On Area and Yield Considerations for Fault-Tolerant VLSI Processor Arrays

ISRAEL KOREN, MEMBER, IEEE, AND MELVIN A. BREUER, SENIOR MEMBER, IEEE

Abstract—**Fault-tolerance is undoubtedly a desirable property of any processor array. However, increased design and implementation costs should be expected when fault-tolerance is being introduced into the architecture of a processor array.**

When the processor array is implemented within a single VLSI chip, these cost increases are directly related to the chip silicon area. Thus, the increase in area should be weighed against the improved performance of the gracefully degrading fault-tolerant processor array. In addition, a larger chip area might reduce the wafer yield to an unacceptable level making the use of fault-tolerant VLSI processor arrays impractical.

The objective of this paper is to devise performance measures for the evaluation of the effectiveness and area utilization of various fault-tolerant techniques. Another goal is to analyze the reduction in wafer yield and investigate the possibility of yield enhancement through redundancy.

Index Terms—**Area utilization, computational availability, fault-tolerance, processor array, reconfiguration strategies, redundancy, VLSI, wafer yield.**

I. Introduction

It has been suggested in [1] to incorporate fault-tolerance and self-testing into the architecture of a processor array within a single VLSI chip in order to achieve the following two goals:

1) verify the proper operation of the array, and

2) provide the ability to remain operational, possibly with some degradation in performance, in the presence of one or more faulty processors.

The inclusion of hardware to achieve fault-tolerance in the architecture of a processor array requires an increase in hardware complexity, and hence in the area of each processor. This increase in area might result in a decrease in the number of processors that can fit into the initial chip area. In addition, to achieve graceful degradation of the processor array when a single processor fails, we have to identify the faulty processor and reconfigure the array to avoid its use. In many cases such a reconfiguration necessitates giving up the use of some other processors which are operating correctly.

Manuscript received November 8, 1982; revised May 6, 1983. This work was supported in part under VHSIC Phase 3 Contract N00039-80-C-0641, administered by the Department of the Navy, Naval Electronics Systems Command (ONR), Pasadena, CA.

I. Koren is with the Computer Science Division, University of California at Berkeley, Berkeley, CA 94720, on leave from the Department of Electrical Engineering, Technion—Israel Institute of Technology, Haifa 32000, Israel.

M. A. Breuer is with the Department of Electrical Engineering Systems, University of Southern California, Los Angeles, CA 90007.

In Section II of this work we investigate the cost in chip area of including fault-tolerance in the architecture of a processor array. Since various fault-tolerant techniques are possible, we need means to evaluate these techniques and choose the one for which the chip area is best utilized.

Another consequence of an increase in the chip area might be a reduction in the wafer yield. At present, the yield of some large area chips is 15 percent and lower, and a further reduction in the wafer yield might lower it to levels which are unacceptable to the semiconductor industry.

In Section III we analyze the reduction in yield due to the inclusion of fault-tolerance and we investigate the possibility of yield enhancement through added redundancy.

II. Area Considerations

Fault-tolerance strategies which do not mask the effect of faults by massive redundancy employ two steps. The first one is testing of the array to identify the faulty processing element (PE). The second is system reconfiguration to avoid the use of the faulty PE. The testing can be done externally or internally (self-testing). For large processor arrays, external testing is either time-consuming, incomplete (there is no access to internal points), or both if the testing is performed periodically. In the case of continuous external testing (e.g., external duplication of the array) the large detection latency (i.e., the time period from fault occurrence until the error manifests itself externally) may prohibit the system error recovery since some of the data needed for recovery may be unavailable. Consequently, internal testing is preferred [1], i.e., the testing of each processor is done locally either by the processor itself (e.g., a self-checking processor), some of its neighbors, or both. Such a distributed testing strategy can be characterized by the following two parameters.

1) The amount of additional area needed. Let A_o denote the area of the basic PE with no fault-tolerant capabilities (simplex PE). If β denotes the ratio of the area of a fault-tolerant PE and the basic PE, then βA_o is the area of a fault-tolerant PE.

2) The coverage probability, i.e.,

$$p = \Pr\{\text{The system recovers (reconfigures) successfully/a fault has occurred}\}.$$

The coverage probability mainly depends on the fault coverage, i.e., the percentage of faults detected by the testing technique. In addition, p also depends on the detection latency,

0018-9340/84/0100-0021$01.00

and a large detection latency results in a substantial increase in the probability that the program integrity will be lost.

Example: If each PE is duplicated and the results obtained by the two PE's are compared before being transmitted to neighboring (duplicated) PE's, then $\beta = 2$. In this case, the fault coverage is almost 100 percent and the detection latency is very small. Thus, the coverage probability is nearly 1.

