John Reilly
Arthur Sutton
Robert Nasser
Robert Griscom

Processor Controller for the IBM 3081

The IBM 3081 represents an important step in the evolution of large-scale data processing systems. The incorporation of LSI technology in its design has resulted in a departure from the use of traditional support tools and techniques. Its increased processing capabilities and its dyadic organization have further accented the requirements for high availability and ease of operation. This paper describes the Processor Controller unit of the 3081 Processor Complex, which handles the functions of maintenance, monitoring, and control of the system. It discusses how this unit has dealt with these changes to keep pace with the demands of advanced technology and improved system availability. In providing the necessary test and support functions of reset and manual control, the Processor Controller exploits the level-sensitive scan design capability of the 3081 to obtain read/write access to all latches and arrays. This design approach, coupled with significantly expanded capabilities for error recovery, configuration control, and diagnostics, has significantly affected the structure and capabilities of the Processor Controller.

Introduction

This paper discusses the structure and functional capabilities of the Processor Controller for IBM's 3081 Processor Complex [1], which contains the first IBM large-scale processor unit [2] implemented in LSI. The technology used and the internal structure of the 3081 Processor Unit represent significant new characteristics that influenced the design of the Processor Controller. The 3081 Processor Unit, comprised of two central processors, 16 or 24 channels, and up to 32M bytes of main storage, is supported by a single 3082 Processor Controller which is dedicated to the performance of functions associated with the monitoring, controlling, and maintaining of the system. In providing these functions, the Processor Controller exploits the Level-Sensitive Scan Design (LSSD) [3] capability of the machine. The LSSD capability, together with a dedicated path to main storage and thorough visibility to power/thermal components, provides the necessary Processor Controller interfaces into the system.

In this paper we review the design objectives and the development process associated with the 3081 and discuss the impact that these had on the design of the Processor Controller. The LSI hardware test cycle, which had a significant influence on the Processor Controller organization, is examined, the 3081 system-level objectives that had to be met are discussed, and the structure and functions of the Processor Controller hardware and microcode are described.

Processor controller development

A basic objective of the 3081 development effort was to bring up the hardware in several stages in order to gain early experience with the new *thermal conduction module* (TCM) technology. To achieve this, it was decided that three discrete successive steps would be taken:

1. *Module test*—which placed a single TCM in a controlled environment with access to all of its input/output pins. Tests were applied, with the logic cycled and monitored for correct results. Probing of the individual chip input/output pads was permitted in this environment.
2. *Subsystem test*—which packaged a number of TCMs on their respective boards to form the various components of the Processor Unit (central processors, channels, and

system controller). Tests were applied in this environment, with all of the signal lines to and from the components controlled and monitored.

3. *System test*—the final stage of hardware bring-up, where all of the components were successively combined to form the complete 3081 Processor Unit.

In all of these steps, the Processor Controller was used as the primary interface into the hardware. This permitted a staged development of this unit and an ongoing analysis of its effectiveness. This also required that the Processor Controller hardware be of a proven technology; thus, existing hardware components were utilized whenever possible to reduce the Processor Controller development and test time.

In addition to these requirements, a set of fundamental objectives for the Processor Controller were established and stressed to offset some of the potential negative factors associated with the TCM package—factors such as high parts cost, no scoping ability, tighter power tolerance, and less physical partitioning. These factors resulted in the following objectives.

1. *Complete visibility* to the logic of the TCM was essential to the engineering bring-up of the hardware, since scoping of the logic was virtually impossible. This was also required to meet the very rigorous objective for diagnostics in the field. To hold repair costs down it was essential to be able to diagnose problems to the minimum number of TCMs.
2. *Logical isolation* of the components and/or elements of the 3081 was required to permit continued operation in the presence of solid problems. Concurrent analysis of the problem was also required to minimize the impact on availability. This logical isolation was very important since the density of the hardware package prevented the desired amount of physical partitioning.
3. *A single point of control* was desired to remove operator complexities associated with the partitioning of the Processor Unit. The operating system was established as this point of control, and this necessitated a communication path between the operating system and the Processor Controller.
4. *Power/thermal monitoring* was required, to the extent that disturbances in these areas could be detected directly, to avoid changing expensive TCMs when in fact a problem might be caused by a power fluctuation or "glitch." Early in the development cycle it was also projected that the TCMs would require very tight tolerance on power. However, as development progressed, the requirement for tight tolerances was relaxed.
5. *Expanded Processor Controller functions* were necessary to improve on system availability and maintainability to allow the 3081 to be maintained by personnel with knowledge of its physical packaging but not necessarily of its internal workings. With more intelligence built into the Processor Controller, the efficiency and cost of maintaining the 3081 would be improved.
6. *Extensive use of microcode* in all areas of the 3081 was desirable to avoid the long turn-around time associated with hardware changes. This objective, coupled with the relatively low density of the arrays in the TCMs, prompted the requirement for a dynamically loadable microcode structure [2]. This structure is discussed in more detail subsequently in this paper. The extensive use of microcode was of paramount importance to the Processor Controller development because the flexibility afforded by microcode was essential in adapting to the various test phases as well as for the quick resolution of problems encountered in the testing of the LSI elements.

