S/390 Parallel Enterprise Server Generation 3: A balanced system and cache structure

by G. Doettling
K. J. Getzlaff
B. Leppla
W. Lipponer
T. Pflueger
T. Schlipf
D. Schmunkamp
U. Wille

Since initiating the information technology industry-wide transition from bipolar to CMOS technology with the first generation of S/390® processors in 1994, IBM reached another major milestone with the introduction of the third generation in September 1996. The balanced system and cache structure and the modularity of the components of Generation 3 support a wide performance range from a uniprocessor to a high-performance multiprocessing system. Because of this modularity, Generation 4 is also based on this structure.

Introduction

The IBM S/390* Parallel Enterprise Server Generation 3 and the IBM S/390 Multiprise* 2000 (both called G3) and the later G4 systems are tightly coupled S/390 symmetrical multiprocessing systems with up to ten processors, a three-level cache hierarchy, and up to 8 GB (for G3) of physical memory. The modularity of the system and cache structure supports a wide performance range based on the same chip set. The G3 spans a range from a uniprocessor with one memory card to a high-end system with ten processors and four memory cards. There are either one or two additional processors in the system structure, which serve as system assist processors (SAPs) for I/O operations. The balanced system and cache structure design makes it possible to retain all elements unchanged for G4, except for replacing the PU (processing unit, or processor) and L2 chips. First an overview of the G3 system and cache structure is given, followed by discussions of the implementation and the features of the PU and L2 chip and the common G3/G4 parts (BSN, MBA, STC/memory card, and CGC). Special focus is given to the error-detection and recovery capabilities of the G3 system.

G3 system structure

The 12-way four-bus system structure and components of the chip set are shown in **Figure 1**. The chip set consists of processor units (PUs), level-2 caches (L2s), bus-switching network adapters (BSNs), storage controllers (STCs) on the memory cards, I/O adapters (MBAs), and

0018-8646/97/$5.00 © 1997 IBM

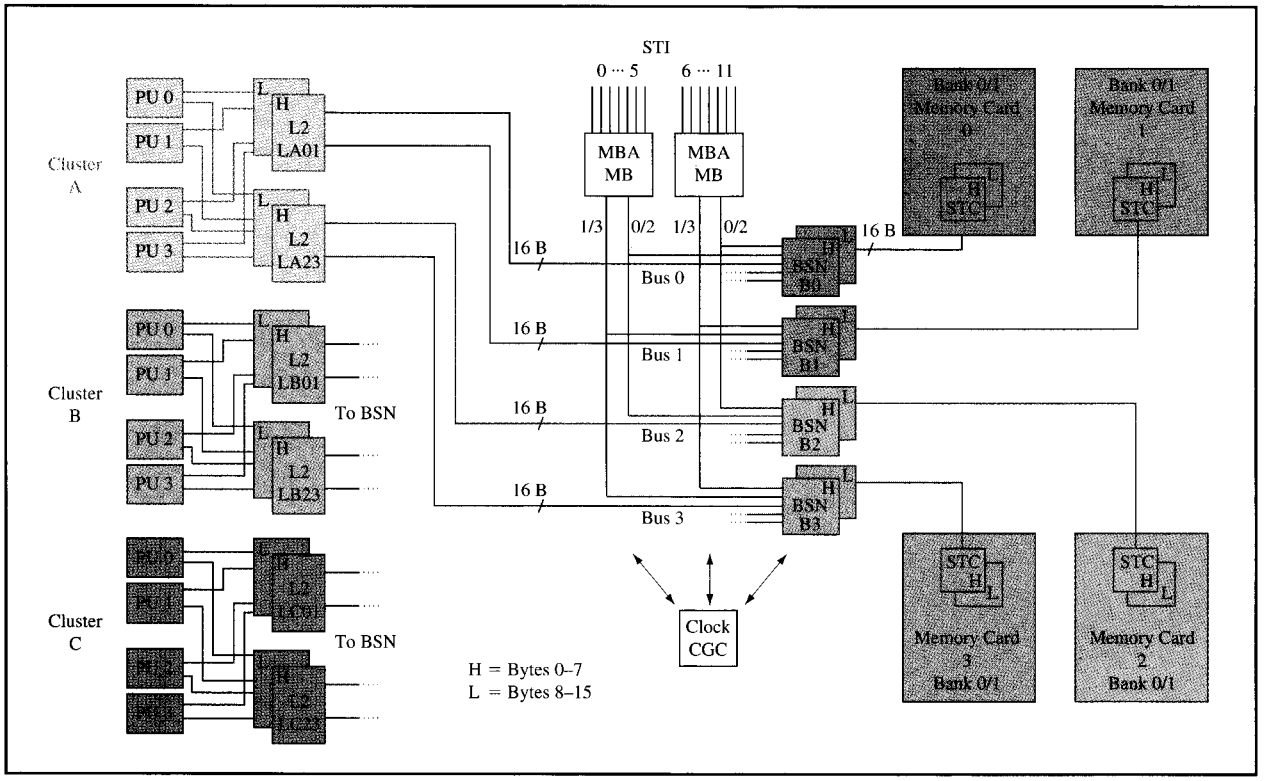


Figure 1

G3 system structure.

the clock chip (CGC). A level-1 cache is included on the PU, and a level-3 cache (called L2.5) on the BSN.

All system buses are 16 bytes wide and bidirectional; L2, BSN, and STC are implemented in pairs of identical chips referred to as the high (H) and low (L) chip because of constraints on chip pad area and silicon area. In the G3 system, a 1:1 ratio between the processor cycle and the bus cycle is reached. The cycle time is 5.9 ns in the high-end system, while the smaller one- or two-bus systems run with a relaxed cycle time.

In the four-bus system the PU, L2, BSN, and MBA chips are packaged on a 127-mm air-cooled multichip module (MCM). **Figure 2** shows the chip placement on the MCM, which is optimized for minimal bus wire length. The 127-mm MCM uses 20 pairs of wiring planes and has a total of 1732 signal module pins. The CGC chip and the STC chips are packaged on single-chip modules (SCMs). Two STC chips are placed on a memory card. The MCM, the CGC chip, and the memory cards are placed on a planar board.

The one- and two-bus systems have a card-on-board (COB) package with all MCMs and SCMs mounted on one card. Four PU chips and two L2 chips are assembled on a 63-mm MCM, and the BSN, MBA, and CGC chips are packaged on SCMs. **Table 1** presents an overview of the capabilities of the chip set to support different system configurations.

Cache structure

The memory subsystem consists of a three-level cache hierarchy and the memory. This cache structure is common throughout the low-end, intermediate, and high-end systems, differing only in cache size and number of buses. The line size for all hierarchy levels is 128 bytes.

- *Level-1 cache (L1)*

The level-1 cache is integrated on the PU chip and is implemented as a unified instruction and data cache. The size is limited to 32 KB to fit on the PU chip and to achieve a one-cycle access. To resolve L1 cache misses, a line request is sent to the private L2 with the address of the missing quadword (16 bytes). This quadword is returned first to allow the PU to proceed immediately. The remaining seven quadwords follow and may wrap within a cache line. The L1 cache is parity-checked. Intermittent errors (soft errors, etc.) are recovered by

Table 1 System configurations.

System	*Four-bus*	*Two-bus*	*One-bus*
Number of PU chips	2–12	2–8	2–4
L2 size per PU	256 KB	128 KB	64 KB
Number of BSN chips	8	4	2
L2.5 size	2 MB	1 MB	512 KB
Number of MBA chips	2	1	1
I/O path	12	6	3
Number of memory cards	4	2	1

reloading the defective line from the L2. The L1 cache includes the following features:

- Store-through concept.
- Size: 32 KB.
- Eight-way set-associative.
- Parity-checked with recovery.
- One-cycle access.
- 16-byte access.

- *Level-2 cache (L2)*

The L2 cache chip comprises four independent cache partitions, each with a size of 64 KB. They can operate concurrently, and each is assigned to one PU to serve as a private L2 cache (see **Figure 3**). The chip is implemented with an 8-byte dataflow and interface. A pair of L2 chips work synchronously in the 16-byte bus structure, providing four PU ports and two bus ports for the L2–BSN bus connection. A PU has a 16-byte bus to each of the L2 pairs and routes the cache-line requests according to the defined address class scheme.

The L2 is a store-in cache. It always holds the actual copy of a line, because its L1 cache stores through. The L2 cache performs bus snooping on all L2–BSN buses to maintain the data coherence within the L1- and L2-cache-level hierarchy. The L2 cache includes the following features:

- Private cache.
- Store-in concept.
- Size: 256 KB per PU.
- Eight-way set-associative.
- ECC-checked.
- Five-cycle access.
- 16-byte access.

- *Level-3 cache (L2.5)*

The L2.5 cache is a shared cache [1] serving all PUs. It is implemented on pairs of BSN chips. A pair of BSN chips performs the required bus arbitration and bus switching, and provides the L2.5 cache with a size of 512 KB per bus. The L2.5 cache contains two banks, which can operate concurrently; its features include

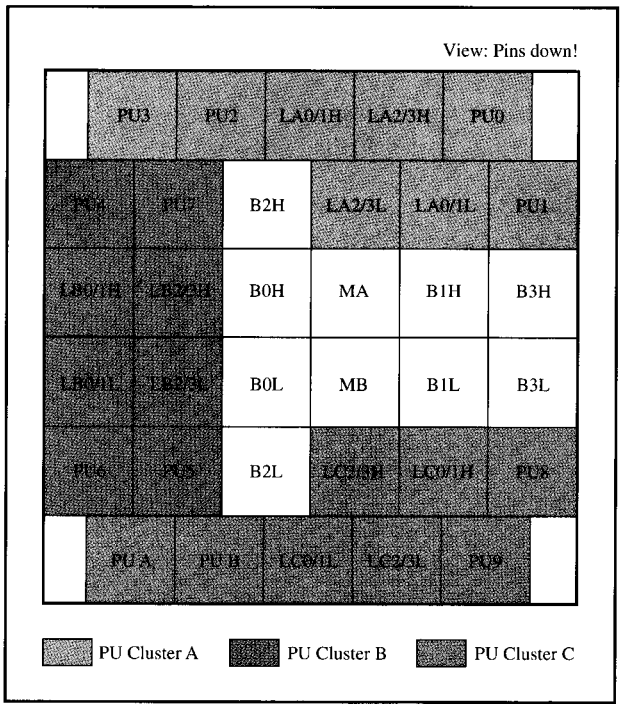


Figure 2

G3 multichip module (127 mm).

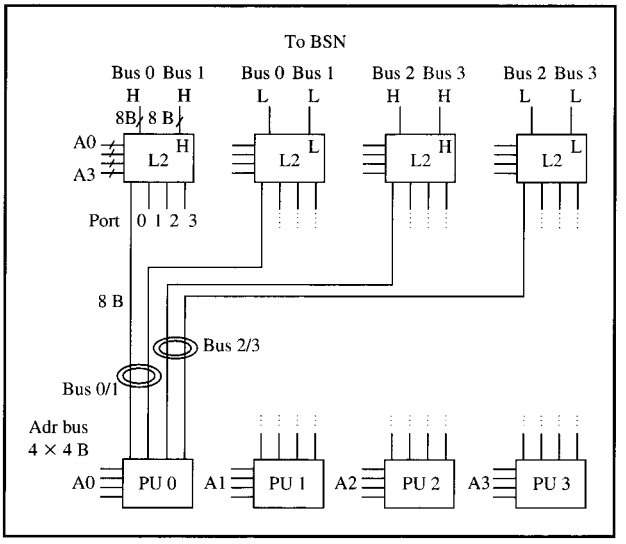


Figure 3

PU–L2 cluster.

- Shared cache.
- Store-through concept.
- 512 KB per bus, 2 MB maximum.
- Eight-way set-associative.
- ECC-checked with deletion feature per bank.
- 14-cycle access.
- 16-byte access.
- Two independent banks.

- *Memory*

The memory is implemented with DRAMs on memory cards, and is controlled by a pair of STC chips which are located on the card and run synchronously. A memory card contains two memory banks, which can operate concurrently; one to four cards may be used depending on the system configuration. The S/390 key storage is implemented on the memory cards. Each card contains the full set of key entries to avoid bus interference for combined data and key operations. The memory card features include

- Shared-system memory.
- Size: up to 2 GB per card.
- One to four cards depending on the configuration.
- ECC-checked, with redundancy bits.
- 32-cycle access.
- Two independent banks.

