N92-14220

Fault-Tolerant Multichannel Demultiplexer Subsystems[1]

Robert Redinbo
Department of Electrical Engineering and Computer Science
University of California
Davis, CA 95616 USA

Abstract

Fault tolerance in future processing and switching communication satellites is addressed by demonstrating new methods for detecting hardware failures in the first major subsystem, the multichannel demultiplexer. An efficient method for demultiplexing frequency slotted channels employs multirate filter banks which contain fast Fourier transform processing. All numerical processing is performed at a lower rate commensurate with the small bandwidth of each baseband channel. The integrity of the demultiplexing operations is protected by using real number convolutional codes to compute comparable parity values which detect errors at the data sample level. High-rate, systematic convolutional codes produce parity values at a much reduced rate, and protection is achieved by generating parity values in two ways and comparing them. Parity values corresponding to each output channel are generated in parallel by a subsystem, operating even slower and in parallel with the demultiplexer that is virtually identical to the original structure. These parity calculations may be time-shared with the same processing resources because they are so similar.

(1) This research was supported by NASA Lewis Research Center through grant NAG-3-1166 and the National Science Foundation through grant MIP-9002664.

Introduction

The new generation of sophisticated processing and switching communication satellites with their extensive digital processing capabilities will be more susceptible to temporary and permanent electronic failures. An overview of a typical satellite's subsystems is given in Figure 1. This paper will concentrate on the demultiplixer subsystem to demonstrate the general principles of fault tolerance needed in implementations that process multiple channels with shared resources. Each subsequent subsystem will have specialized features requiring variations on these fault tolerance techniques. The basic philosophy is still applicable though. Continuing work is addressing the other subsystems.

Data channels in communication systems are easily combined according to frequency division multiplexing (FDM). This method is particularly useful because frequency selectivity is all that is required to extract individual channels from the overall signal constellation. Many satellite communication systems employ this method of multiplexing since there is no requirement for common timing synchronization between data channels. This approach is even more appealing from a hardware implementation viewpoint because very efficient demultiplexer realizations, called polyphase multirate filter banks, are available [1-3].

The basic demultiplexing philosophy envisions a narrow band filter extracting each channel from the multiplexed signals. N multiplexed channels, each with relative bandwidth f_B, are combined into an FDM signal, and the corresponding demultiplexer employs an idealized narrow band filter with Z transform transfer function H(Z), shifted in frequency, to separate the band of frequencies corresponding to a channel. The uniform baseband filter is effectively shifted to each respective band by the scaling phasors, and the output channel only produces samples at a rate $1/N^{th}$ of the input sampling rate [4]. Generally, the uniform filter represented by H(Z) has a finite impulse response (FIR) configuration with the attendant advantage of linear phase [5].

The uniform filter banks can be realized by defining certain segments of the baseband filter's transfer function and then using the outputs from these shorter filters as simultaneous inputs to a discrete Fourier transform operation [1,4]. This approach to demultiplexing is outlined in Figure 2, where a fast Fourier transform (FFT) algorithm realizes the discrete Fourier transform. The relationship between the new shorter segmented filters $H^{(r)}(Z)$, r = 0, 1, ..., N-1, and the original baseband filter H(Z), will be summarized later. The most important feature of Figure 2 is its slower sampling rate applied BEFORE the filtering and FFT operations, permitting the digital hardware implementing these functions to operate at a data rate $1/N^{th}$ that of the input data sampling rate. Nevertheless, the input is still sampled at a suitably high rate commensurate with its wider bandwidth.

There are situations where this efficient form of demultiplexer must be highly reliable. Yet, the very efficient sharing of processing resources makes this form extremely sensitive to even simple failures which can easily contaminate many data channels simultaneously. For example, the new generation of switching and data processing satellites take advantage of these forms of demultiplexers.

There are several considerations when incorporating fault tolerance in such a demultiplexer system. The first important consideration is the detection of failures, whether they are permanent or temporary and transient. Once inaccurate performance is detected, the failed subunit must be identified and located (diagnosis). Finally, if the failures persist, the system must be reconfigured so that adequate performance is still achieved. The work presented here concentrates on the first aspect, fault detection. For, without an indication of improper operation, the other aspects of fault tolerance cannot be invoked.

