Technical Report 2

CODES AND CODING CIRCUITRY FOR AUTOMATIC ERROR CORRECTION WITHIN DIGITAL SYSTEMS

Prepared for:

JET PROPULSION LABORATORIES
4800 OAK GROVE DRIVE
PASADENA, CALIFORNIA

CONTRACT M-44501 UNDER
NASA CONTRACT NASw-6

By: *William H. Kautz*

STANFORD RESEARCH INSTITUTE

MENLO PARK, CALIFORNIA

*SRI

STANFORD RESEARCH INSTITUTE
MENLO PARK, CALIFORNIA

January 1962

Technical Report 2

CODES AND CODING CIRCUITRY FOR AUTOMATIC ERROR CORRECTION WITHIN DIGITAL SYSTEMS

Prepared for:

JET PROPULSION LABORATORIES
4800 OAK GROVE DRIVE
PASADENA, CALIFORNIA

CONTRACT M-44501 UNDER
NASA CONTRACT NASw-6

By: William H. Kautz

SRI Project No. 3196

Approved:

[illegible]
J. R. ANDERSON, MANAGER COMPUTER TECHNIQUES LABORATORY

[illegible]
JERRE D. NOE, DIRECTOR ENGINEERING SCIENCES DIVISION

Copy No. 101

ABSTRACT

27050

Error-correcting codes for use in communication channels have been known for many years. Recently discovered code families have enabled economical encoding and decoding equipment to be designed for use with even very noisy channels. These new codes provide for the automatic correction of several independent errors in a block of transmitted digits, or of bursts of errors in successive digits.

Within a digital computer the "channel" becomes the collection of data paths and storage locations embodied in index, data, and control registers and in the memories and associated input-output transfer links. An attempt to apply existing error-checking codes to this type of channel reveals that these known codes are not necessarily optimal, because (1) appropriate measures of efficiency may differ from those customary in the traditional communications channels, (2) the most probable types of errors (noise) may not be the same, (3) one must also take into account the possibility of errors due to faults in the decoding logic (corrector) itself, and (4) if possible, the codes should be compatible in some sense with arithmetic and other related operations.

In this report we discuss the efficiency, cost, and optimality of codes for use within digital systems, and evaluate to what extent known code families can be applied to automatic error correction. To better satisfy these criteria in the most important remaining cases, some new codes are then described. Particular attention is devoted to error correctors which are "fault-masked"—*i.e.*, whose performance is insensitive to single isolated circuit faults. Also discussed are the simplifications possible if mere detection of an error is adequate. Several examples of encoding, corrector, and detector circuits are offered, exemplifying the use of both gate-type and branch-type logical elements. Author

CONTENTS

ILLUSTRATIONS

TABLES

I INTRODUCTION

The use of redundancy is now well known, if not well practiced, as a technique for improving the reliability of digital systems. It is recognized that one may apply the redundancy within the system at one or more levels—*i.e.*, by replicating the component, the logical element, the network, the subsystem, or even the entire system. While redundant design techniques are available for application at any level, the selection of the best techniques and the optimum levels for a given system, family of faults, and degree of protection is still a problem which is largely unsolved.

In general, redundancy may be applied to a digital network or system either by replicating (in some sense) the logical circuitry directly, or by applying the redundancy to the signals which pass through the system. For the former case of *circuit redundancy*, the extra equipment required depends critically on the particular operation being performed, and it is here that one may take advantage of any redundancy inherent in the operation itself, and in the logical circuitry selected to realize it. The price paid in the latter case of *signal redundancy* lies principally in the extra time required for operation in a serial system, or the larger number of iterated circuit stages required in a parallel configuration. Techniques for utilizing circuit redundancy for detecting correcting, and masking single faults are the subject of earlier reports under the present project.[1,2]*

In this report we will be concerned with signal redundancy techniques for application at the level of the network or subsystem, with emphasis on digital systems such as general-purpose-type computers. Both fault-detection and fault-correction techniques will be discussed. Also, both of the possible families of logical elements will be considered—gate-type elements, such as are conventionally realized in transistor-diode, magnetic, vacuum-tube, and parametron circuitry, and branch-type elements such as relay and switch contacts, cryotrons, and direct-coupled transistor logic—although we will not be overly concerned with detailed circuitry problems.

* References are listed at the end of the report.

The problem of applying redundancy to digital signals for the purpose of detecting or correcting errors due to isolated circuit faults is one of developing and selecting suitable error-detecting and error-correcting codes. Coding theory has by now evolved to the point where numerous codes have been developed for use in communication channels. In particular, many recently developed code families lead to economical encoding and decoding equipment for use with even very noisy channels.[3,4,5]

Within a digital computing system the "channel" becomes the complex of data paths and storage locations embodied in index, data, and control registers, and in the memories and associated input-output transfer links. One may even want to include those portions of the system which transform, analyze, and process the data, such as arithmetic units, code converters, comparitors, etc. (although circuit redundancy appears to be more appropriate for most of these operations). An attempt to apply known error-checking codes directly to this type of channel reveals that these codes are not always optimum or even appropriate, because the criterion of optimality is not the same in a computer as in communication channels. The major differences may be summarized as follows:

(1) The effect of circuit faults is often quite different from effect of noise in communication channels.

(2) The criterion of "efficiency" of a code depends upon circuit cost in a more complex way.

(3) One must also take into account the possibility of faults in the encoder and decoder themselves.

(4) The choice of code should be compatible with the other signals and operations (arithmetic, input and output codes, etc.) besides simple data transfer within the system.

In the following sections we discuss the notions of noise, efficiency, and (to a lesser extent) signal compatibility appropriate to a digital computing system, and investigate the types of codes which should be used. In some cases, known codes or simple modifications of them are suitable. In others, new codes are recommended and are partially developed. Particular attention is directed to some new code families developed at Stanford Research Institute for communications applications, but also well-suited to digital data processing.[5,6,7] A family of "low-density" parity-check codes are shown to be particularly suitable for error-correction in the parallel channel using gate-type logical elements.

Some attention is also devoted to decoders (error detectors and correctors) which are *single-fault-masked*--*i.e.*, their performance is insensitive to single isolated circuit faults. Examples of such decoders are offered.

For most of the ensuing discussion, we assume no specialized knowledge of coding theory, beyond a general appreciation of its objectives and a few of its most elementary concepts—the notions of parity checking, check digits, and encoding and decoding processes, for example. By the same token, the codes to be used are merely described, and not completely derived unless they are new. The reader is referred to textbooks and other literature for these derivations, proofs of optimality, and detailed circuit alternatives for the encoding and decoding equipment.[3]

Principal emphasis is placed on parity-check codes for detecting or correcting isolated *single*-digit errors in blocks of fixed length, with only secondary consideration of multiple errors, bursts, and so-called non-systematic (*i.e.*, non-parity-check) codes. The literature may be consulted for available information on these extensions, and on others such as recurrent (non-block) codes, codes for the erasure channel, and non-binary codes.[3]

Codes for checking simple serial and parallel data transfer and storage operations are discussed in the next two sections. This treatment is primarily a survey and disclosure of what is known and applicable to digital computing systems, but includes some discussion of "low-density" codes which are particularly suited to parallel checking with conventional digital circuitry. Section IV investigates the savings possible when the assumed faults cause binary errors in only one direction—*e.g.*, $1 \rightarrow 0$ errors but not $0 \rightarrow 1$ errors. Some special codes for checking arithmetic operations are presented in Sec. V. Section VI describes codes which are partially serial and partially parallel, introduced with the hope of overcoming some of the limitations of both. Finally, we discuss the masking of faults within the encoding and decoding equipment, and make some partial comparisons of the various codes which have been considered No comparison is attempted between these codes and other redundancy techniques such as circuit redundancy. Such comparisons are known to be critically dependent on the particular computer or system to which the redundancy is to be applied. These considerations and others on the integrated use of redundancy in digital systems form the subject of other effort under the present research project.

The discussion of each code is accompanied where possible with examples to illustrate the type and complexity of circuitry required for encoding and decoding.

II SERIAL CHECKING OF DATA TRANSFER AND STORAGE

Serial data handling will be treated first, because of its greater simplicity, and as a means for introducing some necessary terminology. The more popular gate-type logical elements will be assumed for the time being.

A. CYCLIC CODES

Serial processing of data is characterized by longer operation times but by a considerable saving in the amount of required equipment whenever the operation to be performed is (1) iterative—*i.e.*, when it is essentially the same on each successive data digit of the work or block—and (2) unilateral—*i.e.*, when the interdigital logical dependencies extend in only one direction, forward or backward through the word, but not both.

The first error-correcting codes developed for blocks of binary digits satisfied neither of these properties, so that the encoders and decoders for these codes were forced to be complicated parallel switching circuits.[8,9,10] This complexity effectively prevented widespread use of these codes in digital communication, data processing, and recording systems, even though their advantages for combating noise and disturbances were well established and accepted.

More recently developed families of codes, called *cyclic* codes, satisfy both of the above conditions for serialization, and very economical circuit arrangements for serial error connection can be developed for most of them.[3,11] A wealth of underlying theory allows one to determine (1) which block lengths n can be most efficiently accommodated, (2) what redundancy levels are required, without actually executing the details of code design, and (3) the actual design of the encoder and decoder, with some latitude in the required logic to take into account preferred circuit arrangements. Other closely related code families allow one to correct not only single isolated digit errors, but multiple isolated errors, or (within certain limitations) bursts of errors in adjacent or nearby digit positions.

The reader is referred to the literature for the development and exposition of this theory, and the details of code design.[3] We will confine our attention to a few examples and some bounds on the redundancy required, to indicate the general complexity of the encoding and decoding equipment.

Any systematic code can be described by its *parity check matrix* P, each of whose n columns correspond to one of the n digits of the code, and each of whose k rows correspond to one of the k parity checks. The elements of the matrix are *0*'s and *1*'s, and the positions of the *1*'s in the ith row indicate which digit positions are involved in the ith parity check. Similarly, the positions of the *1*'s in jth column indicate to which parity checks the jth digit contributes. For example, for the conventional Hamming single-error-correcting code,[9] having $k = 3$ check digits and $n = 7$ digits total, the parity check matrix is

$$P = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} .$$

Thus, the first parity check is applied over Positions 1, 3, 5, and 7; the second over Positions 2, 3, 6, and 7; and the third over Positions 4, 5, 6, and 7. The first digit contributes to only the first parity check, etc. Any set of independent columns (*e.g.*, Columns 5, 6, and 7) may be selected to correspond to the check-digit positions in the word.

The fact that each column is different from all others is sufficient condition for any single-digit error to cause a unique pattern of parity check violations. Thus, an error in the third digit (Column 3) will cause the first and second parity checks to be violated, and no other single error could have the same effect. Clearly, then, a matrix with k rows can have a number n of columns as large as $2^k - 1$, corresponding to the set of all k-digit binary numbers except 0 0 ... 0. Thus, the minimum number k of check digits needed for a block of m data digits is the smallest number satisfying $2^k - 1 \geq m + k$, or $k \approx \log_2[m + \log_2(m)] \approx \log_2(m)$. The number of check digits required increases rather slowly with m.

Up to this point, any ordering of the rows and columns is equally preferable. To achieve some degree of uniformity in the parity checking operation over successive digits, however, we might order the columns so that *each row is a cyclically shifted version of the previous row.* For example, in the above matrix P, we might select the column re-ordering as follows:

$$P_c = \left[\begin{array}{cccc:ccc} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \end{array}\right] .$$

Clearly, the second and third rows are (cyclic) horizontal unit displacements of the first and second rows respectively. In this case, an additional feature may be noted; namely that the succession of digits in the first row (and therefore all rows) obeys the simple third-order recurrence relation

$$a_i = a_{i-2} \oplus a_{i-3} .$$

That is, each digit is the exclusive-OR (modulo-2) sum of the two digits two and three positions to its left. The last three columns might logically be selected to correspond to the three check digits. This recurrence property suggests that a simple encoding circuit may be possible, in which the three check digits are attached to the end of a four-data-digit sequence, and for which the calculation of each check digit involves only a third-order dependence over suitably combined past digits of the sequence, in accordance with the recurrence relation.

