Acceptable Testing of VLSI Components Which Contain Error Correctors

RODGER A. CLIFF, MEMBER, IEEE

Abstract—**If a VLSI chip is partitioned into functional units (FU's) and redundant FU's are added, error correcting codes may be employed to increase the yield and/or reliability of the chip. Acceptable testing is defined to be testing the chip with the error corrector functioning, thus obtaining the maximum increase in yield afforded by the error correction. The acceptable testing theorem shows that the use of coding and error correction in conjunction with acceptable testing can significantly increase the yield of VLSI chips without seriously compromising their reliability.**

I. Introduction

VLSI chips which contain error correctors may be tested to be perfect (that is, tested with the error correction mechanisms either bypassed or disabled), or they may be tested to be acceptable (that is, tested with the error correction mechanisms functioning). The first case clearly produces chips of the highest reliability since the entire capability of the codes is used to correct for failures which occur after testing. The second case is of special interest, however, because one has a much higher percentage of acceptable chips than one does of perfect chips. Of course one pays for the increased yield of acceptable chips over the yield of perfect chips with a decrease in reliability. What is significant is that the reliability penalty required to obtain significant yield increase need not be severe.

It is the purpose of this paper to quantify the reliability penalty which results from employing acceptable testing of VLSI chips. A theorem is developed which relates the reliability of chips which are tested to be acceptable to the reliability of untested chips. The reliability of untested chips may be computed from standard IC fault models.

The theorem requires no assumptions about either the nature of the statistics of the defects which occur on the chip, or about the method of partitioning the chips into functional units (FU's), or about the coding and error correction methods. Whereas previous fault-tolerant-VLSI work has emphasized memory arrays on silicon wafers, acceptable testing, as presented in this paper, is applicable to arbitrarily complex and/or irregular logic and any manufacturing technology.

Preparatory to discussing the acceptable testing theorem, a brief review of the current state of the art in fault-tolerant computing as it applies to VLSI is given. This is followed by a discussion of IC fault models. It is demonstrated that consideration of statistically independent random defects only is appropriate to the discussion at hand.

For the purpose of obtaining a qualitative feel for the yield and reliability implications of acceptable testing, two specific examples are worked out. These examples demonstrate that acceptable testing can indeed be a viable technique for increasing yield without unduly sacrificing reliability.

II. Fault-Tolerant Computing as it Applies to VLSI

There is a large amount of literature on fault-tolerant computing, beginning with von Neumann [1] and Moore and Shannon [2]. By 1962, symposia were being held on the subject (Wilcox and Mann [31]). There are early books on the subject by Winograd and Cowan [4] and Pierce [5]. A particularly good review of the state of the art as of 1976 was given by Avizienis [6].

Of the recent literature on fault-tolerant computing, a large number of papers have focussed on fault-tolerant memory. Goldberg *et al.* [7] state that this is so because computer main memory has historically been a major cost item of a computer system: it is the most unreliable part of a system (except for mechanical preipherals), and it is simultaneously the most amenable to fault-tolerance techniques because of its inherent regularity and its exceedingly large number of logic elements. Early papers dealt with core memory, although some of the techniques are apropos any memory technology. More recently, the emphasis has shifted to semiconductor memory. Reviews of semiconductor memory have been given by Eimbinder [8], Riley [9], and Leuke *et al.* [10].

Two interesting papers from the standpoint of semiconductor memory were written by Srinivasan [11], [12]. One of his approaches combines triple-modular-redundancy (TMR) with matching the address decoder to the particular faulty array. The technique is very powerful; unfortunately, its difficult mechanization makes it unlikely to be an economically viable solution. Srinivasan also considers the application of generalized Hamming codes over $GF(2^b)$, where b is the number of bits per byte, as described by Peterson and Weldon [13]. Since the usual methods of correction would be too slow in comparison to the inherent speed of semiconductor memories, he mentions the parallel method described by Bossen [14], and concludes that an even faster approach must be found. Srinivasan's solution is an ingenious scheme for interlacing a

Manuscript received June 8, 1979; revised November 13, 1979.

The author is with the NASA Goddard Space Flight Center, Greenbelt, MD 20771.

single-step majority-decodable code [15], [16] in such a way that it can be used to correct burst errors.

Hsiao [17] describes the sytem used on the IBM 7030 and the IBM System/360 Model 85 main memories. A modification of the translator for the System/360 is delineated by Carter *et al.* [18] who succeed in attaining higher speed. In a related paper, Carter *et al.* [19] emphasize the importance of self-testable checker hardware.

Allen [20] stresses the bit-per-basic-operating-memory (bit/BOM) organization as a means of combatting faults in address and drive circuitry. This allows correction to be made with SEC codes rather than burst error correcting codes. Szygenda and Flynn [21]-[23] describe an unusual core switching array useful in constructing a fault-tolerant magnetic memory.

More apropos large-scale integrated (LSI) semiconductor memories, Siewiorek and McCluskey [24]-[27], and more recently Ingle and Siewiorek [28], discuss hybrid redundancy. This technique consists of a synergism of n-fold-modular-redundancy (NMR) and standby sparing. They describe an approach employing automatically switched spares wherein the switching mechanism is driven by a set of "disagreement detectors."

Taylor [29], [30] took a theoretical tack. He defined the computational capacity of a system to be the limit of the amount of computation divided by the complexity. He gives existence proofs showing that it is possible to construct a stored program computer (including memory) with nonzero capacity from faulty components.

Rao [31] described a system in which coded data passes between the memory and the processor. In the example he considered, the encoders and decoders are located in the processor. An extensive analysis of the cost, both in time and components, for three memory coding schemes was done by Varanasi (student of Rao) [32]. He analyzed the SEC Hamming code, the single-step majority-decodable code, and the two-dimensional iterated parity (cross parity) code.

Switching in of spare LSI chips is proposed by Goldberg *et al.* [33]. They describe a technique using a regular switching network which can be embedded in the LSI chips. On the other hand, Wensley *et al.* [34] treat byte error correcting codes. They develop a theory of framed burst error correcting codes, noticing that if errors are constrained to a single byte, then fewer syndromes are required than for bursts of the same length in arbitrary positions. The system they propose has the decoders on separate LSI chips, but they foresee including the decoders on the same chip as the memory elements. Burst error codes which they considered include the Hamming code over $GF(2^b)$ [13] and Abramson codes [35]. They recommend an LSI realization of the decoder after a method of Levitt and Kautz [36]. They further consider the replacement of failed memory frames with spares and derive the optimum frame width from this standpoint.

In still another paper, Neumann *et al.* [37] consider the tradeoff between interlaced parity codes and the burst error correcting arithmetic codes proposed by Neumann and Rao [38]. The redundancy of interlaced parity or arithmetic codes is found to be slightly greater than that of the more efficient Hong–Patel [39] codes; however, they prefer the arithmetic code because of its utility in the processor.

A large amount of recent work has dealt with reconfiguration (manual or automatic) to improve system performance. Mathur and de Sousa [40] present a technique which uses reconfigurable NMR. Papers by Cox and Carroll [41] and Hartwell *et al.* [42] describe approaches which involve swapping memory bit planes. Hsiao and Bossen [43] use orthogonal latin squares in reconfiguring the address lines of individual bit planes. They thus configure the system to have no more than one error per memory word. This allows single error correction–double error detection (SEC–DED) to function in the face of a large total number of faults.

