A General Method of Applying Error Correction to Synchronous Digital Systems

By D. B. ARMSTRONG

(Manuscript received March 8, 1960)

A general method is presented for applying error correction to synchronous binary digital systems to improve reliability. It includes the familiar scheme of triplication and "vote taking" as a special case. In principle, the method permits the system to operate continuously, even when a fault is present or maintenance is being performed. An efficient maintenance routine, including rapid repair of faults, is an essential adjunct to the scheme if the potentially large increase in reliability made possible by error correction is to be realized.

The percentage redundancy needed to realize the scheme decreases as the complexity of the system to which it is applied increases, but may amount to triplication of equipment even for moderately large systems. The paper describes some error-correcting codes to implement the scheme, discusses error-correcting circuits in a general way, indicates how to estimate the redundancy, and presents a formula for determining the reliability improvement obtainable with a particular maintenance routine. In a companion paper,[1] D. K. Ray-Chaudhuri develops a general theory of minimally redundant codes for this application.

I. INTRODUCTION

This paper describes a general method of applying error correction to synchronous digital data systems. It includes, as a special case, the well-known scheme of triplication with vote taking.[2] Since the scheme employs error-correcting codes, it is capable of detecting errors as well as correcting them. Hence, maintenance personnel can be alerted as soon as a fault occurs. Also, it has the property of enabling the system to which it is applied to continue to function correctly even when faults are present and maintenance is being performed, provided all the faults are confined to any one of the several subunits which comprise the sys-

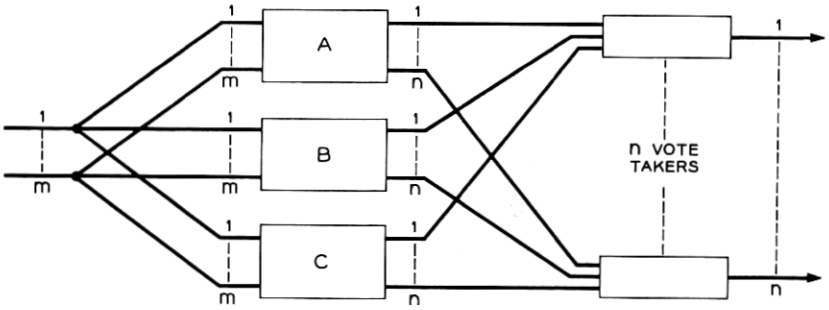

Fig. 1 — Error correction by triplication and vote taking.

tem. Therefore, if faults are found and repaired quickly, so that they do not accumulate to the point where the resulting errors are beyond the error-correcting capabilities of the scheme, the system may be kept in continuous operation for much longer periods than could the equivalent system without error detection and correction.

In this regard, it is estimated that the "mean life"* of the system with error correction can be made several thousand times as long as that of the equivalent nonredundant system, provided faults are repaired sufficiently soon after their occurrence. Such potentially vast increases in reliability depend of course on the availability of rapid diagnostic and fault-repair facilities. Conversely, in the absence of maintenance the mean life of the redundant system will in general be less than that of the nonredundant system. Hence the scheme is not usefully applicable to a system which must operate in an environment where rapid fault repair is impossible — in such situations some other method of building in reliability, such as microlevel redundancy,[3] would be necessary.

In comparison with triplication and vote taking, our procedure will permit more precise localization of faults. Also, for large systems it should result in less over-all equipment redundancy. For small systems, however, an equipment advantage may not always be realized. Since the triplication scheme is fairly well known, we shall start by describing it, but from a slightly different point of view, which shows how it appears as a special case of our procedure.

Fig. 1 shows a system, A, with m inputs and n outputs, and two exact replicas of the system, B and C. Corresponding output wires from A, B and C are fed to "majority" circuits, or vote takers, each of whose out-

* The mean life of a system is here defined as its mean time to failure, assuming it is in perfect condition at the start.

puts agrees with the majority, i.e., with any two or all three, of the inputs which are in agreement. Thus, the system corrects for errors which are confined to the outputs of any one of systems A, B or C.