B. ENCODER AND DECODER

The complete encoder, which generates the redundant digits, is shown in Fig. 1, and operates as follows:

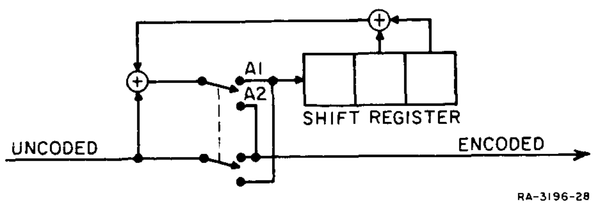

FIG. 1 SERIAL ENCODER FOR A CYCLIC SINGLE-ERROR-CORRECTING PARITY-CHECK CODE

(1) *Mode A1* (switch up): The 4 (generally, $m = n - k$) data digits are applied in

time succession to the 3-stage (generally, k-stage) feedback shift register, as well as to the output. The register operates in accordance with the above recurrence relation.

(2) *Mode A2* (switch down): For the next 3 (generally, k) digit times, the register is flushed into the output through the feedback logic. The input (presumably *0*'s) fills the register during this time.

The string of $m + k = n$ output digits constitute the encoded (redundant) data word, with the k check digits at the end.

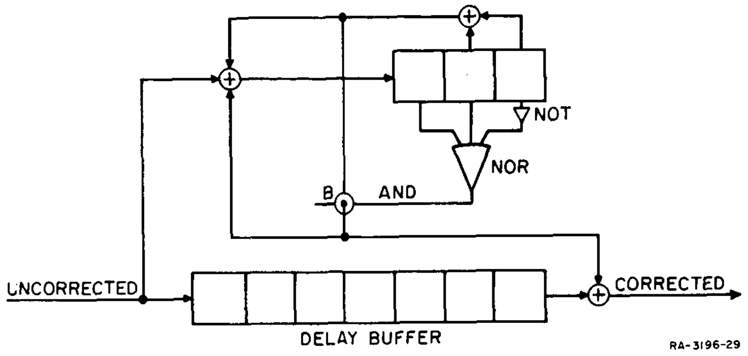

FIG. 2 SERIAL DECODER (CORRECTOR) FOR A CYCLIC SINGLE-ERROR-CORRECTING PARITY-CHECK CODE

The decoder performs automatic correction, and is shown in Fig. 2. It operates as follows:

(1) *Mode A*—All 7 (generally, n) digits are fed in time-sequence into the feedback shift register, as well as into a 7-digit (generally, n-digit) delay buffer. The register operates in accordance with the same recurrence relation as was used for the encoder.

(2) *Mode B*—For the next 7 (generally, n) digit times, the register operates by itself with *0* data input, while the buffer is flushed into the output; if and when the register reaches the state 0 0 1 (generally, 0 0 ... 0 1), as sensed by the NOR-gate, a *1* is fed to the final exclusive-OR gate to complement the corresponding output digit.

The purpose of Mode A is to perform the same parity checks as were performed in the encoder, leaving all *0*'s in the register if the received data word contains no error, and otherwise leaving a "corrector number" in the register indicative of the time-position of any single error in the data word. This position corresponds exactly to the number of shifts required in Mode B to return the register to the state 0 0 1 (generally, 0 0 ... 0 1). The "complement" signal is shown also fed back through the register logic, a connection which returns the register to its all-0 state, and offers a partial check on the success of the correction operation. Register feedback logic is known for all values of k which could conceivably find application.[11]

Serial encoders and decoders should find their major application in situations in which errors arise while the data is in parallel form, since all but short intermittent faults in a serial data channel result in burst or saturation (all *0* or all *1*) type errors, not single isolated errors.

We have not attempted in this description to completely derive and justify the circuits, which are adequately described elsewhere, but offer them as examples of a now well-established technique in modern code design. They illustrate the type and amount of digital equipment required for serial error correction. It is not difficult to show that the amount of circuitry required is either minimum or very close to the minimum needed for any code having the same error-correcting ability. Other equally economical circuit forms are possible. One alternative arrangement for the decoder combines the k-digit shift register with that portion of the buffer corresponding to the k check digits, at the cost of an increase in the timing circuitry (not specifically shown), and a reversal of the order of the data and check digits (which complicates the encoder) to put the latter ahead of the former in time. This arrangement would be particularly appropriate when the buffer can be identified with one of the existing registers of the digital system.

C. OTHER CYCLIC CODES

These techniques of automatic error correction in the serial transfer of information between storage locations in a digital system can be extended to multiple error correction,[3,4,12] both for the case when the multiple errors may occur independently within the block of n digits, and when they are known to be limited to a digit range or "burst" whose length is substantially less than n. Burst errors would typically arise from faults which affect a register or memory location over a range wider than a single binary digit. In both cases, the encoder circuitry has the same form, differing only in the selection of those register stages which are fed back to the input exclusive-OR-gate, and in the required length k of the register. The decoding circuitry also has the same form except that the combinational logic attached to the register (corresponding to the NOR-gate in Fig. 2) may be somewhat more complex.

Optimal or near-optimal codes are known for a wide range of values of n, for any number e of isolated errors, and for burst length b up to about ten. The number of redundant digits required is, very roughly,

$$k \approx e \log_2(n)$$

for isolated errors and

$$k \approx \log_2(n) + b$$

for burst errors, provided k and n are not too small. Note that comparatively less redundancy is required for the correction of a given number of errors if these errors are confined to a burst.

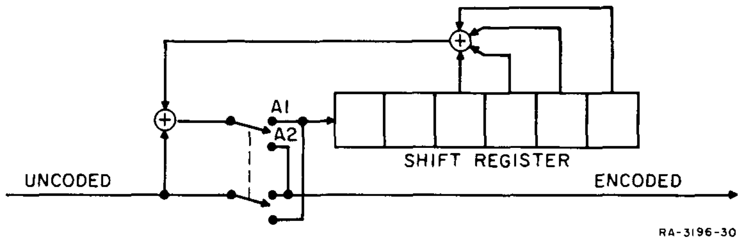

FIG. 3 SERIAL ENCODER FOR A CYCLIC BURST-THREE-CORRECTING CODE

Figures 3 and 4 show an encoder and decoder, respectively, for a code capable of correcting any error burst of width up to $b = 3$ in a block of $n = 15$ digits.[4] Of these digits, $k = 6$ are check digits, and the length of the register is k, leaving $m = n - k = 9$ data digits.

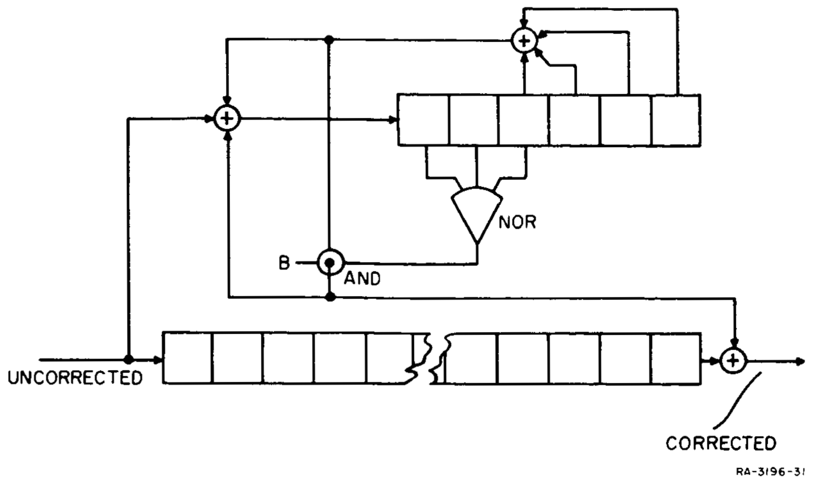

FIG. 4 SERIAL DECODER FOR A CYCLIC BURST-THREE-CORRECTING CODE

D. ERROR DETECTION

If simple error detection is desired, only a single redundant (parity) digit is required: $k = 1$, and $n = m + 1$. The single register stage with exclusive-OR feedback is logically equivalent to a complementing flip-flop. The encoder and decoder are shown in Fig. 5. The presence of an error is indicated in the decoder by a "*1*" output *e* from the flip-flop after the *n*th digit is received.

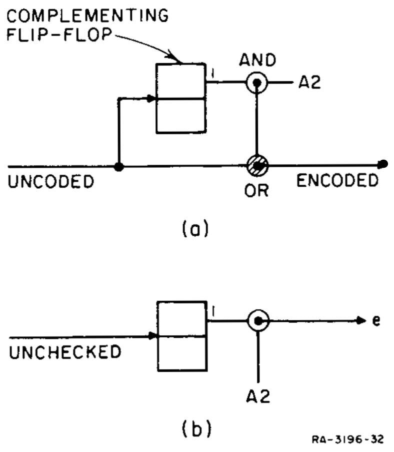

FIG. 5 SERIAL ENCODER AND DECODER (DETECTOR) FOR A SINGLE-ERROR-DETECTING PARITY-CHECK CODE

If the single-error-correcting code is to be used instead for double-error detection, the same encoder is employed as for single-error correction, but the decoder may now be simplified. The buffer register and output gate are unnecessary, and the NOR and AND gates may be replaced by a simple OR-gate over the stages of the register to indicate an error (non-zero contents) at the end of Mode A. Mode B is not needed.

III PARALLEL CHECKING OF DATA TRANSFER AND STORAGE

Purely parallel data handling forces the decoder to be a rather complex combinational switching circuit. Unfortunately, the so-called "optimal" codes minimize the required number k of check digits rather than the complexity of a parallel decoder. We will now describe some "low-density" codes which require a value of k somewhat larger than the minimum, but lead to decoding logic that is more competitive with the amount of circuitry needed for serial decoding.*

We first observe that any serial circuit may be converted into a parallel, iterative, combinational circuit by means of a simple transformation.[14] For example, the encoder of Fig. 1 is first redrawn with the register stages separated from the logic, as in Fig. 6(a). The logic cell is then iterated seven times, as in Fig. 6(b), the switches being absorbed into the wiring in position A1 for the first four cells, and in position A2 for the last three cells, corresponding to the two modes of serial operation described previously. After some obvious simplifications, this circuit is a valid parallel realization of the encoder for a seven-digit cyclic single-error-correcting code.

The same transformation applied to the decoder of Fig. 2 leads to the iterative cell shown in Fig. 7. (For convenience, Cells 8 to 14 have been placed underneath Cells 1 to 7, and the outputs a', b', and c' from Cell 7 should be returned to the inputs a, b, and c on Cell 8.) However, the circuitry displayed in this figure can be simplified considerably by starting with another code, which will now be described.

