Custom S/390 G5 and G6 microprocessors

by M. A. Check
T. J. Slegel

Compared with the G4 microprocessor, the S/390® G5 microprocessor contains many architectural and performance enhancements. The G6 microprocessor represents a technology performance improvement over G5, with system support for additional processors. The G5 processor uses IBM CMOS 6X technology and has a clock frequency of 500 MHz in its fastest models. The G6 uses CMOS 7S technology with a clock frequency up to 637 MHz. The processors include a new IEEE binary floating-point architecture and additional reliability–availability–serviceability (RAS) improvements. The processor has significant performance improvements, including a larger level-1 (L1) cache, enhancements to the instruction fetch buffers, a branch target buffer (BTB), enhancements for a number of instructions, a new quiesce mechanism for instructions that modify translation lookaside buffer (TLB) entries, and a new level-2 (L2) cache and memory subsystem.

Introduction

In 1994 the S/390* Division of IBM began the transformation from bipolar technology to CMOS. The fourth-generation G4 system was generally equivalent in performance to the fastest IBM bipolar systems. The G5, the fifth generation of CMOS machines, has a design goal to provide a significant performance improvement to the S/390 customers, as well as new architectural features to enable new applications. The G5 system increases performance more than twofold over that of the G4 system. It also provides an IEEE-compatible binary floating-point architecture for S/390 in addition to its existing hexadecimal format. The S/390 system platform has provided customers state-of-the-art RAS. These generations offer additional improvements, the most significant of which is transparent processor sparing. The G6 generation offers an additional processor performance improvement due to the technology improvement and additional system performance from two more processors.

The processor improvements will be coupled with a new L2 cache and memory subsystem. This new L2 and memory structure provides a much-improved multiprocessor efficiency and system extensibility over that in the G4 system. For further information, please see the paper by Turgeon et al. [1] in this *Journal* issue, which describes the design of the L2 cache and memory subsystem.

The design of the G5 and G6 microprocessors is based on that of the G4 [2, 3]. The processor design is relatively simple in that it is optimized for a very high clock frequency to obtain excellent performance. A high-level diagram in **Figure 1** shows the functional unit partitions and functional contents of these microprocessors. The processor executes the ESA/390* instruction set architecture [4]. This is a rich Complex Instruction-Set Computer (CISC) architecture that requires some unusual tradeoffs in the design of a processor. For example, the processor is not superscalar but instead tries to reduce the number of clock cycles required for the long-running complex instructions. To simplify the logic design, the processor uses millicode [2] (a form of Licensed Internal

0018-8646/99/$5.00 © 1999 IBM

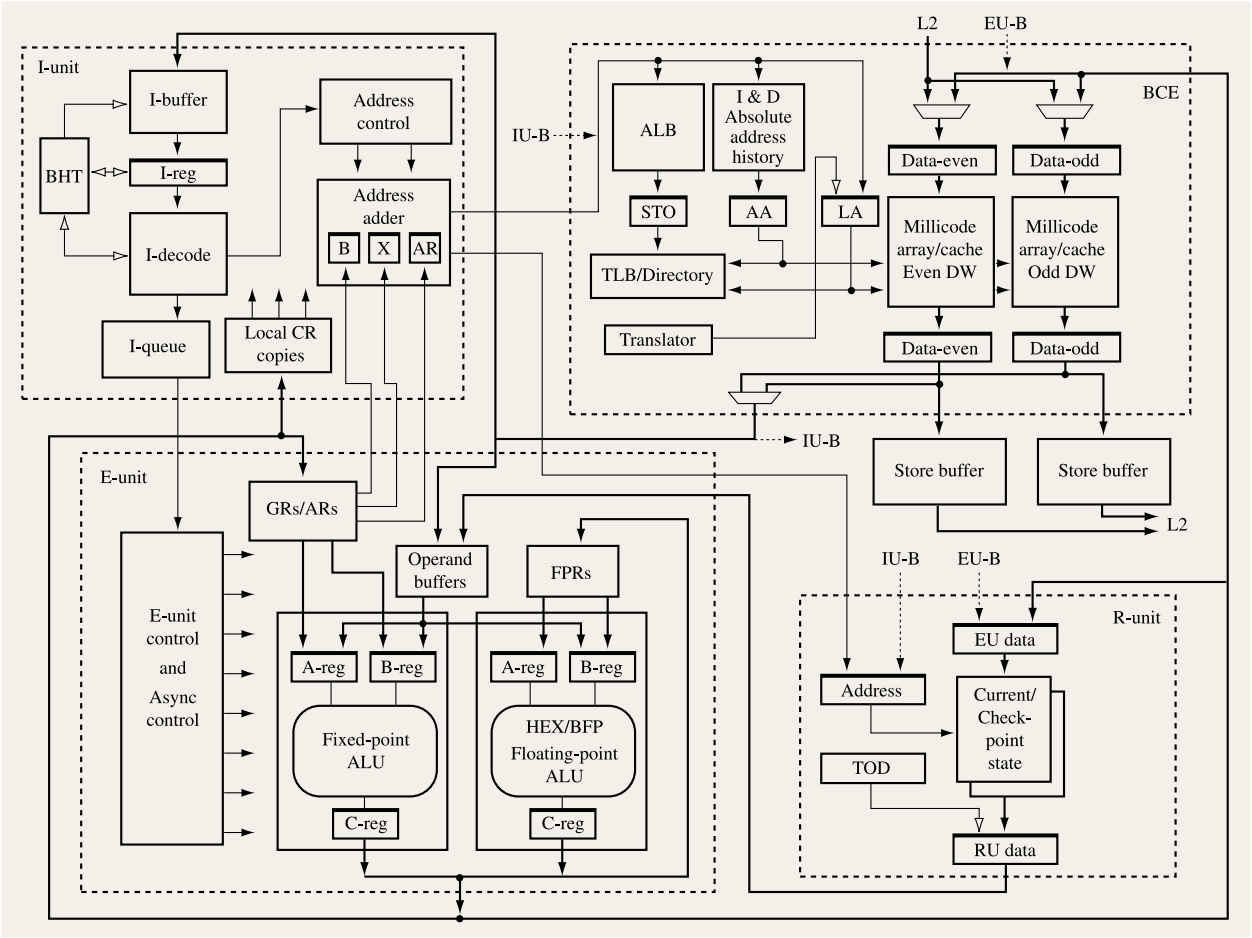

Figure 1

CP high-level overview.