To evaluate the effectiveness of the various fault-tolerant strategies we need some performance measures. One may be *reliability*, $R(t)$, which is the probability that the system operates correctly in the time interval $[o, t]$. Since the array is configured after every fault is detected resulting in a lower computational capacity, we suggest to consider as well *computational availability*, $A_c(t)$, which is the expected available computational capacity [2], [3]. However, these two measures do not take into account the additional area needed when fault-tolerance is introduced into the system. Hence, we define an *area utilization* measure, $U(t)$, in the following way:

$$U(t) = \frac{\text{computational availability } A_c(t)}{\text{total chip area } A}.$$

To calculate these performance measures, we would like to model the fault-tolerant processor array in a way which is independent of the specific technique used. We propose the general Markov model depicted in Fig. 1. In this model, F is the system failure state, while at any other state (i, j) the system is operational in the presence of i faulty processors and j faulty connections. There are states in which the system is incapable of tolerating any more faults. These states, called terminal states, are determined by the reconfiguration strategy employed in the processor array and by the computational capacity requirements of the system. Clearly, if no reconfiguration is provided (o, o) is the only terminal state. In Fig. 1, (m, o), $(m-1, 1), \cdots$, and (o, k) are terminal states. Here, $m(k)$ is the largest number of faulty processors (connections) that the system can tolerate when no faulty connections (processors) are present.

A transition from a nonterminal state (i, j) to the state F takes place when an additional fault occurs and the system fails to recover from its effects. The corresponding transition rate is denoted by $\alpha_{i,j}^{F}$. Similarly, $\alpha_{i,j}^{i+1,j}$ and $\alpha_{i,j}^{i,j+1}$ are the transition rates from state (i, j) to states $(i+1, j)$ and $(i, j+1)$, respectively.

Let

$P_{i,j}(t) = Pr\{\text{The system is in state } (i,j) \text{ at time } t / \text{The system was initially in state } (o,o)\}$

with $P_{o,o}(0) = 1$ and $P_{i,j}(0) = 0$ for $(i, j) \neq (o, o)$.

The Markov model in Fig. 1 is described then by the following differential equations.

$$\frac{dP_{o,o}(t)}{dt} = -\alpha_{o,o}P_{o,o}(t) \tag{1}$$

$$\frac{dP_{i,j}(t)}{dt} = -\alpha_{i,j}P_{i,j}(t) + \alpha_{i-1,j}^{i,j}P_{i-1,j}(t) + \alpha_{i,j-1}^{i,j}P_{i,j-1}(t) \tag{2}$$

where

$$\alpha_{i,j} = \alpha_{i,j}^{i+1,j} + \alpha_{i,j}^{i,j+1} + \alpha_{i,j}^{F}. \tag{3}$$

We denote by s_v any state at level v in Fig. 1, i.e., any state (i, j) satisfying $i + j = v$. Thus, the sequence $s_0, s_1, s_2, \cdots, s_v$ corresponds to a path in Fig. 1 from the state (o, o) to a state at level v.

Using this notation, the solution of (1) and (2) under the condition

$$\alpha_{i,j} \neq \alpha_{a,b} \text{ for all } (a,b) \neq (i,j)$$

which is satisfied in most practical cases, is

$$P_{i,j}(t) = \sum_{\text{A path } s_0,s_1,\cdots,s_{i+j-1},(i,j)} \alpha_{s_0}^{s_1}\alpha_{s_1}^{s_2}\cdots\alpha_{s_{i+j-1}}^{i,j} \sum_{u=0}^{i+j} \frac{e^{-\alpha_{s_u}t}}{\prod_{\substack{v=o \\ v\neq u}}^{i+j} (\alpha_{s_v} - \alpha_{s_u})} \tag{4}$$

and

$$P_{o,o}(t) = e^{-\alpha_{o,o}t}. \tag{5}$$

The summation in (4) is over all $\binom{i+j}{i}$ paths (each of length $i+j+1$) in the Markov model (Fig. 1) from state (o, o) to the state (i, j).

The probability that the system is in state F is clearly

$$P_F(t) = 1 - \sum_{i,j} P_{i,j}(t) \tag{8}$$

where the summation is over all states (i, j) in the Markov model for the system.

The expressions for the above introduced performance measures are [2]

$$R(t) = \sum_{i,j} P_{i,j}(t) \tag{7}$$

$$A_c(t) = \sum_{i,j} c_{i,j}P_{i,j}(t) \tag{8}$$

$$U(t) = \frac{A_c(t)}{A} \tag{9}$$

where $c_{i,j}$ is the computational capacity of the system in state (i, j), expressed for example in instructions per time unit. The computational capacity depends mainly on the number of processors available for computation in state (i, j). This number is at most $N - i$ processors (where N is the number of processors in the fault-free array), and is determined by the reconfiguration strategy. In addition, $c_{i,j}$ depends on the current system structure and application since not all processors within the array are utilized in every possible structure or application. Other system factors like work-load [3] may influence the computational capacity. For simplicity, we assume in what follows that $c_{i,j}$ depends only on the reconfiguration strategy and system structure.

