PowerPC Processors

Anthony Marsala
Basel Kanawati

IBM
PowerPC Embedded Processor Solutions

Abstract

In October of 1991, IBM, Apple, and Motorola formed an alliance to produce a new microprocessor family named the PowerPC. Less than two years later, the alliance has produced the first chip of the PowerPC family, the MPC601. This paper will analyze the PowerPC 6xx family, with an emphasis on the architecture of the MPC601 chip. The 601's Reduced Instruction Set Computer (RISC) architecture will be examined, including the three parallel execution units (integer unit, branch unit, and floating point unit), the instruction unit, the memory management unit, and the system interface. In addition, the software environments for the PowerPC will be discussed. Finally, conclusions about the chip will be drawn and an outlook on the future of the PowerPC family will be made.

Overview

The PowerPC is a follow-on to IBM's RISC chip found in its RS/6000 workstation line. Although the workstation line has been successful, the UNIX market has not experienced the same level of market dominance that the personal-computer market has enjoyed. Most personal-computers today contain an Intel produced 80X86/Pentium processor - about 85%
of personal computers use Intel chips. In an effort to break Intel's hold on the market, IBM has formed an alliance with Apple and Motorola to produce the PowerPC.

IBM has brought its record making RISC architecture to the alliance. Apple's contribution will be more noticed when Taligent's (a joint owned Apple-IBM company) Pink operating system is unveiled to run on the PowerPC. Motorola's chip designers and new submicron chip fabrication plant complete the powerful alliance. By jointly developing the processors at the Somerset Design Center in Austin, Texas, the alliance has been able to share the cost and promote what they hope will be an industry standard in the coming years, displacing Intel's stranglehold on the PC microprocessor market.

Features

There are significant differences between the RISC based 601 chip and the Pentium chip from Intel. The main difference is in its fundamental architecture. The 601 is RISC based, meaning that there are a relatively few number of pipelined instructions to achieve a high throughput on each clock pulse. The Pentium, on the other hand, is a Complex Instruction Set Computer (CISC) architecture. This means that more instructions exist so that the chip may be programmed at a higher level, but each individual instruction typically takes multiple clock cycles to complete. The 80486 and Pentium processors have begun incorporating more RISC type features, but the added support is still there for the older chips in the series.

The size of the 601 is only 11mm x 11mm, fitting in a total of 2.8 million transistors. Pentium, on the other hand, has 3.1 million transistors on a die size of 16.6mm x 17.6mm. The smaller size of the 601 is mainly a result of a 0.65 micron four-layer CMOS process, as opposed to the 0.8 micron BiCMOS process that Intel used. The smaller size has helped sell the 601 for $450, as opposed to approximately $900 for the Pentium. Furthermore, the smaller size and CMOS process has resulted in a smaller power dissipation, only 9 Watts for a 66 MHz 601 as opposed to 16 Watts for the same speed Pentium. This should result in lower system design cost, as the Pentium systems have received a lot of press because of the need for extra cooling fans.

Performance benchmarks on the 601 to date show a one and a half to five times increase over the Pentium, depending on the kind of operation.

PowerPC 6xx Family

The MPC601 is the first of a series of PowerPC chips announced by the alliance. Next year, PowerPC 603 systems should be available. The 603 is a low-power version of the 601 aimed at the notebook computer market, and will contain extra power-saving features. The workstation follow-on to the 601 is the 604. The 604 will have a bigger pipeline and higher parallelism. It will also have more advanced branching techniques to improve system performance. The 604 systems should start appearing by late 1994. The top-of-the-line PowerPC chip will be the 620, and will contain both 64 bit address and data paths. It will also use a higher-throughput processor bus, and contain extra hooks for multiple-chip configurations. Systems with this chip are expected to appear in early 1995.

In addition to the family of processors to be produced by the alliance, both IBM and Motorola have announced independent plans to produce embedded-controllers versions of the PowerPC. Such controllers may have no floating point unit, a small ASIC core with room for application-specific silicon, and increased exception handling routines. These products should be introduced in mid-1994.

