The PowerPC® 440 Core

A high-performance, superscalar processor core for embedded applications

IBM Microelectronics Division
Research Triangle Park, NC

09/21/1999

Overview

The PowerPC 440 CPU core is the latest addition to IBM's family of 32-bit RISC PowerPC embedded processor cores. The PPC440's high-speed, superscalar design and Book E Enhanced PowerPC Architecture ä put it at the leading edge for high performance system-on-a-chip (SOC) designs. The PPC440 core marries the performance and features of standalone microprocessors with the flexibility, low power, and modularity of embedded CPU cores.

Target Applications

The PPC440 Core is primarily designed for applications in which maximum performance and extensive peripheral integration are the critical selection criteria.

Target market segments for the PPC440 core include:

- Consumer applications including digital cameras, video games, set-top boxes, and internet appliances
- Office automation products such as laser printers, thin-client systems, and sub-notebooks
- Storage and networking products such as RAID controllers, routers, ATM switches, cellular basestations, and network cards

Features

- 2-way superscalar design
- Out-of-order issue, execution, and completion
- Dynamic branch prediction
  - Single-cycle branch latency
- Three execution pipelines
- Single-cycle throughput on 32x32 multiply
- 24 DSP operations (16x16+32->32, MAC with single-cycle throughput)
- Real-time non-invasive instruction trace

Typical Application

A typical system on a chip design with the PPC440 Core uses the CoreConnect™ bus structure for system level communication. High bandwidth peripherals and the PPC440 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 1 shows an example PPC440 Core-based system on a chip, illustrating the two-level bus structure and modular core-based design.

