



IBM's ES/9000 Model 982's Fault-Tolerant Design for Consolidation

Consolidated work loads running around the clock means that today's large, general-purpose computers must meet high availability demands. To meet these demands, the Model 982 provides fault tolerance by combining enhanced circuit-level error detection and failure isolation techniques with system-level techniques exploiting inherent redundancy.

Lisa Spainhower

Thomas A. Gregg

*IBM Large Scale
Computing Division*

Ram Chillarege

*IBM T.J. Watson
Research Center*

Recent trends in high-end general-purpose computing point to a need to consolidate commercial information-processing applications and databases. Although this need appears to run contrary to the continuing growth of distributed computing, the commercial world has compelling reasons for consolidating data centers and ultimately computers. These reasons include acquisitions and takeovers, elimination of data centers, and staff reductions. Consolidation reduces costs and simplifies and improves system management and data security. At the same time, however, it increases demands on the remaining data center computing equipment. Thus, the data center computer needs to have very high performance and capacity and fault tolerance that provides close to 100 percent availability.

Computing has become a multiple-time-zone operation or even a global operation. With consolidated computing, an organization may have work loads running in prime shift at some part of the globe at any time. A related trend is the elimination of off-prime or nonprime shift operations—applications such as credit card authorization, automated tellers, and point-of-sale terminals must be on line all the time. As a result, a computing system can never take a scheduled outage. Consolidation, coupled with the nature of today's applications, places stringent availability requirements on high-end computers.

Another result of consolidation is the simultaneous existence of unrelated work loads and

multiple operating systems on the same computer. When applications are added because of acquisitions or site reductions, one larger computer can replace several of lesser capacity without a need to change the applications. The logical partition for each operating system is customized for the capacity of the computer it replaces. Since one computer supports all the partitions, each of which could have a different critical operations window, the overall demand for hardware availability is greater, spanning all the windows. To satisfy this demand, the fault tolerance design objective for IBM's ES/9000 Model 982 computer is to keep running without outage or intervention either at the time of failure or during repair.

Design philosophy

Our general-purpose computer architecture has evolved since 1964 from the System/360, through the System/370 and the System/370 Extended Architecture (XA), to today's System/390 Enterprise Systems Architecture. The current hardware implementation is the ES/9000, which includes three machine types: the 9221, 9121, and 9021, from least to highest performance. The high-end 9021 is available in models with from one to eight CPUs. The most powerful is the eight-CPU Model 982.

Fault tolerance is implemented at two distinct levels in the 982. At the circuit level, a set of objectives and guidelines for consistent error checking and recovery ensures fault tolerance

throughout the computer, regardless of each subsystem's function. At the system level, on the other hand, subsystem functions determine fault tolerance implementation. From a wide range of fault tolerance techniques, the designer selects the best for each subsystem function. Both the circuit and system levels must meet overall product requirements, not just for fault tolerance but also for cost and performance.

Circuit-level issues. A computer's error detection and recovery strategy must handle the most frequent circuit fault models. Logic errors can occur due to hard, intermittent, and transient circuit faults. The hard, or permanent, fault occurs when a digital circuit no longer yields a correct output, given a specific set of inputs. Any time the specific input is repeated, the circuit produces the incorrect output. An intermittent fault occurs when a specific event produces an incorrect result, but the same inputs at a different time may produce the correct result. An intermittent fault can occur as a result of a design error or marginal circuits. Transients occur when environmental conditions, noise, or cosmic particles cause an incorrect result, but the circuit itself functions correctly.

The expectation for the chip technologies used in the 982 is that most faults are transient, so the design focus is to retry operations and recover at the hardware level whenever possible. Once the design has been debugged and the computer installed in a stable environment, intermittent failures are extremely rare. The computer is not specifically designed to handle intermittents. For infrequent intermittents, instructions are retried and recovered; if such errors occur frequently, they exceed thresholds and are considered permanent faults.

The IBM S/360, S/370, and S/390 computers have been known for their data integrity. Traditionally, when an error occurred, these computers checked all computation, retried the operation, and if it was unrecoverable, the computer stopped and perhaps a higher level of fault tolerance (such as software) handled the situation. Consequently, very extensive concurrent error detection was designed into the computers.

To make the 982 fault tolerant, its designers increased the traditional concurrent error detection from the 90 percent range to 100 percent. An error is usually identified and recovery attempted within the same machine cycle. Reporting and recording may take additional cycles. A code protects every latch, and there are no naked latches. All dataflows, arrays, and control buses include parity or ECC, and state machines use techniques such as parity predict or duplication. Expanding to 100 percent coverage added less than 3 percent to the number of logic chips. This increase is so small mainly because many of the hard-to-check logic chips, such as sequence logic, are very limited in pins, and thus the logic can be completely checked without adding chips.

In addition to concurrent error detection of all logic, on-line error correction and repair capability requires exact iden-

tification of the replaceable part that causes an error. For recovery, the computer also should identify the source of the erroneous data. As an example, the design ground rules require that error checkers be placed at the driver of any signals leaving a part and immediately after the receiver of any signals entering a part. This allows most failures to be identified to the correct part without further analysis. On-line failure isolation determines the part to be replaced.¹ The system must be able to determine whether an error is due to a permanent physical failure requiring repair or only a transient fault that can be handled without physical part replacement. In a 982, it is not always easy to distinguish which type of fault has occurred, but the design handles all of them.

