D. C. Bossen
M. Y. Hsiao

Model for Transient and Permanent Error-Detection and Fault-Isolation Coverage

As computer technologies advance to achieve higher performance and density, intermittent failures become more dominant than solid failures, with the result that the effectiveness of any diagnostic procedure which relies on reproducing failures is greatly reduced. This problem is solved at the system level by a new strategy of dynamic error detection and fault isolation based on error checking and analysis of captured information. The model developed in this paper allows the system designer to project the dynamic error-detection and fault-isolation coverages of the system as a function of the failure rates of components and the types and placement of error checkers, which has resulted in significant improvements to both detection and isolation in the IBM 3081 Processor Unit. The model has also resulted in new probabilistic isolation strategies based on the likelihood of failures. Our experiences with this model on several IBM products, including the 3081, show good correlation between the model and practical experiments.

Introduction

Traditionally, failure (fault) isolation in a digital system has been dealt with in terms of testing, with testing procedures or at least specific test patterns generally produced only after the hardware to be tested is designed. Theoretical investigations of the fault-isolation problem have therefore concentrated on methods for producing "efficient" diagnostic test sets [1, 2], including the proposed measure of the diagnosability of systems [3]. An important exception to this testing approach is use of syndromes produced by error-checking circuits during normal machine operation for fault isolation. This general idea is obvious and has been previously discussed [4], although no follow-on work concerning the effectiveness or *coverage* of such an approach has been reported.

In the investigations of fault isolation via testing, the assumed starting point is a *data matrix* $\mathbf{D} = [d_{ij}]$, which has one row for every fault f_i $(1 \leq i \leq n)$ and one column for every test t_j $(1 \leq j \leq m)$. An entry d_{ij} in $\mathbf{D}$ is 1 if fault f_i is detected by test t_j. Based on the characteristics of this matrix, statements can be made about fault distinguishability, fault coverage, and efficiency.

Problems amenable to solution using such a model generally have the implicit assumption that there are more tests available than actually needed. This is rarely the case in practice. One solution has been the development of procedures either for eliminating redundant tests or for selecting tests which isolate faults to a replaceable package level rather than to the particular circuit [1]. Additional work in testing efficiency has resulted in a procedure for selecting the best testing order to achieve the lowest average-test-sequence length [2].

Faults in logic are generally assumed to be of the single stuck-at-0/stuck-at-1 (s-a-0/s-a-1) variety, although other models such as short-circuit faults in LSI and their associated test-generation procedures have been proposed [5]. From a practical standpoint, a fundamental shortcoming with these approaches is that they are based on testing procedures which rely on the presence of permanent faults, whereas in actual practice, most faults are intermittent or transient. It is a well-documented fact that these faults cause most of the serious isolation problems in the field [6–9].

Some work has been reported on the problem of detecting and isolating intermittent faults [10–15] that involves the use of test procedures which attempt to reproduce the assumed randomly occurring intermittent faults. Since the fault may not be present when the test is applied, the

approach taken is to repeat the tests many times. There are various suggestions for computing how much repetition is necessary to ensure a specified probability that a fault is or is not present. However, diagnostic time in a customer's office, even for a single pass-through set of tests, is costly; hence, approaches requiring large repetitions of tests are generally unacceptable.

The approach to fault isolation described in this paper is to capture and interpret the syndromes resulting from error checkers that are active during normal machine operation. The advantage of such an approach is that error checkers can detect all classes of errors which exist during each checking cycle. For example, these errors may be caused by faults of the following types: stuck-at, intermittent, single or multiple, transient, externally (environmentally) induced, pattern-sensitive, or any other fault which causes the computer's state to be outside the code space of the appropriate detection mechanism or error checker.

In order to analyze the fault detection and isolation coverage, we introduce a dynamic fault-probability model for projecting the ability of a system to detect and isolate failures, including intermittent failures, in an operational environment. In fact, this is a fundamental hardware design and diagnostic requirement for the 3081 Processor Unit maintenance strategy [16]. Relevant figures-of-merit are defined and computed using the model. The model is an integral part of an overall design evaluation procedure to enhance a product's reliability, availability, and serviceability.

By using the model, it has been possible to define probabilistic repair strategies and to project their effectiveness in terms of parts replaced per repair action and the probability of correct repair. Such probabilistic repair strategies are extensively used in the 3081 [16]. Further use of this model in the 3081 design has resulted in improved error-checker placement to enhance fault isolation, and the inclusion of improved error-detection mechanisms to cover otherwise weakly checked areas.

The remaining sections of this paper cover a system design strategy for intermittent and solid faults, a new concept of probabilistic fault isolation, the mathematical model for numerically characterizing the dynamic error-detection and fault-isolation probabilities of a system, detection and isolation coverages, the procedure and ground rules for obtaining data for the model, and our experiences with IBM products using the model.

System design and maintenance strategy

Dynamic error-detection mechanisms are usually implemented in various combinations of hardware, microcode, and software, and have had as their historic *raison d'etre* the detection of errors during normal operation so that recovery procedures may be initiated (*i.e.*, retry). A recommended system strategy is to take advantage of and, where necessary, to enhance these capabilities to achieve efficient and effective fault isolation.

To take advantage of these mechanisms for the isolation of intermittent faults, the system architecture must provide for the automatic capture of all necessary error information as it is detected, and before it is destroyed, altered, or overwritten by the recovery process. Thus, such an architecture would allow for what is called *error logging*. The concept of error logging was first introduced in the IBM System/360 [17, 18], although it has not been extensively used as an isolation tool, largely because it was not possible to project the error-detection or fault-isolation coverage. A second requirement is a procedure, preferably automated, for analyzing the error log to determine the physical location of the fault. Ideally, a set of error log analysis routines would replace the conventional set of diagnostic tests as the primary fault-isolation tool. A third requirement is a design-review procedure and fault-probability model to allow advance projections to be made of the dynamic error-detection and fault-isolation probabilities. This is crucial because a strategy based on dynamic error detection generally requires changes to hardware functions (*e.g.*, more error checkers or better placement of them) that would be practically impossible to make after an LSI design was finalized.

