MIPS32® 24Kf™ Processor Core Datasheet

December 19, 2008

The MIPS32® 24Kf™ 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 24Kf core implements the MIPS32 Release 2 Architecture in an 8-stage pipeline. It includes support for the MIPS16e™ application specific extension and the 32-bit privileged resource architecture. This standard architecture allows support by a wide range of industry-standard tools and development systems.

To maintain high pipeline utilization, dynamic branch prediction is included in the form of a Branch History Table and a Return Prediction Stack. The Memory Management Unit (MMU) contains 4-entry instruction and 8-entry data Translation Lookaside Buffers (ITLB/DTLB) and a configurable 16/32/64 dual-entry joint TLB (JTLB) with variable page sizes. Alternatively, for applications not requiring address mapping or protection, the TLBs can be replaced with a simple Fixed Mapping mechanism.

The 24Kf core also features an IEEE 754 compliant Floating Point Unit (FPU). The FPU supports both single and double precision instructions.

The synthesizable 24Kf 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, which enables multiply-intensive algorithms to be performed efficiently. Further, in the 24Kf 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, leaving nearly an entire cycle for the SRAM access.

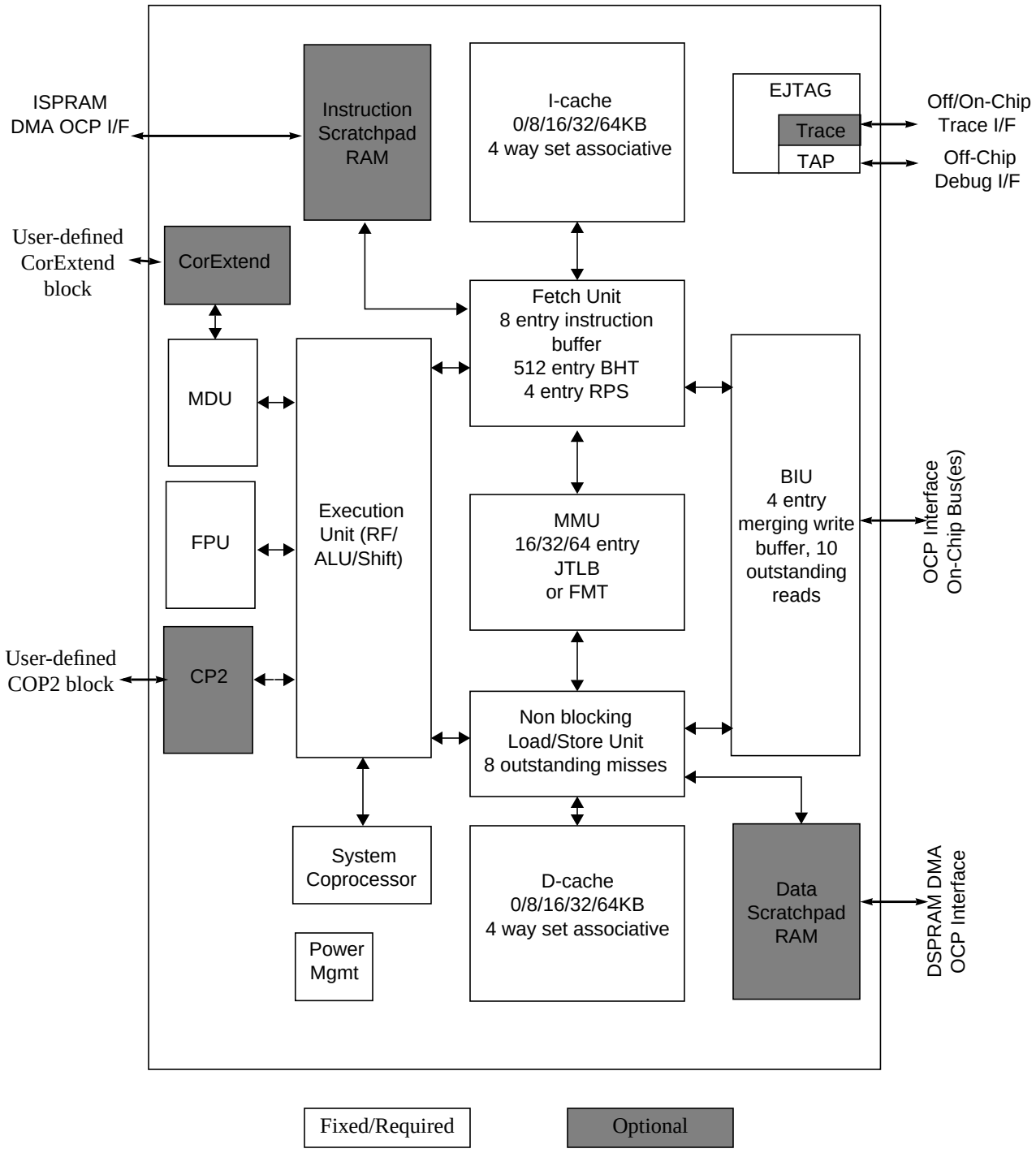
The Bus Interface Unit implements the Open Core Protocol (OCP) which has been developed to address the needs of SOC designers. This implementation features 64-bit read and write data buses to efficiently transfer data to and from the L1 caches. The BIU also supports a variety of core/bus clock ratios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