Code), which is the vertical microcode that executes on this family of processors to implement many complex elements of the architecture. Millicode consists of all of the hardwired ESA/390 instructions plus 102 hardwired assist instructions that can only be executed by millicode.

Performance improvements

Providing significant performance improvements for the S/390 systems from generation to generation required improvements in many areas, including technology, cycles per instruction (CPI), multiprocessor efficiency, and size of the multiprocessor structure. The new memory subsystem structure provided much-improved finite cache effects for multiprocessor efficiency in the G5 and G6 systems, as well as extensions for the G6 system to provide for two additional processors. Each successive processor generation exploited one newer CMOS technology generation. The G5 processor has numerous CPI enhancements, with a small amount of additional tuning for the G6 processor.

- *Technology*

The G5 [5] system is fabricated with IBM CMOS 6X technology. It is a 0.25-μm technology with an 0.15-μm L_{eff} (on the n-FET) and operates at 1.9 V. The processor runs at a clock rate of 500 MHz in the fastest versions, which have a self-contained chiller unit. The rest of the cache and memory subsystem is fabricated with CMOS 6S technology and operates at 250 MHz. The CMOS 6X process has six levels of aluminum wiring and one local wiring level. The G6 processor and L2 cache are fabricated with IBM CMOS 7S technology, which is a 0.20-μm technology with a 0.12-μm L_{eff} and uses copper wiring.

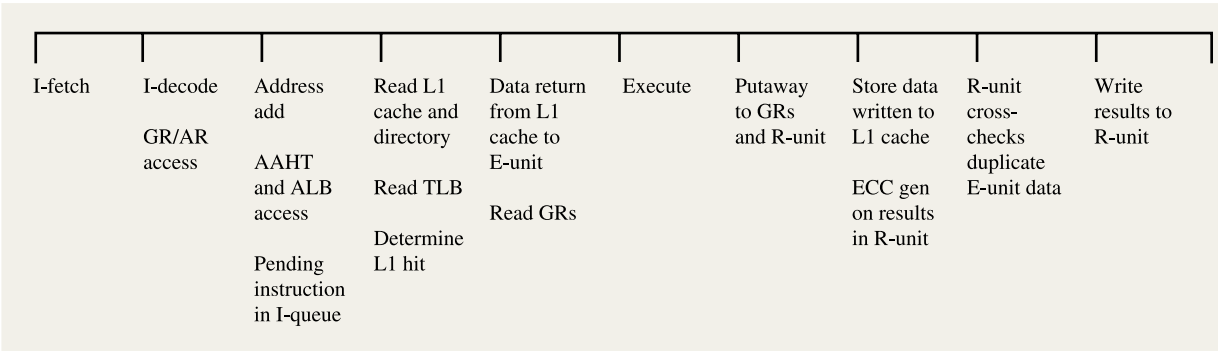

Figure 2

CP pipeline.

To take advantage of the full capability of the technology, significant emphasis was placed on the timing analysis [6] to ensure that the processor is not limited by paths that are dominated by wire delay. This is important to make it possible to use line tailoring of the final product release, allowing the silicon paths to run faster than originally designed. This would not be possible if the limiting paths were wire-delay-dominated, because the faster transistors do not reduce wire delay. In addition, small adjustments have been made to V_{DD} and temperature to achieve the frequency at the high end of the family of systems. The initial modeling of the circuit design is done to allow the exploitation of these adjustments near the end of the design cycle.

- *Cache structure improvements*

The unified L1 cache size has been quadrupled to 256K bytes, with an increase in the line size from 128 bytes to 256 bytes. The line size has been increased to keep the L1 and L2 line sizes consistent and limit the timing effect on the size of the L1 directory arrays. It also allows for some access patterns to bring fewer lines into the cache at the cost of longer transfer times of the line from L2 to L1, which overall had a slight improvement in performance. With this also came increases in the size and associativity of the TLB from a 256-entry two-way to a 1024-entry four-way. In addition, the entire L2 cache and memory structure has been redesigned for the G5 system. The change from a bus-based to a point-to-point switch-based memory subsystem improved the finite cache effects significantly.

In order to process a request for data from the cache, the following steps must be completed: Generate the virtual address; translate the access-list-entry token (ALET) to a segment table origin (STO) using the access register translation lookaside buffer (ALB); translate the virtual address to an absolute address using the TLB; access the cache directory to see where the data is located in the cache; and read the data from the cache. For performance considerations, these steps must not be performed sequentially. As can be seen from **Figure 2**, which shows the instruction pipeline, several of these steps are done in parallel. In the G4 system, a structure called an absolute address history table (AAHT) [2, 7] was used to allow several of these access functions to occur simultaneously. The AAHT structure is used to solve the problem of synonyms. Synonyms occur when bits of a virtual address that are subject to translation are used to address an absolute-addressed cache, because different virtual addresses may map to the same absolute address. Since it is not known what the virtual-to-absolute translation will produce, there are multiple possibilities. The AAHT provides an initial prediction of the value that will be returned from translation, and that value is used for the cache access. Storage pages are 4K in size, and thus bits 1 to 19 of the virtual address are subject to translation. The G4 processor had 64 KB of L1 cache that was four-way associative, with even and odd doubleword interleaves, which required bits 18 to 28 to address it. Bits 18 and 19 were subject to translation. The AAHT predicts the absolute address bits for the cache request before the lookup in the TLB would be completed. This structure has been enhanced for G5 and G6 as in **Figure 3**, including a separate AAHT for instruction fetching.

