Nandakumar N. Tendolkar
Robert L. Swann

Automated Diagnostic Methodology for the IBM 3081 Processor Complex

The concepts of automated diagnostics that were developed for and that are implemented in the IBM 3081 Processor Complex are presented in this paper. Significant features of the 3081 diagnostics methodology are the capability to isolate intermittent as well as solid hardware failures, and the automatic isolation of a failure to the failing field-replaceable unit (FRU) in a high percentage of the cases. These features, which permit a considerable reduction in the time to repair a failure as compared to previous systems, are achieved by designing a machine which has a very high level of error-detection capability as well as special functions to facilitate fault isolation using Level-Sensitive Scan Design (LSSD), and which includes a Processor Controller to implement diagnostic microprograms. Intermittent failures are isolated by analyzing data captured at the detection of the error, and the analysis is concurrent with customer operations if the error is recoverable. A further improvement in the degree of isolation is achieved for solid failures by using automatically generated validation tests which detect and isolate stuck faults in the logic. The diagnostic package was designed to meet a specified value of isolation effectiveness, stated as the average number of FRUs replaced per failure. The technique used to estimate the isolation effectiveness of the diagnostic package and to evaluate proposals for improving isolation is described. Testing of the diagnostic package by hardware bugging indicates very good correlation between projected and measured effectiveness.

Introduction

A major portion of the logic of the IBM 3081 Processor Complex is implemented using *level-sensitive scan design* methodology [1–3]. Up to 118 LSI chips are mounted on a *thermal conduction module* (TCM) [4] which is the *field-replaceable unit* (FRU). A 3081 Processor Unit with 16 channels contains 26 TCMs and one with 24 channels contains 27 TCMs. The subject of this paper is the design of diagnostic microprograms that are capable of isolating failures if they occur in these TCMs, on the boards on which the TCMs are packaged, or in the interboard cables. A major design goal was to isolate automatically all hardware-detected errors caused by hardware failures. This goal has the following advantages: 1) the duration of repair action is reduced because the set of FRUs required to repair the 3081 is identified by the diagnostic microprograms, which operate at microprocessor speed, and is communicated to the *customer engineer* (CE), who can bring these FRUs with him; 2) the level of training required by a CE to repair the 3081 is reduced, since the CE need not know the internal logic; and 3) the same reliable service is available to all customers, since the CE's skill in diagnosing a failure is not a factor. Another design goal for the diagnostic package is that it be able to isolate, to a single TCM or to a set of TCMs, both intermittent and solid (permanent) failures for a specified percentage of failures. The word *isolation* is used here to imply that the diagnostics will identify (call) a set of TCMs such that the failing TCM is in the set. If the set contains no other TCM than the failing one, the isolation is unambiguous and the 3081 can be repaired by replacing the identified TCM. If the set contains more than one TCM, the 3081 can be repaired by replacing at most all TCMs in the set.

A figure of merit to compare different diagnostic strategies is the *isolation effectiveness*, defined here as the average number of FRUs called by diagnostics for a failure. This is an important figure of merit because the cost of the FRUs replaced in the field is directly proportional to the isolation effectiveness. The actual isolation effectiveness is a function of several factors including the ratio of intermittent to solid

faults (we describe this later in the section on diagnostic effectiveness). Although the goal is to identify a failure to the failing FRU unambiguously, the designers of the diagnostics for the 3081 had to design a system that has an isolation effectiveness of 1.2 TCMs per failure or less.

There is a major difference between the technique used for diagnostics in the 3081 and that used in existing systems [5–7], which mainly rely on being able to recreate a failure by running diagnostic tests on the failed machine. In the 3081, a new approach, which uses the data captured at the detection of the error, is used to isolate a failure. Advantages of this method over the former for isolating intermittent failures are discussed in the paper by Bossen and Hsiao [8] in this issue.

This paper describes the techniques used to achieve the previously discussed goals and the evolution of the automated diagnostic methodology. This is followed by a review of the key features of the 3081 Processor Complex hardware that facilitate the implementation of automated diagnostics. A brief description of the Processor Controller's role in supporting diagnostics is given; the fault-isolation process and the four algorithms that form the heart of the methodology are described; and techniques for determining the average number of TCMs-called-per-failure are presented. Highlights of the key features of the automated diagnostic methodology are discussed in the concluding section. The fault-isolation process applies to diagnostics of all failures of the TCMs, the boards on which TCMs are mounted, and the interboard cables, which constitute a major portion of the 3081 hardware.

Overview of automated diagnostic methodology

A primary requirement for designing a diagnostic methodology that automatically isolates failures to a set of FRUs is that some hardware mechanism should detect a very high percentage of failures. Since failures cause errors and since well-known techniques are available to detect errors [9], the IBM 3081 Processor Unit is designed with extensive error-detection mechanisms. It has a capability of instantaneously detecting an estimated 90% of the errors caused by hardware failures; this estimate is based upon the definition and methodology described in the paper by Bossen and Hsiao [8].

The next major requirement is to design a scheme for isolating the detected failures to meet the average-FRUs-per-failure goals for diagnostics. Once an error is detected, certain information that describes the state of the machine at the time of the occurrence of the error is collected. This information determines the *isolation effectiveness* (average number of TCMs called by the diagnostics per failure). A failure is considered to be isolated to a set of FRUs if the detected error could have been caused by failure of some hardware on one of the FRUs in the set and not by the failure of any other FRU not in the set. In order to fix the failure with certainty in a single repair action, all FRUs in the set must be replaced. To meet the stated goals for isolation effectiveness, the hardware should be designed so that the information required to isolate the failure must be available to the diagnostic microprogram. Finally, a given physical failure may cause a set of errors, possibly with distinct symptoms, from which the system can recover. The total information collected for the entire set of errors may result in further isolation of the failure. Thus, a set of algorithms is required to process the information collected for a set of errors to achieve the best possible isolation.