We may consider the outputs from A to carry information bits, and those from B and C to carry check bits, which generate Hamming[4] single error-correcting codes, in the following manner. The matrix below displays the output bits from A, B and C in a matrix consisting of three rows, each row having n entries:

	output #1	output #2	output #3		output #n	
	↓	↓	↓		↓	
$A_1 \rightarrow$	O	O	O	···	O	← outputs from system A
$B_1 \rightarrow$	X	X	X	···	X	← outputs from system B
$C_1 \rightarrow$	X	X	X	···	X	← outputs from system C

O = information bit
X = check bit

Alternatively, the matrix may be thought of as displaying n columns, each with three entries consisting of one information bit and two check bits. For example, the first column contains the information bit A_1, and check bits B_1 and C_1. Two parity checks are constructed from this column; bits A_1 and B_1 satisfy the parity relation

$$A_1 \oplus B_1 = 0,$$

where $\oplus$ represents the sum modulo 2. Bits A_1 and C_1 satisfy the parity relation

$$A_1 \oplus C_1 = 0.$$

These relations merely state that, when the complete system is operating correctly, both B_1 and C_1 will have the same value as A_1.

This coding has the ability to detect any single error in column 1, and moreover tells us which bit is in error, so that corrections can be performed. Therefore, in particular, this scheme permits the correction of any pattern of errors which is confined to a single row of the matrix. Since faults which are confined to one of the systems A, B, or C can cause errors on the outputs of that system alone, this error-correcting scheme will permit the over-all system to operate correctly even when any one of the three systems comprising it is faulty, or is disabled for maintenance purposes.

Obviously, this particular coding is inefficient, because two check bits

are needed for each information bit. It is to be hoped that the use of more efficient codes will result in less equipment redundancy, primarily because fewer check bits will have to be generated. The problem then is to find a way of organizing a system so as to permit error correction with more efficient codes. A scheme for doing this will now be described.

II. A GENERAL SCHEME FOR ERROR CORRECTION

Suppose system A of Fig. 1 is designed so that it breaks down into a number, say r, of electrically independent subunits, each subunit carrying not more than p of the n system outputs, as shown in Fig. 2.

A fault or faults that is confined to any one subunit can at most cause errors on the outputs of that subunit. Therefore, consider the following matrix, in which the outputs of each subunit are displayed in a separate row, with p entries per row. There are $(q - r)$ additional subunits shown in Fig. 2; these provide k check bit outputs:

$$
\left.\begin{array}{l}
\left.\begin{array}{l}
\left.\begin{array}{llll} O & O & \cdots & O \leftarrow \text{outputs from subunit } 1 \\ \vdots & \vdots & & \vdots \\ O & O & \cdots & O \leftarrow \text{outputs from subunit } r \end{array}\right\} r \text{ rows} \\
\left.\begin{array}{llll} X & X & \cdots & X \leftarrow \text{outputs from subunit } r+1 \\ \vdots & \vdots & & \vdots \\ X & X & \cdots & X \leftarrow \text{outputs from subunit } q \end{array}\right\} (q-r) \text{ rows}
\end{array}\right\} q \text{ rows}
$$

Since faults in a single subunit affect only a single row of the matrix, we may, for example, apply Hamming single error-correcting codes on a

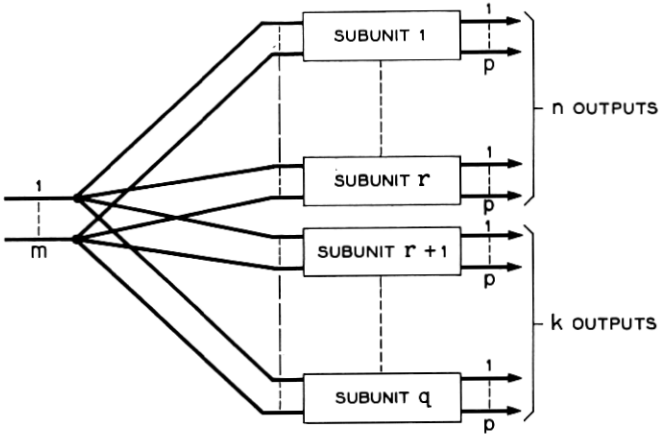

Fig. 2 — Breakdown of system A into subunits.

per-column basis. Thus, if $r = 4$, we need only three check bits per column to provide Hamming single-error correction, and the code redundancy is much less than in the triplication scheme ($\frac{3}{7}$ instead of $\frac{2}{3}$).

Actually, Hamming codes are not the most efficient that could be used for this error-correcting scheme, because they do more than is required. Specifically, they permit correction of any single error per column of the matrix, even though these errors may not be confined to a single row. If we apply the further restriction that all errors be confined to a row, then more efficient codes are possible and are described later.

