Design for Fault-Tolerance in System ES/9000 Model 900

Lisa Spainhower, Jack Isenberg, Ram Chillarege, Joseph Berding
International Business Machines Corporation
South Road, Poughkeepsie, NY 12602

Abstract

The ES/9000 Model 900 is IBM's high-end fault-tolerant commercial processor. Although, high-end commercial processors were traditionally designed to be very reliable, this is the first one that implements a fault-tolerant machine. The design exploits circuit level concurrent-error detection, fault-identification and reconfiguration with system level techniques when multiple functional resources are available. It provides true graceful degradation during Central Processor or Channel re-configuration and repair. This paper:

- Discusses the design point for this processor and the trade-offs involved.
- Shows the error detection and on-line repair process of a Central Processor with the work recovered on an alternate Central Processor, transparent to the application.
- Describes Dynamic Path Selection and the hot-pluggable channels.
- Illustrates the fault-tolerance techniques used in the Level 1 Cache and the Central Store.

1 Introduction

This paper presents the design for fault-tolerance in the ES/9000 Model 900 high-end commercial processor. The Model 900 is a 6-way tightly coupled multiprocessor with a two-level cache, expanded storage and fiber optic channels. Compared to its predecessor, the 3090 model 600J, it provides about twice the processing power and the following reductions in components: a 30% reduction in the number of chips, 40% reduction in the number of Thermal Conduction Modules, (TCMs are the second level multichip packaging vehicle) and a 40% reduction in the number of signal cables. The machine also makes significant advancement in processor design, implementing out of sequence execution, multiple execution elements, etc. This paper focuses only on the design of the fault-tolerant aspects of the machine whereas details of the processor organization can be found in [Liptay92].

The design of a fault-tolerant machine in this market segment involves a complex set of trade-offs. These trade-offs involve, cost, performance, packaging, maintenance strategy, operating system support and customer requirements. Thus, the choices of techniques to achieve fault-tolerance have to be weighed both from a top down and a bottom up view without losing perspective of the final application program or customer view. Since the design team does not start with a clean slate there is a significant evolutionary process from the earlier generation of machines. The earlier generation of machines has been proven to be very reliable, using extensive error checking and recovery techniques in the processor. Furthermore, the packaging, screening and burn-in of components provide additional leverage of very large mean-time-between failures. To turn a high-end machine of this class into a fault-tolerant machine requires a very careful selection of the design point. Clearly, the challenge in this market segment is in designing a fault-tolerant machine that continues to be performance driven and cost competitive.

This paper first discusses the Design Point. The standard circuit design process in the high end enforces the principles of concurrent error detection and fault isolation. To take this design point and make it fault-tolerant one needs to enhance it to provide total concurrent error detection and recovery with on-line non-disruptive reconfiguration [Pradhan86]. This is designed by integrating the circuit level capabilities of error detection with system level redundancy to provide complete recovery and reconfiguration. The resulting fault-tolerant system provides graceful degradation of processing power until repair, with complete transparency to the existing application base.

Following the discussion on the design point, there are three sections discussing the details of the major subsystems, the Central Processors (CP), the Channels and the Storage Subsystem. In each section we have illustrated some of the key ideas used in this design and how they have been implemented. Clearly, no single paper can exhaustively discuss all the elements that

0731-3071/92 $03.00 © 1992 IEEE

go into building the fault-tolerant system. The power subsystem and operating system layer are beyond the scope of this paper. This paper should interest system architects, integrators, and researchers in fault-tolerant computing.

2 The Design Point

A critical part of any such design is a clear recognition of customer requirements. In the high-end there has always been a very strong emphasis on data integrity. Data integrity implies that all computation is checked and in the event of an unrecoverable error, the program aborted or the machine stopped. This is the reason why concurrent error detection is so advanced in most previous IBM high-end processors. Data integrity however, requires only that an error not be allowed to propagate - an unrecoverable error could result in the job or program being terminated. This may result in a system outage if the job being terminated is a critical software subsystem that failed software level recovery. In the case of a permanent hardware failure which requires repair, the processor would be check stopped. In recent years, the requirements have gone beyond data integrity alone. The goal is that no computation be lost, thereby never requiring program termination; that repair be on-line and non-disruptive, providing continuous availability.

