MPR602TSU-01
(IBM Order Number)

MPC602/D
(Motorola Order Number)

2/95

PowerPC™

Advance Information

PowerPC 602™ RISC Microprocessor Technical Summary

This document provides an overview of the PowerPC 602 microprocessor features, including a block diagram showing the major functional components. It provides information about how the 602 implementation complies with the PowerPC Architecture™ definition. This document is divided into two parts:

- Part 1,"PowerPC 602 Microprocessor Overview," provides an overview of the 602 features, including a block diagram showing the major functional components.
- Part 2, "PowerPC 602 Microprocessor: Implementation," gives specific details about the implementation of the 602 as a low-power, low-cost, 32-bit member of the PowerPC™ processor family.

In this document, the term "602" is used as an abbreviation for the phrase, "PowerPC 602 microprocessor." The PowerPC 602 microprocessors are available from IBM as PPC602 and from Motorola as MPC602.

IBM Microelectronics

MOTOROLA

Part 1 PowerPC 602 Microprocessor Overview

This section describes the features of the 602, provides a block diagram showing the major functional units, and gives an overview of how the 602 operates.

The 602 is a low-cost, low-power implementation of the PowerPC microprocessor family of reduced instruction set computer (RISC) microprocessors. The 602 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. Floating-point operations involving 64-bit data types in single-precision format are supported; however, floating-point operations involving 64-bit data types in double-precision format are not implemented in hardware on the 602 and are instead trapped for emulation in software. For more information about how the architecture defines addressing and data types, see *PowerPC Microprocessor Family: The Programming Environments* (also referred to as *The Programming Environments Manual*).

The 602 provides dynamic and static power-saving modes. The three static modes—nap, doze, and sleep—progressively reduce the amount of power dissipated by the processor. Dynamic power management mode allows the processor to reduce power consumption by providing clocking only to those functional units that are active, without affecting operational performance, software execution, or external hardware.

The 602 can simultaneously fold one branch instruction and dispatch one non-branch instruction per clock cycle to any one of three execution units. Instructions can execute out of order; however, the instructions complete and write back in program order.

The 602 integrates four execution units—an integer unit (IU), a floating-point unit (FPU), a branch processing unit (BPU), and a load/store unit (LSU). The ability to execute four instructions in parallel and the use of simple instructions with rapid execution times yield high efficiency and throughput for 602-based systems. Most integer instructions execute in one clock cycle. The FPU is pipelined such that when the FPU pipeline is full, a single-precision multiply-add instruction can complete every clock cycle.

The 602 provides independent on-chip, 4-Kbyte, two-way set-associative, physically addressed caches for instructions and data and on-chip instruction and data memory management units (MMUs). The 602 MMUs contain 32-entry, two-way set-associative, data and instruction translation lookaside buffers (DTLB and ITLB). The TLBs and caches use a least recently used (LRU) replacement algorithm. The 602 also supports block address translation through the use of two independent instruction and data block address translation (IBAT and DBAT) arrays of four entries each. Effective addresses are compared simultaneously with all four entries in the BAT array during block translation. If an effective address matches any entry in the BATs, the BAT entry takes priority over any potential matches in the TLBs.

Applications that do not require address translation through the TLBs defined by the PowerPC architecture can use the 602's protection-only mode to protect up to 4 Mbytes of memory per TLB. However, in protection-only mode, the 602 BATs remain available for protection and translation of the effective address. For details, refer to Section 1.5.1, "Memory Management Units (MMUs)."

The 602 has a single bus interface used for transferring both 32-bit addresses and 64-bit data. This bus is time-multiplexed, as described in Section 2.8, "System Interface." The 602 interface protocol allows multiple masters to compete for system resources through a central external arbiter. The 602 provides a three-state coherency protocol that supports the modified, exclusive, and invalid (MEI) cache states. This protocol is a compatible subset of the MESI (modified/exclusive/shared/invalid) four-state protocol and operates coherently in systems that contain four-state caches. The 602 supports single-beat and burst data transfers for memory accesses and memory-mapped I/O.

The 602 uses an advanced, 3.3-V CMOS process technology and maintains full interface compatibility with TTL devices.

1.1 PowerPC 602 Microprocessor Features

This section describes details of the 602's implementation of the PowerPC architecture. Major features of the 602 are as follows:

- High-performance microprocessor with parallel execution units
  - — One instruction is fetched from the instruction queue per clock
  - — One instruction can be issued and one retired per clock
  - — As many as four instructions in execution per clock
  - — Single-cycle execution for most instructions
- Four independent execution units and two register files
  - — Branch processing unit (BPU)
    - – Zero-cycle branch capability (branch folding)
    - – Programmable static branch prediction on unresolved conditional branches
    - – BPU that performs CR-lookahead operations
  - — A 32-bit integer unit (IU)
    - – Thirty-two 32-bit general-purpose registers (GPRs) for integer operands
  - — A 32-bit floating-point unit (FPU)
    - – Fully IEEE 754-compliant FPU for single-precision operations
    - – Emulation support for double-precision operations
    - – An implementation of the non-IEEE floating-point mode
    - – Thirty-two 32-bit floating-point registers (FPRs) for single-precision operands
  - — LSU for data transfer between data cache and GPRs and FPRs
- Instruction pipelining
  - — Instruction unit capable of simultaneously folding out a branch instruction and dispatching one instruction per clock from the instruction queue
  - — A four-entry instruction queue that provides lookahead capability
  - — Independent pipelines with feed-forwarding that reduces data dependencies in hardware
- Separate caches for instructions and data (Harvard architecture)
  - — 4-Kbyte data cache—two-way set-associative, physically addressed; LRU replacement algorithm
  - — 4-Kbyte instruction cache—two-way set-associative, physically addressed; LRU replacement algorithm
  - — Cache write-back or write-through operation programmable on a per page or per block basis
- Memory management features
  - — Address translation facilities for 4-Kbyte page size, variable block size, and 256-Mbyte segment size
  - — A 32-entry, two-way set-associative ITLB
  - — A 32-entry, two-way set-associative DTLB
  - — Four-entry data and instruction BAT arrays providing 128-Kbyte to 256-Mbyte blocks
  - — Software table search operations and updates supported through fast trap mechanism
  - — 52-bit virtual address; 32-bit physical address

— Protection-only mode—optional configuration of the TLBs that offers protection for up to 4 Mbytes of memory per TLB, but no effective address translation

- Facilities for enhanced system performance
  - — A 64-bit (address and data multiplexed) external data bus with burst transfers
  - — Support for injected snoops by other devices during ownership of bus tenure
  - — Ability to broadcast a line-fill address, during the address tenure of a write-back transaction on the bus
- Integrated power management
  - — Low-power 3.3-volt design
  - — Internal processor/bus clock multiplier that provides 2/1 and 3/1 ratios
  - — Three static power-saving modes—doze, nap, and sleep
  - — Automatic dynamic power reduction on an internal subunit level of granularity, on a per clock basis, when the subunits are idle
- In-system testability and debugging features through JTAG boundary-scan capability

1.2 Block Diagram

The 602 block diagram in Figure 1 illustrates how the execution units—IU, FPU, BPU, and LSU—operate independently and in parallel.

