POWER2: Next generation of the RISC System/6000 family

by S. W. White
S. Dhawan

Since its announcement, the IBM RISC System/6000® processor has characterized the aggressive instruction-level parallelism approach to achieving performance. Recent enhancements to the architecture and implementation provide greater superscalar capability. This paper describes the architectural extensions which improve storage reference bandwidth, allow hardware square-root computation, and speed floating-point-to-integer conversion. The implementation, which exploits these extensions and doubles the number of functional units, is also described. A comparison of performance results on a variety of industry standard benchmarks demonstrates that superscalar capabilities are an attractive alternative to aggressive clock rates.

Introduction

In 1990, IBM announced the RISC System/6000® (RS/6000) family of highly concurrent superscalar workstations and servers, supporting clock rates ranging from 20 MHz to 30 MHz [1]. The 25-MHz Model 530 achieved performance levels which exceeded those of many of its contemporaries (Sun™ 4/200, DECstation 3100, MIPS® M/2000, and Apollo DN10000) by more than 40% on a variety of benchmarks [Dhrystones 1.1, Whetstones, Linpack (dp) Livermore Loops (geometric mean), and SPECmark™] [2]. All models included an 8KB instruction cache (I-cache) and either a 32KB or a 64KB data cache (D-cache). These POWER processors were the first implementations of the IBM POWER (Performance Optimized With Enhanced RISC) Architecture™.

Over the years, the POWER-based RS/6000 offerings have improved incrementally. Desktop, deskside, and rack system clock rates increased up to 62.5 MHz. More than ten of these models support a 32KB I-cache. Additional compiler capability, especially in the area of restructuring data access patterns, has improved benchmarks and customers' code. Changes in the I/O area have increased Micro Channel® bandwidth from 40 MB/s to 80 MB/s peak.

While these changes were taking place, the competition also improved. A dichotomy in design philosophies became apparent. RS/6000 systems and compilers aggressively exploit superscalar capabilities. Other designs, such as Sun SuperSPARC™ and Motorola 88110, also exhibit this philosophy. These superscalar capabilities involve multiple functional units and the hardware complexity to allow the units to function relatively autonomously. Some argue that the complexity makes high clock rates difficult to achieve, and that more performance can be achieved by clock rate

than by aggressive instruction-level parallelism. Examples of this alternative philosophy are the DEC™ 21064 (also known as Alpha™), the HP PA7100, and the MIPS R4000. The debate about the advantages of each approach appears on electronic forums and in articles and editorials. The popularity of the topic has led to the coining of catchy synonyms for the two approaches, such as the "Speed Demons" (high clock rate) versus the "Brainiacs" (complexity) [3].

While it is desirable to pursue both approaches, the goals are often in conflict [4]. For a given technology, there are likely to be sets of clock-rate/instruction-level parallelism pairs which provide near-optimal performance. Although many factors (compiler optimization, as well as chip and system designers' abilities) cloud a comparison, hardware measurement is the generally accepted method of judging the trade-offs. Benchmarks clearly illustrate that the optimal design point is very application-specific.

Tracking the performance of various systems for the past few years has made two points apparent. First, performance improves at a healthy pace in the workstation and server markets; without continual improvements, leaders soon lose their position. Second, performance for a given vendor is a stair-step function. Often the competition is close, with several vendors jockeying for the lead position. While at any instant a system may be dominant, leaders change frequently.

This paper describes the next generation of implementations of the POWER Architecture, POWER2™ processors, and systems. The initial three models are the 55-MHz Model 58H, the 66.5-MHz Model 590, and the 71.5-MHz Model 990. The arrival of the POWER2-based systems moves the RS/6000 family into the lead on many industry standard benchmarks, with a combination of increased clock rate, exploitation of architectural enhancements, doubled functional units, and increased cache capacity. The POWER2 enhanced superscalar capability further widens the gap between the instruction-level parallelism and clock rate approaches.

This paper consists of four major sections. The architecture section discusses enhancements to the programmer's view of the hardware, primarily new instructions which improve storage reference bandwidth, allow hardware square root, and speed floating-point-to-integer conversion. The implementation section provides a description of the POWER2 processor, including functional units, caches, and translation lookaside buffers (TLBs.) The third section describes the fabrication technology. The performance section examines how these changes affect performance and compares the resulting POWER2 performance to that of several competitive systems on a variety of industry standard benchmarks. The performance results demonstrate that superscalar capabilities are an attractive alternative to aggressive clock rates.

Architecture

The RS/6000 systems are implementations of a reduced instruction set computer (RISC) architecture. As is characteristic of many RISC architectures, loads and stores provide the only storage access; arithmetic instructions use only register operands. Several instructions, often considered more complex than a traditional RISC definition, enhance performance. The instructions include a floating-point multiply–add (FMA) instruction, a branch-on-count (BCT) operation, and update forms of storage references.