MPC601 Architecture

The 601 is a follow-on to the IBM Power Optimized With Enhanced RISC (POWER) architecture found in its workstation line. Because it is a RISC processor, it contains fixed length 32-bit wide instructions and three parallel execution units. The three execution units are the Branch Processing Unit (BPU), Integer Unit (IU) and Floating-Point Unit (FPU). Instructions are dispatched to the different execution units via an Instruction Unit, which can queue up to eight instructions and has a dedicated adder for prefetching. Instructions and data are fetched from a 32kB unified eight-way, set-associative cache. This cache uses a least-recently used (LRU) replacement algorithm. Memory management is performed by the Memory Management Unit, which has inside it a 256 entry, two-way set associative unified translation lookaside buffer (UTLB) for virtual to physical memory mapping. The MMU supports demand paging for 4kB pages and supports block addressing for block sizes ranging from 123kB to 8MB. To connect to the world, the 601 has a system bus based on the Motorola 88110 processor bus. This bus has arbitration lines to support multiple bus masters and the 601 can run at an integer multiple of the bus frequency. The MPU601 has a 64 bit data bus and a 32 bit address bus, and contains 32 general-purpose registers (GPRs) and 32 floating-point registers (FPRs). See Figure 1 for an overall block diagram of the MPC601.

Block Diagram of MPU601 PowerPC Microprocessor

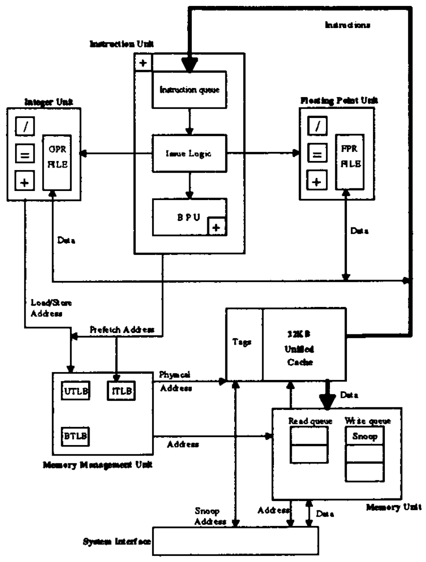

Figure 1 Diagram from Reference [4]

Instruction Unit

The purpose of the Instruction Unit is to provide control for the 601's three independent execution units (See Figure 2). To accomplish this, the instruction unit has an instruction queue which holds up to eight instructions. Up to eight instructions can be fetched from the cache and filled in the queue at once. Assuming that all instructions are in the cache, all fetches will complete in one cycle. Once in the queue, instructions move (are shifted) from the upper half (Q7-Q4) to the lower half (Q3-Q0). The upper half of the queue are merely holding places, while the lower half are used to issue instructions. The entire lower half of the queue is searched for issuing to the BPU and FPU, while Q0 is used for issuing instructions to the integer unit. Since there are three instruction units which can be executing instructions at the same time, the 601 is able to perform up to three instructions in parallel.

Once instructions are issued, a sequential fetcher is used to get more instructions for the instruction queue. The sequential fetcher is possible due to a dedicated adder within the instruction unit. The instruction unit tries to keep the instruction queue filled at all times. The BPU uses static branch prediction to prefetch instructions for the instruction queue. Although instructions can be issued to the IU and FPU while a possible branch exists in the BPU, the instructions will stop at the register write stage of their pipeline and not perform the write-back stage. Using the static branch prediction, execution tries to proceed normally. On unconditional branches, the branch is removed from the instruction stream. This is known as branch folding.

If the branch prediction was correct, normal operation continues with no cycle delays. In the event that a branch prediction was incorrect, the queue will be flushed and instructions will be fetched from the correct location.

Because the instruction unit can issue instructions out of order to the branch processing unit and floating point unit, the strict deterministic ordering of instructions is loosened and higher throughput is achieved.