These algorithms isolate the failure to a failing FRU in many cases, but there is still a class of failures for which the isolation is greater than one FRU. For those cases, two additional techniques are used for further isolation. Using the LSSD facilities, the suspected TCMs can be tested for the existence of *stuck-at* faults (*i.e.*, logic stuck at either 1 or 0 levels). If any TCM has a stuck-at fault, it is identified by these tests. A certain percentage of failures do not fall into the stuck-fault category. For such failures, these algorithms do not identify the single FRU that has failed; we just know the set of FRUs, one of which has failed. A procedure called *sequential FRU replacement* (SFR) has been developed and implemented to reduce the average number of TCMs called by diagnostics to fix such failures. The SFR policy identifies a subset of the set of FRUs such that the failing FRU is in the subset with a very high average probability. These FRUs are replaced in the first repair action when the failure occurs. In a very small percentage of the cases where SFR is used, a second repair action may be required. The implementation of this scheme is described subsequently.

Throughout the development of the diagnostics, it was necessary to evaluate whether they met the stated isolation objectives. The method used for evaluation, described subsequently, was also used to project what percentage improvement would occur in the isolation effectiveness if a proposed hardware or diagnostic algorithm change were to be implemented.

3081 Processor Complex hardware

Hardware features which are most significant to the diagnostics are that 1) the hardware has a very high level of error-detection capability, 2) the concept of the *active source identifier* (described subsequently) is developed and implemented to improve isolation, 3) information required to isolate a failure is stored in the LSSD *shift-register latches* (designated herein as SRLs) and arrays, and 4) the *monitoring and system support adapter* (MSSA) and the *logic support station* (LSS) provide a means for the Processor

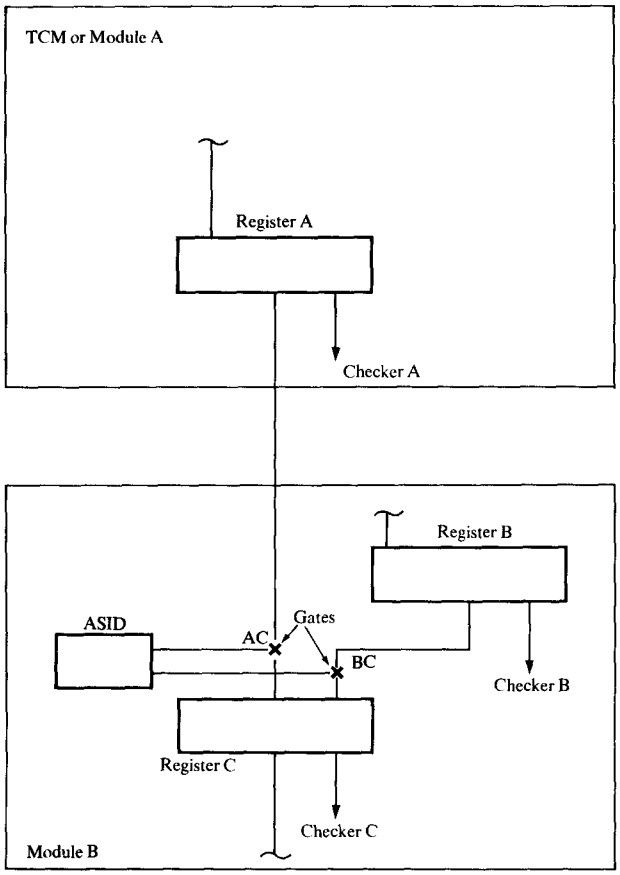


Figure 1 An example of the Active Source Identifier (ASID) concept.

Controller to know about the detection of an error and the ability to scan out the SRLs and arrays that contain the information required for isolation. (Note: characteristics of the Processor Controller are discussed in the paper by Reilly *et al.* [10] in this issue.) As a result of such features, diagnostic microprograms which run on the Processor Controller have the necessary information to isolate a failing TCM.

Error-detection mechanisms are extensively used in the 3081 Processor Complex. Checking techniques are used for hardware facilities such as registers, array chips, adders, decoders, and control mechanisms. When a checker detects an error, certain events are triggered which ultimately lead to the invocation of the diagnostic microprograms. We return to this later.

The concept of the *active source identifier* is illustrated in Fig. 1. Suppose Checker C detects an error whereas Checkers A and B do not indicate any errors. If we did not know whether Register B was ingated or Register A was ingated at the time of the error, then the failure could be either on Module A or on Module B. Two FRUs would have to be called for this detected error. Let us suppose that an SRL (labeled ASID) is used to indicate whether Gate AC or BC was open when the data were transferred into Register C. Then, when a failure is detected while transferring data from Register B to Register C, the ASID would indicate Gate BC, and the failure would be isolated to Module B. Thus the ASID helps in reducing the number of FRUs called by diagnostics where one or another path could have been used at the time of the failure. In general, the *active source identifiers* are SRLs or arrays which identify the data source or the control function which was active at the time of the error. They are used in places where the error-detection mechanism can be set by failure on more than one source module.

Arrays have been provided to store the information required to isolate a failure. Since it takes several cycles before the clocks of a given portion of the hardware (referred to as a *functional element*) that has detected an error can be stopped, the arrays contain several cycles' worth of values of the active source identifiers and control words. The error checkers are SRLs that, once set to *on*, cannot be reset by any mechanism other than the *logic support station* (LSS). The LSS provides a mechanism to stop the functional element that has detected an error. It is connected to the processor controller via the MSSA. The LSS presents an interrupt to the processor controller to indicate that an error is detected.

Role of the Processor Controller

The complete functions of the Processor Controller are discussed in the article by Reilly *et al.* [10]. The relationship of the Processor Controller to other units of the 3081 Processor Complex is shown in Fig. 2. The diagnostic microprograms and the error handler and recovery microprograms run on the Processor Controller. It also supports power and thermal error detection and failure isolation; thus, logic errors due to failure of power or cooling equipment can be separated from those due to failure of TCMs.

When the 3081 hardware detects an error, the LSS presents an interrupt to the Processor Controller and also stops the clocks of the functional element involved. The error handler and recovery microprogram collects the information that is required for both fault isolation and recovery by scanning out the SRLs and arrays. The error handler tries to recover the operation and, if successful, the 3081 can continue data processing operations. The error handler records the scanned-out data on the *Processor Controller File* (PCF). This recording is called a *logout*. The error handler then posts the diagnostics with a message that an error has been detected and passes a pointer to the location of the logout. The diagnostic microprograms use this logout to determine the failing TCM.