In this mechanism, bits from registers that form the virtual address are used to index an array which contains a guess for the absolute address bits for that page index value. Improvements have been made to the mechanism that was used in G4 to improve its accuracy. The original G4 design used bits from the base or index register values for operand requests and a value based on the current instruction stream for instruction requests. From

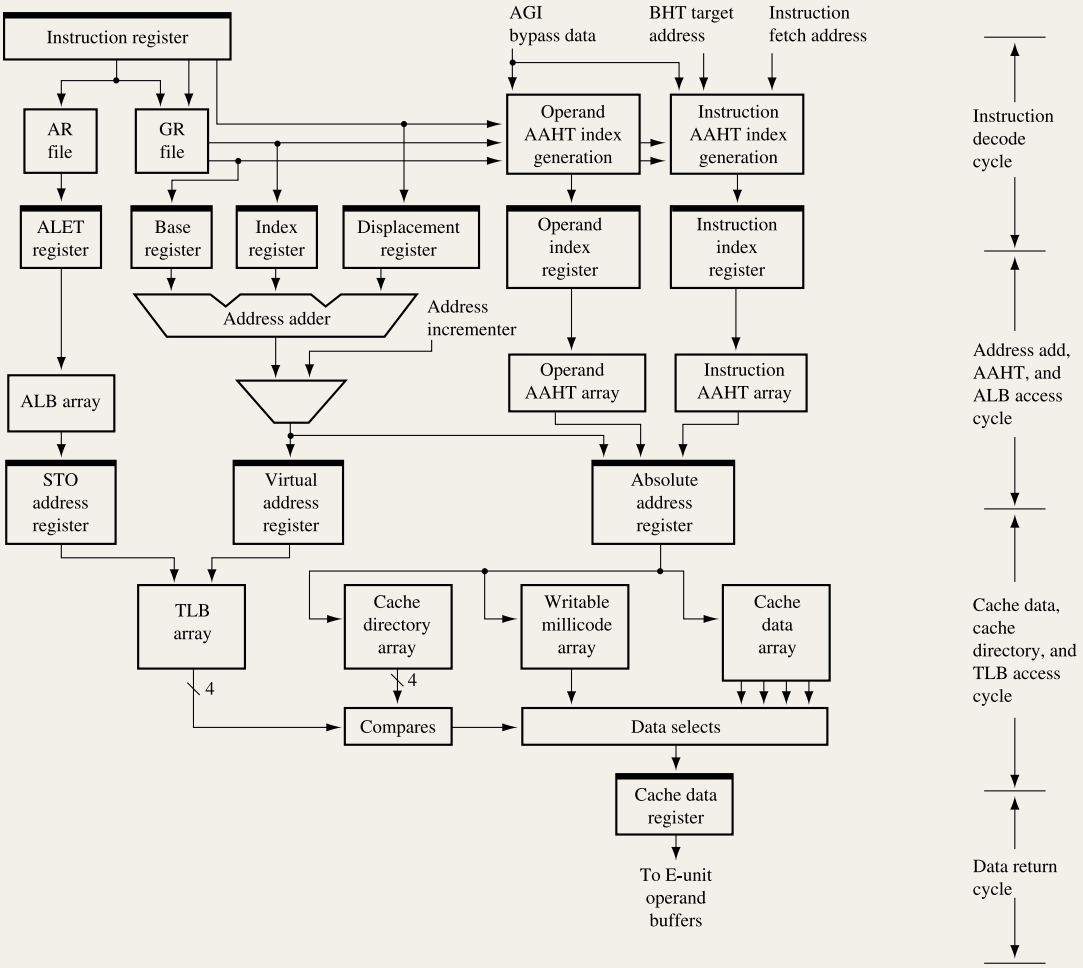

Figure 3

AAHT structure.

performance analysis of operation from the G4 system, it was found that improvements could be made to the operand index value. The inclusion of the BTB would require further changes to the instruction absolute address bit prediction. In the G4 system there were only two absolute address bits that had to be predicted. In the L1 cache system of the G5 system there is a 256KB cache that is four-way associative with two interleaves. This requires that bits 16 to 28 be used to address the cache. Now there are four bits, 16 to 19, that are subject to translation, requiring four bits to be predicted. Predicting a greater number of bits requires these improvements in order to enhance processor performance.

For instruction fetching in the G4, the two bits were predicted on the basis of the bits for the current instruction stream. For most branches, which are normally short, this would be correct. If the branch distance were longer and to a different line, there would still be a 25% chance of guessing correctly. However, with four bits to guess, there would be only a slightly greater than 6% chance that it would be correct for long branches. The BTB would thus be trying to predict the targets of branches and prefetch those targets. If the current absolute bits of the instruction stream were to be used for these branches, the prediction would often be wrong when the branch target was to another line. This would use directory cycles which are already in high demand because of the unified cache.

