MIPS32® 34Kf™ Processor Core Datasheet August 25, 2006

The MIPS32® 34Kf™ core from MIPS Technologies is a high-performance, low-power, 32-bit MIPS RISC core designed for custom system-on-silicon applications. The core is designed for semiconductor manufacturing companies, ASIC developers, and system OEMs who want to rapidly integrate their own custom logic and peripherals with a high-performance RISC processor. Fully synthesizable and highly portable across processes, it can be easily integrated into full system-on-silicon designs, allowing developers to focus their attention on end-user products.

The 34Kf core implements the MIPS32 Release 2 Architecture in a 9-stage pipeline. It is the first MIPS core to implement the MIPS MT application specific extension (ASE) which defines multi-threaded operation. It also implements the MIPS DSP ASE, which provides support for signal processing instructions. It additionally includes support for the MIPS16e™ ASE and the 32-bit privileged resource architecture. This standard architecture allows support by a wide range of industry standard tools and development systems.

The MT ASE allows the core to operate more efficiently by executing multiple program streams concurrently. The core can be configured with 1 or 2 Virtual Processing Elements (VPEs), each of which contain much of the privileged coprocessor 0 state, including a full Memory Management Unit (MMU), to allow multiple OSes to operate concurrently on the processor. Additionally, the core can be configured to have from 1-9 Thread Contexts (TCs). A TC consists of a register file, a Program Counter, and a limited amount of privileged state. TCs offer lightweight multi-threading to allow cooperative or independent threads to run concurrently.

The DSP ASE provides support for a number of powerful data processing operations. There are instructions for doing fractional arithmetic (Q15/Q31) and for saturating arithmetic. Additionally, for smaller data sizes, SIMD operations are supported, allowing 2x16b or 4x8b operations to occur simultaneously. Another feature of the ASE is the inclusion of additional HI/LO accumulator registers to improve the parallellization of independent accumulation routines.

The 34Kf core also features an IEEE 754 compliant Floating Point Unit (FPU). The FPU supports both single and double precision instructions.

The synthesizable 34Kf core includes a high performance Multiply/Divide Unit (MDU). The MDU is fully pipelined to support a single cycle repeat rate for 32x32 MAC instructions. Further, in the 34Kf Pro™ Core, the optional CorExtend block can utilize the HI/LO registers in the MDU block. The CorExtend block allows specialized functions to be efficiently implemented.

Instruction and data level one caches are configurable at 0, 8, 16, 32, or 64 KB in size. Each cache is organized as 4-way set associative. Data cache misses are non-blocking and up to 8 may be outstanding. Two instruction cache misses can be outstanding. Both caches are virtually indexed and physically tagged to allow them to be accessed in the same cycle that the address is translated. To achieve high frequencies while using commercially available SRAM generators, the cache access is spread across two pipeline stages, dedicating nearly an entire cycle for the SRAM access.

The Bus Interface Unit implements the Open Core Protocol (OCP) which has been developed to address the needs of SOC designers. This implementation features 64-bit read and write data buses to efficiently transfer data to and from the L1 caches. The BIU also supports a variety of core/bus clock ratios to give greater flexibility for system design implementations.

An Enhanced JTAG (EJTAG) block allows for software debugging of the processor, and includes a TAP controller as well as optional instruction and data virtual address/value breakpoints.

Figure 1 shows a block diagram of the 34Kf core. The dashed boxes indicate blocks that can be modified by the customer for specific applications.

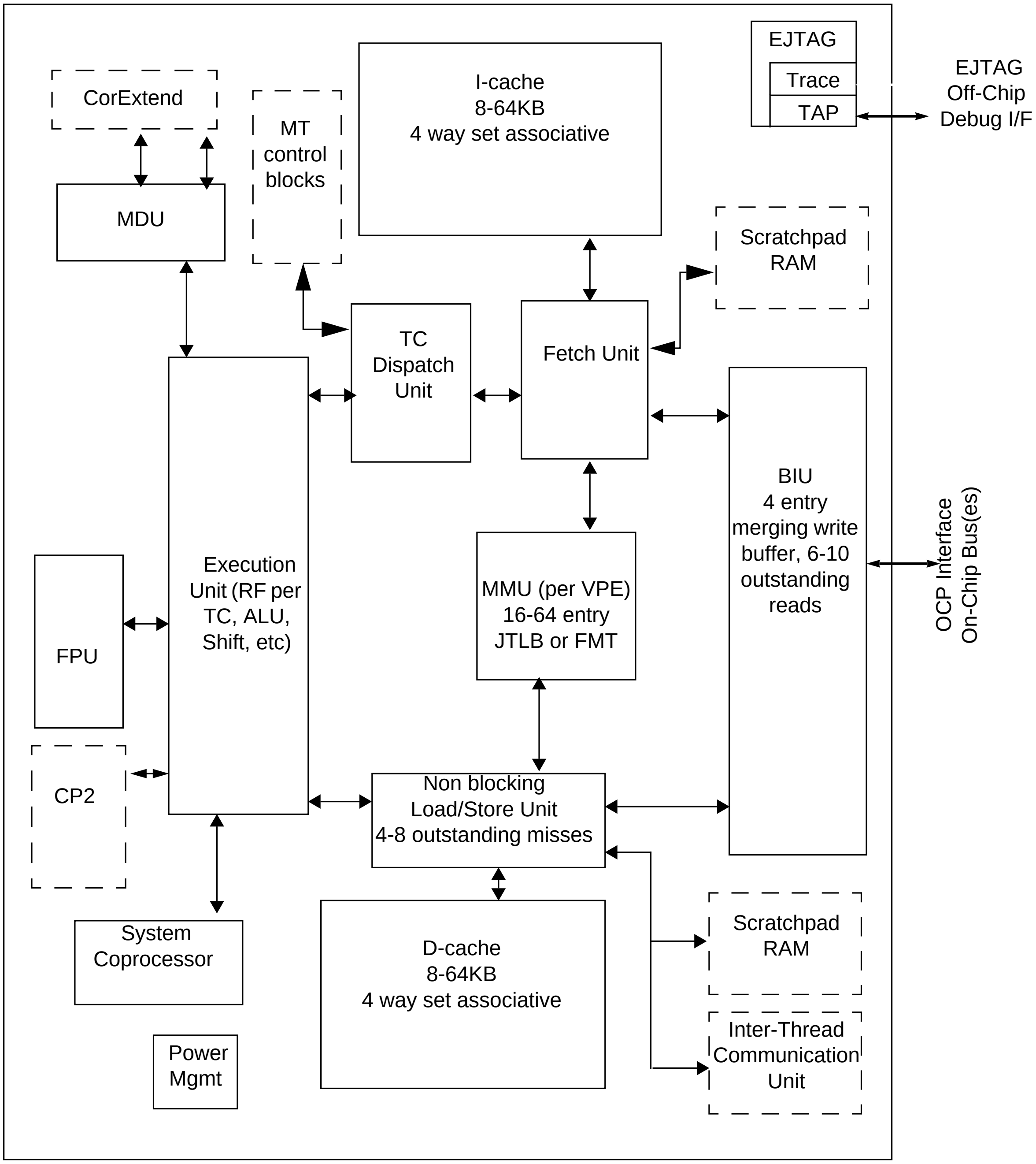
CorExtend
MT control blocks
I-cache 8-64KB 4 way set associative
EJTAG
Trace
TAP
EJTAG Off-Chip Debug I/F
MDU
Scratchpad RAM
TC Dispatch Unit
Fetch Unit
Execution Unit (RF per TC, ALU, Shift, etc)
BIU 4 entry merging write buffer, 6-10 outstanding reads
OCP Interface On-Chip Bus(es)
MMU (per VPE) 16-64 entry JTLB or FMT
FPU
CP2
Non blocking Load/Store Unit 4-8 outstanding misses
System Coprocessor
D-cache 8-64KB 4 way set associative
Scratchpad RAM
Inter-Thread Communication Unit
Power Mgmt

