SURVEY OF BLOCK CODING APPLICATIONS IN COMPUTER SYSTEMS

J. J. STIFFLER
Special Projects Office
Raytheon Company
Sudbury, Massachusetts

Summary Block coding has received increasing attention over the past few years as a technique for detecting and correcting digital computer errors. This paper points out some of the unique aspects of this utilization of error-control codes and examines some specific applications.

Introduction Error-control coding techniques have been investigated extensively over the past decade. The major impetus for this effort has been, and continues to be, the efficacy of these techniques in various communication systems. Interest has been growing in recent years, however, in the applicability of these same, or similar, coding schemes to digital computer for detecting and correcting errors caused by both transient and permanent features[1]. This application of error-control coding differs from the more conventional communications applications for several reasons:

1. The probabilities of the various possible error patterns may be quite different in the two instances.

2. Computer data is frequently parallel (byte or word oriented) rather than serial (bit oriented) in nature.

3. The upper limit on an acceptable decoding delay may be considerably less in computer applications.

4. The possibility of errors arising within the encoders and decoders themselves is not necessarily a negligible factor in assessing the relative advantages of coding for a specific computer application.

5. In many cases, the errors to be detected or corrected can be produced by permanent as well as transient phenomena.

---

[1] See, for example, R.T. Chien, "Coding Applications in Computer Storage Systems", to be published.

The following paragraphs discuss a number of applications of coding techniques to computers. As will be seen, each of these applications illustrates the influence of at least one of the special constraints listed above.

Codes For Arithmetic Operations It has long been recognized that the arithmetic unit of a computer is one "channel", in particular, that can be subjected to error patterns not likely to occur in more conventional communication channels. A single carry error, for instance, can produce a long sequence of adjacent bit errors. One class of codes found to be effective for checking arithmetic operations is the class of residue codes obtained by adjoining to each binary number its residue modulo some integer A[2]. If two numbers are to be added, for example, their residues are also added, modulo A, to form the residue for the sum. Since the sum of the residues should equal the residue of the sum, any discrepancy indicates a failure in the arithmetic unit. It is easy to see that any integer A not a power of 2 is sufficient to assure the detection of a single failure (i.e., a failure affecting only one sum bit or only one carry bit) in an arithmetic unit. The best choice for A depends both on the architecture of the computer in question and on the relative costs attached to hardware complexity and processing delay. In general, the residue of a binary number is most easily determined when A is of the form 2^i-1 for some positive integer i. The IBM 360/91, for example, checks its arithmetic operations using a residue code with A=3 while the JPL STAR computer uses a residue 15 code for the same purposes.

Arithmetic codes, both residue codes and a closely related class of codes called product codes, capable of correcting as well as detecting errors have also been described.[2] Their application in a practical system has thus far been limited by the complexity of the hardware needed to implement such codes. An additional limitation is the fact that these codes are not capable of detecting all possible arithmetic unit failures; in particular, failures occurring during logical as opposed to arithmetic operations.

Codes For Storage Media Errors in magnetic (tapes and discs) and optical storage media are typically characterized by either isolated single errors or by bursts of errors.[3] From this point of view, such media are not especially different than most communication channels. Nevertheless, the most effective coding techniques for these devices differ from those usually used in communication systems for one or more of those reasons itemized earlier. In particular, the increased hardware or the decoding delay, or both, usually associated with communication channel coding are frequently not acceptable in computer applications. To circumvent this problem, the typical approach is to use hardware only to detect the occurrence of an error and to use the computer itself to determine the necessary correction using a "fast" decoding algorithm. Both Fire codes

---

[2] W.W. Peterson, "Error-Correcting Codes", M.I.T. Press, Cambridge, Mass. pp, 236-244, 1961

[3] R.T. Chien, op. cit.

and Reed-Solomon codes have been used quite effectively in this manner to detect and correct storage media errors.[3]

One coding scheme [3,4] used to correct errors in information read from a high-density magnetic tape is of particular interest in that it illustrates the dramatic improvement in the efficacy of a particular coding scheme when certain assumptions concerning the anticipated error patterns can be justified. Information is stored on the tape in nine parallel tracks, with each parallel 9-bit segment representing an 8-bit byte plus a 1-bit overall "vertical" parity-check. Typically, a 9-bit "horizontal" parity-check is periodically added to provide a parity-check on the bits in each of the tracks up to that point. As is well known, it is always possible to correct any single error with an iterative code of this sort, but many error patterns involving as few as two errors are not correctable. Because the large majority of errors perturbing a magnetic tape of this sort are limited to a single track., however, a relatively minor change in the coding scheme can significantly improve its effectiveness without increasing its redundancy.