Distinguishing faults requires the capability to back out and retry internal operations with appropriate thresholds for determining success. Since retrying an operation can involve different paths through the logic than the original error, the system of thresholds is very fine grained and rather complex. For example, if a fault occurs on a directory entry in a cache, the retry of the operation may use a different cache set. Therefore, the threshold must be on the use of the directory address and set, not on the actual operation being performed. This has led to a very extensive threshold implementation.

Failure due to a hard circuit fault might be recoverable through instruction retry since the machine is run nonoverlapped during the retry, thereby using different circuit combinations. Intermittent faults, on the other hand, could cause errors several times and go away later. To deal with either type of fault, multiple retries with appropriate thresholds determine whether a unit needs to be taken out for repair or whether the computer can continue with the unit containing the failure. Chen et al. give a detailed description of the 982's error detection techniques.²

System-level issues. The 982 has three major functional subsystems: the CPUs, the channel subsystem, and the storage hierarchy. It also has a fourth subsystem consisting of the support functions: the processor controller element (PCE), power, and cooling. Together, these four subsystems are called the central processing complex (CPC).

For performance, the CPC has several identical elements in parallel, such as processors and channels. These inherently redundant resources provide a fundamental fault tolerance capability. Each element is designed at the circuit level to have concurrent fault detection, recovery by retry, and fault identification and isolation. The strategy for fault tolerance is to couple the error detection and failure isolation capabilities with system-level techniques exploiting the inherent redundancy in the CPC. As a result, the 982 can identify errors and recover transient and intermittent failures by retry.

The storage hierarchy passes back detected errors to the CPU or channel. Returning these "passed errors" to the

continued on p. 52

continued from p. 49

requester limits the scope of transient faults to a single operation and allows unaffected storage operations to continue. The computer can tolerate many permanent faults, particularly those in array chips, by reconfiguring the array to prevent use of the failed location. For permanent faults requiring repair, the CPC isolates the failing element, reconfigures the system around it, and recovers by rolling back to an error-free consistent state, all without stopping the work load. The design provides the capability to fence off all external interfaces to an element so that reconfiguration and maintenance can be performed while the computer continues in operation. A service technician can replace the failed hardware without affecting the rest of the CPC.

It is crucial that the PCE and the rest of the support do not impinge on the availability of the functional subsystems. Because the support subsystem does not have the performance constraints of the functional subsystems, it can use more traditional fault tolerance techniques such as explicit redundancy—that is, duplication strictly for reliability. The storage hierarchy also employs well-documented methods for data protection; the multiplicity of methods and the resulting design robustness exceed most implementations. For these reasons, we describe the support subsystem and the storage hierarchy only briefly in the following sections. We describe the CPUs and channel subsystem in more detail because they best illustrate the system-level design direction.

The central processor

All S/390 CPCs with more than one CPU can dynamically vary any CPU between on-line and off-line status, and, as long as one CPU is on line, continue instruction execution. Except for special operations (vector facility and integrated cryptographic facility) for which hardware may not be provided on all CPUs, each CPU in a multiprocessing environment can execute any task dispatched by the operating system. Where a particular task will be executed is unpredictable.

The 982 multiprocessing environment is usually one eight-way multiprocessor but may be two four-way multiprocessors, also known as a physically partitioned multiprocessor. Any S/390 multiprocessor can have one native operating system or up to 10 logical partitions. Each logical partition has an independent operating system with some number of logical CPUs assigned to it. The number of CPUs per partition cannot exceed those physically available. Logical CPUs can be dedicated or shared. Any dedicated CPUs can be assigned to only one partition, thus reducing the maximum available to coexisting partitions. For redundancy, a partition is shared or has at least two dedicated CPUs.

A major design goal is to exploit this capability, allowing the computer to continue uninterrupted operation if a CPU becomes unavailable. Accomplishing this goal requires the following procedure:

1. The processing state in the failed CPU must roll back to a consistent, error-free state.
2. The state of the storage system as seen by all other CPUs must be architecturally accurate and free of errors.
3. The PCE must ascertain the failed CPU's state (transient failure or permanent failure).
4. The PCE must remove the failed CPU from the configuration before it propagates any corrupted data.
5. The operating system must transfer the work that was running on the failed CPU to an operational processor.
6. The failed CPU must be repaired and returned to service without a system interruption.

An example involving instruction retry and recovery illustrates the implementation of this procedure. Figure 1 shows the pipeline stages and four issued instructions. This machine implements out-of-sequence execution and exploits this performance technique to boost fault tolerance at the circuit level. Out-of-sequence execution allows multiple instructions to be issued without requiring that they finish in the order of issuance as long as they complete in the same order. There is a fine distinction between *finish* and *complete*. *finish* means reaching the end of the instruction's execution; *complete* means storing the instruction's results. Essentially, once an instruction completes, it cannot be backed out because its results are posted in storage and architecturally the instruction has executed. Prior to completing, an instruction can be backed out, provided all pending stores from instructions issued after it are also canceled and the storage is left in a consistent state.

