MICROPROCESSOR REPORT
THE INSIDERS' GUIDE TO MICROPROCESSOR HARDWARE

VOLUME 6 NUMBER 14 OCTOBER 28, 1992

IBM Delivers First PowerPC Microprocessor

Superscalar 601 Sets the Stage for New Apple Macintosh Line

By Brian Case

IBM and Motorola have formally announced the successful fabrication of PowerPC 601, which is the first implementation of the microprocessor architecture jointly developed by the Apple/IBM/Motorola alliance. The 601 is a superscalar implementation that integrates an integer unit, a floating-point unit, a branch unit, an MMU, and a large cache.

The Apple/IBM/Motorola agreement to jointly develop and use PowerPC microprocessors was inked a little more than a year ago. At that time, there was considerable doubt that the first implementation would be delivered on time. The doubts were based not on technical considerations but on the aggressive schedule—calling for 601 samples in fall '92—and the belief that the three companies would waste considerable time simply figuring out how to work together. As it turns out, first samples of the 601 have been delivered to Apple exactly on schedule.

The 601 integrates 2.8 million transistors, which is large, even by today's standards. What is most impressive, however, is the die size: a mere 10.95 cm square. The 601 is only 40% bigger than the 0.8-micron 486 but has over twice the transistors and four times the cache. IBM's aggressive fabrication process is the key to this density: 0.6-micron CMOS with four layers of metal plus a fifth metal layer for local interconnect.

The chip will be offered in 50 MHz and 66 MHz speeds. While these clock rates are nothing special by the standards of recent high-end processors, the superscalar implementation will yield competitive performance. The chip requires a 3.6-volt power supply and dissipates 9 watts at 50 MHz. Interface levels are TTL or CMOS compatible, and the chip is packaged in a 304-pin QFP with 184 signal pins. While key customers have received early samples, general sampling is expected to begin in December with volume production scheduled for the middle of 1993. No pricing has been announced.

The initial Apple/IBM/Motorola plan calls for three other PowerPC microprocessors. At the Microprocessor Forum, Chuck Moore said that IBM and Motorola are planning to have all four parts available in at least sample quantities by the end of 1993. The 601 is considered the processor for low-cost desktop computers. The 603 will be about the same performance as the 601 but aimed at low-cost, low-power applications. The 604 will be the mainstream processor for desktop computers and will be faster than the 601. The 620 will be for use in the highest-performance workstation-class machines.

The 601 will be manufactured by IBM but sold by Motorola. The future family members will be designed by the joint IBM/Motorola/Apple team and built by Motorola for sale on the merchant market. IBM may build the chips for its own use and is also seeking OEM business.

601 Ancestry

The PowerPC architecture, jointly defined by representatives of Apple, IBM, and Motorola, is an evolution of the POWER architecture used in IBM's RS/6000 workstations and servers. The PowerPC architecture deletes some POWER instructions and adds some new ones, as shown in Table 1.

The 601 is based on IBM's RSC (RIOS single chip) microprocessor, which is a low-end implementation of the POWER architecture used in IBM's Model 220. For at least two reasons, the 601 implements both the full POWER and PowerPC architectures (i.e., it includes the POWER instructions not required for PowerPC). First, it was easier to build upon the RSC core by leaving existing functions intact. Second, having the complete POWER architecture present in the 601 will let IBM run old software without trap overhead.

Chip Overview

As with nearly all recent, high-end microproces-

sors, the 601 integrates the entire processor subsystem on a single chip. As shown in Figure 1, it has pipelined integer (fixed-point) and floating-point execution units, a separate branch unit, a TLB-based MMU, and a 32-Kbyte cache. One unusual feature is the combined instruction/data cache; most other processors—with the notable exception of the 486—have separate instruction and data caches. The 601 cache's eight-way set associativity is also unusually high.

The block diagram shows many logical cache ports, but there are really only separate 256-bit input and output ports. Hardware to arbitrate and prioritize requests for cache access make sure cache bandwidth is allocated to maximize performance.