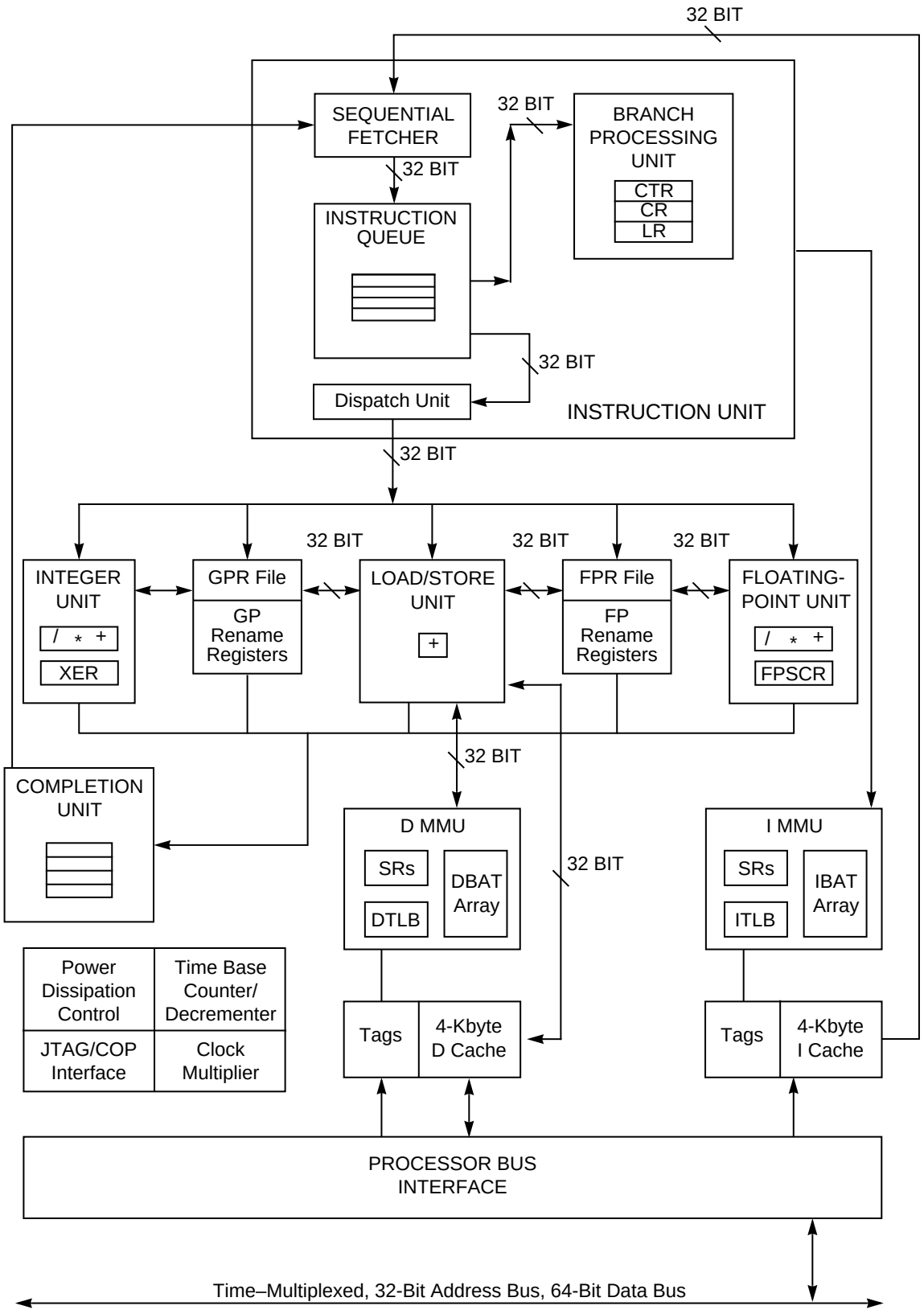


Figure 1. PowerPC 602 Microprocessor Block Diagram

1.3 Instruction Pipeline

As shown in Figure 2, the instruction pipeline in the 602 has four stages:

- Fetch—During this stage, instructions are fetched from the instruction cache.
- Decode and dispatch—During this stage, instructions are decoded, branch instructions are folded out, and instructions are dispatched for execution once all the resources needed for execution are available.
- Execute—During this stage, instructions are executed in the LSU, IU, or FPU.
- Complete and write-back—During this stage, all results are committed to the architectural registers.

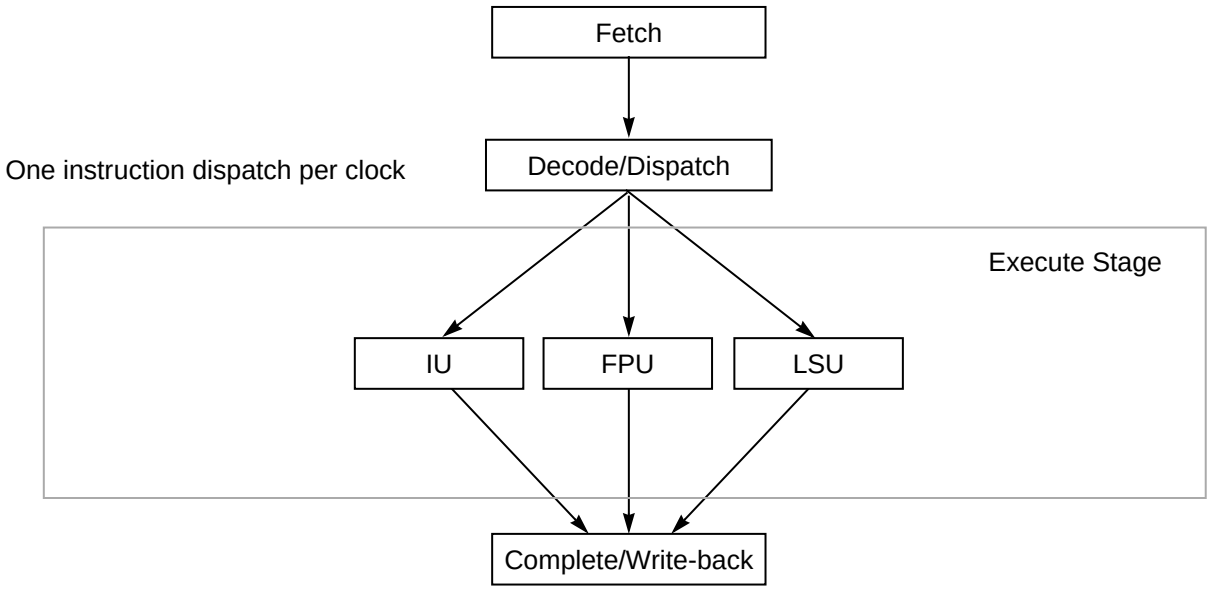


Figure 2. Pipeline Diagram

Note that an instruction may remain in a single pipeline stage for multiple processor cycles and may simultaneously occupy more than one processor cycle.

If exceptions are detected during the execute or the complete/write-back stages, all following instructions are cancelled, their execution results in rename buffers are discarded, and the correct instruction stream is fetched for subsequent execution. For an overview and definition of exceptions supported in the 602, see Section 2.5.1, “PowerPC Exception Model.”

The 602 provides address translation and protection facilities, including an ITLB, DTLB, and instruction and data BAT arrays. Instruction fetching and issuing are handled in the instruction unit. Translation of addresses for cache or external memory accesses are handled by the MMUs. For more information, see Sections 1.3.1, “Instruction Unit,” and 1.5.1, “Memory Management Units (MMUs).”

1.3.1 Instruction Unit

As shown in Figure 1, the 602 instruction unit, which contains a fetch unit, instruction queue, dispatch unit, and BPU, provides centralized control of instruction flow to the execution units. The instruction unit determines the address of the next instruction to be fetched based on information from the sequential fetcher and from the BPU.

The instruction unit fetches the instructions from the instruction cache into the instruction queue. The BPU extracts branch instructions from the instruction queue and uses the static branch prediction defined by the PowerPC architecture specification on unresolved conditional branches. This allows the instruction unit to fetch instructions from a predicted target instruction stream while a conditional branch is evaluated. The BPU folds out branch instructions for unconditional branches or for conditional branches unaffected by instructions in progress in the execution pipeline.

Instructions issued beyond a predicted branch do not complete execution until the branch is resolved, preserving the programming model of sequential execution. If any of these instructions are to be executed in the BPU, they are decoded but not issued. Instructions to be executed by the FPU, IU, and LSU are issued and allowed to complete up to the register write-back stage. Write-back is allowed when a correctly predicted branch is resolved, and instruction execution continues without interruption along the predicted path. An instruction is dispatched only if there is an entry available for it in the completion unit. If no completion buffers are available, instruction dispatch stalls until an entry is available.

If branch prediction is incorrect, the instruction unit flushes all predicted path instructions and instructions are issued from the correct path.

1.3.1.1 Instruction Queue and Dispatch Unit

The instruction queue (IQ), shown in Figure 1, holds as many as four instructions and loads one instruction from the instruction unit during a single cycle. The instruction fetch unit continuously loads as many instructions as space in the IQ allows. If one of the instructions loaded is a branch instruction, it is dispatched to the BPU. One non-branch instruction per cycle can be dispatched to any one of the three other execution units. Dispatching is facilitated to the IU, FPU, and LSU by the provision of a reservation station at each unit. The dispatch unit checks for source and destination register dependencies, determines dispatch serializations, and inhibits subsequent instruction dispatching as required.

For a more detailed overview of instruction dispatch, see Section 2.7, “Instruction Timing.”

1.3.1.2 Branch Processing Unit (BPU)

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

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 602 fetches instructions from the predicted target stream until the conditional branch is resolved.

The BPU contains an adder for computing branch target addresses and three user-control registers—the link register (LR), the count register (CTR), and the condition register (CR). The BPU calculates the return pointer for subroutine calls and saves it into the LR for certain types of branch instructions. The LR also contains the branch target address for the Branch Conditional to Link Register (**bclr***x*) instruction. The CTR contains the branch target address for the Branch Conditional to Count Register (**bcctr***x*) instruction. The contents of the LR and CTR can be copied to or from any GPR. 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.

1.3.1.3 Completion Unit

When an instruction is dispatched, it is reserved a place in the completion buffer, which tracks the program order and ensures that instructions complete in program order. Completing an instruction commits the 602 to any architectural register changes caused by that instruction. In-order completion ensures the correct architectural state when the 602 must recover from a mispredicted branch or an exception.

Instruction state and other information required for completion is kept in a first-in-first-out (FIFO) queue of four completion buffers. A single completion buffer is allocated for each instruction as it enters the dispatch unit. If no completion buffers are available, instruction dispatch stalls. A maximum of one instruction per cycle is completed in order from the queue.

This unit is responsible for ensuring that exceptions are handled in an orderly way.

1.4 Independent Execution Units

The PowerPC architecture's support for independent execution units allows the implementation of processors with out-of-order instruction execution. For example, because branch instructions do not depend on GPRs or FPRs, branches can often be resolved early, eliminating stalls caused by taken branches.

Branch instructions do not execute in the same sense that arithmetic, logical, or load/store instructions do. As shown in Figure 1, the IU, FPU, and LSU are arranged in parallel to one another. The execution units are described in the following sections.

1.4.1 Integer Unit (IU)

The IU executes all integer instructions. The IU executes one integer instruction at a time, performing computations with its arithmetic logic unit (ALU), multiplier, divider, and integer exception register (XER). Most integer instructions are single-cycle instructions. Thirty-two 32-bit GPRs are provided to support integer operations. Stalls due to contention for GPRs are minimized by the automatic allocation of rename registers. The 602 writes the contents of the rename registers to the appropriate GPR when integer instructions are retired by the completion unit.

The IU executes all integer arithmetic instructions, condition register logical instructions, synchronization, and move to/from instructions.

