

There has been extensive research on buffer control techniques in the source coding literature [5][6]. However, to the best of our knowledge, virtually no computation control has been used for variable-complexity channel decoding algorithms. In a relevant work [7], Layland proposed a buffer-management strategy called *feedback-queueing memory management* (FBQM), for the purpose of reducing the block erasure (dropping) probability of a sequential decoder with a finite buffer size. In this paper, we present a computation control algorithm and demonstrate that the proposed computation control algorithm not only achieves lower data block loss rates than the conventional fano decoders, but also requires less buffer space than Layland's FBQM scheme for the same block loss rate.

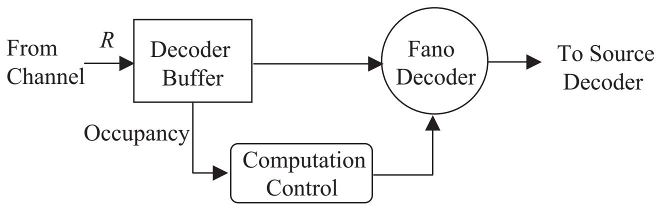

Fig. 1. Computation control of a Fano decoding system, where blocks of data are input at a constant rate R (bits/sec) to the Fano decoder buffer, where they will be decoded (and thus removed) at a variable rate since the decoding complexity is variable.

The remainder of the paper is organized as follows. Section II establishes the conditions leading to block losses, based on which Section III presents the adaptive computation control (ACC) algorithm for AWGN channels, and points out the differences between the ACC scheme and Layland's FBQM scheme. Simulation results are discussed in Section IV. The paper is concluded in Section V.

II. DECODING COMPLEXITY AND BIT ERRORS

We assume that concatenated error-control codes are used, where the inner code is a convolutional code, and the outer code is a Reed-Solomon (RS) code. A block is considered *lost* if (i) it is dropped due to buffer overflow, or (ii) it contains too many bit errors to be correctable by the outer error-correcting code. In the following, we establish thresholds for decoding complexity and bit errors, which will be used to design computation control algorithm that seeks to minimize the overall block loss rate.

Consider a Fano decoding system with a buffer size of B_{max} (in bits) as illustrated in Fig. 1. The Fano processor runs at a clock frequency of f Hz. Decoding a given block takes c cycles (note that c can vary from block to block). During the decoding time $T_d = (c/f)$, one block (L bits) will be decoded and then removed from the decoder buffer by the processor, while $(T_d \times R)$ bits will be fed into the buffer from the channel. To avoid overflow, the buffer needs to have enough space to store all the incoming bits while decoding the current block.

Paper approved by A. K. Khandani, the Editor for Coding and Information Theory of the IEEE Communications Society. Manuscript received February 9, 2007; revised February 1, 2008.

This paper was presented in part at IEEE GlobeCom 2001, San Antonio, TX, November 2001.

W. D. Pan is with the Department of Electrical and Computer Engineering, University of Alabama in Huntsville, Huntsville, AL 35899 USA (e-mail: dwpan@ece.uah.edu).

A. Ortega is with the Signal and Image Processing Institute, Department of Electrical Engineering, University of Southern California, Los Angeles, CA 90089-2564 USA (e-mail: antonio.ortega@sipi.usc.edu).

Digital Object Identifier 10.1109/TCOMM.2009.06.070582

Given the buffer occupancy $O(t)$ at time t, if $T_d \times R - L + O(t) \leq B_{max}$, then there exists a threshold C_0 as given in (1), such that if the processor is to decode the block with complexity $c \leq C_0$, then no buffer overflow will occur.

$$C_0 = [B_{max} + L - O(t)] \left(\frac{f}{R}\right). \tag{1}$$

Since $O(t) \leq B_{max}$, we have

$$C_0 \geq C_0^{min} = \frac{fL}{R}, \tag{2}$$

where C_0^{min} is the lower bound of the thresholds. Since decoding a certain block may require more than C_0 cycles (because c is variable), there exists a family of complexity thresholds C_m as given in (3), where m is a non-negative integer, such that if the decoding complexity $c < C_m$, then at most m incoming blocks will be lost due to buffer overflow:

$$C_m = [B_{max} + (m+1)L - O(t)] \left(\frac{f}{R}\right). \tag{3}$$

From (1) and (3), we have

$$C_m = C_0 + m\left(\frac{fL}{R}\right) = C_0 + mC_0^{min}, \tag{4}$$

which relates C_m ($m > 0$) to C_0.

