Fault-tolerance design of the IBM Enterprise System/9000 Type 9021 processors

by C. L. Chen
N. N. Tendolkar
A. J. Sutton
M. Y. Hsiao
D. C. Bossen

The 9021-type processors offer the highest performance of the IBM Enterprise System/9000™ (ES/9000™) series. They also have the highest levels of concurrent error detection, fault isolation, recovery, and availability of any IBM general-purpose processor. High availability is achieved by minimizing component failure rates through improvements of the base technology, and design techniques that permit hard and soft failure detection, recovery and isolation, and component replacement concurrent with system operation. In this paper, we discuss fault-tolerant design techniques for array, logic, and storage subsystems. We also present diagnostic strategy, fault isolation, and recovery techniques. New features such as the redundant power system and Processor Availability Facility are described. The overall recovery design is described, as well as specific implementation schemes. The design process to verify the error detection, fault isolation, and recovery is also described.

1. Introduction

The IBM Enterprise System/9000™ (ES/9000™) Type 9021 Models 820, 860, and 900 are general-purpose data processing systems. They consist of a number of units, including a processor unit, a Processor Controller, a coolant distribution unit, and a power unit. A major portion of the system is implemented using thermal conduction modules (TCMs). A logic chip contains up to 5620 circuits, and a TCM contains either 100 or 121 logic and array chips. The ES/9000 Model 900 system has 44 to 58 TCMs, the Model 860 has 40 to 52 TCMs, and the Model 820 has 36 to 46 TCMs, depending upon the optional features used. In this paper, the ES/9000 Models 820, 860, and 900 are referred to as the 9021 system.

The subject of this paper is the fault-tolerant design used in the 9021 system to detect, recover, and isolate failures of circuits on TCMs and other components. Fault-tolerant design affects two figures of merit that are of interest to the customer: continuous availability and duration of repair. A key design goal of the 9021 system is to provide high levels of reliability, availability, and serviceability (RAS).

At the broadest level, the 9021 system is designed to provide continuous availability and the highest level of

data integrity in the presence of hardware failures. To provide these levels of fault tolerance, the hardware, the software modules in support of the 9021 system, and the Licensed Internal Code (LIC)[1] have been designed with special characteristics to support fault tolerance. All system hardware components including logic, storage arrays, and power have significantly increased levels of coverage in error detection, fault isolation, error correction, recovery effectiveness, and concurrent maintenance.

The bedrock upon which all of the hardware fault tolerance and data integrity in the 9021 system rests is concurrent error detection. Section 2 of this paper presents a technical description of the specific design techniques used in the 9021 system to achieve nearly the error-detection effectiveness of logic duplication and comparison, with less than 30% circuit overhead. Along with this, the added hardware design to accomplish greatly refined fault isolation is described.

Factors such as hardware reliability and recovery do help in reducing machine repairs. To provide high availability, the mean time required to repair the machine must be kept to a minimum for standalone systems, and means for recovery must be provided for sysplex (systems complex) [1] configurations that have backup facilities.

Built-in hardware error detection and FRU isolation have been successfully used in IBM 308X and IBM 3090™ processors to quickly isolate failures to a failing field-replaceable unit (FRU) [2, 3]. A key design characteristic of the IBM 308X and 3090 processors which is also used in the 9021 design is to a) detect errors during normal machine operation, b) capture machine status information at the time of error detection, and c) isolate the failing FRU by analysis of the data captured at the detection of the error. Fault-isolation design is discussed in Section 3.

A major goal of the 9021 system is to provide user-transparent recovery for the majority of hard and soft failures. From the hardware point of view, the processor unit is made up of logic and arrays. Error detection and retry are used for recovery from logic errors. Concurrent error correction and standby spares are used for array fault tolerance. Techniques used to recover from errors in arrays are discussed in Section 4. Sections 5 and 6 are devoted respectively to fault-tolerant techniques used in central storage and in expanded storage. Highlights of system-level recovery functions are given in Section 7. They include the 9021 design process, the Processor Availability Facility, channel recovery, concurrent repair, $N + 1$ power design for fault tolerance, Processor Controller fault-tolerant features, and sysplex recovery. A summary of the 9021 fault-tolerance design is given in Section 8.

[1] Licensed Internal Code (LIC) is software provided for use on specific IBM machines and licensed to customers under the terms of the IBM Customer Agreement.

2. Fault tolerance for logic circuits

- *Concurrent error detection*

In this section we discuss concurrent error detection of logic circuit failures. The 9021 machines are designed with an extensive concurrent error-detection and fault-isolation (ED/FI) capability which is the basis for error recovery, diagnosis, and repair. The term *error detection* refers to the ability of special error-detection circuitry to detect errors due to hardware faults during the actual run time of a computer. The result of such a detected error would be a number of internal activities, broadly described as error logging and recovery. The logged error state of a machine, prior to recovery, is used in a subsequent fault-isolation process.

Fault-isolation capability is the diagnostic resolution of the logged error caused by the single occurrence of an error [3]. The ED/FI error domain for any particular error checker is the set of faults which can turn this checker on. As the number of error checkers, and hence error domains, is increased, the average size of the domains is decreased, and thus the diagnostic resolution improves. Specific parameters to quantify the isolation capability are the average, the minimum, and the maximum number of parts (field-replaceable units, modules, chips, logic blocks, nets) contained in, or implicated by, the error domains. The target values for the 9021 machines were specified prior to design, on the basis of state-of-the-art error-checking mechanisms and error-reporting design experience from the 308X and 3090 systems.

From first principles, the only way to achieve 100% concurrent error-detection coverage is to use complete replication along with appropriately placed comparison circuits. With reasonably chosen comparison circuits, the system total for such an approach would be greater than 100% overhead, possibly as high as 120%. In IBM products, error-detecting logic design has been refined so that it is possible to achieve an error-detection level of nearly 100% with less than 30% overhead at the circuit level. Since the 9021 system uses gate array chips, the 30% overhead reduces to less than 10% at the chip level, and approaches 0% at TCM level; i.e., the total number of TCMs in a system is not affected by the 30% extra checking circuits. This is because most functional areas are pin-limited at package level, while error checking generally requires few external pins.

- *Error-detection mechanisms*

In order to achieve nearly 100% concurrent error-detection coverage with less than 30% circuit overhead, a set of well-chosen error-checking principles was developed and employed throughout the system design. These can best be described in terms of the following categories:

1. Parity for data flow registers.
2. Parity for control registers.
3. Parity predict for transformation logic.
4. Parity predict for sequential controls.
5. Decode and invalid combination checks.
6. Residue checking for arithmetic functions.

These categories correspond to the hardware function categories used to build a computer.

Use of parity checking

Parity checking is the most cost-effective method of error detection. Parity (usually, but not always, byte parity) is maintained on all registers, physical interfaces, and arrays, with the exception of those facilities encoded with some other qualified checking code such as ECC (error-correcting code) or m-of-n code [4]. The design goal for the 9021 system is to maintain and check parity or an equivalent code of every storage element (latch or array bit) in the machine. Error checking occurs on every machine cycle. This includes program-accessible data, instructions, internal control store information, and internal machine state information.

Coverage of parity checking

A parity bit in a logic register can be viewed as a cheap form of duplication of the register. The compromise in error-detection capability of parity versus duplication comes about because some single points of failure in logic (gating, clock powering, etc.) circuitry may corrupt multiple bits of the checked field, depending on data bit values, and may or may not be detected. Such physical failures are always detected, however, if they persist for multiple machine cycles. For this reason, parity is viewed as a legitimate compromise to substitute for duplication and comparison of all register bits.

Parity predict for transformation logic

Given the parity of the operands to an adder, hardware algorithms have been devised and implemented which operate in parallel with the adder and compute the expected, or predicted, parity of the sum. The predicted parity signal is then compared with the actual parity of the sum in order to detect adder failures. This check is performed every cycle the adder is used.