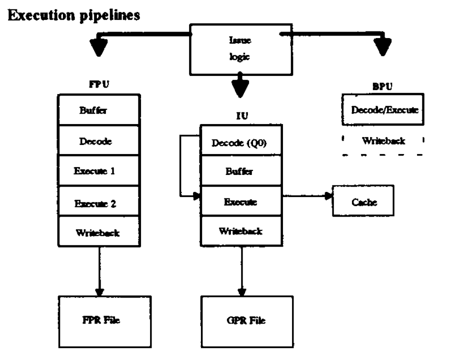

Figure 2 Figure taken from Reference [4]

In addition, the instruction unit performs feed forwarding. In feed forwarding, an instruction which has the same register output as the input to the next instruction is routed automatically to the next instruction without performing the usually required extra read. The reduces the number of clock cycles an execution unit must wait to use a register.

Branch Processing Unit (BPU)

The branch processing unit tries to prevent the delays normally incurred by branches. Typically, when a branch occurs, extra cycles are needed because the new target address is not in the instruction queue and quite possibly not in the cache. The branch processing unit tries to anticipate branches and correctly predict conditional branches (ones that must wait for the condition to be calculated) so that prefetching can be done at the new target address. The instruction queue will then remain filled and execution pipeline units will keep busy. Although the BPU is an independent execution unit, it is logically a part of the instruction unit since its goal is to keep the instruction queue full and other execution units busy.

For conditional branches, the 601 uses a bit in its instruction coding to specify how to predict the branch. The default operation is that backwards branches (previous instructions) are predicted taken and forward branches (instructions not yet reached) are predicted untaken. This static approach (not condition dependent) is in contrast to the dynamic method of other popular processors such as the Pentium, which predicts branches based on the history of branches taken. The static approach is also simpler to implement, thereby saving real-estate on the die as well as design time, power, etc.

There are two rules the BPU must adhere to for conditional branches in order to ensure proper sequential program execution. First, no instruction following a branch can write results to a register. And secondly, no additional branch instructions can be issued to the BPU while a branch prediction is in process.

The BPU uses a dedicated adder and three special-purposes registers for branch prediction - the link register (LR), count register (CTR), and control register (CR). By using special-purpose registers instead of general-purpose registers, the BPU's execution is not dependent on the availability of the GPRs and hence can operate independent of the FPU and IU.

The branch unit is essentially a one-stage pipeline. Unconditional branches are folded as described previously. For conditional branches, a prefetch occurs at the predicted target address. Current instructions subsequent to the branch are stopped; when new prefetched instructions arrive, the stopped instructions are discarded. If, however, the branch condition is evaluated and the branch prediction is found to be incorrect *before* the prefetched instructions arrive, the stopped instructions will restart and the prefetched instructions will be discarded.

Although there is a penalty for wrongly predicted branches, the rewards of a zero-delay branch prediction outweighs the cost.

Integer Unit (IU)

The integer unit executes arithmetic and logical instructions as well as memory access instructions (including floating point memory access instructions). It contains an arithmetic logic unit (ALU), an integer multiplier, divider, and the general-purpose register file with 32 registers. The register file contains two write-back ports so that two accesses can be done in parallel (eg. from the cache and integer execute unit). It also contains the integer exception register (XER).

To access information, the integer unit has interfaces to both the cache and memory management unit. All load and store instructions are issued and translated in program order, although accesses can occur out of order. To calculate an address, one source register (or zero) is added to either a second source register or a 16 bit immediate field imbedded within the instruction. One instruction can be issued to the IU each clock cycle.

The ALU performs integer adds and subtracts, as well as the standard logical functions such as compare, rotate, and shift. The MPC601 also contains additional POWER rotate and shift instructions for additional compatibility that will not exist in future PowerPC implementations.

The IU is implemented as a three stage pipeline, which may be expanded to four. The bottom of the instruction unit queue (Q0) is decoded and issued to the execute stage if the stage is empty. If the execute stage is not empty, the instruction is passed to a buffer stage (the possible fourth stage) until the execute stage becomes available. The buffer stage tries to overcome stalls in the IU pipeline due to multicycle operations in the execute stage. After the instruction completes the execute stage, results enter the write back stage, where integer instructions are written to the GPR file.

