# Sun Makes MAJC With Mirrors

## *Dual On-Chip Mirror-Image Processor Cores Cooperate for High Performance*

*by Brian Case*

At Microprocessor Forum this month, Marc Tremblay of Sun described the first implementation of its new MAJC architecture (see MPR 8/23/99, p. 13 and MPR 9/13/99, p. 12). With two identical and independent but cooperative processor cores, the MAJC-5200 is one of the first microprocessors to implement chip multiprocessing (CMP), though Sun prefers to classify the chip as a multiprocessor system on a chip (MPSOC). This first chip will offer a relatively high clock rate, eight powerful function units, a unique geometry decompression engine, and copious amounts of off-chip data bandwidth. While this chip has two CPUs, the MAJC architecture will allow future MAJC chips to incorporate hundreds of MAJC processor cores on the same die.

Compared with the Java-specific picoJava and microJava cores, MAJC takes a different and far more sensible approach to supporting Java. In contrast to these previous Java cores, MAJC is not wedded to the Java bytecode, allowing Sun to pull a non-Java application out of its MAJC hat. In most applications, MAJC chips will execute Java bytecode through a dynamic-compiler JVM (Java virtual machine), such as Sun's HotSpot, rendering bytecode the intermediate language.

Sun plans to tape out the MAJC-5200 design by early December and begin sampling chips in the second quarter of next year. At 200 $mm^2$ in a 0.22-micron six-layer-copper process, the chip will not be an inexpensive embedded controller, but it could offer an excellent price-performance ratio. Sun says more than one committed internal customer is ready and waiting for the chip. As Table 1 shows, however, the follow-on MAJC-5200+ (Sun says this is only a working name) will have a much smaller die size. The later chip can be used as either a cost- and power-reduced version of the initial chip—keeping performance roughly the same at 500 MHz—or a cheaper and faster version of the initial chip. While still not an inexpensive embedded controller chip, the MAJC-5200+ should make the architecture appropriate for a much broader range of products.

| | MAJC-5200 | MAJC-5200+ | |
|---|---|---|---|
| CMOS Process | 0.22µ, 6-metal copper | 0.18µ, 7-metal copper | |
| Die Size | 220 $mm^2$ | 130 $mm^2$ | |
| Operating Voltage | 1.8 V | 1.5 V | |
| Frequency | 500 MHz | 500 MHz | 700 MHz |
| Power Dissipation | 15 W | 10 W | 15 W |

**Table 1.** Characteristics of the first and second MAJC chips from Sun. The second chip uses a process shrink to reduce die size and either increase performance or reduce power.

### Copy-and-Paste Reduces Design Time

Figure 1 shows a block diagram of the MAJC-5200. The two identical processor cores occupy most of the chip, and each core consists of three identical instruction pipelines ($FU_1$, $FU_2$, and $FU_3$) plus a fourth, slightly different, pipeline ($FU_0$). The $FU_0$ pipes can execute only one-quarter of the MAJC instructions, because two of the opcode bits in instruction 0 are used to indicate the packet length.

Each processor core has its own 16K two-way set-associative instruction cache. The I-caches use 32-byte lines and LRU replacement. The single 16K four-way set-associative data cache uses 32-byte lines and not-most-recently-used (NMRU) replacement. The data cache is shared by the two processor cores and is the primary communication path between them. The D-cache is a true dual-ported design, so both processors can access any part of the cache at the same time. In most other "dual-ported" caches, such as those found in most x86 chips, simultaneous access is allowed only when two accesses hit in different banks.