Hardware structure

The physical organization of the 3081 Processor Complex is shown in Fig. 1. It is composed of three major units: the *electronic frame* (Frame S), the *power frame* (Frame X), and the *Processor Controller frame* (Frame E/J). The electronic frame contains LSI hardware for the two *central processors* (PROC boards CP0 AND CP2), the channels (EXDC board), and the *system controller* (SCUB board)—which is the path for CPU, channels, and Processor Controller to main storage. *Main storage*, not shown, is controlled by logic resident on the SCUB and EXDC boards. Power for this electronic frame is provided from the X frame.

The E/J frame contains the Processor Controller, which consists of a *functional controller* and an array of adapters required to provide the interface and storage media essential to the Processor Controller's role. The functional controller contains 128K bytes of storage and an eight-level hardware interrupt structure. For communicating to operating personnel, both on-site and remote, the *display cluster adapter* (DCA) and the *communication common adapter* (CCA) have been provided. The *Processor Controller file adapter*, a large-capacity fixed file, and the *auxiliary storage adapter*, a 512K-byte monolithic storage device, are used to hold the Processor Controller microcode and system diagnostics. The auxiliary storage adapter is also used as a paging device for performance reasons. The *diskette adapter* is used for reading engineering-change data from portable diskettes. All of this hardware was previously designed and built for other IBM products.

Simple hardware paths to the system were established, as shown in Fig. 1, by developing a *monitoring and system support adapter* (MSSA) through which a number of *logic support stations* (LSSs) and *device support stations* (DSSs) could be attached to the controller. In this configuration, the MSSA acts as a multiplexor, gating interrupts and

