

Digital Channel Error Correction Coding Design Tools

Error correction coding theory describes ways to add overhead to transmitted or stored messages to insure that they can be understood when received or played back. Challenging implementations of error correction coding theory are responsible for many advances in digital communications applications ranging from deep space exploration to high-density disk drives, cellular telephones and pager networks.

Introduction

The increased use of error correction techniques by digital communications designers has created a demand for tools to evaluate and exercise error correction coding approaches before they are committed to expensive ASICs or firmware. Techniques based on studying the exact bit location of errors as they occur in the underlying digital channel, can be used to evaluate error correction coding performance before the error correction layers are added. Using these types of real-time analyzers is no more difficult than using familiar bit error rate testers and the results can create efficient systems in significantly less time.

Error correction coding in modern systems

Practically all systems that transmit or store digital data use some form of error control technique. In the presence of real-world phenomenon, designers of these systems have to face the certain outcome that some transmitted or stored bits of information will not be received the way they were intended. The choices of how to handle errors range from doing nothing to using elaborate error detection and correction methods. Selecting between the choices depends on the accuracy, speed, and latency requirement of the information, and whether or not there is simultaneous bi-directional communications between the sender and receiver.

The accuracy requirement of information is derived from the eventual use of the data. When storing data files used in computer systems, data accuracy of one error in 10^{13} bits is considered acceptable—a bit error rate of 1×10^{-13} . This amounts to one error in 1,250 Gigabytes of data. When sending instrumentation sensor data across a link, a bit error rate of 1×10^{-9} is quite acceptable. When playing back digital video for broadcasting, a bit error rate of 1×10^{-7} typically exceeds the requirements. In digitized PCM voice communications, bit error rates of 1×10^{-4} can provide acceptable performance. In some cases, random errors are more tolerable than burst errors allowing a system with a worse bit error rate to be perceived as higher quality.

The speed and latency of information is also derived from the application. Small transaction processing, such as databases accesses or communications in wireless networks require low latency between the time when data is received and when it is ready to be used (e.g., any post-processing such as an error correction step). Applications of data recording onto tape can typically use as much latency as is desired. Large latency concession allows error correction code words to be spread through more and more user data, generally providing better burst error correction ability. This comes at the expense of significant buffering and delay.

The data rate of a communication channel can often limit the practical error correction approach. In a streaming application, it is imperative to keep up with the real-time data. Depending on the level of hardware support for the error correction step and the distribution of error statistics, certain error correction algorithms cannot be performed

fast enough to keep up with the data rate. Digital channels implemented in firmware devices have much less error correction horsepower than a hardware modem or controller with highly integrated error correction chips built in.

A bi-directional communication between a sender and receiver in some cases can eliminate the need for forward error correction systems. In these environments, systems can be created with very low bit error rates by using powerful and simple error detection with a means of asking the transmitter to re-send data that was not received correctly. This is the basis of many network protocols. This same practice can be applied to magnetic recording systems that have an ability to verify their write data while they are writing. Errors detected during writing can trigger re-writing the errored data block in anticipation of trouble during the eventual read-back.

Design Methodology for Error Correction Systems

The first step when designing an error handling approach is to identify the requirements for the error statistics of the final channel. This should not only be specified as a bit error rate, but should also define burst profiles. Systems have been delivered that meet the bit error rate specification yet fail to operate as intended because of isolated burst events.

The next step is to characterize the expected underlying error characteristics of the raw link. This should be done particularly in the face of anticipated failure modes. In a disk drive, this would mean looking at the channel when the head is not centered directly over the track. In a satellite communications application, this would mean looking at error statistics during rain/snow storms or during high sunspot activity. When designing error correction systems, characterization of the raw error performance of a channel is much more demanding than a simple bit error rate measurement. Burst statistics and the relationship between bit and burst errors to eventual data blocking factors are of tremendous importance.

Beyond the basic bit error rate, useful digital channel error analysis includes a comprehensive study of the bit location of errors in the link. Bit error location analysis can come in the form of a histogram of burst lengths, a histogram of intervals between errors, error auto-correlation, block-oriented error distributions and more. Errors in a digital channel are often the result of a few different phenomena. For example, it is typical to have a component of random noise induced errors along with a burst error generating aspect (e.g., lightning, media defects, etc.). Each component of the error statistics of a channel may be combated in a different way so it is important to isolate errors of different types.

