The PowerPC 405™ Core

IBM Microelectronics Division
Research Triangle Park, NC 27709

11/2/98

Overview

The PowerPC 405 CPU Core is a new addition to the 32-bit RISC PowerPC Embedded Processor family. The 405 Core possesses all of the qualities necessary to make system-on-a-chip designs a reality. This core occupies a small die area, consumes minimal power, and provides a high performance 100% PowerPC compatible platform capable of taming the most demanding embedded applications.

Target Applications

The 405 Core enables high performance designs in which low cost, low power and versatility are the critical selection criteria.

Target market segments for the 405 core include:

- Consumer video applications including digital cameras, video games, set-top boxes, and soft modems
- Portable products such as cellular phones, PDAs, and handheld GPS receivers
- Office automation products such as printer, X-terminals, and fax machines
- Networking and storage products such as disk drive controllers, routers, ATM switches, high performance modems, and network cards

Typical Application

A typical system on a chip design with the 405 Core uses a two level bus structure for system level communication. High bandwidth peripherals and the 405 Core communicate with one another over the processor local bus (PLB). Less demanding peripherals share the on-chip peripheral bus (OPB) and communicate to the PLB through the OPB Bridge. The PLB and OPB provide common interfaces for peripherals and enable quick turnaround, custom solutions for high volume applications.

Figure 2 shows an example 405 Core-based system on a chip, illustrating the two-level bus structure and modular core-based design.

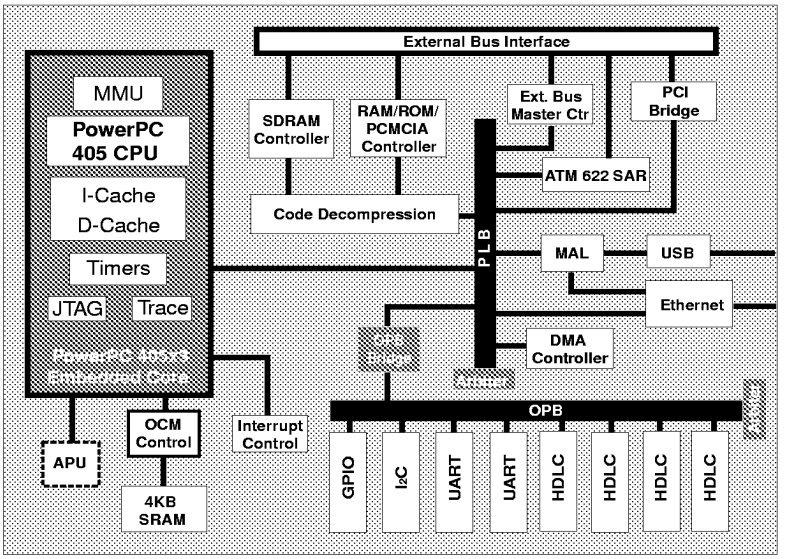

Figure 2. Example 405x3 core+ASIC

Specifications

- Technology: 0.25 µm CMOS process (0.18 µm L_{eff})
- Frequency: 0-200MHz (200MHz at WC process, 85C, 2.3V)
  - 276MHz nominal
- Performance: 228 Dhrystone 2.1 MIPS@200MHz (est.)
- Supply Voltage: 2.5V
- Die Size: 2.0 mm^2 for CPU only (est.)
- Power (typ.): 400mW @ 200MHz (est.), CPU only
- 3.04 mW/MHz with 16KB ICU/8KB DCU (est.)
- Qualified operating range –40C to 125C, 2.3V to 2.7V

Embedded Design Support

The 405 Core, as a member of the PowerPC 400 Series, is supported by the IBM PowerPC Embedded Tools™ program, in which over 80 third party vendors have combined with IBM to provide a complete tools solution. Development tools for the 405 Core include C/C++ compilers, debuggers, bus functional models and real-time operating systems. As part of the tools program, IBM maintains a complete set of development tools by offering the High C/C++ Compiler, RISCWatch™ debugger with RISCTrace™, OS Open™ real-time operating system, VHDL and Verilog simulation models and a 405 Core reference design kit.