Bus structure

The bus structure of a high-end system (Figure 1) comprises four logical buses numbered 0 to 3. Each of these buses is controlled by a pair of BSN chips [2]. A "logical" bus can consist of up to three "physical" buses to connect the PU–L2 clusters to a pair of BSN chips and two additional buses to connect the MBA chips. Each of the pairs of BSN chips controls one memory bus, with a pair of STC chips located on the memory card. All system buses are 16 bytes wide, parity-checked, and bidirectional, and are connected by pairs of L2, BSN, and STC chips. The synchronous operation of a pair is checked every cycle. Command and address are duplicated on each half of the bus during the command/address cycle to allow both chips to run synchronously.

- *PU–L2 cluster*

PU and L2 cache chips are grouped in clusters (A, B, C). Each cluster contains one to four PUs and one or two pairs of L2 chips (Figure 3). A PU owns a 64KB cache partition on each of the four L2 chips, giving a total of 256 KB per PU in the four-bus system. The cluster contains two L2 chips in the one- or two-bus system, with a total of 64 KB or 128 KB per PU. A pair of L2 chips has four PU ports, each with a 16-byte data bus and a 4-byte address bus. These are private buses with a simple handshaking protocol (PU–L2 bus). They are used by the PUs

- To request cache lines from the L2 with the address put on the address bus.
- To perform store-through operations with the address put on the address bus and a 16-byte datablock on the data bus.
- To communicate with other PUs, MBAs, or memory via the L2 cache.

- *Logical L2–BSN bus*

A pair of L2 chips contains two bus ports, each 16 bytes wide. These are the L2–BSN buses, which connect the PU–L2 clusters with pairs of BSN chips. The four-bus system contains two pairs of L2 chips within each cluster, providing a connectivity of 4 × 16-byte buses to the memory via the BSN chips. A simple routing scheme is used to select the bus and bank for a memory access. Bits 22 and 23 of the line address define the bus number, and bit 21 selects one of the two memory banks. This maps the cache lines with the low-order line address 0 and 1 in the first memory card, cache lines with the line address 2 and 3 in the next card, and so on. This fine granularity provides an equally distributed load on the buses.

The L2–BSN buses are shared between the PUs and the MBAs and are controlled by pairs of BSN chips with a simple handshake protocol. There is no separate address bus available; command/address and data are multiplexed on the same bus. The BSN controls access to the bus, granting access to one of the PUs or to the MBAs. It redrives the command/address cycle to the memory via the STC and to the other PU–L2 clusters to allow bus snooping.

A cache line is transferred in eight datashots of 16 bytes each; it is redriven to the STC for line-store operations and to the PU–L2 clusters for line-fetch operations. The L2–BSN bus supports two-way interleaving by using the latency between the command/address cycle and the first data transfer cycle for line-fetch operations. In this gap another command/address cycle can be issued to the other bank of this bus, utilizing the two independent banks of the L2.5 cache and the memory. A pair of BSN chips has four L2–BSN bus ports to support a maximum of four PU–L2 clusters (three are used in the G3 system), two MBA–BSN bus ports, and one BSN–STC bus port to connect the STC chips and the memory.

- *BSN–STC bus*

The BSN–STC bus is controlled by a pair of BSN chips. The command/address format and the basic protocol are similar to those of the L2–BSN bus, since the BSN simply operates as a switch between them. There is an internal latency of two cycles for the command/address cycle and

Table 2 L2 cache bus-snooping response.

Operation	*Old state*				*New state*				*Cast-out*	*Memory update*
	M	*E*	*S*	*I*	*M*	*E*	*S*	*I*		
LF–DFETCH				x				x		
			x				x			
		x					x			
	x						x		x	x
LF–DSTORE				x				x		
			x					x		
		x						x		
	x							x	x	

Cache-line status: *M* = Modified, *E* = Exclusive, *S* = Shared, *I* = Invalid.

for a data-transfer cycle. (This bus works in interleave mode as well.) The bus protocol allows the cancellation of line-fetch operations that have already been started without knowing whether the line is in the L2.5 cache.

Bus operations

- *Bus commands*

The bus structure connects all components of the system; it provides communication among them and allows access to internal objects such as the key storage on the memory card or to I/O adapters via the MBA. The bus operations can be broken down into eight command groups:

- Line fetch.
- Line store/line fetch.
- Cast-out.
- Line-invalidate.
- Absolute fetch/store operations with 1–8 bytes.
- Key-storage operations.
- Sense/control operations.
- Fetch/store MBA with 1–128 bytes.

There are several line-fetch commands; they specify whether the line is requested for a fetch or store operation (LF–DFETCH or LF–DSTORE), an instruction fetch (LF–IFETCH), or the key from the key storage is requested together with the data (LFK). The line-store/line-fetch command combines an LRU cast-out with the line fetch to free up an entry in the L2 cache. This LRU cast-out is also a separate command, used for cache-to-cache cast-out operations as well (cross-cache cast-out). A line-invalidation command is signaled from one PU to all other PUs via the L2 cache to become the single owner (exclusive state) of a line. Absolute fetch/store commands allow access to the memory, bypassing the caches. They are used for internal functions. S/390 key-storage commands perform read/write operations to a key-storage entry or modification to the reference and change bits.

Sense/control commands are used for many internal purposes. They permit the exchange of control and status information among the PU and L2, BSN, MBA, or the STC and are also used for communication between PUs via the L2 caches.

MBA fetch/store commands provide direct memory access for the I/O adapters, connected to the MBA with self-timed interfaces (STIs). This access is monitored by the L2 caches to maintain the data coherence within the memory subsystem.

- *Bus arbitration*

Centralized bus arbitration is performed by the BSN chips for each bus and bank. It prioritizes the access of the MBA and keeps the access to the banks in balance.

- *Cache coherence*

Data coherence within the memory subsystem is controlled by the L2 caches using a bus-snooping mechanism. The snooping is limited to the line address class of one logical bus; there is no interference with other buses. Line address hits during bus snooping may cause the L2 cache(s) to change the status of the line in their directories or to cast out a modified line to the requestor and to the memory for update. **Table 2** shows the actions of L2 caches for line fetch due to fetch (LF–DFETCH) or due to store (LF–DSTORE), depending on the state of the cache line. The implemented scheme follows the MESI protocol.

The snooping function requires the command/address information from all L2 caches of one logical bus. The BSN receives the command/address from the requestor on the L2–BSN bus and distributes it to all other PU–L2 clusters with one cycle of latency on each of the other physical L2–BSN buses. Hits in a private L2 cache do not require cross-cache communication. The advantage of this concept is that there is no need for a central control element or central directory copies; it therefore allows concurrent operations on four buses. No extra bus cycles

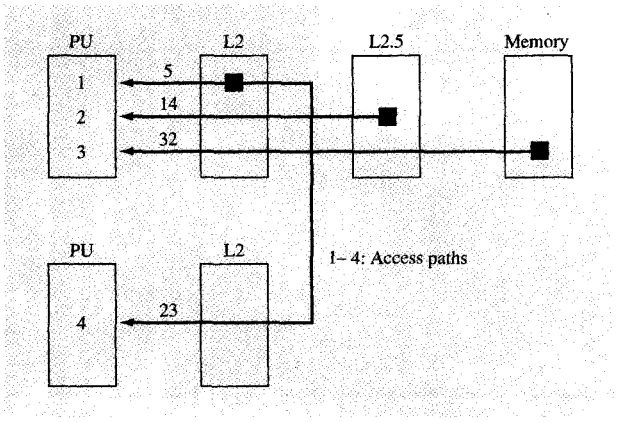


Figure 4

Line-fetch access paths with the latency in processor cycles.

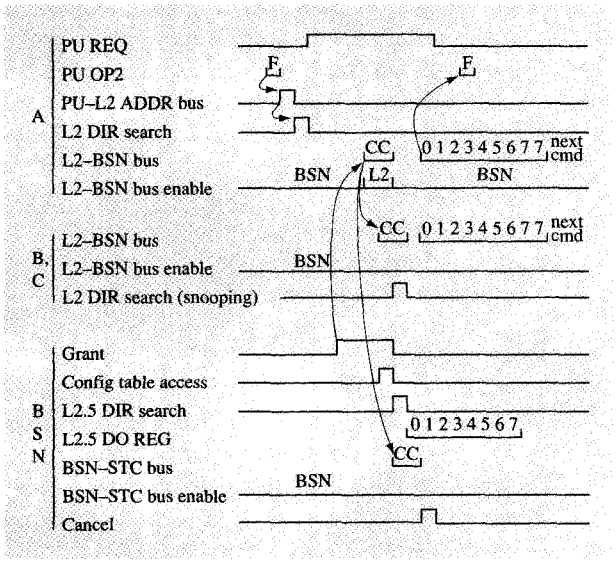


Figure 5

Bus timing of line fetch (A) with hit in L2.5 for a shared line.

are spent in the critical access path, since the bus snooping is done in parallel.

- *Line-fetch operation*

The execution of a line-fetch operation depends on the type of line-fetch command and on the state of the line in the private L2 cache, in the other L2 caches, or in the L2.5 cache. **Figure 4** shows the different sources and paths (1 to 4) for a requested line and the minimum access times in processor cycles (including the repetition of the failing instruction). The possible paths are as follows.

Path 1 The L2 cache has a hit and provides the line, usually within five cycles.

Path 2 The L2.5 cache has a hit and returns the line in 14 cycles if it is "shared" or the command is a line fetch for instructions (LF–IFETCH). Those lines are shared by definition. Otherwise, the L2.5 cache must wait for the result of bus snooping, because the line may be modified in one L2 cache; eight or more waiting cycles are added in this case. The line is also stored in the L2 cache.

Path 3 There is no hit in the L2 or L2.5 cache. The memory returns the line in 32 cycles or more, depending on its availability. The memory also provides the line for a line fetch with key command (LFK), regardless of a hit in the L2 or L2.5 cache, because the key from the key storage must be delivered. The line is also stored in the L2 and L2.5 caches.

Path 4 Another L2 cache has the requested line in "modified" state and returns it in 23 cycles or more, depending on its availability. During this cross-cache cast-out, the line is put on the L2–BSN bus for the requestor and on the BSN–STC bus for a memory update. This update is not done for a line fetch that is due to store (LF–DSTORE), because the requestor changes the line again immediately. The line is also stored in the L2 and L2.5 caches.

- *Example of bus timing*

An example of bus timing for a line-fetch operation for a shared line with an L2.5 cache hit is shown in **Figure 5**. A PU from PU–L2 cluster A gets an L1 cache miss for a fetch operation (F). The failing address is sent to the L2 cache, which performs an L2 directory search (miss) and raises the request line to the BSN. The BSN accepts the request and gives the bidirectional L2–BSN bus to the L2 cache (L2–BSN bus enable = L2). The requesting L2 cache puts command and address (C) on the L2–BSN bus, and the BSN redrives command and address on the L2–BSN buses for the PU–L2 clusters B, C. The L2 caches in clusters B, C search their directories in the next cycle (bus snooping).

The BSN accesses the configuration table (see the related section) and sends the command and the physical address via the BSN–STC bus to the memory to start the memory operation in parallel. The BSN searches the L2.5 directory in the same cycle and fetches the first quadword from the L2.5 cache. The BSN gets the hit and the shared state for this line from the L2.5 directory and immediately

puts the first and the following datashots on the L2–BSN buses (L2–BSN bus enable = BSN). The BSN also cancels the memory operation.

Command/address (C) and the last datashot (7) on the bus are always driven for two cycles in order to quiesce the bus. Finally the requesting L2 drops the request with the first data cycle and sends the data with one cycle of latency to its PU. The PU repeats the failing instruction (F) and continues with processing while the remaining data blocks are transferred.

- *Storage access penalty*

The storage access penalty is the latency between the data request to the cache hierarchy and the first returned 16-byte data block, plus the time required to repeat the missing instruction. The L1 cache access of one cycle is counted as part of the instruction processing.

Table 3 shows the cache sizes, the access penalties, and the cache hit rate for each level of the cache hierarchy for a ten-way and a five-way system. The penalty is given in processor cycles of 5.9 ns, the hit rate in percent per instruction for a typical S/390 commercial workload with heavy memory and bus load. Both systems have the same L1 cache size with a hit rate of 89%, so 11% of the storage references must be resolved by the following cache hierarchy or memory; 5% of the remaining references are covered by the L2 cache (4% in the five-way), 3% by the L2.5 cache, and 3% by the memory (4% in the five-way). Both systems have almost identical hit rates, providing equally balanced bus and cache behaviors. On the basis of the results of Table 3, three design points were considered: a cache hierarchy 1) with L1; 2) with L1 and L2; 3) with L1, L2, and L2.5.