Like all descendants of the original IBM RIOS chip set, the 601 is a superscalar implementation. While the RSC can issue up to two instructions in a cycle from a three-instruction window, the 601 can issue up to three instructions per cycle from a four-instruction window.

Figure 2 shows the pipeline structure for the major instruction subsets. The 601 has a dedicated branch offset adder in the branch unit to allow branch instructions to be executed quickly in a two-stage pipeline. The RSC, on the other hand, uses the integer ALU to compute branch addresses, which requires an extra cycle and prevents superscalar dispatch of relative branches and integer ALU instructions.

Integer instructions use a traditional four-stage pipeline. Memory-reference instructions are executed by the fixed-point pipeline, but compared to integer ALU instructions, the cache access adds an extra pipeline stage and an extra cycle of latency.

The bus interface is a slightly modified version of Motorola's 88110 bus. The basic 88110 bus protocols and the 64-bit data bus width are retained, but modifications were made to improve the support for second-level caches in multiprocessor systems. For example, pins were added to tell a second-level cache which of the eight sets in the on-chip cache is being replaced on a cache miss. For systems that maintain inclusion in the second-level cache, a pin has been added to let the second-level cache know whether or not a dirty push (writing cached data to memory) leaves a copy of the data behind in the on-chip cache.

Also, while the 88110 bus protocol allowed pipelining of outstanding transactions on the bus, the 88110 did not implement it. The 601 can pipeline up to two outstanding bus operations, which IBM says provides a significant performance boost.

IBM claims any ASICs that were designed to work with the 88110 can easily be modified to work with the 601. Apple would certainly have lobbied for compatibility in the interest of leveraging whatever work it had completed on its 88110-based machine. Rumors persist that NeXT

cntlzd[.]	Count leading zeros dblwd
dcbf	Data cache block flush
dcbst	D-cache blk. store
dcbt	D-cache blk. touch
dcbtst	D-cache blk. touch for store
dcbz	D-cache block set to zero
divd[o][.]	Divide dblwd
divdu[o][.]	Divide dblwd unsigned
divw[o][.]	Divide word
divwu[o][.]	Divide word unsigned
eieio	Enforce in-order exec. of I/O
extsb[.]	Extend sign byte
extsw[.]	Extend sign word
fadds[.]	FP add single
fcfid[.]	FP convert from integer dblwd
fctid[.]	FP cvt. to int. dblwd
fctidz[.]	FP cvt. to int. dblwd round to zero
fctiw[.]	FP cvt. to int. word
fctiwz[.]	FP cvt. to int. word rnd to zero
fdivs[.]	FP divide single
fmadds[.]	FP mul-add single
fmsubs[.]	FP mul-sub single
fmuls[.]	FP mul single
fnmadds[.]	FP negative mul-add sngl
fnmsubs[.]	FP negative mul-sub sngl
fres[.]	FP reciprocal est. sngl
frsqrte[.]	FP recip. sq. root est.
fsel[.]	FP select
fsqrt[.]	FP sq. root
fsqrts[.]	FP sq. root single
fsubs[.]	FP subtract single
icbi	I-cache block inval.
isync	Instruction synch.
ld	Load dblwd
ldarx	Ld dblwd and reserve indexed
ldu	Ld dblwd with update
ldux	Ld dblwd with update indexed
ldx	Ld dblwd indexed
lwa	Ld word algebraic
lwarx	Ld word and reserve indexed
lwaux	Ld word alg. with update indexed
lwax	Ld word alg. indexed
mftb	Mv. from time base
mftbu	Mv. from time base upper
mulhd[.]	Mul high dblwd
mulhdu[.]	Mul high dblwd unsgnd
mulhw[.]	Mul high word
mulhwu[.]	Mul high word unsgnd
rldcl[.]	Rot. left dblwd, clear left
rldcr[.]	Rot. left dblwd, clear right
rldic[.]	Rot. left dblwd imm., clear
rldicl[.]	Rot. left dblwd imm., clear left
rldicr[.]	Rot. left dblwd imm., clear right
rldimi[.]	Rot. left dblwd imm., mask insert
sld[.]	Shift left dblwd
srad[.]	Shift right algebraic dblwd
sradi[.]	Shift right algebraic dblwd imm.
srd[.]	Shift right dblwd
std	Store dblwd
stdcx	St. dblwd cond. indexed
stdu	St. dblwd with update
stdux	St. dblwd indexed, update
stdx	St. dblwd indexed
stfiwx	St. FP as int. wd indexed
stwcx.	St. word cond. indexed
subf[o][.]	Subtract from
sync	Synchronize
td	Trap dblwd
tdi	Trap dblwd immediate

Table 1. New instructions in the PowerPC architecture.