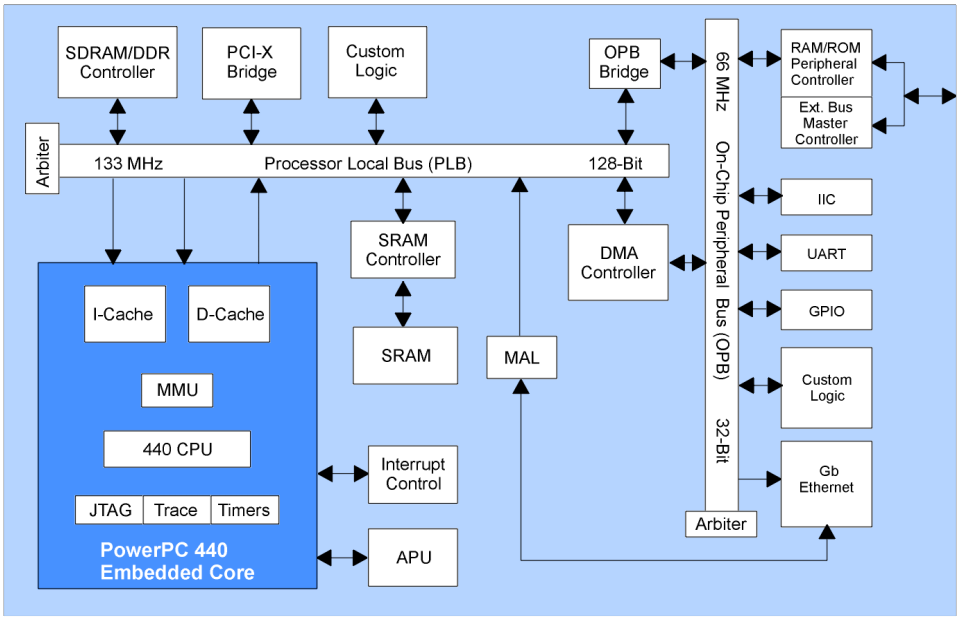

Figure 1. Example PPC440 Core + ASIC

Specifications

Performance (Dhrystone 2.1)	1000 MIPS @ 555MHz (est.), Nominal silicon, 1.8V, 55°C 720 MIPS @ 400MHz (est.), Slow silicon, 1.65V, 85°C
Frequency	0 – 400MHz , Slow silicon, 1.65V, 85°C 555MHz nominal
Power Dissipation	2.5mW / MHz @ 1.8V (est.), hard core with 32KI / 32KD caches
Architecture	32-bit PowerPC Book E compliant, application code compatible with all PowerPC processors
Die Size	4.0 mm^2 for CPU only (est.)
Caches	0-64KB, 32-way to 128-way associative
Technology	0.18 µm CMOS copper technology 0.12 µm L_{eff} , 4 levels of metal
Power Supply	1.8 Volts
Transistors	5.5M, hard core with 32KI / 32KD caches
Operating Range	-40°C to 125°C, 1.6V to 1.9V
Data Bandwidth	Up to 6.4 GB/sec via three 128-bit, 200MHz CoreConnect bus interfaces

Table 1- 440 CPU Core Specifications

Embedded Design Support

The PPC440 Core, as a member of the PowerPC 400 Family, 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 PPC440 include C/C++ compilers, debuggers, bus functional models, hardware/software co-simulation environments, 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™ trace interface, VHDL and Verilog simulation models and a PPC440 Core Superstructure development kit.

PPC440 CPU Core Organization

PPC440 CPU

The PPC440 CPU operates on instructions in a dual issue, seven stage pipeline, capable of dispatching two instructions per clock to multiple execution units and to optional Auxiliary Processor Units (APUs). The PPC440 core is shown in Figure 2.

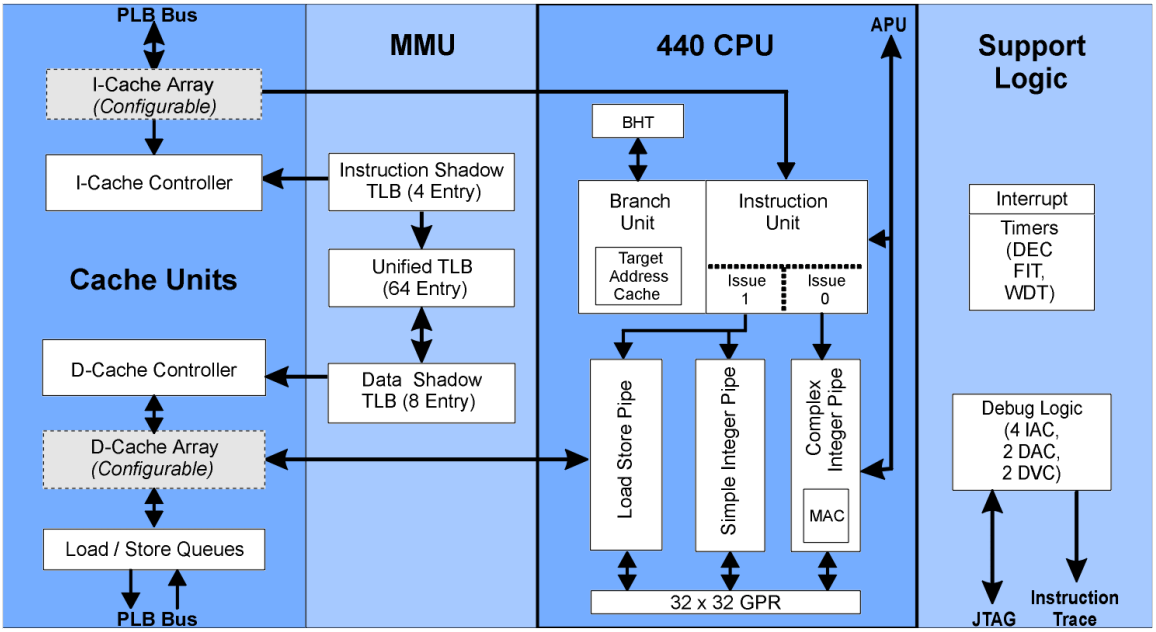

Figure 2 - PPC440 Core Block Diagram

The pipeline contains the following stages, as shown in Figure 3:

