IBM's S/390 G5 Microprocessor Design

A high-frequency CISC processor, microarchitecture features, and reliability enhancements were crucial in building CMOS-based systems that could outperform those based on bipolar circuit technology.

Timothy J. Slegel, Robert M. Averill III, Mark A. Check, Bruce C. Giamei, Barry W. Krumm, Christopher A. Krygowski, Wen H. Li, John S. Liptay, John D. MacDougall, Thomas J. McPherson, Jennifer A. Navarro, Eric M. Schwarz, Kevin Shum, and Charles F. Webb
IBM Corporation

The IBM S/390 G5 microprocessor in IBM's newest CMOS mainframe system provides more than twice the performance of the previous generation, the G4. The G5 system offers improved reliability and availability, along with new architectural features such as support for IEEE floating-point arithmetic and a redesigned L2 cache and processor interconnect. IBM announced the G5 system on 7 May 1998 and began volume shipments in September 1998.

The G5 system implements the ESA/390 instruction-set architecture,[1] which is based on and compatible with the original S/360 architecture introduced in 1964. Therefore, it has no RISC (reduced-instruction-set computing) concepts and is one of the most complex of all CISC (complex-instruction-set computing) architectures. Designers had to meet a unique set of challenges to achieve the G5's level of performance—for example, achieving a very high frequency given the complexity of the architecture.

Background

Until the early 1990s, IBM designed all its mainframe systems with bipolar circuit technology. This was the best performing technology for its time, but it required water cooling of the circuit modules and led to systems consuming large amounts of floor space. The last of these bipolar mainframes was announced in 1993. The following year, IBM announced the first step in its revolutionary shift to CMOS technology for its mainframe systems. Each year thereafter, IBM introduced a new-generation CMOS-based system.[2]

Although these systems gave users very good price/performance, they could not compete with the last-generation bipolar system in terms of raw MIPS (millions of instructions per second). IBM continued to ship bipolar systems for users requiring that level of performance. IBM also introduced the ability to couple multiple systems together so that users could seamlessly execute and manage jobs across systems. In 1997 IBM announced the S/390 G4 system,[3,4] with performance equivalent to the last bipolar system. Less than a year later, the S/390 G5 system debuted.[5]

The microprocessors run at 500 MHz in the fastest version of the G5 system. These models use a self-contained chiller unit to cool the multichip module (MCM) containing the processors, the L2 cache, and other support chips. IBM also ships purely air-cooled systems running at 417 and 385 MHz. The L2 and support chips run at half the frequency of the microprocessor on all versions of the system.

The S/390 G5 microprocessor uses IBM's CMOS 6X technology. This is a 0.25-micron

0272-1732/99/$10.00 © 1999 IEEE

technology with an L_{eff} of 0.15 micron on the nFET with six levels of aluminum wiring. The 14.6-mm × 14.7-mm die contains approximately 25 million transistors. These comprise seven million transistors for logic, 13 million for the L1 cache, and five million for other arrays. The chip operates at 1.9 volt at the circuit level and dissipates approximately 25 watts of power. The junction temperature is about 20°C for the models using the chiller unit.

The design team used a full-custom approach[6] for the dataflow logic. We synthesized most of the control logic and then manually optimized schematics and layout to improve cycle time. The design predominantly uses static logic, but there is some dynamic logic in certain critical paths. Figure 1 shows a micrograph of the processor.

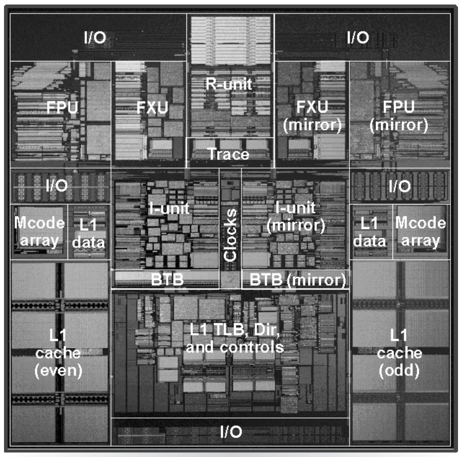


Figure 1. Micrograph of the IBM S/390 G5 microprocessor.

The G5 microprocessor's design is based on that of the previous-generation G4, but with numerous enhancements that yield a faster cycle time, fewer cycles per instruction (CPI),[7] new IEEE-compatible floating-point architecture, and significant improvements in reliability and availability. In addition, the design of the L2 cache and processor interconnect structure is entirely new for the G5.

Microarchitecture

An efficient implementation of the highly complex ESA/390 architecture leads to some unusual trade-offs in a processor's design. The G5 design team weighed these trade-offs and selected a rather simple microarchitecture that could yield a very high frequency. Nevertheless, compared with the G4, the G5 processor does implement many significant features to further improve CPI.