An arbitrary scale is used for the storage access penalty (see **Figure 6**), with the L1 cache-only design normalized to 1. The storage access penalty decreases rapidly from point 1 to point 2 and moderately from point 2 to point 3. The diagram illustrates with this sample the efficiency of the three-level cache hierarchy and the benefit of the L2.5 cache.

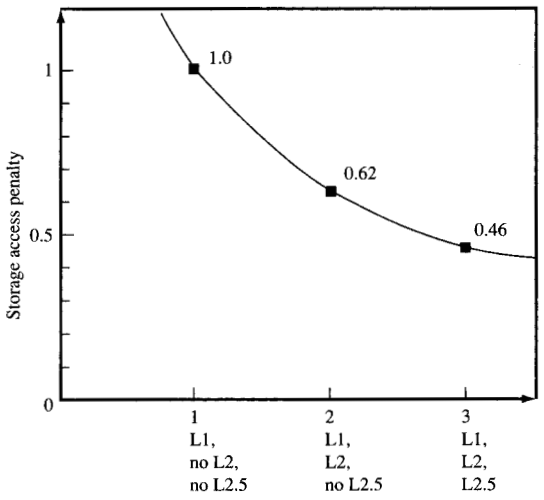


Figure 6

Storage access penalty improvement by the cache hierarchy levels.

- *Bus bandwidth*

The bus bandwidth determines the dynamic part of the storage penalty. An insufficient bandwidth of the memory subsystem increases the latency from the processor's viewpoint, because the queueing on the bus leads to more waiting cycles. This bus structure minimizes the impact by utilizing

- 16-byte-wide buses.
- Up to four concurrently operating buses.
- Two-way interleaving on each bus.
- Low bus cycle time.
- Bus protocol.

The bus protocol avoids unnecessary waiting cycles once a command has been granted by the BSN arbiter and issued by an L2. The command is routed immediately to the L2.5 cache and further to the memory. Bus snooping by the

Table 3 Typical cache hit rate in a ten-way/five-way system.

Memory subsystem	*Access penalty* (PU cycles)	*Ten-way, four buses*		*Five-way, two buses*	
		Cache size	*Hit rate* (%)	*Cache size*	*Hit rate* (%)
L1 cache	1	32 KB	89	32 KB	89
L2 cache	5	256 KB/PU	5	128 KB/PU	4
L2.5 cache	14	2 MB	3	1 MB	3
Memory	32	8 GB max	3	4 GB max	4

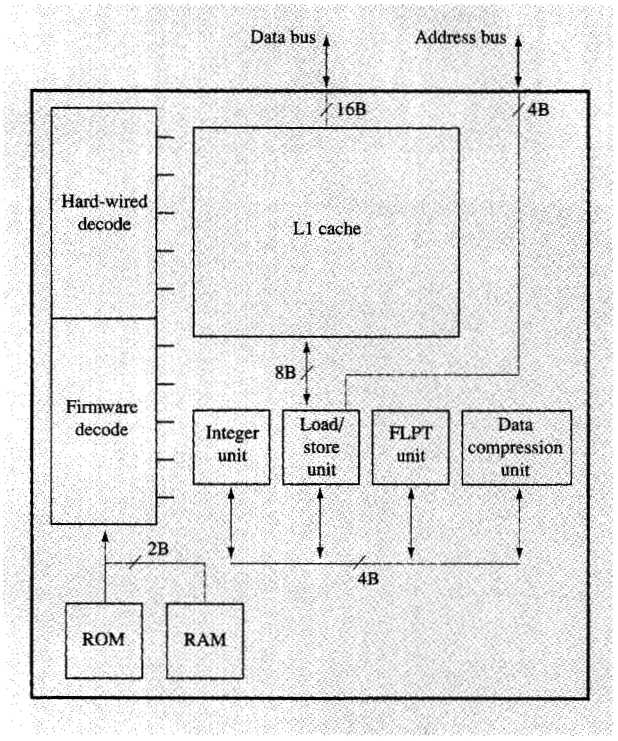


Figure 7

Processing unit.

other L2 caches and the L2.5 directory lookup are done in parallel and may cause a late cancellation of the memory operation.

The peak data rate on the L2–BSN bus is reached with permanent requests for shared lines and L2.5 cache hits; 256 bytes can be transferred in 26 bus cycles, including the command/address cycles, the access cycles, and the eight data cycles per cache line. This results in a data rate of 1.67 GB/s per bus, or a total of 6.68 GB/s for a four-bus system at 5.9-ns cycle time.

A bus-timing sequence is given in Figure 13 (shown later) with a line fetch from memory interleaved with a line fetch from the L2.5 cache at a peak data rate of 1.4 GB/s per bus. This example shows the bus interleaving effect. Overall, a four-bus system will provide a sustained throughput of 5–6 GB/s. Calculation of the bus utilization for typical S/390 workloads shows a utilization of 30–40% for the L2–BSN bus and 20–30% for the BSN–STC bus.

- *Bus cycle time*

All system buses run at the cycle time of the processor. This avoids additional latency in the PU or L2 chip for cycle-time adaptation and improves the bus bandwidth. This is achieved by the point-to-point characteristic of the buses (with the exception of the MBA–BSN bus). Optimized chip placement on the MCM provides balanced wiring density and short bus wire length. The bus wire length on the MCM does not exceed four chip spaces. The module was carefully wired with min/max wiring rules for each net in order to reach the bus cycle time.

Processing unit (PU)

- *Overview*

The processing unit (**Figure 7**) supports the S/390 architecture instruction set [3]. The most frequently used commercial instructions (108) and all floating-point instructions (54) are hard-wired. The remaining instructions are executed by the firmware.

The features of the PU chip include the following:

- 14.5-mm chip size, CMOS 5X, 7.2 million transistors.
- 5.9-ns cycle time.
- 744 chip I/Os used.
- 32KB unified cache (L1).
- 32KB RAM and 32KB ROM for firmware.
- Floating-point unit.
- Data-compression unit.
- S/390 timers.
- Two 16-byte-wide bidirectional buses to the L2 for data transfer and 4 × 4-byte address buses.
- System measurement instrument (SMI) to support performance verification.
- Trace buffers for system debug.
- Escape logic to support system debug.
- Serial link to the clock chip.

- *S/390 implementation*

Addressing and dataflow are implemented in 32-bit CISC processors. As in RISC processors, 108 instructions are executed by hardware, with a four-stage pipelining concept. The pipeline stages are *i-fetch*, *decode*, *execution*, and *writeback*. If an instruction requires more than one execution cycle, it is not sent for writeback until decode and execution are complete.

The i-fetch obtains data from the L1 cache if no other cache operation takes place. The fetched instructions are routed to two 16-byte instruction buffers. Address bit 27 of the virtual instruction address selects which buffer is loaded. To get the instruction buffers filled, an indicator is set if one of the two buffers is empty and a request for reload is raised. The reload is started if the instruction in the decode stage will not use the cache in the execution stage. The decode stage performs the address calculation and operand register read for the S/390 instructions. It loads and starts the firmware program for instructions that are not hard-wired. Firmware and hard-wired instructions such as integer and cache operations are processed in the

execution stage. Additional hardware is implemented to speed up firmware programs. The results from the execution stage are transferred in the writeback stage either to an architectured register or to the L1 cache. This instruction implementation leads to an infinite-cache cycles-per-instruction (cpi) performance of 2.4 in a typical commercial environment.

- *L1 cache*

Since most of the S/390 instructions have a data storage reference, it is important that the L1 cache can be accessed within one cycle. Access to the translation lookaside buffer (TLB), the L1 directory, and the L1 cache is done in parallel. The L1 is an eight-way set-associative cache with store-through and is addressed with the low-order real address bits. In the case of a cache miss, the requested cache line is transferred from the L2 in eight shots of 16 bytes to the L1 fetch buffer (see **Figure 8**). The instruction which caused the cache miss is continued after the transmission of the first 16-byte shot from the L1 fetch buffer into the L1 cache. During the processing of the following S/390 instructions, the remaining data from the L1 fetch buffer are copied to the L1 cache if they are not being used by instruction processing. Store data and L1 fetch buffer data are merged if the quadword addresses are the same.

Systemwide data integrity is maintained by a broadcasting mechanism between the L1 and L2 caches. After each instruction or between the execution of units of operations, the PU is interruptible by a broadcasting request to update the L1 cache directory.

Virtual addresses have to be translated into absolute addresses. Both are kept in a four-way set-associative TLB with 64 entries in each set, together with the S/390 storage key and protection information. In the case of a cache miss, the TLB provides the 4KB page address to form the complete 35-bit absolute address.

- *Firmware*

The firmware is used to execute complex instructions and the interrupt handling of the PU. It resides in RAM, ROM, or the memory DRAM subsystem. The addressable firmware region is 128 KB in size; 64 KB are held in the RAM and ROM, and the remaining part is transferred on request from the memory DRAM subsystem to a 128-byte register inside the PU. The firmware instructions have a vertical format; each instruction has a size of 2 bytes. The pipe depth is the same as for the hard-wired S/390 instructions.

- *Floating-point*

The main floating-point dataflow consists of an add/subtract flow with 116-bit width, and a signed multiplier of 60 × 60 bits. All floating-point operations except for divide, square root, and extended multiply require only one cycle in pipelined mode. Synchronization for instructions which require more than one cycle is done by the floating-point interface, which stalls the PU. A zero-cycle branch is available in conjunction with the floating-point unit, which speeds up the loop processing.

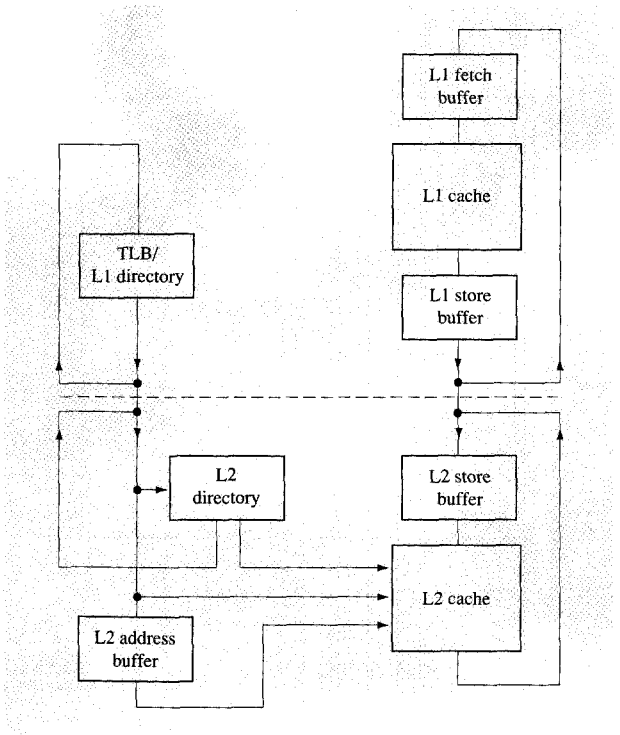


Figure 8

L1/L2 dataflow.

- *Data compression*

Data compression is based on a Lempel–Ziv algorithm and is fully implemented in hardware. Expansion of 1 byte takes two cycles, and eight cycles are needed for compression.

- *Error detection and recovery*

To maintain data integrity throughout the system, all data paths, including external buses, are parity-checked on byte boundaries. The multiplier in the floating-point unit includes a residue-checking scheme. The occurrence of a parity check or residue check within a PU will check-stop this PU and propagate the check-stop state to the clock chip and the correspondent L2. The defective PU is fenced "on the fly," and the remaining system, including all L2 caches, continues with processing, thus enhancing availability to the user. Soft errors in large arrays are also detected by parity checkers. In the case of an error, the

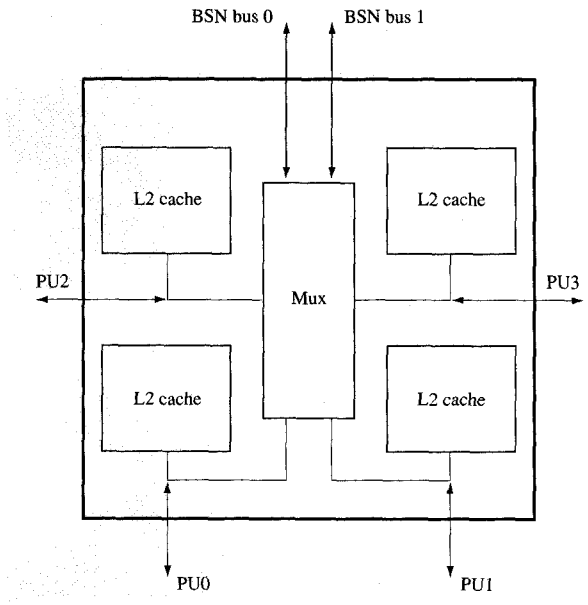


Figure 9

L2 cache.

faulty part is reloaded automatically. If, after the reload, the error comes up again, it is handled as a check-stop condition. Problems during debug can be analyzed by reading the 256-entry-deep trace buffer. With this information, it is possible to track what happened within the PU in the last 256 consecutive cycles. To circumvent hardware problems, a programmable "escape logic" is used to change the behavior of the hardware.