Once statistics are gathered and system requirements are defined, the next step is to match error distributions and processing requirements to error correction approaches.

Correction Architectures

There are many practical error correction architectures from which to choose. These include maximum likelihood detectors (e.g., Viterbi decoder), Fire Codes, Reed-Solomon block codes, Hamming codes and CRC error detection with retransmit. Along with block error correctors, it is typical to incorporate shuffling (or interleaving) of data in order to scramble burst error distributions among many error correction code words.

Maximum likelihood detectors typically operate early in the bit detection processing to help make bit or symbol thresholding decisions based on a priori knowledge and maximum likelihood bit patterns. In a magnetic recorder, for example, two neighboring bit samples of a disk drive PRML waveform cannot both be North poles (they must alternate from North to South). A detector that sees an S-N-N-N-S sequence will correctly decode an S-N-S-N-S sequence. Maximum likelihood detectors have greatest effects on small, random errors and are therefore often used in conjunction with other methods to achieve desired error performance. The operation of a maximum likelihood detector is typically implemented using a RAM look-up table approach. This method finds the most likely bit sequence by assigning quality metrics to legal bit sequence transitions accepting the highest quality bit sequence.

Fire codes are ideal for rare occasions of smaller bursts. A Fire code is an error correction strategy that appends a small number of bits (e.g., 16-32 bits) to a block of data during encoding so that a single small burst of errors in each data block can be corrected at the receiver. There are many benefits to Fire codes in that the same checksum used for commonplace error detection can be further used for a limited amount of error correction. A typical 32-bit checksum used in telecommunications and data storage applications will correct single error bursts that are less than 11 bits long. The total length of the "block" covered by the Fire Code is also limited in a similar, but more restrictive, way as a CRC (Cyclic Redundancy Code) is for error detection. Fire codes operate because of the careful selection of the particular CRC generating polynomial. These special polynomials can be factored into two

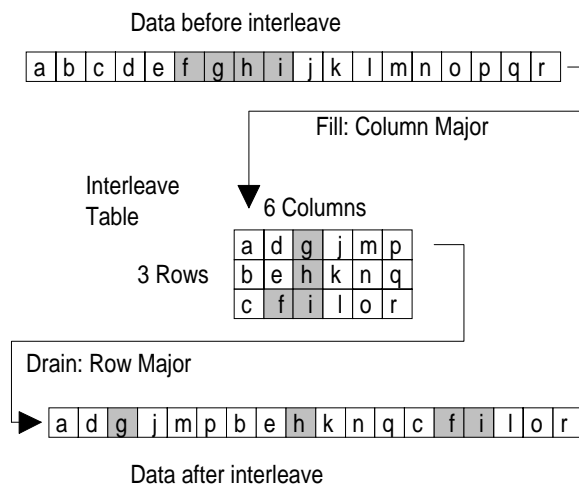
prime polynomials each of which are used to locate a detected error. Knowing the location of a transmitted error modulo the degree of each of these factored prime polynomials allows application of the Chinese Remainder Theorem to locate the exact bit error. Although this math sounds complex, it is easily implemented with D-flip-flops and XOR gates, and can operate at very high speeds.

Reed-Solomon block codes are very popular in both communications and data storage applications. Chips to implement very high-speed real-time correctors of this type are commercially available as well as DSP software solutions. Generally, a Reed-Solomon corrector will provide a number, say T, of symbol corrections in an N-symbol code word independent of the location of the errors inside the code word. Used with complex interleaving, large burst errors can easily be corrected using this approach. Further refinements of the algorithm allow even more correction ability in cases where the location of the error is known by some other means. These “soft error” indicators, for example, can come from upstream decoding violations. Like Fire codes, Reed-Solomon codes are implemented by appending symbols to the end of the transmission that are used during decoding to locate and correct errors. The detection process is logically similar but, because multi-bit symbols are used, it is significantly more processing intense.

Hamming codes are typically found in RAM memory sub-systems. Hamming codes offer simple decoding and correction of very small code words at very high speeds. Hamming codes typically protect against a set number of bits in the code word allowing multiple bit errors to still cause problems. Hamming codes add to each protected word a number of check bits used to increase the uniqueness between words. Uniqueness, in this case, is measured by the number of bit positions that any two words differ from each other. This is known as the Hamming distance. The minimum Hamming distance for all possible words, d, defines the correction ability of the Hamming Code to be $[d - 1]/2$. Hamming codes are most typically implemented using simple XOR gates or RAM lookup tables to compute the most likely data word from a corrupted code word.