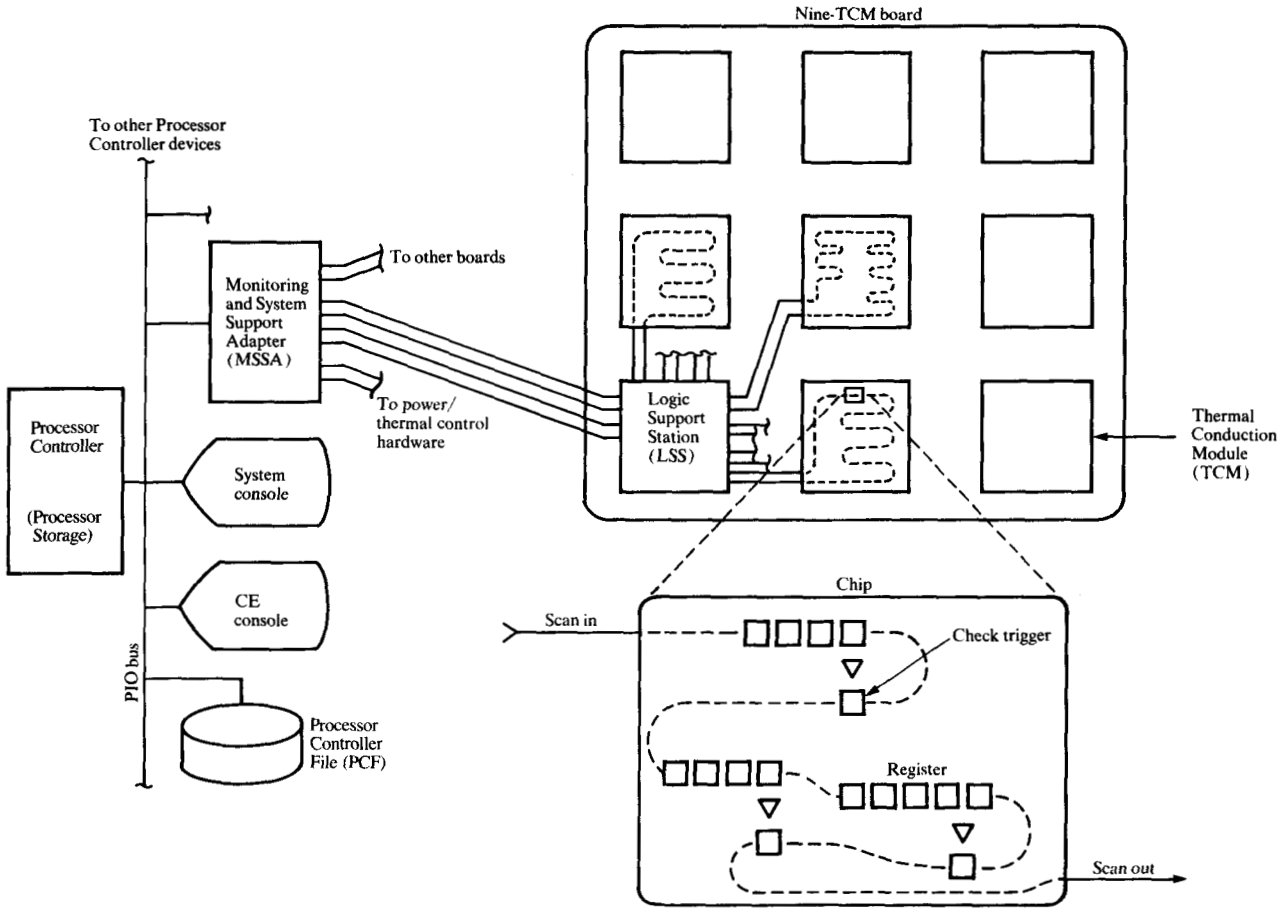


Figure 2 Block diagram illustrating relationship of the Processor Controller to other units of the 3081 Processor Complex.

The Processor Controller also provides facilities for the diagnostic microprograms to communicate to the *customer engineer* (CE). A display console is provided for the CE to obtain information about any detected failure and the failing TCMs identified by diagnostics for each failure. If the frequency of errors is higher than a threshold value or if a processor is in check-stop state, the diagnostic microprogram also displays a code on the system console indicating which TCMs the CE needs to bring with him to fix the problem.

Fault-isolation process

The function of the fault-isolation process is to determine a minimal set of FRUs that must be replaced to fix a failure or a fault that has caused an error detected by the hardware. There are four elements that make up the fault-isolation process: *direct isolation, intersection isolation, validation tests* (VTs), and *sequential FRU replacement* (SFR). The fault-isolation process is illustrated in Fig. 3. Whenever an error is detected by the 3081 hardware, the error handler makes a logout and invokes diagnostic microprograms. The logouts are processed by *fault-isolation analysis routines* (FIARs). This phase corresponds to direct isolation. If the result of direct isolation is a multiple FRU call, intersection isolation is invoked to further isolate the failure. If the frequency of the error is over a threshold (currently set at 24 errors in any continuous two-hour period) or if a processor is put in check-stop state because it is impossible or undesirable to continue operation, a message is sent to the system console indicating that a CE be called, and a coded number is displayed which is used by the CE to bring the necessary set of TCMs.

The diagnostic microprograms maintain a file that contains, for each detected fault, a block of information called a *fault-isolation record control block* (FIRCB). The CE can look at an FIRCB and determine the FRUs called by diagnostics and any further processing that may be required. If direct isolation and intersection do not result in

