MICROPROCESSOR REPORT

THE INSIDERS' GUIDE TO MICROPROCESSOR HARDWARE

VOLUME 9 NUMBER 3 MARCH 6, 1995

Hal Reveals Multichip SPARC Processor

High-Performance CPU for Hal Systems Only—No Merchant Sales

by Linley Gwennap

Emerging bleary-eyed and relieved after a long, bumpy flight, Hal Computer Systems revealed the design of its first microprocessor, which the company calls the Sparc64, in a number of ISSCC and Compcon presentations. The multichip processor falls short of repeated claims by former CEO Andrew Heller to deliver the industry's best performance. Nevertheless, the company estimates that its decoupled superscalar design will produce 256 SPECint92 and 330 SPECfp92, respectable performance for a high-end processor.

Hal, now a subsidiary of Fujitsu, plans to use the Sparc64 in its own line of SPARC systems; neither the startup nor its parent expects to sell the new processor on the merchant market. We expect the first Hal systems, based on the Sparc64, to ship in 3Q95. These products, code-named R1, will be uniprocessor workstations for technical and commercial applications.

The Sparc64 design, also known as the PM1, carries performance and schedule expectations similar to those of Sun's UltraSparc *(see **081301.PDF**)*. Both implement the 64-bit SPARC V9 architecture. The Sparc64 goes beyond UltraSparc (and most other microprocessors) by implementing high-reliability features suitable for mission-critical enterprise servers. Hal's future R2 products, due in 1996, will target this market.

The decoupled superscalar design is similar to the HP PA-8000 or Intel P6. Unlike these chips, the Hal processor tracks instructions using tags, allowing 64 to be active at once, more than any other processor. An unusual memory subsystem design should allow high performance even on applications with large working sets. The target clock speed in an MCM package is 154 MHz, faster than any other multichip microprocessor.

Multichip Design Allows Large Caches

The Sparc64 processor consists of two logic chips and four memory chips. As Figure 1 (see below) shows, the CPU chip contains the superscalar processor along with a 4K instruction cache and control logic for the off-chip instruction and data caches. The off-chip caches are each 128K in size using two 64K cache-memory chips. The MMU chip translates memory addresses and connects the processor to separate memory and I/O buses.

The off-chip caches are fully pipelined with a three-cycle latency, similar to pipelined synchronous SRAMs. They are virtually indexed and virtually tagged, using 128-byte lines. Each line is divided between two chips, creating two banks per cache. The banks can be addressed individually, allowing each cache to service two requests per cycle, as long as they use different banks. The caches are also nonblocking, continuing to service requests while up to two miss transactions are pending. Either cache can continue to service accesses from one bank as the other is handling a refill.

Although the caches are identical, they are used somewhat differently on the instruction and data sides. The 128K instruction cache acts as a level-two cache behind the 4K on-chip cache. The smaller cache (which Hal calls a level-zero cache) has a single-cycle latency, reducing branch penalties. Because the CPU fetches from a single instruction stream, both I-cache chips receive the same address on each cycle.

There is no on-chip data cache, so the off-chip cache is the primary data storage. The CPU drives separate addresses to the two D-cache chips, simulating a dual-ported cache using a two-bank structure.

Figure 2 shows a detailed diagram of the CPU chip, which contains a prefetch unit that controls accesses to the L2 instruction cache. It attempts to stay two 64-byte cache lines ahead of the fetch unit. The CPU contains a prefetch buffer that holds up to two cache lines as they are returned from the L2 cache. Data read from the L2 cache is stored in the on-chip cache once it is validated.

As instructions are read from the L2 cache, they are predecoded (which Hal refers to as recoding) before being stored in the L1 cache. The predecode unit can handle four SPARC instructions per cycle, keeping up with the bandwidth of the L2 cache. Because the third

stage of the cache access consists mainly of moving data from the cache chip to the CPU, there is enough time to predecode the instructions in this third stage, so no delay is incurred.

Predecoded instructions contain four extra bits, which simplify the later decoding process. The predecode unit also performs a target calculation for branch and call instructions, adding the offset to the current PC. The lower bits of the target are stored back into the offset field of the original instruction; the carry bit is kept as one of the four extra bits, so the full target can be recalculated later. This calculation speeds branch handling.