1. IFTH – Fetch instructions from instruction cache
2. PDCD – Pre-decode; partial instruction decode
3. DISS – Decode/Issue; final decode and issue to units
4. RACC – Register Access; read from multi-ported General Purpose Register (GPR) file
5. EXE1/AGEN – Execute stage 1; complete simple arithmetics, generate load/store address
6. EXE2/CRD – Execute stage 2; multiplex in results from units in preparation for writing into GPR file, Data Cache access
7. WB – Writeback; write results into GPR file from integer operation or load operation

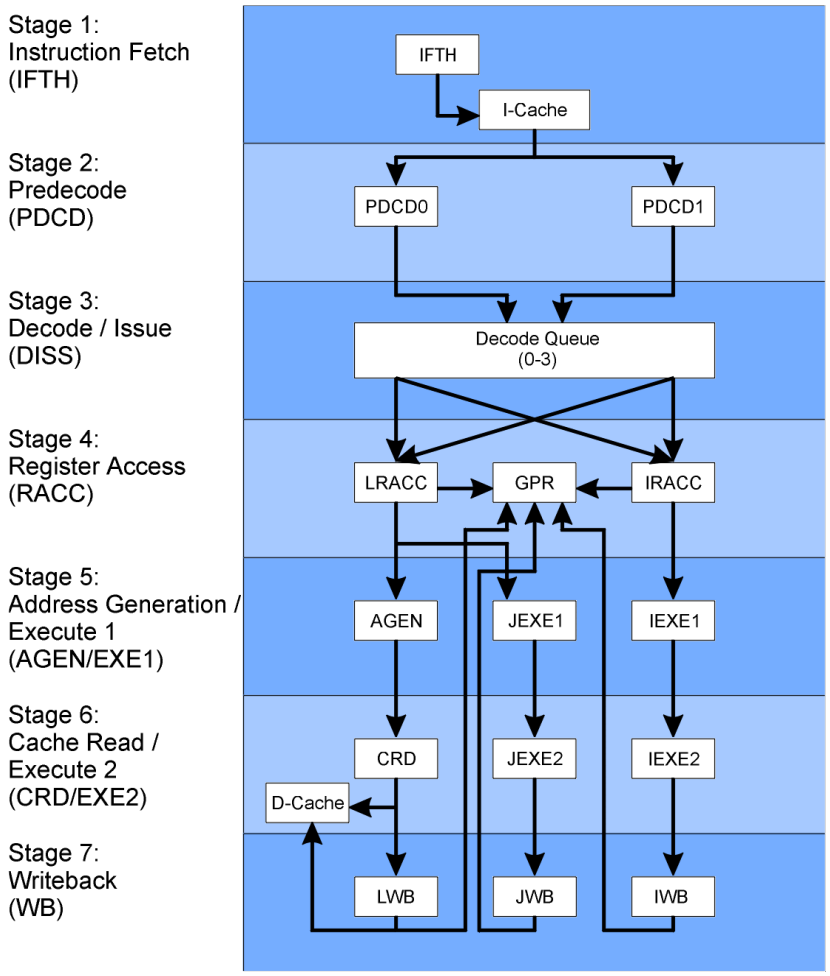

Figure 3 - PPC440 CPU Pipeline

Instruction Fetch and Pre-decode

During the Instruction Fetch stage (IFTH), an entire cache line (eight words) is read into the instruction cache line read buffer. From there, the next two instructions in the pre-decode buffers PDCD0 and PDCD1 during the PDCD stage. The instruction cache is virtually indexed and tagged, and translation is performed in parallel with the cache access.

Branch Unit

The PPC440 uses a Branch History Table (BHT) to maintain dynamic branch prediction of conditional branches. To perform dynamic branch prediction, a 2-bit counter in the BHT is used to decide whether prediction should agree or disagree with the normal PowerPC static branch prediction. The counter counts up if branch determination agrees, and down if it disagrees. Once the counter saturates, it can only count away from saturation. Therefore, four valid states exist: “Strongly agree”, “Agree”, “Disagree”, and “Strongly disagree”. By agreeing or disagreeing with static branch prediction, different branches can use the same counter in the BHT and have opposite static predictions, without the machine necessarily mispredicting a branch.