Assume a failure occurs at time T2 as shown in Figure 1. The clocks are immediately frozen at time T2. Note that the last completed instruction was instruction 1, which completed at time T1. Although instructions 3 and 4 have both finished (their results have been calculated and put into temporary facilities awaiting completion), they cannot be flagged as completed since instruction 2 has not completed. Since the machine maintains instruction execution results in temporary facilities until it determines the instruction is complete, it can back out of all intermediate pipeline operations (including the finished instructions) by flushing the pipeline and temporary facilities. This leaves the executing program in an error-free processing state—that is, at a point corresponding to the completion of instruction 1. Assuming a failure in the CPU hardware, the physically separate PCE examines the pipeline contents, architectural facilities, and temporary facilities, and puts them back into a consistent state. The PCE obtains the CPU state by scanning out the internal facilities, using totally separate scan clocks so that the logic clocks can remain stopped.

Next, the PCE must ensure that the storage system is consistent. The storage hierarchy design facilitates this step. An advantage of its store-through level 1 data cache is that it

isolates internal CPU faults. Because the shared level 2 cache contains any critical system data resulting from CPU instruction execution, the level 1 cache copy is not critical and can be discarded upon detection of an error.

Instruction retry determines whether the failure detected was a transient error or a true permanent circuit failure. The CPU register manager,³ which controls out-of-sequence instruction execution, provides an audit trail of changes effected by incomplete instructions; the audit trail is used for checkpoint restart in instruction retry. To perform retry, the PCE sets the CPU's internal logic state to the state it would be in at the completion of instruction 1. The PCE then steps the logic clocks to see if the following instructions execute successfully with no errors. If they do, a transient error exists, and instruction execution recommences. During retry and for some time after, instruction execution is nonoverlapped. Each instruction must complete before the next sequential instruction is decoded. If retry is successful, normal overlapped instruction execution will start.

If retry is unsuccessful after exceeding a retry threshold, the failure is permanent. The computer can tolerate many permanent faults, especially those in large arrays. Examples are the level 1 caches, their directories, a cache of previously translated virtual addresses called the directory look-aside table, and a cache of previously taken branch addresses called the branch history table. In these arrays, individual addressable locations and/or groups of locations can be deconfigured when a permanent fault is identified. For most other permanent CPU failures, the work load must be moved. When one of these occurs, the PCE puts the failed processor into the CPU check-stopped state.

CPU check-stop⁴ indicates severe CPU damage and terminates processing on that CPU. In a uniprocessor, CPU check-stop is the equivalent of system check-stop, and all instruction processing ceases. In older models, when there was more than one CPU, the operating system would initiate alternate CPU recovery, which logically removes the failing CPU from the system.⁵ The 982 uses the processor availability facility (PAF), which physically isolates the failing CPU for repair while the work in progress is completely

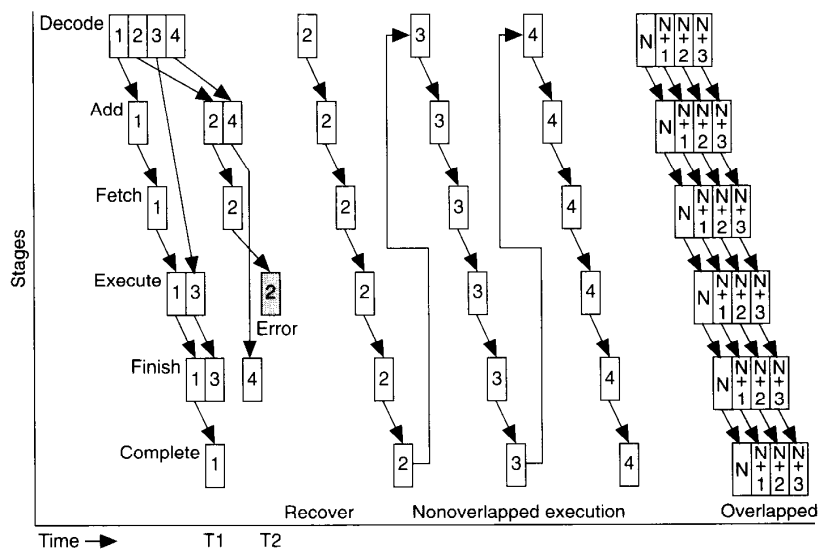


Figure 1. Out-of-sequence instruction recovery (N = instruction at which overlapped execution resumes).

recovered and run on the remaining CPUs.

Processor availability facility. PAF presents the check-stopped condition, the error status details, and the complete status of the process in execution to the operating system. Upon examination of the error status, the operating system stores the task in the normal dispatch queue to allow resumption of execution on another processor according to normal dispatch priorities.⁶

In the past, when retry was not successful, alternate CPU recovery would abnormally end (ABEND) the task and initiate software recovery. Depending on the robustness of the recovery and the criticality of the ABENDED task, the work load would or would not continue. For general control program tasks where recovery is quite robust, recovery was successful. If the task was an application, it probably would not recover, but its termination generally would not stop the work load. However, for many subsystem tasks and other critical component tasks, it was common for the work load to stop.