The scheme is as follows: a phase-shifted version of each track is encoded using an (n,k) cyclic (or pseudo-cyclic) code with n-k=9 and the modulo-two sum of the nine 9-bit parity-check vectors is recorded in place of the horizontal parity-check referred to earlier. Since, presumably, at most one track experiences any errors between any two such parity checks, the modulo-two sum of the observed and recorded parity checks characterizes the error pattern perturbing that single track. The pattern of parity errors, as determined by the vertical parity checks, should then produce a phase-shifted version (modulo the generator polynomial) of this same error syndrome, with the relative phase shift identifying which track actually experienced the error, and the vertical parity checks then identifying the specific errors committed. The major advantage in recording the modulo-two sum of the nine sets of check bits is not in the resulting reduction in redundancy but rather in the fact that the nine encoder-decoders which would normally be required can be replaced with only one which, in effect, encodes or decodes all nine tracks simultaneously. As long as the assumption that only one track is subjected to errors at any given time is valid, this hardware reduction is achieved with no loss in error-correction capability.

Coding For Permanent Hardware Failures Codes capable of detecting or correcting transient errors are, of course, capable of correcting the same error patterns even when produced by permanent hardware failures. In the latter event, however, significant advantages sometimes accrue by the fact that the identification of a permanent error can materially increase the effectiveness of a particular code. Consider, for example, a

---

[4] D.T. Brown and F.F. Sellers, Jr., "Error Correction for IBM 800-bit-per-inch Magnetic Tape", IBM J. of Res. and Dev., Vol. 14, pp. 384-389, July 1970

random-access memory designed to store 32-bit words. Typically, 50% of the failures in such a memory are due to the failure of a single bit-line (i.e., a failure affecting the ability to read or write only one of the 32 bits in each word). If it were desired to code a memory of this type of protect against, say, three consecutive failures of this sort by using conventional error correcting techniques, the number of bits which would have to be stored per word would increase by 36%, from 32 bits to 50 bits. In contrast, if the decoder were designed to recognize failures as permanent, an increase of only 18%, to 39 bits, would provide identical protection. In this case, the decoder need only identify those bits stored in failed locations as erasures and correct them accordingly. Since the code used is less redundant, moreover, the required decoder can be correspondingly less complex.

Further simplification is afforded in man cases by switching out each defective bit line as soon as it is identified and replacing it with a spare line. In this case, up to three failures can be corrected in a 32-bit per word memory by using only four redundant bit lines for an 11% increase in word length. That is, a single parity-bit is used to identify the existence of an error. The error is then easily located using software diagnostics, the defective line is switched out, and one of the three spares switched in as a replacement. The efficacy of this latter approach vis-a-vis the other two obviously depends in part upon the complexity of the switching hardware plus the parity-check apparatus relative to that of the decoders required for the other schemes. Clearly, the complexity of the (36)-pole switch capable of selecting any subset of 33 of the available 36 bit lines could potentially overwhelm any advantage in simplifying the decoder and decreasing the storage redundancy. In actuality, however, the necessary.switching can generally be accomplished with considerably less hardware than thatneeded to implement the more complex encoders and decoders.[5]

Another technique frequently used to protect against permanent hardware failures involves the use of majority-logic devices for masking the effect of a fault. The common triplemodular redundant (TMR) concept is a simple application of this principle. In this configuration, the device to be protected is triplicated and the three identical units run simultaneously. Any output from the triplicated device is determined by a majority vote on the outputs of the three devices individually.

An interesting generalization of this concept was recently proposed by Larsen and Reed.[6] Rather than simply replicating each device (i.e., instead of using a simple repetition

---

[5] J. J. Stiffler, “Coding for Fault-Tolerant Data Storage”, to be published.

[6] R. W. Larsen-and I. S. Reed, “Redundancy by Coding Versus Redundancy by Replication for Failure-Tolerant Sequential Circuits”, IEEE Trans. on Computers, Vol. C-21, pp. 130-137, Feb. 1972

code), they suggested identifying each of the state variables associated with the device and defining a majority-logic decodable code across these variables. Using this technique, the authors were able to show that, at least under some circumstances, a given improvement in overall reliability could be achieved more efficiently using coding than by simple replication. In another paper[7], this same general approach is used to design fault-tolerant binary counters.

Conclusion Several examples were presented in which block coding techniques have been used effectively in digital computer systems. In most cases, the nature of the application resulted in significant modifications in the structure of the code or in the decoding algorithm or both. While the number of such applications is at present not large, the area is one of growing interest, and, it is safe to predict, will be the subject of considerable attention for some time to come.

[7] I. S. Reed and A.C.L. Chiang, "Coding Techniques for Failure-Tolerant Counters", IEEE Trans. on Computers, Vol. C-19, pp. 1035-1038, Nov. 1970