1.4.2 Floating-Point Unit (FPU)

The FPU contains a single-precision multiply-add array and the floating-point status and control register (FPSCR). The multiply-add array allows the 602 to efficiently implement multiply, add, and multiply-add operations. The FPU is pipelined so that single-precision instructions can be issued back-to-back. Thirty-two FPRs support single-precision floating-point operations. Stalls due to contention for FPRs are minimized by the automatic allocation of rename registers. The 602 writes the contents of the rename registers to the appropriate FPR when floating-point instructions are retired by the completion unit.

In the 602, all double-precision arithmetic operations, floating-point load or store operations that involve double-precision operands, and operations producing denormalized numbers are handled by emulation software. The 602 traps to an exception handler when it encounters these operands or operations while in the IEEE mode.

The 602 can also be operated in the non-IEEE floating-point mode. For a result of divide by zero, invalid, overflow, or underflow, this mode allows the 602 to produce predetermined values that may not conform to IEEE-754. The non-IEEE mode is useful for time-critical applications such as graphics applications. For inputs of QNANs or SNANs, the hardware delivers QNANs.

1.4.3 Load/Store Unit (LSU)

The LSU executes all load and store instructions and provides the data transfer interface between the GPRs, FPRs, and the cache/memory subsystem. The LSU calculates effective addresses, performs data alignment, and traps on load/store string instructions. Load/store string instructions are emulated in software.

Load and store instructions are issued and translated in program order; however, the actual memory accesses can occur out of order. Synchronizing instructions are provided to enforce strict ordering.

Cacheable loads, when free of data dependencies, execute speculatively with a maximum throughput of one per cycle and a two-cycle total latency. Data returned from the cache is held in a rename register until the completion logic commits the value to a GPR or FPR. Store instructions cannot be executed speculatively and are held in the store queue until the completion logic signals that the store operation is to be completed to memory. The time required to perform the actual load or store operation depends on whether the operation involves the cache or the system memory.

The LSU executes all load, store, cache control, and memory control instructions.

1.5 Memory Subsystem

The 602 provides support for cache and memory management through separate instruction and data memory management units. The 602 also provides separate 4-Kbyte instruction and data caches and an efficient processor bus interface to facilitate access to main memory and other bus subsystems. The memory subsystem support functions are described in the following sections.

1.5.1 Memory Management Units (MMUs)

The 602's MMUs support up to 4 Petabytes (2^{52}) of virtual memory and 4 Gigabytes (2^{32}) of physical memory (referred to as real memory in the architecture specification) for instruction and data. The MMUs also control access privileges for these spaces on block and page granularities. Referenced and changed status is maintained by the processor for each page to assist implementation of a demand-paged virtual memory system.

The LSU calculates effective addresses for data load and store operations, and performs data alignment to and from cache memory. The instruction unit calculates the effective addresses for instruction fetching.

After an address is generated, the higher-order bits of the effective address are translated by the appropriate MMU into physical address bits. Simultaneously, the lower-order address bits (which are untranslated and the same for both logical and physical addresses), are directed to the on-chip caches where they form the index into the two-way set-associative tag array. After translating the address, the MMU passes the higher-order bits of the physical address to the cache, and the cache lookup completes. For cache-inhibited accesses or accesses that miss in the cache, the untranslated lower-order address bits are concatenated with the translated higher-order address bits; the resulting 32-bit physical address is then used by the memory unit and the system interface, which accesses external memory.

The MMU also translates addresses and enforces memory protection supervisor/user privilege level of the access in relation to whether the access is a load or store.

For instruction accesses, the MMU performs an address lookup in the 32 entries of the ITLB, and in the IBAT array. If an effective address hits in both the ITLB and the IBAT array, the IBAT array translation takes priority. Data accesses cause a lookup in the DTLB and DBAT array for the physical address translation. In most cases, the physical address translation resides in one of the TLBs or BATs, and the physical address bits are readily available to the on-chip cache.

When the physical address translation misses in the TLBs, the 602 provides hardware assistance for software to perform a search of the translation tables in memory. The hardware assist consists of the following features:

- Automatic storage of the missed effective address in the 602-specific IMISS and DMISS registers
- Automatic generation of the primary and secondary hashed real address of the page table entry group (PTEG), which are readable from the HASH1 and HASH2 register locations. The HASH data is generated from the contents of the IMISS or DMISS register. The register selected depends on whether an instruction or data miss was acknowledged last.
- Automatic generation of the first word of the page table entry (PTE) for which the tables are being searched
- A real page address (RPA) register that matches the format of the lower word of the PTE
- Two 602-specific TLB access instructions (**tlbli** and **tlbld**) that are used to load an address translation into the instruction or data TLBs
- Shadow registers for GPR0–GPR3 that allow missed code to execute without corrupting the state of any of the GPRs. These shadow registers are used only for servicing a TLB miss.

The 602-specific protection-only mode enables each TLB to protect up to 128 Kbytes per entry (4 Mbytes per TLB). Effective address translation is not performed for TLBs in protection-only mode; however, BATs are not affected by running the processor in protection-only mode and can still implement both protection and translation of the effective addresses as described earlier. The default mechanism is described in *The Programming Environments Manual*.

See Section 2.6.2, “PowerPC 602 Microprocessor Memory Management,” for more information about memory management for the 602.

1.5.2 Cache Units

The 602 provides independent 4-Kbyte, two-way set-associative instruction and data caches. The cache block size is 32 bytes (eight words). The caches adhere to a write-back policy, but, as defined by the PowerPC architecture, the 602 allows control of cacheability, write policy, and memory coherency at the page and memory block levels. The caches use an LRU replacement policy.

As shown in Figure 1, the caches provide a 32-bit interface to the instruction fetch unit and load/store unit. The surrounding logic selects, organizes, and forwards the requested information to the requesting unit. Store operations to the cache can be performed on a byte basis, and a complete read-modify-write operation to the cache can occur in each cycle.

The load/store unit and instruction fetch unit provide the caches with the address of the data or instruction to be fetched. In the case of a cache hit, the cache returns two words to the requesting unit.

Since the 602 data cache tags are single-ported, simultaneous load or store and snoop accesses cause resource contention. Snoop accesses have the highest priority and are given first access to the tags, unless the snoop access coincides with a tag write. In this case the snoop is retried and must rearbitrate for access to the cache. Load or store operations that are deferred due to snoop accesses are executed on the clock cycle following the snoop.

1.6 Processor Bus Interface

The 602 bus interface is a time-multiplexed, 32-bit address, 64-bit data interface. In its first phase, the multiplexed interface is used by the address tenure. Following the completion of the address tenure, the interface is then used for the data tenure.

The 602 on-chip caches can be configured in the write-through or write-back modes. In the write-back mode, the predominant type of transaction for most applications is burst-read memory operations, followed by burst-write memory operations and single-beat (noncacheable or write-through) operations. Additionally, there can be address-only operations, variants of the burst and single-beat operations, (for example, global memory operations that are snooped and atomic memory operations), and address retry activity (for example, when a snooped read access hits a modified block in the cache).

Memory accesses occur in single-beat (1–8 bytes) and four-beat burst (32 bytes) data transfers.

Access to the system interface is granted through an external arbitration mechanism that allows devices to compete for bus mastership. This arbitration mechanism allows the 602 to be integrated into systems that implement various fairness and bus parking procedures to avoid arbitration overhead.

Typically, memory accesses in the 602 are weakly-ordered—sequences of operations, including load/store multiple instructions, do not necessarily complete in the order they begin—maximizing the efficiency of the bus without sacrificing coherency of the data. The 602 allows read operations to precede store operations (except when a dependency exists). Because the processor can dynamically optimize run-time ordering of load/store traffic, overall performance is improved.

1.7 System Support Functions

The 602 implements several support functions that include power management, time base/decrementer registers for system timing tasks, a watchdog timer, an IEEE 1149.1(JTAG)/common on-chip processor (COP) test interface, and a phase-locked loop (PLL) clock multiplier. These system support functions are described in the following subsections.

1.7.1 Power Management

The 602 provides four power modes selectable by setting the appropriate control bits in the machine state register (MSR) and hardware implementation register 0 (HID0) registers. The four power modes are as follows:

- Full-power–This is the default power state of the 602. The 602 is fully powered and the internal functional units are operating at the full processor clock speed. If the dynamic power management mode is enabled, functional units that are idle automatically enters a low-power state without affecting performance, software execution, or external hardware.
- Doze–All the functional units of the 602 are disabled except for the time base/decrementer registers and the bus snooping logic. The 602 returns to the full-power state upon the occurrence of any asynchronous exception. Asynchronous exceptions implemented on the 602 are listed in Table 1. The 602 in doze mode maintains the PLL in a fully-powered state and locked to the system external clock input (SYSCLK) so a transition to the full-power state takes only a few processor clock cycles.
- Nap–The nap mode further reduces power consumption by disabling bus snooping, leaving only the time base register and the PLL in a powered state. The 602 returns to the full-power state upon the occurrence of any asynchronous exception. Asynchronous exceptions implemented on the 602 are listed in Table 1. A return to full-power state from a nap state takes only a few processor clock cycles.
- Sleep–Sleep mode reduces power consumption to a minimum by disabling all internal functional units, after which external system logic may disable the PLL and SYSCLK. Returning the 602 to the full-power state requires the enabling of the PLL and SYSCLK, followed by any external exception after the time required to relock the PLL.

1.7.2 Time Base/Decrementer

The time base is a 64-bit register (accessed as two 32-bit registers) that is incremented once every four bus clock cycles; external control of the time base is provided through the time base enable (TBEN) signal. The decrementer is a 32-bit register that generates a decrementer exception after a programmable delay. The contents of the decrementer register are decremented once every four bus clock cycles, and the decrementer exception condition is generated as the count passes through zero.