A. LOW-DENSITY CODES

The decoding logic required for any systematic code can be attributed to (1) the execution of the set of k parity check operations defined by the parity check matrix P, in order to determine the k-digit "corrector" number (a non-zero value of this number indicates the location of the error);

* Some low density codes were previously introduced by Gallager.[13]

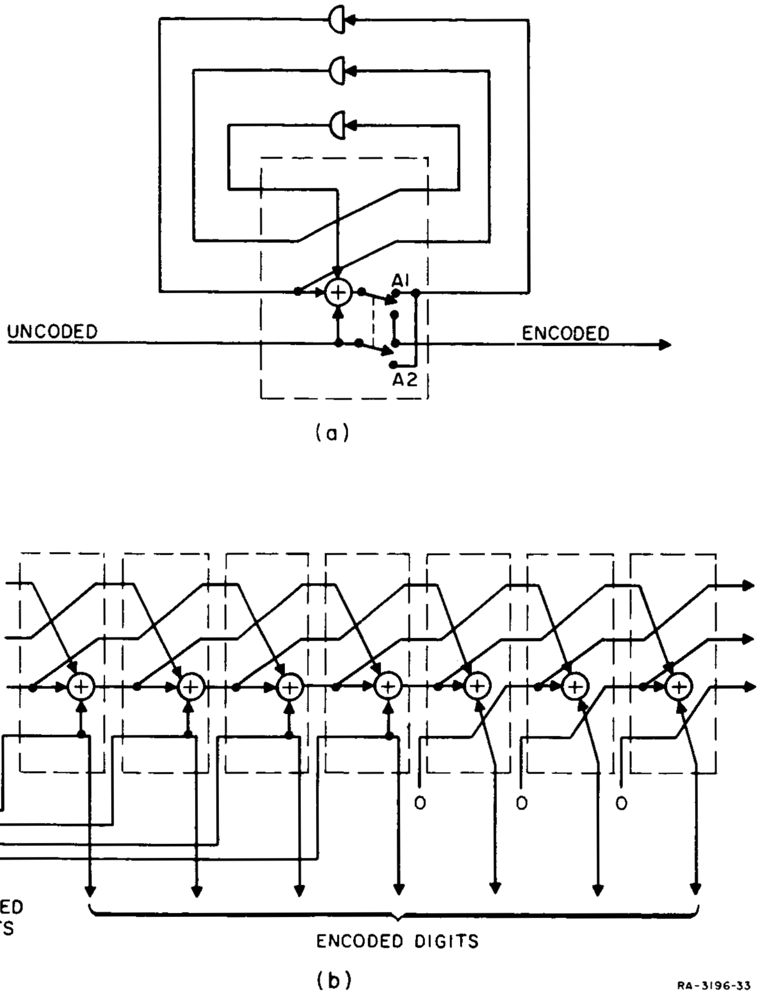

FIG. 6 ITERATIVE PARALLEL REALIZATION OF THE SERIAL ENCODER OF FIG. 1

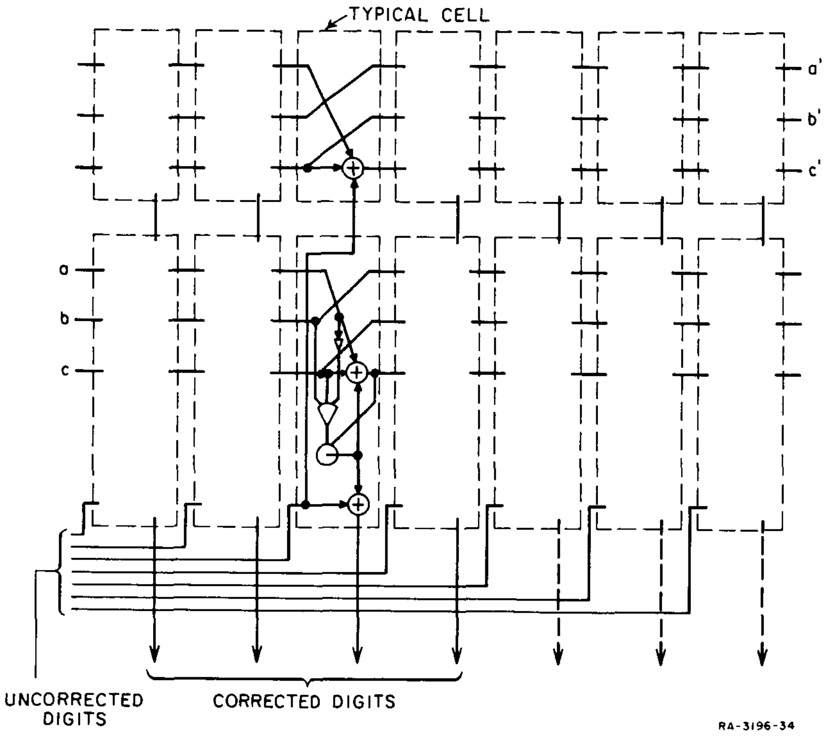

FIG. 7 ITERATIVE PARALLEL REALIZATION OF THE SERIAL DECODER OF FIG. 2

(2) the decoding of the corrector-number into separate signal lines, one for the correction of each of the $n - k = m$ data digits;[*] and (3) the actual correction of these digits.

Taking these factors in reverse order, we may observe that the cost of (3) is fixed at m two-input exclusive-OR-gates, with no immediate possibility for direct simplification. The cost of (2) is the cost of a k-input, m-output decoding tree; this cost increases with m at a slightly less than linear rate.

The major contribution to the total cost of the corrector is the set of k multi-input exclusive-OR-gates for (1). With most known types of digital logic devices, the cost of a μ-input exclusive-OR-gate increases with μ much faster than linearly. Consequently, we must be prepared to reduce the number μ of digit positions over which each of the k parity checks are performed, even if k must be increased somewhat to do this. In terms of the parity check matrix, the conditions for this type of single-error correction may be expressed as follows:

(1) Each of the k rows has no more than a fixed number μ of *1*'s.

(2) Each of the $n = m + k$ columns is different, and contains at least one *1*.

(3) m is maximum for given k and μ.

The columns of such a matrix should therefore be formed from a set of k-digit binary numbers, each of which has a minimum number of *1*'s: the k numbers having a single *1* (these being most logically assigned as check digits, in the last k columns), and as many of the $\binom{k}{2}$ binary numbers having two *1*'s, $\binom{k}{3}$ numbers having three *1*'s, etc., as are needed and as are consistent with Condition (1). Since the first m columns of P can contain no more than $k(\mu - 1)$ *1*'s in all, with at least two *1*'s per column, the number of columns for data digits is immediately bounded by

$$m \leq \frac{k(\mu - 1)}{2} \quad .$$

[*] We have assumed through this section that only the *data* digits are to be corrected. If the entire redundant code word is to be reused, so that the check digits need to be corrected as well, then the cost of the decoder will increase by a fraction very nearly equal to k/m applied to portions (2) and (3) of the decoder.

We will now show by construction that this bound can actually be achieved, for all but very small values of m and k, except for the fractional remainder ½ which results from the division by 2 in case both k and $\mu - 1$ are odd:

$$m = \left[\frac{k(\mu - 1)}{2}\right] , \quad k \geq \mu \quad .$$

(The brackets denote the integer part of the quantity within.)

For $\mu = 3$, then $m \leq k$, and this bound is achievable provided only that at least k different columns with just two *1*'s can be formed:

$$\binom{k}{2} \geq k$$

i.e., provided $k \geq 3$. In fact, the matrix whose first k columns have *1*'s in Columns j and $j + 1$ (mod k) and Row j ($j \leq k$) will always be adequate. *E.g.*, for $k = 5$, we may use the parity check matrix

$$P = \left[\begin{array}{ccccc:ccccc} 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \end{array}\right]$$

Thus, for $\mu = 3$ and $k \geqq 3$, $m = k$ is optimum.

For any value of $\mu \geq 2$, there exist $k(\mu - 1)/2$ columns having two *1*'s each whenever

$$\binom{k}{2} \geq \frac{k(\mu - 1)}{2}$$

or $k \geq \mu$, and the bound is achievable. For example, for $\mu = 4$ and $k = 5$, then $m = 7$:

$$P = \left[\begin{array}{ccccccc|ccccc} 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \end{array}\right]$$

and for $\mu = 5$ and $k = 5$, then $m = 10$:

$$P = \left[\begin{array}{cccccccccc|ccccc} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 \end{array}\right] .$$

Note that when k and $\mu - 1$ are both odd, one row at P need have only $\mu - 1$ *1*'s.

We may now invert this expression to obtain the required number k of redundant digits in terms of m, as follows:

$$\begin{array}{lclcll} \mu = 2 & : & k & = & 2m & \text{for} \quad m \geq 1 \\ \mu = 3 & : & k & = & m & \text{for} \quad m \geq 3 \\ \mu = 4 & : & k & = & \left[\dfrac{2m + 2}{3}\right] & \text{for} \quad m \geq 5 \\ \mu = 5 & : & k & = & \left[\dfrac{m + 1}{2}\right] & \text{for} \quad m \geq 9 \end{array} .$$

(Matrices with more than two *1*'s per column may be formed to cover values of m below these limits, but these cases are probably not of particular interest.) For larger values of μ, the bound may be similarly achieved:

$$m = \left[\frac{k(\mu - 1)}{2}\right] , \qquad k \geq \mu .$$

The entire decoder now requires:[*]

---

[*] It is barely possible that a less costly decoder could be achieved by combining the three portions. Except for a few degenerate cases (such as $m = 1$, $k = 2$), no economies appear to result from such merging, aside from some small savings sometimes possible by decomposing the bank of μ-input exclusive-OR-gates into two or more levels of gates with fewer inputs, but with some sharing in the earliest levels.

k μ-input exclusive-OR-gates, for determination of the corrector number

m 2-input AND-gates, for decoding the corrector number, and

m 2-input exclusive-OR-gates, for the final correction.

The portion of the circuit which generates each corrected output digit is illustrated in Fig. 8. On a per-data-digit basis, the first contribution is the only variable part. Here, $k \approx 2m/(\mu - 1)$. Thus, if the cost of a μ-input exclusive-OR gate increases with μ according to a linear proportion,

$$\text{cost} \sim \frac{\mu - 1}{2}$$

then all values of μ are equally preferable for the decoder itself. A lower rate of increase would favor a higher value of μ, while a higher rate of increase favors a lower value of μ. The latter alternative is probably more realistic, so that the choice is really between:

$\mu = 2$: two 2-input exclusive-OR-gates per data digit or

$\mu = 3$: one 3-input exclusive-OR-gate per data digit or

$\mu = 4$: about two-thirds of a 4-input exclusive-OR-gate per data digit

for the only part of the decoder whose complexity depends on the value of μ.

The circuit of Fig. 9 displays the decoder for the example of P given above for $\mu = 3$, $k = 5$, $m = 5$.

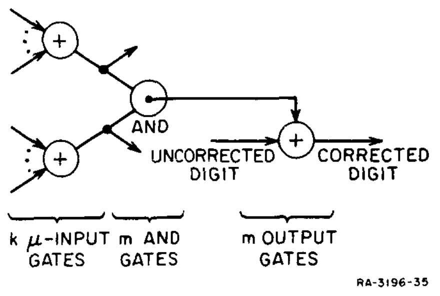

FIG. 8 A ONE-DIGIT PORTION OF A PARALLEL DECODER FOR A SINGLE-ERROR-CORRECTING, LOW-DENSITY, PARITY-CHECK CODE

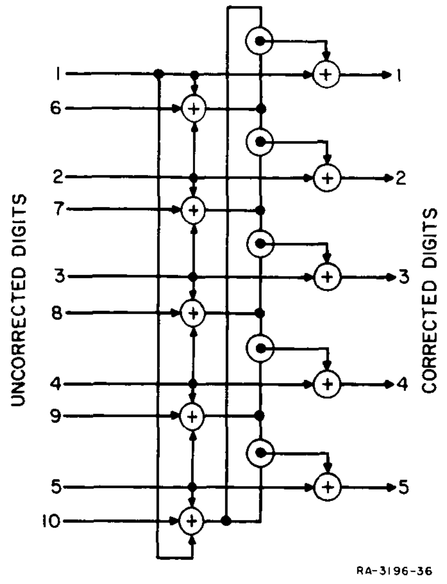

FIG. 9 DECODER FOR A LOW-DENSITY SINGLE-ERROR-CORRECTING PARITY-CHECK CODE, $\mu = 3, m = 5, k = 5$

B. HAMMING DECODER

The choice of the optimum value of μ, hence k, also depends upon the amount of register, memory, and other "channel" equipment whose size will be increased as k is increased. As the amount of this equipment becomes large in proportion to the cost of the encoder and decoder, the optimum value of μ will tend to increase, up to the limit set by the Hamming single-error-correcting codes introduced earlier for serial single-error correction. The Hamming codes require a number k of check digits such that

$$m \leq 2^k - k - 1$$

or, inverted,

$$k = 1 + \{\log_2 [m + 1 + (\log_2 m)]\} \quad .$$

The Hamming decoder differs from the decoder for the low-density codes, however. For the *lossless* case, where the maximum number of data digits are used for a fixed number of check digits, then

$$n = m + k = 2^k - 1$$

and each parity check extends over about half of the columns: $\mu = 2^{k-1}$. The decoding tree, the second portion of the decoder, is an almost complete one.[15] It may consist of either m k-input AND-gates, or $\rho(n) - (k+1)$ 2-input AND-gates, according to a well-known structure,* where

$$\rho(n) = \rho\left(\left[\frac{n}{2}\right]\right) + \rho\left(n - \left[\frac{n}{2}\right]\right) + 2^n$$

and

$$\rho(2) = 4 \quad , \quad \rho(1) = 0 \quad .$$

As the value of m is reduced below $2^k - 1 - k$, those columns of P having the largest number of *1*'s may be deleted first, to minimize the number of *1*'s per row in the remaining portion of the matrix. Table I lists for each value of m up to 25 the largest number μ_0 of *1*'s per row in P, and the number ρ_0 of 2-input AND-gates required for the decoding tree, which is now much less than complete and is therefore somewhat less costly. The complete cost of the Hamming decoder is therefore about

$\underline{k}$	μ_0-input exclusive-OR-gates
ρ_0	2-input AND-gates
$\underline{m}$	2-input exclusive-OR-gates.