Parity for sequential control logic

Here, the idea is to encode the states of the state machine with a parity code, and then design the next-state logic so that physical failures result in a state with bad parity. The best way to do this is to implement the next-state function of each latch with separate logic, independent of the other latches comprising the state. This technique is used in the 9021 system in all binary counters, shifters, and control state machines.

Decode and invalid combination checks

The remaining control hardware of the 9021 system which does not fall into either the category of a register (parity checked) or sequential state machine (parity predict and checked) is classified as combinational logic. A general checking technique for combinational logic is to detect invalid output combinations. Most often, combinational logic is in the form of a decoder. The valid state at the outputs of a decoder is one in which *one and only one* output is active. Careful analysis of such circuitry has led the 9021 designers to choose a standardized *decode* check to give high coverage at very low overhead compared with the full *one and only one* check usually described in the literature [4].

Residue checking

The high-speed multiplier is a function for which the most efficient check is to predict the residue modulo 3 of the product. Such a check gives complete coverage for single errors, including carry logic, if checking occurs on every cycle as the partial products are generated and summed. In order to provide a more robust checking coverage and maintain maximum performance, an overall residue modulo 15 is used instead, for multicycle operations.

• *Error-checking usage summary*

Each hardware function in the 9021 machines is checked with a qualified error-checking circuit. **Table 1** summarizes the types of functions found in the 9021 system from a hardware standpoint, the approximate contribution of each function to a total bill of materials, the type of error checking appropriate to the function, and, finally, the approximate circuit overhead for the appropriate checking means. From this breakdown, the approximate percentage of total circuits used for error checking is in the range of 19 to 27%.

• *Verification of error detection and fault isolation*

Since the fault tolerance of the 9021 machines represents a significant additional set of functions, both hardware and software, the design process must include steps and tools necessary to verify these fault-tolerance functions. In order to verify the error-detection and fault-isolation characteristics of the system, an extensive family of tools and procedures has been developed to locate potentially unchecked hardware during development, so that necessary changes can be made before hardware is actually built. Similar procedures and tools are used during the recovery design.

For error-detection coverage, logic backtrace is used to locate all logic not directly connected to a qualified error checker. Unchecked logic groups are then resolved before the logic design is released. Fault injection in a simulation model of the machine is used to verify the correctness of the error domains for purposes of fault isolation.

Table 1 System function vs. checking technique and overhead.

Major function	*System total hardware* (%)	*Approximate checking overhead* (%)	*Checking technique*
Data flow arrays, interfaces	65	12.5	Parity
"Easy" control logic: Counters Control registers Address registers Shifters, etc. Comparators	15	20	Parity on small fields Parity predict
Sequential control logic: Control triggers Arithmetic logic	10	40–60	Parity and residue predict Illegal combination
Combinational control logic: Decoders Encoders, etc.	10	40–100	Decoder check Illegal combination Selective duplication and compare

3. Diagnostic fault isolation

The diagnostics goal for the 9021 system is to isolate 95% of the failures to a single FRU. For 5% of the failures, two TCMs plus any boards and wires that interconnect the TCMs are candidates for fault identification.

Let us consider the FRU isolation situation. It is desirable to confine 100% of failures to a single FRU to minimize the service cost. This is not always possible in a multi-FRU system, for reasons given below. In a multi-FRU system the board provides wires for inter-FRU communication. When a bad signal is received on the receiving FRU, it is not possible to tell whether the driver on the sending FRU has failed, the receiver on the receiving FRU has failed, or the board wire that interconnects them has failed. Hence, when an error is detected on an inter-FRU communication path, the failure is isolated to two FRUs. The percentage of failures for which two FRUs are called can be kept to a minimum by checking the signal on the sending FRU and checking the received signal on the receiving FRU. Roughly 5% of the total number of circuits on a FRU are used for inter-FRU communication, so the design permits calling two FRUs plus the board for failures of these 5% of circuits and calling a single FRU for all remaining failures.

Next, we describe two design techniques that are used to achieve a high percentage of single-FRU isolation. We also present the role played by the processor controller in FRU isolation. Finally, the fault-isolation process is described.

• *Design techniques for FRU isolation*

To isolate failures to a single FRU, it is necessary to minimize situations where failure of a circuit on one FRU is detected by a checker on another FRU. Two techniques are used to accomplish this. They are a) rules for checking of signals that cross FRU boundaries, and b) the active source identifier. These techniques are described below.

Checking of signals crossing FRU boundaries

An example of using error checkers at FRU boundaries to improve FRU isolation is shown in **Figure 1**. Signals to be sent originate at a parity-checked register on FRU 1. They enter combinational logic, array, or gating logic. Error checker E is used to isolate the failure of combinational logic circuits to a single FRU. When these circuits fail, checkers D and E are off and checker F is on.

Active source identifier

Besides the identity of the error checker that detected an error, certain machine status information must be captured in the error log to facilitate FRU isolation. The specific information called *active source identifier* (ASI) was first introduced in the IBM 308X processors [2] and is briefly explained here. It is used to point to the source of data in a situation where one source out of many supplies the data that are checked.

An example of an active-source identifier is shown in **Figure 2**. Register A or Register B could be the source of data that are sent to Register C. A latch (ASI) points to the source of data. Thus, ASI contains, e.g., a 0 if data came from register B, and a 1 if data came from register A. Consider data transfer from register B to C. The ASI would be 0. If parity checker C detects an error, the failure is isolated to FRU 2. In the absence of the ASI we would have to call both FRU 1 and FRU 2 when parity checker C comes on.

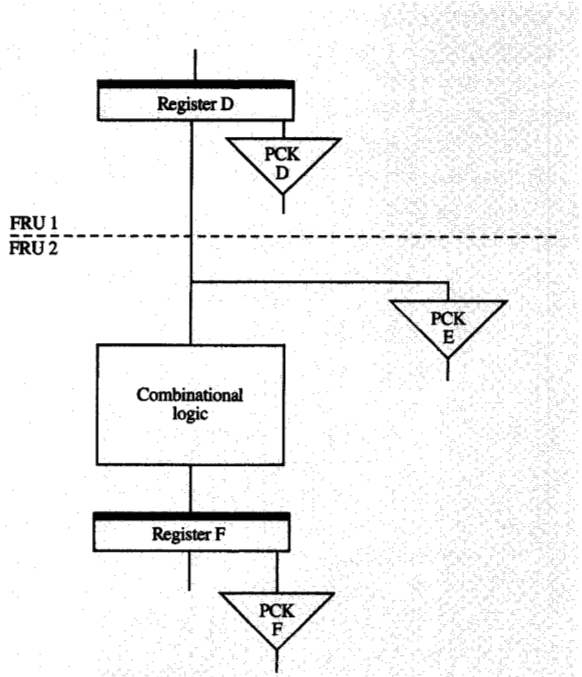

Figure 1

Checking the inter-FRU path. Checker E helps to isolate logic/array failures to a single FRU. PCK is a parity check operation.