1.7.3 IEEE 1149.1 (JTAG)/Common On-Chip Processor (COP) Test Interface

The 602 provides IEEE 1149.1 and COP functions for facilitating board testing and chip debug. The IEEE 1149.1 test interface provides a means for boundary-scan testing the 602 and the board to which it is attached. The COP function shares the IEEE 1149.1 test port, provides a means for executing test routines, and facilitates chip and software debugging.

1.7.4 Clock Multiplier

The internal clocking of the 602 is generated from and synchronized to the external clock signal, SYSCLK, by means of a voltage-controlled, oscillator-based PLL. The PLL provides programmable internal processor clock rates of either two or three times the externally supplied clock frequency. The bus clock is the same frequency and is synchronous with SYSCLK.

1.7.5 Watchdog Timer

The 602-specific watchdog timer can be used to generate a periodic exception based on the operation of the time-base register. The watchdog timer is enabled and programmed through the timer control register (TCR), which is a supervisor-level SPR specific to the 602. Supervisor-level software can set bits in the TCR to select one of four time periods for the interrupts, and other aspects of the watchdog timer operations. When a watchdog timer exception occurs, instruction fetching begins at vector offset 0x1500 (as shown in Table 2). The watchdog timer can be programmed such that if the exception handler does not reset the timer, a second watchdog-timer interrupt condition will cause a soft reset (system reset exception).

Part 2 PowerPC 602 Microprocessor: Implementation

The PowerPC architecture is derived from the IBM 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 602 as a low-power, 32-bit member of the PowerPC processor family.

- Features—Section 2.1, “Features,” describes general features that the 602 shares with the PowerPC microprocessor family.
- Registers and programming model—Section 2.2, “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 additional registers that are unique to the 602.
- Instruction set and addressing modes—Section 2.3, “Instruction Set and Addressing Modes,” describes the PowerPC instruction set and addressing modes for the PowerPC operating environment architecture, and defines and describes the PowerPC instructions implemented in the 602.

- Cache implementation—Section 2.4, “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 602 cache implementation.
- Exception model—Section 2.5, “Exception Model,” describes the exception model of the PowerPC operating environment architecture and the differences in the 602 exception model.
- Memory management—Section 2.6, “Memory Management,” describes generally the conventions for memory management among the PowerPC processors. This section also describes the 602’s implementation of the 32-bit PowerPC memory management specification.
- Instruction timing—Section 2.7, “Instruction Timing,” provides a general description of the instruction timing provided by the parallel execution supported by the PowerPC architecture and the 602.
- System interface—Section 2.8, “System Interface,” describes the signals implemented on the 602.

2.1 Features

The 602 is a high-performance, low-cost microprocessor for consumer electronics and computers. It is designed for use in advanced home entertainment and educational devices with audio/video, multimedia, and complex graphics requirements. The 602 is also applicable for low-power business and commercial devices with speech recognition and synthesis, wireless communications, or handwriting recognition.

The following sections summarizes the features of the 602, including both those that are defined by the architecture and those that are unique to the 602 implementation.

The PowerPC architecture consists of the following layers, and adherence to the PowerPC architecture can be measured 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 602 does not implement the double-precision floating-point instructions and the load/store string instructions in hardware. Barring these exceptions, the 602 implements the levels of architecture as mentioned above. Specific features of the 602 are listed in Section 1.1, “PowerPC 602 Microprocessor Features.”

For more information, see *The Programming Environments Manual*.

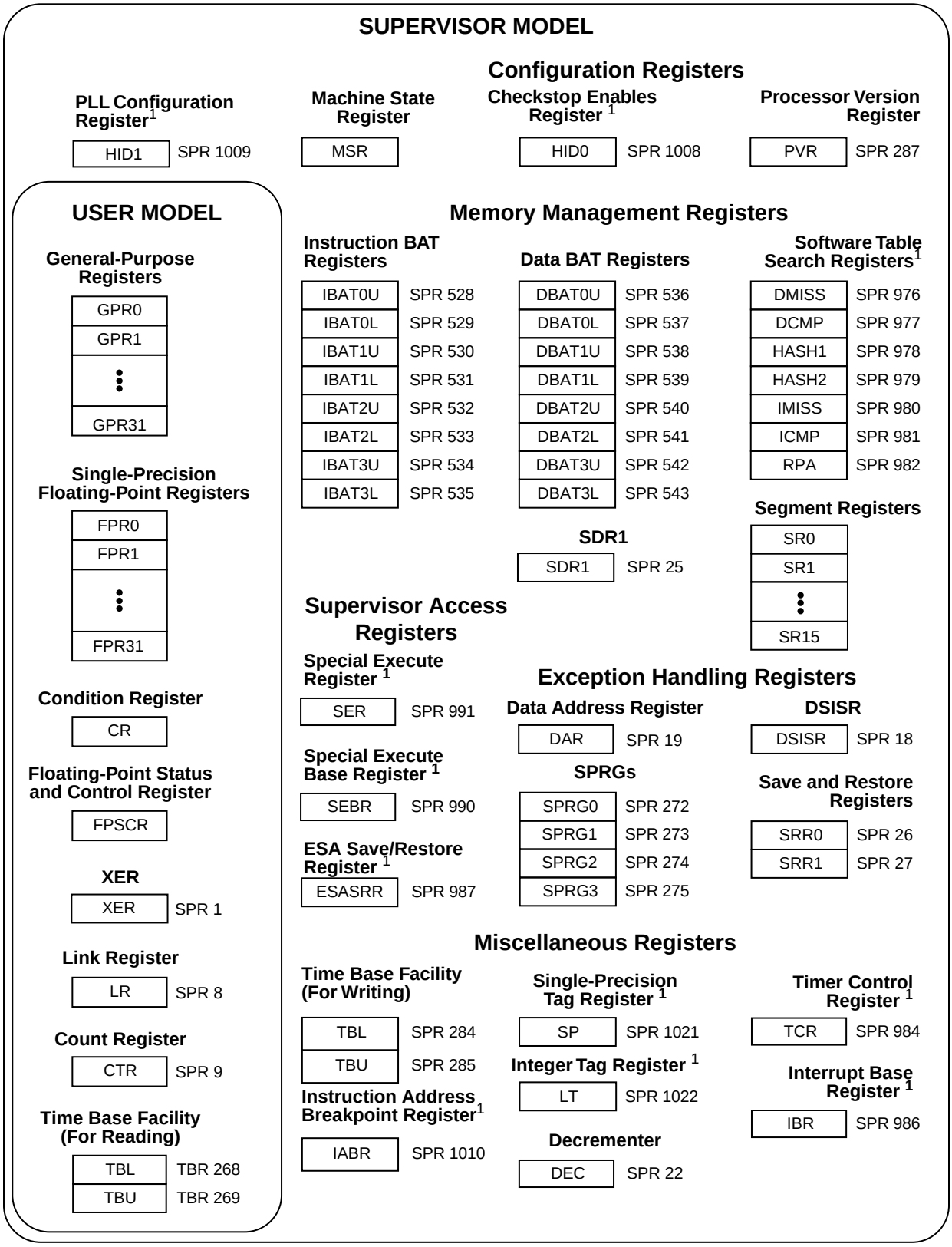
2.2 PowerPC Registers and Programming Model

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 level (typically used by the operating system) and user level (used by the application software). The programming models incorporate 32 GPRs, 32 FPRs, special-purpose registers (SPRs), and several miscellaneous registers. Each PowerPC microprocessor may also have its own unique set of implementation-specific registers. The registers implemented in the 602 are shown in Figure 3 and described in the following sections. Registers defined by the PowerPC architecture are described in *The Programming Environments Manual*.

Access to supervisor-level 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 3 shows the registers implemented in the 602 and indicates whether these registers are accessible to user- or supervisor-level software. The numbers to the right of the SPRs indicate the number that is used in the syntax of the instruction operands to access the register.



[1] These registers are 602–specific registers. They may not be supported by other PowerPC processors.

Figure 3. PowerPC 602 Microprocessor Programming Model—Registers

The following sections summarize the PowerPC registers that are implemented in the 602.

2.2.1 General-Purpose Registers (GPRs)

The PowerPC architecture defines 32 user-level, general-purpose registers (GPRs) that serve as the data source or destination for all integer instructions.

2.2.2 Floating-Point Registers (FPRs)

The UISA portion PowerPC architecture defines 32 user-level, 64-bit floating-point registers (FPRs). The FPRs serve as the data source or destination for floating-point instructions. These registers can contain data objects of either single- or double-precision floating-point formats.However, because the 602 is optimized for systems that perform single- and not double-precision floating-point arithmetic, the 602 implements thirty-two 32-bit FPRs. Double-precision operations are trapped for emulation in software.

2.2.3 Condition Register (CR)

The CR is a 32-bit, user-level register that consists of eight four-bit fields that reflect the results of certain operations, such as move, integer and floating-point compare, arithmetic, and logical instructions, and provide a mechanism for testing and branching.

2.2.4 Floating-Point Status and Control Register (FPSCR)

The floating-point status and control register (FPSCR) is a user-level register that contains all exception signal bits, exception summary bits, exception enable bits, and rounding control bits needed for compliance with the IEEE 754 standard.

2.2.5 Machine State Register (MSR)

The machine state register (MSR) is a supervisor-level register that defines the state of the processor. The contents of this register are saved when an exception is taken and typically restored when the exception handling completes. The 602 implements the MSR as a 32-bit register.

2.2.6 Segment Registers (SRs)

For memory management, 32-bit PowerPC microprocessors implement sixteen 32-bit segment registers (SRs). To speed access, the 602 implements the segment registers as two arrays—a main array (for data memory accesses) and a shadow array (for instruction memory accesses). Loading a segment entry with the Move to Segment Register (**mtsr**) instruction loads both arrays.