The core features optional support for external interfaces to a coprocessor block and scratchpad RAMs. Separate instruction and data scratchpads are supported, with reference designs featuring external OCP interfaces for system access to the arrays.

An Enhanced JTAG (EJTAG) compliant block allows for software debugging of the processor and includes a TAP controller as well as optional instruction and data virtual address/value breakpoints. Additionally, real-time tracing of instruction program counter, data address and data values can be supported.

Figure 1 shows a block diagram of the 24Kf core.

Figure 1 24Kf™ Core Block Diagram



24Kf™ Core Features

- 8-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
- Programmable Memory Management Unit
  - 16/32/64 dual-entry JTLB with variable page sizes
  - 4-entry ITLB
  - 8-entry DTLB
  - Optional simple Fixed Mapping Translation (FMT) mechanism
  - Individually configurable instruction and data caches sizes of 0/8/16/32/64 KB
  - 4-Way Set Associative
  - Up to 8 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
  - Maximum size of 1MB each
  - Interface allows back-stalling the core
  - Reference designs available featuring two 64 bit OCP interfaces for external DMA
- Bus Interface
  - OCP 2.1 compliant
  - OCP interface with 32-bit address and 64-bit data
  - Flexible core:bus clock ratio support
  - 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 12 and maximum 38 clock latency (dividend (*rs*) sign extension-dependent)
- CorExtend™ User Defined Instruction Set Extensions (available in 24Kf Pro™ core)
  - Separately licensed; a core with this feature is known as the 24Kf Pro™ core
  - Allows user to define and add instructions to the CPU 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 local gated clocks
- EJTAG Debug
  - Support for single stepping
  - Virtual instruction 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
  - PDTrace version 4.1 compliant
- Testability
  - Full scan design achieves test coverage in excess of 99% (dependent on library and configuration options)
  - Optional memory BIST for internal SRAM arrays

Architecture Overview

The 24Kf core contains a variety of blocks some of which are always present, while others are optional.

The required blocks are as follows:

- Fetch Unit
- Execution Unit
- MIPS16e recode
- System Control Coprocessor (CP0)
- Memory Management Unit (MMU)
- Cache Controllers
- Bus Interface Unit (BIU)

- Power Management
- Floating Point Unit

Optional blocks include:

- CorExtend™ User Defined Instruction (UDI) support
- Enhanced JTAG (EJTAG) breakpoints
- MIPS Trace (PDtrace™) support
- Instruction/Data scratchpad
- COP2 interface

Pipeline Flow

The 24Kf core implements an 8-stage pipeline. Three 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.

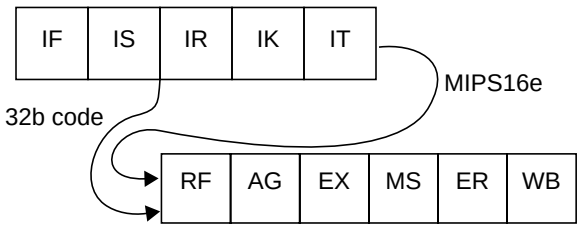
The 24Kf core pipeline consists of the following stages:

- IF - Instruction Fetch First
- IS - Instruction Fetch Second
- IR - Instruction Recode (MIPS16e only)
- IK - Instruction Kill (MIPS16e only)
- IT - Instruction Fetch Third (MIPS16e only)
- RF - Register File access
- AG - Address Generation
- EX - Execute
- MS - Memory Second
- ER - Exception Resolution
- WB - WriteBack

The 24Kf core 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 register and then read it back.