The fetch unit takes up to four instructions per cycle from the prefetch buffer. If the needed data is not in the prefetch buffer (due to a misprediction), the fetch unit can instead read the data from the L1 cache with no delay. The L1 cache is nonblocking and capable of providing data at the same time that it is accepting data from the prefetch buffer. If an access misses the L1 cache as well, the needed instructions must be fetched from the L2 cache, requiring a delay of three cycles. The deep queues in the execution engine typically mask this delay.

As long as instructions are available, the fetch unit will read four arbitrarily aligned instructions and pass them to the issue unit. The only exception occurs if the end of a cache line is reached; because the fetch unit cannot access two cache lines at once, it will finish one line, then begin the next line on the subsequent cycle.

Register Windows Complicate Renaming

The issue unit finishes decoding the instructions, reads the source operands, renames the destination registers, and issues the instructions to the appropriate reservation station. The predecode bits help to fit this entire sequence of events into a single clock cycle.

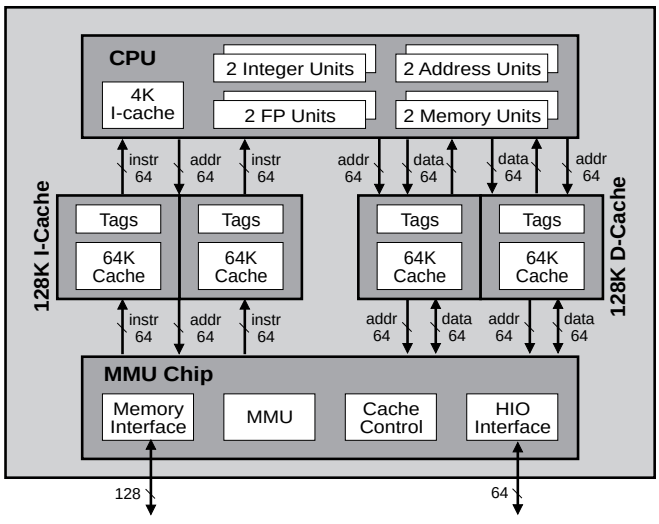

Figure 1. The Sparc64 processor consists of the CPU chip, the MMU (system-interface) chip, and four cache chips.

The issue unit can handle four instructions per cycle as long as there are no more than two floating-point instructions, two load/store instructions, and one branch. Four instructions can be issued even if all are integer calculations, as long as at least two are simple ALU operations (no shifts, multiplies, or divides).

To meet these issue requirements, the integer register file has ten read ports and four write ports, while the floating-point register file has six read ports and three write ports. Combining this many ports with the large number of registers required by SPARC register windows can consume large amounts of die area. UltraSparc avoids this problem with a patented method of compressing the register file *(see **081301.PDF**)*, but Hal could not use this technique. Instead, the Sparc64 implements only four register windows, compared with the seven or eight in typical SPARC processors. This choice keeps the size of the register file within reason but may cause more register spills to memory.

Four register windows require 78 integer registers along with the standard 32 FP and 5 condition-code registers in SPARC V9. To support renaming, the Sparc64 implements 38 additional integer registers along with 27 extra condition-code and 24 spare FP registers. Renaming prevents stalls when one instruction attempts to overwrite the contents of a register before a previous instruction can access that register, a situation that occurs frequently in out-of-order processors like the Sparc64.

To keep track of the proper program order of instructions, each instruction is assigned a unique 6-bit tag by the issue unit. This avoids the need to store instructions in a reorder buffer, as in AMD's K5 and others. The width of the tags allows up to 64 instructions to be active in the machine at any given time, giving the Sparc64 more flexibility in reordering instructions than any other announced microprocessor.

Figure 3 shows conceptually how the tags are used. In this chart, the oldest active instructions (2–4) are ready to be retired, as all previous instructions are already completed. Instruction 5 is still executing, preventing instructions 6–9 from being retired, even though they have already completed. Subsequent instructions are in a variety of states, with the most recent instructions (18–20) being issued. In this example, only 19 instructions are active.