405 CPU Core Organization

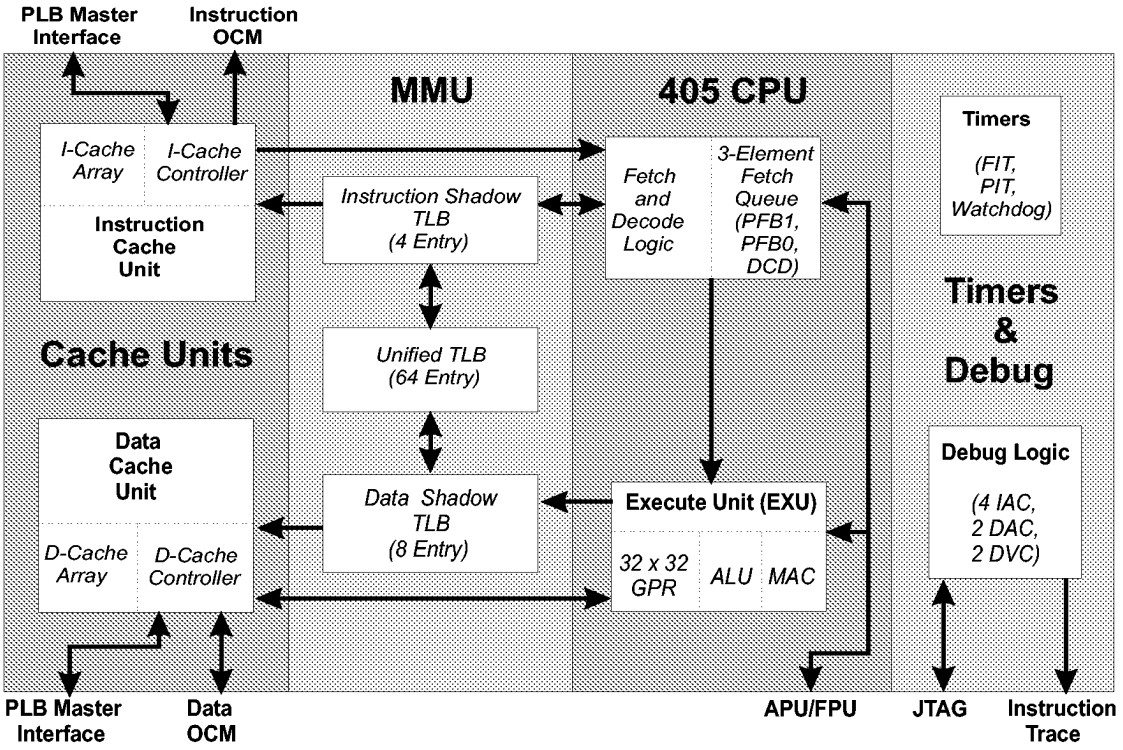

Figure 1. PowerPC 405 CPU core block diagram

405 CPU

The 405 CPU operates on instructions in a five stage pipeline consisting of a fetch, decode, execute, write-back, and load write-back stage. A five-stage pipeline increases throughput and makes operating at 200 MHz or more possible. Figure 1 contains a high-level block diagram of the 405 CPU core.

Fetch and Decode Logic

The fetch and decode logic maintains a steady flow of instructions to the execute unit by placing up to two instructions in the fetch queue. The fetch queue consists of three buffers: pre-fetch buffer 1 (PFB1), pre-fetch buffer 0 (PFB0) and decode (DCD). When the queue is empty, instructions flow directly into the decode stage.

Static branch prediction as implemented on the 405 CPU core takes advantage of some standard statistical properties of code. Branches with negative address displacement are by default assumed taken. Branches that do not test the condition or count registers are also predicted as taken. The 405 CPU core bases branch prediction upon these default conditions when a branch is not resolved and speculatively fetches along the predicted path. The default prediction can be overridden by software at assembly or compile time.

Branches are examined in the decode and pre-fetch buffer 0 fetch queue stages. Two branch instructions can be handled simultaneously. If the branch in decode is not taken, the fetch logic fetches along the predicted path of the branch instruction in pre-fetch buffer 0. If the branch in decode is taken, the fetch logic ignores the branch instruction in pre-fetch buffer 0.

Execution Unit

The 405 CPU core has a single issue execute unit which contains the register file, arithmetic logic unit (ALU) and the multiply-accumulate (MAC) unit.. The execution unit performs all 32-bit PowerPC integer instructions in hardware and is compliant with the IBM PowerPC Embedded Environment specification. Table 1 lists instruction latencies and throughputs.