An emerging alternate method of fault tolerance, termed by some Algorithm-Based Fault Tolerance (ABFT), views the algorithmic operations and the data sample flow as the important items to protect regardless of the underlying hardware realization. The first use of this technique was in protecting matrix operations [9], and there have been many other applications investigated [10-17]. Most research has been directed to protecting linear algorithms.

The fundamental approach in ABFT employs real number error-detecting codes to define parity values associated with a group of data samples. These codes can be either block or convolutional codes [18-20]. In either case, the original processing algorithm is combined with the parity generation process, generally leading to a composite, efficient, simplified parity generation algorithm that produces independent parity values which are associated with the output data. Then comparable parity values are computed directly from the original processing algorithm's output data. The respective parity values, one from each set but computed in different ways, should be identical, except possibly for some small round-off error differences since they are evaluated in two dissimilar ways. Errors are detected when the respective parity values differ significantly. This type of fault tolerance will be applied to protecting the demultiplexer.

Basics of Filter Banks

An analysis bank of filters will be examined where each of the L transfer functions $H_0(Z), H_1(Z), \ldots, H_{L-1}(Z)$ bandlimit their respective signal outputs so that each may be sampled at a rate 1/N of the input rate. This general setting is depicted in Figure 3a. The L transfer functions will be assumed FIR types, for the purposes of the exposition. Each of the L filter paths can be analyzed separately and the generic situation is isolated in Figure 3b, for further development. The Z transform quantities shown in these figures employ the two-side Z transform. Infinite limits in the summations are included in its definition below, even though only a finite number of nonzero terms appear for the FIR filter case.

The impulse response $\{h_p(m)\}$, corresponding to the transfer function H(Z), is segmented into N subsequences and the weighting operation separated into N parallel convolutions.The N parallel convolutions employ segmented impulse responses related to the original p^{th} channel impulse response in the following way.

$$y_p(r) = \sum_{v=0}^{N-1} \left[\sum_{u=-\infty}^{+\infty} h_p^{(-v)}(r-u)\, x(uN+v) \right]$$

$$= \sum_{v=0}^{N-1} \left[\sum_{u=-\infty}^{+\infty} h_p^{(v)}(a)\, x(r-u)\, N - v \right] ; \qquad (1a)$$

$$h_p^{(v)}(a) = h_p(aN+v) \; ; \quad \begin{array}{l} v = 0, 1, \ldots, N-1 \\ a = 0, \pm 1, \pm 2, \ldots \end{array} \qquad (1b)$$

where r in (1a) = 0, ±1, ±2, ... The same number of operations are performed in this approach, but each parallel self-contained path can operate at a rate reduced by factor N. The ability to use a slower processing rate in each disjoint parallel path provides a serial-to-parallel tradeoff.

Demultiplexers can be viewed as a special form of Figure 3a, wherein L = N and the N transfer functions are constrained to be related to one common baseband transfer function H(Z). It is easy to demonstrate the effects of the scaling phase sequence $\{e^{j\frac{f_s}{N}pr}\}_{r=-\infty}^{+\infty}$ is to shift the filter response [1]. Under these conditions, the output of the p^{th} filter path may be written in terms of the input weighted by segments of the impulse response and scaled by a phasor.

$$y_p(r) = \sum_{v=0}^{N-1} e^{j\frac{f_s}{N}pv} \left\{ \sum_{u=-\infty}^{+\infty} h^{(-v)}(r-u)\, x(uN+v) \right\} \quad (2)$$

The final summation over index variable v in equation (2) is equivalent to forming the p^{th} discrete Fourier transform coefficient for the N outputs of the segmented impulse response filters $\{h^{(-v)}(r)\}_{v=0}^{N-1}$. Furthermore, the only change needed to get an output for a different output, say $\{y_m(r)\}$, is to modify the scaling coefficients, $\left\{e^{j\frac{f_s}{N}mv}\right\}_{v=0}^{N-1}$, affecting the outer sum. The same segmented impulse response filters are employed, but the scaling values change. Thus Figure 2 represents the general case where an FFT form of a discrete Fourier transform is applied to the respective outputs of the segmented filter functions. All N channel outputs of the demultiplexer are obtained simultaneously.

