MIPS TECHNOLOGIES

MIPS Verified™

MIPS32® 1074K™ Coherent Processing System Datasheet

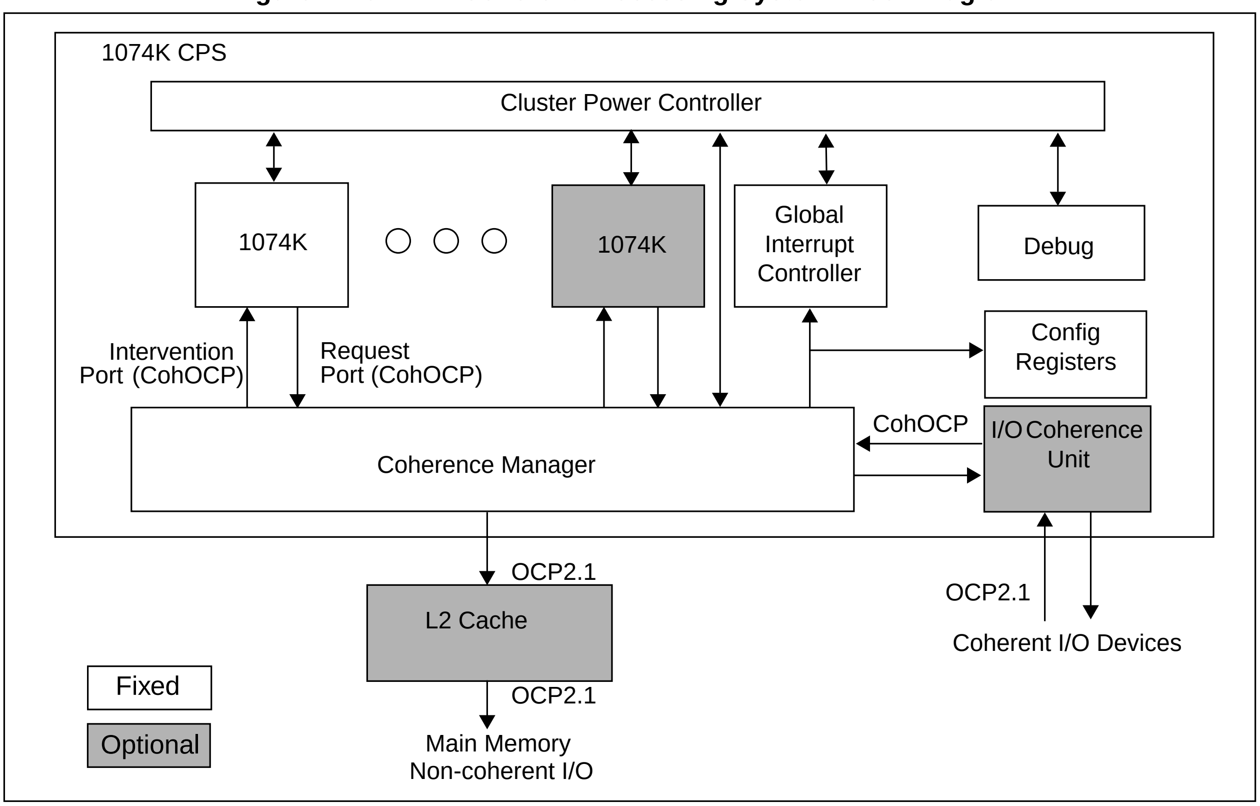
June 03, 2011

The MIPS32® 1074K™ Coherent Processing System (CPS) from MIPS Technologies is a high-performance coherent multiprocessor cluster of one to six MIPS32® 1074K™ CPUs and an optional coherent I/O port. The 1074K CPU features an out-of-order, superscalar execution pipeline, Digital Signal Processing Application Specific Extensions and is based on the proven MIPS32® 74K™ CPU. Multi-CPU coherence is managed in hardware by a Coherence Manager (CM), using extensions to the OCP-IP protocol and an OCP-based intervention port on each CPU. Figure 1 shows a block diagram of the Cluster.

The 1074K CPS Cluster can optionally be connected to the MIPS® SOC-it® L2 Cache Controller. When connected to the L2 cache, 256-bit datapaths are utilized to take advantage of the full bandwidth available from the L2 cache design. When the L2 cache is not present, the interface is restricted to 64-bit datapaths so that it matches the memory interface of existing non-coherent CPU products from MIPS Technologies. The L2 or memory interface uses non-coherent OCP protocols.

The 1074K Cluster includes other modules that handle common system-level functions. The optional I/O Coherence Unit supports HW I/O coherence by bridging a non-coherent OCP I/O interconnect to the CM and handling ordering requirements. The Global Interrupt Controller handles the distribution of interrupts between and among the CPUs in the Cluster. Additionally, the Cluster Power Controller can gate off the clocks and power to idle cores in order to save power.

Figure 1 1074K™ Coherent Processing System Block Diagram



1074K™ CPS Features

- 1 - 4 coherent MIPS32 1074K™ CPU cores
- Power Controller to shut down idle CPU cores
- Optional hardware I/O coherence port
- MESI coherence states in L1 data caches
- Supports cache to cache data transfers
- Speculative memory reads to reduce latency
- Out-of-order data return
- Optimized 256b interface to SOC-it L2 cache controller
- Separate clock ratios on Memory, L2, and IOCU OCP ports.
- Core-to-CM clock ratios of 1:1, 5:4, 3:2 and 2:1. All Cores operate at same frequency.

Supported Configurations

Refer to Section “Build-Time Configuration Options” for a complete list of configuration options.

Software Support

Modifications to the Linux operating system are available from MIPS Technologies that support SMP and SMVP build options.