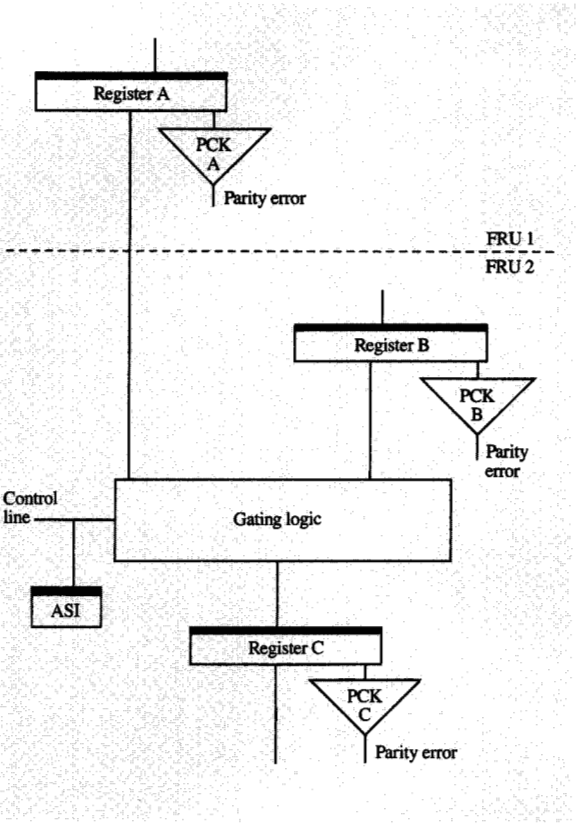

Figure 2

Use of active source identifier (ASI) for FRU isolation. PCK is a parity check operation.

• *Error log, syndromes, and domains*

To be able to determine the failing FRU, we need to collect data at the time of detection of an error. We first define the data collected at the detection of an error and then explain how they are processed to determine the failing FRU.

For every error checker there is a latch which is set to ON when the checker detects an error. Upon detection of an error, the status of each error latch and ASI is captured and recorded. This information is called an error log. A syndrome is a specific combination of values of bits in the logout. Values of error latches and ASIs uniquely specify a syndrome. The domain of a syndrome S, D(S), is a set of circuits such that the failure of any circuit in D(S) could cause the syndrome S to occur. If a circuit is not in D(S), its failure cannot cause the syndrome S. F(S) is a set of FRUs such that FRU i belongs to F(S) if and only if at least one circuit from FRU i is in D(S).

It is clear from the definition of the domain that if any circuit in D(S) fails, syndrome S occurs, and it is isolated to FRUs in F(S). By using trace-back programs, the domain of each syndrome is identified and the corresponding FRUs are identified. This map of syndromes to FRUs to be replaced is encoded and is an integral part of the diagnostics Licensed Internal Code (LIC) that runs on the Processor Controller for isolating failures to the failing FRU.

• *Role of the Processor Controller in FRU isolation*

The role of the Processor Controller in FRU isolation is similar to that in IBM 308X and 3090 machines. The Processor Controller is a separate independent processor that provides operation monitoring, control, and maintenance support for the 9021 processor complex. It provides extensive error recording, recovery, and diagnostic support for the processor complex. It is connected to the 9021 processor complex via a logic support element (LSE), as shown in **Figure 3**. The diagnostics LIC runs on the Processor Controller. A display console is provided for the customer engineer (CE) to obtain information about any detected failure and the FRU or FRUs that must be replaced in order to repair the machine.

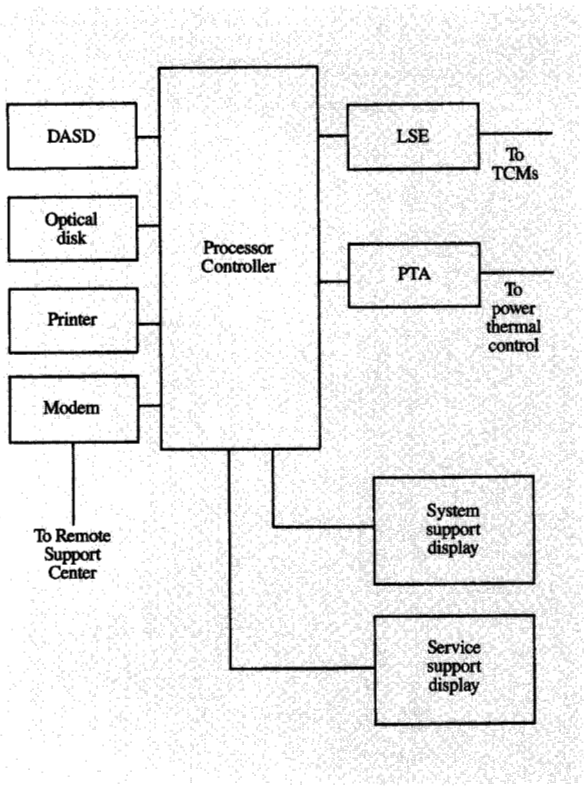

Figure 3

Processor Controller.

- *Fault-isolation process*

The function of the fault-isolation process is to process the error log and determine the failing FRU (two FRUs are called when an inter-FRU path fails). The fault-isolation process is illustrated in **Figure 4**.

When a hardware error is detected by an error checker, the Processor Controller performs recovery and creates an error log. If recovery is successful, the 9021 processor continues operating while the Processor Controller analyzes the error log. Analysis routines (ARs) analyze the error log and determine the error syndrome, and then identify the FRU to be replaced. Intersection isolation is used to further isolate or consolidate FRU calls when several errors occur in a short period of time [2]. If the frequency of the error is over a threshold or if a processor, a subelement, or the entire machine is put in a check-stop state because it is impossible or undesirable to continue operation, a message is sent to the Remote Support Center requesting service. The message also indicates the FRU that needs to be replaced. A history of all detected errors including any part replaced by the CE is maintained on the Processor Controller DASD (disk storage). The last step is to repair the machine by replacing the failing FRU.

4. Internal array fault tolerance

In this section we consider the techniques used to provide fault tolerance for hard and soft failures in various internal arrays used in the central processor, system controller, and interconnect communication elements.

An array is made up of cells each of which can store one bit of data. When a single-bit soft failure occurs, the information stored in a cell is lost, but the cell itself does not suffer any permanent damage; i.e., new data can be stored and retrieved from the cell when a soft failure affects it. On the other hand, a hard failure results in permanent damage to some portion of the array. Data cannot be stored in the cells affected by a hard failure. A hard failure may affect just one array cell, or it may affect several cells.

The design goal for the 9021 machines is to recover from most soft failures in internal arrays. Major arrays are also designed with redundancy schemes that permit hard fault tolerance. The primary fault-tolerance methods for arrays are the use of error-correcting code (ECC) for error correction or the use of error detection and retry. In the

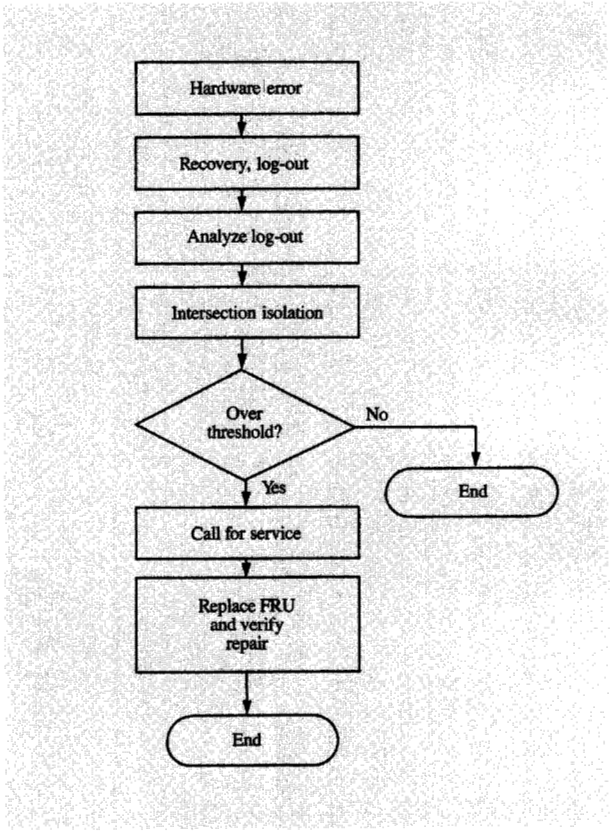

Figure 4

Fault-isolation process.

latter case, backup data must be kept so that the array that has bad data can be restored. An important fault-tolerance characteristic is to prevent the occurrence of uncorrectable errors and undetectable errors in arrays when hard errors are allowed to accumulate. There are two methods available for handling hard errors in arrays to prevent the occurrence of uncorrectable and undetectable errors. One, called *line delete*, is used to delete the portion of an array that has a hard failure. The system continues processing data without the portion that was deleted (each delete reduces the capacity of the array). The second method, called *relocate*, uses spare word lines in the array. The array is made up of M word lines, whereas the system needs N, $N < M$, at a time. Hardware selects N good word lines out of M. When a hard failure occurs, the bad line is replaced by one of the good spare lines. Line delete and relocate can both be used for fault tolerance of an array.