Real Convolutional Codes

Convolutional codes have been defined traditionally over finite field alphabets [21, 22], but recent research results show how they may be extended to systems using either integer or real arithmetic [18, 20, 14]. Nevertheless, the basic approach to convolutional codes remains the same, particularly with regard to a matrix description of the encoding and parity checking functions. Only systematic forms of convolutional codes will be considered primarily because the normal filtering operations are not altered and such forms are automatically noncatastrophic [22]. Only the detecting capabilities of such codes are used; any correcting operations could easily exceed the original processing requirements.

The encoding matrix for a systematic convolutional code, G, has a block-type format involving m fundamental finite sized matrices whose dimensions are related to the rate and number of parity check positions in the code. The parameter m determines the constraint length of the code.

$$G = \begin{pmatrix} G_0 & G_1 & \cdots & \cdots & G_m & 0 & \cdots \\ 0 & G_0 & \cdots & \cdots & G_{m-1} & G_m & \cdots \\ 0 & 0 & \cdots & \cdots & \cdots & G_{m-1} & \cdots \\ 0 & 0 & \vdots & \vdots & \cdots & \cdots & \vdots \\ 0 & 0 & \cdots & \cdots & \cdots & \cdots & \cdots \\ \vdots & \vdots & \vdots & G_0 & \cdots & \cdots & \vdots \\ \cdots & \cdots & \cdots & \vdots & G_0 & G_1 & \cdots \\ \vdots & \vdots & \vdots & \cdots & 0 & G_0 & \cdots \\ \cdots & \cdots & \cdots & \cdots & 0 & 0 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{pmatrix} \quad (3)$$

The $k \times n$ submatrices G_j, $j = 0, 1, \ldots, m$, have distinctive forms and divide into two types.

$$G_0 = (I \mid P_0) \quad ; I,\ k \times k \quad \text{Identity Matrix}$$
$$P_0,\ k \times (n-k) \quad \text{Parity - Check Matrix} \quad (4a)$$

$$G_j = (0 \mid P_j) \quad ; 0, k \times k \quad \text{Zero Matrix}$$
$$; P_j, k \times (n-k) \quad \text{Parity - Check Matrix} \quad (4b)$$

The entries in the parity check submatrices P_i may be either 0 or 1 even for the real Marshal code case [18, 20], or in the more general case, real numbers [14, 23].

The parity positions are a function of possibly $M = (m+1)k$ input samples through the action of the P_j parts of each G_j. The stack of these parity weighting values will be denoted by an $\{M \times (n-k)\}$ matrix Q with respective columns $\{\underline{q}_r\}$. The $(n-k)$ parity position associated with the input values are obtained by the weighing action of columns $\underline{q}_r$. Each parity value may be viewed as the output of an FIR filter, described notationally using the Z transform of column $\underline{q}_c$:

$$Q_c(Z) = \sum_{j=0}^{M-1} q_c^{(M-1-j)}\, Z^{-1}, \quad (5)$$

where $c = 0, 1, 2, \ldots, (n-k-1)$.

Real convolutional codes can also be imbued with a distance structure similar to the usual one applied to finite field symbol codes. It is possible to define a metric in terms of a real Hamming weight.

High rate convolutional codes with only one parity channel will be used for protecting output data channels emanating from a demultiplexer. Binary-based codes, for which there exist tables of high performance codes [24], will be chosen. In particular, a rate K/(K+1) systematic convolutional code is defined by a single parity weight filter, equations (5). A single parity value for every K input sample is produced by sampling an FIR filter with transfer function denoted by Q(Z). The data flow normally and are simultaneously tapped to this FIR parity filter, Q(Z). A convenient view of the parity production process is shown in Figure 4. The data flow normally and are simultaneously tapped to this FIR parity filter, Q(Z). The downsampling symbol $\downarrow$K indicates that after every K data samples, one parity value is produced.