The product is also supported by third party software releases as part of the on-going development of the ecosystem by MIPS.

Coherence Protocol

The coherence protocol is characterized by the following properties:

- Writeback cache - Uses a writeback cache to ensure high performance (only modified lines are written to memory).
- Cache-line based - Coherence and ownership are maintained per cache line.
- Snoopy protocol - Each CPU snoops the stream of transactions and updates its cache state accordingly.
- Invalidate - A line is invalidated in the cache (possibly with a writeback to memory) when a store from another processor is seen.

Cache States

MESI cache states are used in the L1 data cache. A cache line can exist in one of the following states:

- *Modified:* The line has been modified. This is the only valid copy of the data in the system. No other L1 data caches will have a copy of this line in a valid state.
- *Exclusive*: No other L1 data caches have a copy of this line in a valid state. This CPU ‘owns’ the line and can modify it. The data for this line is consistent with memory.
- *Shared*: Cache line potentially exists in more than one L1 data cache. Read-only in all caches.
- *Invalid*: The line is not valid in this cache.

Ordering

Weak ordering is used within the Cluster; that is, reads and writes to different addresses can occur out of program order unless explicitly ordered through the use of a SYNC instruction.

Coherent OCP Extensions

The 1074K Cluster makes use of extensions to the OCP protocol to include information about coherent traffic. These extensions include:

- Coherent Request Types as shown in Table 1.
- Coherent state slave responses. On an intervention, indicates what MESI state the CPU had the line in. On a request, indicates what MESI state the line should be installed in.

- Dataless response - allows a dataless Upgrade transaction and is also a common response on the intervention port.

Table 1 Coherent Requests

Request	Description
(Legacy)	Non-coherent Reads/Writes.
CohReadShared	Request for a line in a Shared state (load miss). Data can remain in other data caches in the Shared state. Line may be upgraded to Exclusive if there are no other Sharers.
CohReadOwn	Request for Exclusive ownership of the line (store miss). Following this request, the line cannot remain in any other L1 data caches.
CohUpgrade	Request to upgrade line from shared to exclusive (store hit on shared). Following this request, the line will not remain in any other data caches. The upgrade can be done without transferring data.
CohWriteBack	Writeback of data (eviction). CPU is writing data back to memory. This request is not sent to other CPUs, as it is known they will not have valid copies of the line.
CohInvalidate	Invalidate the line in all caches (PREF Prepare for Store, CACHE HitInvalidate).
CohCopyBack	Write dirty data back to memory from any cache (CACHE HitWrite-Back).
CohCopyBackInv	Write dirty data back to memory from any cache and invalidate in all (CACHE HitWriteBackInvalidate).
CohWriteInvalidate	Write data to memory (I/O write). This write data is replacing the existing data. A partial line write will be merged with data from the caches and memory, and a full line write will invalidate the line in the caches and overwrite memory.
CohReadSharedDiscard	Read data from coherence domain (I/O read). Gets the most up-to-date data, but because the data is not going to a coherent cache, the state of the line does not need to change.
CohReadSharedAlways	Request line in Shared state only - cannot be installed as Exclusive or Modified. (Not used in this system - could be used by a caching I/O agent or for a coherent I-Cache fetch.)

Intervention Port

Devices with coherent caches include a second OCP port referred to as the *intervention port*. Coherency is maintained by sending all coherent requests to all devices via the intervention port. This includes requests by the device itself; referred to as a *self-intervention*, this provides a mechanism for the agent to determine the global ordering of its request with respect to requests from other agents.

Each device updates its cache state for the intervention and responds when the state transition has completed. The previous state of the line is indicated in the response. If a read type intervention hits on a line that the CPU has in a Modified or Exclusive state, the CPU returns the cache line with its response.

A cacheless device, such as the IOCU, does not require an intervention port.

MIPS32 1074K CPU

The MIPS32® 1074K™ CPU from MIPS Technologies is a high-performance, low-power, 32-bit RISC Superscalar CPU designed for custom system-on-chip (SoC) applications. The CPU 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 SoC designs, allowing developers to focus their attention on end-user products.

The 1074K CPU implements the MIPS32 Release 2 Architecture in a superscalar, out-of-order execution pipeline. The deeply pipelined CPU can support a peak issue and graduation rate of 2 instructions per cycle. The 1074K CPU also implements the MIPS DSP ASE - Revision 2.0, which provides support for signal processing instructions, and includes support for the MIPS16e™ ASE and the 32-bit privileged resource architecture. This architecture is supported by a wide range of industry-standard tools and development systems.

The 1074K CPU has a Level-1 (L1) Instruction Cache, which is configurable at 0, 16, 32, or 64 KB in size. It is organized as 4-way set associative. Up to four instruction cache misses can be outstanding. The instruction cache is virtually indexed and physically tagged to make the data access independent of virtual to physical address translation. Instruction cache tag and data access are staggered across 2 cycles, with up to 4 instructions fetched per cycle.

The superscalar 1074K CPU can dispatch up to 2 instructions per cycle into one of the arithmetic-logic unit (ALU) or address generation (AGEN) pipes. The AGEN pipe executes all Load/Store and Control Transfer instructions while the ALU pipe executes all other instructions. Instructions are issued and executed out-of-order; however, the results are buffered and the architectural state of up to 2 instructions per cycle is updated in program order.

The L1 Data Cache is configurable at 16, 32, or 64 KB in size. It is organized as 4-way set associative. Data cache misses are non-blocking and up to seven may be outstanding. The data cache is virtually indexed and physically tagged to make the data access independent of virtual-to-physical address translation. The tag array also has a virtual address portion, which is used to compare against the virtual address being accessed and generate a data cache hit prediction. This virtual address hit prediction is always backed up by a comparison of the translated physical address against the physical tag. To achieve high frequencies while using commercially available SRAM generators, the cache access and hit determination is spread across three pipeline stages, dedicating an entire cycle for the SRAM access.