We now wish to show how it is possible to break down a system into electrically independent subunits. Digital systems may be classified into two types: those which perform only combinational logic (have no memory), and those which perform sequential logic (have memory). The latter type is of more interest, but it is useful to deal with the former first. We assume throughout that the data on the input wires are not in error, and that faults in the system do not cause errors on the input wires.

Suppose then that the r subunits in Fig. 2, which produce the n system outputs, consist entirely of combinational logic. It is evident that the system can be broken down into such subunits because, for example, each output can be realized by designing a separate combinational logic circuit which generates the appropriate Boolean function of the m input variables. Alternatively, some savings in logic elements may be possible by designing multifunctional logic circuits, each generating only the p outputs of a single subunit.

To provide check outputs, additional subunits are needed, and are designated $(r + 1)$ through q in Fig. 2. To design these, it is necessary to be able to express each check output as a Boolean function of the m input variables. This can be done because the structure of the error-correcting code will specify each check output to be the sum modulo 2 of some set of information outputs, and since the latter are known functions of the inputs, we can therefore express the check outputs directly as functions of the inputs. We may, of course, work with truth tables instead of functional representations.

In the case of sequential logic, a complication is introduced which may be explained with the aid of Fig. 3. In this figure, a sequential system is represented as consisting of two major units. Unit 1 consists entirely of combinational logic and unit 2 consists entirely of memory.*

* Some authors replace the memory elements by unit delay elements. See for example, Fig. 1 of Unger.[5] His paper deals with asynchronous circuits, whereas we are treating synchronous circuits of the type designated "*PP*" by Cadden.[6]

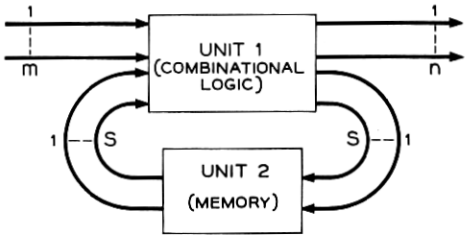

Fig. 3 — A possible configuration for a finite-state sequential system.

The combinational logic generates two sets of outputs:

(a) The n system outputs;

(b) s "feedback" outputs which provide the inputs to the memory unit.

The s outputs of the memory unit, in conjunction with the m system inputs, comprise the inputs to the combinational logic unit.

Suppose that unit 1 is designed as r electrically independent subunits. In general, the $(m + s)$ inputs to unit 1 will feed all r subunits. A fault in a single subunit will cause errors on the output of that subunit, and these will feed back via the memory unit to the inputs of some or all of the other subunits. Hence, in a few cycles of operation it is possible that the outputs of all subunits will be in error because of a fault in just one subunit. This situation can be remedied by applying error correction to some or all of the s feedback wires in addition to the n system output wires. These additional corrections should be made between the outputs of unit 2 and the inputs of unit 1, in order to correct errors caused by faults in unit 2 as well as in unit 1.

Alternatively, it is possible to design the system so as to avoid correcting the internal feedback wires, and yet insure that a fault affects not more than p of the n system outputs. For example, instead of breaking down the system into r subunits, one could replicate the system r times and utilize only outputs $1, 2, \cdots, p$, from the first replica, outputs $p + 1, p + 2, \cdots, 2p$, from the second replica $\cdots$ and outputs $n - p + 1, n - p + 2, \cdots, n$, from the rth replica.

No doubt this alternative realization could be achieved without using r complete replicas of the system. However, the necessary design procedures are not well formulated and the resulting equipment redundancy is difficult to estimate. In contrast, the design procedure for the first mentioned method is straightforward, its redundancy is easier to estimate and, at least with present devices and techniques, it appears to result in considerably less over-all redundancy. Therefore, in the remainder of the paper we shall assume that the first method is to be used.

Accordingly, the correction of a sequential system requires that unit 1 of Fig. 3 be designed as r independent subunits and that it be augmented by a combinational logic unit which generates k check outputs, where k is large enough to provide the necessary parity checks for correcting $(n + s)$ wires (we assume that all s feedback wires may require correction). As in the purely combinational case, each check bit can be expressed as a sum modulo 2 of an appropriate subset of the n outputs of unit 1 and the s outputs of unit 2, and since these are known Boolean functions of the $(m + s)$ inputs to unit 1, each check bit output can likewise be expressed as a Boolean function of these same inputs.