Composite Filtering and Parity Generation

This section develops methods for combining the parity generation operations with filter banks, such as shown in Figure 3, forming a cascaded system, depicted in Figure 5. A generic channel with signal value notation overlaid is presented in the middle of this figure. The output of the t^{th} filter, $H_t(Z)$, is denoted by $Y_t(Z) \leftrightarrow \{y_t(r)\}$. The parity output $\{p_t(a)\}$, after downsampling by factor K, may be written in terms of the t^{th} channel signal $\{y_t(r)\}$, which in turn can be expressed using segments of the filter's impulse response.

$$p_t(a) = \sum_{v=0}^{N-1} \sum_{u=-\infty}^{+\infty} x(uN+v)\; g_t^{(v)}(aK-u) \quad (6a)$$

The composite weighting functions $\{g_t^{(v)}(r)\}$ contain every N^{th} sample of the filter weighting, properly offset by index v.

$$g_t^{(v)}(s) = \sum_{r=-\infty}^{+\infty} q(r)\, h_t((s-r)N-v), \quad (6b)$$

where $v = 0, 1, \ldots, N-1$. The output sample index a is scaled by K in the argument of $g_t^{(v)}(\ )$ inside the definition of $p_t(a)$, equation (6a), while it is further scaled by N in this definition, equation (6b). The net effect has the input data weighted by values every N^{th} point, in steps of KN with respect to the data indices. The Z transform of the composite impulse responses, equation (6b), will be denoted by $G_t^{(v)}(Z)$.

The real savings in computing the respective channel parities occur for the case of uniform filters at the critically sampled rate, L = N. With the filter bank as in Figure 2, the outputs of each $H_t(Z)$ are scaled by a complex phasor as in equation (2). This translates the parity channel output $p_t(a)$ into a modified equations (6).

$$p_t(a) = \sum_{v=0}^{N-1} \sum_{u=-\infty}^{+\infty} [x(uN+v)\, e^{j\frac{f_s}{N} tv}]\, g^{(v)}(ak-u) \qquad (7)$$

The uniform filter weighting function $g^{(v)}(s)$ is defined similarly to equation (6b). The complex roots of unity are functions only of the outer index v, and, when all N channels are considered, the complete set of parity values may be calculated by a DFT operation, as described earlier with regard to the polyphase multirate filter banks. The calculation rate is reduced by a factor KN, even though the individual composite channels accept data at intervals of N.

Protecting a Polyphase Filter Demultiplexing System

The parity values are calculated in two ways, one by a parallel composite parity generation process as described in the last section. The second comparable parity values are computed directly from the channel's demultiplexed output. The first set of parities are calculated according to equations (6) employing the composite weighting. The other parity estimates are computed directly from the individual channel outputs using the parity weighting Q(Z). These two versions of $p_t(a)$, labeled $p_t'(a)$ and $p_t''(a)$ are compared in a totally self-checking comparator. The combined protection system is detailed for generic channels r in Figure 6; identical calculations for each of the N outputs would be made.

The full details of this generalized version of a totally self-checking equality checker [7] are contained in a book chapter [25]. The threshold value Δ in this comparator is selected to allow small differences between the two versions of comparable parity samples, accounting for roundoff noise discrepancies arising because they are computed by different subsystems. The parity weight filters, $G^{(v)}(Z)$ blocks in Figure 6, combine the effects of Q(Z) and H(Z). However, the computational rate is reduced further by a factor of K, making this scheme an efficient protection approach. Since each channel compares a pair of parity values every K^{th} output value, errors are detected with a latency of at most K output samples. The detecting capability of the code is sometimes specified in terms of the minimum distance for a constraint length.

Conclusions and Future Work

This paper has demonstrated how real convolutional codes can be employed efficiently for protecting demultiplexer filter banks. Each demultiplexer output channel has two forms of low rate parity calculation associated with it. One value is computed directly from the output using an FIR parity filter dictated by the structure of a real convolutional code. The memory in the parity filter is determined by the constraint length of the code while its very favorable downsampled processing rate is governed by the code rate. The other parity value is computed in parallel with the normal processing by a composite filter operating at a reduced rate as governed by the convolutional code choice. These parallel parity calculations can be implemented very efficiently by a polyphase multirate filter bank, virtually identical with the main demultiplexer bank, except operating at a much lower rate.

Research is continuing on the fault tolerance aspects of other subsystems shown in Figure 1. The control sections inherent in each subassembly are not always visible to the system designer. The data level protection techniques promoted in this paper provide coverage for many control action failures. However, there are control steps that are not directly covered, for example, those actions associated with parity comparison results or reconfiguration decisions. There are evolving techniques that can be applied at the microcode level employing embedded checks recomputed by a small hardware monitor at run time and compared [26-27].