The AAHT index and array for operand requests existed in the G4 processor, and a similar structure for

instruction requests has been added in the G5 processor. Once the structure exists, it can be used to predict absolute bits for surprise branches. In the case of branches, the branch address is rarely based on both a base and an index register. For these surprise branch target instruction fetches, only the base or index register contents are used, depending on which register the branch instruction used. The BTB entry incorporates the virtual target address. Thus, the value of the instruction AAHT table index is based on bits of the virtual address of the current instruction address, BTB target address, or part of the target address.

For the operand absolute address history array index, the G4 used only bits from the base or index register. If the instruction used a base register, only bits from the base register were used. If a base register was not used, only bits from the index register were used. This method was found to be less than optimal for several important cases. The first is when the effective address uses both base and index values, or when a large displacement value is used, such as in a table lookup. For this problem, the ideal case would be to have the full effective address for the table lookup. However, the effective address is being calculated during the same cycle as the AAHT array access and is not yet available. Thus, a quick hash function based on the bits of the base, index, and displacement fields is performed during the instruction decode cycle, which is the cycle prior to the AAHT array access. The second important case concerns an instruction which updates a general register (GR) that is still pending execution when a subsequent instruction that requires the use of that register to generate its storage address is decoded. For performance reasons, the processor allows data from internal buffers for LOAD (L) or LOAD ADDRESS (LA) instructions to be used to generate the correct storage address before one of these instructions completes execution. In this case, the data for the effective address add is not coming from the general registers but is being passed from the internal buffers from a previous instruction. This data must be used in place of the general register output values.

- *BTB and instruction fetch buffers*

Significant improvements in performance have been gained with the inclusion of a BTB (**Figure 4**) and more instruction fetch buffers in the G5 processor. The BTB contains 2048 entries in a 2 × 1024-entry two-way associative format. The instruction buffers are increased from four 32-byte buffers in G4 to six 32-byte buffers in G5. This allows the G5 processor in all but the tightest loops to find branches and fetch their target instruction streams before the branches are decoded.

The BTB in the G5 holds only taken branches and uses a two-bit algorithm to indicate strongly taken, weakly taken, and changing target branches. This enables the BTB to deal with branches that change direction or target frequently. When a taken branch is first encountered, the branch is installed as strongly taken. If a branch in the BTB currently marked strongly taken is subsequently not taken, it is moved to weakly taken. If a weakly taken branch is not taken, it is removed from the BTB. A branch that is currently weakly taken is changed to strongly taken if the branch is taken again. This allows better prediction in cases where the branch is alternating between taken and not taken. For changing target branches, the entry is marked as such. In this case, instruction fetching is not started to an unreliable address, since more time would be required to move to the correct stream when the real target is known. Instead, instruction fetch waits until the branch reaches the address computation stage and fetches the correct target address.

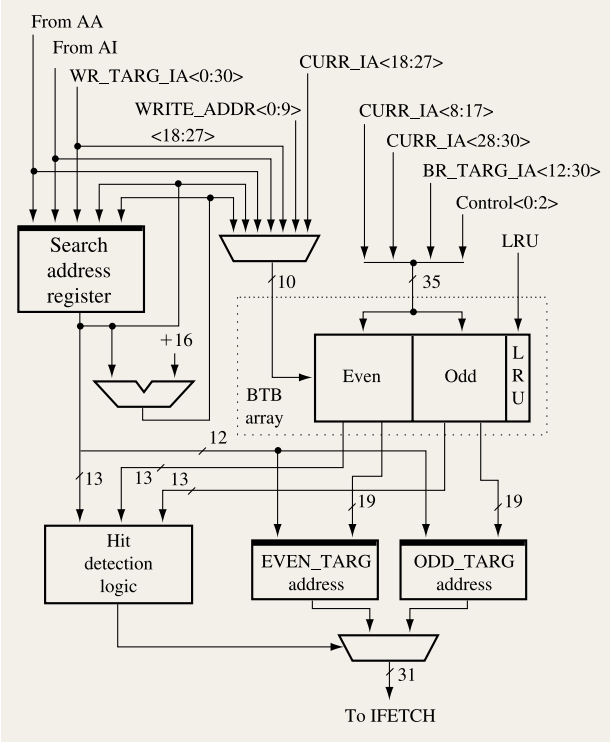

Figure 4

BTB structure.

The G5 BTB also holds branch information on all types of instruction code, including normal S/390 code and all forms of Licensed Internal Code (LIC) such as millicode. This information is carried with the entry and allows the internal code to benefit from branch prediction of the BTB in the same way as S/390 code. In the past, either the

internal forms did not benefit from the BTB, since it could not distinguish between the different branches, or two different structures would have to be created. A single control bit is used to distinguish between the S/390 and all forms of LIC.

The array contains 2048 35-bit entries which are indexed by 10 bits of search address that each contain 13 bits of current instruction address, 19 bits of branch target address, and three control bits. There is a least-recently-used (LRU) bit for each of the set entries. Special attention is given to the handling of the LRU bit when a tight loop is encountered. It is not desirable to have the same branch written into both sets for a given entry address. LRU updates are also performed each time an existing branch is encountered. In the G6 system, a mechanism has been added so that instruction decode is prevented for a few cycles when a tight loop is detected that the BTB is not able to search and predict soon enough. This allows the BTB to predict the branch loop. The performance gain from this step is greater than the cost of the delayed decode cycles. To limit the size of the array, the number of instruction address and branch target address bits is less than the full address. The number split between current instruction address and target address was determined on the basis of performance modeling. The remainder of the bits are assumed to be the current instruction address bits in order to generate the predicted target address.

Once a matching entry is found in the BTB, the target address is forwarded to the instruction fetch logic, which requests the target instruction stream from the cache. At the time the branch is decoded, the correct target address is calculated; it is then used to ensure that the predicted address is correct. If so, the target has already been fetched, and decoding of the target stream occurs during the next cycle. If the target was incorrect, a *branch wrong target* is indicated with the branch execution, and the correct target stream is then fetched.