The Branch Target Address Cache (BTAC) is used to predict branches and deliver their target addresses before the instruction cache can deliver the same data. It is accessed during IFTH, whereas normal branch prediction would not occur until PDCD, and therefore avoids a one cycle penalty. The BTAC is made up of an odd and even BTAC containing eight entries each. Only unconditional branches and bdnz

instructions are stored, which gives a significant performance boost while keeping the design straightforward.

Decode and Issue
The four-entry decode queue accepts up to two instructions per clock **submitted** from the pre-decode buffers. Instructions always enter the lowest empty or emptying queue position, behind any instructions already in the queue. Therefore, the queue fills from the bottom up, instructions stay in order, and no bubbles exist in the queue. A significant portion of decode is performed in the lowest two positions (DISS0 and DISS1). Up to two instructions exit the queue based on the instructions' decode and pipeline availability, and are **issued** to the RACC stage. DISS1 can issue out of order with respect to DISS0.

Register Access
Conceptually, the GPR file consists of thirty-two, 32-bit general purpose registers. It is implemented as two 6-port arrays, (one array for LRACC, one for IRACC) each with thirty-two, 32-bit registers containing three write ports and three read ports. On all GPR updating instructions, the appropriate GPR write ports will be written in order to keep the contents of the files the same. On GPR reads, however, the GPR read ports are dedicated to instructions that are **dispatched** to a RACC's associated pipe(s).

Execution Pipelines
The PPC440 contains three execution pipes: a load/store pipe ("L-pipe"), a simple integer pipe ("J-pipe"), and a complex integer pipe ("I-pipe"). The L-pipe and J-pipe instructions are dispatched from the LRACC; I-pipe instructions are dispatched from IRACC. The three pipes together perform all 32-bit PowerPC integer instructions in hardware compliant with the PowerPC Book E specification. Table 2 lists the rules for dispatching to each of the three execution pipes.

L-pipe only	Loads/stores[1], cache instructions, mbar, msync
I-pipe or J-pipe[2]	Add, addi, addis, and, andc, cntlzw, eqv, extsb, extsh, nand, neg, nor, or, orc, ori, oris, xori, xoris, rlwimi, rlwinm, rlwnm, slw, srw, subf
I-pipe only	Branches, multiplies, divides, move to/from DCR/SPR, indirect XER updates, indirect LR/CTR updates, indirect CR updates, CR-logicals, MAC instructions, mcrf, mcrxr, mtcrf, mfcr, compares, dlmzb, isync, rfi, rfci, sc, wrtee, wrteei, mtmsr, mfmsr, traps

Table 2 – Rules for Instruction Issue

The MAC unit is an auxiliary processor unit (APU) which adds 24 operations to the PPC440 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 single cycle throughput. The MAC unit is contained within the I-pipe.

[1] The stwcx. instruction goes down both the L-pipe as well as the I-pipe, in order to update the CR.
[2] Instructions which update the CR or XER are not issued to the J-pipe.

Instruction and Data Caches

Processor Local Bus (PLB) Memory Access
The PPC440 has three independent 128-bit Processor Local Bus (PLB) master interfaces, one for instruction fetches, one for data reads, and a third for data writes. Memory accesses are performed through the PLB interfaces to/from the instruction cache (I-Cache) or data cache (D-Cache) units. Having three independent bus interfaces for the cache units provides maximum flexibility for designs to optimize system throughput. Memory accesses (loads/stores) which hit in the cache achieve single-cycle throughput.