The $FU_0$ pipes are geographically near the shared data cache, because these pipelines are the only ones that execute memory operations in the MAJC-5200 implementation. This restriction is an implementation choice, not an architectural requirement, but it does mean that a MAJC-5200

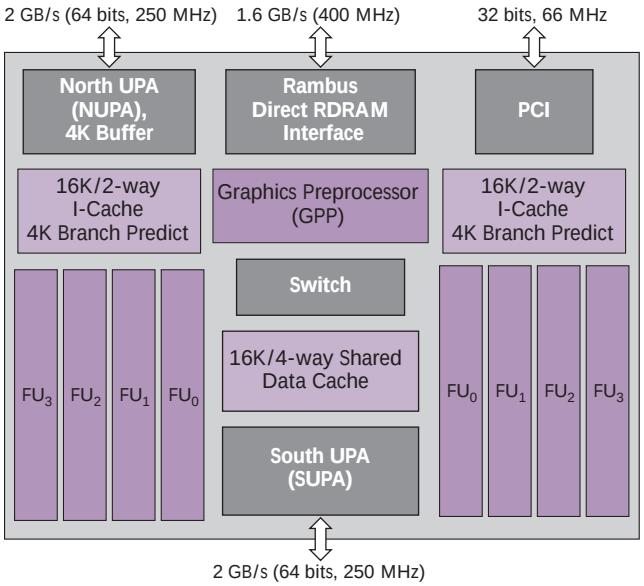


**Figure 1.** MAJC-5200 block diagram. Sun has not yet released a die plot, but this block diagram corresponds roughly to the layout of the chip. The processor cores are mirrored, with the $FU_0$ instruction pipes located near the data cache, which they share. The central Switch block connects the three I/O blocks, the GPP, and the Rambus interface to the data cache.

binary might not be optimal for a later MAJC implementation that can allow loads and stores in any pipe. It also means that code might not be backward compatible from later to earlier chips. Since most applications of MAJC chips will use a dynamic-compiler JVM that is written to take best advantage of the particular MAJC implementation on which it runs, exposed implementation constraints should not have any ill effects on performance efficiency of migrated Java bytecode; the dynamic JVM will generate optimal, implementation-specific code sequences in real time.

Pipeline $FU_0$ is also the only one that can execute instructions such as branches, divides, reciprocal square roots, cache flushes, memory barriers, and software traps.

The processor pipeline is effectively nine stages deep, as Figure 2 shows. The instruction cache and branch-prediction logic are accessed in the Fetch stage. The branch predictor is a modern but modest two-level Gshare design with 4,096 two-bit entries and global 12-bit history register. The Align stage is needed because instruction packets can begin on arbitrary four-byte boundaries.

Each of the four distinct instruction pipelines has a single decode and four execution stages; no predecode is needed, because an instruction's position within the packet determines the target pipeline. The decode stage handles the register file access. Each FU has a 128-register file with three read and five write ports. Three read ports are needed to handle the four-operand instructions, and five write ports are required to handle writes from each pipeline plus writes from the data cache.

Memory operations are started in stage E2, where the data cache and load/store buffers are accessed. Since the chip has no L2 cache—and no option to provide one—either there will be a hit in the L1 cache or the DRAM will be accessed. With the disparity between cache and DRAM latency, performance could suffer greatly when 16K provides too little cache, but Sun counters that an L2 wouldn't help MAJC's data-streaming applications anyway. To help improve performance, the data cache is nonblocking and supports out-of-order data return, and streamed data can bypass the cache entirely. As with any modern high-performance processor, the compiler must try to schedule loads in advance of data use.

To implement the nonblocking cache and out-of-order data return, each $FU_0$ pipe implements a load buffer with space for five outstanding 32-byte loads and a store buffer with space for eight outstanding 8-byte stores. When memory operations hit, the data cache can provide up to 4 GBytes/s of data to each CPU.

MAJC specifies precise exceptions, which are implemented in the 5200 chip by recording a bit vector of exceptions during the execute stages and then recognizing any event in the Trap pipeline stage. The software trap handler uses the bit vector to prioritize events and determine which of potentially many exceptions to service.

In the MAJC architecture, all instructions within a packet must complete together. In the MAJC-5200 implementation, this requirement is met, but while a packet's instructions are in the decode stage, there is some elasticity. One or more instructions can stall in their decode stage while the others proceed to E1. All FUs then wait until the decode stage(s) can proceed; all instructions move from E1 together. Nonblocking loads provide another form of elasticity, allowing a packet of instructions to complete, even though the load is many cycles from completion.