The synthesizable 1074K CPU includes a high performance Multiply/Divide Unit (MDU). The MDU is fully pipelined to support a single cycle repeat rate for 32×32 MAC instructions. The CorExtend® block can utilize the accumulator registers in the MDU block, allowing specialized functions to be efficiently implemented.

The MIPS DSP ASE - Revision 2.0 provides support for a number of powerful data processing operations. There are instructions for fractional arithmetic (Q15/Q31) and for saturating arithmetic. Additionally, for smaller data sizes, SIMD operations are supported, allowing 2×16 bit or 4×8 bit operations to occur simultaneously. Another feature of the ASE is the inclusion of additional HI/LO accumulator registers to improve the parallelization of independent accumulation routines. All 32-bit operand arithmetic DSP instructions (except multiply) are executed in the ALU pipe while the 64-bit operand arithmetic and multiply class DSP instructions are executed in the MDU pipe.

The 1074Kf™ CPU features an optional IEEE 754 compliant Floating Point Unit (FPU). The FPU supports both single- and double-precision instructions. The main pipe is connected to the FPU through an in-order instruction queue, and up to 2 instructions can be written into this queue. The FPU can read up to 2 instructions from this queue and issue into its execution pipes.

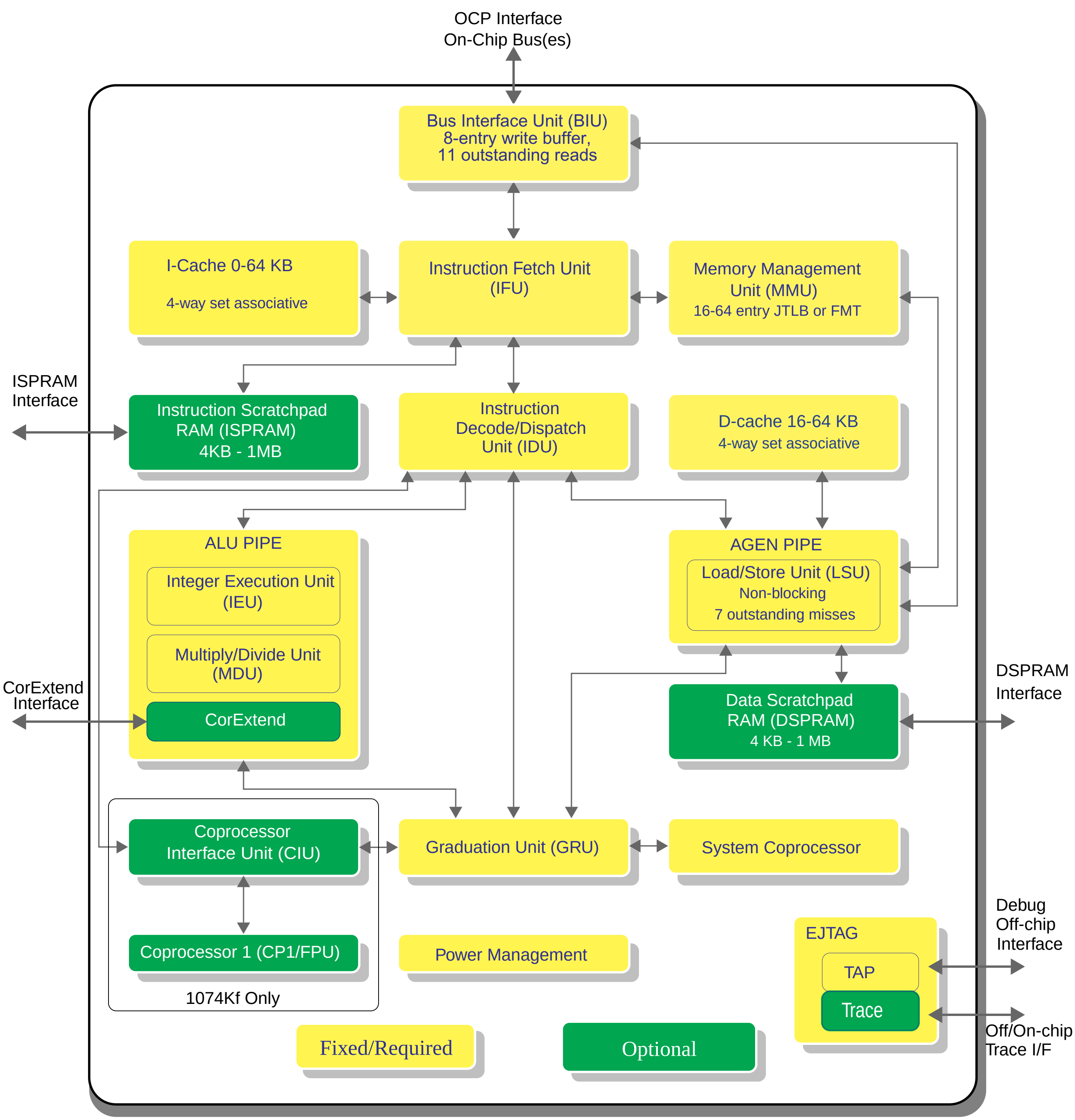
The Bus Interface Unit (BIU) implements the Open Core Protocol (OCP), which has been developed to address the needs of SoC designers. This implementation features 64-bit read and write data buses to efficiently transfer data to and from the L1 caches. The BIU also supports a variety of CPU/bus clock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

Optional support for external Instruction and Data Scratchpad RAM arrays, with reference design supporting DMA interfaces for loading the arrays.