We further illustrate the relation between C_0 and the *normalized* buffer occupancy (i.e., the degree of fullness), $F(t) = O(t)/B_{max}$, which is a system parameter directly measurable. Assume that the buffer can hold at most M data blocks, i.e., the buffer size is $B_{max} = M \times L$, where M is a positive integer. Then from (1) and (2), we have

$$C_0 = M\left[1 + \frac{1}{M} - F(t)\right] C_0^{min}, \tag{5}$$

which gives

$$F(t) = 1 - \frac{1}{M}\left(\frac{C_0}{C_0^{min}} - 1\right). \tag{6}$$

As will be discussed in Section III, (6) provides a useful mapping from a given C_0 to $F(t)$. As shown in (7), C_0 can also be related to $Q(t) = [B_{max} - O(t)]/L$, which denotes the absolute number of vacant blocks in the buffer.

$$Q(t) = \frac{C_0}{C_0^{min}} - 1. \tag{7}$$

On the other hand, if the number of error bits b in a decoded block exceeds a certain bit error threshold E_t, then the block contains uncorrectable bit errors and is declared a lost block. E_t is determined by the error correcting capability of the outer code in use.

III. The Computation Control Algorithm

We use Δ as a control parameter, which is chosen from a discrete set of n admissible values, $\Delta_s = \{\Delta_1, \Delta_2, ..., \Delta_n\}$. Our goal is to find the optimal control function $\Delta = f(O(t))$, which allows the decoder to choose one element from the set Δ_s, so as to minimize the overall *Probability of Block Loss* (PBL) $= \lim_{t\to\infty} \frac{[D(t)+U(t)]\times L}{N(t)}$, where L denotes the number of bits in a block, $D(t)$ represents the number of blocks dropped due to buffer overflow, $U(t)$ represents the number of decoded blocks having excessive number of bit errors that are uncorrectable by the outer code, and $N(t)$ is the total number of bits that have come into the buffer up to time t (obviously, $N(t) = R \cdot t$).

Given the joint distribution of decoding complexity and bit errors, $P(b, c|\Delta_i)$, which can be obtained empirically [8], we can determine the probability of m blocks being dropped due to buffer overflow as

$$P_m = \sum_{c\in(C_{m-1},C_m]} \sum_{All\ b} P(b, c|\Delta_i), \tag{8}$$

where C_m ($m \in [1, M]$) are thresholds of decoding complexities that can be determined by using (4) in Section II. Therefore, the *expected* number of blocks dropped due to buffer overflow is $\sum_{m=1}^{M}(m \cdot P_m)$, where M is such that $C_{max} \in (C_{M-1}, C_M]$, with C_{max} being the highest possible decoding complexity for a block. On the other hand, $\sum_{b>E_t}\sum_{All\ c} P(b, c|\Delta_i)$ gives the probability of the current block being decoded in error, as it computes the probability mass in the region where $b > E_t$. By combining these two types of lost blocks, we can then select the Δ^* (from the set of Δ values, Δ_s) that minimizes the *average* number of lost blocks:

$$\Delta^* = \arg\min_{\Delta_i\in\Delta_s} \left\{\sum_{m=1}^{M}(m \cdot P_m) + \sum_{b>E_t}\sum_{All\ c} P(b, c|\Delta_i)\right\}. \tag{9}$$

A sequence of Δ^* values selected based on (9) constitute a control policy that ensures that each block is decoded with an amount of computation, which *collectively* will cause the smallest number of blocks to be lost, after a set of blocks have been decoded. Although such a control policy does not permit lookahead and thus is *greedy*, the best one can do in real-time channel conditions is to make the decoding decision based on the buffer state at a give time point and choose the best Δ value according to (9). Therefore, we are not attempting to optimize the decoding performance of the very next block, but rather, we seek to minimize the average number of lost blocks, which is a global metric based on a sequence of blocks.

We can construct a lookup table that stores the (C_0, Δ^*) pairs given by (9). Note that in a practical system, more than one lookup table may be needed to account for different channel conditions, since the joint distributions $P(b, c|\Delta_i)$ in (9) vary with the channel SNR. For a certain channel SNR, a corresponding lookup table can be constructed off-line based on the empirical complexity / BER statistics collected under that particular channel SNR during the training stage. However, should there be a mismatch between the actual channel condition and the channel characteristics assumed in the design, the control performance will suffer. Under such circumstances, some SNR estimation techniques for AWGN channels [9] could be employed, so that the control can be switched among multiple lookup tables that cover the SNR range of interest.

