L. J. Boland
G. D. Granito
A. U. Marcotte
B. U. Messina
J. W. Smith

The IBM System/360 Model 91: Storage System

Abstract: This paper discusses the design concepts employed in the development of the IBM System/360 Model 91 storage system. Particular attention is paid to the exploitation of System/360 capabilities in the areas of large storage capacity, concurrent operation, and flexibility, as they apply to the highly overlapped Model 91 system.

An interleaved set of main storage modules is used with the Model 91 to help mask the difference between machine cycle time and storage access time. The set is connected to the central processor, peripheral storage control element and maintenance console by three time shared busses—one for addresses, one for data-in, and one for data-out. The main storage control element (MSCE) controls these busses to maximize the storage access rate. To achieve minimum access time, requests are normally sent directly to the storage modules. The proper module is selected by the MSCE, the address gated in, and the storage cycle started. If the module is busy from a previous request, the request is stored in a request stack for a later attempt. If the request is accepted, it is stored in an accept stack. This stack controls the data-out gating of the storage modules, and notifies the CPU of the destination of returning data. It also furnishes module busy information which controls the recycling of rejected requests.

An important feature is the ability of the MSCE to logically sequence store/fetch requests, by interlocking the rejected requests with the current request without any degradation of minimum access time. Additionally, each address sent to the MSCE is compared with the addresses of waiting and in-process requests. This allows serial fetching of two adjacent single words of a double-word storage cycle. Fetches following stores to the same location can be executed without waiting for a fetch storage cycle.

Peripheral storage is provided in the system for both block transfers of data and individual word fetches and stores. All requests to peripheral storage are sent via the peripheral storage control element.

The MSCE is synchronized with the CPU and uses the same machine cycle. Ideally, a request can be honored each machine cycle, but the actual rate is determined by storage module conflicts. The storage system performance is measured in access rate and access time. The MSCE has been simulated to measure the effects of storage speeds, degree of interleaving, and changes in MSCE controls.

Introduction

In the development of a highly concurrent processing system, there are two principal considerations:

- The maintenance of a high rate of information flow through the processor requires a storage control system capable of transferring large volumes of data with a minimum of interference.
- Throughput requirements dictate that the input/output handling and buffering capability be equally efficient.

The versatility of the Model 91 can be partly ascribed to a highly overlapped and flexible storage control system which satisfies these criteria. This system is based on a hierarchic concept of storage, with implications of a wide performance range which can vary according to application.

This paper is intended as a description of that system. It is divided into four major sections, to correspond to the points of view from which the design may be considered. The first section discusses the hierarchic concept and overall design objectives. Section two describes the main storage control element (MSCE) which serves as receptor of processor references and controller of all high-performance main storage references. It identifies the requirements imposed on the MSCE from the points of view of the processor and the peripheral storage control element and explains how these requirements were met. (A 4-way, interleaved, 750-nanosecond main storage is assumed for the description.) Section three deals with the peripheral storage control element (PSCE), which controls the flow of data among the peripheral elements of the system. The fourth section explains the characteristic interaction of the elements of the hierarchy, as exemplified by the main and peripheral storage control elements.

General design considerations

- *The hierarchic concept*

The hierarchic storage is characterized by its multilevel structure, consisting of a number of separate but interconnected components of varying sizes and speeds. Successful operation requires a versatile control scheme and depends upon the ability of the operating system and controls to move freely from one level to another in the hierarchy.

The Model 91 storage system has three principal media:

1) High performance main storage: a storage of intermediate size and capacity. Variations in this capacity and in speed and amount of interleaving provide a measured performance characteristic which lends itself to application "tailoring."
2) Extended main storage: This medium extends the capacity of core storage to accommodate the total addressing potential of System/360.
3) File storage and input/output

Each of these elements is monitored and controlled by its own storage control function (Fig. 1). Within the definition of a continuous addressing spectrum (pipeline*), it is possible to establish boundaries within which each control function can operate.

To fully exploit the hierarchic concept, the interconnection scheme must be able to move large quantities of data in several directions and at varying rates. As an example of this requirement, consider the *K-K′* configuration of the Model 91. The *K* component, a high performance main storage, can develop a 172 megabyte/sec rate. The *K′* component, extended main storage, can also develop a 172 megabyte/sec rate, even though it is relatively less accessible to the processor. (Although interference factors can decrease the actual rates, interference is minimized because there is a choice in the configuration of 32 storage units from which to select.) In addition the scheme must accommodate the concurrency of operation required by the processor (potentially 133 megabytes/sec), a storage channel, or data mover (capable of transfers from any point to any point at a rate up to 64 megabytes/sec), and files and I/O (5–10 megabytes/sec combined rate).

To meet these requirements the control system would ideally be able to create as much data transfer potential as there is storage potential. In the Model 91 *K-K′* there exists a potential data demand of 270 megabytes/sec and a potential storage availability of 344 megabytes/sec. (The demand is equivalent to a 72-bit word every 30 nanoseconds required to fully satisfy all users of storage.) Other configurations can extend the potential availability to more than 1,000 megabytes/sec.

*"Pipeline," in this case, means a continuous stream of operations or instructions, capable of being executed concurrently without neglecting any serial dependence among successive operations.

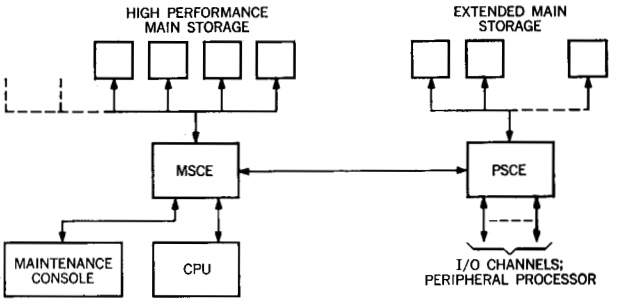
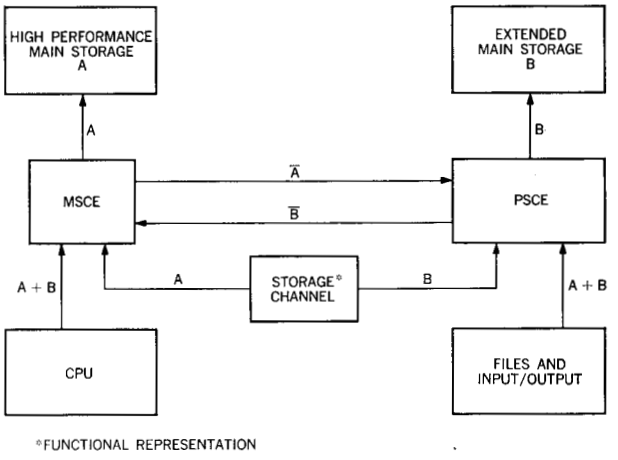


Figure 1 Block diagram of storage system.

Figure 2 Model 91 address flow.



In practice, however, there are constraints. For example, the machine cycle (60 nsec) generally determines the maximum pipeline transfer rate, and the word length cannot be substantially greater than 72 bits because of cable and skew problems. Consequently, one must look to concurrency of transfer to achieve high efficiency. The organization of the Model 91 storage control system is therefore characterized, within the hierarchic concept, by this multiple transfer potential. The separate, bounded control functions mentioned above provide each type of storage with its own transfer path (Fig. 2). Control of high performance main storage resides in the main storage control element (MSCE), while extended main storage and input/output are controlled by the peripheral storage control element (PSCE).

The paragraphs below summarize the specific design objectives which were developed for these units, and subsequent sections describe the operation of each.