The importance of a timely evaluation of any aspect of the system reliability or fault-tolerant coverage cannot be overstressed. Reference [19] graphically illustrates the current lack of rigorous state-of-the-art methods for the projection of fault-tolerant coverage. Error detection and fault isolation are important aspects of fault-tolerant coverage.

With this recommended system strategy, the fault-isolation characteristics of a system are not treated as an "add-on" feature provided by diagnostic tests; rather, they are a direct result of the hardware functions of error detection and are, thus, the primary responsibility of the hardware designer. Reference [4(b)] gives design information for error-detection circuits and methods.

Dynamic fault-probability model for detection and isolation

• *Coverage figures-of-merit for fault-tolerant systems*

The proposed strategy requires several figures-of-merit for the coverage in order to properly characterize the fault-tolerant performance of the system and to provide useful feedback for making design improvements. The first is the *error-detection coverage* or *percentage* (referred to as *ED*).

Another, the *fault-isolation coverage* (*FI*), is more complex to describe. First, it is possible to characterize a system by its fault-isolation distribution $(I_1, I_2, \cdots, I_n)$, where I_1 is the portion of all faults which implicate a single field-replaceable unit (FRU). In general, I_i is the portion of all faults which implicate exactly i FRUs. This is an unambiguous way of specifying the fault-isolation coverage which is independent of isolation strategy. (Other figures-of-merit for fault-isolation coverage are presented in a later section, along with a discussion of isolation strategies.)

• *Notation and definitions*

The system consists of N FRUs ($\text{FRU}_1, \text{FRU}_2, \cdots, \text{FRU}_N$). Each FRU_i contains n_i faults, $f_{i1}^*, f_{i2}^*, \cdots, f_{in_i}^*$. The total system contains $N^* = \Sigma n_i$ faults. Associated with each FRU_i is a fault probability p_i, which is the sum of the fault probabilities for faults contained on FRU_i. That is,

$$p_i = \sum_{j=1}^{n_i} p_{ij}^*,$$

where p_{ij}^* is the probability of fault f_{ij}^*. For practical systems, the size of the fault set N^* is unmanageably large. For a FRU containing m lines, N^* will be $3^m - 1$; therefore, in order to analyze any practical systems of even medium size, it is mandatory that procedures for grouping faults be developed.

Based on the use of error checkers and captured machine-state information, a system contains M nonzero, mutually exclusive syndromes, $S_1, S_2, \cdots, S_M$, together with the null syndrome S_{null}. A *syndrome* is the logical product (Boolean "AND") of identifiers of an error-checker and its active source at the time the error is captured; thus, it is the result of a dynamically detected error. (Examples of syndromes are given in the following section.)

The dynamic error-detection characteristics of the system are represented by a fault-conditional-probability data matrix and a vector of fault probabilities. Figure 1 shows the full fault-conditional-probability data matrix **D*** of dimension $N^* \times M$ and the full fault-probability vector **P*** of length N^*. **D*** has a row for each fault and a column for each syndrome. The matrix entry d_{ijk}^* is the conditional probability that S_k occurs, given fault f_{ij}^*. It is convenient to deal with this set of information in a compressed form, *i.e.*, in the form of a compressed fault-conditional-probability data matrix **D**, which has dimension $N \times M$ and has one row for each FRU_i and one column for each syndrome S_j. The compressed full fault-probability vector **P** has one entry p_i for each FRU_i, equal to the sum of all fault probabilities on FRU_i (see Fig. 2). The entry d_{ij} of **D** is simply obtained as the sum of products:

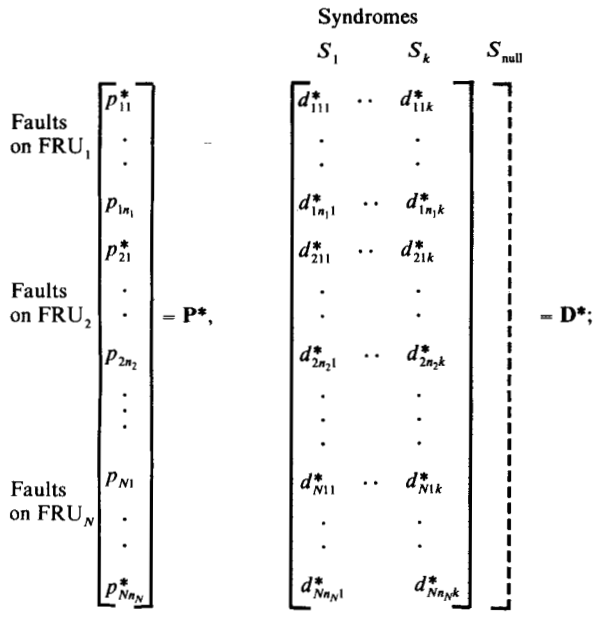


Figure 1 Full fault-probability vector **P*** and fault-conditional-probability data matrix **D***.

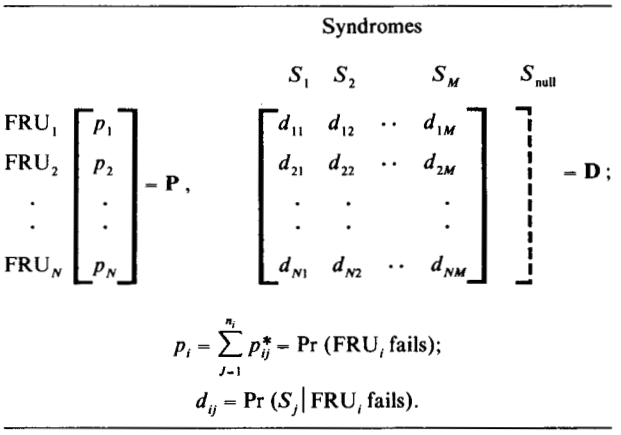


Figure 2 Compressed fault-probability vector **P** and compressed fault-conditional-probability data matrix **D**.

$$d_{ij} = \frac{\sum_{k=1}^{n_i} p_{ik}^* d_{ikj}^*}{\sum_{k=1}^{n_i} p_{ik}^*} = \Pr(S_j \mid \text{FRU}_i \text{ fails}),$$

where $\Pr(S_j \mid \text{FRU}_i \text{ fails})$ is the probability of S_j due to FRU_i failure. In practice, the full matrix **D*** is not explicitly