Figure 1 34Kf™ Core Block Diagram

34Kf™ Core Features

- 9-stage pipeline
- 32-bit address paths
- 64-bit data paths to caches and external interface
- MIPS32 Release2 Instruction Set and Privileged Resource Architecture
- MIPS16e™ Code Compression
- MIPS MT Application Specific Extension (ASE)

  - Support for 1 or 2 Virtual Processing Elements (VPEs)
  - Support for 1-9Thread Contexts (TCs)
  - Inter-Thread Communication (ITC) memory for efficient communication & data transfer.
- MIPS DSP ASE
  - 3 additional pairs of accumulator registers.
  - Fractional data types (Q15, Q31)
  - Saturating arithmetic
  - SIMD instructions operate on 2x16b or 4x8b simultaneously.
- Programmable Memory Management Unit
  - 16/32/64 dual-entry JTLB per VPE
  - JTLBs are sharable under software control
  - 4-9 entry MT-optimized ITLB
  - 8-entry DTLB
  - Optional simple Fixed Mapping Translation (FMT) mechanism
- Programmable L1 Cache Sizes
  - Individually configurable instruction and data caches
  - Cache sizes of 0/8/16/32/64 KB
  - 4-Way Set Associative
  - Up to 9 outstanding load misses
  - Write-back and write-through support
  - 32-byte cache line size
  - Virtually indexed, physically tagged
  - Cache line locking support
  - Non-blocking prefetches
  - Optional parity support
- Scratchpad RAM support
  - Separate RAMs for Instruction and Data
  - Independent of cache configuration
  - 2 64 bit OCP interfaces for external DMA
  - OCP port runs at the same core/bus clock ratio as that of BIU interface
- Bus Interface
  - OCP interface with 32-bit address and 64-bit data
  - OCP interface runs at core/bus clock ratios of 1, 1.5, 2, 2.5, 3, 3.5, 4, 5 or 10 via a separate synchronous bus clock
  - Burst size of four 64-bit beats
  - 4 entry write buffer
  - “Simple” byte enable mode allows easier bridging to other bus standards
  - Extensions for front-side L2 cache
- Multiply/Divide Unit
  - Maximum issue rate of one 32x32 multiply per clock
  - 5 cycle multiply latency
  - Early-in iterative divide. Minimum 11 and maximum 34 clock latency (dividend (*rs*) sign extension-dependent)
- CorExtend™ User Defined Instruction Set Extensions
  - Separately licensed; a core with this feature is known as the 34Kf Pro™ core
  - Allows user to define and add instructions to the core at build time
  - Maintains full MIPS32 compatibility
  - Supported by industry standard development tools
  - Single or multi-cycle instructions
  - Includes access to HI and LO registers
- Floating Point Unit (FPU)
  - IEEE-754 compliant Floating Point Unit
  - Compliant to MIPS 64b FPU standards
  - Supports single and double precision datatypes
  - Optionally run at 1:1 or 2:1 core/FPU clock ratio
- Coprocessor 2 interface
  - 64 bit interface to a user designed coprocessor
- Power Control
  - Minimum frequency: 0 MHz
  - Power-down mode (triggered by WAIT instruction)
  - Support for software-controlled clock divider
  - Support for extensive use of fine-grained clock gating
- EJTAG Debug
  - Support for single stepping
  - Instruction address and data address/value breakpoints
  - TAP controller is chainable for multi-CPU debug
  - Cross-CPU breakpoint support
- MIPS Trace
  - PC, data address and data value tracing w/ trace compression
  - Support for on-chip and off-chip trace memory
- Testability
  - Full scan design achieves test coverage in excess of 99% (dependent on library and configuration options)
  - Optional memory BIST for internal SRAM arrays

Pipeline Flow

The 34Kf core implements a 9-stage pipeline. Two extra fetch stages are conditionally added when executing MIPS16e instructions. This pipeline allows the processor to achieve a high frequency while maintaining reasonable area and power numbers.

Figure 2 shows a diagram of the 34Kf core pipeline.

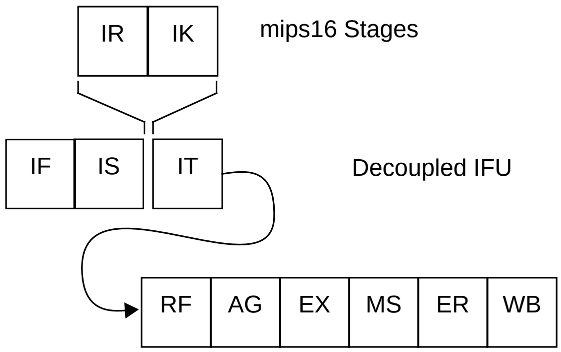

Figure 2 34Kf™ Core Pipeline

IF Stage: Instruction Fetch First

- I-cache tag/data arrays accessed
- Branch History Table accessed
- ITLB address translation performed
- Instruction watch and EJTAG break compares done

IS - Instruction Fetch Second

- Detect I-cache hit
- Way select
- Branch prediction

IR - Instruction Recode

- MIPS16e instruction recode

IK - Instruction Kill

- MIPS16e instruction kill

IT - Instruction Fetch Third

- Instruction Buffer
- Branch target calculation

RF - Register File Access

- Register File access
- Instruction decoding/dispatch logic
- Bypass muxes

AG - Address Generation

- D-cache Address Generation
- bypass muxes

EX - Execute/Memory Access

- skewed ALU
- DTLB
- DCache SRAM access
- Branch Resolution
- Data watch and EJTAG break address compares

MS - Memory Access Second

- DCache hit detection
- Way select mux
- Load align

ER- Exception Resolution

- Instruction completion
- Register file write setup
- Exception processing

WB - Writeback

- Register file writeback occurs on rising edge of this cycle

34Kf™ Core Logic Blocks

The 34Kf core consists of the following logic blocks, shown in Figure 1. These logic blocks are defined in the following subsections.