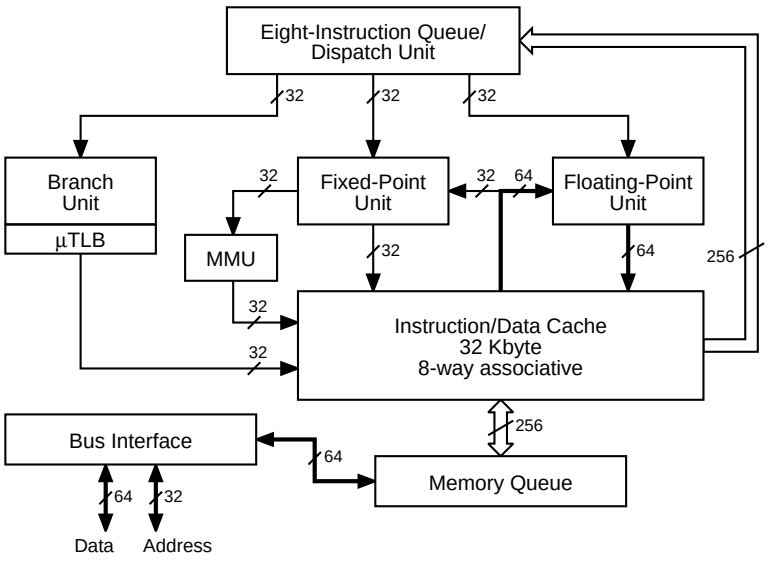


Figure 1. PowerPC 601 block diagram.

is planning to introduce an 88110-based computer, but it is possible that the relatively compatible bus and early silicon of the 601 might cause them to switch.

In view of the effort put into the design of the 601 bus, it seems likely that future members of the PowerPC family, especially the 603 and 604, will maintain bus compatibility with the 601. Doing so would make it easy for Apple to leverage existing motherboard designs and provide an Intel-like, "overdrive" upgrade strategy.

Integer Unit

For the most part, the integer (fixed-point) unit is traditional. In the original RIOS (RS/6000) chip set, the integer unit was decoupled from the instruction dispatch logic by an integer instruction queue. In RIOS, two integer instructions could be dispatched to the queue in a single cycle (but executed only one at a time).

The 601 couples the integer unit more closely to the dispatch unit by eliminating the queue. All exceptions are kept precise with respect to integer instructions. Sequence tags are attached to integer instructions or to the holes left in the instruction queue by dispatched integer instructions. Exceptions or modifications of machine state beyond the currently executing integer instruction are not allowed to take effect until the current integer instruction completes.

The PowerPC architecture has load/store string (byte data) and load/store multiple (32-bit word data) instructions. These operations are executed by hardware in the integer unit, even though the PowerPC architecture permits these instructions to trap under certain circumstances. Integer multiply and divide are executed by bit-serial hardware in the integer unit (instead of using the floating-point hardware as is done in some processors). Integer multiply takes 5 to 9 cycles (data dependent), and integer divide takes 34 cycles.

The most unusual aspect of the 601 (for a RISC processor) is that it has support for most unaligned data accesses. Loading and storing arbitrarily aligned half-word, word, and double-word data is performed transparently by hardware (except when the access crosses a page boundary). Most other RISC processors either ignore low-order address bits (thereby forcing alignment) or trap on unaligned memory references. Since Macintosh software is filled with unaligned memory references (due to the leniency of 68000-family microprocessors), having unaligned-access support in the 601 eases emulation of existing Macintosh applications.