Error management with CRC error detection is accomplished by appending to each data block message a small number of overhead bits, termed the CRC, at the end of the message. These bits are carefully constructed to create a zero CRC when the message is decoded. There are a number of failure modes of this error detection process relating to the size of the generator polynomial. Typical CRCs add 16 or 32 bits to a message. This can typically be turned into an error correction approach only if a back channel is supported allowing a data block retransmit request. CRC error checksum polynomials are generated and checked using simple D-flip-flop and XOR gate linear feedback shift registers that can operate at very high speeds.

Interleaving is a simple way to spread apart dense errors, which would otherwise overload a single error correction code word, into an acceptable error distribution.



In interleaving systems, user data is placed into an interleave table in one axis and then read out in the other axis. The size of the table defines the interleaving properties. In the example shown, a simplistic three row by six column interleave is used, which successfully separates a four-symbol burst into more easily correctable one and two symbol errors.

Interleaving introduces latency because enough data to fill an entire interleave must be available before encoding or decoding can begin. Interleaves also require that the granularity of data availability be at least the size of one full

interleave. Two and three-dimensional interleaves are used in communications and data storage products that can correct error bursts of thousands of bytes in real-time.

ECC dependence on error distributions

Each of these error correction approaches has overload characteristics that will cause them to fail. For example, if there are two errors separated by more than 11 bits in a 32-bit Fire code protected block, it will fail to correct. If a Reed-Solomon code word has more than T symbol errors, it will not be able to both locate and correct the error (it could make corrections if the locations were already known).

The location of errors with respect to each other and with respect to symbol and block boundaries is very important. Identification of the amounts and correlation of random and burst errors is needed when choosing the correction strength or the interleave depth of advanced correctors. Even the distribution of burst events becomes important if burst events are correlated. An error correction system that is designed to withstand the typical observed burst size will probably fail if one burst predicts another burst which falls within the same code word interleave (for example, the correlated bursts that occur in a magnetic read head bouncing off a large media defect).

Advanced Analysis Techniques

Analysis techniques used when designing and evaluating error correction systems include measuring bit and burst error rates, burst length histograms, error auto-correlation, block error distributions and error free interval histograms. All of these analyses rely on knowing the exact bit location of errors a channel.

Identifying the amount of isolated bit errors and burst errors is an important first step. The best burst error correctors can be ineffective with an overload of background errors. Isolated errors might be pattern dependent allowing focused attention on signal equalization and coding to make bit error rate improvements. If all errors are isolated (and random) there is no need to interleave user data because they are already randomly distributed with respect to correction code words.

Determining burst errors depends on categorizing neighboring errors into a single burst event. Some tolerance is needed when grouping errors into bursts to allow for a certain number of error-free bits within a burst. Common instruments allow users to set a maximum error free interval (in bits) for this purpose. For example in Fire codes, the maximum error free interval should be artificially set to the blocking length of the data to assure that any two errors in the data block show up as a single burst.

Once bursts are identified, a histogram can be used to show the probability distribution of having bursts of different length. This cannot be used alone for designing error correctors because it is not known yet where any two bursts are with respect to each other. Burst length histograms typically reflect the underlying physics of the digital channel. Factors such as lightning strike duration, magnetic head shapes and defect sizes contribute to this. Figure A shows a histogram of burst lengths in a channel that is already interleaved with a 1632-bit interleave using a very large minimum error free interval setting. Set up in this way, the histogram shows that many bursts are measured between 4,000 and 5,000 bits indicating that raw pre-interleaved errors were five symbols long.