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. Additionally, real-time tracing of instruction program counter, data address and data values can be supported.

Figure 2 shows a block diagram of the 1074K CPU.

Figure 2 1074K™ CPU Block Diagram



1074K™ CPU Features

- 14-stage ALU and 15-stage AGEN pipelines
  - 12-stage ALU fetch and execution pipe
  - 13-stage AGEN fetch and execution pipe
  - Common 2-stage graduation pipe
- 32-bit address paths
- 128-bit data path for instruction cache and data path for data cache
- 64-bit data paths to external interface
- MIPS32 Release2 Instruction Set and Privileged Resource Architecture
- MIPS16e Code Compression
- MIPS DSP ASE - Revision 2.0
  - 3 additional pairs of accumulator registers
  - Fractional data types (Q15, Q31)
  - Saturating arithmetic
  - SIMD instructions operate on 2×16 bit or 4×8 bit simultaneously
- Instruction Fetch Unit
  - 4-instruction fetch per cycle
  - 8-entry Return Prediction Stack
  - Combined Majority Branch Predictor using three 256-entry Branch History Tables (BHT).
  - 64-entry (4-way associative) jump register cache to predict target for indirect jumps.
  - Hardware prefetching of the next 1 or 2 sequential cache lines on a miss. Number of prefetched lines (0, 1, or 2) controllable via configuration bits.
- Dual Out-of-Order Instruction Issue
  - Separate ALU and AGEN pipes
  - AGEN pipe executes load/store and control transfer instructions
  - ALU pipe executes all other instructions
  - 32 (18 ALU, 14 AGEN) completion buffers hold execution results until instructions are graduated in program order
- Programmable Memory Management Unit
  - 16/32/48/64 dual-entry, dual-ported TLB shared by Instruction and Data MMU
  - 4-entry ITLB (4KB, 16KB page size)
  - 4K, 16K, 64K, 256K, 1M, 4M, 16M, 64M, 256M byte page size supported in JTLB
  - Optional simple Fixed Mapping Translation (FMT) mechanism
- Programmable L1 Cache Sizes
  - Individually configurable instruction and data caches
  - Instruction Cache sizes of 0/16/32/64 KB
  - Data Cache sizes of 16/32/64 KB
  - 4-way set associative
  - 32-byte cache line size
  - Virtually indexed, physically tagged
  - Cache line locking support
  - Up to 4 outstanding I-cache misses
  - Virtual tag based hit prediction in data cache
  - Up to 7 unique outstanding D-cache line misses and 9 total load misses
  - Writeback support in data cache
  - Non-blocking data cache prefetches
  - Optional parity support

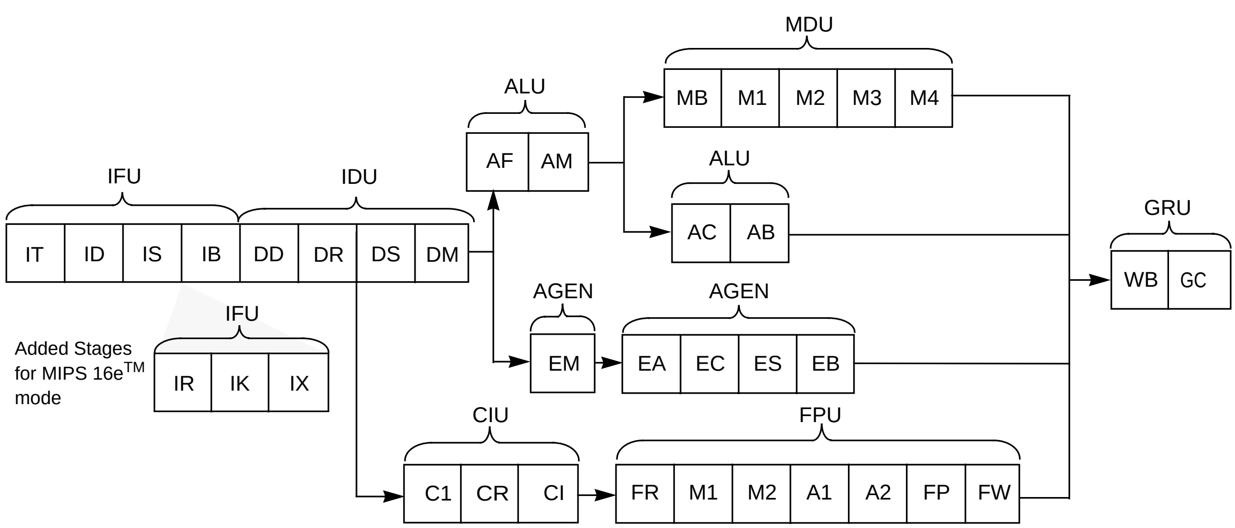
- Scratchpad RAM support
  - Independent Instruction and Data Scratch-pad RAMs
  - Scratchpad RAM size from 4KB to 1MB
  - Independent of cache configuration
  - 64-bit OCP interfaces for external DMA
  - OCP port runs at the same CPU/bus clock ratio as the BIU interface
- Front-side L2 support
  - Support for inline L2 cache.
  - L2 cache can be configured to be bypass-able.
- Bus Interface
  - OCP version 2.1 interface with 32-bit address and 64-bit data
  - OCP version 2.1 interface runs at CPU/bus clock ratios of 1:1, 3:2 or 5:4 via a separate synchronous bus clock
  - Clock ratio can be changed dynamically
  - Burst size of four, 64-bit beats
  - 8-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 32×32 multiply per clock
  - 7-cycle multiply latency
  - Iterative SRT divide algorithm. Minimum 10 and maximum 50 clock latency (dividend (*rs*) sign extension-dependent)
- Power Control
- CorExtend® User Defined Instruction Set Extensions
  - Allows user to define and add instructions to the CPU at build time
  - Maintains full MIPS32® compatibility
  - Includes access to GPRs and Accumulator registers
  - Instruction operand format (source/destination registers) and latency specified by a programmable template
  - Allows latencies of 3, 5, or >5 cycles when destination is a GPR/Accumulator. Single-cycle latency is allowed when there is no modification to the architectural state of the 1074K CPU.
  - Allows in-order issue of CorExtend instructions that do not modify the 1074K CPU architectural state
  - Supported by industry-standard development tools
- Floating Point Unit (FPU)
  - IEEE-754 compliant Floating Point Unit
  - Compliant to MIPS 64-bit FPU standards
  - Supports single- and double-precision data types
  - Optionally runs at 1:1, 3:2 or 2:1 CPU/FPU clock ratio
  - Separate in-order dual-issue pipeline decoupled from integer pipeline
  - Asymmetric dual-issue. Load/Store and MoveToCP1/MoveFromCP1 instructions execute in one pipe, while all other instructions execute in the other pipe.
- Relocatable Reset Vector
  - Support for user (pin) programmable reset vector in a multi-CPU environment.
  - Minimum frequency: 0 MHz