The array is searched at a rate of 16 bytes per cycle. The search is conducted on the current 256-byte line and the next sequential line. The search can be started from a number of different locations including the current instruction address, the next sequential line of the current address, the target address of a surprise branch predicted as taken (one that is not BTB-predicted), a BTB-predicted target address, the correct address of a branch that was predicted as taken but with an incorrect target, or the target address of a branch predicted as not taken that was actually taken.

The BTB also has a mechanism to turn itself off and on under millicode control. This is done because there are some functions that require tight control on cache activity while operations to the rest of the system structure occur. Since the BTB searches and fetches branch targets on its own, it is temporarily turned off in order to prevent these unexpected fetches. There are also a few specific loops in certain performance-sensitive millicoded instructions for which, because of the nature of their function, performance is better with the BTB turned off. The presence of this mechanism allows millicode to exploit this capability.

The instruction buffers in this family of processors are each 32 bytes in length and are assigned one per instruction path by cache line. As a doubleword of instruction text is consumed in instruction decode, further doublewords are fetched into the buffer. As fetching nears the end of a line, a different instruction buffer is assigned to fetch the next sequential line. In the G5 system, the number of instruction buffers has been increased from four buffers to six to provide the best use of predictions from the BTB. In the four-buffer G4 system, one would be used for the current line, one for the next sequential line, and one each for two possible outstanding branch paths. Thus, the BTB would predict a future branch, but it would not be possible to fetch the target stream in some cases. With two more buffers added, the targets of predicted branches can be fetched or allow sequential lines of branch targets near the end of lines to be fetched as well.

- *Instruction performance enhancements*

The G5 processor includes performance improvements for two classes of instructions. The first is the set of instructions that operate on decimal data. Many applications used on S/390 systems, particularly in the financial industry, require good decimal performance. The G4 system included an eight-digit decimal adder and performed add/subtract/compare operations in hardware. The remainder of the decimal instructions were implemented using millicode. The G5 design includes a decimal multiplier that uses a lookup table to generate partial products, with the entire decimal multiply instruction done under hardware control. For decimal divide, a hardware assist instruction is included which produces one digit of the quotient. Millicode uses this assist instruction iteratively to produce the full quotient. Finally, there is dedicated hardware to implement the convert to/from integer instructions and the pack/unpack instructions under complete hardware control. The two instructions CONVERT TO BINARY (CVB) and CONVERT TO DECIMAL (CVD) switch an operand between binary and decimal formats. The PACK (PACK) and UNPACK (UNPK) instructions convert from one to the other of the two decimal data formats (“zoned format” and “packed format”) described in Chapter 8 of the *ESA/390 Principles of Operation* [4].

The other class of instructions that achieve a significant performance improvement over G4 are those that manipulate the program status word (PSW), including

LOAD PSW (LPSW), SET SYSTEM MASK (SSM), STORE THEN OR SYSTEM MASK (STOSM), STORE THEN AND SYSTEM MASK (STNSM), SET ADDRESS SPACE CONTROL (SAC), SET ADDRESS SPACE CONTROL FAST (SACF), and SET PSW KEY FROM ADDRESS (SPKA). These instructions change the fundamental state of the processor in one or more of the following areas: enablement of dynamic address translation, address space where subsequent instructions and/or operands will be fetched, and enablement of interruptions. These instructions are architecturally complex, and the G4 design used millicode to implement them. Avoiding the process of entering and exiting millicode for these instructions can save seven to ten cycles for each execution. For a few of these instructions, determination of certain conditions via millicode instructions can take multiple cycles, whereas the hardware-executed implementation can be performed in a single execution cycle.

A common thread that runs through the implementation of many of these instructions is whether or not processor operations must be serialized after their execution. In the processor, serialization is defined as discarding partially executed instructions that have not yet completed and any instructions that may have been prefetched. Since a serialization action costs approximately 12 clock cycles in the G5 processor, it must be avoided if possible. Serialization may be required either for ESA/390 architectural reasons (although it may be possible to avoid it in many cases, given certain characteristics of the G5 microarchitecture for which it is known that the ESA/390 architectural requirements have been met in other ways), or for practical reasons based on the implementation.

The SPKA instruction changes the storage protection key used to access subsequent instructions and data. In the G5 processor, serialization is normally not required following the SPKA in order to meet the ESA/390 architectural requirements, since instruction decode and therefore operand fetches are inhibited after an SPKA decodes and until it completes. The problem is that a subsequent instruction is likely to have been prefetched and already in an instruction buffer. The L1 cache would have checked for I-fetch data for protection exceptions against the old PSW key and would not be aware that it has been changed. The G5 includes logic to carefully monitor prefetched instructions against changes made to the key via an SPKA instruction. If it detects a possible conflict, it serializes the processor to allow a protection exception to be taken when the data is refetched. However, this serialization is now a rare occurrence and does not affect performance.

The STOSM and STNSM instructions modify the system mask portion of the PSW and receive special treatment to avoid unnecessary serializations. Although these instructions can modify any portion of the system mask, they are typically used to enable and disable I/O and external interrupts. If one of these instructions modifies any other portion of the system mask, the logic unconditionally serializes the processor. If the instruction modifies only the interrupt mask bits, serialization is normally not performed. However, the ESA/390 architecture requires an interrupt to occur immediately following an STOSM if an asynchronous interrupt is pending. Therefore, there is logic in the G5 which detects that an interrupt may be pending regardless of the current interrupt mask settings. If one is, the processor is serialized to allow the asynchronous interrupt logic to correctly present the interrupt. In addition, owing to a delay of several clock cycles in the "handshaking" between the PSW logic and the asynchronous interrupt logic, interrupts are artificially blocked, and the processor is serialized if a pending interrupt is detected during this time.