2.2.7 Special-Purpose Registers (SPRs)

The PowerPC operating environment architecture 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 3, depending on the program's access privilege (supervisor or user, determined by the privilege-level (PR) bit in the MSR). Note that registers such as the GPRs and FPRs are accessed through operands that are part of the instructions. Access to registers can be explicit (that is, through the use of specific instructions for that purpose such as Move to Special-Purpose Register (**mtspr**) and Move from Special-Purpose Register (**mfspr**) instructions) or implicit, as the part of the execution of an instruction. Some registers are accessed both explicitly and implicitly.

2.2.7.1 User-Level SPRs

The following SPRs are accessible by user-level software:

- Link register (LR)—The 32-bit link register can be used to provide the branch target address and to hold the return address after branch and link instructions.
- Count register (CTR)—The 32-bit CTR is decremented and tested automatically as a result of branch-and-count instructions.
- Integer exception register (XER)—The 32-bit XER contains the summary overflow bit, integer carry bit, and overflow bit.
- Hardware implementation-dependent register 1 (HID1) is a user-level, read-only register that stores the PLL configuration bits. This register is 602-specific, and is not defined by the PowerPC architecture.

2.2.7.2 Supervisor-Level SPRs

The 602 also contains SPRs that can be accessed only by supervisor-level software. See Figure 3 for a list of the SPR numbers. These registers consist of the following:

- The 32-bit DSISR register defines the cause of data access and alignment exceptions.
- The data address register (DAR) is a 32-bit register that holds the address of an access after an alignment or DSI exception.
- Decrementer register (DEC) is a 32-bit decrementing counter that provides a mechanism for causing a decrementer exception after a programmable delay.
- The 32-bit SDR1 register specifies the page table format used in virtual-to-physical address translation for pages. (Note that physical address is referred to as real address in the architecture specification.)
- The machine status save/restore register 0 (SRR0) is a 32-bit register that is used by the 602 for saving the address of the instruction that caused the exception, and the address to return to when a Return from Interrupt (**rfi**) instruction is executed.
- The machine status save/restore register 1 (SRR1) is a 32-bit register used to save machine status on exceptions and to restore machine status when an **rfi** instruction is executed.
- The 32-bit SPRG0–SPRG3 registers are provided for operating system use.
- The time base registers (TBL and TBU) together provide a 64-bit time base register. The registers are implemented as a 64-bit counter, with the least-significant bit being the most frequently incremented.
- The processor version register (PVR) is a 32-bit, read-only register that identifies the version (model) and revision level of the PowerPC processor.
- Block address translation (BAT) registers—The PowerPC architecture defines 16 BAT registers, divided into four pairs of data BATs (DBATs) and four pairs of instruction BATs (IBATs).

The following supervisor-level SPRs are implementation-specific to the 602:

- The DMISS and IMISS registers are read-only registers that are loaded automatically upon an instruction or data TLB miss.
- The HASH1 and HASH2 registers contain the physical addresses of the primary and secondary page table entry groups (PTEGs).
- The ICMP and DCMP registers contain a duplicate of the first word in the page table entry (PTE) for which the table search is looking.

- The required physical address (RPA) register is loaded by the processor with the second word of the correct PTE during a page table search.
- The hardware implementation-dependent register 0 (HID0) provides means for enabling the 602's checkstops and features.
- The instruction address breakpoint register (IABR) is loaded with an instruction address that is compared to instruction addresses in the dispatch queue. When an address match occurs, an instruction address breakpoint exception is generated.
- The special execute register (SER) is a 32-bit register used in the protection-only mode. Each bit is used to assert special execute privileges on a 4-Kbyte page basis.
- The special execute base register (SEBR) contains the base address of the 128-Kbyte region that is protected by the special execute (SE) bits in SER.
- The enable supervisor access save and restore register (ESASRR) is a 32-bit register that saves selected MSR bits (SA, EE, PR, AP) when the **esa** instruction is executed. Executing the **dsa** instruction restores these bits to the MSR.
- The single-precision tag register (SP) is a 32-bit register for which each bit (SP0–SP31) corresponds to one of the 32, 32-bit FPRs (FPR0–FPR31). An SP bit is set if the corresponding FPR holds a single-precision value.
- The integer tag register (LT) is a 32-bit register for which each bit (LT0–LT31) corresponds to one of the 32, 32-bit FPRs (FPR0–FPR31). An LT bit is set if the corresponding FPR holds an integer value.
- The timer control register (TCR) provides bits for enabling and programming the watchdog timer.
- The interrupt base register (IBR) contains an exception vector offset address used by exception handlers if the MSR[IP] is cleared when an exception occurs; if MSR[IP] is set, the default offset address 0xFFF00000 is used.

2.3 Instruction Set and Addressing Modes

The following subsections describe the PowerPC instruction set and addressing modes in general.

2.3.1 PowerPC Instruction Set and Addressing Modes

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.

2.3.1.1 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 instructions
  - — 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 instructions
  - — Synchronize instruction
  - — Instruction synchronize
- Memory control instructions—These instructions provide control of caches, TLBs, and segment registers.
  - — Supervisor-level cache management instructions
  - — User-level cache instructions
  - — Segment register manipulation instructions
  - — Translation lookaside buffer management instructions

Note that this grouping of the instructions 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) operands. Floating-point double-precision operations are trapped for emulation in software. The PowerPC architecture uses instructions that are four bytes long and word-aligned. It provides for byte, half-word, and word operand load and store operations between memory and 32 GPRs. It also provides for word and double-word operand loads and stores between memory and 32 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 causing an exception, or by an asynchronous event. Either kind of exception may cause one of several components of the system software to be invoked.

2.3.1.2 Calculating Effective Addresses

The effective address (EA) is the 32-bit address computed by the processor when executing a memory access or branch instruction or when fetching the next sequential instruction.

The PowerPC architecture supports two simple memory addressing modes:

- EA = (**r**A|0) + offset (including offset = 0) (register indirect with immediate index)
- EA = (**r**A|0) + **r**B (register indirect with index)

These simple addressing modes allow efficient address generation for memory accesses. Calculation of the effective address for aligned transfers occurs in a single clock cycle.

For a memory access instruction, if the sum of the effective address and the operand length exceeds the maximum effective address, the memory operand is considered to wrap around from the maximum effective address to effective address 0.

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.

2.3.2 PowerPC 602 Microprocessor Instruction Set

The 602 instruction set is defined as follows:

- The 602 provides hardware support for most 32-bit PowerPC instructions. The 602 does not support double-precision floating-point, load/store string, **eciwx**, and **ecowx** instructions in hardware.
- The 602 provides two implementation-specific instructions used for software table search operations following TLB misses and two that enable/disable access into the supervisor mode (without having to branch or invoking a system call):
  - TLB Load Data (**tlbld**)
  - TLB Load Instruction (**tlbli**)
  - Enable Supervisor Access (**esa**)
  - Disable Supervisor Access (**dsa**)
- The 602 implements the following instructions that are defined as optional by the PowerPC architecture:
  - Floating Select (**fsel**)
  - Floating Reciprocal Estimate Single-Precision (**fres**) instruction. On the 602, **fsel** is implemented as a divide rather than an estimate.
  - Floating-Point Convert to Integer Word with Round toward Zero (**fctiwz**)

2.4 Cache Implementation

The following subsections describe the PowerPC architecture's treatment of cache in general, and the 602-specific implementation, respectively.

2.4.1 PowerPC Cache Characteristics

The PowerPC architecture does not define hardware aspects of cache implementations. For example, some PowerPC processors, including the 602, have separate instruction and data caches (Harvard architecture), while others, such as the PowerPC 601™ microprocessor, implement a unified cache.

PowerPC microprocessors control the following memory access modes on a page or block basis:

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

Note that in the 602, a cache block is defined as being eight words wide. The VEA defines cache management instructions that application programmers can use to affect the cache contents and coherency state.

2.4.2 PowerPC 602 Microprocessor Cache Implementation

The 602 has two 4-Kbyte, two-way set-associative (instruction and data) caches. 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. A block diagram of the data cache organization is shown in Figure 3.

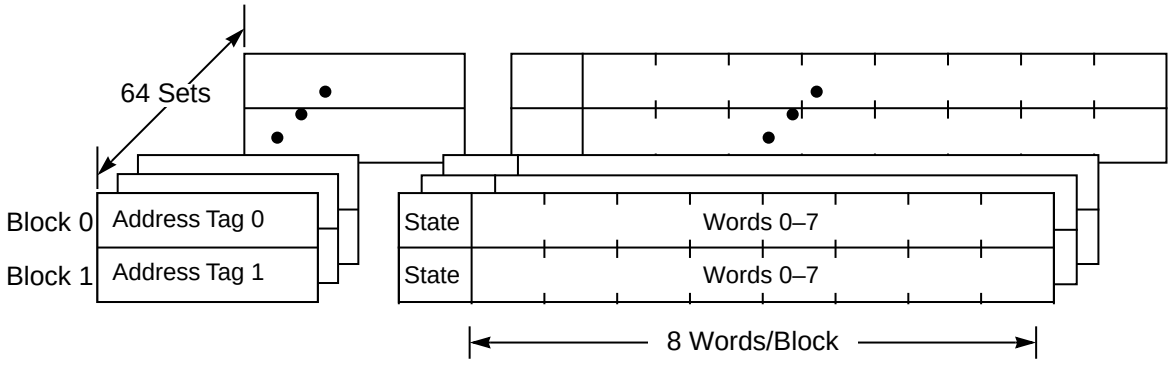


Figure 4. Data Cache Organization

The data cache is configured as 64 sets of two cache blocks each. Each cache block consists of eight 32-bit words, two state bits, and an address tag. The two state bits implement the three-state MEI (modified/exclusive/invalid) protocol. Note that the PowerPC architecture defines the term block as the cacheable unit. For the 602, the block size is equivalent to a cache block.