A working experimental memory system which uses switchable spares has been reported by Carter and McCarthy [44]. Meanwhile, Stiffler [45] has developed a correction scheme for bit plane oriented errors. He asserts that this technique uses less hardware than spares switching. Taking a different tack, Davida and Robinson [46] propose a dual-distance decoding scheme which gains a speed advantage by first doing error detection, and then proceeding to error correction only if required.

The use of on-chip spares to increase LSI memory yield has been treated by several authors. As long ago as 1967, Tammaru and Angell [47] and Petritz [48] showed that limited discretionary wiring in combination with spares could increase memory chip yield. Schuster [49] suggested using latches, laser customizing, or an on-chip EPROM to select the spares. A total system, consisting of chips with on-board spares, automatic test equipment, and automatic laser customizing, has been reported by Cenker *et al.* [50].

The discretionary wiring techniques (even if fully automatic, as in Cenker *et al.*) necessitate special handling of individual memory chips. Aubusson and Catt [51] state that after considerable activity in the 1960's, discretionary wiring flopped in 1969 because of this handling problem. They strive for wafer scale integration (WSI) in which sufficiently many sections of a wafer are logically concatenated under external control after fabrication. Manning [52] has taken a similar approach.

Although these methods allow the construction of an array which achieves a sizable fraction of the total area of a wafer, they have the disadvantage of requiring individual (although automatic) customizing of each wafer. Of course customizing on a per wafer basis rather than a per chip basis does cut down the amount of handling. A further disadvantage of these schemes is that the customization is stored in latches on the wafer and is therefore volatile. Recustomization must be done after power interruption. This does have the advantage, however, that it permits recovery from faults which occur after the initial customization.

Although numerous authors have treated on-chip spares, the previous papers did not envision the placement of autonomous error correctors within LSI memory chips. (A partial exception is Goldberg *et al.* [7], who did propose a scheme for switchable spare LSI memory chips wherein the switches were embedded in the memory chips.)

In contrast, Cliff and Rao [53] described the yield increase which could be obtained from on-chip error correction in LSI memory chips. Subsequently, Cliff [54] showed that both yield and reliability could be simultaneously enhanced by the use of on-chip error correction. He also demonstrated that the application of concatenated codes [55], [56] was attractive. He described a memory organization using both coded chips and an overall error correction scheme. Truly remarkable performance can be achieved with this technique—so good, in fact, that one can envision constructing a usable memory system from untested chips.

III. Fault Models for VLSI

For over 15 years there has been a controversy over the proper technique to use for modeling IC faults. In 1964, Murphy [57] stated that area defects would be caught at slice test, line defects (such as scratches) would not exist in a well controlled process, and that spot defects would limit IC yield. Referring to spot defects, he said, "this is the predominant type of defect causing losses at line test and is responsible for the dependence of yield on area." He also pointed out that chip-to-chip or slice-to-slice variation of the defect density would make the model pessimistic.

Tammaru and Angell [47] agreed in 1967 that in a refined process random defects would control yield, but that a model based on random defects would be pessimistic if the defects were in reality nonrandom. Seeds [58] did an empirical analysis in 1967 which seemed to show that the random model was pessimistic.

Uniformly distributed random defects should give a yield versus area relationship of the form

$$Y = Y_0^{A/A_0} \tag{1}$$

however, in 1970, Moore [59] stated that experience at Intel showed that

$$Y = Y_0^{(A/A_0)^{1/2}} \tag{2}$$

was closer to reality.

Warner [60] showed in 1974 that he could fit Moore's data with two random distributions, each covering half the area of the slice. Warner also stated, "the investigator noting a yield affected by line or area defects and assuming that he faced only point defects, would tend to produce pessimistic projections concerning the effect of increasing IC area." He went on to say, "virtually all the good IC's come from the lowest density subarea, within which point defects are randomly distributed." In other words, once the really bad areas are removed from consideration, the mean density of defects which would be inferred from yield data is realistically low and (1) is a good fit to experiment. Warner also pointed out that the effective "cost" of a line defect varies as $A^{1/2}$ (the number of chips a line can intersect varies as $L/A^{1/2}$). It is possible that Intel's experience was colored by a large number of line defects.

Numerous attempts have been made to explain why observed variation of yield versus area does not agree with (1): Ansley, 1968 [61], Yanagawa, 1972 [62], Gupta and Lathrop, 1972 [63], Gupta *et al.*, 1974 [64], Muehldorf, 1975 [65], and Paz and Lawson, 1977 [66]. One novel attempted explanation put forth by Price in 1970 [67] was that the workers in this field were incorrectly applying Boltzman statistics, when, in fact, they should have been using Bose-Einstein statistics. Murphy refuted this assertion in 1971 [68].

In 1973, Stapper [69] did an experiment with 20 wafers each with 50 "monitors" per chip. Although he presented a sophisticated model to explain his data, he conceded that Poisson statistics (random spot defects) could not be rejected on a statistical basis. He also stated that visual examination showed that the "monitors" failed from single defects. This suggests that he was indeed seeing random defects. Again, in 1975, Stapper [70] tried several models and showed that none of them could be rejected on the basis of statistical significance.

For the purposes of the analyses presented in Sections VI and VII of this paper, it will be assumed that defects are indeed randomly distributed and statistically independent. This is justified as follows. 1) Area defects: according to Schuster [49], "gross imperfections cause large areas of the chip to be bad and are not affected by redundancy techniques." 2) Line defects: these should not occur in a well controlled process (Murphy [57]). 3) Clusters of spot defects: Muehldorf [65] wrote, "a cluster has, in essence, the same damaging effect for a chip as a solitary fault, and hence for refined chip yield predictions, a cluster should be counted like a single fault." This is indeed true for uncoded IC's. However, if the FU's are appreciably smaller than a cluster, or if the FU's are interwoven so that a cluster typically hits several FU's (in different words), then the behavior is closer to the case of independent defects. If, on the other hand, FU's are significantly larger than a cluster, then there is a certain probability that an FU will be faulty (it is irrelevant whether the fault is caused by a single defect or a cluster) and the FU's will be effectively statistically independent.

Furthermore, since it has not been unambiguously shown that clustering of spot defects is the cause of the observed deviations from (1), there seems to be no valid reason to unduly complicate the mathematics. It should also be pointed out that the results of this paper are not confined to present day technology. Clustering may or may not occur in tomorrow's technology. Another point to be made, is that the object of this paper is to demonstrate the utility of acceptable testing; the fact that the model used might be pessimistic can only make that case stronger.

With respect to failures during use (in contrast to defects discovered at initial test) the same arguments apply. Schnable *et al.* [71] state that the effect of complexity on reliability is controversial. Data presented by Koppel [72] show that in the case of Intersil 16K RAM's, 89.6 percent of operational failures involve a single bit. This is taken as evidence for statistically independent failures. Therefore random failures, as well as random manufacturing defects, will be assumed in the calculations.

Data on failure rate versus time is not given. We assume a constant failure rate for the calculations. This is no doubt unrealistic, since a classical "bathtub" curve could be expected. It does greatly simplify the calculations, and we are primarily interested in comparisons of coded versus uncoded reliability anyway, not in the absolute failure rate of chips. A varying failure rate would simply correspond to a nonuniform time axis in our examples.