- *Control system design objectives*

The key to success in a highly parallel pipeline processor is the ability to react quickly when the pipeline is diverted. Diversion occurs in the form of branching in general, and data-dependent branching in particular. Effective storage

reaction implies that a new flow be initiated in minimum time. Consequently, storage access time must be made as short as possible. Diversion of the pipeline in the concurrent system also implies that many accesses will be initiated without being used. It follows, therefore, that the storage control system must be able to provide many more transfers than can actually be used by a particular problem.

The design objective of the Model 91 was the achievement of a wide performance range which could vary with system application. This imposed on the storage system the following general requirements:

- Control of highly interleaved storages of different speeds.
- Overlap of a multiplicity of high speed I/O devices.
- Development of a very high performance storage channel.
- Minimum disturbance to the processor to achieve a wide performance range.
- Ability to accommodate advanced storage and I/O devices.
- Flexibility to react quickly to various application needs.

To develop these objectives, simulation methods were used to test proposed designs for the MSCE. The effects of interleaving, storage speed, and buffering were observed to determine their impact on processor performance. The impact of various memory cycles on overlapped I/O channels, under each of several design conditions, was also observed by simulation, to determine the form of the PSCE.

With the refinement provided by simulation, the general requirements were defined in terms of the following objectives for the final storage system design:

1) An overall design relatively insensitive to storage speed.
2) Minimization of access time to the processor while maintaining high data rates.
3) Control of large numbers of small storage arrays.
4) I/O control for optimization of overlap operations.
5) Elimination of multiple busses to the individual storage units.
6) Buffering of mismatch between fast I/O and slow storage.
7) Storage protection for highly interleaved variable sets.

By combining proven techniques with novel concepts the design of the MSCE and the PSCE has met these objectives very well. The sections which follow describe that design (and its operation) in detail.

Main storage control element

- *High performance storage principles*

From a CPU viewpoint, the ideal storage system would be one large storage unit with a cycle time equal to the basic machine cycle. The CPU can then issue storage requests on any, or every, cycle. Since this is impossible with the fast cycle of the Model 91, the technique of interleaving is used. Consider a main storage system composed of several (4 to 16) self-contained storage units, which are capable of simultaneous operation. Contiguous addresses are interleaved among the units in a sequential manner. For example, the sequential address string N, $N + 1$, $N + 2$, $N + 3$ would be stored in four different units. The storage system can service a string of sequential requests by starting, or selecting, a storage unit every cycle until all are busy.

Interleaving also improves the servicing of a string of random addresses, since the large number of units reduces the probability that an address will go to a busy unit. Thus the access rate of the storage system is a function of the number of interleaved units, i.e., of the interleaving factor. In practice, the interleaving factor is a binary number, which permits storage address allocation to be determined be decoding the low order bits of the address (for example, the three low order bits for interleaving by 8, or 2^3).

Since the CPU issues storage requests at a one per cycle (or slower) rate, the use of a common set of busses on a time-shared basis is suggested. That is, on every cycle a new address can be transmitted to all storage units over an address bus. The same is true for data words on the busses to and from storage. Since bus cycles can be wasted because of storage conflicts, the control of the busses affects the maximum data rate.

Another performance criterion for the storage system is the access time, which is the time which elapses between the issuing of an address by the CPU and the return of data to the proper sink register. The minimum access time is the sum of the storage unit read time and the cable and logic delays in the MSCE. The average, or probabilistic, access time (which includes the effect of storage conflicts) is limited by the interleaving factor and the storage cycle time. It also depends upon the organization of the MSCE, which must make some provision for conflicts.

- *Design requirements*

Since the ultimate performance of the storage system is limited by the storage units themselves, obvious requirements are that the MSCE must minimize its share of the access time and optimize the data rates by properly controlling the time-shared busses. Certain logical requirements are imposed upon the MSCE by the design of other elements of the Model 91, particularly the instruction unit and the PSCE.

The input to the MSCE from the instruction unit is a storage request, which consists of an address, a return or sink address to route the returning data, control bits to define the operation more precisely, and data for store operations. The MSCE must act on the request by selecting the proper storage unit and furnishing it with an address

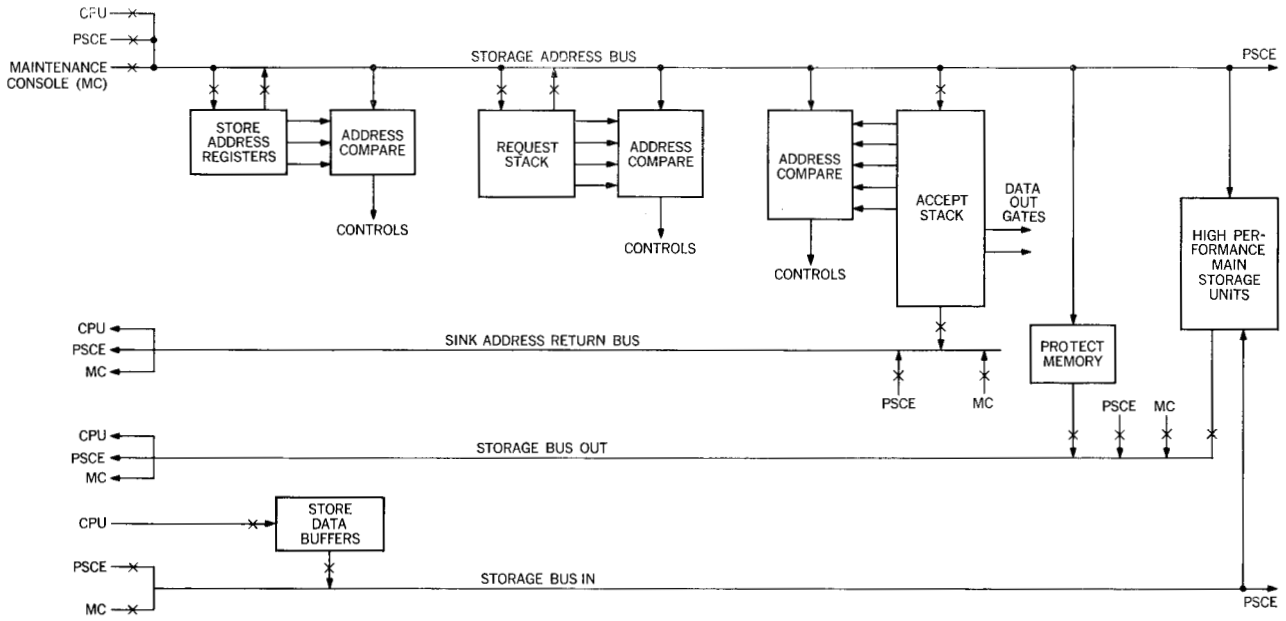


Figure 3 Block diagram of MSCE organization.

and data for stores. In case of a conflict, the MSCE must hold the request for later recycling, generally without stopping the instruction unit. Several requests can be in the MSCE and in storage at the same time, and these need not necessarily be handled in sequence. The various returning data words are correlated with their sinks by the MSCE, which sends the sink address to the CPU a cycle before the data.

In general, requests do not require servicing in sequence, but can instead be serviced in an order which will optimize bus utilization. There is a requirement, however, that the MSCE be able to correctly sequence several stores, or stores and fetches, to the same address.

The storage units considered in the design of the Model 91 have word sizes of 72 bits, including parity, and most units of the CPU are designed to use this word size. Fixed-point and single-precision operations, however, require 36-bit words. Since addresses sent to the MSCE and to the storage units define 72-bit words, two storage cycles could be used in accessing the two halves of a storage word. To avoid this performance degradation, the requirement known as Multi-Access was placed upon the MSCE. This feature allows a memory data register to be read out as many times as desired without recycling the storage unit.