Fetch Unit

This block is responsible for fetching instructions for all Thread Contexts (TCs). Each TC has an 8 entry instruction buffer (IBF) that decouples the fetch unit from the execution unit. When executing instructions from multiple TCs, a portion of the IBF is used as a skid buffer. Instructions are held in the IBF after being sent to the execution unit. This allows stalled instructions to be flushed from the execution pipeline without needing to be refetched.

In order to fetch instructions without intervention from the execution unit, the fetch unit contains branch prediction logic. A 512-entry Branch History Table (BHT) is used to predict the direction of branch instructions. It uses a bimodal algorithm with two bits of history information per entry. Also, a 4-entry Return Prediction Stack (RPS) is a simple structure to hold the return address from the most recent subroutine calls. The link address is pushed onto the stack whenever a JAL, JALR, or BGEZAL instruction is seen. Then that address is popped when a JR instruction occurs. The BHT is shared by all TCs on the processor, while the RPS is dynamically associated with a single TC.

Thread Schedule Unit (TSU)

This unit is responsible for dispatching instructions from different Thread Contexts (TCs). An external policy manager assigns priorities for each TC. The TSU determines which TCs are runnable and selects the highest priority one available. If multiple are available, a round-robin mechanism will select between them fairly.

The policy manager is a customer configurable block. Simple round-robin or fixed priority policies can be implemented by tying off signals on the interface. A reference policy manager is also included that implements a weighted round-robin algorithm for long-term distribution of execution bandwidth.

Execution Unit

The 34Kf core execution unit implements a load/store architecture with single-cycle ALU operations (logical, shift, add, subtract) and an autonomous multiply/divide unit. Each TC on a 34Kf core contains thirty-one 32-bit general-purpose registers used for integer operations and address calculation. The register file consists of two read ports and one write port and is fully bypassed to minimize operation latency in the pipeline.

The execution unit includes:

- 32-bit adder used for calculating the data address
- Logic for verifying branch prediction
- Load aligner
- Bypass multiplexers used to avoid stalls when executing instructions streams where data producing instructions are followed closely by consumers of their results
- Leading Zero/One detect unit for implementing the CLZ and CLO instructions
- Arithmetic Logic Unit (ALU) for performing bitwise logical operations
- Shifter & Store Aligner

MIPS16e™ Application Specific Extension

The 34Kf core includes support for the MIPS16e ASE. This ASE improves code density through the use of 16-bit encodings of many MIPS32 instructions plus some MIPS16e-specific instructions. PC relative loads allow quick access to constants. Save/Restore macro instructions provide for single instruction stack frame setup/teardown for efficient subroutine entry/exit.

Multiply/Divide Unit (MDU)

The 34Kf core includes a multiply/divide unit (MDU) that contains a separate pipeline for integer multiply and divide operations. This pipeline operates in parallel with the integer unit pipeline and does not stall when the integer pipeline stalls. This allows any long-running MDU operations to be masked by instructions on other TCs and/or other integer unit instructions.

The MDU consists of a pipelined 32x32 multiplier, result/accumulation registers (HI and LO), a divide state machine, and the necessary multiplexers and control logic.

The MDU supports execution of one multiply or multiply accumulate operation every clock cycle.

Divide operations are implemented with a simple 1 bit per clock iterative algorithm. An early-in detection checks the sign extension of the dividend (*rs*) operand. If rs is 8 bits wide, 23 iterations are skipped. For a 16-bit-wide rs, 15 iterations are skipped, and for a 24-bit-wide rs, 7 iterations are skipped. Any attempt to issue a subsequent MDU instruction while a divide is still active causes a pipeline stall until the divide operation is completed.

Table 1 lists the repeat rate (peak issue rate of cycles until the operation can be reissued) and latency (number of cycles until a result is available) for the 34Kf core multiply and divide instructions. The approximate latency and repeat rates are listed in terms of pipeline clocks. For a more detailed discussion of latencies and repeat rates, refer to Chapter 9 of *Programming the MIPS32 34K™ Core Family.*

Table 1 34Kf™ Core Integer Multiply/Divide Unit Latencies and Repeat Rates

Opcode	Operand Size (mul *rt*) (div *rs*)	Latency	Repeat Rate
MULT/MULTU, MADD/MADDU, MSUB/MSUBU	32 bits	5	1
MUL	32 bits	5	1*

* If there is no data dependency, a MUL can be issued every cycle.

Table 1 34Kf™ Core Integer Multiply/Divide Unit Latencies and Repeat Rates (Continued)

Opcode	Operand Size (mul *rt*) (div *rs*)	Latency	Repeat Rate
DIV/DIVU	8 bits	12/14	12/14
	16 bits	20/22	20/22
	24 bits	28/30	28/30
	32 bits	36/38	36/38
* If there is no data dependency, a MUL can be issued every cycle.			

Floating Point Unit (FPU) / Coprocessor 1

The 34Kf core Floating Point Unit (FPU) implements the MIPS64 ISA (Instruction Set Architecture) for floating-point computation. The implementation supports the ANSI/IEEE Standard 754 (IEEE Standard for Binary Floating-Point Arithmetic) for single and double precision data formats. The FPU contains thirty-two 64-bit floating-point registers used for floating point operations.

The FPU can be configured at build time to run at either the same or one-half the clock rate of the integer core. The FPU is not as deeply pipelined as the integer core so the maximum core frequency will only be attained with the FPU running at one-half the core frequency. The FPU is connected via an internal 64-bit coprocessor interface. Note that clock cycles related to floating point operations are listed in terms of FPU clocks, not integer core clocks.

The performance is optimized for single precision formats. Most instructions have one FPU cycle throughput and four FPU cycle latency. The FPU implements the MIPS64 multiply-add (MADD) and multiply-sub (MSUB) instructions with intermediate rounding after the multiply function. The result is guaranteed to be the same as executing a MUL and an ADD instruction separately, but the instruction latency, instruction fetch, dispatch bandwidth, and the total number of register accesses are improved.

IEEE denormalized input operands and results are supported by hardware for some instructions. IEEE denormalized results are not supported by hardware in general, but a fast flush-to-zero mode is provided to optimize performance. The fast flush-to-zero mode is enabled through the FCCR register, and use of this mode is recommended for best performance when denormalized results are generated.

The FPU has a separate pipeline for floating point instruction execution. This pipeline operates in parallel with the integer core pipeline and does not stall when the integer pipeline stalls. This allows long-running FPU operations, such as divide or square root, to be partially masked by system stalls and/or other integer unit instructions. Arithmetic instructions are always dispatched and completed in order, but loads and stores can complete out of order. The exception model is 'precise' at all times. The FPU is also denoted as "Coprocessor 1".