We define a normalized computational capacity factor which is the ratio of the array's computational capacity in state (i, j) to the initial computational capacity in state (o, o). The normalized factors satisfy

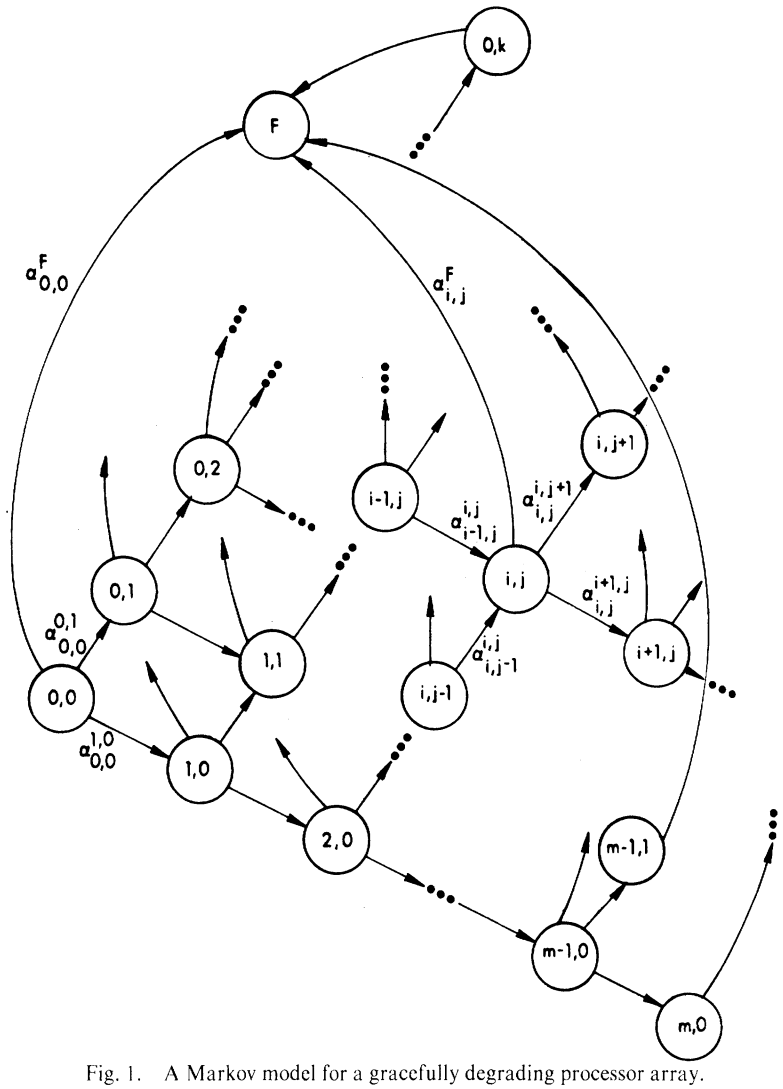


Fig. 1. A Markov model for a gracefully degrading processor array.

$$0 \le \max\{c_{i+1,j}, c_{i,j+1}\} \le c_{i,j} \le \min\{c_{i-1,j}, c_{i,j-1}\} \le c_{o,o} = 1 \quad (10)$$

and as a result

$$0 \le A_c(t) \le 1. \quad (11)$$

Let $c_{\min}$ denote the lowest computational capacity at which the system can still function. Thus, a state (a, b) is included in the Markov model of the system iff $c_{a,b} \ge c_{\min}$. This inequality also determines the terminal states of the system.

Example: The processor array presented in [1] is a rectangular grid in which every cell (PE) is connected to its four immediate neighbors. This grid may be structured as a linear array, a square array or a binary tree and appropriate structuring algorithms have been presented in [1]. When a fault in a processor or a connecting bus (between processors) occurs, the following strategy has been suggested. In the first step the fault is located. If the fault occurs within a processor, then all the processors in the corresponding row and column are declared as connecting elements (CE's) and do not participate in later processing. If a connecting bus is faulty, only the processors in the appropriate row or column turn into connecting elements. In the second step the rectangular grid is restructured. Fig. 2 depicts a binary tree structured on a grid with a faulty processor. For such a binary tree the computational capacity factor is

$$c_{i,j} = 2^{-2([\log_2(\sqrt{N}+1)]-[\log_2(\sqrt{N}-i-j/2+1)])}. \quad (12)$$

where N is the number of PE's in the grid.

If the rectangular grid is simply used as a square array we obtain,

$$c_{i,j} = \left[1 - \frac{i + \frac{j}{2}}{\sqrt{N}}\right]^2 \quad (13)$$

Similar expressions can be derived for other reconfiguration strategies in square grids and other grid topologies like triangular, hexagonal and octagonal grids.