The PSCE carries different requirements because it is connected to the storage and input/output channels. The storage channel objectives include the ability for the PSCE to make requests to the MSCE in bursts, at a one per cycle rate. In addition, requests by the input/output channels via the PSCE could not tolerate uncontrolled delays in the MSCE caused by storage or bus conflicts, because overruns would result. Thus it was decided to give the PSCE the ability to monitor the busy status of main storage, and to reserve main storage units. Given this ability, the PSCE can control its requests to the MSCE so that they are guaranteed acceptance. This will be discussed more fully in later sections of this paper.

The third requesting unit is the maintenance console, which stores and fetches from manual keys, and also initiates the logging in storage of machine status for diagnostic purposes. Since a high data rate is not important, it was judged sufficient to allow the console one request in process at any time.

- *MSCE Organization*

A main storage control element, designed to meet the above requirements, is diagrammed in Fig. 3. It consists of the following functional areas:

- Store address registers (SAR's), which hold addresses of stores pending availability of store data.
- Store data buffers, which hold store data words from all areas of the processor pending availability of the proper storage unit.
- The request stack, a set of four registers which holds rejected requests from the processor pending availability of the storage unit, and thus buffers the processor from storage conflicts.
- The accept stack, a set of registers which holds information on accepted requests in process.
- The storage address bus (SAB), which transmits addresses to all storage units and to the PSCE.

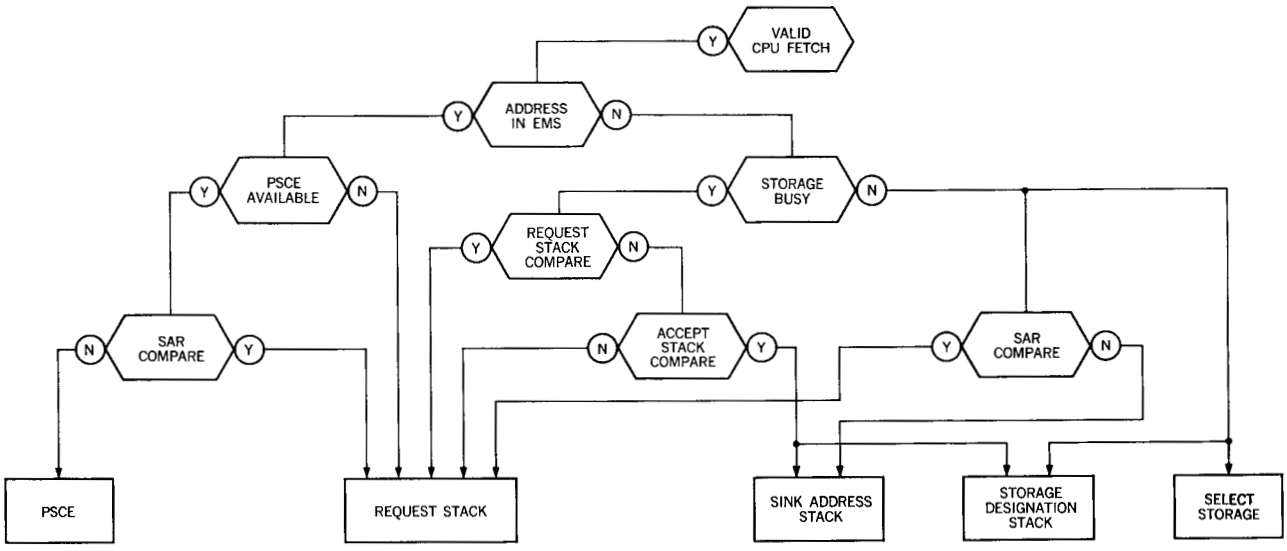


Figure 4 Flow chart of CPU fetch.

- The sink return bus, which transmits the sink address fetch to all sink registers one cycle before the fetch data.
- The storage-bus-out (SBO), which transmits 64 bits and parity of data from storage units, the PSCE, protect storage and the maintenance console keys to all data sinks.
- The storage-bus-in (SBI), which transmits 64 bits and parity of data from processor-filled data buffers, the PSCE, and the maintenance console to storage.
- Protect storage, which stores the keys required for the System/360 Protection Feature.
- Controls

If there are no conflicts, the MSCE can accept a request each cycle from one of its sources—the processor, PSCE, or maintenance console. During each cycle the MSCE controls determine which source will be allowed a request, and gates are conditioned to put the address on the address bus. This bus is also used to load the SAR's, which hold store addresses until the data word is generated by the processor. The main feature of the address bus is its direct path to storage, with no intervening buffers, to minimize access time.

The general organization can best be understood by considering a simple fetch request. During the cycle in which a request is gated on the address bus, the address bus controls determine its disposition. A successful request is sent to the proper storage unit, or to the PSCE if extended main storage is requested. A main storage request is also gated into the top position of the accept stack, the push-through stack which holds pertinent information about all requests in process.

Any rejected processor request is stored in a position of the request stack for later recycling. Requests are not taken from the PSCE or maintenance console unless they can be guaranteed acceptance, and thus never reside in the request stack.

Each address on the address bus is compared with addresses in the SAR's, the request stack and the accept stack. Comparison with an SAR forces rejection of the request and it is stored in the request stack, since its acceptance would cause an out-of-sequence fetch. Comparison with an address in the accept stack implies that the desired word is being fetched by a previous request, and can be obtained again without selecting a storage unit or waiting for it to "go not busy." This is the Multi-Access feature discussed in a previous section. As implemented, it applies to a fetch following either a fetch or a store.

Comparison with an address in the request stack causes the request to be tagged for a future Multi-Access operation, and to be gated into another position of the request stack. The presence in the stack of an outstanding request for the particular address causes the second request to be rejected, keeping the two in the proper sequence. The flow chart in Fig. 4 summarizes the handling of the fetch request by the address bus logic. If the request is accepted, the MSCE generates a select pulse to start the proper storage unit. The selected unit latches the address, which is on the bus common to all units, and starts its cycle.

While the storage unit is cycling, the request is moving down the accept stack, one position each machine cycle. The stack contains the bit code designating the selected unit for n-2 positions, the word address for five positions,