Cache Configuration
The PPC440 has separate instruction and data caches with 8 word (32 byte) cache lines. Instruction and data cache sizes are factory-configurable to any combination of 0KB, 8KB, 16KB, 32KB, or 64KB cache sizes. Configurable cache sizes provide designers with a parameter for optimizing the PPC440 to a desired price-performance for a particular application. The caches are highly associative, with associativity varying with cache size as shown in Table 3. High associativity enables advanced cache functions such as locking and transient memory regions (see "Cache Partitioning" below).

Cache Size	Ways
8 KB	32
16KB	64
32KB	64
64KB	128

Table 3 – Number of Ways for Different PPC440 Cache Sizes

The cache arrays are non-blocking. Non-blocking caches allow the PPC440 to overlap execution of load/store instructions while instruction fetches take place over the PLB. The caches, therefore, continue supplying data and instructions without interruption to the pipeline. The PPC440 replaces cache lines according to a round-robin replacement policy.

The initial PPC440A4 core offering will include a 32KB instruction cache and 32KB data cache. These caches are physically constructed using two, 16KB CAMRAM macros, each consisting of 8, 2KB sub-banks (or "sets"). This organization facilities low-power operation and fast hit/miss determination.

Cache Partitioning
The PPC440 caches have the ability to be separated into "normal", "transient", and "locked" regions. Normal regions are what is traditionally thought of regarding cache replacement. Transient regions are used for data that is used temporarily and then not needed again, such as the data in a particular JPEG image. A separate transient region avoids castouts of more commonly accessed code in the normal region. The locked region is for code that is not to be cast out of the cache, and is the resulting region not included in the normal and transient regions. The regions are set via "victim" ceiling and floor pointers, as shown in Figure 4. Figure 4 shows two examples of cache partitioning, the left side shows separate transient and normal regions, and the right side shows part of the normal region overlapping with the transient region. The normal ceiling is defined as the top of the cache.