- *Quiesce mechanism*

The ESA/390 architecture contains two relatively frequently used instructions that can have a significant performance impact on larger SMP systems. These are INVALIDATE PAGE TABLE ENTRY (IPTE) and SET STORAGE KEY EXTENDED (SSKE). In both cases, the ESA/390 architecture requires local copies of page table entries or storage keys, typically kept in the TLB, on other processors to be invalidated before the instruction completes on the issuing processor.

IBM has used several different implementations in past systems to achieve the architectural requirements. A commonly used one (and the one used on the G4 system) worked on the principle of quiescing operations on all processors during the execution of one of these instructions. When a processor, called the master, began executing an IPTE or SSKE instruction, it would first notify the system control (SC) element that it wanted to quiesce the system. The SC would then broadcast this quiesce request to all processors in the system, called slave processors, as an interrupt. When each slave processor reached an interruptible point, it would inform the SC that it was now quiesced and would wait in a millicode loop for further commands. Only when all slave processors reached a quiesce point would the master processor issue a command to have them make the appropriate changes to their TLBs. The master processor, after modifying its own TLB, would then modify the common facility: a page-table entry in storage for IPTE or a storage key for SSKE. Finally, the master processor would issue a command to all slave processors informing them that the quiesce had completed and they were free to continue normal operations.

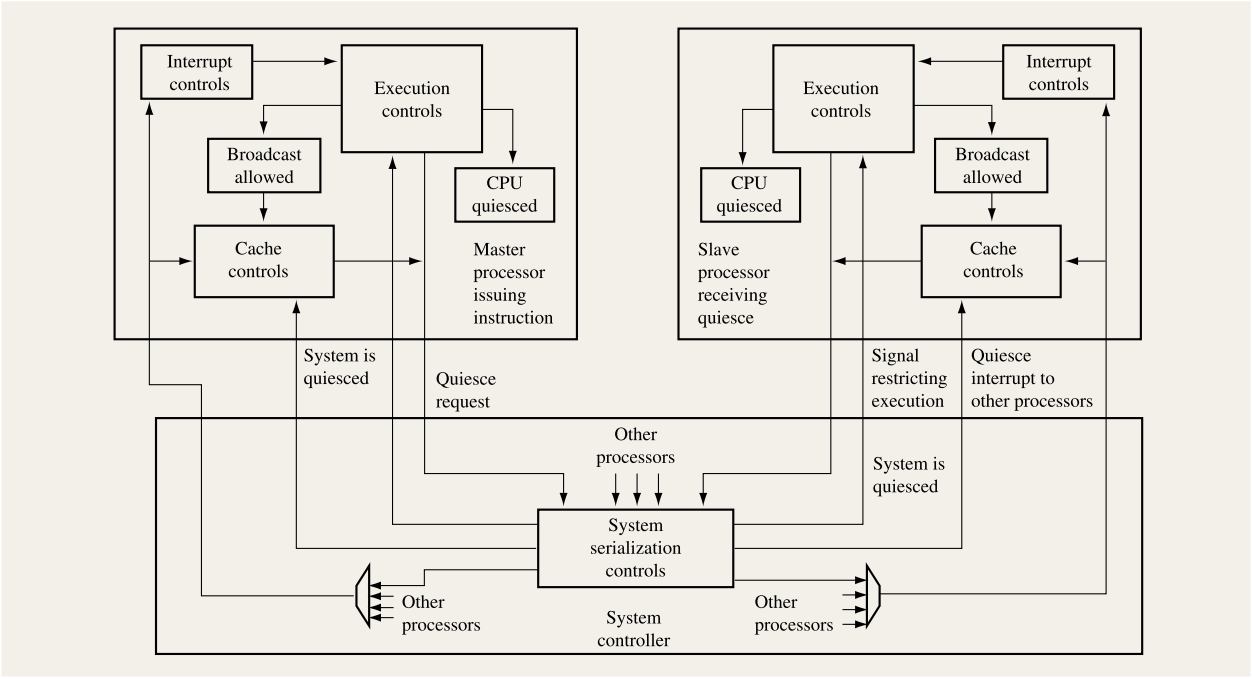

Figure 5

Quiesce operation.

This scheme meets the architectural requirements for IPTE and SSKE but has one major drawback: All processors in the system must wait, doing no useful work, until the last processor in the system responds to the quiesce request. The performance impact of this effect becomes worse for larger SMP systems. The problem can be mitigated, to some extent, by optimizing the hardware and millicode by checking for pending system quiesce requests more frequently, and this has been done on the G4 system. However, when the design of the G5 system was begun, this system quiesce penalty was a significant impediment to the improvement to CPI, and a better solution was desired.

The G5 uses a novel approach to the system quiesce problem for IPTE and SSKE which is shown in **Figure 5**. The first improvement is the reduction of handshaking between the master processor and the SC by combining the command to quiesce the system and the command to perform the actual operation on the slave processors. When the SC receives this command from the master processor, it initializes a state machine and broadcasts the request to all slave processors. A slave processor which receives this quiesce interrupt notifies the SC that it has reached a quiesce point. In contrast to the G4 system, a slave processor immediately makes the updates to its own TLB and is then free to continue normal execution of instructions, subject to certain limitations. If the slave processor requires any updates to its TLB, it must stop further execution and wait until the quiesce operation completes. The most important examples are a TLB miss requiring translation, or a required access to a storage key. The SC tracks the slave processors to determine when all of them have reached a quiesce point. When they have, it notifies the master processor so that it can change the common facility. It then informs the SC that this quiesce operation is complete and normal system operation can resume without limitations. Any slave processors that were stopped because of a TLB miss can now continue. Since the typical time between TLB misses is longer than the typical time required to perform an IPTE or SSKE operation, this design eliminates most of the "quiesce wait" penalty on the slave processors.