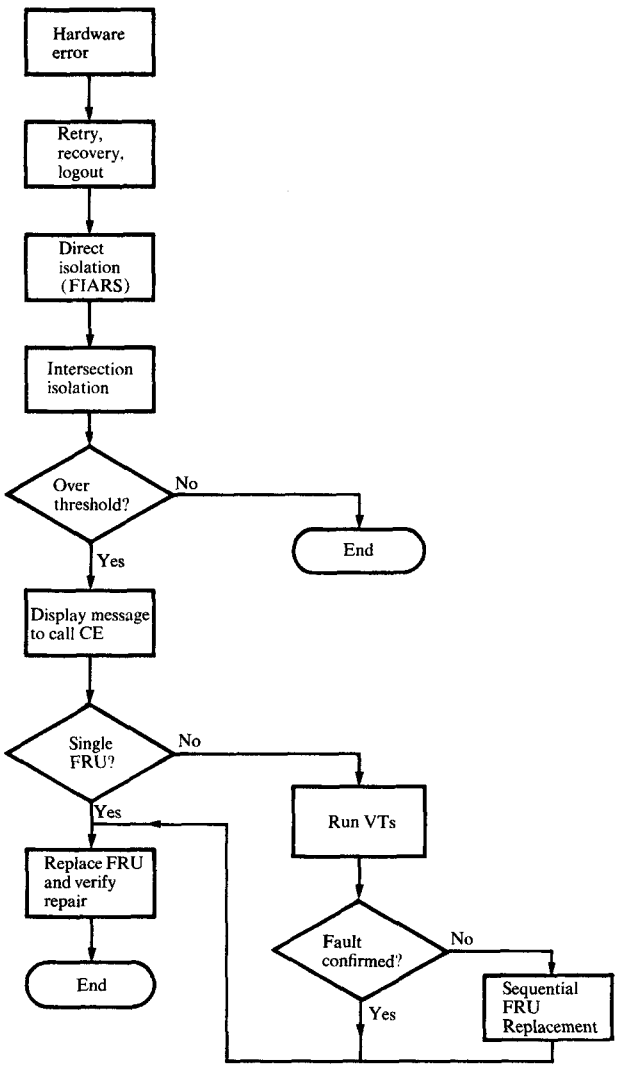


Figure 3 Fault-isolation process.

isolation to a single TCM, the diagnostic microprogram indicates (in the FIRCB) to the CE that the system operator should release the indicated hardware to run VTs.

The CE comes to repair the machine either because the diagnostic microprograms detected an over-threshold condition or the customer determined that the rate of errors is unacceptable and the machine needs repair. If the fault was isolated by direct or intersection isolation to a single TCM, the CE proceeds to replace the indicated TCM. If it was not isolated to a single TCM, the CE makes the necessary hardware available to diagnostics (with the customer's permission) for running VTs. If the VTs detect a fault and identify the failing TCM or TCMs to be replaced, the CE replaces them. If the fault was not detected by the VTs, the SFR policy is invoked and the set of TCMs to be replaced in the first call are conveyed to the CE, who replaces them. The last step in the process is verification of the replacement. In this step, we ensure that there are no stuck faults remaining in the system after replacement by running VTs to detect such faults.

After replacement, the diagnostic file contains information on which TCMs were replaced and when. If the fault recurs in the two-week period following a repair, one of the following applies: sequential FRU replacement policy was used and the second call is required; the failure may be on a board or a cable if the domain of the error so indicates; or all indicated FRUs, including boards and cables, have been replaced and the error still persists. In the first case, the CE proceeds to replace TCMs indicated in the second SFR call. In the second and third cases, the CE calls a support center for help because further analysis is required, in either case, before any repair action can be taken.

- *Direct isolation*

Direct isolation is defined as the process of isolating a failure based only on the logout information collected by the error handler for a single detected error. The isolation of failures from the logout would be a complicated, if not infeasible, process if an unknown number of physically distinct failures could simultaneously occur. If the failures occur as a random process, and if the failure rate is such that the probability of a second failure occurring before the first failure is repaired is very small (1% or less), a single fault will be present 99% or more of the time when an error is detected. For the technology used in the 3081 Processor Unit, projection of failure rates based on testing in the laboratory environment indicates that this condition is satisfied and the single-fault assumption is justified.

An important function of the direct-isolation process is also to create information used by the other elements of the fault-isolation process. We now introduce the concept of a *direct-isolation domain* (DID), which is a collection of hardware entities, and the concept of error syndromes. We show how a relationship is established between an error syndrome and a direct-isolation domain. The process of direct isolation consists of examining the logout to determine which error syndrome has occurred and, using the relationship between error syndromes and DIDs, determining the DID that contains the failure.

Error syndromes A *syndrome* is a specific combination of values of the bits in the logout. The logout consists of a certain number of bits of information on the state of the machine at the time of the detection of the error. Included is information on the status (*on* or *off*) of the error checkers, active source identifiers, and other information stored in the arrays for fault-isolation purposes (*e.g.*, what function was being performed, etc.).

Direct isolation domain (DID) of a syndrome The DID of a syndrome is a set of hardware entities (SRLs, registers, encoders, decoders, or some collection of circuits) such that when any of these fail, they could cause the error syndrome to occur, and further, there is no other hardware entity outside the set whose failure could cause the same syndrome. Clearly, the definition of the DID of a syndrome implies that the error syndrome could be caused by the failure of only those FRUs on which the DID of the syndrome lies. Thus, the problem of isolating the failure requires finding the DID of the syndrome.

Consider a network of interconnected circuits that is a subset of the 3081 Processor Unit hardware. Furthermore, this network has a fixed number of inputs and outputs. Let the set of inputs be such that for a given syndrome there is no error in the inputs, and the set of outputs be such that it corresponds to the error syndrome detected. In this network, the output is a function of the inputs and the logic of the network. Since inputs are error-free and an output has errors in it (if there is a single fault in the system), the fault is in the set of circuits in this network. If we can find a network that satisfies these conditions, but no proper subset of it satisfies them, the set of circuits in the network is the DID of the syndrome. DIDs are determined by back-tracking algorithms and are verified by simulation. Using the previously described process, a DID is determined for each valid error syndrome by the diagnostician. An example of a DID is shown in Fig. 4.