In most commercial environments a slight loss of performance is acceptable during the process of repair. This is particularly true considering that a) the processors are expensive and an expensive idle processor is usually unacceptable; b) although systems are operated at almost full utilization (around 90%) they are rarely fully utilized. A small shortfall in processing capability can be handled by workload management algorithms in the dispatcher or by shedding some less important workloads. With the repair becoming non-disruptive, true graceful degradation becomes the goal in these environments.

The design of a high-end processor is primarily driven by performance and cost trade offs. For performance reasons, the organization of the Central Processor Complex (CPC) has several identical elements, in parallel, such as processors, channels, levels of storage, etc. Thus there is a fundamental capability for high availability from the redundant resources that are deployed for performance. At the same time, each of these elements 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 enhance the existing Error Detection and Fault Isolation (EDFI) techniques and couple them with system level techniques exploiting the identical elements in the CPC. The resulting design can identify errors and recover transient and intermittent failures by retry. For permanent faults requiring repair, the CPC will isolate the failing element, reconfigure the system around it, recover by rolling back to an error-free consistent state and resume execution. The failed hardware can be replaced without affecting the rest of the system.

In the following sub-sections we describe the error handling and maintenance capability and then describe the design goals for each of the Subsystems.

Error Handling and Maintenance

In order to ensure data integrity and provide on-line diagnosis, the circuits in the high-end employ concurrent error detection and fault isolation. To achieve the fault-tolerance and continuous-availability goals, it was critical that this capability be enhanced. Thus the design goal was to increase the level of error detection from what was typically in the 90% range to total concurrent error detection. The design point for error detection is to ensure that an error is first identified within the same machine cycle. The reporting and recording may take additional cycles. Every latch is protected by a code and there are no naked latches. All dataflows, arrays, and control busses have ECC, or checking, and state machines have parity predict, etc. Expanding to this level of coverage added less than 3% to the number of logic chips. One reason for the small increase in overhead is that many of the *hard to check* logic chips, like sequence logic, are very pin limited and thus the logic could be duplicated on the chip for checking without costing more chips. Another reason was that in a heavily pipelined machine at the high-end, a lot of the circuits are in parallel where the checking levels are already very high.

There are a several failure modes that this Design Point addresses however, for the purpose of understanding the error detection and recovery strategy we discuss a few of the dominant circuit fault models. Broadly, errors in the logic can occur due to solid hard circuit faults, intermittent faults, and transients faults. The solid hard circuit faults, commonly called permanent faults, occur when a digital circuit no longer yields a correct output given a specific set of inputs. Any time the specific input is repeated, the incorrect output is produced. An intermittent fault is an occurrence where a specific event produces an incorrect result, but the same inputs at a different point in time may produce the correct result. This can occur as a result of a design error, marginal circuits or loading conditions. Transients occur when there are environmental conditions, noise transients, or cosmic particles cause an incorrect result, but the circuit itself functions correctly.

To provide online error correction and repair not only should the error be identified but also resolved whether it is a transient or permanent. For the case of transient, the recovery should be guided to the source of the data that had an error and for a permanent, the exact identity of the replaceable part. As an example, the design ground rules require that error checkers be placed at the driver for any signals that leave the replaceable part, and immediately after the receiver of any signals entering a replaceable part. This allows most failures to be identified to the correct part without further analysis. The on-line EDFI would determine the Field Replaceable Unit (FRU) [Bossen82] to be replaced. It is neces-

sary to be able to determine whether an error is due to a permanent physical failure requiring repair, or only a transient that can be handled without physical part replacement. In an ES/9000 system it is not always easy to distinguish which type of fault has occurred, but the design handles all of them. The entire logic structure called for the capability to back out and retry all internal operations with appropriate thresholds for determining success. Since the retry of 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 and, 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 system implementation. A detailed description of the error detection techniques can be found in [Bossen92]. Note that failure due to a hard circuit fault might be recoverable through instruction retry since the machine is run unoverlapped during the retry, thereby using different circuit combinations. Intermittents 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 are used to determine whether a unit needs to be taken out for repair or whether the system can continue with the unit containing the failure.