IV. VLSI Chip Model and Definitions

For the purposes of this paper, a VLSI chip is modeled as a complex nonrepairable entity which may be accessed only at its input and output terminals. The chip is assumed to contain redundant hardware and one or more error correction mechanisms which function by the application of error correcting codes. The details of the error correction mechanisms and the choice of coding techniques are outside the scope of this paper.

The VLSI chip is assumed to contain multiple FU's each of unit complexity. It is assumed that redundancy is added in the form of additional FU's, again each of unit complexity. It is further assumed that either the total complexity of all the correction mechanisms is significantly less than the complexity of the aggregate of the FU's, or that the complexity of the correction mechanisms increase no worse than linearly with the number of FU's. This means that either the correction mechanisms are a "hard core" whose complexity can be ignored for the purpose of determining an upper limit to the reliability of the chip, or the complexity of the correction mechanisms can be subsumed into that of the FU's. Observe that an FU may be as simple as a single memory cell. Alternatively, an FU could be as complex as, or even more complex than, an entire microcomputer (complete with processor, memory, and I/O ports).

Let us assume that the FU's on a chip are organized into w "words," each of which contains k nonredundant FU's (required for the intended function of the chip) and n-k redundant FU's (required for error correction). It is further assumed that the error correction mechanisms function on these "words." These need not be "words" in the conventional sense. The only requirement is that at each time the outputs of the FU's are sampled, the outputs of the FU's of a given "word" relate in such a way that the error correction mechanism can function.

The strategies to be used in subdividing a complex VLSI chip into "words" and in subdividing these "words" into FU's depend on the level of complexity obtainable on a single chip and upon the type of functions to be performed. There is much room for research on this subject. It is the purpose of this paper to demonstrate the performance enhancement that can be obtained once this subdivision has been accomplished; partitioning per se will not be covered except to note that NMR at the FU level can certainly be used. Recent work by Davies and Wakerly [73] on "synchronization voting" in NMR shows how the FU's may use self-contained independent clocks. With the addition of redundant power and ground distribution, a totally redundant system is possible.

Let us define an acceptable word to be one whose outputs (after correction) are correct for all possible inputs. Similarly, let us define an acceptable chip to be one whose outputs (again, after correction) are correct for all possible inputs. In the uncoded case, all FU's on the chip must function properly in order for the chip to be acceptable. For a coded chip, however, it is only necessary that the outputs of the correction mechanism associated with each word be correct, i.e, that every word be acceptable. Observe that even for the case of single-error-correcting codes, it is possible for a chip to function properly in the face of several failed FU's if the failures occur in different words.

Let us define $\overset{*}{q}(t)$ to be the probability that a chip is acceptable at time t. Let us further stipulate that testing is performed at $t = 0$. We must consider uncoded chips, both untested and tested perfect. We must also consider coded chips in the untested, tested perfect, and tested acceptable cases. For the most part, we will be concerned with a c-fold multiple-error-correcting code, which we shall call call cMEC for convenience. (Frequently, the letter t is used to denote the number of errors which can be corrected. We choose to use c instead, because in this paper t is used to denote time.) Consideration will also be given to single-error-correcting codes (the special case $c = 1$), which we shall call SEC. Table I lists the various subscripts which will be appended to $\overset{*}{q}(t)$ to specify which case we are considering. Observe that $\overset{*}{q}(0)_{c\mathrm{MEC}\,*\,\mathrm{UT}}$ is the yield of good chips obtained at testing ($t = 0$), and that $\overset{*}{q}(0)_{c\mathrm{MEC}\,*\,\mathrm{TP}} = 1$, and $\overset{*}{q}(0)_{c\mathrm{MEC}\,*\,\mathrm{TA}} = 1$, because only perfect or acceptable chips, respectively, are retained after testing.

V. Definition and Proof of the Acceptable Testing Theorem

If a coded chip is tested perfect, then the entire capability of the code is available to correct faults which accumulate during life. If, on the other hand, a chip is tested acceptable, then there may be faulty FU's present at time $t = 0$. These faulty FU's will use up some of the error correction capability of the code. Accordingly, we expect that the probability versus time of having an acceptable word should lie somewhere between that for an untested chip and that for a tested perfect chip. We shall now explore in detail the nature of the probability versus time of having an acceptable chip.

We will first state the acceptable testing theorem for cMEC $*$ TA chips and briefly discuss its import. Subsequently a proof of its validity will be given.

Theorem:

$$\overset{*}{q}(t)_{c\mathrm{MEC}\,*\,\mathrm{TA}} = \frac{\overset{*}{q}(t)_{c\mathrm{MEC}\,*\,\mathrm{UT}}}{\overset{*}{q}(0)_{c\mathrm{MEC}\,*\,\mathrm{UT}}}. \tag{3}$$

What the theorem says is that acceptable testing does not change the functional form of the probability versus time of having an acceptable word. This functional form remains the same as in untested chips. What does happen is that the untested probability versus time of an acceptable word is multiplied by the constant required to make the tested probability at time $t = 0$ equal to 1. This constant is, of course, $[\overset{*}{q}(0)_{c\mathrm{MEC}\,*\,\mathrm{UT}}]^{-1}$. Since we know that $\overset{*}{q}(0)_{c\mathrm{MEC}\,*\,\mathrm{TA}} = 1$, the value of this constant is not a surprise; what is not obvious, however, is that $\overset{*}{q}(t)_{c\mathrm{MEC}\,*\,\mathrm{TA}}$ should have the same functional form as $\overset{*}{q}(t)_{c\mathrm{MEC}\,*\,\mathrm{UT}}$. The utility of the acceptable testing theorem is that it allows us to determine the reliability of a chip after we have used a part of the error correcting code's capability to increase yield.

We now prove the acceptable testing theorem. Let A be the event that a particular coded untested chip is acceptable at time t. From the definition of $\overset{*}{q}(t)_{c\mathrm{MEC}\,*\,\mathrm{UT}}$, it follows that

TABLE I

Case	Subscript
Uncoded, Untested	UNC*UT
Uncoded, Tested Perfect	UNC*TP
c-Fold Multiple-Error-Correcting Coded, Untested	cMEC*UT
c-Fold Multiple-Error-Correcting Coded, Tested Perfect	cMEC*TP
c-Fold Multiple-Error-Correcting Coded, Tested Acceptable	cMEC*TA
Single-Error-Correcting Coded, Untested	SEC*UT
Single-Error-Correcting Coded, Tested Perfect	SEC*TP
Single-Error-Correcting Coded, Tested Acceptable	SEC*TA

$$\overset{*}{q}(t)_{c\mathrm{MEC}\,*\,\mathrm{UT}} = P[A]. \tag{4}$$

Let B be the event that the particular chip in question was acceptable at time $t = 0$. It then follows that

$$\overset{*}{q}(0)_{c\mathrm{MEC}\,*\,\mathrm{UT}} = P[B]. \tag{5}$$

From the definition of $\overset{*}{q}(t)_{c\mathrm{MEC}\,*\,\mathrm{TA}}$, we see that

$$\overset{*}{q}(t)_{c\mathrm{MEC}\,*\,\mathrm{TA}} = P[A/B], \tag{6}$$

the conditional probability that the chip is still acceptable at time t, given that it was previously found to be acceptable at time $t = 0$. Consulting a text on probability theory [74], we find that

$$P[A/B] = \frac{P[AB]}{P[B]} \tag{7}$$

and that

$$P[AB] = P[B/A] \ P[A]. \tag{8}$$

By combining (5) and (6), we find that

$$P[A/B] = \frac{P[B/A] \ P[A]}{P[B]}. \tag{9}$$

However, since A implies B, it follows that $P[B/A] = 1$. Therefore,

$$P[A/B] = \frac{P[A]}{P[B]}. \tag{10}$$

Now we substitute (4)-(6) into (10). Q.E.D.