The register file is comprised of thirty-two 32 bit general purpose registers (GPR) which are accessed with three read ports and two write ports. During the decode stage, data is read out of the GPR's and feed to the execute unit. Likewise, during the write-back stage, results are written to the GPR. The use of the five port on the register file enables either a load or a store operation to execute in parallel with an ALU operation.

Instruction	Latency (Clock Cycles)	Throughput (Clock Cycles)
Arithmetic	1	-
Traps	2	-
Logical	1	-
Shifts/rotates	1	-
Multiply two sign extended 16 bit half words in 32 bit registers (16 x 16 → 32)	2	-
Multiply half word (16 x 16 → 32)	2	1[a]
Multiply one sign extended 16 bit half word in a 32 bit register with a 32 bit word (16 x 32 → least significant 32 bits of a 48 bit product)	3	2[a]
Multiply (32 x 32 → least significant 32 bits of a 64 bit product)	4	3[a]
Multiply (32 x 32 → least significant 32 bits of a 64 bit product) with overflow detection	5	4[a]
Multiply (32 x 32 → most significant 32 bits of 64 bit product) with or without overflow detection enabled	5	4[a]
MAC (16 x 16 + 32 → 32)	2	1[a]
Divide	35	-
Loads[b]	1[b]	1[b]
Load multiples/strings (caches hit)	1/transfer	-
Stores	1	-
Store multiples/strings (caches hit)	1/transfer	-
Move to/from Device Control Register	3	-
Move to/from Special Purpose Register	1	-
Branch known taken	1[d] or 2[e]	-
Branch known not taken	1	-
Taken branch predicted taken[c]	1[d] or 2[e]	-
Taken branch predicted not taken[c]	2[e] or 3[f]	-
Not taken branch predicted taken[c]	2[e] or 3[f]	-
Not taken branch predicted not-taken[c]	1	-

Notes:
a. Throughput applies when no dependencies exist between the result of the multiplication and the source of the subsequent multiplication. However, the target accumulator from one MAC instruction may be used as the source accumulator for the next MAC instruction.
b. Load instructions have a 1 cycle use penalty. Data acquired during the execute pipeline stage is not available until after the completion of the write-back pipeline stage. Write-back takes one clock cycle.
c. These cycles are for conditional branches that have a dependency.
d. The branch is resolved during the fetch pipeline stage while in PFB0.
e. The branch is resolved during the decode pipeline stage while in DCD.
f. The branch is resolved during the execute pipeline stage.

Table 1. Instruction Latencies and Throughputs

The MAC unit is an auxiliary processor unit (APU) which adds 9 operations to the 405 CPU core instruction set. MAC instructions operate on either signed or unsigned 16 bit operands and accumulate the results in a 32 bit GPR. All MAC unit instructions have a single cycle throughput.

Table 2 lists the six MAC operations and three halfword multiply operations supported by the MAC unit. RA and RB in Table 2 are the 16 bit operands and RT is the 32-bit accumulate source and target. Most of the instructions have four flavors. Instructions may be unsigned, signed, modulo or saturate. Modulo and saturate determine the behavior of the instruction when the accumulated value overflows.

Description	Operation	Instruction mnemonic
multiply accumulate high halfword to word	$Prod_{0:31} \leftarrow (RA)_{0:15} \times (RB)_{0:15}$ $RT \leftarrow Prod_{0:31} + (RT)$	machhw – modulo signed machhwu - modulo unsigned machhws – saturate signed machhwsu - saturate unsigned
multiply accumulate low halfword to word	$Prod_{0:31} \leftarrow (RA)_{16:31} \times (RB)_{16:31}$ $RT \leftarrow Prod_{0:31} + (RT)$	maclhw - modulo signed maclhwu - modulo unsigned maclhws - saturate signed maclhwsu - saturate unsigned
multiply accumulate cross halfword to word	$Prod_{0:31} \leftarrow (RA)_{16:31} \times (RB)_{0:15}$ $RT \leftarrow Prod_{0:31} + (RT)$	macchw - modulo signed macchwu - modulo unsigned macchws - saturate signed macchwsu - saturate unsigned
negative multiply accumulate high halfword to word	$Prod_{0:31} \leftarrow (RA)_{0:15} \times (RB)_{0:15}$ $RT \leftarrow - Prod_{0:31} + (RT)$	nmachhw – modulo signed nmachhws – saturate signed
negative multiply accumulate low halfword to word	$Prod_{0:31} \leftarrow (RA)_{16:31} \times (RB)_{16:31}$ $RT \leftarrow - Prod_{0:31} + (RT)$	nmaclhw – modulo signed nmaclhws – saturate signed
negative multiply accumulate cross halfword to word	$Prod_{0:31} \leftarrow (RA)_{0:15} \times (RB)_{0:15}$ $RT \leftarrow - Prod_{0:31} + (RT)$	nmacchw – modulo signed nmacchws – saturate signed
multiply high halfword to word	$RT \leftarrow (RA)_{0:15} \times (RB)_{0:15}$	mulhhw – modulo signed mulhhwu – modulo unsigned
multiply low halfword to word	$RT \leftarrow (RA)_{16:31} \times (RB)_{16:31}$	mullhw – modulo signed mullhwu - modulo unsigned
multiply cross halfword to word	$RT \leftarrow (RA)_{16:31} \times (RB)_{0:15}$	mulchw – modulo signed mulchwu -modulo unsigned