Figure 2 shows a diagram of the 24Kf core pipeline.

Figure 2 24Kf™ 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
- MIPS32 Branch prediction

IR - Instruction Recode

- MIPS16e instruction recode
- MIPS16e branch prediction

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
- Start DCache access
- Branch Resolution
- Data watch and EJTAG break address compares

MS - Memory Access Second

- Complete DCache access
- DCache hit detection
- Way select mux
- Load align
- EJTAG break data value compare

ER- Exception Resolution

- Instruction completion
- Register file write setup
- Exception processing

WB - Writeback

- Register file writeback occurs on rising edge of this cycle

24Kf™ Core Logic Blocks

The 24Kf core consists of the following logic blocks, shown in Figure 1. These logic blocks are defined in the following subsections:

- Fetch Unit
- Execution Unit
- Floating Point Unit (FPU) / Coprocessor 1
- MIPS16e support
- System Control Coprocessor (CP0)
- Memory Management Unit (MMU)
- Cache Controller
- Bus Interface Unit (BIU)
- Power Management

Fetch Unit

The 24Kf core fetch unit is responsible for fetching instructions and providing them to the rest of the pipeline, as well as handling control transfer instructions (branches, jumps, etc.). It calculates the address for each instruction fetch and contains an instruction buffer that decouples the fetching of instructions from their execution.

The fetch unit contains two structures for the dynamic prediction of control transfer instructions. 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.

Execution Unit

The 24Kf core execution unit implements a load/store architecture with single-cycle ALU operations (logical, shift, add, subtract) and an autonomous multiply/divide unit. The 24Kf core contains thirty-two 32-bit general-purpose registers used for integer operations and address calculation. Optionally, one or three additional register file shadow sets (each containing thirty-two registers) can be added to minimize context switching overhead during interrupt/exception processing. 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

Floating Point Unit (FPU) / Coprocessor 1

The 24Kf 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

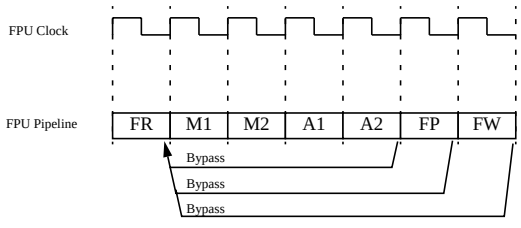
The FPU implements a high-performance 7-stage pipeline:

- 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 1 contains the floating point instruction latencies and repeat rates for the 24Kf 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 1 24Kf™ 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 = Long-word		

MIPS16e™ Application Specific Extension

The 24Kf core includes support for the MIPS16e ASE. This ASE improves code density through the use of 16-bit encoding 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 24Kf 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 partially masked by system stalls 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 2 lists the latencies (number of cycles until a result is available) and repeat rates (peak issue rate of cycles until the operation can be reissued) for the 24Kf 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 2 of the *MIPS32 24K Processor Core Family Software User's Manual*.

Table 2 24Kf™ 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[1]

Table 2 24Kf™ Core Integer Multiply/Divide Unit Latencies and Repeat Rates

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

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

The MIPS architecture defines that the result of a multiply or divide operation be placed in the HI and LO registers. Using the Move-From-HI (MFHI) and Move-From-LO (MFLO) instructions, these values can be transferred to the general-purpose register file.

In addition to the HI/LO targeted operations, the MIPS32 architecture also defines a multiply instruction, MUL, which places the least significant results in the primary register file instead of the HI/LO register pair.

Two other instructions, multiply-add (MADD) and multiply-subtract (MSUB), are used to perform the multiply-accumulate and multiply-subtract operations. The MADD instruction multiplies two numbers and then adds the product to the current contents of the HI and LO registers. Similarly, the MSUB instruction multiplies two operands and then subtracts the product from the HI and LO registers. The MADD and MSUB operations are commonly used in DSP algorithms.

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.