Floating Point Unit (FPU)

The PowerPC floating point unit contains a single precision multiply array, double precision add array, a divider, 32 floating point registers (FPRs), and a floating point status and control register. It is used to operate on all IEEE 754 floating point data types (not a number [NaN], normalized, denormalized, zero, and infinity). By having the floating point operations in hardware, the 601 is able to avoid the software latency of exception routines to handle floating point operations. This is similar to the Intel 486, which for the first time put its floating point unit on chip instead of creating a 487 math coprocessor as the company had done with past products.

The floating point unit is broken up into five pipelined stages: buffer stage, decode stage, execute1 stage, execute2 stage, and writeback stage. Unlike the IU, the floating point unit does not have feed forwarding, so instructions which depend on data from a previous instruction must wait until the previous instruction performs its writeback to registers. The FPU pipeline makes it possible to execute most single precision and many double precision floating point operations in a single clock cycle.

In contrast to the IU, all FPU instructions must spend at least one cycle in the buffer stage. And like the BPU, the floating point unit can access the bottom half of the instruction queue (Q3-Q0), thereby reducing bottlenecks and allowing non-data dependent instructions to be executed out of order. And like the GPR file of the IU, the FPR's are dual ported to allow access by the cache and the writeback stage of the FPU pipeline in the same cycle.

Memory Management Unit (MMU)

The memory management unit is responsible for both translating virtual to physical addresses and for enforcing protection with regard to supervisor and user privileges. See Figure 3 for a block diagram of the MMU.

The MPC601 is able to address up to 4 terabytes of virtual memory (2^{52}) and 4 gigabytes of main memory (2^{32}). The instruction unit generates instruction addresses and the integer unit generates data accesses. When the memory management unit needs to perform a memory access, it issues a virtual address. This virtual address is then translated and the physical address passed to the cache. In the event of the cache being inhibited or a cache miss, the 32 bit physical address is then passed to the system interface for retrieval.

MMU Operations

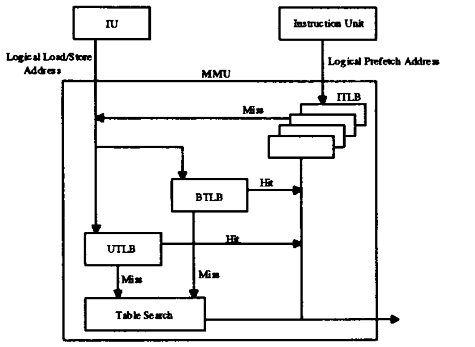

Figure 3 Figure from Reference [4]

When the MMU passes the address to the cache, the lower bits from the virtual address are the same as the physical address; these bits go directly to the cache as the index into the tag array. To perform the translation of the upper address bits, the MMU may access up to three of its translation lookaside buffers (TLB). Each of the three TLBs utilize a least recently used (LRU) algorithm for keeping storing information. The MMU keeps the last four instruction accesses in a fully associative four-entry instruction translation lookaside buffer (ITLB). The ITLB capitalizes on the fact of locality of reference for sequential programs. It also reduces contention for the MMU between instruction and data accesses.

If the ITLB lookup is a miss, it will check the 256 entry two-way set associative UTLB (unified data and instruction TLB) and four-entry BTLB (block TLB) . These lookups will guarantee a hit for most address translations, and the corresponding upper physical address bits will be sent to the cache as the cache tag. When all TLB's produce a miss, the MMU must access page tables in memory to perform the necessary virtual to real address translation.

By using a physical address to access the cache instead of a virtual address, the MPC601 is able to avoid the update and coherency problem of having two virtual addresses that map to the same physical address. And the extra translation time needed to map from virtual to physical address is not a problem because the translation is performed during the execute stage of IU; no extra clock cycles are needed.