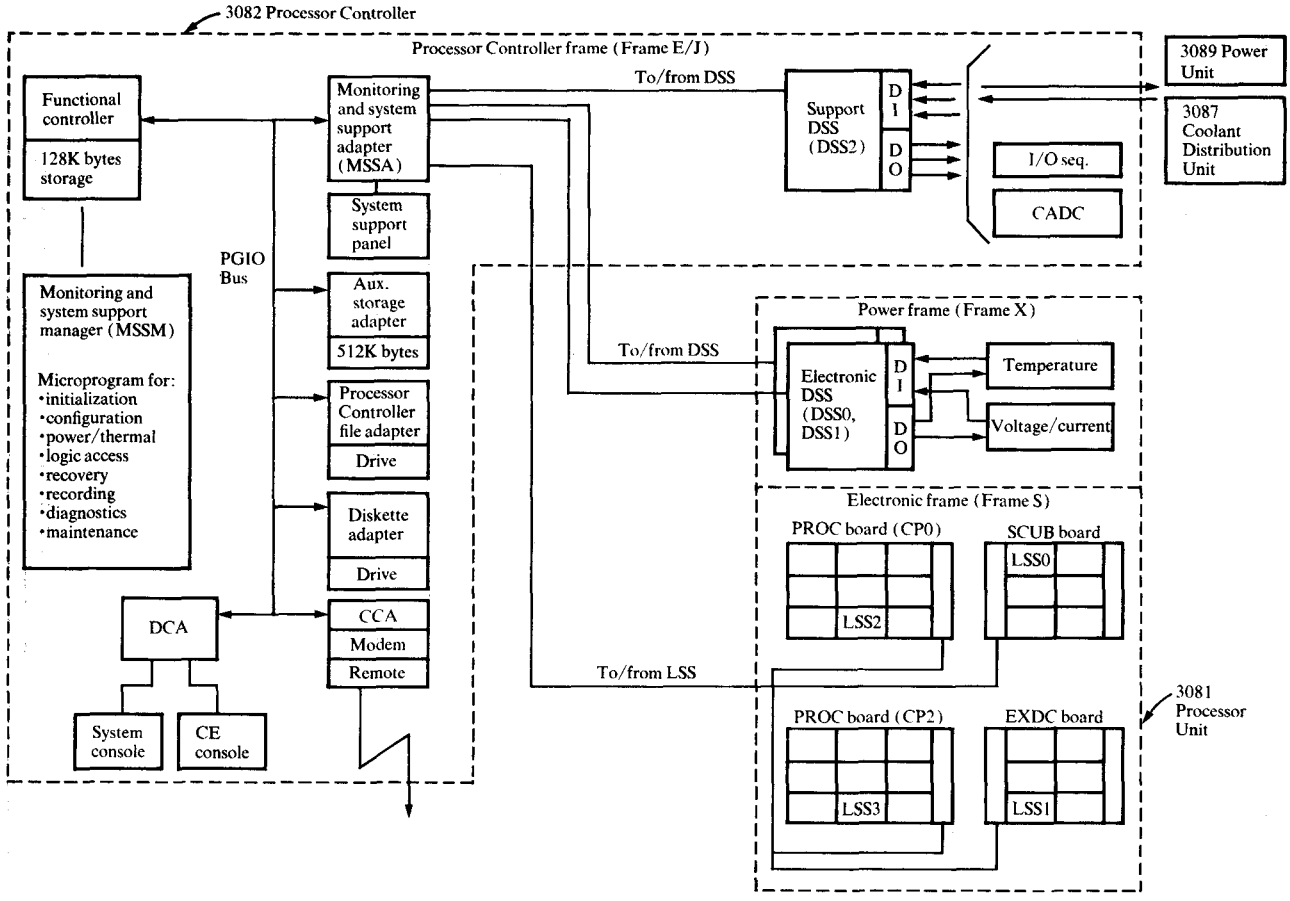

Figure 1 Physical organization of major units of the IBM 3081 Processor Complex. (Note: DCA = Display Cluster Adapter, CCA = Communication Common Adapter, LLS = Logic Support Station, DSS = Device Support Station, PGIO Bus = Programmed Input/Output Bus, DI = Data In, DO = Data Out, CADC = Central Analog/Digital Converter.)

commands between the Processor Controller and an appropriate support station. The LSSs are packaged in the TCM technology, have common logic functions, and provide a path for the Processor Controller into the hardware, with the following basic functional characteristics:

1. *Scanning* capability for the logic of the TCMs, using the LSSD scan-in scan-out method. This technique provides thorough read/write capability for all TCM circuits and arrays.
2. *Partition and clock control* for the components and/or elements of the system. Manipulation of a control register in the LSS, by means of Processor Controller microcode, permits isolation and separate clock control of the various parts of the system.
3. A *read/write of main storage* capability is provided for the Processor Controller through the LSS associated with the system controller (LSS0). This permits the Processor Controller to dynamically access main store without impacting processor and channel activity.
4. *Interrupts*, and associated interrupt codes, from the functional logic and error checker of the TCM hardware are presented to the processor controller through an intermittent path capability.

The DSSs are packaged in card-on-board technology, are designed with common logic functions, and provide control and visibility of the power and thermal elements of the system. Because of the lower reliability of the power and thermal sensors relative to that of other parts of the system, redundant paths were provided to permit the verification of an event through alternate means. The capabilities provided by the DSS hardware are the following:

1. *On/off control and sensor readout* are provided for all components of the power/thermal system.