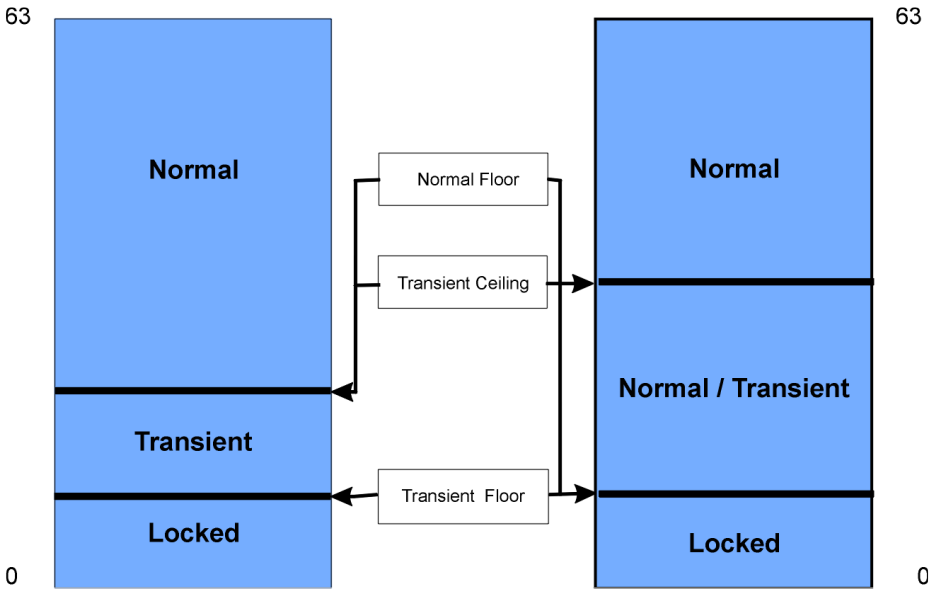

Figure 4 – Two Examples of Cache Partitioning

I-Cache Speculative Pre-fetching

The I-Cache utilizes a programmable speculative pre-fetch mechanism to enhance performance. Software can enable up to three additional lines to be speculatively pre-fetched, using a burst protocol, upon any instruction cache miss. When this mode is enabled, the I-Cache controller will automatically inspect the I-Cache on a miss to see if any of up to the next three lines are also misses. If so, the hardware will present a burst request to the PLB immediately after the original line fill request. This speculative burst request takes advantage of the throughput capability of standard memory architectures such as SDRAM and brings in anticipated subsequent instructions after a miss. Furthermore, if the instruction stream branches away from the lines which are being speculatively filled, the burst request which is filling the speculative lines can be abandoned in the middle, and a new fill request at the branch target location immediately initiated. There is a programmable "threshold" to determine when to abandon a speculative line fill that may have been in progress at the time of a branch redirection. This threshold designates how many doublewords of the speculative cache line must be received to *not* abandon a current line fill. In this fashion, the speculative pre-fetch mechanism can be carefully tailored to provide optimum performance for specific applications and memory subsystems.

D-Cache Line Fills

The D-Cache contains three line fill buffers and can queue up to four load misses to three separate cache lines. The PPC440 will then execute past these load misses, until the queue is full or the pipes are held waiting for a load value. The D-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.

D-Cache Non-cacheable Store Gathering

The D-Cache “gathers” up to 16 bytes for non-cacheable, write-through, and w/o allocate stores, and will burst the quadword to the PLB for fast writes to non-cacheable memory.

D-Cache Write-Back and Write-Through Modes

The D-Cache supports write-back or write-through mode. 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. (Although write-through requests to non-cacheable memory can be gathered as previously mentioned).

D-Cache Store Allocation

The D-Cache can be programmed whether or not to allocate a line on a D-Cache store miss. Write-on-allocate is enabled by default. In this mode, a store miss to cacheable memory forces the data cache controller to allocate a line in the data cache and generate a line fill. In contrast, when "without allocate" is enabled, a store miss to cacheable memory will *not* allocate a line data cache and will simply write the data to memory.

Big Endian and Little Endian Support

The PPC440 supports big endian or little endian byte ordering for instructions and data stored in external memory. The PowerPC Book E architecture is endian neutral; each page in memory can be configured for big or little endian byte ordering via a storage attribute contained in the TLB entry for that region. Strapping signals on the PPC440 core initialize the beginning TLB entry's endian attribute, so the PPC440 can boot from little or big endian memory.

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 TLB efficiency and minimize the number of TLB misses. The PPC440 gives programmers the flexibility to have any combination of the following eight possible page sizes in the translation look-aside buffer (TLB) simultaneously: 1KB, 4KB, 16KB, 64KB, 256KB, 1MB, 16MB and 256MB. Having an extremely large page size allows users to define system memory with a minimal number of TLB entries, thereby simplifying TLB allocation and replacement. Small page sizes prevent the wasting of memory when allocating small areas of data.

Each page of memory is accompanied by a set of storage attributes. These attributes include cacheability, write through/write back mode, big/little endian, guarded and four user-defined attributes. The user-defined attributes 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 big or little endian byte ordering. Write through/write back specifies whether memory is updated in addition to the cache during store operations.

Two of the user-defined storage attributes can be programmed for special functions inside the core. One can be enabled to designate normal or transient cache regions. Another can be enabled to control whether or not store misses allocate a line in the D-Cache.

Access control bits in the TLB entries enable system software to control read, write, and execute access for programs in both user and supervisor states.

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.

Interrupt Handling Logic

The PPC440 services exceptions generated by error conditions, the internal timer facilities, debug events, and the external interrupt controller (EIC) interface. Altogether, there are sixteen different interrupt types supported.

Interrupts are divided into two classes, critical and non-critical. Each class of interrupt has its own pair of save/restore registers for holding the program counter and machine state. Separate save/restore registers allow the PPC440 to quickly handle critical interrupts even within a non-critical interrupt handler.

When an interrupt is taken, the PPC440 automatically writes the program counter and machine state to save/restore register SRR0 and SRR1 respectively for non-critical interrupts, or CSRR0 and CSRR1 respectively for critical interrupts. The machine status and program counter are automatically restored at the end of an exception handler when the return from interrupt (rfi) or return from critical interrupt (rfci) instruction is executed.

