New Classes of Unidirectional Error-Detecting Codes

Behrooz Parhami

Dept. of Electrical & Computer Engineering
University of California
Santa Barbara, CA 93106, USA

Abstract

Unidirectional errors arise as a result of common VLSI failure modes and certain connection faults, particularly with serial data transmission. We consider a totally ordered set of symbols to be encoded in an order-preserving manner for unidirectional error detection. General order-preserving encoding and the special case of difference-preserving encoding (when the value v is encoded with its separate data part representing v + b to facilitate arithmetic operations) are considered and optimal codes are devised for each case.

I. Introduction

The theory of error-detecting and error-correcting codes plays an important role in the design of dependable and fault-tolerant digital systems [BOSE86a], [FUJI90]. An error affecting multiple bits in a codeword is *unidirectional* if the bit inversions characterizing the error are all of the same type (0→1 or 1→0). For example, a unidirectional error may change the codeword 00101101 to 01101111 (only 0→1 inversions) or 00000000 (only 1→0 inversions) but not to 01100101. Unidirectional errors arise as a result of common VLSI failure modes and certain connection faults, particularly when data is transmitted serially [COOK73], [PARH73], [PARH78], [WAKE75]. Thus many researchers have been motivated to study the properties of unidirectional errors and methods for their detection and correction in digital systems.

Unidirectional errors are easier to detect or correct than multiple-bit random errors involving the same pattern or number of bits in the sense that they require a lower level of redundancy and simpler encoding/decoding algorithms. Initial research on unidirectional error codes dealt only with their detection [BERG61], [FREI62]. Numerous researchers subsequently extended the class of unidirectional error-detecting codes by considering different types of codes (e.g., separable, non-separable), certain error subclasses (e.g., burst errors, byte errors), and varying system contexts [BLAU88] [BOSE84] [BOSE85] [BOSE86] [BOSE89] [DUNN90] [NICO86] [PARH73] [WAKE75] [PARH78]. Error correction and combined correction/detection schemes were also investigated [BLAU89] [BOSE81] [BOSE82] [BOSE82a] [BRUC89] [KUND90] [MONT90] [NIKO86] [PRAD80] [TAOD88] [VENK89a].

Encoding and decoding are fairly difficult for non-separate unordered codes and except through costly table lookup, virtually no arithmetic operation can be performed on the codewords without first decoding them. In this paper, we discuss the two classes of order-preserving and difference-preserving unordered codes. With the first class, straight binary comparison of two codewords or parts thereof can reveal the relative order of the corresponding symbols (or magnitudes in the case of numerical values). With the second class, the difference between the ranks of two symbols or magnitudes of two numbers can be found by direct subtraction of codewords or parts thereof.

The original motivation for this study was provided by the need to encode "dependability tags" (d-tags) for data objects in a data-driven dependability assurance scheme [PARH89d]. As each data object in this scheme must carry a d-tag, efficient encoding of d-tag values is of utmost importance. This application is such that d-tags are frequently compared and subjected on occasion to simple arithmetic transformations. If these transformations are effected through table-lookup (which is practical because d-tags are typically small), then order preservation is the only requirement for the code. Otherwise the more stringent requirement of difference preservation must be enforced. However the results presented here are of general interest since they lead to highly efficient arithmetic-type unordered codes.

Notes on notation: $\lg x$ denotes $\log_2 x$; $C_{n:q} = n!/[q!\,(n-q)!]$; $v(X)$ is the base-2 value of the binary vector X; $b(x)$ is the binary vector representing x in base 2; $b_k(x)$ is the k-bit binary representation of x (modulo 2^k).

II. Order-Preserving Codes

Consider an error code mapping the totally ordered set of values $x_1 < x_2 < \ldots < x_s$ into encoded representations or bit-vectors $E_1, E_2, \ldots, E_s$. The code is said to be *order-preserving* with respect to the subvector D of its bits if for all $i < j$, we have $v(D_i) < v(D_j)$. We call D the data-

CH3040-3/91/0000/0574$01.00 © 1991 IEEE

ordering subvector and the rest of the bits, say C with $E_i = \langle D_i, C_i \rangle$, the check subvector. Note that some checking may have been incorporated into D_i bits also, but whereas all bits of D_i are required to establish the correct relative order of two codewords, the bits in C_i are redundant in this respect and can be ignored to simplify the comparisons. Additionally, even though data-ordering and redundant bits are separate, we don't call the codes "separable" since the data-ordering part need not contain the standard binary encoding of the values being represented. In coding theory parlance, a separable error-detecting code is obtained when one starts with a particular set of k-tuples as the data parts and attaches to them suitable redundant bits. Our codes are not separable in this sense.