2. *Digital adjust* capability for regulators is provided through the DSSs. That is, the output of each regulator can be changed by the Processor Controller microcode to any specified value within the adjustment range of the regulator (usually ±7% around nominal).
3. *Monitoring of the regulator output* signals is accomplished by an element of the DSS called the *tracking analog/digital and comparator* (TAD) subsystem (see Fig. 2). This subsystem provides continual monitoring of regulator voltages and currents against thresholds programmed into the TAD subsystem by means of the Processor Controller microcode. Each signal is represented in the subsystem by an eight-bit digital value. This value is updated every 12 microseconds to reflect the latest state of the signal. Associated with each TAD value are two additional eight-bit quantities—the high and low threshold values. These high and low thresholds are established by the Processor Controller microcode. The TAD hardware compares these thresholds against the current eight-bit value representing the tracked signal. Any excursion of the signal outside the programmed thresholds is interpreted as an analog event, and this results in an interrupt being generated by the DSS to the Processor Controller microcode. This interrupt requests the microcode to analyze the power/thermal hardware and, if the signal remains outside the programmed thresholds, to power-off the affected power partition.
4. *Monitoring of TCM temperatures* is also performed in a manner very similar to that for the regulators. The primary difference is that only a high threshold is monitored for TCM temperature.
5. *Digital readout* of voltages, currents, and coolant temperatures is provided by the *central analog/digital converter* (CADC). In fact, the CADC replaces the meter used by service personnel on predecessor systems and provides a simple and accurate means to display the critical voltages, currents, and temperatures of the system.
6. *Interrupts* from the power/thermal elements are presented to the Processor Controller via the DSS. These interrupts signal any abnormal event and initiate an analysis by means of the Processor Controller microcode.

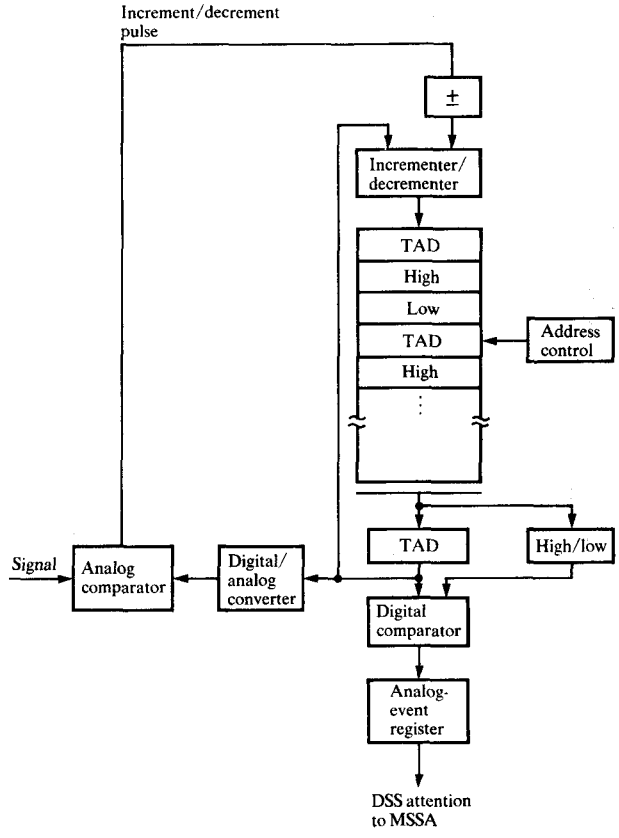

Figure 2 Tracking analog/digital and comparator (TAD) subsystem.

Microcode structure

To provide the required microcode support in an efficient manner, common supervisor routines are used by all support functions. These routines, together with the hardware interrupt structure of the functional controller, provide for fast invocation of high-priority functions. During task switching, the execution of the lower-priority support function is suspended, with the associated code and data rolled out into the auxiliary storage adapter. This process is repeated as high-priority functions are invoked. Upon completion of the high-priority function, the code and data for the rolled-out functions are then rolled in and restarted. This task-switching capability is significant since the larger number of lower-priority Processor Controller functions, such as fault analysis, should not be permitted to lock out a higher-priority function such as error recovery or system reset.

The console structure is designed to permit concurrent activities of operation and either local or remote maintenance. To ensure independence of this concurrent activity, a protection mechanism is built into the microcode which associates the hardware elements of the system with a particular console. Support microcode controlling the interface from the MSSA verifies the authorized association prior to issuing commands to the hardware elements. If the hardware is not associated with the console from which the request originated, the request is rejected and an error message is generated. This association is established during the configuration process, which is described in more detail subsequently.