The demodulators are very difficult to protect because of their internal phase and timing tracking loops. On the other hand, the forward error correctors (FEC) directly following contain redundancy checks. These devices generally use convolutional codes and implement the Viterbi Algorithm [21-22]. Hardware failures in a channel demodulator appear as channel errors to the respective FEC devices. Hence, if an individual FEC subassembly is fault-free, any errors detected by it lead to suspecting failures in the preceding DEMOD unit. This is a "sandwich" approach to fault tolerance. If succeeding and preceding units are fault-free and errors are detected by the protected succeeding unit, the item in the middle of the "sandwich" suspect. Work is progressing on introducing data level fault tolerance in Viterbi algorithm type decoders, primarily by attaching real parities to groups of path metrics [21-22]. Again, the principle of algorithm-based fault tolerance is used. Real parity values are computed and recomputed and then compared.

References

[1] M. Bellanger, *Digital Processing of Signals: Theory and Practice* (2nd Edition). New York: John Wiley & Sons, 1989.

[2] M. Bellanger and J. L. Daguet, "TDM-FDM Transmultiplexer: Digital Polyphase and FFT," *IEEE Transactions on Communications*, Vol. COM-22, pp. 1199-1205, 1974.

[3] M. Bellanger, G. Bonnert and M. Coudreuse, "Digital Filtering by Polyphase Network: Application to Sample-Rate Alteration and Filter Banks," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. ASSP-24, pp. 109-114, 1976.

[4] R. E. Crochiere and L. R. Rabiner, *Multirate Digital Signal Processing*. Englewood Cliffs: Prentice-Hall, 1983.

[5] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing*. Englewood Cliffs: Prentice-Hall, 1989.

[6] D. K. Pradhan, Editor, *Fault-Tolerant Computing Theory and Techniques*, Vol. 1., Englewood Cliffs: Prentice-Hall, 1986.

[7] J. Wakerly, *Error Detecting Codes, Self-Checking Circuits and Applications*. New York: North-Holland, 1978.

[8] B. W. Johnson, *Design and Analysis of Fault-Tolerant Digital Systems*. Reading, MA: Addison-Wesley Publishing Company, 1989.

[9] K.-H. Huang and J. A. Abraham, "Algorithm-Based Fault Tolerance for Matrix Operations," *IEEE Transactions on Computers*, Vol. C-33, pp. 518-528, 1984.

[10] J.-Y. Jou and J. A. Abraham, "Fault-Tolerant Matrix Arithmetic and Signal Processing on Highly Concurrent Computing Structures," *Proceedings of the IEEE (Special Issue on Fault Tolerance in VLSI)*, Vol. 74, pp. 732-741, 1986.

[11] J. A. Abraham, "Fault Tolerance Techniques for Highly Parallel Signal Processing Architectures," *SPIE Highly Parallel Signal Processing Architectures*, Vol. 614, pp. 49-65, 1986 (K. Bromley, Editor).

[12] C. J. Anfinson and F. T. Luk,"A Linear Algebraic Model of Algorithm-Based Fault Tolerance," *IEEE Transactions on Computers*, Vol. C-37, pp. 1599-1604, 1988.

[13] F. T. Luk and H. Park,"An Analysis of Algorithm-Based Fault Tolerance Techniques," *SPIE Advanced Algorithms and Architectures for Signal Processing*, Vol. 696, pp. 222-227, 1986.

[14] W. G. Bliss,"Area-Time Efficient and Fault-Tolerant VLSI Arrays for Digital Processing," Ph.D. Thesis, Department of Electrical and Computer Engineering, University of Colorado, 1988.

[15] C. J. Anfinson, R. P. Brent and F. T. Luk, "A Theoretical Foundation for the Weighted Checksum Scheme," *SPIE Advanced Algorithms and Architectures for Signal Processing*, Vol. 975, pp. 10-18, 1988.

[16] R. P. Brent, F. T. Luk and C. J. Anfinson, "Choosing Small Weights for Multiple Error Detection," SPIE High Speed Computing, Vol. 1058, pp. 16-1–16-7, 1989.

[17] F. T. Luk, "Algorithm-Based Fault Tolerance for Parallel Matrix Equation Solvers," *SPIE Real-Time Signal Processing*, Vol. 564, pp. 49-53, 1985 (W. J. Miceli and K. Bromley, Editors).

[18] T. G. Marshall, Jr., "Coding of Real-Number Sequences for Error Correction: A Digital Signal Processing Problem," *IEEE Journal on Selected Areas in Communications*, Vol. SAC-2, pp. 381-392, 1984.