The FMA compound instruction consists of a floating-point multiply and a *dependent* add. On POWER and POWER2 implementations, the FMA operation performs the multiply and add with a total latency of only two cycles. Independent FMA instructions can start every cycle. The FMA operation allows a peak MFLOPS rate equal to twice the MHz rate while using a single functional unit. Many experts credit the FMA instruction as a key component of the RS/6000 processor's outstanding floating-point performance. The HP PA7100 has a similar compound operation that allows a floating-point multiply and an *independent* add. Simple coding of common constructs, such as inner product or daxpy, often involves dependent pairs of operations, requiring additional compiler complexity to exploit the HP compound operations.

The BCT form of a conditional branch decrements and tests a special-purpose register, the count register, to determine the outcome of the branch. Often a loop-closing branch can be coded using the BCT form; the programmer loads the count register with an iteration count for the loop, and the branch unit decrements and tests this value independently of other fixed-point unit (FXU) work. In many other architectures, a general-purpose register (GPR) is used to hold the iteration count, and the FXU performs the decrement and test. The FXU forwards the test result to the branch unit in the form of a condition code result. The RS/6000 BCT instruction and count register are examples of architectural separation of resources that enhance the implementer's ability to exploit instruction-level parallelism. The FXU can offload the loop count decrement and test operations, while the branch unit can accurately determine the fetch path without FXU synchronization. This results in a zero-cycle branch from the FXU's point of view.

Both addressing forms of storage references, indexed and displacement, support an "update form." This pre-update of the base register (with the effective address) greatly decreases the need for explicit address arithmetic. The multiple operations, which comprise each of the FMA, the BCT, and the update forms, allow designers an opportunity to provide instruction-level parallelism beyond

```
DO 100 i=1,n                          loop: lfd     fpr2,8(r11)          #a(i,j)
   a(i,j) = a(i,j) + a(i,k) * atemp         lfdu    fpr4,8(r12)          #a(i,k)
ENDDO                                       fma     fpr6,fpr2,fpr4,fpr8  #fpr8=atemp
                                            stfdu   fpr6,8(r11)          #a(i,j)
                                            bct     loop
   ************************************************************************
DO 100 i=1,n,2                        loop: lfq     fpr2,fpr3,16(r11)    #a(i,j), a(i+1,j)
   a(i,j) = a(i,j) + a(i,k) * atemp         lfqu    fpr4,fpr5,16(r12)    #a(i,k), a(i+1,k)
   a(i+1,j)=a(i+1,j) + a(i+1,k) * atemp     fma     fpr6,fpr2,fpr4,fpr8  #fpr8=atemp
ENDDO                                       fma     fpr7,fpr3,fpr5,fpr8
                                            stfqu   fpr6,fpr7,16(r11)    #a(i,j), a(i+1,j)
                                            bct     loop
```

Figure 1

Quad-word storage reference benefit on Linpack benchmark.

both the number of functional units and the available dispatch bandwidth.

POWER2 supports a superset of the POWER Architecture. New instructions provide performance opportunity: quad-word floating-point storage references, square root, and convert to integer. Virtual address translation changes improve performance and add capability. The architecture also adds hardware performance monitoring. Support of all POWER instructions maintains upward compatibility for programs.

- *New instructions*

The architecture adds high-performance floating-point storage access instructions, load quad and store quad, which support all of the addressing forms for double-precision storage references: indexed and displacement, with and without update forms. The quad-word (128 bits) loads move two adjacent double-precision storage operands into two adjacent floating-point registers (FPRs).

Because of the BCT branch and the implicit register updates available in storage reference instructions, most RS/6000 floating-point loops simply consist of storage references, floating-point arithmetic, and a BCT-type branch. The FXU (which executes all storage reference instructions) and floating-point unit (FPU) operate fairly autonomously. Therefore, either the number of storage references or the number of arithmetic instructions usually limits the number of cycles required to execute an iteration of a floating-point loop. When storage references limit the performance of a loop, load quad and store quad instructions can provide improvement.

The dominant loop from the Linpack benchmark [5] shown in **Figure 1** illustrates the quad-word benefit. The top code block represents the dominant loop after inlining but without unrolling. The pseudo-assembly code shows three storage reference instructions, the performance limiter for this code on a POWER processor. After unrolling this stride-1 loop (by a factor of two), the new loop contains three pairs of storage references. Each pair involves two adjacent storage locations and two adjacent FPRs. As shown in the bottom code block, a quad-word reference can replace each pair. Because an iteration of the unrolled loop with quad-word storage references requires the same number of cycles as an iteration of the original loop, the quad-word storage reference capability almost doubles the performance of storage-reference-limited code such as the Linpack benchmark.

In addition to the quad-word storage reference instructions, the architecture adds a square-root instruction. On previous RS/6000 systems, a library routine call provided the square-root function. When the call is replaced with a single instruction, the number of cycles per operation drops from about 50 to roughly 25. In the SPEC CFP92 suite, hardware square root provides a substantial gain on the ORA benchmark, which spends about 50% of its time in the library square-root routine. Application areas that exhibit performance gains from the square-root instruction include computational physics and graphics.

