Buffer Control for Variable Complexity Fano Decoders

Wendi Pan, Antonio Ortega
Integrated Media Systems Center, Department of Electrical Engineering-Systems
University of Southern California, Los Angeles, CA 90089-2564

***Abstract*— Fano sequential decoders are variable complexity convolutional decoders, which have the desirable property of operating with very low computation at high SNR. In portable mobile communications, it is often desirable to trade BER with decoder complexity/power consumption. However, the variable complexity nature of the Fano algorithm means that buffers are required for the Fano decoder due to large variations in processing delays. In this paper, we formulate the buffer control problem as one that seeks to minimize the overall probability of block loss, subject to a finite buffer size constraint. The overall probability of block loss is comprised of two terms, corresponding to loss due to excessive bit errors and decoder buffer overflow, respectively. This leads to an interesting trade-off, as faster decoding often means higher bit error rate. Based on the joint distribution of decoding complexity and BER, at each decoding stage, we find an optimal Fano decoder parameter (Δ) to minimize block loss. In addition, we propose a simple real-time table lookup algorithm that implements the Δ control policy. Simulation results demonstrate the superior performance of the proposed algorithm.**

I. Introduction

Sequential decoding was introduced in 1961 by Wozencraft as a method of maximum likelihood sequence estimation with typically lower computational complexity than the Viterbi algorithm [1]. Of several versions of sequential decoding algorithms, the Fano algorithm is generally considered to be the most practical to implement [2][3][4].

A Fano decoder explores one hypothetical data sequence at a time by locally encoding it and comparing it with the noisy encoded version that is actually received. the decoder examines the metric of a potential path. If the metric value dips below a threshold T, the decoder backs up and begins to examine other paths. If no path can be found whose metric value stays above the threshold, the threshold is then loosened ($T \leftarrow T - \Delta$) and the decoder moves forward again with a lower threshold. As long as the decoder moves forward to a node as a first visit, the threshold is tightened ($T \leftarrow T + \Delta$) to ensure no endless loop occurs and the decoder eventually reaches the end of the tree. The threshold increment Δ is a parameter that controls the tradeoff between the bit error rate (BER) and the decoder complexity, which translates to power consumptions of the decoder. An analogy can be made between Δ and the quantization parameter (step size) in image/video coding, since the quantizer controls the tradeoff between decoded image distortion and bit rate [5].

In a mobile communications system, there is a need for designs that efficiently use the battery power of the mobile communication terminal. In [6], a chip design based on the Fano algorithm achieves significantly lower energy consumption for an AWGN channel with relatively high SNR, compared to the Viterbi decoder. We can save computation (energy) if we could tolerate a degraded bit error performance. As another example, a user that is located close to a base station should be able to operate at lower power than users roaming further afield.

On the other hand, the Fano decoder has inherent non-deterministic processing delay, which necessitates buffering of data before and after the decoder in practical real-time decoding systems. Therefore we are faced with an interesting tradeoff — if the decoder runs faster by choosing a large Δ in order to avoid buffer overflow, the "coarser" decoding will cause more blocks to be decoded in error, which will be considered lost. Alternatively, if the decoder uses a small Δ in order to achieve "finer" decoding, then the decoded blocks will tend to contain few bit errors. However, the decoder will be slowed down, and the buffer will tend to fill up. If the buffer becomes full, blocks have to be dropped. Thus our objective is to design a buffer control algorithm that can minimize the average number of lost blocks under the limited buffer size constraint.

There has been extensive research on buffer control techniques in the source coding literature [7][8][9] . How-

This research has been funded in part by the NSF grant NCR-CCR-9726391, STMicroelectronics Inc, and by the Integrated Media Systems Center, a National Science Foundation Engineering Research Center, Cooperative Agreement No. EEC-9529152.

ever, to the best of our knowledge, no solution has been proposed to the specific buffer control problem under consideration. Our approach is to pick the best Δ that minimizes the probability mass corresponding to lost blocks, based on the joint distribution of the decoding complexity and the BER. Note that in this paper, we consider only changing Δ, but the clock speed of the Fano processor can also be changed in order to prevent buffer overflow. Note also that buffer sizes cannot be too long in interactive applications.