Three pointers manage this process. The issue pointer contains the last tag value that has been issued. The commit pointer indicates the oldest instruction that is still executing; all previous instructions are eligible for retirement. The retirement pointer points to the oldest active instruction. New instructions can be issued as long as the issue pointer does not catch up to the retirement pointer.

This method is simpler than the reorder buffer used in other decoupled designs. A reorder buffer stores all ac-

tive instructions and their results, acting as a large associative register file. In the Sparc64, speculative results are held in rename registers, eliminating the associative access. Furthermore, the instructions that generated these results are not stored after executing; the tags are all that is needed to ensure proper ordering.

Using tags allows the window to be expanded to 128 instructions or more simply by increasing the width of the tag field. Hal chose a 6-bit tag because simulations showed that the 64-instruction window is rarely filled; stalls are more likely to come from a shortage of rename registers or a lack of adequate parallelism in the instruction stream.

A Plethora of Function Units

Renamed instructions are issued to one of four reservation stations (RS) or to the branch monitor, as Figure 2 shows. Load and store instructions are issued to both the address RS and the memory RS. The reservation stations for the integer, floating-point, and address-generation units can each hold 8 instructions; the load/store RS has 12 entries. Each reservation station can accept two instructions per cycle.

Instructions wait in the reservation stations until their operands are available before being dispatched to the function units. If more instructions than function units are available, the oldest instructions that are ready for execution are dispatched. Instructions are issued directly to the function units if the reservation stations are empty.

The integer RS can dispatch two instructions per cycle. One goes to a full integer unit, while the other goes to a restricted unit that has an ALU and shifter but no multiplier or divider. Two address-generation instructions can be dispatched at the same time. The address-generation units can also handle integer ALU instructions, so up to four integer arithmetic instructions can execute in a single cycle.

The FP reservation station can dispatch one instruction per cycle to either the general FPU or the FP divider. The FPU has a four-stage multiply-add pipeline. The self-timed divider generates a double-precision value in 8–9 cycles. Table 1 shows the cycle times for long-latency integer and FP operations. Note that the Sparc64 traps several V9 instructions and emulates them in software; these include all quad-precision FP instructions, FP square root, and population count (POPC).

The dispatch rules for loads and stores are more complex than for other types of instructions. Program errors could occur if a load and a store to the same memory address execute out of order. The Sparc64 supports two memory modes, based on SPARC V9 definitions. In TSO (total store ordering) mode, all loads and stores are executed in strict program order. In RMO (relaxed memory ordering) mode, loads and stores can execute out of order; the processor does a partial address check to ensure that dependent loads and stores are not reordered.

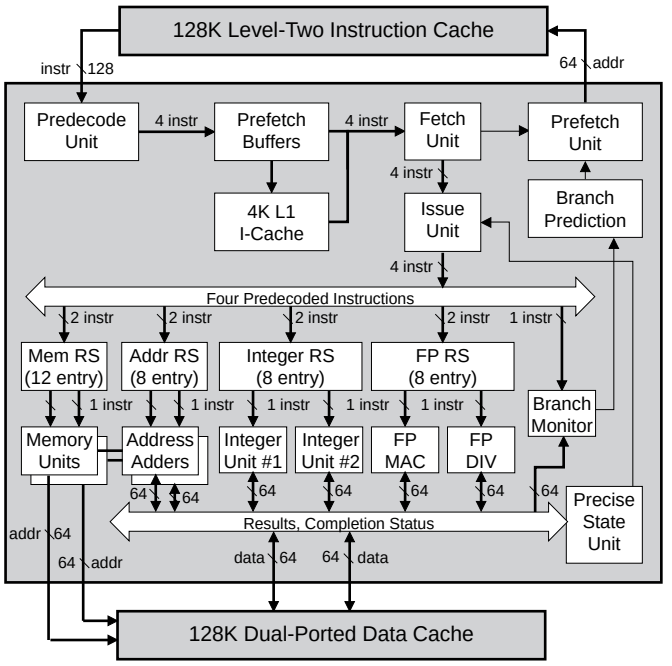

Figure 2. The Sparc64 CPU can issue up to four instructions per cycle, which are then executed by the decoupled function units.