Some examples of fault-tolerance schemes used for arrays are given below to illustrate the concepts. We shall discuss cache arrays in the central processor and system controller, and the control store array.

The 9021 system contains cache arrays in the processor and the system controller. The 9021 central processor L1 cache is a store-through cache consisting of an instruction cache and a data cache. Both caches are protected by parity. The 9021 machine also has an L2 cache which contains a copy of L1 data. Data in L2 are protected by an ECC that corrects single errors and detects double errors. Single-bit failures in L1 are corrected by the backup copy from L2. Error rates are monitored by the error handler module in the Processor Controller to identify hard failures. When a hard failure is detected in L1, the line containing the failure is deleted and is no longer available for use. This prevents the occurrence of a 2-bit failure in the future which could result in an undetected error. Further, L1 caches also have spare word lines available for hard fault tolerance. When line-delete capacity is used up, one of the spare lines replaces one of the failing lines. A single-bit failure in L2 is corrected by the ECC. If the frequency of errors in a given line in L2 is over a threshold, the line containing the failure is deleted.

The central processor LIC is stored in the control store array, which is also parity-protected. The Processor Controller has a copy of the data in the control store array. When an error is detected in the control store array, the Processor Controller refreshes the data. The processor control store array also has spare capacity built into the array chips. Upon detection of a hard error in the array, the faulty line is replaced by a spare line. The Processor Controller reloads the control store data into the spare line and the processing continues.

A flow chart that describes the handling of errors in arrays in general is shown in **Figure 5**. The process begins with the detection of an error when data are read out of the array. The first step is to correct the error and provide good data to the requester. For ECC-protected arrays, error-correction logic is used to correct single-bit errors in data. For some arrays, duplicate copies exist for performance or functional reasons. For some other arrays, duplicate copies are used for correcting errors. If an error occurs in one array, the backup copy is used to reload the data into the array.

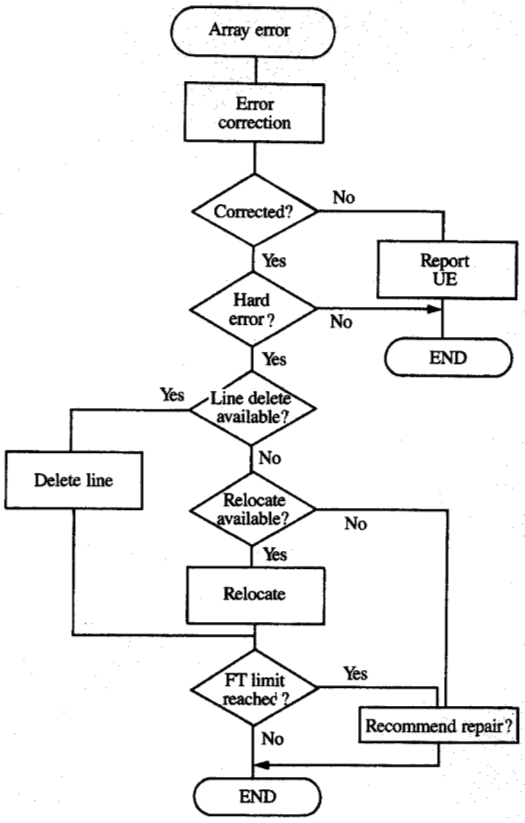

Figure 5

Array fault tolerance flow chart.

If an error is corrected, the next step is to determine whether the error is a hard error. For each array, a history of the location and frequency of prior errors is maintained. If the frequency of errors is over a certain threshold value, it is assumed that the errors are hard errors.

When a hard error occurs and is corrected, the following actions are taken. If a line-delete mechanism is available for the array, the affected line is deleted. A line is a

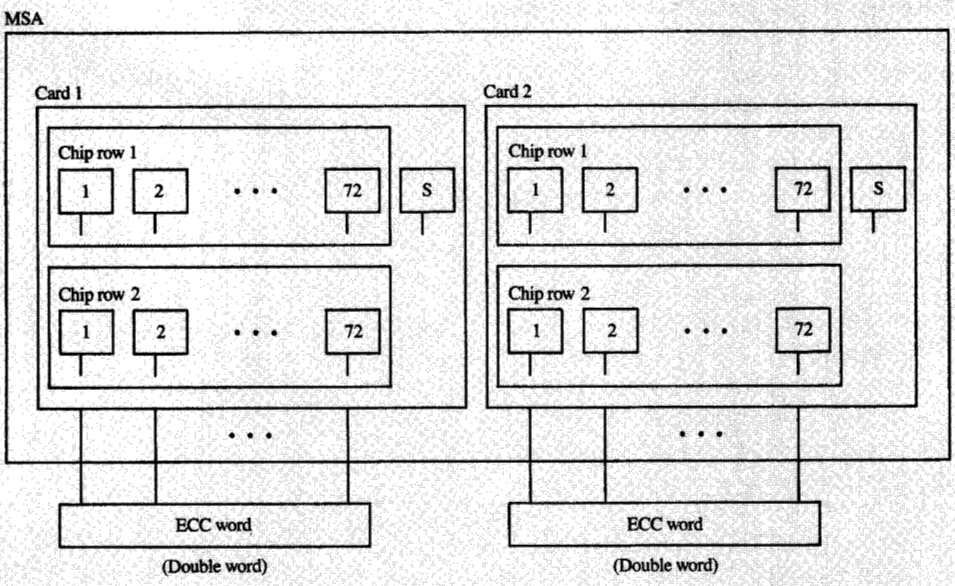

Figure 6

Largest main storage array (MSA) configuration.

particular portion of an array, and the number of bytes (for example, 128 bytes) in a line is different for different arrays. If line deletion is not possible or available, a relocate is attempted for arrays that have spare word lines. Some arrays have two spare word lines and some have five to support relocation.

Finally, spare capacity left for further fault tolerance is examined. If it is determined that the next error in any array is going to be uncorrectable or undetectable, a recommendation is made for repairing the machine at a scheduled time, or concurrently (as discussed earlier) for replacing the module that contains the array.

5. Central storage fault tolerance

Central storage in the 9021 system may consist of one or two main storage elements (MSE). An MSE contains up to four main storage arrays (MSAs). Each MSA has two memory array cards; each array card can be populated with up to 145 DRAM chips, one of which is a spare chip. The maximum MSE capacity is 512 MB (megabytes) of data.

A number of fault-tolerance features have been designed into the central storage subsystem. An error-correcting code (ECC) is designed to correct all single errors and detect all double errors in every ECC word. A double-complement procedure [5] is implemented to recover data from double errors by a fetch with retry request to MSA. A memory-scrubbing scheme is used to prevent the accumulation of soft errors and also to monitor hard errors in each memory chip. In addition, the spare memory chip is dynamically activated to replace a failing chip that contains a preset threshold of failing bits.

The reliability improvement of using ECC over simple parity checking ranges from two to three orders of magnitude, depending on the failure rates and failure modes of the memory chips. Spare chip replacement of a failing active chip offers at least another order of magnitude reliability improvement over the ECC. The Fault-Tolerant Memory Simulator (FTMS) [6] has been used to study fault-tolerant memory design options and analyze the 9021 central storage reliability.