Level-2 cache (L2)

• *Overview*

The L2 caches (**Figure 9**) are private caches, each associated with a dedicated processor chip. One chip holds four cache macros of 64 KB each. The dataflow per chip is 8 bytes wide. As with the BSN chip, there is also a switching function. The buses of four PUs are multiplexed to obtain access to two BSN buses. This structure keeps the number of chip I/Os on PU, L2, and BSN chips balanced.

The L2 chip features are as follows:

- 16.4-mm chip size, CMOS 5X, 17.9 million transistors.
- 5.9-ns cycle time.
- 928 chip I/Os used.
- Four 64KB private caches, eight-way set-associative.
- Three cycles of access time.
- Line-fetch operations in parallel with bus snooping.
- Pipelining of subsequent line fetches.
- Caches with ECC, directories with parity and duplicated.
- Fencing against defective PUs.

• *L2 cache access*

The L2 cache chip is optimized for fast access time. To achieve this, a separate bidirectional address bus in addition to the bidirectional data bus between the PU and the L2 chip is provided. The address bus carries the absolute address for each L1 reference in order to maintain L1/L2 cache consistency. This address is used by the L2 to address port 1 of the directory array (also called tag RAM). This array has a second address port to support bus snooping via the BSN bus. Using a two-port array avoids an arbitration cycle between the PU and BSN bus addresses. When the cache array is not busy, directory access and first-cache access are done in the same cycle. Fast address-compare macros at the directory outputs create a late column-select signal. The data of the selected column are latched in a data-out register and sent to the PU interface register via the error-correction logic (ECC). This results in a three-cycle access from address-in to data-out.

• *Pipelining of data and addresses*

The L2 cache performs a store-in scheme. Store-through from the L1 cache is done in units of 16 bytes, disregarding the fact that the PU can store only $1 \cdots 8$ bytes per cycle. The L1 cache merges new store data bytes with old data bytes, thus sending 16 bytes on quadword boundaries. This avoids a time-consuming prefetch, byte merge, and ECC generation in the L2 cache.

Store-throughs have lower priority than line fetches in the L2 cache. Store datashots prior to a line fetch are buffered in an L2 store buffer (see Figure 8), which is two entries deep. The buffer content is transferred into the cache when a line-fetch operation results in an "L2 miss." Thus, all store operations are completed before an LRU cast-out can occur.

A line fetch with an L2 hit requires eight array-read cycles and needs the bidirectional data bus for eight cycles, plus an extra "quiesce" cycle prior to bus direction change. The term *quiesce* means that the last pattern is repeated, so the reflections from the far end of the bus are clamped in the driver circuit, which is still in low-impedance state. During these nine cycles the bus is not available for store-through operations by the PU. Therefore, the PU has implemented one L1 store buffer per L2 pair. These buffers are four entries deep. The address bus is not busy during the line-fetch operation, so subsequent fetch, store, or L1 miss addresses can be sent. In the case of store-through operations, associated address/column information is buffered in the L2 address

buffer. The interface ensures that the PU store buffer and the L2 address buffer fill and empty in synchronism.

Figure 10 shows a timing diagram for a sequence of five PU store operations *a–e*, where *c* and *e* encounter an L1 miss, *e* additionally an L2 miss. The normal pipeline of stores *a* and *b* is interrupted when the L1 miss condition for store *c* is signaled to the L2 chip. Datashots *a* and *b* are saved in the L2 store buffer. The write cache is suppressed, and eight cache-read cycles are performed to provide the data for store *c*. The PU repeats the store *c* when the first fetched quadword arrives in the L1 receive register. Now the L1 references *d* and *e* can be processed. The L2 address buffer holds the corresponding cache row address and column for stores *c* and *d*, and the PU buffers the datashots *c* and *d*. This buffer is emptied at the end of the line-fetch operation in the PU, resulting in the L2 write cycles *c* and *d*. The pending stores *a* and *b* are executed after the end of the L2 line fetch. Store *e* leads to L1 miss and L2 miss. The L2 starts the bus operation by raising the L2–BSN request in parallel with the current line-fetch operation for store *c*.

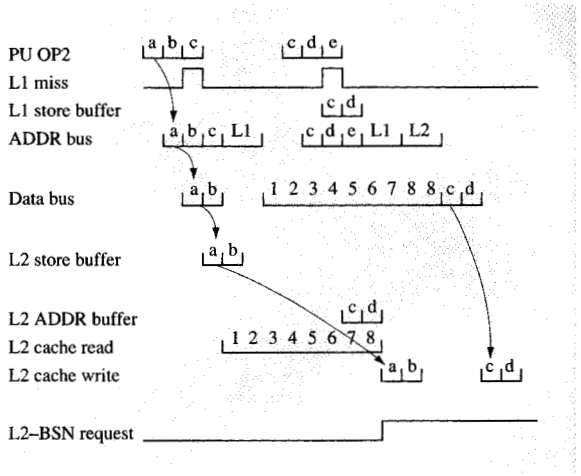


Figure 10

Store-through line-fetch sequences.

• *L1/L2 cache consistency*

The data integrity across all caches in the system is guaranteed by the MESI protocol (*M* = modified, *E* = exclusive, *S* = shared, *I* = invalid). The *M* state is known only to the L2 cache. The L1 does not need to know this state because of the L1 store-through scheme. Data integrity between L1 and the private L2 must also be maintained.

L1 store-through requires that the L1 content must always be a subset of the L2 content. Generally, this requirement is fulfilled as long as the L2 cache associativity is equal to or greater than the L1 cache associativity and as long as the LRU replacement algorithms of both caches are perfect. A difficulty arises in systems with depopulated L2 caches (two-bus or one-bus systems). Here, two or even four L1 rows are mapped into one L2 row. Therefore, the L2 must enforce the invalidation of L1 cache lines when the L2 replaces a line with the same address. A fixed protocol is used for this purpose. The third and fourth cycles after the missing address *e* (see Figure 10) are reserved on the bidirectional address bus for the L2. When an L2 miss occurs, the L2 sends its LRU line address, indicated as *l2*, to the L1 cache. The L1 performs a directory search and an invalidation in the case of a match. Prior to the L2 LRU address, the L1 LRU address is on the address bus. This is used for a different purpose, the so-called "mini-broadcast."

Mini-broadcast

The L2 cache directory maintains a copy of the L1-valid bits, as well as the indicator bits used by the MESI protocol. An L1-valid bit in the L2 is always set after the missing address for a line fetch has appeared at the L2. An L1-valid bit is reset after the L1 LRU address has been sent over the address bus. Using the L1-valid bit gives some performance advantages in the MP system. Other processors may request a change of a cache-line state via the L2 and BSN chips. The command (line fetch due to fetch, line fetch due to store, line invalidate, I/O fetch or store, etc.) and the address are routed over the L2–BSN bus, and an L2 directory access is done at the second directory port. When an L2 hit occurs, the line state must be changed according to Table 2. An L2-to-L1 protocol ensures that the states of the L2 and L1 cache lines change concurrently. The PU is interrupted for five cycles to perform the appropriate action. This performance loss can be avoided when the L1-valid bit is off. The L2 does the state change alone in a "read–modify–write" action on the directory.

• *Error detection and recovery*

PU fencing

The clock-generation chip reacts differently under PU check-stop conditions than under L2, L2.5, or memory check-stop conditions. The system continues its operation even when a PU enters check-stop state. Thus, system availability is increased significantly. Since the L2 may contain modified memory data, it is mandatory that the L2–BSN interface is not disturbed. The PU fencing function is implemented simply by a negative active "PU check-stop" signal which is latched in the L2 chip. Incoming control signals are suppressed, and the address bus-in register switches over to hold its old value. The

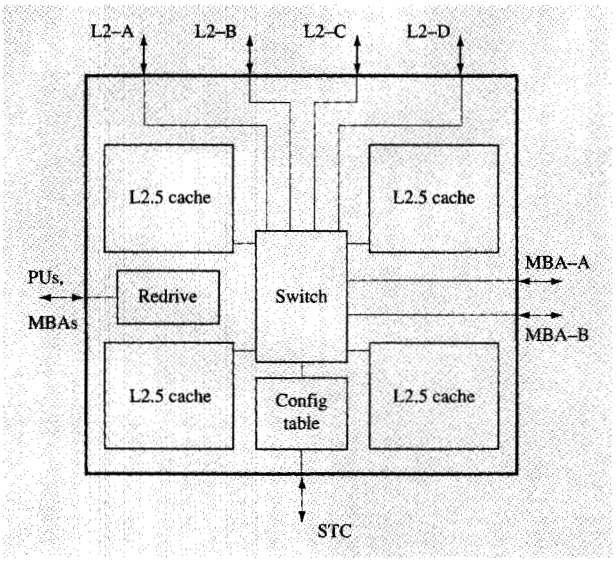


Figure 11

Bus-switching network.

handshaking for broadcast operations between L2 and PU due to cross-cache interrogation is emulated by the L2 chip.

The clock-generation chip forces all I/O drivers of a defective PU to high-impedance state. The signal levels of all receivers go to minus because of the integrated pull-down resistors. The negative active PU check-stop signal utilizes this effect and stays active. Systems with depopulated or deactivated PU chips behave similarly; the open PU port at the L2 chip is fenced.

Soft errors in large arrays

Soft errors due to alpha particles in large arrays must not create check-stop situations. These errors are usually single-bit errors. ECC with single-error correct, double-error detect capability is implemented on the L2 data caches. The directories require an equivalent correction/detection mechanism. The implementation of ECC would have cost one extra cycle of L2 access; instead, a duplication of arrays was chosen and the following rules obeyed:

- Parity check in one array only: Use the other one.
- No parity check but array outputs unequal: Check-stop.

Synchronism check

The control parts of two L2 chips are identical and must always be in synchronism. The error-detection capability is improved when the state of the control logic for these two chips is compared every cycle. The implementation is simple. The main control signals from each of the four caches on a chip are exclusive-ORed, driven via latches to the second chip of a pair, and compared there with the equivalent XOR sum. A mismatch leads to check-stop. The XOR tree is large. For a better isolation down to the source of a defect, it is advisable to implement indicator latches for groups of signals. The latches can be inspected in the scan chain after a sync check has occurred. Mismatching latch states between the chips point to the error.

Bus-switching network (BSN)

- *Overview*

The BSN chip (**Figure 11**) is required to connect different physical data buses to one logical data bus. To support a high system throughput, the bus control logic, caches, and a memory address translation are provided. The BSN chip features include

- 14.5-mm chip size, CMOS 5X, 16.6 million transistors.
- 5.9-ns cycle time.
- 758 chip I/Os used.
- Four 64KB shared caches (L2.5) per chip.
- Configuration table (CFT) for DSR/2 support.
- High-speed switch for seven electrically decoupled ports, four L2s, two MBAs, and one STC.
- Support for L2 and L2.5 cache coherence.
- 8-byte-wide buses on BSN chips.
- System measurement instrument (SMI) support.
- Redrive logic for processor-to-processor communication.
- Trace buffers for system debug.

- *Switching part*

The BSN is used as bus controller and bus-switching chip. Up to four L2 cache chips can be connected to the chip with point-to-point nets for buses and control signals. Up to two MBAs can be connected to the BSN with three-point nets (one MBA and two BSNs). Finally, one memory card with its STC is connected with point-to-point nets. This structure allows the chip as well as the buses to operate with system cycle speed; no speed-matching buffers are required. Additionally, the switching network must connect various internal units to the external buses:

- Four L2.5 caches that are especially designed to hold shared data for all PUs.
- Configuration table (CFT), used to implement the S/390 DSR/2 feature (dynamic storage reconfiguration).
- Parts of the system measurement instrument (SMI) to support performance measurements, especially in the multiprocessing environment.
- Cycle and command trace buffers for system bring-up/debug.