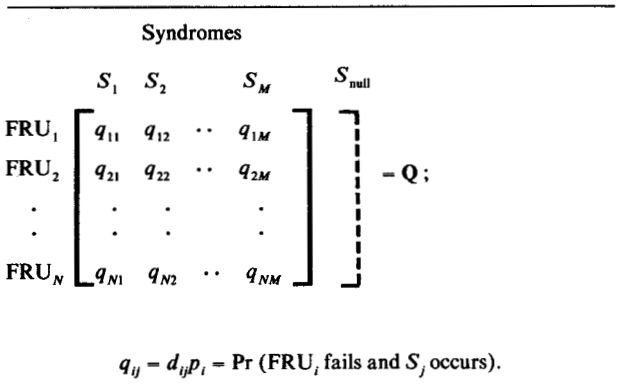


Figure 3 Joint compressed-probability matrix **Q**.

obtained. What is obtained is a matrix whose size is between **D*** and **D**, which represents an initial grouping of faults according to functions.

The computational algorithms all have as their basis the compressed matrix **D**, the compressed fault-probability vector $\mathbf{P} = (p_1, p_2, \cdots, p_N)$, and the joint compressed-probability matrix **Q** (see Fig. 3). Matrix **Q** is useful since it contains the maximum information; *i.e.*, the sum of each row is equal to the detected fault probability for FRU_i, while the sum of each column is equal to the syndrome-occurrence probability for syndrome S_j. Each "normalized" (for the total number of detected faults) column equals the distribution of likelihoods for FRU failure candidates, given a particular syndrome occurrence. The total sum of all the entries in **Q** is the error-detection probability (*ED*); thus $\Pr(S_{null}) = 1 - ED$. The error-detection probability for FRU_i equals the sum of the row divided by p_i. Since the fault isolation (*FI*) is normally defined with respect to detected errors only, for purposes of computing isolation-coverage figures-of-merit, the matrix **Q** is normalized by dividing it by $[1 - \Pr(S_{null})]$.

● *Fault-isolation strategies*

Before computing figures-of-merit for dynamic fault isolation of a system, it is useful to introduce the concept of *isolation strategy*. A strategy is a definition, for each syndrome S_i, of the specific repair action to be taken given the occurrence of S_i. The action or strategy is based on the particular FRU implicated by S_i. In a system perfectly designed for fault isolation, every syndrome implicates a single FRU, and a discussion of strategy is unnecessary. However, this is not generally the case and when an ambiguous syndrome occurs a choice must be made between replacing either all of the implicated FRUs or some subset of them. For the latter case, there is a risk involved since the FRU containing the actual fault may not be replaced; thus a subsequent occurrence of the fault may arise, necessitating another repair call. Based on the syndrome-probability matrix, all important factors in such a trade-off can be quantified.

Examples of various strategies include the following. For first-call repairs we use the *deterministic isolation procedure* (DIP) strategy; all implicated FRUs are replaced. For second-call repairs, the action can be either (a) to replace the single-most-likely FRU on the first call and replace the remainder on a subsequent call; or (b) on the first call, to replace a subset of FRUs that is sufficient to account for a threshold (*e.g.*, 90%) portion. This is a limited-risk strategy. For *N*th-call repairs, we use the *sequential isolation procedure* (SIP) strategy; the action taken depends on the number of calls. On the first call, the most-likely FRU is replaced; on the second call, the next-most-likely FRU is replaced, etc., until on the *N*th call, the least-likely FRU is replaced. The choice among these strategies reflects trade-offs among the costs for parts, labor, and customer-outage associated with multiple calls to fix a problem.

● *Figures-of-merit for fault-isolation coverage*

It was previously noted that the fault-isolation distribution characterizes fault-isolation coverage independently of the fault-isolation strategy. However, in order to reflect the very real practical differences between possible isolation strategies, additional figures-of-merit for the coverage are required. Two useful measures are the average number of FRUs replaced per detected fault (*NFRU*) and the average number of service calls per detected fault (*NSC*). Having calculated these values, a direct estimate of the cost of each strategy over the life of the system or product can be computed. *NFRU* and *NSC* can be computed directly from the normalized **Q**, **P**, and the previously stated strategy definitions.

For each syndrome S_i, d_i is defined as the number of FRUs implicated by syndrome S_i. Based on the syndrome probability matrix **Q**, d_i is the number of nonzero entries in column i of **Q**. The probability of isolating to one FRU, given a system failure, is computed by summing all entries in the columns of **Q** which have $d_i = 1$. Using DIP strategy, this probability is given by

$$\sum_{i=1}^{N} q_{ij}; j \ni d_j = 1.$$

The remainder of the fault-isolation distribution can be computed using the general relation of the probability of isolating to K FRUs,

$$\sum_{i=1}^{N} q_{ij}; j \ni d_j = K.$$

The sequential isolation bound reflects the maintenance strategy of selecting for replacement the most likely FRU

implicated by an ambiguous syndrome. An ambiguous syndrome is represented in the matrix **Q** by a column containing more than one nonzero entry. Choosing the most likely FRU for such a syndrome corresponds to choosing the FRU associated with maximum entry in each column. Let $\text{Max}_j\,(q_{ij})$ be the largest entry in column j of **Q.** Then for the sequential isolation, the probability of isolating to one FRU is given by

$$\sum_{j=1}^{M} \text{Max}_j\,(q_{ij}).$$

Note that this will include all the entries summed for the deterministic isolation bound.