- *Other performance enhancements*

In addition to the major performance enhancements in the G5 processor that have already been discussed, there are many smaller improvements. Each one, by itself, provides a relatively modest increase in CPI. However, the sum of all of them yields a significant CPI improvement.

In the G4, G5, and G6 processors, integer multiply and divide operations are actually performed in the floating-point unit (FPU) rather than in the fixed-point unit

(FXU). The FPU already has the dataflow logic in place to perform these operations and requires only a small amount of additional control logic to perform the integer counterpart of these instructions. However, because the general registers (GRs) are located in the FXU, the data must be first sent to the FPU for processing and the results returned to the FXU. In G5, this handshaking between the two units has been optimized and is one clock cycle shorter than in G4.

Two types of store instructions have also been improved in G5. First, the MOVE CHARACTER (MVC) instruction with one-byte destructive overlap and the EXCLUSIVE OR CHARACTER (XC) instruction with exact overlap are commonly used to clear areas of memory (or to set all bytes to the same value). In G5, logic has been added to detect this special case and perform the stores at 16 bytes per clock cycle. In addition, floating-point store instructions may now be pipelined with other types of floating-point instructions to further improve performance.

Another area of improvement involves the hardware assists and setup that are done to make millicode operations more efficient. The G5 provides assists for millicode to use for the TRANSLATE (TR), TRANSLATE AND TEST (TRT), EDIT (ED), and EDIT AND MARK (EDMK) operations. These allow hardware to perform the time-consuming parts of these instructions while millicode is, essentially, a shell around the hardware assist iterating through the overall ESA/390 instruction. Also, for many ESA/390 millicoded instructions, to optimize performance the G5 hardware performs much more extensive setup of millicode working registers and detects various conditions that cause millicode to branch.

The G6 processor contains hardware support for up to 14 physical processors in a system. It has also optimized the overlapped fetching of subsequent operand data following the return of data misses in the L1 cache.

Architectural extensions

The S/390 architecture has undergone many improvements and additions over the years to allow new applications and workloads to be handled by the platform. A major new architectural function that was added to the G5 system is the IEEE 754 binary floating-point (BFP) architecture. This architecture has enabled the S/390 platform to provide support for new applications such as Java.** In addition, the S/390 platform is now able to provide compatibility for floating-point data from non-S/390 platforms. For further details on this architecture, please see the papers on the S/390 BFP facility by Schwarz and Krygowski [8] and Abbott et al. [9] in this *Journal* issue.

RAS improvements

The S/390 systems have long provided excellent reliability, availability, and serviceability to our customers. From the start of all designs, such provisions are part of the system design. In particular, the G4/G5/G6 systems are designed to detect and recover from errors and protect the architected state of the system. These processors have duplicated copies of the instruction unit (IU), fixed-point unit (FXU), and floating-point unit (FPU). There is a single copy of the cache and the register unit (RU). The cache uses error-correcting codes (ECC) on unique data, and parity on all other data that resides elsewhere in the system. The RU also uses ECC protection on the state of the processor. The cache and RU check all results coming from the replicated units for mismatches that indicate a possible error. When an error occurs, the processor is reset, the protected state is restored, and processing resumes.

- *Millicode update*

The cache also received an improvement in serviceability. The G4 system contained a 32KB millicode read-only array that holds the 64 most commonly used routines to improve performance. In the G5 this has been replaced by a 32KB writable storage area that is loaded at system initial millicode load (IML) time. This Licensed Internal Code may be updated to allow for functional enhancements until the final version is ready for shipment or even after it has been shipped. The G5 and G6 microprocessors, like the G4, are able to update the millicode stored in the system area concurrently with normal system operation.

- *Transparent processor sparing*

A major new recovery function called transparent processor sparing has been added in the G5 microprocessor. When a processor encounters a permanent failure, the current state of the application that was executing at that time is relocated to a spare processor in the system if one is available. More details of exactly how this works are available in the paper by Spainhower and Gregg [10] in this *Journal* issue concerning the fault-tolerant design and recovery of the system.

- *BTB array errors*

The cache arrays in the system have robust error detection and the ability to function with hardware errors. This has also been brought into the BTB array design. With the replicated copies of the instruction unit, it is possible to cross-check the values coming from the array. If an error is detected, the processor goes through a recovery action that will clear the entire array. If it is a transient error, this action alone will correct the problem. If it is a permanent array failure, this is detected if a second error has been detected within a time period determined by array failure information for the technology. With the

checking performed by the mirrored instruction units, the copies of the BTB must provide identical results. This requires the disabling of any portion of the BTB that has failed. Once detected, one and/or both sets in the array can be disabled, allowing the processor to operate correctly but in a performance-degraded mode. At the next IML, a spare processor is automatically brought on line to replace the defective one.

Conclusions

In the early 1990s a strategy was put in place to move the S/390 system line from bipolar to CMOS technology to provide a better price/performance curve for our customers. The G4 system provided performance comparable to that of the last IBM bipolar system. The G5 system represents a major performance and functional enhancement to the G4 system. The system delivers more than twice the performance of G4. The G6 system provides our customers an additional 32% processor and 51% system performance improvement over the G5 system. They complete the transformation of the mainframe systems from the bipolar technology to the CMOS technology. With support for the IEEE 754 binary floating-point architecture, new application support is enabled and continues to provide new functionality for our customers. Additionally, the G5 microprocessor introduces new RAS features which provide greater system stability to the customers.