Hardware support microcode structure

As opposed to building support functions (such as RESETs and DISPLAYs) into the LSI hardware of the system, and also to offset the lack of scoping capability, a primary

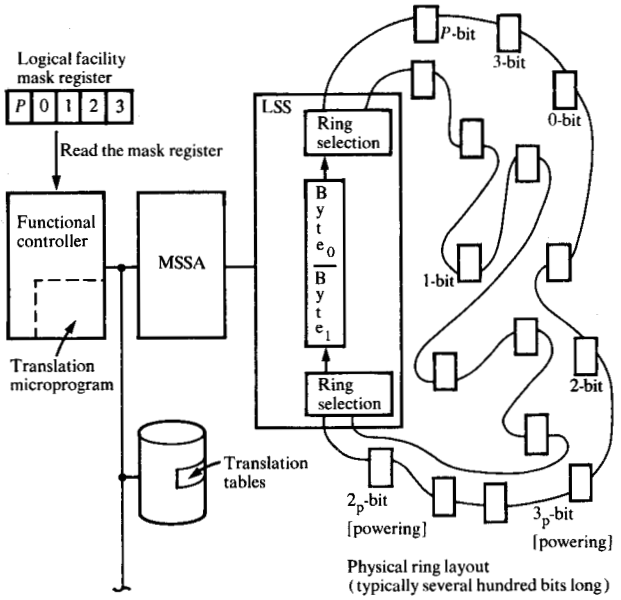

Figure 3 LSSD implementation.

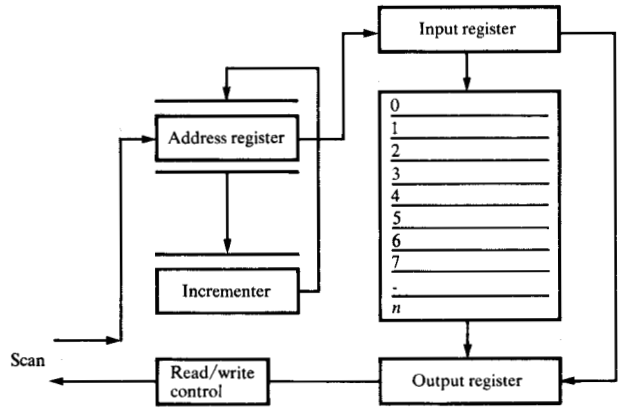

Figure 4 Array scanning with address incrementer.

objective was to provide these functions via the Processor Controller and the LSSD capability. This would provide a simpler hardware design with better overall flexibility. To accomplish this, a structure was established in the Processor Controller microcode to permit easy access of LSI facilities (*e.g.*, registers, arrays) by their symbolic names (*e.g.*, AREG). The relationship associating the symbolic names and the physical LSSD scan rings is held in tables to ensure independence from facility structure variations and engineering changes. The capabilities of this structure can be best understood with the aid of Fig. 3, which shows a typical LSSD implementation with two scan rings connected to a logic support station (LSS). The request to read a facility is received by the support structure, which then indexes into the translation tables to determine the scan rings and bit position that make up the facility. The data from the rings are scanned into the Processor Controller, and the facility is constructed per this table information. Correct facility parity, as well as powering (which is the replication of a facility for logic driving purposes), are also checked. When the operation is complete, the hardware is restored to its original state. On a write operation, only the requested facility is changed. The Processor Controller automatically determines and sets the correct state for the parity and powering bits associated with the facility. The preservation of unrelated data is accomplished by rotating the data in the scan ring, stopping the rotation to read or write selected bits, such that completion results in the ring being shifted to its original position.

The accessing of arrays is more involved and time-consuming. Figure 4 shows a typical array implementation. For a write-array operation, the write control, the address register, and the input register must be scanned. The array clocks are started for one cycle to write the input register into the array. To initialize an entire array, the following process is followed for every location of the array: scanning out of the address register, incrementing of the address, scanning of the address and input data back to the array, and cycling of the array clocks again. To improve the performance of this read operation, and for a write operation such as reset wherein the same data are stored into all locations of an array, a hardware address register incrementer was added to arrays such that, when the array clocks are cycled, the address is incremented by one. This hardware incrementer is shown in Fig. 4. The operation for writing the same data to consecutive locations can be a series of write-array clock commands. For a read operation, a series of scans followed by read-array clock commands are sufficient to collect data from all locations. This technique significantly reduces the total array write/read time and is used extensively by reset and logout functions.

