Chapter 1 Overview

This chapter provides an overview of the MPC7450 microprocessor features, including a block diagram showing the major functional components. It also provides information about how the MPC7450 implementation complies with the PowerPC™ and AltiVec™ architecture definitions.

1.1 MPC7450 Microprocessor Overview

This section describes the features and general operation of the MPC7450 and provides a block diagram showing major functional units. The MPC7450 is an implementation of the PowerPC microprocessor family of reduced instruction set computer (RISC) microprocessors. The MPC7450 consists of a processor core, 32-Kbyte separate L1 instruction and data cache, a 256-Kbyte L2 cache, and an internal L3 tag and controller which support a glueless backside L3 cache through a dedicated high-bandwidth interface. The core is a high-performance PowerPC superscalar design supporting multiple execution units, including four independent units that execute AltiVec instructions.

The MPC7450 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 MPC7450 provides virtual memory support for up to 4 Petabytes (2^{52}) of virtual memory and real memory support for up to 64 Gigabytes (2^{36}) of physical memory.

The MPC7450 also implements the AltiVec instruction set architectural extension. The MPC7450 is a superscalar processor that can dispatch and complete three instructions simultaneously. It incorporates the following execution units:

- 64-bit floating-point unit (FPU)
- Branch processing unit (BPU)
- Load/store unit (LSU)
- Four integer units (IUs):
  - Three shorter latency IUs (IU1a–IU1c)—execute all integer instructions except multiply, divide, and move to/from special-purpose register (SPR) instructions.
  - Longer latency IU (IU2)—executes miscellaneous instructions including condition register (CR) logical operations, integer multiplication and division instructions, and move to/from SPR instructions.

- Four vector units that support AltiVec instructions:
  — Vector permute unit (VPU)
  — Vector integer unit 1 (VIU1)—performs shorter latency integer calculations
  — Vector integer unit 2 (VIU2)—performs longer latency integer calculations
  — 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 MPC7450-based systems. Most integer instructions (including VIU1 instructions) have a one-clock cycle execution latency.

Several execution units are multiple-stage pipelines; that is, the tasks they perform are broken into subtasks executed in successive stages. Typically, instructions follow one another through the stages, so a four-stage unit can work on four instructions when its pipeline is full. So, although an instruction may have to pass through several stages, the execution unit can achieve a throughput of one instruction per clock cycle.

AltiVec computational instructions are executed in the four independent, pipelined AltiVec execution units. A maximum of two AltiVec instructions can be issued in order to any combination of AltiVec execution units per clock cycle. Moreover, the VIU2, VFPU, and VPU are pipelined, so they can operate on multiple instructions. The VPU has a two-stage pipeline, the VIU2 and VFPU each have four-stage pipelines. As many as 10 AltiVec instructions can be executing concurrently.

Note that for the MPC7450, double- and single-precision versions of floating-point instructions have the same latency. For example, a floating-point multiply-add instruction takes five cycles to execute, regardless of whether it is single- (**fmadds**) or double-precision (**fmadd**).

The MPC7450 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, as described in Chapter 5, “Memory Management.” For information about the L1 caches, see Chapter 3, “L1, L2, and L3 Cache Operation.”

The MPC7450’s L2 cache is implemented with an on-chip, 256-Kbyte, eight-way set-associative physically addressed memory available for storing data, instructions, or both. The L2 cache supports parity generation and checking for both tags and data. It responds with a nine-cycle load latency for an L1 miss that hits in L2. The L2 cache is fully pipelined for single-cycle throughput. For information about the L2 cache implementation, see Chapter 3, “L1, L2, and L3 Cache Operation.”

The L3 cache is implemented with an on-chip, eight-way set-associative tag memory, and with external, synchronous SRAMs for storing data, instruction, or both. The external SRAMs are accessed through a dedicated L3 cache port that supports a single bank of 1 or 2 Mbytes of synchronous SRAMs for L3 cache data. The L3 data bus is 64-bits wide and provides multiple SRAM options as well as quick quad-word forwarding to reduce latency. Alternately, the L3 interface can be configured to use half (1 Mbyte) or all of the SRAM area as a direct-mapped, private memory space. For information about the L3 cache implementation, see Chapter 3, "L1, L2, and L3 Cache Operation."

The MPC7450 has three power-saving modes, nap (with bus snoop), sleep, and deep sleep, which 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 MPC7450 also provides a thermal assist unit (TAU) and a way to reduce the instruction fetch rate for limiting power dissipation. Power management is described in Chapter 10, "Power and Thermal Management."

Figure 1-1 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.

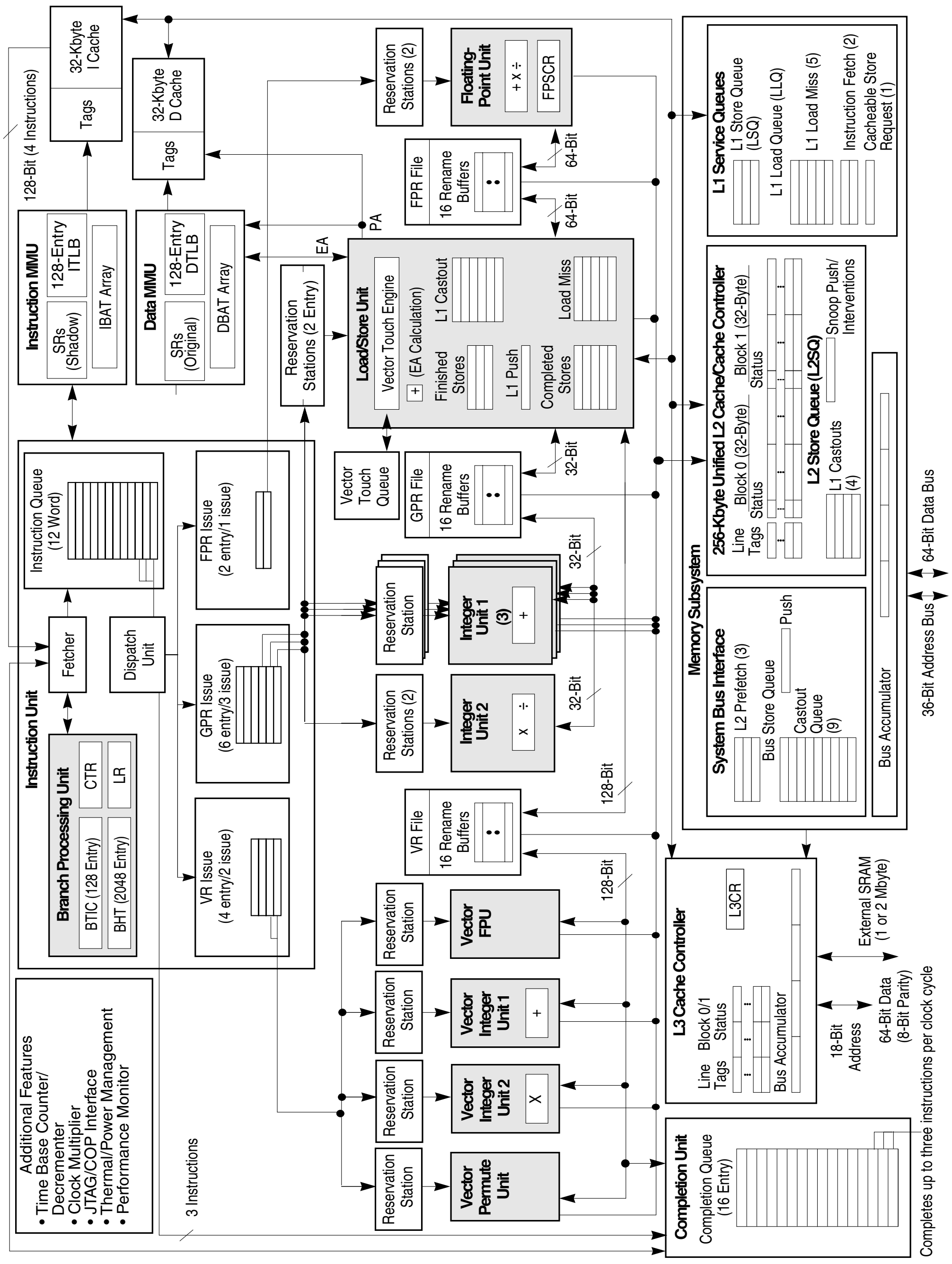


Figure 1-1. MPC7450 Microprocessor Block Diagram

1.2 MPC7450 Microprocessor Features

This section describes the features of the MPC7450. The interrelationships of these features are shown in Figure 1-1.

1.2.1 Overview of the MPC7450 Microprocessor Features

Major features of the MPC7450 are as follows:

- High-performance, superscalar microprocessor
  - As many as 4 instructions can be fetched from the instruction cache at a time
  - As many as 3 instructions can be dispatched to the issue queues at a time
  - As many as 12 instructions can be in the instruction queue (IQ)
  - As many as 16 instructions can be at some stage of execution simultaneously
  - Single-cycle execution for most instructions
  - One instruction per clock cycle throughput for most instructions
- Eleven independent execution units and three register files
  - Branch processing unit (BPU) features static and dynamic branch prediction
    - 128-entry (32-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, a fetch that hits the BTIC provides the first four instructions in the target stream.
    - 2048-entry branch history table (BHT) with two bits per entry for four levels of prediction—not-taken, strongly not-taken, taken, strongly taken
    - Up to three outstanding speculative branches
    - Branch instructions that do not update the count register (CTR) or link register (LR) are often removed from the instruction stream.
    - 8-entry link register stack to predict the target address of Branch Conditional to Link Register (**bclr**) instructions.
  - Four integer units (IUs) that share 32 GPRs for integer operands
    - Three identical IUs (IU1a, IU1b, and IU1c) can execute all integer instructions except multiply, divide, and move to/from special-purpose register instructions.
    - IU2 executes miscellaneous instructions including the CR logical operations, integer multiplication and division instructions, and move to/from special-purpose register instructions.
  - Five-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
- Thirty-two 64-bit FPRs for single- or double-precision operands

— Four vector units and 32-entry vector register file (VRs)

- Vector permute unit (VPU)
- Vector integer unit 1 (VIU1) handles short-latency AltiVec integer instructions, such as vector add instructions (**vaddsbs**, **vaddshs**, and **vaddsws**, for example)
- Vector integer unit 2 (VIU2) handles longer -latency AltiVec integer instructions, such as vector multiply add instructions (**vmhaddshs**, **vmhraddshs**, and **vmladduhm**, for example).
- Vector floating-point unit (VFPU)

— Three-stage load/store unit (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
- Three-cycle GPR and AltiVec load latency (byte, half word, word, vector) with 1-cycle throughput
- Four-cycle FPR load latency (single, double) with 1-cycle throughput
- No additional delay for misaligned access within doubleword 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
- Supports both big- and little-endian modes, including misaligned little-endian accesses

• Rename buffers

— 16 GPR rename buffers

— 16 FPR rename buffers

— 16 VR rename buffers

• Completion unit

— The completion unit retires an instruction from the 16-entry completion queue (CQ) 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 after a mispredicted branch
- Retires as many as three instructions per clock cycle

- 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 basis
  - Instruction cache can provide four instructions per clock cycle; data cache can provide four words per clock cycle
  - Caches can be disabled in software
  - Caches can be locked in software
  - MESI data cache coherency maintained in hardware
  - Separate copy of data cache tags for efficient snooping
  - Parity support on cache and tags
  - No snooping of instruction cache except for **icbi** instruction
  - Data cache supports AltiVec LRU and transient instructions, as described in Section 1.3.2.2, “AltiVec Instruction Set.”
  - Critical double- and/or quad-word forwarding is performed as needed. Critical quad-word forwarding is used for AltiVec loads and instruction fetches. Other accesses use critical double-word forwarding.
- Level 2 (L2) cache interface
  - On-chip, 256-Kbyte, 8-way set associative unified instruction and data cache
  - Fully pipelined to provide 32 bytes per clock cycle to the L1 caches
  - A total 9-cycle load latency for an L1 data cache miss that hits in L2
  - Pseudo least-recently-used (PLRU) replacement algorithm
  - Copyback or write-through data cache (on a per page basis)
  - 64-byte, two-sectored line size
  - Parity support on cache
- Level 3 (L3) cache interface
  - Provides critical double-word forwarding to the requesting unit
  - Internal L3 cache controller and tags
  - External data SRAMs
  - Support for 1- and 2-Mbyte L3 caches