Acknowledgments

The authors wish to recognize the other logic design leaders, Barry Krumm, John MacDougall, and Eric Schwarz. Key technical leaders were John Liptay and Charles Webb. Other key members of the design team included James Andre, Michael Campbell, Carl Dorestant, Bruce Giamei, Chris Krygowski, Wen Li, Thomas McPherson, Jennifer Navarro, Ashok Shenoy, Kevin Shum, and Scott Swaney. The logic design team would also like to acknowledge the contributions of the microcode, verification, and physical design teams.

References

1. P. R. Turgeon, P. Mak, M. A. Blake, M. F. Fee, C. B. Ford III, P. J. Meaney, R. Seigler, and W. W. Shen, "The S/390 G5/G6 Binodal Cache," *IBM J. Res. Develop.* **43,** No. 5/6, 661–670 (1999, this issue).
2. C. F. Webb and J. S. Liptay, "A High-Frequency Custom CMOS S/390 Microprocessor," *IBM J. Res. Develop.* **41,** No. 4/5, 463–473 (July/September 1997).
3. C. F. Webb, C. J. Anderson, L. Sigal, K. L. Shepard, J. S. Liptay, J. D. Warnock, B. Curran, B. W. Krumm, M. D. Mayo, P. J. Camporese, E. M. Schwarz, M. S. Farrell, P. J. Restle, R. M. Averill III, T. J. Slegel, W. V. Houtt, Y. H. Chan, B. Wile, T. N. Nguyen, P. G. Emma, D. K. Beece, T.-C. Ching, and C. Price, "A 400MHz S/390 Microprocessor," *IEEE J. Solid-State Circuits* **32,** No. 11, 1165–1175 (November 1997).
4. *Enterprise Systems Architecture/390 Principles of Operation*, Order No. SA22-7201; available through IBM branch offices.
5. T. J. Slegel, R. M. Averill III, M. A. Check, B. C. Giamei, B. W. Krumm, C. A. Krygowski, W. H. Li, J. S. Liptay, J. D. MacDougall, T. J. McPherson, J. A. Navarro, E. M. Schwarz, K. Shum, and C. F. Webb, "IBM's S/390 G5 Microprocessor Design," *IEEE Micro* **19,** No. 2, 12–23 (March/April 1999).
6. D. E. Hoffman, R. M. Averill, B. Curran, Y. H. Chan, A. Dansky, R. Hatch, T. McNamara, T. J. McPherson, G. Northrop, L. Sigal, A. Pelella, and P. M. Williams, "Deep Submicron Design Techniques for the 500MHz IBM S/390 G5 Custom Microprocessor," *Proceedings of the 1998 International Conference on Computer Design (ICCD '98)*, Austin, TX, October 1998, pp. 258–263.
7. K. Hua, A. Hunt, L. Liu, J.-K. Peir, D. Pruett, and J. Temple, "Early Resolution of Address Translation in Cache Design," *Proceedings of the 1990 International Conference on Computer Design (ICCD '90)*, Austin, TX, September 1990, pp. 408–412.
8. E. M. Schwarz and C. A. Krygowski, "The S/390 G5 Floating-Point Unit," *IBM J. Res. Develop.* **43,** No. 5/6, 707–721 (1999, this issue).
9. P. H. Abbott, D. G. Brush, C. W. Clark III, C. J. Crone, J. R. Ehrman, G. W. Ewart, C. A. Goodrich, M. Hack, J. S. Kapernick, B. J. Minchau, W. C. Shepard, R. M. Smith, Sr., R. Tallman, S. Walkowiak, A. Watanabe, and W. R. White, "Architecture and Software Support in IBM S/390 Parallel Enterprise Servers for IEEE Floating-Point Arithmetic," *IBM J. Res. Develop.* **43,** No. 5/6, 723–760 (1999, this issue).
10. L. Spainhower and T. A. Gregg, "IBM S/390 Parallel Enterprise Server G5 Fault Tolerance: A Historical Perspective," *IBM J. Res. Develop.* **43,** No. 5/6, 863–873 (1999, this issue).

Received November 18, 1998; accepted for publication May 28, 1999

Mark A. Check *IBM System/390 Division, 522 South Road, Poughkeepsie, New York 12601 (check@us.ibm.com).*
Mr. Check received a B.S.E.E. degree from the University of Wisconsin at Madison in 1988 and an M.S.C.E. from Syracuse University in 1993. He joined IBM in 1988 at the IBM Product Development Laboratory in Poughkeepsie in the Processor Development organization. Mr. Check has worked on the ES/9000 and the G4 CMOS processor families in the instruction unit; he was the instruction unit leader for the G5 and G6 processors. He has received a Fourth-Plateau IBM Invention Achievement Award, an IBM Outstanding Innovation Award, and an IBM Outstanding Technical Achievement Award. Mr. Check is an Advisory Engineer and a member of the IEEE. He is currently working on the design of future IBM microprocessors.

Timothy J. Slegel *IBM System/390 Division, 522 South Road, Poughkeepsie, New York 12601 (slegel@us.ibm.com).*
Mr. Slegel received his B.S.E.E. and M.S.E.E. degrees from Lehigh University in 1980 and 1982, respectively, joining IBM in 1982. He has worked in many areas of processor design, including arithmetic units, vector processors, and caches, and was the overall team leader for the G5 and G6 processors. Mr. Slegel has received a Fifth-Plateau IBM Invention Achievement Award, an IBM Outstanding Innovation Award, and two IBM Outstanding Technical Achievement Awards. He is a Senior Technical Staff Member, currently working on the design of future IBM microprocessors.