- *Central storage ECC*

Data in the memory chips are protected by a (72, 64) SEC–DED code [5] that is capable of correcting all single errors and detecting all double errors in every ECC word of 72 bits, 64 of which are data bits and eight of which are check bits. An MSA array card contains one or two chip rows of 72 memory chips (**Figure 6**). The data are organized in one bit per chip so that each of the 72 bits of an ECC word is stored in a different chip. Regardless of the number of failing bits, a failing memory chip can

corrupt at most one bit of an ECC word. Thus, the memory can tolerate a total failure of one of the 72 memory chips associated with the same set of ECC words. Multiple errors in different chips are also correctable as long as the errors do not align in the locations associated with the same ECC word.

Memory errors are classified into hard and soft errors. Hard errors are reproducible errors due to permanent physical or electrical damaging of memory cells or the peripheral circuitry of the memory chips. Depending on the nature of the failures, the number of hard errors per failure event may be one bit or multiple bits. Multiple hard errors in a memory chip may be generated from a single bit-line failure, a single word-line failure, or other failures that affect a large area of a memory chip. Soft errors are temporary errors caused by alpha particles or other transient electrical disturbances; they comprise mostly single bit errors per failure event. A soft error is corrected when the location that contains the error is stored with new data.

As the machine continues to operate, the number of failing memory cells increases. In the beginning, the memory errors are all correctable by the ECC. As the failures accumulate over a period of time, it becomes more likely that two or more failures will align in the locations that belong to the same ECC word. Thus, the probability of uncorrectable errors increases with time. To reduce the probability of uncorrectable errors in the memory, a background memory-scrubbing scheme, described below, is designed into the 9021 central storage system to prevent memory error accumulation. If there is an uncorrectable error event, a fetch with retry request is issued to exercise a double-complement procedure for the recovery of double errors.

- *Fetch with retry request*

In the 9021 system, a *double word* represents 8 bytes or 64 bits of data and is stored as a 72-bit ECC word in a set of 72 separate memory chips in an array card. A *quad word* is composed of two double words. A *double line* consists of 16 quad words or 256 data bytes. Data fetch requests from the 9021 processor to MSA are on a double-line basis. Physically, a double line of data is stored in 144 array chips that are evenly spread over two array cards in the same MSA. Thus, a double-line fetch involves a processing of 16 ECC words (double words) from each of the two array cards, i.e., 16 quad words from two array cards.

In a data fetch request, if one of the ECC words in the double-line boundary contains a UE (uncorrectable error), the processor may issue a fetch with retry request to recover the data. The execution of the request is a macro operation of four storage accesses (two stores and two fetches) for the MSA involved. During the macro operation, other requests for the MSA are put on hold until the operation is finished.

The fetch and retry request procedure is designed to correct double errors in double words. All double errors can be corrected provided that one of each double error is a hard error. The algorithm used involves a complement and a recomplement of data and is known as the double-complement algorithm [5]. Once the double errors in the double words of a double line have been corrected and the data have been sent to the requestor, the page (composed of 16 double lines) that contains the double line may be deallocated to prohibit its further use.

- *Central storage scrubbing*

The memory-scrubbing process implemented in the 9021 central storage serves two functions. The first function is to eliminate the accumulation of soft errors in the memory array chips. The purpose is to reduce the likelihood of the alignments of existing soft errors and future hard or soft errors. The second function of memory scrubbing is to identify and record hard errors in the memory chips. Multiple hard errors are prime targets for any other error to line up in the same ECC word and result in a UE (uncorrectable error) event. Once a memory chip with multiple hard errors is identified, a spare chip replacement procedure may be invoked to transfer data from the failing chip to the spare chip on the same memory card. The failing chip then is set to become inactive.

At a selected interval, the Hardware Assisted Memory Tester (HAMT) of Central Storage Control initiates a scrub request to an MSA. The scrubbing process requires four double-line accesses: two fetches and two stores. The process runs in the background, with minimal interference with the normal system operations. The entire central storage is scrubbed in less than four minutes. A set of 144 10-bit error counters is used to record the number of hard errors in a chip row of 144 memory chips in two array cards of an MSA.

All single soft errors in double words are corrected and removed from the memory by the scrubbing procedure in a matter of a few minutes. The probability of having two independent soft errors aligned in the same double word is essentially zero. The scrubbing procedure also keeps a record of the number of hard errors for each of the 144 data bit positions of the two array cards in an MSA.

- *Dynamic chip sparing*

Each memory array card has a spare memory chip. It can be used to dynamically replace any failing memory chip on the same card. The replacement process is invoked when the number of errors in a memory chip reaches a preset threshold.

The purpose of replacing a failing memory chip is to prevent UE events. Since the memory array is organized

in one bit per chip, the ECC is capable of correcting all errors in one memory chip. Thus, the first memory chip failure does not affect the normal system operation. As the system continues to operate, it is likely that there will be a second failing memory chip some time during the lifetime of the machine. If the failure mode in both failing chips is a single cell, the two failing cells are not likely to align in the locations associated with the same ECC word. On the other hand, if the failure mode of a failing chip involves multiple cells, it is likely that two failing cells of two chips would align in locations belonging to the same ECC word. Thus, to prevent UE events, the spare chips are used to replace failing memory chips containing multiple failing cells.

When the entire address range of a chip row in an MSA has been scrubbed, the HAMT starts to examine the number of hard errors in the 144 error counters. The error counters are divided into two groups of 72 each corresponding to a chip row in each of the two array cards. Two separate sets of counts are kept for the two groups of 72 error counters. In each group, the first counter whose value exceeds the threshold value determines the chip location within the chip row to be replaced by the spare chip on the array card. All error counters are reset to zero at the end of the examination process.

When the failing chip location for replacement has been determined and the spare memory chip is available, the HAMT sends a sparing vector to MSA and starts the spare chip replacement procedure. The sparing vector contains the location of the memory chip to be replaced and the sparing mode. The three sparing modes are sparing not active, store only, and full fetch/store modes. The sparing mode is initially set at "sparing not active." The spare chip replacement procedure is the following:

1. The sparing mode is set to "store only."
2. The data bit in the chip row of the memory chip, say i, to be replaced is fetched, passed through the ECC, and stored back. In the store operation, bit i of a double word is stored in memory chip i as well as the spare chip.
3. The sparing mode is set to "full fetch/store" mode.

6. Expanded-storage fault tolerance

Expanded storage is a high-capacity electronic extension of central storage. The 9021 expanded storage consists of four to eight expanded storage array (ESA) boards. The number of array cards on each ESA board ranges from 4 to 16. The number of memory chips on an array card depends on whether the card is designed for 1Mb (megabit) or 4Mb memory chips. There are 292 memory chips per card for the 1Mb chips, and there are 146 memory chips per card for the 4Mb chips. The maximum capacity of a four-board ESA is 4 GB (gigabytes) of data.

The expanded storage of the 9021 system has many of the fault-tolerance features of central storage. It uses an ECC and has spare memory chips. It uses background scrubbing to eliminate soft-error accumulation and to record failing cells in the memory chips. It also dynamically replaces a failing memory chip with a spare memory chip when the number of failing bits of a chip reaches a threshold. A double-complement algorithm is also implemented to recover data from uncorrectable errors. The background scrubbing, the double-complement algorithm, and the spare memory replacement algorithm implemented in the expanded storage are similar to those in the central storage and are not discussed further. However, there are major differences between the expanded storage and the central storage in the design of spare memory and ECC.

The memory chips of the expanded storage are organized in one bit per chip, the same as in the central storage. In the central storage, an ECC word is stored in a single array card, and there is only one spare memory chip per card. In the expanded storage, an ECC word is stored in eight separate array cards. There are two spare memory chips for every group of 144 active memory chips on an array card. In each group, the two spare chips can be sequentially and dynamically switched into active duty to replace any two of the 144 active memory chips that have failed in more than a preset threshold number of bits.