The instruction cache also consists of 64 sets of two blocks, and each block consists of 32 bytes, an address tag, and a valid bit. The instruction cache may not be written to except through a line-fill operation. The instruction cache is not snooped, and cache coherency must be maintained by software. A fast hardware invalidation capability is provided to support cache maintenance. The organization of the instruction cache is very similar to the data cache shown in Figure 3.

Each cache block contains eight contiguous words from memory that are loaded from an 8-word boundary (that is, bits A27–A31 of the effective addresses are zero); thus, a cache block never crosses a page boundary. Accesses that cross a page boundary can incur a performance penalty.

The 602's cache blocks are loaded in four beats of 64 bits each. The burst load is performed as critical double-word first. The cache that is being loaded is blocked to internal accesses until the load completes. The critical double word is simultaneously written to the cache and forwarded to the requesting unit, thus minimizing stalls due to load delays.

To ensure coherency among caches in a multiprocessor (or multiple caching-device) implementation, the 602 implements the MEI protocol. These three states, modified, exclusive, and invalid, indicate the state of the cache block as follows:

- Modified—The cache block is modified with respect to system memory; that is, data for this address is valid only in the cache and not in system memory.
- Exclusive—This cache block holds valid data that is identical to the data at this address in system memory. No other device has this data.
- Invalid—This cache block does not hold valid data.

Cache coherency is enforced by on-chip bus snooping logic. Since the 602's data cache tags are single ported, a simultaneous load or store and snoop access represent a resource contention. The snoop access is given first access to the tags. The load or store then occurs on the clock following the snoop. All read operations on the bus are treated as read-with-intent-to-modify operations, as are all burst-read operations from a snooping perspective.

2.5 Exception Model

The following subsections describe the PowerPC exception model and the 602 implementation, respectively.

2.5.1 PowerPC Exception Model

The PowerPC exception mechanism allows the processor to change to supervisor state as a result of external signals, errors, or unusual conditions arising in the execution of instructions, and differ from the arithmetic exceptions defined by the IEEE for floating-point operations. 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 occurs in supervisor mode.

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

The PowerPC architecture requires that a processor be able to handle instruction-caused exceptions in program order; therefore, although a particular implementation may recognize exception conditions out of order, they are presented strictly in order. When an instruction-caused exception is recognized, any unexecuted instructions that appear in the instruction stream prior to the instruction causing the interrupt, are required to complete before the exception is taken. Any exceptions caused by those instructions are handled first. Likewise, exceptions that are asynchronous and precise are recognized when they occur, but are not handled until the instruction currently in the completion stage successfully completes execution or generates an exception, and the completed store queue is emptied.

Unless a catastrophic condition causes a system reset or machine check exception, only one exception is handled at a time. If, for example, an instruction encounters multiple exception conditions, those conditions are processed sequentially. After the exception handler handles an exception, the instruction execution continues until the next exception condition is encountered. However, in many cases there is no attempt to re-execute the instruction. This method of recognizing and handling exception conditions sequentially guarantees that exceptions are recoverable.

Exception handlers should save the information stored in SRR0 and SRR1 early to prevent the program state from being lost due to a system reset and machine check exception or to an instruction-caused exception in the exception handler, and before enabling external interrupts.

The PowerPC architecture supports four types of exceptions:

- Synchronous, precise—These are caused by instructions. All instruction-caused exceptions are handled precisely; that is, the machine state at the time the exception occurs is known and can be completely restored. This means that (excluding the trap and system call exceptions) the address of the faulting instruction is provided to the exception handler and that neither the faulting instruction nor subsequent instructions in the code stream completes execution before the exception is taken. Once the exception is processed, execution resumes at the address of the faulting instruction (or at an alternate address provided by the exception handler). When an exception is taken due to a trap or system call instruction, execution resumes at an address provided by the handler.
- Synchronous, imprecise—The PowerPC architecture defines two imprecise floating-point exception modes, recoverable and nonrecoverable. Even though the 602 provides a means to enable the imprecise modes, it implements these modes identically to the precise mode (that is, all enabled floating-point exceptions are always precise on the 602).
- Asynchronous, maskable—The external, system management, and decrementer interrupts are maskable asynchronous exceptions. When these exceptions occur, their handling is postponed until the next instruction, and any exceptions associated with that instruction, completes execution. If there are no instructions in the execution units, the exception is taken immediately upon determination of the correct restart address (for loading SRR0).
- Asynchronous, nonmaskable—There are two nonmaskable asynchronous exceptions: system reset and the machine check exception. These exceptions may not be recoverable, or may provide a limited degree of recoverability. All exceptions report recoverability through the MSR[RI] bit.

2.5.2 PowerPC 602 Microprocessor Exception Model

As specified by the PowerPC architecture, all 602 exceptions can be described as either precise or imprecise and either synchronous or asynchronous. Asynchronous exceptions (some of which are maskable) are caused by events external to the processor's execution; synchronous exceptions, which are all handled precisely by the 602, are caused by instructions. The 602 exception classes are shown in Table 1.

Table 1. PowerPC 602 Microprocessor Exception Classifications

Synchronous/Asynchronous	Precise/Imprecise	Exception Type
Asynchronous, nonmaskable	Imprecise	Machine check System reset
Asynchronous, maskable	Precise	External interrupt Decrementer System management interrupt Watchdog timer
Synchronous	Precise	Instruction-caused exceptions

Although exceptions have other characteristics as well, such as whether they are maskable or nonmaskable, the distinctions shown in Table 1 define categories of exceptions that the 602 handles uniquely. Note that Table 1 includes no synchronous imprecise instructions. Although the PowerPC architecture supports imprecise handling of floating-point exceptions, the 602 implements these exception modes as precise exceptions.

The 602's exceptions, and conditions that cause them, are listed in Table 2. The vector offset column indicates the offset from either 0xFFF00000 or 0x00000000 depending on the setting of MSR[IP], as per the architecture. The 602 provides a privileged instruction and a register (IBR) that allows the vector offset to be an address other than the defaults. If the MSR[IP] bit is set and the **mtspr** instruction has moved a valid address into the IBR, the IBR contents are used. Exceptions that are specific to the 602 are indicated.

Table 2. Exceptions and Conditions

Exception Type	Vector Offset (hex)	Causing Conditions
Reserved	00000	—
System reset	00100	A system reset is caused by the assertion of either $\overline{\text{SRESET}}$ or $\overline{\text{HRESET}}$.
Machine check	00200	A machine check is caused by the assertion of the $\overline{\text{TEA}}$ signal during a data bus transaction, assertion of $\overline{\text{MCP}}$.
DSI	00300	The cause of a DSI exception can be determined by the bit settings in the DSISR, listed as follows: 4 Set if a memory access is not permitted by the page or DBAT protection mechanism; otherwise cleared. 5 Set if memory access is attempted to a direct-store segment; otherwise cleared. 6 Set for a store operation and cleared for a load operation.
ISI	00400	An ISI exception is caused when an instruction fetch cannot be performed for any of the following reasons: • The effective (logical) address cannot be translated. That is, there is a page fault for this portion of the translation, so an ISI exception must be taken to load the PTE (and possibly the page) into memory. • The fetch access violates memory protection. If the key bits (Ks and Kp) in the segment register and the PP bits in the PTE are set to prohibit read access, instructions cannot be fetched from this location.
External interrupt	00500	An external interrupt is caused when MSR[EE] = 1 and the $\overline{\text{INT}}$ signal is asserted.
Alignment	00600	An alignment exception is caused when the 602 cannot perform a memory access for any of the following reasons: • The operand of a floating-point load or store is not word-aligned. • The operand of a floating-point memory access is to a direct store segment. • The operand of **lmw**, **stmw**, **lwarx**, or **stwcx.** is not word-aligned. • The operand of **dcbz** is in a page that is write through or cache inhibited, for a virtual mode access. • A little-endian access is misaligned, or a multiple access is attempted with the little-endian bit set.

Table 2. Exceptions and Conditions (Continued)

Exception Type	Vector Offset (hex)	Causing Conditions
Program	00700	A program exception is caused by one of the following exception conditions, which correspond to bit settings in SRR1 and arise during execution of an instruction: • Floating-point enabled exception—A floating-point enabled exception condition is generated when the following condition is met: (MSR[FE0] | MSR[FE1]) & FPSCR[FEX] is 1. FPSCR[FEX] is set by the execution of a floating-point instruction that causes an enabled exception or by the execution of one of the “move to FPSCR” instructions that results in both an exception condition bit and its corresponding enable bit being set in the FPSCR. • Illegal instruction—An illegal instruction program exception is generated when execution of an instruction is attempted with an illegal opcode or illegal combination of opcode and extended opcode fields (including PowerPC instructions not implemented in the 602). These do not include those optional instructions treated as no-ops. • Privileged instruction—A privileged instruction type program exception is generated when the execution of a privileged instruction is attempted and the MSR register user privilege bit, MSR[PR], is set. In the 602, this exception is generated for **mtspr** or **mfspr** with an invalid SPR field if SPR[0] = 1 and MSR[PR] = 1. This may not be true for all PowerPC processors. • Trap—A trap type program exception is generated when any of the conditions specified in a trap instruction is met.
Floating-point unavailable	00800	A floating-point unavailable exception is caused by an attempt to execute a floating-point instruction (including floating-point load, store, and move instructions) when the floating-point available bit is disabled, (MSR[FP] = 0).
Decrementer	00900	The decrementer exception occurs when the most significant bit of the decrementer (DEC) register transitions from 0 to 1. Must also be enabled with the MSR[EE] bit.
Reserved	00A00–00BFF	—
System call	00C00	A system call exception occurs when a System Call (**sc**) instruction is executed.
Trace	00D00	A trace exception is taken when MSR[SE] =1 or when the currently completing instruction is a branch and MSR[BE] =1.
Floating-point assist	00E00	Not implemented in the 602. Double-precision floating-point instructions no supported in hardware trap to the emulation trap exception (0x01600).
Reserved	00E10–00FFF	—
Instruction translation miss	01000	An instruction translation miss exception is caused when an effective address for an instruction fetch cannot be translated by the ITLB.
Data load translation miss	01100	A data load translation miss exception is caused when an effective address for a data load operation cannot be translated by the DTLB.
Data store translation miss	01200	A data store translation miss exception is caused when an effective address for a data store operation cannot be translated by the DTLB; or when a DTLB hit occurs and the change bit in the PTE must be set due to a data store operation.