  - Copyback or write-through data cache (on a per page basis)
  - 64-byte (1 M), or 128-byte (2 M) sectored line size
  - Private memory capability for half (1-Mbyte minimum) or all of the L3 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
  - Supports parity on cache and tags
  - Configurable core-to-L3 frequency divisors
  - 64-bit external L3 data bus can sustain 64 bits per L3 clock cycle
- Separate memory management units (MMUs) for instructions and data
  - 52-bit virtual address; 32- or 36-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- or software-reloadable (that is, on a TLB miss a page table search is performed in hardware or by system software)
- Efficient data flow
  - Although the VR/LSU interface is 128 bits, the L1/L2/L3 bus interface allows up to 256 bits.
  - The L1 data cache is fully pipelined to provide 128 bits/cycle to/from the VRs
  - L2 cache is fully pipelined to provide 256 bits per processor clock cycle to the L1 cache.
  - As many as 8 outstanding, out-of-order, cache misses are allowed between the L1 data cache and L2/L3 bus.
  - As many as 16 out-of-order transactions can be present on the bus
  - Store 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).
  - Three-entry finished store queue and five-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, MESI cache coherency protocols for data cache
  - Load/store with reservation instruction pair for atomic memory references, semaphores, and other multiprocessor operations
- Power and thermal management
  - 1.8 Volt processor core
  - The following three power-saving modes are available to the system:
    - Nap—Instruction fetching is halted. Only those clocks for TAU, time base, decrementer, and JTAG logic remain running. The part goes into the doze state to snoop memory operations on the bus and then back to nap using a $\overline{\text{QREQ}}/\overline{\text{QACK}}$ processor-system handshake protocol.
    - Sleep—Power consumption is further reduced by disabling bus snooping, leaving only the PLL in a locked and running state. All internal functional units are disabled.
    - Deep sleep—When the part is in the sleep state, the system can disable the PLL resulting. The system can then disable the SYSCLK source for greater system power savings. Power-on reset procedures for restarting and relocking the PLL must be followed upon exiting deep sleep state.
  - Thermal management facility provides software-controllable thermal management. Thermal management is performed through the use of three supervisor-level registers and an MPC7450-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

1.2.2 Instruction Flow

As shown in Figure 1-1, the MPC7450 instruction unit provides centralized control of instruction flow to the execution units. The instruction unit contains a sequential fetcher, 12-entry instruction queue (IQ), dispatch unit, and branch processing unit (BPU). It determines the address of the next instruction to be fetched based on information from the sequential fetcher and from the BPU.

See Chapter 6, “Instruction Timing,” for a detailed discussion of instruction timing.

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 MPC7450-specific dynamic branch prediction or the architecture-defined static branch prediction.

Branch instructions that do not affect the LR or CTR are often removed from the instruction stream. Section 6.4.1.1, “Branch Folding and Removal of Fall-Through Branch Instructions,” describes when a branch can be removed from the instruction stream.

Instructions dispatched 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.

1.2.2.1 Instruction Queue and Dispatch Unit

The instruction queue (IQ), shown in Figure 1-1, holds as many as 12 instructions and loads as many as 4 instructions from the instruction cache during a single processor clock cycle.

The fetcher attempts to initiate a new fetch every cycle. The two fetch stages are pipelined, so as many as four instructions can arrive to the IQ every cycle. All instructions except branch (**b***x*), Return from Exception (**rfi**), System Call (**sc**), Instruction Synchronize (**isync**), and no-op instructions are dispatched to their respective issue queues from the bottom three positions in the instruction queue (IQ0–IQ2) at a maximum rate of three instructions per clock cycle. Reservation stations are provided for the three IU1s, IU2, FPU, LSU, VPU, VIU2, VIU1, and VFPU. The dispatch unit checks for source and destination register dependencies, determines whether a position is available in the CQ, and inhibits subsequent instruction dispatching as required.

Branch instruction can be detected, decoded, and predicted from entries IQ0–IQ7. See Section 6.3.4, “Instruction Dispatch and Completion Considerations.”

1.2.2.2 Branch Processing Unit (BPU)

The BPU receives branch instructions from the IQ and executes them early in the pipeline, achieving the effect of a zero-cycle branch in some cases.

Branches with no outstanding dependencies (CR, LR, or CTR unresolved) can be processed and resolved immediately. For branches in which only the direction is unresolved due to a CR or CTR dependency, the branch path is predicted using either architecture-defined static branch prediction or MPC7450-specific dynamic branch prediction. Dynamic branch prediction is enabled if HID0[BHT] is set. For **bclr** branches where the target address is unresolved due to a LR dependency, the branch target can be predicted using the hardware link stack. Link stack prediction is enabled if HID0[LRSTK] is set.

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.

Dynamic prediction is implemented using a 2048-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 MPC7450 executes instructions from the predicted target stream although the results are not committed to architected registers until the conditional branch is resolved. Unresolved branches are held in a three-entry branch queue. When the branch queue is full, no further conditional branches can be processed until one of the conditions in the branch queue is resolved.

When a branch is taken or predicted as taken, instructions from the untaken path must be flushed and the target instruction stream must be fetched into the IQ. The BTIC is a 128-entry, four-way set associative cache that contains the most recently used branch target instructions (up to four instructions per entry) for **b** and **bc** branches. When a taken branch instruction of this type hits in the BTIC, the instructions arrive in the IQ two clock cycles later, 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 in the LR for certain types of branch instructions. The LR also contains the branch target address for Branch Conditional to Link Register (**bclr***x*) instructions. The CTR contains the branch target address for Branch Conditional to Count Register (**bcctr***x*) instructions. 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.

1.2.2.3 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 16-entry CQ. The completion unit tracks instructions from dispatch through execution and retires them in program order from the three bottom CQ entries (CQ0–CQ2).

Instructions cannot be dispatched to an execution unit unless there is a CQ vacancy.

Branch instructions that do not update the CTR or LR are often removed from the instruction stream. Those that are removed do not take a CQ entry. Branches that are not removed from the instruction stream follow the same dispatch and completion procedures as non-branch instructions but are not dispatched to an issue queue.

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 MPC7450 must recover from a mispredicted branch or any exception. An instruction is retired as it is removed from the CQ.

For a more detailed discussion of instruction completion, see Section 6.3.4, “Instruction Dispatch and Completion Considerations.”

1.2.2.4 Independent Execution Units

In addition to the BPU, the MPC7450 provides the ten execution units described in the following sections.

1.2.2.4.1 AltiVec Vector Permute Unit (VPU)

The VPU execute permutation instructions such as pack, unpack, merge, splat, and permute on vector operands.

1.2.2.4.2 AltiVec Vector Integer Unit 1 (VIU1)

The VIU1 executes simple vector integer computational instructions, such as addition, subtraction, maximum and minimum comparisons, averaging, rotation, shifting, comparisons, and Boolean operations.

1.2.2.4.3 AltiVec Vector Integer Unit 2 (VIU2)

The VIU2 executes longer-latency vector integer instructions, such as multiplication, multiplication/addition, and sum-across with saturation.

1.2.2.4.4 AltiVec Vector Floating-point Unit (VFPU)

The VFPU executes all vector floating-point instructions.

A maximum of two AltiVec instructions can be issued in order to any combination of AltiVec execution units per clock cycle. Moreover, the VIU2, VFPU, and VPU are pipelined, so they can operate on multiple instructions.

1.2.2.4.5 Integer Units (IUs)

The integer units (three IU1s and IU2) are shown in Figure 1-1. The IU1s execute shorter latency integer instructions, that is, all integer instructions except multiply, divide, and move to/from special-purpose register instructions. IU2 executes integer instructions with latencies of 3 cycles or more.

IU2 has a 32-bit integer multiplier/divider and a unit for executing CR logical operations and move to/from SPR instructions. The multiplier supports early exit for operations that do not require full 32- x 32-bit multiplication.

1.2.2.4.6 Floating-Point Unit (FPU)

The FPU, shown in Figure 1-1, is designed such that double-precision operations require only a single pass, with a latency of five cycles. As instructions are dispatched to the FPUs 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 start execution from the bottom reservation station only and execute in program 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 MPC7450 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 occurs after four consecutive, independent floating-point arithmetic instructions execute 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 16 floating-point rename registers. The MPC7450 writes the contents of the rename registers to the appropriate FPR when floating-point instructions are retired by the completion unit.

The MPC7450 supports all IEEE 754 floating-point data types (normalized, denormalized, NaN, zero, and infinity) in hardware, eliminating the latency incurred by software exception routines.

1.2.2.4.7 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 also calculates effective addresses and aligns data.

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 clock cycle from the perspective of the LSU. Loads to FPRs require a four-cycle total latency. 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 MPC7450 executes store instructions with a maximum throughput of one per clock cycle and a three-cycle total latency to the data cache. The time required to perform the load or store operation depends on the processor:bus clock ratio and whether the operation involves the on-chip caches, the L3 cache, system memory, or an I/O device.

1.2.3 Memory Management Units (MMUs)

The MPC7450's MMUs support up to 4 Petabytes (2^{52}) of virtual memory and 64 Gigabytes (2^{36}) 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 memory management units are contained within the load/store unit.

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 MPC7450 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. Lower-order address bits are untranslated and so are the same for both logical and physical addresses. These bits 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- or 36-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 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 or system software 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 MPC7450's instruction and data TLBs are 128-entry, two-way set-associative caches that contain address translations. The MPC7450 can initiate a hardware or system software search of the page tables in memory on a TLB miss.

1.2.4 On-Chip Instruction and Data Caches

The MPC7450 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 8 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 forwarded to the requesting unit, minimizing stalls due to load delays. For vector loads, the critical quad word is handled similarly but is transferred on the second beat. The cache being loaded is not blocked to internal accesses while the load completes.

The MPC7450 L1 cache organization is shown in Figure 1-2.

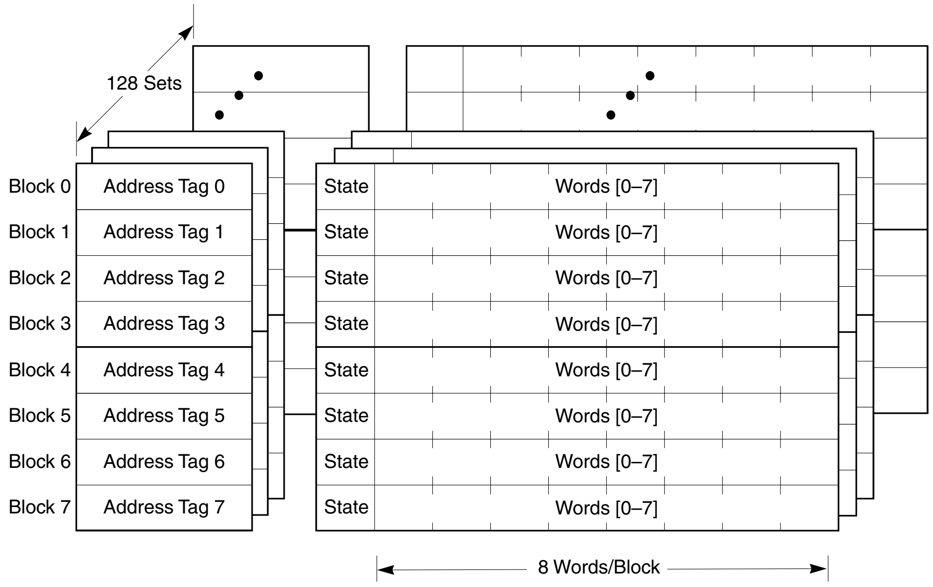


Figure 1-2. L1 Cache Organization

The instruction cache provides up to four instructions per clock 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 clock 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 MPC7450 also implements a 128-entry (32-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 four instructions in the target stream.