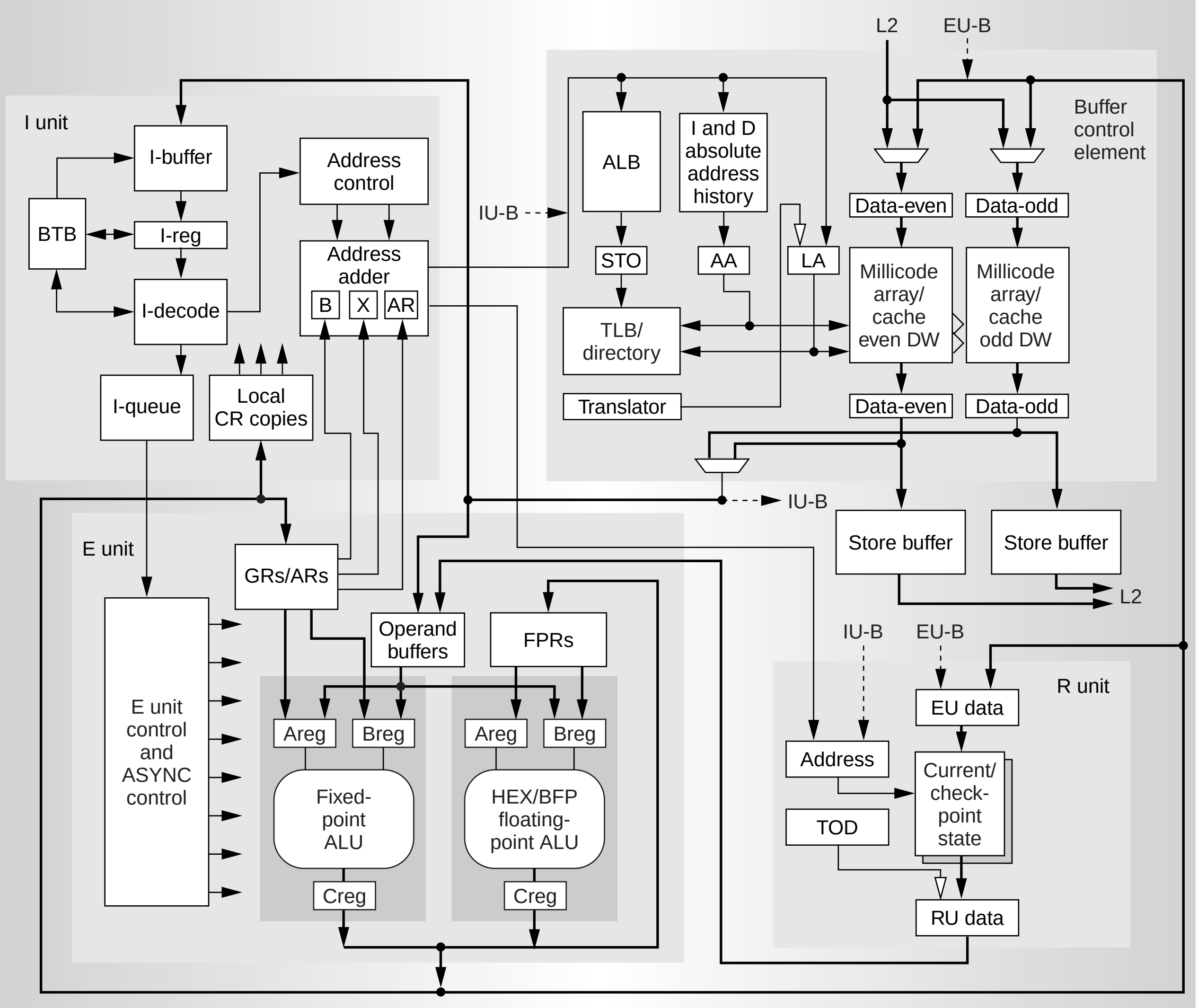
The most obvious trade-off is that the processor is not superscalar, unlike previous IBM bipolar mainframes, which had very sophisticated superscalar engines. The ESA/390 architecture has numerous, relatively commonly used, instructions that require tens, hundreds, or even thousands of clock cycles to execute no matter what implementation is chosen. Therefore, instead of executing more than one instruction per cycle, the design optimizes the number of cycles for these long-running instructions to achieve a good CPI.

Another complication of the ESA/390 architecture is that it is not a load-store architecture. The most common instruction format, RX-format, operates on data from a register and storage; this would be equivalent to two RISC instructions. The processor's pipeline design is optimized for executing these RX-format instructions in one cycle. In addition, there is a rich set of storage-to-storage instructions in the ESA/390 architecture that takes variable-length operands from two different memory locations and stores the result. Finally, the architecture has many other complexities that need efficient implementation: instructions that operate on decimal data, many addressing modes, multiple address spaces, precise interrupts, virtual machine emulation, and two different floating-point architectures.

Any practical implementation of the ESA/390 architecture must use some form of microcode to handle the more complex instructions; a purely hardware implementation would have too many design bugs to be shipped on a reasonable schedule. The G5 processor therefore uses millicode (a form of Licensed Internal Code) to implement many of the architecture's complex elements. Millicode is our name for the vertical microcode that executes on the processor.

Figure 2 shows a high-level diagram of the G5 microprocessor, which is logically partitioned into four units:

- The L1 cache, or buffer control element (BCE), contains the cache data arrays, cache directory, translation-lookaside



BTB	Branch target buffer	STO	Segment table origin	I&D	Instruction and data
CR	Control register	AR	Access registers	TOD	Time of day
AA	Absolute address	GR	General registers and millimode general registers	EU-B	Duplicated E unit
LA	Logical address			IU-B	Duplicated I unit

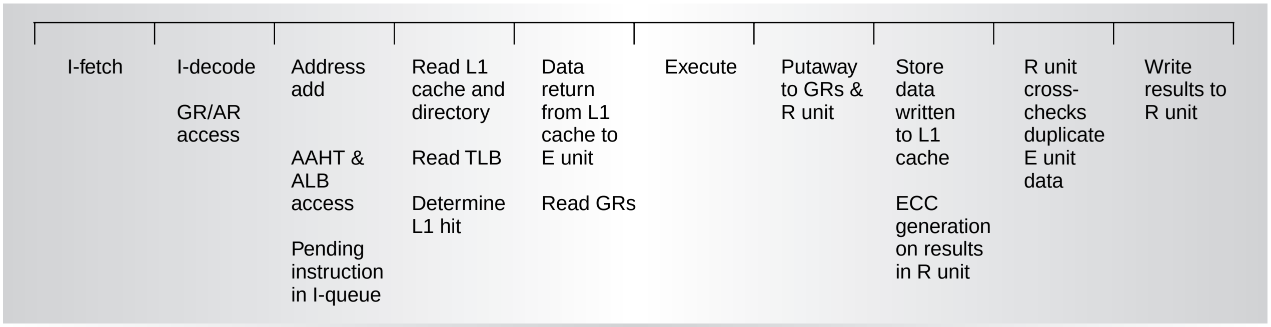
Figure 2. Diagram of the G5 microprocessor.

buffer (TLB), and address translation logic.