The paper is organized as follows. In Section II, we discuss the Δ control in the buffer framework. Next, the relations between the buffer occupancy, complexity and block errors are established in Section III. Section IV presents the solution to the Δ control problem. We finally describe a simple table lookup algorithm, followed by its simulation results.

II. Buffer for the Fano Decoder

The variable complexity advantage of the Fano decoder comes with the price of requiring buffers in a practical system. As shown in Fig.1, it is necessary to introduce a buffer memory to store and queue the incoming data blocks until the Fano processor can decode them.

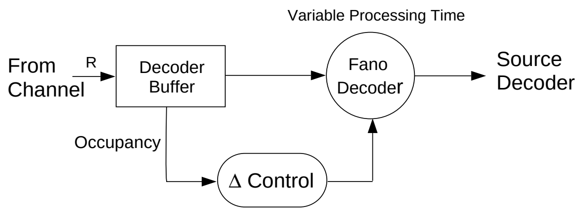
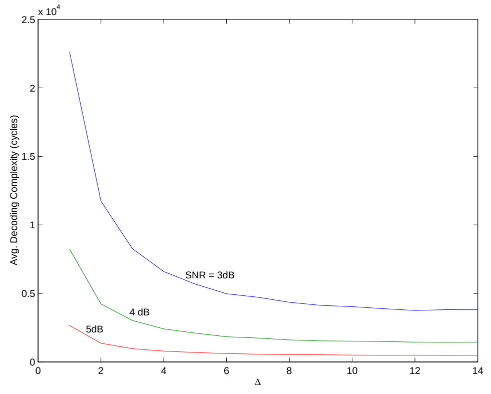


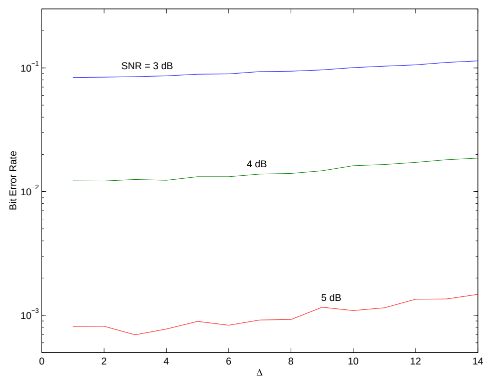
Fig. 1. Fano decoder buffer. From an AWGN channel, blocks of data are input at a constant rate R to the Fano decoder buffer, where they will be removed at a variable rate since the decoding complexity is a random variable.

In our experiments, we use concatenated error-control codes, which are prevalent in nowadays' GSM systems. The inner code is a constraint length 7, rate 1/2 convolutional code with outer code being the Reed-Solomon(RS) code with certain degree of error correction capability.

In the Fano decoder, Δ is a control parameter. As can be seen in Fig.2, if Δ is chosen to be small, a block is less likely to be decoded in error, however, the decoding complexity increases. As shown in Fig.1, data blocks keep coming into the buffer at a constant speed from the channel, while the data in the buffer are taken away from the buffer and decoded by the decoder. If the decoder is emptying the buffer slowly, some incoming blocks might be dropped due to buffer overflow. Conversely, if the Δ is chosen to be large, the decoding complexity will decrease, and the probability of overflow decreases due to faster decoding operations. Nevertheless, decoded blocks are likely to contain greater number of bits in error because of higher BER's. If a data block contains too many error bits to be corrected by the outer RS code, the entire blocks will be rendered useless for the source decoder, therefore, this block shall also be considered lost.