A. Use of Existing Codes

Any unordered code can be used as an order-preserving code. All that needs to be done is to assign the codewords in increasing order of their numerical values (as binary numbers) to the symbols in their ascending order. The third column in Table I shows how a 2-out-of-5 code can be used as an order-preserving code for the ten BCD digits.

Table I: Examples of Order-Preserving Codes.

Info Part: BCD Data Words	Berger Check (No. of Zeros)	Order-Preserving 2-out-of-5 Code	Another Five-Bit (4-plus-1) Code
0 0 0 0	1 0 0	0 0 0 1 1	0 0 1 1 0
0 0 0 1	0 1 1	0 0 1 0 1	0 1 0 1 0
0 0 1 0	0 1 1	0 0 1 1 0	0 1 1 0 0
0 0 1 1	0 1 0	0 1 0 0 1	0 1 1 1 1
0 1 0 0	0 1 1	0 1 0 1 0	1 0 0 1 0
0 1 0 1	0 1 0	0 1 1 0 0	1 0 1 0 0
0 1 1 0	0 1 0	1 0 0 0 1	1 0 1 1 1
0 1 1 1	0 0 1	1 0 0 1 0	1 1 0 0 0
1 0 0 0	0 1 1	1 0 1 0 0	1 1 0 1 1
1 0 0 1	0 1 0	1 1 0 0 0	1 1 1 0 1

The following result allows us to ignore the rightmost ("least significant") code bit in ordering of two codewords.

Theorem 1: In any order-preserving unidirectional error code, removal of the rightmost ("least significant") code bit will leave the ordering unchanged. In general, no other bit can be removed in the same way.

Proof: There are no two codewords that differ only in the rightmost bit, since the minimum distance between pairs of codewords in a unidirectional error code is 2. Thus if $E = E_{n-1}E_{n-2} \ldots E_1E_0$ and $E' = E'_{n-1}E'_{n-2} \cdots E'_1E'_0$ are codewords, we have $v(E) < v(E')$ if and only if $v(E_{n-1}E_{n-2} \ldots E_1) < v(E'_{n-1}E'_{n-2} \ldots E'_1)$. Hence the rightmost bits E_0 and E'_0 need not be compared in determining order. To show that the same cannot be said about any other bit position in general, we need to provide a single counter-example. The 2-out-of-5 code in Table I is sufficient for this purpose. In this example, it is easy to see that removal of any other bit position from the 5-bit code will reverse the ordering of some consecutive pair of codewords.

By Theorem 1, the non-separable 2-out-of-5 code shown in Table I can be viewed as a pseudo-separable code (consisting of 4 "data" bits followed by one "check" bit) with respect to ordering of codewords. Clearly as far as order preservation is concerned, the 2-out-of-5 code is superior to the 7-bit Berger code with 4 data bits.

B. Modified Berger Codes

As mentioned earlier, for many values of s, Berger codes are not guaranteed to be optimal. The following result allows us to use more efficient codes which we call modified Berger codes.

Theorem 2: The unidirectional error detection property of Berger codes is preserved if Berger check values are encoded by any order-preserving code to obtain a modified Berger code.

Proof: Consider two arbitrarily chosen Berger-code codewords $\langle D_1, B_1 \rangle$ and $\langle D_2, B_2 \rangle$ with $B_1 < B_2$ and the order-preserving mapping h. We show that if B_i is replaced by $C_i = h(B_i)$, no unidirectional error can transform $\langle D_1, C_1 \rangle$ into $\langle D_2, C_2 \rangle$. By assumption, D_1 has fewer 0s than D_2. Thus if a unidirectional error changes D_1 into D_2, it must do so through 1-to-0 inversions. But 1-to-0 inversions in C_1 can only decrease its value and thus cannot transform it into C_2. A similar argument shows the impossibility of changing $\langle D_2, C_2 \rangle$ into $\langle D_1, C_1 \rangle$ through 0-to-1 inversions.

A natural question of interest at this point is to find a general procedure for constructing optimal modified Berger codes given the size s of the symbol set to be encoded.

C. Optimal Order-Preserving Codes

Optimal order-preserving codes for unidirectional error detection are obtained if we choose the s data parts so that a minimal number of Berger check values are required. We need at least $\lceil \lg s \rceil$ bits for the data part.