- *NFRU using DIP and SIP strategies*

With d_j as defined above, *NFRU* can be computed using DIP strategy with the relation

$$\sum_{j=1}^{M} d_j \sum_{i=1}^{N} q_{ij},$$

which is simply the summation of the number of FRUs implicated per syndrome times the syndrome probability. In order to compute *NFRU* using SIP strategy, define $\{q_{ij}\}_j$ to be the arrangement of column j in descending numerical order. That is,

$$q_{1j} \geq q_{2j} \geq \cdots \geq q_{Nj}.$$

Then the average number of FRUs per syndrome j, d_j (avg) is given by

$$\frac{\sum_{i=1}^{N} iq_{ij}}{\sum_{i=1}^{N} q_{ij}} = d_j(\text{avg}),$$

and the average number of FRUs per failure is therefore

$$NFRU(\text{SIP}) = \sum_{j=1}^{M} d_j\,(\text{avg}) \sum_{i=1}^{N} q_{ij} = \sum_{j=1}^{M} \sum_{i=1}^{N} iq_{ij}.$$

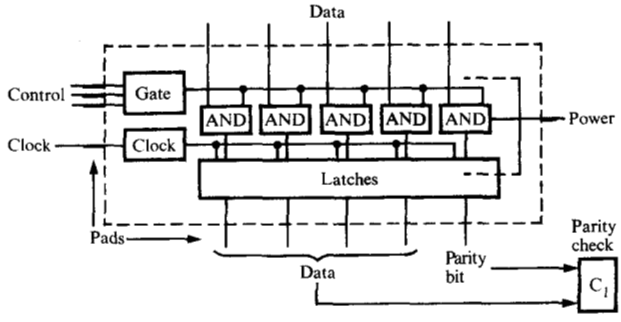
- *Average number of service calls per fault (NSC)*

For the DIP strategy, the number of service calls per failure is equal to one, since this strategy specifies that all implicated FRUs will be replaced on the first call. In the case of SIP strategy, the average number of service calls is equal to the average number of FRUs replaced:

$$NSC(\text{SIP}) = NFRU(\text{SIP}).$$

The EDFI process—obtaining data for the fault-conditional data matrices

It has been found practical to obtain data for the fault-probability models where circuits have been grouped into functions. This corresponds to a matrix **D***, which is somewhat larger than the compressed matrix **D.** It has further been found practical to have the designer provide data for each FRU in terms of a data set called a *basic information*



Component name	*Probability of failure* (rate)	*Functions of the component*	*Probability of function failure*	*Syndrome name*	*Syndrome likelihood*
A-chip logic	0.9	Gate logic	1/9	C_1	0.5
		Clock logic	1/9	C_1	0.5
		Register-bit logic	7/9	C_1	1.0
A-chip pads	0.05	Data pads	1/2	C_1	1.0
		Clock/gate pads	1/2	C_1	0.5
A-chip power	0.05	On-chip distribution	1/1	C_1	1.0

Figure 4 Parity-checked register example with detailed basic information table entries.

table, which would correspond to a horizontal slice or set of rows of the matrix **D***. Data provided in this fashion is reduced by programs to the single row **D** and the corresponding FRU fault-probability entry in vector **P.**

Each row of the basic information table is a vector of data consisting of component name, component failure rate, function(s) of the component, failure probabilities of the functions, syndromes of each function, and syndrome probabilities given function failure. See Fig. 4. It is quite possible for each row in the basic information table to correspond to a single circuit fault, in which case **D*** would be represented. Guidelines provided to designers suggest that entries should correspond to functions such as registers, decoders, parity checkers, arrays, ALUs, etc., each of which may have hundreds of circuits, and represent hundreds of faults in **D***. The association of a failing function with the error checker that detects its failure (according to the system design) and also with the syndrome (checker combined with machine state) is the designer's job.

Where more than one nonzero syndrome is possible, for example due to varying path usage or instruction mix, the relative likelihood is also estimated by the designer. Many circuit groups performing a function, such as a register or decoder, are such that their faults are not uniform in detection probability. For example, a single failure in

Components	*Probability of failure* (rate)	*Detection probability for schemes:*			
		I	*II*	*III*	*IV*
Register cells	0.7	1.0	1.0	1.0	1.0
Clock	0.1	0.5	0.5	1.0	0.75
Gates	0.1	0.5	0.5	1.0	0.75
Power	0.05	1.0	0.5	1.0	0.75
Pads	0.05	0.75	0.75	1.0	0.88
Total	1.00	0.89	0.86	1.0	0.93

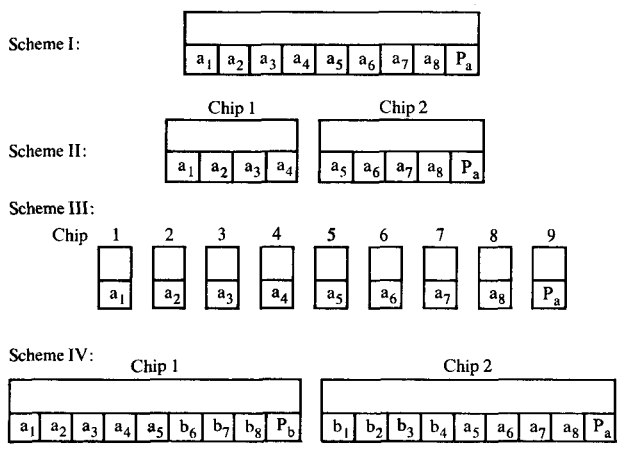


Figure 5 Four packaging schemes of register and array chips with the chip-detection probabilities. $a_1, a_2, \cdots, a_8, P_a$ is an 8-bit byte with its parity bit; $b_1, b_2, \cdots, P_b$ corresponds to another 8-bit byte.

circuits in gating and clock distribution of an LSI register causes multiple bit errors and hence causes the null syndrome with probability 0.5. On the other hand, the latches will not cause the null syndrome.

This gives rise to the concept of *checker effectiveness,* which represents the weighted average across the failures in a representative function of the detection probabilities. Although it is called *checker* effectiveness, it actually is associated with the function being checked. Its utility is in avoiding repeated detailed circuit input to the basic information table when a common design function is repetitively used.

By using the concept of checker effectiveness in this manner, a designer can provide as input to the evaluation an additional table called the *checker information table* consisting of a row for each identified checker in the system, the row containing the checker name, and the effectiveness of the checker.

Checker	*Effectiveness*	*Location*	*Description*
C_1	0.9	Data bus	Byte parity checker
.	.	.	.
.	.	.	.

This allows the designer to represent the entire function as a single row in the basic information table.

Component	*Failure probability* (rate)	*Function*	*Failure probability*	*Syndrome of function*	*Syndrome likelihood*
A chip	1.0	Gated register	1.0	C_1	1.0
.	.	.	.	.	.
.	.	.	.	.	.