To achieve good system performance, the BSN must guarantee a high bus bandwidth combined with a low bus latency. For the required bandwidth, two-way bus and memory interleaving is supported. A two-cycle bus operation (low latency) from L2 cache to L2 cache is achieved by a special bus-switching logic on the BSN chip. This logic is placed between master- and slave-clocked latches (see **Figure 12**) and causes command and data to be flushed through the BSN chip, disregarding the chip-to-chip clock skew, which is about 20% of the system cycle time. This concept allows a two-cycle operation with only one clock skew added to the total path delay; the cycle time of the BSN buses and control signals can therefore keep pace with the system. The switching logic shown in Figure 12 is also used for the chip-to-chip control signals to implement a fast, low-latency protocol for data transfer and bus snooping. Two round-robin arbiters for up to twelve PUs and two MBAs reduce the number of wires in the MP system and lead to a good bus utilization. The BSN control logic must maintain and support data integrity and cache coherence for all connected chips.

Besides the two-way memory interleaving, the switching logic generates gaps on the bus to allow the L2s and MBAs to put new commands on the bus [4], therefore improving bus utilization. An example of a typical bus sequence is shown in **Figure 13**. In this case two interleaving line fetches are processed; one line fetch is served by the DRAM and the other by the L2.5 cache on the BSN chip.

The timing diagram shows three different ports (two L2–BSN buses and the BSN–STC bus): *F* indicates a line-fetch command which is driven/redriven for two cycles on the buses, and $0 \cdots 77$ indicates the requested line-fetch data of 128 bytes, transferred with eight datashots ($0 \cdots 7$), each 16 bytes wide. For electrical reasons, the last shot of a command or data packet must always be driven for two cycles. The . . . *Sel* signals validate the command on the different ports; the *XFer* . . . signals validate the line-fetch data. The second *STC Sel* for the interleaving line fetch cancels the line-fetch request to the STC and is caused by the L2.5 match.

- *L2.5 cache*

The L2.5 cache reduces the latency for data not kept in the L1 and L2 caches of the requesting processor. It provides the data immediately if the requested cache line is already shared by other processors. Memory access latency is thereby avoided, saving 18 cycles in the cache-line "fetch" operation. This increases the overall processor performance by about 12% in the G3 systems, where this three-level cache hierarchy scheme was first implemented and tested.

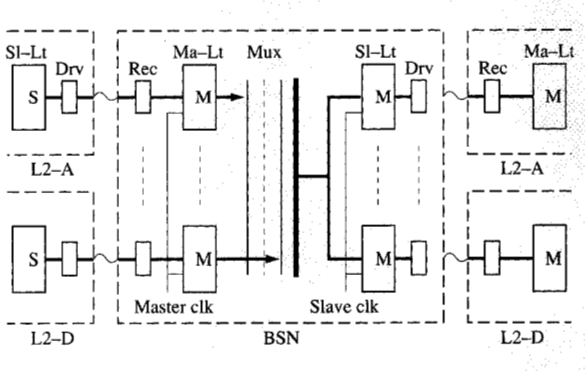


Figure 12

BSN switching logic for L2–BSN connections.

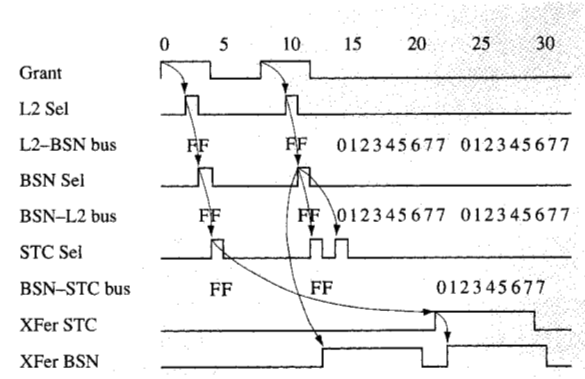


Figure 13

Bus-timing sequence with interleaving.

L2.5 cache organization

The total L2.5 cache is organized to support the cache-line interleaving mechanism; it is therefore split into four identical independent parts called quadrants, as shown in Figure 11. A L2.5 cache bank comprises two quadrants. The switch selects the banks by address bit 21 and the quadrants by address bit 20. Each quadrant holds an eight-way set-associative cache of 64 KB with a line size of 128 bytes. The design effort could be kept at a minimum by using the same array macros as for the L2 chip in the G3 system.

L2.5 cache-line-state handling

The data integrity within the three-level cache hierarchy is controlled according to the MESI protocol. The *M* state is not included in the L2.5 cache, since data integrity for

Table 4 L2.5 cache-line state before and after line-fetch operation.

LF operation type	*Cache-line state before line fetch*	*New state*
LF–DFETCH	*I*	*E/S*
	E	*S*
	S	*S*
LF–IFETCH	*I*	*S*
	E	*S*
	S	*S*
LF–DSTORE	*I*	*S*
	E	*E*
	S	*E*

modified data is maintained by L2 caches. All other MESI states are handled by the L2.5 cache, and its behavior is optimized to hold actual shared data. Any request for a missing line in the processor's L1 and L2 caches is routed to the L2.5 cache and to the memory. If the requested data are kept in the L2.5 cache and shared among other processors (*S* state), the L2.5 cache will provide the data immediately. If the data are available but not yet shared (*E* state), the L2.5 waits to provide data until all L2 caches have searched their directories, with the result that none of them holds the line in the *M* state. For an active *M* state in an L2, this L2 provides the actual data, the switch in the BSN routes the data to the requesting PU, and the L2.5 is updated in parallel. In both cases the line is marked as *S* if it is requested because of a "fetch"-type instruction in the processor. For a "store"-type instruction, it remains in the *E* state in the L2.5 cache. Data are provided by the memory only if the requested line is not kept in the L2.5 cache (*I* state). The switch provides the data to the requesting PU, and the L2.5 cache is updated in parallel. The same rules apply to the transfer into the *S* or *E* state in the L2.5 cache.

Table 4 summarizes cache-line-state handling by the L2.5 cache. It shows the state of a cache line before and after the execution of a line-fetch operation. The L2.5 cache receives different line-fetch commands, according to the type of instruction which caused the line-fetch operation (LF) in the processor; these are line fetch for data-fetch-type instructions (LF–DFETCH); line fetch for instruction fetch (LF–IFETCH); and line fetch for data-store-type instructions (LF–DSTORE). If a cache line is not kept in the L2.5 cache (see the *I*-state column) for an LF–DFETCH, the new state can be either *E* if no other processor owns the cache line or *S* if another processor has this line in the *E* or *S* state already. This is indicated in the table by *E/S*.

Table 4 does not show the fact that for all line-fetch operations in the *I* or *E* state, the requested data could be kept in another L2 cache in the *M* state. In this case, this L2 would provide the data (cross-cache cast-out), which would be loaded in parallel into the L2.5 cache. The implemented scheme is optimized for processor performance. Actual measurements show its effectiveness, since more than 80% of the cache lines kept in the L2.5 are in the *S* state. This cache-line state provides the most benefit, since processor data latency is reduced by 18 cycles compared to an access to memory.

- *Configuration table*

The configuration table is the implementation of the dynamic storage reconfiguration (DSR/2) facility. It is used to map the absolute addresses coming from the PUs or MBAs into physical memory addresses. Depending on the memory size, up to 512 storage elements, which can be different in size, are supported. The elements can be reconfigured during normal system operation. The address mapping is done within one cycle, before the address is routed to the memory card. The PUs can write and read the configuration table with special controls and senses. A bypass of the table can be activated and deactivated, and some special memory functions always bypass the table (i.e., senses and controls to the memory card).

- *System configurations*

Different systems (G3 and G4) with their different PU/L2 chips, memory cards with different sizes and access times, and varying configurations (one-, two-, and four-bus systems) require considerable programmable logic inside the switch to achieve a good bus performance; i.e., different DRAMs require different access times.

Additional logic is spent for PU-to-PU communication signals and MBA-to-PU interrupt signals. To reduce the required I/O pin count for the high-end configuration, this "redrive logic" is spread over multiple chips. To provide the same functions in smaller systems, especially the one-bus system with only two BSN chips, this logic has to be programmable. At system start-up, the BSN chips are "personalized" for these configurations.

- *Error detection and recovery*

For a highly available and reliable system, error detection and error recovery are very important in avoiding data loss and ensuring data integrity.

Switching part

Because the BSN is stimulated by external signals, the first step for error detection is the checking of all chip inputs. Additionally, the internal states of the chip must be checked. The implemented functions are as follows:

- The buses and the internal dataflow are parity-checked on byte boundaries and cause a system check-stop in the event of an error.
- The control signal inputs of the chip, which are wired point-to-point, use the fact that two BSN chips per bus always work in parallel. So-called sync checks observe the correct function of these inputs, and, in the case of a mismatch, the system is stopped. The information concerning which other chip caused the error is stored and can be accessed by the service element.
- The powering trees for the internal multiplexors are checked in order to avoid incorrect routing of data through the system.
- Bus protocol functions check important control signals (check the checkers).
- Addressing invalid entries in the configuration table suppresses command forwarding to the memory card. This is signaled to the PU or MBA.
- The memory card can handle certain "accept errors" which are routed via the BSN to the requesting PU or MBA. Examples of accept errors are "address exceeds maximum range," "bad parity on data," and "illegal command sequence."
- Especially for problem fixes during system bring-up, so-called "escape logic" is implemented to allow the detection and correction of protocol errors, i.e., time-outs.

L2.5 cache

The entire dataflow of the L2.5 cache [5] includes parity, which is checked for correctness in all operations. In addition, parity is included in the L2.5 cache directory, which keeps track of the lines included in the L2.5 cache and their state. The data in the cache include a double-bit error-detection and single-bit error-correction scheme (ECC) [6]. Single-bit errors are corrected before the data are transferred to the switch. A hardware "deconfiguration" scheme is implemented for both parity errors in the directory and double-bit errors in the data. In these cases, the failing quadrant is "deleted"; i.e., it no longer participates as a cache. All further accesses are automatically routed to the memory subsystem. No data loss occurs, since the L2.5 does not include the *M* state. Because of this mechanism, the inclusion of a cache in the BSN does not decrease system reliability and availability, but increases overall processor performance.

Configuration table

The configuration table comprises two arrays (per chip) which hold the same address-mapping tables. In the case of a single-bit error, the "good" array is used. The correct address is routed to the memory card, and the single-bit error is latched in the BSN. The error latch can be sensed by the PU, and the configuration table can then be rewritten.

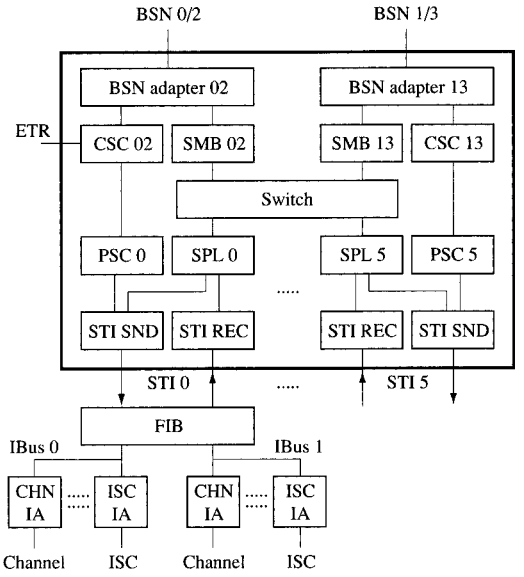


Figure 14

Memory bus adapter.

Memory bus adapter (MBA)

- *Overview*

The MBA (**Figure 14**) is a high-speed, low-latency DMA controller that provides the connection between memory and I/O. The key features of the MBA are the following:

- Bandwidth of 2 GB/s through two BSN adapters.
- High-speed self-timed interface (STI) with cable lengths between 5 cm and 20 m.
- CMOS 5X technology.
- 15.5-mm chip size.
- 770 chip I/Os used.
- 3.6 million transistors.
- 5.9-ns cycle time.

- *Main functional units*

There are two primary types of storage operations: Fetch operations (transfer data from memory to I/O), and store operations (transfer data from I/O to memory). Programming of the MBA is done with sense and control instructions issued by a PU: Control instructions set/modify registers, and sense instructions read registers. The main functional units on the MBA are described in the following subsections.