The relatively large cost of this decoder clearly favors the low-density codes, unless the data storage locations and data paths dictate the use of a small or minimal number k of check digits.

Table I

PARAMETERS OF COMPLEXITY OF PARALLEL ENCODERS AND DECODERS FOR THE HAMMING CODE

m	k	μ_0	ρ_0	m	k	μ_0	ρ_0
2	3	3	3	14	5	8	29
3	3	3	5	15	5	8	30
4	3	4	7	16	5	9	31
5	4	4	11	17	5	10	32
6	4	4	14	18	5	10	33
7	4	5	15	19	5	11	34
8	4	6	16	20	5	11	36
9	4	7	17	21	5	12	37
10	4	7	18	22	5	13	38
11	4	8	19	23	5	14	39
12	5	7	27	24	5	15	40
13	5	7	28	25	5	15	41

C. ENCODERS

The encoder for a low-density code of weight μ consists of just k $(\mu - 1)$-input exclusive-OR-gates, one gate corresponding to each row of the parity check matrix. Clearly, no reduction in this total number of gates is possible. Note that when $\mu = 2$, these gates reduce to direct connections. The cost of an encoder for a Hamming code is the same, with

* The number $(k+1)$ is subtracted for the unneeded check-digit correctors and for the "no-error" condition.

μ replaced by μ_0, whose value is given in Table I. The complexity of the encoder therefore varies nearly in proportion to that of the decoder, so that the cost of the latter may be taken as an indication of the complexity of both.

D. ERROR DETECTION

For single-error detection instead of single-error correction, a single parity check over all digits gives the least redundancy, and requires an m-input and an $m + 1$-input exclusive-OR-gate for the encoder and decoder, respectively.

Low-density codes offer the possibility of applying k parity checks over just μ digits at a time. Each of the checks covers up to $\mu - 1$ of the data digits, therefore, so that the value of k required is roughly $m/(\mu - 1)$. More precisely, since the division may not come out even,

$$k = 1 + \left[\frac{m - 1}{\mu - 1}\right] .$$

The decoder then consists of k μ-input exclusive-OR-gates feeding a single k-input OR-gate which provides the error output e. The encoder consists of k $(\mu - 1)$-input exclusive-OR-gates, as before.

E. THRESHOLD ELEMENTS

If the encoder and decoder are to be constructed from threshold elements rather than AND-gates and OR-gates, an upper bound to the cost can be obtained by a direct conversion of the gate circuits derived above. A μ-input exclusive-OR gate can be replaced by $1 + [\log_2 \mu]$ threshold elements.[16] An AND-gate driving a 2-input exclusive-OR-gate can be replaced by just two threshold elements. Thus, for single-error correction using the Hamming code, the encoder has a cost

$$\tau_e \leq k + k[\log_2 (\mu_0 - 1)]$$

and the decoder has a cost

$$\tau_d \leq 2m + k + k[\log_2 \mu_0] .$$

For the lossless case, where $\mu_0 = 2^{k-1}$, these bounds become

$$\tau_e \leq k^2 - k$$

and

$$\tau_d \leq 2m + k^2 = 2^{k+1} + k^2 - 2k + 2$$

respectively. For example, for $m = 4$, then $k = 3$, and no more than 17 elements are required for the decoder. A decoding network utilizing 16 elements* is shown in Fig. 10. For $m = 1$, then $k = 2$, and a single majority gate ($\tau = 1$) will suffice.

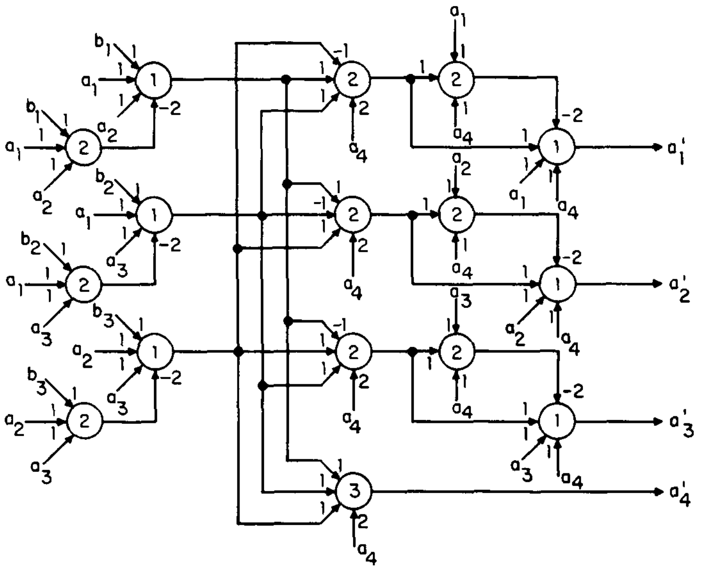

FIG. 10 PARALLEL DECODER FOR 7-DIGIT HAMMING CODE USING THRESHOLD ELEMENTS

For low-density codes, the same conversion of gates gives costs $\tau_d \leq 4m$ and $\tau_e \leq 3m$, for $\mu = 3$ and $\mu = 4$. For $\mu = 2$, $\tau_d = m$ simple majority gates are adequate for the decoder, and no gates for the encoder ($\tau_e = 0$). These latter values are obviously minimal.

For single-error detection the conversion gives $\tau_e = 1 + [\log_2 m]$ and $\tau_d = 1 + [\log_2 (m + 1)]$.

* Another circuit requiring only 15 elements has recently been found.

F. BRANCH-TYPE ELEMENTS

If parallel error correction is to be accomplished with branch-type elements (such as switch contacts) instead of gate-type elements, circuits based on either the low density or the Hamming codes may be used. Parallel single-error-correcting encoders and decoders for the low-density codes are shown in Fig. 11(a), (b), and (c), for the cases $\mu = 2$, $\mu = 3$, and $\mu = 4$, respectively. In each case, the encoder consists of k circuits like the one shown, and the decoder consists of m circuits like the one shown. The digits of the encoded word are designated (a_1, a_2, ... a_m, b_1, b_2, ... b_k), and the corrected word (a_1', a_2', ... a_m'). The labeling of elements corresponds to the location of *1*'s in the parity check matrix P.

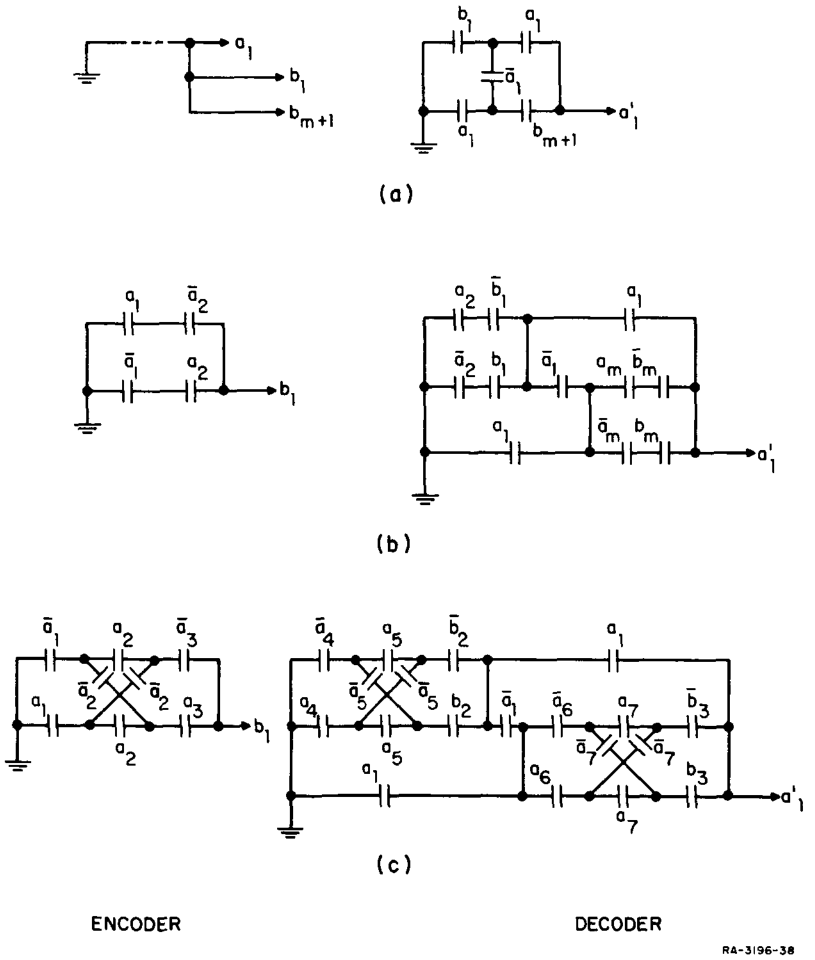

FIG. 11 PARALLEL SINGLE-ERROR-CORRECTING ENCODERS AND DECODERS FOR THE LOW-DENSITY CODES FOR (a) $\mu = 2$, (b) $\mu = 3$, AND (c) $\mu = 4$, USING BRANCH-TYPE ELEMENTS

The total numbers c_d and c_e of branch-type elements required for the decoder and encoder, respectively, are seen to be as follows:

$$\mu = 2 \quad : \quad c_d = 5m \quad \text{and} \quad c_e = 0$$

$$\mu = 3 \quad : \quad c_d = 11m \quad \text{and} \quad c_e = 4m$$

$$\mu = 4 \quad : \quad c_d = 26m \quad \text{and} \quad c_e \approx \frac{16}{3}\, m$$

$$\mu = 5 \quad : \quad c_d = 53m \quad \text{and} \quad c_e \approx 6m \quad .$$

The last values are obtained by a direct extension of this circuit form. These circuits are not known to be minimal.

Hamming-code encoder and decoder circuits for $n = 3$ and $n = 7$ are shown in Fig. 12(a) and (b). The encoders cost 0 and 24 elements, and the decoders cost 5 and 128 elements, for $n = 3$ and $n = 7$, respectively.