Loads and stores must first wait for their addresses to be computed by the address units. The load/store RS can then dispatch two instructions per cycle. Because the data cache is divided into two banks, one memory unit handles accesses to the even bank while the other controls the odd bank. Loads take one cycle to compute the address and two additional cycles to retrieve the needed data from the cache. Stores also take three cycles to complete. Like the instruction cache, the data cache is fully pipelined, sustaining two accesses per cycle.

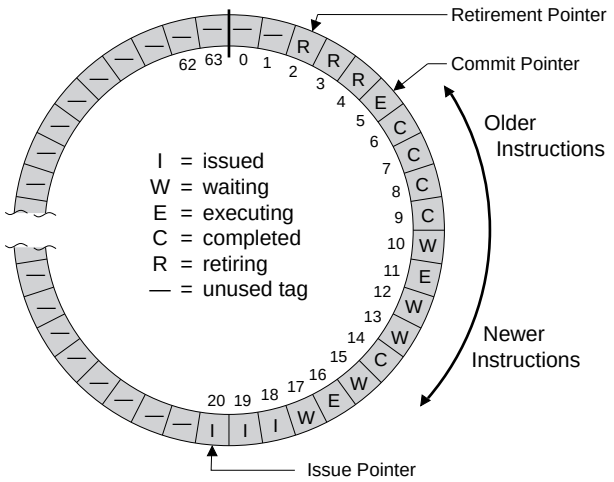

Figure 3. The Sparc64 uses 6-bit tags to ensure that instructions are retired in correct program order even when executed out of order.

Hal Outlasts Founder Heller

Hal was founded in the summer of 1990 by Andrew Heller, a key designer of IBM's RS/6000 architecture. His vision was bold: to build a world-class computer company from scratch. Hal planned to build its own microprocessors (using its own design tools) to power its own system hardware and operating system for use in a broad range of workstation and server products. By early 1991, Fujitsu agreed to fund the venture, although Heller and other employees held a minority stake, hoping to profit when the company went public.

As Heller and his early cohorts began work on what would become the Sparc64, the instruction set was undefined. Fujitsu's backing pushed Hal into the SPARC camp. But Hal felt that commercial markets would require a 64-bit processor, so the company drove the development of the V9 standard *(see **070201.PDF**)* that extended SPARC to 64 bits and added other performance enhancements.

Heller expected to deliver the first systems by the end of 1993 and staffed the company accordingly, building a sales and marketing group along with huge hardware and software development teams. But it soon became apparent that Hal had bitten off more than it could chew, as the mammoth development efforts began to stall from the number of new people and the sheer amount of work to be done. With the company running low on funds and no product revenue in sight, Hal cut back its marketing and sales force.

As the design effort continued to slip, the mercurial Heller clashed with both his staff and his board of directors, and he was forced to leave Hal in August 1993. Ultimately, Fujitsu supplied additional funding, buying out the other shareholders (primarily employees) in November 1993.

To simplify its software effort, the company decided to use Sun's Solaris rather than developing its own OS from scratch. Hal still retains a software team that is developing 64-bit extensions for Solaris.

Fujitsu also owns SPARC processor vendor Ross Technology and manufactures both Hal's Sparc64 and Ross's HyperSparc chips. The Hal processor provides Fujitsu with better performance than the midrange HyperSparc without moving to the UltraSparc design, which is built by SPARC rival Texas Instruments. Although the company does not plan to sell the Sparc64 openly, the processor could be used in SPARC systems sold by Fujitsu or by two companies in which Fujitsu holds a controlling interest: Amdahl and the British minicomputer vendor ICL.

Now under the management of Scott Metcalf, Hal's 400 employees are finally nearing their goal of delivering working systems. The Sparc64 achieved first silicon in June 1994 and is currently running Solaris in test systems. With financial and manufacturing support from parent Fujitsu, the company appears likely to ship systems by the fall of 1995.

Branches Cause Checkpoints

Branches are always a source of excitement in decoupled superscalar processors, as a single misprediction can cause dozens of calculations to be thrown away. The Sparc64 has a 1,024-entry branch history table (BHT) that uses saturating two-bit counters to predict branches using the common Smith algorithm. This BHT is larger than those in most competitive processors, improving the branch-prediction accuracy.