Theorem 3: Given $k \geq \lceil \lg s \rceil$ bits available for the data part, the number of check bits required in an optimal order-preserving unidirectional error-detecting code representing a set of s symbols is the minimum value of c for which the inequality

$$\Sigma_{i=\alpha..\beta} \; C_{k:i} \geq s$$

holds, where $\alpha = \max(0, \lfloor k/2 \rfloor - 2^{c-1} + 1)$ and $\beta = \min(k, \lfloor k/2 \rfloor + 2^{c-1})$.

Proof: Given c check bits, we can represent 2^c different check values and can thus have 2^c different weight classes in our data parts. By taking these different weights as close to $k/2$ as possible (i.e., from $\lfloor k/2 \rfloor - 2^{c-1} + 1$ to $\lfloor k/2 \rfloor + 2^{c-1}$), the number of codewords is maximized. If this maximum is at least s, we can successfully construct the code with c check bits. Otherwise we need more check bits.

Algorithm 1: To construct an optimal order-preserving unidirectional error code:

1. Initialize $k \leftarrow \lceil \lg s \rceil$ and $n \leftarrow \infty$.
2. Use Theorem 3 to construct a $(k + c)$-bit code for the smallest possible c.
3. If $n > k + c$, then set $n \leftarrow k + c$, $k \leftarrow k + 1$, and return to Step 2.
4. The last code that was constructed is the answer.

III. Difference-Preserving Codes

Difference-preserving codes constitute a more restricted class of order-preserving codes in which if D_i and D_j are the data parts in the encodings of values x_i and x_j, then:

$$x_j - x_i = v(D_j) - v(D_i)$$

In other words, there must exist an integer bias b such that $v(D_i) = x_i + b$. We will restrict our treatment of difference-preserving codes to the special case where $x_{i+1} - x_i = 1$ for all i; i.e. when a set of consecutive integer values are to be encoded.

Theorem 4: With k-bit data parts, the longest possible sequence of consecutive data vectors in which no more than w different weights occur is:

2^k if $w = k + 1$
$2^k - 1$ if $w = k$
$2^{w+1} - 2$ if $w < k$.

Proof: The set of all k-bit data words contains $k + 1$ different weights. Thus for $w = k + 1$, all the 2^k data words can be accommodated. In the case of $w = k$, one of the $k + 1$ weights must be excluded. By choosing the excluded weight to be 0 or k, we obtain $2^k - 1$ data words which is obviously the maximum possible. The proof in the final case of $w < k$ is more involved. Let the number of consecutive data words in this case be $f(w)$. We prove by induction on w that $f(w) = 2^{w+1} - 2$ with the data words being $b_k(1)$ through $b_k(2^{w+1} - 2)$; i.e., the binary vectors $0^{(k-1)}1$ through $0^{(k-w-1)}1^{(w)}0$ where $\sigma^{(r)}$ represents a subvector consisting of r repetitions of the symbol σ. Clearly the assertion holds for $w = 1$ when the two consecutive data words $b_k(1) = 0^{(k-1)}1$ and $b_k(2) = 0^{(k-2)}10$ represent a maximal set of size 2. Let $f(u) = 2^{u+1} - 2$ for $u < k - 1$ with the data strings $0^{(k-1)}1$ through $0^{(k-u-1)}1^{(u)}0$. The following construction shows that $f(u + 1) \geq 2^{u+2} - 2 = 2f(u) + 2$. In addition to the code vectors $0^{(k-1)}1$ through $0^{(k-u-1)}1^{(u)}0$ having weights 1 through u, we include in the new code the two strings $0^{(k-u-1)}1^{(u+1)}$ and $0^{(k-u-2)}10^{(u+1)}$ having weights $u + 1$ and 1, respectively. Then for each vector $0^{(k-u-1)}S$ in the original set, we include the vector $0^{(k-u-2)}1S$ in the new code, where S is any subvector of length $u + 1$. These new vectors will have weights 2 through $u + 1$. To complete the proof, we observe that the above number of data words is maximal (i.e. we have $f(w) \leq 2^{w+1} - 2$) since any sequence of $2^{w+1} - 1$ consecutive data words involves at least $w + 1$ different weights.