[19] J. K. Wolf, "Redundancy, the Discrete Fourier Transform, and Impulse Noise Cancellation," *IEEE Transactions on Communications*, Vol. COM-31, pp. 458-461, 1983.

[20] T. G. Marshall, Jr., "Real Number Transform and Convolutional Codes," *Proceedings 24th Midwest Symposium on Circuits and Systems*, Albuquerque, NM, pp. 650-653, June 1981.

[21] W. W. Peterson and E. J. Weldon, Jr., *Error-Correcting Codes* (Second Edition). Cambridge, MA: The MIT Press, 1972.

[22] S. Lin and D. J. Costello, Jr., *Error Control Coding Fundamentals and Applications*. Englewood Cliffs: Prentice-Hall, 1983.

[23] V. S. S. Nair and J. A. Abraham,"Real Number Codes for Fault-Tolerant Matrix Operations on Processor Arrays", *IEEE Transactions on Computers*, Vol. C-39, pp. 426-435, 1990.

[24] J. Hagenauer, "High Rate Convolutional Codes with Good Distance Profiles," *IEEE Transactions on Information Theory*, Vol. IT-23, pp. 615-618, 1977.

[25] G. R. Redinbo "Real Codes in the Fourier Domain for Fault-Tolerant Signal Processing." Chapter in *Spectral Techniques: Theory and Applications*. North-Holland: Amsterdam, 1991. (Moraga and Creutzburg, Editors.)

[26] Kent D. Wilken, "Optimal Signature Placement for Processor-Error Detection using Signature Monitioring", *Proceedings Twenty-First International Symposium on Fault-tolerant Computing*, Monteral, Canada, pp.180-187, June 1991.

[27] K. Wilken and J. Shen, "Continuous Signature Monitoring: Low-cost Concurrent-Detection of Processor Control Errors", *IEEE Transactions on Computer-Aided Design*, col. CAD-9, pp.629-641, June 1990.