Interrupt Handling

The 24Kf core 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 24Kf core, so the VInt bit will always read as a 1 for the 24Kf 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. The presence of this mode is denoted by the VEIC bit in the *Config3* register. Again, this mode is architecturally optional. On the 24Kf 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.

The reset state of the processor is to interrupt compatibility mode such that a processor supporting Release 2 of the Architecture, like the 24Kf core, is fully compatible with implementations of Release 1 of the Architecture.

VI or EIC interrupt modes can be combined with the optional shadow registers to 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.

GPR Shadow Registers

Release 2 of the MIPS32 Architecture optionally removes the need to save and restore GPRs on entry to high priority interrupts or exceptions, and to provide specified processor modes with the same capability. This is done by introducing multiple copies of the GPRs, called *shadow sets*, and allowing privileged software to associate a shadow set with entry to kernel mode via an interrupt vector or exception. The normal GPRs are logically considered shadow set zero.

The number of GPR shadow sets is a build-time option on the 24Kf core. Although Release 2 of the Architecture defines a maximum of 16 shadow sets, the core allows one (the normal GPRs), two, or four shadow sets. The highest number actually implemented is indicated by the $SRSCtl_{HSS}$ field. If this field is zero, only the normal GPRs are implemented.

Shadow sets are new copies of the GPRs that can be substituted for the normal GPRs on entry to kernel mode via an interrupt or exception. Once a shadow set is bound to a kernel mode entry condition, reference to GPRs work exactly as one would expect, but they are redirected to registers that are dedicated to that condition. Privileged software may need to reference all GPRs in the register file, even specific shadow registers that are not visible in the current mode. The RDPGPR and WRPGPR instructions are used for this purpose. The CSS field of the *SRSCtl* register provides the number of the current shadow register set, and the PSS field of the *SRSCtl* register provides the number of the previous shadow register set (that which was current before the last exception or interrupt occurred).

If the processor is operating in VI interrupt mode, binding of a vectored interrupt to a shadow set is done by writing to the *SRSMap* register. If the processor is operating in EIC interrupt mode, the binding of the interrupt to a specific shadow set is provided by the external interrupt controller, and is configured in an implementation-dependent way. Binding of an exception or non-vectored interrupt to a shadow set is done by writing to the ESS field of the *SRSCtl* register. When an exception or interrupt occurs, the value of $SRSCtl_{CSS}$ is copied to $SRSCtl_{PSS}$, and $SRSCtl_{CSS}$ is set to the value taken from the appropriate source. On an ERET, the value of $SRSCtl_{PSS}$ is copied back into $SRSCtl_{CSS}$ to restore the shadow set of the mode to which control returns.

Modes of Operation

The 24Kf 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 15 for more information on debug mode.

Memory Management Unit (MMU)

The 24Kf core contains a configurable Memory Management Unit (MMU) that is primarily responsible for converting virtual addresses to physical addresses and providing attribute information for different segments of memory.

Two types of MMUs are possible on the 24Kf core, selectable when the core is synthesized. Software can identify the type of MMU present by querying the MT field of the *Config* register.

1. Translation Lookaside Buffer (TLB)-style MMU. The basic TLB functionality is specified by the MIPS32 Privileged Resource Architecture (PRA). A TLB provides mapping and protection capability with per-page granularity. The 24Kf implementation allows a wide range of page sizes to be present simultaneously.
2. Fixed Mapping Translation (FMT)-style MMU. 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.

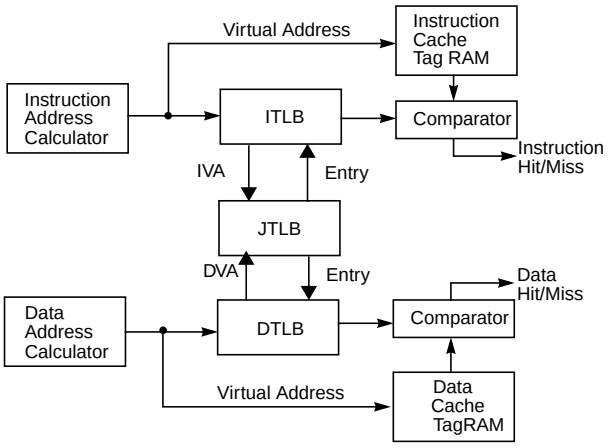
Translation Lookaside Buffer (TLB)

The basic TLB functionality is specified by the MIPS32 Privileged Resource Architecture. A TLB provides mapping and protection capability with per-page granularity. The 24Kf 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 24Kf core.

