Order Number: MPC7410TS/D
Rev . 0, 9/2000

PowerPC

Advance Information

MPC7410 RISC Microprocessor Technical Summary

This document provides an overview of the MPC7410 PowerPC™ microprocessor features, including a block diagram showing the major functional components. It also provides information about how the MPC7410 implementation complies with the PowerPC™ architecture definition.

AltiVec™ Technology and the MPC7410

AltiVec technology features are described in the following sections:

- AltiVec registers are described in Table 3.
- AltiVec instructions are described in Section 3.2.2, “AltiVec Instruction Set.”
- Execution units for AltiVec instructions are described in Section 2.2.4.1, “AltiVec Vector Permute Unit (VPU),” and Section 2.2.4.2, “AltiVec Vector Arithmetic Logic Unit (VALU).”

Part I MPC7410 Microprocessor Overview

This section describes the features and general operation of the MPC7410 and provides a block diagram showing major functional units. The MPC7410 is an implementation of the PowerPC microprocessor family of reduced instruction set computer (RISC) microprocessors. The MPC7410 implements the 32-bit portion of the PowerPC architecture, which provides 32-bit effective addresses, integer data types of 8, 16, and 32 bits, and floating-point data types of 32 and 64 bits.

The MPC7410 also implements the AltiVec instruction set architectural extension of the PowerPC architecture. The MPC7410 is a superscalar processor that can dispatch and complete two instructions simultaneously. It incorporates the following execution units:

freescale™ semiconductor

- Floating-point unit (FPU)
- Branch processing unit (BPU)
- System register unit (SRU)
- Load/store unit (LSU)
- Two integer units (IUs):
  - IU1 executes all integer instructions.
  - IU2 executes all integer instructions except multiply and divide instructions.
- Two vector units that support AltiVec instructions:
  - Vector permute unit (VPU)
  - Vector arithmetic logic unit (VALU), which consists of the following independent subunits:
    - Vector simple integer unit (VSIU)
    - Vector complex integer unit (VCIU)
    - Vector floating-point unit (VFPU)

The ability to execute several instructions in parallel and the use of simple instructions with rapid execution times yield high efficiency and throughput for MPC7410-based systems. Most integer instructions (including VSIU instructions) have a one-clock cycle execution latency.

The FPU and VFPU are pipelined; that is, the tasks they perform are broken into subtasks executed in successive stages. Typically, a floating-point instruction occupies only one of the three FPU stages at a time, freeing the previous stage to operate on the next floating-point instruction. Thus, three floating-point instructions can be in the FPU execute stage at a time and one floating-point instruction can finish executing per processor clock cycle. The VFPU has four pipeline stages when executing in non-Java mode and five when executing in Java mode.

Note that for the MPC7410, double- and single-precision versions of floating-point instructions have the same latency. For example, a floating-point multiply-add instruction takes three cycles to execute, regardless of whether it is single- (**fmadds**) or double-precision (**fmadd**).

Figure 1 shows the parallel organization of the execution units (shaded in the diagram). The instruction unit fetches, dispatches, and predicts branch instructions. Note that this is a conceptual model that shows basic features rather than attempting to show how features are implemented physically.

The MPC7410 has independent on-chip, 32-Kbyte, eight-way set-associative, physically-addressed L1 (level-one) caches for instructions and data and independent instruction and data memory management units (MMUs). Each MMU has a 128-entry, two-way set-associative translation lookaside buffer (DTLB and ITLB) that saves recently used page address translations. Block address translation is implemented with the four-entry instruction and data block address translation (IBAT and DBAT) arrays, defined by the PowerPC architecture. During block translation, effective addresses are compared simultaneously with all four BAT entries.

The L2 cache is implemented with an on-chip, two-way, set-associative tag memory, and with external, synchronous SRAMs for data storage. The external SRAMs are accessed through a dedicated L2 cache port that supports a single bank of 256 Kbytes, 512 Kbytes, 1 Mbyte, or 2 Mbytes of synchronous SRAM. Alternately, the L2 interface can be configured to use half (256 Kbytes minimum) or all of the SRAM area as a direct-mapped, private memory space.

The MPC7410 has four software-controllable power-saving modes. Three static modes, doze, nap, and sleep, progressively reduce power dissipation. When functional units are idle, a dynamic power management mode causes those units to enter a low-power mode automatically without affecting operational performance, software execution, or external hardware. The MPC7410 also provides a thermal assist unit (TAU) and a way to reduce the instruction fetch rate for limiting power dissipation.

The MPC7410 uses an advanced CMOS process technology and is fully compatible with TTL devices.

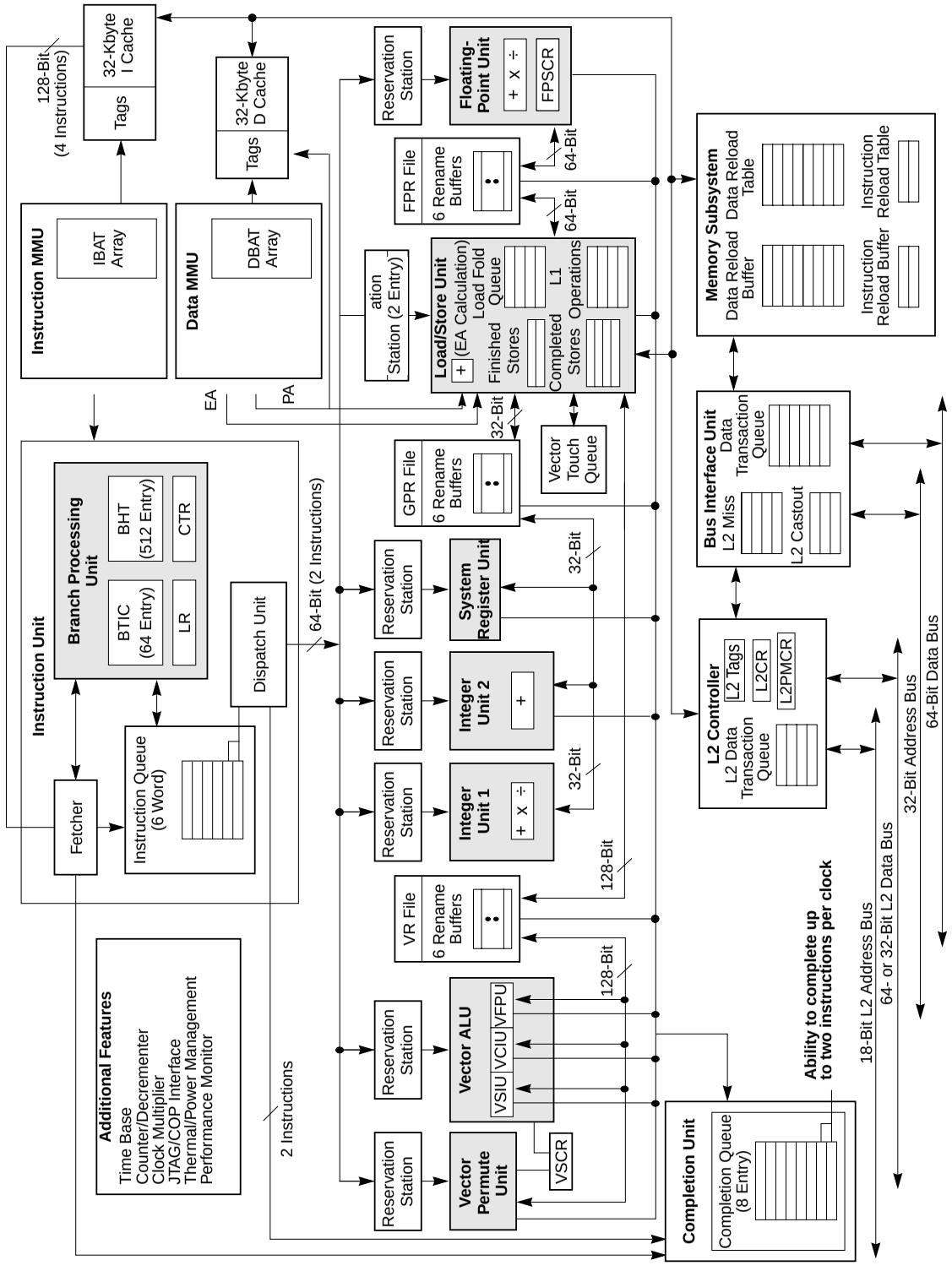


Figure 1. MPC7410 Microprocessor Block Diagram

Part II MPC7410 Microprocessor Features

This section describes features of the MPC7410. The interrelationships of these features are shown in

2.1 Overview of the MPC7410 Microprocessor Features

Major features of the MPC7410 are as follows:

- High-performance, superscalar microprocessor
  - As many as four instructions can be fetched from the instruction cache per clock cycle
  - As many as two instructions can be dispatched per clock
  - As many as eight instructions can execute per clock (including two integer instructions and four AltiVec instructions)
  - Single-clock-cycle execution for most instructions
  - One instruction per clock throughput for most instructions
- Eight independent execution units and three register files
  - Branch processing unit (BPU) features static and dynamic branch prediction
    - 64-entry (16-set, four-way set-associative) branch target instruction cache (BTIC), a cache of branch instructions that have been encountered in branch/loop code sequences. If a target instruction is in the BTIC, it is fetched into the instruction queue a cycle sooner than it can be made available from the instruction cache. Typically, if a fetch access hits the BTIC, it provides the first two instructions in the target stream.
    - 512-entry branch history table (BHT) with two bits per entry for four levels of prediction—not-taken, strongly not-taken, taken, strongly taken
    - Branch instructions that do not update the count register (CTR) or link register (LR) are removed from the instruction stream.
  - Two integer units (IUs) that share 32 GPRs for integer operands
    - IU1 can execute any integer instruction.
    - IU2 can execute all integer instructions except multiply and divide instructions (shift, rotate, arithmetic, and logical instructions). Most instructions that execute in IU2 take one cycle to execute. The IU2 has a single-entry reservation station.
  - Three-stage FPU and a 32-entry FPR file
    - Fully IEEE 754-1985-compliant FPU for both single- and double-precision operations
    - Supports non-IEEE mode for time-critical operations
    - Hardware support for denormalized numbers
    - Single-entry reservation station
    - Thirty-two 64-bit FPRs for single- or double-precision operands
  - Two vector units and 32-entry vector register file (VRs)
    - Vector permute unit (VPU)
    - Vector arithmetic logic unit (VALU), which consists of the three independent subunits: vector simple integer unit (VSIU), vector complex integer unit (VCIU), and vector floating-point unit (VFPU)

- Two-stage LSU
  - Supports integer, floating-point and vector instruction load/store traffic
  - Four-entry vector touch queue (VTQ) supports all four architected AltiVec data stream operations
  - Two-entry reservation station
  - Single-cycle, pipelined load or store cache accesses (byte, half, word, double word, quad word) including misaligned accesses within a double-word boundary
  - Dedicated adder calculates effective addresses (EAs)
  - Supports store gathering
  - Performs alignment, normalization, and precision conversion for floating-point data
  - Executes cache control and TLB instructions
  - Performs alignment, zero padding, and sign extension for integer data
  - Hits under misses (multiple outstanding misses) supported
  - Six-entry store queue
  - Sequencing for load/store multiples and string operations
  - Supports both big- and little-endian modes, including misaligned little-endian accesses
- SRU handles miscellaneous instructions
  - Executes CR logical and move to/move from SPR instructions (**mtspr** and **mfspr**)
  - Single-entry reservation station

- Rename buffers
  - Six GPR rename buffers
  - Six FPR rename buffers
  - Six VR rename buffers
  - Condition register buffering supports two CR writes per clock
- Completion unit
  - The completion unit retires an instruction from the eight-entry reorder buffer (completion queue) when all instructions ahead of it have been completed, the instruction has finished execution, and no exceptions are pending.
  - Guarantees sequential programming model (precise exception model)
  - Monitors all dispatched instructions and retires them in order
  - Tracks unresolved branches and flushes instructions from the mispredicted branch
  - Retires as many as two instructions per clock
- Separate on-chip L1 instruction and data caches (Harvard architecture)
  - 32-Kbyte, eight-way set-associative instruction and data caches
  - Pseudo least-recently-used (PLRU) replacement algorithm
  - 32-byte (eight-word) L1 cache block
  - Physically indexed/physical tags
  - Cache write-back or write-through operation programmable on a per-page or per-block basis
  - Instruction cache can provide four instructions per clock; data cache can provide four words per clock
  - Caches can be disabled in software
  - Caches can be locked in software

- Data cache coherency (MEI, MESI, and MERSI) maintained in hardware
- Separate copy of data cache tags for efficient snooping
- No snooping of instruction cache except for **icbi** instruction
- Data cache supports AltiVec LRU and transient instructions, as described in Section 3.2.2, “AltiVec Instruction Set.”
- The critical double word is made available to the requesting unit when it is burst into the reload data queue. The caches are nonblocking, so they can be accessed during this operation.