- The I unit handles instruction fetching, decoding, and address generation and contains the queue of instructions awaiting execution.
- The E unit contains the various execution units, along with the local working copy of the general, access, and floating-point registers.
- The R unit is the recovery unit that holds a checkpointed copy of the entire microarchitected state of the processor, timing facility, and various other miscellaneous state information.

L1 cache

The 256-Kbyte L1 cache is four-way set associative and four times larger than the L1 cache in the G4 processor. The cache is unified in that it holds instruction, operand, and millicode data and is a store-through design. The cache is two-way interleaved to support two simultaneous requesters. The L1 cache



I-fetch
I-decode
GR/AR access
Address add
AAHT & ALB access
Pending instruction in I-queue
Read L1 cache and directory
Read TLB
Determine L1 hit
Data return from L1 cache to E unit
Read GRs
Execute
Putaway to GRs & R unit
Store data written to L1 cache
ECC generation on results in R unit
R unit cross-checks duplicate E unit data
Write results to R unit

Figure 3. I unit and E unit instruction pipeline.

supports a data rate of 4 gigabytes/s for data filling from L2.

Since the cache is much larger than in the G4, we carefully evaluated the line size and eventually selected 256 bytes as optimal. The performance improvement this gives over a smaller cache line size more than makes up for the trailing-edge penalty of waiting for the last bytes of a line to be loaded into L1.

The cache is accessed by a real address, or to be more precise, by an absolute address. Because of the large cache size, four virtual bits of the address are needed in selecting a congruence class, along with all the address bits that are not subject to translation. The G5 uses an interesting approach to this problem by having an absolute address history table (AAHT) predict the absolute address value for these four bits before the request accesses the cache directory.[8] If the prediction is later determined to be incorrect after the TLB is checked, the cache hit is suppressed and the request is recycled after updating the AAHT with the correct value. There is one AAHT for instruction addresses and another for operand addresses.

The TLB contains 1,024 entries and is four-way set associative. All dynamic address translation and access register translation (ART) is done in hardware without assists from millicode for all the ESA/390 addressing modes. The ART lookaside buffer (ALB) contains eight entries and is fully associative.

The L1 cache unit also contains a 32-Kbyte writable millicode array containing the millicode for the 64 most commonly used ESA/390 instructions implemented in millicode. This array is loaded at initial power-on reset time and is not modified during normal system operation.

I unit and E unit

Figure 3 shows the instruction pipeline, as implemented in the I unit and E unit. The pipeline is effectively seven stages from the instruction fetch cycle to the writing of results for RX-format instructions. Note that for register-to-register instructions, the RR-format, no operand fetch is necessary from the L1 cache, so the pipeline length is reduced by one stage, depending on the format of adjacent instructions. The R unit has three additional stages for checkpointing of results, but these cannot cause pipeline stalls.

The I unit calculates operand addresses in the address add cycle. It then sends these requests to the L1 cache for processing and later informs the E unit when operand data is present in the operand buffers.

A new feature in the G5 processor is a 2,048-entry BTB that is two-way associative. It uses a 2-bit algorithm for branch prediction and is active for both ESA/390 code and for millicode. Both types of code share the same BTB.

The E unit is partitioned into a fixed-point unit and a floating-point unit. The fixed-point unit contains a 64-bit binary adder, a 64-bit logical/shift/and-insert-under-mask unit, and an 8-digit decimal adder. We describe the floating-point unit in more detail later.

The ESA/390 architecture is unique among today's architectures in that it contains a rich set of instructions that operate on decimal data. Decimal instructions are used intensively in the financial industry, so performance is very important. The G4 processor included an 8-digit decimal adder and executed add/subtract/compare operations in hardware while using millicode to perform more-complex operations such as decimal multiply, divide, pack/unpack, and convert between decimal and binary format.

The G5's decimal performance is significantly improved by having a decimal multiplier. It also has hardware to generate one decimal digit of the quotient and uses a millicode routine that iterates to perform the entire division operation. In addition, we added dedicated control logic to perform the other decimal instructions purely in hardware.

Instructions that manipulate the program status word (PSW) are also somewhat performance critical, but they are architecturally complex. To execute them easily, the G4 design used millicode. To improve the CPI, the G5 processor implements all these instructions in hardware.

The ESA/390 architecture requires very precise handling of interrupts and exceptions. These requirements impose a significant burden on designers. Although we optimize their design for "normal" operations, we must still obey the architectural specifications without impacting cycle time.

The G5 design uses a concept called single-instruction mode to handle interrupts and exceptions. Normally, the processor is running fully pipelined. However, when the L1 cache or the I unit detects that there may be an exception associated with an upcoming instruction, it informs the E unit, which then requests the entire processor to serialize operations. (In certain cases the E unit may detect the exception itself.) This serialization causes all instructions waiting for execution or in an I-buffer to be flushed. The next instruction then executes in the single-instruction mode without anything else in the pipeline. In this second pass, if there really is an exception or an interrupt, the hardware invokes a millicode interrupt handler to perform all the steps the architecture requires. In some cases, the normal-speed detection is only a gross, pessimistic check. Only in the single-instruction-mode pass are the precise checks made. When the processor enters the single-instruction mode, the instructions executing at that time slow down significantly. However, this mode is entered so infrequently (when there is not going to be a true interrupt) that the overall impact on performance is negligible.