## Fast I/O for Interchip Cooperation

The chip has four I/O blocks that communicate with the on-chip caches through the switch block, shown in the middle of Figure 1. The north and south UPA blocks are general-purpose off-chip communication paths and implement a scaled-down version of the UltraSparc's 128-bit UPA interface. Both the NUPA and SUPA are 64 bits wide, support seven outstanding reads and seven writes, and provide up to 2 GBytes/s of bandwidth, but they are not identical. With a 4K data buffer, the NUPA is intended for use as an input channel, while the SUPA provides only a small buffer and is thus better suited to sending data off the chip. For compute-intensive applications that can't be tackled by a single MAJC chip, Sun envisions chains of MAJC chips connected through the UPA interfaces.

The Rambus interface connects to Direct RDRAM at 400 MHz, which results in a potential 1.6 GBytes/s of memory bandwidth. In the real world, Sun expects to reap a maximum of about 1.4 GBytes/s in data-streaming applications, such as graphics and multimedia.

The PCI interface, which is Sun's third-generation implementation, provides another way to get data to and from the chip. The 66-MHz 32-bit interface is capable of up to 220 MBytes/s.

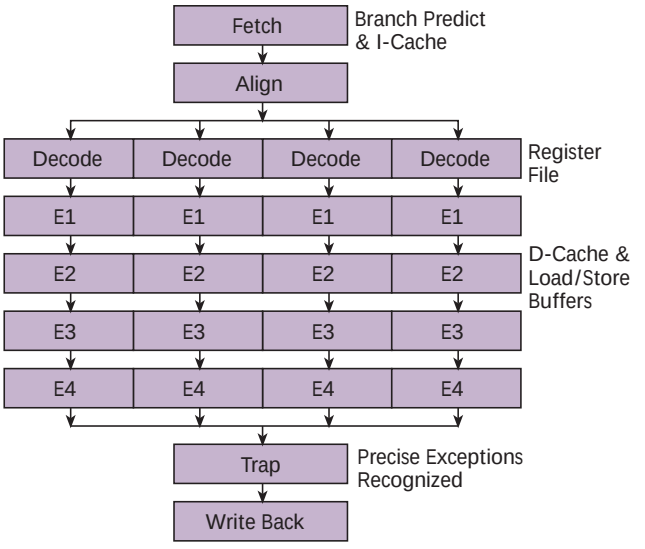


**Figure 2.** All instructions flow through the four execution stages of the MAJC-5200 pipeline, and each pipe is somewhat elastic relative to the others, which allows one pipe to get ahead of another. All instructions in a packet complete at the same time, however.

## Graphics Unit Speeds Triangle Transmission

Most of the blocks in the MAJC-5200 are general purpose in nature, but the graphics preprocessor (GPP) has a specific purpose: to dramatically speed up the receipt of triangles from an external geometry engine. The GPP implements a vertex compression scheme that is part of the Java3D specification and is readily licensable. Part of the idea is to transmit to the MAJC chip only unique vertices in a patch of triangles and then use various compression techniques on the coordinates, normals, and color characteristics of the vertices. The result is a compression ratio of between 6 and 20. The GPP can also retrieve and process uncompressed vertex data for compatibility with applications that don't adhere to Java3D.

The GPP retrieves this compressed vertex stream with its own DMA engine, which can transfer data between any two of the chip's I/O ports. The GPP then parses the stream, decompresses it, sorts the vertex data, and distributes the data to two output buffers, one for each CPU core. The GPP attempts to balance the vertex processing load to keep the two CPUs equally busy.

When real code is run and the communication channel is fast, the effect of the compression on performance is much less than a factor of six, as Table 2 shows, but the effect can still be significant, depending on other bottlenecks in the graphics pipeline. In each of the three cases shown in Table 2, the GPP compression prevents triangle communication from limiting performance. When the communication channel is bandwidth constrained, the vertex compression pays a much bigger dividend.