The transition rates $\alpha^{i+1,j}_{i,j}$, $\alpha^{i,j+1}_{i,j}$ and $\alpha^{F}_{i,j}$ in Fig. 1 depend upon the failure rate of a single processor, the failure rate of the commmunication bus between two adjacent processors, and the coverage probability. We adopt here the common assumption that failures obey a Poisson distribution with a time-independent failure rate. This failure rate depends on the complexity of the processor (or the communication bus), and hence on the silicon area occupied by the processor (or the

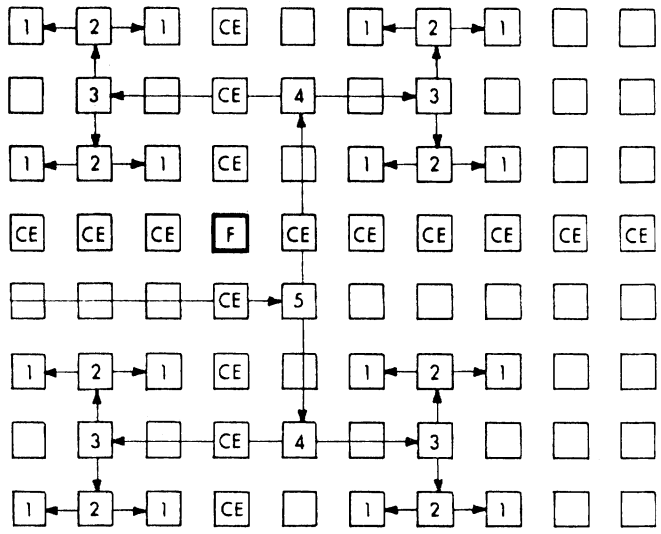


Fig. 2. A five-level binary tree structured on a grid (F denotes a faulty processor and CE denotes a connecting element).

communication bus). As a first approximation we assume a linear dependence between the failure rate and the area. Hence, if λ_o is the failure rate of a simplex PE with area A_o, then $\beta\lambda_o$ is the failure rate of a fault-tolerant PE with area βA_o. We denote by λ_c the failure rate of the communication bus connecting two adjacent PE's.

When calculating the transition rate $\alpha_{i,j}^{i+1,j}$ (i.e., an additional faulty PE), the failure rate $\beta\lambda_o$ has to be multiplied by the number of active processors serving as PE's. While for calculating $\alpha_{i,j}^{i,j+1}$ (i.e., an additional faulty connection), the failure rate $\beta\lambda_o$ has to be multiplied by the number of processors serving as connecting elements, and the rate λ_c has to be multiplied by the number of active communication buses.

The exact form of the expressions for the various transition rates depend upon the chosen reconfiguration strategy for the given topology of the processing array. For a square processing array and the strategy suggested in [1] we obtain in a straightforward manner

$$\alpha_{i,j}^{i+1,j} = \left(\sqrt{N} - i - \frac{j}{2}\right)^2 \beta\lambda_0 p. \tag{14}$$

$$\alpha_{i,j}^{i,j+1} = 2\left(\sqrt{N} - i - \frac{j}{2}\right)\left[\left(i + \frac{j}{2}\right)\beta\lambda_0 + (\sqrt{N} - 1)\lambda_c\right] p. \tag{15}$$

The third transition rate, namely $\alpha_{i,j}^{F}$, is related to the previous ones in the following way,

$$\alpha_{i,j}^{F} = \frac{1-p}{p}\left(\alpha_{i,j}^{i+1,j} + \alpha_{i,j}^{i,j+1}\right). \tag{16}$$

Substituting these transition rates into (3) yields

$$\alpha_{i,j} = \left(\sqrt{N} - i - \frac{j}{2}\right)\left[\left(\sqrt{N} + i + \frac{j}{2}\right)\beta\lambda_o + 2(\sqrt{N} - 1)\lambda_c\right]. \tag{17}$$

Equations (14)–(17) can now be substituted in expression (4) for $P_{i,j}(t)$. The latter can then be substituted into (7)–(9) to obtain expressions for the three performance measures.

The reliability, computational availability and area utilization measures can be used for comparison of various testing strategies characterized by the parameters β and p, and reconfiguration strategies characterized by the $c_{i,j}$'s.

Example: Continuing the previous example we consider now a square array of 10×10 processors and compare three possible designs. The first one is an array with simplex PE's, i.e., no fault-tolerance and no reconfiguration capabilities. The second is an array with self-checking PE's for which $\beta \simeq 1.4$ and $p \simeq 0.7$ [4]. The third is an array with duplicated PE's for which $\beta \simeq 2$ and $p \simeq 1$ [5]. For the second and third designs we use the reconfiguration strategy proposed in [1] and as a result

$$c_{i,j} = \left(1 - \frac{i + \frac{j}{2}}{10}\right)^2.$$

Suppose that a given application can still use the array if its size is reduced to 7×7 (with degraded performance). The resulting Markov model for the second and third designs is shown in Fig. 3. In it, all states (i, j) satisfying $i + j/2 \leq 3$ are included, e.g., the array can tolerate at most three faulty PE's or six faulty communication buses.