With the PAF operation, the task is not ABENDED, and no other software recovery mechanisms need be invoked. PAF provides on-line reconfiguration, which, combined with concurrent error detection and fault identification, makes the 982 a fault-tolerant processor.⁷ During and after the PAF operation, $N-1$ CPUs continue to execute instructions normally, no tasks are lost, and no intervention is required. The normal dispatching priority ensures optimum use of the sys-

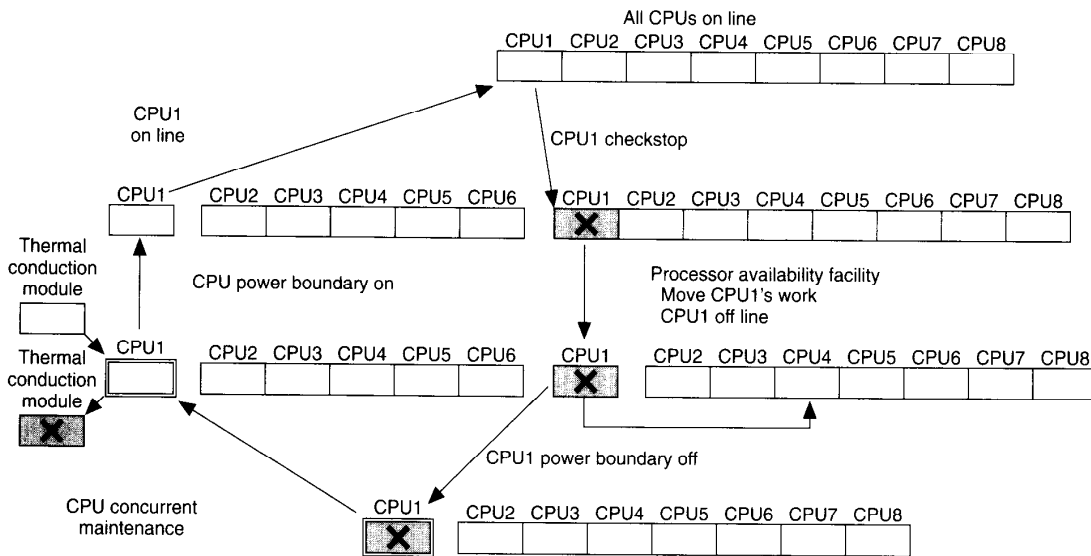


Figure 2. Concurrent CPU repair.

tem while it is running with one less CPU.

Nondisruptive repair. One of our objectives was to provide concurrent maintenance for redundant subsystems, whether inherent or explicit. Figure 2 shows the procedure for concurrent repair of a CPU. The failed CPU must be both logically and physically isolated from the operational components of the CPC. Physical isolation is provided by a power boundary. Each CPU has its own independent power supplies, which are not shared with any other hardware and which can be powered up and down independently of the power status of other power boundaries. Logical isolation is provided by fencing, an internal mechanism by which any physically connected, on-line logical unit disables the receipt of signals from a CPU. The hardware automatically invokes fencing.

The PCE also automatically invokes an auto-call for repair by a customer service engineer. At a time convenient to the customer, the engineer performs on-line maintenance: powering down the CPU boundary, replacing the failed part, powering the boundary back on, and putting the CPU back into the configuration. Again, throughout this repair process, $N-1$ CPUs continue uninterrupted instruction processing, and upon its completion, all CPUs are on line, executing tasks.

The channel subsystem

This subsystem connects the 982 to the I/O control units, which in turn connect to the I/O devices: disk and tape drives, terminals (usually personal computers), and printers.

Because of the large number of channels, each of which can communicate with many devices concurrently, the channel subsystem allows recovery actions affecting as few I/O operations as possible.

When a program or operating system communicates with a device, it first builds a channel program listing I/O commands (read, write, locate, and so on), central storage addresses for data, and several flags controlling execution of the channel program.⁴ Next, the CPU executes an instruction that starts the channel program by placing it on the I/O work queue. At this point, execution of the I/O operation passes to the channel subsystem and the CPU is no longer involved; it is free to do other work.

One of the I/O processors (IOPs) in the channel subsystem examines the work queue and finds an I/O operation to perform. This processor selects a path and passes the operation to the selected channel. The channel then fetches and executes the channel program.

Some operations to I/O devices take a considerable amount of time without requiring data. For example, a seek operation to a disk may take several milliseconds. To better utilize the channel path during these long operations, certain bits in the channel program instruct the channel to allow the I/O control unit to disconnect from the channel interface, allowing other operations to be initiated to other I/O control units and devices. As a result, I/O operations are multiplexed within a channel. When the I/O device finishes the opera-