While the central storage uses a single-error-correcting and double-error-detecting (SEC–DED) code, the expanded storage uses a double-error-correcting and triple-error-detecting (DEC–TED) code. The DEC–TED code is a (144, 128) code that uses 16 bits to check a set of 128 data bits [7]. Since there are more memory chips in the expanded storage, there is a higher probability of memory chip failures in comparison to the central storage. The DEC–TED code is required to maintain high reliability of the expanded storage. In the rest of this section, we describe for the first time the construction and the decoding algorithm for the code.

• *Expanded-storage ECC*

The (144, 128) DEC–TED code used in the 9021 expanded storage uses 16 check bits for 128 data bits. The code is obtained by shortening a (255, 239) cyclic BCH code C whose generator polynomial contains α and α^3 as roots, where α is a root in the finite field GF(2^8) of a binary primitive polynomial of degree 8 [8]. The positions of code C are labeled as α^i, $0 \le i \le 254$. Define the *trace* of α^i by

$$T(\alpha^i) = \sum_{m=0}^{7} \alpha^{i2^m},$$

which has a binary value. Then the null space of code C contains the vector $V = [T(1), T(\alpha^3), T(\alpha^{3\times 2}), \cdots,$

$T(\alpha^{3\times 254})]$. Let $I = \{i | T(\alpha^{3i}) = 1, 0 \le i \le 254\}$. There are 144 elements in I. Vector V has ones at positions α^i for all $i \in I$ and has zeros elsewhere. Thus, vector V has 144 ones and 111 zeros. Let F be the code obtained from code C by deleting the positions where V has zeros. Then code F is a (144, 128) code. Since the null space of code F has an all-ones vector that provides an overall parity check of all bits in a code word, code F is a DEC–TED code, having a minimum distance of 6.

Single-error-correcting codes capable of detecting multibit symbol errors have been introduced in [9]. SEC–DED codes capable of detecting symbol errors have also been studied [10, 11]. The concept of symbol error detection can also be extended to DEC–TED codes. In particular, bit positions of code F can be arranged in a certain order to provide error detection of all single 4-bit symbol errors.

A DEC–TED code is capable of detecting single 4-bit symbol errors if all 4-bit symbol error syndromes are different from the set of correctable error syndromes, which include the all-zero syndrome, all single-error syndromes, and all double-error syndromes. The (144, 128) DEC–TED code used in the 9021 expanded storage is the same as code F with bit positions arranged in a special order described in [7]. The code is capable of detecting all single 4-bit symbol errors. In fact, the code has the additional property that all single 4-bit symbol error syndromes are distinct. Given the occurrence of a symbol error, the syndrome can be used to uniquely identify the symbol error location and the associated 4-bit error pattern. This property is useful in isolating memory support logic failures as described in [7].

When a 144-bit ECC word is fetched from the storage, a set of 16 parity check equations are used to generate the error syndrome. If the syndrome is an all-zero 16-bit vector, no error is assumed in the fetched word. If the syndrome is not an all-zero vector, the ECC decoder determines the positions of the errors. Assuming that there are two errors, at positions X_1 and X_2, the syndrome can be represented by two 8-bit vectors, S_1 and S_3, with $S_1 = X_1 + X_2$, and $S_3 = X_1^3 + X_2^3$. Given the syndrome (S_1, S_3), the decoding algorithm finds the error positions X_1 and X_2, where the S and X values are considered elements of the finite field GF(2^8) of 256 elements.

The decoding algorithm described in [12] has been used to decode the expanded-storage DEC–TED code. The algorithm is derived from the relation

$$\begin{aligned} D &= S_1^3 + S_3 \\ &= X_1^3 + X_1^2X_2 + X_1X_2^2 + X_2^3 + X_1^3 + X_2^3 \\ &= X_1^2X_2 + X_1X_2^2 \\ &= X_1X_2(X_1 + X_2) \\ &= S_1^2X_1 + S_1X_1^2. \end{aligned}$$

Note that the squaring of a finite field element is a linear operation. The expression $S_1^2X + S_1X^2$ can be represented by S_1T_x, where T_x is an 8×8 binary matrix uniquely defined by X. The double-error-decoding algorithm can be described as follows:

1. Compute $D = S_1^3 + S_3$.
2. For each of the 144 code word positions X, compute $Q_x = S_1T_x$. The computation of all 144 Q_x values can be carried out in parallel in hardware involving exclusive-OR circuits.
3. If $Q_x = D$ for a particular X, then X is a position of error. The data at position X are then corrected by an inversion of the data bit.

7. Recovery design

Recovery is the final link of fault tolerance. Internal TCM array recovery has been discussed in previous sections. In this section, we describe the recovery of other system components and the overall system recovery scheme. The recovery design of the 9021 system is based on proven techniques used in IBM 308X and 3090 systems. However, significant improvements have been made in the 9021 system to provide recovery for most hardware failures.

In the rest of this section, we describe salient features of the recovery design of the 9021 system.

- *History*

Since recovery design is an extremely complex task, and different system functions are designed by different groups of engineers, recovery design needs careful coordination to avoid design errors. Therefore, a Recovery Design Council (RDC) was formed to set the direction of hardware recovery. This RDC consisted of designers who had participated in the recovery design reviews of the predecessor systems, IBM 308X systems and IBM 3090 systems. The RDC has as its members a representative from each of the main system elements, such as Central Processor, System Controller, Input/Output Subsystem, and Processor Controller, as well as representatives from the Recovery System Simulation and Recovery System Test organizations. This RDC reviewed the 9021 system design to ensure that the design has no data integrity problems, conforms to ESA (Enterprise Systems Architecture) and machine check architecture, and meets the goals set for overall recovery effectiveness and performance specifications.

In the early phase of system logic design, the RDC created the overall system recovery rules document, which contained the rules the designer must follow. The rules addressed the following topics:

- Recovery of logic and array failures.
- Recording of error data.

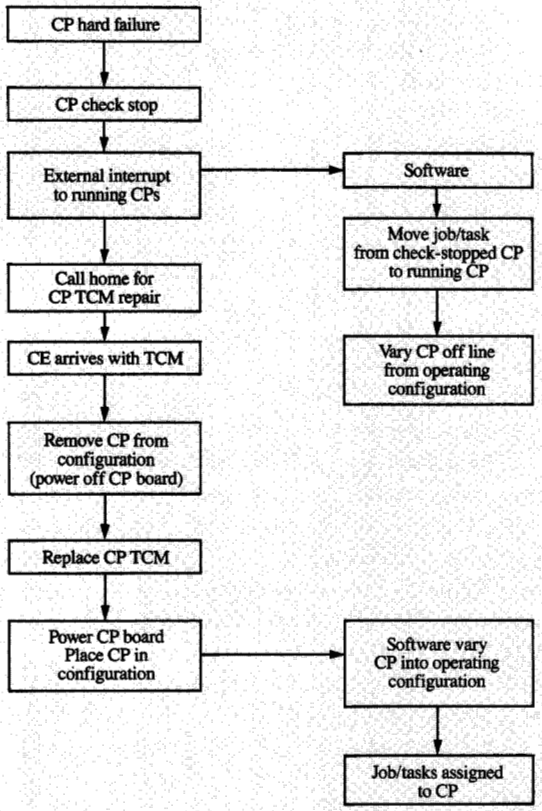

Figure 7

Processor Availability Facility and central processor (CP) TCM concurrent maintenance.

- Thresholding of errors.
- Recovery resets.
- Recovery performance and effectiveness.

- *Design process*

The early design philosophy concentrated on the recovery of intermittent hardware failures. The recovery design goal is that all logic will be recoverable for an intermittent (soft) logic failure. The same recovery goal applied to array chip failures, i.e., that all single-cell array failures will be recovered.