R unit

The entire microarchitected state of the processor is kept in the R unit. There are 256 registers, half of them 32 bits and the other half 64 bits. All are protected by error-correcting code (ECC). These registers contain a copy of the data in ESA/390-architected registers such as general registers, control registers, floating-point registers, PSW, and so on. However, the R unit also contains a copy of working registers for millicode plus numerous processor and system control registers that only millicode can access.

For cycle time reasons, frequently used registers, such as general and floating-point registers, have their master copy in the E unit. The R unit registers have a single read port and a single write port, whereas the E unit copy has multiple read and write ports. Some of the more commonly used control registers are also shadowed in the I unit, E unit, and L1 cache as needed.

For every clock cycle in which the E unit produces a result, that value is also written into the R unit copy. First the R unit checks whether the result was correct (see the later section on processor recovery) and then it generates ECC on that result. Finally, this checkpointed result is written into the R unit registers along with its ECC. The contents of R unit registers represent the complete checkpointed state of the processor during any given cycle, should it be necessary to recover from a hardware error.

Millicode

The G5 microprocessor implements 284 ESA/390 instructions totally in hardware. Millicode is used to implement the remainder of the ESA/390 instructions that are either more complex or relatively infrequently used and therefore do not warrant a hardware implementation. The millicode instruction set consists of all the ESA/390 hardware instructions plus 102 instructions that only millicode can use.

Executing a complex ESA/390 instruction is like executing a hardwired subroutine call. The I unit detects at decode time that millicode will be required to execute an instruction. Processor hardware then sets up certain working registers that millicode will need later. This setup is optimized for different types of instructions, to minimize the amount of millicode processing necessary. Finally, the processor's instruction address is changed to point to the starting address for the required millicode routine. From then on, the processor is said to be executing in *millimode*—a highly privileged mode of operation in which millicode can essentially do anything it wants to the processor and the system.

When in millimode, the processor fetches and executes instructions just as it would in normal ESA/390 mode. When the millicode routine has completed, it executes an instruction that is similar to a hardwired subroutine return. This exits millimode, putting the processor back in ESA/390 mode, and restores the instruction address to point to the next ESA/390 instruction. The ESA/390 program is not aware that any of this has occurred; it just appears as if its next instruction has been executed. Obviously, if an ESA/390 program tries to execute an instruction that is only valid in millimode, the hardware will cause a program exception that will prevent it from executing.

When the processor is executing in millimode, the millicode has complete read/write access to all R unit registers. To reduce the number of clock cycles required when entering and leaving millimode, the millicode uses a completely different set of general registers and access registers from those normally used for ESA/390 programs.

Besides executing the complex S/390 instructions, millicode also performs various service functions. These include logging data associated with any hardware errors that may have occurred, scrubbing memory for correctable errors, supporting operator console functions, and controlling low-level I/O operations.

Virtual machine emulation

The ESA/390 architecture provides for up to two levels of virtual machines running on a processor (that is, a virtual machine running under a virtual machine running under the native hardware). This capability is exploited by IBM's PR/SM (Processor Resource/Systems Manager) environment, which lets users run multiple operating systems simultaneously on the system. Likewise, the Virtual Machine operating system uses this capability to give customers the appearance of a complete ESA/390 system for their own use.

An efficient implementation of these virtual machines requires extensive hardware support, along with millicode to manage the transition between virtual machine levels. Hardware support includes three complete copies of all 16 ESA/390-architected control registers and three copies of the timing facility registers. These three copies represent the host mode, first-level guest, and second-level guest emulation environments. In addition, logic included in the I unit detects various conditions that require interception by a higher level of emulation.

Table 1. ESA/390 floating-point architecture data formats.

Format	Sign	Exponent bits	Exponent bias	Fraction	Total width (bits)
HFP short	1	7	64	24	32
HFP long	1	7	64	56	64
HFP extended	1	7	64	112	128
BFP single	1	8	127	24	32
BFP double	1	11	1,023	53	64
BFP quad	1	15	16,383	113	128

IEEE floating-point architecture

IBM's mainframes have included floating-point arithmetic in their architecture since the 1960s. As Table 1 shows, this architecture includes 32-, 64-, and 128-bit operands, each with seven bits of exponent. A 1-bit change in the exponent corresponds to a 4-bit shift in the fraction. Therefore, it is called hexadecimal floating point (HFP).

In recent years, users have been interested in interoperability with other platforms and therefore require efficient and compatible data exchange. Most other platforms have a floating-point architecture that conforms, more or less, with the IEEE 754 standard. Also, Java implementations require adherence to this standard, and emulating the operations is slow and inefficient. Therefore, IBM wanted to give mainframe programs the ability to perform arithmetic in IEEE 754 floating-point format.

The G5 microprocessor contains a complete implementation of the IEEE 754 standard, along with continued full support for IBM's traditional HFP format. To distinguish between the two floating-point architectures, the IEEE-compatible version is called binary floating point, or BFP.