- Level 2 (L2) cache interface
  - L2 is fully pipelined to provide 64 or 32 bits per L2 clock cycle to the L1 caches
  - On-chip two-way set-associative L2 cache controller and tags
  - External data SRAMs
  - Support for 256-Kbyte, 512-Kbyte, 1-Mbyte, and 2-Mbyte L2 caches
  - Copyback or write-through data cache (on a per page basis, or for all L2)
  - 32-byte (256 K and 512 K), 64-byte (1 M), or 128-byte (2 M) sectored line size
  - Direct-mapped, private memory capability for half (256 Kbytes minimum) or all of the L2 SRAM space
  - Supports pipelined (register-register) synchronous burst SRAMs, PB3 pipelined (register-register) synchronous burst SRAMs, and pipelined (register-register) late-write synchronous burst SRAMs
  - Configurable core-to-L2 frequency divisors
  - Configurable for 64- or 32-bit L2 data bus
- Separate memory management units (MMUs) for instructions and data
  - 52-bit virtual address; 32-bit physical address
  - Address translation for 4-Kbyte pages, variable-sized blocks, and 256-Mbyte segments
  - Memory programmable as write-back/write-through, cacheable/noncacheable, and coherency enforced/coherency not enforced on a page or block basis
  - Separate IBATs and DBATs (four each) also defined as SPRs
  - Separate instruction and data translation lookaside buffers (TLBs)
    - Both TLBs are 128-entry, two-way set associative, and use LRU replacement algorithm
    - TLBs are hardware-reloadable (that is, the page table search is performed in hardware)
- Efficient data flow
  - All data buses between VRs, LSU, L1 and L2 caches, and the bus are 128 bits wide
  - The L1 data cache is fully pipelined to provide 128 bits/cycle to/from the VRs
  - The L2 cache is fully pipelined to provide 64 or 32 bits per L2 clock cycle to the L1 caches
  - Up to 8 outstanding, out-of-order, cache misses allowed between the L1 data cache and L2/bus
  - Up to seven out-of-order transactions on the bus, one in progress and six pending
  - Load folding to fold new L1 data cache misses into older, outstanding load and store misses to the same line

- Store miss merging for multiple store misses to the same line. Only coherency action taken (address-only) for store misses merged to all 32 bytes of a cache block (no data tenure needed).
- Two-entry finished store queue and 4-entry completed store queue between the LSU and the L1 data cache
- Separate additional queues for efficient buffering of outbound data (such as cast outs and write throughs) from the L1 data cache and L2

- Multiprocessing support features include the following:
  - Hardware-enforced, cache coherency protocols for data cache
    - 3-state (MEI) similar to the MPC750
    - 4-state (MESI) similar to the MPC604
    - 5-state (MERSI), where the new R state allows shared intervention
  - Load/store with reservation instruction pair for atomic memory references, semaphores, and other multiprocessor operations
- Power and thermal management
  - Three static modes, doze, nap, and sleep, progressively reduce power dissipation:
    - Doze—All the functional units are disabled except for the time base/decrementer registers and the bus snooping logic.
    - Nap—The nap mode further reduces power consumption by disabling bus snooping, leaving only the time base register and the PLL in a powered state.
    - Sleep—All internal functional units are disabled, after which external system logic may disable the PLL and SYSCLK.
  - Thermal management facility provides software-controllable thermal management. Thermal management is performed through the use of three supervisor-level registers and an MPC7410-specific thermal management exception.
  - Instruction cache throttling provides control of instruction fetching to limit power consumption.
- Performance monitor can be used to help debug system designs and improve software efficiency.
- In-system testability and debugging features through JTAG boundary-scan capability

2.2 Instruction Flow

As shown in Figure 1, the MPC7410 instruction unit provides centralized control of instruction flow to the execution units. The instruction unit contains a sequential fetcher, six-entry instruction queue (IQ), dispatch unit, and BPU. It determines the address of the next instruction to be fetched based on information from the sequential fetcher and from the BPU.

The sequential fetcher loads instructions from the instruction cache into the instruction queue. The BPU extracts branch instructions from the sequential fetcher. Branch instructions that cannot be resolved immediately are predicted using either the MPC7410-specific dynamic branch prediction or the architecture-defined static branch prediction.

Branch instructions that do not affect the LR or CTR are removed from the instruction stream. The BPU folds branch instructions when a branch is taken (or predicted as taken); branch instructions that are not taken, or predicted as not taken, are removed from the instruction stream through the dispatch mechanism.

Instructions issued beyond a predicted branch do not complete execution until the branch is resolved, preserving the programming model of sequential execution. If branch prediction is incorrect, the instruction unit flushes all predicted path instructions, and instructions are fetched from the correct path.

2.2.1 Instruction Queue and Dispatch Unit

The instruction queue (IQ), shown in Figure 1, holds as many as six instructions and loads up to four instructions from the instruction cache during a single processor clock cycle. The instruction fetcher continuously attempts to load as many instructions as there were vacancies in the IQ in the previous clock cycle. All instructions except branch, Return from Exception (**rfi**), System Call (**sc**), and Instruction Synchronize (**isync**) instructions are dispatched to their respective execution units from the bottom two positions in the instruction queue (IQ0 and IQ1) at a maximum rate of two instructions per cycle. Reservation stations are provided for the IU1, IU2, FPU, LSU, SRU, VPU, and VALU. The dispatch unit checks for source and destination register dependencies, determines whether a position is available in the completion queue, and inhibits subsequent instruction dispatching as required.

Branch instructions can be detected, decoded, and predicted from anywhere in the instruction queue.

2.2.2 Branch Processing Unit (BPU)

The BPU receives branch instructions from the sequential fetcher and performs CR lookahead operations on conditional branches to resolve them early, achieving the effect of a zero-cycle branch in many cases.

Unconditional branch instructions and conditional branch instructions in which the condition is known can be resolved immediately. For unresolved conditional branch instructions, the branch path is predicted using either the architecture-defined static branch prediction or the MPC7410-specific dynamic branch prediction. Dynamic branch prediction is enabled if HID0[BHT] = 1.

When a prediction is made, instruction fetching, dispatching, and execution continue from the predicted path, but instructions cannot complete and write back results to architected registers until the prediction is determined to be correct (resolved). When a prediction is incorrect, the instructions from the incorrect path are flushed from the processor and processing begins from the correct path. The MPC7410 allows a second branch instruction to be predicted; instructions from the second predicted instruction stream can be fetched but cannot be dispatched.

Dynamic prediction is implemented using a 512-entry branch history table (BHT), a cache that provides two bits per entry that together indicate four levels of prediction for a branch instruction—not-taken, strongly not-taken, taken, strongly taken. When dynamic branch prediction is disabled, the BPU uses a bit in the instruction encoding to predict the direction of the conditional branch. Therefore, when an unresolved conditional branch instruction is encountered, the MPC7410 executes instructions from the predicted target stream although the results are not committed to architected registers until the conditional branch is resolved. This execution can continue until a second unresolved branch instruction is encountered.

When a branch is taken (or predicted as taken), the instructions from the untaken path must be flushed and the target instruction stream must be fetched into the IQ. The BTIC is a 64-entry, four-way set associative cache that contains the most recently used branch target instructions, typically in pairs. When an instruction fetch hits in the BTIC, the instructions arrive in the instruction queue in the next clock cycle, a clock cycle sooner than they would arrive from the instruction cache. Additional instructions arrive from the instruction cache in the next clock cycle. The BTIC reduces the number of missed opportunities to dispatch instructions and gives the processor a one-cycle head start on processing the target stream.

The BPU contains an adder to compute branch target addresses and three user-accessible registers—the link register (LR), the count register (CTR), and the condition register (CR). The BPU calculates the return pointer for subroutine calls and saves it into the LR for certain types of branch instructions. The LR also contains the branch target address for the Branch Conditional to Link Register (**bclr***x*) instruction. The CTR contains the branch target address for the Branch Conditional to Count Register (**bcctr***x*) instruction. Because the LR and CTR are SPRs, their contents can be copied to or from any GPR. Also, because the BPU uses dedicated registers rather than GPRs or FPRs, execution of branch instructions is largely independent from execution of integer and floating-point instructions.

2.2.3 Completion Unit

The completion unit operates closely with the instruction unit. Instructions are fetched and dispatched in program order. At the point of dispatch, the program order is maintained by assigning each dispatched instruction a successive entry in the eight-entry completion queue. The completion unit tracks instructions from dispatch through execution and retires them in program order from the two bottom entries in the completion queue (CQ0 and CQ1).

Instructions cannot be dispatched to an execution unit unless there is a vacancy in the completion queue. Branch instructions that do not update the CTR or LR are removed from the instruction stream and do not take an entry in the completion queue. Instructions that update the CTR and LR follow the same dispatch and completion procedures as non-branch instructions, except that they are not issued to an execution unit.

Completing an instruction commits execution results to architected registers (GPRs, FPRs, VRs, LR, and CTR). In-order completion ensures the correct architectural state when the MPC7410 must recover from a mispredicted branch or any exception. An instruction is retired as it is removed from the completion queue.

2.2.4 Independent Execution Units

In addition to the BPU, the MPC7410 provides the seven execution units described in the following sections.

2.2.4.1 AltiVec Vector Permute Unit (VPU)

The VPU performs the following permutations on vector operands:

- Pack—Vector pack instructions truncate the contents of two concatenated source operands (grouped as eight words or sixteen half words) into a single result of eight half words or sixteen bytes, respectively.
- Unpack—Vector unpack instructions unpack the eight low or high bytes (or four low or high half words) of one source operand into eight half words (or four words) using sign extension to fill the most significant bytes (MSBs).
- Merge—Byte vector merge instructions interleave the eight low bytes (or eight high bytes) from two source operands producing a result of 16 bytes. Similarly, half-word vector merge instructions interleave the four low half words (or four high half words) of two source operands producing a result of eight half words, and word vector merge instructions interleave the two low words (or two high words) from two source operands producing a result of four words. The vector merge instruction has many uses, and it can be used to efficiently transpose SIMD vectors.
- Splat—Vector splat instructions prepare vector data for performing operations for which one source vector is to consist of elements that all have the same value (for example, multiplying all elements of a vector register by a constant). Vector splat instructions also can move data. For example to multiply all elements of a vector register by a constant, the vector splat instructions can be used to splat the scalar into a vector register. Likewise, when storing a scalar into an arbitrary memory location, it must be splatted into a vector register, and that register must be specified as the source of the store. This guarantees that the data appears in all possible positions of that scalar size for the store.

- Permute—Permute instructions allow any byte in any two source vector registers to be directed to any byte in the destination vector. The fields in a third source operand specify from which field in the source operands the corresponding destination field is to be taken. The Vector Permute (**vperm**) instruction provides many useful functions. For example, it can be used efficiently to perform table lookups and data alignment. For an example of how to align data, see Section 3.1.6, "Quad-Word Data Alignment," in the *AltiVec Technology Programming Environments Manual*.
- Select—Data flow in the vector unit can be controlled without branching by using a vector compare instruction and the vector select (**vsel**) instruction. In this case, the compare result vector is used directly as a mask operand of a vector select instruction. The **vsel** instruction selects one field from one or the other of two source operands under control of its mask operand. Use of the TRUE/FALSE compare result vector with select in this manner produces a two-instruction equivalent of conditional execution on a per-field basis.

These instructions are described in detail in Chapter 2, "Addressing Modes and Instruction Set Summary," in the *AltiVec Technology Programming Environments Manual*.

2.2.4.2 AltiVec Vector Arithmetic Logic Unit (VALU)

As shown in Figure 1, the VALU consists of the following three independent subunits:

- Vector simple integer unit (VSIU)—executes simple vector integer computational instructions, such as addition, subtraction, maximum and minimum comparisons, averaging, rotation, shifting, comparisons, and Boolean operations
- Vector complex integer unit (VCIU)—executes longer-latency vector integer instructions, such as multiplication, division, multiplication/addition, and sum-across with saturation
- Vector floating-point unit (VFPU)—executes all vector floating-point instructions

Although only one instruction can be dispatched to the VALU per processor clock cycle, all three subunits can execute simultaneously. For example, if instructions are dispatched one at a time to the VFPU, VCIU, and VSIU, all three subunits can be executing separate instructions, and, if enough VR rename resources are available, two of them can write back their results in the same clock cycle.

2.2.4.3 Integer Units (IUs)

The integer units IU1 and IU2 are shown in Figure 1. IU1 can execute any integer instruction; IU2 can execute any integer instruction except multiplication and division instructions. Each IU has a single-entry reservation station that can receive instructions from the dispatch unit and operands from the GPRs or the rename buffers.

Each IU consists of three single-cycle subunits—a fast adder/comparator, a subunit for logical operations, and a subunit for performing rotates, shifts, and count-leading-zero operations. These subunits handle all one-cycle arithmetic instructions; only one subunit can execute an instruction at a time.

The IU1 has a 32-bit integer multiplier/divider as well as the adder, shift, and logical units of the IU2. The multiplier supports early exit for operations that do not require full 32- x 32-bit multiplication.

Each IU has a dedicated result bus (not shown in Figure 1) that connects to rename buffers.

2.2.4.4 Floating-Point Unit (FPU)

The FPU, shown in Figure 1, is designed such that single-precision operations require only a single pass, with a latency of three cycles. As instructions are dispatched to the FPU's reservation station, source operand data can be accessed from the FPRs or from the FPR rename buffers. Results in turn are written to the rename buffers and are made available to subsequent instructions. Instructions pass through the reservation station in dispatch order.