Additional new instructions allow more efficient conversion of a floating-point value to an integer value. The fcir and fcirz instructions provide the conversion with default rounding and with round toward zero, respectively. They improve random number generation where the seed is a floating-point value but the modulo arithmetic calculations require integer inputs. Other examples of use include histogram updates and table-lookup routines that convert a floating-point input value into an integer value for indexing a table or array. Furthermore, interpolation can use the floating-point-to-integer conversion to determine which two adjacent grid points (integer indices) surround a calculated point on a grid. The compiler can use fcirz to provide the Fortran INT intrinsic function.

- *Enhanced translation*

In addition to the newly added instructions, users benefit from a performance gain from the modifications to the virtual address translation process. As in most virtual memory systems, the operating system manages a set of architecturally defined page tables which maintain a mapping of virtual pages to real pages. To increase the efficiency of this process, TLBs cache translation information for recently accessed pages. When a storage reference requires translation information which is not available in the TLBs, a TLB miss occurs, and hardware searches the page tables for the translation information and either updates the TLBs or signals the occurrence of a page fault.

The POWER hardware TLB miss search consists of hashing bits from the virtual address, indexing a hash anchor table (HAT) to obtain a pointer into the page frame table (PFT), and following the pointers through a chain of PFT entries to find a match. Because the translation information is noncacheable, this process requires a minimum of two cache misses per TLB miss: one for the HAT entry and one for the PFT entry. Since this PFT is an inverted page table, entries along a PFT chain tend to be scattered through the table; each additional search step along a PFT chain usually incurs an additional cache miss. Another attribute of an inverted page table is that it can map only one virtual page to a given real page. Support for aliasing at the page level results in "ping-ponging" of PFT entries.

Rather than chaining PFT entries together, the POWER2 translation scheme [6] places PFT entries (for pages whose addresses hash to the same value) in a contiguous group. The virtual address hashing results in a pointer directly to the first entry in the group, rather than indirectly through the HAT, reducing the number of storage references. The POWER2 PFT entries are cacheable. As a result, the table walk associated with a TLB miss often requires at most one cache miss. The new definition of the PFT structure (away from an inverted PFT) simplifies page-level aliasing.

- *Performance monitor*

While the preceding architectural enhancements aim at improving performance, a set of performance monitors allows observation of many processor aspects that affect performance [7]. This facility allows the measurement of

- Instruction/data cache/TLB misses.
- Functional unit utilization.
- Instruction distribution by a functional unit.
- Number of instructions executed from a particular class or group.

This performance monitor facility provides a wealth of information that allows the detection of performance bottlenecks. Examples of how this monitor facility can be used on applications can be found in [8].

Implementation

As shown in **Figure 2**, the POWER2 processor complex consists of eight semi-custom chips partitioned in a fashion similar to POWER: an instruction cache unit (ICU) which also processes branches, the FXU, the FPU, four data cache units (DCUs), and a storage control unit (SCU). The ICU prefetches instructions from the I-cache and places them in instruction buffers. ICU control logic decodes or analyzes the instructions in the buffers. The ICU executes ICU instructions (primarily branches), sometimes affecting the prefetch path. The ICU *dispatches* non-ICU instructions to the FXU and FPU over a four-instruction-wide *instruction dispatch* bus (I-bus). The FXU and FPU process their respective arithmetic instructions. The FXU also processes storage reference instructions by generating and translating the addresses before placing them on the cache address bus.

The FXU contains the D-cache directories, while the ICU contains the I-cache directories and arrays. When a data (instruction) cache miss occurs, the FXU (ICU) arbitrates for the processor bus (P-bus). After the FXU (ICU) places the cache miss request on the P-bus, the SCU "sees" the request and generates the corresponding memory control signals to start a memory operation. The returning data arrive at the DCUs, which place the data in the D-cache (or else forward the data to the ICU on the instruction reload bus).

The FXU and FPU each contain two execution units. The memory, instruction reload, and instruction dispatch buses, as well as the interface from the DCU to the FXU and FPU, are wider than in the POWER implementation to support these additional execution units. The POWER2 implementation can execute six instructions (branch, condition register, two fixed-point, and two floating-point) per cycle. Like POWER, the POWER2 FPU supports