The new architecture adds 121 new instructions to the existing ESA/390 architecture: 87 BFP instructions, 26 additional HFP instructions, and eight support instructions. This is

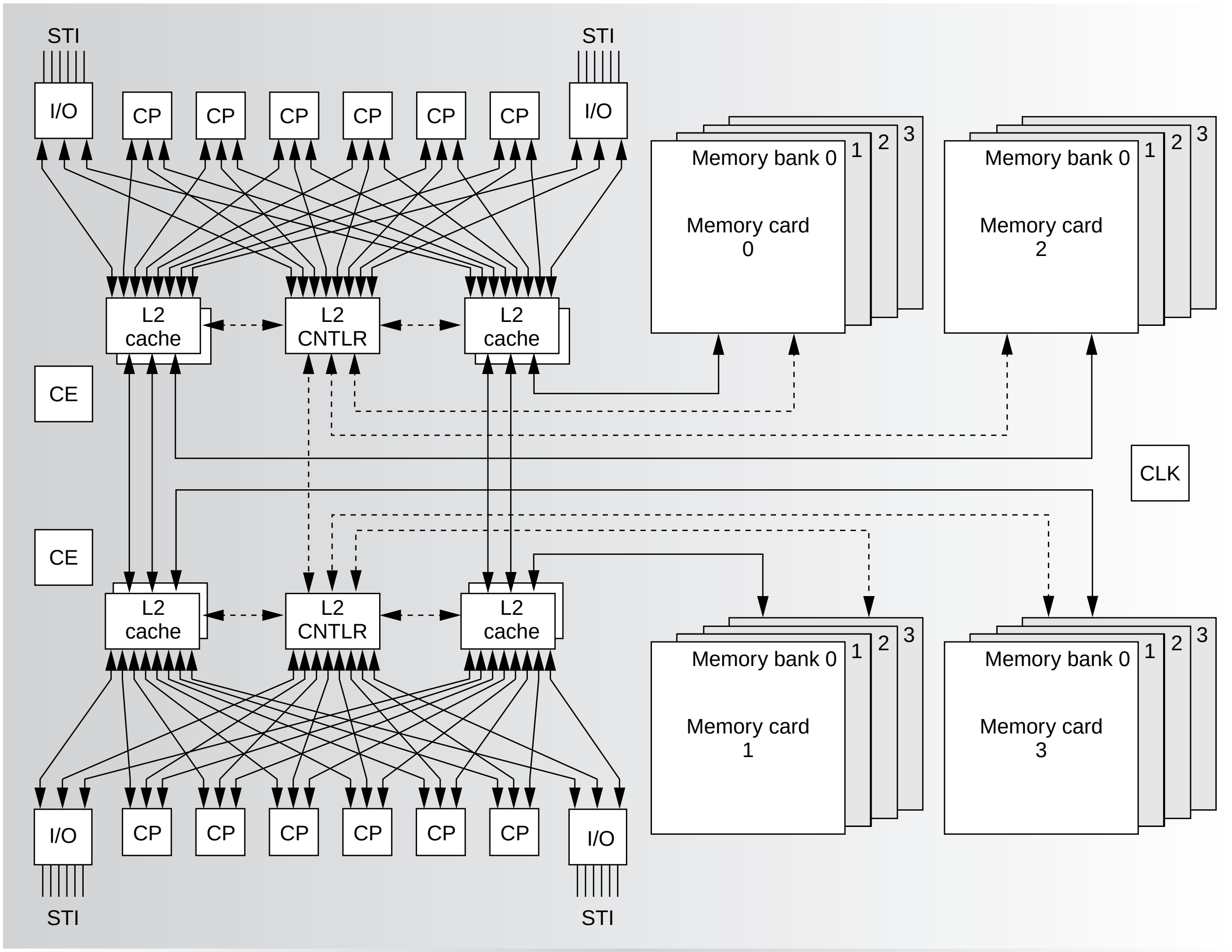


Figure 4. Diagram of the S/390 G5 system structure. Solid lines represent 128-bit data buses, and dashed lines represent address and control buses.

in addition to the 54 existing HFP instructions. Also, the number of floating-point registers has grown from four to 16, all of them 64 bits wide.

We expect that the traditional HFP instructions will be used predominantly for the next few years while applications are being developed to exploit the BFP architecture. Therefore, the G5 design is optimized to perform HFP arithmetic, while implementing BFP arithmetic as simply as possible. We accomplished this by evolving the G4 floating-point-unit (FPU) design point[9] to incorporate BFP.[10]

The FPU resides in the E unit. All floating-point operations are performed within the FPU in HFP format. When a BFP instruction is executing, the operands are first converted to HFP format immediately after passing through registers Areg and Breg in Figure 2. At the end of the arithmetic operation, the result is converted back to BFP format immediately before entering the Creg in the figure. This includes rounding required by BFP operations.

Obviously, there is much control logic, along with small pieces of dataflow logic, to handle the idiosyncrasies of BFP arithmetic and instances when numbers cannot be fully represented in HFP format. All three data formats (32-bit, 64-bit, and 128-bit), including all special-case operands, are handled in hardware without any millicode intervention. All this adds up to a design producing results that are fully compatible with the IEEE 754 standard while using relatively little additional die area compared with the G4 FPU.

The FPU operates on HFP operands in a fully pipelined manner at a rate of one result per cycle with three cycles of latency. For BFP, it is pipelined with two cycles per result with five cycles of latency.

System structure

In addition to excellent uniprocessor performance, the S/390 G5 system also excels in symmetric multiprocessor (SMP) performance. The G5 features up to 12 processors interconnected by a sophisticated system controller that contains the L2 cache and data-switching logic. The system structure is binodal, and each node contains essentially half a system. The two nodes are in a tightly coupled configuration and mounted on the same MCM. All processors in the system can access the entire memory with essentially uniform access time.

Figure 4 shows the G5 system structure. All data buses shown in this figure are 128 bits wide and run at 250 MHz. There are up to 12 processors, labeled CP (central processor) in the figure. A maximum of 10 can be configured as ESA/390 CPUs. The remaining CPs serve as I/O processors that manage the huge volume of data moving into and out of the I/O subsystem, or they may be configured as spares in the event of a processor failure.