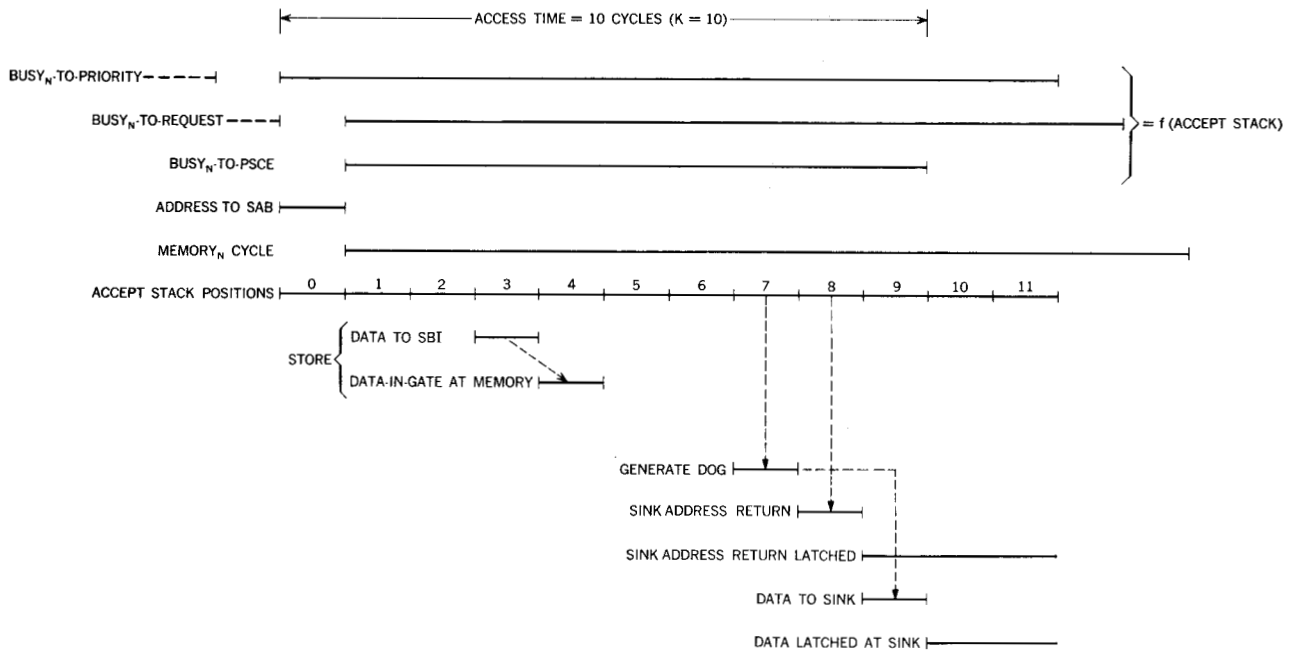


Figure 5 Timing diagram for CPU fetch.

and the sink address of the fetch for k-2 positions, where n is the number of machine cycles per storage cycle, and k is the smallest number of machine cycles required for access. Storage-busy information is obtained from the designation field of the stack, since a unit is busy only if its code is in any stack position. The same field is used to generate the data out gate (DOG), by decoding the k-3 position, and sending the decoded DOG lines to the proper data lines from storage. The DOGs are generated by the MSCE rather than by the individual storage units, to allow for Multi-Access operations. The full address is kept in five positions for comparisons for Multi-Access, as previously described.

The sink address is delayed in k-2 positions, the last position being used to determine the sink for which the fetch was made. The sink address is decoded, and the correct sink register is conditioned one cycle before the fetch data appear on the SBO. If there was no memory conflict, the data word is gated into the conditioned sink register k cycles after initiation of the request. Figure 5 is a timing chart for a simple fetch, where $k = 10$, the case for a 750-nanosecond memory unit. For completeness, store timing is also shown. As shown in the flow chart, the request could be rejected and gated into the request stack for one of the following reasons: (1) The requested storage unit was busy; (2) the request was to the PSCE, and the PSCE inhibit line (queue full) was on; or (3) the address compared with an address in a SAR or the request stack. Priority logic controls the re-cycling of rejected requests in a manner that optimizes the use of the address bus, protects against improper sequences, and guarantees acceptance of the request.

- *Accept stack*

The accept stack deserves a more detailed explanation since it generates many of the necessary control functions. The relation between memory cycle time, access time and depth of the stack has been given above. Figure 6 diagrams a stack with proper depths for a 750-nanosecond unit, with overall access to the processor of ten cycles.

The problem that led to the adoption of the accept stack was the requirement for three kinds of busy information from each main storage unit. This was complicated by the fact that multi-accesses to a unit could effectively make it busy for varying periods. The accept stack solved this problem, with several by-products, as shown by the following list of its functions:

1) Stores the coded designation of each busy storage unit, from which busy information is derived.
2) Delays the sink addresses, and correlates them with their respective data words.
3) Generates data out gates to gate fetched data words to the SBO at the proper time.
4) Stores the main storage address for five cycles, to compare with requests on the bus and identify Multi-Access cases.
5) Aids in maintenance, by effectively allowing single cycling of main storage fetches, and by correlating various errors with the requests causing them.

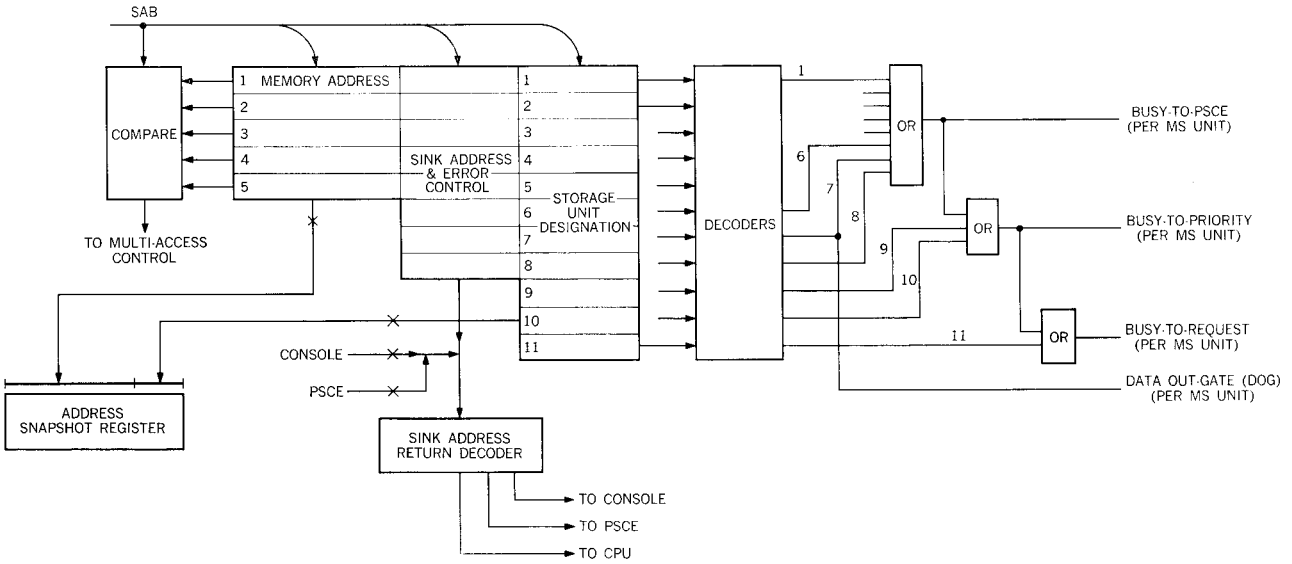


Figure 6 Block diagram of accept stack organization (750-nsec storage unit).

In Fig. 6, the stack itself is seen to consist of three major fields with varying depths. The deepest field, storage unit designation, is used to generate busy information and data out gates. Each position has a decoder, the output of which identifies a storage unit as busy to any request placed on the SAB. In cases of Multi-Access, the same code can appear in more than one position. The three types of busy information, busy-to-select, busy-to-priority and busy-to-PSCE are generated by examining eleven, ten and eight positions respectively as shown. They represent varying degrees of "look-ahead" on busy status. Their use will be explained more fully in the section on controls.

Position 7 is decoded to generate data-out gates. This means that the seven-cycle delay through the seven positions plus the communication time to the storage unit equals the internal access time of the unit.

The "sink and control" field is used mainly to delay the sink address sufficiently to correlate it with returning data. A total of eight cycles in the push-through stack, plus communication time to the sink registers, equals the proper delay to select the sinks one cycle before the return of data. This field is used also to carry error information, which is inserted in the proper position depending upon the source of the error. For example, if an address parity error is detected at storage an error bit is inserted in position three, since the request causing the error has been shifted down to that position.

The address field is used to compare for Multi-Access conditions. Because of circuit loading limitations on the address bus, the depth is limited to five positions. However, machine simulation runs were made on a variety of problems with different depths, and fortunately no improvement was noted with increased depth. The address field is used also, in conjunction with the aforementioned error bits, as a maintenance aid. If an error bit is detected in position 5, indicating that an error was associated with the particular request, the address field is gated into a special "snapshot" register and saved for later use in diagnostics.