Floating-Point Unit

As with all modern processors, the floating-point unit is IEEE-754 compatible and implements single- and double-precision. The 88110 includes hardware support for IEEE extended precision only because Apple demanded it; Apple's SANE math package makes extended-precision part of the Macintosh programming interface. Since the Motorola designers probably resisted including the costly extended-precision hardware, it is ironic that Apple will be using the 601 without extended-precision support, and that the 88110—the microprocessor tailor-made for a RISC-based Macintosh—will have limited use due to lack of support from Apple.

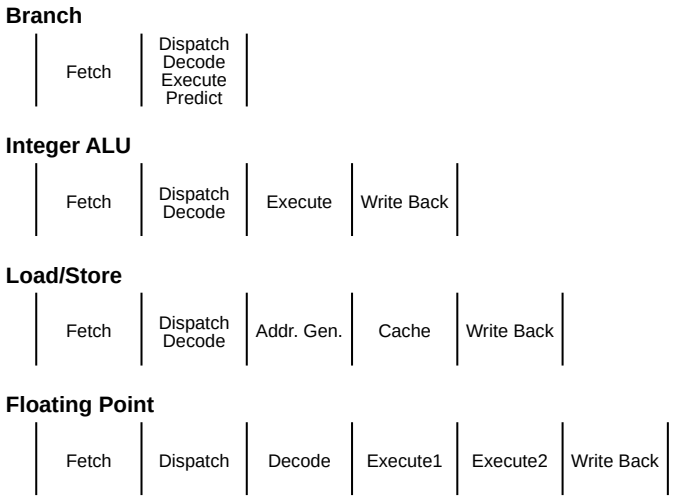


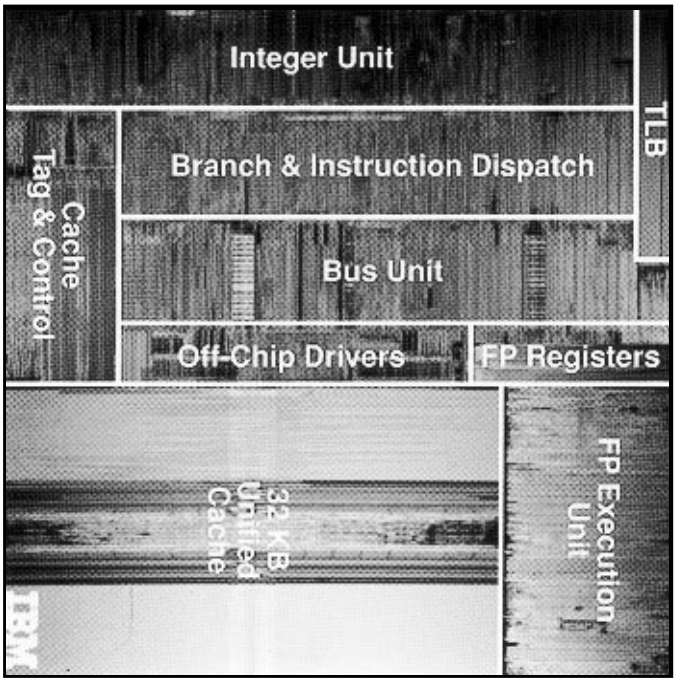
Figure 2. PowerPC 601 execution-unit pipelines.

The floating-point unit implements the hardware needed for the basic arithmetic operations and the composite multiply-add-fused (MAF) instructions. (MAF performs a multiplication followed by an add, but with only one rounding step instead of two as in most processors.) Table 2 lists the 601 floating-point performance characteristics. A new operation can be started each cycle for all operations except division and double-precision operations involving multiply, which have a two-cycle throughput.

As in the RIOS chip set, instruction flow from the dispatch unit to the floating-point execution hardware is buffered by a two-instruction queue. This allows the dispatch logic to empty the main dispatch window of floating-point instructions so that subsequent integer and branch instructions can be found and dispatched.

As Figure 1 shows, the floating-point unit has 64-bit data paths to the cache. Floating-point loads and stores are executed by the integer unit.