(a)



(b)

Fig. 2. (a) Average complexity and (b) average BER as a function of Δ and channel SNR. The complexity can be decreased at the expense of an increased BER.

Hence, our objective is to design a Δ control policy such that the overall probability of block loss is minimized, subject to the constraint of a finite buffer size.

III. Buffer Occupancy, Complexity and Block Errors

In Fig.3, at time t, the Fano processor has to fetch the next block from the buffer for decoding. Assume that the Fano processor runs at a clock frequency of f Hz, and that decoding a given block takes c cycles (note that c is a variable), then during the decoding time $T_d = (c/f)$, one block (L bits) will be removed from the decoder buffer by the processor, while $(T_d \times R)$ bits will be fed into the buffer from the channel. Given the buffer occupancy $O(t)$, and the buffer size B_{max}, we can determine a threshold complexity C_t, such that if the processor is to decode the block with a complexity $c \leq C_t$, then no buffer overflow will occur.

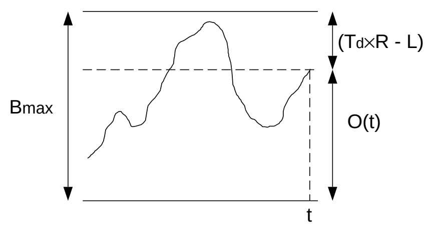


Fig. 3. Mapping of the buffer occupancy to the decoder complexity.

To avoid overflow, the buffer needs to have enough space to store all incoming bits while decoding of the current block proceeds, i.e.,

$$\frac{c}{f} \times R - L + O(t) \leq B_{max}, \quad (1)$$

or equivalently,

$$c \leq C_t := [B_{max} + L - O(t)] \times \frac{f}{R}. \quad (2)$$

Since the decoding time of the Fano processor is a random variable, decoding of a certain block may require more than C_t cycles, thus leading to a buffer overflow. We can further determine a family of complexity thresholds C_m, where m is an integer. C_m is the decoding complexity such that if the actual complexity $c < C_m$ then at most m blocks are lost due to buffer overflow. If $m = 0$, then $C_0 = C_t$, i.e., no buffer overflow. We have

$$\frac{C_m}{f} \times R - L + O(t) = B_{max} + m \times L, \quad (3)$$

and therefore

$$C_m = C_{m-1} + \delta, \quad (4)$$

where $\delta = L \times \frac{f}{R}$. δ is independent of the buffer occupancy. Note that if an incoming block gets dropped because the buffer is already full, then the entire block gets dropped. Hence, any decoding complexity that lies within the region $(C_{m-1}, C_m]$ translates to m blocks dropped (Fig.4).

In order to devise an adaptive Δ control policy, we study the joint distribution of the number of bit errors in a block b after it is decoded by the Fano decoder, and its decoding complexity c. The two-dimensional sample space (b,c) can be partitioned into four quadrants (Fig.5) by the complexity threshold C_t and the error bit number threshold E_t. Characteristics of each quadrant are summarized in Table I. For example, any sample

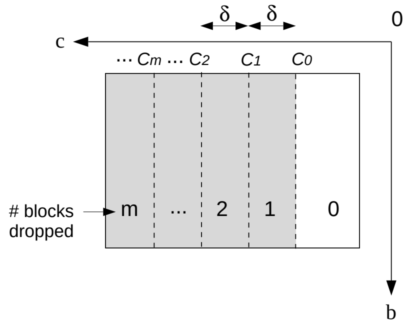


Fig. 4. A family of complexity thresholds C_i for determining the number of blocks dropped due to buffer overflow.

(b,c) in quadrant I ($c \leq C_t$, and $b \leq E_t$) corresponds to the case where the decoder runs so fast that overflow is avoided. Meanwhile, the decoded blocks, if containing any bit errors, can still be corrected by the RS code.