As instructions are fetched, their addresses are checked in the BHT. If the BHT predicts a taken branch, the instruction contains enough bits of the target address (as computed by the predecode unit) to initiate a fetch from the instruction cache. If the target address hits in the level-one instruction cache, there is no penalty for a correctly predicted taken branch. The presence of (possibly annulled) delay slots complicates the handling of branches in a superscalar SPARC processor.

To improve the handling of subroutine returns, which have indirect targets that cannot be precomputed, the Sparc64 implements a four-entry return-address table that is indexed by the current window pointer used by SPARC processors. During normal subroutine processing, the return address is stored in the table when the subroutine is called and is available to immediately redirect the fetch stream when the subroutine returns.

Every time a branch is encountered, the processor stores important machine state in shadow registers. Checkpoints are also made before issuing instructions that alter nonrenamed software-visible state. The number of registers that must be checkpointed is relatively small, as the general-purpose registers are all renamed; only the register map, not the registers themselves, must be backed up. The Sparc64 supports 16 levels of checkpointing.

The checkpointing mechanism allows the machine state to be restored in a single cycle. The register maps are restored to their previous state, effectively discarding the values computed speculatively. Other machine state is restored from the shadow registers.

The branch monitor holds all active branches and monitors the output of the function units to verify that the branch condition is computed to the predicted value. If a deviation is detected, the branch unit signals the precise state unit, which handles the process of restoring the system to its proper state.

In the event of an exception (e.g., a failing instruction), it may take several cycles to restore the machine state. The processor begins by restoring the state to the first checkpoint after the failing instruction; this takes a single cycle, as for mispredicted branches. The CPU can then "undo" four instructions every two cycles, backstepping through the register map until it reaches the failing instruction. The exception is then reported to software

with the processor in the correct state, as if it had executed all instructions up to the failing one.

Figure 4 shows the Sparc64 pipeline. The basic integer pipeline is only four stages. Thus, if a branch flows directly through the pipeline, the misprediction penalty can be as little as three cycles (including one to recover), although it can be longer if the source of the condition sits in the reservation station for a while. Three stages at the end of the pipeline handle instruction retirement, but these do not impact performance.

The figure also shows the three-stage prefetch pipeline, in which instructions are fetched from the L2 cache. This pipeline is transparent to the CPU as long as the prefetch is correct, but these extra stages reduce performance if a taken branch misses the L1 cache. The load pipeline also adds three stages, resulting in a three-cycle load-use penalty for loads that hit the L1 cache.

System Interface Allows Local Memory

The poorly named MMU chip connects the CPU and cache to the rest of the system. In addition to the MMU, it controls the 128-bit memory interface and a separate 64-bit I/O bus. This partitioning allows the CPU and cache chips to work entirely in a virtual address space while the MMU chip keeps these addresses consistent with the rest of the system. Although other 64-bit processors force some of the upper address bits to be zeroes, the Hal design implements full 64-bit addresses throughout the processor.

Unlike most processors, the Sparc64 supports three levels of addressing. Virtual addresses are first mapped to logical addresses (or "views"), which are used for the I/O subsystem as well as for remote memory in a multiprocessor configuration. These logical addresses are then converted to real addresses for the local memory system. The intermediate logical addresses provide a convenient method of mapping and sharing I/O devices and memory.

Virtual-to-logical translation is speeded by the view lookaside buffer (VLB), a fully associative translation cache with 128 entries. Logical-to-real translation is handled by a standard TLB with 1,024 entries in a four-way associative structure. Separating the MMU from the CPU allows such a large TLB; this in turn should improve performance on commercial applications and other software with large data sets.

Because the CPU can generate three addresses per cycle (one from the instruction cache and one from each bank of the data cache), the MMU includes a three-port micro-TLB with eight entries. The µTLB translates directly from virtual to real addresses in parallel with the cache access. Addresses that miss the µTLB are handled by the VLB/TLB combination. VLB and TLB misses are handled in hardware through a table-walk algorithm.