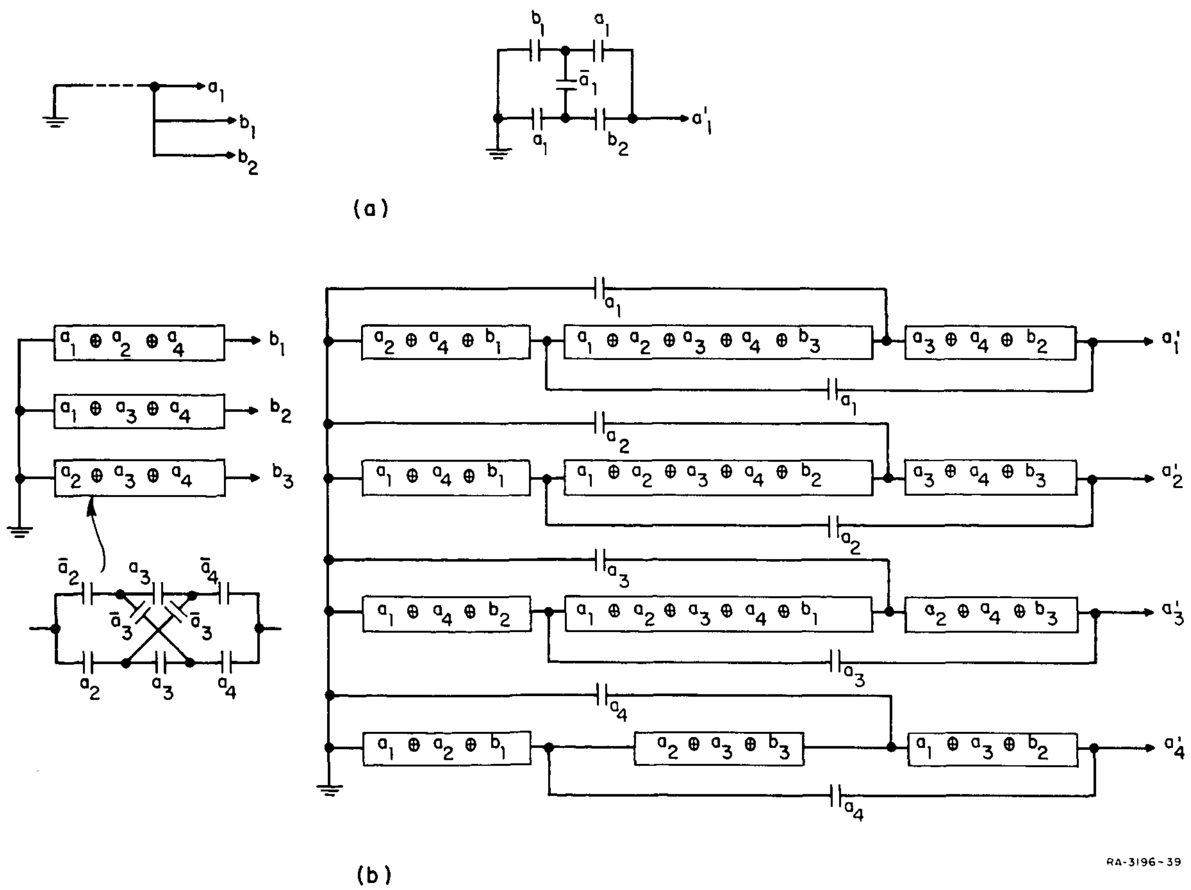

FIG. 12 ENCODER AND DECODER CIRCUITS FOR THE HAMMING CODES FOR (a) n = 3 AND (b) n = 7, USING BRANCH-TYPE ELEMENTS

For detection only, a simple parity check needs only $4m - 4$ and $4m$ elements in the encoder and decoder, respectively. These values can be extended directly to the low-density error-detecting codes, from the knowledge that an r-digit parity circuit requires $4r - 4$ branch-type elements:[12]

$$c_e \approx \frac{4(\mu - 1)(\mu - 2)}{\mu^2} m$$

$$c_d \approx \frac{4(\mu - 1)^2}{\mu^2} m \quad .$$

IV ASYMMETRIC DIGIT ERRORS

By analogy with fault-checked switching circuits, it may be possible to reduce the required amount of redundant circuitry if it can be assumed that all errors are due only to the *loss* of *1*'s—*i.e.*, to just $1 \rightarrow 0$ errors, rather than to both $0 \rightarrow 1$ and $1 \rightarrow 0$ errors. Codes for this purpose are called *1-error* correcting codes, and are said to be adapted to the binary *asymmetric* channel.[18] Present interest in them stems from a presumed dissymmetry in the nature of faults in certain types of devices such as magnetic cores, diodes, some transistor types, etc.

Unfortunately, only a few code families of this type are known, and most of these codes are either no more efficient than their symmetric counterparts, or require excessive decoding circuitry, or both. We will survey briefly those 1-error correcting codes which may have some practical advantages, and evaluate the encoding and decoding circuitry for each.

A. USE OF SYMMETRIC-CHANNEL CODES

It should first be pointed out that any of the codes described above for serial or parallel error correction or detection may be *used* for *1*-error correction or detection. In the case of *1*-error correction, a small saving in the cost of the decoder is possible because of the simple type of correction that need be made: one or more erroneous *0*'s should be changed back to *1*'s. With gate-type elements, the output exclusive-OR-gate(s) may therefore be replaced by inclusive-OR-gate(s).

In parallel, branch-type, single-*1*-error correctors, some saving is usually possible, as follows:

Hamming code, $n = 3$:	still 5 elements
Hamming code, $n = 7$:	70 instead of 128 elements
Low-density codes, $\mu = 2$:	still $5m$ elements
$\mu = 3$:	$9m$ instead of $11m$ elements
$\mu = 4$:	$25m$ instead of $26m$ elements
$\mu = 5$:	$49m$ instead of $53m$ elements.

With threshold elements, the parallel decoders derived previously now require *one less element* per output for both the low-density codes and the Hamming code.

B. DUPLICATION

In the asymmetric case, duplication rather than triplication of individual digits of the data word is adequate for single *1*-error correction, since a pair of disagreeing digits could have arisen only from two *1*'s, not two *0*'s. Thus, $k = m$ check digits will allow all single (and many multiple) *1*-errors to be corrected. In the parallel case the encoder is now trivial, and the decoder consists of only k 2-input inclusive-OR-gates (or two series branch-type elements per output, or one threshold element per output). In the serial case, an m-digit or a *1*-digit delay is required for the encoder and decoder, depending on whether the set of redundant digits follows the data digits as a block or whether the redundant digits are interspersed individually. The rest of the serial decoder consists of a single 2-input inclusive-OR-gate for the final correction.

C. BERGER CODES

Any number of *1*-errors may be detected in a code whose m data digits are augmented with $k = 1 + [\log_2 m]$ check digits, the latter selected to be the binary representation of the number of *0*'s in the block of data digits. This code suggested by Berger[19] will also detect any number of *0*-errors, provided *1*-errors and *0*-errors are never mixed in the same block of digits. This code has been shown to require the least redundancy of any all-*0*-error or all-*1*-error-detecting *separable* codes;[20] that is, those codes for which the check digits may be specifically separated from the data digits. Its error-detecting ability rests on the fact that a binary number representation is a *weighted-digit* representation, so that any loss of *1*'s in the check-digit portion of the encoded word reduces the binary number so represented. Loss of *1*'s in the data-digit portion increases the number of *0*'s. Thus, either type of error creates a checking discrepancy in the same direction: check number < number of *0*'s in data digits. Similarly, any one or more *0*-errors will create a discrepancy in the opposite direction. Thus, no combination of *1*-errors or combination of *0*-errors can occur which will allow the check to remain satisfied.

The serial encoder and decoder shown in Fig. 13(a) and (b), respectively, are fairly obvious realizations of the counting operation, requiring in each case a set of k flip-flops which can both operate as a binary counter for the first m digits, and then be shifted out serially for the next k digits. The gate-type parallel versions are apparently best generated as k-cell iterative equivalents. Branch-type parallel networks are most naturally realized as reduced m-variable symmetric-function networks.[20] The number c of elements required for the encoder is shown as a function of m in Table II, and an example of an encoder for $m = 5$, $k = 3$, is shown in Fig. 14. The decoder network costs about 10% more than the encoder, and has an almost identical structure.

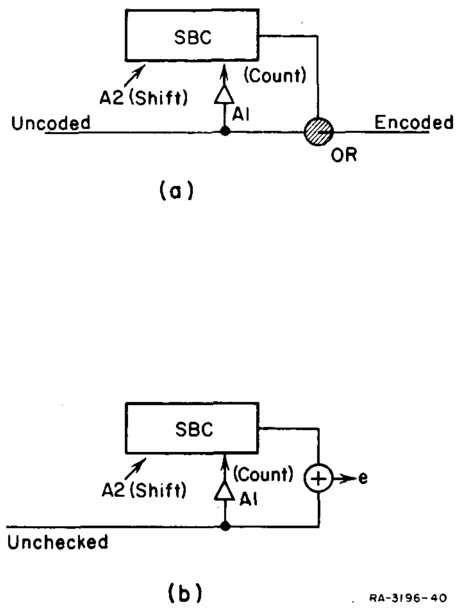

FIG. 13 SERIAL ENCODER (a) AND DECODER (b) FOR BERGER ALL-1-ERROR-DETECTING CODE

Table II

REQUIRED REDUNDANCY AND COST OF PARALLEL ENCODER FOR BERGER CODE USING BRANCH-TYPE ELEMENTS

m	k	c
2	2	6
3	2	13
4	3	29
5	3	48
6	3	68
7	3	88
8	4	123
9	4	166
10	4	209
11	4	252
12	4	296
13	4	340
14	4	384
15	4	428
23	5	1140
31	5	1876

D. NON-SEPARABLE CODES

A few codes for the asymmetric channel are available which are *non-separable*—that is, the codes do not allow a separation of the digits into data digits and check digits. In general, these codes do not even provide for a number of data-digit code combinations equal to a power of two. Typical of such codes is the family of *fixed-weight* codes.[20] In the most efficient case for all-*1*-error-detection, every valid code combination contains just $[n/2]$ *1*'s. The decoder is then expected to verify that a received code word is one of the $\binom{n}{[n/2]}$ words having just this number of *1*'s. A simple counter (parallel or serial) is adequate. These codes are sometimes used in conjunction with memories to provide more reliable operation of the access circuitry, but their use for data purposes is normally excluded because of their non-separable property. Also, the logical circuitry required for conversion between fixed-weight codes and more standard codes is quite complex.

A few efficient single *1*-error correcting codes are known, but these are also non-separable.[19]

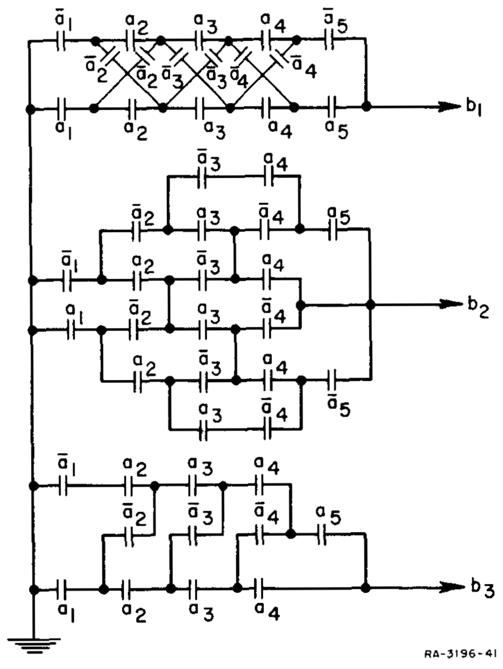

FIG. 14 ENCODER FOR BERGER ALL-1-ERROR DETECTING CODE USING BRANCH-TYPE ELEMENTS

V CODES FOR ARITHMETIC OPERATIONS

A family of simple codes developed by Brown finds a natural application in the detection and correction of single errors in data words subject to processing through an arithmetic unit.[6] As was the case with the serial use of single-error codes in a previous section, Brown's codes are most appropriate for checking errors which occurred during parallel rather than serial transfer and processing of data. (The encoding and decoding may be done either serially or in parallel, however.) They are capable of detecting or correcting any single-digit error originating in the transfer or storage of either operand, or of the sum (or difference), or in any single carry digit, even if any of these errors should themselves result in the propagation of a chain of one or more carries. Compared to the parity-check codes already discussed, only an extra digit or two of redundancy is required for this extra arithmetic protection. The encoding and decoding processes can be readily implemented, and their serial realizations are not very much more complicated than the corresponding encoders and decoders for non-arithmetic codes.