The BTIC can be disabled and invalidated through software. As with other aspects of MPC7450 instruction timing, BTIC operation is optimized for cache-line alignment. If the first target instruction is one of the first five instructions in the cache block, the BTIC entry holds four instructions. If the first target instruction is the last instruction before the cache block boundary, it is the only instruction in the corresponding BTIC entry. If the next-to-last instruction in a cache block is the target, the BTIC entry holds two valid target instructions, as shown in Figure 1-3.

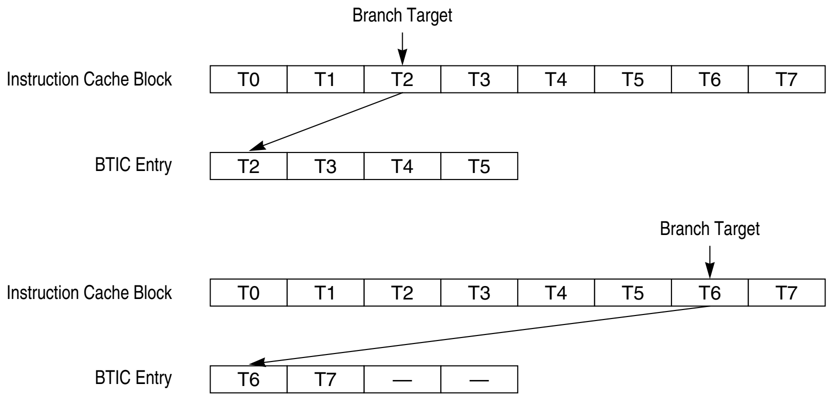


Figure 1-3. Alignment of Target Instructions in the BTIC

BTIC ways are updated using a FIFO algorithm.

For more information and timing examples showing cache hit and cache miss latencies, see Section 6.3.2, “Instruction Fetch Timing.”

1.2.5 L2 Cache Implementation

The L2 cache is a unified cache that receives memory requests from both the L1 instruction and data caches independently. The integrated L2 cache on the MPC7450 is a unified (containing instruction and data) 256 Kbyte on-chip cache. It is 8-way set associative and organized with 32-byte blocks and two blocks/line.

Each line consists of 64 bytes of data, organized as two blocks (also called sectors). Although all 16 words in a cache line share the same address tag, each block maintains the three separate status bits for the 8 words of the cache block, the unit of memory at which coherency is maintained. Thus, each cache line can contain 16 contiguous words from memory that are read or written as 8-word operations.

The MPC7450 integrated L2 cache organization is shown in Figure 1-4.

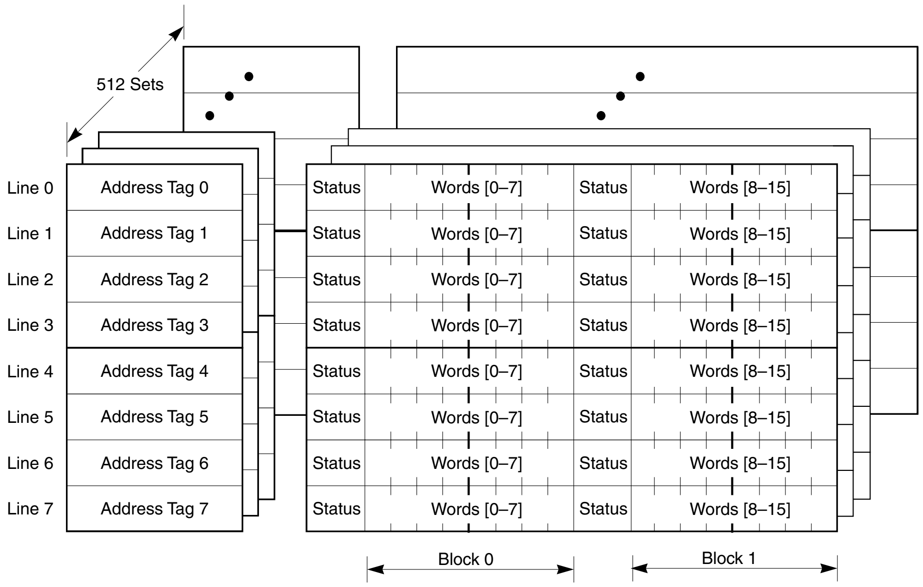


Figure 1-4. L2 Cache Organization

The L2 cache controller contains the L2 cache control register (L2CR), which:

- includes bits for enabling parity checking on the L2
- provides for instruction-only and data-only modes
- provides hardware flushing for the L2
- selects between two available replacement algorithms for the L2 cache.

The L2 implements the MESI cache coherency 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 L3 cache if they miss.

The L2 cache tags are fully pipelined and non-blocking for efficient operation. Thus the L2 cache can be accessed internally while a load for a miss is pending (allowing hits under misses). A reload for a cache miss is treated as a normal access and blocks other accesses for only one cycle.

For more information, see Chapter 3, “L1, L2, and L3 Cache Operation.”

1.2.6 L3 Cache Implementation

The unified L3 cache receives memory requests from L1 and L2 instruction and data caches independently. The L3 cache interface is implemented with an on-chip, two-way set associative tag memory with 2,048 (2K) tags per way and a dedicated interface with support for up to 2 Mbyte of external synchronous SRAMs.

Tags are sectored to support either two or four cache blocks per tag entry, depending on the L2 cache size. Each sector (32-byte cache block) in the L3 cache has three status bits that are used to implement the MESI cache coherency protocol. Accesses to the L3 cache can be designated as write-back or write-through and the L3 maintains cache coherency through snooping.

The L3 interface can be configured to use 1 or 2 Mbytes of the SRAM area as a private memory space. Accesses to private memory does not propagate to the system bus. The MPC7450 can also be configured to use 1 Mbyte of SRAM as L3 cache and a second Mbyte as private memory. Also, in this case, private memory accesses do not propagate to the L3 cache or to the external system bus.

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 L3 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. For more information, see Chapter 3, “L1, L2, and L3 Cache Operation.”

The L3 cache control register (L3CR) provides control of L3 cache configuration and interface timing. The L3 private memory control register (L3PM) configures the private memory feature.

The L3 cache interface provides two clock outputs that allow the clock inputs of the SRAMs to be driven at select frequency divisions of the processor core frequency.

Requests from the L3 cache generally result from instruction misses, data load or store misses, write-through operations, or cache management instructions. Requests from the L1 and L2 cache are compared against the L3 tags and serviced by the L3 cache if they hit; they are forwarded to the bus interface if they miss.

1.2.7 System Interface

The MPC7450 supports two interface protocols—MPX bus protocol and a subset of the 60x bus protocol. The full 60x bus protocol is described in the *PowerPC Microprocessor Family: The Bus Interface for 32-Bit Microprocessors*. Note that although this protocol is implemented by the MPC603e, MPC604e, MPC740, and MPC750 processors, it is referred to as the 60x bus interface. The MPX bus protocol is derived from the 60x bus protocol. The MPX bus interface includes several additional features that provide higher memory bandwidth than the 60x bus and more efficient use of the system bus in a multiprocessing environment. As the MPC7450 is optimized for MPX bus, the MPX bus is preferred for usage on the MPC7450 over the 60x bus.

The MPC7450 bus interface includes a 64-bit data bus with 8 bits of data parity, a 36-bit address bus with 5 bits of address parity, and additional control signals to allow for unique system level optimizations.

The bus interface protocol is configured using the $\overline{\text{BMODE0}}$ configuration signal at reset. If $\overline{\text{BMODE0}}$ is asserted at the negation of $\overline{\text{HRESET}}$, the MPC7450 uses the MPX bus protocol; if $\overline{\text{BMODE0}}$ is negated during the negation of $\overline{\text{HRESET}}$, the MPC7450 uses a limited subset of the 60x bus protocol. Note that the inverse state of $\overline{\text{BMODE}}$[0:1] at the negation of $\overline{\text{HRESET}}$ is saved in MSSCR0[BMODE].

1.2.8 MPC7450 Bus Operation Features

The MPC7450 has a separate address and data bus, each with its own set of arbitration and control signals. This allows for decoupling the data tenure from the address tenure of a transaction and provides for a wide range of system-bus implementations including:

- Nonpipelined bus operation
- Pipelined bus operation
- Split transaction operation

The MPC7450 supports only the normal memory-mapped address segments defined in the PowerPC architecture. Access to direct store segments results in a DSI exception.

1.2.8.1 MPX Bus Features

The MPX bus has the following features:

- Extended 36-bit address bus plus 5 bits of odd parity (41 bits total)
- 64-bit data bus plus 8 bits of odd parity (72 bits total)
- Support for up to 16 out-of-order transactions using four data transaction index (DTI[0:3]) signals
- Support for a four-state (MESI) cache coherence protocol
- On-chip snooping to maintain data cache coherency in multiprocessing and DMA environments

- Full data streaming
- Address pipelining
- Support for data intervention in multiprocessor systems

1.2.8.2 60x Bus Features

The following list summarizes the 60x bus interface features:

- 36-bit address bus (plus 5 bits of odd parity)
- 64-bit data bus (plus 8 bits of odd parity); a 32-bit data bus mode is not supported
- Support for a four-state (MESI) cache coherence protocol
- On-chip snooping to maintain L1 data cache, L2, and L3 cache coherency for multiprocessing applications
- Support for address-only transfers (useful for a variety of broadcast operations in multiprocessor applications)
- Support for up to 16 outstanding transactions (15 pending plus one data tenure in progress).

1.2.9 Overview of System Interface Accesses

The system interface includes address register queues, prioritization logic, and a bus control unit. The system interface latches snoop addresses for snooping in the L1 data, L2, and L3 caches, the memory hierarchy address register queues, and the reservation controlled by the Load Word and Reserve Indexed (**lwarx**) and Store Word Conditional Indexed (**stwcx.**) instructions. Accesses are prioritized with load operations preceding store operations.

Instructions are automatically fetched from the memory system into the instruction unit where they are issued to the execution units at a peak rate of three instructions per clock cycle. Conversely, load and store instructions explicitly specify the movement of operands to and from the integer, floating-point, and AltiVec register files and the memory system.

When the MPC7450 encounters an instruction or data access, it calculates the effective address and uses the lower-order address bits to check for a hit in the on-chip, 32-Kbyte L1 instruction and data caches. During L1 cache lookup, the instruction and data memory management units (MMUs) use the higher-order address bits to calculate the virtual address, from which they calculate the physical address (real address). The physical address bits are then compared with the corresponding cache tag bits to determine if a cache hit occurred in the L1 instruction or data cache.

Loads and cacheable write-back store accesses that miss in the corresponding cache are sent to the L1 load queue (LLQ). LLQ transactions are sent to the internal 256-Kbyte L2 cache and L3 cache controller simultaneously. Write-through and cache-inhibited stores and cache and synchronizing operations are sent to the L1 store queue (LSQ) regardless of

L1 data cache state. LSQ transactions are queued in the L2 cache controller and sent to the L3 cache if necessary. If no match is found in the L2 or L3 cache tags, the physical address is used to access system memory.

In addition to loads, stores, and instruction fetches, the MPC7450 performs hardware table search operations following TLB misses; L1, L2, and L3 cache castout operations; and cache-line snoop push operations when a modified cache line detects a snoop hit from another bus master.

1.2.9.1 System Interface Operation

The primary activity of the MPC7450 system interface is transferring data and instructions between the processor and system memory. There are three types of transfer 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-byte) data transfers—Generated to support caching-inhibited or write-through AltiVec loads and stores (only generated in MPX bus mode in MPC7450) and for cache inhibited instruction fetches in MPX mode.
- Four-beat burst (32 bytes) data transfers—Initiated when an entire cache block is transferred into or out of the internal caches. Because the first-level caches on the MPC7450 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.