Figure 4 Address Translation During a Cache Access.



Joint TLB (JTLB)

The 24Kf core implements a fully associative JTLB containing 16, 32, or 64-dual-entries mapping up to 128 virtual pages to their corresponding physical addresses. The purpose of the TLB is to translate virtual addresses and their corresponding ASIDs into a physical memory address. The 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 a small 4-entry, fully associative TLB dedicated to performing translations for the instruction stream. 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 a small 8-entry, fully associative TLB dedicated to performing translations for loads and stores. 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.

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 bits of physical address, a valid bit, a lock bit, and an optional parity bit per way. The instruction data entry holds two instructions (64 bits) per way, as well as 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). The LRU replacement bits (6b) 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.

The 24Kf core 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.

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 bits of physical address, a valid bit, a lock bit, and an optional parity bit per way. The data entry holds 64 bits of data per way, with optional parity per byte. There is an additional array holding dirty bits and LRU replacement algorithm bits (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 both 32KB and 64KB data cache options, there is an implementation option to eliminate virtual aliasing. If this option is not selected, software must take care of any aliasing issues by using a page coloring scheme or some other mechanism.

In addition to instruction-cache locking, the 24Kf core also supports a data-cache locking mechanism identical to the instruction cache. Critical data segments are locked into the cache on a “per-line” basis. 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.

Cache Memory Configuration

The 24Kf 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 24Kf core instruction and data cache attributes.

Table 3 24Kf™ 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 24Kf 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.

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 24Kf core. This table lists characteristics about the core and the specific OCP functionality that is supported.

Table 4 OCP Performance Report

Core name	24Kf
Core Identity	TBD
Vendor Code	TBD
Core Code	0x93, visible in ProcessorID field of CP0 *PrID* register
Revision Code	Visible in Revision field of *PrID* register
Process dependent	No
Frequency range for this core	Synthesizable, so varies based on process, libraries, and implementation
Area	Synthesizable, so varies based on process, libraries, and implementation
Power Estimate	Synthesizable, so varies based on process, libraries, and implementation
Special reset requirements	No

Table 4 OCP Performance Report (Continued)

Number of Interfaces	1 OCP master, 2 OCP slave (DMA access for SPRAMs)
Interface Information: • Name • Type	 OCPMasterInterface 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 tags supported and use of those tags	Total of 12 tags: 10 tags for outstanding RD's, 1 tag for WR & 1 tag for SYNC
Connection ID and use of connection information	None
Use of sideband signals	None
Implementation restrictions	1. MReqInfo handled in a user-defined way. 2. MAddrSpace is used (2 bits) to indicate L2/L3 access. 3. Core clock is synchronous but a multiple of the OCP clock. Strobe inputs to the core control input and output registers to establish the core:bus clock ratio.
Interface Information: • Name • Type	 OCPSlaveInterface Slave
Slave OCP Interfaces (DMA interface to scratchpad)	
Operations accepted	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 is not supported.
Maximum number of operations outstanding	2 outstanding operations which includes both RD & WR.
Burst support	Burst access is not supported
High level flow control	Back pressure from slave on data and command accept. Slave assumes no back pressure from the master.
Number of tags supported and use of those tags	Total of 8 tags. Any tag number can be used for read and write operation.
Connection ID and use of connection information	None
Use of sideband signals	None

Table 4 OCP Performance Report (Continued)

Implementation restrictions	The slave interface operates at the same clock ratio as that of the master OCP interface.

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.

Write Through

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.

Write Back

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.

Uncached Accelerated

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 and double word stores only.

Gathering of uncached accelerated stores will start on cache-line aligned addresses, i.e. 32 byte aligned addresses. Once an uncached accelerated store starts 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. An uncached accelerated store that does not merge and does not go to an aligned address will be treated as a regular uncached store.

SimpleBE Mode

To aid in attaching the 24Kf 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 reads 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.

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 a different frequency. The core does not contain an explicit OCP input clock; all flops are actually controlled by *SI_ClkIn*. Additional inputs control when inputs should be sampled and outputs should be driven
- 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*.