The FPU contains a single-precision multiply-add array and the floating-point status and control register (FPSCR). The multiply-add array allows the MPC7410 to efficiently implement multiply and multiply-add operations. The FPU is pipelined so that one single- or double-precision instruction can be issued per clock cycle. Note that an execution bubble may occur after three consecutive, independent floating-point arithmetic instructions to allow for a normalization special case. Thirty-two 64-bit floating-point registers are provided to support floating-point operations. Stalls due to contention for FPRs are minimized by automatic allocation of the six floating-point rename registers. The MPC7410 writes the contents of the rename registers to the appropriate FPR when floating-point instructions are retired by the completion unit.

The MPC7410 supports all IEEE 754 floating-point data types (normalized, denormalized, NaN, zero, and infinity) in hardware, eliminating the latency incurred by software exception routines.

2.2.4.5 Load/Store Unit (LSU)

The LSU executes all load and store instructions as well as the AltiVec LRU and transient instructions and provides the data transfer interface between the GPRs, FPRs, VRs, and the cache/memory subsystem. The LSU calculates effective addresses, performs data alignment, and provides sequencing for load/store string and multiple instructions.

Load and store instructions are issued and translated in program order; however, some memory accesses can occur out of order. Synchronizing instructions can be used to enforce strict ordering. When there are no data dependencies and the guarded bit for the page or block is cleared, a maximum of one out-of-order cacheable load operation can execute per cycle from the perspective of the LSU, with a two-cycle total latency on a cache hit. Data returned from the cache is held in a rename register until the completion logic commits the value to a GPR, FPR, or VR. Stores cannot be executed out of order and are held in the store queue until the completion logic signals that the store operation is to be completed to memory. The MPC7410 executes store instructions with a maximum throughput of one per cycle and a three-cycle total latency to the data cache. The time required to perform the actual load or store operation depends on the processor/bus clock ratio and whether the operation involves the on-chip cache, the L2 cache, system memory, or an I/O device.

2.2.4.6 System Register Unit (SRU)

The SRU executes various system-level instructions, as well as condition register logical operations and move to/from special-purpose register instructions. To maintain system state, most instructions executed by the SRU are execution-serialized; that is, the instruction is held for execution in the SRU until all previously issued instructions have executed. Results from execution-serialized instructions executed by the SRU are not available or forwarded for subsequent instructions until the instruction completes.

2.3 Memory Management Units (MMUs)

The MPC7410's MMUs support up to 4 Petabytes (2^{52}) of virtual memory and 4 Gigabytes (2^{32}) of physical memory for instructions and data. The MMUs control access privileges for these spaces on block and page granularities. Referenced and changed status is maintained by the processor for each page to support demand-paged virtual memory systems.

The LSU calculates effective addresses for data loads and stores; the instruction unit calculates effective addresses for instruction fetching. The MMU translates the effective address to determine the correct physical address for the memory access.

The MPC7410 supports the following types of memory translation:

- Real addressing mode—In this mode, translation is disabled by clearing bits in the machine state register (MSR): MSR[IR] for instruction fetching or MSR[DR] for data accesses. When address translation is disabled, the physical address is identical to the effective address.
- Page address translation—translates the page frame address for a 4-Kbyte page size
- Block address translation—translates the base address for blocks (128 Kbytes to 256 Mbytes)

If translation is enabled, the appropriate MMU translates the higher-order bits of the effective address into physical address bits. The lower-order address bits (that are untranslated and therefore, considered both logical and physical) are directed to the on-chip caches where they form the index into the eight-way set-associative tag array. After translating the address, the MMU passes the higher-order physical address bits to the cache and the cache lookup completes. For caching-inhibited accesses or accesses that miss in the cache, the untranslated lower-order address bits are concatenated with the translated higher-order address bits; the resulting 32-bit physical address is used by the memory subsystem and the bus interface unit, which accesses external memory.

The TLBs store page address translations for recent memory accesses. For each access, an effective address is presented for page and block translation simultaneously. If a translation is found in both the TLB and the BAT array, the block address translation in the BAT array is used. Usually if the BAT array lookup results in a miss, the translation is in a TLB and the physical address is readily available to the on-chip cache. When a page address translation is not in a TLB, hardware searches for one in the page table following the model defined by the PowerPC architecture.

Instruction and data TLBs provide address translation in parallel with the on-chip cache access, incurring no additional time penalty in the event of a TLB hit. The MPC7410's instruction and data TLBs are 128-entry, two-way set-associative caches that contain address translations. The MPC7410 automatically generates a search of the page tables in memory on a TLB miss.

2.4 On-Chip Instruction and Data Caches

The MPC7410 implements separate L1 instruction and data caches. Each cache is 32-Kbyte and eight-way set associative. As defined by the PowerPC architecture, they are physically indexed. Each cache block contains eight contiguous words from memory that are loaded from an 8-word boundary (that is, bits EA[27–31] are zeros); thus, a cache block never crosses a page boundary. An entire cache block can be updated by a four-beat burst load across a 64-bit system bus. Misaligned accesses across a page boundary can incur a performance penalty. The data cache is a nonblocking, write-back cache with hardware support for reloading on cache misses. The critical double word is transferred on the first beat and is simultaneously written to the cache and forwarded to the requesting unit, minimizing stalls due to load delays. The cache being loaded is not blocked to internal accesses while the load completes.

The MPC7410 cache organization is shown in Figure 2.

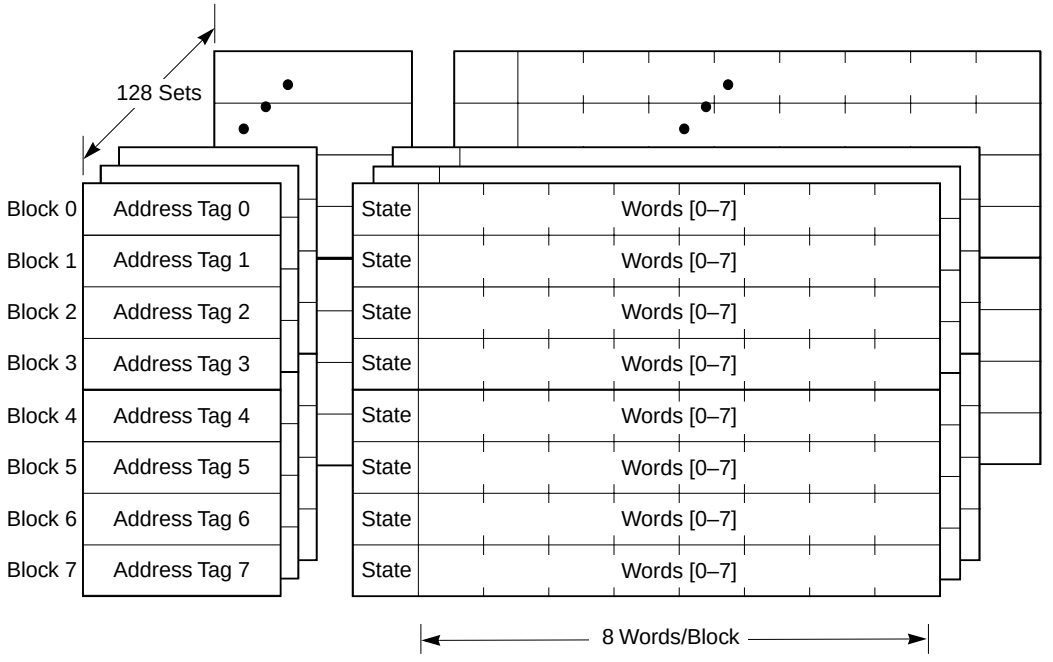


Figure 2. L1 Cache Organization

The instruction cache provides up to four instructions per cycle to the instruction queue. The instruction cache can be invalidated entirely or on a cache-block basis. It is invalidated and disabled by setting HID0[ICFI] and then clearing HID0[ICE]. The instruction cache can be locked by setting HID0[ILOCK]. The instruction cache supports only the valid/invalid states.

The data cache provides four words per cycle to the LSU. Like the instruction cache, the data cache can be invalidated all at once or on a per-cache-block basis. The data cache can be invalidated and disabled by setting HID0[DCFI] and then clearing HID0[DCE]. The data cache can be locked by setting HID0[DLOCK]. The data cache tags are dual-ported, so a load or store can occur simultaneously with a snoop.

The MPC7410 also implements a 64-entry (16-set, four-way set-associative) branch target instruction cache (BTIC). The BTIC is a cache of branch instructions that have been encountered in branch/loop code sequences. If the target instruction is in the BTIC, it is fetched into the instruction queue a cycle sooner than it can be made available from the instruction cache. Typically the BTIC contains the first two instructions in the target stream. The BTIC can be disabled and invalidated through software.

2.5 L2 Cache Implementation

The L2 cache is a unified cache that receives memory requests from both the L1 instruction and data caches independently. The L2 cache is implemented with an on-chip, two-way, set-associative tag memory, and with external, synchronous SRAMs for data storage. The external SRAMs are accessed through a dedicated L2 cache port that supports a single bank of 256-Kbyte, 512-Kbyte, 1-Mbyte, or 2-Mbyte synchronous SRAMs. The L2 cache normally operates in write-back mode and supports system cache coherency through snooping.

Depending on its size, the L2 cache is organized into 32-, 64-, or 128-byte lines. Lines are subdivided into 32-byte sectors (blocks), the unit at which cache coherency is maintained.

The L2 cache controller contains the L2 cache control register (L2CR), which includes bits for enabling parity checking, setting the L2-to-processor clock ratio, and identifying the type of RAM used for the L2 cache implementation. The L2 cache controller also manages the L2 cache tag array, which is two-way set-associative with 8K tags per way. Each sector (32-byte cache block) has its own valid, shared, and modified status bits. The L2 implements the MERSI protocol using three status bits per sector.

Requests from the L1 cache generally result from instruction misses, data load or store misses, write-through operations, or cache management instructions. Requests from the L1 cache are compared against the L2 tags and serviced by the L2 cache if they hit; they are forwarded to the bus interface if they miss.

The L2 cache can accept multiple, simultaneous accesses. The L1 instruction cache can request an instruction at the same time that the L1 data cache is requesting data. The L1 data cache requests are handled through the data reload table (shown in Figure 1), which can have up to eight outstanding data cache misses. The L2 cache also services snoop requests from the bus. If there are multiple pending requests to the L2 cache, snoop requests have highest priority. The next priority are load and store requests from the L1 data cache. The next priority are instruction fetch requests from the L1 instruction cache.

Alternately, the L2 interface can be configured to use half (256 Kbytes minimum) or all of the SRAM area as a direct-mapped, private memory space. The private memory space provides a low-latency, high-bandwidth area for critical data or instructions. Accesses to the private memory space do not propagate to the L2 cache nor are they visible to the external system bus. The private memory space is also not snooped, so the coherency of its contents must be maintained by software or not at all.

2.6 System Interface/Bus Interface Unit (BIU)

The MPC7410 processor bus interface is based on the 60x bus, but it includes several features that allow it to provide significantly higher memory bandwidth. The MPC7410 can be configured to support either an MPC750-compatible 60x mode or an expanded bus mode called MPX bus mode.

The MPC7410 has a separate address and data bus, each with its own set of arbitration and control signals. This allows for the decoupling of the data tenure from the address tenure of a transaction, and provides for a wide range of system bus implementations including:

- Non-pipelined bus operation
- Pipelined bus operation
- Split transaction operation
- Enveloped transaction operation

The MPC7410 supports only the normal memory-mapped address segments defined in the PowerPC architecture.

The 60x bus interface has the following features:

- 32-bit address bus (plus 4 bits of odd parity)
- 64-bit data bus (plus 8 bits of odd parity); a 32-bit data bus mode is not provided
- Supports two cache coherency protocols:
  - Three-state (MEI) similar to the MPC750
  - Four-state (MESI) similar to the MPC604 processors
- On-chip snooping to maintain L1 data cache and L2 cache coherency for multiprocessing applications

- Supports address-only transfers (useful for a variety of broadcast operations in multiprocessor applications)
- Support for limited out-of-order transactions
- Support for up to seven transactions (six pending plus one data tenure in progress)
- TTL-compatible interface

In addition to the 60x bus features, to gain increased performance, the MPX bus mode has the following features:

- Increased address bus bandwidth by eliminating dead cycles under some circumstances
- Full data streaming for burst reads and burst writes
- Increased levels of address pipelining
- Support for full out-of-order transactions
- Support for data intervention in multiprocessing systems
- Support for third cache coherency protocol: Five-state (MERSI), where the new R state allows shared intervention
- Improved electrical timings (for example, programmable option for keeping address bus driven)

2.6.1 System Interface Operation

The primary activity of the MPC7410 system interface is transferring data and instructions between the processor and system memory. There are three types of 60x bus memory accesses:

- Single-beat transfers—These memory accesses allow transfer sizes of 8, 16, 24, 32, or 64 bits in one bus clock cycle. Single-beat transactions are caused by uncacheable read and write operations that access memory directly (that is, when caching is disabled), cache-inhibited accesses, and stores in write-through mode.
- Two-beat burst (16 bytes) data transfers—Generated to support caching-inhibited or write-through AltiVec loads and stores (only generated in MPX bus mode in MPC7410).
- Four-beat burst (32 bytes) data transfers—Initiated when an entire cache block is transferred. Because the first-level caches on the MPC7410 are write-back caches, burst-read memory operations are the most common memory accesses, followed by burst-write memory operations, and single-beat (noncacheable or write-through) memory read and write operations.