Memory accesses can occur in single-beat (1, 2, 3, 4, and 8 bytes), double-beat (16 bytes), and four-beat (32 bytes) burst data transfers. For memory accesses, the address and data buses are independent to support pipelining and split transactions. The bus interface can pipeline as many as 16 transactions and, in MPX bus mode, supports full out-of-order split-bus transactions.

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 MPC7450 to be integrated into systems that implement various fairness and bus-parking procedures to avoid arbitration overhead.

Typically, memory accesses are weakly ordered to maximize the efficiency of the bus without sacrificing coherency of the data. The MPC7450 allows load operations to bypass store operations (except when a dependency exists). Because the processor can dynamically optimize run-time ordering of load/store traffic, overall performance is improved.

Note that the synchronize (**sync**) and enforce in-order execution of I/O (**eieio**) instructions can be used to enforce strong ordering.

This is a synchronous interface; all MPC7450 inputs are sampled and all outputs are driven on the rising edge of the bus clock cycle. The *MPC7450 RISC Microprocessor Hardware Specifications* gives timing information.

The system interface is specific for each PowerPC microprocessor implementation.

1.2.9.2 Signal Groupings

Signals are provided for implementing the bus protocol, clocking and control of the L2 and L3 caches, as well as separate L2 and L3 address and data buses. Test and control signals provide diagnostics for selected internal circuits.

The MPC7450 MPX and 60x bus interface protocol signals are grouped as follows:

- Address arbitration—The MPC7450 uses these signals to arbitrate for address bus mastership.
- Address transfer start—These signals indicate that a bus master has begun a transaction on the address bus.
- Address transfer—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—These signals provide information about the type of transfer, such as the transfer size and whether the transaction is bursted, write-through, or cache-inhibited.
- Address transfer termination—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—The MPC7450 uses these signals to arbitrate for data bus mastership.
- Data transfer—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 transfer termination—Data termination signals are required after each data beat in a data transfer. In a single-beat transaction, data termination signals also indicate 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. Data termination signals also indicate whether a condition exists that requires the data phase to be repeated.

Many other MPC7450 signals control and affect other aspects of the device, aside from the bus protocol. They are as follows:

- L3 cache address/data—The MPC7450 has separate address and data buses for accessing the L3 cache.
- L3 cache clock/control—These signals provide clocking and control for the L3 cache.

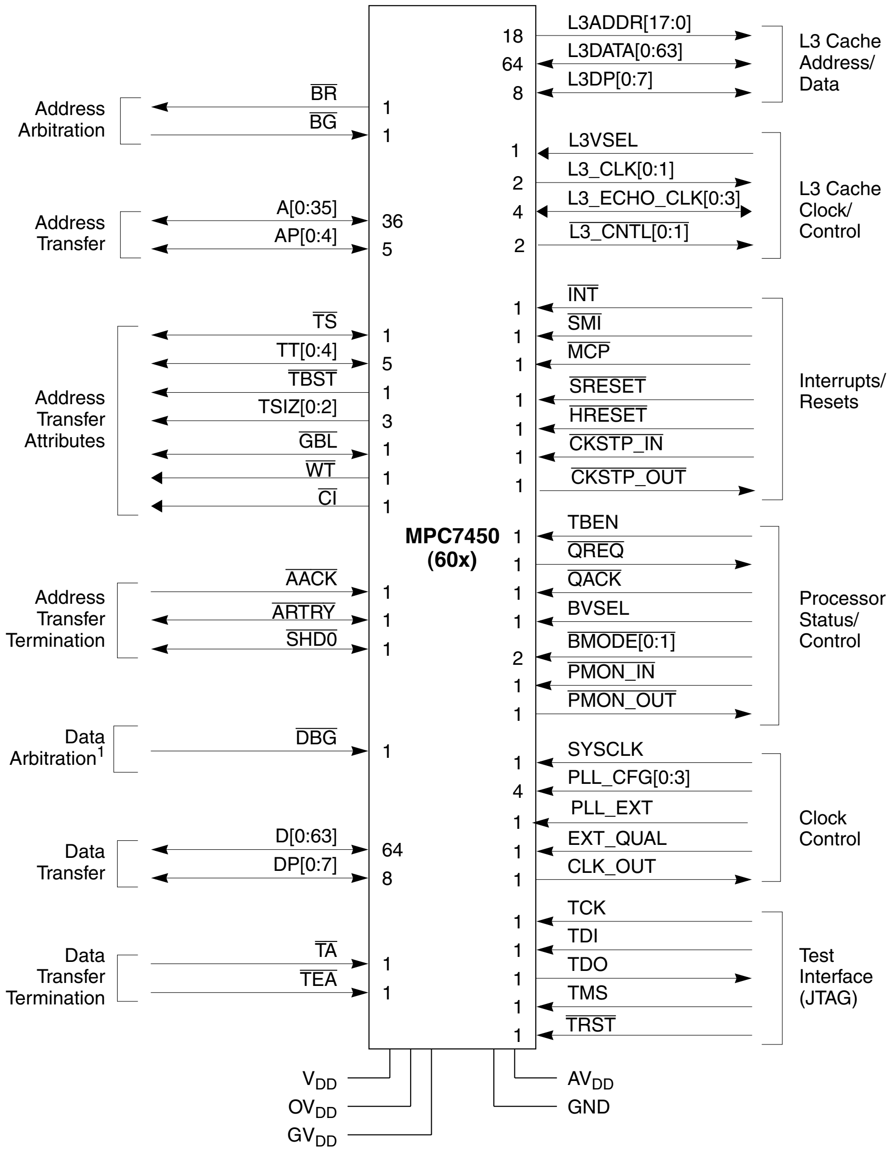
- Interrupts/resets—These signals include the external interrupt signal, checkstop signals, and both soft reset and hard reset signals. They are used to interrupt and, under various conditions, to reset the processor.
- Processor status and control—These signals enable the time-base facility and are used to select the bus mode and control sleep mode.
- Clock control—These signals determine the system clock frequency. They are also used to synchronize multiprocessor systems.
- Test interface—The JTAG (IEEE 1149.1a-1993) interface and the common on-chip processor (COP) unit provide a serial interface to the system for performing board-level boundary-scan interconnect tests.
- Voltage selection—This signal controls bus voltages of the device.

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.

1.2.9.3 MPX Bus Mode Functional Groupings

Figure 1-5 illustrates the MPC7450's signal configuration in MPX bus mode, showing how the signals are grouped. A pinout showing pin numbers is included in the *MPC7450 RISC Microprocessor Hardware Specifications*. Note that the left side of the figure depicts the signals that implement the MPX bus protocol and the right side of the figure shows the remaining signals on the MPC7450 (not part of the bus protocol).



1 The DTI[0:3] signal is not functional in 60x mode.

Figure 1-5. MPX Bus Signal Groups

Signal functionality is described in detail in Chapter 8, "Signal Descriptions," and Chapter 9, "System Interface Operation."

1.2.9.3.1 Clocking

For functional operation, the MPC7450 uses a single clock input signal, SYSCLK, from which clocking is derived for the processor core, the L3 interface, and the MPX bus interface. Additionally, internal clock information is made available at the pins to support debug and development.

The MPC7450'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 *MPC7450 RISC Microprocessor Hardware Specifications*. The bus clock is maintained at the same frequency as SYSCLK. SYSCLK does not need to be a 50% duty-cycle signal.

The MPC7450 generates the clock for the external L3 synchronous data RAMs. The clock frequency for the RAMs is divided down from (and phase-locked to) the MPC7450 core clock frequency using a divisor selected through L3CR[L3CLK].

1.3 MPC7450 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 MPC7450 as a low-power, 32-bit member of the PowerPC processor family. The structure of this section follows the user's manual organization; each subsection provides an overview of each chapter.

- Registers and programming model—Section 1.3.1, "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 MPC7450.

  Instruction set and addressing modes—Section 1.3.2, "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 MPC7450. The information in this sections is described more fully in Chapter 2, "Programming Model."
- Cache implementation—Section 1.3.3, "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 MPC7450 cache implementation. The information in this section is described more fully in Chapter 3, "L1, L2, and L3 Cache Operation."

- Exception model—Section 1.3.4, “Exception Model,” describes the exception model of the PowerPC operating environment architecture and the differences in the MPC7450 exception model. The information in this section is described more fully in Chapter 4, “Exceptions.”
- Memory management—Section 1.3.5, “Memory Management,” describes generally the conventions for memory management among the PowerPC processors. This section also describes the MPC7450’s implementation of the 32-bit PowerPC memory management specification. The information in this section is described more fully in Chapter 5, “Memory Management.”
- Instruction timing—Section 1.3.6, “Instruction Timing,” provides a general description of the instruction timing provided by the superscalar, parallel execution supported by the PowerPC architecture and the MPC7450. The information in this section is described more fully in Chapter 6, “Instruction Timing.”
- AltiVec implementation—Section 1.3.7, “AltiVec Implementation,” points out that the MPC7450 implements AltiVec registers, instructions, and exceptions as described in the *AltiVec Technology Programming Environments Manual*. Chapter 7, “The AltiVec Technology Implementation,” provides complete details.
- Power management—Section 1.3.8, “Power Management,” describes how the power management can be used to reduce power consumption when the processor, or portions of it, are idle. The information in this section is described more fully in Chapter 10, “Power and Thermal Management.”
- Thermal management—Section 1.3.9, “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. The information in this section is described more fully in Chapter 10, “Power and Thermal Management.”
- Performance monitor—Section 1.3.10, “Performance Monitor,” describes the operation of the performance monitor diagnostic tool. This functionality is fully described in Chapter 11, “Performance Monitor.”

The following sections summarize the features of the MPC7450, distinguishing those that are defined by the architecture from those that are unique to the MPC7450 implementation.

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 MPC7450 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 MPC7450 features are listed in Section 1.2, "MPC7450 Microprocessor Features."

1.3.1 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, 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 a 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 1-6 shows all the MPC7450 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. For more information, see Chapter 2, "Programming Model."

The OEA defines numerous SPRs 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 1-6, 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.