A. SINGLE-ERROR DETECTION

To encode an m-digit binary number $a = (a_{m-1}, a_{m-2}, \ldots a_1, a_0)$ in Brown's single-error-detecting code, we simply multiply it by *3*. The redundant form $A = 3a$ may be added or subtracted to other redundant numbers without violating the "divisible by three" condition:

$$C = A \pm B = 3a \pm 3b = 3(a \pm b) = 3c \quad .$$

The number k of redundant digits added is clearly just *two*:

$$n = m + 2 \quad .$$

This code can be considered as a "fortified" version of the simple parity-check code, which requires only *one* redundant digit.

In handling numbers A, B, etc., through an arithmetic unit, a single error in any *one* of A, B, C, or a carry signal has the effect of adding or subtracting a power of two to the n-digit output number C. Since every power of two is *not* divisible by 3, the resulting incorrect value of C will not satisfy a "divisible by three" check.

For the encoder, the operation of multiplying a number a by three can be implemented by adding a to itself left-shifted once—that is,

$$A = 3a = a + 2a \quad .$$

In a serial encoder, the shifting corresponds to a one-digit delay. Figure 15(a), (b) shows two realizations of such a circuit, which is an enlarged version of a serial full adder. The parallel version is most naturally the iterative equivalent of one of these circuits, and requires n identical cells with the same logic as in Fig. 15(a) or (b), prior to end-cell simplification.

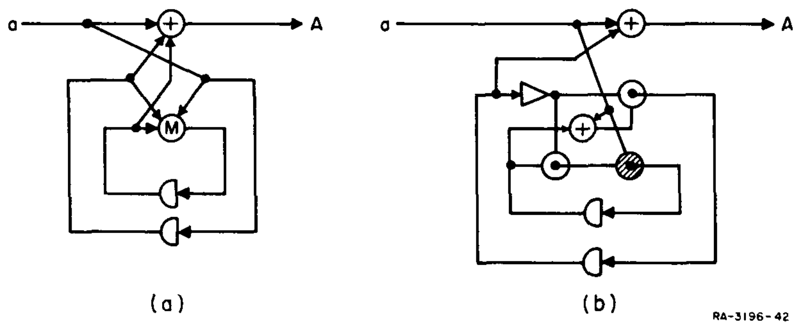

FIG. 15 TWO VERSIONS OF A SERIAL "MULTIPLY BY THREE" NETWORK

For the decoder, it is only necessary to verify that the redundant number $A = \sum_{i=0}^{n-1} A_i 2^i$ is zero to the modulus 3:

$$\sum_{i=0}^{n-1} A_i 2^i \equiv 0 \pmod 3 \quad .$$

But since $2^i \equiv 1$ or -1, according as i is even or odd, respectively, this test is equivalent to

$$\sum_{\substack{i=0 \\ \text{even}}}^{n-1} A_i - \sum_{\substack{i=0 \\ \text{odd}}}^{n-1} A_i \equiv 0 \pmod{3} \quad .$$

Thus a 3-counter arranged to count cyclically by an amount +1 or −1 on alternate *1*-digits of A can perform the test. Any non-zero content in the counter after n digits have passed indicates an error. Figure 16 displays one form of such a circuit. The center delay unit in this figure operates as a 2-counter:*

$$y_3' = \overline{y}_3$$

to provide the alternation between +1 and −1 on the digits of A. The other two delay units are interconnected as a reversible 3-counter:

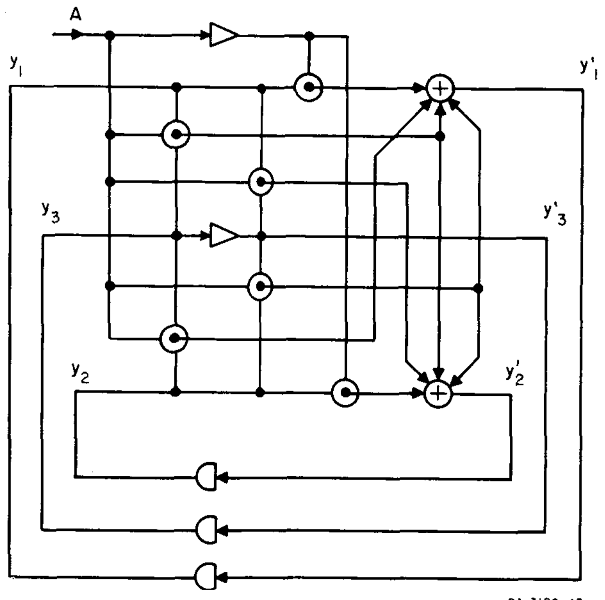

FIG. 16 ONE FORM OF A NETWORK TO TEST A SERIAL DATA SIGNAL FOR DIVISIBILITY BY THREE

* See Ref. 21 for the sequential-logic notation.

$$+1 \quad : \quad 01 \longrightarrow 10 \longrightarrow 11 \longrightarrow (01)$$

$$-1 \quad : \quad 01 \longrightarrow 11 \longrightarrow 10 \longrightarrow (01)$$

with logic

$$y_1' = \bar{A}y_1 \oplus Ay_2y_3 \oplus Ay_2\bar{y}_3 \oplus Ay_1y_3$$

$$y_2' = \bar{A}y_2 \oplus Ay_1\bar{y}_3 \oplus Ay_2\bar{y}_3 \oplus Ay_1y_3 \quad .$$

Conversion to parallel logic allows one to dispense with the 2-counter, and simply to employ an alternation of two different kinds of cells:

$$+1 \text{ cell } (y_3 = 0) \quad : \quad y_1' = \bar{A}y_1 \oplus Ay_2$$

$$y_2' = y_2 \oplus Ay_1$$

$$-1 \text{ cell } (y_3 = 1) \quad : \quad y_1' = y_1 \oplus Ay_2$$

$$y_2' = \bar{A}y_2 \oplus Ay_1 \quad .$$

Other operations besides addition and subtraction can also be performed with this code, but some adjustment or correction of the result may be necessary. For example, two's-complementation of a number, normally obtained by complementing individually each binary digit:

$$b = 2^m - 1 - a$$

now requires a correction to be made:

$$B = 3b = 3 \cdot 2^m - 3 - 3a$$

$$= (2^n - 1 - A) - (2^{n-2} + 2) \quad .$$

Thus, following binary complementation, the number 0100...0010 = $2^{n-2} + 2$ must be subtracted from the result to obtain the complement of a redundant number. Brown[6] proposed the alternative redundant representation

$$A = 3a + 2^{n-3} + 1$$

instead of $A = 3a$, to reduce the correction term to zero, but this now forces the correction to be made after each *addition*.

A binary counter may be simply modified to count in the redundant code—that is, by "three's." The logic is easily derived by known methods:[21]

$$x'_k = x_k \oplus x_{k-1}x_{k-2} \ldots x_3(x_2 + x_1) \cdot 1 \qquad k = 3, 4, \ldots n$$

and $x'_1 = \overline{x}_1$, $x'_2 = x_2 \oplus \overline{x}_1$. The usual equations are

$$x'_k = x_k \oplus x_{k-1}x_{k-2} \ldots x_3x_2x_1 \cdot 1 \qquad k = 1, 2, \ldots n \quad .$$

B SINGLE-ERROR CORRECTION

The same family of codes is appropriate for the correction of errors, except that the multiplier should not be *three*, but one of certain permissible larger numbers. Table III lists some of these values of the multiplier γ, along with the maximum m which can be used for each, and the required number k of redundant digits.[6] For comparison purposes, the number k_H of check digits required for the single-error-correcting Hamming code having the same value of m is also listed.

Table III

PERMISSIBLE MULTIPLIERS γ AND ASSOCIATED CODE PARAMETERS FOR SINGLE-ERROR-CORRECTING CODES FOR ARITHMETIC OPERATIONS

γ	m	k	n	k_H
13	2	4	6	3
19	4	5	9	3
23	6	5	11	4
29	9	5	14	4
37	12	6	18	5
47	17	6	23	5
53	20	6	26	5
59	23	6	29	5
61	24	6	30	5
67	26	7	33	5
71	28	7	35	6
79	32	7	39	6
83	34	7	41	6
101	42	8	50	6
103	43	8	51	6

The encoding operation, "multiply by γ," is now more complex, and standard arithmetic procedures appear to be most natural, except possibly for a few values of γ for which the binary multiplication might be "wired in" rather than programmed. For example, serial multiplication of a number a by $\gamma = 37 = 100101$ could be accomplished in two stages of full addition by adding together a, D^2a, and $D^3(D^2a)$ (where the operator D indicates one digit of delay). Similarly, multiplication by 13 or 19 could be accomplished with just two full adders and a few delays.

To decode, standard arithmetic procedures may again be employed to determine the residue (remainder) after division by γ. If this residue is zero, no error has occurred. A single error in the rth digit position will leave a residue of $\pm 2^r$, reduced modulo γ. The number r of the digit to be corrected, and the sign of the correction, might be determined by a table look-up procedure. The table would provide r and the sign of the correction for each of the possible residues 1, 2, ... $\gamma - 1$. [Note that $n = (\gamma - 1)/2$.]

Brown gives an alternative algorithm, which has been implemented by Peterson in a circuit for serial correction.[3]

C. ASYMMETRIC ERROR CORRECTION

Codes for the correction of single *1*-errors instead of all single errors may be easily derived by using the same values of γ as in Table III, but allowing n to increase from $(\gamma - 1)/2$ to $\gamma - 1$. k stays the same, so that the maximum allowable m is also increased by $(\gamma - 1)/2$. Thus, the fractional redundancy required is about one-half the previous value. This reduction is possible because of the fact that the set of all possible errors in A is now smaller, since these errors are all of the same sign (-2^r) instead of either sign ($\pm 2^r$, for $r = 0, 1, 2, \ldots n - 1$).

The reduction in the required value of γ for the same value of m is reflected in the encoder in a simplified multiplication circuit. For example, the earlier example ($\gamma = 37$) gave $m = 12$ data digits and required two full adders and five unit delays for encoding. For *1*-error correction, a smaller value of $\gamma (\gamma = 19$, say) allows just as many data digits ($m = 13$), but requires two full adders and only four unit delays. The decoder is similarly simplified.

VI SERIES-PARALLEL CODES

Some of the disadvantages of both series and parallel codes may be avoided through the use of codes in which the digits are dispersed both in time and space. The diagram of Fig. 17 depicts a double-parity-check code. A two-dimensional array of data digits is augmented with a row of check digits, selected to satisfy parity checks over each column individually, and a column of check digits, selected to satisfy parity checks over each row individually. It may be shown that (1) the corner check digit is consistent, satisfying both the row and column checks of which it is a part, and (2) the resultant array is capable of either triple error detection, or single-error-correction plus double-error-detection. If we identify one of the dimensional directions of the array (say, the horizontal direction) with *time*, then an encoder and decoder can be envisaged in which the row checks are performed serially and the column checks in parallel. Subject to certain assumptions about the kinds of errors to be expected in such a channel, this series-parallel arrangement can be expected to require less equipment for the same total number m of data digits and the same degree of fault protection.