- *Controls*

The controls in the MSCE consist explicitly of two functions. First, a decision must be made as to which addressing source should be gated to the address bus. Second, given some address on the address bus, a decision must be made as to which storage unit should be selected, if any. Within each control function there are, of course, many other subtle decisions required to effect logical sequencing. Implicit control of SBO, SBI, and sink address returns is a function of the push-down codes in the accept stack.

For the first decision, priority, the general order of service is:

1) PSCE to (main) storage.
2) Maintenance console to storage.
3) Request stack to (main) storage for Multi-Access.
4) SAR to storage.
5) Request stack to storage.
6) Processor to storage.

A new priority decision is made every cycle, resulting in a time-multiplexed pipeline of priority—address bus—fetch/store.

When no requesting source of priority higher than the processor requires the SAB, processor fetch requests are gated to the SAB (usually on the cycle following the address generation) without a prior test of storage availability. With a sufficient interleave factor and a short storage cycle the requests seldom encounter busy units. Thus the requirement of minimum access time can be attained. Although store requests are held in SAR's where a storage-busy test could be performed, the SAR's are gated to the SAB without the test. Thus the SAR's are unloaded as soon as possible to make them available again to the instruction unit, which considers a store operation completed once it loads a SAR.

Simulation has demonstrated that recycling of requests from the request stack after a fixed-time wait results in secondary rejections which reduce bus efficiency and complicate the control of real-time PSCE requests. Hence, initially rejected processor addresses are recycled only once to the optimized address bus, according to storage-available and first-in, first-out discipline.

Because PSCE requests for main storage require a very high data rate, the address bus efficiency for the PSCE must also be high. Hence, PSCE requests to main storage are granted priority within the PSCE itself as a function of impending availability of specific storage units.

To optimize the overlap of storage units, in priority and on the address bus, their imminent availability (i.e., non-busy status) is as valuable as their actual availability. Hence, three levels of busy status are decoded for each storage unit:

1) Busy-to-PSCE signal, which turns off four cycles before actual time-out of the storage unit.
2) Busy-to-priority signal, which turns off two cycles before actual time-out of the storage unit.
3) Busy-to-select signal, which turns off one cycle before actual time-out of the storage unit.

These signals allow the MSCE to "look-ahead" in order:

- To respond to a PSCE reservation for a storage unit, acknowledging the availability of the unit. Four cycle look-ahead covers the communication delay between the MSCE and PSCE and allows the PSCE to execute a priority cycle (busy-to-PSCE).

- To execute priority for the maintenance console or request stack, each of which needs to know when a specific storage unit is to be available during an address bus cycle (busy-to-priority).

- To allow the generation of a select signal for a busy storage unit during address bus time, if the unit is to be available on the following cycle (busy-to-select).

The second control function is concerned with the disposition of the contents of the address bus on the cycle following the associated priority cycle. When the address bus is valid, the decoded main storage unit or the PSCE is selected if available. If the unit to be selected is not available, the request is routed into the request stack where it resides until the desired storage unit becomes available. When a main storage unit is selected, certain fields are routed into the push-through accept stack for use later in controlling data out gate (DOG) generation, sink address returns, SBI, SBO, storage busy decode, and Multi-Access compares. Special interlocks in the form of address comparators (address bus vs. pending SAR's and pending requests in the request stack) order stores to the same address and recognize and re-order out-of-sequence store/fetch requests to the same address. These interlocks also link these same store/fetch or fetch/fetch requests to the same address for Multi-Access.

If extended storage is decoded, the address bus is gated into a buffer in the PSCE, from which point the address is decoded further to select the appropriate storage unit according to the discipline of the PSCE. If this buffer is not available the request is gated into the request stack in the MSCE until the buffer becomes available. Note that because of the PSCE-MSCE single buffer interface, the PSCE can expand capacity, increase speed, etc., without affecting MSCE control design.

- *Storage protection*

The storage protection feature in the Model 91 performs the same function as in other members of the System/360 family, which is the protection from unauthorized fetches and stores. All attached storage is considered to be in blocks of 2048 bytes, and a 4-bit key is kept in a protect storage for each block. Each request initiates the read-out of the proper address key, which is compared with a key furnished by the requesting source. A mismatch effectively cancels the operation.

Since a protect operation can be required on every MSCE cycle, this suggests either an interleaved set of protect storage units, or one unit with a 60-nsec cycle. If an interleaved set is used, each unit must be of sufficient size to store all keys required by the storage system. Furthermore, the access time of the unit must be fast enough to cancel stores when mismatches are detected, a requirement which becomes more difficult as the attachment of faster units is considered. These factors, as well as the requirement for adaptability to various storage units, led to the adoption of a single high-speed protect storage. The 60-nsec cycle requirement is met by implementing the protect storage in extra-high-performance logic.

- *Variations of storage*

Interfaces between the MSCE and other units have been designed to allow variations in interleaving factor, capacity, and storage speed. Simulation of a random addressing source has demonstrated the relative improvement in