The references to checker C_1 in the *Syndrome of function* column will automatically cause a cross reference to the checker information table to obtain the detection probability or effectiveness of 0.9, a value previously determined.

A limitation on the use of "checker effectiveness" arises in a design when a particular checker checks information from more than one source and the two sources have different fault distributions. For example, a parity checker checks a storage array as well as a data register, and the fault distributions are different. In this case, the basic information table entries must list the detailed fault distributions of each function.

Figure 4 shows an LSI parity-checked register chip along with its basic information table. The relative failure rates within the chip of the three categories, logic, mechanical interconnections (pads), and power distribution within the chip, are provided by the technology developers using a projection methodology analogous to the one described in [20]. Circuit and pad counts are provided by the logic designer. Performing the arithmetic to compute an overall detection probability for this chip gives 0.8875. If this same chip part number is used elsewhere in the system, it is worthwhile to use the checker effectiveness idea (checker information table) to avoid repeated detailed data entry.

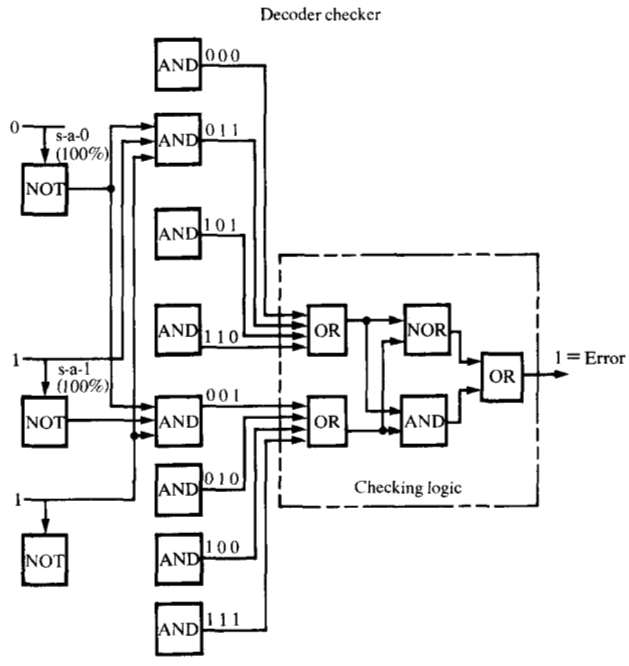
- *Enhancing parity-check detection effectiveness*

Detection probability of 0.5 is assigned to those circuit faults which can cause multiple bits to be in error in a parity-checked field, for example, gate and clock circuits. If the gate or clock circuit is actually controlling data, *e.g.*, from two bytes instead of from one, the detection probability for this class of circuit is $[1 - (0.5)^2] = 0.75$. This fact is useful in the implementation of multi-byte data flows, in order to increase the parity-check-detection probability by packaging

Table 1 Detection probabilities.

Detection mechanism	*Detection (%)*	*Comments, package, etc.*
Parity check		
Arrays		
MS-C*, 256 × 9	95	1 field/chip
	96	2 fields/chip
	100	bit slice
MS-D, 32 × 18	90	1 field/chip
	95	2 fields/chip
	99	4 fields/chip
	100	bit slice
MS-E, 256 × 54	95	1 field/chip
	97	2 fields/chip
Registers		
MS-A	90	all bits in 1 chip
	93	2 fields/chip
	100	1 bit/chip/field
MS-B	95	1 field/chip
Parity predict		
adder	81	
counter	90	
Decoder check	75	
Cyclic redundancy check	99.9	
ECC (error detection only)	$1-2^{-r}$	r check bits
Illegal pattern checking	$(M/N)100$	M = number of illegal patterns N = total patterns
Time out	100	Delayed
Sum check	99.9	16 bits
Duplication	100	
Sensor detection	100	Will be lowered if low threshold sensitivity
Power check	100	Sensing
	M	M = over-detection % for logic supplied by this power source
Background diagnostics	$N \times D$	N = run time/total time D = computed diagnostic effectiveness

*The abbreviations MS-C, D, E, A, B represent masterslice C, D, E, A, B, etc.



Number of inputs	*Number of gates*				*Checker circuit overhead*	*Detection probability*
	Decoding logic		*Checking logic*			
	NOT	*AND*	*OR*	*AND*		
2	2	4	4	1	0.83	0.68
3	3	8	4	1	0.45	0.72
4	4	16	8	1	0.45	0.71
5	5	32	12	1	0.35	0.71

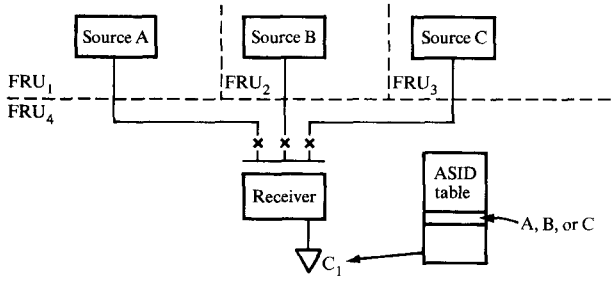
Figure 6 Decoder check with error-detection probabilities and check-circuit overhead for various size decoders.

portions of different bytes on the same LSI chip and under a common gate and clock control. In general, for parts of n bytes under common control, the detection probability is $[1 - (0.5)^n]$.

Depending on the ratio of such common logic faults to those faults which cause only single bits in error, such packaging variations can have a pronounced effect on the overall detection probability. Figure 5 shows four packaging schemes for parity-checked LSI registers. A common failure of a single byte is now divided into multiple parity-check cases; therefore, the average detection probabilities differ among these four schemes. Some of these variations for the 3081 processor technologies have been evaluated and tabulated as design guidelines in Table 1. The detection variations among the array technologies reflect different functional distributions, and hence, different failure distributions internal to the chip. Failure mode distributions for array technologies have been published [20–22] which identify categories such as bit-line, word-line, cell, and chip kill failures.