Since this translation function required a large amount of table information stored on the Processor Controller file, the performance for certain functions (such as resets and diagnostics) was not acceptable. To resolve this, a compiler was developed which performs this translation process, yielding data for complete scan rings, thus permitting the bypassing of the translation function.

Hardware clock control microcode structure

The use of LSSD requires the stopping of system clocks to permit scanning. Clock control is provided in the LSSs such that clocks may be stopped on a per-component basis, thus permitting the scanning of a component while the remaining components are unaffected. To prevent spurious signals from

the TCM board being scanned, the components which are running are signaled, by the Processor Controller via the LSS, to degate all lines from the component being scanned. This capability permits as much concurrency between system operation and support function as possible. For example, recovery can run on a central processor component concurrently with system operation utilizing the remaining Processor Unit components.

Processor Controller storage path

A section of main storage is reserved by the Processor Controller for hardware support purposes. This *hardware system area* (HSA) is used to hold loadable microcode, control blocks, and trace data required or generated by the processor and channel hardware. A dedicated path to main storage is provided via the system controller such that access to main storage by the Processor Controller can be concurrent with system activity. This capability has been exploited to develop a real-time communication mechanism between the Processor Controller and the processors, channels, and *system control program* (SCP). Most important of these is the interface to the SCP, which is used to simplify the operation of the system, as is described subsequently.

Support functions

The support microcode structure, built on top of the hardware structure and paths into the system, provided an excellent base for the development of necessary application microcode. The structure provided all of the support function and system visibility that the applications required to initialize, control, recover, and maintain the system.

Initialization of the 3081, to the point where the SCP can be loaded, is controlled by the Processor Controller. Hardware controls in the power distribution frame and MSSA initialize the Processor Controller, which then powers on the remainder of the system. The regulator thresholds are established and the output of the regulator is measured by the Processor Controller microcode. This value is then compared with the expected nominal voltage for that regulator. If they are not equal, the regulator is adjusted and another measurement and comparison made. This measure-compare-adjust process is continued until the desired voltage is obtained. At this point, the tracking analog-to-digital value for that regulator represents the nominal output, and the high and low thresholds are established around this value. The temperature thresholds for each TCM have been predefined and are set at this time.

The LSI logic of the system is then reset using the LSS scan function. As part of the initialization process, the Processor Controller dynamically locates an error-free main-storage location for the assignment of the SCP *initial program load* (IPL) area, starting with address zero, as well

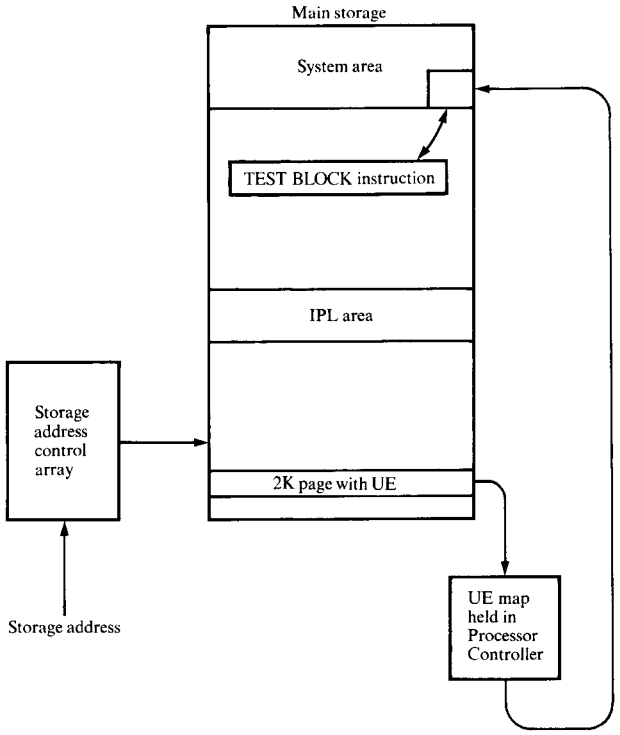

Figure 5 Storage configuration. (Note: UE = uncorrectable error.)

as an error-free area to be used as the hardware system area. (See Fig. 5.) This is done by a processor microcode routine which validates main storage. Main-storage page locations containing a double-bit uncorrectable error, or that fail the pattern tests, are passed to the Processor Controller, which maintains a map of these failing pages. Assignment of storage is accomplished by the Processor Controller analyzing this map and initializing a storage-address-control array contained in the System Controller. The Processor Controller places this map of failing locations into the hardware system area in such a form that it can be referenced by the new TEST BLOCK instruction, which is utilized by the SCP to determine failing storage pages which must not be used [4].