The MPC7410 also supports address-only operations, variants of the burst and single-beat operations (for example, atomic memory operations and global memory operations that are snooped), and address retry activity (for example, when a snooped read access hits a modified block in the cache). Because all I/O is memory-mapped, I/O accesses use the same protocol as memory accesses. The MPX bus also supports data-only operations to provide data intervention in systems that use the MERSI protocol.

Access to the system interface is granted through an external arbitration mechanism that allows devices to compete for bus mastership. This arbitration mechanism is flexible, allowing the MPC7410 to be integrated into systems that implement various fairness and bus parking procedures to avoid arbitration overhead.

Typically, memory accesses are weakly ordered—sequences of operations, including load/store string and multiple instructions, do not necessarily execute in the order they begin—maximizing the efficiency of the bus without sacrificing data coherency. The MPC7410 allows read operations to be performed ahead of store operations (except when a dependency exists, or in cases where a noncacheable access is performed). The MPC7410 provides support for a write operation to be performed ahead of a previously queued read data tenure (for example, letting a snoop push be enveloped between address and data tenures of a read operation) in 60x bus mode and full data tenure reordering in MPX bus mode. Because the MPC7410 can dynamically optimize run-time ordering of load/store traffic, overall performance is improved.

The system interface supports address pipelining, which allows the address tenure of one transaction to overlap the data tenure of another. The extent of the pipelining depends on external arbitration and control circuitry. Similarly, the MPC7410 supports split-bus transactions for systems with multiple potential bus masters—one device can have masterMPX bus protocolsship of the address bus while another has mastership of the data bus. Allowing multiple bus transactions to occur simultaneously increases the available bus bandwidth for other activity.

The system interface is specific for each PowerPC microprocessor implementation.

2.6.2 Signal Groupings

The MPC7410 signals are grouped as shown in Figure 3. Signals are provided for implementing the bus protocol, clocking and control of the L2 caches, as well as separate L2 address and data buses. Test and control signals provide diagnostics for selected internal circuits.

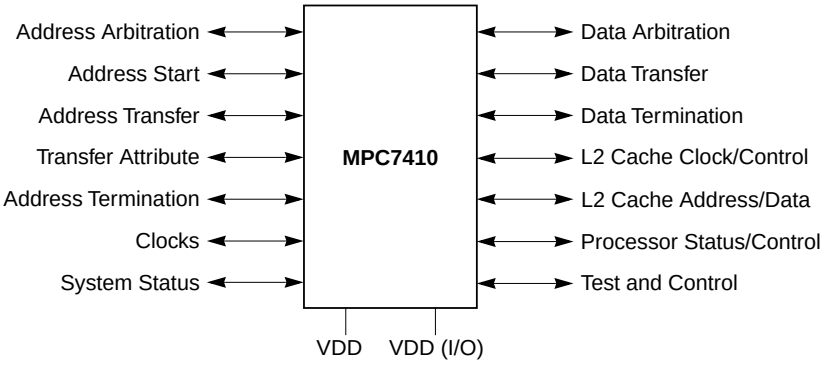


Figure 3. System Interface

The signals used for the 60x and the MPX bus protocols are largely identical except that the MPX bus differs in the following ways:

- Does not use the $\overline{\text{ABB}}$ and $\overline{\text{DBB}}$ output signals
- Uses three DTI[0:2] signals instead of a single $\overline{\text{DBWO}}$ signal
- Uses two $\overline{\text{SHD}}$[0:1] signals instead of a single $\overline{\text{SHD}}$ signal

The MPC7410 bus protocol signals are grouped as follows:

- Address arbitration signals—The MPC7410 uses these signals to arbitrate for address bus mastership.
- Address start signals—These signals indicate that a bus master has begun a transaction on the address bus.
- Address transfer signals—These signals include the address bus and address parity signals. They are used to transfer the address and to ensure the integrity of the transfer.
- Transfer attribute signals—These signals provide information about the type of transfer, such as the transfer size and whether the transaction is bursted, write-through, or caching-inhibited.
- Address termination signals—These signals are used to acknowledge the end of the address phase of the transaction. They also indicate whether a condition exists that requires the address phase to be repeated.
- Data arbitration signals—The MPC7410 uses these signals to arbitrate for data bus mastership.

- Data transfer signals—These signals, which consist of the data bus and data parity signals, are used to transfer the data and to ensure the integrity of the transfer.
- Data termination signals—Data termination signals are required after each data beat in a data transfer. In a single-beat transaction, a data termination signal also indicates the end of the tenure; in burst accesses, data termination signals apply to individual beats and indicate the end of the tenure only after the final data beat. They also indicate whether a condition exists that requires the data phase to be repeated.

The remaining signals are used for functions other than the bus protocol and they are grouped as follows:

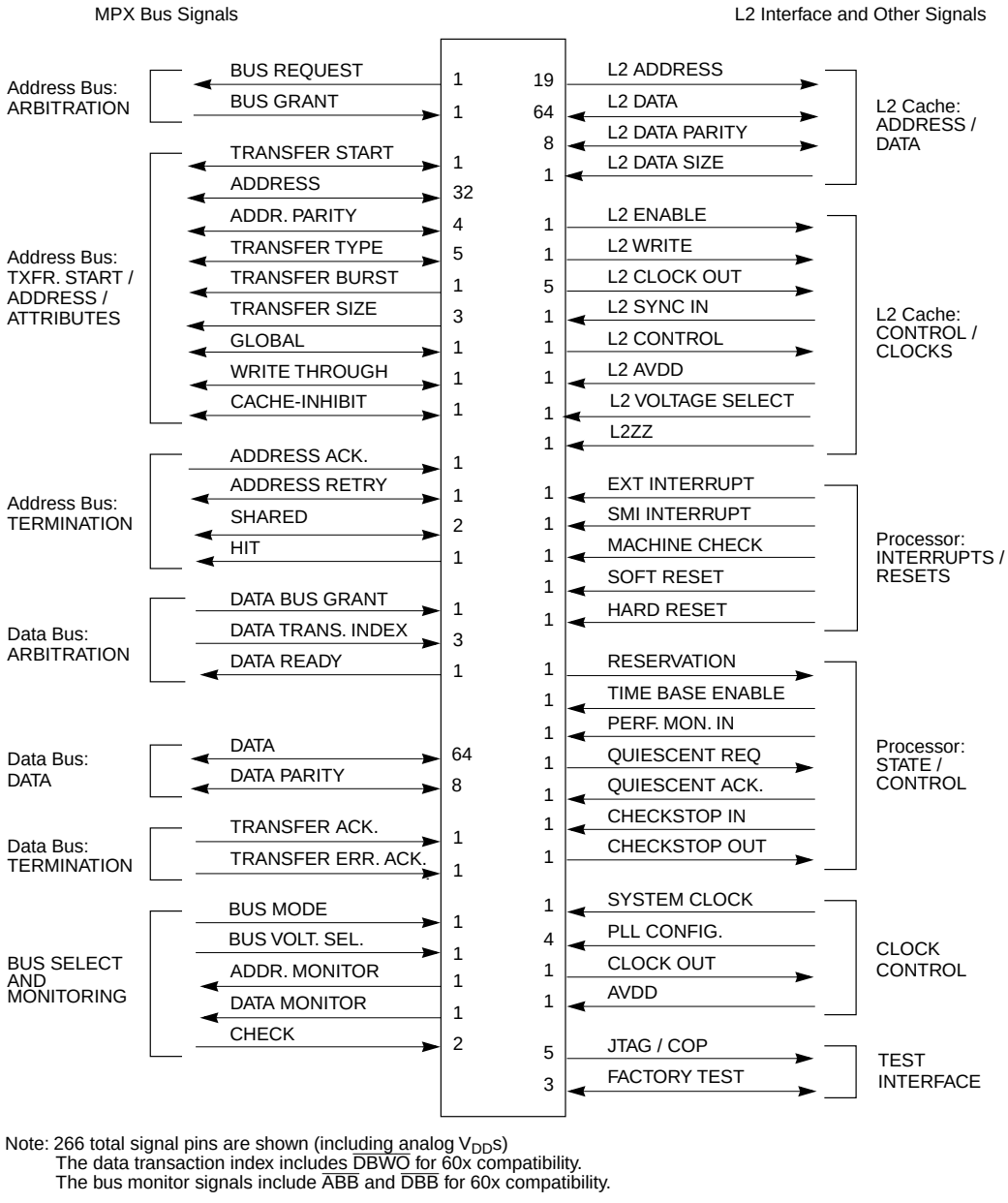
- L2 cache clock/control signals—These signals provide clocking and control for the L2 cache.
- L2 cache address/data—The MPC7410 has separate address and data buses for accessing the L2 cache.
- Interrupt and reset signals—These signals include the interrupt signal, checkstop signals, and both soft reset and hard reset signals. These signals are used to generate interrupt exceptions and, under various conditions, to reset the processor.
- Processor status/control signals—These signals are used to set the reservation coherency bit, enable the time base, and other functions.
- Miscellaneous signals—These signals are used in conjunction with resources such as the time base facility.
- JTAG/COP interface signals—The common on-chip processor (COP) unit provides a serial interface to the system for performing board-level boundary scan interconnect tests.
- Clock signals—These signals determine the system clock frequency. These signals can also be used to synchronize multiprocessor systems.

NOTE:

Active-low signals are shown with overbars—for example, $\overline{\text{ARTRY}}$ (address retry) and $\overline{\text{TS}}$ (transfer start). Active-low signals are referred to as asserted (active) when they are low and negated when they are high. Signals that are not active low, such as AP[0:3] (address bus parity signals) and TT[0:4] (transfer type signals) are referred to as asserted when they are high and negated when they are low.

2.6.2.1 Signal Configuration

Figure 4 shows the MPC7410's logical pin configuration. The signals are grouped by function.



Note: 266 total signal pins are shown (including analog V_{DD}s)
The data transaction index includes $\overline{\text{DBWO}}$ for 60x compatibility.
The bus monitor signals include $\overline{\text{ABB}}$ and $\overline{\text{DBB}}$ for 60x compatibility.

Figure 4. MPC7410 Microprocessor Signal Groups

2.6.2.2 Clocking

For functional operation, the MPC7410 uses a single clock input signal, SYSCLK, from which clocking is derived for the processor core, the L2 interface, and the MPX bus interface. Additionally, internal clock information is made available at the pins to support debug and development.

The MPC7410's clocking structure supports a wide range of processor-to-bus clock ratios. The internal processor core clock is synchronized to SYSCLK with the aid of a VCO-based PLL. The PLL_CFG[0–3] signals are used to program the internal clock rate to a multiple of SYSCLK as defined in the MPC7410

hardware specification. The bus clock is maintained at the same frequency as SYSCLK. SYSCLK does not need to be a 50% duty-cycle signal.

The MPC7410 generates the clock for the external L2 synchronous data RAMs. The clock frequency for the RAMs is divided down from (and phase-locked to) the core clock frequency of the MPC7410. The core-to-L2 frequency divisor for the L2 PLL is selected through L2CR[L2CLK].

Part III MPC7410 Microprocessor: Implementation

The PowerPC architecture is derived from the POWER™ architecture (Performance Optimized with Enhanced RISC architecture). The PowerPC architecture shares the benefits of the POWER architecture optimized for single-chip implementations. The PowerPC architecture design facilitates parallel instruction execution and is scalable to take advantage of future technological gains.

This section describes the PowerPC architecture in general, and specific details about the implementation of the MPC7410 as a low-power, 32-bit member of the PowerPC processor family. The structure of this section follows the organization of the user's manual; each subsection provides an overview of each chapter

- Registers and programming model—Section 3.1, "PowerPC Registers and Programming Model," describes the registers for the operating environment architecture common among PowerPC processors and describes the programming model. It also describes the registers that are unique to the MPC7410.
- Instruction set and addressing modes—Section 3.2, "Instruction Set," describes the PowerPC instruction set and addressing modes for the PowerPC operating environment architecture, and defines and describes the PowerPC instructions implemented in the MPC7410.
- Cache implementation—Section 3.3, "On-Chip Cache Implementation," describes the cache model that is defined generally for PowerPC processors by the virtual environment architecture. It also provides specific details about the MPC7410 cache implementation.
- Exception model—Section 3.4, "Exception Model," describes the exception model of the PowerPC operating environment architecture and the differences in the MPC7410 exception model.
- Memory management—Section 3.5, "Memory Management," describes generally the conventions for memory management among the PowerPC processors. This section also describes the MPC7410's implementation of the 32-bit PowerPC memory management specification.
- Instruction timing—Section 3.6, "Instruction Timing," provides a general description of the instruction timing provided by the superscalar, parallel execution supported by the PowerPC architecture and the MPC7410.
- Power management—Section 3.7, "Power Management," describes how the power management can be used to reduce power consumption when the processor, or portions of it, are idle.
- Thermal management—Section 3.8, "Thermal Management," describes how the thermal management unit and its associated registers (THRM1–THRM3) and exception can be used to manage system activity in a way that prevents exceeding system and junction temperature thresholds. This is particularly useful in high-performance portable systems, which cannot use the same cooling mechanisms (such as fans) that control overheating in desktop systems.
- Performance monitor—Section 3.9, "Performance Monitor," describes the performance monitor facility, which system designers can use to help bring up, debug, and optimize software performance.