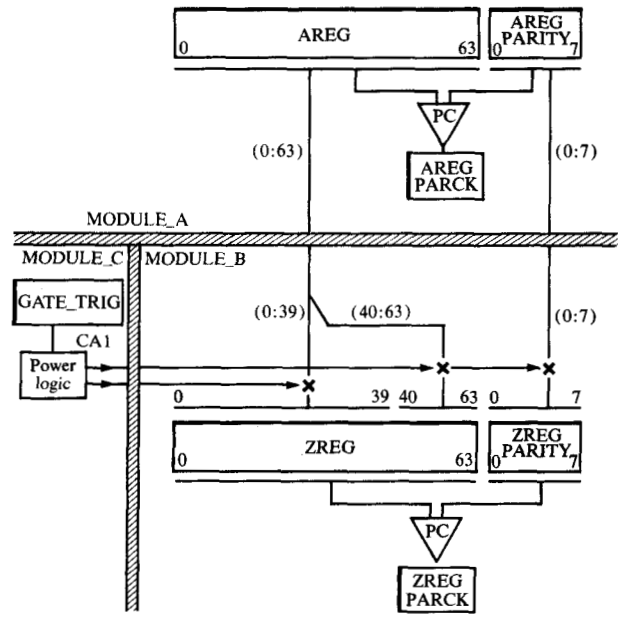
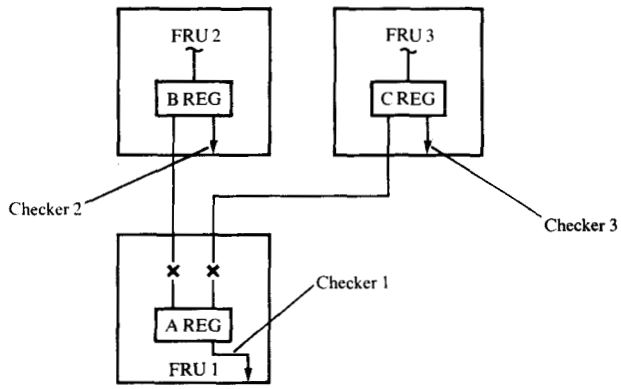


Figure 4 Example of a Direct-Isolation Domain (DID). The syndrome is ZREG PARCK *on* and AREG PARCK *off*.

• *Intersection isolation*

Intersection isolation is the technique of isolating a failure to a single FRU when the failure results in a variety of error syndromes over a period of time. An examination of the hardware of the 3081 Processor Unit indicated that there are hardware entities which, when they fail, can cause many different syndromes to occur depending upon the function being performed by the machine. Direct isolation would produce a DID for each such syndrome that occurs. Advantage can be taken of the collective information provided by several syndromes to further isolate the failure beyond what was accomplished by direct isolation.

Principle of intersection isolation When a group of errors occurs in a small time interval such that it is reasonable to assume that a single fault exists, the fault must be in the hardware area common to (*i.e.*, the logical intersection of) all error domains of the group. If this hardware area is on a single FRU, the fault is isolated to a single FRU for all errors of the group. This principle is illustrated by an example shown in Fig. 5. Error 1 occurs when data flow is from B REG to A REG. The domain of Error 1 contains the A REG on FRU 1 and the B REG on FRU 2. Sometime later, Error 2 occurs when the data flow is from the C REG to the A REG. The domain of Error 2 contains the A REG on FRU 1 and the C REG on FRU 3. Applying the principle of intersection isolation, it is concluded that the failure is on FRU 1, A REG, which is the area common to both domains.



Error	*Detected by checker*	*Domain* (FRU, area)	*Data flow*
1	1	(1, A REG) and (2, B REG)	B REG to A REG
2	1	(1, A REG) and (3, C REG)	C REG to A REG

Figure 5 Intersection isolation example.

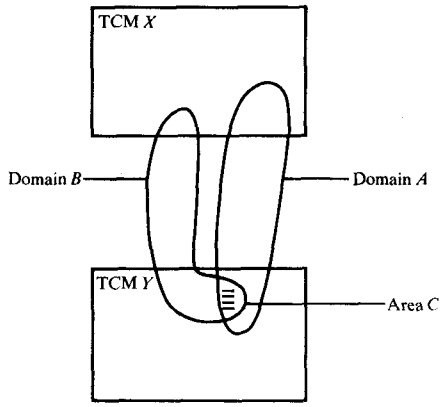


Figure 6 Intersection using TCMs.

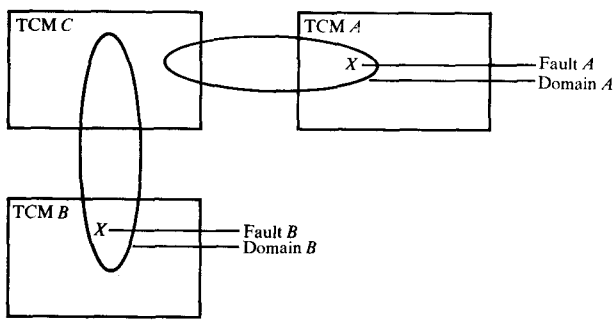


Figure 7 Intersection when two faults exist.

Hence, FRU 1 is the failing FRU. In the example shown in Fig. 5, the number of FRUs required to correct the problem is reduced from three to one.

To apply intersection isolation, we first have to identify the hardware facilities common to a set of (direct isolation) domains of a set of error syndromes. These hardware facilities could be latches, triggers, registers, etc. Since the hardware facilities in each domain are specified for the machine, one can determine the hardware facilities common to any set of domains. In the following discussion, the term *area* is used to denote a collection of hardware facilities such as latches, triggers, registers, etc.

Intersection isolation requirements The domains have to be defined in such a way that the area common to various domains can be identified; this requires the following:

1. Each FRU is divided into areas, and each area should meet the following criteria: a. No two areas have any hardware in common; b. The area lies entirely on a FRU; c. The area is either completely inside or outside a domain.
2. There is a domain D_i associated with each error syndrome i. This D_i is identified by direct isolation. For a given D_i a set S_i is defined as follows:

   $S_i = \{(M, A) \mid (M, A) \in D_i\}$,

   where $(M, A) \in D_i$ means that Area A on FRU M is a part of the domain of error i.

Intersection-isolation technique Given a set of error syndromes and associated domains, we define a set S^* as follows:

$S^* = \{(M, A) \mid (M, A) \in S_i$ for each error $i\}$.

The following three cases can arise:

1. If there is a FRU M^* such that for all $(M, A) \in S^*$, $M = M^*$, then the fault is isolated to FRU M^*.
2. If there are at least two distinct FRUs M_1 and M_2 such that $(M_1, A_1) \in S^*$ and $(M_2, A_2) \in S^*$, then the fault is not isolated to a single FRU.
3. If S^* is empty, then at least two failures (faults) exist in the system. This case can be converted into Cases 1 or 2 by partitioning errors into groups. A group is a set of errors such that there is at least one (M, A) that belongs to the domain of every error in the set. (S^* is not empty for all errors that belong to a group.)