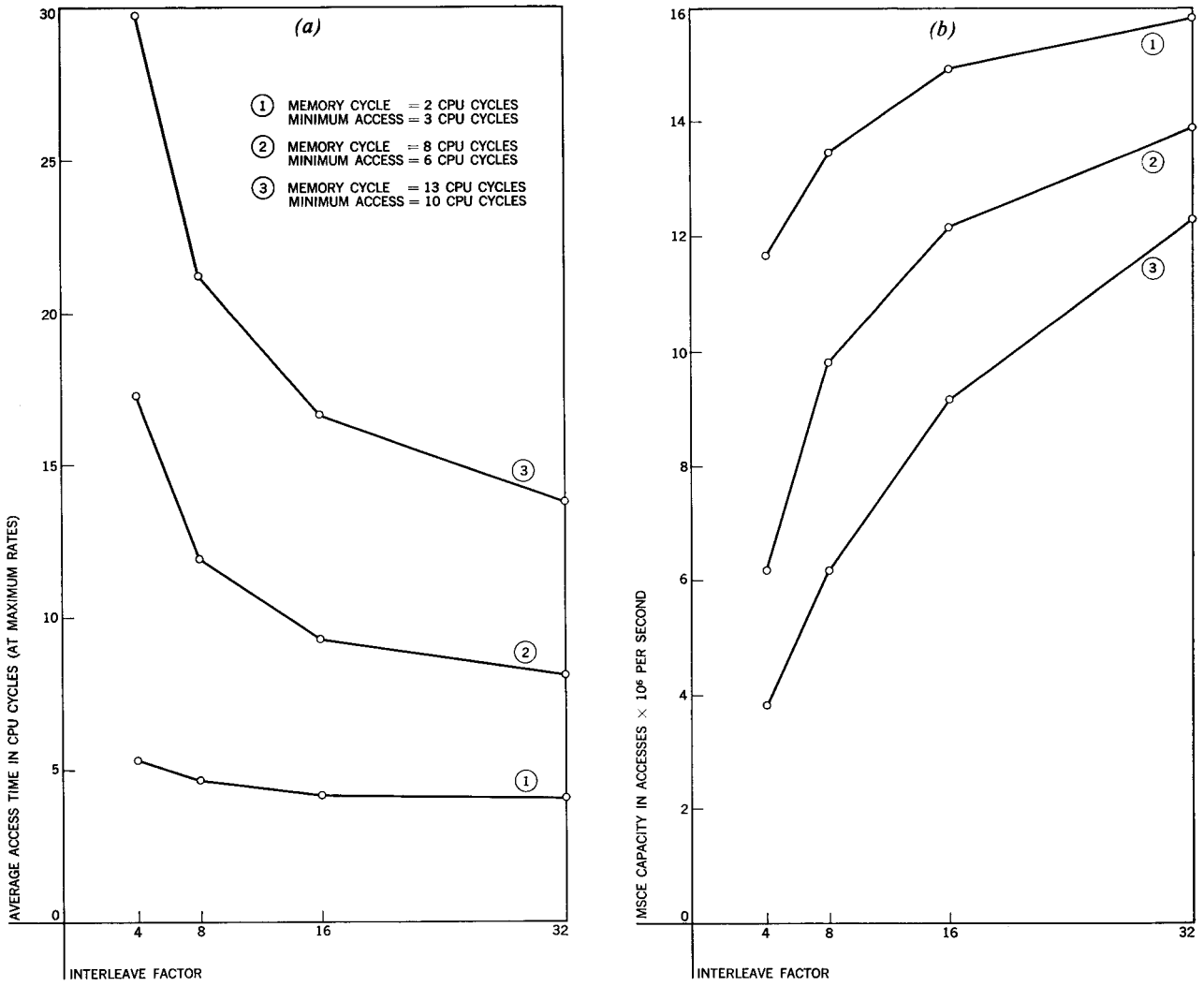


Figure 7 Simulation results. (a) Average access time vs. interleave factor; (b) MSCE capacity vs. interleave factor.

average data rate that is encountered as interleaving and speed are improved (see Fig. 7). The same simulation indicates the average access time on a given storage unit as a function of interleave factor. MSCE design accommodates the flexibility in storage configuration by modular circuit replacement, yielding interleave factors of 4, 8, or 16. Storage cycle time can range up to 750 nsec by varying the number of vertical push-down positions in the accept stack.

Note that the data rate and the average access time improve almost linearly with improvement in storage speed, whereas the interleave factor yields an exponentially diminishing improvement (see Fig. 7). The theoretical improvement due to interleaving can be achieved only by diligent attention to optimal physical distribution of logic and the connection of the interleaved units. Total logic and cable delay from a requesting source to storage to the data sink has been normalized to two cycles. Interleaving of the operation of storage units greater than 16 ways would exceed the two-normalized logic and cable delay cycles. The result is an increase in access time because the physical expansion of logic and cabling is not linear. The effect of interleaving, then, needs to be considered more than casually.

Peripheral storage control element (PSCE)

- *Design requirements*

The governing requirements for the PSCE were twofold. The first requirement was compatibility with other high performance models of System/360. The second requirement was the maximization of storage utilization. The

first requirement helped define the minimum number, types, and speed of I/O units that must communicate through the PSCE. This did not, however, define the maximum capability. The second requirement pointed out a need for something more elaborate than a switch to handle the congestion that could develop from an attempt to transfer data to and from extended main storage or high performance main storage.

Bench marks

The requirements were such that a set of bench marks was required to prove the superiority of any particular concept. Three bench marks were defined and used during the initial design phase for cost/performance evaluations.

Bench mark definition was difficult since little was known about all the likely applications or I/O configurations that would develop for the Model 91. Much was known of special applications and problems but little was known about the use of extended main storage. Before the bench marks were defined, an attempt was made to answer the following questions:

- How much I/O would a typical Model 91 configuration contain?
- What are the maximum data rates of the I/O units?
- With the availability of high performance I/O, what is the requirement for overlapping I/O?
- What is the chaining requirement for high performance I/O?
- What aggregate I/O rates would the PSCE be expected to handle?
- What is the storage range that the PSCE would be expected to handle?
- What is a typical size for storage configuration?
- Should storage be interleaved? If yes, what should the interleave factor be?
- What are typical random rates for processor activity?
- Should the Model 91 be able to share storage with peripheral processors?
- What is the minimum acceptable storage channel rate?
- What is the maximum transfer rate expected for the peripheral processor?

The questions were not simply resolved. It was difficult to put limits on any condition because the best performance was desired in all areas. It was possible, however, to develop a small number of reasonable alternatives, and the following bench marks were defined for comparing alternate PSCE designs:

I. Storage: 4 or 8 way interleave, 8 μsec cycle
   I/O*: 2 — 1.25 megabyte/sec devices
   1 — .150 megabyte/sec device
   Peripheral processor (PPE): 6.66 megabyte/sec
   Storage channel (SC)*: maximum rate

II. Storage: 4 or 8 way interleave, 8 μsec cycle
   I/O*: 2 — 1.25 megabyte/sec devices
   1 — .150 megabyte/sec device
   1 — 90 kilobytes/sec device
   PPE*: —6.66 megabyte/sec
   Central processor (CPU)†: maximum rate

III. Extended Main Storage: 4 or 8 way interleave
   I/O*: 1 — 1.25 megabyte/sec device
   1 — .150 megabyte/sec device
   1 — 90 kilobyte/sec device
   SC*: maximum rate

These bench marks were used to help select the design approach with the most potential. They were not used as ultimate objectives. Once a design concept was selected, simulation was used to help evaluate cost/performance.

Speed matching and other problems

The speed matching problems were as varied as the combination of interfaces and speed variations possible at each interface of the PSCE. The variations in the interfaces are due in part to the different storage technologies, storage hierarchies, bussing needs and circuit requirements that may exist for various system configurations. The storage hierarchies present many unique engineering problems in the area of the boundary detection which is used for interleaving, bus assignment, and storage protection.

One of the most difficult speed matching problems that the PSCE had to contend with was that of allowing high speed I/O to operate into a storage unit with a cycle time greater than the cycle time of the requesting I/O unit. This same mismatch exists for the processor and the storage channel (SC) but, since these units can wait indefinitely for service, they are not subject to overrun as are the I/O units. The traditional approach to the I/O problem has been to allow I/O units to have priority over any other element in a system. If the priority approach did not solve the problem, then it was usually necessary to bypass any bus in the path to storage and create an independent path for I/O. This approach had led to the development of "multi-tailed" storages.

Standard solutions were found to be inadequate because their implementation would only partially solve the I/O problem and still do nothing to improve the processor or SC rates. In the past, if the I/O rate into storage was such that it could not tolerate a conflict, the I/O would block the processor until the risk of overrun was past.

It was decided that any new solution would be acceptable only if it met the following requirements:

- CPU, peripheral processor and SC to have access to storage without the use of multi-input storage units.
- More than one high speed I/O channel to be able to use storage in an overlapped mode.

* Sequential addressing.
† Random addressing.

Table 1 PSCE bench mark evaluation.

Case	Bench mark	No. of Storage Units @ 8 μ sec	No. of buffers	SC[a] store or fetch	SC rate	CPU fetch rate	PPE[b] fetch rate	Storage utilization
1	I	8	12	All fetch	2.35	···	0.53	69%
2	I	8	12	All store	4.0	···	0.6	90%
3	I	8	12	50% fetch, 50% store	2.96	···	0.56	80%
4	I	8	16	All fetch	3.64	···	0.6	82%
5	I	4	12	All fetch	0.96	···	0.3	100%
6	II	8	12	···	···	2.35	0.6	70%
7	II	4	12	···	···	.5	0.42	88%
8	III	8	12	All fetch	5.7	···	···	86%
9	III	8	12	All store	6.7	···	···	100%
10	III	8	12	50% fetch, 50% store	6.2	···	···	93%
11	III	4	12	All fetch	2.7	···	···	100%

NOTE: All rates in megabytes/sec.
[a] Storage channel
[b] Peripheral processor (PPE)

- A processor request for a busy storage unit should not inhibit its ability to make other requests.
- Storage interleaving must be possible in order to improve the accessibility of requested data.
- No unit must resort to blocking storages in order to guarantee their availability at a later time.
- Due to the built-in overhead of a storage control unit, it must be able to handle requests in a pipeline fashion, a pipeline technique being one which allows concurrent execution of multiple operations while taking into consideration the serial dependence of the operations.
- The design must lend itself to growth and be able to adjust to different storage hierarchies.
- The design must be balanced to maximize the use of storage to all users.
- The design must be able to adjust to different storage configurations for proper boundary detection.