BSN adapter unit
There are two BSN adapter units on each MBA chip. The BSN adapter connects MBA–BSN bus 0/2 (MBA–BSN bus 1/3) to the speed-matching buffers, which contain the command and data for the storage operations. In addition, it provides an interface to the central sense/control unit. One BSN adapter has a bandwidth of 1 GB/s.

Speed-matching buffer (SMB)
For every BSN adapter there are three speed-matching buffers, to hold commands, store data, and fetch data. These buffers are necessary to adapt the word-wide STI macro interface to the quadword-wide BSN interface. The command buffer has room for four fetch and four store commands; the store and fetch data buffers each have room for four lines.

Switch
The switch connects the STI path logic to the speed-matching buffers. To realize a high bandwidth, a split transaction protocol is implemented, allowing the concurrent execution of two store data transfers, two fetch data transfers, and one command transfer.

STI path logic (SPL)
This logic has three purposes:

1. It splits information packets received from the STI–REC macro into a command part and a data part, which are sent over the switch to the speed-matching buffers.
2. It receives fetch data from the switch and builds an information packet (IP) as required by the STI–SND macro.
3. It receives from the port sense/control logic the data to send sense/control commands down the STI link. Again, it builds the IPs as required by the STI–SND macro.

Central sense/control (CSC)
This unit provides the PU with access to the registers in the MBA. To balance sense/control loads on the BSN buses, there are two CSC units on each MBA chip, and each CSC is connected to the BSN 02 adapter and the BSN 13 adapter. STI ports 0, 1, and 2 are accessed via central sense/control BSN 02; STI ports 3, 4, and 5 are accessed via central sense/control BSN 13. Central sense/control BSN 02 connects to the ETR (external time reference). In addition, it contains all logic for the master TOD (time of day) and the facilities to synchronize several local TODs to the master TOD.

Port sense/control (PSC)
This logic provides data and commands for sense/control signals sent via the STI. It contains a set of registers to manage interrupt and busy conditions on a channel basis. Further, it permits programming the STI interface to run with a byte-transfer rate of 3 ns or 4 ns.

STI macro
The STI link provides the connection via the fast internal bus adapter (FIB) to the channel [7] and to the intersystem channel (ISC) [8]. The system supports up to 256 channels and up to 32 intersystem channels. The macro consists of two parts, a receive and a send macro. The STI [9] is a byte-wide very high-speed data interface using differential drivers/receivers. It is a full-duplex bus with a raw data rate of 250/333 MB/s in each direction. In addition to the eight data bits and the parity bit, a clock is sent in each direction. With every clock edge, data are transmitted/received. Information is transmitted on the link in "information packets" (IP) consisting of header and data blocks. The link protocol causes overhead, which leads to an effective data rate of approximately 200 MB/s in each direction.

There are nine different clock domains on the chip (six STI receive clocks, one STI send clock, one system clock, and one ETR clock). One of the goals of the design was to minimize latency, which is caused by crossing asynchronous clock boundaries. For example, if an IP arrives at the STI receive macro, this is signaled to the SPL. The SPL, which runs with the system clock, starts to read out an IP buffer of the STI if sufficient data have been received in the IP buffer. Since there are many combinations of system clock speeds and STI clock speeds, a set of programmable counters were implemented in order to determine the best read-out start time.

- *Error detection and recovery*

All registers in the data path and most state machines in the control flow are parity-checked. If a check occurs, this is signaled to the originator of the operation (e.g., the channel) and the corresponding operation is retried. If the error is of intermittent nature, the system continues to run; if it is a permanent error, the system tries to continue operation in a deconfigured mode. A special mechanism was designed to check for failures in the speed-matching buffer.

The principle is shown in **Figure 15**. In a horizontal data-checking mechanism, each doubleword of data (cmd) is protected by a corresponding parity byte. In a vertical data-checking approach, a block of data is protected by a longitudinal redundancy check (LRC) byte. This protection mechanism is typical for link protocols. Both approaches can be combined in a concurrent signature-checking approach. Every time cmd/data are written into the buffers, a new signature is generated from the actual data to be written and the previous written signature. This signature is saved as in the horizontal data-checking

scheme. Every time cmd/data are read out of the buffers, a new signature is generated from the actual data read and the previous signature read. The new calculated signature must be identical in every cycle with the signature stored in the array. The scheme described has a modest circuit overhead and very good error-detection capabilities. Note that the number of array bits is the same as in the horizontal checking approach.

- *Verification*

Compared to its predecessor, the MBA G3 increased the bandwidth by a factor of 10 (two GB/s versus 200 MB/s). To achieve such a high bandwidth, a heavily queued chip had to be built; therefore, the number of concurrent operations increased from 3 to 30. Verification of a chip with such huge numbers of concurrent operations was a challenging task. Part of the switch logic was verified using formal verification [10].

Storage controller (STC) and memory subsystem

- *Overview*

The memory subsystem [11] is designed to serve as S/390 main and expanded storage for the G3 and G4 systems. Both levels are located on the same physical unit, a separate memory card. To reduce the bus traffic between PU and memory, certain memory-related operations are implemented on the memory card, as well as the basic store and fetch line functions. The S/390 storage key protection also resides on the memory card, with all the necessary logic. Special features have been implemented which increase the memory reliability and availability. Memory card characteristics (**Figure 16**) are the following:

- Maximum card size 6 GB, based on 64Mb DRAM technology.
- 4Mb/16Mb/64Mb Extended Data Output (EDO) DRAM support (50-ns and 60-ns RAS access).
- Latency 17 cycles at 5.9 ns (50 ns) or 18 cycles at 6.25 ns (60 ns).
- Busy time 25 cycles at 5.9 ns for one line.
- Card technology mixed-grid array (MGA) 12 signal, 10 power layers (9 × 11 in.).
- DRAM package correction ECC; two spare DRAMs per card.

Figure 17 is a photograph of the card.

STC characteristics (**Figure 18**) are the following:

- 12.7-mm chip size, CMOS 5X, approximately 1.3 million transistors.
- 5.9-ns cycle time.
- 748 chip I/Os used for signals.

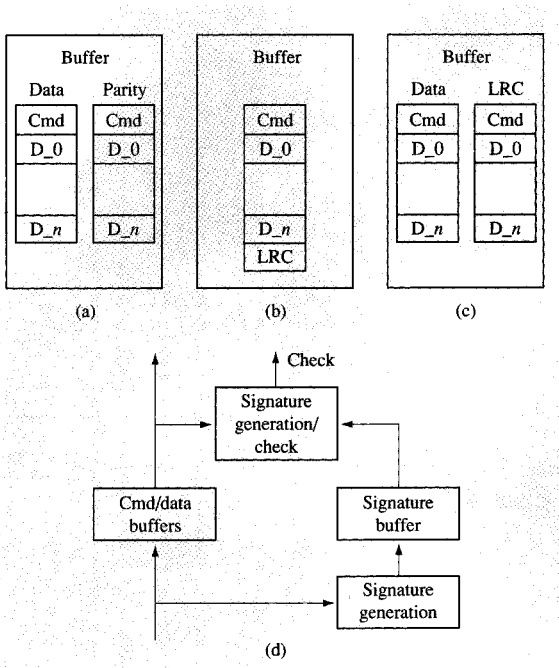
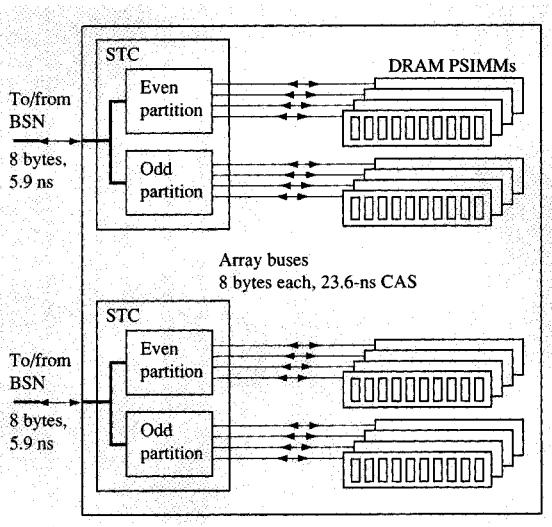


Figure 15

Principle of concurrent signature checking: (a) Horizontal checking; (b) vertical checking; (c) horizontal and vertical checking; (d) concurrent signature checking.



PSIMM = pinned single in-line memory module
CAS = column-access strobe

Figure 16

Logical data bus structuring on memory card.

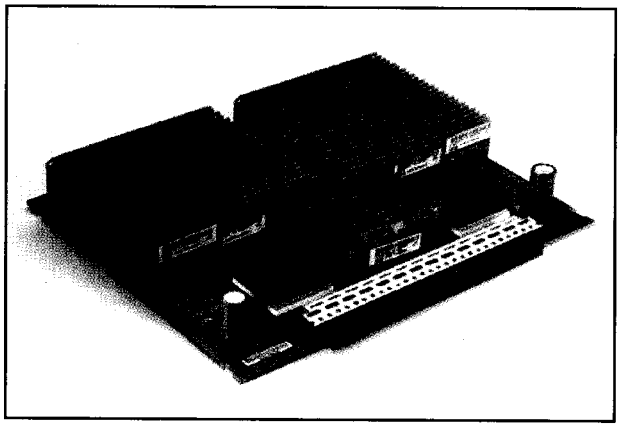


Figure 17

G3/G4 memory card.

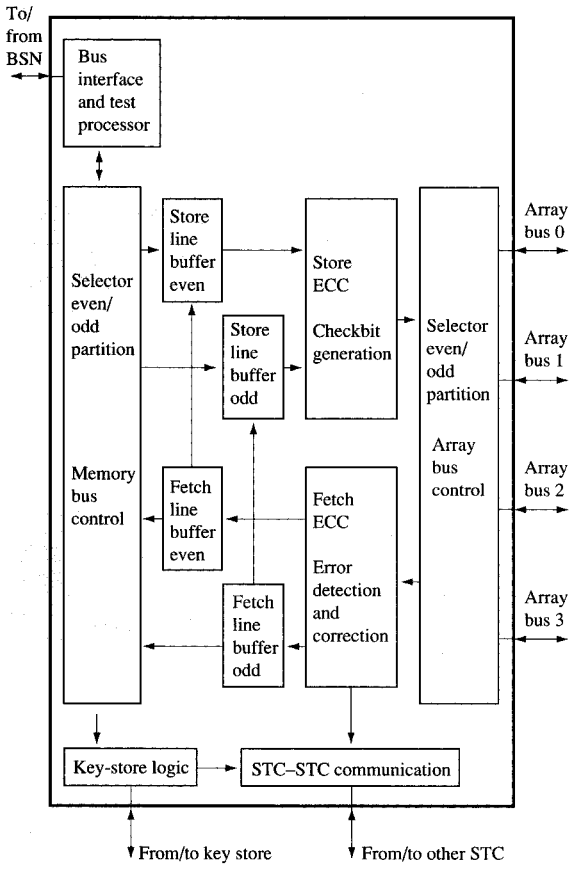


Figure 18

Simplified dataflow of one storage controller.

- Independent line buffers for store and fetch operations.
- Line interleaving on one memory card.
- Expanded storage support.
- Fetch before store for fast LRU cast-outs.
- Wraparound feature; first quadword of line served first.
- Redundancy setting per DRAM module on the fly.
- Active parallel redundancy for the key store.
- Trace buffers for system debug.
- Programmable DRAM and SRAM self-test performed by STC.

- *Memory card structure*

Two independent storage banks

Because of the relatively long DRAM access time (50 ns or 60 ns), two independent storage banks have been implemented. This permits utilization of the BSN–STC bus for command or data transfer of one storage bank while the other bank is active with RAM accesses (even/odd interleave on the BSN–STC bus). This maximizes the BSN–STC bus utilization by filling the "latency gap" on an individual bus. Each storage bank utilizes independent store and fetch buffers in order to decouple the activities on the BSN–STC bus from those on the buses to the DRAMs.

On-card array bus structure

To meet the bandwidth requirements, every STC physically interleaves four array buses (each array bus is 64 + 12 bits wide), and for one line transfer it selects an individual DRAM module twice. This sequence supplies eight doublewords per STC to complete one line. This structure exploits the CAS cycle time of 25 ns for 60-ns standard EDO RAMs (4×6.25 ns = 25 ns).