Notice that the acceptable testing theorem is valid regardless of whether or not the faults are correlated, and furthermore, that no assumptions need be made regarding the method of partitioning the chip into FU's or the method of error correction. However, all of these considerations do affect $\overset{*}{q}(t)_{c\mathrm{MEC}\,*\,\mathrm{UT}}$, and, hence, the final results obtained from coding.

VI. Discussion of the Theorem with Examples

The acceptable testing theorem gives us a way to obtain quantitative information on the probability versus time that a tested acceptable chip will continue to function. Let us now consider some specific examples in order to gain some qualitative insight into the utility of acceptable testing.

For the purpose of calculation of the examples, we assume that the defects on the chip are statistically independent. As we pointed out in Section II, this leads to possibly pessimistic results; however, it greatly simplifies the calculations, because if the probability that a given FU is good is q, then the probability that j FU's are all simultaneously good is q^j [74].

Let us define the probability versus time that a given FU functions properly to be $q(t)$. Then the probability versus time that a given FU has failed is

$$p(t) = 1 - q(t). \tag{11}$$

It is convenient to further express $q(t)$ as

$$q(t) = q_0 \cdot q_t \tag{12}$$

where q_0 is the value of $q(t)$ at time $t = 0$ (the time at which we test the chip), and $q_t = 1$ at time $t = 0$ and decreases monotonically thereafter. We shall find one more definition to be useful

$$p_t = 1 - q_t. \tag{13}$$

Let us define the probability versus time that a particular word of the chip is acceptable (functions properly as seen at the output of the correction mechanism) to be $\hat{q}(t)$. Then, given $\hat{q}(t)$, we can easily determine the probability $\overset{*}{q}(t)$ that an entire chip is acceptable. For a chip containing w words

$$\overset{*}{q}(t) = [\hat{q}(t)]^{w}. \tag{14}$$

In the case of an uncoded (nonredundant) chip, there are k FU's per word. Therefore,

$$\hat{q}(t)_{\mathrm{UNC}\,*\,\mathrm{UT}} = [q(t)]^{k}. \tag{15}$$

If the chip is tested perfect (there is no meaning to acceptable testing of nonredundant chips) then $q(t) = q_t$, and

$$\hat{q}(t)_{\mathrm{UNC}\,*\,\mathrm{TP}} = [q_t]^{k}. \tag{16}$$

For coded chips, however, $\hat{q}(t)$ depends on the characteristics of the code being used and the manner in which the chip was tested (whether the chip is untested, tested acceptable, or tested perfect).

Let us employ a cMEC code. If a word contains c or fewer faulty FU's then the output of the corrector will be correct and the word will be acceptable. Standard probability theory [74] tells us that the probability of exactly j functional FU's on an untested chip is

$$P_j(t) = \binom{n}{j} q(t)^{n-j} p(t)^j. \tag{17}$$

Standard probability theory also tells us that the probability of the union of statistically independent events is the sum of the probabilities of the individual events. Therefore, the probability that a given word of an untested chip is acceptable is found by summing $P_j(t)$ for $0 \leqslant j \leqslant c$.

$$\hat{q}(t)_{c\mathrm{MEC}\,*\,\mathrm{UT}} = \sum_{j=0}^{c} \binom{n}{j} q(t)^{n-j} p(t)^j. \tag{18}$$

An equivalent form which we will frequently find to be more useful is obtained by factoring out $q(t)^n$. This form appears as (19).

$$\hat{q}(t)_{c\text{MEC} * \text{UT}} = q(t)^n \sum_{j=0}^{c} \binom{n}{j} \left(\frac{p(t)}{q(t)}\right)^j. \tag{19}$$

Equations (17)-(19) apply to chips in the as-manufactured (untested) state. Let us now consider chips which have been tested. When a chip is tested perfect, the effect in the coded case is the same as that in the uncoded case, namely to force the condition $q_0 = 1$ in the chips which are not discarded. Therefore, using (12), we see that for chips which are tested perfect

$$\hat{q}(t)_{c\text{MEC} * \text{TP}} = q_t^n \sum_{j=0}^{c} \binom{n}{j} \left(\frac{p_t}{q_t}\right)^j. \tag{20}$$

The tested acceptable case (cMEC * TA) is computed from the untested case (cMEC * UT) by observing that the acceptable testing theorem applies to words as well as entire chips. The SEC cases are obtained from the cMEC cases by setting $c = 1$. Table II summarizes the results for the eight cases under consideration. Note that the SEC cases have been algebraically rearranged to aid in computation.

For the example which follows, consider each FU to have a single output, and assume that a word contains four nonredundant FU's. Using the results in Table II, Fig. 1 shows how $\hat{q}(t)$ varies as a function of $q(t)$ for two SEC codes, each with $k = 4$. These codes are the (7,4) Hamming code and an unspecified (8,4) code which is implemented such as to only correct single errors. The uncoded case and triple modular redundancy on each bit (TMR)[4] are also included for comparison.

The graph uses logarithmic coordinates; therefore, the uncoded case plots as a straight line. Notice that all the coded cases start out being better than the uncoded case, but that eventually, they all become worse than the uncoded case. It can be seen that the comparative performance of a code (the 8,4) code) which has an information rate only slightly less than that of a perfect code [the (7,4) Hamming code, or (TMR)[4]] degrades quite rapidly compared to that of the perfect code as $q(t)$ decreases. The reason, of course, is that the extra check cells associated with a nonperfect code mean increased probability of encountering a fault, with no corresponding increase in code capability.

One more comment about Fig. 1 is in order; no distinction is made between untested and tested cases. This is so because the abscissa is $q(t)$. Tested perfect chips start at the point 1,1. Untested chips start where their curve crosses $q(t) = q_0$. The tested acceptable case is not covered in this figure.

Now let us examine the effect of testing on $\hat{q}(t)$. Up to this point, no specific reliability model has been used. It is convenient to employ the constant failure rate model. It will be assumed that FU's have a constant failure rate r throughout the useful life of the chip and that FU's are manufactured with an initial probability q_0 of being good. It follows that

$$q(t) = q_0 \cdot e^{-rt}. \tag{21}$$

We have five cases to consider. The first is the uncoded tested perfect case (which has in practice been the usual case). The second is the uncoded untested case (which is not normally encountered, but is included here for comparison). The third is the coded tested perfect case. In this case all the capability of the code is used for reliability improvement, and no yield improvement is obtained. This case is an upper limit to the reliability improvement which can be obtained. The fourth case—coded tested acceptable—is the one in which we are most interested. It allows us to obtain both yield improvement and reliability improvement. The fifth case—coded untested—is of less interest in itself, but it is included because the coded tested acceptable case is derived from it.