The PowerPC architecture consists of the following layers, and adherence to the PowerPC architecture can be described in terms of which of the following levels of the architecture is implemented:

- PowerPC user instruction set architecture (UISA)—Defines the base user-level instruction set, user-level registers, data types, floating-point exception model, memory models for a uniprocessor environment, and programming model for a uniprocessor environment.
- PowerPC virtual environment architecture (VEA)—Describes the memory model for a multiprocessor environment, defines cache control instructions, and describes other aspects of virtual environments. Implementations that conform to the VEA also adhere to the UISA, but may not necessarily adhere to the OEA.
- PowerPC operating environment architecture (OEA)—Defines the memory management model, supervisor-level registers, synchronization requirements, and the exception model. Implementations that conform to the OEA also adhere to the UISA and the VEA.

The MPC7410 implementation supports the three levels of the architecture described above. For more information about the PowerPC architecture, see *PowerPC Microprocessor Family: The Programming Environments*.

Specific features of the MPC7410 are listed in Part II, “MPC7410 Microprocessor Features.”

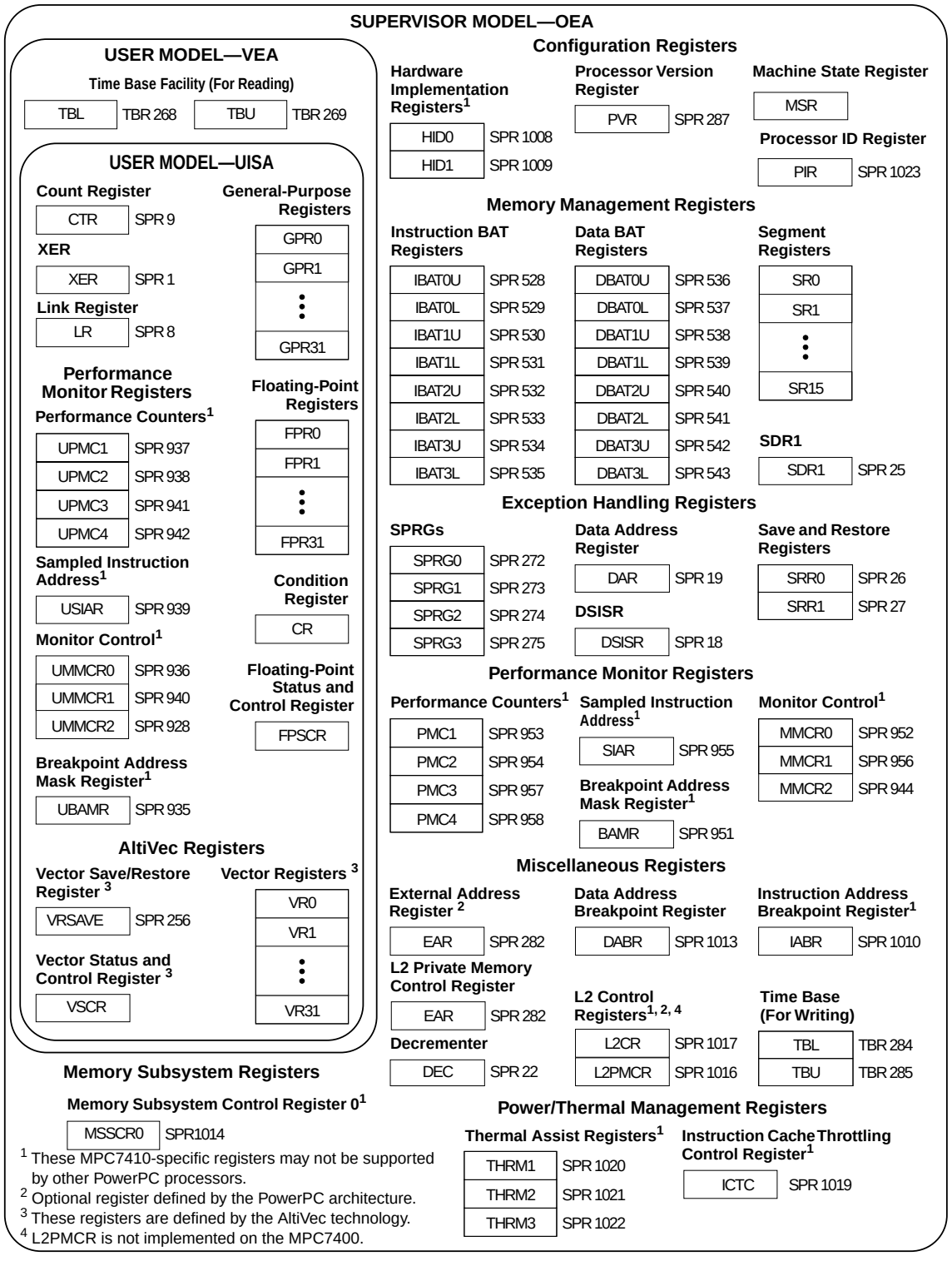
3.1 PowerPC Registers and Programming Model

The PowerPC architecture defines register-to-register operations for most computational instructions. Source operands for these instructions are accessed from the registers or are provided as immediate values embedded in the instruction opcode. The three-register instruction format allows specification of a target register distinct from the two source operands. Load and store instructions transfer data between registers and memory.

PowerPC processors have two levels of privilege—supervisor mode of operation (typically used by the operating system) and user mode of operation (used by the application software). The programming models incorporate 32 GPRs, 32 FPRs, special-purpose registers (SPRs), and several miscellaneous registers. The AltiVec extensions to the PowerPC architecture augment the programming model with 32 VRs, one status and control register, and one save and restore register. Each PowerPC microprocessor also has its own unique set of implementation-specific registers to support functionality that may not be defined by the PowerPC architecture.

Having access to privileged instructions, registers, and other resources allows the operating system to control the application environment (providing virtual memory and protecting operating-system and critical machine resources). Instructions that control the state of the processor, the address translation mechanism, and supervisor registers can be executed only when the processor is operating in supervisor mode.

Figure 5 shows all the MPC7410 registers available at the user and supervisor level. The numbers to the right of the SPRs indicate the number that is used in the syntax of the instruction operands to access the register.



1 These MPC7410-specific registers may not be supported by other PowerPC processors.
2 Optional register defined by the PowerPC architecture.
3 These registers are defined by the AltiVec technology.
4 L2PMCR is not implemented on the MPC7400.

Figure 5. MPC7410 Microprocessor Programming Model—Registers

The following tables summarize the PowerPC registers implemented in the MPC7410; Table 1 describes registers (excluding SPRs) defined by the PowerPC architecture.

Table 1. PowerPC Architecture-Defined Registers on the MPC7410 (Excluding SPRs)

Register	Level	Function
CR	User	The condition register (CR) consists of eight four-bit fields that reflect the results of certain operations, such as move, integer and floating-point compare, arithmetic, and logical instructions, and provide a mechanism for testing and branching.
FPRs	User	The 32 floating-point registers (FPRs) serve as the data source or destination for floating-point instructions. These 64-bit registers can hold either single- or double-precision floating-point values.
FPSCR	User	The floating-point status and control register (FPSCR) contains the floating-point exception signal bits, exception summary bits, exception enable bits, and rounding control bits needed for compliance with the IEEE-754 standard.
GPRs	User	The 32 GPRs serve as the data source or destination for integer instructions.
MSR	Supervisor	The machine state register (MSR) defines the processor state. Its contents are saved when an exception is taken and restored when exception handling completes. The MPC7410 implements MSR[POW], (defined by the architecture as optional), which is used to enable the power management feature. The MPC7410-specific MSR[PM] bit is used to mark a process for the performance monitor.
SR0–SR15	Supervisor	The sixteen 32-bit segment registers (SRs) define the 4-Gbyte space as sixteen 256-Mbyte segments. The MPC7410 implements segment registers as two arrays—a main array for data accesses and a shadow array for instruction accesses; see Figure 1. Loading a segment entry with the Move to Segment Register (**mtsr**) instruction loads both arrays. The **mfsr** instruction reads the master register, shown as part of the data MMU in Figure 1.

The OEA defines numerous special-purpose registers that serve a variety of functions, such as providing controls, indicating status, configuring the processor, and performing special operations. During normal execution, a program can access the registers, shown in Figure 5, depending on the program's access privilege (supervisor or user, determined by the privilege-level (PR) bit in the MSR). GPRs and FPRs are accessed through operands that are part of the instructions. Access to registers can be explicit (that is, through the use of specific instructions for that purpose such as Move to Special-Purpose Register (**mtspr**) and Move from Special-Purpose Register (**mfspr**) instructions) or implicit, as the part of the execution of an instruction. Some registers can be accessed both explicitly and implicitly.

In the MPC7410, all SPRs are 32 bits wide. Table 2 describes the architecture-defined SPRs implemented by the MPC7410. *The Programming Environments Manual* describes these registers in detail, including bit descriptions.

Table 2. PowerPC Architecture-Defined SPRs Implemented by the MPC7410

Register	Level	Function
LR	User	The link register (LR) can be used to provide the branch target address and to hold the return address after branch and link instructions.
BATs	Supervisor	The architecture defines 16 block address translation (BAT) registers, which operate in pairs. There are four pairs of data BATs (DBATs) and four pairs of instruction BATs (IBATs). BATs are used to define and configure blocks of memory.
CTR	User	The count register (CTR) is decremented and tested by branch-and-count instructions.
DABR	Supervisor	The optional data address breakpoint register (DABR) supports the data address breakpoint facility.

Table 2. PowerPC Architecture-Defined SPRs Implemented by the MPC7410 (Continued)

Register	Level	Function
DAR	User	The data address register (DAR) holds the address of an access after an alignment or DSI exception.
DEC	Supervisor	The decrementer register (DEC) is a 32-bit decrementing counter that provides a way to schedule decrementer exceptions.
DSISR	User	The DSISR defines the cause of data access and alignment exceptions.
EAR	Supervisor	The external access register (EAR) controls access to the external access facility through the External Control In Word Indexed (**eciwx**) and External Control Out Word Indexed (**ecowx**) instructions.
PIR	Supervisor	The processor ID register (PIR) is used to differentiate between processors in a multiprocessor system.
PVR	Supervisor	The processor version register (PVR) is a read-only register that identifies the processor.
SDR1	Supervisor	SDR1 specifies the page table format used in virtual-to-physical page address translation.
SRR0	Supervisor	The machine status save/restore register 0 (SRR0) saves the address used for restarting an interrupted program when a Return from Interrupt (**rfi**) instruction executes.
SRR1	Supervisor	The machine status save/restore register 1 (SRR1) is used to save machine status on exceptions and to restore machine status when an **rfi** instruction is executed.
SPRG0–SPRG3	Supervisor	SPRG0–SPRG3 are provided for operating system use.
TB	User: read Supervisor: read/write	The time base register (TB) is a 64-bit register that maintains the time of day and operates interval timers. The TB consists of two 32-bit fields—time base upper (TBU) and time base lower (TBL).
XER	User	The XER contains the summary overflow bit, integer carry bit, overflow bit, and a field specifying the number of bytes to be transferred by a Load String Word Indexed (**lswx**) or Store String Word Indexed (**stswx**) instruction.

Table 3 describes the registers defined by the AltiVec technology.

Table 3. AltiVec-Specific Registers

Register	Level	Function
VRs	User	The 32 vector registers (VRs) serve as the data source or destination for AltiVec instructions.
VSCR	User	The 32-bit vector status and control register (VSCR). A 32-bit vector register that is read and written in a manner similar to the FPSCR.
VRSAVE	User	The 32-bit vector save (VRSAVE) register is defined by the AltiVec technology to assist application and operating system software in saving and restoring the architectural state across process context-switched events.

Table 4 describes the supervisor-level SPRs in the MPC7410 that are not defined by the PowerPC architecture.