If a failure is determined to be a physical failure requiring repair, the design requires the capability to fence off all external interfaces to the element being repaired so that reconfiguration and maintenance can be performed while the remainder of the system continues in operation. Thus all interfaces have the capability to "fence" or de-gate the drivers and/or receivers of the interfaces. When a failure is determined to require repair, a service request is initiated. The Processor Controller Element is provided with a Remote Support Facility (RSF) which can report failure information automatically. An auto-dial for service from the RSF will result in the correct routing of the information to the customer service engineer, parts stocking location, and, if further analysis of the failure information is required, to the remote support center.

Design Goals for Subsystems

The design choices for fault-tolerance pay careful attention to both the physical layout of the machine and the logical boundaries. Figure 1 shows the physical structure of the Model 900. Each central processor (CP) is contained on 1 board in 4 TCMs which includes two 128K byte store through Level 1 Caches (L1). Each CP contains multiple execution elements (each for different types of instructions) and implements out of sequence instruction execution. There are two System Control Element (SCE) boards each with 6 TCMs and each board has 8 card positions to accept up to 512 MBytes of central storage. The SCE includes several elements, a store-in Level 2 cache (2MBytes per SCE), the directories and store buffers. There are two Inter-

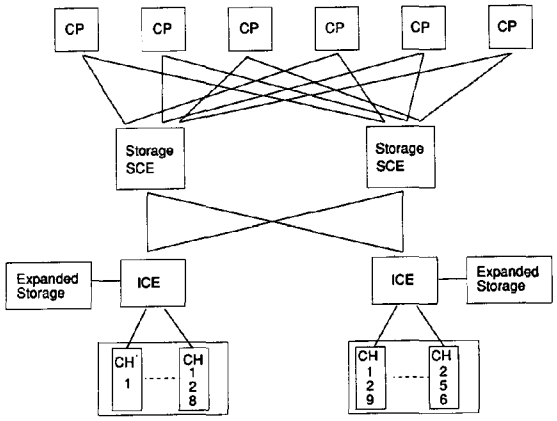

Figure 1: ES9000/900 System Structure

communications Connection Elements (ICE) per System - each is packaged on a board with 5 TCMs. This element contains an I/O processor, channel control and switching function and the Expanded Storage controls. The Expanded Storage is packaged on cards and up to 8 GBytes is available. The system can be configured with up to a total of 256 channels, 96 of which could be Parallel channels and the remainder Enterprise Systems Connection Architecture (ESCON) fiber optic channels. To avoid confusion between an individual CP and the package including the TCM boards, storage and channels, the package is called the Central Processor Complex (CPC) or simply the processor.

The CPC hardware can be divided into 3 logical elements. Figure 2 shows the Central Processors (CPs), Storage Subsystem, and the Channel Subsystem (CSS). A different approach was taken in each logical section depending on element redundancy and the concurrent error detection and reconfiguration capable in the element. We examine the goals of each of these logical elements here and describe the details of the design in the following sections.

- The Model 900 is a 6-way tightly coupled multiprocessor which is an inherently redundant design point. The implementation takes care that a program dispatched on any of the CPs and will execute architecturally correctly. If a CP takes a catastrophic failure, the work can be moved to another CP and the failing CP fenced out of the processor resource pool and repaired. Degrading from 6 CPs to 5 CPs is less than $1/6th$ (due to lower MP degradation), which is quite tolerable for most commercial environments. This graceful degradation together with non-disruptive repair provides a very cost-effective solution in a high-end CPC. Additional steps taken in internal design of the CP have the additional benefit of greatly reducing the likelihood of a CP taking a catastrophic failure. The operating system's primary function

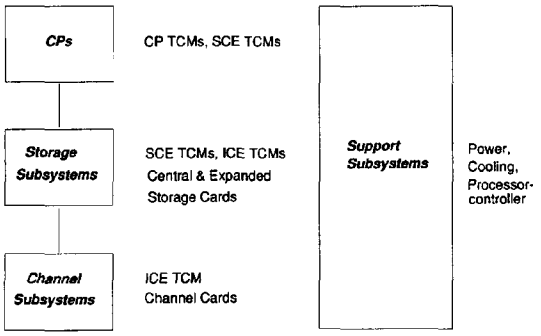

Figure 2: ES9000/900 Logical Structure

is to schedule work to the CPs using algorithms to best manage the workload. Achieving good performance for the important jobs even in a degraded configuration is a natural fallout of the operating system scheduler.