Timers

The PPC440 contains a 64-bit time base and three timers: the Decrementer (DEC), the Fixed Interval Timer (FIT), and the WatchDog Timer (WDT). 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 DEC is a 32-bit register that decrements at the time base increment rate. The user loads the DEC register with a value to create the desired delay. When the register reaches zero, the timer stops decrementing and generates a decrementer interrupt. Optionally, the DEC can be programmed to auto-reload the value last written to the DEC auto-reload register, after which the DEC 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 PPC440 generates a FIT exception.

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 PPC440 can be accessed through the debug logic. Upon a debug event, the PPC440 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.

Internal Debug Mode
In internal debug mode, a debug event enables exception-handling software at a dedicated interrupt vector to take over the PPC440 and communicate with a debug tool. Exception-handling software has read-write access to all registers and can set hardware or software breakpoints. ROM monitors typically use the internal debug mode.

External Debug Mode
In external debug mode, the PPC440 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 PPC440 via the JTAG interface. Once the PPC440 is in stop state, the debug tool can start the PPC440, step an instruction, freeze the timers or set hardware or software break points. In addition to PPC440 control, the debug logic is capable of writing instructions into the instruction cache, eliminating the need for external memory during initial board bring up.

Debug Wait Mode
Debug wait mode offers the same functionality as external debug mode with one difference; in debug wait mode, the PPC440 will respond to interrupts and temporarily leave stop state to service them before returning to debug wait mode. In external debug mode, by contrast, interrupts are disabled while in stop state. Debug wait mode is particularly useful when debugging real-time control systems.

Real-Time Trace Debug Mode
In real-time trace debug mode, instruction trace information is continuously broadcast to the trace port. When a debug event occurs, an external debug tool saves instruction trace information before and after the event. The number of traced instructions depends only on the memory buffer depth of the trace tool.

Debug Events
Debug events signal the debug logic to either stop the PPC440, put the PPC440 in debug wait state, cause a debug exception, or save instruction trace information, depending on the debug mode. Table 4 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 a taken branch instruction.
Instruction Completion	The Instruction Completion debug event occurs after the completion of any instruction.
Return from Interrupt	The Return From Interrupt debug event occurs after the completion of an rfi or rfci instruction.
Interrupt	The Interrupt 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). Alternatively, the registers can be combined to cause an IAC debug event 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). Alternatively, the registers can be combined to cause a DAC 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 (or within a DAC range) 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 DVC1 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 PPC440.

Table 4 - Debug Events

Power Management

The PPC440 core, in keeping with the IBM PowerPC 400 family tradition, utilizes aggressive power management techniques for minimizing power. The PPC440 utilizes three key techniques: redundant operand registers, half-cycle latch stabilization, and dynamic clock gating.

Redundant Operand Registers
Redundant operand registers are used at various pipeline stages for feeding operands to each of the execution units. This saves power by preventing unused units from seeing the operand values being used by other units and improves performance by reducing loading and wire length in critical stages.

Half-Cycle Latch Stabilization
Half-cycle stabilization latches minimize the propagation of glitches to downstream logic. This is easily employed since the PPC440 core contains a master/slave latch arrangement for scan-test purposes. Therefore, a master-only latch is simply needed in the logic path that is switching in the first half of a cycle. For example, if the select lines for a mux are being determined in the first half of a cycle, then by putting a master-only latch on these select lines before delivering them to the mux, the mux outputs are prevented from glitching while the select lines are being determined. Conversely, if the data lines are unstable in the first half of a cycle, a stabilization latch may be used on the data inputs, while leaving the select lines alone.

Dynamic Clock Gating
The most important feature of the PPC440's dynamic power management is the extensive use of clock gating. Given the PPC440's master/slave latch organization, there are two possible gates that can be used. The relationship between them, and their relative affect on the clock splitter and hence power are shown in Figure 5.