TABLE II

Testing Case	Probability of Acceptable Word, $\hat{q}(t)$
UNC*UT	$q(t)^k = q_0^k \cdot q_t^k$
UNC*TP	q_t^k
cMEC*UT	$q(t)^n \sum_{j=0}^{c} \binom{n}{j}\left(\frac{p(t)}{q(t)}\right)^j$
cMEC*TP	$q_t^n \sum_{j=0}^{c} \binom{n}{j}\left(\frac{p_t}{q_t}\right)^j$
cMEC*TA	$\dfrac{q(t)^n \sum_{j=0}^{c} \binom{n}{j}\left(\frac{p(t)}{q(t)}\right)^j}{q_0 \sum_{j=0}^{c} \binom{n}{j}\left(\frac{p_0}{q_0}\right)^j}$
SEC*UT	$q(t)^{n-1} \left[n - (n-1)\, q(t)\right]$
SEC*TP	$q_t^{n-1} \left[n - (n-1)\, q_t\right]$
SEC*TA	$\dfrac{q(t)^{n-1} \left[n - (n-1)\, q(t)\right]}{q_0^{n-1} \left[n - (n-1)\, q_0\right]}$

The probability $\hat{q}(t)$ of an acceptable word is plotted in Fig. 2 versus normalized time rt for each of the five cases mentioned in the preceding paragraph. In this specific example, the (7,4) Hamming code was used.

The dotted straight line starting at the upper left-hand corner of the figure applies to the uncoded tested perfect case. The uppermost curve applies to the coded tested perfect case. Observe that for quite some time the curve for the coded tested acceptable case lies in between those of the two tested perfect cases. In this interval, tested acceptable chips are more likely to be functional than uncoded tested perfect chips.

Note that in Fig. 2, the curve for the coded untested case always lies below that for the uncoded tested perfect case. This would not normally be true; it happens here because an abnormally low value of q_0 has been chosen for the example. This choice was made for the purpose of spreading the curves apart to more vividly demonstrate the differences between cases. If the value of q_0 used in computing the curves of the figure had a more typical value, say 0.99, then the three curves for the three coded cases would be nearly identical and they would all lie above the line for the uncoded tested perfect case throughout most of their length.

We may further observe from Fig. 2 that the coded tested acceptable case is related to the coded untested case by a multiplicative constant. The curve for the former case is obtained by a vertical translation of the curve for the latter case. The amount of the translation is just that required to move the coded tested acceptable cases start at $\hat{q}(0) = 1$.

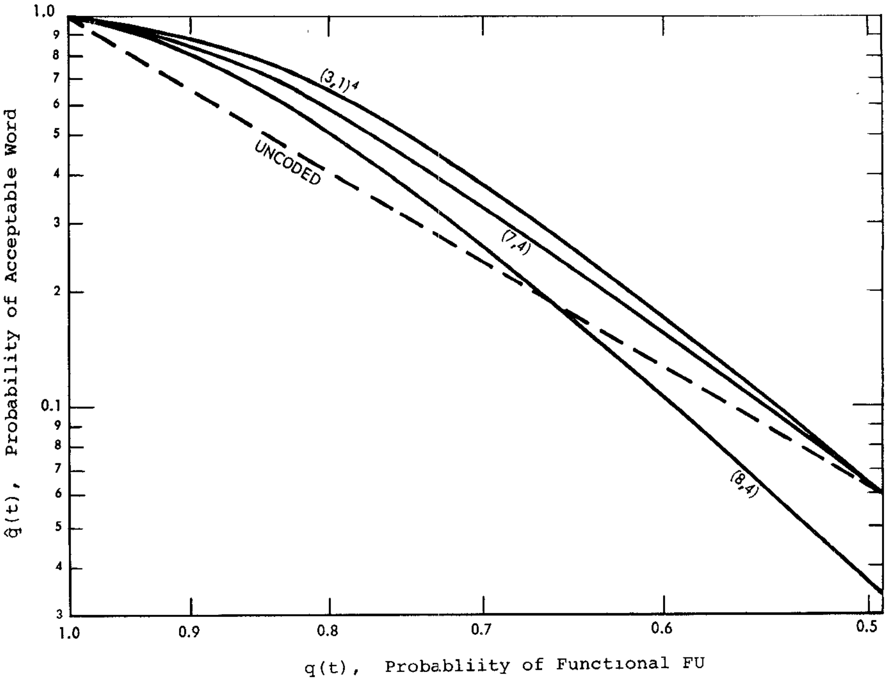


Fig. 1. Probability of acceptable 4-bit word.

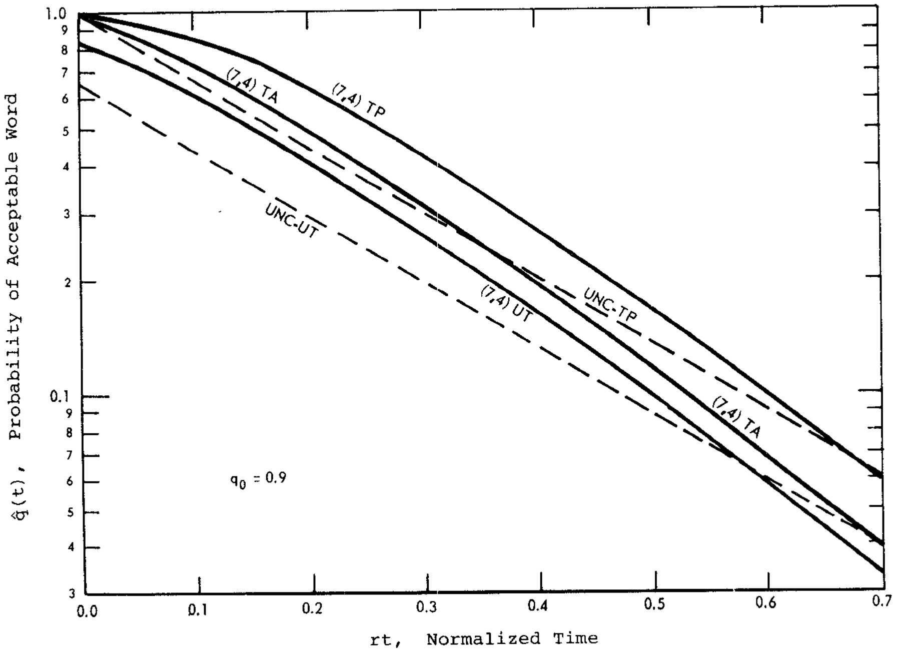


Fig. 2. Effect of testing on $\hat{q}(t)$.

Let us now consider, as a second example, a chip which uses TMR on the single bit outputs of each of $K = 1000$ FU's. Table III compares the probability $\overset{*}{q}(0)$ of having an acceptable chip at time $t = 0$ to the probability $\overset{*}{q}(1/Kr)$ of having an acceptable chip at time $t = 1/Kr$ (the time at which an uncoded tested perfect chip has probability $1/e$ of being acceptable). This comparison is made for each of the same five cases we have just discussed. However, in contrast to Fig. 2, three more realistic values of q_0 have been used. Also, an entire chip is considered in this example, rather than just one word. Observe that the performance of tested acceptable chips is nearly as good as that of tested perfect chips. This will be further discussed in the conclusion.