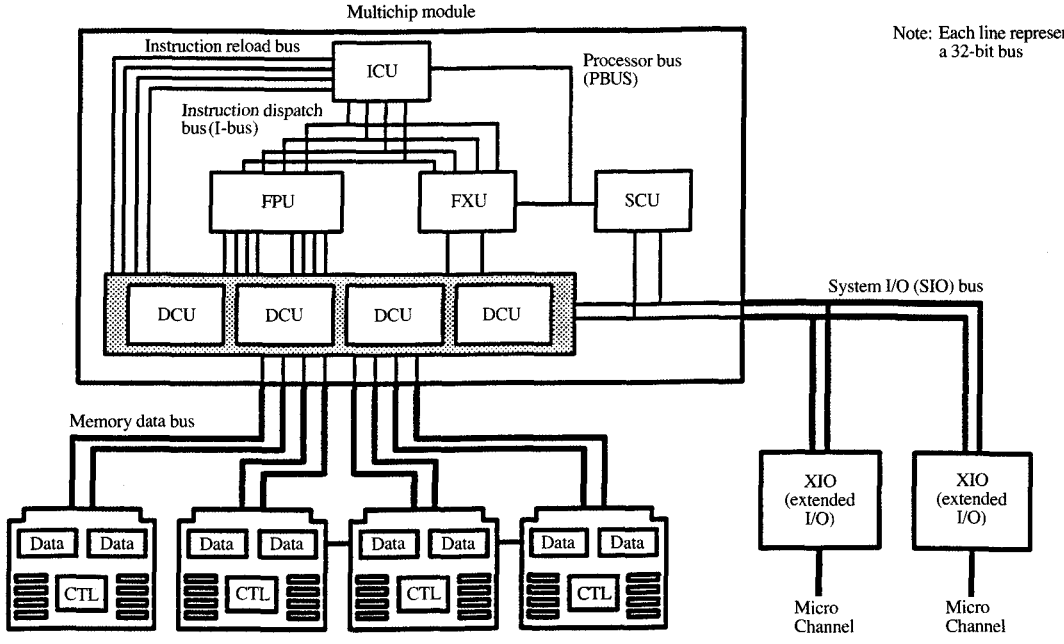

Figure 2

POWER2 eight-word system.

the compound FMA instruction. Because each of the two floating-point instructions can produce both an add result and a multiply result, the peak execution rate is eight operations per cycle. The interfaces between the multichip module (MCM) and the memory and I/O units are compatible with those in POWER systems.

The POWER2 processor chip set offers two system configurations: a four-word memory bus with a 128KB D-cache and an eight-word memory bus with a 256KB D-cache. The I-cache is 32KB for either configuration. Figure 2 shows an eight-word memory system. A four-word system differs in that it has only two memory cards and supports transfers of four words per cycle between memory and the MCM.

Upgrade paths from POWER-based systems (for example, from Models 570 and 580 to the Model 590) exist. The upgrade process replaces the CPU planar along with some miscellaneous parts such as an air dam. An upgraded system requires new memory cards; however, the 80-ns single in-line memory modules (SIMMs) from the upgraded system may populate the new memory cards.

- *Instruction cache unit*

The ICU fetches and dispatches instructions. It contains a two-way set-associative 32KB I-cache with 128-byte lines, the associated I-cache directories, and a 128-entry two-way set-associative instruction TLB (I-TLB). The ICU control logic fetches instructions from the I-cache and places them in one of two instruction buffers, depending on the expected path and the occurrence of conditional branches. The buffer for sequential path instructions has 16 elements. The ICU removes unconditional branches (and instructions following these branches) from the sequential buffer, alters the fetch path, and appends instructions along the new path to those remaining in the sequential buffer. The fetch logic places instructions from a conditional branch target path in an eight-entry target buffer. The ICU processes instructions in the sequential buffer. It executes branch and condition-register instructions and dispatches the remaining instructions to the FXU and FPU. The ICU can fetch eight instructions per cycle from the I-cache regardless of alignment. It can dispatch six instructions per cycle: two internally and four externally to the FXU and FPU.

Generally, the FXU and FPU receive an uninterrupted instruction stream; often the ICU can process branches without introducing pipeline delays. The ICU dispatches the sequential path instructions to the FXU and FPU. The ICU conditionally dispatches instructions beyond an unresolved conditional branch; the FXU and FPU hold conditionally dispatched instructions in their instruction buffers until the branch is resolved. Then the FPU or FXU issues or cancels the conditionally dispatched instructions.

To minimize conditional branch delay after an incorrect guess, the ICU also fetches the branch-taken path from the I-cache and dispatches the target instructions over the I-bus. Until the branch is resolved, the FXU and FPU ignore these target path instructions. If the branch is resolved as "not taken," the instructions for the correct path are in the FXU/FPU instruction buffers and usually incur no penalty. If the ICU or FXU resolves the branch as "taken," the FXU and FPU cancel the conditionally dispatched instructions and load the target instructions (from the I-bus) into the FXU and FPU instruction buffers.

Unconditional branches, not-taken conditional branches, and "taken but resolved" branches often cause no pipeline delay. Taken branches which are not resolved when they are processed often result in a one-cycle delay in the FXU/FPU pipelines. More ICU details can be found in [9].

- *Fixed-point unit*

The FXU performs all storage references, integer arithmetic, and logical operations. The FXU contains the GPRs, two fixed-point execution units, the data cache directory, the data TLB (D-TLB), and the TLB reload hardware. Two eight-port (4R/4W) register files, one for each execution unit, implement the thirty-two 32-bit GPRs. The register files implement full bypass to minimize delay between dependent operations. The FXU performs register scoreboarding so that D-cache accesses do not hold off subsequent independent register-to-register instructions.

The dual execution units can execute a total of two instructions per cycle. An eight-element (six-element in POWER) FXU instruction buffer, which holds only fixed-point instructions, feeds the dual execution units. Each unit contains an adder and a logic functional unit. The second unit also contains a multiply and divide unit. The multiply/divide unit executes a multiply in two cycles (three to five cycles in POWER) and a divide in 13 to 17 cycles (19–20 cycles in POWER).