The reliability $R(t)$, computational availability $A_c(t)$ and area utilization $U(t)$ for the three designs are plotted in Figs. 4 and 5. In these graphs the time has been normalized and one time unit is equal to the average lifetime of a simplex PE, i.e. $1/\lambda_o$. As might be expected, the array with duplicated PE's has the highest reliability and computational availability. However, it has the lowest area utilization. The self-checking array has a better area utilization, even higher than the simplex array, for longer mission times.

Similar analysis can be done for other topologies of processing arrays in order to compare various fault-tolerant techniques.

III. Yield Considerations

The introduction of fault-tolerance into the architecture of a single chip processing array results in an increase in the hardware complexity and therefore in the chip area. The yield of large chips in the current technology is 15 percent and lower, and a further increase in the area may reduce the wafer yield to unacceptable levels. Yield enhancement can be achieved by introducing redundancy into the chip so that faulty cells (PE's) can be tolerated. This idea of using redundancy to enhance yield is already being employed by several manufacturers of VLSI memory chips (64 kbit) [6], [7]. The technique used consists of adding extra word lines and bit lines which are available to replace failing lines and to bypass failing bit cells using laser programming [7]. Other faults, such as defects in address lines, clock lines and power lines are considered fatal and are not correctable by redundancy. In this section we analyze the reduction in yield due to the inclusion of fault-tolerance and examine the possibility of yield enhancement using redunancy.

We first describe the defects which may occur during the

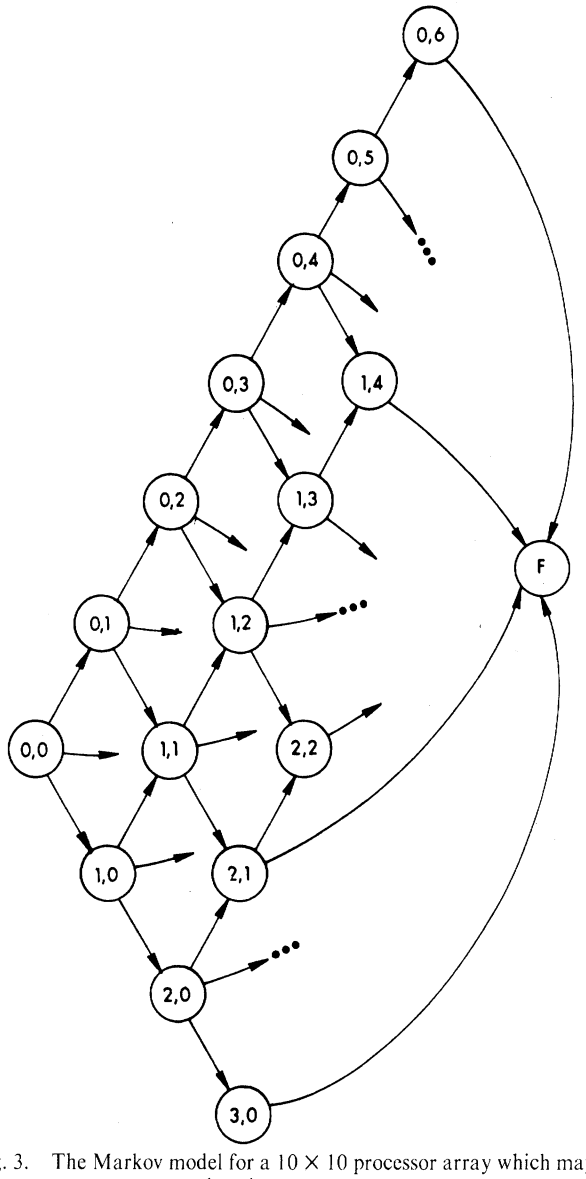


Fig. 3. The Markov model for a 10 × 10 processor array which may be reduced to a 7 × 7 array.

manufacturing process and their distribution. Fabrication defects can be classified as spot defects, line defects and area defects (e.g., [8]). Spot defects constitute the majority of the defects and each of them affects a single processor in a multiprocessor chip. Even line and area defects may in most cases result in a single faulty cell in a processor array. Hence, we assume in what follows that only a single processor is affected by any defect.

Fabrication defects obey mixed Poisson statistics using a gamma distribution as a mixing function [6]. This results in a Polya-Eggenberger distribution of the form

$$\Pr\{X = x\} = \frac{\Gamma(x+\alpha)}{x!\Gamma(\alpha)} \cdot \frac{\left(\frac{\bar{\lambda}}{\alpha}\right)^x}{\left(1+\frac{\bar{\lambda}}{\alpha}\right)^{\alpha+x}} \tag{18}$$

where x is the number of defects per chip; $\bar{\lambda}$ is the average number of defects per chip and α is a parameter that depends on the defect density variation [6]. This two-parameter distribution has a mean of $\bar{\lambda}$ and a variance of $\bar{\lambda}(1+\bar{\lambda}/\alpha)$. The mean and variance of data obtained from many wafer samples are used to estimate the two parameters $\bar{\lambda}$ and α.