Implementation of intersection isolation This technique would require a process to divide DIDs into areas. Because of the large number of DIDs involved in the 3081 and the lack of an automated process, an alternative way of using just the TCMs for intersection was investigated. Thus, if S_1, S_2, $\cdots$, S_n are the sets of TCMs corresponding to the domains of errors E_1, E_2, $\cdots$, E_n detected in a time period where single fault assumption applies, then S^*, where $S^* = S_1 \cap S_2 \cap \cdots \cap S_n$, is the set of TCMs to which the failure is isolated.

If there exists a single fault in the machine, the failing FRU is still in the intersection of the FRUs called for each DID, but there is a possibility that intersection using TCMs would not result in as good an isolation as obtained by using areas. An example of this is shown in Fig. 6. For both Domain A and Domain B, TCMs X and Y are called. The intersection of these two domains on a TCM basis still gives TCMs X and Y. If the failure was in Area C, it would not be isolated to TCM Y if just TCMs are used for intersection. The second concern is that there is a small probability that two distinct faults could occur which would result in a wrong (not failing) TCM called by intersection. This is shown in Fig. 7. There are two faults, one on TCM B and another on TCM A. The TCM C is the common TCM for their domains. The intersection would result in calling TCM C.

An evaluation of the impact of these two concerns on the isolation effectiveness was performed using random selection

of failure locations. The study showed that, based on the projected failure rates of TCMs, if we allowed only syndromes occurring within a two-week period of each other to intersect, the average number of TCMs per failure using TCMs for intersection would be 3% higher than that obtained by using areas. The major effort required to implement intersection using areas was not justifiable, so we decided to intersect the set of TCMs called by DIDs to isolate a failure. This means that when a set of DIDs occur in a two-week time period, the failure is isolated to TCMs common to each DID that occurs. To enable us to take corrective action in those rare cases where intersection does not call the correct set of TCMs, a record is kept of DIDs that were intersected for a given fault. If the error recurs, the identity of TCMs corresponding to the particular DID that recurred are still available to fix the problem.

• *Validation tests*

Validation tests (VTs) are tests for detecting and isolating stuck faults in combinational networks. Their generation and application in fault isolation for systems using LSSD has been extensively reported. References [11–13] contain techniques of generating tests for logic networks. References [1–3, 14] describe the concepts of LSSD and show how tests can be generated to isolate "stuck-at" faults for a machine designed with LSSD. For the 3081, validation tests are generated using the techniques described in these references. We now describe the role of validation tests in the overall diagnostic strategy and give some of the characteristics that are unique to the 3081.

The role of validation tests as part of the diagnostic microprograms is to confirm the existence of a fault in a TCM and, if a fault is found, to determine the failing TCM. The validation tests are also used to verify the repair action taken by the CE. VTs are primarily used for isolating those faults for which direct isolation and intersection isolation call more than one TCM. When the CE is called to fix such a problem, he is given instructions to vary off line the portion of hardware on which the VTs need to run. The CE signals the availability of the hardware, and the diagnostic microprograms automatically run the appropriate set of VTs. If the VTs find a fault, the names of the failing FRU or FRUs are displayed on the CE console and the CE is asked to replace them. If VTs do not detect a fault, sequential FRU replacement (described in the next section) is used to call FRUs.

Characteristics of validation tests Validation tests are part of the diagnostic microprograms which reside on the Processor Controller File. They are generated automatically by programs from the description of the logic of the 3081. The processes of scheduling the tests to be run, running the tests when hardware becomes available, and calling the failing FRUs are also automated.

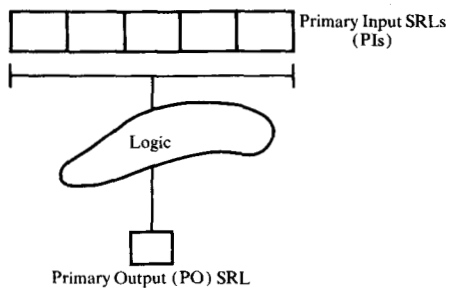


Figure 8 Segment of an SRL (shift-register latch).

The logic of the entire machine is divided into segments. A segment is defined for each SRL, called the *primary output* (PO) of the segment, in the machine. It consists of all SRLs such that the value of the PO SRL of interest at the end of this machine cycle depends on the current value of this set of SRLs [called *primary inputs* (PIs)] plus the combinational network whose output is the PO SRL. This relationship is shown in Fig. 8. A stuck fault in a segment can be detected by scanning a particular set of bits into the SRLs that are PIs and POs, running the machine one clock cycle, and comparing the value of the PO SRL to that of the same SRL of a fault-free machine. If the two values do not agree, there is a fault in the segment.

The running of VTs requires the scanning of data (test pattern) into a group of SRLs, running the clocks for one cycle, and scanning out a group of SRLs to detect a fault. A number of patterns are required to detect all possible sets of faults, and the procedure of scan-in, scan-out, and compare is repeated until a fault is found or all test patterns are exhausted and no fault detected. The latter case is assumed to be an intermittent fault. The scan-in, scan-out, and running of clock for one cycle is controlled by the Processor Controller. To minimize the time required to detect and isolate a given fault when multi-FRU DIDs are present, the entire set of tests is divided into groups of tests called a VT. Each has a unique name and there is a group of segments that the VT covers or finds faults in. Thus, only those tests that cover the suspected portion of the hardware need be run to isolate a given fault.

• *Sequential FRU replacement policy*

It was pointed out earlier that, for certain sets of failures, the direct and intersection isolation and validation tests cannot pinpoint the failing TCM (FRU). The DIDs of such failures are spread over more than one TCM. It is possible to fix such a failure with more than one repair action using, on an average, fewer replacement TCMs per failure than one

repair action would require. Sequential FRU replacement (SFR) policy applies to this set of failures, and it minimizes the average number of TCMs replaced per failure for a given average probability of fixing the failure in one repair action. A given failure is fixed in at most two repair actions. For a slight risk of not fixing a problem in the first repair action, a considerable reduction in the average number of TCMs per failure is achieved. From the point of minimizing impact to customer operations it was assumed that the average risk of requiring a second repair action for a failure in an SFR case should not exceed a very small value.