- *New concepts*

It was decided that the most promising concept for meeting the basic requirements was a bus organized around a buffer stack, or queue. The use of buffers was certainly not new but the manner in which they were to be used provided the flexibility and performance that was desired. The queue developed has the following operating characteristics:

- A variable number of queue positions are dynamically reserved for I/O inputs. The number reserved depends on the speed and number of I/O units in operation.
- The queue is used to store outstanding requests made by all users.
- Input requests to the queue are on a first come, first served basis except for simultaneous requests, which are handled in a fixed priority order.
- Output from the queue to storage is based on a three-level decision. The first decision level checks for available storage. The next decision level determines the unit that will have priority out of those requesting an available storage. The last decision level selects the first request for the unit getting priority.
- Output from the queue destined for channels is handled on a request basis. All other output (peripheral processor, CPU, and storage channel) is handled when no higher requests are outstanding.
- The queue can overlap all input and output operations. That is, at any point in time it can handle data returns from storage, two input priority requests, an output priority storage selection, and the return of a word to a channel and to the CPU.
- Outstanding requests in the queue may be handled out of sequence.
- Queue positions must be available for use by the processor and the storage channel when not reserved by channels.

- *Simulation*

The operating characteristics listed above were selected and developed for the PSCE only after a study of data obtained by simulating different bus designs. Simulation, based on the bench marks previously described, pointed to bottlenecks that would have caused an unbalanced bus under certain I/O configurations. The simulation results shown in Table 1 give an indication of probable storage utilization with a PSCE design of 12 queues working into an 8-μsec storage.

- *Queue design*

The decision to use a bus design with a shared buffer stack (queue) was made after studies indicated that all of the

proposed bus designs contained several buffers. However, the designs differed in the way the buffers were used and distributed among the individual control sections of the bus. It was found that, for approximately the same cost as designs with distributed buffers, it was possible to build a bus with a shared stack. It was necessary to build the registers of the central stack so that they could be used by all control sections of the bus. This required that they be more elaborate than would be the case if they had been designed to the specific requirements of one application. The increased complexity is more than balanced by the improved data rates that are possible for the processor and storage channel. The storage channel rates as a function of available queues and storage access time are shown in Table 2. (A more detailed explanation of these rates is given below.) It is obvious by looking at the table that the best storage channel rate is obtained by using all available queues. It was decided that 8 registers should be used in the queue because it was found that assuming a 3/4-μsec storage cycle and 4 channels with 1.25 megabyte devices, at least 6 registers were needed to handle the simultaneous operation of 6 or more channels.

Table 2 Storage channel best case transfer rates.

Number of queues available to storage channel	*Transfer rate, megabytes/sec* [b]*Access = 11 cycles*	[b]*Access = 10 cycles*
1[a]	7.74	8.35
2	15.7	16.7
3	23.5	25.0
4	31.4	33.3
5	39.2	41.7
6	47.1	50.0
7	54.7	58.3
8	62.5	66.6
0	SC Locked Out	

[a] For single word boundary, use this entry only.
[b] Access = Storage access time measured at PSCE tailgate for 3/4 μsec storage.

- *PSCE Organization*

The design of the PSCE merges several functions into one integrated unit. This organization consists of four major areas: queue and busses, queue priority, common channel controls, and storage channel (Fig. 8).

Queue and busses

The effectiveness of the queue depends in great measure upon its accessibility. The busses that communicate with the queue were planned with a goal of allowing simultaneous execution whenever feasible. These busses can be grouped in four main categories:

Unit entry busses There is a unit entry bus shared by all requesting units which provides access to all queues. In addition, two unit entry address busses are provided, one for the CPU and another shared by the other requesting units. The independent processor address path was provided because of the high incidence of fetch requests as compared to store requests and, since access to extended main storage depends in part upon access to the queue, the separate address path improves the overall access time.

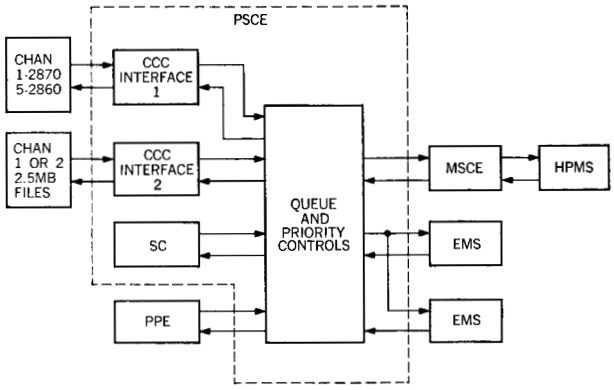


Figure 8 Block diagram of PSCE organization.

Unit return busses Two unit return busses are provided, one for the processor and another for all other requesting units. These busses are fed from each queue and return data to the requesting units. The separate CPU bus is included to minimize access time. In general, when a fetch is made, a queue register is held as a sink for the returning data. Data returns from storage are placed in the queue and can be sent immediately to the requesting unit or held until requested by I/O channels. The separate processor return bus allowed improvements in this procedure since, with the separate bus, the processor will not encounter conflicts with higher-priority unit returns. A further improvement was realized by utilizing the processor return bus in such a way that data returning from extended main storage could be sent directly to the processor without passing through the queue. This approach allows queues being used for fetches to be made available immediately after the fetch request is sent to storage. This means that fewer queues are required to handle any given processor request rate and more queues are available to other units.

Storage request busses Two storage request busses are provided, one for extended main storage and another for high performance main storage via the MSCE. These busses provide independent select paths to the two groups of storage and, since traffic to both groups can be high, these paths eliminate unnecessary inter-group conflicts which tend to increase access time.

Storage return busses Three independent storage busses

are provided in order to simultaneously handle up to three different storage access times. One is for returns from the MSCE and the others provide for up to two different storage speeds.

Queue priority

Queue priority can best be described in two parts, input priority and output priority.

Input priority In general, input priority is concerned with entry to the queue. The key functions performed are:

- Maintain dynamic queue availability status.
- Reserve queues for I/O channels upon request from the common channel control section of the PSCE.
- Assign unreserved queues to incoming requests based upon unit priority and queue availability.
- Assign reserved queues to I/O channel requests. A pre-priority function among individual I/O channels is performed by the channel controls which present a single request to input priority.
- Route returning storage data to the proper queues and update queue status as a result of these returns.

Output priority Output priority continually monitors the status of the queue in order to determine actions to be taken on the storage request busses and unit return busses. The following decision mechanism is simultaneously applied for each storage request bus:

- Compare available storage status with all requests in the queue in order to determine which request should access storage.
- If more then one request finds an available storage unit then select the highest priority unit among those requesting.
- If there is more than one outstanding request from a selecting unit, storage accesses are made on a first-in, first-out basis.

For the unit return busses the following actions are taken:

- Route memory returns to the central processor.
- Inform other units of returns available in the queue and return them on a first-in, first-out basis when requested.

Storage channel

The function of the storage channel is to provide fast data transfer from storage to storage, overlapped with other system activity. The storage channel operates as an independent unit with respect to the queue and is not treated as "just another channel." Communication delays encountered in conventionally independent I/O channels have been eliminated by integrating the storage channel with the PSCE.