Floating-point exceptions are normally imprecise (they can be signalled sometime after the exception actually occurs) to allow maximum speed, but a precise-exception mode is available for program debugging and algorithms that require it.



Die Photo of the PowerPC 601, which incorporates 2.8 million transistors on a 10.95-cm-square die.

MMU

The memory-management unit contains an unusually large 256-entry, two-way set-associative TLB. Most microprocessors have 32- or 64-entry TLBs, but with a fully-associative organization; perhaps the large size was needed to make up for the relatively poor hit rate of the two-way organization. Hardware miss handling updates the TLB on a miss.

The PowerPC architecture specifies virtual address capabilities in two levels: a 32-bit and a 64-bit level. A PowerPC implementation can conform to either level depending on its intended application. The 64-bit architecture level calls for a true, 64-bit linear address capability and instruction-set extensions similar to that implemented by the R4000, Alpha, and the SPARC version-9 architectures.

The 601 conforms to the 32-bit linear address level but extends virtual addresses with a 52-bit, segmented address space. Virtual address translation begins by using the top four bits of the 32-bit linear address to index into a table of sixteen 24-bit segment identifiers. The 24-bit segment ID and the 28-bit segment offset are concatenated to form the 52-bit virtual address. (As far as programmers are concerned, this is still a 32-bit linear address space.) The MMU translates this virtual address to a 32-bit physical address. The page size is 4 Kbytes.

The MMU also provides block address translation capability separate from the TLB hardware. Four block- address-translation registers can map areas from 128K to 8M each. These would typically be used to map the operating system and a video frame buffer. In addition, a single 256M mapping entry is intended to be used for a memory-mapped I/O space.

601 FP Characteristic	FP add	FP mult	FP div	FP MAF
SP latency	4	4	17	4
SP throughput	1	1	15	1
DP latency	4	4	31	5
DP throughput	1	2	29	2

Table 2. PowerPC 601 floating-point latencies and throughputs.

Cache

The 601's 32K on-chip cache is one of the largest yet implemented in a single-chip microprocessor. Intel's i860XP equals it with separate 16K caches, and TI's SuperSPARC just beats it with a total of 36K (20K instruction, 16K data). The forthcoming R4000A will match the total cache size, but Intel's P5 will have only half as much.

The cache is write-back and uses an eight-way set-associative organization. The line size is 64 bytes with two 32-byte sectors per line. For multiprocessor systems, the cache implements a MESI coherency protocol. To improve data integrity, the cache maintains byte parity. The cache's large size combined with its high degree of set associativity should yield a high hit rate.

The cache is indexed with the physical address from the MMU, and the tags store physical addresses. Most other processors use a scheme that sends the upper part of the virtual address to the TLB and the lower part to the cache simultaneously so that TLB and cache lookup can proceed in parallel. This technique was not necessary for the 601.

A cache with physical tags usually has a higher hit rate than a cache with virtual tags because the operating system usually does not have to flush the cache on a process switch. A virtual cache can add process identifiers to reduce cache flush frequency, but this increases the size of each tag. Eliminating cache flushes is more important for large caches than for small ones.

The cache can perform a read-modify-write every cycle, which means it can handle a store instruction on each cycle even when the operand is misaligned. The cache always reads the entire cache line along with the cache tags. As the tag is being compared to detect a hit, the data to be written is "spliced" into place in a buffer. Late in the cycle, if a hit is detected, the entire line is written back into the cache, thus effecting the store; if a miss is detected, the write into the cache is simply disabled.

This strategy works only if the operand lies entirely within the line. When the operand straddles a line boundary, two read or read-modify-write operations must be performed, requiring an extra cycle.

The cache is isolated from the bus interface by two memory queues. The load queue has two entries to buffer addresses for pending read requests, while the store queue has enough capacity to buffer three 32-byte cache sectors of dirty data waiting to be written back to memory. Coherency between the cache, the memory queues, and the bus is maintained in hardware by snooping the write queue.