It is of course necessary that the check bit logic circuits also be designed as independent subunits with not more than p outputs per subunit.

III. ERROR-CORRECTING CODES

Before discussing specific codes, we wish to establish lower bounds on the number of check bits, k, needed to fulfill our error-correcting requirements. Specifically, referring to the matrix above for the outputs from r subunits of system A, we ask what minimum value of k is required to permit correction of every possible pattern of errors in any single row of the q rows.

Actually, two lower bounds are applicable. The first bound, which is also the larger of the two when $q > (2^p + 1)$, p being the number of entries per row, is easily derived as follows: Observe that the number of possible error patterns in a single row is $(2^p - 1)$, if we exclude the no-error pattern. Therefore, the total number of error patterns in all q rows is $q(2^p - 1)$. Obviously, k must be large enough to permit as many "parity failure" patterns as there are error patterns. This requires that k satisfy the inequality

$$(2^k - 1) \geqq q(2^p - 1).$$

That is,

$$k \geqq [\log_2 (q2^p - q + 1)], \tag{1}$$

where the square bracket denotes the smallest integer which is

$$\geqq \log_2 (q2^p - q + 1).$$

The second lower bound, which is larger than the first when $q \leqq (2^p + 1)$, and which is therefore of greater practical significance, is:

$$k \geqq 2p. \tag{2}$$

It is derived by determining the maximum number of code words that

can be chosen out of a set of 2^{pq} binary words of lengths pq, and which fulfill the specified error-correcting requirements. Its derivation is relegated to the Appendix.

Surprisingly, it was found not too difficult to construct codes for most values of p and q in the range $2 < p < 10$, $3 < q < 9$ which achieved the appropriate lower bound.

Subsequent to the work described here, Ray-Chaudhuri[1] developed a general theory of minimally redundant codes for this application. However, it will not be out of place to exhibit here some of the codes previously derived, since they are also minimally redundant, and since the error-correcting equipment required to implement either them or the Ray-Chaudhuri codes is of the same general character and complexity. Three families of codes* are exhibited below in matrix form, corresponding to three values of p, as follows:

Family 1: $p = 2$; $q = 3$, 4 and 5.
Family 2: $p = 3$; $q = 5$, 6, 7, 8 and 9.
Family 3: $p = 4$; $q = 4$, 5, 6 and 7.

Family 1 ($p = 2$)

1	2
X	X
3	4
X	X
14	23
O	O
13	124
O	O
24	123
O	O

Family 2 ($p = 3$)

1	56	24
X	O	O
3	15	25
X	O	O
5	146	36
X	O	O
6	34	2
X	O	X
4	126	35
X	O	O
13	46	1245
O	O	O
16	456	23
O	O	O
14	125	236
O	O	O
45	136	256
O	O	O

Family 3 ($p = 4$)

1	5	23	46
X	X	O	O
2	6	38	57
X	X	O	O
3	7	14	58
X	X	O	O
4	8	12	67
X	X	O	O
18	25	36	47
O	O	O	O
16	24	58	378
O	O	O	O
13	27	56	478
O	O	O	O

* These codes were constructed by George Allen at Bell Telephone Laboratories in Summer, 1959.

The set of digits beside each bit position indicates the parity groups which that bit enters. For example, in family 1, the digit set attached to the last bit in the last row is 123, indicating that this bit enters parity check groups 1, 2 and 3. As a further illustration, the bits in family 1 that are labeled 1, 14, 13, 124 and 123 enter parity group 1; their sum modulo 2 is zero when there are no errors. The bit positions which carry only a single digit are check bit positions. They are denoted by X's and the information bits are denoted by O's. The matrix displayed for family 1 is a 2×5 matrix; however, if a 2×4 or a 2×3 matrix is desired, one omits respectively the last row or the last two rows. Similar remarks apply to families 2 and 3.

Several remarks should be made at this point. First, observe that in the original matrix we represented the check bits as being located entirely in the last $(q - r)$ rows. However, this is not necessary; the check bits may appear along with information bits in some or all rows, as is the case in the three matrices above. The arrangement is dictated by the structure of the code, but bits may be permuted within each row with impunity.

Secondly, it is possible to delete information bits from any row without destroying the utility of a code; in such cases, the deleted bits will be omitted from the parity checks in which they would normally participate.