The electrical redrive challenge

On the 6GB card (maximum configuration), the two STCs must drive at their DRAM interface 48 PSIMMs, 19 DRAM modules per PSIMM, four data I/Os, 12 addresses, and three controls per DRAM. Overall, 17366 DRAM I/Os and 600 SRAM I/Os must be attached to the support logic. Of the 748 signal I/Os per storage controller, 520 were available for the DRAM interface; they were arranged in the following matrix:

- 304 data I/Os for four data buses, 76 bits per data bus (both storage banks share the data bus).
- 88 address I/Os for four data buses and two storage banks.
- 48 address I/Os for every SIMM for the most critical address.
- 48 CAS I/Os for every SIMM.
- 24 RAS I/Os for every SIMM pair.

- 8 write-enable I/Os for four data buses and two storage banks.

Because of the pin-count limitation, one STC supports the SRAM addresses while the other supports data and enabling signals, logically multiplexed over the same physical pins. The rest of the STC I/Os are used for the BSN–STC bus, STC-to-STC communication, spare DRAMs, clock signals, and test support. Overall, this array wiring structure supports cycle times at the DRAM interface down to 5.1 ns if DRAMs with 50-ns RAS access time are used.

Memory scalability and granularity
One of the major requirements in designing the memory subsystem was to support the modular concept, as well as a very wide range of memory sizes for different product offerings with just one design point (e.g., one raw card and one storage controller part number). From the point of view of memory card design, the overall G3/G4 system memory size ranges from 64 MB to 24 GB (factor 384), although not all options are actually used in the system. This has been achieved with the following steps:

- Factor 4 by number of buses in the system (one-, two-, four-bus system).
- Factor 2 by using only one storage bank on the card (even/odd interleaving disabled).
- Factor 3 by populating PSIMMs for one, two, or three RAS banks (address depth).
- Factor 16 by DRAM technology (4 Mb, 16 Mb, 64 Mb).

- *Functions complying with S/390 architecture*
The following S/390 architecture storage-related functions have been implemented on the memory card.

Key store
The S/390 architecture requires a storage-protection mechanism which ensures that only those elements of the system which have a correct access key can obtain access to storage locations. The access key protects storage in increments of 4KB pages.

The storage for this key and the related fetch, reference, and change information has been implemented as a decentralized fast SRAM array ("key store") residing on the memory card. The STC compares the access key of a requestor with the key stored on the card and permits or denies the alteration of the storage location. The time required for the key control is included in the latency numbers, mentioned earlier. If permission is denied, the STC communicates with the other STC to suppress the storage of data.

Fast LRU cast-out
To support fast LRU cast-out, the STC contains separate fetch and store buffers for each storage bank. The command is sent together with the store data to the STC. While the STC initiates the fetch access to the DRAMs, the store data are deposited in the store buffer. After the fetch data are delivered on the BSN–STC bus, the STC begins a second access cycle to store the data (see Figure 18). During this time, the other storage bank can concurrently receive and execute other commands.

Expanded storage support
A similar mechanism, called "move page fetch/move page store," is used to support the exchange of lines between the S/390 main storage and expanded storage. The lines are fetched from the source address into the fetch buffer and subsequently stored through the store buffer to the destination address.

Data handling within one line
Certain operations require that individual bytes or sequences of bytes be changed within one line. This is also done in the storage controller. It examines the byte address and the access key, initiates the prefetch of the line, merges the data, and stores the changed line back to the memory. This mechanism allows the alteration of $1 \cdots 127$ bytes within one 128-byte line. Examples of this operation are partial stores from the I/O and the conditional marking of individual lines.

- *Error detection and recovery*
Because of the high number of RAM modules, the memory card contains several error-detection/correction schemes to provide high reliability and availability.

DRAM ECC
A (76, 64) ECC scheme has been implemented for the DRAMs which corrects all errors within one DRAM module, including completely defective DRAM modules (four out of four DRAM I/Os defective). This gives the capability of DRAM chip-kill correction on the fly, without impact to the running application. The ECC scheme also detects a very high percentage of failures of two or three DRAM modules within one ECC word. In addition to the correction of defective data, the DRAM address has been included in the check-bit generation in order to detect misaddressing. Errors detected by one STC are communicated to the other STC to ensure synchronous data delivery.

SAP-controlled error scrubbing
In a background process, the system assist processor periodically monitors the DRAMs by scanning the complete address space. If a line has correctable errors,

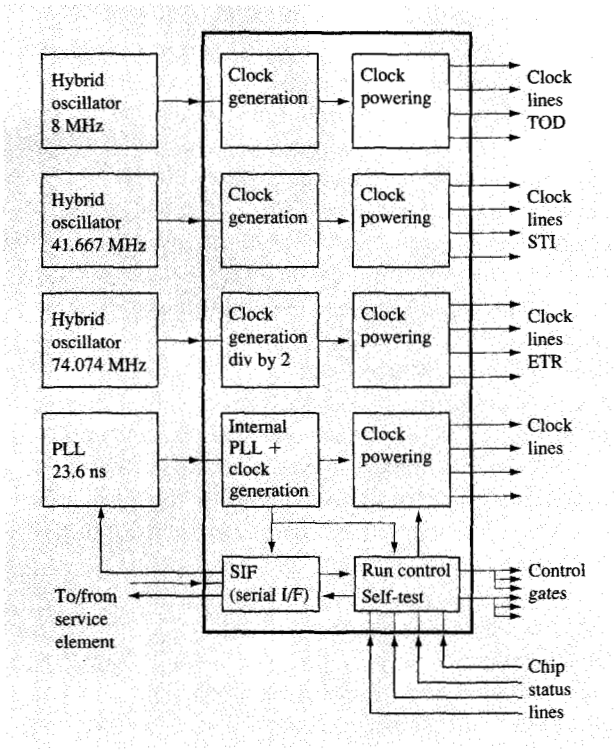


Figure 19

Block diagram of G3/G4 clock chip functional islands.

it is corrected by the STC and restored into the memory. Depending on the nature of the defect, this eliminates or "scrubs" soft errors in memory areas which are seldom used.

DRAM sparing on the fly

During the scrubbing process a hardware error map is written which contains error counters, defective PSIMM locations, defective DRAM modules on a PSIMM, and the defective I/Os, as well as the nature of the failure (correctable/uncorrectable). The error map is also used in card manufacturing.

If defined thresholds are exceeded, the redundant module is activated, replacing the most defective DRAM on the card. This is done dynamically; i.e., the data that are still correctable are corrected and copied to the redundant module. After the copy is complete, the defective DRAM is shut off. The redundant module on each half of the card can be assigned to any DRAM location. This mechanism ensures an uninterrupted customer operation.

Key-store redundancy

To provide high availability, the key store uses redundancy; i.e., each key-store entry is stored twice in two separate SRAM modules. In the case of a failure, the entry with the correct parity is used. Either of the two key stores can be disabled.

- *Integrated test support*

Traditionally memory cards are tested on separate, stand-alone card testers in manufacturing or in a laboratory environment. With the progress of CMOS technology, it became difficult and very expensive to provide equipment which keeps pace with the cycle-time requirements. Therefore, a freely programmable test processor has been implemented in the storage controller which can generate practically any test sequence.

Internal test mode

In this mode the test processor stimulates the BSN–STC bus with the same commands or command sequences as those issued by the BSN. It analyzes the response of the memory card and monitors protocol and data correctness. The card "tests itself."

External test mode

In the external mode, the test processor sends the stimuli to the memory card connector. Thus, a memory card without DRAMs and SRAMs, with only two STCs, can stimulate a normal memory card via the card connector. The card "tests another card."

As additional equipment external power supplies, a PC and a clock generator card are sufficient. Compared to commercially available stand-alone test equipment, this concept results in very low cost for test equipment, in the laboratory and in manufacturing. The test processor program can be loaded in the G3/G4 system during power-on from the service element. The major advantage of the integrated test processor is that the technology of the "tester" migrates with the technology of the device to be tested.

Clock-generation chip (CGC)

- *Overview*

The clock-generation chip (**Figure 19**) provides the necessary clock pulses and run control signals for all S/390-related hardware building blocks of the G3/G4 processor subsystem. The key features of the CGC are the following:

- CMOS 5X technology.
- 10.0-mm chip size, 0.5 million transistors.
- 414 chip I/Os used.
- 44-mm MLCI single-chip module.
- 5.9-ns cycle time.

- <0.4-ns on-chip clock skew, <1.2-ns chip-to-chip clock skew.
- Two supply voltages.
  - 2.5 V for connections to other CMOS 5X chips.
  - 3.6 V for connections to the ETR and the service element.
- Clock and control-signal generation.
  - Power-on reset recognition.
  - Five-wire interface to service element.
  - Start/stop control of entire G3/G4 processor subsystem (including check recognition and handling).
  - Serial link to all PU chips.
  - External time reference (ETR) support.
  - Programming of an external PLL.
  - Self-test control for all connected chips.

- *Power-on reset*

After recognition of an external power-on reset signal or an equivalent command from the service element, the CGC controls the hardware initialization of all chips, consisting of the following phases:

- Detection of all chips connected to the CGC.
- Shift in zeros through the serial SRL network (initial self-test data).
- Test all embedded arrays via ABIST (array built-in self-test) and subsequent initialization of embedded arrays (storing zeros with good parity and directory array valid bits OFF).
- Self-test of the chip internal logic via LBIST (logic built-in self-test).
- Set up the S/390 processor(s) for transfer of bootstrapping code by the service element.

Any chip that does not pass the ABIST or LBIST remains disabled and is no longer available for system operation. However, the system may operate in a degraded mode until the defective component has been replaced.

- *Attachment to the service element*

The clock chip connects to the service element subsystem via the five-wire interface, which is the physical connection between the service element and the CGC and consists of the following lines:

- *Shift gate* instructs the CGC to shift 1-bit information on the data-in/data-out lines.
- *Address/data* declares the information transported on the data lines as either address or data, depending on the polarity.
- *Set pulse* is used to strobe data and for commands.
- *Data-in* is the input line for serial bit transport.
- *Data-out* is the output line for serial bit transport.

The CGC implements an address register holding the address shifted in by the service element and a decoder to address an individual target based on the contents of the address register. Each chip connected to the CGC has a unique address, making it an individual target. Some addresses are reserved for facilities on the CGC itself, e.g., for the "status register," which provides status information about the entire system to the service element. Another example of a CGC address is the command to initiate an initial microprogram load (IML).

During address mode, a 9-bit address consisting of eight address bits and one parity bit is shifted into the address register, which selects the target for the following set pulse or shift gate in data mode. While the shift gate in data mode propagates a single bit into an SRL chain, the set pulse is used to assist the array access on the chips and to control the CGC. The five-wire interface supports service functions such as

- Serial R/W access of all SRL chains and embedded arrays on building blocks connected.
- Serial R/W access of SRL chains on the CGC itself.
- Single cycle.
- Start/stop of the G3/G4 processor.

In addition to the five-wire interface, the CGC has the capability to alert the service element by raising the interrupt line to indicate an asynchronous event (e.g., a check condition). This mechanism eliminates the need for polling for certain conditions and increases the overall performance.

- *Clock and control-signal generation*

The CGC provides the clocks and control signals as well as the address decoding for the service element subsystem for the following maximum configuration:

- Twelve PU chips.
- Twelve L2 chips.
- Eight BSN chips.
- Eight STC chips (four memory cards).
- Two MBA chips.
- One ETR chip.
- Two crypto chips.

The control signals of the CGC are optimized for a high-end system implemented entirely in chips. The clock chip supports any combination of chips. The service element must validate each configuration and disable particular chips if necessary. The valid configurations for a G3 and a G4 processor subsystem differ from one another. All chips except the ETR chip receive clock signals that are synchronous with the system cycle of 5.9 ns. Each clock chip generates its 5.9-ns clock signal, locally derived from

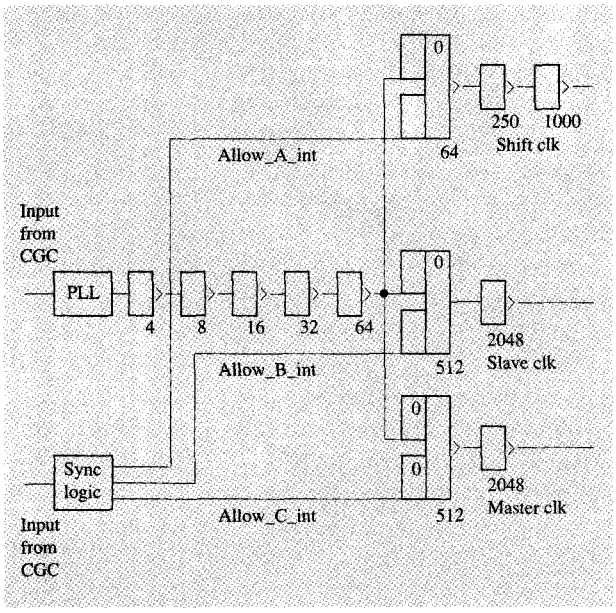


Figure 20

OPCG schematic.

a 23.6-ns external reference clock, via an on-chip PLL. The ETR and those parts of the MBA chips that communicate with the ETR chip get a separate clock with a 27-ns cycle. The CGC has the capability to start and stop each PU chip on an individual basis. The CGC starts the other chips if at least one PU chip is active. All start/stop actions execute synchronously, which means that the clocks for all chips are enabled or disabled within the same cycle. There is also a clock domain available that never stops once it has been started after power-on. This "continuous-running" clock domain is used on the PU and MBA chips for the timer facilities and on the STC chips to control the refresh of the DRAMs.