On a cache miss caused by a load, the cache normally requests only the 32-byte sector (half-line) needed to read the requested data. If the bus is idle, the memory queue will automatically create a read request for the other sector in the line, thus implementing a degree of prefetch for instructions and data. The PowerPC "touch" instruction performs a similar prefetch operation, so the logic and sequencing to perform this operation were already present.

Branch Unit

Although much of the 601 core is copied from the RSC, the branch unit has been completely redesigned. A dedicated branch offset adder is now included to allow the branch unit to execute all branch instructions without assistance from the integer unit. A dedicated branch adder is common in high-performance processors; it is curious that the RSC omitted it.

Static branch prediction is used to allow branches to be speculatively executed. The predicted direction is taken for backward branches and not-taken for forward branches.

To improve performance on a given program, the results of profiling can be used by the compiler to change the built-in prediction behavior. When profiling shows the default prediction direction is incorrect more than 50% of the time, the "reversal" bit can be set in the branch instruction. This is a clever and inexpensive technique for tuning branch prediction, but dynamic techniques, such as the branch target buffer in the P5, are more adaptable and offer the potential for better performance. Of course, future PowerPC implementations may also use dynamic techniques.

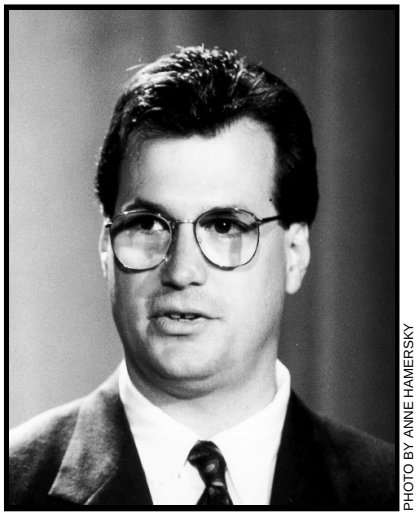


PHOTO BY ANNE HAMERSKY

"We wanted to establish the PowerPC architecture in the marketplace as early as possible. As a result, we put the 601 on an aggressive 12-month development schedule.... From IBM, we took the RSC... and substantially enhanced the internal structure to decrease the CPI and achieve our overall performance goals. From Motorola, we took the 88110 bus interface, with some enhancements, and developed the 601 bus interface."

Chuck Moore, IBM

Branch prediction on the 601 can be quite aggressive. For example, a compare followed by a conditional branch that uses the result of the compare can be issued simultaneously. The branch will be predicted and a fetch of the target stream from the cache will be initiated. When the result of the compare is available, it is sent directly to the instruction buffer (instead of to the branch unit) where it will cause the buffer to either accept or ignore the target stream. In case the prediction is wrong, the previous program counter address is saved in a register so that sequential fetching can be resumed.

The 601 has only one main TLB for storing both instruction and data address translations. To prevent this central resource from becoming a bottle neck, the branch unit implements an instruction μTLB. A μTLB is an inexpensive alternative to a full-sized TLB, and it typically gives good performance because instruction accesses tend to be highly local. μTLBs are used in the early MIPS processors, such as the the R3000, and the microSPARC. The 601's four-entry μTLB improves on the RSC's single-entry design.

One peculiarity of the POWER and PowerPC architectures is the link register. The link register is where the subroutine call instructions save their return address and where indirect jump (subroutine return) instructions get their target address. (This is no doubt an artifact of the original multi-chip RIOS implementation; now, it looks like an architectural flaw.) In contrast, most architectures save return addresses directly in a general-purpose register.

Since a subroutine call can be a conditional branch, it can be predicted. To allow the call to be "undone" in the case of a mis-prediction, the link register is backed up by a two-entry stack to allow the previous address to be restored.

Instruction Dispatch Unit

The 601 is a modestly superscalar design along the lines of hyperSPARC and Alpha: it allows multiple dispatch and execution using the traditional execution units found in most processors. It does not, however, allow two integer operations to be dispatched and executed in the same cycle.