Table 4. MPC7410-Specific Registers

Register	Level	Function
BAMR	Supervisor	Breakpoint address mask register is used in conjunction with the events that monitor IABR and DABR hits.
HID0	Supervisor	The hardware implementation-dependent register 0 (HID0) provides checkstop enables and other functions.
HID1	Supervisor	The hardware implementation-dependent register 1 (HID1) allows software to read the configuration of the PLL configuration signals.
IABR	Supervisor	The instruction address breakpoint register (IABR) supports instruction address breakpoint exceptions. It can hold an address to compare with instruction addresses in the IQ. An address match causes an instruction address breakpoint exception.
ICTC	Supervisor	The instruction cache-throttling control register (ICTC) has bits for controlling the interval at which instructions are fetched into the instruction queue in the instruction unit. This helps control the MPC7410's overall junction temperature.
L2CR	Supervisor	The L2 cache control register (L2CR) is used to configure and operate the L2 cache. It has bits for enabling parity checking, setting the L2-to-processor clock ratio, and identifying the type of RAM used for the L2 cache implementation.
L2PMCR	Supervisor	The L2 private memory control register (L2PMCR) is used to configure the private memory function of the L2 interface. This register is not implemented on the MPC7400.
MMCR0–MMCR2	Supervisor	The monitor mode control registers (MMCR0–MMCR1) are used to enable various performance monitoring interrupt functions. UMMCR0–UMMCR1 provide user-level read access to MMCR0–MMCR1.
MSSCR0	Supervisor	The memory subsystem control register is used to configure and operate the memory subsystem.
PMC1–PMC4	Supervisor	The performance monitor counter registers (PMC1–PMC4) are used to count specified events. UPMC1–UPMC4 provide user-level read access to these registers.
SIA	Supervisor	The sampled instruction address register (SIA) holds the EA of an instruction executing at or around the time the processor signals the performance monitor interrupt condition. The USIA register provides user-level read access to the SIA.
THRM1, THRM2	Supervisor	THRM1 and THRM2 provide a way to compare the junction temperature against two user-provided thresholds. The thermal assist unit (TAU) can be operated so that the thermal sensor output is compared to only one threshold, selected in THRM1 or THRM2.
THRM3	Supervisor	THRM3 is used to enable the TAU and to control the output sample time.
UBAMR	User	The user breakpoint address mask register (UBAMR) provides user-level read access to BAMR.
UMMCR0–UMMCR2	User	The user monitor mode control registers (UMMCR0–UMMCR1) provide user-level read access to MMCR0–MMCR2.
UPMC1–UPMC4	User	The user performance monitor counter registers (UPMC1–UPMC4) provide user-level read access to PMC1–PMC4.
USIA	User	The user sampled instruction address register (USIA) provides user-level read access to the SIA register.

3.2 Instruction Set

All PowerPC instructions are encoded as single-word (32-bit) opcodes. Instruction formats are consistent among all instruction types, permitting efficient decoding to occur in parallel with operand accesses. This fixed instruction length and consistent format greatly simplifies instruction pipelining.

3.2.1 PowerPC Instruction Set

The PowerPC instructions are divided into the following categories:

- Integer instructions—These include computational and logical instructions.
  - Integer arithmetic instructions
  - Integer compare instructions
  - Integer logical instructions
  - Integer rotate and shift instructions
- Floating-point instructions—These include floating-point computational instructions, as well as instructions that affect the FPSCR.
  - Floating-point arithmetic instructions
  - Floating-point multiply/add instructions
  - Floating-point rounding and conversion instructions
  - Floating-point compare instructions
  - Floating-point status and control instructions
- Load/store instructions—These include integer and floating-point load and store instructions.
  - Integer load and store instructions
  - Integer load and store multiple instructions
  - Floating-point load and store
  - Primitives used to construct atomic memory operations (**lwarx** and **stwcx.** instructions)
- Flow control instructions—These include branching instructions, condition register logical instructions, trap instructions, and other instructions that affect the instruction flow.
  - Branch and trap instructions
  - Condition register logical instructions
- Processor control instructions—These instructions are used for synchronizing memory accesses and management of caches, TLBs, and the segment registers.
  - Move to/from SPR instructions
  - Move to/from MSR
  - Synchronize
  - Instruction synchronize
  - Order loads and stores
- Memory control instructions—These instructions provide control of caches, TLBs, and SRs.
  - Supervisor-level cache management instructions
  - User-level cache instructions
  - Segment register manipulation instructions
  - Translation lookaside buffer management instructions

This grouping does not indicate the execution unit that executes a particular instruction or group of instructions.

Integer instructions operate on byte, half-word, and word operands. Floating-point instructions operate on single-precision (one word) and double-precision (one double word) floating-point operands. The PowerPC architecture uses instructions that are four bytes long and word-aligned. It provides for byte, half-word, and word operand loads and stores between memory and a set of 32 GPRs. It also provides for word and double-word operand loads and stores between memory and a set of 32 floating-point registers (FPRs).

Computational instructions do not modify memory. To use a memory operand in a computation and then modify the same or another memory location, the memory contents must be loaded into a register, modified, and then written back to the target location with distinct instructions.

PowerPC processors follow the program flow when they are in the normal execution state. However, the flow of instructions can be interrupted directly by the execution of an instruction or by an asynchronous event. Either kind of exception may cause one of several components of the system software to be invoked.

Effective address computations for both data and instruction accesses use 32-bit unsigned binary arithmetic. A carry from bit 0 is ignored in 32-bit implementations.

3.2.2 AltiVec Instruction Set

The AltiVec instructions are divided into the following categories:

- Vector integer arithmetic instructions—These include arithmetic, logical, compare, rotate and shift instructions.
- Vector floating-point arithmetic instructions—These include floating-point arithmetic instructions, as well as a discussion on floating-point modes.
- Vector load and store instructions—These include load and store instructions for vector registers. The AltiVec technology defines LRU and transient type instructions that can be used to optimize memory accesses.
  - LRU instructions. The AltiVec architecture specifies that the **lvxl** and **stvxl** instructions differ from other AltiVec load and store instructions in that they leave cache entries in a least-recently-used (LRU) state instead of a most-recently-used state.
  - Transient instructions. The AltiVec architecture describes a difference between static and transient memory accesses. A static memory access should have some reasonable degree of locality and be referenced several times or reused over some reasonably long period of time. A transient memory reference has poor locality and is likely to be referenced a very few times or over a very short period of time.

    The following instructions are interpreted to be transient:
    - **dstt** and **dststt** (transient forms of the two data stream touch instructions)
    - **lvxl** and **stvxl**
- Vector permutation and formatting instructions—These include pack, unpack, merge, splat, permute, select and shift instructions.
- Processor control instructions—These instructions are used to read and write from the vector status and control register (VSCR).
- Memory control instructions—These instructions are used for managing the caches (user level and supervisor level).

3.2.3 MPC7410 Microprocessor Instruction Set

The MPC7410 instruction set is defined as follows:

- The MPC7410 provides hardware support for all 32-bit PowerPC instructions.
- The MPC7410 implements the following instructions optional to the PowerPC architecture:
  - External Control In Word Indexed (**eciwx**)
  - External Control Out Word Indexed (**ecowx**)
  - Data Cache Block Allocate (**dcba**)
  - Floating Select (**fsel**)
  - Floating Reciprocal Estimate Single-Precision (**fres**)
  - Floating Reciprocal Square Root Estimate (**frsqrte**)
  - Store Floating-Point as Integer Word (**stfiwx**)

3.3 On-Chip Cache Implementation

The following subsections describe the PowerPC architecture's treatment of cache in general, and the MPC7410-specific implementation, respectively.

3.3.1 PowerPC Cache Model

The PowerPC architecture does not define hardware aspects of cache implementations. For example, PowerPC processors can have unified caches, separate L1 instruction and data caches (Harvard architecture), or no cache at all. PowerPC microprocessors control the following memory access modes on a page or block basis:

- Write-back/write-through mode
- Caching-inhibited mode
- Memory coherency

The caches are physically addressed, and the data cache can operate in either write-back or write-through mode as specified by the PowerPC architecture.

The PowerPC architecture defines the term 'cache block' as the cacheable unit. The VEA and OEA define cache management instructions a programmer can use to affect cache contents.

3.3.2 MPC7410 Microprocessor Cache Implementation

The MPC7410 cache implementation is described in Section 2.4, "On-Chip Instruction and Data Caches," and Section 2.5, "L2 Cache Implementation." The BPU also contains a 64-entry BTIC that provides immediate access to cached target instructions. For more information, see Section 2.2.2, "Branch Processing Unit (BPU)."

3.4 Exception Model

The following sections describe the PowerPC exception model and the MPC7410 implementation.

3.4.1 PowerPC Exception Model

The PowerPC exception mechanism allows the processor to interrupt the instruction flow to handle certain situations caused by external signals, errors, or unusual conditions arising from the instruction execution.

When exceptions occur, information about the state of the processor is saved to certain registers and the processor begins execution at an address (exception vector) predetermined for each exception. Exception processing occurs in supervisor mode.

Although multiple exception conditions can map to a single exception vector, a more specific condition may be determined by examining a register associated with the exception—for example, the DSISR and the FPSCR. Additionally, some exception conditions can be enabled or disabled explicitly by software.

The PowerPC architecture requires that exceptions be handled in program order; therefore, although a particular implementation may recognize exception conditions out of order, they are handled in order. When an instruction-caused exception is recognized, any unexecuted instructions that appear earlier in the instruction stream, including any that are undispatched, are required to complete before the exception is taken, and any exceptions those instructions cause must also be handled first. Likewise, asynchronous, precise exceptions are recognized when they occur, but are not handled until the instructions currently in the completion queue successfully retire or generate an exception, and the completion queue is emptied.

Unless a catastrophic condition causes a system reset or machine check exception, only one exception is handled at a time. For example, if one instruction encounters multiple exception conditions, those conditions are handled sequentially. After the exception handler handles an exception, the instruction processing continues until the next exception condition is encountered. Recognizing and handling exception conditions sequentially guarantees that exceptions are recoverable.

When an exception is taken, information about the processor state before the exception was taken is saved in SRR0 and SRR1. Exception handlers should save the information stored in SRR0 and SRR1 early to prevent the program state from being lost due to a system reset or machine check exception, or due to an instruction-caused exception in the exception handler. The contents of SRR0 and SRR1 should also be saved before enabling external interrupts.

The PowerPC architecture supports four types of exceptions:

- Synchronous, precise—These are caused by instructions. All instruction-caused exceptions are handled precisely; that is, the machine state at the time the exception occurs is known and can be completely restored. This means that (excluding the trap and system call exceptions) the address of the faulting instruction is provided to the exception handler and that neither the faulting instruction nor subsequent instructions in the code stream will complete execution before the exception is taken. Once the exception is processed, execution resumes at the address of the faulting instruction (or at an alternate address provided by the exception handler). When an exception is taken due to a trap or system call instruction, execution resumes at an address provided by the handler.
- Synchronous, imprecise—The PowerPC architecture defines two imprecise floating-point exception modes: recoverable and nonrecoverable. Even though the MPC7410 provides a means to enable the imprecise modes, it implements these modes identically to the precise mode (that is, enabled floating-point exceptions are always precise).
- Asynchronous, maskable—The PowerPC architecture defines external and decrementer interrupts as maskable, asynchronous exceptions. When these exceptions occur, their handling is postponed until the next instruction, and any exceptions associated with that instruction, completes execution. If no instructions are in the execution units, the exception is taken immediately upon determination of the correct restart address (for loading SRR0). As shown in Table 5, the MPC7410 implements additional asynchronous, maskable exceptions.
- Asynchronous, nonmaskable—There are two nonmaskable asynchronous exceptions: system reset and the machine check exception. These exceptions may not be recoverable, or may provide a limited degree of recoverability. Exceptions report recoverability through the MSR[RI] bit.

3.4.2 MPC7410 Microprocessor Exception Implementation

The MPC7410 exception classes described above are shown in Table 5.

Table 5. MPC7410 Microprocessor Exception Classifications

Synchronous/Asynchronous	Precise/Imprecise	Exception Type
Asynchronous, nonmaskable	Imprecise	Machine check, system reset
Asynchronous, maskable	Precise	External, decrementer, system management, thermal management, and performance monitor interrupts
Synchronous	Precise	Instruction-caused exceptions

Although exceptions have other characteristics, such as priority and recoverability, Table 5 describes categories of exceptions the MPC7410 handles uniquely. Table 5 includes no synchronous imprecise exceptions; although the PowerPC architecture supports imprecise handling of floating-point exceptions, the MPC7410 implements these exception modes precisely.

Table 6 lists MPC7410 exceptions and conditions that cause them. Exceptions specific to the MPC7410 are indicated. Note that only three exceptions may result from execution of an AltiVec instruction:

- AltiVec unavailable exception. Taken if there is an attempt to execute any non-stream vector instruction with MSR[VA] = 0. After this exception is handled, execution resumes at offset 0x00F20. This exception does not occur for stream instructions (**dst**[**t**], **dstst**[**t**], or **dss**). Note that the contents of the VRSAVE register are not protected by this exception, which is consistent with the AltiVec specification.
- A DSI exception. Taken if a vector load or store operation encounters a page fault (does not find a valid PTE) or a protection violation. Also a DSI occurs if a vector load or store attempts to access T = 1 direct store space.
- AltiVec assist exception. Taken in some cases if a vector floating-point instruction detects denormalized data as an input or output in Java mode.