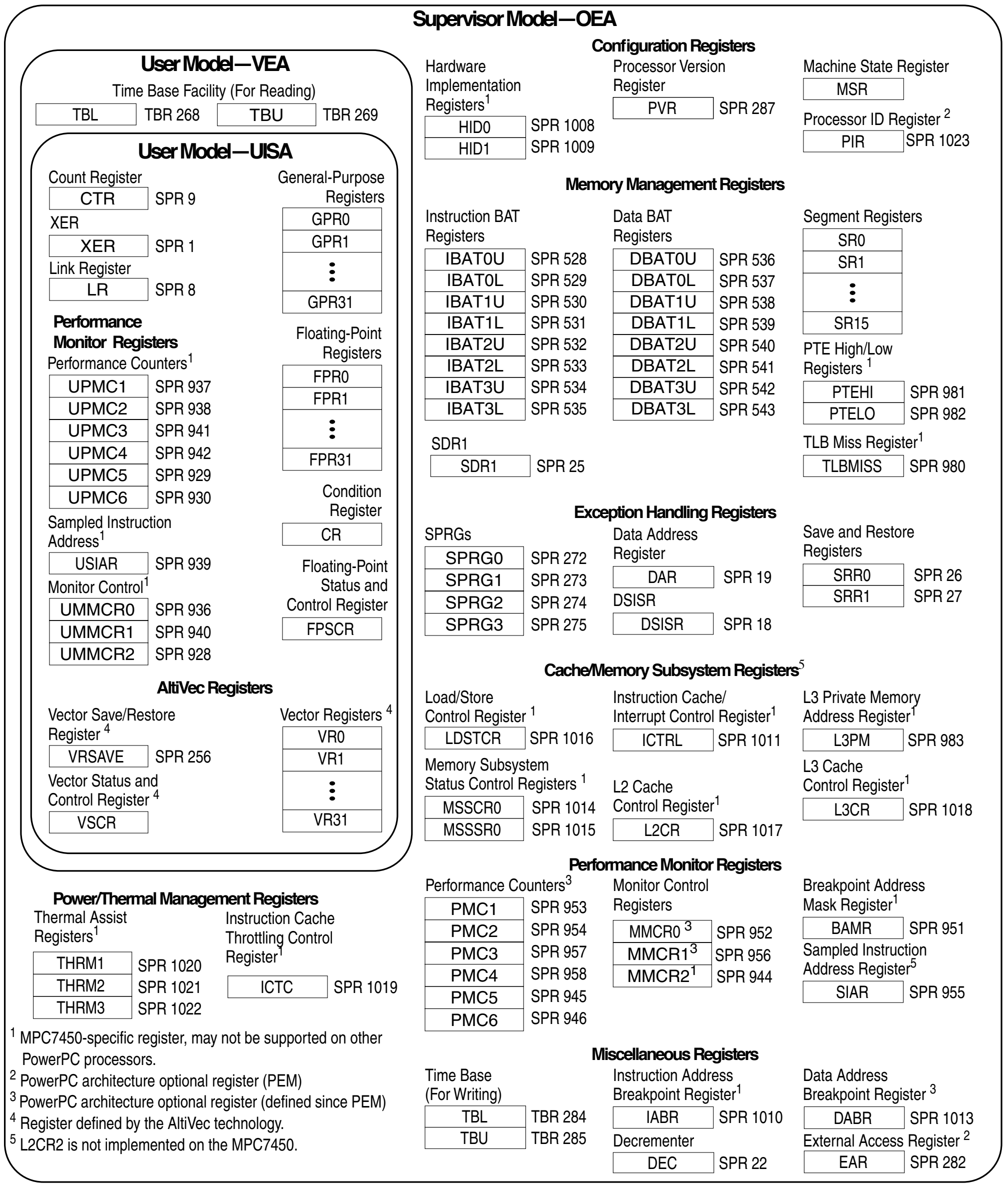


Figure 1-6. MPC7450 Microprocessor Programming Model—Registers

Some registers can be accessed both explicitly and implicitly. In the MPC7450, all SPRs are 32 bits wide. Table 1-1 describes register implemented by the MPC7450.

Table 1-1. Register Summary for the MPC7450

Name	SPR	Description	Reference
UISA Registers			
CR	—	Condition register. The 32-bit CR consists of eight 4-bit fields, CR0–CR7, that reflect results of certain arithmetic operations and provide a mechanism for testing and branching.	PEM
CTR	9	Count register. Holds a loop count that can be decremented during execution of appropriately coded branch instructions. The CTR can also provide the branch target address for the Branch Conditional to Count Register (**bcctr**x) instruction.	PEM
FPR0–FPR31	—	Floating-point registers (FPRn). The thirty-two FPRs serve as the data source or destination for all floating-point instructions.	PEM
FPSCR	—	Floating-point status and control register. Contains floating-point exception signal bits, exception summary bits, exception enable bits, and rounding control bits for compliance with the IEEE 754 standard.	PEM
GPR0–GPR31	—	General-purpose registers (GPRn). The thirty-two GPRs serve as data source or destination registers for integer instructions and provide data for generating addresses.	PEM
LR	8	Link register. Provides the branch target address for the Branch Conditional to Link Register (**bclr**x) instruction, and can be used to hold the logical address of the instruction that follows a branch and link instruction, typically used for linking to subroutines.	PEM
UMMCR0[1] UMMCR1[1] UMMCR2[1]	936, 940, 928	User monitor mode control registers (UMMCRn). Used to enable various performance monitor exception functions. UMMCRs provide user-level read access to MMCR registers.	11.3.2.1 11.3.3.1 11.3.4.1
UPMC1–UPMC6[1]	937, 938 941, 942 929, 930	User performance monitor counter registers (UPMCn). Used to record the number of times a certain event has occurred. UPMCs provide user-level read access to PMC registers.	11.3.6.1
USIAR[1]	939	User sampled instruction address register. Contains the effective address of an instruction executing at or around the time that the processor signals the performance monitor exception condition. USIAR provides user-level read access to the SIAR.	11.3.7.1
VR0–VR31[3]	—	Vector registers (VRn). Data source and destination registers for all AltiVec instructions.	7.1.1.4
VRSAVE[3]	256	Vector save/restore register. Defined by the AltiVec technology to assist application and operating system software in saving and restoring the architectural state across process context-switched events. The register is maintained only by software to track live or dead information on each AltiVec register.	7.1.1.5
VSCR[3]	—	Vector status and control register. A 32-bit vector register that is read and written in a manner similar to the FPSCR.	7.1.1.4
XER	1	Indicates overflows and carries for integer operations. **Implementation Note**—To emulate the POWER architecture **lscbx** instruction, XER[16–23] are be read with **mfspr**[XER] and written with **mtspr**[XER].	PEM

Table 1-1. Register Summary for the MPC7450 (Continued)

Name	SPR	Description	Reference
		VEA	
TBL, TBU	TBR 268, TBR 269	Time base facility. Consists of two 32-bit registers, time base upper and lower registers (TBU/TBL). These registers can be written only by supervisor-level instructions, but can be read by user- and supervisor-level software. TBU and TBL can be read from the TBR 268 and 269, respectively, with the move from time base register (**mftb**) instructions. TBU and TBL can be written to TBR 284 and 285, respectively, with the move to special purpose register (**mtspr**) instruction. Reading from SPR 284 or 285 causes an illegal instruction exception.	PEM
		OEA	
BAMR[1]	951	Breakpoint address mask register. Used in conjunction with the events that monitor IABR hits.	11.3.5
DABR[2]	1013	Data address breakpoint register. Optional register implemented in the MPC7450 and is used to cause a breakpoint exception if a specified data address is encountered.	PEM
DAR	19	Data address register. After a DSI or alignment exception, DAR is set to the effective address (EA) generated by the faulting instruction.	PEM
DEC	22	Decrementer register. A 32-bit decrementer counter used with the decrementer exception. **Implementation Note**—In the MPC7450, DEC is decremented and the time base increments at 1/4 of the system bus clock frequency.	PEM
DSISR	18	DSI source register. Defines the cause of DSI and alignment exceptions.	PEM
EAR[1]	282	External access register. Used with **eciwx** and **ecowx**. Note that the EAR and the **eciwx** and **ecowx** instructions are optional in the PowerPC architecture.	PEM
HID0[1], HID1[1]	1008, 1009	Hardware implementation-dependent registers. Control various functions, such as the power management features, and locking, enabling, and invalidating the instruction and data caches. The HID1 includes bits that reflects the state of PLL_CFG[0:3] clock signals and control other bus-related functions.	2.1.5.1 2.1.5.2
IABR[1]	1010	Instruction address breakpoint register. Used to cause a breakpoint exception if a specified instruction address is encountered.	2.1.5.6
IBAT0U/L IBAT1U/L IBAT2U/L IBAT3U/L DBAT0U/L DBAT1U/L DBAT2U/L DBAT3U/L	528, 529 530, 531 532, 533 534, 535 536, 537 538, 539 540, 541 542, 543	Block-address translation (BAT) registers. The PowerPC OEA includes an array of block address translation registers that can be used to specify four blocks of instruction space and four blocks of data space. The BAT registers are implemented in pairs: four pairs of instruction BATs (IBAT0U–IBAT3U and IBAT0L–IBAT3L) and four pairs of data BATs (DBAT0U–DBAT3U and DBAT0L–DBAT3L). Because BAT upper and lower words are loaded separately, software must ensure that BAT translations are correct during the time that both BAT entries are being loaded. The MPC7450 implements IBAT[G]; however, attempting to execute code from an IBAT area with G = 1 causes an ISI exception.	PEM, 5.3.1
ICTC[1]	1019	Instruction cache throttling control register. Has bits for enabling instruction cache throttling and for controlling the interval at which instructions are fetched. This controls overall junction temperature.	10.4

Table 1-1. Register Summary for the MPC7450 (Continued)

Name	SPR	Description	Reference
ICTRL[1]	1011	Instruction cache and interrupt control register. Used in configuring interrupts and error reporting for the instruction and data caches.	2.1.5.5.4
L2CR[1]	1017	L2 cache control register. Includes bits for enabling parity checking, setting the L2 cache size, and flushing and invalidating the L2 cache.	2.1.5.5.2
L3CR[1]	1018	L3 cache control register. Includes bits for enabling parity checking, setting the L3-to-processor clock ratio, and identifying the type of RAM used for the L3 cache implementation.	2.1.5.5.3
L3PM[1]	983	The L3 private memory register. Configures the base address of the range of addresses that the L3 uses as private memory (not cache).	2.1.5.5.5
LDSTCR[1]	1016	Load/store control register. Controls data L1 cache way-locking .	2.1.5.5.5
MMCR0[2], MMCR1[2], MMCR2[1]	952, 956, 944	Monitor mode control registers (MMCRn). Enable various performance monitor exception functions. UMMCR0–UMMCR2 provide user-level read access to these registers.	2.1.5.9.1, 11.3.2 2.1.5.9.3, 11.3.3 2.1.5.9.5, 11.3.4
MSR	—	Machine state register. Defines the processor state. The MSR can be modified by the **mtmsr**, **sc**, and **fi** instructions. It can be read by the **mfmsr** instruction. When an exception is taken, MSR contents are saved to SRR1. See Section 4.3, “Exception Processing.” The following bits are optional to the PowerPC architecture. Note that setting MSR[EE] masks decrementer and external interrupt exceptions and MPC7450-specific system management, performance monitor, and thermal management exceptions.	PEM, 4.3

Bit	Name	Description
6	VEC	AltiVec available. MPC7450 and AltiVec technology specific; optional to the PowerPC architecture. 0 AltiVec technology is disabled. 1 AltiVec technology is enabled. Note: When a non-stream AltiVec instruction accesses VRs or the VSCR when VEC = 0 an AltiVec unavailable exception is generated. This does not occur for data streaming instructions (**dst(t)**, **dstst(t)**, and **dss**); the VRs and the VSCR are available to data streaming instructions even if VEC = 0. VRSAVE can be accessed even if VEC = 0.
13	POW	Power management enable. MPC7450-specific and optional to the PowerPC architecture. 0 Power management is disabled. 1 Power management is enabled. The processor can enter a power-saving mode determined by HID0[NAP,SLEEP] when additional conditions are met. See Table 2-4.
29	PMM	Performance monitor marked mode. MPC7450-specific and optional to the PowerPC architecture. See Chapter 11, “Performance Monitor.” 0 Process is not a marked process. 1 Process is a marked process.