One final comment with respect to the last example is in order. $K = 1000$ is chosen as representative of the number of nontrivial FU's per chip which could be achieved in the not too distant future. A trivial FU is a single memory bit, for which $K = 10^5$ to $K = 10^6$ will soon be appropriate. On the other hand, if the FU's are microprocessors, then with present size wafers and present densities, we are limited to the range

TABLE III

p_0	UNCODED				TMR CODED					
	Untested		Tested Perfect		Untested		Tested Acceptable		Tested Perfect	
	$\overset{*}{q}(0)$	$\overset{*}{q}\left(\frac{1}{Kr}\right)$	$\overset{*}{q}(0)$	$\overset{*}{q}\left(\frac{1}{Kr}\right)$	$\overset{*}{q}(0)$	$\overset{*}{q}\left(\frac{1}{Kr}\right)$	$\overset{*}{q}(0)$	$\overset{*}{q}\left(\frac{1}{Kr}\right)$	$\overset{*}{q}(0)$	$\overset{*}{q}\left(\frac{1}{Kr}\right)$
.001	.368	.135	1.000	.368	.997	.988	1.000	.991	1.000	.997
.005	.067	.025	1.000	.368	.928	.898	1.000	.968	1.000	.997
.025	1.01×10^{-11}	3.68×10^{-12}	1.000	.368	.158	.137	1.000	.864	1.000	.997

$K = 100$ to $K = 1000$, even if the chips are entire wafers. At present it is not at all clear what level of complexity will be optimum for an FU. The object of the example is merely to show what kind of performance could be expected for a K which has approximately the right order of magnitude.

VII. Conclusion

There are two conclusions to be drawn from the foregoing analysis. First, when a chip has been tested perfect (uncoded or coded), its probability as a function of time of being acceptable does not depend at all on the yield obtained at the time of testing; it only depends on q_t. This is because there are no faulty FU's in a tested perfect chip. The second conclusion is that although the yield at the time of testing does affect the probability as a function of time of being acceptable for coded tested acceptable chips, this dependence is not particularly strong, even when the yield is relatively bad. We shall now consider why this should be so.

Let us examine, for instance, the last case tabulated in the previous example (Table III). There will be, on the average, 25 faulty FU's per uncoded chip ($Kp_0 = 1000 \times 0.025 = 25$). Since the statistical uncertainty associated with observing N random events is $N^{1/2}$, almost every chip will have between 20 and 30 faults. The yield of uncoded chips is abysmally bad, there being only 1 in 10^{11} which contains no faults. Coded chips, in contrast, will on the average have 75 faults each ($nKp_0 = 3 \times 1000 \times 0.025 = 75$). Therefore, almost every chip will have between 66 and 84 faults). However, the yield of acceptable coded chips is 16 percent, which is not bad when compared to 10^{-11}. The important point is that the remaining coded chips after acceptable testing will each contain between 66 and 84 faults. The testing simply eliminates all chips which have more than one fault *per word*.

If any of the words which already contains a fault from manufacturing accumulates another fault due to aging, then the chip will fail. However, since only about 8 percent of the words on the chip suffer this difficulty, the effect on the probability that the chip is acceptable is quite mild. Even in this extreme case, a tested acceptable chip has a probability of 0.864 of being acceptable at the time when an uncoded tested perfect chip has a probability of 0.368 of being acceptable. Of course, at this same time, a coded tested perfect chip would have probability 0.997 of being acceptable, but the yield of such chips is essentially zero (only one out of 10^{33} coded chips is perfect in this example)!

In conclusion, the acceptable testing theorem shows that the use of coding and error correction in conjunction with acceptable testing can significantly increase the yield of VLSI chips without seriously compromising their reliability.

Acknowledgment

The author is indebted to Dr. T. R. N. Rao who originally suggested the line of research which has evolved into the results presented here. The author also wishes to thank the reviewers for numerous helpful suggestions which have greatly improved this paper. In particular, one of the reviewers suggested the proof of the acceptable testing theorem which was presented in the paper. The author's original proof was excessively long and complicated, and it was valid only for the case of statistically independent defects.

References

[1] J. von Neumann, "1952 lectures" in *Automata Studies*, C. E. Shannon and J. McCarthy, Eds. Princeton, NJ: Princeton University Press, 1956, pp. 43–98.

[2] F. H. Moore and C. E. Shannon, "Reliable circuits using less reliable relays," *J. Franklin Inst.*, vol. 262, pp. 191–208, 281–297.

[3] *Redundancy Techniques for Computing Systems*, R. H. Wilcox and W. C. Mann, Eds. Washington, DC: Spartan, 1962.

[4] S. Winograd and J. D. Cowan, *Reliable Computation In the Presence of Noise*. Cambridge, MA: 1963.

[5] W. H. Pierce, *Failure Tolerant Computer Design*. New York: Academic, 1965.

[6] A. Avizienis, "Fault-tolerant systems," *IEEE Trans. Comput.*, vol. C-25, pp. 1304–1311, Dec. 1976.

[7] J. Goldberg, K. N. Levitt, and J. H. Wensley, "An organization for a highly survivable memory," *IEEE Trans. Comput.*, vol. C-23, pp. 693–705, July 1974.

[8] *Semiconductor Memories*, J. Eimbinder, Ed. New York: Wiley, 1971.

[9] W. B. Riley, "Special report: Semiconductor memories are taking over data-storage application," *Electronics*, vol. 46, Aug. 2, 1973.

[10] G. Luecke, J. P. Mize, and W. N. Carr, *Semiconductor Memory Design and Application*. New York: McGraw-Hill, 1973.

[11] C. V. Srinivasan, "Codes for error correction in high-speed memory systems—Part I: Correction of cell defects in integrated memories," *IEEE Trans. Comput.*, vol. C-20, pp. 882–888, Aug. 1971.

[12] ——, "Codes for error correction in high-speed memory systems—Part II: Correction of temporary and catastrophic errors," *IEEE Trans. Comput.*, vol. C-20, pp. 1514–1520, Dec. 1971.

[13] W. W. Peterson and E. J. Weldon, Jr., *Error Correcting Codes*, 2nd ed. Cambridge, MA: MIT Press, 1972.

[14] D. C. Bossen, "*b*-adjacent error correction," *IBM J. Res. Develop.*, vol. 14, pp. 402–408, 1970.

[15] R. G. Gallager, "Low density parity check codes," Sc.D. dissertation, Massachusetts Institute of Technology, Cambridge, MA, 1960; MIT Press Res. Monograph 21.

[16] J. L. Massey, "Threshold decoding," MIT Press Res. Monograph 20, 1963.

[17] M. Y. Hsiao, "A class of optimal minimum odd-weight-column

SEC–DED codes," *IBM J. Res. Develop.*, vol. 14, pp. 395–401, July 1970.

[18] W. C. Carter, K. A. Duke, and D. C. Jessep, "Look-aside techniques for minimum circuit memory translators," in *Dig. 1972 Int. Symp. on Fault Tolerant Computing*, June 1972.

[19] W. C. Carter, D. C. Jessep, and A. Wadia, "Error free decoding for failure tolerant memories," in *Proc. IEEE Int. Comput. Group Conf.*, June 1970, pp. 229–239.