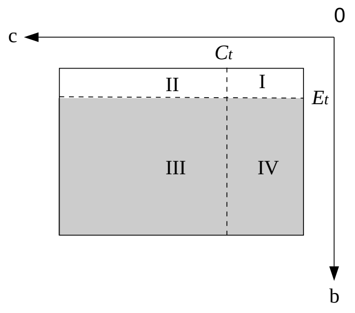


Fig. 5. Partition of the sample space (b, c). C_t is a threshold such that any complexity above C_t will cause buffer overflow. The shaded area ($b > E_t$) represents blocks that are declared in error after the Fano decoding, since the succeeding RS code cannot correct over E_t error bits in a block.

TABLE I

Quadrant	Overflow	Block Error
I	No	No
II	Yes	No
III	Yes	Yes
IV	No	Yes

Our simulation of the Fano decoder is based on the software used in a related work [6] [1]. An example of a resulting set of 2-D histograms is shown in Fig.6. The main conclusion to be drawn from the histogram is that the distributions of b and c are far from being independent. Thus, in what follows, we will use the joint distribution in our optimization.

[1]We would like to thank the authors of [6] for their source codes of Fano decoders.

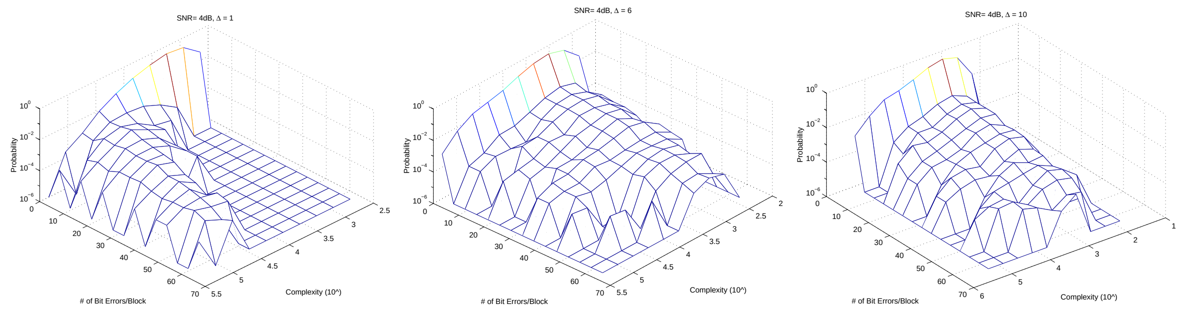


Fig. 6. Histograms of (b,c) pairs, for SNR= 4dB, and $\Delta = 1, 6$ and 10.

IV. Optimal Δ Control

A. Formulation

The adaptive channel decoder buffer control problem can be formulated as follows:

Given a certain channel SNR and a finite buffer size B_{max}, at a given point in time t, the Fano decoder is to decode a block already in the decoder buffer. The occupancy $O(t)$ of the decoder buffer is:

$$O(t) = \begin{cases} N(t) & \text{if } t \le \triangle T_d; \\ min(N(t) - R_p(t - \triangle T_d), B_{max}) & \text{if } t > \triangle T_d. \end{cases} \tag{5}$$

In Eq.(5), $N(t)$ is the number of bits input to the buffer up to time t, ΔT_d is the initial delay (for preloading the buffer with some blocks in order to avoid the underflow at the very beginning of the decoding process), and $R_p(t)$ represents the number of bits decoded by the Fano decoder.

The optimal control function $\Delta = f(O(t))$ allows the decoder to choose one element from a discrete set of n admissible values, $[\Delta_1, \Delta_2, ..., \Delta_n]$, so that the overall *Probability of Block Loss*, as in Eq.(6) is minimized.

$$PBL = \lim_{t \to \infty} \frac{[D(t) + E(t)] \times L}{N(t)}, \tag{6}$$

where L denotes the number of bits in a block, $D(t)$ represents the number of blocks dropped due to buffer overflow, and $E(t)$ represents the number of decoded blocks having excessive number of bit errors that are uncorrectable by the RS code. Obviously, $N(t) = R \cdot t$.