The effect of a defect in a PE might be different from that of a defect in the communication bus connecting two adjacent PE's. We distinguish, therefore, between the two and denote by i and j, the number of PE defects and the number of communication bus defects per chip, respectively. We also denote by δ the percentage of communication bus defects out of all chip defects. The average values of i and j are, consequently $(1-\delta)\bar{\lambda}$ and $\delta \cdot \bar{\lambda}$, respectively. A first approximation for δ might be the percentage of silicon area devoted to communications buses out of the total chip area.

Thus, instead of using (18), we use the following expression,

$$\Pr\{I = i, J = j\} = \frac{\Gamma(i+j+\alpha)}{i!j!\Gamma(\alpha)} \frac{\left(\frac{\bar{\lambda}}{\alpha}\right)^{i+j}}{\left(1+\frac{\bar{\lambda}}{\alpha}\right)^{\alpha+i+j}} \delta^j(1-\delta)^i. \tag{19}$$

As long as the number of defects per chip is small, we may tolerate them and still use the chip if redundant processors have been added. Replacing faulty cells by redundant ones can not be done as in memory chips. However, we may use the reconfiguration strategy which was intended to be employed only later on to avoid the use of defective cells, and use instead some of the redundant ones.

The addition of redundant cells and the introduction of fault-tolerance into the array increase the chip area and as a result a smaller number of chips will now fit into the same wafer area. Hence, instead of comparing yield = $\Pr\{I = 0, J = 0\}$ which is the probability that a single chip is acceptable, we have to compare the expected number of acceptable chips from a given wafer.

Let B denote the number of simplex chips (without fault-tolerance) that fit into a wafer. The expected number of acceptable simplex chips is then

$$B \cdot \Pr\{I = 0, J = 0\} = \frac{B}{\left(1+\frac{\bar{\lambda}}{\alpha}\right)^{\alpha}}. \tag{20}$$

Let s denote the number of defective cells that we would like to tolerate and let γ_s denote the increase in chip area (the addition of redundancy) needed to tolerate these s faulty cells. Since the area of a fault-tolerant chip is $N\beta A_o$, the area of such a chip with the additional redundancy is $\gamma_s \cdot N\beta A_o$. The factor γ_s is called the redundancy factor and it depends among other factors on the reconfiguration strategy.

If s defective cells can be tolerated, the probability that a chip is acceptable is therefore

$$\sum_{i=0}^{s} \sum_{j=0}^{2(s-i)} \Pr\{I = i, J = j\}. \tag{21}$$

We may not, however, use in (21) expression (19) for $\Pr\{I = i, J = j\}$ as we have done in (20) since a chip with area $\gamma_s \beta N A_o$ has a larger average number of defects than a chip with area NA_o. We have therefore to substitute $\bar{\lambda}$ in (19) by $\gamma_s \cdot \beta\bar{\lambda}$.

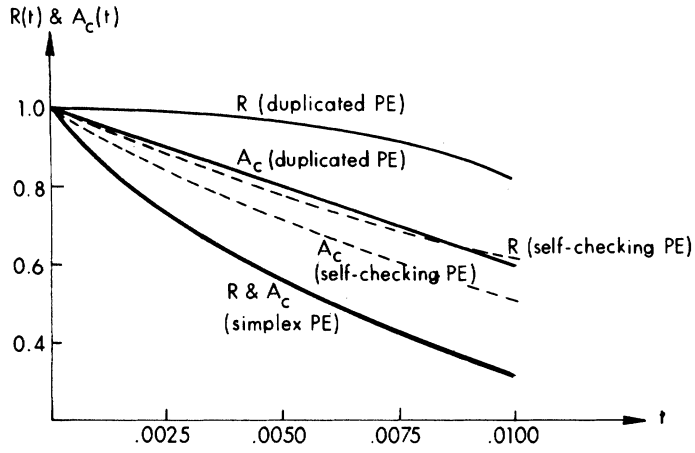


Fig. 4. Reliability and computational availability of three processor array designs.

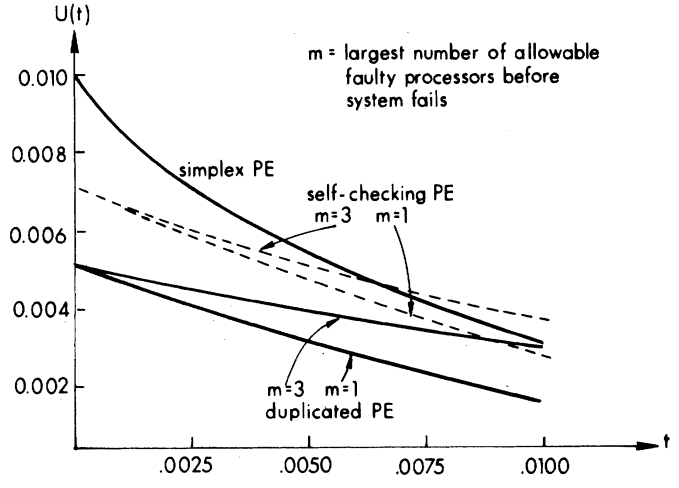


Fig. 5. Area utilization of the three designs.