- *On-product clock generation (OPCG)*

In order to achieve this function, the CGC distributes a raw oscillator signal to all chips with an individual line for each chip. Each chip including the CGC itself receives this oscillator signal and multiplies the frequency of this signal by four by means of an internal PLL. The output of the PLL feeds a clock-generation network (CGN). The CGN uses the PLL output and some control lines driven by the CGC and generates the master, slave, shift, and array clocks locally on each chip. The CGC drives the control lines as multidrop nets, and each chip synchronizes them locally and generates "Allow_x_int" signals.

Figure 20 shows the logic of the CGN implemented on each chip. The CGN uses the standard book set; higher drive capability is achieved by paralleling the standard circuits. The numbers in the diagram show the maximum number of parallel circuits. The output load of each circuit is tightly controlled during the physical design process, resulting in a clock skew of less than 0.4 ns for any two latches on the same chip [12]. Some additional adders contribute to the clock skew between any two latches on different chips:

- Different driver characteristics of the drivers of the CGC.
- Tolerances in card/module wiring.
- On-chip PLL phase error plus jitter.

All of these contributors add up to a total clock skew of less than 1.2 ns for any two latches on different chips. The technique of distributing just an oscillator signal and several control lines and using OPCG has several advantages compared to distributing all master, slave, shift, and array clocks from the CGC:

- Only the reference clock signals must be length-adjusted on all packaging levels.
- Clock-gating signals may be implemented as multidrop signals, reducing the pin count of the CGC.

- *Error detection*

Each chip can signal a check condition to the clock chip via an individual line. The action performed by the CGC depends on the source of the check. These external checks are grouped into two different groups: All PU chips, and all L2, STC, BSN, MBA, and crypto chips. If the check is raised by any PU chip, only this PU is stopped by turning off the appropriate control signals. The CGC informs the remaining PUs, which continue to run without interruption of this situation, by sending a "malfunction alert" to all other PUs via the clock–PU serial link. Any check signaled to the clock chip by any other chip stops the entire system. In addition, the CGC detects internal malfunctions and alerts the service element accordingly. In all cases the service element must perform further investigation of the check.

- *Clock–PU serial link*

The communication between the clock chip and the PUs is done by means of a synchronous serial interface and is bidirectional [13]. Several checking mechanisms have been introduced to improve reliability. Some of the conditions signaled via this interface are active PUs, malfunction alert, start pulse, soft stop state, and wait state.

The clock chip is the master of all communication. A communication package consists of a specific header, a PU field, a command field, and a checksum sent by the clock chip. All data besides the header are return-to-zero (RZ)

Table 5 Complete frame.

Bits	*Cycles*	*Content*
	3	Header
16	32	PU field
16	32	Command field
8	16	Checksum (number of ones in PU and command field)
8	16	Response bits PU0 · · · PU15
—	99	Total cycles

coded, and one bit takes two cycles to transmit. The header is B'110'. This is the only case in which two consecutive ones are sent. This information is used to inform all PUs about the beginning of a new frame. The header is the only source for synchronization. Two consecutive ones occurring anywhere else in the frame lead to a check. After sending the header, the CGC sends the 16-bit PU field. Each bit in the PU field selects a specific PU. Depending on the command field following, these bits take on a different meaning. The command following specifies the command to be executed after the checksum has been verified by the PU. There are commands that must be executed by each PU and commands that are valid for individual PUs. Finally, the CGC sends an 8-bit checksum to provide error detection on the interface. Each PU then sends its 8-bit response to the CGC, leading to the layout for a complete frame given in **Table 5**.

Summary

The IBM S/390 Parallel Enterprise Server Generation 3 is based on a well-balanced cache and modular system structure. The G3 processor chip set covers a wide performance range from a uniprocessor to a high-performance multiprocessing system. A three-level cache hierarchy and a high-speed processor interconnection scheme reduce the data latency for the processor significantly. High reliability and availability are guaranteed by the error-detection and recovery mechanisms implemented through the entire chip set. The design quality of the G3 chip set has been outstanding since the first silicon was functional to the extent that all hardware verification programs including the operating systems could be tested. Only one metallization change (metal EC) was needed for each of the G3 chips to achieve full functionality. This was the key to keeping the time from first silicon power-on to general availability of the G3 system to only eight months.

Acknowledgments

The authors would like to thank all colleagues in the IBM development laboratories in Boeblingen, Poughkeepsie, and Endicott who contributed to the success of the S/390 Parallel Enterprise Server Generation 3.

*Trademark or registered trademark of International Business Machines Corporation.

References

1. G. Doettling, K. J. Getzlaff, K. Jackson, K. Langston, B. Leppla, P. Mak, W. Shen, H. W. Tast, and U. Wille, "Shared Cache Memory Device," European Patent Application PCT/EP95/02847, July 19, 1995.
2. K. J. Getzlaff, B. Leppla, H. W. Tast, and U. Wille, "Bus Structure for a Multiprocessor System," European Patent Application PCT/EP95/01140, March 27, 1995.
3. *Enterprise Systems Architecture/390 Principles of Operation*, Order No. SA22-7201-00; available through IBM branch offices.
4. K. J. Getzlaff, B. Leppla, K. Muenzner, L. Reichl, and U. Wille, "Method of Forced Bus Interleaving," *IBM Tech. Disclosure Bull.* **39,** No. 1, 403–404 (January 1996).
5. G. Doettling, K. J. Getzlaff, B. Leppla, and U. Wille, "High Available Error Self-Recovering Shared Cache for Multiprocessor Systems," European Patent Application PCT/EP95/01453, April 18, 1995.
6. E. Fujiwara and M. Hamada, "Single b-Bit Byte Error Correcting and Double Bit Error Detecting Codes for High-Speed Memory Systems," *Proceedings of the 1992 IEEE International Symposium on Information Theory*, pp. 494 ff.
7. *IBM Enterprise Systems Architecture/390 ESCON I/O Interface*, Order No. SA22-7202; available through IBM branch offices.
8. J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach* (fourth printing), Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1990.
9. T. A. Gregg, "S/390 CMOS Server I/O: The Continuing Evolution," *IBM J. Res. Develop.* **41,** No. 4/5, xxx–xxx (1997, this issue).
10. T. Schlipf, T. Buechner, R. Fritz, M. Helms, and J. Koehl, "Formal Verification Made Easy," *IBM J. Res. Develop.* **41,** No. 4/5, 567–576 (1997, this issue).
11. B. W. Curran and M. H. Walz, "IBM Enterprise System/9000 Type 9121 System Controller and Memory Subsystem Design," *IBM J. Res. Develop.* **35,** No. 3, 357–366 (1991).
12. B. Kick, U. Baur, J. Koehl, and T. Pflueger, "Standard-Cell-Based Design Methodology for High-Performance Support Chips," *IBM J. Res. Develop.* **41,** No. 4/5, 505–514 (1997, this issue).
13. R. Braun, K. J. Getzlaff, W. Haller, T. Pflueger, and D. Schmunkamp, "Processing Unit to Clock Interface," European Patent Application PCT/EP95/01451, April 18, 1995.

Received January 13, 1997; accepted for publication July 15, 1997

Gerhard Doettling *IBM Entwicklung GmbH, Schoenaicherstrasse 220, 71032 Boeblingen, Germany (GDOETTLI at BOEVM4).* Mr. Doettling studied electrical engineering at the University of Stuttgart and received his graduate degree in 1978. For the next three years he worked for SEL in Stuttgart, where he was involved in the design of a digital telephone switching system. In 1981 he joined the IBM Development Laboratory in Boeblingen. Since then he has worked on several CMOS processor logic design projects, primarily for S/390 systems, but also for a RISC-based and an AS/400* system. Mr. Doettling is currently Manager of the CPU Development Department in Boeblingen.

Klaus Joerg Getzlaff *IBM Entwicklung GmbH, Schoenaicherstrasse 220, 71032 Boeblingen, Germany (GETZ at BOEVM2)* Mr. Getzlaff studied electrical engineering at the Technical University of Bielefeld, graduating in 1964. He worked for two years with the AEG Company on the design of test equipment for electronics industry devices. In 1966 he joined IBM Germany as a product engineer for S/370 intermediate systems. After holding assignments at the development locations in Endicott and Boeblingen for the design of the S/370 Model 3145 and the 4300 systems, Mr. Getzlaff moved to the S/370 Development Laboratory in Boeblingen in 1979, focusing on CMOS processor design and multiprocessor concepts of the 9221 series and the follow-on systems. Since 1994 he has been responsible for S/390 processor design at the Boeblingen laboratory.

Bernd Leppla *IBM Entwicklung GmbH, Schoenaicherstrasse 220, 71032 Boeblingen, Germany (bleppla@vnet.ibm.com).* Mr. Leppla studied electrical engineering at the Fachhochschule Heilbronn, graduating in 1990 and joining IBM the same year. Since 1993 he has worked on the design of the Generation 2 and Generation 3 BSN chips and is now working in the CPU development group.

Walter Lipponer *IBM Entwicklung GmbH, Schoenaicherstrasse 220, 71032 Boeblingen, Germany (lipponer@vnet.ibm.com).* Mr. Lipponer studied electrical engineering at the Berufsakademie Stuttgart, graduating in 1979. In 1981 he joined IBM at its former DRAM manufacturing plant in Sindelfingen, where he worked for five years in semiconductor quality and reliability engineering. After an assignment at IBM Burlington, Vermont, he joined the Boeblingen Development Laboratory, focusing on component and subsystem qualification of CMOS processor memories. Since 1993 Mr. Lipponer has been a member of the memory design team working on memory card and storage controller designs.

Thomas Pflueger *IBM Entwicklung GmbH, Schoenaicherstrasse 220, 71032 Boeblingen, Germany (PFLUEGER AT BOEVM3).* Mr. Pflueger is an Advisory Engineer in the S/390 processor development group. He received an M.S. degree in electrical engineering from the Technical University of Munich in 1983, joining IBM the same year. Mr. Pflueger works on processor logic design.

Thomas Schlipf *IBM Entwicklung GmbH, Schoenaicherstrasse 220, 71032 Boeblingen, Germany (SCHLIPF at BOEVM3).* Mr. Schlipf studied electrical engineering at the University of Karlsruhe. In 1985, after working for a time at the Robert Bosch Company, he joined the IBM S/390 Development Laboratory in Boeblingen. Since then he has been working on the hardware design of I/O chips and now leads the MBA team. Mr. Schlipf's interests are in the areas of computer architecture and formal verification. He is a member of the IEEE.

Dietmar Schmunkamp *IBM Entwicklung GmbH, Schoenaicherstrasse 220, 71032 Boeblingen, Germany (schmunkamp@vnet.ibm.com).* Mr. Schmunkamp studied electrical engineering at the Technische Hochschule Darmstadt and received his graduate degree in 1984, joining IBM the same year. Since then he has been working on the design of CMOS microprocessors, with special interests in clocking and service interface. Mr. Schmunkamp is a member of the VDE.

Udo Wille *IBM Entwicklung GmbH, Schoenaicherstrasse 220, 71032 Boeblingen, Germany (uwille@vnet.ibm.com).* Mr. Wille studied electrical engineering at the Ingenieurschule Bremen, graduating in 1966. For the next four years he worked for the AEG on the development of synchronous generators. In 1970 he joined IBM in Boeblingen to work in product assurance. He has held assignments in both hardware test and hardware development, and has designed I/O adapter chips and CPU chips. He now specializes in the design of level-2 caches.