Given the buffer size M (blocks), we can use (6) to construct another lookup table that maps the normalized buffer occupancy $F(t)$ to Δ^* (see Table I for an example). Thus the computation control algorithm proceeds as follows:

- *(Step 1) Before decoding the current block in the buffer, measure the current normalized buffer occupancy $F(t)$.*
- *(Step 2) Find the Δ^* corresponding to $F(t)$ using table lookup.*
- *(Step 3) Decode the current block by using Δ^*.*
- *(Step 4) Go back to Step 1 to decode the next block.*

In [7], it was shown by Layland that a buffer-management strategy for variable-complexity sequential decoders, known as *feedback-queueing memory management* (FBQM) scheme, was capable of achieving a block erasure probability very close to the theoretical lower bound. For a detailed description of FBQM scheme, refer to [7]. Here we point out several major differences between our adaptive computation control (ACC) scheme and the FBQM scheme. First of all, unlike the FBQM scheme, the ACC scheme uses a linearly managed buffer. Since data blocks are decoded in a first-in-first-out fashion, there is no need for block reordering after decoding. Second, the ACC scheme can adaptively vary the amount of computation on noisy blocks by choosing a different Δ value, in order to avoid blocks being dropped due to buffer overflow. In contrast, the FBQM scheme assumes that the same Δ value is used for each block. Thus the decoding complexity of a certain block is considered fixed. Furthermore, a second buffer is required for the FBQM scheme to reorder (unscramble) the decoded blocks that are released from the decoder buffer. A decoded block cannot leave the reordering buffer until all other blocks that arrived at the decoder buffer prior to this block have departed from the reordering buffer. The total buffer space required is then increased due to the need for the unscrambling buffer, which should be large enough to accommodate all the decoded blocks emitted from the decoder buffer. The size of the unscrambling buffer can be determined through simulations.

IV. Simulation Results and Discussions

In the simulations, BPSK is used for transmission. Encoded data are transmitted over an AWGN channel, and hard-decision decoding is used as an example, without loss of generality. For the inner code, the following $(2,1,7)$ non-systematic convolutional code is used in the simulation with generator polynomials $G^{(0)}(D) = 1 + D + D^2 + D^5 + D^7$, $G^{(1)}(D) = 1+D^3+D^4+D^5+D^6+D^7$. The best Δ value for decoding a block is selected from a small set $\Delta_s = \{1, 5, 13\}$, whose elements were determined empirically to provide the desirable tradeoffs between the decoding complexity and the associated bit error rates [8]. In practice, we can certainly consider more candidate Δ values to achieve finer granularity in computation control, albeit at the expense of increased complexity in generating the lookup table.

Fig. 2(a) shows the relation between buffer sizes and the probability of block loss (PBL) between the following three decoding schemes (the AWGN channel has a SNR = 4 dB): (i) the conventional linear buffer scheme with a fixed Δ used for decoding of all blocks; (ii) the adaptive computation control based on the algorithm given in Section III, where the lookup table used is shown in Table I. The PDF's $P(b, c|\Delta_i)$ used to generate the lookup table are estimated from empirical traces consisting of 10^5 (b, c) pairs, obtained from a decoding system

TABLE I
The lookup table (Normalized buffer occupancy versus Δ^*). NA: not applicable.

Buffer Size (blocks)	$\Delta^* = 1$	$\Delta^* = 5$	$\Delta^* = 13$
1	NA	0	1
5	NA	$[0, 4/5]$	1
10	NA	$[0, 9/10]$	1
20	NA	$[0, 19/10]$	1
25	$[0, 3/25]$	$[4/25, 24/25]$	1
30	$[0, 8/30]$	$[9/30, 29/30]$	1
40	$[0, 17/40]$	$[18/40, 39/40]$	1
50	$[0, 27/50]$	$[28/50, 49/50]$	1

with $L = 128$ bits, $f = 10^8$ Hz, $R = 10^6$ bits/sec, and $E_t = 8$ bits. (iii) Layland's FBQM scheme, which uses a fixed Δ and requires a reordering/unscrambling buffer in addition to the block decoding buffer.