[20] C. A. Allen, "Design of digital memories that tolerate all classes of defects," Tech. Rep. AFAL-TR-66-223 or SU-SEL-66-031, 1966. Prepared under USAF Contract AF 33(657)-11586, Stanford Electronics Laboratories, Stanford, CA.

[21] S. A. Szygenda and M. J. Flynn, "Failure analysis of memory organizations for utilization in a self-repair memory system," *IEEE Trans. Rel.*, vol. R-20, pp. 64–70, 1971.

[22] ——, "Coding techniques for failure recovery in a distributive modular memory organization," in *AFIPS Proc. of the SJCC*, May 1971.

[23] ——, "Self-diagnosis and self-repair in memory: An integrated system approach," *IEEE Trans. Rel.*, vol. R-22, pp. 2–12, Apr. 1973.

[24] D. P. Siewiorek and E. J. McCluskey, "Switch designs for hybrid redundancy," Digital Systems Lab., Stanford Univ., Tech. Note 13, 1972; presented at the *Comput. Syst. Des. 1972 West Conf.*, Anaheim, CA, Feb. 1972.

[25] ——, "An iterative cell switch design for hybrid redundancy," Stanford Electronic Laboratories, Stanford, CA, Tech. Rep. 20, Dec. 1971; SU-SEL-71-064, Dec. 1971.

[26] ——, "A measure of switch complexity in systems with standby spares," Stanford Electronic Laboratories, Stanford, CA, Tech. Rep. 21, 1971; SU-SEL-71-065, Dec. 1971.

[27] ——, "An iterative cell switch design for hybrid redundancy," *Dig. 1972 Int. Symp. on Fault Tolerant Computing*, June 1972.

[28] A. D. Ingle and D. P. Siewiorek, "A reliability model for various switch designs in hybrid redundancy," *IEEE Trans. Comput.*, vol. C-25, pp. 115–133, Feb. 1976.

[29] M. G. Taylor, "Reliable information storage in memories designed from unreliable components," *Bell Syst. Tech. J.*, vol. 47, pp. 2299–2237, Dec. 1968.

[30] "Reliable computation in computing systems designed from unreliable components," *Bell Syst. Tech. J.*, vol. 47, pp. 2239–2366, Dec. 1968.

[31] T. R. N. Rao, "Use of error correcting codes on memory words for improved reliability," *IEEE Trans. Rel.*, vol. R-17, pp. 91–96, June 1968.

[32] M. R. Varanasi, "Systematic codes and their applications to computer systems," Ph.D. dissertation, Univ. Maryland, College Park, MD, 1973.

[33] J. Goldberg, K. N. Levitt, and J. H. Wensley, "An organization for a highly survivable memory," in *Dig. 1973 Int. Symp. on Fault Tolerant Computing*, Palo Alto, CA, June 1973, pp. 59–73.

[34] J. H. Wensley, K. N. Levitt, M. W. Green, J. Goldberg, and P. G. Neumann, "Design of a fault tolerant airborne digital computer, volume I–Architecture," Stanford Res. Inst., Menlo Park, CA, Final Rep. Contr. Oct. 1973.

[35] N. M. Abramson, "A class of systematic codes for non independent errors," *IRE Trans. Inform. Theory*, vol. IT-5, pp. 150–157, Dec. 1959.

[36] K. N. Levitt and W. H. Kautz, "Cellular arrays for the parallel implementation of binary error correcting codes," *IEEE Trans. Inform. Theory*, vol. IT-15, pp. 597–607, Sept. 1969.

[37] P. G. Neumann, K. N. Levitt, J. Goldberg, and J. H. Wensley, "A study of fault tolerant computing," Stanford Res. Inst., Menlo Park, CA, Final Rep. Contr. N00014-72-C-0254.

[38] P. G. Neumann, and T. R. N. Rao, "Error correction in byte organized arithmetic processors," in *Dig. 1973 Int. Symp. on Fault Tolerant Computing*, Palo Alto, CA, June 1973, pp. 53–58.

[39] S. J. Hong and A. Patel, "A general class of maximal codes for computer applications," *IEEE Trans. Comput.*, vol. C-21, pp. 1322–1331, Dec. 1973.

[40] F. P. Mathur and P. T. de Sousa, "Reliability models of NMR systems," *IEEE Trans. Rel.*, vol. R-24, pp. 108–113, June 1975.

[41] G. W. Cox and B. D. Carroll, "Reliability modeling and analysis of fault-tolerant memories," *IEEE Trans. Rel.*, vol. R-27, pp. 49–54, Apr. 1978.

[42] W. T. Hartwell, C. W. Hoffner, and W. N. Toy, "A fault tolerant memory for duplex systems," *IEEE Trans. Rel.*, vol. R-27, pp. 134–138, June 1978.

[43] M. Y. Hsiao, and D. C. Bossen, "Orthogonal latin square configuration for LSI memory yield and reliability enhancement," *IEEE Trans. Comput.*, vol. C-24, pp. 512–516, May 1975.

[44] W. C. Carter and C. E. McCarthy, "Implementation of an experimental fault-tolerant memory system," *IEEE Trans. Comput.*, vol. C-25, pp. 557–568, June 1976.

[45] J. J. Stiffler, "Coding for random access memories," *IEEE Trans. Comput.*, vol. C-27, pp. 526–531, June 1978.

[46] G. I. Davida and J. P. Robinson, "Detection-correction decoding for system reliability," *IEEE Trans. Rel.*, vol. R-19, pp. 188–190, Nov. 1970.

[47] E. Tammaru and J. B. Angell, "Redundancy for LSI yield enhancement," *IEEE J. Solid-State Circuits*, vol. SC-2, pp. 172–182, Dec. 1967.

[48] R. L. Petritz, "Current status of LSI technology," *IEEE J. Solid-State Circuits*, vol. SC-2, pp. 130–147, Dec. 1967.

[49] S. E. Schuster, "Multiple word/bit line redundancy for semiconductor memories," *IEEE J. Solid-State Circuits*, vol. SC-13, pp. 698–703, Oct. 1978.

[50] R. P. Cenker *et al.*, "A fault-tolerant 64K dynamic random-access memory," *IEEE Trans. Electron Devices*, vol. ED-26, pp. 853–860, June 1979.

[51] R. Aubusson and I. Catt, "Wafer scale integration–A fault-tolerant procedure," *IEEE J. Solid-State Circuits*, vol. SC-13, pp. 339–344, June 1978.

[52] F. B. Manning, "An approach to highly integrated, computer maintained cellular arrays," *IEEE Trans. Comput.*, vol. C-26, pp. 536–552, June 1977.

[53] R. A. Cliff and T. R. N. Rao, "Improving the yield of LSI memory chips by the application of coding," in *Proc. 8th Annu. Princeton Conf. Inform. Sci. Syst.*, 1974, pp. 386–390.

[54] R. A. Cliff, "The application of coding to improve the yield and reliability of semiconductor memories," Ph.D. dissertation, Univ. Maryland, College Park, MD, 1974. (Available from University Microfilms, Ann Arbor, MI.)

[55] G. D. Forney, "Concatenated codes," Sc.D. dissertation, Massachusetts Inst. Tech., Cambridge, MA, 1965; MIT Press Research Monograph 37, 1966.