When both functional units are ready to accept a new instruction, the issue logic sends the (logically) first instruction to the first unit. If a second FXU instruction exists in the FXU instruction buffers, and its input operands are available, the issue logic sends it to the second unit. To minimize data dependencies, the upper portion of the second unit has a three-input adder. As a result, if an add-type operation (or storage reference with update) in the first unit is updating the source registers for the second FXU instruction, the second instruction can execute in parallel.

Since address generation uses the GPRs in the FXU, the FXU executes all storage reference instructions, including the address translation and D-cache directory search. It contains the sixteen 32-bit segment registers (effective to virtual) and a 512-entry (128-entry in POWER) two-way set-associative D-TLB (virtual to real) for address translation. It also has TLB reload hardware for processing both I-TLB and D-TLB misses. Dual-ported D-cache arrays and dual-ported directories support the two execution units. The cache supports first-order nonblocking accesses; subsequent load/store operations can overlap with D-cache misses.

Single-element loads incur the same load-use delay as POWER; one cycle separates the load from a dependent arithmetic. The dual execution units allow two storage references (including quad-word) per cycle. In POWER systems, hardware handles the most frequent cases for unaligned accesses. POWER2 extends hardware support for unaligned accesses to include quad-word accesses on any odd-word boundary. Both POWER2 FXU execution units work together on multicycle load/store operations. The issue logic loads the opcode into both units, and two load/store operations execute per cycle. A detailed description of the FXU implementation can be found in [6].

- *Floating-point unit*

The FPU includes the FPRs and two double-precision (64-bit) execution units. Dual units allow execution of two floating-point instructions per cycle. As in POWER, POWER2 implements FPR renaming to increase FXU/FPU autonomy. The number of physically implemented registers exceeds the number (32) defined by the architecture. Mapping hardware selects a "free" physical register to hold the destination value for each floating-point load operation. Because loads do not overwrite the previous data in the architected register, functional units may execute a load before the completion of a logically previous arithmetic instruction, even if it loads an FPR which is a source operand for the arithmetic. This is a substantial benefit on short loops, where register renaming cannot be done in software. Moving the loads further ahead allows the effects of load–use delay to be minimized. To support the increased number of functional pipelines, the FPU supports 54 physical registers (40 in POWER processors).

The functional units in the FPU are more symmetrical and autonomous than in the FXU. Usually the FXU can only issue an instruction to the second functional unit

if two instructions are ready for issue simultaneously. In the FPU, whenever an instruction is ready to be issued and a functional unit is available, the FPU can issue the instruction. Except for compares, both FPU units perform all operations, including divide and square root instructions. An eight-element instruction buffer, which may contain eight floating-point interruptible instructions, feeds the execution units. Arithmetic results conform to the IEEE 754 binary floating-point standard.

As in POWER, the POWER2 FPU supports the compound multiply–add instruction. The pipelines perform two operations with a single rounding of the result and with the same latency as a single multiply or add operation. Dual units enable the FPU to execute two double-precision multiply/add instructions every cycle, resulting in up to four floating-point operations per cycle.

When loops contain multiple divides, or when single divide loops are unrolled, proper scheduling may allow divide operations to be performed in parallel, providing roughly a factor of two advantage over POWER. Since the square-root instruction is roughly twice as fast as the POWER library sequence, dual execution of square-root instructions gives POWER2 roughly a factor-of-four advantage over POWER for this function.

To exploit the bandwidth potential of the quad-word storage reference instructions, a dual quad-word interface to the data cache supports the dual FPU execution units. Dual units and quad-word storage references can load up to four FPRs per cycle.

The POWER2 FPU has a separate unit for normalizing store data, allowing stores to execute in parallel with arithmetic operations. As a result, floating-point stores effectively require zero FPU cycles, whereas they require one FPU cycle in POWER. Additional FPU details can be found in [10].

- *Data cache unit*

POWER2 has a four-way set-associative dual-ported D-cache that consists of four identical chips. These four chips generate two one-word data buses to the FXU, two quad-word buses to the FPU, a four-word instruction reload bus to the ICU, and a two-word system I/O (SIO) bus to the I/O subsystem for DMA data (see Figure 2). For increased reliability, the DCUs support

- Memory scrubbing.
- Single-bit correction double-bit detection error checking code (ECC).
- Bit steering.

Two DCU configurations are possible: a 256KB capacity cache (256-byte lines) fed with an eight-word memory bus or a 128KB data cache (128-byte lines) with a four-word memory bus. (POWER supports 64-byte and 128-byte lines.) The memory system can include two, four, or eight memory cards. When the system contains two memory cards, it configures itself as a four-word memory system. If there are more than two memory cards, an eight-word memory system exists. A detailed description of the DCU, SCU, and memory and I/O interfaces can be found in [6]. The store-back cache design minimizes memory bus traffic.

- *Storage control unit*

The SCU contains the controls and configuration registers for memory, and arbitrates for all communications between the CPU (ICU, FXU, DCU), the memory, and the SIO bus. It generates the controls for the SIO bus and a data path for programmed I/O. The SIO bus enhancements allow prefetching of DMA read data from memory, enabling the system to sustain high DMA rates using the streaming data protocol on the Micro Channel. POWER systems can use memory cards from POWER2 systems.