There are two aspects of SFR policy: the first problem is to determine the optimal set of TCMs to be replaced for a given failure to meet the objectives stated earlier, and the second problem is to implement the optimal strategy on the processor controller.

Optimal SFR policy An SFR strategy is completely specified by defining for each DID which TCMs are to be replaced in the first (repair action) call and which are to be replaced in the second. Corresponding to each strategy there is an average number of TCMs replaced per failure and a risk of second call. A computational procedure was developed by Rutledge [15] to determine the replacement strategy that has the minimum average TCMs per failure and for which the average risk of second call does not exceed a given percentage value. To calculate the average TCMs per failure and risk of second call, we use the following definitions. There are k DIDs in the system that call two or more TCMs. The failure rate of DID_i is the number of failures of circuits of DID_i in some specified time period T. It is proportional to the number of circuits, f_i, in the DID. Let Q_i be the conditional probability that, given there is a failure in the hardware covered by these DIDs, it is in DID_i. Then

$$Q_i = \frac{f_i}{\sum_{i=1}^{k} f_i}.$$

If DID_i has circuits on N_i TCMs, then $P_{j/i}$ is the likelihood that, given that a failure exists in DID_i, it is on TCM_j. $P_{j/i}$ is given by the following equation:

$$P_{j/i} = \frac{\text{Number of circuits of DID}_i \text{ on TCM}_j}{\text{Total number of circuits in DID}_i}.$$

For a given DID, for any strategy, TCMs are replaced in the order of decreasing likelihood. This reduces the average number of TCMs per failure and the risk of second call for a fixed number of TCMs that are replaced in the first call. If a failure occurs in DID_i, we replace TCM_j before $TCM_{j'}$ if $P_{j/i} > P_{j'/i}$. Therefore, there are N_i strategies to be considered for DID_i. The rth strategy, $1 \le r \le N_i$, is the one in which the r most likely TCMs are replaced in the first call.

For the entire set of DIDs, the replacement strategy is completely specified by a k-element vector V such that V_i, $i = 1, 2, \cdots, k$, is the number of TCMs replaced in the first call for the DID_i. If the V_i most likely TCMs of DID_i are replaced in the first call, then the probability of fixing the failure for DID_i in the first call, α_i, is given by

$$\alpha_i = \sum_j P_{j/i},$$

where j belongs to the set of V_i most likely TCMs of DID_i. The probability of second call for DID_i is $1 - \alpha_i$. The average number of TCMs for the two calls for DID_i, M_i, is given by

$$M_i = \alpha_i V_i + (1 - \alpha_i) N_i.$$

Therefore, the overall average number of TCMs per failure, $\overline{M}$, and probability of second call, $\overline{P}_2$, across all k DIDs are given by

$$\overline{M} = \sum_{i=1}^{k} Q_i M_i, \text{ and } \overline{P}_2 = \sum_{i=1}^{k} Q_i (1 - \alpha_i).$$

Both $\overline{M}$ and $\overline{P}_2$ are functions of V. The replacement strategy that minimizes $\overline{M}$ subject to the condition that $\overline{P}_2$ is less than or equal to a specified very small value meets the objectives of the SFR policy. This strategy is identified by the algorithm given in [15].

The SFR policy is implemented by associating an SFR bit with each TCM called by a DID. The SFR bit is set to 1 if the TCM is to be replaced in the first call for the DID and is set to 0 if it is to be replaced in the second call. These bits are set according to the optimal SFR policy identified previously and are permanently stored on the Processor Controller File.

When the CE comes to repair a hardware-detected intermittent failure for which multiple TCMs are called, the TCMs that are to be replaced in the first SFR call are identified by diagnostics and replaced by the CE. The diagnostic microprograms keep a record of the TCMs replaced. Two situations can arise when not all TCMs are replaced in the first call: either the faulty TCM is replaced, or it is not. We define that the nonreplacement case is implied if the same error is detected within two weeks of the replacement of TCMs in the first call. If no error is detected in that period, any subsequent occurrence of an error in the DID is treated as an occurrence of a new fault, because beyond two weeks the probability of a second failure is comparable to that of missing the failing TCM in the first call.

Diagnostic (isolation) effectiveness

One of the major requirements was that the average-number-of-TCMs-called-per-detected-TCM-failure had to

be 1.2 TCMs or less. The average number of TCMs per failure is a function of the percentage of faults that are solid faults. On the basis of the technology of the 3081 and experience on previous machines, 25% of the faults are assumed to be solid faults for the 3081. A secondary objective was to isolate a high percentage of the failures to a single TCM on the sole basis of the data captured at the time of error detection. This would limit the percentage of failures where a portion of the machine would not be available to the customer while diagnostics are running and would result in improved availability. This second objective means that a high percentage of the failures should be isolated to a single TCM by a combination of direct and intersection isolation. To evaluate the fault-isolation process in terms of these objectives, we first define certain concepts that enable us to calculate the isolation effectiveness. The actual evaluation is based on statistical sampling since this is a more cost-effective method than the analytical technique.

- *Analytical definition of effectiveness*

Any given area has a known failure rate. If C_i is the failure rate of area i and the failure of this group of circuits results in N_i TCMs to be called by the fault-isolation process, then the average number of TCMs per failure, $\overline{N}$, is given by

$$\overline{N} = \frac{\Sigma N_i C_i}{\Sigma C_i}.$$

Let $D_i = 1$ if, and only if, a failure in area i can be isolated to one TCM by direct and intersection isolation, and 0 otherwise. Then the average percentage of failures isolated to one TCM by direct plus intersection isolation (P_{DI}) is given by

$$P_{DI} = \frac{100\Sigma D_i C_i}{\Sigma C_i}.$$

Thus knowing N_i, D_i, and C_i we can calculate the necessary statistics. The 3081 Processor Unit consists of blocks (groups of circuits), each of which satisfies the criteria defined for an area. But there are hundreds of thousands of these blocks and an automated computer program would be required to determine N_i, D_i, and C_i for each block. Such a program could not be justified because of the effort required to develop it. Therefore, a statistical sampling approach was used to determine diagnostic effectiveness.