As shown in Table I, the buffer sizes can be roughly divided into three ranges: (i) When the buffer size M (blocks) is small, i.e., $M \in [1, 20]$, $\Delta = 1$ is never chosen by the ACC scheme, which suggests that larger Δ values could offer better tradeoffs between buffer overflow and bit errors. The PBL of the ACC scheme goes far below that of the linear buffer scheme with a fixed $\Delta = 1$. For the extreme case where the buffer can hold only one block, $\Delta = 13$ is almost always chosen by the ACC scheme due to the buffer being full most of the time. For all other buffer sizes in this range, $\Delta = 5$ is always selected by the ACC scheme when the buffer is not full, so that the ACC performs very closely to the $\Delta = 5$ scheme and better than the $\Delta = 13$ scheme. (ii) As the buffer gets larger, i.e., $M \in [25, 40]$, the benefit of using the ACC becomes much more pronounced. For example, as shown in Table I, if the buffer size is 30 blocks, we should choose $\Delta = 1$ if the buffer has less than 9 blocks in it. When the normalized buffer occupancy rises above 30%, $\Delta = 5$ will be selected. When the buffer is full, $\Delta = 13$ (the largest possible value) will be chosen to minimize the probability of overflow. In this range of buffer sizes, the ACC scheme outperforms all fixed Δ schemes, while achieving a PBL significantly lower than that of the scheme with $\Delta = 13$. (iii) As the buffer size increases ($M \geq 40$ blocks), the probability of buffer overflow decreases. Hence the ACC scheme tends to choose $\Delta = 1$ over a wider range of buffer occupancies. As can be expected, if the buffer size is further increased, the ACC scheme will almost always pick $\Delta = 1$, thereby gradually degenerating into a fixed Δ scheme without control.

Furthermore, as shown in Fig. 2 (a), except for very small buffer sizes ($<$ 7 blocks), the ACC scheme achieves lower PBL than the FBQM scheme regardless of the Δ value. For very small decoder buffer, the impact of using an extra reordering buffer would be insignificant. The reason is that the number of decoded blocks that might become out-of-order depends heavily on the size of the decoder buffer. Therefore, if reordering is taken account, the FBQM would be beneficial only for small decoder buffer sizes. The same observation can also be made by comparing the average latency experienced by any decoded block passing through a decoding system. As can be seen in Fig. 2 (b), for a given Δ, the linearly managed (FIFO) buffer incurs a smaller average delay than the FBQM-unscrambler combination whenever the buffer size