	Single Precision		Double Precision	
	Throughput	Latency	Throughput	Latency
Integer Multiply	3 cycles	3 cycles	5 cycles	5 cycles
Integer Divide	~9 cycles	~9 cycles	~17 cycles	~17 cycles
FP Add	1 cycle	4 cycle	1 cycle	4 cycle
FP Multiply	1 cycle	4 cycle	1 cycle	4 cycle
FP Multiply-Add	1 cycle	4 cycle	1 cycle	4 cycle
FP Divide	~4 cycles	~5 cycles	~7 cycles	~8 cycles

Table 1. Floating-point add and multiply operations share a single four-stage pipeline. Integer multiply and divide operations are also handled by a single unit and cannot execute in parallel.

The Sparc64 module connects to main memory through a 128-bit interface controlled by the MMU chip. That chip receives and queues miss requests from the caches and controls access to main memory. An external memory controller is required to generate the necessary control signals for the DRAM. The memory system returns the critical word first when servicing a cache miss. The penalty for an L1 cache miss is about 21 cycles, depending on the type and speed of the memory system.

The 64-bit HIO (Hal I/O) bus connects the processor to I/O devices. The multiplexed HIO bus operates at 50 MHz using 3.3-V signal levels. The CPU and HIO clocks are essentially asynchronous. The bus executes one transaction at a time instead of using a more efficient split-transaction protocol. Hal is developing bridges to SBus and SCSI as well as an HIO-based graphics card.

Designed for High Reliability

Taking a lesson from the eponymous but fictional HAL 9000, the company has included many features for error detection and correction in the Sparc64 design. The L1 instruction cache is protected by parity; corrupted data is reread from the L2 cache. The data cache arrays are protected by full ECC, using 8 bits for each 64 data bits; this allows single-bit error correction and double-bit error detection. Similarly, the main-memory subsystem is protected by ECC. The VLB and TLB are protected by parity, as translations can always be retrieved from the

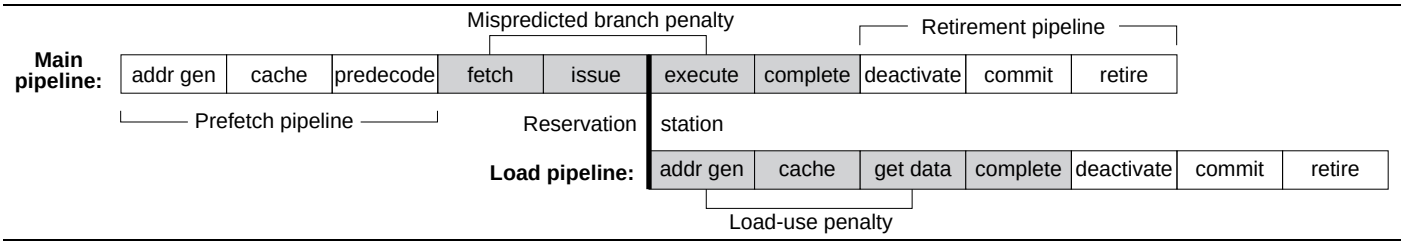

Figure 4. The Sparc64 uses a basic integer pipeline (in gray) of four stages with extra stages for a prefetch pipeline and a retirement pipeline.

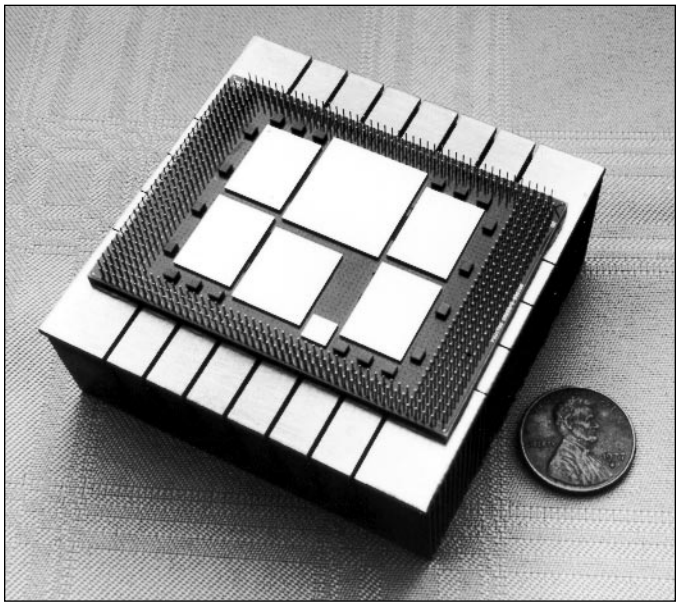

Figure 5. The Sparc64 module contains a large CPU chip and a smaller MMU chip surrounded by four identical cache chips and a tiny analog clock chip. The 565-pin ceramic PGA package measures 6.1 × 4.5 cm, not including the large heat sink.

page tables in memory. All signals that cross from chip to chip within the module are parity protected.