- *I/O unit*

The POWER2 I/O unit is the same as the one in the RS/6000 Models 580 and 980. The I/O unit implements the 64-bit streaming data protocol on the Micro Channel at 10 MHz. The I/O unit implements dual 64-byte buffers per DMA channel so that operations over the SIO bus and Micro Channel can fully overlap. The I/O unit, along with some logic on the I/O planar, reduces the arbitration time on the Micro Channel from 400 ns to 100 ns. This improves bandwidth and bus utilization. In addition, enhancements to the protocol for the SIO bus include prefetch data commands from the I/O unit so that the DMA data from memory are available to the I/O unit with minimum delay.

For DMA transfers to memory, the I/O unit keeps a record of every modified byte in the 64-byte buffer. If all bytes are modified, a write to memory results. If only some of the bytes in the 64-byte buffer are modified, the I/O unit performs a read–modify–write. The previous version of the I/O unit performs a read–modify–write for all transfers to memory. This implementation almost doubles the throughput for all DMA operations to memory.

Chip and packaging technology

Table 1 shows the transistor counts and die sizes for the chips contained on the MCM, as well as the external signal I/O count for the MCM. The POWER2 processor chip set contains over 23 million transistors on a silicon area of 1217 mm^2. The chips incorporate CMOS technology with an effective channel length of 0.45 μm. The chips contain one level of polysilicon and four levels of metal wiring.

POWER2 systems mount the chips on a ceramic MCM that provides most of the chip-to-chip wiring. While the total chip I/O is 3181, all but 512 connections are internal to the MCM. The MCM includes a 64-mm by 64-mm pin

Table 1 Physical attributes of the POWER2 processor chip set.

Chip values and MCM totals	*Transistor count* (thousands)		*Die size* (mm × mm)	*Signal I/O*
	Logic	*Memory*		
FXU	583	848	12.7 × 12.7	473
FPU	1001	315	12.7 × 12.7	504
ICU	547	2277	12.7 × 12.7	464
DCU (×4)	1117	16000	12.7 × 12.7	366
SCU	349	—	9.4 × 9.4	276
MCM	3597	19440	1217 sq. mm	512

Table 2 Industry standard benchmark comparison.

System	*RS/6000*		*HP PA7100*	*DEC 10000*
	Model 580	*Model 590*	*Models 735/755*	*Model 610 AXP™*
Clock rate	62.5 MHz	66.5 MHz	99 MHz	200 MHz
SPECint92	73.3	117.0	80.6	116.5
SPECfp92	134.6	242.4	149.8	193.6
Linpack	38	130	41	43
TPP	104	236	107	155
TPC-C (tpmC)	N/A	726.13	613.80*	N/A
$/tpm-C	N/A	$1603	$2488*	N/A

*The TPC values for HP are not available for Models 735/755. The TPC-C values shown in Table 2 for HP are for the 96-MHz HP 9000 series 800 Model H50, which is based on the HP PA7100 processor.

grid array with 512 signal I/Os. At 66.5 MHz, the total power dissipated by the MCM is about 65 W. More MCM and packaging information can be found in [11].

Performance

There are two categories for the performance gains associated with the improvements described previously. The first category applies to code that a compiler may have generated prior to POWER2-based systems. Without recompiling, many programs obtain benefits from the larger caches and TLB structures, as well as the enhanced address translation process. Calls to library routines such as ESSL obtain POWER2-specific tuning benefits when new libraries are linked. Furthermore, the additional functional units exploit the instruction-level parallelism which has been exposed in the compiled code.

The second category includes the enhanced exploitation of the functional units obtainable through recompiling for a POWER2 target. The compiler can expose additional instruction-level parallelism after more aggressive loop unrolling. POWER2 scheduling, which takes into account POWER2 latencies and interlocks, can improve performance. Recompiling also allows applications to benefit from the new instructions. Reference [12] contains benchmark data comparing the performance effects of existing binaries versus recompiled applications.

The previous three sections described architectural, implementation, and technology details that allow the RS/6000 processor to increase its clock rate while significantly increasing its superscalar abilities. This paper opened with the debate topic: "Which delivers more performance, instruction-level parallelism or clock rate?" The complexity of the trade-offs and their interactions makes this a difficult question. Proof of a good answer is overall performance. **Table 2** compares a POWER2-based system with a POWER-based system and two competitive workstations/servers [5, 13–16]. While new announcements continually move the performance bar forward, Table 2 shows the publicly available data as of October 1, 1993. While the HP and DEC systems have substantially higher clock rates, the POWER2-based system uses instruction-level parallelism to move ahead on all of these industry-standard benchmarks. (During the final review of this paper, DEC announced slightly higher values which are roughly equivalent to those for the 71.5-MHz IBM Model 990.)