FPU Pipeline

The FPU implements a high-performance 7-stage pipeline:

- Decode, register read and unpack (FR stage)
- Multiply tree - double pumped for double (M1 stage)
- Multiply complete (M2 stage)
- Addition first step (A1 stage)
- Addition second and final step (A2 stage)
- Packing to IEEE format (FP stage)
- Register writeback (FW stage)

The FPU implements a bypass mechanism that allows the result of an operation to be forwarded directly to the instruction that needs it without having to write the result to the FPU register and then read it back.

Figure 3 shows the FPU pipeline.

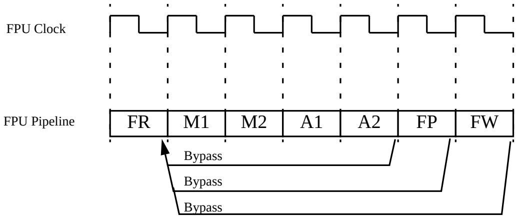

Figure 3 FPU Pipeline

FPU Instruction Latencies and Repeat Rates

Table 2 contains the floating point instruction latencies and repeat rates for the 34Kf core. In this table 'Latency' refers to the number of FPU cycles necessary for the first instruction to produce the result needed by the second instruction. The ‘Repeat Rate’ refers to the maximum rate at which an instruction can be executed per FPU cycle

Table 2 34Kf™ Core FPU Latency and Repeat Rate

Opcode*	Latency (FPU cycles)	Repeat Rate (FPU cycles)
ABS.[S,D], NEG.[S,D], ADD.[S,D], SUB.[S,D], C.cond.[S,D], MUL.S	4	1
MADD.S, MSUB.S, NMADD.S, NMSUB.S, CABS.cond.[S,D]	4	1
CVT.D.S, CVT.PS.PW, CVT.[S,D].[W,L]	4	1
CVT.S.D, CVT.[W,L].[S,D], CEIL.[W,L].[S,D], FLOOR.[W,L].[S,D], ROUND.[W,L].[S,D], TRUNC.[W,L].[S,D]	4	1
MOV.[S,D], MOVF.[S,D], MOVN.[S,D], MOVT.[S,D], MOVZ.[S,D]	4	1
MUL.D	5	2
MADD.D, MSUB.D, NMADD.D, NMSUB.D	5	2
RECIP.S	13	10
RECIP.D	26	21
RSQRT.S	17	14
RSQRT.D	36	31
DIV.S, SQRT.S	17	14
DIV.D, SQRT.D	32	29
MTC1, DMTC1, LWC1, LDC1, LDXC1, LUXC1, LWXC1	4	1
MFC1, DMFC1, SWC1, SDC1, SDXC1, SUXC1, SWXC1	1	1
* Format: S = Single, D = Double, W = Word, L = Longword		

System Control Coprocessor (CP0)

In the MIPS architecture, CP0 is responsible for the virtual-to-physical address translation and cache protocols, the exception control system, the processor’s diagnostic capability, the operating modes (kernel, user, supervisor, and debug), and whether interrupts are enabled or disabled. Configuration information, such as cache size and associativity, presence of features like MIPS16e or floating point unit, is also available by accessing the CP0 registers.

Coprocessor 0 also contains the logic for identifying and managing exceptions. Exceptions can be caused by a variety of sources, including boundary cases in data, external events, or program errors.

Most of CP0 is replicated per VPE. A small amount of state is replicated per TC and some is shared between the VPEs.

Interrupt Handling

Each 34Kf VPE includes support for six hardware interrupt pins, two software interrupts, a timer interrupt, and a performance counter interrupt. These interrupts can be used in any of three interrupt modes, as defined by Release 2 of the MIPS32 Architecture:

- Interrupt compatibility mode, which acts identically to that in an implementation of Release 1 of the Architecture.
- Vectored Interrupt (VI) mode, which adds the ability to prioritize and vector interrupts to a handler dedicated to that interrupt, and to assign a GPR shadow set for use during interrupt processing. The presence of this mode is denoted by the VInt bit in the *Config3* register. This mode is architecturally optional; but it is always present on the 34Kf core, so the VInt bit will always read as a 1 for the 34Kf core.
- External Interrupt Controller (EIC) mode, which redefines the way in which interrupts are handled to provide full support for an external interrupt controller handling prioritization and vectoring of interrupts. This presence of this mode denoted by the VEIC bit in the *Config3* register. Again, this mode is architecturally optional. On the 34Kf core, the VEIC bit is set externally by the static input, *SI_EICPresent*, to allow system logic to indicate the presence of an external interrupt controller.

If a TC is configured to be used as a shadow register set, the VI and EIC interrupt modes can specify which shadow set should be used upon entry to a particular vector. The shadow registers further improve interrupt latency by avoiding the need to save context when invoking an interrupt handler.

Modes of Operation

The 34Kf core supports four modes of operation: user mode, supervisor mode, kernel mode, and debug mode. User mode is most often used for application programs. Supervisor mode gives an intermediate privilege level with access to the ksseg address space. Supervisor mode is not supported with the fixed mapping MMU. Kernel mode is typically used for handling exceptions and operating system kernel functions, including CP0 management and I/O device accesses. An additional Debug mode is used during system bring-up and software development. Refer to "EJTAG Debug Support" on page 12 for more information on debug mode.

Memory Management Unit (MMU)

Each 34Kf VPE contains a Memory Management Unit (MMU) that is primarily responsible for converting virtual addresses to physical addresses and providing attribute information for different segments of memory. At synthesis time, the type of MMU can be chosen independently for each VPE from the following options:

- Translation Lookaside Buffer (TLB)
- Fixed Mapping Translation (FMT)

In a dual-TLB configuration, each VPE contains a separate JTLB so that the translations for each are independent from each other. However, there is further configuration option where the JTLBs can be shared. This requires special OS support, but enables a higher-performance MMU with less area impact.

The following sections explain the MMU options in more detail.

Translation Lookaside Buffer (TLB)

The basic TLB functionality is specified by the MIPS32 Privileged Resource Architecture. A TLB provides mapping and protection capability with per-page granularity. The 34Kf implementation allows a wide range of page sizes to be present simultaneously.

The TLB contains a fully associative Joint TLB (JTLB). To enable higher clock speeds, two smaller micro-TLBs are also implemented: the Instruction Micro TLB (ITLB) and the Data Micro TLB (DTLB). When an instruction or data address is calculated, the virtual address is compared to the contents of the appropriate micro TLB (uTLB). If the address is not found in the uTLB, the JTLB is accessed. If the entry is found in the JTLB, that entry is then written into the uTLB. If the address is not found in the JTLB, a TLB exception is taken.

Figure 4 shows how the ITLB, DTLB, and JTLB are implemented in the 34Kf core.