Correctable errors are logged and signaled to software via a low-priority interrupt. Errors that cannot be corrected by hardware cause a high-priority interrupt, forcing the CPU into RED state as defined by the V9 architecture. In this state, the Sparc64 fetches instructions from a diagnostic processor across a serial bus controlled by the MMU; these instructions are executed serially. This procedure, although slow, allows diagnosis of the error even if cache, main memory, or some portions of the CPU itself have failed. In many cases, diagnostic software can disable the failed hardware unit, abort affected software processes, and restart the system.

These protection features, and the diagnostic processor, are similar to techniques used in mainframes but are not found in most microprocessor-based systems. Only IBM's POWER processors (which Hal founder Heller helped develop) offer a similar level of data checking. Hal expects that these features will give its systems an edge when competing for mainframe-replacement business in accounts that demand high availability and data integrity.

	Logic Transistors	Total Transistors	Die Area
CPU Chip	2.2 million	2.7 million	297 mm^2
MMU Chip	0.6 million	2.0 million	163 mm^2
Cache Chip	0.3 million	4.3 million	142 mm^2
Module (total)	4.0 million	21.9 million	1,028 mm^2

Table 2. The Sparc64 module consumes nearly 22 million transistors, although most are in the four cache chips.

Multichip Packaging Reduces Cost

The six chips are packaged in a single ceramic multichip module, as Figure 5 shows. Based on Fujitsu's mainframe technology, the module uses a ceramic substrate to provide interchip connections. The Sparc64 die are mounted face down ("flip-chip") on the substrate, eliminating the need for bond wires.

The large number of signals used by the CPU and MMU chips makes it practically impossible to package the chip set discretely; as Figure 1 shows, each chip has more than 600 I/O signals and would require packages with nearly 1,000 pins to provide adequate power and ground pins. But most of these signals run from one chip to another within the module; only 286 connect externally, most to the memory and I/O subsystems. Including power and ground, the module requires 565 pins, a large but feasible number.

In addition to reducing cost compared with discrete packages, the module also improves performance. Interchip signals go from die to substrate to die, avoiding the electrical discontinuities of two sets of bond wires and pins. In the Sparc64, no signal makes more than one chip-to-chip crossing per cycle, and those that go between chips typically have most of a cycle to make that trip. As a result, the Hal processor is able to achieve a higher clock rate than any other multichip processor, reaching a cycle time of 6.5 ns.

The Sparc64 die are manufactured in Fujitsu's 0.4-micron CMOS-55 process with four metal layers. This process is more advanced than Fujitsu's CMOS-50, which HyperSparc-2 uses, and a step ahead of what other vendors have yet announced.

As Table 2 details, the MCM contains a total of 21.9 million transistors, although most of these are in the cache chips. The CPU chip has 2.7 million transistors, but nearly all are for logic, which packs less densely than memory. Thus, the aggressive 0.4-micron process was needed to pack the Sparc64 CPU into 297 mm^2 of silicon, as Figure 6 shows.

The MPR Cost Model estimates that the entire Sparc64 module costs roughly $770 to build, far more than the $180 manufacturing cost of the similarly configured but lower performance HyperSparc module. UltraSparc's estimated cost is $420, but this cost does not include any external cache. Adding 256K of 166-MHz synchronous SRAM, at list price, to the cost of an UltraSparc chip brings the total cost to roughly that of the Sparc64 module.

The Sparc64 uses a supply voltage of 3.3 V. Hal is still characterizing the power of its processor but estimates that the module will dissipate about 60 W at 154 MHz. The package has a relatively large surface area but still requires a hefty heat sink, shown in Figure 5, for adequate cooling.