- The Channel Subsystem (CSS) is the means through which the processor attaches to DASD and tape storage devices, display controllers, switches, and communication networks. It is made up of TCM data path and control logic and up to 256 card-on-board channels. The CSS has considerable inherent redundancy. Redundant paths are used in such a way that if one is busy, an alternate is used, thus increasing performance rather than waiting for the busy one to be freed. Performance and availability requirements determine the number of redundant paths to any particular I/O device. These are generally sufficient to tolerate the loss of one path with no noticeable performance degradation. If a channel should fail, the failing channel can be varied out of the active configuration and concurrently repaired. Channel packaging of one parallel channel per card or two ESCON channels per card facilitates the concurrent repair. ESCON [Elliot92] architecture further facilitates the ease with which redundant paths can be configured so that an alternate path can be used to address an I/O device. The Channel Subsystem TCM provides configuration boundaries for fault tolerance in the data funnel from the individual channel to Central Storage.

- The Storage Subsystem is responsible for the handling of system data. It is made up of Central Storage, Expanded Storage, and the L2 Cache. This data is usually accessed synchronously with the CP's operation. The storage subsystem controls manage the data through the use of Least Recently Used(LRU) algorithms. The storage system is not duplicated since this has adverse performance effects and the cost implications are prohibitive. However, several techniques are applicable to make storage fault-tolerant: ECC, scrubbing, array reconfiguration and sparing. Array reconfiguration is the ability to delete a section of storage and move the data for the target address to another area in the storage array. For the central and expanded storage array, spare memory array chips are added for automatic replacement of a failing memory chip. Fault avoidance has also been used most extensively in this element. An entire level of packaging was removed, compared to the Model 600J, by integrating the Central Storage cards into the System Control Element.

- Support subsystems which are not critical to the performance or cost/performance characteristics of the Model 900 widely use explicit redundancy. Each major physical element (e.g. SCE, CP, ICE) has its own power boundary. Within that boundary there are power supplies for each required voltage level. For each voltage level within the power boundary there is one more supply than needed to meet the power requirements. All N+1 supplies are normally active. When one fails, the remaining N automatically re-adjust their output to meet the needs of the load. Likewise, subsequent to a repair, all N+1 automatically re-adjust their output to meet the needs which were being provided by N supplies. Details of this can be found in [Covi92]. The air cooling subsystem fans and blowers are similarly designed. Pumps are redundant in the water cooling subsystem and are periodically switched to detect any latent faults in the inactive pump. The Processor Controller Element (PCE), which provides system operations, service interfaces as well as participating in recovery and reconfiguration, is duplexed. One side is active and the backup continuously runs diagnostic programs and communicates with the active via "I am well" signals. These explicitly redundant subsystems - power supplies, fans, blowers, pumps, PCE - are all concurrently maintainable.

The following sections describe the design of three subsystems that were described in this section, the CPs, Channel Subsystem, and Storage Subsystem.

3 The Central Processor

All System 390 Central Processor Complexes with more than one Central Processor have the ability to dynamically vary any CP on-line and off-line, and, as long as one of the N CPs is on-line, continue instruction execution. With the exclusion of special operations for which hardware may not be provided on all CPs (i.e. Vector Facility and Integrated Cryptographic Facility), each CP in a multiprocessing environment is capable of executing any task dispatched by the operating system and it is unpredictable where any particular task will be