For *controlling* the system, prior to the IPL of the SCP, the operator selects desired configuration and control options by using the Processor Controller functions from the system console. The Processor Controller configures the hardware per the selected options and, at SCP IPL time, passes the configuration information to the SCP via the Processor Controller's interface to main storage [5]. Once the SCP has completed its initialization, configuration changes can only be accomplished by directing the request to the SCP. When the SCP has effected the logical part of the request, it signals the Processor Controller to perform the physical reconfiguration [6]. In the case of reconfiguring

storage, the Processor Controller, in conjunction with the SCP, moves data from one storage location to another to permit taking a storage unit off line for maintenance purposes.

This SCP-to-Processor-Controller communication is accomplished by a new processor instruction MSFCALL, an extension of the DIAGNOSE instruction. Execution of the instruction causes a processor to set up a control block in the hardware system area and signal the Processor Controller. The processor then continues instruction processing while the Processor Controller executes the request. Upon completion of the request, the Processor Controller signals the SCP via an external interrupt.

Exception handling for the 3081 system is performed by the Processor Controller. With the configuration established and the SCP running, the Processor Controller concentrates on monitoring the elements of the system, polling and displaying system status at the system console. In this mode, the Processor Controller is primarily waiting for exceptions, in the form of errors, to occur in the system. The objective is to handle these exceptions in a minimally disruptive fashion while collecting as much data as required for use in the automatic fault-isolation procedures which run on the Processor Controller. A technique used to minimize the effect of a hardware error is to stop only the clocks of the failing component, permitting the remaining components of the system to continue. If the error is an *uncorrectable error* (UE) in memory, the Processor Controller updates the storage error maps described earlier.

When a central processor encounters a solid error and is placed in the check-stop state by the Processor Controller, the store-in-cache design results in master data being locked in the cache of the processor. By accessing the System Controller, the Processor Controller can determine that this condition exists for specific storage addresses. The Processor Controller then issues a special command to the System Controller to force bad data for each address of main storage corresponding to this locked data [7]. In this manner, data integrity is maintained and, since most data contained in the cache at the time of error pertain to a unique task, only that task will be affected.

Another example of how the Processor Controller is utilized to improve system availability is in the recovery from storage errors caused by alpha particles [8]. On detected UEs, the Processor Controller, via a special algorithm, applies a test pattern to the data area to determine where the hard error exists. Through the algorithm, the soft error (alpha-particle-induced) is corrected and the data are restored. The central processor that detected the error is then retried successfully.

The power and thermal components of the system are continuously monitored and tracked. Exceeding the operating limits causes an interrupt to the Processor Controller, which logs all associated data for fault analysis and takes the least-disruptive action where this does not conflict with hardware protection. A key ability of this exception handling is the automatic correlation of a power/thermal exception to hardware errors that are likely to occur at essentially the same time. This capability is used to prevent the replacement of hardware when in fact the hardware error was caused by the power/thermal event.

Maintenance of the 3081 is a key role of the Processor Controller. To meet the maintenance time and cost objectives of the 3081, both error-detection and fault-diagnosing capabilities were enhanced. Error detection was stressed and evaluated early in the design cycle when improvement could be easily made [9]. In addition, the Processor Controller was exploited to perform thorough error data collection and analysis. In most cases, sufficient data are collected on the first occurrence of a fault to permit direct isolation to a very small number of *field-replaceable units* (FRUs). The correlation of multiple faults, both power/thermal and hardware, significantly enhances the Processor Controller's ability to isolate the problem to a small set of FRUs. In addition to fault analysis, a set of very comprehensive hardware tests can be run to isolate solid problems and to verify the repair action. These techniques are described in detail in the paper by Tendolkar and Swann in this issue [10].

Should these maintenance procedures fail to resolve the problem, the Processor Controller provides a large assortment of tools to be used by service personnel to identify and fix the problem. These tools capitalize on the LSSD and clock control functions for thorough visibility of the hardware. A remote capability exists which permits all Processor Controller functions to be used by highly trained personnel in a support center.