Finally, we observe that the three matrices above provide only for values of $p \leqq 4$ and $q \leqq 9$. If for any reason we wish to form matrices with $p > 4$, or $q > 9$, this may be done by building up the over-all matrix, either vertically or horizontally, or both, from several of the above matrices. Thus a wide variety of equipment arrangements can be accommodated. However, if we build up vertically, we will sacrifice minimal code redundancy. For example, if we form a matrix with 3 columns and 18 rows by using two matrices of family 2, we shall have included $2 \times 6 = 12$ check bits, whereas the minimum is given by bound 1, namely

$$[\log_2 (18 \times 2^3 - 18 + 1)] = 7 \text{ bits.}$$

IV. ERROR-CORRECTING CIRCUITS

A discussion of error-correcting circuits is included here to indicate roughly the amount of equipment involved in error correction, and to provide a basis for a maintenance routine which is proposed later.

It was explained in Section II that, with the present scheme, it is necessary to apply error correction either to the n system outputs of a purely combinational system, or to the $(n + s)$ outputs and feedback

connections of a sequential system. To do this, k parity check bits must be generated, where k is determined by the structure of the error-correcting code employed. Therefore, the inputs to the error-correcting circuits consist of $(N + k)$ wires, where $N = n + s$ ($s = 0$ for a purely combinational system) and k wires carry the check bits. The output of the error-correcting circuits consists of N wires which carry the corrected versions of the corresponding N inputs.

We shall want to distinguish between the correcting circuits and the circuits which are corrected or correctable. The latter comprise the set of q subunits which generate the $(N + k)$ outputs; for example the circuits in Fig. 2. Therefore, we shall hereafter refer to the q subunits as "the system," without modifier.

The error correcting-circuits may be considered to perform the following three functions:

1. Reconstruct the parity checks to determine which, if any, have failed.
2. From the pattern of parity check failures, determine which of the $(N + k)$ input wires are carrying erroneous bits.
3. Correct that subset of the N wires which are carrying erroneous bits. Erroneous check bits do not require correction.

Circuits to perform the above tasks may be realized in several ways, and with varying degrees of redundancy to assure reliability. At one extreme, the error-correcting circuits could be nonredundant, in which case an efficient preventive maintenance routine would be required to insure that they perform for long periods without error. Alternatively they could be built with microlevel redundancy, in which case preventive maintenance would again be necessary but would be applied less frequently. A third alternative would be to make some or all of the error-correcting circuits "self-error-detecting." Those parts which were self-error-detecting would be subjected to maintenance only when a fault was detected; the parts which were not self-error-detecting would require preventive maintenance.

As a fourth alternative, it might be attempted to make the error-correcting circuits completely self-error-correcting. However, a simple heuristic argument can be given which indicates that it is impossible to achieve this goal.

Fig. 4 shows a block diagram of a proposed error-correcting circuit, designed according to the third alternative above. Box 1 in Fig. 4 contains the units which perform functions 1 and 2 above. Box 2 performs function 3 above, and also an error-sensing and alarm function. For simplicity, sets of wires in this figure are represented by single directed

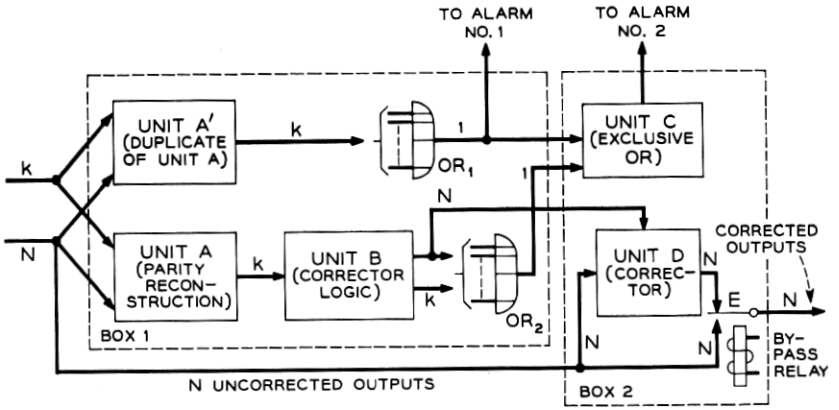

Fig. 4 — A proposed error-correcting circuit.

lines with an associated symbol indicating the number of wires. Box 1 is designed to be self-error-detecting, but box 2 is not. A detailed description of the operation of the circuits in Fig. 4 does not contribute significantly to an understanding of the over-all scheme, and so is omitted.