Name	SPR	Description	Reference
MSSCR0[1]	1014	Memory subsystem control register. Used to configure and operate many aspects of the memory subsystem.	2.1.5.3

Table 1-1. Register Summary for the MPC7450 (Continued)

Name	SPR	Description	Reference
MSSSR0[1]	1015	Memory subsystem status register. Used to configure and operate the parity functions in the L2 and L3 caches for the MPC7450.	2.1.5.4
PIR	1023	Processor identification register. Provided for system use. MPC7450 does not change PIR contents.	PEM
PMC1–PMC6[2]	953, 954 957, 958 945, 946	Performance monitor counter registers (PMCn). Used to record the number of times a certain event has occurred. UPMCs provide user-level read access to these registers.	2.1.5.9.9 11.3.6
PTEHI PTELO	981 982	The PTEHI and PTELO registers are used by the **tlbld** and **tlbli** instructions to create a TLB entry. When software table searching is enabled (HID0[STEN] = 1), and a TLB miss exception occurs, the bits of the page table entry (PTE) for this access are located by software and saved in the PTE registers.	2.1.5.7.2
PVR	287	Processor version register. Read-only register that identifies the version (model) and revision level of the PowerPC processor.	PEM, 2.1.4.1
SDA	—	Sampled data address register. The MPC7450 does not implement SDA or the user-level, read-only USDA registers.	—
SDR1	25	Sample data register. Specifies the base address of the page table entry group (PTEG) address used in virtual-to-physical address translation. **Implementation Note**—The SDR1 register has been modified (with the SDR1[HTABEXT] and SDR1[HTMEXT] fields) for the MPC7450 to support the extended 36-bit physical address (when HID0[XAEN] = 1]).	PEM, 5.5.1, 5.5.3
SIAR[2]	955	Sampled instruction address register. Contains the effective address of an instruction executing at or around the time that the processor signals the performance monitor exception condition. USIAR provides user-level read access to the SIAR.	2.1.5.9.11 11.3.7
SPRG0–SPRG3	272–275	SPRGn. Provided for operating system use.	PEM
SR0–SR15	—	Segment registers (SRn). Note that the MPC7450 implements separate instruction and data MMUs. It associates architecture-defined SRs with the data MMU. It reflects SRs values in separate, shadow SRs in the instruction MMU.	PEM
SRR0, SRR1	26, 27	Machine status save/restore registers (SRRn). Used to save the address of the instruction at which execution continues when **rfi** executes at the end of an exception handler routine. SRR1 is used to save machine status on exceptions and to restore machine status when **rfi** executes. **Implementation Note**—When a machine check exception occurs, the MPC7450 sets one or more error bits in SRR1. Refer to the individual exceptions for individual SRR1 bit settings.	PEM, 4.3
TBL, TBU	284, 285	Time base. A 64-bit structure (two 32-bit registers) that maintains the time of day and operating interval timers. The TB consists of two registers—time base upper (TBU) and time base lower (TBL). The time base registers can be written to only by supervisor-level software, but can be read by both user- and supervisor-level software. Note that reading from SPR 284 or 285 causes an illegal instruction exception.	PEM

Table 1-1. Register Summary for the MPC7450 (Continued)

Name	SPR	Description	Reference
THRM1,[1] THRM2,[1] THRM3[1]	1020, 1021, 1022	Thermal management registers (THRMn). Used to enable and set thresholds for the thermal management facility. THRM1, THRM2—Provide the ability to compare the junction temperature against two user-provided thresholds. Dual thresholds give thermal management software differing degrees of action in lowering the junction temperature. The TAU can be also operated in a single threshold mode in which the thermal sensor output is compared to only one threshold in either THRM1 or THRM2. THRM3—Used to enable the thermal management assist unit (TAU) and to control the comparator output sample time.	2.1.5.8.1
TLBMISS[1]	980	The TLBMISS register is automatically loaded when software searching is enabled (HID0[STEN] = 1) and a TLB miss exception occurs. Its contents are used by the TLB miss exception handlers (the software table search routines) to start the search process.	2.1.5.7.1

1 MPC7450-specific register may not be supported on other PowerPC processors.

2 Optional register defined by the PowerPC architecture.

3 Register is defined by the AltiVec technology.

1.3.2 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.

For more information, see Chapter 2, “Programming Model.”

1.3.2.1 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.

1.3.2.2 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, described in Section 2.5.5, “Vector Permutation and Formatting Instructions.”
- Processor control instructions—These instructions are used to read and write from the AltiVec Status and Control Register., described in Section 2.3.4.6, “Processor Control Instructions—UISA.”
- Memory control instructions—These instructions are used for managing of caches (user level and supervisor level), described in Section 2.3.5.3, “Memory Control Instructions—VEA.”

1.3.2.3 MPC7450 Microprocessor Instruction Set

The MPC7450 instruction set is defined as follows:

- The MPC7450 provides hardware support for all 32-bit PowerPC instructions.
- The MPC7450 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**)
  - — Load Data TLB Entry (**tlbld**)
  - — Load Instruction TLB Entry (**tlbli)**

1.3.3 On-Chip Cache Implementation

The following subsections describe the PowerPC architecture's treatment of cache in general, and the MPC7450-specific implementation, respectively. A detailed description of the MPC7450 cache implementation is provided in Chapter 3, "L1, L2, and L3 Cache Operation."

1.3.3.1 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.

1.3.3.2 MPC7450 Microprocessor Cache Implementation

The MPC7450 cache implementation is described in Section 1.2.4, "On-Chip Instruction and Data Caches," Section 1.2.5, "L2 Cache Implementation," and Section 1.2.6, "L3 Cache Implementation." The BPU also contains a 128-entry BTIC that provides immediate access to cached target instructions. For more information, see Section 1.2.2.2, "Branch Processing Unit (BPU)."

1.3.4 Exception Model

The following sections describe the PowerPC exception model and the MPC7450 implementation. A detailed description of the MPC7450 exception model is provided in Chapter 4, "Exceptions."

1.3.4.1 PowerPC Exception Model

The OEA portion of the PowerPC architecture defines the mechanism by which PowerPC processors implement exceptions. Exception conditions may be defined at other levels of the architecture. For example, the UISA defines conditions that may cause floating-point exceptions; the OEA defines the mechanism by which the exception is taken.

The PowerPC exception mechanism allows the processor to change to supervisor state as a result of unusual conditions arising in the execution of instructions and from external signals, bus errors, or various internal conditions. 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. Processing of exceptions begins in supervisor mode.

Although multiple exception conditions can map to a single exception vector, often a more specific condition may be determined by examining a register associated with the exception—for example, the DSISR and the floating-point status and control register (FPSCR). Also, software can explicitly enable or disable some exception conditions.

The PowerPC architecture requires that exceptions be taken in program order; therefore, although a particular implementation may recognize exception conditions out of order, they are handled strictly in order with respect to the instruction stream. When an instruction-caused exception is recognized, any unexecuted instructions that appear earlier in the instruction stream, including any that have not yet entered the execute state, are required to complete before the exception is taken. In addition, if a single instruction encounters multiple exception conditions, those exceptions are taken and handled sequentially. Likewise, exceptions that are asynchronous and precise are recognized when they occur, but are not handled until all instructions currently in the execute stage successfully complete execution and report their results.

To prevent loss of state information, exception handlers must save the information stored in the machine status save/restore registers, SRR0 and SRR1, soon after the exception is taken to prevent this information from being lost due to another exception event. Because exceptions can occur while an exception handler routine is executing, multiple exceptions can become nested. It is the exception handler's responsibility to save the necessary state information if control is to return to the excepting program.

In many cases, after the exception handler handles an exception, there is an attempt to execute the instruction that caused the exception. Instruction execution continues until the next exception condition is encountered. Recognizing and handling exception conditions sequentially guarantees that the machine state is recoverable and processing can resume without losing instruction results.

In this book, the following terms are used to describe the stages of exception processing:

- Recognition—Exception recognition occurs when the condition that can cause an exception is identified by the processor.
- Taken—An exception is said to be taken when control of instruction execution is passed to the exception handler; that is, the context is saved and the instruction at the appropriate vector offset is fetched and the exception handler routine begins executing in supervisor mode.
- Handling—Exception handling is performed by the software at the appropriate vector offset. Exception handling is begun in supervisor mode.

In this book, the term 'interrupt' is used to describe the external interrupt, the system management interrupt, and sometimes the asynchronous exceptions, in general. Note that the PowerPC architecture uses the word 'exception' to refer to IEEE-defined floating-point exception conditions that may cause a program exception to be taken; see Section 4.6.7, "Program Exception (0x00700)." The occurrence of these IEEE exceptions may or may not cause an exception to be taken. IEEE-defined exceptions are referred to as IEEE floating-point exceptions or floating-point exceptions in this book.

1.3.4.2 MPC7450 Microprocessor Exceptions

As specified by the PowerPC architecture, exceptions can be either precise or imprecise and either synchronous or asynchronous. Asynchronous exceptions are caused by events external to the processor's execution; synchronous exceptions are caused by instructions.

The types of exceptions are shown in Table 1-2. Note that all exceptions except for the performance monitor, AltiVec unavailable, instruction address breakpoint, system management, AltiVec assist, thermal management, and the three software table search exceptions are described in Chapter 6, "Exceptions," in *The Programming Environments Manual*.

Table 1-2. MPC7450 Microprocessor Exception Classifications

Synchronous/Asynchronous	Precise/Imprecise	Exception Types
Asynchronous, nonmaskable	Imprecise	System reset, machine check
Asynchronous, maskable	Precise	External interrupt, system management interrupt, decrementer exception, performance monitor exception, thermal management exception
Synchronous	Precise	Instruction-caused exceptions

The exception classifications are discussed in greater detail in Section 4.2, "MPC7450 Exception Recognition and Priorities." For a better understanding of how the MPC7450 implements precise exceptions, see Chapter 6, "Instruction Timing." Exceptions implemented in the MPC7450, and conditions that cause them, are listed in Table 1-3. Table 1-3 notes when an exception is MPC7450-specific.

The three software table search exceptions support the software page table searching and are enabled by setting HID0[STEN]. See Section 4.6.15, "TLB Miss Exceptions," and Chapter 5, "Memory Management."

Table 1-3. Exceptions and Conditions

Exception Type	Vector Offset	Causing Conditions
Reserved	0x00000	—
System reset	0x00100	Assertion of either $\overline{\text{HRESET}}$ or $\overline{\text{SRESET}}$ or at power-on reset
Machine check	0x00200	Assertion of $\overline{\text{TEA}}$ during a data bus transaction, assertion of $\overline{\text{MCP}}$, an address bus parity error on MPX bus, a data bus parity error on MPXbus, an L1 instruction cache error, and L1 data cache error, a memory subsystem detected error including the following: • L2 data parity error • L2 cache tag parity error • L3 SRAM error • L3 tag parity errors. MSR[ME] must be set.
DSI	0x00300	As specified in the PowerPC architecture. Also includes the following: • A hardware table search due to a TLB miss on load, store, or cache operations results in a page fault. • Any load or store to a direct-store segment (SR[T] = 1). • A **lwarx** or **stwcx.** instruction to memory with cache-inhibited or write-through memory/cache access attributes.
ISI	0x00400	As specified in the PowerPC architecture
External interrupt	0x00500	MSR[EE] = 1 and $\overline{\text{INT}}$ is asserted
Alignment	0x00600	• 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 • An operand of a **dcbz** instruction is on a page that is write-through or cache-inhibited for a virtual mode access. • An attempt to execute a **dcbz** instruction occurs when the cache is disabled or locked.
Program	0x00700	As specified in the PowerPC architecture

Table 1-3. Exceptions and Conditions (Continued)

Exception Type	Vector Offset	Causing Conditions
Floating-point unavailable	0x00800	As specified in the PowerPC architecture
Decrementer	0x00900	As defined by the PowerPC architecture, when the msb of the DEC register changes from 0 to 1 and MSR[EE] = 1
Reserved	0x00A00–00BFF	—
System call	0x00C00	Execution of the System Call (**sc**) instruction
Trace	0x00D00	MSR[SE] =1 or a branch instruction is completing and MSR[BE] =1. The MPC7450 operates as specified in the OEA by taking this exception on an **isync**.
Reserved	0x00E00	The MPC7450 does not generate an exception to this vector. Other PowerPC processors may use this vector for floating-point assist exceptions.
Reserved	0x00E10–00EFF	—
Performance monitor	0x00F00	The limit specified in PMCn is met and MMCR0[ENINT] = 1 (MPC7450-specific)
AltiVec unavailable	0x00F20	Occurs due to an attempt to execute any non-streaming AltiVec instruction when MSR[VEC] = 0. This exception is not taken for data streaming instructions (**dstx**, **dss,** or **dssall**). (MPC7450-specific)
ITLB miss	0x01000	An instruction translation miss exception is caused when HID0[STEN] = 1 and the effective address for an instruction fetch cannot be translated by the ITLB (MPC7450-specific).
DTLB miss-on-load	0x01100	A data load translation miss exception is caused when HID0[STEN] = 1 and the effective address for a data load operation cannot be translated by the DTLB (MPC7450-specific).
DTLB miss-on-store	0x01200	A data store translation miss exception is caused when HID0[STEN] = 1 and the effective address for a data store operation cannot be translated by the DTLB, or when a DTLB hit occurs, and the changed bit in the PTE must be set due to a data store operation (MPC7450-specific).
Instruction address breakpoint	0x01300	IABR[0–29] matches EA[0–29] of the next instruction to complete and IABR[BE] = 1 (MPC7450-specific)
System management interrupt	0x01400	MSR[EE] = 1 and $\overline{\text{SMI}}$ is asserted (MPC7450-specific)
Reserved	0x01500–015FF	—
AltiVec assist	0x01600	This MPC7450-specific exception supports denormalization detection in Java mode as specified in the AltiVec Technology Programming Environments Manual.
Thermal management	0x01700	Generated when the thermal management assist unit detects that the temperature has exceeded the programmed threshold. (MPC7450-specific)
Reserved	0x01800–02FFF	—

1.3.5 Memory Management

The following subsections describe the memory management features of the PowerPC architecture, and the MPC7450 implementation, respectively.