Table 2. Exceptions and Conditions (Continued)

Exception Type	Vector Offset (hex)	Causing Conditions
Instruction address breakpoint	01300	An instruction address breakpoint exception occurs when the address (bits 0–29) in the IABR matches the next instruction to complete in the completion unit and the IABR enable bit (bit 30) is set.
System management interrupt	01400	A system management interrupt is caused when MSR[EE] =1 and the $\overline{\text{SMI}}$ input signal is asserted.
Watchdog timer	01500	The watchdog timer interrupt exception is taken when a carry occurs out of a bit specified by the user. If the watchdog timer is not reset by the interrupt service routine, a second watchdog timer exception forces an internal reset of the 602.
Emulation trap	01600	The emulation trap exception is taken when any of the following instructions are encountered: • Double-precision floating-point instructions • Load/store string instructions • The **eciwx** and **ecowx** instructions (optional Instructions in the PowerPC architecture)
Reserved	01700–02FFF	—

2.6 Memory Management

The following subsections describe the memory management features of the PowerPC architecture, and the 602 implementation, respectively.

2.6.1 PowerPC Memory Management

The primary functions of the MMU are to translate logical (effective) addresses to physical addresses for memory accesses, I/O accesses (I/O accesses are assumed to be memory-mapped), and to provide access protection on blocks and pages of memory.

There are two types of accesses generated by the 602 that require address translation—instruction accesses and data accesses to memory generated by load and store instructions.

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

The hashed page table is a variable-sized data structure that defines the mapping between virtual page numbers and physical page numbers. The page table size is a power of 2, and its starting address is a multiple of its size.

The page table contains a number of page table entry groups (PTEGs). A PTEG contains eight page table entries (PTEs) of 8 bytes each; therefore, each PTEG is 64 bytes long. PTEG addresses are entry points for table search operations.

Address translations are enabled by setting bits in the MSR—MSR[IR] enables instruction address translations and MSR[DR] enables data address translations.

2.6.2 PowerPC 602 Microprocessor Memory Management

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. A TLB is a cache of the most recently used page table entries. Software is responsible for maintaining the consistency of the TLB with memory. The 602's TLBs are 32-entry, two-way set-associative caches that contain instruction and data address translations. The 602 provides hardware assist for software table search operations through the hashed page table on TLB misses. Supervisor software can invalidate TLB entries selectively.

The instruction and data memory management units provide 4 Gbytes of logical address space accessible to supervisor and user programs with a 4-Kbyte page size and 256-Mbyte segment size. Block sizes range from 128 Kbytes to 256 Mbytes and are software-selectable. In addition, the 602 uses an interim 52-bit virtual address and hashed page tables for generating 32-bit physical addresses. The MMUs in the 602 rely on the exception processing mechanism for the implementation of the paged virtual memory environment and for enforcing protection of designated memory areas.

As specified by the PowerPC architecture, the hashed page table is a variable-sized data structure that defines the mapping between virtual page numbers and physical page numbers. The page table size is a power of 2, and its starting address is a multiple of its size.

Also as specified by the PowerPC architecture, the page table contains a number of page table entry groups (PTEGs). A PTEG contains eight page table entries (PTEs) of 8 bytes each; therefore, each PTEG is 64 bytes long. PTEG addresses are entry points for table search operations.

For applications requiring protection for areas of memory larger than 256 Kbytes, the 602 provides an optional configuration of its TLBs. Upon reset, the operating system can configure the TLBs in the protection-only mode, which is described in detail in Section 2.6.2.1, "Protection-Only Mode."

The 602 also provides independent four-entry BAT arrays for instructions and data that maintain address translations for blocks of memory. These entries define blocks that can vary from 128 Kbytes to 256 Mbytes. The BAT arrays are maintained by system software. An address match with an entry in the BATs takes priority over a hit in the TLBs for the same effective address.

2.6.2.1 Protection-Only Mode

In protection-only mode, the 602 allows protection for up to 4 Mbytes of memory per TLB. Effective addresses are not translated through the TLBs in this mode. ISI and DSI exceptions due to address translation through the TLBs are not invoked in this mode. However, such exceptions can still be caused by access protection violations.

This mode has no impact on the usage of the BATs which are available for protection and translation, and are maintained by the system software. An effective address match in the BATs takes priority over a hit in the TLBs.

The MMU features that are provided in this mode that are different from the default configuration of the TLBs are as follows:

- 32-entry, two-way set-associative data and instruction TLBs each provide protection only for thirty-two 4-Kbyte pages per TLB entry. A total of 4 Mbytes of memory can be protected in each TLB. Protection consists of 1 bit per 4-Kbyte page that inhibits execution in the ITLB and enables writes in the DTLB.
- One 24-bit VSID. Only one segment entry can be used in this mode (entry 0). Other entries can be written to but are not used in the address translation in this mode.
- Execution protection for ITLB (on 128-Kbyte blocks of memory at a granularity of 4-Kbyte pages).

2.7 Instruction Timing

The 602 is a pipelined processor with parallel execution units. A pipelined processor is one in which the processing of an instruction is reduced into discrete stages. Because the processing of an instruction is broken into a series of stages, an instruction does not require the entire resources of an execution unit. For example, after an instruction completes the decode stage, it can pass on to the next stage, while the subsequent instruction can advance into the decode stage. This improves the throughput of the instruction flow. For example, it may take three cycles for a floating-point instruction to complete, but if there are no stalls in the floating-point pipeline, a series of floating-point instructions can have a throughput of one instruction per cycle.

The instruction pipeline in the 602 has four major pipeline stages, described as follows:

- The fetch pipeline stage primarily involves retrieving instructions from the memory system and determining the location of the next instruction fetch. Additionally, the BPU decodes branches during the fetch stage and folds out branch instructions before the dispatch stage if possible.
- The decode and dispatch pipeline stage is responsible for decoding the instructions supplied by the instruction fetch stage, and determining which of the instructions are eligible to be dispatched in the current cycle. In addition, the source operands of the instructions are read from the appropriate register file and dispatched with the instruction to the execute pipeline stage. At the end of the dispatch pipeline stage, the dispatched instructions and their operands are latched by the appropriate execution unit.
- During the execute pipeline stage each execution unit that has an executable instruction executes the selected instruction (perhaps over multiple cycles), writes the instruction's result into the appropriate rename register, and notifies the completion stage that the instruction has finished execution. In the case of an internal exception, the execution unit reports the exception to the completion/write-back pipeline stage and discontinues instruction execution until the exception is handled. The exception is not signaled until that instruction is the next to be completed. Execution of most floating-point instructions is pipelined within the FPU allowing up to three instructions to be executing in the FPU concurrently. The pipeline stages for the floating-point unit are multiply, add, and round-convert. Execution of most load/store instructions is also pipelined. The load/store unit has two pipeline stages. The first stage is for effective address calculation and MMU translation and the second stage is for accessing the data in the cache.
- The complete/write-back pipeline stage maintains the correct architectural machine state and transfers the contents of the rename registers to the GPRs and FPRs as instructions are retired. If the completion logic detects an instruction causing an exception, all following instructions are cancelled, their execution results in rename registers are discarded, and instructions are fetched from the correct instruction stream.

The 602 has four independent execution units, one each for integer instructions, floating-point instructions, branch instructions, and load/store instructions. The IU and the FPU each have dedicated register files for maintaining operands (GPRs and FPRs, respectively), allowing integer calculations and floating-point calculations to occur simultaneously without interference.

Because the PowerPC architecture can be applied to such a wide variety of implementations, instruction timing among various PowerPC processors varies accordingly.

2.8 System Interface

The system interface is specific for each PowerPC microprocessor implementation.

The 602 interface includes a time-multiplexed, 32-bit address and 64-bit data bus, and 56 control and information signals (see Figure 5). The system interface allows for address-only transactions as well as address and data transactions. The 602 control and information signals include the bus arbitration, address start, address transfer, transfer attribute, address termination, data transfer, data termination, and processor state signals. Test and control signals provide diagnostics for selected internal circuits.