Table 6. Exceptions and Conditions

Exception Type	Vector Offset (hex)	Causing Conditions
Reserved	00000	—
System reset	00100	Assertion of either $\overline{\text{HRESET}}$ or $\overline{\text{SRESET}}$ or at power-on reset
Machine check	00200	Assertion of $\overline{\text{TEA}}$ during a data bus transaction, assertion of $\overline{\text{MCP}}$, or an address, data, or L2 bus parity error. MSR[ME] must be set.
DSI	00300	As specified in the PowerPC architecture. For TLB misses on load, store, or cache operations, a DSI exception occurs if a page fault occurs. The MPC7410 takes a DSI if a **lwarx** or **stwcx.** instruction is executed to an address marked write-through or if the data cache is enabled and locked.
ISI	00400	As defined by the PowerPC architecture.
External interrupt	00500	MSR[EE] = 1 and $\overline{\text{INT}}$ is asserted.
Alignment	00600	A floating-point load/store, **stmw**, **stwcx.**, **lmw**, **lwarx**, **eciwx** or **ecowx** instruction operand is not word-aligned. A multiple/string load/store operation is attempted in little-endian mode. The operand of **dcbz** is in memory that is write-through-required or caching-inhibited or the cache is disabled

Table 6. Exceptions and Conditions (Continued)

Exception Type	Vector Offset (hex)	Causing Conditions
Program	00700	As defined by the PowerPC architecture.
Floating-point unavailable	00800	As defined by the PowerPC architecture.
Decrementer	00900	As defined by the PowerPC architecture, when the most significant bit of the DEC register changes from 0 to 1 and MSR[EE] = 1.
Reserved	00A00–00BFF	—
System call	00C00	Execution of the System Call (**sc**) instruction.
Trace	00D00	MSR[SE] = 1 or a branch instruction completes and MSR[BE] = 1. Unlike the architecture definition, **isync** does not cause a trace exception on MPC7410.
Reserved	00E00	The MPC7410 does not generate an exception to this vector. Other PowerPC processors may use this vector for floating-point assist exceptions.
Reserved	00E10–00EFF	—
Performance monitor[1]	00F00	The limit specified in a PMC register is reached and MMCR0[ENINT] = 1
AltiVec unavailable[1]	00F20	Occurs due to an attempt to execute any non-stream AltiVec instruction while MSR[VA] = 0. This exception is not taken for stream instructions (**dst**[**t**], **dstst**[**t**] or **dss**).
Instruction address breakpoint[1]	01300	IABR[0–29] matches EA[0–29] of the next instruction to complete, and IABR[BE] = 1.
System management interrupt[1]	01400	MSR[EE] = 1 and $\overline{\text{SMI}}$ is asserted.
Reserved	01500–015FF	—
AltiVec assist[1]	01600	Supports denormalization detection in Java mode as defined by the AltiVec specification.
Thermal management exception[1]	01700	Thermal management is enabled, the junction temperature exceeds the threshold specified in THRM1 or THRM2, and MSR[EE] = 1.
Reserved	01800–02FFF	—

[1] MPC7410-specific

3.5 Memory Management

The following subsections describe the memory management features of the PowerPC architecture, and the MPC7410 implementation, respectively.

3.5.1 PowerPC Memory Management Model

The primary functions of the MMU are to translate logical (effective) addresses to physical addresses for memory accesses and to provide access protection on blocks and pages of memory. There are two types of accesses generated by the MPC7410 that require address translation—instruction accesses, and data accesses to memory generated by load, store, and cache control instructions.

The PowerPC architecture defines different resources for 32- and 64-bit processors; the MPC7410 implements the 32-bit memory management model. The memory management model provides 4 Gbytes of

logical address space accessible to supervisor and user programs with a 4-Kbyte page size and 256-Mbyte segment size. In addition, it defines an interim 52-bit virtual address and hashed page tables for generating 32-bit physical addresses.

The architecture also provides independent four-entry BAT arrays for instructions and data that maintain address translations for blocks of memory. These entries define blocks that can vary from 128 Kbytes to 256 Mbytes. The BAT arrays are maintained by system software.

The PowerPC MMU and exception model support demand-paged virtual memory. Virtual memory management permits execution of programs larger than the size of physical memory; demand-paged implies that individual pages are loaded into physical memory from system memory only when they are first accessed by an executing program.

The hashed page table is a variable-sized data structure that defines the mapping between virtual page numbers and physical page numbers. The page table size is a power of 2, and its starting address is a multiple of its size. The page table contains a number of page table entry groups (PTEGs). A PTEG contains eight page table entries (PTEs) of eight bytes each; therefore, each PTEG is 64 bytes long. PTEG addresses are entry points for table search operations.

Setting MSR[IR] enables instruction address translations and MSR[DR] enables data address translations. If the bit is cleared, the respective effective address is the same as the physical address.

3.5.2 MPC7410 Microprocessor Memory Management Implementation

The MPC7410 implements separate MMUs for instructions and data. It maintains a copy of the segment registers in the instruction MMU; however, read and write accesses to the segment registers (**mfsr** and **mtsr** are handled through the segment registers in the data MMU. The MPC7410 MMU is described in Section 2.3, “Memory Management Units (MMUs).”

The R (referenced) bit is updated in the PTE in memory (if necessary) during a table search due to a TLB miss. Updates to the C (changed) bit are treated like TLB misses. A complete table search is performed and the entire TLB entry is rewritten to update the C bit.

3.6 Instruction Timing

The MPC7410 is a pipelined, superscalar processor. A pipelined processor is one in which instruction processing is divided into discrete stages, allowing work to be done on different instructions in each stage. For example, after an instruction completes one stage, it can pass on to the next stage leaving the previous stage available to the subsequent instruction. This improves overall instruction throughput.

A superscalar processor is one that issues multiple independent instructions into separate execution units, allowing instructions to execute in parallel. The MPC7410 has eight independent execution units, two for integer instructions, and one each for floating-point, branch, load/store, system register, vector permute, and vector arithmetic logic unit instructions. Having separate GPRs, FPRs, and VRs allows integer, floating-point, and vector calculations, and load and store operations to occur simultaneously without interference. Additionally, rename buffers are provided to allow operations to post execution results for use by subsequent instructions without committing them to the architected FPRs, GPRs, and VRs.

As shown in Figure 6, the common pipeline of the MPC7410 has four stages through which all instructions must pass—fetch, decode/dispatch, execute, and complete/write back. Some instructions occupy multiple stages simultaneously and some individual execution units have additional stages. For example, the floating-point pipeline consists of three stages through which all floating-point instructions must pass.

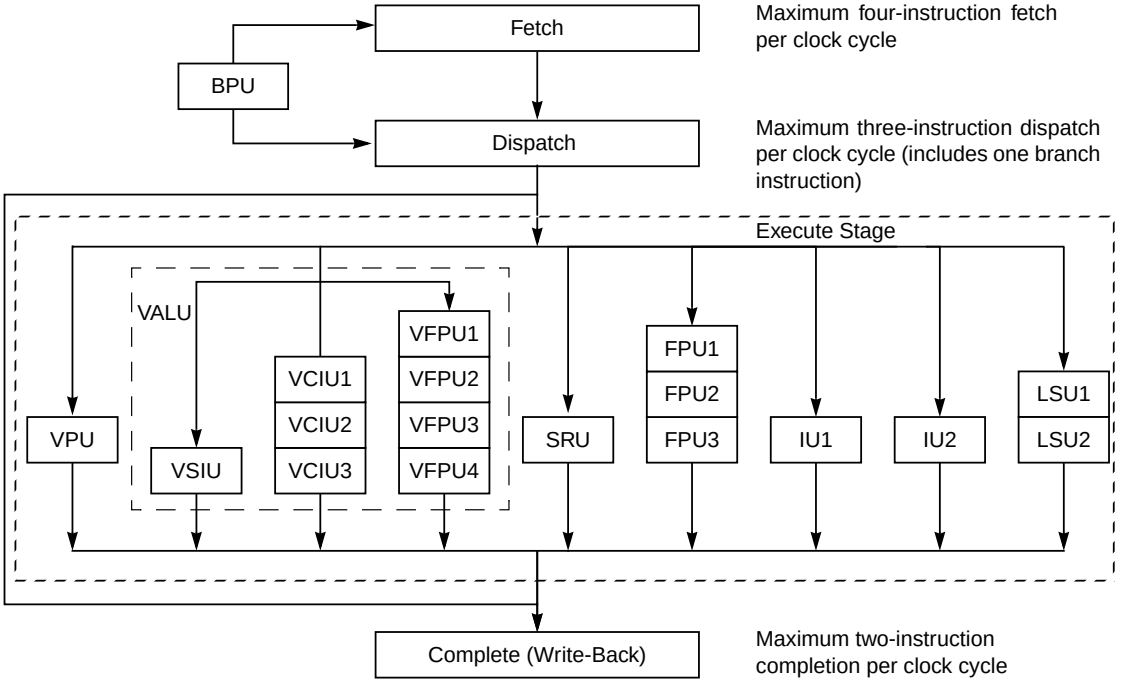


Figure 6. Pipeline Diagram

Note that Figure 6 does not show features, such as reservation stations and rename buffers that reduce stalls and improve instruction throughput.

The instruction pipeline in the MPC7410 has four major pipeline stages, described as follows:

- The fetch pipeline stage primarily involves retrieving instructions from the memory system and determining the location of the next instruction fetch. The BPU decodes branches during the fetch stage and removes those that do not update CTR or LR from the instruction stream.
- The dispatch stage is responsible for decoding the instructions supplied by the instruction fetch stage and determining which instructions can be dispatched in the current cycle. A rename ID is given to instructions with a target destination. If source operands for the instruction are available, they are read from the appropriate register file or rename register to the execute pipeline stage. If a source operand is not available, dispatch provides a tag that indicates which rename register will supply the operand when it becomes available. At the end of the dispatch stage, the dispatched instructions and their operands are latched by the appropriate execution unit.
- Instructions executed by the IUs, FPU, SRU, LSU, VPU, and VALU are dispatched from the bottom two positions in the instruction queue. In a single clock cycle, a maximum of two instructions can be dispatched to these execution units in any combination. When an instruction is dispatched, it is assigned a position in the eight-entry completion queue. A branch instruction can be issued on the same clock cycle for a maximum three-instruction dispatch.
- During the execute pipeline stage, each execution unit that has an executable instruction executes the selected instruction (perhaps over multiple cycles), writes the instruction's result into the appropriate rename register, and notifies the completion stage that the instruction has finished execution. In the case of an internal exception, the execution unit reports the exception to the completion pipeline stage and (except for the FPU) discontinues instruction execution until the exception is handled. The exception is not signaled until that instruction is the next to be completed.

Execution of most floating-point instructions is pipelined within the FPU allowing up to three instructions to be executing in the FPU concurrently. The FPU stages are multiply, add, and round-convert. Execution of most load/store instructions is also pipelined. The load/store unit has two pipeline stages. The first stage is for effective address calculation and MMU translation and the second stage is for accessing the data in the cache.

- The complete pipeline stage maintains the correct architectural machine state and transfers execution results from the rename registers to the GPRs and FPRs (and CTR and LR, for some instructions) as instructions are retired. As with dispatching instructions from the instruction queue, instructions are retired from the two bottom positions in the completion queue. If completion logic detects an instruction causing an exception, all following instructions are cancelled, their execution results in rename registers are discarded, and instructions are fetched from the appropriate exception vector.

Because the PowerPC architecture can be applied to such a wide variety of implementations, instruction timing varies among PowerPC processors, and this pipeline description is specific to the MPC7410.

3.7 Power Management

The MPC7410 provides four power modes, selectable by setting the appropriate control bits in the MSR and HID0 registers. The four power modes are as follows:

- Full-power—This is the default power state of the MPC7410. The MPC7410 is fully powered and the internal functional units are operating at the full processor clock speed. If the dynamic power management mode is enabled, functional units that are idle will automatically enter a low-power state without affecting performance, software execution, or external hardware.
- Doze—All the functional units of the MPC7410 are disabled except for the time base/decrementer registers, the thermal assist unit, and the bus snooping logic. When the processor is in doze mode, an external asynchronous interrupt, a system management interrupt, a decrementer exception, a hard or soft reset, or machine check brings the MPC7410 into the full-power state. The MPC7410 in doze mode maintains the PLL in a fully powered state and locked to the system external clock input (SYSCLK) so a transition to the full-power state takes only a few processor clock cycles.
- Nap—The nap mode further reduces power consumption by disabling bus snooping, leaving only the decrementer/time base registers, the thermal assist unit, the PLL, and the DLL (for L2 RAM clocks) in a powered state. The MPC7410 returns to the full-power state upon receipt of an external asynchronous interrupt, a system management interrupt, a decrementer exception, a hard or soft reset, or a machine check input ($\overline{\text{MCP}}$). A return to full-power state from a nap state takes only a few processor clock cycles. When the processor is in nap mode, if $\overline{\text{QACK}}$ is negated, the processor is put in doze mode to support snooping.
- Sleep—Sleep mode minimizes power consumption by disabling all internal functional units, after which external system logic may disable the PLL and SYSCLK. Returning the MPC7410 to the full-power state requires the enabling of the PLL and SYSCLK, followed by the assertion of an external asynchronous interrupt, a system management interrupt, a hard or soft reset, or a machine check input ($\overline{\text{MCP}}$) signal after the time required to relock the PLL.

3.8 Thermal Management

The MPC7410's thermal assist unit (TAU) provides a way to control heat dissipation. This ability is particularly useful in portable computers, which, due to power consumption and size limitations, cannot use desktop cooling solutions such as fans. Therefore, better heat sink designs coupled with intelligent thermal management is of critical importance for high performance portable systems.

Primarily, the thermal management system monitors and regulates the system's operating temperature. For example, if the temperature is about to exceed a set limit, the system can be made to slow down or even suspend operations temporarily in order to lower the temperature.