Hardware Reset

Unlike previous MIPS cores, a 24Kf core only has a single reset input. Historically, cold reset was used to reset a PLL. In synthesizable cores without a PLL, the two inputs were ORed together internally and then treated identically (except for a *Status* bit indicating which reset was seen). The 24Kf interface has removed the second reset type and only includes the *SI_Reset* pin.

The *SI_Reset* input is used to initialize critical hardware state. It can be asserted either synchronously or asynchronously to the core clock, *SI_ClkIn*, and will trigger a Reset exception. The reset signal is active high, and must be asserted for a minimum of 5 *SI_ClkIn* cycles. The falling edge triggers the Reset exception. The reset signal must be asserted at power-on or whenever hardware initialization of the core is desired.

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.

Power Management

The 24Kf 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 24Kf 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 bit is available externally via the *SI_RP* signal. 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, $Status_{EXL}$, $Status_{ERL}$, 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 24Kf core is in a low power state. Depending on what type of exception is taken, one of these three bits will be asserted and reflected on the *SI_EXL*, *SI_ERL*, or *EJ_DebugM* outputs. The external agent can look at these signals and determine whether to leave the low power state to service the exception.

The following 4 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* signal represents the state of the RP bit (27) in the CP0 *Status* register.
- The *SI_EXL* signal represents the state of the EXL bit (1) in the CP0 *Status* register.
- The *SI_ERL* signal represents the state of the ERL bit (2) in the CP0 *Status* register.
- The *EJ_DebugM* signal represents the state of the DM bit (30) in the CP0 *Debug* register.

The coprocessor interface is extensible and standardized on MIPS cores, allowing design reuse. The 24Kf core supports a

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_NMI*, *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 24Kf core asserts the *SI_Sleep* signal, which is part of the system interface, whenever the WAIT instruction is executed. The assertion of *SI_Sleep* indicates that the clock has stopped and the 24Kf core is waiting for an interrupt.

Local clock gating

A majority of the power consumed by the 24Kf 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.

CorExtend™ User Defined Instruction Extensions

The optional CorExtend User Defined Instruction (UDI) block enables the implementation of a small number of application-specific instructions that are tightly coupled to the core’s integer execution unit.

The interface to the CorExtend block is similar to the Multiply-Divide Unit, allowing non-blocking, pipelined multi-cycle operations. A portion of the hooks into the MDU control logic and also allows the HI/LO accumulation registers to be used by the CorExtend block.

CorExtend instructions may operate on a general-purpose register, immediate data specified by the instruction word, or local state stored within the UDI block. The destination may be a general-purpose register, HI/LO, or local UDI state. The operation may complete in one cycle or multiple cycles, if desired.

Coprocessor 2 interface

The 24Kf core can be configured to have an interface for an on-chip coprocessor. The interface allows the coprocessor to be tightly coupled to the processor core, allowing high performance solutions, like integrating a graphics accelerator or custom DSP.

subset of the full coprocessor interface standard: single issue, 64 bit in-order data transfers.

The coprocessor interface is designed to ease integration with customer IP. The interface allows high-performance communication between the core and coprocessor. There are no late or critical timing signals on the interface.

Data Scratchpad RAM (DSPRAM)

The 24Kf core can be configured to include an optional Data scratchpad RAM independent of the data cache configuration. A separate OCP slave interface allows a DMA master to access the data scratchpad RAM.

To demonstrate use of the scratchpad capability, MIPS provides a default design that includes one contiguous 8 KB RAM with cache-like access. A DSPRAM hit supersedes a data cache hit. DSPRAM is indexed by virtual address. The hit information is based on the physical address in the base register. DSPRAM can be mapped to either cacheable or non-cacheable address space. A sophisticated arbitration scheme and instruction slip in the pipe prevents unnecessary stalls.

Only store instructions which are guaranteed to complete and hit in the DSPRAM, arbitrate for the RAM. The DMA access priority with respect to the core access is determined by the input pin *SI_DMA_Priority*. The DSPRAM interface supports multi-cycle access to the RAM array to accommodate slow devices or larger memory sizes. The interface allows addressing of DSPRAM sizes up to 1MB. The interface also supports 64-bit wide data access and provides a mechanism to back-stall the core pipeline.