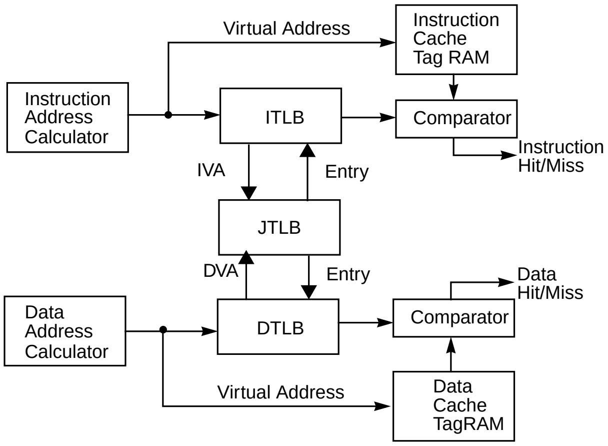

Figure 4   Address Translation During a Cache Access

Joint TLB (JTLB)

The JTLB is a fully associative TLB cache containing 16, 32, or 64-dual-entries mapping up to 128 virtual pages to their corresponding physical addresses. The address translation is performed by comparing the upper bits of the virtual address (along with the ASID) against each of the entries in the *tag* portion of the joint TLB structure.

The JTLB is organized as pairs of even and odd entries containing pages that range in size from 4 KB to 256 MB, in factors of four, into the 4 GB physical address space. The JTLB is organized in page pairs to minimize the overall size. Each *tag* entry corresponds to two data entries: an even page entry and an odd page entry. The highest order virtual address bit not participating in the tag comparison is used to determine which of the data entries is used. Since page size can vary on a page-pair basis, the determination of which address bits participate in the comparison and which bit is used to make the even-odd determination is decided dynamically during the TLB look-up.

Instruction TLB (ITLB)

The ITLB is dedicated to performing translations for the instruction stream. The ITLB is a hybrid structure having 3 entries that are shared by all TCs plus an additional entry dedicated to each TC. Thus, the ITLB may be as large as 9 entries, but each TC may only have its translations in up to 4 places.

The ITLB only maps 4 KB or 1 MB pages/subpages. For 4 KB or 1 MB pages, the entire page is mapped in the ITLB. If the main TLB page size is between 4 KB and 1 MB, only the current 4 KB subpage is mapped. Similarly, for page sizes larger than 1 MB, the current 1 MB subpage is mapped.

The ITLB is managed by hardware and is transparent to software. The larger JTLB is used as a backing structure for the ITLB. If a fetch address cannot be translated by the ITLB, the JTLB is used to attempt to translate it in the following clock cycle, or when available. If successful, the translation information is copied into the ITLB for future use. There is a minimum two cycle ITLB miss penalty.

Data TLB (DTLB)

The DTLB is an 8-entry, fully associative TLB dedicated to performing translations for loads and stores. All entries are shared by all TCs. Similar to the ITLB, the DTLB only maps either 4 KB or 1 MB pages/subpages.

The DTLB is managed by hardware and is transparent to software. The larger JTLB is used as a backing structure for the DTLB. If a load/store address cannot be translated by the DTLB, a lookup is done in the JTLB. If the JTLB translation is successful, the translation information is copied into the DTLB for future use. The DTLB miss penalty is also two cycles.

Fixed Mapping Translation (FMT)

The FMT is much simpler and smaller than the TLB-style MMU, and is a good choice when the full protection and flexibility of the TLB is not needed. Like a TLB, the FMT performs virtual-to-physical address translation and provides attributes for the different segments. Those segments that are unmapped in a TLB implementation (kseg0 and kseg1) are handled identically by the FMT.

Data Cache

The data cache is an on-chip memory block of 0/8/16/32/64 KB, with 4-way associativity. Since the data cache is virtually indexed, the virtual-to-physical address translation occurs in parallel with the cache access. A tag entry holds 20 or 21 bits of physical address, a valid bit, a lock bit, and an optional parity bit. The data entry holds 64 bits of data per way, with optional parity per byte. There are 4 data entries for each tag entry. The tag and data entries exist for each way of the cache. There is an additional array that holds the dirty and LRU replacement algorithm bits for all 4 ways(6b LRU, 4b dirty, and optionally 4b dirty parity).

Using 4KB pages in the TLB and 32 or 64KB cache sizes it is possible to get virtual aliasing. A single physical address can exist in multiple cache locations if it was accessed via different virtual addresses. For a 32KB data cache, there is an implementation option to eliminate virtual aliasing. If this option is not selected, or a 64KB cache is implemented, software must take care of any aliasing issues by using a page coloring scheme or some other mechanism.

The 34Kf core supports data-cache locking. Cache locking allows critical code or data segments to be locked into the cache on a “per-line” basis, enabling the system programmer to maximize the efficiency of the system cache. The locked contents can be updated on a store hit, but will not be selected for replacement on a cache miss.

The cache-locking function is always available on all data-cache entries. Entries can then be marked as locked or unlocked on a per entry basis using the CACHE instruction.

Instruction Cache

The instruction cache is an on-chip memory block of 0/8/16/32/64 KB, with 4-way associativity. Because the instruction cache is virtually indexed, the virtual-to-physical address translation occurs in parallel with the cache access rather than having to wait for the physical address translation. A tag entry holds 20 or 21 bits of physical address, a valid bit, a lock bit, and an optional parity bit. The instruction data entry holds two instructions (64 bits), 6 bits of pre-decode information to speed the decode of branch and jump instructions, and 9 optional parity bits (one per data byte plus one more for the pre-decode information). There are four data entries for each tag entry. The tag and data entries exist for each way of the cache. The LRU replacement bits (6b) are shared among the 4 ways and are stored in a separate array.

The instruction cache block also contains and manages the instruction line fill buffer. Besides accumulating data to be written to the cache, instruction fetches that reference data in the line fill buffer are serviced either by a bypass of that data, or data coming from the external interface. The instruction cache control logic controls the bypass function.

Just like the data cache, with certain cache and TLB page sizes, it is possible to have virtual aliasing in the instruction cache. This is less of a problem because the instruction cache is not written so the aliases are always consistent. If instruction memory is modified, all of the aliases should be flushed from the instruction cache.

The 34Kf core also supports instruction-cache locking. Cache locking allows critical code or data segments to be locked into the cache on a "per-line" basis, enabling the system programmer to maximize the efficiency of the system cache.

The cache-locking function is always available on all instruction-cache entries. Entries can then be marked as locked or unlocked on a per entry basis using the CACHE instruction.

Cache Memory Configuration

The 34Kf core incorporates on-chip instruction and data caches that are usually implemented from readily available single-port synchronous SRAMs and accessed in two cycles: one cycle for the actual SRAM read and another cycle for the tag comparison, hit determination, and way selection. The instruction and data caches each have their own 64-bit data paths and can be accessed simultaneously. Table 3 lists the 34Kf core instruction and data cache attributes.