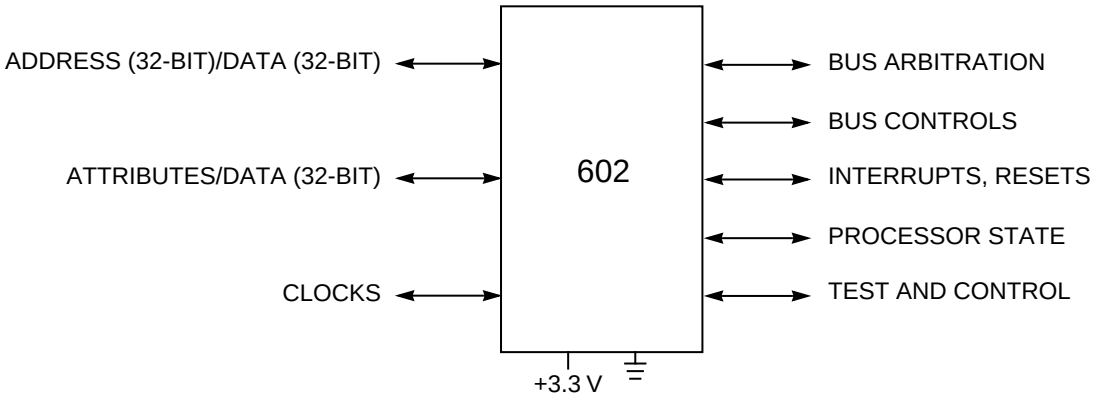


Figure 5. System Interface

The 602 supports multiple masters through a bus arbitration scheme that allows various devices to compete for the shared bus resource. The arbitration logic can implement priority protocols, such as fairness, and can park masters to avoid arbitration overhead. The MEI protocol ensures coherency among multiple devices and system memory. Also, the 602's on-chip caches and TLBs and optional second-level caches can be controlled externally.

The 602’s clocking structure allows the bus to operate at integer multiples of the processor cycle time.

The following sections describe the 602 bus support for memory. Note that some signals perform different functions depending upon the addressing protocol used.

2.8.1 Memory Accesses

The 602 memory accesses allow transfer sizes of 8, 16, 24, 32, or 64 bits in one bus clock cycle. Data transfers occur in either single-beat transactions or four-beat burst transactions. Single-beat transactions are caused by noncached accesses that access memory directly (that is, read and write operations when caching is disabled, cache-inhibited accesses, and stores in write-through mode). Four-beat burst transactions, which always transfer an entire 32-byte cache block, are initiated when a block is read from or written to memory. The 602 initiates two distinct bus transactions in cases of misaligned accesses.

2.8.2 PowerPC 602 Microprocessor Signals

The 602 signals are grouped as follows:

- Bus arbitration signals—The 602 uses these signals to arbitrate for address bus mastership.
- Address transfer start signals—These signals indicate that a bus master has begun a transaction on the address bus.
- Address transfer signals—These signals consist of the address, and prefetch line-fill address buses.

- Transfer attribute signals—These signals provide information about the type of transfer, such as the transfer size and whether the transaction is a burst, write-through, or cache-inhibited transaction.
- Address transfer termination signals—These signals are used to acknowledge the end of the address phase of the transaction. They also indicate whether a condition exists that requires the address phase to be repeated.
- Data transfer signals—These signals consist of the 64-bit data bus.
- Data transfer termination signals—Data termination signals are required after each data beat in a data transfer. In a single-beat transaction, the data termination signals also indicate the end of the tenure, while in burst accesses, the data termination signals apply to individual beats and indicate the end of the tenure only after the final data beat.
- System status signals—These signals include the interrupt signal, checkstop signals, and both soft- and hard-reset signals. These signals are used to interrupt and, under various conditions, to reset the processor.
- Processor state signals—These signals enable the time base, and provide machine quiesce control.
- IEEE 1149.1(JTAG)/COP interface signals—The IEEE 1149.1 test unit and the common on-chip processor (COP) unit are accessed through a shared set of input, output, and clocking signals. The IEEE 1149.1/COP interface provides a means for boundary-scan testing and internal debugging of the 602.
- Test interface signals—These signals are used for production testing.
- Clock signals—These signals determine the system clock frequency. These signals can also be used to synchronize multiprocessor systems.

NOTE

A bar over a signal name indicates that the signal is active low—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, for example, TT0–TT4 (transfer type signals), are referred to as asserted when they are high and negated when they are low.

2.8.3 Signal Configuration

Figure 6 illustrates the 602's logical pin configuration, showing how the signals are grouped.

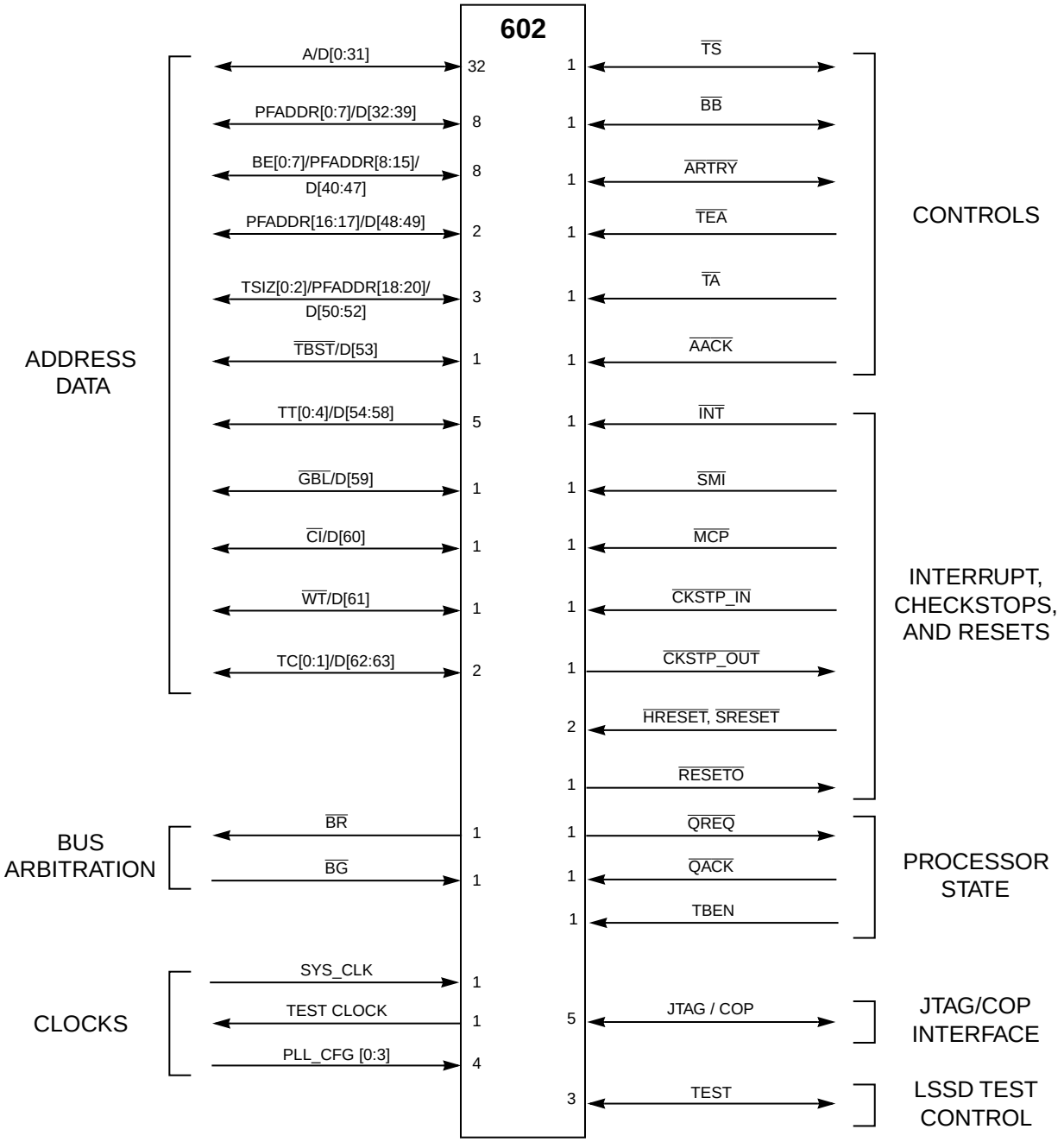


Figure 6. PowerPC 602 Microprocessor Signal Groups

Motorola Literature Distribution Centers:
USA: Motorola Literature Distribution, P.O. Box 20912, Phoenix, Arizona 85036.
EUROPE: Motorola Ltd., European Literature Centre, 88 Tanners Drive, Blakelands, Milton Keynes, MK14 5BP, England.
JAPAN: Nippon Motorola Ltd., 4-32-1, Nishi-Gotanda, Shinagawa-ku, Tokyo 141 Japan.
ASIA-PACIFIC: Motorola Semiconductors H.K. Ltd., Silicon Harbour Centre, No. 2 Dai King Street, Tai Po Industrial Estate, Tai Po, N.T., Hong Kong.
Technical Information: Motorola Inc. Semiconductor Products Sector Technical Responsiveness Center; (800) 521-6274.
Document Comments: FAX (512) 891-2638, Attn: RISC Applications Engineering.

IBM Microelectronics:
USA: IBM Microelectronics, Mail Stop A25/862-1, PowerPC Marketing, 1000 River Street, Essex Junction, VT 05452-4299; Tel.: (800) PowerPC [(800) 769-3772]; FAX (800) POWERfax [(800) 769-3732].
EUROPE: IBM Microelectronics, PowerPC Marketing, Dept. 1045, 224 Boulevard J.F. Kennedy, 91105 Corbeil-Essonnes CEDEX, France; Tel. (33) 1-60-88 5167; FAX (33) 1-60-88 4920.
JAPAN: IBM Microelectronics, PowerPC Marketing, Dept., R0260, 800 Ichimiyake, Yasu-cho, Yasu-gun, Shinga-ken, Japan 520-23; Tel. (81) 775-87-4745; FAX (81) 775-87-4735.