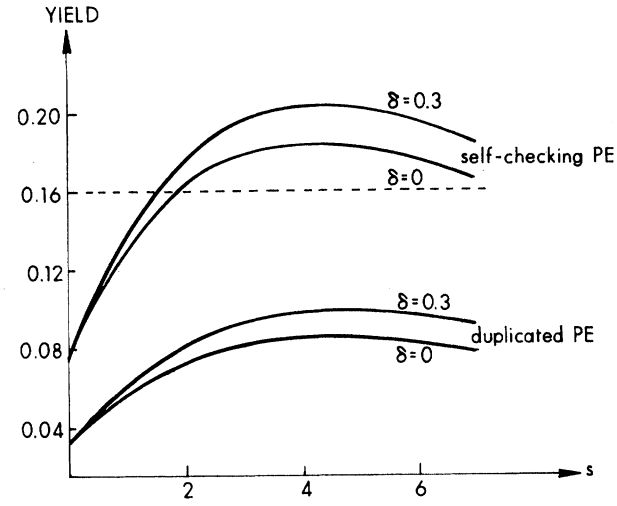


Fig. 6. The equivalent yield of fault-tolerant chips with added redundancy.

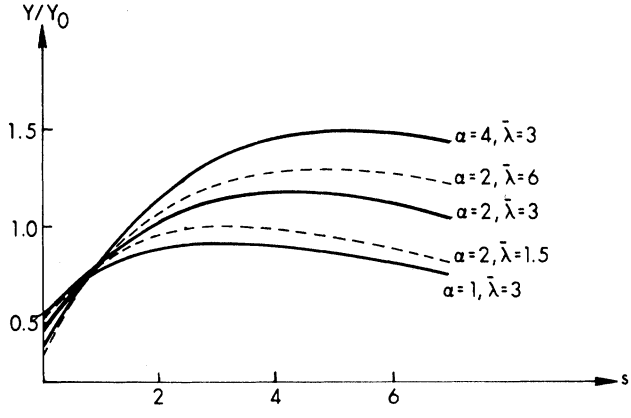


Fig. 7. Sensitivity of the maximum yield of a self-checking array to wafer parameters ($\delta = 0.1$).

The number of chips that fit now into the same wafer area is clearly $B/\gamma_s \cdot \beta$ and the resulting expected number of accepted chips is

$$\frac{B}{\gamma_s \cdot \beta} \sum_{i=0}^{s} \sum_{j=0}^{2(s-i)} \Pr\{I = i, J = j\}. \tag{22}$$

For simplicity we may divide (20) and (22) by B and compare the yield of a simplex chip

$$Y_0 = \frac{1}{\left(1 + \left(\frac{\bar{\lambda}}{\alpha}\right)\right)^{\alpha}} \tag{23}$$

with the equivalent yield of a fault-tolerant chip with added redundancy

$$Y_s = \frac{1}{\gamma_s \cdot \beta} \sum_{i=0}^{s} \sum_{j=0}^{2(s-i)} \frac{\Gamma(i+j+\alpha)}{i!j!\Gamma(\alpha)} \times \frac{\left(\frac{\bar{\lambda}\beta\gamma_s}{\alpha}\right)^{i+j}}{\left(1 + \left(\frac{\bar{\lambda}\beta\gamma_s}{\alpha}\right)\right)^{i+j+\alpha}} \delta^j (1-\delta)^i. \tag{24}$$

By comparing (23) and (24) we can determine whether it is beneficial when yield is considered to have built-in fault-tolerance and how many redundant cells should we add.

Example: Consider a chip consisting of a square grid of $N = 15 \times 15 = 225$ simplex PE's with the following defect distribution parameters: $\bar{\lambda} = 3$ and $\alpha = 2$ [6]. The resulting yield of the chip is $Y_0 = 0.16$. We compare the simplex chip with two possible designs of a fault-tolerant chip with added redundancy to enhance yield. The first is a chip with self-checking PE's, i.e., $\beta = 1.4$. The second is a chip with duplicated PE's, i.e., $\beta = 2$. For both designs we use the reconfiguration strategy that has been proposed in [1] and reviewed in Section II. The redundancy factor is, therefore,

$$\gamma_s = \left(1 + \frac{s}{\sqrt{N}}\right)^2.$$

The equivalent yield Y_s of the two chips is plotted in Fig. 6 as a function of the added redundancy with δ as a parameter. For a chip with duplicated PE's the maximum obtainable yield is 0.098 with $s = 5$ (i.e., five defective PE's can be tolerated) for $\delta = 0.3$, and is 0.086 with $s = 4$ for $\delta = 0$. For a chip with self-checking PE's the maximum yield is 0.205 and 0.185 for $\delta = 0.3$ and $\delta = 0$, respectively, both with $s = 4$. Hence, a chip with a 19×19 array of self-checking PE's has a higher yield than a chip with a 15×15 array of simplex PE's.