  - Power-down mode (triggered by WAIT instruction)
  - Support for software-controlled clock divider
  - Support for top-level, block-level, fine-grained and data cache clock gating
- EJTAG Debug 5.0
  - Support for single-stepping
  - Instruction address and data address/value breakpoints
  - TAP controller is chainable for multi-CPU debug
  - Cross-CPU breakpoint support
  - Relocatable debug handler
- MIPS Trace
  - PC, data address, data value, performance counter value, processor pipeline inefficiency tracing with trace compression
  - Support for on-chip and off-chip trace memory
  - PDtrace version 6 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

Pipeline Flow

The 1074K CPU implements a 14/15-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 optimal area and power numbers. Figure 3 shows the 1074K CPU pipeline.

Figure 3 1074K™ CPU Pipeline



Instruction Fetch Unit (IFU)

IT: Instruction Tag Read

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

ID: Instruction Data Read

- I-cache data array accesses
- Tag compare, Detect I-cache hit

IS: Instruction Select

- Way select
- Target calculation start

IB: Instruction Buffer

- Instruction Buffer write
- Target calculation done

IR: Instruction Recode

- MIPS16e instruction recode

IK: Instruction Decode

- MIPS16e branch decode
- MIPS16e target validate

IX: Instruction Expansion

- MIPS16e macro expansion

Instruction Decode and Dispatch Unit (IDU)

DD: Decode

- Access Rename Map, get source register availability to resolve source dependency
- Decode instructions and assign pipe and instruction identifier
- Check execution resources

DR: Rename

- Update Rename Map at destination register to resolve output dependency
- Send instruction information to Graduation Unit (GRU) and Coprocessor Interface Unit (CIU)
- Send instruction to Decode and Dispatch Queue (DDQ)

DS: Select for Dispatch

- Check for operand and resource availability and mark valid instructions as ready for dispatch
- Select 1 out of 8 (6-entry DDQ + 2 staging registers) ready instructions in each ALU and AGEN pipe independently

DM: Instruction Mux

- Read out the selected instruction from the previous stage and update the selection information
- Generate controls for source-operand bypass mux
- ALU pipe will start premuxing operands based on the selected instruction.
- AGEN pipe will starting reading source operands from Register File and Completion Buffers.

Integer Execution Unit (IEU)

AF: ALU Register file Read

- AGEN pipe will complete reading source operands from Register File and Completion Buffers.

AM: ALU Operand Mux

- Select source operands and set up for execution

AC: ALU Compute

- Integer Execution start. Logical operations, some shift and arithmetic operations complete and bypass the results.

AB: ALU Results Bypass

- Complete Integer Execution and bypass results

EM: AGEN Operand Mux

- Select source operands for Load/Store index computation and set up for execution

EA: AGEN Effective Address Compute

- Compute Effective Address for Load/Store instructions
- Select source operands for Store data and Branch/Jump instructions
- Start JTLB access

Load/Store Unit (LSU)

EC: Cache Access

- Access D-cache and D-tag arrays. Read Virtual and Physical tags along with data
- Continue JTLB access
- AGEN pipe resolves conditional branch and Jump instruction

ES: D-Cache way select

- Select D-cache way based on Virtual tag match with Effective Address
- Start Physical Tag compare with JTLB data
- AGEN pipe redirects IFU in the event of branch mis-predict or register indirect jump

EB: Cache Data Bypass

- Complete data selection and align load data
- Bypass results (selected data) to both AGEN and ALU pipes
- Validate Virtual tag match with Physical tag comparison

Coprocessor Interface Unit (CIU)

C1: Coprocessor Write Pointer Adjust

- Adjust InOrder Instruction Queue write pointer

CR: Coprocessor Instruction Read

- Read instruction from InOrder Instruction Queue

CI: Coprocessor Instruction Issue

- InOrder Instruction dispatch to Coprocessor1

Graduation Unit (GRU)

WB: Writeback

- Consolidate and propagate D-cache hit/miss information
- Write execution results into ALU and AGEN completion buffers
- Update all GRU structures to indicate instruction completion
- Oldest 2 entries that have completed execution are identified and their addresses are obtained to read the completion buffers and associated information to graduate 2 instructions

GC: Graduation Complete

- Two instructions are graduated and Register File data is obtained for update
- Load misses are graduated with their destination marked unavailable
- Load misses and Stores (hits and misses) are activated in the LSU for further processing

1074K™ CPU Logic Blocks

The 1074K CPU consists of the logic blocks defined in the following subsections (see Figure 2).