Table 3 34Kf™ Core Instruction and Data Cache Attributes

Parameter	Instruction	Data
Size	0, 8, 16, 32, or 64 KB*	0, 8, 16, 32, or 64 KB
Organization	4 way set associative	4 way set associative
Line Size	32 Bytes*	32 Bytes
Read Unit	64 bits*	64 bits
Write Policies	N/A	write-through without write allocate, write-back with write allocate
Miss restart after transfer of	miss word	miss word
Cache Locking	per line	per line
*Logical size of instruction cache. Cache physically contains some extra bits used for precoding the instruction type.		

Cache Protocols

The 34Kf core supports the following cache protocols:

- **Uncached:** Addresses in a memory area indicated as uncached are not read from the cache. Stores to such addresses are written directly to main memory, without changing cache contents.
- **Write-through, no write allocate:** Loads and instruction fetches first search the cache, reading main memory only if the desired data does not reside in the cache. On data store operations, the cache is first searched to see if the target address is cache resident. If it is resident, the cache contents are updated, and main memory is also written. If the cache look-up misses, only main memory is written.
- **Write-back, write allocate:** Stores that miss in the cache will cause a cache refill. Store data, however, is only written to the cache. Caches lines that are written by stores will be marked as dirty. If a dirty line is selected for replacement, the cache line will be written back to main memory.
- **Uncached Accelerated:** Like uncached, data is never loaded into the cache. Store data can be gathered in a write buffer before being sent out on the bus as a bursted write. This is more efficient than sending out individual writes as occurs in regular uncached mode.

Scratchpad RAM

The 34Kf core allows blocks of scratchpad RAM to be attached to the load/store and fetch units. These allow low-latency access to a fixed block of memory.

These blocks can be modified by the customer. A reference design is provided which includes an SRAM array as well as an external DMA port to allow the system to directly access the array.

InterThread Communication Unit (ITU)

This block provides a mechanism for efficient communication between TCs using gating storage. This block has a number of locations that can be accessed using different views. These views provide the mechanisms to implement a number of useful communication methods such as mailboxes, FIFO mailboxes, mutexes, and semaphores.

This block can be modified by the customer to target a specific application. A reference ITU design is included with the core that implements some basic views and functionality.

Bus Interface (BIU)

The Bus Interface Unit (BIU) controls the external interface signals. The primary interface implements the Open Core Protocol (OCP). Additionally, the BIU includes a write buffer.

OCP Interface

Table 4 shows the OCP Performance Report for the 34Kf core. This table lists characteristics about the core and the specific OCP functionality that is supported.

Table 4 OCP Performance Report

Core name	34Kf
Vendor Code	0x4d50
Core Code	0x10a
Revision Code	0x1
Core Identity	Additional identification is available through the *PrID* and *EBase* coprocessor0 registers.
Process dependent	Yes
Frequency range for this core	Synthesizable, so varies based on process, libraries, and implementation
Area	
Power Estimate	
Special reset requirements	No
Number of Interfaces	1 OCP master
Master OCP Interface	
Operations issued	RD, WR
Issue rate (per OCP cycle)	One per cycle, for all of the types listed above except for a non-standard RD (SYNC) which depends on ack latency.
Maximum number of operations outstanding	10 read operations. All writes are posted, so the OCP fabric determines the maximum number of outstanding writes.
Burst support and effect on issue rates	Fixed burst length of four 64b beats with single request per burst. Burst sequences of WRAP or XOR supported.
High level flow control	None
Number of threads supported and use of those threads	All transactions utilize a single thread
Connection ID and use of connection information	None
Use of sideband signals	None
Implementation restrictions	1. MReqInfo handled in a user defined way. 3 bits used to send cacheable attribute information or encode type of L2 CACHE instruction, 1 bit used to signify SYNC. 2. MAddrSpace is used (2 bits) to indicate L2/L3 access. 3. Core clock is synchronous but a multiple of the OCP clock. The ratios supported are 1:1, 1:1.5, 1:2, 1:2.5, 1:3, 1:3.5, 1:4, 1:5, 1:10. A helper pulse is required by the core to transfer data from/to the OCP interface without any hazards.

Write Buffer

The BIU contains a merging write buffer. The purpose of this buffer is to store and combine write transactions before issuing them to the external interface. The write buffer is organized as four 32-byte buffers. Each buffer contains data from a single 32-byte aligned block of memory.

When using the write-through cache policy, the write buffer significantly reduces the number of write transactions on the external interface and reduces the amount of stalling in the core due to issuance of multiple writes in a short period of time.

The write buffer also holds eviction data for write-back lines. The load-store unit opportunistically pulls dirty data from the cache and sends it to the BIU. It is gathered in the write buffer and sent out as a bursted write.

For uncached accelerated references, the write buffer can gather multiple writes together and then perform a bursted write to increase the efficiency of the bus. Uncached accelerated gathering is supported for word.

Gathering of uncached accelerated stores will start on cache-line aligned addresses, i.e. 32 byte aligned addresses. Uncached accelerated stores that do not to meet the conditions required to start gathering will be treated like regular uncached stores.

Once an uncached accelerated store meets the requirements needed to start gathering, a gather buffer is reserved for this store. All subsequent uncached accelerated word or double word stores to the same 32B region will write sequentially into this buffer, independent of the word address associated with these latter stores. The uncached accelerated buffer is tagged with the address of the first store.

SimpleBE Mode

To aid in attaching the 34Kf core to structures which cannot easily handle arbitrary byte enable patterns, there is a mode that generates only "simple" byte enables. Only byte enables representing naturally aligned byte, halfword, word, and doubleword transactions will be generated.

The only case where a read can generate "non-simple" byte enables is on an uncached tri-byte load (LWL/LWR). In SimpleBE mode, such a read will be converted into a word read on the external interface.

Writes with non-simple byte enable patterns can arise when a sequence of stores is processed by the merging write buffer, or from uncached tri-byte stores (SWL/SWR). In SimpleBE mode, these stores will be broken into multiple write transactions.

EJTAG Debug Support

The 34Kf core includes an Enhanced JTAG (EJTAG) block for use in the software debug of application and kernel code. In addition to standard user/supervisor/kernel modes of operation, the 34Kf core provides a Debug mode that is entered after a debug exception (derived from a hardware breakpoint, single-step exception, etc.) is taken and continues until a debug exception return (DERET) instruction is executed. During this time, the processor executes the debug exception handler routine.

The EJTAG interface operates through the Test Access Port (TAP), a serial communication port used for transferring test data in and out of the 34Kf core. In addition to the standard JTAG instructions, special instructions defined in the EJTAG specification define what registers are selected and how they are used.

There are several types of simple hardware breakpoints defined in the EJTAG specification. These breakpoints stop the normal operation of the CPU and force the system into debug mode. There are two types of simple hardware breakpoints implemented in the 34Kf core: Instruction breakpoints and Data breakpoints.