The mechanics used for recovery could vary depending upon the amount of logic to be addressed. Logic failures could be recovered by retrying a command or by retrying a full element such as a CP. The choice of implementation is a trade-off of the amount of special support hardware and Processor Controller recovery code versus a generalized recovery design of recovering the whole element for all types of errors. Logic recovery algorithms are also invoked for timing problems in which one element may time-out waiting for response from another element. Retry of the command or element will correct the problem by altering the state of the elements, e.g., purging the cache and restarting. The use of this recovery capability is helpful in bring-up of the system by making it possible to recover from design errors until the design is corrected.

The 9021 system contains a large amount of nonfunctional logic which in past systems was so integrated with the functional logic that a failure in the nonfunctional logic could permanently prevent the functional logic from recovering. Examples of nonfunctional logic are error-recording logic, hardware trace logic, system activity display logic, address compare logic, scrub logic, and sparing logic. This condition does not exist in the 9021 system. The recovery design in the 9021 is to separate this nonfunctional logic so that its errors, when detected, will be thresholded; if over threshold, the logic is physically fenced from the rest of the element. The nonfunctional error interrupt is masked, an error report is made, and the field-replaceable part is scheduled to be repaired at a noncritical time.

- *Processor Availability Facility*

The Processor Availability Facility (PAF) of the 9021 system allows the system to continue running when a processor has a hard error. It is implemented by a combination of hardware and software. When a processor hard failure occurs, PAF is invoked. With PAF, the system recovers the task that was running on a failed processor by moving it to another processor. The Processor Controller does the following when PAF is invoked:

1. Puts the failing processor in check-stopped state.
2. Puts data stored in all the architected facilities (e.g., general-purpose registers, and other facilities needed for program execution) of the failing processor into the register save area of main storage. The validity bits for the data are set in the machine check interrupt code (MCIC) area. The data now can be accessed by the other processors.
3. Signals other processors in the configuration.

Software modules in MVS/ESA™, VM/ESA®, and PR/SM™ [13] recognize the signal generated by the Processor Controller by inspecting the MCIC. These software modules then execute the task that was running on the failing processor on one of the remaining processors in the configuration. **Figure 7** shows the steps of the recovery process for a CP failure in a multiprocessor system. This recovery operation is transparent to the task

that was running. In some rare cases, certain MVS/ESA tasks for example, PAF may not be able to recover the failure.

- *Channel recovery*

The channel subsystem attempts recovery for channel subsystem failures. For some channel failures, recovery might not be possible. When recovery is not possible, all damaged operations are terminated and the affected element (for example, a channel) is removed from the configuration. Upon termination of the operation, the system control program initiates a recovery action which is successful in many cases.

- *Concurrent repair*

The 9021 system design permits the replacement of a failed CP TCM while the system is running. After repair the CP becomes available for processing data.

When a hard failure occurs in a CP, the PAF is invoked to move the failing CP off-line, as described earlier. The CE is notified that the machine needs repair and is told which specific TCM needs to be replaced. When the CE arrives at the installation with the new TCM, the operator logically removes the CP from the configuration via a software command, unless the CP has already been removed from the configuration. The CE then physically removes the CP from the configuration and powers down the power boundary of the CP board containing the failing TCM by executing a service language command (SLC) of the service console of the Processor Controller. The TCM is replaced, and power is returned as the SLC is executed to physically bring the CP into the configuration. The operator then logically configures the CP into the operating configuration. A reset of the logic is done to the CP, and, if successful, the CP is returned to the operating configuration and software assigns tasks to the CP. If the reset fails, the Remote Support Center is called for assistance.

Prior to the introduction of the 9021 machines, channel card replacement required a scheduled system outage. The 9021 system is designed for concurrent channel replacement when a channel card fails. To replace a faulty channel card, one channel (two for ESCON™ [14]) is taken off-line. The channel is put in single-channel service mode. The faulty card is replaced and the channel is varied back on-line.

- *$N + 1$ power design for fault tolerance*

The 9021 system is designed to tolerate the failure of any one power supply [15]. When a power supply fails, the system continues operating and the recovery is transparent to the user. The 9021 system is divided into various power boundaries. For example, a CP in a six-way multiprocessor system forms one power boundary. If one power boundary normally requires N power units to supply enough power, an extra power unit is added to form a $N + 1$ power unit for the power boundary. The spare power supply is switched in whenever one of the active power supplies fails. Repair of the failing power supply is done at a time convenient for the customer and is concurrent with system operation.

- *Processor Controller fault-tolerant features*

The 9021 Processor Controller is a duplex system and has been designed to be fault-tolerant in single-image mode. When it is physically partitioned, it will support a multiprocessor (MP) installation as two separate complexes with a Processor Controller on each side. When an MP system is operating in single-image mode (both sides under one operating system), only one Processor Controller is active. The other Processor Controller is a backup. The two Processor Controllers have an "I am well" communication, and the backup can force a takeover of the active one when it detects that the active controller has a failure.

- *Sysplex recovery*

Today many large-system customers are using sysplexes [1] and Processor Resource/Systems Manager™ (PR/SM™) [13] capabilities. Two 9021 systems, one called active and the other called backup, can be used in a sysplex so that both have access to shared data. In this situation, we need to ensure that if one 9021 machine fails, the other can continue operation and has access to the shared data. The 9021 system provides the system operator an optional mode of recovery for these environments. This mode assists automatic takeover by the backup when an active system fails.

The ESA design of the I/O interface allows a system in a shared I/O environment, such as shared DASD, to initiate a reserve to the DASD so that it blocks the other system's access to the DASD until it completes its operation. When the active system hardware fails, the recovery action is to free up access to all of the shared input/output resources that were reserved by the active system. This is done so that the backup system can access the resources. To free up the resources, the failing 9021 machine resets all of its outstanding reserves when in this optional mode.

Software failures in the active system can result in MVS/ESA or VM/ESA putting the system in a nonrestartable wait state. If this happens, operator intervention and possibly an initial program load (IPL) is required to restart the system. The following actions are taken to recover from this situation. There is a new instruction issued by the operating system to the Processor Controller in this optional mode, to reset the I/O reserves when selected by the operator to do so. A special message is displayed on the System Console, when the system is

check-stopped in this mode. These messages, when monitored by a focal point system,[2] can be used to activate a backup system to begin its takeover. This minimizes takeover delay and avoids manual operator intervention. This same support is provided in a PR/SM environment.

8. Summary

The IBM 9021 system contains many significant design improvements in fault tolerance over previous systems. It has an extremely high level of concurrent error-detection capability, better concurrent failure-isolation capability, and very high diagnostic resolution, as well as an error-recovery capability for failures of on-TCM array chips and the central and the expanded storage arrays. The DEC–TED code, which is unique to the IBM machine, is employed in the expanded storage to provide a high degree of fault tolerance. A new feature of the 9021 system is the use of the $N + 1$ power design concept for fault tolerance of power supply failures. Another new design feature is the Processor Availability Facility, which allows the system to continue operation when a processor has a hard failure in a system that has two or more processors. Recovery makes most failures transparent to the users, and concurrent maintenance enhances availability. For many failures, when repair is needed, the repair can be deferred to a time convenient for the user. All these features contribute to providing high system availability to the customers.

Acknowledgment

Many designers have made significant contributions to the ES/9000 Type 9021 fault-tolerance design, and it is not possible to mention all of their names. However, we would like to thank Dan Kolor, Ann Merenda, Mike Pettigrew, Guru Rao, William Shen, and Lisa Spainhower for their inputs in preparing this paper.