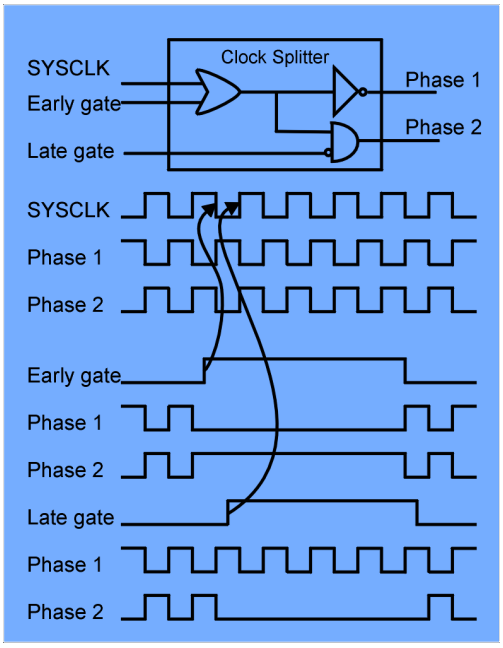

Figure 5 - PPC440 Clock Gating

In this figure, the early gate blocks the phase 1 clock and prevents the master latch from loading, while the late gate blocks the phase 2 clock and prevents the slave latch from loading. As illustrated in the simplified block diagram of the clock splitter, the early gate must arrive by mid-cycle -- which is when the system clock falls. If the gate is activated by this point, then the net effect is that internal to the clock splitter the fall on the system clock is never observed, and both the phase 1 and the phase 2 clock splitter

outputs remain stable, preventing any downstream master latches from loading, and hence their associated slave latches will not change either. This affords the maximum power savings, with the downstream logic dissipating no power other than leakage, and the clock splitter itself using almost zero power.

In the event that the gate for a given latch cannot be determined by mid-cycle, the late gate can be used, which does not prevent the system clock fall and consequent phase 1 clock rise, but does prevent the corresponding next phase 2 clock rise. This does not save as much power, but the timing is much more relaxed and the power savings are still considerable.

Core External Interfaces

Processor Local Bus (PLB) interface
The PPC440 accesses system resources through three independent PLB interfaces: one for instruction fetches, one for data reads, and a third for data writes. Each PLB controller is a 128-bit PLB master. The PLB is the high performance CoreConnect bus optimized for SOC design.

DCR Bus interface
The Device Control Register (DCR) bus is a configuration bus for components external to the PPC440. Using the DCR bus to manage status and configuration registers reduces PLB traffic and improves system bandwidth and 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.

Auxiliary Processor Unit (APU) Interface
The APU interface enables a custom design implementation to tightly couple coprocessor-type macros to the PPC440.

The APU interface provides sufficient functionality to attach macros such as a full PowerPC Floating Point Unit (single or double precision), a multimedia macro, DSP, or other custom functions implementing algorithms appropriate for the system application. The APU interface supports dual-issue pipeline designs, and utilizes a full 128-bit load/store path to the D-Cache. The interface can be used with macros that contain their own register files, or with simpler macros which use the CPU's register file for source and/or target operands.

The APU interface provides customers the capability to execute instructions that are not part of the PowerPC Book E architecture concurrently with the PPC440. Accordingly, areas have been reserved within the architected instruction space to allow for customer- or application-specific extensions.

External Interrupt Controller (EIC) Interface
The EIC interface extends interrupt support to logic external to the PPC440 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 PPC440, consisting of the JTAG and instruction trace ports, offer access to resources internal to the core and assist in software development. The JTAG port provides the ability for external debug tools to gain control of the processor for debug purposes. This interface provides debuggers such as RISCWatch with processor control that includes stepping, stopping, and starting the PPC440. The Trace port furnishes programmers with a mechanism for acquiring instruction traces. This trace information is captured via an external trace tool, such as RISCTrace. The PPC440 is capable of tracing before, around, or after an occurring debug event.

For further information regarding the PowerPC 440 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/