Table 2. Integer MAC Unit Instructions

Exception Handling Logic

The 405 CPU services exceptions generated by error conditions, the internal timer facility, debug events, and the external interrupt controller (EIC) interface. All together, there are nineteen possible exceptions including the two provided by the EIC interface.

Exceptions are divided into two classes, critical and non-critical. Each class of exceptions has its own pair of save/restore registers for holding the machine status. Separate save/restore registers allows the 405 CPU core to quickly handle critical interrupts.

When an interrupt is taken, the 405 CPU automatically writes the machine status to save/restore register SRR0 and SRR1 for non-critical interrupts or SRR2 and SRR3 for critical interrupts. SRR0 and SRR2 are

written with the address of the excepting instruction or the next sequential instruction. SRR1 and SRR3 are written with a copy of the machine state register. The machine status is automatically restored at the end of an exception handler when the return from interrupt (rfi) or return from critical interrupt (rfci) instruction is executed.

Instruction and Data Cache

The 405 CPU core accesses memory through the instruction (ICU) and data cache units (DCU). The cache units each include a 64 bit PLB master interface, cache arrays and a cache controller. The ICU and DCU handle cache misses as request over the PLB to another PLB device such as an external bus interface unit. Cache hits are handled as single cycle memory accesses to the instruction and data caches.

The 405 CPU core has separate two-way set-associative instruction and data caches with 8 word (32 byte) cache lines. Instruction and data cache sizes are factory-configurable to any combination of 0KB, 4KB, 8KB, 16KB, or 32KB cache sizes. Configurable cache sizes provide designers with a parameter for optimizing the 405 CPU core to a desired price-performance for a particular application.

The cache arrays are non-blocking. Non-blocking caches allow the 405 CPU core to overlap execution of instructions while line reads over the PLB take place. The caches, therefore, continue supplying data and instructions without interruption to the pipeline.

The cache controllers replace cache lines according to the least recently used (LRU) replacement policy. The LRU bit determines which line of a set (congruence class) to replace. The most recently accessed line in the set is retained and the other line is marked as LRU, using the LRU bit. The cache controller replaces the LRU line during a line fill.

Instruction Cache Unit (ICU)

ICU Line Fills
The ICU utilizes an eight word fill buffer to enhance cache performance. For cacheable memory, a requested cache line first passes through the fill buffer prior to being written into the cache arrays. When the fill buffer contains all eight words of a cache line, the ICU writes the contents of the fill buffer to the cache. Using the fill buffer reduces cache blocking on line fills from eight to two cycles.

While the cache line is being captured in the fill buffer, the ICU forwards the target word (the word requested by the fetcher) to the fetch queue, bypassing the cache arrays. If the fetch logic makes further requests of words in the cache line, the ICU places them on the bypass path to expedite the request.

Non-cacheable memory accesses benefit from the bypass path and the fill buffer when 4 or 8 word non-cacheable line reads are enabled.. In response to a PLB fetch request, the ICU places the target word on the bypass path while the fill buffer captures the remaining instructions. The fill buffer will hold the instructions until overwritten by another request. A performance boost occurs when the fetch logic requests instructions held in the fill buffer since the fill buffer has the same access latency as a cache hit.