In order to achieve this goal, we need to select a sequence of Δ's that minimize the additive cost — *average number of lost blocks*. Thus for a given SNR, if we assume that the joint conditional probability mass function (PMF), $P(b, c|\Delta)$, is known, then we can find a Δ^* as

$$\Delta^* = \arg \min_{\Delta_i \in [\Delta_1, \Delta_n]} \left\{ \sum_{m=1}^{M} [m \times \sum_{c \in (C_{m-1}, C_m]} P(b, c|\Delta_i)] + \sum_{b > E_t} P(b, c|\Delta_i) \right\}, \tag{7}$$

where M is such that $C_{max} \in (C_{M-1}, C_M]$, with C_{max} being the largest possible decoder complexity.

Eq.(7) selects the Δ that minimizes the sum of two terms: the first term is the *expected* number of blocks dropped due to buffer overflow, whereas the second term is the probability of the current block being decoded in error.

A sequence of Δ^* based on Eq.(7) constitutes a *greedy* control policy statistically since it ensures that each block is decoded at a speed that causes the smallest number of blocks lost *on average* during the decoding of the current block. Thus this control policy is optimal at each decoding stage if the current buffer occupancy is given. Since the channel is memoryless, and Δ is coarsely quantized, our solution is a good *approximation* to the optimal control policy over the entire decoding process.

B. Algorithm

In practice, rather than computing Eq.(7) on the fly, we can pre-compute the optimal Δ^* for each decoder complexity $C_t \in [C_{min}, C_{max}]$, by substituting C_0 with C_t in Eq.(7). A lookup table T that stores the (C_t, Δ^*) pairs can be constructed in this way. Thus the buffer control algorithm can be stated as follows:

- *(Step 1) To decode a block already in the buffer at time t, use Eq.(2) to obtain the target decoding complexity C_t based on the current buffer occupancy $O(t)$.*
- *(Step 2) Find the Δ^* value to be used for decoding this block in the lookup table T. Finish decoding this*

block and repeat the first step for the decoding of the next block.

C. Simulation Results

In the buffer simulation, we investigate the relation between buffer sizes and the probability of block losses for both the fixed and adaptive Δ control algorithms. Δ is chosen from 1 to 14.

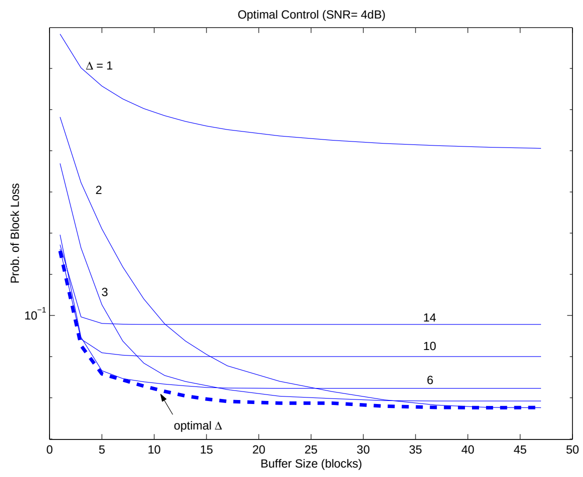


Fig. 7. Comparison with fixed Δ control policies. The thicker, dashed curve represents the proposed Δ^* control policy based on the lookup table in Fig.8.

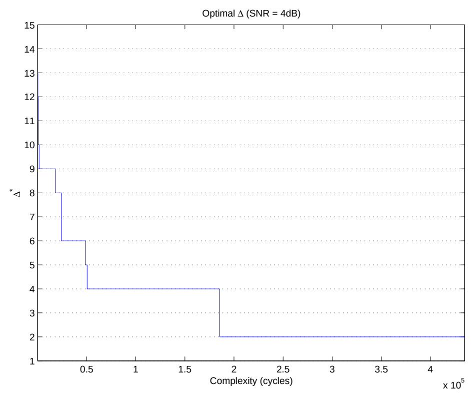


Fig. 8. The lookup table (C_t, Δ^*) for SNR= 4dB.