The system controller consists of two chips, labeled L2 CNTLR in the figure. These contain the L2 directories and configuration array; they manage all data switching and buffers in the system controller. The system controller also contains eight L2 cache dataflow/array chips labeled L2 cache. Each of these chips contains 1 Mbyte of L2 cache and all data switches and buffers needed to move data between the processors, L2 cache, memory, and the I/O subsystem. The total L2 cache is large—8 Mbytes.

Each node of the system controller contains two I/O interface chips that connect with the L2 chips. The I/O subsystem is attached to these I/O interface chips via self-timed interfaces (STIs), with six STIs per chip. Each STI port runs at 333 Mbytes/s in each direction, and all 24 can run simultaneously, producing a total I/O bandwidth of 16 Gbytes/s.

The G5 system contains four memory cards, each with four banks. All four memory cards can transfer data simultaneously to yield a total memory bandwidth of 16 Gbytes/s. The maximum installable memory for the system is 24 Gbytes. (Memory above 2 Gbytes is used efficiently in the PR/SM environment or as expanded memory for paging.)

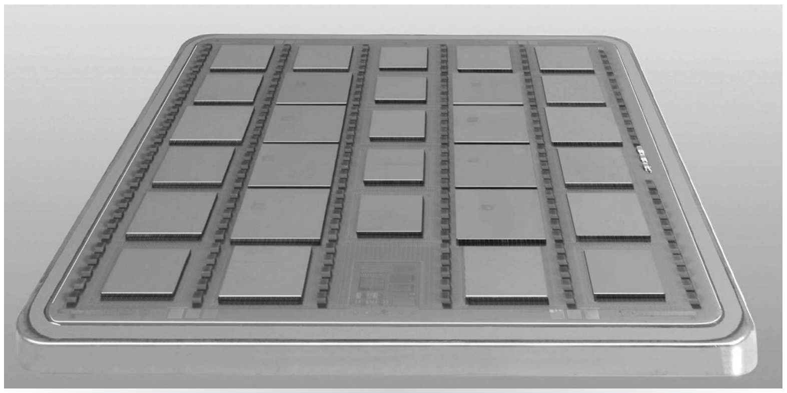


Figure 5. Photograph of the multichip module used in the S/390 G5 system.

The system also has two cryptographic coprocessor chips, labeled CE in the figure, that support RSA and DES functions in hardware. The MCM also contains a clock chip that distributes the master oscillator to all other chips and contains the service interface to the service element. The service element is a laptop computer running OS/2 that manages system operation.

Figure 5 is a photograph of the high-end MCM. In its lower end models, IBM ships essentially the upper half of Figure 4. This has a maximum of six processors, 4 Mbytes of L2 cache, two I/O interface chips, and two memory cards.

Reliability and availability

S/390 mainframe systems have long been known for outstanding reliability and availability. The G5 system continues this tradition with additional improvements in this area. CMOS is inherently an extremely reliable technology, and the vast majority of systems will never encounter a hardware error. Nevertheless, errors can and do occur. Since S/390 systems are often used for mission-critical and enterprise-wide applications, strict data integrity and continuous availability are key product requirements.

The microprocessor and many other system parts have essentially 100% error detection and recoverability from any transient error. In the rare occurrence of a solid error in the logic, several system features enable the use of redundant hardware, often automatically and transparently, to allow continued operation.

Processor hardware error recovery

Previous IBM mainframe systems were designed with traditional types of error-checking logic: parity, state checking, local duplication of control logic, and so on. The general rule of thumb was that 20% to 30% of all logic was devoted to error detection and recovery. This approach has several significant drawbacks. First, the checking logic is often in critical paths and can be a limiting factor in cycle time. It also introduces design complexities that can impact schedules. Finally, it is not well suited to a full-custom circuit design methodology, particularly in arithmetic units.

For the G4 and G5 microprocessors, the design team took a radically different approach, completely duplicating the I unit and E unit. On every clock cycle, signals coming from these units, including instruction results, are cross-compared in the R unit and the L1 cache. (These signals from the duplicated I and E units are shown by the dashed arrows in Figure 2.) If the signals don't match, hardware error recovery is invoked. This checking scheme solves the problems associated with traditional checking, although at an additional cost in die area.

The R unit and L1 cache use traditional error-checking approaches. All arrays in the L1 cache unit are protected with parity except for the store buffers, which are protected with ECC. Since the L1 is a store-through design, another valid copy of the data will always be in L2 or in memory. As an aside, since the L2 is a store-in design, it is protected by ECC, because it often holds the only valid copy of data.

If the R unit or L1 cache detects an error, the processor automatically enters an error recovery mode of operation. This process is done purely in hardware without any millicode intervention, since the processor may be in some indeterminate state that may not be able to run millicode. This error recovery mode also lets the processor recover while it is executing in millimode. The process includes the following steps:

1. The R unit freezes its checkpoint state and does not allow any pending instructions to update it.
2. The L1 cache forwards any store data that it may have buffered to the L2 for instructions that have already been checkpointed.
3. Certain critical latches in the I unit, E unit, and L1 cache are reset.
4. All arrays in the L1 cache unit and the BTB are reset.
5. Each R unit register is read out in sequence, with ECC logic correcting any errors it may find, and the corrected values are written back into the register file. In parallel, all shadow copies of these registers in the I unit, E unit, and L1 cache are updated.
6. All R unit registers are read a second time to ensure there are no solid correctable errors. If there are, the processor is check-stopped; that is, the clocks are stopped on that chip and it is no longer available for system operation.
7. The E unit forces a serialization interrupt, which restarts instruction fetching and execution. If recovery was successful, neither an ESA/390 program nor the operating system is aware that this recovery sequence has occurred.
8. An asynchronous interrupt tells millicode to log trace array and other data for later analysis by IBM product engineering.