Big Endian and Little Endian Support
The 405 CPU core supports big endian or little endian byte ordering for instructions stored in external memory. Since the PowerPC architecture is big endian internally, the ICU rearranges the instructions stored as little endian into the big endian format. Therefore, the instruction cache always contains instructions in big endian format so that the byte ordering is correct for the execute unit. This feature allows the 405 core to be used in systems designed to function in a little endian environment.

Data Cache Unit (DCU)

DCU Line Fills
For cacheable line fills, data passes through the fill buffer and the write buffer prior to being written into the data cache. After the fill buffer receives the eighth word of the cache line, the data cache controller writes the contents of the fill buffer into the write buffer. The cache controller then takes two clock cycles to write the contents of the write buffer into the cache arrays. Using the write buffer reduces cache blocking from 8 to 3 cycles and frees the fill buffer for a subsequent line reads.

Cacheable data requested by the execute unit is sent on the by pass path directly into the operand registers of the execute unit. The data cache controller places the target word on the bypass path as the fill buffer captures data words off the PLB. Additional requests of the cache line held in the fill buffer are also forwarded directly to the operand registers in the execute unit.

For non-cacheable line accesses, the DCU utilizes the fill buffer even though the data will not be written into the data cache. As with cacheable line accesses, the targeted data word is sent on the by pass path directly to the operand registers of the execute unit. The data captured in the fill buffer is accessible to the execute unit until it is overwritten by a subsequent read (load) request. Data accessed in the fill buffer achieves cache hit performance which gives non-cacheable accesses a noticeable performance boost.

DCU Line Flushes
Single queued flushes are non-blocking since the DCU contains a 2-line flush queue. A 2-line flush queue enables the DCU to access the data cache arrays for load and store cache hits while handling a single line flush. Flushed data is first collected in the line buffer built into the output of the data cache arrays. The DCU then passes the flushed data cache line to the second queue position, the flush buffer. From the flush buffer, the data is sent to memory by way of the PLB.

DCU Write Strategies
The DCU supports write-back or write-through mode and allocate-on-write. In write-back mode, store hits are written to the cache and not to main memory. Main memory is later modified if and when the line is flushed from the cache. In Write-through mode, the data cache controller writes main memory for store misses as well as store hits; every store operation generates a PLB write request. Both modes can be used with allocate-on-write. When allocate-on-write is enabled, a store miss to cacheable memory forces the data cache controller to allocate a line in the data cache, generate a line fill and flush the LRU line if marked dirty.

DCU Load Strategies
For cacheable memory, the DCU typically places an 8 word read request on the PLB and allocates a line in the data cache when a load miss to the cache occurs. But the DCU may also be configured to request either a word or an 8 word line and not allocate a cache line. If a cache line is not allocated, the DCU handles the requested data as if it is non-cacheable. Controlling allocation of cacheable memory allows programmers to hold critical data in the data cache.

Byte Steering Logic
The 405 CPU core has hardware support for big endian and little endian unaligned accesses. Since byte ordering and alignment depends on the data size (byte, half word or word) accessed, the data cache exactly mirrors external memory. Alignment and byte ordering are handled by byte steering logic between the DCU and the GPR during load operations. Byte steering logic in the DCU returns data to its original byte ordering before writing it back to the data cache, on-chip memory or memory accessed over the PLB. Having byte steering logic integrated into 405 CPU core eliminates the difficulty of integrating peripherals originally designed for a little endian system.

Memory Management Unit (MMU)

The MMU supports multiple page sizes as well as a variety of storage protection attributes and access control options. Multiple page sizes improve memory efficiency and minimizes the number of TLB misses. The 405 CPU core gives programmers the flexibility to have any combination of the following eight possible pages sizes in the translation look-aside buffer (TLB) simultaneously: 1KB, 4KB, 16KB, 64KB, 256KB, 1MB, 4MB and 16MB.

Each page of memory is accompanied by a set of storage attributes. These attributes include cacheability, write through/write back mode, allocate on write, big/little endian, guarded and user-defined. The user-defined attribute can be used to mark a memory page with an application specific meaning. The guarded attribute controls speculative accesses. The big/little endian attribute marks a memory page as having a particular byte ordering. Write through/write back specifies whether memory is updated in addition to the cache during store operations.