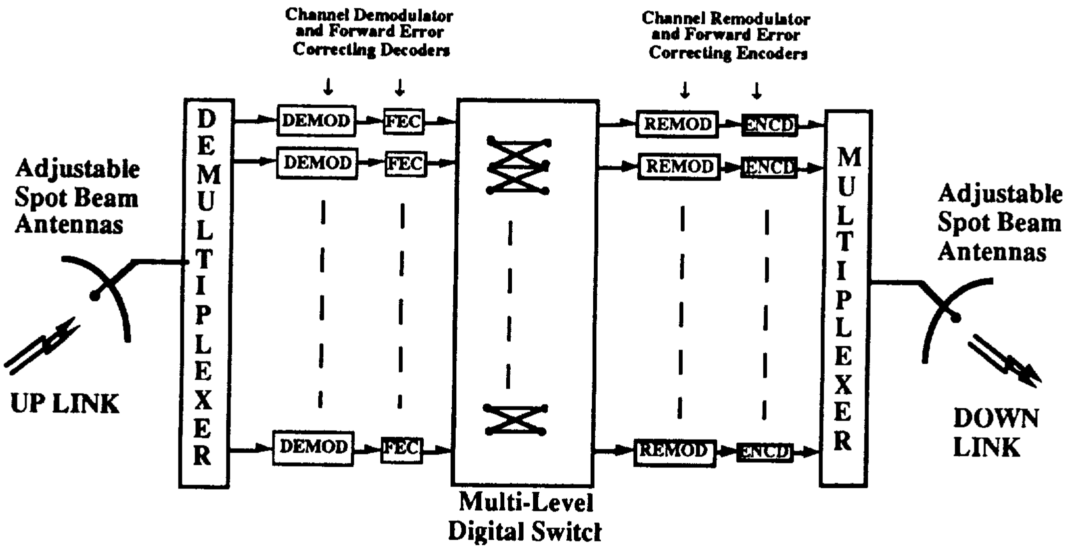


Figure 1: Typical Communication Satellite Subsystems

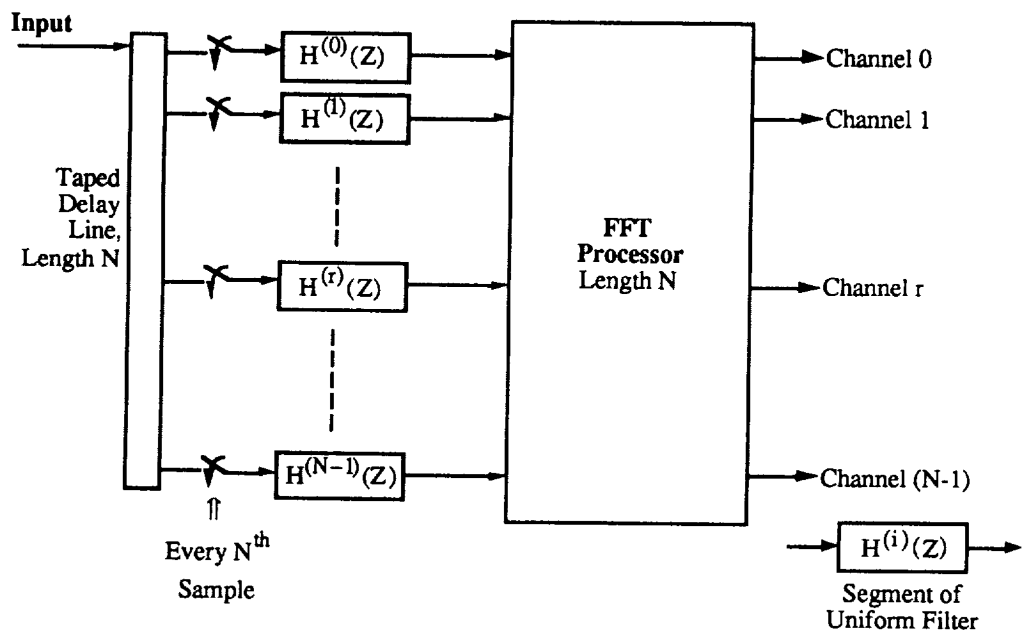
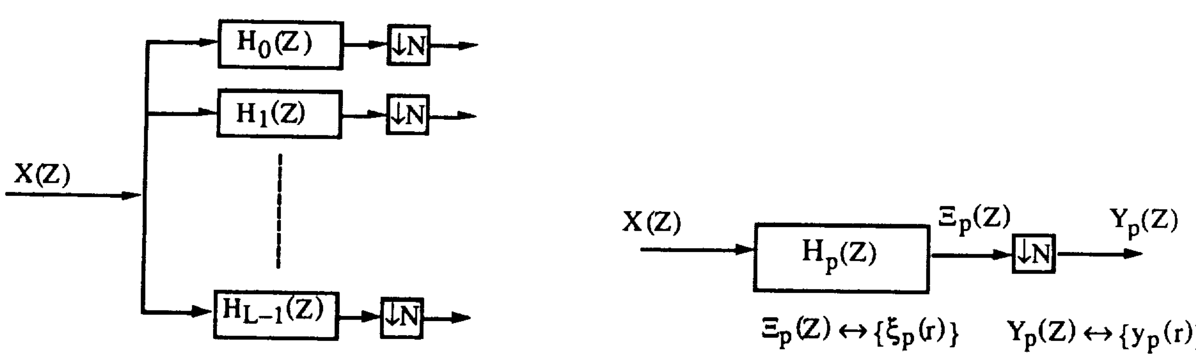