Instruction Scratchpad RAM (ISPRAM)

The 24Kf core can be configured to include an optional instruction scratchpad RAM independent of the instruction cache configuration. A separate OCP slave interface allows a DMA master to access the instruction scratchpad RAM.

To demonstrate use of the scratchpad capability, MIPS provides a default design that includes one contiguous 8KB RAM with cache like access. ISPRAM hit supersedes instruction cache hit. ISPRAM is indexed by virtual address. The hit information is based on the physical address in the base register. ISPRAM can be mapped to either cacheable or non-cacheable address space.

The DMA access priority with respect to the core access is determined by the input pin *SI_IDMA_Priority*. The ISPRAM interface supports multi-cycle access to the RAM array to accommodate slow devices or larger memory sizes. The interface allows addressing of ISPRAM sizes up to 1MB. The interface also supports 64-bit wide data access and provides a mechanism to back-stall the core pipeline.

EJTAG Debug Support

The 24Kf 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 24Kf 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 24Kf 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.

Debug Registers

Three debug registers (*DEBUG*, *DEPC*, and *DESAVE*) have been added to the MIPS Coprocessor 0 (CP0) register set. The *DEBUG* register shows the cause of the debug exception and is used for setting up single-step operations. The *DEPC*, or Debug Exception Program Counter, register holds the address on which the debug exception was taken. This is used to resume program execution after the debug operation finishes. Finally, the *DESAVE*, or Debug Exception Save, register enables the saving of general-purpose registers used during execution of the debug exception handler.

To exit debug mode, a Debug Exception Return (DERET) instruction is executed. When this instruction is executed, the system exits debug mode, allowing the normal execution of application and system code to resume.

EJTAG Hardware Breakpoints

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 24Kf core: Instruction breakpoints and Data breakpoints.

During synthesis, the 24Kf core can be configured with or without hardware breakpoints. The following breakpoint options are supported:

- 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/store transactions. Breakpoints are set on virtual address and ASID values, similar to the Instruction breakpoint. Data breakpoints can be set on a load, a store, or both. 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.

MIPS Trace

The 24Kf 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 8 MB; 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.

Testability

Testability for production testing of the core is supported through the use of internal scan and memory BIST.

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

Memory BIST for the cache arrays, scratchpad memories and on-chip trace memory is optional, but can be implemented either through the use of integrated BIST features provided with the core, or inserted with an industry-standard memory BIST CAD tool.

Integrated Memory BIST

The core provides an integrated memory BIST solution for testing the internal cache SRAMs, scratchpad RAMs and on-chip trace RAM, using BIST controllers and logic tightly coupled to the cache subsystem. Several parameters associated with the integrated BIST controllers are configurable, including the algorithm (March C+ or IFA-13).

User-specified Memory BIST

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 24Kf 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 24K 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
Integer register file sets	1, 2, or 4	$SRSCtl_{HSS}$
Integer register file implementation style	Flops or generator	N/A
Memory Management Type	TLB or FMT	$Config_{MT}$
TLB Size	16, 32, or 64 dual entries	$Config1_{MMUSize}$
TLB data array implementation style	Flops or generator	N/A
* These bits indicate the presence of an external block. Bits will not be set if interface is present, but block is not.		

Table 5 Build-time Configuration Options

Option	Choices	Software Visibility
Number of outstanding data cache misses	4 or 8	N/A
Number of outstanding Loads	4 or 9	N/A
Instruction hardware breakpoints	0 or 4	DCR_{IB}, IBS_{BCN}
Data hardware breakpoints	0 or 2	DCR_{DB}, DBS_{BCN}
MIPS Trace support	Present or not	$Config3_{TL}$
MIPS Trace memory location	On-core, off-chip or both	$TCBCONFIG_{OnT}$ $TCBCONFIG_{OfT}$
MIPS Trace on-chip memory size	256B - 8MB	$TCBCONFIG_{SZ}$
MIPS Trace triggers	0 - 8	$TCBCONFIG_{TRIG}$
MIPS Trace source field bits in trace word	0 or 2	$TCBCONTROLB_{TWSrcWidth}$
CorExtend interface (Pro only)	Present or not	$Config_{UDI}$*
FPU clock ratio relative to integer core	1:1 or 1:2	$Config7_{FPR}$
Coprocessor2 interface	Present or not	$Config1_{C2}$*
Instruction ScratchPad RAM interface	Present or not	$Config_{ISP}$*
Data ScratchPad RAM interface	Present or not	$Config_{DSP}$*
I-cache size	0, 8, 16, 32, or 64 KB	$Config1_{IL}$, $Config1_{IS}$
D-cache size	0, 8, 16, 32, or 64 KB	$Config1_{DL}$, $Config1_{DS}$
D-cache hardware aliasing support	Present or not (for 32KB only)	$Config7_{AR}$
Cache parity	Present or not	$ErrCtl_{PE}$
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
PrID Company Option	0x0-0x7f	$PrID_{CompanyOption}$