Instruction Fetch Unit (IFU)

The Instruction Fetch Unit (IFU) is responsible for fetching instructions from the Instruction Cache, Instruction Scratchpad or Memory and feeding them to the execution units. The IFU can fetch up to 4 instructions at a time from an aligned PC. The IFU uses majority branch prediction based on a gshare predictor. There are three, 256-entry Branch History Tables that are indexed by different combinations of instruction PC and Global History. The majority of these 3 predictions are used to determine the predicted direction of a conditional branch. The IFU also has an 8-entry Return Prediction Stack to predict subroutine return addresses and a 64-entry jump indirect target address predictor. A 4-way, 16-entry/way buffer learns and predicts the target addresses for indirect jumps.

The IFU has a 4-entry microTLB which is used to translate the virtual address into the physical address. This translated physical address is used to compare against tags in the instruction cache to determine a hit.

The functionality of the IFU is spread across 4 CPU- visible pipeline stages in MIPS32 mode. Additional stages are in the shadow of execution and do not account for the minimum recirculation path in the event of a PC redirection. In the MIPS16e™ mode, the IFU takes an additional 3 stages to recode and expand the compressed code.

There is a 12-entry Instruction Buffer to decouple the instruction fetch from execution. Up to 4 instructions can be written into this buffer, but a maximum of 2 instructions can be read from this buffer by the IDU.

The IFU can also be configured to allow for hardware prefetching of cache lines on a miss. When an instruction cache miss is detected, the IFU can prefetch the next 0, 1, or 2 lines (besides the missed line) to reduce average miss latency. The number of prefetched lines can be configured by software via *Config7* register settings.

MIPS16e™ Application-Specific Extension

The 1074K CPU includes support for the MIPS16e ASE. This ASE improves code density by using 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 set-up/teardown for efficient subroutine entry/exit.

Instruction Decode and Dispatch Unit (IDU)

This unit is responsible for receiving instructions from the IFU and dispatching them to the execution units when their operands and required resources are available. Up to two instructions can be received in-order from the IFU per cycle. The instructions are assigned an instruction ID and a completion buffer ID, which identifies a buffer location to hold results temporarily. The instruction is also renamed by looking up in a Rename Map, and the source registers are replaced (if necessary) by completion buffer IDs of producer instructions, so that operands may be bypassed as soon as possible.

Renamed instructions are assigned to one of two pipes (ALU or AGEN) and written into the Decode and Dispatch Queue (DDQ). The oldest instruction that has all the operands ready and meets all resource requirements is dispatched independently to the corresponding pipe. It is possible that instructions will be dispatched out-of-order relative to program order. Dispatched instructions do not stall in the pipe and write the results into the completion buffer.

The IDU also keeps track of the progress of the instruction through the pipe, updating the availability of operands in the Rename Map and in all dependent instructions in the DDQ.

The IDU also writes the instruction ID, completion buffer ID, and related information into structures in the Graduation Unit (GRU). The GRU reads instructions and corresponding results from the completion buffer, graduates the instructions, and updates the architectural state of the machine.

Execution Units

The 1074K CPU execution unit implements two pipes: an ALU pipe for handling all arithmetic operations (logical, shift, add, subtract) and an AGEN pipe for handling all load/store operations and control transfer instructions and an autonomous multiply/divide unit (MDU) and CorExtend unit. The MDU and CorExtend pipe share control logic with the ALU pipe. There is a 31-entry, 32-bit register file that is shared by both the pipes. There is a separate 18-entry, 64-bit completion buffer for the ALU pipe, and a 14-entry, 32 bit completion buffer for the AGEN pipe.

ALU Pipe

The ALU pipe spans four stages as follows:

- The first two stages (AF, AM) of the ALU pipe are used to prepare operands, read the register file and completion buffer, and mux select all operands for the arithmetic operation.

- Execution is performed in the AC stage, which includes:
  - Arithmetic Logic Unit (ALU) for performing arithmetic and bitwise logical operations
  - Shifter
  - Leading Zero/One detect unit for implementing the CLZ and CLO instructions
  - All logical operations, some arithmetic operations {ADD (rt=0), ADDU (rt=0), LUI, SEH, SEB, ZEH, ZEB, SLT, SLTI, SLTIU, SLTU, SLL (shift<=8) and SRL (31<=shift<=25)} will complete and bypass the results from AC stage to both ALU and AGEN pipe consumers.
  - ADD, ADDU, ADDI, ADDIU instructions can bypass the results in AC to the consumers in the ALU pipe. If the consumer instructions are in the AGEN pipe, these instructions will bypass the results from the AB stage.
- The AC stage is aligned with the start of the Multiply/Divide Unit (MDU) and the CorExtend unit.
- Results bypass for all operations is performed in the AB stage. The results are also prepared for writing into the completion buffer in the following cycle. One exception to this rule are the ADD operations bypassing to the consumer instructions in the ALU pipe.

The latency of the ALU operation is 1 or 2 cycles. For 2-cycle operations, the first cycle is required to perform the arithmetic operation, and the second cycle is required to select and forward the results to potential consumer instructions. The ALU supports a throughput of 1 operation per cycle.