## Speculation Speeds Single Thread

Given the game of leapfrog in CPU performance claims, the raw speed of the MAJC-5200 is impressive. At 500 MHz the chip achieves 6.2 GFLOPS for single-precision data, 1.5 GFLOPS for double-precision data, 7 GOPS for 32-bit integer data, and 13 GOPS for 16-bit integer data. The chip can execute up to six operations per cycle from a grab bag of bit extract, byte shuffle, shift, move/pick conditional, convert, and compare instructions, which yields 3 GOPS for these operations. For the standard bcopy loop (a useful primitive Unix function), the chip achieves 1 GByte/s.

Table 2 shows some of the graphics-processing performance the MAJC-5200 can achieve. At tens of millions of fully processed triangles per second, this chip would form part of a world-beating workstation 3D graphics system today. By the time the MAJC-5200 is available, however, some 3D graphics chips may surpass this performance level, perhaps approaching 100 Mtriangles/s.

Performance for multimedia applications should also be impressive. Sun claims, for example, the chip will be capable of decoding six simultaneous 128-Kbits/s video streams while encoding a single stream at 15 frames per second in the H.263 format for video conferencing.

Perhaps the most interesting performance metric, however, is a measure of how much benefit can be reaped from the second on-chip processor for common, single-threaded applications. For multithreaded Java applications, the benefit of the second CPU is clear and easy to exploit: simply find two ready threads and start them on separate CPUs. To get a sense of what is possible, Sun simulated six benchmarks from the SpecJVM suite, both on a single processor and using the two processors. Figure 3 shows the measured speedup.

The performance improvement was gained from speculative method execution, which is implemented through a combination of hardware and software. The hardware components comprise high-speed data transfers, lock exchanges, and interrupts between CPUs, while the software component is the ability of the JVM to select a method and then ask the second CPU to run it in a protected memory and register context. In hardware, a data transfer or lock exchange between CPUs can be completed in eight cycles, while an inter-CPU interrupt takes 10 cycles. For this dual-processor configuration, there are the usual synchronization issues but no cache-coherence limitation, since both CPUs share a single data cache. The result is fast communication between CPUs that makes it feasible to

| Process | Limits | Compressed | Uncompressed |
|---|---|---|---|
| Transforms XYZ, UV | GPP limit | 92 | 71 |
| | CPUs limit | 113 | 113 |
| | Output limit | 107 | 93 |
| | Chip limit | 92 | 71 |
| Transforms+light XYZ, $N_XN_YN_Z$ | GPP limit | 110 | 77 |
| | CPUs limit | 83 | 83 |
| | Output limit | 107 | 93 |
| | Chip limit | 83 | 77 |
| Transforms+light XYZ, $N_XN_YN_Z$, RGB | GPP limit | 66 | 62 |
| | CPUs limit | 83 | 83 |
| | Output limit | 107 | 93 |
| | Chip limit | 66 | 62 |

**Table 2.** This table shows the 5200's graphics performance limits. Numbers are in millions of triangles per second. (Source: Sun)

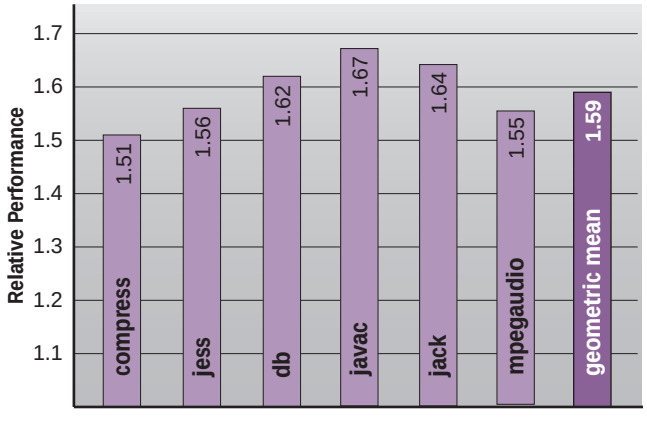


**Figure 3.** The benefit of speculative method execution for single-threaded code on two CPUs can be significant. Six unmodified single-threaded programs from the SpecJVM suite show an aggregate speedup of 1.59. (Source: Sun)

ask the second CPU to take a "shot in the dark" on an upcoming method. If there are dependencies between the method running on the first CPU and the speculative method on the second CPU, they can be detected quickly and either resolved or used to cancel the speculative method.