[56] L. S. Schwartz, *Principles of Coding Filtering and Information Theory*. Washington, DC: Spartan, 1963.

[57] B. T. Murphy, "Cost-size optima of monlithic integrated circuits," *Proc. IEEE*, pp. 1537–1545, Dec. 1964.

[58] R. B. Seeds, "Yield, economic and logistic models for complex digital arrays," in *IEEE Int. Convention Rec.*, pt. 6, Mar. 20–23, 1967, pp. 60–61.

[59] G. E. Moore, "What level of integration is best for you?," *Electronics*, pp. 126–130, Feb. 16, 1970.

[60] R. M. Warner, Jr., "Applying a composite model to the IC yield problem," *IEEE J. Solid-State Circuits*, vol. SC-9, pp. 86–95, June 1974.

[61] W. G. Ansley, "Computation of integrated circuit yields from the distribution of slice yields for individual devices," *IEEE Trans. Electron Devices*, vol. ED-15, pp. 405–406, June 1968.

[62] T. Yanagawa, "Yield degradation of integrated circuits due to spot defects," *IEEE Trans. Electron Devices*, vol. ED-19, pp. 190–197, Feb. 1972.

[63] A. Gupta and J. W. Lathrop, "Yield analysis of large integrated-circuit chips," *IEEE J. Solid-State Circuits*, vol. SC-7, pp. 389–395, Oct. 1972.

[64] A. Gupta, W. A. Porter, and J. W. Lathrop, "Defect analysis and yield degradation of integrated circuits," *IEEE J. Solid-State Circuits*, vol. SC-9, pp. 96–103, June 1974.

[65] E. I. Muehldorf, "Fault clustering: Modeling and observation on experimental LSI chips," *IEEE J. Solid-State Circuits*, vol. SC-10, pp. 237–244, Aug. 1975.

[66] O. Paz and T. R. Lawson, Jr., "Modification of Poisson statistics: Modeling defects induced by diffusion," *IEEE J. Solid-State Circuits*, vol. SC-12, pp. 540–546, Oct. 1977.

[67] J. E. Price, "A new look at yield of integrated circuits," *Proc. IEEE*, vol. 58, pp. 1290–1291, Aug. 1970.

[68] B. T. Murphy, "Comments on 'A new look at yield of integrated circuits,' " *Proc. IEEE*, vol. 59, p. 1128, July 1971.

[69] C. H. Stapper, Jr., "Defect density distribution for LSI yield calculations," *IEEE Trans. Electron Devices*, vol. ED-20, pp. 655-657, July 1973.

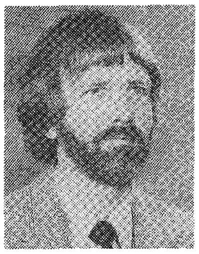
[70] C. H. Stapper, Jr., "On a composite model to the IC yield problem," *IEEE J. Solid-State Circuits*, vol. SC-10, pp. 637-639, Dec. 1975.

[71] G. L. Schnable, H. J. Ewald, and E. S. Schlegel, "MOS integrated circuit reliability," *IEEE Trans. Rel.*, vol. R-21, pp. 12-19, Feb. 1972.

[72] R. Koppel, "RAM reliability in large memory systems—Improving MTBF with ECC," *Comput. Design*, Mar. 1979.

[73] D. Davies and J. F. Wakerly, "Synchronization and matching in redundant systems," *IEEE Trans. Comput.*, vol. C-27, pp. 531-539, June 1978.

[74] W. Feller, *An Introduction to Probability Theory and Its Applications*, vol. I, 2nd ed. New York: Wiley, 1950.



Rodger A. Cliff (S'60–M'62) was born in Washington, DC. He received the B.S. degree in electrical engineering from the Massachusetts Institute of Technology, Cambridge, and the M.S. and Ph.D. degrees in electrical engineering from the University of Maryland, College Park, with minors in computer science, physics and astronomy, and math.

Since 1961, he has been employed by the NASA Goddard Space Flight Center, Greenbelt, MD. He did early work in source encoding and special purpose digital computers for Explorer class spacecraft. In 1966 he designed and constructed a demonstration model stored program computer, the SDP-1. The demonstration was successful and be became principal Investigator for a flight experiment to demonstrate the utility of stored program computers on Explorer class spacecraft. The result was the SDP-3 computer, which he designed, and which was flown successfully on Explorer 43 in 1971. He has lectured in France on spacecraft computers and he served as a consultant on spacecraft computers to the Dutch government for the Astronomical Netherlands Satellite project. He was intimately involved with the successful introduction of CMOS IC's into spacecraft use (on the Atmosphere Explorer series). For several years he was Systems Engineer for the Geosynchronous Operational Environmental Satellite (GOES) project. He is presently with the Preliminary Systems Design Group. Since 1974 he has been performing research on *in situ* measurement of space radiation damage to electronic components. He was Principal Investigator for the Component Radiation Effects Measurement (CREM-I) experiment which flew on Explorer 55. He is presently coinvestigator for the CREM-II experiment, which is in the breadboard stage. CREM-II will measure *in situ* space radiation damage to CMOS and I^2L microprocessors, and other components. His present research interests include ultra large-scale integration (ULSI), artificial intelligence, computer architecture, and fault-tolerant computing.

Dr. Cliff is a member of the Association for Computing Machinery and the American Association for the Advancement of Science.

A High-Speed Microprogrammable Digital Signal Processor Employing Distributed Arithmetic

JAN ZEMAN AND H. TROY NAGLE, JR., SENIOR MEMBER, IEEE

Abstract—**This paper describes a general-purpose digital-signal processor which is constructed with 4 bit bipolar microprocessor slices. The signal processor is microprogrammable and contains special features which allow it to employ distributed arithmetic. Hence, the processor can achieve high sampling rates without using a hardware multiplier unit. The processor's architecture is presented and its micro-order structure is examined. The processor wordlength is 16 bit; its basic cycle time, 300 ns; its data memory size, 2K words; its control store size, 256 × 56 bits. It consumes 48 W of power and has special address processing hardware. Experimental results with a twelfth-order digital filter are demonstrated. The signal processor is also compared with several other signal processors of its class described in the literature.**

Manuscript received March 23, 1979; revised September 17, 1979. This work was supported by the Stiftung Hasler-Werke Arbeitsgemeinschaft für Elektrische Nachrichtentechnik (AGEN), Bern, Switzerland.

J. Zeman is with the Institute of Telecommunications, Swiss Federal Institute of Technology, Zurich, Switzerland.

H. T. Nagle, Jr. is with the Department of Electrical Engineering, Auburn University, Auburn, AL 36830.

I. Introduction

In recent years many different digital signal processors have been described in the literature. They range in complexity from small special-purpose processors to implement digital filters, to large high-speed programmable fast Fourier transform (FFT)-oriented machines. In this paper, we will consider machines which fall in the middle range between these two classes. Specifically, we will examine microprogrammable processors whose purpose is high-speed digital signal processing. Even with these restrictions there are still many such processors described in the literature, some designs only exist on paper [1]-[7] while others have been constructed and operated [8]-[21].

In earlier years, the processors were constructed with off-the-shelf SSI and MSI components, resulting in large part counts, high power consumption, and expensive construction costs.