The thermal management facility also ensures that the processor's junction temperature does not exceed the operating specification. To avoid the inaccuracies that arise from measuring junction temperature with an external thermal sensor, the MPC7410's on-chip thermal sensor and logic tightly couples the thermal management implementation.

The TAU consists of a thermal sensor, digital-to-analog convertor, comparator, control logic, and the dedicated SPRs described in Section 3.1, "PowerPC Registers and Programming Model." The TAU does the following:

- Compares the junction temperature against user-programmable thresholds
- Generates a thermal management exception if the temperature crosses the threshold
- Enables the user to estimate the junction temperature by way of a software successive approximation routine

The TAU is controlled through the privileged **mtspr**/**mfspr** instructions to the three SPRs provided for configuring and controlling the sensor control logic, which function as follows:

- THRM1 and THRM2 provide the ability to compare the junction temperature against two user-provided thresholds. Having dual thresholds gives the thermal management software finer control of the junction temperature. In single threshold mode, the thermal sensor output is compared to only one threshold in either THRM1 or THRM2.
- THRM3 is used to enable the TAU and to control the comparator output sample time. The thermal management logic manages the thermal management exception generation and time multiplexed comparisons in the dual threshold mode as well as other control functions.

Instruction cache throttling provides control of the MPC7410's overall junction temperature by determining the interval at which instructions are fetched. This feature is accessed through the ICTC register.

3.9 Performance Monitor

The MPC7410 incorporates a performance monitor facility that system designers can use to help bring up, debug, and optimize software performance. The performance monitor counts events during execution of instructions related to dispatch, execution, completion, and memory accesses.

The performance monitor incorporates several registers that can be read and written to by supervisor-level software. User-level versions of these registers provide read-only access for user-level applications. These registers are described in Section 3.1, "PowerPC Registers and Programming Model." Performance monitor control registers, MMCR0 or MMCR1, can be used to specify which events are to be counted and the conditions for which a performance monitor exception is taken. Additionally, the sampled instruction address register, SIA (USIA), holds the address of the first instruction to complete after the counter overflowed.

Attempting to write to a user-read-only performance monitor register causes a program exception, regardless of the MSR[PR] setting.

When a performance monitoring exception occurs, program execution continues from vector offset 0x00F00.

3.10 Differences between the MPC7410 and the MPC7400

The MPC7410 is a derivative of the MPC7400 microprocessor design. Table 7 summarizes the differences between the two microprocessors.

Table 7. Differences between the MPC7410 and the MPC7400

Feature	Difference
Private memory	The MPC7410 supports using the L2 SRAMs as direct-mapped private memory. The private memory feature on the MPC7410 is configured by a new supervisor-level, special-purpose register, the L2 private memory control register (L2PMCR). The MPC7400 does not support private memory. As such, the MPC7400 does not implement the L2PMCR.
L2 data bus width	The MPC7410 supports a 32- or 64-bit L2 data bus. The MPC7400 supports only a 64-bit L2 data bus.
L2 address bus width	The MPC7410 adds an L2 address signal, L2ADDR[18], to support up to 2 Mbyte of L2 cache with a 32-bit data bus.
Processor Version Register (PVR)	The PVR for MPC7410 is 0x800C_1nnn . The PVR for the MPC7400 is 0x000C_0nnn .
Core and I/O voltages	The electrical characteristics of the MPC7410 are different from the MPC7400. See the corresponding hardware specifications for each device.
Frequency of operation and core/clock ratios	The clock AC specifications and PLL configuration of the MPC7410 are different from the MPC7400. See the corresponding hardware specifications for each device.

3.11 Differences between the MPC7410 and the MPC750

The design philosophy on the MPC7410 (and the MPC7400) is to change from the MPC750 base only where required to gain compelling multimedia and multiprocessor performance. The MPC7410's core is essentially the same as the MPC750's, except that whereas the MPC750 has a 6-entry completion queue and has slower performance on some floating-point double-precision operations, the MPC7410 has an 8-entry completion queue and a full double-precision FPU. The MPC7410 also adds the AltiVec instruction set, has a new memory subsystem, and can interface to the improved MPX bus. Differences are summarized in Table 8.

Table 8. Differences between the MPC7410 and the MPC750

Feature	Difference		
Core			
Sequencing	The MPC750 has a 6-entry IQ and a 6-entry CQ. For each clock, it can fetch four instructions, dispatch two instructions, fold one branch, and complete two instructions. The MPC7410 is identical, except for an eight-entry CQ, as shown in Figure 1. The extra CQ entries reduce the opportunity for dispatch bottlenecks to the MPC7410's additional execution units.		
FPU	On the MPC750, single-precision operations involving multiplication have a 3-cycle latency, while their double-precision equivalents take an additional cycle. Because the MPC7410 has a full double-precision FPU, double- and single-precision multiplies have the same latency: 3 cycles. Floating-point divides have the same latency for both designs (17 cycles for single-precision, 31 for double-precision).		
	MPC750	Double-precision floating-point multiply	4 cycles
		All other floating-point add and multiply	3 cycles
	MPC7410	All floating-point add and multiply	3 cycles
AltiVec technology	The MPC7410 implements all instructions defined by the AltiVec specification. Two dispatchable AltiVec functional units were added, a vector permute unit (VPU) and a vector ALU unit (VALU). The VALU comprises a simple integer unit, a complex integer unit, and a floating-point unit. As shown in Figure 1, the MPC7410 also adds 32 128-bit vector registers (VRs) and 6 VR rename registers. The VPU handles permute and shift operations and the VALU handles calculations. The LSU handles AltiVec load and store operations. To support AltiVec operations, all memory subsystem data buses are 128 bits wide (as opposed to 64 bits in the MPC750). Queues have been added and queue sizes have been increased to sustain heavy AltiVec technology usage. The AltiVec technology is designed to improve the performance of vector-intensive code in applications such as multimedia and digital signal processing. AltiVec-targeted code can accelerate 2D and 3D graphics functions 3–5 times, especially core functions in 3D engines and game-related 2D functions.		
Memory Subsystem			
The MPC7410 has a new memory subsystem designed to support AltiVec technology loads, the new MPX bus protocol, and 5-state multiprocessing capabilities. Queues and queue sizes are designed to support more efficient data flow. For example, the MPC750 has a three-entry LSU store queue, while the MPC7410 has a six-entry LSU store queue. The MPC7410 adds an eight-entry reload buffer, where L1 data cache misses can wait for their data to be loaded. This enables load miss folding and store miss merging.			
Load miss folding	In the MPC750, if a second load misses to the same cache block, the second load must wait for the critical word of the first load before it can access its data, and subsequent accesses are also stalled. In the MPC7410, the first load or store causes an entry to be allocated in the reload buffer. A subsequent load to the same cache block is placed aside in the load fold queue (LFQ), and it can return its data immediately when available. Also, subsequent accesses to the cache are not blocked and can be processed. For example, on the MPC750 if a load or store (access A) misses in the data cache, a subsequent load (access B) to the same cache block must wait until the critical word for A is retired. Because of this, any subsequent loads or stores after access B also cannot access the data cache until the reload for access A completes. On the other hand, with the MPC7410 if a load or store access A misses in the data cache, up to four subsequent misses to the same cache block can be folded into the LFQ, and subsequent instructions can access the data cache. Loads are blocked only when the reload table or the LFQ are full.		
Store miss merging	In the MPC750, if a second store misses to the same cache block, it must wait for the critical word of the first store before it can write its data. The MPC7410 can merge several stores to the same cache block into the same entry in its reload buffer. If enough stores merge to write all 32 bytes of the cache block (usually via two back-to-back AltiVec store misses), then no data needs to be loaded from the bus and an address-only transaction (KILL) is broadcast instead.		

Table 8. Differences between the MPC7410 and the MPC750 (Continued)

<table>
<tr><th>Feature</th><th colspan="2">Difference</th></tr>
<tr><td colspan="3">Cache</td></tr>
<tr><td>Allocate on reload</td><td colspan="2">Both designs have the same L1 cache size, but differ in their block allocation policy. The MPC750 has an allocate-on-miss policy, while the MPC7410 has an allocate-on-reload policy, which allows better cache allocation and replacement and more efficient use of data bus bandwidth.
If access A misses in the cache, the MPC750 immediately identifies the victim block (call it X) if there is one and allocates its space for the new data (call it Y) to be loaded. If a subsequent access (access B) needs this victim block, even if access B occurs before Y has been loaded, then it will miss because as soon as X is victimized it is no longer valid. After Y has loaded (and, if X is modified, after X has been cast out), X must be reloaded, and B must wait until its data is valid again.
The MPC7410, on the other hand, delays allocation/victimization until the block reload occurs. In the example above, while Y is being loaded, B can hit block X, and a different block is victimized. This allows more efficient use of the cache and can reduce thrashing.
On the MPC7410, allocation occurs in parallel with reload which uses the cache more efficiently.
<table>
<tr><th>MPC750</th><th>MPC7410</th></tr>
<tr><td>1-cycle load arbitration</td><td>1-cycle load arbitration</td></tr>
<tr><td>1-cycle allocate</td><td rowspan="3">4-beat reload</td></tr>
<tr><td>4-cycle victimization (if castout needed)</td></tr>
<tr><td>4-beat reload (64 bits/beat)</td></tr>
<tr><td>Total = 6 or 10 cycles</td><td>Total = 5 cycles</td></tr>
</table></td></tr>
<tr><td>Outstanding misses</td><td colspan="2">The MPC750 allows one outstanding data cache miss and one outstanding instruction cache miss (accessing the L2 or the bus) at any time. The MPC7410 allows one instruction cache miss and up to eight data side misses. Note that the L2 can queue up to four hits but with a fast L2 (1:1 mode) it is impossible to fill this queue with data cache misses. The L2 miss queue can queue four transactions waiting to access the processor address bus.</td></tr>
<tr><td>Miss under miss</td><td colspan="2">While processing a miss, the MPC750’s data cache allows subsequent loads and stores to hit in the data cache (hit under miss), but it blocks on the next miss until the first miss finishes reloading. The MPC7410 allows subsequent accesses that miss in the data cache to propagate to the L2 and beyond (miss under miss).</td></tr>
<tr><td>L2 cache</td><td colspan="2">The MPC7410 has twice as many on-chip L2 tags per way (8192) than the MPC750 and can support twice the L2 cache size (up to 2 Mbyte). The sectoring configuration differs as follows:
<table>
<tr><th colspan="2">MPC750</th><th colspan="2">MPC7410</th></tr>
<tr><td></td><td></td><td>2 Mbyte</td><td>4 sectors/tag</td></tr>
<tr><td>1 Mbyte</td><td>4 sectors/tag</td><td>1 Mbyte</td><td>2 sectors/tag</td></tr>
<tr><td>512 Kbyte</td><td>2 sectors/tag</td><td>512 Kbyte</td><td>1 sector/tag</td></tr>
<tr><td>256 Kbyte</td><td>2 sectors/tag</td><td>256 Kbyte</td><td>1 sector/tag</td></tr>
</table>
Assigning fewer sectors per tag uses the cache more efficiently.
The MPC7410 and MPC750 also have different cache reload policies. On the MPC750, an L1 cache miss that also misses in the L2 causes a reload from the bus to both L1 and L2. On the MPC7410, misses to the L1 instruction cache behave the same way, but misses to the L1 data cache cause data to be reloaded into the L1 only. Thus, with respect to the L1 data cache, the L2 holds only blocks that are cast out; it acts as a giant victim cache for the L1 data cache. This improves performance because the data is duplicated in the L1 data cache and L2 less often.</td></tr>
<tr><td>L2 data bus width</td><td colspan="2">The MPC7410 supports a 32- or 64-bit L2 data bus.
The MPC7400 supports only a 64-bit L2 data bus.</td></tr>
</table>

Table 8. Differences between the MPC7410 and the MPC750 (Continued)

Feature	Difference
L2 address bus width	The MPC7410 L2 address bus has two additional bits: MPC7410 L2ADDR[18:0] MPC750 L2ADDR[16:0]
Private memory	The MPC7410A's L2 interface supports using the SRAM area as a direct-mapped, private memory space. This feature is not supported on the MPC750. The private memory space provides a low-latency, high-bandwidth area for critical data or instructions. Accesses to the private memory space do not propagate to the L2 cache nor are they visible to the external system bus.
60x bus/ MPX bus	The MPC7410 supports the 60x bus used by the MPC750, but it also supports a new bus (MPX bus). It implements a 5-state cache-coherency protocol (MERSI) and the MESI and MEI subsets. This provides better hardware support of multiprocessing. For example, the MPX bus supports data intervention. On the 60x bus, if one processor performs a read of data that is marked modified in another processor's cache, the transaction is retried and the data is pushed to memory, after which the transaction is restarted. The MPX bus allows data to be forwarded directly to the requesting processor from the processor that has it cached. (The MPC7410 also supports intervention for data marked exclusive and shared.) The MPC7410 supports up to seven simultaneous transactions on the 60x or MPX bus interface (one in progress and six pending); the MPC750 supports only two.

How to Reach Us:

Home Page:
www.freescale.com

E-mail:
support@freescale.com

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
+1-800-521-6274 or +1-480-768-2130
support@freescale.com

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
support@freescale.com

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

freescale™
semiconductor