Block addressing is available for applications that require contiguous memory space (such as a read-only memory [ROM]; block sizes can range from 128kB to 8MB.

Cache Unit

Much has already been said about the cache unit (Figure 4).

Cache Interface

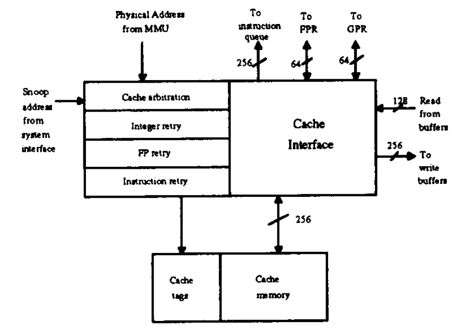

Figure 4 Figure from Reference [4]

It is a 32kB, eight-way set associative cache and uses a LRU replacement algorithm. It is unified, that is, both data and instructions reside in the same cache, thereby allowing for a simple implementation. Cache lines are 64 bytes, divided up into two, eight-word (32 bit) sectors. Each sector can be snooped, loaded, cast-out, or invalidated.

Although the cache is designed to use a write-back policy, the ability to cache, the write policy, and the memory coherency are controllable at the page and block level. Write operations can be performed on data sizes as small as a byte, and a read-modified-write can occur in as little as one cycle.

When the instruction unit requests the next instruction, the cache will attempt to fill up the instruction queue with as many as eight instructions (the full size of the queue). Therefore, if the queue is empty, an entire sector can be loaded into the queue in parallel.

The cache has a dedicated port for instruction fetch and load/stores as well as a dedicated port for snooping operations. By having a dedicated port for snooping, it does not require additional clock cycles to perform a snoop unless the snooping operation results in a cache hit. A snooping cache hit would result in extra cycles needed to perform the snoop.

The cache has its own retry queues to buffer floating point, integer, and instruction fetch accesses, and arbitration is performed to determine the highest priority cache requests. The priority for cache request is cache reloads (highest), snoops, floating point stores, integer accesses, and instruction fetches.

By supporting snooping and MESI protocols, cache coherency can be achieved in high performance multiprocessor applications.

Memory Unit

The memory unit is used to buffer operations between the cache and the system interface. Operations which depend on the memory unit are load and store cache misses, cache replacement operations, and page table search operations (TLB misses). The memory unit employs a two element read queue and a three element write queue to perform the necessary buffering.

Each element in the queue is eight words, or one cache line of data. The write queue has an element dedicated to snooping operations to guarantee a high priority for bus snoops.

Although all loads and stores are issued and translated in program order, ones that hit in the cache are likely to complete before ones that miss, therefore loads and stores can complete out of strict deterministic order. The MPC601 does address checking for translated loads and stores and prevents out of order loads and stores which have matching addresses. It is also possible to force strict program order of loads and stores by using the synchronization instructions provided by the 601.

System Interface

The system interface is the MPC601's connection from the memory unit to the outside world. The interface consists of a 64 bit data bus, a 32 bit address bus , and bus arbitration and other control signals. The system interface implemented by the 601 is based on the system interface of Motorola's 88110 RISC processor. Because of the 601's large on-chip write-back policy cache, most of the system operations are burst-memory reads followed by burst-memory writes (cache writebacks). Other operations include I/O operations, non-cache memory accesses, and snooping operations.

Address arbitration is provided to allow memory accesses in multiprocessor configurations. The system is flexible enough to allow different external arbitrations mechanisms to exist. The MPC601 also has additional multiprocessor support through the external control of the on-chip cache and the TLB, as well as expandability to a second external cache.

The MPC601 is somewhat unlike other RISC chips in that it allows for misaligned loads and stores.

Software Environments

Managing and controlling the software development life cycle depends greatly on the tools set used.

Operating Systems

In today's market, having a fast chip is not enough to be successful. There must also be a large number of applications available. Therefore, the operating systems that the PowerPC will run becomes crucial. As IBM learned with the introduction of OS/2 1.0, users are not very willing to jump to a new hardware platform if there are few applications for the new platform (and the current platform does not provide backward compatibility with existing applications!)