1.3.5.1 PowerPC Memory Management Model

The primary function of the MMU in a PowerPC processor is the translation of logical (effective) addresses to physical addresses (referred to as real addresses in the architecture specification) for memory accesses and I/O accesses (I/O accesses are assumed to be memory-mapped). In addition, the MMU provides access protection on a segment, block, or page basis. Note that the MPC7450 does not implement the optional direct-store facility.

Two general types of memory accesses generated by PowerPC processors require address translation—instruction accesses and data accesses generated by load and store instructions. In addition, the addresses specified by cache instructions and the optional external control instructions also require translation. Generally, the address translation mechanism is defined in terms of the segment descriptors and page tables that the PowerPC processors use to locate the effective-to-physical address mapping for memory accesses. The segment information translates the effective address to an interim virtual address, and the page table information translates the virtual address to a physical address.

The segment descriptors, used to generate the interim virtual addresses, are stored as on-chip segment registers on 32-bit implementations (such as the MPC7450). In addition, two translation lookaside buffers (TLBs) are implemented on the MPC7450 to keep recently used page address translations on-chip. Although the PowerPC OEA describes one MMU (conceptually), the MPC7450 hardware maintains separate TLBs and table search resources for instruction and data accesses that can be performed independently (and simultaneously). Therefore, the MPC7450 is described as having two MMUs, one for instruction accesses (IMMU) and one for data accesses (DMMU).

The block address translation (BAT) mechanism is a software-controlled array that stores the available block address translations on-chip. BAT array entries are implemented as pairs of BAT registers that are accessible as supervisor special-purpose registers (SPRs). There are separate instruction and data BAT mechanisms. In the MPC7450, they reside in the instruction and data MMUs, respectively.

The MMUs, together with the exception processing mechanism, provide the necessary support for the operating system to implement a paged virtual memory environment and for enforcing protection of designated memory areas. Section 4.3, “Exception Processing,” describes how the MSR controls critical MMU functionality.

1.3.5.2 MPC7450 Microprocessor Memory Management Implementation

The MPC7450 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 MPC7450 MMU is described in Section 1.2.3, “Memory Management Units (MMUs).”

The MPC7450 implements the memory management specification of the PowerPC OEA for 32-bit implementations but adds capability for supporting 36-bit physical addressing. Thus, it provides 4 Gbytes of physical address space accessible to supervisor and user programs, with a 4-Kbyte page size and 256-Mbyte segment size. In addition, the MPC7450 MMUs use an interim virtual address (52 bits) and hashed page tables in the generation of 32-bit or 36-bit physical addresses (depending on the setting of HID0[XAEN]). PowerPC processors also have a BAT mechanism for mapping large blocks of memory. Block range from 128 Kbytes to 256 Mbytes and are software programmable.

The MPC7450 provides table search operations performed in hardware. The 52-bit virtual address is formed and the MMU attempts to fetch the PTE that contains the physical address from the appropriate TLB on-chip. If the translation is not found in either the BAT array or in a TLB (that is, a TLB miss occurs), the hardware performs a table search operation (using a hashing function) to search for the PTE. Hardware table searching is the default mode for the MPC7450; however, if HID0[STEN] = 1, a software table search is performed.

The MPC7450 also provides support for table search operations performed in software (if HID0[STEN] is set). In this case, the TLBMISS register saves the effective address of the access that requires a software table search. The PTEHI and PTELO registers and the **tlbli** and **tlbld** instructions are used in reloading the TLBs during a software table search operation. The following exceptions support software table searching if HID0[STEN] is set and a TLB miss occurs:

- For an instruction fetch, an ITLB miss exception,
- For a data load, an DTLB miss-on-load exception,
- For a data store, an DTLB miss-on-store exception.

The MPC7450 implements the optional TLB invalidate entry (**tlbie**) and TLB synchronize (**tlbsync**) instructions that can be used to invalidate TLB entries. For more information on the **tlbie** and **tlbsync** instructions, see Section 5.4.4.2, “TLB Invalidation.”

1.3.6 Instruction Timing

This section describes how the MPC7450 microprocessor performs operations defined by instructions and how it reports the results of instruction execution. It gives detailed descriptions of how the MPC7450 execution units work and how those units interact with other parts of the processor, such as the instruction fetching mechanism, register files, and caches. It gives examples of instruction sequences, showing potential bottlenecks and how to minimize their effects. Finally, it includes tables that identify the unit that executes each instruction, the latency for each instruction, and other information useful to assembly language programmers.

The MPC7450 design minimizes average instruction execution latency, which is the number of clock cycles it takes to fetch, decode, dispatch, issue, and execute instructions and make results available for subsequent instructions. Some instructions, such as loads and stores, access memory and require additional clock cycles between the execute phase and the write-back phase. Latencies depend on whether an access is to cacheable or noncacheable memory, whether it hits in the L1, L2, or L3 cache, whether a cache access generates a write back to memory, whether the access causes a snoop hit from another device that generates additional activity, and other conditions that affect memory accesses.

To improve throughput, the MPC7450 implements pipelining, superscalar instruction issue, branch folding, removal of fall-through branches, three-level speculative branch handling, and multiple execution units that operate independently and in parallel.

As an instruction passes from stage to stage, the subsequent instruction can follow through the stages as the former instruction vacates them, allowing several instructions to be processed simultaneously. Although it may take several cycles for an instruction to pass through all the stages, when the pipeline is full, one instruction can complete its work on every clock cycle. Figure 1-7 represents a generic four-stage pipelined execution unit, which when filled has a throughput of one instruction per clock cycle.

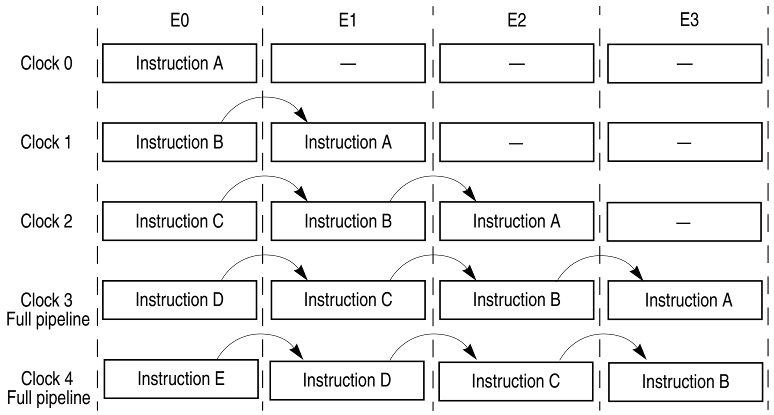


Figure 1-7. Pipelined Execution Unit

Figure 1-8 shows the entire path that instructions take through the fetch1, fetch2, decode/dispatch, execute, issue, complete, and write-back stages, which is considered the MPC7450's master pipeline. The FPU, LSU, IU2, VIU2, VFPU, and VPU are multiple-stage pipelines.

The MPC7450 contains the following execution units:

- Branch processing unit (BPU)
- Three integer unit 1s (IU1a, IU1b, and IU1c)—execute all integer instructions except multiply, divide, and move to/from SPR instructions.
- Integer unit 2 (IU2)—executes miscellaneous instructions including the CR logical operations, integer multiplication and division instructions, and move to/from special-purpose register instructions
- 64-bit floating-point unit (FPU)
- Load/store unit (LSU)
- The AltiVec unit contains the following four independent execution units for vector computations and the latencies are shown in Table 7-10
  - AltiVec permute unit (VPU)
  - AltiVec integer unit 1 (VIU1)
  - Vector integer unit 2 (VIU2)
  - Vector floating-point unit (VFPU)

  A maximum of two AltiVec instructions can be issued in order to any combination of AltiVec execution units per clock cycle. Moreover, the VIU2, VFPU, and VPU are pipelined, so they can operate on multiple instructions.

The MPC7450 can complete as many as three instructions on each clock cycle. In general, the MPC7450 processes instructions in seven stages—fetch1, fetch2, decode/dispatch, issue, execute, complete, and writeback as shown in Figure 1-8. Note that the pipeline example in Figure 6-1 is similar to the four-stage VFPU pipeline in Figure 1-8.

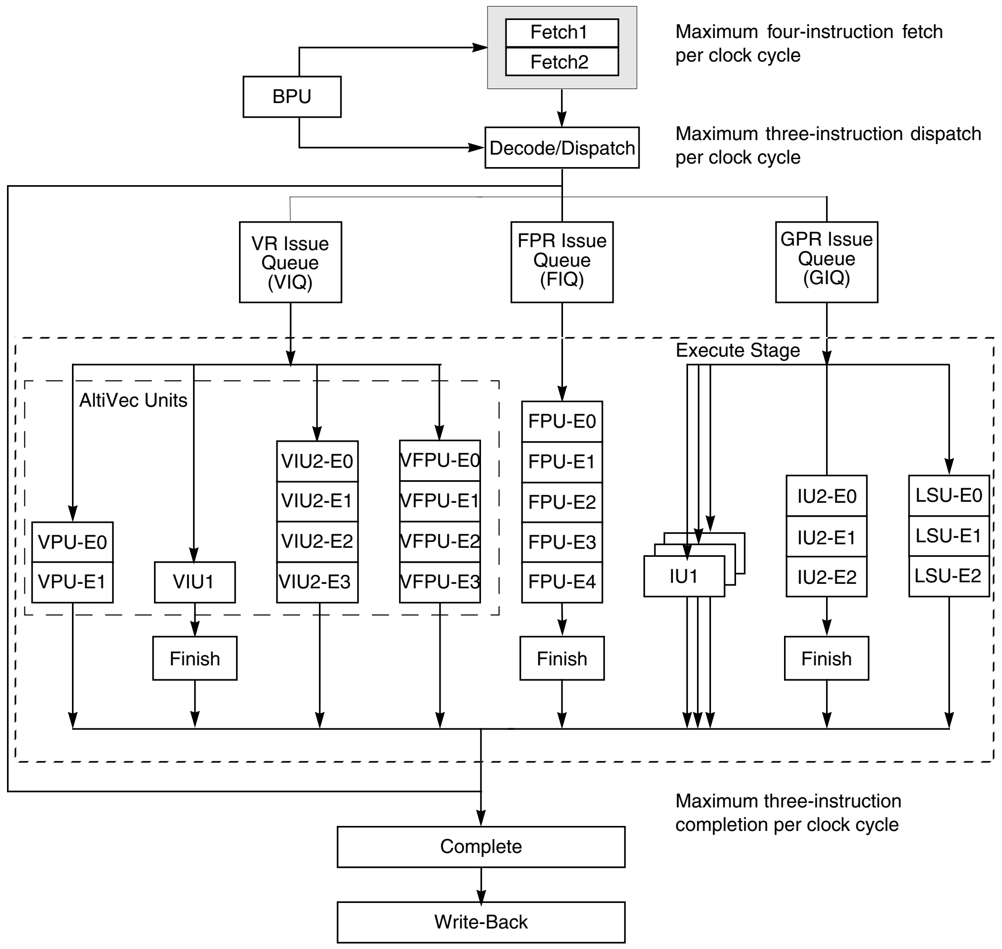


Figure 1-8. Superscalar/Pipeline Diagram

The instruction pipeline stages are described as follows:

- Instruction fetch—Includes the clock cycles necessary to request an instruction and the time the memory system takes to respond to the request. Instructions retrieved are latched into the instruction queue (IQ) for subsequent consideration by the dispatcher.