• *Other detection mechanisms—detection effectiveness*

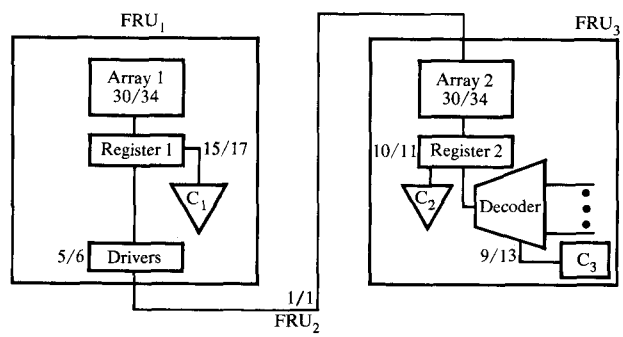
Parity checking has been emphasized in the preceding examples because, in most digital systems, parity checking accounts for 70 to 80% of the detection coverage. Control logic which consists of counters and registers may also be parity checked. There is a portion of control logic, however, for which parity is not appropriate in general. For example, decoders have the natural characteristic that a one-and-only-one check may be performed. Since this type of check amounts to a practical duplication of the logic, it is generally



	Model input function	*Number of circuits*	*Syndrome name*	*Syndrome likelihood*
FRU_1	Source A circuits	82	C_1 (and) A	1.0
FRU_2	Source B circuits	67	C_1 (and) B	1.0
FRU_3	Source C circuits	80	C_1 (and) C	1.0
FRU_4	Receiver and checker circuits	65	C_1 (and) A	0.33*
			C_1 (and) B	0.33*
			C_1 (and) C	0.34*

*Determined by frequency of path usage.

Figure 7 Active source identifier (ASID) as part of the syndrome.



	Function	*Probability of failure*	*Syndrome name*	*Syndrome likelihood* (rate)
FRU_1	Array 1	34/116	C_1	30/34 (0.88)
	Register 1 + checker C_1	17/116	C_1	15/17 (0.88)
	Drivers	6/116	C_2	5/6 (0.83)
FRU_2	Cable	1/116	C_2	1/1 (1.00)
FRU_3	Array 2	34/116	C_2	30/34 (0.88)
	Register 2 + checker C_2	11/116	C_2	10/11 (0.91)
	Decoder + checker C_3	13/116	C_3	9/13 (0.69)

Figure 8 Example system with basic information table entries for the example.

avoided in favor of some form of a reduced decoder check; see Fig. 6 and [3]. The effectiveness of such a checker may be evaluated by considering, as the set of all faults, each gate stuck-at-zero (s-a-0) and stuck-at-one (s-a-1). When the check circuit itself is included in the function, the overall "effectiveness" is shown in Fig. 6. The complexity of such an analysis shows that the detection effectiveness concept applied to commonly repeated portions of a design is very useful in practice in reducing the amount of the input required.

- *Syndrome definition for improved isolation*

Syndrome entries in the basic information table may in general consist of Boolean combinations of checkers and machine status. Figure 7 shows data flow where a downstream checker C_1 checks three sources on three different FRUs. The implicated FRU set for the condition "C_1 active" would be all three sources as well as the receiver, or four FRUs. The addition of active source identifier logic to capture the name of the active path will improve isolation. The input for this situation is also shown in Fig. 7, where it is seen that isolation is now to two FRUs.

One important point which this example brings out is the fact that isolation is generally improved by including machine-state information with the error checker in order to define the syndrome. In other words, a single physical error checker can give rise to a number of unique syndromes, each with its own set of implicated failures, when machine state is included together with checker output. This is a very useful observation when the design changes are being considered in order to improve isolation coverage. This fact also illustrates the importance of error logging for producing good isolation. Exactly what gets logged, in addition to error check outputs, will have a tremendous impact on the isolation coverage in general. Using the ED/FI model, it is possible to rapidly assess the impact on fault-isolation coverage of proposed design changes, as well as enhanced machine state error logging.

- *Illustrative example*

The system under consideration is shown in Fig. 8, along with its basic information table. It consists of two logic cards (FRUs) and an interconnecting cable, also a FRU. The denominator of the fraction shown with each item is the failure rate expressed in some consistent units. The numerator is obtained as the product of the failure rate times the error-detection probability for the item, determined according to the ground rules previously stated.

The checker information table is not used since functions checked by checker C_2 have different detection probabilities. The compressed fault-probability vector **D**, the compressed fault-conditional-probability matrix **P**, and the compressed joint-probability matrix **Q** are shown in Fig. 9. Dividing **Q** by $[1 - \Pr(S_{null})]$ gives the "normalized" joint-probability

matrix **Q** of Fig. 9, where the set of d_i giving FRUs per syndrome is also indicated. The error-detection probability equals $[1 - \Pr(S_{null})] = 1 - 16/116 = 0.86$. Using the normalized **Q** the DIP and SIP isolation probabilities are computed as

Pr (isolate to 1 FRU, DIP) = Pr (C_1 or D_1) = 0.45 + 0.09 = 0.54;

Pr (isolate to 1 FRU, SIP) = the summation of the maximum entry in each column of the normalized **Q** = 0.45 + 0.40 + 0.09 = 0.94.

Other figures-of-merit are the average number of FRUs using DIP strategy, *NFRU*(DIP) = 1(0.45) + 3(0.46) + 1(0.09) = 1.92; the average number of FRUs using SIP strategy, *NFRU*(SIP) = 1(0.45) + [1(0.4) + 2(0.05) + 3(0.01)] + 1(0.09) = 1.07; the average number of service calls using DIP strategy, *NSC*(DIP) = 1; and the average number of service calls using SIP strategy, *NSC*(SIP) = *NFRU*(SIP) = 1.07.

ED/FI experience with IBM products

The ED/FI model and evaluation procedures described in this paper were developed to meet a practical and real need, and they have been used throughout IBM since 1975 for assessing error-detection and fault-isolation coverages of numerous product designs. The definitions and ground rules have been extended beyond the logic and electronic areas into the electromechanical products such as printers and disk drives. The initial experience in developing the model came about in the early 3081 processor development, where the system designers, diagnostic developers, and field engineers agreed that the change in technology required a diagnostic strategy to handle intermittent errors. Therefore, there was a strong need to assess the error-detection and fault-isolation coverage. There were at that time, however, no definitions, ground rules, or procedures for projecting such coverage within IBM or reported in the literature [19].