In the case of self-checking PE's, the effect of a change in the wafer parameters (namely, $\bar{\lambda}$ and α) on the best choice for s (needed to obtain the maximum yield) has been examined. The results are plotted in Fig. 7. The conclusion that

might be drawn from the curves in Fig. 7 is that we may choose $s = 4$ for a wide range of values of $\bar{\lambda}$ and α. Hence, an exact estimation of the wafer parameters is not needed.

IV. Conclusions

The consequences of introducing fault-tolerance into the architecture of a (single chip) VLSI processor array have been analyzed in this paper. A method for evaluating the effectiveness and area utilization of a given fault-tolerant technique has been presented. This method can be useful when comparing various fault-tolerance techniques to be incorporated into the VLSI chip.

In the second part of the paper, the impact of fault-tolerance on the wafer yield has been examined. It has been shown that adding redundancy may increase rather than decrease the wafer yield.

References

[1] I. Koren, "A reconfigurable and fault-tolerant VLSI multiprocessor array," in *Proc. 8th Annu. Syump. Comput. Arch.*, May 1981, pp. 426–441.

[2] M. D. Beaudry, "Performance-related reliability measures for computing systems," *IEEE Trans. Comput.*, vol. C-27, pp. 540–547, June 1978.

[3] F. A. Gay and M. L. Ketelsen, "Performance evaluation for gracefully degrading systems," in *Proc. 9th Annu. Symp. Fault-Tolerant Comput.*, June 1979, pp. 51–58.

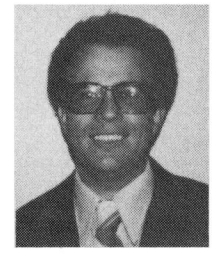
[4] W. C. Carter *et al.*, "Cost effectiveness of self-checking computer design," in *Proc. 7th Symp. Fault-Tolerant Comput.*, June 1977, pp. 117–123.

[5] R. M. Sedmak and H. L. Liebergot, "Fault tolerance of a general purpose computer implemented by very large scale integration," *IEEE Trans. Comput.*, vol. C-29, pp. 492–500, June 1980.

[6] C. H. Stapper, A. N. McLaren, and M. Dreckmann, "Yield model for productivity optimization of VLSI memory chips with redundancy and partially good product," *IBM J. Res. Develop.*, vol. 24, pp. 398–409, May 1980.

[7] 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.

[8] R. A. Cliff, "Acceptable testing of VLSI components which contain error correctors," *IEEE Trans. Comput.*, vol. C-29, pp. 125–134, Feb. 1980.



Israel Koren (S'72–M'76) was born on June 23, 1945. He received the B.Sc. (cum laude), M.Sc., and D.Sc. degrees from the Technion—Israel Institute of Technology, Haifa, in 1967, 1970, and 1975, respectively, all in electrical engineering.

From 1968 to 1971 he was with the Computer Center, Israel Ministry of Defense. In 1972 he joined the Department of Electrical Engineering at the Technion, where he became a Lecturer in 1975. From 1976 to 1978 he was an Assistant Professor in the Department of Electrical Engineering and Computer Science at the University of California, Santa Barbara. In 1978 he was an Assistant Professor in the Department of Electrical Engineering—Systems, at the University of Southern California, Los Angeles. Since 1979 he has been with the Department of Electrical Engineering, Technion—Israel Institute of Technology. Presently he is on sabbatical leave from the Technion, with the Computer Science Division, Department of Electrical Engineering and Computer Sciences, University of California, Berkeley. His current research interests are fault-tolerant architectures, VLSI arrays, and reliability of computer systems.

Melvin A. Breuer (S'58–M'65–SM'73) was born in Los Angeles, CA on February 1, 1938. He received the B.S. degree in engineering with honors from the University of California, Los Angeles, in 1959, the M.S. degree in engineering, also from the University of California, Los Angeles, in 1961, and the Ph.D. degree in electrical engineering from the University of California, Berkeley in 1965.

In 1965 he joined the staff of the Department of Electrical Engineering, University of Southern California, Los Angeles, where he is currently a Professor. His main interests are in the area of switching theory, computer aided design of computers, fault tolerant computing, and VLSI circuits.

Dr. Breuer is a member of Sigma Xi, Tau Beta Pi, and Eta Kappa Nu. He is the editor and co-author of *Design Automation of Digital Systems: Theory and Techniques*, (Prentice-Hall), editor of *Digital System Design Automation: Languages, Simulation and Data Base*, (Computer Science Press), and co-author of *Diagnosis and Reliable Design of Digital Systems*, (Computer Science Press). He has published over 70 technical papers and was formerly the editor-in-chief of the *Journal of Design Automation and Fault Tolerant Computing* and the co-editor of the *Journal of Digital Systems*. In addition to his research activities, he has been at the forefront of developing courses on design automation and fault tolerant computing at universities and at a number of research institutes.