Theorem 5: A separable difference-preserving unidirectional error code for s symbols can be constructed with $k = \lceil \lg s \rceil$ data bits and c check bits where:

$c = \lceil \lg(k + 1) \rceil$ if $s = 2^k$
$c = \lceil \lg k \rceil$ if $s = 2^k - 1$
$c = \lceil \lg \lceil \lg(1 + s/2) \rceil \rceil$ if $s < 2^k - 1$.

It is interesting to note that the first two cases in the statement of Theorem 5 (i.e. $s = 2^k$ and $s = 2^k - 1$) can be represented by the single expression $c = \lceil \lg \lceil \lg(1 + s) \rceil \rceil$.

Proof: By Theorem 4, if $s = 2^k$, the data parts of our codewords will contain $w = k + 1$ different weights. Thus encoding of the corresponding check values requires $\lceil \lg(k+1) \rceil$ bits in this case. Similarly, if $s = 2^k - 1$, Theorem 4 suggests that $w = k$ different weights will be involved. This implies $\lceil \lg k \rceil$ bits for the check part. The final case of $s < 2^k - 1$ can be handled as follows. By Theorem 4, w different weights will be involved such that:

$$2^{w+1} - 2 \geq s$$

The above inequality yields

$$w \geq \lceil \lg(1 + s/2) \rceil$$

The number c of check bits must be such that $2^c \geq w$. Combining this inequality with the above, we obtain $c \geq \lceil \lg \lceil \lg(1+s/2) \rceil \rceil$ as required.

Algorithm 2: To construct an optimal difference-preserving unidirectional error code:

1. Set $k \leftarrow \lceil \lg s \rceil$.
2. Use Theorem 5 to find the number w of weights and the number c of check bits required.
3. If $s \neq 2^k$, $s \neq 2^k - 1$, and $w = 2^c$, pick the s consecutive data vectors to start from $b_k(1) = 0^{(k-1)}1$, resulting in a bias of $b = 1$. Otherwise start them with $b_k(0) = 0^{(k)}$.

We note from Algorithm 2 that in most cases, the optimal choice will have a bias of $b = 0$, resulting in a separable code. In the special case of $s \neq 2^k$, $s \neq 2^k - 1$, and $w = 2^t$ for some t, the bias can be 1. No other value for the bias is ever necessary to achieve optimality.

IV. Conclusion

We have introduced the two classes of order-preserving and difference-preserving unidirectional error-detecting codes. With the first class, straight binary comparison of

two codewords or parts thereof can reveal the relative order of the corresponding symbols (or magnitudes in the case of numerical values). With the second class, the difference between the ranks of two symbols or magnitudes of two numerical values can be obtained by direct subtraction of codewords or parts thereof.

For order-preserving UED coded, it was shown that in many cases, efficient pseudo-separable codes can be devised for which the determination of order between a pair of codewords is based on a subset of the bits (the "data" part). For difference-preserving UED codes, it was shown that modified Berger codes, which differ from regular Berger codes in that the data part does not necessarily cover all the 2^k possible combinations and the check part is modified by an order-preserving transformation, are optimal.

Further work is needed to replace the iterative procedure of Algorithm 1 with a formula that yields the number n of bits in the optimal code as a function of s, the symbol set size. This will be of theoretical interest though, since practically, the number of iterations is always fairly small.

References

Abbreviations: *IEEETC = IEEE Transactions on Computers, FTCS = Proc. Int'l Symp. on Fault-Tolerant Computing.*

[ALBA89] Al-Bassam, S. and B. Bose, "Design of Efficient Balanced Codes", *FTCS*, June 1989, pp. 229-236.

[BERG61] Berger, J.M., "A Note on Error Detection Codes for Asymmetric Channels", *Information and Control*, Vol. 4, pp. 68-73, Mar. 1961.

[BLAU88] Blaum, M., "Systematic Unidirectional Burst Detecting Codes", *IEEETC* **37**:453-457, Apr. 1988.

[BLAU89] Blaum, M. and H. van Tilborg, "On t-Error Correcting / All Unidirectional Error Detecting Codes", *IEEETC* **38**:1493-1501, Nov. 1989.

[BOSE81] Bose, B., "On Systematic SEC-MUED Codes", *FTCS*, June 1981, pp. 265-267.

[BOSE82] Bose, B. and T.R.N. Rao, "Theory of Unidirectional Error Detecting/Correcting Codes", *IEEETC* **31**:521-530, June 1982.

[BOSE82a] Bose, B. and D.K. Pradhan, "Optimal Unidirectional Error Correcting/Detecting Codes", *IEEETC* **31**:564-568, June 1982.