References

1. *MVS/ESA Planning: Sysplex Management*, Order No. GC28-1620, March 1991; available through IBM branch offices.
2. Nandakumar N. Tendolkar and Robert L. Swann, "Automated Diagnostic Methodology for the IBM 3081 Processor Complex," *IBM J. Res. Develop.* **26**, 78–88 (January 1982).
3. 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 (January 1982).
4. F. F. Sellers, Jr., M. Y. Hsiao, and L. W. Bearnson, *Error Detection Logic for Digital Computers*, McGraw-Hill Book Co., Inc., New York, 1968.
5. C. L. Chen and M. Y. Hsiao, "Error-Correcting Codes for Semiconductor Memory Applications: A State-of-the-Art Review," *IBM J. Res. Develop.* **28**, 124–134 (March 1984).
6. C. L. Chen and R. A. Rutledge, "Fault-Tolerant Memory Simulator," *IBM J. Res. Develop.* **28**, 184–195 (March 1984).
7. C. L. Chen, "Double Error Correction–Triple Error Detection Code," U.S. Patent 4,509,172, April 2, 1985.
8. W. W. Peterson and E. J. Weldon, Jr., *Error-Correcting Codes*, Second Ed., MIT Press, Cambridge, MA, 1972.
9. D. C. Bossen, L. C. Chang, and C. L. Chen, "Measurement and Generation of Error Correcting Codes for Package Failures," *IEEE Trans. Computers* **C-27**, 203–207 (March 1978).
10. C. L. Chen, "Error-Correcting Codes with Byte Error Detection Capability," *IEEE Trans. Computers* **C-32**, 615–621 (July 1983).
11. T. R. N. Rao and E. Fujiwara, *Error-Control Coding for Computer Systems*, Prentice Hall, Inc., Englewood Cliffs, NJ, 1989.
12. U. Olderdissen and H. Schumacher, "Decoding of BCII Double Error Correction–Triple Error Detection Codes," U.S. Patent 4,556,977, December 3, 1985.
13. *IBM ES/3090 Processor Complex Processor Resource/Systems Manager Planning Guide*, Order No. GA22-7123, September 1991; available through IBM branch offices.
14. *Introducing Enterprise Systems Connection*, Order No. GA23-0383, September 1991; available through IBM branch offices.
15. K. R. Covi, "Three-Loop Feedback Control of Fault-Tolerant Power Supplies in IBM Enterprise System/9000 Processors," *IBM J. Res. Develop.* **36**, 781–789 (1992, this issue).

Received January 9, 1992; accepted for publication February 26, 1992

[2] A point system is a system that operates at the customer's operations command center and is used to control other systems within the customer's enterprise. The type of control could be local or remote.

C. L. (Jim) Chen *IBM Enterprise Systems, P.O. Box 950, Poughkeepsie, New York 12602 (CLCHEN at TDCSYS3).* Dr. Chen is a Senior Technical Staff Member in the IBM Mid-Hudson Valley Development Laboratory. He received a B.S. degree from National Taiwan University and a Ph.D. degree in electrical engineering from the University of Hawaii. Prior to joining IBM, he was a faculty member at the University of Illinois at Urbana–Champaign. He has worked in the areas of error-correcting codes and computer reliability, and has received six IBM Invention Achievement Awards, four IBM Outstanding Innovation Awards, and an IBM Corporate Award. Dr. Chen is a Fellow of the Institute of Electrical and Electronics Engineers, and a member of the IBM Academy of Technology.

Nandakumar N. Tendolkar *IBM Enterprise Systems, P.O. Box 950, Poughkeepsie, New York 12602 (TENDOL at TDCSYS3).* Dr. Tendolkar is a Senior Engineer in the Systems Technology Noise Laboratory. He is currently involved in the design and evaluation of large-system error detection, FRU isolation, and recovery support hardware and microcode. He joined IBM in 1967, and began work with his current group in 1980. Prior to that, he worked on developing and implementing the diagnostic strategy for the System 308X and follow-on machines. He has also worked in the areas of industrial engineering and planning and system performance evaluation. Dr. Tendolkar received the M.S. degree in operations research from Cornell University and the Ph.D. degree in computer and information science from Syracuse University. He has received an IBM Outstanding Innovation Award for his work on 308X diagnostics, and has two patent applications on file. Dr. Tendolkar is a Senior Member of the IEEE. He is also an Adjunct Professor of Computer Science at Marist College, Poughkeepsie, New York.

Arthur J. Sutton *IBM Enterprise Systems, P.O. Box 950, Poughkeepsie, New York 12602 (retired).* Mr. Sutton received his B.S. degree from St. Lawrence University in 1951 and his master's degree in mathematics from Columbia University in 1956. He joined IBM in 1956 at the Poughkeepsie Development Laboratory, where he was involved in the system design of system RAS functions, specifically system recovery design, and also design of Processor Controller system functions such as the error handler and reconfiguration for 308X, 3090, and ES/9000 systems. He is now retired from IBM. Mr. Sutton has received an IBM Eighth-Level Invention Achievement Award. He is a Senior Member of the Institute of Electrical and Electronics Engineers and a member of the Computer Society.

M. Y. (Ben) Hsiao *IBM Enterprise Systems, P.O. Box 950, Poughkeepsie, New York 12602 (HSIAO at TDCSYS3).* Dr. Hsiao is an IBM Fellow and Functional Manager of the Systems Technology Noise Laboratory. His current professional interests include research and development in computer reliability, availability, serviceability, error-correcting codes, error detection, failure-isolation techniques, and system noise/SER analysis. He joined IBM in Poughkeepsie in the Advanced Reliability Technology Department in 1960. From 1965 to 1967, he was on educational leave to the University of Florida, after which he returned to IBM as Advisory Engineer in the Reliability and Diagnostic Engineering Department. In 1969 he was promoted to Senior Engineer and Manager of the Reliability Technology Department. He assumed his present position in 1984. Dr. Hsiao received his B.S. in electrical engineering in 1956 from Taiwan University, Taipei, his M.S. in mathematics in 1960 from the University of Illinois, and his Ph.D. in electrical engineering in 1967 from the University of Florida. He has eight IBM Invention Achievement Awards, three IBM Outstanding Innovation Awards, and an IBM Corporate Award in the areas of error-correction codes, error detection, and failure-isolation techniques. He has authored and co-authored two books published in 1964 and 1968. Dr. Hsiao is a Fellow of the Institute of Electrical and Electronics Engineers and a member of the Fault-Tolerant Computing Committee and IFIPS Committee on Reliable Computing and Fault Tolerance.

Douglas C. Bossen *IBM Enterprise Systems, P.O. Box 950, Poughkeepsie, New York 12602 (DBOSSEN at TDCSYS2).* Dr. Bossen is a Senior Technical Staff Member in the IBM Mid-Hudson Valley Development Laboratory, where he joined IBM in 1968. He is a member of the Noise Detection and Recovery Design Department, which has general responsibility for advanced reliability techniques, including error-correcting codes, error-detection mechanisms, fault tolerance, fault-isolation techniques, and reliability modeling. Formerly he was manager of Advanced RAS Design Technology. Dr. Bossen has lectured extensively in IBM on ED/FI hardware design. His work on the application of error-correcting codes to computers includes the invention of the b-adjacent codes which are used in many IBM products as well as in the industry for DASD and semiconductor storage error-correction systems. He received the B.S., M.S., and Ph.D. degrees in electrical engineering, all from Northwestern University. Since joining IBM Dr. Bossen has received an IBM Sixth-Level Invention Achievement Award; he has also received two IBM Outstanding Innovation Awards and an IBM Corporate Award for his work in error detection and fault isolation. He was elected in 1990 to membership in the IBM Academy of Technology. Dr. Bossen has 13 issued U.S. patents, four pending, and 27 invention publications; he has published 17 papers. He is a Fellow of the Institute of Electrical and Electronics Engineers, and a member of Sigma Xi, Tau Beta Pi, and Eta Kappa Nu. In 1973 he received honorable mention by Eta Kappa Nu as an Outstanding Young Electrical Engineer.