During synthesis, the 34Kf core can be configured to support the following breakpoint options per VPE:

- Zero or four instruction breakpoints
- Zero or two data breakpoints

Instruction breaks occur on instruction fetch operations, and the break is set on the virtual address. Instruction breaks can also be made on the ASID value used by the MMU. A mask can be applied to the virtual address to set breakpoints on a range of instructions.

Data breakpoints occur on load and/or store transactions. Breakpoints are set on virtual address and ASID values, similar to the Instruction breakpoint. Data breakpoints can also be set based on the value of the load/store operation. Finally, masks can be applied to both the virtual address and the load/store value.

In debug mode, EJTAG can request that a 'soft' reset be masked. This request is signalled via the *EJ_SRstE* pin. When this pin is deasserted, the system can choose to block some sources of soft reset. Hard resets, such as power-on reset or a reset switch should not be blocked by this signal.

MIPS Trace

The 34Kf core includes optional MIPS Trace support for real-time tracing of instruction addresses, data addresses and data values. The trace information is collected in an on-chip or off-chip memory, for post-capture processing by trace regeneration software. On-chip trace memory may be configured in size from 0 to 1MB; it is accessed through the existing EJTAG TAP interface and requires no additional chip pins. Off-chip trace memory is accessed through a special trace probe and can be configured to use 4, 8, or 16 data pins plus a clock.

Clock and Test Considerations

The following sections describe clocking, power management, and testability features.

Clocking

The core has 3 primary clock domains:

- Core domain - This is the main core clock domain, controlled by the *SI_ClkIn* clock input.
- OCP domain - This domain controls the OCP bus interface logic. This domain is synchronous to *SI_ClkIn*, but can be run at lower frequencies. Core to bus ratios of 1:1, 3:2, 2:1, 5:2, 3:1, 7:2, 4:1, 5:1 and 10:1 are supported.
- TAP domain - This is a low speed clock domain for the EJTAG TAP controller, controlled by the *EJ_TCK* pin. It is asynchronous to *SI_ClkIn*.

Power Management

The 34Kf core offers a number of power management features, including low-power design, active power management, and power-down modes of operation. The core is a static design that supports slowing or halting the clocks, which reduces system power consumption during idle periods.

The 34Kf core provides two mechanisms for system-level low power support:

- Register-controlled power management
- Instruction-controlled power management

Register-Controlled Power Management

The RP bit in the CP0 Status register provides a software mechanism for placing the system into a low power state. The state of the RP bits is available externally via the *SI_RP* and *SI_RP_1* signals. The external agent then decides whether to place the device in a low power mode, such as reducing the system clock frequency.

Three additional bits, $\text{Status}_{\text{EXL}}$, $\text{Status}_{\text{ERL}}$, and Debug_{DM} support the power management function by allowing the user to change the power state if an exception or error occurs while the 34Kf core is in a low power state. Depending on what type of exception is taken, one of these three bits will be asserted. The *SI_EXL*, *SI_ERL*, and *EJ_DebugM* outputs reflect the state on VPE0 and the VPE1 state is shown on *SI_EXL_1*, *SI_ERL_1*, and *EJ_DebugM_1*. The external agent can look at these signals and determine whether to leave the low power state to service the exception.

The following 8 power-down signals are part of the system interface and change state as the corresponding bits in the CP0 registers are set or cleared:

- The *SI_RP[_1]* signal represents the state of the RP bit (27) in the CP0 *Status* register.
- The *SI_EXL[_1]* signal represents the state of the EXL bit (1) in the CP0 *Status* register.
- The *SI_ERL[_1]* signal represents the state of the ERL bit (2) in the CP0 *Status* register.
- The *EJ_DebugM[_1]* signal represents the state of the DM bit (30) in the CP0 *Debug* register.

Instruction-Controlled Power Management

The second mechanism for invoking power-down mode is through execution of the WAIT instruction. When the WAIT instruction is executed, the internal clock is suspended; however, the internal timer and some of the input pins (*SI_Int[5:0], SI_Int_1[5:0], SI_NMI, SI_NMI_1, and SI_Reset*) continue to run. Once the CPU is in instruction-controlled power management mode, any interrupt, NMI, or reset condition causes the CPU to exit this mode and resume normal operation.

The 34Kf core asserts the *SI_Sleep* signal, which is part of the system interface, whenever it has entered low-power operation and gone to sleep. It will enter sleep mode when all bus transactions are complete and all TCs are not running instructions. This happens when a TC is:

- Blocked due to a WAIT instruction
- Blocked due to an outstanding ITC operation
- Yielded
- Halted
- Not Active

Local clock gating

A significant portion of the power consumed by the 34Kf core is often in the clock tree and clocking registers. The core has support for extensive use of local gated-clocks. Power-conscious implementors can use these gated clocks to significantly reduce power consumption within the core.

Internal Scan

Full mux-based scan for maximum test coverage is supported, with a configurable number of scan chains. ATPG test coverage can exceed 99%, depending on standard cell libraries and configuration options.

Memory BIST

The core provides an integrated memory BIST solution for testing the internal cache SRAMs, SPRAM, and the on-chip trace memory using BIST controllers and logic tightly coupled to the cache subsystem. These BIST controllers can be configured to utilize the following algorithms: March C+ or IFA-13.

Memory BIST can also be inserted with a CAD tool or other user-specified method. Wrapper modules and signal buses of configurable width are provided within the core to facilitate this approach.

Build-Time Configuration Options

The 34Kf core allows a number of features to be customized based on the intended application. Table 5 summarizes the key configuration options that can be selected when the core is synthesized and implemented.

For a core that has already been built, software can determine the value of many of these options by querying an appropriate register field. Refer to the *MIPS32 34Kf Processor Core Family Software User's Manual* for a more complete description of these fields. The value of some options that do not have a functional effect on the core are not visible to software.

Table 5 Build-time Configuration Options

Option	Choices	Software Visibility
Number of VPEs	1 or 2	$\text{MVPConf0}_{\text{PVPE}}$
Number of TCs	1-9	$\text{MVPConf0}_{\text{PTC}}$
Integer register file implementation style	Flops or generator	N/A
Number of outstanding data cache misses	4 or 8	N/A
Number of outstanding Loads	4 or 9	N/A
Memory Management Type (per VPE)	TLB or FMT	$\text{Config}_{\text{MT}}$
TLB Size (per VPE)	16, 32, or 64 dual entries	$\text{Config1}_{\text{MMUSize}}$
TLB data array implementation style	Flops or generator	N/A
Instruction hardware breakpoints (per VPE)	0 or 4	DCR_{IB}, IBS_{BCN}
Data hardware breakpoints (per VPE)	0 or 2	DCR_{DB}, DBS_{BCN}
MIPS Trace support	Present or not	$\text{Config3}_{\text{TL}}$
MIPS Trace memory location	On-core, off-chip or both	$\text{TCBCONFIG}_{\text{OnT}}$, $\text{TCBCONFIG}_{\text{OfT}}$
MIPS Trace on-chip memory size	256B - 8MB	$\text{TCBCONFIG}_{\text{SZ}}$
MIPS Trace triggers	0 - 8	$\text{TCBCONFIG}_{\text{TRIG}}$
MIPS Trace source field bits in trace word	0 or 2	$\text{TCBCONTROLB}_{\text{TWSrcWidth}}$
CorExtend interface (Pro only)	Present or not	$\text{Config}_{\text{UDI}}$*
FPU clock ratio relative to integer core	1:1 or 1:2	$\text{Config7}_{\text{FPR}}$
Coprocessor2 interface	Present or not	$\text{Config1}_{\text{C2}}$*
Data ScratchPad RAM interface	Present or not	$\text{Config}_{\text{DSP}}$*

* These bits indicate the presence of external blocks. Bit will not be set if interface is present, but block is not.