V. RELIABILITY ANALYSIS

We shall first describe a maintenance routine which is appropriate to the particular mode of error correction realized by the circuits of Fig. 4. We shall then apply an approximate formula which indicates the improvement in reliability of the redundant error-correcting system over the original nonredundant system when our particular maintenance routine is employed. For brevity, derivation of this formula is omitted.

The maintenance routine is as follows. If a fault occurs in some subunit of the system, unit D under control of box 1 corrects the resulting errors. Simultaneously, box 1 operates alarm 1. The faulty subunit is then located and repaired as quickly as possible. In principle, the system can continue to operate correctly even when the faulty subunit is being replaced or repaired, provided a fault does not develop in another subunit or in the error-correcting circuits during the repair of the original fault.

If box 1 fails, alarm 2 is operated, and possibly alarm 1 also, and simultaneously relay E is switched to the bypass position, thus causing essentially no interruption in system operation. Box 1 must also be repaired quickly, because if a fault occurs in the system during the repair of box 1, the system will fail, since it is not now being error cor-

rected. If units C or D fail, they do not operate an alarm since they are not provided with error detection. The probability of failure of these units must, therefore, be minimized by a preventive maintenance routine. To minimize the outage times of units C and D, two copies of each could be provided, one operating and one standby. They would be interchanged at regular intervals T, and preventive maintenance would be applied to the units currently in the standby condition. In this way, the system would not lose its error correcting capability during preventive maintenance. Relay E must be switched to the bypass position to permit continuous operation while unit C or D is being replaced. We assume that the relay switching time is short enough so as not to interrupt system operation.

We now wish to determine a quantitative measure of the reliability improvement of the maintained error-correcting system over that of the nonredundant system. A useful measure is the ratio of their respective mean lives; namely, $R_L = L_r/L_i$, where L_r is the mean life of the redundant system and L_i is the mean life of the nonredundant system. In general, the derivation of R_L is quite complicated, but by making suitable simplifying assumptions we can obtain an approximate formula which is useful. These assumptions are:

1. All components have an exponential survival probability function and the same mean life, which is taken to be the time unit. Therefore, the survival probability of any component is exponential $(-t)$.
2. Components fail independently.
3. Failure of any component in the nonredundant system causes that system to fail.
4. The bypass relay has zero probability of failure.
5. The times taken to repair faulty circuits are assumed constant.

These and other parameters are now defined:

Δ_1 = repair time of a subunit of the system,

Δ_2 = repair time of box 1,

Δ_3 = time that unit C or unit D is removed from circuit when being exchanged with its standby,

T = time interval between successive replacements of unit C or D by its standby

n_1 = number of components in each subunit of the system,

n_2 = number of components in box 1,

n_3 = number of components in box 2,

q = total number of subunits in the system,

r = number of subunits which provide the N system outputs.

From these assumptions plus some others concerning the relative

values of the above parameters, we can obtain the following formulas:

$$R_L = \frac{n_1 q}{2\delta}, \tag{3}$$

where

$$\delta = \tfrac{3}{2} n_3 T + r n_1 \frac{\Delta_3}{T} + q n_1 [(q-1) n_1 \Delta_1 + n_2(\Delta_1 + \Delta_2)]. \tag{4}$$

As an application of (3) and (4), consider a system having parameter values as follows:

time unit = mean component life = 1000 years,

$\Delta_1 = \Delta_2$ = one-half hour = $\frac{1}{2} \times 10^{-7}$ units (time to repair a system subunit or box 1),

Δ_3 = 8 seconds = 2×10^{-10} units (time to replace unit C or unit D by its standby),

T = one month = 10^{-4} units (maintenance interval for units C and D),

$n_1 = n_3 = 333$ components,

$n_2 = 666$ components,

$q = 6$,

$r = 4$.

Substituting these values in (3) and (4) results in R_L equal to 2900. That is, the mean life of the redundant system is 2900 times that of the nonredundant system. Actually, this figure could be improved if we reduced the repair times Δ_1 and Δ_2 from one-half hour to, say, five minutes. Such a reduction would be possible if the system were built of small modular packages and a highly automated diagnostic routine were available to locate faults in the order of a minute or less. Thus, a fault could be pinpointed to one or two particular packages, and these packages could be replaced immediately by good standby packages, thus permitting correction of the fault in minutes. The faulty packages could then be tested and repaired in more leisurely fashion, and this latter time would not be chargeable to Δ_1 or Δ_2.