The instruction dispatch unit contains an eight-

Microprocessor	SPECint89	SPECint89	SPECint92	SPECfp92
PowerPC 601 (50 MHz)	40*	60*	–	–
PowerPC 601 (66 MHz)	50*	80*	–	–
R4000PC (50/100 MHz)	30.0	44.5	36.8	40.0
microSPARC (50 MHz)	–	–	22.8	18.4
SuperSPARC (36 MHz)	–	–	44.2	52.9
P5 (66 MHz, 256KB L-2 cache)	–	–	60**	55**
486DX2 (66 MHz, 256KB L-2 cache)	34.0	21.2	32.4	16.1

Table 3. The PowerPC 601 compares favorably to its competitors on the SPEC benchmarks. IBM declined to provide SPEC92 numbers. (*IBM estimates, **Microprocessor Report estimates)

entry instruction queue from which instructions are dispatched to the three execution units. The dispatch logic looks at the bottom four entries in the queue to determine which instructions will be dispatched next. Up to three instructions can be dispatched per cycle, one to each of the execution units. The queue is refilled from the cache in eight-instruction chunks via a wide, dedicated bus.

Instructions can be dispatched out of order so long as dependencies permit. For example, a branch in the fourth slot can be executed even if the first three instructions cannot be dispatched. Since exceptions are synchronized to the integer pipeline, some instructions that are issued out of order will not be allowed to complete until a following integer instruction has completed.

IBM claims the 601 has a 30% better clocks-per-instruction average than the RSC. Improvements in the fetch, dispatch, and branch processing logic account for about 11%; the increase in the cache size accounts for the other 19%.

Performance

Table 3 shows the performance estimates given by IBM for a 601 system with a memory system that requires 12-cycles for a random access and 3 cycles for sequential burst accesses (page-mode accesses without interleaved memory) with no second-level cache. Only the P5 and 486 numbers assume a second-level cache. The 601's integer performance is competitive, while the floating-point performance appears to be on par with its fastest competitors (although it is hard to compare SPEC89 and SPEC92 ratings).

One of the considerations for Apple is the performance of the 601 for Macintosh emulation. Note that floating-point performance is virtually irrelevant for the majority of Macintosh applications and even less important for a 680x0 instruction set emulator, which will spend most of its time executing integer operations to fetch and decode 680x0 opcodes.

The superscalar dispatch of integer operations and branches will definitely be of value in a 680x0 emulator. Another plus for 680x0 emulation is the extensive set of PowerPC bit-field operations. These should help speed the extraction of instruction fields from 680x0 opcodes and extension words.

While the 32K on-chip cache is large by single-chip microprocessor standards, some of the fastest emulation techniques use large tables. Thus, depending on how Apple writes its 680x0 emulator, the 32K cache may have a low hit rate. If Apple adds an external cache to improve the performance of emulation, the estimates given in Table 3 might prove conservative.

Conclusions

The 601 is an extremely compact die given its level of integration. Considering only die size would lead to the conclusion that the 601 will be a relatively inexpensive chip. IBM's five-layer-metal CMOS process is, however, complex by the standards of other manufacturers, where triple-level or even dual-level metal is the state of the art. The complexity of the process could offset the yield advantage of the small die.

For Apple's main purpose—migrating the Macintosh application base to a RISC processor—the 601 is a good match. Apple plans to put the 601 in an under-$3000 machine; other high-end processors, including the P5, are probably going into more expensive machines. Macintosh emulation would benefit greatly, however, from the ability to dual-issue integer instructions, and the 88110 would have provided this capability.

If the Apple/IBM/Motorola alliance can achieve its goal of at least sampling all four planned family members by the end of next year, PowerPC will quickly match many competing architectures in terms of implementation breadth. In any case, the 601 will, for the first time, give Apple a personal computer able to compete with—and even beat—some workstations. If Apple can convince application developers to write native applications, the first PowerPC Macintosh could redefine the standard of personal computer performance—that is, unless the P5 does it first.

Some designers at IBM feel that the computer giant has a negative reputation for producing large, costly chips behind schedule, and they wanted the 601 to dispel that impression. The on-time delivery and impressive density of the 601 should go a long way toward achieving that goal. Now the test is to see if IBM/Motorola can deliver the quantities that Apple requires at a price the personal computer market can afford. ♦