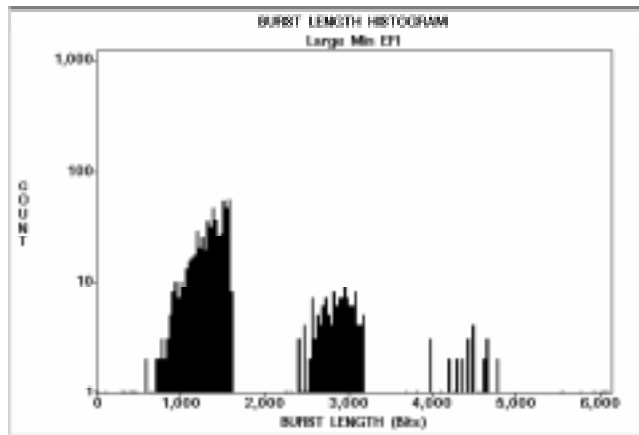


Figure A

Error auto-correlation will show how errors are correlated to each other. This analysis will answer the question, “Does one error predict another error?” The first part of an error auto-correlation looks very similar to a burst length histogram. The more interesting relationships occur farther away from the initial error and help to define interleave depths. For example, Figure B shows a sample auto-correlation in a channel used for data storage. In this case, there is a high correlation between having one error and then having more errors 34,848 bits away. In this system, it is due to defects in the media manufacturing process, which are inherent to magnetic media. These defects will not likely be removed, so error correctors must deal with them. Interleave depths should be chosen to separate errors by at least this degree to isolate these highly correlated error events into separate code words.

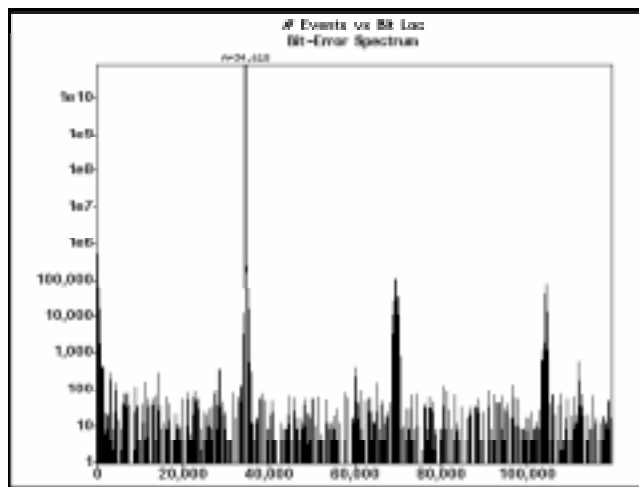


Figure B.

Block error distributions are histograms of the number of errors that occur within a user-programmed block size independent of their location within the block. This analysis is good to use on correctors that are position independent such as Reed-Solomon. A block error analysis can be used to choose the correction strength, T, needed to correct each block. Any entries in the block histogram larger than the T value selected will not be corrected.

Evaluating ECC Using Emulation

Evaluation of error correction strategies can be done by emulating interleaves and correction operation. This is done in commercial digital channel error analyzers. In order for this to happen, the exact bit position of errors in the channel must be known and plotted in a table mirroring the behavior of blocking and interleaving. Once the table is full, a check of the limits of the corrector can be done to evaluate which errors would be removed and which would cause overload.

The result of error correction emulation is a modified stream of error locations that can now be studied in the same way as an uncorrected stream. Post correction error rates, burst lengths, correlation and even further levels of error correction can be applied.

Instrumentation Impacts

Equipment to perform this type of analysis connects into systems in the exact way as older bit error rate testers. Users can send pseudo-random data patterns or more complex formatted data streams through their evaluation channels and then send them to the error analyzer. The analyzer will synchronize to the users data stream and then start locating error positions. Users may also supply marker information that assigns blocking factors that are used during analysis.

Systems where traditional bit error rate testers have not found large applicability can often be interfaced to digital channel error analyzers. Many of the limiting factors that have caused bit error rate testers to be difficult to use include their common bit-serial interfaces and their reliance on pseudo-random data patterns. New digital channel error analyzers offer serial and parallel interfaces with user-programmable patterns that can include sections for equalization training, phase lock loop detection, synchronization patterns and more.

Analysis techniques can be performed in either real-time mode or as a post-processing step after error data logging. Post-processing an error data set allows users to evaluate the effect of different correction interleaves and strengths on the same raw error data. This apples-for-apples comparison holds the raw channel errors constant while varying only the correction parameters.

Conclusion

Location, location, location. These are not only words for realtors. When designing modern error correction systems, the location of errors and their relationships to each other and symbol or code word boundaries is more important than measuring bit error rate. Analyzers that study error locations are as easy to use as earlier bit error rate testers and provide invaluable insight into a digital channel's behavior.



3475-D Edison Way
Menlo Park, California 94025
Voice: 650 364-1853
Fax: 650 364-5716
Email: info@synthesysresearch.com
Web: www.synthesysresearch.com