Therefore, it appears that, with an efficient maintenance routine, the mean life of the error-correcting system can be several thousand times that of the nonredundant system.

VI. REDUNDANCY AND SPEED PENALTIES

It would be desirable to estimate the amount of equipment redundancy in the scheme described, and any attendant reduction in the operating speed of the system. The equipment redundancy cannot be specified

in simple terms, but the contributing sources can be delineated and roughly evaluated. They are:

(a) the circuits which generate the check bits;

(b) the circuit redundancy which results from designing the system as q independent subunits;

(c) the error-correcting circuits.

The redundancy contribution of item (a) can reasonably be estimated to be in the ratio k/N to the amount of equipment in the original non-redundant system. That is, if the amount of equipment in the non-redundant system is treated as one "unit" of equipment, the amount of equipment needed to generate the check bits would be roughly k/N "units." As indicated by the codes in Section III, values of k/N as small as $\frac{1}{3}$ are achievable for $N \geqq 18$. The assumption underlying this estimate is that the amount of circuitry required to generate each check output is the same as the amount required to produce each original system output.

The redundancy contribution of item (b) is believed to be insignificant compared to the other contributions, especially for large systems, provided optimal design techniques are employed. The contribution of item (c) is by far the largest, and is the most difficult one to estimate. It depends on the type of logic technology employed, on the amount of time delay that the error-correcting function is permitted to introduce, and to some extent on the particular error-correcting code used. The following estimates may suggest an order of magnitude for item (c), in the particular case of the correcting circuits proposed in Fig. 4, and assuming the use of diode logic. Based on "paper" designs of these circuits, the author estimated that the correcting circuits might require roughly 60 to 70 "equivalent" diodes per wire corrected (the number of wires corrected is N). Transistors were counted as equivalent to two diodes, resistors etc. were not counted.

This estimate assumes that units A and A′ of Fig. 4 both employ $(c - 1)$ EXCLUSIVE OR circuits per parity check over c bits, and that unit B of Fig. 4 is realized with two-stage logic. Thus, if these error-correcting circuits were to be applied to a system which, in its non-redundant form, was realizable with 30 to 35 "equivalent" diodes per wire corrected, it is evident that the correcting circuits would comprise two "units" of equipment. This ratio is not considered to be unrealistically high.

In general, therefore, it is to be expected that the scheme in question may introduce an amount of redundancy equivalent to at least triplication of the original equipment. However, it has the potential of being

less redundant than the triplication and vote-taking scheme, in which generation of the check bits alone results in triplication and the correcting circuits are additional.

In this connection, it is remarked that the triplication scheme is usually thought of as including relatively simple vote-taking circuits which perform corrections but are incapable of detecting errors and operating appropriate alarms. These latter features would have to be included in order to render the scheme truly comparable to the more general error-correcting procedure described here.

In principle, it appears that the redundancy penalty might be made to decrease monotonically as the systems to which error correction is applied becomes increasingly complex, provided the following two assumptions are valid:

(a) as the number N of corrections is increased, the coding efficiency also increases; that is, k/N becomes smaller;

(b) the amount of equipment per correction in the original system increases faster than the amount of equipment per correction in the correcting circuits.

Assumption (a) is realizable, but (b) cannot be verified. Indeed, (b) may be plausible only provided the correcting circuits use an increasing number of logic stages, which can be expected to result in an increase in time taken to perform corrections; that is, an increasingly severe speed penalty is imposed.

In this regard, the parity check circuits referred to earlier in this section require $2 \times [\log_2 c]$ logical stages. (The square bracket denotes the smallest integer which is equal to or greater than $\log_2 c$.) For the special codes described in Section III, the number of bits per parity check, c, is typically equal to q, the number of subunits in a system, and q must increase in order to increase the coding efficiency. It therefore follows that greater coding efficiency can be achieved only at the expense of greater delay in the corrector, or more complex correcting circuits, or both, and a compromise must be reached.

Finally, a remark should be made concerning the impact of this scheme on the over-all design of a sequential system when the method which requires both feedback and output corrections is used. To minimize the number of corrections necessary, the number of feedback and output wires should be kept to a minimum. The designer usually is able to exercise some control over both. In particular, there are roughly as many feedback connections in a sequential system as there are binary memory elements; therefore it would be desirable to minimize the number of memory elements. At present, large systems are frequently

designed with many more memory elements than necessary, presumably because this results in simpler design procedures. It may, therefore, be desirable to find methods which lead to designs having nearly minimal numbers of memory elements, in order to make our error-correcting procedure more attractive.