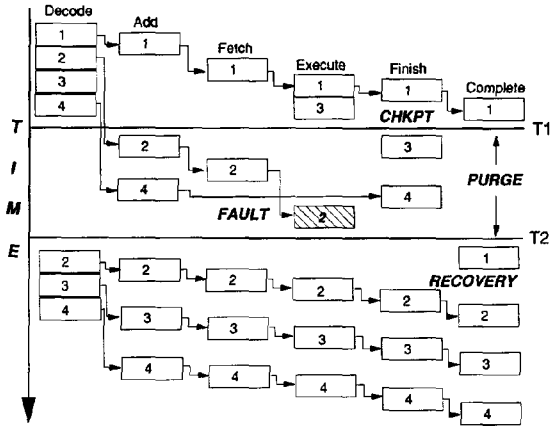

Figure 3: Instruction Retry and Recovery

executed. A major goal of the Model 900 design was to exploit this capability and continue uninterrupted operation in the event that one CP becomes unavailable. Accomplishing this goal required the following designs:

1. The processing state in the failed CP must be rolled back to a consistent, error free state.
2. The state of the storage system as seen by all other CPs must be architecturally accurate and error free.
3. The state of the hardware capability of the failed CP (transient failure vs permanent failure) must be determined.
4. The failed CP must be removed from the configuration before it propagates any corrupted data.
5. The work that was running on the failed CP must be transferred to an operational processor.
6. The failed CP must be repaired and returned to service without a system interruption.

The implementation of this is best illustrated by an example involving instruction retry and recovery. Figure 3 illustrates the pipeline and four instructions that are issued. Note that this machine implements out-of-sequence execution and this example also illustrates the leveraging of a performance technique to boost fault-tolerance at the circuit level. Out-of-sequence execution allows for multiple instructions to be issued without requiring that they *finish* in the same order that they are issued so long as they *complete* in the same order. There is a fine distinction between *finish* and *complete* – finish is when the instruction finishes execution, and complete is when the results of the instruction are put away into storage. Essentially, once an instruction completes it cannot be backed out since its results are posted in storage and architecturally the instruction has executed. Prior to completing, an instruction can be backed out provided all other pending stores from instructions issued after it are also cancelled and the storage is left in a consistent state.

Assume a failure occurs at time T2 as shown in Figure 3. The clocks are immediately frozen at time T2. Note that the last completed instruction was instruction number 1 which completed at time T1. Although instructions 3 and 4 had both finished (they had had their results calculated and put into temporary facilities awaiting completion), they can not be flagged as completed since instruction number 2 has not completed. Since instruction execution results are maintained in temporary facilities until the instruction is determined to be complete, it is possible to back out of all intermediate pipeline operations (including the finished instructions) by flushing the pipeline and temporary facilities. This leaves the processing state, of the program in execution, at an error free state – that is, at a point in time corresponding to the completion of instruction number 1. Since there is assumed to be a failure in the hardware of the CP, a physically separate Processor Controller, examines the pipeline contents, architected facilities, and temporary facilities, and puts them back into a consistent state. The Processor Controller obtains the CP state by scanning out the internal facilities using totally separate scan clocks so that the logic clocks can remain stopped.

Next, the storage system must be insured to be consistent. The Model 900 particularly adapts to this because of its storage hierarchy design. One of the advantages of the store-through L1 Data Cache is providing isolation of internal Central Processor faults. Because the shared L2 Cache contains any critical system data resulting from CP instruction execution, the L1 Cache copy is not critical and can be discarded upon detection of an error.

Determining whether the failure detected was a transient error or a true permanent circuit failure is done by retry. The Model 900 CP register management [Liptay90], the main function of which is to control out-of-sequence instruction execution, provides an audit trail of changes effected by incomplete instructions which is used by instruction retry for checkpoint restart. In order to do retry, the Processor Controller sets the internal state of the logic in the CP to the state it would be in at the completion of instruction number 1. The logic clocks are then stepped to see if the following instructions execute successfully with no errors. If they do, the error was a transient, and normal operation follows. If retry is unsuccessful after a thresholded number of retries, the failure is determined to be permanent and the work must be moved. At this point the failed processor is put into the CP check-stopped state.

CP Checkstop [ESA/390] indicates severe CP damage for which processing on that CP must be terminated. In a uniprocessor, CP Checkstop is the equivalent of System Checkstop and all instruction processing ceases. In the past, when there was more than one CP and Processor Availability Facility (PAF) was not available, the operating system would initiate Alternate CP Recovery (ACR). ACR logically removes the failing

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.