In cases where the Processor Controller hardware contains a solid fault, built-in diagnostics and analysis routines are utilized. In this case, the support personnel must perform parts replacement using a four-digit reference code indicator on the system support panel (Fig. 1). Should this repair be unsuccessful, the Maintenance Device [11] is attached to a receptacle provided in the Processor Controller. The support personnel are led through a question-and-answer dialogue, at the Maintenance Device keyboard, intended to find and fix all residual hardware failures not found by the built-in diagnostics.

Conclusion

The Processor Controller represents a significant evolution in meeting the needs of a large-scale LSI computer system.

It has extended the role of processor controllers on predecessor systems in key areas, offsetting potential problems associated with LSI hardware. It has provided traditional functions, such as resets, in a cost-efficient fashion, reducing the number of circuits required in the 3081 by exploiting LSSD to provide these functions. In addition, the flexibility it provides in these areas has contributed to efficient hardware bring-up and to reducing hardware change activity in the IBM 3081 Processor Complex. The Processor Controller design has also been successfully exploited to attain very high objectives in the area of improved system availability and operational characteristics. By providing improved clock-control granularity, automatic configuration control, and automatic fault isolation, system incidents and outages due to hardware failures or operator errors should be reduced.

Acknowledgments

The authors wish to recognize the significant contributions to the Processor Controller development effort made by G. D. Granito and J. D. Vowell, who established many of the key objectives and who were persistent in their efforts to ensure that those objectives were met. Acknowledging the efforts of the many engineers responsible for the design effort would be difficult. However, the authors wish to recognize a few who made significant contributions to the design direction: T. A. Stranko for the hardware design, L. A. Guadagno for the power/thermal structure, E. Borchsenius for the LSI service tools, and C. L. Baldwin and S. P. Geiger for the structure of the Processor Controller microcode.

References

1. M. S. Pittler, D. M. Powers, and D. L. Schnabel, "System Development and Technology Aspects of the IBM 3081 Processor Complex," *IBM J. Res. Develop.* **26,** 2–11 (1982, this issue).
2. R. M. Gustafson and F. J. Sparacio, "IBM 3081 Processor Unit: Design Considerations and Design Process," *IBM J. Res. Develop.* **26,** 12–21 (1982, this issue).
3. E. B. Eichelberger, "Level Sensitive Logic System," U. S. Patent 3,783,254, 1974.
4. E. L. Richardson, R. M. Smith, A. J. Sutton, and J. D. Vowell, "Retention of Failing Storage and Storage Protect Key Address Through Initial Program Loads," *IBM Tech. Disclosure Bull.* **22,** No. 1 (January 1980).
5. B. B. Moore, J. T. Rodell, A. J. Sutton, J. D. Vowell, and P. J. Wanish, "System Control Program Initialization in a Multiprocessing System," *IBM Tech. Disclosure Bull.* **22,** No. 9 (February 1980).
6. B. B. Moore, J. T. Rodell, A. J. Sutton, and J. D. Vowell, "Automatic Vary of a Central Processor in a Multiprocessor System with Non-Store-Through Cache," *IBM Tech. Disclosure Bull.* **22,** No. 4 (September 1979).
7. E. J. Annunziata, B. L. McGilvray, and A. J. Sutton, "Cache Purge in Store-in-Cache System," *IBM Tech. Disclosure Bull.* **20,** No. 9 (February 1978).
8. D. C. Bossen and M. Y. Hsiao, "A System Solution to the Memory Soft Error Problem," *IBM J. Res. Develop.* **24,** 390–397 (1980).
9. D. C. Bossen and M. Y. Hsiao, "Model for Transient and Permanent Error-Detection and Fault-Isolation Coverage," *IBM J. Res. Develop.* **26,** 67–77 (1982, this issue).
10. Nandakumar N. Tendolkar and Robert L. Swann, "Automated Diagnostic Methodology for the IBM 3081 Processor Complex," *IBM J. Res. Develop.* **26,** 78–88 (1982, this issue).
11. *Processor Controller Maintenance Manual,* Order No. SY22-7063, IBM Corporation; available through IBM branch offices.

Received June 9, 1980; revised July 6, 1981

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