VII. SUMMARY

We have described an error-correcting scheme which is generally applicable to synchronous digital systems, and which includes the triplication and vote-taking scheme as a special case. It permits systems to which it is applied to operate continuously even when faults are present and maintenance is being performed. The scheme can lead to very large increases in system reliability, but only if augmented by a maintenance routine which effects rapid repair of faults.

Two types of error-correcting codes have been discussed, Hamming codes and special codes. The Hamming codes are universally applicable, but are not minimally redundant in this application. The special codes are minimally redundant but not universally applicable, in that they have not been developed for a large range of values of p and q.

The equipment redundancy required to implement the scheme may be equivalent to at least triplication for moderately large systems, but should be less for more complex systems. It is not specifiable in simple terms and can be determined accurately only by carrying through the detailed design of the specific systems. Such detailed applications have not yet been made.

APPENDIX

Proof That 2p Is a Lower Bound on the Number of Check Bits

We shall derive this bound by showing that an upper bound on the number of code words of length pq which satisfy our error-correcting criterion is $2^{(q-2)p}$ words. This implies that the maximum number of bits which can be assigned values arbitrarily is $(q - 2)p$ bits. The remaining bits must be check bits; therefore, a lower bound on the number of check bits is $qp - (q - 2)p = 2p$ bits.

Proof That an Upper Bound on Number of Code Words Is $2^{(q-2)p}$

It is useful to think of the pq bits which comprise a code word as being arranged in a single row, with each successive block of p bits being replaced by a single symbol, D_i, which can take on any one of the 2^p

different values. In this alternative representation, a typical q-symbol word would be

$$D_1D_2D_3 \cdots D_q .$$

In terms of this representation, our error-correcting criterion requires that an error in any one of the q symbols be correctable. This implies that admissible code words must differ in more than two symbol positions. For, consider the following two words which differ only in the first two symbol positions:

$$\text{word 1:} \quad D_1D_2D_3 \cdots D_q ,$$

$$\text{word 2:} \quad D_1'D_2'D_3 \cdots D_q .$$

It is possible for an error in the first symbol of word 1 and in the second symbol of word 2 to cause both to become, for example, the word

$$D_1'D_2D_3 \cdots D_q .$$

Hence, we cannot determine whether word 1 or 2 was the correct word; therefore, admissible code words must differ in more than two symbol positions.

Let S be the set of all q-symbol words. There are 2^{qp} such words. Partition S into disjoint subsets S_i, $i = 1, 2, 3, \cdots$, such that two elements of S belong to the same subset if they are identical in the last $(q - 2)$ symbol positions. Thus there are as many subsets as there are truncated words $D_3D_4 \cdots D_q$, namely, $2^{(q-2)p}$ subsets, and each subset contains 2^{2p} elements.

Now arbitrarily choose a (q-symbol) word from subset S_1 to be a code word. Then no other words from subset S_1 can be chosen as code words, because any two words from S_1 differ only in the first two-symbol positions. By the same argument, at most one word can be selected as a code word from S_2, etc. Therefore, there cannot be more code words than subsets. Hence, an upper bound on the number of code words is $2^{(q-2)p}$.

REFERENCES

1. Ray-Chaudhuri, D. K., this issue, p. 595.
2. Von Neumann, J., Probabilistic Logics and the Synthesis of Reliable Organisms from Unreliable Components, in Shannon, C. E. and McCarthy, J., eds., *Automata Studies*, Annals of Math. Studies, No. 34, Princeton Univ. Press, Princeton, N. J., 1956, pp. 44–98 (particularly Section 8.3).
3. Moore, E. F. and Shannon, C. E., Reliable Circuits Using Less Reliable Relays, J. Frank. Inst., **262**, 1956, pp. 191; 281.
4. Hamming, R. W., Error Detecting and Error Correcting Codes, B.S.T.J., **29**, 1950, p. 147.
5. Unger, H., Hazards and Delays in Asynchronous Sequential Switching Circuits, I.R.E. Trans., **CT-6**, 1959, p. 12.
6. Cadden, W. J., Equivalent Sequential Circuits, I.R.E. Trans., **CT-6**, 1959, p. 30.