AGEN Pipe

The AGEN pipe spans 5 stages as follows:

- The first stage (EM) is used to select the operands that are read from the register file and completion buffer. The register file and completion buffer read stage overlays the DM stage of the IDU and does not contribute to the pipestage delay of the instruction.
- The data address for load/store operations is calculated using a 32-bit adder in the EA stage.
- Data cache access and JTLB access for load/store instructions is performed in the EC stage.
- The EC stage is also used for resolving conditional branches and register indirect jumps.
- The ES and EB stages are used by the load/store instructions to select the appropriate way of data from the data cache, to compare the JTLB results with the physical tags, align the data, resolve any exceptions, and to bypass the data (if applicable) back into the ALU and AGEN pipes.
- The ES stage is also used to send the redirect PC to the IFU, if there is a mis-predicted branch/jump instruction.

Multiply/Divide Unit (MDU)

The 1074K CPU includes a multiply/divide unit (MDU) that contains a separate pipeline for integer multiply and divide operations. This unit also executes multiply class instructions in the DSP-ASE. This pipeline operates in parallel with the integer unit pipeline and has a separate write port to the ALU completion buffer.

The MDU consists of a pipelined 32×32 multiplier, result/accumulation registers (HI and LO), a divide state machine, and the necessary multiplexors 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 radix 2 iterative SRT algorithm. The operands are always normalized, i.e., leading zeroes in the divisors and dividend are removed. This reduces the total number of cycles required to produce the result. Divide operations block the MDU and will not allow another MDU operation to enter until the current operation is complete. The MDU, however, looks ahead and informs the IDU that a divide operation is about to complete, which prevents any bubbles in the MDU pipeline.

Table 2 lists the repeat rate (i.e., peak rate (in cycles) at which these operations may be issued consecutively) and latency (number of cycles until a result is available) for the 1074K CPU multiply and divide instructions. The approximate latency and repeat rates are listed in terms of pipeline clocks.

Table 2 1074K™ CPU 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	7	1[1]
DIV, DIVU	8 bits	Min: 11 Max: 20	Min: 11 Max: 20
	32 bits	Min: 11 Max: 50	Min: 11 Max: 50

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

CorExtend® Unit

The CorExtend unit allows the user to add a functional unit to the 1074K CPU pipeline with access to all programmer-visible GPR and Accumulator state.

The user will be provided with a template to define the operand format and latency for the new instruction(s) to be added. Up to 15 new instructions may be added. Each instruction may select up to 2 source GPRs and/or 1 Accumulator from the complete architectural state of 32 GPRs and 4 accumulators. The instruction may have a destination of either a GPR, an accumulator, or a private state. The latency for each instruction is also selectable to be either 3, 5, or >5 cycles. Instructions with a destination of private state have a latency of 1 cycle. The CorExtend unit may also have private architectural state, and the existence of such state can be indicated in the template to restrict out-of-order issue. If there is no private state or there is no dependence on private state, then the IDU along with the ALU and MDU pipes manage the dependency checking, operand delivery, and results update. If a CorExtend instruction has its source and/or destination operands from its own private state, it will be issued in program order.

The CorExtend unit is synthesized along with the CPU and will have an external interface for access to any state within that unit. The number of completion buffers for CorExend instructions is selectable at synthesis configuration time (from 1 to 15), and this will determine the number of CorExtend instructions that can be in flight before graduating. This is analogous to the ALU and AGEN completion buffers. The repeat rate of CorExtend instructions that can be issued back to back is also configurable at synthesis time. This parameter controls the repeat rate of instructions that may either read or write private state.

Coprocessor Interface Unit (CIU) (1074Kf CPU Only)

The CIU provides an interface between the main integer CPU and the CP1 blocks. The CIU holds an in-order instruction queue (IOIQ) for the Coprocessor1 interface, a coprocessor load data queue (CLDQ), a coprocessor Store Data Queue (CSDQ), and a few other structures to pass data to and from the coprocessors.

Instructions are written in order into the IOIQ by the IDU. Coprocessor1 Load/Store instructions are sent to the AGEN pipe and to the corresponding IOIQ. Instructions exit the IOIQ and enter the Coprocessor, at which point they will read the data from the CLDQ if it is ready. If the data is not ready, the Coprocessor will either wait for the data or issue ahead, depending on its capability. A CLDQ entry is released only when its data has been consumed by the Coprocessor. If data is to be written from the Coprocessor back into the integer CPU, it is sent back through the CSDQ.

Even though some Coprocessor instructions do not go through the main integer pipeline, they are assigned an instruction identifier. This identifier is tracked in the Graduation Unit to generate a synchronization signal that is used to indicate to the Coprocessor that the coprocessor instruction has been cleared of all speculation and exception conditions in the integer pipe. Only coprocessor instructions that have reached such a state are allowed to commit results in the Coprocessor.

Coprocessor-based conditional branches are handled in the GRU, with condition-code information passed through the CIU.

Floating Point Unit (FPU) / Coprocessor 1 (1074Kf CPU Only)

The 1074K CPU 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, also called "Coprocessor 1", contains thirty-two, 64-bit floating point registers used for floating point operations.

The FPU can be configured at build time to run at the same, one-half, or two-thirds the frequency of the integer CPU. The FPU is not as deeply pipelined as the integer CPU, so the maximum CPU frequency will only be attained when the FPU is running at a fraction of the CPU frequency. The FPU is connected via an internal 64-bit coprocessor interface. Note that clock cycles of floating point operations are shown using FPU clocks, not integer CPU clocks.

The performance of the FPU is optimized for single-precision formats. Most instructions have a 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 required 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 *FCSR* register, and use of this mode is recommended for best performance when denormalized results are generated.

The FPU has two separate pipelines for floating point instruction execution—one for load/store instructions and another for all other instructions. These pipelines operate in parallel with the integer CPU pipeline and do 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 integer CPU will perform the data access for load/store operations and transfer data to and from the FPU using the CIU. Load data may arrive in the FPU out-of-order relative to program order. The exception model is 'precise' at all times.