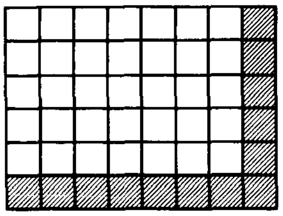

FIG. 17 DIGIT ARRAY FOR A DOUBLE-PARITY-CHECK CODE

Actually, there are so many ways in which two-dimensional and multidimensional codes can be formed, utilized, and implemented that a comprehensive coverage of the possibilities is out of the question here. The individual codes themselves and the notion of two-dimensional checking are adequately discussed in the literature,[22,23] and most of the principles of implementation have been discussed earlier in this report for the serial and parallel portions separately. Consequently, we will confine our attention to a single example of some potential practical utility.

Let a block of $m = m_1 m_2$ data digits be augmented first with $k_1 = 1 + [\log m_1]$ extra columns of check digits, selected to form a *Berger* code individually on each row, then with one row of check digits, selected to satisfy individually all column parity checks (Fig. 18). (Note that the corner check digits cannot now be consistent, but are generated only by the parity checks.) Thus, a separate serial encoder and decoder for each row can provide for the detection of any number of *1*-errors or any number of *0*-errors in any one or more rows (but not both in the same row). If such errors are confined to a single row, however, a parallel column parity check can effectively indicate just which columns are in error, so that such errors may be readily corrected.

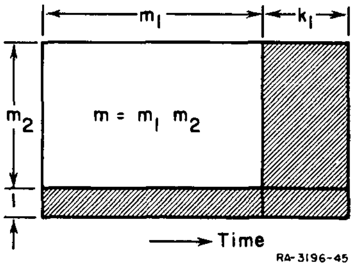

FIG. 18 TYPICAL DIGIT ARRAY FOR A SERIES-PARALLEL ERROR-CHECKING CODE

The encoder and decoder are shown in Figs. 19 and 20, respectively. Each shifting binary counter (SBC) operates as for the purely serial Berger code (Fig. 14), counting *0*'s for the first m digits, then shifting out serially for the next k digits. The parity digit is generated with the large exclusive-OR-gate. In the decoder, these m_1 counters first operate to determine which row, if any, contains the errors. A set of m_1 flip-flops then retain this information so that the delayed block of digits may be corrected by means of the column parity check.

This circuit arrangement is therefore capable of automatically correcting any number of *1*-errors or any number of *0*-errors (but not both) in a single row. With the addition of a small amount of additional logic circuitry (the network L), automatic error *detection* can be obtained for the same types of errors in *any* number of rows, and for any set of mixed *0*-errors and *1*-errors in any one row (and most of those in multiple rows, too). Furthermore, many types of faults in the decoder itself are corrected and detected by this arrangement.

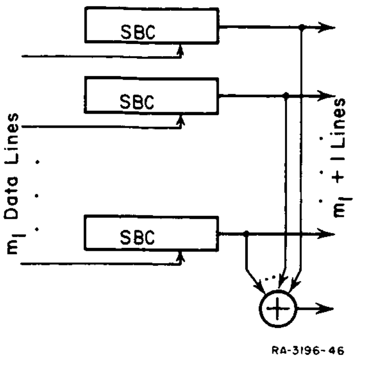

FIG. 19 ENCODER FOR SERIES-PARALLEL CODE

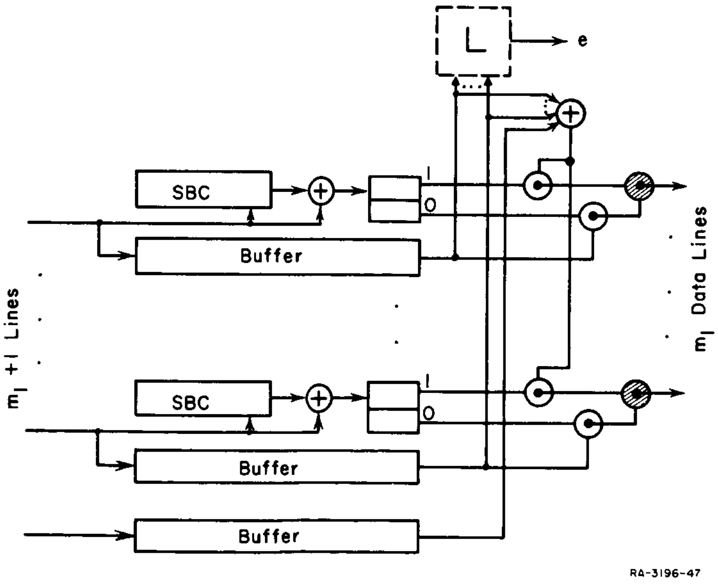

FIG. 20 DECODER FOR SERIES-PARALLEL CODE

Series-parallel codes appear to hold considerable promise for reducing the cost of the coding circuitry below the large values required by purely parallel codes, but still providing protection against the types of faults which are normally not checked without triplication in a serial channel. Also, their potential for arithmetic checking merits investigation.

VII FAULT-MASKED ENCODERS AND DECODERS

Consider now what steps need be taken to render the encoder and decoder insensitive to their own faults. A number of such *fault-masking* redundancy techniques are available. The two most popular are (1) triplication with voting, usually attributed to von Neumann[24] [Fig. 21(a)], and requiring 3^+:1 redundancy, and (2) a form of component replication, which is applicable to either branch-type logical elements, as studied by Moore and Shannon[1] [Fig. 21(b)], or to gate-type elements, as proposed by Tryon[25] [Fig. 21(c)], both with 4:1 redundancy.

In most purely serial networks, these two techniques (or simple extensions of them) appear to be the only reasonable alternatives for protection against arbitrary single faults, because of the lack of any inherent redundancy in the operation being performed or in the circuitry

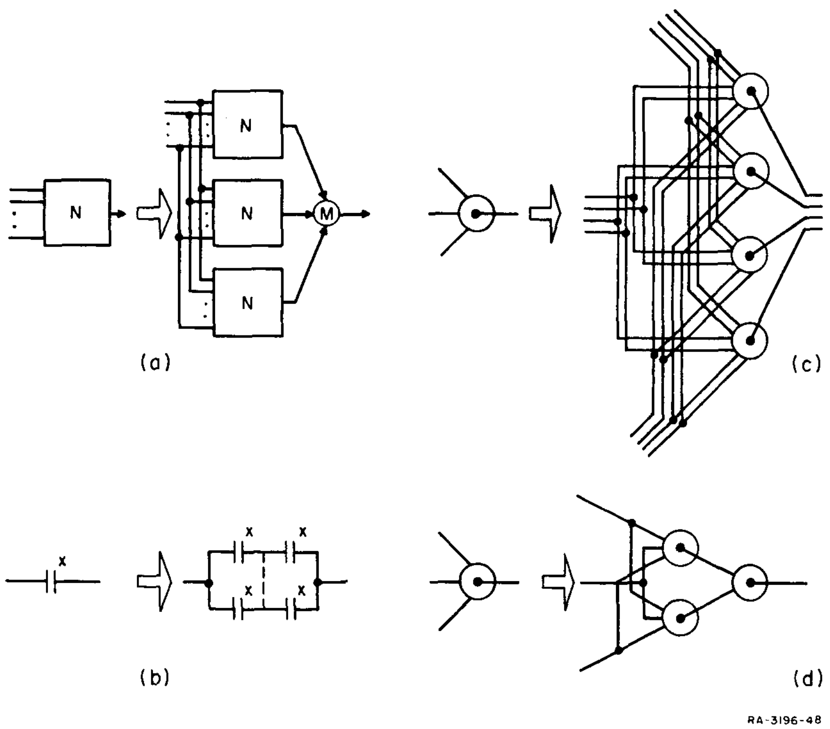

FIG. 21 SOME ELEMENTARY FAULT-MASKING TECHNIQUES

itself. In parallel circuitry, however, it is usually possible to take advantage of existing redundancy in the operation (*i.e.*, the multiple outputs) or in the circuit to reduce the redundancy ratio below 3:1 or 4:1. This is particularly true of iterative networks, and therefore of some of the parallel encoding and decoding networks.[26]

With regard to parallel encoders, it should first be observed that so long as the outputs are formed separately (as in the case with almost all those discussed), a single fault can have no greater effect than a single error in the "channel" itself. Consequently, provided that the encoder and channel *together* are assumed to be limited to a single fault, the encoder is protected along with the channel.

For parallel decoders, faults in the input lines are similarly handled, but the possibility of faults in the exclusive-OR-gates which compose the first and major portion of the decoder requires that additional parity checks be performed, in such a manner that the corrector number (the set of outputs of these k multi-input exclusive-OR-gates) is itself made redundant. This can be done by applying a *second* error-correcting code to this corrector number, and augmenting the circuitry accordingly.

For a decoder based on the Hamming single-error-correcting code, having the parity-check matrix (for $k = 4$, $n = 15$, for example)

$$P = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

we want to increase the number of parity checks so that any single erroneous parity-check digit can be corrected. This is easily done by applying a Hamming single-error-correcting code to the *columns* of P, treating each column as the 4-digit data portion of a code word in the Hamming code.[9] The 3 check digits required will then add three more rows to the matrix:

$$P_c = \left[\begin{array}{ccccccccccccccc}
1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\
0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
\hline
1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \\
1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\
1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1
\end{array}\right]$$

The first portion of the decoder therefore consists of not $k = 4$, but $K = 7$ exclusive-OR-gates. Generally

$$K = k + 1 + \left[\log_2(k + 1 + [\log_2 k])\right]$$

where

$$k = 1 + \left[\log_2(m + 1 + [\log_2 m])\right] \quad .$$

In terms of $\nu(x) = 1 + [\log x]$ = the number of digits in the binary representation of x, then

$$k = \phi(m) + \phi\big(\phi(m)\big)$$

where $\phi(x) = \nu\big(x + \nu(x)\big)$. [In this same notation, $k = \phi(m)$.]

The second portion of the decoder consists of a decoding tree having 7 inputs and 11 outputs (generally, K inputs and m outputs). Each output takes on the value *1* for one of the valid input combinations of the Hamming code, plus all non-valid input combinations which differ from it in just one digit. This type of tree is irredundantly realized most naturally as a collection of AND-gates and inclusive-OR-gates. One of the redundancy techniques of Fig. 21 may then be applied. If the AND- and OR-gates are realized with diodes in the conventional manner, the less expensive circuit of Fig. 21(d) may be used to protect against faults in the diodes themselves, neglecting the possibility of defective output resistors.

There may be some (as yet unknown) way to fault-mask the decoding tree by taking advantage of the fact that a single system fault cannot render both the input signals incorrect *and* the tree itself defective. Thus, the tree need be internally masked for only the valid (error-free) input-variable combinations. It should also be possible to take advantage of the fact that only one output of the tree has the value *1* at any one time. These two possibilities have not been explored.

Faults in the third portion of the decoder—the bank of output exclusive-OR-gates—will generally go uncorrected, so long as the representation of the output data is not itself redundant. This is a basic limitation. If such faults are felt to be sufficiently likely, the point in the system at which the redundancy is dropped (*i.e.*, the decoder) should be moved further down the channel.

The method of applying a second error-checking code over the first, as proposed above, can also be used for the low-density codes. The Hamming codes may be used for this purpose, although the fact that each column of P has only one or two *1*'s may allow some economy in the number of additional rows required when k is large, or in the decoding circuitry.

Decoders (error detectors) for error-detecting codes may be given the protection of single-fault-detection by the same principle. *One* additional check digit must be provided ($K = k + 1$) to guarantee that a parity check is satisfied over the set of all check digits. For the simple parity check code, for which $P = [1\ 1\ 1\ \dots\ 1 \vdots 1]$, the parity checker is simply duplicated:

$$P_c = \left[\begin{array}{ccccc:c} 1 & 1 & 1 & \dots & 1 & 1 \\ 1 & 1 & 1 & \dots & 1 & 1 \end{array}\right] .$$

An output two-input OR-gate then indicates when either check fails. For the low-density error-detecting codes, a single m-input checker would suffice, but an additional k checkers are needed if the limitation to μ-input gates is retained. In both cases, therefore, the cost of the decoder is approximately doubled.