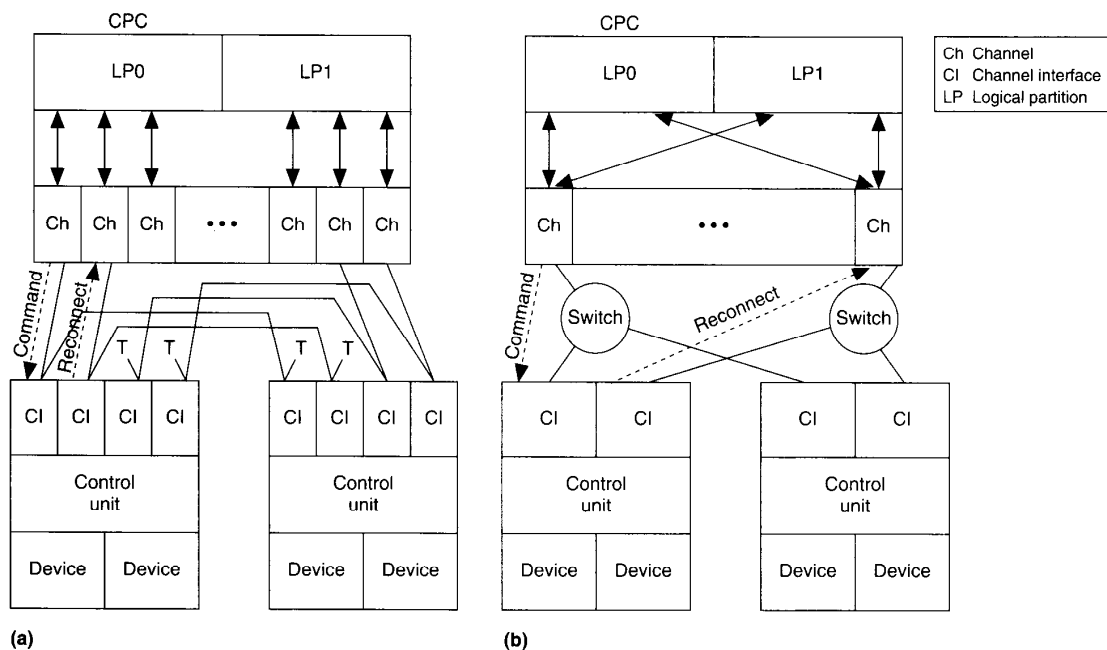


Figure 3. I/O configuration: parallel I/O interface (a); serial ESCON interface (b).

tion, it interrupts the channel subsystem over one of the available channel interfaces. This multiplexing over a channel interface requires that the channels themselves execute many channel programs concurrently. If the channel detects an error when many channel programs are active, the recovery algorithm usually affects only the single channel program actively communicating over the I/O interface.

Figure 3 illustrates how S/390s are attached to I/O control units. Figure 3a shows two control units attached to a system by parallel I/O cables; Figure 3b shows the same connection using the serial Enterprise Systems connection (ESCON) cables through two switches. Each information route from an operating system to a control unit and ultimately to an I/O device is called a path. In both examples, the CPC includes two operating systems running on two logical partitions.

The parallel I/O interface attaches to the I/O control units using a multidrop bus configuration.⁸ Each of the 35 conductors (two 9-bit unidirectional data buses and 16 control lines) connects the channel to a number of I/O control units. The conductors actually pass through the I/O control units until they reach the last I/O control unit, where there is a terminating resistor (indicated by a "T" in the figure). The serial ESCON interface attaches to the I/O control units by means of pairs of point-to-point optical cables, one cable in

each direction.^{9,10} The switches, called ESCON directors, provide improved connectivity to the I/O control units.¹¹

Dynamic path selection. Multiple paths to the I/O control units improve system performance. Before the 370/XA architecture was developed, the program or operating system was responsible for selecting the path to the I/O device. The 370/XA architecture further improved performance by moving the path selection function into the channel subsystem. Now, when an operation encounters a busy condition, the channel subsystem can choose another route to the device. The channel subsystem controls the status of available physical connections, the choice of physical connection, and the routing of operations.

Figure 3 shows that an I/O command initiated along one physical path (indicated by a "Command" in the figure) can complete along a different path ("Reconnect") chosen by the I/O control unit—a capability known as dynamic reconnection. The program or operating system is aware only of the I/O devices; the channel subsystem and the I/O control unit recognize the physical connections between the system and the I/O device.

Multiple paths to the I/O control units also provide redundancy and have long been a requirement of high-end general-purpose processors. Because redundancy is inherent,

the channel subsystem can exploit it transparently to the operating system. Just as the channel subsystem can get around busy conditions, it can also get around permanent failures in a single channel or channel path, rerouting if a channel is no longer available.

The channel subsystem records status information about work queues and data in central storage. A channel that has a permanent failure between active I/O operations can be removed from the configuration without affecting any other in-process I/O operations. Dynamic reconnection, the ability of an I/O device to continue an I/O operation with an attached channel other than the one that initiated the operation, also provides fault tolerance of permanent failures in a single channel. If a channel experiences a permanent failure while actively communicating with an I/O device, the channel subsystem may continue the operation on another channel or it may request recovery of the operation from the operating system.

Channel subsystem recovery. In the simplest fault-tolerant design, errors that occur in the channel subsystem would be recovered by the program or operating system. The channel would simply indicate the error to the program or operating system, which would then carry out the appropriate recovery action. Sometimes this recovery action would be to repeat the operation. Other recovery actions might be more complicated, involving a sense operation to the I/O device to determine the state before retrying the operation.

Unfortunately, some older I/O devices and the programs that drive them cannot recover from transient errors. An example is a punched-card reader, which cannot back up and reread a card. When the reader presents an error to the program, the application is terminated, and human intervention is required to get the application going again. A similar situation exists when errors cause a loss of knowledge of the tape position in older tape drives. To handle such less-than-perfect recovery by the program, a robust recovery scheme in the channel subsystem keeps many of the errors from reaching the program. Newer I/O control units and devices, especially disk drives, are much more recoverable by software.

Since not all devices are 100 percent recoverable by software, the channel subsystem should recover from as many transient errors as is practical. Therefore, in the 982, a channel can recover from transient errors within itself and from errors on the I/O interface. The IOPs and channel buffers can also recover from transient errors; however, the recovery of these elements sometimes affects multiple channels. Another goal of channel subsystem recovery design is to limit the scope of the recovery actions so that they affect as few channel programs and I/O devices as possible. The 982's design limits the scope of recovery by defining three types of recovery actions in the channels and other, more drastic recovery actions for the IOPs and channel buffer. Some of

these actions involve handling passed errors. If a channel or an IOP receives an error return code from a central storage operation running on its behalf, the central storage operation keeps running and the error is passed to and recovered by the channel or IOP.