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

Document Revision History

Change bars (vertical lines) in the margins of this document indicate significant changes in the document since its last release. Change bars are removed for changes that are more than one revision old. This document may refer to

Architecture specifications (for example, instruction set descriptions and EJTAG register definitions), and change bars in these sections indicate changes since the previous version of the relevant Architecture document.

Revision	Date	Description
00.90	July 17, 2003	Initial version
00.91	July 31, 2003	Updates based on early feedback
00.92	August 8, 2003	Preliminary external release
00.93	September 12, 2003	• Described several updates to the OCP interface (Thread model, SYNC behavior, MReqInfo, MAddrSpace) • Added burst order section • Added core-to-bus clocking relationship waveform and description
00.94	September 29, 2003	• Removed I/O SError. Use interrupts instead for async bus errors • EJ_DINT type should be A • Added 4 L2 performance counter signals to I/O list • Added sync pattern table to *SI_OCPSync* in external interface section.
00.95	December 3, 2003	• Added Uncached Accelerated flush condition on store to a different 32B region • Trademark updates
01.00	December 10, 2003	• Updated template
01.01	January 27, 2004	• Noted option of running FPU at same clock rate as integer core. • Changed names of BIST-related interface signals; they now begin with *MB_*. • Clarified that an OCP write data phase starts one cycle after the command phase is accepted.
01.02	February 11, 2004	• Clarified number of possible hardware breakpoints. • Fixed document template
02.00	March 5, 2004	• Removed unused *SI_OCPClkIn* input pin. • Renamed *gscan{in,out}_x* pins to *gscan{in,out}[n-1:0].* • Increased number of parity bits in I-cache data array, if parity is enabled. • Updated MDU latencies
02.01	May 26, 2004	• Added new static inputs to control selection of integrated memory BIST algorithm. • Added Table 5 summarizing build-time configuration options.
03.00	September 15, 2004	• Described SPRAM and COP2 interfaces. • Added breakpoint status output pins. • Updated OCP interface to OCP version 2.1, with use of TagID fields.
03.01	November 10, 2004	• OCP Sync waveform clarified. • Other minor improvements.
03.02	March 15, 2005	• Described the MIPS Trace interface. • Other minor updates
03.03	April 26, 2005	• Described the ISPRAM interface. • Other minor updates
03.04	June 30, 2005	• Various enhancement updates
03.05	December 14, 2005	• 8KB cache support • Clock-ratio resynchronization • Pin changes for OCP compliance • New scan control pin

Revision	Date	Description
03.06	June 29, 2006	• Add configuration option for trace word source width • 8 outstanding load misses • 1:10 clock ratio support • Hardware aliasing support for 64KB D-cache • New pins to support MIPS SBL2
03.07	December 19, 2006	• Added signals to override exception base when $Status_{BEV}$ is 1 • Added new L2 inputs: indication that L2 is in bypass mode and performance counter for ECC events • Added new input to indicate that downstream system can handle external SYNC transactions
03.08	January 22, 2007	• Updated document template to nDb1.03
03.10	November 1, 2007	• Updated to consistent names for TagLo and DataLo registers • Added configuration options for number of data cache misses, load misses, and PrID company option field • Added new input *gscanramenable* to qualify whether *gscanramwr* affects the RAM strobes during scan mode • Added CP0 *UserLocal* register with conditional access via RDHWR instruction
04.00	December 19, 2008	• Removed sections of detailed information that was replicated in other core documentation: Pin Lists, Instruction List, TLB operation description • Minor typographical updates to the Architecture Overview bullet list

Template: nDb1.03, Built with tags: 2B