Experience has shown the value of using the ED/FI evaluation procedure repetitively as the design progresses. Early evaluation of 3081 system error-detection coverage yielded about 60% for the CPU. The current level of better than 90% was achieved by dedicated concentration to put checking on the initially weak areas as shown by the model.

Because designers became educated and sensitive to the need for good error-detection coverage, a number of innovations in error checking were made. These were primarily in the control areas of the machine, and included decoder checks, illegal pattern checking, encoder checks, and the application of parity to address and control fields as standard practice. Packaging arrangements to enhance parity-error detection, as pointed out in Table 1, were also extensively used. Fault-isolation coverage was enhanced by better placement of checkers and by the identification of machine-state information to produce better syndromes. Some sample output reports are shown in Figs. 10 and 11 for a representative large system. Reports are typically produced on a per-FRU basis for designer feedback.

$$\mathbf{P} = \begin{bmatrix} P_1 \\ P_2 \\ P_3 \end{bmatrix} = \begin{matrix} \text{FRU}_1 \\ \text{FRU}_2 \\ \text{FRU}_3 \end{matrix} \begin{bmatrix} 57/116 \\ 1/116 \\ 58/116 \end{bmatrix}$$

$$\mathbf{D} = \begin{matrix} \\ \text{FRU}_1 \\ \text{FRU}_2 \\ \text{FRU}_3 \end{matrix} \begin{matrix} \begin{matrix} C_1 & C_2 & C_3 \end{matrix} \\ \begin{bmatrix} 45/57 & 5/57 & 0 \\ 0 & 1/1 & 0 \\ 0 & 40/58 & 9/58 \end{bmatrix} \end{matrix} \begin{matrix} S_{null} \\ \begin{bmatrix} 7/57 \\ 0 \\ 9/58 \end{bmatrix} \end{matrix}$$

$$\mathbf{Q} = \begin{matrix} \\ \text{FRU}_1 \\ \text{FRU}_2 \\ \text{FRU}_3 \end{matrix} \begin{matrix} \begin{matrix} C_1 & C_2 & C_3 \end{matrix} \\ \begin{bmatrix} 45/116 & 5/116 & 0 \\ 0 & 1/116 & 0 \\ 0 & 40/116 & 9/116 \end{bmatrix} \end{matrix} \begin{matrix} S_{null} \\ \begin{bmatrix} 7/116 \\ 0 \\ 9/116 \end{bmatrix} \end{matrix}$$

$$\mathbf{Q}\text{ normalized} = \begin{matrix} \\ \text{FRU}_1 \\ \text{FRU}_2 \\ \text{FRU}_3 \end{matrix} \begin{matrix} \begin{matrix} C_1 & C_2 & C_3 \end{matrix} \\ \begin{bmatrix} 45 & 5 & 0 \\ 0 & 1 & 0 \\ 0 & 40 & 9 \end{bmatrix} \end{matrix}$$

number of FRUs/syndrome: 1 3 1

Figure 9 Vector **P** and matrices **D**, **Q**, and **Q** normalized to 100 for the example.

Independent verification of the projections, by hardware bugging using a limited sample size on the 3081, shows good correlation to the coverage values projected by the model. This model and the ED/FI process have been extended and used in IBM to project isolation coverage for error-recreation diagnostic programs. The circuit and fault coverages of individual tests are determined by test generation and simulation programs. Each test is treated as an additional checker added to the hardware checker lists and the same ED/FI analysis procedure follows to compute an ED/FI percentage for the solid-failure case using diagnostics. This allows a complete evaluation of the service plan for

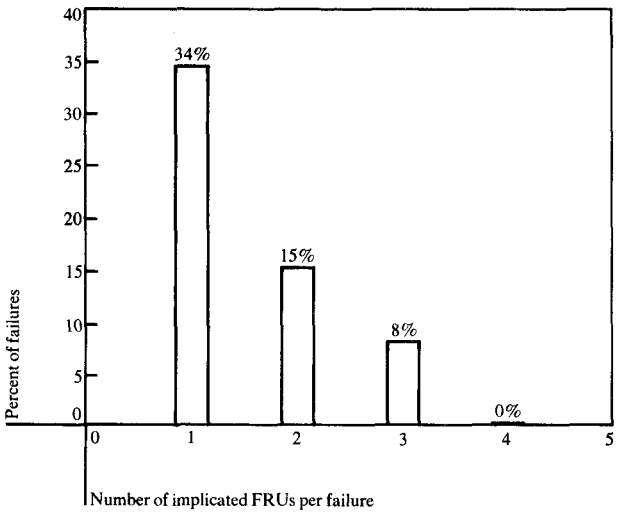


Figure 10 Representative sample output showing isolation distribution.

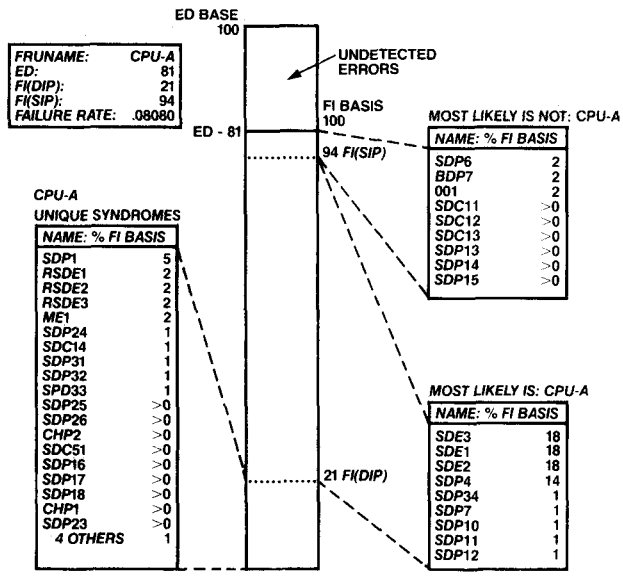


Figure 11 Sample FRU output report showing syndrome contribution to isolation. The various abbreviations listed are simply identified syndrome names for the particular system tested.

a system in both the operational environment (error checkers and log analysis only) and the maintenance environment (where error re-creation may be used).