The first, most limited type of channel recovery action handles errors on the I/O interface. On the parallel interface, these errors are most commonly parity errors on the data bus. On the ESCON interface, these errors are due to bit errors on the serial link, and may be much more frequent than errors on the parallel interface. This recovery action affects only the device currently communicating on the I/O interface. If the channel detects an error, it can signal the device to attempt a command retry. If the device detects an error, it can request the channel to retry the operation. If the recovery action is successful, the channel program is not interrupted and recovery is transparent.

The second type of channel recovery deals with either internal or passed errors, which can be recovered by the channel itself without intervention by another element such as the PCE. In this case, the channel determines that there has been no damage to information describing the status of the multiple I/O operations. At most, the channel may have to terminate operation of the device logically connected to the channel at the time of the error. If an I/O operation is terminated, the channel program will see an error condition and perform the recovery operation.

The third type of recovery action also handles either internal or passed errors, but these errors damage critical information in the channel. In this case, all I/O operations terminate and the channel restarts. All the channel programs using the channel receive an error indication, and they all perform their recovery actions.

Recovery of the IOPs depends on the integrity of the channel subsystem control blocks in main storage. At most, if an IOP has an error while it is locking a control block, the other IOP or the PCE can determine which control block is locked and perform the appropriate recovery action for the affected I/O operation. If an IOP has a permanent failure, another IOP can take over the work without a loss of channels. When the channel buffer has an error, recovery involves purging all storage requests from the channels and IOPs and purging all communication buffers between the IOPs and the channels. After the purge, all the IOPs and channels are sent a signal indicating the purged condition, and it is then up to each element to perform its own recovery. This purging and notification of lost storage requests and communications is another type of passed error unique to the channel subsystem.

ESCON configuration advantages. The ESCON Multiple Image Facility (EMIF) allows channels to be shared by multiple logical partitions. Returning to Figure 3, each parallel channel is dedicated to a particular partition. The serial ESCON channels use an additional interface protocol that

allows individual channels to be shared by multiple logical partitions. This means that fewer channels are needed to provide all the necessary connections between the CPC and I/O control units. As a result, the configuration uses fewer I/O cables while maintaining redundant paths for performance and fault tolerance.

ESCON supports dynamic reconfiguration management, the ability to change the I/O configuration and describe these changes to the operating system while the system is running. I/O control units, devices, and channel paths can be added, deleted, or modified. The ability to add and delete channel paths adds more concurrent repair opportunities. For example, one can repair a trunk cable by supplying alternate paths and then removing the damaged trunk cable.

Hot pluggable channels. The 982 design allows removal and replacement of individual channel cards concurrently with continued operation of the rest of the CPC. Each channel has a dedicated interface to the rest of the channel subsystem logic; no signals are shared with any other channel. These unique interfaces allow channels to be individually isolated (or fenced) from the rest of the system. On the other hand, up to 128 channels are on the same power boundary. This presents a challenge for the power supply isolation of a single card.

Our implementation involves multiple-length card-to-board pins, which allow a card to be inserted and removed from an active board without introducing electrical noise in the remaining cards on the same board. The channels on those cards continue to execute I/O operations without interruption during the removal or replacement of the permanently failed channel card.

Figure 4 shows how the multiple-length pins gradually charge the load when a card is inserted into the board. As the card is inserted, the longest pins (numbered 1) make contact first. These pins are the ground connection. The next pins to make contact (numbered 2) supply power to the drain of an FET (field-effect transistor). The third-longest pins (numbered 3) supply the gate voltage to the FET through an RC (resistor-capacitor) network. At this time, the power supply voltage on the card is being slowly increased, preventing a large inrush of current to charge the high-capacitance load. When the card is fully inserted, the shortest pins (numbered 4) are contacted. These pins connect the board power directly to the card load. This pin length is also used for all card-to-board signal connections.

Detection of a permanent channel failure initiates an auto-dial for concurrent channel maintenance. Automatic maintenance procedures assure that the channel or channels on that card are taken off line. LEDs on each ESCON card allow each channel to indicate its on-line/off-line status. Every card, ESCON or parallel, also has an on-line maintenance LED, activated when the customer service engineer initiates concurrent channel maintenance. This positive visual indication

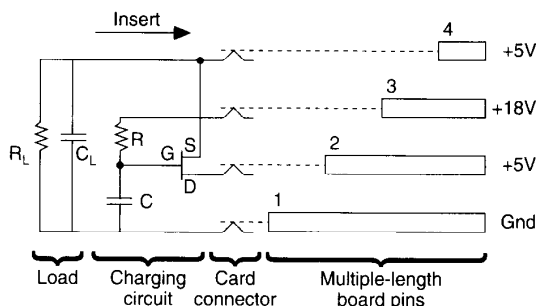


Figure 4. Hot plug charging.

assures that the correct card is replaced. By providing physical and logical isolation of individual card-on-board channels, the 982 allows channels to be repaired concurrently.