It may be considered adequate to provide only fault-detection in the decoder which implements an error-correcting code.[26] If this is the case, we may add a single extra row to P: $K = k + 1$. The decoder will be increased by one additional exclusive-OR-gate in the first portion, and by one K-input exclusive-OR-gate for fault detection in the second portion. The first of the gates is automatically protected, and faults in the second can be masked by duplication, if desired.

Fault-masked decoders realized with branch-type elements appear to be very intricate. Lofgren[27] offers the network of Fig. 22 as a single-fault-masked version of the network shown in Fig. 11(a) and Fig. 12(a). This network is insensitive to any single short or open circuited element. The redundancy ratio is 14/5 = 2.8, which is clearly better than the 4.0 required by the method illustrated in Fig. 21(b). Masked networks with less than 4:1 redundancy which correspond to the other networks of Figs. 10 and 12 are not known.

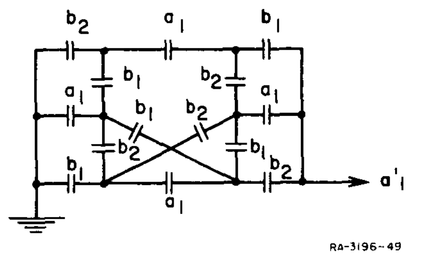

FIG. 22 LOFGREN'S SINGLE-FAULT-MASKED SINGLE-ERROR-CORRECTING NETWORK, CORRESPONDING TO THE IRREDUNDANT NETWORK OF FIG. 11(a)

VIII CONCLUSIONS

The objectives of the program of which this report forms a part are concerned principally with redundancy techniques and their application to improving the reliability of digital systems. This report purports to describe known and new techniques for this purpose. No serious attempt has been made to compare at this time their applicability and value with other techniques discussed previously. The integrated use of various redundancy techniques at one or more levels throughout a digital system is the subject of current study under the present program, and will be reported upon at a later time.

We have attempted in the preceding sections to provide techniques and examples of the use of error-checking codes for compensating for a limited number of faults in certain principal portions of a digital system. Because the problem is not so much one of the existence but the economy of the techniques and their resultant circuits, approximate costs of the encoder and decoders have been calculated, under a reasonable and typical set of assumptions as to the type of circuitry being used.

Table IV summarizes the combined costs of the encoder and decoder for the parallel realizations of most of the single-error-correcting codes discussed in previous sections. The number of conventional gate-type elements, threshold elements, and branch-type elements are shown separately. The final decision as to the "best" code clearly depends upon the type of elements being used, and would usually also depend upon other unlisted or partially listed features of these elements—*e.g.*, the number of element inputs. As mentioned before, the optimum value of k is a compromise between simplified checking circuitry and the reduced size of the redundant portion of the "channel."

The costs in Table IV do not include those for masking faults in the decoder itself. The increase required for masking varies considerably with the logical elements used, the types of faults assumed, and the degree of protection desired.

It has been shown elsewhere that signal redundancy suffers from certain basic limitations: for all but a definable minority of logical

Table IV

COMBINED COSTS OF PARALLEL ENCODERS AND DECODERS FOR SINGLE-ERROR-CORRECTING CODES

CODE	k	NUMBER OF GATE-TYPE ELEMENTS				NUMBER OF THRESHOLD ELEMENTS	NUMBER OF BRANCH-TYPE ELEMENTS
		2-Input AND	2-Input EX-OR	r-Input EX-OR			
				(number)	(r)		
Low-density, $\mu = 2$	$2m$	m	$3m$	0	--	m	$5m$
$\mu = 3$	m	m	$2m$	m	3	$6m$	$15m$
$\mu = 4$	$\left[\frac{2m+2}{3}\right]$	m	m	$\left\{\begin{matrix}k\\k\end{matrix}\right.$	$\left.\begin{matrix}3\\4\end{matrix}\right\}$	$\frac{16}{3}m$	$\frac{94}{3}m$
$\mu = 5$	$\left[\frac{m+1}{2}\right]$	m	m	$\left\{\begin{matrix}k\\k\end{matrix}\right.$	$\left.\begin{matrix}4\\5\end{matrix}\right\}$	$5m$	$59m$
Hamming, $m = 1$	2	2	1	0	--	1	5
$m = 4$	3	4	0	$\left\{\begin{matrix}3\\3\end{matrix}\right.$	$\left.\begin{matrix}3\\4\end{matrix}\right\}$	22	152
general	$\phi(m)$	ρ_0	m	$\left\{\begin{matrix}k\\k\end{matrix}\right.$	$\left.\begin{matrix}\mu_0\\\mu_0-1\end{matrix}\right\}$	$2m + 2k^2 - k$ *	

* For lossless case; in general, $\tau \leq 2n + k[\log_2 \mu_0] + k[\log_2 (\mu_0 - 1)]$.

circuits, there exist some types of faults whose errors no amount of signal redundancy can correct,[7,28] This theoretical limitation need not overly concern us, however, for three reasons. First, this minority includes several circuit operations of considerable practical importance, such as simple data transfer, parity checking, and linear (*i.e.*, pure exclusive-OR) logical circuits generally. Second, we rarely need to protect a circuit against *all* possible faults, but only a selected class deemed to be the most likely. Third, it has recently been shown that, under certain reasonable assumptions, a network can be made arbitrarily reliable with the proper *combination* of signal and circuit redundancies.†

We may legitimately conclude that several known and some new codes can be fruitfully and economically applied to the problem of increasing the reliability of a digital network or system, provided only that the limitation to "single-fault-checking" actually covers all but a small fraction of the expected types of failures. This will be the case provided the irredundant version of the network or the system is already sufficiently reliable so that the likelihood of double faults is truly negligible. This point of view is identical to that taken in coding theory in order to remove from an otherwise logical-combinatorial problem

† Private communication from J. D. Cowan of MIT.

all probabilistic considerations. Thus, we normally speak of single-error-correcting or double-error-detecting codes, in preference to codes having a fixed probability of uncorrected or undetected error.

Further work is warranted in refining the costs of some of the decoding circuits discussed, particularly those based on codes for checking arithmetic operations. Also, it is conceivable that future effort in coding theory will uncover some improved codes of this class whose encoders and decoders are as simple, say, as those in Figs. 1 and 2 for the single-error-correcting codes. Finally, improved circuits for fault-masked correctors are needed for most of the cases considered in this discussion, but particularly for gate-type realizations of serial, parallel, and series-parallel codes for arithmetic checking.

REFERENCES

1. E. F. Moore and C. E. Shannon, "Reliable Circuits Using Less Reliable Relays," *J. Franklin Inst.* **262**, 3, pp. 191-208 (September 1956); **262**, 4, pp. 281-297 (October 1956).

2. W. H. Kautz, "Development of Techniques for Improving the Reliability of Digital Systems Through Logical Redundancy," Quarterly Letter Report 3, SRI Project 3196, Stanford Research Institute, Menlo Park, Calif. (20 January 1961).

3. W. W. Peterson, "Error-Correcting Codes," (John Wiley & Sons, Inc., New York City, 1961).

4. J. E. Meggitt, "Error Correcting Codes and Their Implementation for Data Transmission Systems," *IRE Trans., PGIT*-**7**, 4, pp. 234-244 (October 1961).

5. B. Elspas, "Design and Instrumentation of Error-Correcting Codes," Interim Tech. Report, Project 3318, Stanford Research Institute, Menlo Park, Calif. (October 1961).

6. D. T. Brown, "Error Detecting and Correcting Binary Codes for Arithmetic Operations," *IRE Trans., PGEC*-**9**, pp. 333-337 (1960).

7. P. Elias, "Computation in the Presence of Noise," *IBM Jour.* **2**, 4, pp. 346-353 (October 1958).

8. M. J. E. Golay, "Notes on Digital Coding," *Proc. IRE* **37**, p. 657 (1949).

9. R. W. Hamming, "Error Detecting and Error Correcting Codes," *Bell. System Tech. J.* **29**, pp. 147-160 (1950).

10. I. S. Reed, "A Class of Multiple-Error-Correcting Codes and the Decoding Scheme," *IRE Trans., PGIT*-**4**, pp. 38-49 (1954).

11. E. Prange, "Cyclic Error-Correcting Codes in Two Symbols," AFCRC-TN-57-103, Air Force Cambridge Research Center, Cambridge, Mass. (September 1957).

12. B. Elspas and R. A. Short, "A Note on Optimum Burst-Error-Correcting Codes," *IRE Trans., PGIT*-**8**, 1 (January 1962).

13. R. G. Gallager, "Low Density Parity Check Codes," *IRE National Convention Record* **8**, 4, p. 126 (1961).

14. E. J. McCluskey, "Iterative Combinational Switching Circuits," *IRE Trans., PGEC*-**7**, 4, pp. 285-291 (December 1958).

15. A. W. Burks *et al.*, "Complete Decoding Nets—General Theory and Minimality," *J. Soc. for Industrial and Applied Math.* **2**, 4, pp. 201-243 (1954).

16. W. H. Kautz, "The Realization of Symmetric Switching Functions with Linear-Input Logical Elements," *IRE Trans., PGEC*-**10**, 3, pp. 371-8 (September 1961).

17. S. H. Caldwell, *Switching Circuits and Logical Design* (John Wiley & Sons, Inc., New York City, 1958); see Chap. 7.

18. W. H. Kim and C. V. Freiman, "Single-Error-Correcting Codes for Asymmetric Binary Channels," *IRE Trans., PGIT*-**5**, 2, pp. 62-66 (June 1959).

19. J. M. Berger, "A Note on Error Detection Codes for Asymmetric Channels," *Info. and Control* **4**, pp. 68-73 (1961).

20. C. V. Freiman, "Protective Block Codes for Asymmetric Binary Channels," Research Report RC-457, IBM Corp Research Center, Yorktown, N.Y. (May 23, 1961).

21. W. H. Kautz, "State-Logic Relations in Autonomous Sequential Networks," *Proceedings of Eastern Joint Computer Conference*, 1958, AIEE Special Publication T-114, pp. 119-127.

22. W. H. Kautz, "A Class of Multiple-Error-Correcting Codes for Data Transmission and Recording," Tech. Report 5, Project 2124, Stanford Research Institute, Menlo Park, Calif. (August 1959).

REFERENCES

23. P. Calingaert, "Two-Dimensional Parity Checking," *J. Assoc. for Computing Machinery* 8, 2, pp. 186-200 (April 1961).

24. J. Von Neumann, "Probabilistic Logics and the Synthesis of Reliable Organisms from Unreliable Components," in *Automata Studies,* C. E. Shannon and J. McCarthy, Eds. (Princeton University Press, Princeton, N.J., 1956), pp. 43-98.

25. J. G. Tryon, "Redundant Logic Circuitry," U.S. Patent 2,942,193 (June 21, 1960).

26. W. H. Kautz, "Automatic Fault Detection in Combinational Switching Networks," Tech. Report 1, Project 3196, Stanford Research Institute, Menlo Park, Calif. (April 1961); also, *AIEE* Publication S-134, "Switching Circuit Theory and Logical Design," (September 1961).

27. L. Lofgren, "Qualitative Limits for Automatic Error Correction-Self-Repair," Tech. Report 6, University of Illinois Electrical Engineering Research Laboratory, Urbana, Ill. (21 June 1961); see p. 30.

28. W. W. Peterson and M. O. Rabin, "On Codes for Checking Logical Operations," *IBM Jour.* 3, 2, pp. 163-168 (April 1959).