Summary

This paper describes a model for projecting error-detection and fault-isolation coverages. Implicit in interpreting the results of the model is a maintenance strategy of fault isolation based on dynamically detected errors. The model has been applied in many practical design cases, and evaluation results have suggested weak areas in both error-detection and fault-isolation coverage, with improvements having been made accordingly. Designers are given specific ground rules for generating inputs to the model, and are provided error-detection coverage numbers and computational guidelines for most error checkers in use.

The following lists some of the ED/FI evaluation benefits:

1. Product designers are aware of the need for error detection and FRU isolation early in the design phases. The evaluation procedure gives early feedback regarding areas needing improvement. (It is noteworthy that inventions are often made in response to such needs.) Improvements in error detection made in this iterative fashion, especially in LSI technologies, have had minimum impact on product cost, performance, and schedule.
2. By placing primary emphasis on error checkers which can detect errors from all causes, including intermittent errors, problems associated with a maintenance strategy of reproducing errors with diagnostic programs or procedures are eliminated. This benefit results in greatly reduced mean-time-to-repair, as well as reduced parts costs.
3. Data integrity, which depends first and foremost on the detection of errors, can be designed directly into the hardware. Using the model for error-detection coverage, quality can be projected in advance.

Acknowledgments

The authors wish to recognize the early participation of L. C. Chang in the development of the model, A. P. Czarneicki and R. A. Houdtzagers for their role in coordinating the early ED/FI evaluations of the 3081 system, and C. L. Chen for his work in programming the model.

References

1. H. Y. Chang, "An Algorithm for Selecting an Optimum Set of Diagnostic Tests," *IEEE Trans. Electron. Computers* **EC-14,** 706–711 (1965).
2. H. Y. Chang, "A Distinguishability Criterion for Selecting Efficient Diagnostic Tests," *AFIPS Conf. Proc., Spring Joint Computer Conf.* **32,** 529–534 (1968).
3. A. D. Friedman, "A New Measure of Digital System Diagnosis," *Proceedings of the Fault-Tolerant Computing Symposium No. 5,* Paris, 1975, IEEE, New York, pp. 167–169.
4. See for example F. F. Sellers, Jr., M. Y. Hsiao, and L. W. Bearnson, *Error Detecting Logic for Digital Computers,* McGraw-Hill Book Co., Inc., New York, 1968 and W. C. Carter, "Theory and Use of Checking Circuits," *Infotech State of the Art Report 20,* Infotech Information, New York, 1974, pp. 413–455.
5. A. D. Friedman, "Diagnosis of Short-Circuit Faults in Combinational Circuits," *IEEE Trans. Computers* **C-23,** 746–752 (1974).
6. M. Ball and F. Hardy, "Effects and Detection of Intermittent Failures in Digital Systems," *AFIPS Conf. Proc., Fall Joint Computer Conf.* **35,** 329–335 (1969).

7. O. Tasar and V. Tasar, "A Study of Intermittent Faults in Digital Computers," *Proceedings of the 1977 National Computer Conference,* AFIPS Press, Montvale, NM, pp. 807–811.
8. S. McConnel and D. P. Siewiorek, "C.vmp: The Implementation, Performance, and Reliability of a Fault Tolerant Microprocessor," *Report CMU-CS-78-108*, Carnegie-Mellon University, Pittsburgh, PA, March 1978.
9. Y. K. Malaiya and S. Y. H. Su, "A Survey of Methods for Intermittent Fault Analysis," *Proceedings of the 1979 National Computer Conference,* AFIPS Press, Montvale, NJ, pp. 577–585.
10. M. A. Breuer, "Testing for Intermittent Faults in Digital Circuits," *IEEE Trans. Computers* **C-22,** 241–246 (1973).
11. S. Kamal and C. V. Page, "Intermittent Faults: A Model and Detection Procedure," *IEEE Trans. Computers* **C-23,** 713–719 (1974).
12. S. Kamal, "An Approach to the Diagnosis of Intermittent Faults," *IEEE Trans. Computers* **C-24,** 461–467 (1975).
13. I. Koren and Z. Kohavi, "Diagnosis of Intermittent Faults in Combinational Networks," *IEEE Trans. Computers* **C-26,** 1154–1158 (1977).
14. S. Y. H. Su, I. Koren, and Y. K. Malaiya, "A Continuous-Parameter Markov Model and Detection Procedure for Intermittent Faults," *IEEE Trans. Computers* **C-27,** 567–570 (1978).
15. S. Mallela and G. M. Masson, "Diagnosable Systems for Intermittent Faults," *IEEE Trans. Computers* **C-27,** 560–566 (1978).
16. Nandakumar N. Tendolkar and Robert L. Swann, "Automated Diagnostic Methodology for the IBM 3081 Processor Complex," *IBM J. Res. Develop.* **26,** 78–88 (1982, this issue).
17. W. C. Carter, H. C. Montgomery, R. J. Preiss, and J. H. Reinheimer, "Design of Serviceability Features for the IBM System/360," *IBM J. Res. Develop.* **8,** 115–126 (1964).
18. M. Y. Hsiao, W. C. Carter, J. W. Thomas, and W. R. Stringfellow, "Reliability, Availability, and Serviceability of IBM Computer Systems: A Quarter Century of Progress," *IBM J. Res. Develop.* **25,** 453–465 (1981).
19. A. L. Hopkins, Jr., "Fault Tolerant System Design: Broad Brush and Fine Print," *Computer* **13,** No. 3, 39–45 (1980).
20. 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.* **24,** 398–409 (1980).
21. H. C. Rickers, "Microcircuit Device Reliability/LSI Data," *RADC/RBRAC, MDR-3, Winter 1975–76*, Reliability Analysis Center, U.S. Air Force Rome Air Development Center, Rome, NY.
22. W. C. Carter and C. E. McCarthy, "Implementation of an Experimental Fault-Tolerant Memory System," *IEEE Trans. Computers* **C-25,** 557–568 (1976).

Received June 26, 1980; revised May 7, 1981

The authors are located at the IBM Data Systems Division laboratory, Poughkeepsie, New York 12602.