Table 5 Build-time Configuration Options (Continued)

Option	Choices	Software Visibility
Instruction ScratchPad RAM interface	Present or not	$\mathrm{Config}_{\mathrm{ISP}}$*
I-cache size	0, 8, 16, 32, or 64 KB	$\mathrm{Config1}_{\mathrm{IL}}$, $\mathrm{Config1}_{\mathrm{IS}}$
D-cache size	0, 8, 16, 32, or 64 KB	$\mathrm{Config1}_{\mathrm{DL}}$, $\mathrm{Config1}_{\mathrm{DS}}$
D-cache hardware aliasing support	Present or not (for 32KB only)	$\mathrm{Config7}_{\mathrm{AR}}$
Cache parity	Present or not	$\mathrm{ErrCtl}_{\mathrm{PE}}$
Memory BIST	Integrated (March C+ or March C+ plus IFA-13), custom, or none	N/A
Clock gating	Top-level, integer register file array, FPU register file array, TLB array, fine-grain, or none	N/A
* These bits indicate the presence of external blocks. Bit will not be set if interface is present, but block is not.		

Revision History

In the left hand page margins of this document you may find vertical change bars to note the location of significant changes to this document since its last release. Significant changes are defined as those which you should take note of as you use the MIPS IP. Changes to correct grammar, spelling errors or similar may or may not be noted with change bars. Change bars will be removed for changes which are more than one revision old.

Please note: Limitations on the authoring tools make it difficult to place change bars on changes to figures. Change bars on figure titles are used to denote a potential change in the figure itself. Certain parts of this document (Instruction set descriptions, EJTAG register definitions) are references to Architecture specifications, and the change bars within these sections indicate alterations since the previous version of the relevant Architecture document.

Table 6

Revision	Date	Description
00.01	August 12, 2004	Initial version
00.50	August 30, 2004	Pre-pre Sales Version.
00.60	February 4, 2005	Fixed some consistency errors on number of outstanding loads and misses
00.70	May 24, 2005	Added ISPRAM details and misc. cleanup
01.00	September 26, 2005	Production release
01.01	March 7, 2006	• Add option for 8KB instruction and data caches • Describe trace capability and options
01.02	August 25, 2006	• Updated to reflect support for 9 TCs

Copyright © 2005,2006 MIPS Technologies, Inc. All rights reserved.

Unpublished rights (if any) reserved under the copyright laws of the United States of America and other countries.

This document contains information that is proprietary to MIPS Technologies, Inc. ("MIPS Technologies"). Any copying, reproducing, modifying or use of this information (in whole or in part) that is not expressly permitted in writing by MIPS Technologies or an authorized third party is strictly prohibited. At a minimum, this information is protected under unfair competition and copyright laws. Violations thereof may result in criminal penalties and fines.

Any document provided in source format (i.e., in a modifiable form such as in FrameMaker or Microsoft Word format) is subject to use and distribution restrictions that are independent of and supplemental to any and all confidentiality restrictions. UNDER NO CIRCUMSTANCES MAY A DOCUMENT PROVIDED IN SOURCE FORMAT BE DISTRIBUTED TO A THIRD PARTY IN SOURCE FORMAT WITHOUT THE EXPRESS WRITTEN PERMISSION OF MIPS TECHNOLOGIES, INC.

MIPS Technologies reserves the right to change the information contained in this document to improve function, design or otherwise. MIPS Technologies does not assume any liability arising out of the application or use of this information, or of any error or omission in such information. Any warranties, whether express, statutory, implied or otherwise, including but not limited to the implied warranties of merchantability or fitness for a particular purpose, are excluded. Except as expressly provided in any written license agreement from MIPS Technologies or an authorized third party, the furnishing of this document does not give recipient any license to any intellectual property rights, including any patent rights, that cover the information in this document.

The information contained in this document shall not be exported, reexported, transferred, or released, directly or indirectly, in violation of the law of any country or international law, regulation, treaty, Executive Order, statute, amendments or supplements thereto. Should a conflict arise regarding the export, reexport, transfer, or release of the information contained in this document, the laws of the United States of America shall be the governing law.

The information contained in this document constitutes one or more of the following: commercial computer software, commercial computer software documentation or other commercial items. If the user of this information, or any related documentation of any kind, including related technical data or manuals, is an agency, department, or other entity of the United States government ("Government"), the use, duplication, reproduction, release, modification, disclosure, or transfer of this information, or any related documentation of any kind, is restricted in accordance with Federal Acquisition Regulation 12.212 for civilian agencies and Defense Federal Acquisition Regulation Supplement 227.7202 for military agencies. The use of this information by the Government is further restricted in accordance with the terms of the license agreement(s) and/or applicable contract terms and conditions covering this information from MIPS Technologies or an authorized third party.

MIPS, MIPS I, MIPS II, MIPS III, MIPS IV, MIPS V, MIPS-3D, MIPS16, MIPS16e, MIPS32, MIPS64, MIPS-Based, MIPSsim, MIPSpro, MIPS Technologies logo, MIPS RISC CERTIFIED POWER logo, MIPS-VERIFIED, 4K, 4Kc, 4Km, 4Kp, 4KE, 4KEc, 4KEm, 4KEp, 4KS, 4KSc, 4KSd, M4K, 5K, 5Kc, 5Kf, 20K, 20Kc, 24K, 24Kc, 24Kf, 24KE, 24KEc, 24KEf, 25Kf, 34K, 34Kc, 34Kf, R3000, R4000, R5000, ASMACRO, Atlas, "At the core of the user experience.", BusBridge, CorExtend, CoreFPGA, CoreLV, EC, JALGO, Malta, MDMX, MGB, PDtrace, the Pipeline, Pro Series, QuickMIPS, SEAD, SEAD-2, SmartMIPS, SOC-it, and YAMON are trademarks or registered trademarks of MIPS Technologies, Inc. in the United States and other countries.

All other trademarks referred to herein are the property of their respective owners.

Template: D1.14, Built with tags: 2B