[BOSE84] Bose, B. and T.R.N. Rao, "Unidirectional Codes for Shift Register Memories", *IEEETC* **33**:575-578, 1984.

[BOSE85] Bose, B. and D.J. Lin, "Systematic Unidirectional Error Detecting Codes", *IEEETC* **34**:1026-1032, Nov. 1985.

[BOSE86] Bose, B., "Burst Unidirectional Error Detecting Codes", *IEEETC* **35**:350-353, Apr. 1986.

[BOSE86a] Bose, B. and J. Metzner, "Coding Theory for Fault-Tolerant Systems", Chap. 4 in *Fault-Tolerant Computing: Theory and Techniques*, Ed. by D.K. Pradhan, Prentice-Hall, 1986, pp. 265-335.

[BOSE87] Bose, B., "On Unordered Codes", *FTCS*, July 1987, pp. 102-107.

[BOSE89] Bose, B., "Byte Unidirectional Error Correcting Codes", *FTCS*, June 1989, pp. 222-228.

[BRUC89] Bruck, J. and M. Blum, "Some New EC/AUED Codes", *FTCS*, June 1989, pp. 208-215.

[COOK73] Cook, R.W. et al, "Design of Self-Checking Microprogram Control", *IEEETC* **27**:302-308, 1973.

[DUNN90] Dunning, L.A., G. Dial, and M.R. Varanasi, "Unidirectional Byte Error Detecting Codes for Computer Memory Systems", *IEEETC* **39**:490-503, Apr. 1990.

[FREI62] Freiman, C.V., "Optimal Error Detecting Codes for Completely Asymmetric Channels", *Information and Control*, Vol. 5, pp. 64-71, Mar. 1962.

[FUJI90] Fujiwara, E. and D.K. Pradhan, "Error-Control Coding in Computers", *Computer* **23**:63-72, July 1990.

[KNUT86] Knuth, D.E., "Efficient Balanced Codes", *IEEE Trans. Info. Theory* **32**:51-53, Jan. 1986.

[KUND90] Kundu, S. and S.M. Reddy, "Symmetric Error Correcting and All Unidirectional Error Detecting Codes", *IEEETC* **39**:752-761, June 1990.

[MONT90] Montgomery, B.L. and B.V.K. Vijaya Kumar, "Systematic Random Error Correcting & All Unidirectional Error Detecting Codes", *IEEETC* **39**:836-840, June 1990.

[NICO86] Nicolaidis, M. and B. Courtois, "Design of NMOS Strongly Fault Secure Circuits Using Unidirectional Error Detecting Codes," *FTCS*, July 1986, pp. 22-27.

[NIKO86] Nikolos, D., N. Gaitanis, and G. Philokyprou, "Systematic t-Error Correcting / All Unidirectional Error Detecting Codes," *IEEETC* **35**:394-402, May 1986.

[PARH73] Parhami, B. and A. Avizienis, "Application of Arithmetic Error Codes for Checking of Mass Memories," *FTCS*, June 1973, pp. 47-51.

[PARH78] Parhami, B. and A. Avizienis, "Detection of Storage Errors in Mass Memories Using Low-Cost Arithmetic Error Codes," *IEEETC* **27**:302-308, 1978.

[PARH89d] Parhami, B., "A Data-Driven Dependability Assurance Scheme with Applications to Data and Design Diversity," *Proc. of the Int'l Conf. Dependable Computing for Critical Applications*, Aug. 1989, pp. 105-112.

[PRAD80] Pradhan, D.K., "A New Class of Error-Correcting/Detecting Codes for Fault-Tolerant Computer Applications," *IEEETC* **29**:471-481, June 1980.

[SMIT84] Smith, J.E., "On Separable Unordered Codes," *IEEETC* **33**:741-743, Aug. 1984.

[TAOD88] Tao, D.L., C.R.P. Hartman, and P.K. Lala, "An Efficient Class of Unidirectional Error Detecting/ Correcting Codes," *IEEETC* **37**:879-882, July 1988.

[VENK89a] Venkatesan, R., "t-Error Correcting / d-Error Detecting / All Unidirectional Error Detecting Codes," *Proc. of the IEEE Pacific Rim Conf. on Communications, Computers & Signal Processing*, June 1989, pp. 412-415.

[WAKE75] Wakerly, J.F., "Detection of Multiple Unidirectional Errors Using Low-Cost Arithmetic Codes," *IEEETC* **24**:210-212, 1975.