Currently,seven operating systems are being ported to run on the PowerPC: Apple's System 7, IBM's AIX and OS/2 operating systems as well as their embedded operating sytem OS Open, Sun's Solaris, PowerOpen, and Pink - the object-oriented operating system under development by Taligent. There has also been discussion about Novell's Netware, UNIX System V, and Windows NT being under development.

PowerOpen is probably the most interesting of the six operating systems. It is a standard of application programming interfaces (APIs) and application binary interfaces (ABIs) developed by the PowerOpen Association, a consortium of companies supporting the PowerPC. PowerOpen will make it possible for different user interfaces (Windows, UNIX, Mac OS) and applications to coexist on the same PowerPC system by providing a standard set of APIs. The common APIs will allow application software to have hardware independent calling routines.

Additionally, Insignia Solutions, which currently provides emulation technology for Windows application to run under Windows NT, is working on a port of its SoftPC X86 and Macintosh emulation technology to run natively on a PowerPC chip.

These and other emulation companies expect to achieve 486 levels of performance of X86 software under emulation on a PowerPC.

With such a choice of operating systems and user-interfaces that will be available on the PowerPC platform, a large body of applications will exist to entice users to jump to the new hardware.

But the real "power" of the PowerPC will show forth when applications are written to run natively, without emulation, to make the maximum use of the pipeline and independent parallel execution units and higher clock speed.

Development Tools

In addition to operating systems, there have been a number of announced tool developers that have announced plans for PowerPC support. The tools range from hardware debuggers to compilers to simulators to operating systems. IBM and Motorola have put together a catalog of development tools to advertise the support of third party vendors for these tools, which are essential for program development. Unlike the POWER architecture, which was limited to IBM machines, the PowerPC architecture will need new compilers, debuggers, assemblers, GUI-builders, and other tools to facilitate development by third-parties. Even IBM, who previously had treated its XLC compilers as company jewels, has announced plans to sell the compilers as packages to run on any PowerPC platform (i.e.. not an IBM box). The IBM AIX XLC Compiler features advanced optimization techniques which were refined for the IBM RISC/6000 workstation family. Although many of the techniques used in the RS/6000 C Compiler are well-known and have been used in other optimizing compilers, several new optimizations and many improvements on existing techniques have been added. This should allow consumers to pick and choose the hardware and software they desire like never before.

Conclusion

The PowerPC architecture combines advanced reduced instruction computing and pipelining techniques into a small, low-power CMOS chip. It has advanced pipelining, independent execution units which allow up to three instructions to be executed in parallel, while at the same time having simple static branching techniques and a relatively simple unified data and instruction cache. These simplified design techniques have left room for a relatively large cache (32k) to provide a high cache hit ratio. And because the chip was designed without pushing the fabrication technology to the limit, the PowerPC family of processors has room to grow and add higher performance features.

The 64 bit data path, the multiprocessing support with cache snooping, and the 52 bit virtual address bus show that the MPC601 is ready to handle the future of computing technology. At the same time, the large number of operating systems and emulation tools being ported to the PowerPC will provide a needed application base for user acceptance.

Although it will be difficult for the IBM/Apple/Motorola alliance to wrestle the market lead from today's 486/Pentium Intel systems, the low cost, high performance, and room for increased performance of the MPC601 will have an impact in the computer industry for years to come.

Bibliography

1. IBM & Motorola, PowerPC Tools Catalog, 1993.

2. Steve Lohr, "Technology alliance seeks leading-edge chips", The Raleigh News and Observer, May 27, 1993.

3. Motorola Inc., PowerPC 601 RISC Microprocessor User's Manual, 1993.

4. Bob Ryan, "RISC Drives PowerPC", BYTE, August 1993.

5. Tom Thompson, "PowerPC Performs for Less", BYTE, August 1993.

6. Loring Wirbel, "PowerPC on fast track", Electronic Engineering Times, May 11, 1992.