The SPEC CINT92 suite consists of six integer codes representing compilers, spreadsheets, and so on. The SPEC CFP92 suite consists of 14 floating-point benchmarks from the workstation and server market. SPEC rules do not permit hand optimization of the codes in either suite. The overall measure in each suite is the geometric mean for the benchmarks within the suite. The Model 590 win in the SPECint92™ race confirms that superscalar ability can compensate for a three times greater clock rate advantage, even on SPECint92. Because of the characteristics of SPECfp92™, one expects even greater opportunity to exploit a superscalar approach. The 25% win of the POWER2 system over competitors with up to a factor-of-three clock rate advantage, as well as the large gain over POWER-based systems, demonstrate the additional superscalar opportunity in SPECfp92.

The Linpack benchmark represents one specific problem area: solving dense systems of equations. The benchmark rules do not allow hand optimization while solving the 100 × 100 matrix problem. For POWER, the optimized inner loop asymptotically approaches a load and a store per element update, resulting in an FXU limit of two cycles. The HP PA7100 requires two cycles for a store. The resulting limit of three cycles per update counteracts the rough 3:2 clock rate advantage of PA7100 over POWER (Model 580). As a result, the Linpack performance of these two systems is very similar. As discussed earlier, POWER2 systems benefit substantially on Linpack from quad-word storage references. Linpack performance receives almost another factor-of-two improvement as the number of functional units doubles.

While the Linpack performance of the POWER-based system almost matches that of the competitive systems, POWER2 systems boast more than a factor-of-three advantage.

The TPP (Toward Peak Performance) benchmark solves the same type of problem as Linpack, on a 1000×1000 size problem, but allows hand optimization of the code. Because this is a numerically intensive benchmark which exploits hand-coded library routines on many systems, this benchmark often approaches the peak MFLOPS rate of a system. The TPP value for POWER2 easily surpasses even the theoretical peak rates of the HP and DEC Alpha systems, 198 and 200 MFLOPS, respectively.

TPC-C™ is a prominent commercial benchmark. Unlike the other benchmarks in this comparison, which run a controlled source version of a program, TPC-C is a specification of a workload that "is a mixture of read-only and update intensive transactions that simulate the activities found in complex OLTP [on-line transaction processing] application environments" [17]. Detailed TPC-C characteristics and POWER2's TPC-C performance are described in [18].

Summary

A popular debate topic concerns the design philosophy dichotomy evident in the workstation and server markets. Aggressive superscalar characterizes the RS/6000 POWER-based systems. DEC Alpha and HP PA7100 systems illustrate the aggressive clock rate approach. POWER2's quad-word storage references and additional functional units make the divergence even more pronounced. While the leader in this performance race changes frequently, industry standard benchmarks demonstrate how POWER2-based systems allowed the RS/6000 family to move ahead of other implemented design points. This result confirms that instruction-level parallelism is a justifiable alternative to clock rate in the performance race.

Acknowledgments

The POWER2 extension of the IBM RISC System/6000 product family results from the efforts of the entire IBM Austin team, as well as many people primarily at the Yorktown, Burlington, and Toronto sites. Many of the key technical individuals are the authors of the companion papers and cited references. Management leadership for the chips, systems, and compilers includes Phil Hester, George Lerom, Rudy Pirovitz, Dwayne Crider, Don Johnson, Bob Gerber, and Karen Bennet. We thank Ron Seaman for an outstanding job of managing the POWER2 system development schedule and for making early systems available. Finally, we thank John Reysa and Kate Stewart for reviewing this manuscript and providing many useful suggestions.

References

1. H. B. Bakoglu, G. F. Grohoski, and R. K. Montoye, "The IBM RISC System/6000 Processor: Hardware Overview," *IBM J. Res. Develop.* **34,** No. 1, 12–22 (January 1990).
2. John Cocke and V. Markstein, "The Evolution of RISC Technology at IBM," *IBM J. Res. Develop.* **34,** No. 1, 4–11 (January 1990).
3. Linley Gwennap, "Speed Kills? Not for RISC Processors," *Microprocessor Report*, March 8, 1993, p. 3.
4. Steven W. White, Phil D. Hester, Jack W. Kemp, and G. Jeanette McWilliams, "How Does Processor MHz Relate to End-User Performance?," *IEEE Micro*, Part 1: Vol. 13, No. 4, 8–16 (August 1993); Part 2: Vol. 13, No. 5, 79–89 (October 1993).
5. Jack Dongarra, "Performance of Various Computers Using Standard Linear Equations Software," *Computer Architecture News (USA)* **20,** No. 3, 22–44 (June 1992). (*Note:* Since its original publication, the data in this article have been updated several times by its author without formal republication. Information cited above in Table 2 is actually taken from the April 15, 1993 version. Current updates may be obtained by sending e-mail to netlib@ornl.gov with the message "send performance from benchmark.")
6. D. J. Shippy, T. W. Griffith, and Geordie Braceras, "POWER2 Fixed-Point, Data Cache, and Storage Control Units," *PowerPC and POWER2: Technical Aspects of the New IBM RISC System/6000*, pp. 29–44; Order No. SA23-2737, 1994; available through IBM branch offices. [D. J. Shippy and T. W. Griffith, "POWER2 Fixed-Point, Data Cache, and Storage Control Units," *IBM J. Res. Develop.* **38,** No. 5, 503–524 (September 1994, this issue).]
7. E. H. Welbon, C. C. Chan-Nui, D. J. Shippy, and D. A. Hicks, "POWER2 Performance Monitor," *PowerPC and POWER2: Technical Aspects of the New IBM RISC System/6000*, pp. 55–63; Order No. SA23-2737, 1994; available through IBM branch offices. [E. H. Welbon, C. C. Chan-Nui, D. J. Shippy, and D. A. Hicks, "The POWER2 Performance Monitor," *IBM J. Res. Develop.* **38,** No. 5, 545–554 (September 1994, this issue).]
8. Sohel R. Saiyed, J. Michael O'Connor, and Maurice Franklin, "POWER2 CPU-Intensive Workload Performance," *PowerPC and POWER2: Technical Aspects of the New IBM RISC System/6000*, pp. 129–136, Order No. SA23-2737, 1994; available through IBM branch offices.
9. Jama Barreh, Baba Arimilli, Robert Golla, and Paul Jordan, "POWER2 Instruction Cache Unit," *PowerPC and POWER2: Technical Aspects of the New IBM RISC System/6000*, pp. 19–28, Order No. SA23-2737, 1994; available through IBM branch offices. [J. I. Barreh, R. T.