Figure 2: Polyphase Multirate Filter Bank



(a) Bank of L Filters Followed by Downsamplers

(b) Generic Filter Path

Figure 3: General Analysis Bank

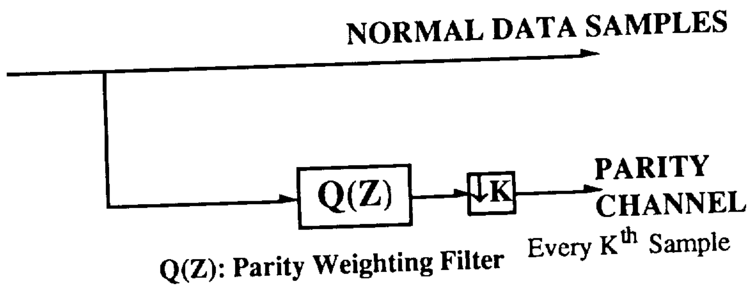


Figure 4: Parity Generation in a Rate (K/K+1) Systematic Convolutional Code

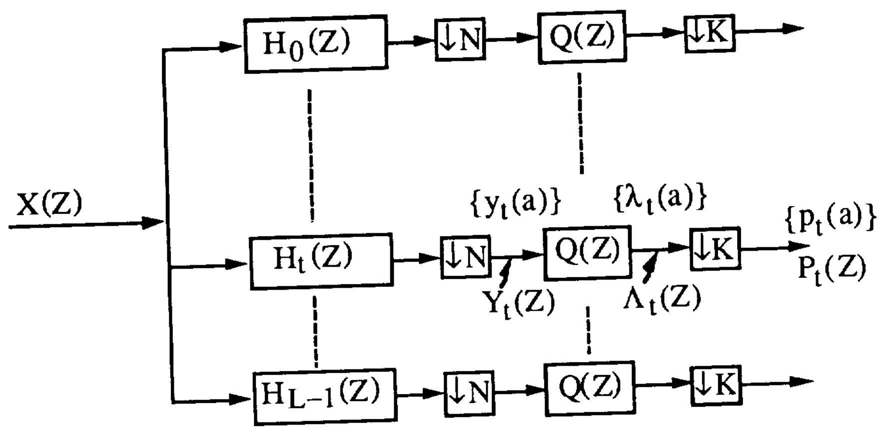


Figure 5: Parity Generation for Analysis Bank Outputs

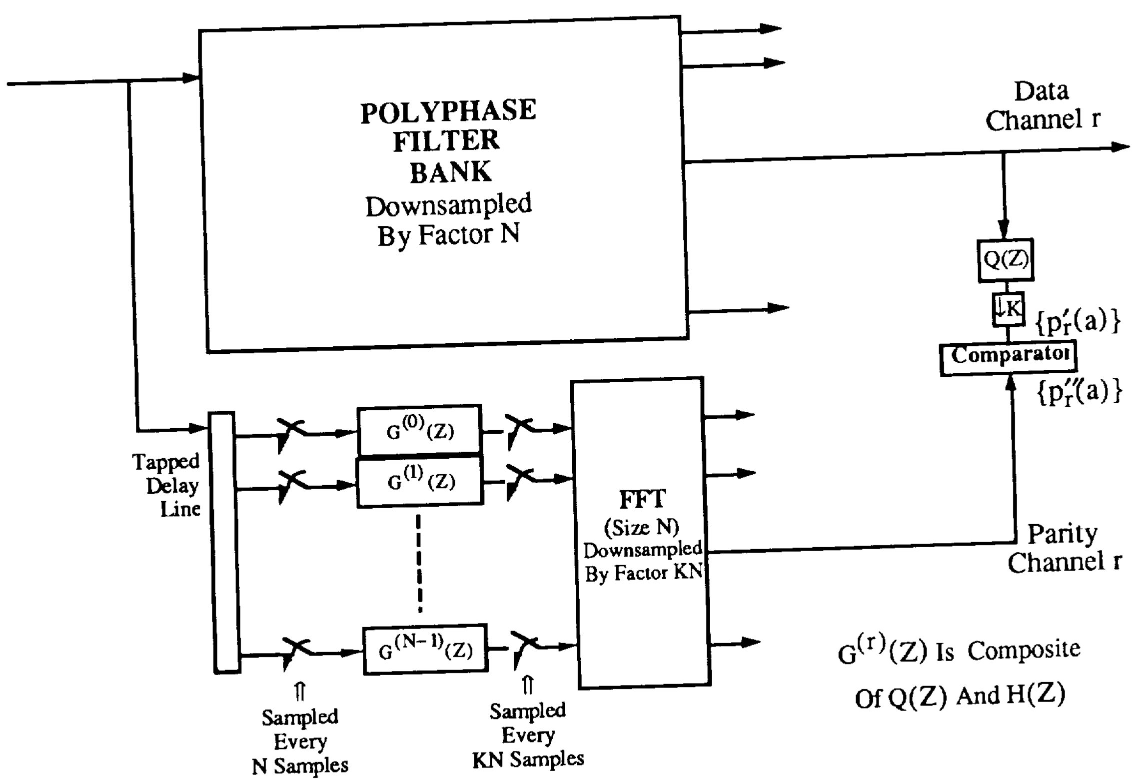


Figure 6: Protection of Demultiplexer Channels