Two conditions may cause recovery to fail: an uncorrectable error during step 5, or another error occurring during step 7 before an instruction is successfully completed. Both cases result in a check-stop condition, and another type of recovery action will then be invoked.

This whole sequence takes several thousand clock cycles to complete, but since hardware errors are so rare, performance is not critical. For any type of hardware error in the processor (E unit cross-compare, cache parity error, L1 control error, and so on), this same recovery algorithm is invoked, since it can handle all cases.

Array recovery features

The algorithm is essentially 100% effective for any type of transient error in the processor. However, solid errors can occur—though less frequently. The most common type of solid error occurs in arrays, since they take up a significant portion of the die. The G5 processor features an automatic array delete mechanism to deal with such errors. There is logic that records the cache address when a cache parity error is detected. If another cache parity error occurs on the same physical line or set, the logic automatically deletes the corresponding line or set from further use. Processor operation then continues normally.

Most arrays have built-in redundant word-lines to increase manufacturing yield. Chip testing uses a laser to blow fuses and thereby utilize these spare word-lines and make all arrays 100% usable when the chips leave the factory. However, the arrays rarely have defects when the chip is built, so these redundant word-lines go unused. In the G5 we added the capability to exploit these spare word-lines to automatically replace defective sections of an array at a customer's site. The service element runs self-test on the chips at power-on reset time, and any marginal arrays can be detected. The service element then implements an algorithm that scans values into latches on the chip that mirror the fuses. This causes the redundant word-lines to be used instead of the failing section of the array.

System recovery features

When a solid error occurs that is not in an array, the processor will go into a check-stopped state. For many years, IBM's mainframe systems have had several different mechanisms that attempt to recover the job running on a check-stopped processor and to prevent system performance degradation.

One of these mechanisms is the processor availability facility (PAF). The service element scans out the latches from the check-stopped processor and extracts the ESA/390-architected state. It then sends this data back to the system and stores it in an area set aside for machine check interrupt. The operating system, which is still running on the system, is informed via a machine-check interrupt that a processor has check-stopped. The operating system can then use this saved data to resume executing that job on another processor. In this case, the check-stop is not visible to the application program that ran on the failed processor.

Another system recovery mechanism is concurrent processor sparing. In most cases, when a customer orders a certain number of processors in an SMP system, IBM actually ships one or more spare processors on the MCM. These spare processors are not visible to the customer. In a running system, they execute a millicode idle loop. Upon a processor check-stop, the customer can issue a command on the console that lets the operating system use one of these spare processors. The combination of PAF and concurrent sparing effectively gives the customer a full-performance system without loss of any jobs.

IBM typically ships SMP systems with one or more spare processors, though they are not visible to the customer.

Finally, concurrent I/O processor sparing is an automatic mechanism for allowing a spare processor (or a functional processor) to be placed into service as an I/O processor in the event of a check-stop on one of the I/O processors.

Transparent processor sparing

All of the mechanisms just described provide excellent recoverability from processor check-stops. However, there are some drawbacks. Some do not work on uniprocessor systems or in certain logically partitioned environments. Also, the customer is aware when one of these mechanisms is invoked. The G5 system, however, introduces an even more advanced recovery scheme called transparent processor sparing. This mechanism moves the microarchitected state of a failed processor to a spare processor in the system.

With transparent processor sparing, when a processor check-stops, the service element scans out all latches on the failed processor. It then processes this data to extract the contents of all R unit registers that contain the complete checkpointed microarchitected state of the processor. In parallel with this, all remaining processors are notified of the failure and can select, under millicode control, which spare processor will take over for the failed processor. The service element then sends the state information back to the system, where it is temporarily saved in a work area in storage. The millicode running on the spare processor then makes any required changes to the microarchitected state in storage—for example, changing the processor ID. The millicode then executes a hardware instruction that loads the entire contents of its own R unit from storage in one atomic operation. When this instruction completes, the spare processor begins fetching and executing instructions where the failed processor stopped.

This entire process is automatic and totally transparent to the operating system and the customer. Transparent processor sparing is also effective when a processor is executing in millimode at the time of failure.

The IBM S/390 G5 system realizes a performance increase well above the industry growth curve. The system's processor achieves excellent performance, and it incorporates reliability and recovery features that we feel surpass those of other systems. We look for these trends to continue in future generations of IBM's mainframe systems, with higher clock frequencies, more processors per system, and further architectural extensions. MICRO

Acknowledgments

We recognize the physical design, array, test, simulation, and millicode teams for their contributions to the G5 microprocessor. In particular, we thank James Andre, Dean Bair, Michael Campbell, Carl Dorestant, Mark Farrell, William Lewis, Paul Moore, Michael Mullen, Robert Murray, Ashok Shenoy, and Scott Swaney.