Access control options enable system software to control general access for programs in the problem state, and control write and execute permissions for all pages. The access control options are entries in the Zone protection register (ZPR). The ZPR contains 16 zones. A zone is an arbitrary identifier for grouping TLB entries for purposes of protection. Each memory page is marked as belonging to a particular zone.

The MMU includes a 64-entry fully-associative unified TLB to reduce the overhead of address translation. Contention for the main TLB between data address and instruction address translation is minimized through the use of a four-entry instruction shadow TLB (ITLB) and an eight-entry data shadow TLB (DTLB). The ITLB and DTLB shadow the most recently used entries in the unified TLB. The MMU manages the replacement strategy of the ITLB and DTLB leaving the unified TLB to software control. Real-time operating systems are free to implement their own replacement algorithm for the unified TLB.

The 405 CPU core offers real mode as an alternative to address translation and MMU storage attributes. Real mode supports the same storage attributes as the MMU. Disabling address translation for instruction addresses and/or data addresses enables real mode for all instruction and/or data addresses. Real mode offers separate control of the storage attributes for all instruction and data addresses through a set of registers that divide the 4GB address space into thirty-two, 128 MB fixed memory regions.

Timers

The 405 CPU core contains a 64-bit time base and three timers: the Programmable Interval Timer (PIT), the Fixed Interval Timer (FIT), and the watchdog timer. The time base counter increments synchronously with the CPU clock or an external clock source. The three timers are synchronous with the time base.

The PIT is a 32-bit register that decrements at the same rate as the time base is incremented. The user loads the PIT register with a value to create the desired delay. When the register reaches zero, the timer stops decrementing and generates a PIT interrupt. Optionally, the PIT can be programmed to auto-reload the last value written to the PIT register, after which the PIT continues to decrement.

The FIT generates periodic interrupts based on one of four selectable bits in the time base. When the selected bit changes from 0 to 1, the 405 CPU core generates a FIT interrupt.

The watchdog timer provides a periodic critical-class interrupt based on a selected bit in the time base. This interrupt can be used for system error recovery in the event of software or system lockups. Users may select one of four time periods for the interval and the type of reset generated if the watchdog timer expires twice without an intervening clear from software. If enabled, the watchdog timer generates a reset unless an exception handler updates the watchdog timer status bit before the timer has completed two of the selected timer intervals.

Debug Logic

All architected resources on the 405 CPU core can be accessed through the debug logic. Upon a debug event, the 405 CPU core provides debug information to an external debug tool. Three different types of tools are supported depending on the debug mode: ROM Monitors, JTAG debuggers and instruction trace tools.

In internal debug mode, a debug event enables exception-handling software at a dedicated interrupt vector to take over the CPU core and communicate with a debug tool. The debug tool has read-write access to all registers and can set hardware or software breakpoints. ROM monitors typically use the internal debug mode.

In external debug mode, the CPU core enters stop state (i.e., stops instruction execution) when a debug event occurs. This mode offers a debug tool non-invasive read-write access to all registers in the 405 CPU core. Once the CPU core is in stop state, the debug tool can start the CPU core, step an instruction, freeze the timers or set hardware or software break points. In addition to CPU core control, the debug logic is capable of writing instructions into the instruction cache, eliminating the need for external memory during initial board bring up. Communication to a debug tool using external debug mode is through the JTAG port.

Debug wait mode offers the same functionality as external debug mode with one exception. In debug wait mode, the CPU core goes into wait state instead of stop state after a debug event. Wait state is identical to stop state until an interrupt occurs. In wait state, the 405 CPU core can vector to an exception handler, service an interrupt and return to wait state. This mode is particularly useful when debugging real time control systems.

Real-time trace debug mode is always enabled. The debug logic continuously broadcasts instruction trace information to the trace port. When a debug event occurs, the debug logic signals an external debug tool to save instruction trace information before and after the event. The number of instructions traced depends on the trace tool.

Debug events signal the debug logic to stop the CPU core, put the CPU core in debug wait state, cause a debug exception or save instruction trace information. Table 3 on the following page lists the possible debug events and their description.