- *Statistical technique*

The statistical technique is based on sampling theory [16]. A random sample of n blocks is chosen from the total set of blocks in the 3081 Processor Unit such that the probability of choosing a block is proportional to its circuit count. The isolation of a failure of a block i is determined by the following rules:

1. If the block is in a set of DIDs such that there is a single TCM common to all DIDs, then for this block $N_i = 1$ and $D_i = 1$.
2. If the block is in a set of DIDs such that there is more than one TCM in the intersection of these DIDs, it is randomly assigned to one of the two following categories, with 25% probability that a block will be in *Category 1* (stuck fault) and 75% probability that it will be in *Category 2* (intermittent fault). For each block in Category 1, N_i is the number of TCMs called by VTs. This is available from the information produced by programs that generate VTs. For each block in Category 2, N_i is given by the SFR policy for the DID. Thus N_i is assigned according to the category in which the block falls. D_i is 0 for all blocks.

The sample mean and standard deviation are calculated and the 95% confidence interval is calculated from them. The formulas for calculating these intervals are as follows:

a. The average number of TCMs per failure is in the interval given by the following relation, with 95% confidence:

$$m \pm 1.96\frac{S}{\sqrt{n}},$$

where $m = \frac{\Sigma N_i}{n}$ and $S = \sqrt{\frac{\Sigma(N_i - m)^2}{n-1}}$.

b. The probability that a given failure is isolated to one TCM by direct and intersection isolation lies in the interval given by the following relation, with 95% confidence:

$$P \pm 1.96\sqrt{\frac{P(1-P)}{n}}, \quad \text{where} \quad P = \frac{\Sigma D_i}{n}.$$

A sample size of 662 failures was analyzed. The sample average number of TCMs per failure was 1.19, and the 95% confidence interval was 1.15 to 1.23 TCMs per failure. The estimated mean percentage of failures isolated to one TCM by direct and intersection isolation indicated that the design goals were met.

After the diagnostic package was developed, it was tested by bugging the hardware. These tests indicated very good correlation between measured and projected effectiveness.

Summary and conclusion

In this paper, we have presented the approach to the diagnostics of the 3081 system implemented in a new TCM-based technology. We developed and implemented a diagnostic methodology that automatically identifies the failing TCM in a customer environment. A novel feature of this

methodology is that diagnostic microprograms isolate intermittent as well as solid failures. A high percentage of the time, the failing TCM is identified without running any tests on the hardware. It was also shown how we met the *a priori* stated objectives for the diagnostic package.

The main trend in integrated circuit technology is to higher-and-higher-density packages (such as TCMs) to give users highly reliable and available machines that are easy to repair. An integrated approach such as the one presented in this paper is mandatory for meeting those objectives and for keeping service costs at a reasonable level.

Acknowledgments

The authors wish to acknowledge the work of J. H. Miller on defining a procedure for obtaining direct-isolation domains. Thanks are due to D. W. Krueger for providing an example of a direct-isolation domain, and to H. Schiller and H. C. Long for reviewing the paper and providing helpful comments. The members of the Diagnostic Development group implemented the strategy and participated in evaluating the isolation effectiveness.

References

1. E. B. Eichelberger and T. W. Williams, "A Logic Design Structure for LSI Testability," *Proceedings of the 14th Design Automation Conference,* New Orleans, LA, June 1977, pp. 462–468.
2. J. Pomeranz, R. Nijhuis, and C. Vicary, "Customized Metal Layers Vary Standard Gate-Array Chip," *Electronics,* 105–107 (March 15, 1979).
3. H. W. Curtis, "Integrated Circuit Design, Production, and Packaging for System/38," *IBM System/38 Technical Developments,* 11–15 (1978), Order No. G580-0237, available through IBM branch offices.
4. A. J. Blodgett and D. R. Barbour, "Thermal Conduction Module: A High-Performance Multilayer Ceramic Package," *IBM J. Res. Develop.* **26,** 30–36 (1982, this issue).
5. *A Guide to the IBM 3033 Processor Complex, Attached Processor Complex, and Multiprocessor Complex of System/370,* Order No. GC20-1859, available through IBM branch offices.
6. R. S. Swarz, "Reliability and Maintainability Enhancements for the VAX-11/780," *Proceedings of the 1978 International Symposium on Fault-Tolerant Computing, FTCS-8,* Toulouse, France, June 21–23, 1978, pp. 24–28.
7. A. Shah, "A Technique for Failure Isolation in a Large Computer System," *Proceedings, IEEE ICCC80,* Rye, NY, October 1980, pp. 238–240.
8. D. C. Bossen and M. Y. Hsiao, "Model for Transient and Permanent Error-Detection and Fault-Isolation Coverage," *IBM J. Res. Develop.* **26,** 67–77 (1982, this issue).
9. 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.
10. John Reilly, Arthur Sutton, Robert Nasser, and Robert Griscom, "Processor Controller for the IBM 3081," *IBM J. Res. Develop.* **26,** 22–29 (1982, this issue).
11. W. C. Carter, H. C. Montgomery, R. J. Preiss, and H. J. Reinheimer, "Design of Serviceability Features for the IBM System/360," *IBM J. Res. Develop.* **8,** 115–126 (1964).
12. P. S. Bottoroff, R. E. France, N. H. Garges, and E. J. Orosz, "Test Generation for Large Logic Networks," *Proceedings of the 14th Design Automation Conference,* New Orleans, LA, 1977, pp. 479–481.
13. T. W. Williams and K. P. Parker, "Testing Logic Networks and Designing for Testability," *IEEE Computer* **12,** No. 10, 9–22 (October 1979).
14. N. C. Berglund, "Processor Development in the LSI Environment," *IBM System/38 Technical Developments,* 7–10 (1978), Order No. G580-0237, available through IBM branch offices.
15. R. Rutledge and J. A. Santillo, IBM Corporation, Poughkeepsie, NY, private communication (1977).
16. W. A. Cochran, *Sampling Techniques,* 3rd edition, John Wiley & Sons, Inc., New York, 1977, Chs. 2 and 3.

Received December 14, 1979; revised July 9, 1981

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