Golla, L. B. Arimilli, and P. J. Jordan, "POWER2 Instruction Cache Unit," *IBM J. Res. Develop.* **38,** No. 5, 537–544 (September 1994, this issue).]
10. Troy N. Hicks, Richard E. Fry, and Paul E. Harvey, "POWER2 Floating-Point Unit: Architecture and Implementation," *PowerPC and POWER2: Technical Aspects of the New IBM RISC System/6000*, pp. 45–54, Order No. SA23-2737, 1994; available through IBM branch offices. [*IBM J. Res. Develop.* **38,** No. 5, 525–536 (September 1994, this issue).]
11. Rathna Reddy and David Galvin, "POWER2 Electrical and Mechanical Packaging," *PowerPC and POWER2: Technical Aspects of the New IBM RISC System/6000*, pp. 64–68, Order No. SA23-2737, 1994; available through IBM branch offices.
12. John Reysa, Da-Gung Lu, Kate Stewart, Ahmed Chabib, and Sohel R. Saiyed, "Migration and Compatibility," *PowerPC and POWER2: Technical Aspects of the New IBM RISC System/6000*, pp. 122–128, Order No. SA23-2737, 1994; available through IBM branch offices.
13. Kaivalya Dixit and Jeff Reilly, "SPEC Developing New Component Benchmark Suites," *SPEC Newsletter* **3,** No. 4, 14–17 (December 1991).
14. *SPEC Newsletter* **5,** Nos. 1 and 2, March 1993 (HP) and June 1993 (DEC on p. 28 and p. 54).
15. IBM Announcement, Sept. 21, 1993, New York, Order No. G221-3223-07; available through IBM branch offices.
16. HP TPC-C Full Disclosure Report, Sept. 7, 1993, available from Shanley Public Relations, San Jose, CA; (408) 295-8894; e-mail shanley@cup.portal.com.
17. "TPC Benchmark C," Standard Specification Revision 1.1, Transaction Processing Performance Council (TPC), June 1, 1993, p. 1.
18. Maurice Franklin, William Alexander, Rajiv Jauhari, Ann Marie Grizzaffi Maynard, and Bret Olszewski, "POWER2 Commercial Workload Performance," *PowerPC and POWER2: Technical Aspects of the New IBM RISC System/6000*, pp. 137–144, Order No. SA23-2737, 1994; available through IBM branch offices. [M. T. Franklin, W. P. Alexander, R. Jauhari, A. M. G. Maynard, and B. R. Olszewski, "Commercial Workload Performance in the IBM POWER2 RISC System/6000 Processor," *IBM J. Res. Develop.* **38,** No. 5, 555–561 (September 1994, this issue).]

Received June 3, 1993; accepted for publication April 16, 1994

Steven W. White *IBM RISC System/6000 Division, 11400 Burnet Road, Austin, Texas 78758 (white@austin.ibm.com).*
Dr. White is a senior engineer in the Processor Architecture and Performance group at IBM Austin. He received his B.S.E.E., M.S.E.E., and Ph.D. degrees from Texas A&M University, where he also taught in the Electrical Engineering Department for three years. In 1982, he joined IBM to work on mainframe scientific and engineering processor development, architecture, and system design. He had a two-year assignment with the Computational Physics Group at Livermore National Laboratory, and moved to his current position in 1990. Dr. White has published papers in a variety of areas including VLSI design, parallel processing, numerical algorithms, code optimization, system design, performance analysis, human genome research, and compilers. His primary interest is scientific and engineering performance. He is a member of IEEE and a registered professional engineer.

Sudhir Dhawan *IBM RISC System/6000 Division, 11400 Burnet Road, Austin, Texas 78758 (DHAWAN at AUSVM6).*
Dr. Dhawan is a senior engineer in the Processor/System Development area at IBM Austin. He received a Ph.D. in computer engineering from the University of New Mexico at Albuquerque. Dr. Dhawan joined IBM at Austin in 1983; he has worked in I/O, processor, and system development on IBM workstations. He has filed five patents and published several papers.