Aside from fast data and control communication between the two processor cores, speculative method execution is implemented mostly in software. The JVM scans ahead in the Java bytecode stream, locates what it determines is likely to be an independent method, then asks the second CPU to run the method in a separate address space. If the speculation succeeds, the two memory spaces are merged to create a single space. If the speculation fails, the speculative method is canceled and its space is garbage-collected. There is no overhead for canceling the speculative method, but some overhead is suffered to start the independent thread.

What is especially impressive here is that a "simple" CPU replication and some hardware and software cleverness results in a speedup that would be difficult to achieve by extending today's out-of-order superscalar microprocessor organizations. These results were achieved using Java on an architecture and implementation somewhat tailored for Java, but the larger reality is that the results may bode well for chip multiprocessing implementations of other architectures. Even if it is true that MAJC has better support for collaborative use of dual processors, other architectures can adopt the technique too, just as many old ISAs have been augmented with some form of predicated execution and SIMD operations. Sun's MAJC results seem to say that chip multiprocessing can be generally beneficial, given the right software support.

MICHAEL MUSTACCHI

**MAJC chief architect Marc Tremblay describes the 5200's dual-processor structure at Microprocessor Forum.**

Of course, the two CPUs on the MAJC-5200, and all the CPUs on future MAJC chips, need not be used only for speculative method execution. For applications that are already multithreaded, it makes sense to simply distribute the threads to independent CPUs. And in an environment where there are independent system tasks or other parallel processes, it makes sense to simply distribute the processes to independent CPUs.

## Good Start, Uphill Battle

As impressive as the MAJC-5200 chip is, it is also the victim of compromise. While MAJC is nominally a 64-bit architecture, the first chip isn't. First, the register files, data paths, and memory pointers are all 32 bits. This means that double-precision FP and long-integer operands will occupy two adjacent register file locations. This won't be true, however, for 64-bit MAJC implementations, so there will be an issue of binary compatibility for statically compiled code. As mentioned before, this shouldn't be much of an issue for Java bytecode binaries since the final code executed by the MAJC chip will be generated on the fly by a dynamic compiler JVM.

Second, the caches are relatively small at 16K each, and even though the Rambus memory system will provide high bandwidth, it has long latency relative to the cache. Many multimedia applications will be able to stream data in from memory or I/O interfaces and keep the streams from polluting the cache, however, 16K is still quite small, especially given that the data cache is shared by two processors that are ostensibly executing independent threads. Additionally, the caches may be too small to run the dynamic compiler at top speed, although the code it generates for inner loops will perform well and may dominate program run time.

Thus, while it is impressive that Sun designed two processors on a single chip, it is nonetheless likely the engineers could have built a higher-performance single-CPU chip. With one CPU, the caches could have been bigger, the limitation of only $FU_0$ executing memory ops could have been relaxed, and the chip could have implemented the full 64-bit MAJC architecture. The chip might even have been smaller. Sun counters that Java isn't helped by issuing more than four instructions, that two four-issue CPUs deliver more performance than a single eight-issue CPU; that cycle time is better for two simple CPUs; and that design time is greatly reduced versus a complex out-of-order implementation.

The innovation in MAJC is the Java-centric architecture and implementation combined with attention to chip multiprocessing. Java is perhaps the best current software vehicle for threaded programming, and threads may be the best way to surpass the limits of performance improvement through ILP. As Java use rises, a Java-centric microprocessor might find significant volume in purpose-built devices.

As with any new architecture, however, MAJC faces a steep uphill climb to achieving a state of profitability sufficient to sustain its development. To its credit, Sun has hitched MAJC to a couple of sturdy new performance bandwagons—chip multiprocessing and speculative method execution—and the MAJC-5200 has proved, at least in simulations, these techniques are one way to keep improving the performance of single-threaded applications. M

### Price & Availability

Sun Microelectronics expects a December tapeout for the 5200 with internal samples in 2Q00. The 5200+ should sample in 3Q00. For more information, go to *www.sun.com/microelectronics/MAJC/*.