Of the units that require access to storage, the storage channel was given lowest priority. I/O channels require higher priority because of their overrun nature; the peripheral processor, which may also have I/O channels operating, also requires higher priority; since CPU accesses imply an immediate need for storage and storage channel accesses imply a future or less immediate requirement, the CPU was also given higher priority. This decision was based on the fact that delayed processor access delays the system immediately while delayed storage channel access *may* delay the system in the future.

I/O activity and especially CPU activity will cause conflicts with storage channel activity. Every storage conflict will delay the storage channel and high interleaving of storage will help only to reduce the probability of such conflicts. However, if the storage channel can circumvent a current conflict and attempt its next access, the probability of a second storage conflict is considerably reduced. This, of course is the central philosophy of the PSCE, i.e., to permit units to access storage out of sequence in order to better utilize the high interleave and thereby increase both the overall rate of each unit and the effective use of storage.

The storage channel can initiate a fetch request every cycle and does so as long as space is available in the queue. Each queue register used becomes a sink for the data from storage. If this data were returned to the storage channel, it would eventually be sent back to the queue for storing in memory. It is not desirable to permit this round trip of data from queue to channel and back, because it would add handling time and buffering requirements to the storage channel. Instead, the data remain in the queue and a mechanism is provided to bring the store address to the queue.

The fact that fetches are made out of sequence from the queue implies that these store addresses would have to be supplied out of sequence in order to complete the data transfer as quickly as possible. Generation of out-of-sequence store addresses entails extensive address buffering and sequence controls, and a different mechanism, called re-address, is used. When the storage channel sends a fetch request to the queue, the data field of the queue register is "empty." In addition to the normal entry of the fetch address, the store address for that word is generated and placed in the "empty" data field. Whenever the fetch address is sent from the queue to storage, the store address is moved from the data field to the address field where it waits for the data to return from storage. As soon as the data return the store can be made, thereby allowing storage channel stores to be made out of sequence as well as fetches.

Since queue availability is a decisive factor in storage channel transfer rates, any factor which tends to increase queue availability also tends to increase this rate. As storage access time decreases, queues become available more quickly because the total fetch-store time is decreased. Table 2 shows the effect of the access time of the 3/4-μsec storage unit on storage channel transfer rates.

As system activity into storage increases, storage conflicts also increase and the number of queues available becomes more important in order to provide some minimum storage channel rate. Table 3 illustrates this fact by showing estimated transfer rates of the storage channel as a function of queues available and of storage conflicts, assuming a ten cycle access time. It should be noted that since a simple algorithm was used, these rates are only representative, but they do serve to illustrate the point.

Common channel control

The common channel control (CCC) uses one interface to provide for the attachment of up to five IBM 2860 Selector Channels plus one IBM 2870 Multiplexor Channel to the Model 91. The Selector Channels communicate directly with the CCC which in turn communicates with the input and output control sections of the PSCE. The CCC also has the potential for attachment of two very high performance channels (2.5 megabyte/sec rate) through a second interface. This interface is designed to minimize the communication time required to service a channel request. It is expected that the time required to service a channel will be reduced from 1 μsec for the standard interface to approximately 0.2 μsec for this second interface.

The CCC will accept a modified Selector Channel interface. The change in the interface was made to provide buffer control for channels that control devices with rates $\leq$1.25 megabytes. The changes allow words to be returned to different channels in a sequence different from that in which the requests were generated. This permits an aggregate channel rate which is higher than is normally possible over the standard Selector Channel interface. In addition, the changes permit the CCC to pre-fetch data for 1.25 megabyte/sec devices. Pre-fetching permits the overlapping of storage access time with channel service time. This overlapping allows the CCC to control the operation of four IBM 2860 Selector Channels with 1.25 megabyte/sec devices (no data chaining) into a 3/4 μsec. memory without the risk of overrun. Without pre-fetching it would only be possible to run two Selector Channels with the same restrictions stated above.

- *PSCE serviceability*

A feature included in the design of the PSCE permits it to be operated in a mode that does not depend on the availability of I/O equipment, storage or processors. The PSCE queue and controls can be exercised in a loop to repeat particular patterns. A separate maintenance panel is provided for the special PSCE maintenance features. The panel also permits maintenance on the PSCE to be overlapped with maintenance of other parts of the CPU. Serviceability of the PSCE is enhanced by these features:

- Variable effective queue size permits failures to be isolated.

Table 3 Storage channel transfer rates (assuming 750-nsec extended main storage and 750-nsec high performance main storage.

No. of queues available to storage channel	*Transfer rate, megabytes/sec*		
	Case I	*Case II*	*Case III*
1[a]	8.3	6.0	4.8
2	16.6	11.9	19.5
3	24.9	17.5	13.7
4	33.3	23.0	17.0
5	41.6	28.4	21.7
6	49.8	33.6	25.4
7	58.3	38.6	29.0
8	66.4	43.5	32.4

CASE I: Probability of EMS busy = 0
Probability of HPMS busy = 0
CASE II: Probability of one of EMS or HPMS busy = 1/2
Probability of both EMS and HPMS busy = 0
CASE III: Probability of both EMS and HPMS busy = 1/2
Probability of both EMS and HPMS not busy = 1/2.

[a] Use this entry for addresses on single word boundaries.

- Queue contents on stores can be saved until storage advance time. This allows correlation of storage-detected errors with the contents of the queue register that generated the storage select.
- A register in the queue can "freeze" its contents on error. This feature allows all data associated with a request to be retained for future display or log out.

PSCE-MSCE interaction

It should be noted that the queue buffers are general purpose whereas the processor request stack in the MSCE has no provision for data bits, uses the processor data buffers, and is tailored to processor requests. Consequently, the PSCE buffering capabilities (8 queue positions) are utilized by both PSCE-to-main storage requests and by processor-to-extended storage requests. PSCE-to-main storage requests appear at the MSCE from the queue only after the specific storage unit has been reserved and becomes available for selection. Thus, two-way communication, on a main storage interleave-factor basis (4, 8, or 16), exists between the MSCE and the PSCE for PSCE-to-main storage requests. In other words, the PSCE monitors the state of each main storage unit.

Processor-to-extended storage requests can appear at the PSCE at any time (provided queue positions are available), independently of the immediate availability of the desired extended storage. Again, the PSCE buffering capability is utilized by stacking processor requests in the queue until service time. In effect, then, the processor requests monitor the state of the queue, rather than the

state of extended storage units. Processor-to-extended storage requests are (1) transmitted across the interface, (2) buffered in the queue, (3) transmitted to priority controls, and (4) permitted to select extended main storage. Conversely, PSCE-to-main storage requests (1) are buffered in the queue (2) enter priority (3) are transmitted across the interface, and (4) select main storage. Both types of requests use the queue as a buffer and enter priority only after the requesting address enters the queue.

The MSCE and the PSCE are synchronized to receive fetched data from each other on any cycle, although the control technique is different in each control element. The PSCE has unique data paths for main storage data returns and extended storage returns to the queue. The MSCE accepts data from extended storage by orthogonally multiplexing the main storage and extended storage.

Acknowledgments

The design of the storage system was a group effort with many contributors. In particular Messrs. M. C. Dales, S. A. Calta, H. A. Carlson, A. Gomez, L. W. Kaumeyer, J. Kloepping, and J. V. Mizzi contributed in the early phases of design. Also, the instruction unit design group gave valuable suggestions and criticisms. The simulations mentioned were developed by Mr. P. S. Cheng and Mr. J. R. Johnson.

Received November 1, 1965.