Price & Availability

Neither Hal nor Fujitsu plans to sell the Sparc64 processor on the merchant market. Hal will use the Sparc64 in systems expected to begin shipping in 3Q95. For more information, contact Clark Hoyle of Hal Computer at 408.379.7000 x1391; fax 408.379.2786.

Good Performance for Commercial Code

Hal's performance estimates are projections from initial measurements; the current prototypes do not yet deliver the full 256 SPECint92. Assuming that both the Sparc64 and UltraSparc meet their performance and schedule goals, the Sun processor will be just slightly faster on integer code; both should appear in systems at about the same time. Sun's chip has the advantage of its unique multimedia extensions *(see **081604.PDF**)*.

Hal's processor should do well on programs with larger working sets than the SPEC92 benchmarks, including the forthcoming SPEC95 suite and typical commercial applications. The large pipelined caches and huge TLB will assist these types of programs. The processor's projected floating-point performance lags that of next-generation Alpha, MIPS, and PA-RISC chips but slightly exceeds that of UltraSparc.

Had Hal been able to get the Sparc64 to market within a year of its original schedule, it would have been the first decoupled superscalar processor in production and would have led the field in performance, even in the original CMOS-50 process. At this point, however, it is just one of many such designs, and its performance will be good but not outstanding.

Although the manufacturing cost of the Sparc64 module is comparable to that of UltraSparc with external cache, the custom interface chips used by the Hal processor will be at a lower volume than the interface chips used by UltraSparc, putting Hal at a cost disadvantage. Other next-generation RISC chips should deliver better performance than the Sparc64 at a lower cost and with higher volumes.

Hal's biggest differentiators are the robust data-integrity features of its processor. Although HP and Sun are actively pursuing "mainframe downsizing" customers, neither offers similar features, which may make these customers more comfortable with a Hal offering. These customers must wait for Hal's second round of products, however; the market for Hal's initial workstations is less interested in this level of data integrity.

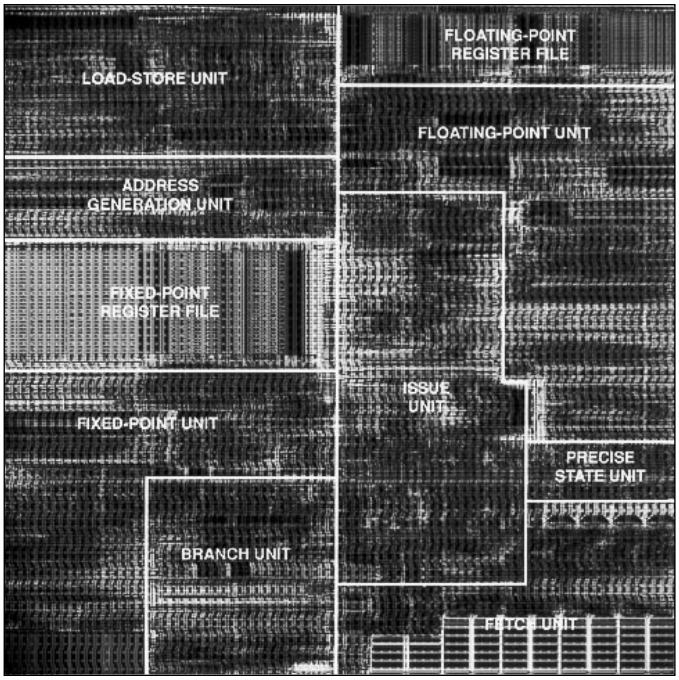

Figure 6. The Sparc64 CPU measures 17.5 × 16.9 mm and consumes 2.7 million transistors in 0.4-micron four-layer-metal CMOS. There is no pad ring due to the flip-chip mounting process. The fetch unit includes a 4K instruction cache; the data cache is off chip.

Hal has surprised many skeptics by getting this close to shipping a product. Yet even if the company delivers on its promises, it faces an uphill struggle to establish itself in a marketplace full of aggressive competitors. When the company was started, Sun was not selling much to the commercial market, but now that company will be Hal's biggest competitor. Fujitsu's deep pockets should keep Hal going long enough to establish a foothold in that market, which may be all the small company needs to survive. ♦