Storage hierarchy

The 982 has a four-level storage hierarchy, and fault tolerance is extensive at all levels. Level 1 is the high-speed cache of each CPU. It is divided into a 128-Kbyte data cache (level 1D) and a 128-Kbyte instruction cache (level 1I). The basic data element of both caches is a 128-byte line, and both caches are four-way set associative. Address mapping allows each line of data to be stored in any one of four sets or columns but always in a specific one of 256 rows. Level 1D is store-through cache to the next hierarchical level, the level 2 cache. Data changed by the CPU in level 1D is also held in an ECC (error correction code) protected buffer until its successful receipt by level 2.

Level 1 has parity for error detection, refetch for transient error recovery, and line delete, set delete, and relocate for hard error recovery. Line delete reduces to three the number of columns available in a particular row after a hard failure. Set delete reduces all rows to three columns. Beyond certain column thresholds, line delete is superseded by relocate, which uses built-in spare rows on the array chips to circumvent hard failures. See Spainhower et al.¹² for details of level 1 fault tolerance.

Level 2 is a store-in cache to the next hierarchical level, central storage or level 3. Level 2 has a current copy of all level 1 data. In the 982, there are two 4-Mbyte level 2s, each having data affinity with its local level 3. Level 2 has a 64-data and 8-syndrome bit, single-error correct, double-error detect ECC for checking and recovery from single-bit transient and permanent errors. For multibit permanent error recovery, level 2, which like level 1 is four-way set associative, has line and set deletes.

Central storage, also known as level 3, has a 2-Gbyte total capacity. It is protected by a 64-data and 8-syndrome bit, sin-

gle-error correct, double-error detect ECC for single-bit errors, transient or hard. Each bit of the ECC word comes from a different array chip, so all single array chip failures are correctable. Combinations of one hard and one transient or two hard failures in one ECC word are corrected by an algorithm using an exclusive-OR of complemented and recomplemented copies of the word. Spare memory chips are dynamically activated when certain single-chip thresholds that increase the likelihood of multibit alignment are exceeded. When multibit hard errors are detected, the CPC notifies the operating system to stop using the storage location. Spainhower et al.¹¹ includes a description of central storage fault tolerance.

Expanded storage, or level 4, is a high-speed electronic storage for frequently used data that would otherwise be stored on magnetic devices. Level 4 data is transferred to level 3 on 4-Kbyte-page boundaries. Level 4 is protected by a 128-data and 16-syndrome bit, double-error correct, triple-error detect ECC. Like level 3, it uses an exclusive-OR algorithm to correct beyond the power of its code. Also like level 3, level 4 has spare array chips.

Throughout the storage hierarchy, addresses, storage protection data, and other control information is protected by ECC, duplication, set deletes, line deletes, relocates, or some combination of these. All major data paths within storage control have ECC.

The support subsystem

The power and cooling subsystems make wide use of explicit redundancy. In the case of power supplies, fans, and blowers, all the redundant elements are normally active. The load is split across one more element—power supply, fan, or blower—than is necessary to meet physical requirements. An advantage is that the elements normally operate in a derated, less-than-full-capacity mode. This improves the reliability of many component parts. Also, regardless of which element fails, those necessary to provide uninterrupted power or cooling to functional elements are already configured and on line, eliminating any delay due to start-up or switch-over. The remaining elements simply increase their contribution to meet the needs of the load. A third advantage is that all the elements are known to be operational without latent fault diagnostics or procedures.

The power supplies, fans, and blowers can be repaired concurrently with continued operation of the 982 with no performance degradation or service outage. Visual indicators and interactive guided maintenance procedures assist service personnel in identifying elements to be replaced. For extra protection, each type of power supply has a unique connection configuration to prevent misplugging.¹³

Pumps for water cooling are also explicitly redundant but use an idle-standby scheme. To avoid situations where the active pump fails, only to switch to a backup that has a latent

fault, the PCE periodically switches from the functional active pump to the idle pump. If the idle pump is not working, the PCE makes another switch-over and places a call for service. If the idle pump is functional, it changes roles with the active pump until the next scheduled switch-over.

The PCE has active and backup sides. Each side is an identical, complete PCE system with processor, memory, and disk. The backup scheme is active standby. The active side of the PCE performs all needed service and system functions for the 982. The backup side constantly runs diagnostics. The two sides are in regular communication via "heartbeat monitoring." If the backup detects itself in error, it places a service call while the active remains in control. If the active side detects itself in error, or if the backup does not receive heartbeat signals from the active, a switch-over takes place. The backup becomes the active side, and places a service call. When a side is being repaired, it is placed in a third mode, service mode. Service mode is also used to apply updates to the microcode (formally called licensed internal code).

The PCEs, IOPs, channels, and CPUs all have microcode. Service personnel usually can change the CPC's current active microcode level without any downtime. Microcode changes can be downloaded via TP link to the PCE disk at any time. The PCE is placed in service mode, and the backup side disk merges the changes with the current code to form the new microcode level. The new level is then written to the active side and the appropriate control storages within the CPC. The backup side is taken out of service mode and returned to backup mode. The one exception to this is a physically partitioned multiprocessor—for example, one 982 operating as two independent four-way CPCs. In that case, each side of the PCE is normally active and dedicated to one of the multiprocessor sides.