Debug Event	Description
Branch Taken	A Branch Taken debug event occurs prior to the execution of taken branch instruction.
Instruction Completion	The Instruction Completion debug event occurs after the completion of any instruction.
Exceptions	The Exception debug event occurs after an interrupt is taken.
Trap	The Trap debug event occurs prior to the execution of a trap instruction, where the trap condition is met.
Instruction Address Compare (IAC)	The IAC debug event occurs prior to the execution of an instruction at an address that matches the contents of one of four IAC registers (IAC1, IAC2, IAC3, and IAC4).
Instruction Address Range	The Instruction Address Range debug event occurs prior to the execution of an instruction at an address contained in one of the following ranges as specified by the four IAC registers: IAC1 <= range < IAC2 (inclusive), IAC3 <= range < IAC4 (inclusive), range low < IAC1 < IAC2 <= range high (exclusive), or range low < IAC3 < IAC4 <= range high (exclusive).
Data Address Compare (DAC)	The DAC debug event occurs prior to the execution of an instruction that accesses a data address matching the contents of one of the two DAC registers (DAC1 and DAC2).
Data Address Range	The Data Address Range debug event occurs prior to the execution of an instruction that accesses a data address within one of the following ranges specified by the two DAC registers: DAC1 <= range < DAC2 (inclusive), or range low < DAC1 < DAC2 <= range high (exclusive).
Data Value Compare (DVC)	The Data Value Compare debug event occurs prior to the execution of an instruction that accesses a data address matching one of the two DAC registers and containing a particular data value as specified by one of the two DVC registers. The DVC debug event may occur when a selected data byte, half-word or word matches the corresponding element in DVC 1 or DVC2.
Unconditional Event	An unconditional debug event is set by a debug tool through the JTAG port or by ASIC logic external to the 405 CPU core.

Table 3. Table of Debug Events

Core Interfaces

Processor Local Bus (PLB) interface
The 405 CPU core accesses system resources through PLB interfaces on the instruction and data cache controllers. The cache controllers are both 64-bit PLB masters. The PLB is a high performance on-chip bus optimized for Core+ASIC development or system-on-a-chip design.

DCR Bus interface
The Device Control Register (DCR) bus is a configuration bus for components external to the 405 CPU core. Using the DCR bus to manage status and configuration registers reduces PLB traffic and improves system integrity. System resources on the DCR Bus are protected or isolated from wayward code since the DCR bus is not part of the system memory map.

OCM interfaces
Access to direct-mapped memory, added to the 405 CPU core, is through the on-chip memory (OCM) interfaces. The OCM interfaces have the same access time as a cache hit. Memory attached to the 405 CPU core may be accessed through the instruction OCM interface and/or the data OCM interface. Instruction side local-memory is often used to hold critical code such as an interrupt handler that requires guaranteed low-latency access. Data side local-memory offers the same fixed low-latency access and is used to hold critical data such as filter coefficients for a DSP application.

Auxiliary Processor Unit (APU)/Floating Point Unit (FPU) Interface
The APU/FPU interface enables a custom design implementation to use an auxiliary processor to execute instructions that are not part of the PowerPC Architecture or IBM PowerPC Embedded Environment, or to execute PowerPC Floating Point instructions

External Interrupt Controller (EIC) Interface
The EIC interface extends interrupt support to logic external to the 405 CPU core through the external and critical interrupt signals. These inputs are level sensitive. The critical interrupt and external interrupt signals are conceptually logic OR's of all implementation-specific critical and non-critical interrupts outside the core.

Debug interface
Debugging interfaces on the 405 CPU core, consisting of the JTAG and Trace ports, offers access to resources internal to the core and assist in software development. The JTAG port provides basic JTAG chip testing functionality as well as the ability for external debug tools to gain control of the processor for debug purposes. The Trace port furnishes programmers with a mechanism for acquiring instruction traces.

The JTAG port complies with IEEE Std 1149.1, which defines a test access port (TAP) and boundary scan architecture. Extensions to the JTAG interface provide debuggers such as RISCWatch with processor control that includes stopping, starting and stepping the 405 CPU core. These extensions are compliant with the IEEE 1149.1 specifications for vendor-specific extensions.

The Trace port provides instruction trace information to an external trace tool, such as RISCTrace. The 405 CPU core is capable of back trace and forward trace. Back trace is the tracing of instructions prior to a debug event while forward trace is the tracing of instructions after a debug event.

For further information regarding the PowerPC 405 core, contact an IBM Microelectronics sales representative. To identify your local sales representative, view the listing on the WWW at: http://www.chips.ibm.com/support/