References

1. *Enterprise Systems Architecture/390 Principles of Operation*, order number SA22-7201, available through IBM branch offices.
2. G.S. Rao et al., "IBM S/390 Parallel Enterprise Servers G3 and G4," *IBM J. Research and Development*, Vol. 41, No. 4/5, July/Sept. 1997, pp. 397-403.
3. C.F. Webb and J.S. Liptay, "A High-Frequency Custom CMOS S/390 Microprocessor," *IBM J. Research and Development*, Vol. 41, No. 4/5, July/Sept. 1997, pp. 463-473.
4. C.F. Webb et al., "A 400 MHz S/390 Microprocessor," *IEEE J. Solid-State Circuits*, Vol. 32, No. 11, Nov. 1997, pp. 1,665-1,675.
5. T.J. Slegel et al., "IBM S/390 G5 Microprocessor," *Hot Chips 10 Conf. Record*, Aug. 1998 (copies available from the author).
6. R.M. Averill III et al., "Deep Submicron Design Techniques for the 500 MHz IBM S/390 G5 Custom Microprocessor," *Proc. Int'l Conf. Computer Design*, IEEE Computer Soc. Press, Los Alamitos, Calif., Oct. 1998.
7. *Large Systems Performance Reference*, order number SC28-1187, available through IBM branch offices.
8. K. Hua et al., "Early Resolution of Address Translation in Cache Design," *Proc. Int'l Conf. Computer Eng.*, Sept. 1990.
9. E.M. Schwarz, L. Sigal, and T.J. McPherson, "CMOS Floating-Point Unit for the S/390 Parallel Enterprise Server G4," *IBM J. Research and Development*, Vol. 41, No. 4/5, July/Sept. 1997, pp. 475-488.
10. E.M. Schwarz, R.M. Smith, and C.A. Krygowski, "The S/390 G5 Floating-point Unit Supporting Hex and Binary Architectures," *Proc. 14th IEEE Symp. Computer Arithmetic*, to be published by IEEE Computer Soc. Press, Los Alamitos, Calif., Apr. 1999.

Timothy J. Slegel is a senior engineer with the IBM S/390 Server Division in Poughkeepsie, New York, working on the design of future microprocessors. He was the overall technical leader for the G5 microprocessor and has worked in various areas of processor design, including arithmetic units, vector processors, and caches. He received BSEE and MSEE degrees from Lehigh University.

Robert M. Averill III is an advisory engineer in IBM's Poughkeepsie hardware laboratory and the physical design leader for a future microprocessor. His work includes high-performance deep-submicron CMOS microprocessor design. He received a BS from Northwestern University and an MS from Syracuse University, both in electrical engineering.

Mark A. Check is an advisory engineer in the custom microprocessor development group with the IBM S/390 Server Division. He received a BSEE from the University of Wisconsin–Madison and an MSCE from Syracuse University. He is a member of the IEEE.

Bruce C. Giamei is an advisory engineer in the custom microprocessor development group. He worked on CPU design for the

ES/9000 processor and the S/390 G4 and G5 CMOS processors. He received a BSEE and an MSEE, both from Purdue University.

Barry W. Krumm, an advisory engineer in the IBM S/390 Server Division, led the L1 cache design team for the G4 and G5 microprocessors. He has worked on the logic design of the buffer control element and storage control element for large system processors, including the bipolar ES/9000 systems. He received a BA in computer science modified with electrical engineering from Dartmouth College.

Christopher A. Krygowski, an advisory engineer with IBM's Server Division in Poughkeepsie, is leading the logic design of the execution unit of a future S/390 microprocessor. His research interests include floating-point arithmetic, high-frequency microprocessor design, and fault-tolerant computing. He received a bachelor's degree in electrical engineering from Clarkson University and will receive a master's in electrical engineering from National Technological University in Boulder, Colorado.

Wen H. Li is an advisory engineer with IBM's S/390 Server Division in Poughkeepsie. She has worked on logic and circuit design for the execution unit of microprocessors. She received a BSEE and an MSEE from City College New York.

John S. Liptay, an IBM senior technical staff member, worked on the design of S/360, S/370, and S/390 processors. He received a BEE and an MEE, both from Rensselaer Polytechnic Institute. He is a member of the ACM and the IEEE.

John D. MacDougall is a staff engineer and a technical leader in the development of future high-frequency CMOS microprocessors. He is on assignment at IBM Boeblingen (Germany) as a liaison between the Poughkeepsie and Boeblingen design centers. He has worked in both the logical and physical design areas of the S/390 Division. MacDougall received a BS in computer engineering from Worcester Polytechnic Institute and is completing his MS in computer engineering at Syracuse University.

Thomas J. McPherson, an advisory engineer in S/390 microprocessor development, was the frequency leader for the G5 microprocessor and is now working on future S/390 microprocessors. He received a BS in electrical engineering from Rutgers University and an MS in computer engineering from Syracuse University.

Jennifer A. Navarro is an advisory engineer in the S/390 Division at IBM in Poughkeepsie. She is a lead designer of L1 cache logic for a future CMOS microprocessor, having worked on both logic and circuit designs in the G4 and G5 systems. She holds a BS in electrical engineering from Polytechnic University, New York.

Eric M. Schwarz is a senior engineer in IBM's S/390 Division in Poughkeepsie, where he is CPU logic technical leader for a future mainframe processor. His research interests include both computer arithmetic and computer architecture. He received a BSc in engineering science from Pennsylvania State University, an MSc in electrical engineering from Ohio University, and a PhD in electrical engineering from Stanford University. He is a member of the IEEE and the IEEE Computer Society.

Kevin Shum, an advisory engineer, is leading the cache design for a future microprocessor. He has worked on logic and circuit design for various IBM mainframe central processors and on front-end design processes and methodologies within the microprocessor team. He received BS and MS degrees in electrical engineering from Columbia University.

Charles F. Webb is a senior technical staff member at IBM's Poughkeepsie laboratory, where he works on CPU and system design for S/390 processors. His technical interests include microarchitecture, server system design, and performance analysis. He received an MEng. in computer and systems engineering from Rensselaer Polytechnic Institute and is a member of the IEEE Computer Society.

Direct questions concerning this article to Timothy J. Slegel, IBM Corp., MS P310, 522 South Road, Poughkeepsie, NY 12602; slegel@vnet.ibm.com.