ON A SOLID FOUNDATION of concurrent error detection, Model 982 uses many different techniques to achieve fault tolerance. Subsystem functions mainly determine the techniques used. The support subsystem uses traditional explicit redundancy for the PCE, power supplies, water cooling, and air cooling. The implementation for each of these is different, suited to the particular subsystem component. The memory hierarchy uses a wide range of well-known techniques, including ECC and duplication, to ensure that data and control information are preserved and that the CPC continues operation even after a permanent failure. The implementation is customized for each major array in the memory hierarchy subsystem.

The CPUs and channel subsystem have extensive recovery mechanisms to survive transient and permanent faults. These two subsystems also achieve fault tolerance by gracefully degrading after a permanent hardware fault. The big

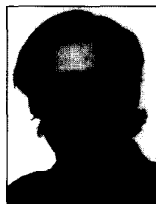
advantage of this design feature over more common explicit redundancy is that it permits full performance almost all the time and degrades performance only when a failure occurs. We have measured the mean time between permanent failures of a CPU, for instance, in decades. The inherent redundancy of multiple CPUs and multiple channels that exist mainly for performance also contribute greatly to the 982's fault tolerance. ■

References

1. D. Bossen and B. Hsiao, "Model for Transient and Permanent Error Detection and Fault Isolation Coverage," *IBM J. Research and Development*, Vol. 26, No. 1, Jan. 1982, pp. 67-75.
2. C.L. Chen et al., "Fault-Tolerance Design of the IBM Enterprise System/9000 Type 9021 Processors," *IBM J. Research and Development*, Vol. 36, No. 4, July 1992, pp. 765-779.
3. J.S. Liptay, "Design of the IBM Enterprise System/9000 High-End Processor," *IBM J. Research and Development*, Vol. 36, No. 4, July 1992, pp. 713-731.
4. IBM Corp., *Enterprise Systems Architecture/390 Principles of Operation*, Order No. SA22-7201.
5. IBM Corp., *MVS/ESA Component Diagnosis and Logic: Alternate CPU Recovery*, Order No. LY-1432.
6. J.C. Daly and F. Frey, "Nondisruptive Resuming of Work from Checkstopped Central Processor," *IBM Technical Disclosure Bulletin*, N10b, 03-92, 1991, pp. 204-205.
7. D.K. Pradhan, *Fault Tolerant Computing Theory and Techniques*, Vol. II, Prentice-Hall, Englewood Cliffs, N.J., 1986.
8. IBM Corp., *Enterprise Systems Architecture/390 IBM System/360 and System/370 I/O Interface Channel to Control Unit Original Equipment Manufacturers' Information*, Order No. GA22-6974.
9. J.C. Elliott and M.W. Sachs, "The IBM Enterprise Systems Connection (ESCON) Architecture," *IBM J. Research and Development*, Vol. 36, No. 4, July 1992, pp. 577-591.
10. J.R. Flanagan, T.A. Gregg, and D.F. Casper, "The IBM Enterprise Systems Connection (ESCON) Channel: A Versatile Building Block," Vol. 36, No. 4, July 1992, pp. 577-591.
11. C.J. Georgiou et al., "The IBM Enterprise Systems Connection (ESCON) Director: A Dynamic Switch for 200Mb/s Fiber Optic Links," *IBM J. Research and Development*, Vol. 36, No. 4, July 1992, pp. 593-616.
12. L. Spainhower et al., "Design for Fault Tolerance in ES 9000 Model 900," *Proc. Fault-Tolerant Computing Symp.*, IEEE Computer Society Press, Los Alamitos, Calif., 1992, pp. 38-47.
13. K.R. Covi, "Three-Loop Feedback Control of Fault Tolerant Power Supplies in ES/9000 Processors," *IBM J. Research and Development*, Vol. 36, No. 4, July 1992, pp. 781-785.



Lisa Spainhower is a senior engineer in the Systems Technology organization of IBM's Large Scale Computing Division in Poughkeepsie, N.Y. She is the systems availability member of several strategic product teams. She previously developed reliability, availability, and serviceability strategy for the ES/9000 Model 9021 and availability modeling for the Model 3090. She is a graduate of the University of Michigan.



Thomas A. Gregg is a senior engineer in the Connectivity Solutions organization of IBM's Large Scale Computing Division. His interests include the design of high-speed serial communications channels for large-scale computers. He contributed significantly to the development of the ESCON channel. He received an ScB in engineering and an ScM in electrical engineering from Brown University.



Ram Chillarege is the manager of the Center for Software Quality at IBM T.J. Watson Research Center. His work has focused on experimental evaluation of reliability and failure characteristics in systems. He invented orthogonal defect classification, for which he was awarded the IBM Outstanding Innovation award, and he developed the concept of failure acceleration for fault injection experiments.

Chillarege received a bachelor's degree in physics and mathematics from the University of Mysore, another in electrical engineering, and a master's degree in automation from the Indian Institute of Science. His PhD in electrical engineering is from the University of Illinois at Urbana-Champaign. He chaired the Workshop on Fault and Error Models, is an associate editor of *IEEE Transactions on Reliability*, a senior member of the IEEE, and a member of the Computer Society.

For information about this article, contact Lisa Spainhower, M/S P314, IBM, 522 South Rd., Poughkeepsie, NY 12602; or lisa@pkedvm9.vnet.ibm.com.

Reader Interest Survey

Indicate your interest in this article by circling the appropriate number on the Reader Service Card.

Low 159

Medium 160

High 161