As can be seen in Fig.7, our adaptive control policy consistently outperforms all other fixed Δ control policies. Moreover, our policy outperforms various well-known adaptive control policies (Fig.9).

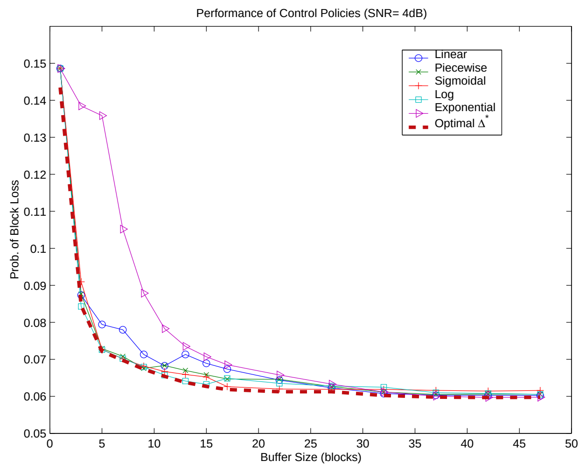


Fig. 9. Comparison with several adaptive control policies in Fig.10.

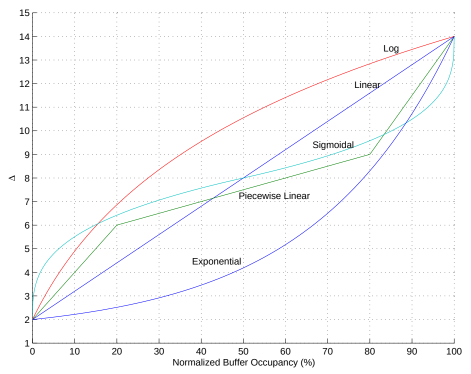


Fig. 10. Adaptive control policies.

References

[1] S. Lin and D. C. (Jr), *Error control coding : fundamentals and applications.* Prentice-Hall, 1983.

[2] R. Fano, "A heuristic discussion of probabilistic decoding," *IEEE Trans. Information Theory*, vol. IT-9, pp. 64–74, April 1963.

[3] A. Viterbi and J. Omura, *Principles of digital communication and coding.* McGraw-Hill, 1979.

[4] J. Wozencraft and I. Jacobs, *Principles of communication engineering.* Wiley, 1965.

[5] A. Ortega, K. Ramchandran, and M. Vetterli, "Optimal trellis-based buffered compression and fast approximations," *IEEE Trans. on Image Processing*, vol. 3, pp. 26–40, Jan. 1994.

[6] S. Singh, P. Thienniviboon, R. Ozdag, S. Tugsinavisute, R. Chokkalingam, P. Beerel, and K. Chugg, "Algorithm and circuit co-design for a low-power sequential decoder," in *Proc. of Asilomar Conf. on Signal, Systems and Comp.*, Oct. 1999.

[7] J. Zdepski, D. Raychaudhuri, and K. Joseph, "Statistically based buffer control policies for constant rate transmission of compressed digital video," *IEEE Trans. Comm.*, pp. 947–957, June 1991.

[8] C.-Y. Hsu, A. Ortega, and M. Khansari, "Rate control for robust video transmission over wireless channels," *IEEE Journal on Sel. Areas in Comm.*, vol. 17, pp. 756–773, May 1999.

[9] J. I. Ronda, M. Eckert, F. Jaureguizar, and N. Garcia, "Rate control and bit allocation for MPEG-4," *IEEE Trans. Circuits Syst. Video Technol.*, pp. 1243–1258, Dec. 1999.