  Instruction fetch timing depends on many variables, such as whether an instruction is in the branch target instruction cache (BTIC), the on-chip instruction cache, or the L2 or L3 cache. Those factors increase when it is necessary to fetch instructions from system memory and include the processor-to-bus clock ratio, the amount of bus traffic, and whether any cache coherency operations are required.

- The decode/dispatch stage fully decodes each instruction; most instructions are dispatched to the issue queues (branch, **isync**, **rfi**, and **sc** instructions do not go to issue queues).
- The three issue queues, FIQ, VIQ, and GIQ, can accept as many as one, two, and three instructions, respectively, in a cycle. Instruction dispatch requires the following:
  - — Instructions are dispatched only from the three lowest IQ entries—IQ0, IQ1, and IQ2.
  - — A maximum of three instructions can be dispatched to the issue queues per clock cycle.
  - — Space must be available in the CQ for an instruction to dispatch (this includes instructions that are assigned a space in the CQ but not an issue queue).
- The issue stage reads source operands from rename registers and register files and determines when instructions are latched into the execution unit reservation stations. The GIQ, FIQ, and VIQ (AltiVec) issue queues have the following similarities:
  - – Operand lookup in the GPRs, FPRs, and VRs, and their rename registers.
  - – Issue queues issue instructions to the proper execution units.
  - – Each issue queue holds twice as many instructions as can be dispatched to it in 1 cycle; the GIQ has six entries, the VIQ has four, and the FIQ has two.

  The three issue queues are described as follows:
  - — The GIQ accepts as many as three instructions from the dispatch unit each cycle. IU1, IU2, and all LSU instructions (including floating-point and AltiVec loads and stores) are dispatched to the GIQ.
  - — Instructions can be issued out-of-order from the bottom three GIQ entries (GIQ2–GIQ0). An instruction in GIQ1 destined for an IU1 does not have to wait for an instruction in GIQ0 that is stalled behind a long-latency integer divide instruction in the IU2.
  - — The VIQ accepts as many as two instructions from the dispatch unit each cycle. All AltiVec instructions (other than load, store, and vector touch instructions) are dispatched to the VIQ. As many as two instructions can be issued to the four AltiVec execution units, but unlike the GIQ, instructions in the VIQ cannot be issued out of order.
  - — The FIQ can accept one instruction from the dispatch unit per clock cycle. It looks at the first instruction in its queue and determines if the instruction can be issued to the FPU in this cycle.
- The execute stage accepts instructions from its issue queue when the appropriate reservation stations are not busy. In this stage, the operands assigned to the execution stage from the issue stage are latched.

  The execution unit executes the instruction (perhaps over multiple cycles), writes results on its result bus, and notifies the CQ when the instruction finishes. The execution unit reports any exceptions to the completion stage. Instruction-generated exceptions are not taken until the excepting instruction is next to retire.

Most integer instructions have a 1-cycle latency, so results of these instructions are available 1 clock cycle after an instruction enters the execution unit. The FPU, LSU, IU2, VIU2, VFPU, and VPU units are pipelined, as shown in Figure 7-3.

Note that AltiVec computational instructions are executed in the four independent, pipelined AltiVec execution units. The VPU has a two-stage pipeline, the VIU1 has a one-stage pipeline, and the VIU2 and VFPU have four-stage pipelines. As many as 10 AltiVec instructions can be executing concurrently.

- The complete and write-back stages maintain the correct architectural machine state and commit results to the architected registers in the proper order. If completion logic detects an instruction containing an exception status, all following instructions are cancelled, their execution results in rename buffers are discarded, and the correct instruction stream is fetched.

  The complete stage ends when the instruction is retired. Three instructions can be retired per clock cycle. If no dependencies exist, as many as three instructions are retired in program order. Section 6.7.4, "Completion Unit Resource Requirements," describes completion dependencies.

The write-back stage occurs in the clock cycle after the instruction is retired.

1.3.7 AltiVec Implementation

The MPC7450 implements the AltiVec registers and instruction set as they are described by the *AltiVec Technology Programming Environments Manual*. Two additional implementation specific exceptions have been added, that is:

- the AltiVec assist exception which is used in handling denormalized numbers in Java mode and
- an alignment exception for cache-inhibited AltiVec loads and stores and write-through stores that execute when in 60x bus mode

Both exceptions are described fully in Chapter 4, "Exceptions." Also, the default setting for VSCR[NJ] bit has changed from being non-Java compliant (VSCR[NJ] = 1) in the MPC7400/7410 to having a default setting of Java–compliant (VSCR[NJ] = 0) in the MPC7450. The AltiVec implementation is described fully in Chapter 7, "The AltiVec Technology Implementation."

1.3.8 Power Management

The MPC7450 has two software-controllable power-saving modes, nap and sleep, which progressively reduce power dissipation. Nap mode also implements a doze state to allow the processor to snoop transactions on the MPX/60x bus then return to nap.

The MPC7450 has four power states, two of which are selectable through the MSR and HID0 registers. The four power modes are as follows:

- Full-power (default). The MPC7450 is fully powered and the internal functional units operate at the full processor clock speed. If dynamic power management mode is enabled, idle functional units automatically enter a low-power state without affecting performance, software execution, or external hardware.
- Nap. Instruction fetch and dispatch stop and bus snooping is disabled. Only the decrementer/time base registers, the thermal assist unit, the PLL, and JTAG logic remain in a powered state. The MPC7450 returns to the full-power state upon receipt of an external asynchronous interrupt, a system management interrupt, a hard or soft reset, or a machine check input ($\overline{\text{MCP}}$), a decrementer exception, or a thermal assist unit exception. Return to full-power state from a nap state takes only a few processor clock cycles. When the processor is in nap mode, the system causes the processor to enter doze state by negating $\overline{\text{QACK}}$. In doze state, the processor snoops bus operations and performs appropriate memory coherency operations. The system return to nap mode when $\overline{\text{QACK}}$ reasserts and the processor has completed any memory operations.
- Sleep. All internal functional units are disabled. The PLL remains locked and running but no clocks propagate to functional units. Note that the time base is not operational in sleep mode. It must be reset by an external source upon exiting Sleep. The MPC7450 returns to the full-power state upon receipt of an external asynchronous interrupt, a system management interrupt, a hard or soft reset, or assertion of $\overline{\text{MCP}}$. Return to full-power state from a Sleep state takes only a few processor clock cycles
- Deep sleep. The system can disable the PLL and in turn disable the SYSCLK source. Power-on reset procedures for restarting and relocking the PLL must be followed on exiting deep sleep. Returning the MPC7450 to full-power state requires enabling of the PLL and SYSCLK. After the time required to relock the PLL, one of the following must occur: an external asynchronous interrupt, a system management interrupt, a hard or soft reset, or assertion of $\overline{\text{MCP}}$.

Chapter 10, “Power and Thermal Management,” describes power saving and thermal management modes for the MPC7450.

1.3.9 Thermal Management

The MPC7450’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 MPC7450'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 1.3.1, "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 MPC7450's overall junction temperature by determining the interval at which instructions are fetched. This feature is accessed through the ICTC register.

Chapter 10, "Power and Thermal Management," provides information about power saving and thermal management modes for the MPC7450.

1.3.10 Performance Monitor

The MPC7450 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 1.3.1, “PowerPC Registers and Programming Model.” Performance monitor control registers, MMCR0, MMCR1, and MMCR2 can be used to specify which events are to be counted and the conditions for which a performance monitoring exception is taken. Additionally, the sampled instruction address register, SIAR (USIAR), 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 monitor exception occurs, program execution continues from vector offset 0x00F00.

Chapter 11, “Performance Monitor,” describes the operation of the performance monitor diagnostic tool incorporated in the MPC7450.

1.4 Differences between the MPC7450 and the MPC7400/7410

Table 1-4 compares the key features of the MPC7450 with the earlier MPC7400/MPC7410. To achieve a higher frequency, the number of logic levels per clock cycle is reduced. In addition, the pipeline of the MPC7450 is extended (compared to the MPC7400), while maintaining the same level of performance (in terms of number of instructions executed per clock cycle. Table 1-8 shows these differences.

Table 1-4. MPC7450 and MPC7400/MPC7410 Feature Comparison

Microarchitectural Feature	MPC7450	MPC7400/MPC7410
Basic Pipeline Functions		
Pipeline stages up to execute	5	3
Total pipeline stages (minimum)	7	4
Pipeline maximum instruction throughput	3 + branch	2 + branch
Pipeline Resources		
Instruction queue size	12	6
Completion queue size	16	8
Renames (GPR, FPR, VR)	16, 16, 16	6, 6, 6

Table 1-4. MPC7450 and MPC7400/MPC7410 Feature Comparison (Continued)

Microarchitectural Feature	MPC7450	MPC7400/MPC7410
Maximum Execution Throughput		
Short-latency integer units (IU1s)	3	2
Vector units	2 (any 2 of 4 units)	2 (permute/integer)
Floating-point unit	1	1
Out-of-Order Window Size in Execution Queues		
Short-latency integer units	1 entry x 3 queues	1 entry x 2 queues
Vector units	In order, 4 queues	In order, 2 queues
Floating-point unit	In order	In order
Branch Processing Resources		
Prediction structures	BTIC, BHT, link stack	BTIC, BHT
BTIC size, associativity	128 entry, 4 way	64 entry, 4 way
BHT size	2K entry	512 entry
Link stack depth	8	none
Unresolved branches supported	3	2
Branch taken penalty (BTIC hit)	1	0
Minimum misprediction penalty	6	4
Execution Unit Timings (Latency-Throughput)		
Aligned load (integer, float, vector)	3-1, 4-1, 3-1	2-1, 2-1, 2-1
Misaligned load (integer, float, vector)	4-2, 5-2, 4-2	3-2, 3-2, 3-2
L1 miss, L2 hit latency	6 (9)	9 (11)[1]
IU1s (adds, subs, shifts, rotates, compares, logicals)	1-1	1-1
Integer multiply (32x8, 32x16, 32x32)	3-1, 3-1, 4-2	2-1, 3-2, 5-4
Scalar floating-point	5-1	3-1
VIU1 (vector integer unit 1—shorter latency vector integer)	1-1	1-1
VIU2 (vector integer unit 2—longer latency vector integer)	4-1	3-1
VFPU (vector floating-point)	4-1	4-1
VPU (vector permute)	2-1	1-1
Caches		
MMUs (instruction and data)	128 entry, 2 way	128 entry, 2 way
Table search mechanism	Hardware and software	Hardware

Table 1-4. MPC7450 and MPC7400/MPC7410 Feature Comparison (Continued)

Microarchitectural Feature	MPC7450	MPC7400/MPC7410
L1 Instruction Cache/Date Cache Features		
Size	32K/32K	32K/32K
Associativity	8 way	8 way
Locking granularity/style	4-Kbyte/way	Full cache
Parity on instruction cache	Word	None
Parity on data cache	Byte	None
Number of data cache misses (load/store)	5/1	8 (any combination)
Data stream touch engines	4 streams	4 streams
On-Chip L2 Cache Features		
Cache level	L2	Tags only (see off-chip cache support below)
Size/associativity	256 Kbyte/8 way	
Access width	256 bits	
Number of 32 byte sectors/line	2	
Parity	Byte	
Off-Chip Cache Support		
Cache level	L3	L2
On-chip tag logical size	1 Mbyte, 2 Mbyte	0.5 Mbyte, 1 Mbyte, 2 Mbyte
Associativity	8 way	2 way
Number of 32-byte sectors/line	2, 4	1, 2, 4
Off-chip data SRAM support	MSUG2 DDR, LW, PB2	LW, PB2, PB3
Data path width	64	64
Private memory SRAM sizes	1 Mbyte, 2 Mbyte	0.5 Mbyte, 1 Mbyte, 2 Mbyte
Parity	Byte	Byte

[1] Numbers in parentheses are for 2:1 SRAM.