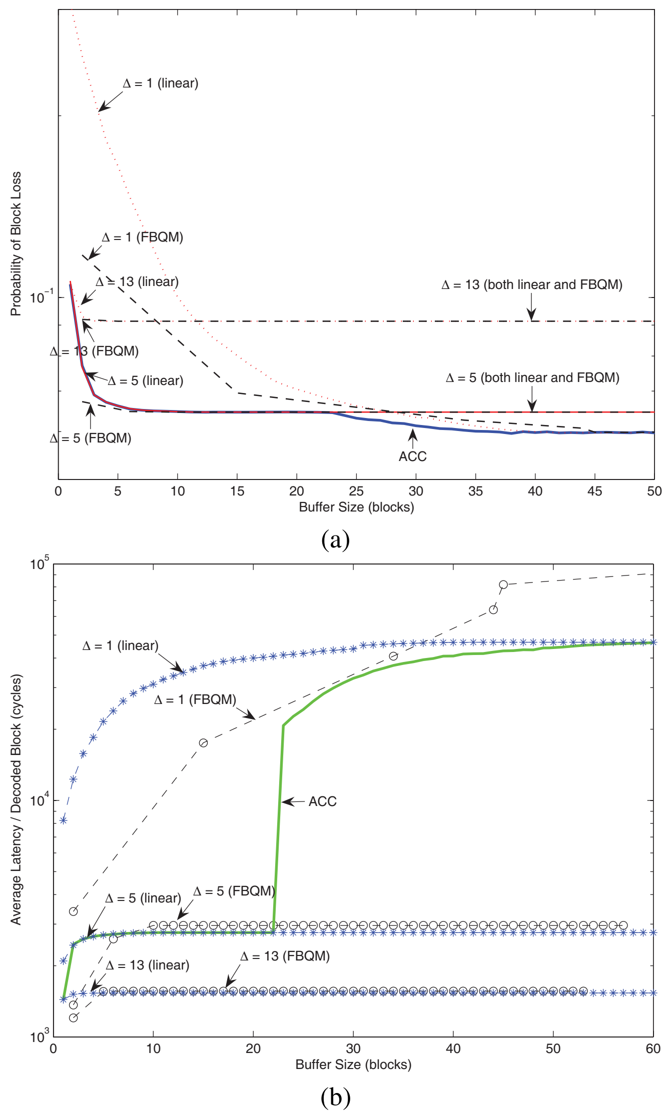

Fig. 2. (a) Probability of block loss as a function of buffer size for three schemes: linear buffer scheme (with fixed Δ's), linear buffer with adaptive computation control (ACC) by varing Δ's, and Layland's FBQM scheme with a reordering buffer), and (b) Average latency (in clock cycles) per decoded block.

is greater than a certain threshold. In the literature, sufficiently large buffer sizes are generally recommended for avoiding buffer overflow in sequential decoders [10][11]. In practical implementations of Fano decoders, the actual buffer size in use is usually selectable from a range of values. For example, buffer sizes of up to 16 blocks of data have been used in the simulation of a high-speed Fano decoder for deep space communications [12], with varying probabilities of block loss due to buffer overflow.

V. Conclusions

We introduce an efficient computation control (ACC) algorithm for Fano decoders with buffers. The ACC algorithm dynamically determines the best Δ value based on the normalized buffer occupancy by consulting lookup tables, which can be constructed offline using empirically obtained statistics. Our study showed that the ACC algorithm significantly outperforms the conventional Fano decoding algorithm using a fixed Δ value. For reasonably large buffer sizes, the ACC scheme compares favorably to Layland's FBQM scheme, which is known for its theoretical lower-bound approaching performance for sequential decoders, especially after taking into account factors such as the block loss rate, decoder latency, and the complexity of the buffer control logic.

References

[1] J. M. Wozencraft and I. M. Jacobs, *Principles of Communication Engineering*. Wiley, 1965.
[2] R. M. Fano, "A heuristic discussion of probabilistic decoding," *IEEE Trans. Inform. Theory*, vol. 9, pp. 64–74, Apr. 1963.
[3] S. Lin and D. J. Costello, Jr., *Error Control Coding: Fundamentals and Applications*, 2nd ed. Upper Saddle River, NJ: Pearson Prentice Hall, 2004.
[4] S. Lin, D. J. Costello, Jr., and M. J. Miller, "Automatic-repeat-request error control schemes," *IEEE Commun. Mag.*, vol. 22, no. 12, pp. 5–17, Dec. 1984.
[5] C.-Y. Hsu, A. Ortega, and M. Khansari, "Rate control for robust video transmission over wireless channels," *IEEE J. Select. Areas Commun.*, vol. 17, pp. 756–773, May 1999.
[6] P. A. A. Assuncao and M. Ghanbari, "Buffer analysis and control in CBR video transcoding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 10, pp. 83–92, Feb 2000.
[7] J. W. Layland, "Buffer management for sequential decoding," *IEEE Trans. Commun.*, vol. 22, pp. 1685–1690, Oct. 1974.
[8] W. Pan and A. Ortega, "Buffer control for variable complexity fano decoders," in *Proc. IEEE Global Telecommun. Conf. (Globecom)*, Nov. 2001, pp. 176–180.
[9] D. R. Pauluzzi and N. C. Beaulieu, "A comparison of SNR estimation techniques for the AWGN channel," *IEEE Trans. Commun.*, vol. 48, pp. 1681–1691, Oct. 2000.
[10] C. R. Cahn, G. K. Huth, and C. R. Moore, "Simulation of sequential decoding with phase-lock demodulation," *IEEE Trans. Commun.*, vol. 21, no. 2, pp. 89–97, Feb. 1973.
[11] O. R. Jensen and E. Paaske, "Forced sequence sequential decoding: A concatenated coding system with iterated s equential inner decoding," *IEEE Trans. Commun.*, vol. 46, no. 10, pp. 1280–1291, Oct. 1998.
[12] J. W. Layland and W. A. Lushbaugh, "A flexible high-speed sequential decoder for deep space channels," *IEEE Trans. Commun. Technol.*, vol. 19, no. 5, pp. 813–1971, Oct. 1971.