FPU Pipeline

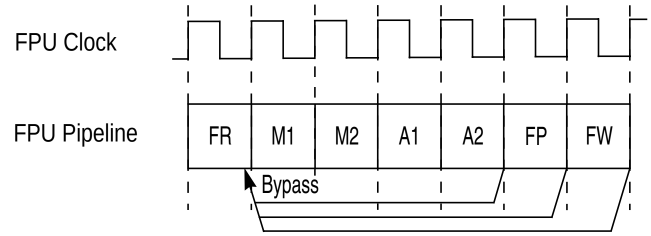
The FPU implements a high-performance pipeline with the following 7 stages:

- 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 4 shows the FPU pipeline.

Figure 4 FPU Pipeline



FPU Instruction Latencies and Repeat Rates

Table 3 contains the floating point instruction latencies and repeat rates for the 1074K CPU. 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 3 1074K™ CPU FPU Latency and Repeat Rate

Opcode[1]	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[2]	1
MFC1, DMFC1, SWC1, SDC1, SDXC1, SUXC1, SWXC1	1	1

1. Format: S = Single, D = Double, W = Word, L = Longword
2. Best case assuming 2:1 clock ratio

Load/Store Unit (AGEN pipe)

The Load/Store Unit is responsible for interfacing with the CPU pipe and handling load/store instruction to read/write data from data caches and/or memory. This unit is capable of handling loads and stores issued out-of-order. Loads, however, are not issued by the IDU until all prior stores have been issued.

Data cache sizes of 16K, 32K and 64K bytes are supported. The cache is 4-way set associative and uses an LRU replacement algorithm. There are separate virtual and physical tag arrays corresponding to the data array. The virtual tag is accessed in parallel with the data cache array and is compared against the virtual address to predict the way. The physical tag is always compared with the result of the JTLB to validate the way selection.

In addition to the data cache, the LSU also supports a scratchpad RAM for sizes ranging from 4KB to 1MB. The LSU interfaces to a 16/32/48/64 dual-entry JTLB. The LSU can handle both integer and floating point load/store instructions and has a 64-bit data path.

Loads are non-blocking in the 1074K CPU. Loads that miss in the data cache are allowed to graduate with their destination register marked unavailable. Consumers of this destination register are held back at the IDU until all their operands become available. Consumers that have already been dispatched are replayed through the pipe and held back at the IDU on its second pass through the pipe. Loads that hit in the data cache and bypass to the AGEN pipe have a 4-cycle load-use latency, while those that bypass to the ALU pipe will have a 3-cycle load-use latency.

Graduated load misses and store hits and misses are sent in order to the Load/Store Graduation Buffer (LSGB). The LSGB has corresponding data and address buffers to hold all relevant attributes. LSGB entries are processed in a FIFO order, with data cache updates and requests made at one canonical point. Cache fill requests are merged and processed at this point. An 8-entry Fill Store Buffer (FSB) tracks outstanding fill requests and fills the data cache when the line is completely received. Each FSB entry can hold an entire cache line. The Load Data Queue (LDQ) keeps track of outstanding load misses and forwards the critical data to the main pipe as soon as it becomes available. The FSB also holds data for store instructions (regardless of a hit or miss in the cache) that have not yet updated the cache. Loads that reference the same line as the pending store in the FSB will receive the store data bypassed (if they are younger than the store), and the incoming line is merged with the store data before being written into the cache. One FSB entry is reserved for intervention processing. Loads that are older than the store are tracked in the Load Data

Queue (LDQ) and will receive the data when it arrives from the BIU.

Graduation Unit (GRU)

The Graduation Unit is responsible for committing execution results into architectural state and releasing buffers and resources used by these instructions. The GRU is also responsible for evaluating the exception conditions reported by execution units and taking the appropriate exception. Asynchronous interrupts are also funneled into the GRU, which prioritizes those events at the existing conditions and takes the appropriate interrupt.

The GRU receives information about the program order of instruction from the Graduation FIFO (GFIFO). The GFIFO is written by the IDU at dispatch time. The GFIFO entry has a pointer to the completion buffer and associated structures where various attributes such as PC, exception information, etc. are held.

The GRU will read up to 2 completed instructions from the GFIFO every cycle and then read the corresponding completion buffer and associated information. After processing the exception conditions, the destination register(s) are updated and the completion buffers are released. The GRU also sends graduation information to the IDU, so that it can update the rename maps to reflect the state of execution results (i.e., GPRs, Accumulators, etc.). The GRU also sends resolved branch information to the IFU, so that branch history tables can be updated.

Load misses and store hits/misses are sent to the LSGB for further processing. When the LSU receives the data back from outside, it directly updates the architectural state, but the GRU ensures that the LSGB is kept up-to-date, so that only the latest data is written. If there is no space in the LSGB, the GRU will stop graduating load/store instructions, which holds the releasing of completion buffers.

The GRU also handles instructions such as CACHE, MTC0, and TRAP-on-condition type operations that require serialized operation. During such operations, the GRU throttles down to graduating 1 instruction per cycle; otherwise, the GRU will always attempt to graduate 2 instructions per cycle.

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, and the presence of features like MIPS16e or a floating point unit, are also available by accessing the CP0 registers.

CP0 also contains the state used 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 1074K CPU supports 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. As it is always present on the 1074K CPU, the *VInt* bit will always read 1.
- External Interrupt Controller (EIC) mode, which redefines the way in which interrupts are handled, in order to provide full support for an external interrupt controller that handles prioritization and vectoring of interrupts. This mode is optional in the Release 2 architecture. The presence of this mode is denoted by the *VEIC* bit in the *Config3* register. On the 1074K CPU, 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 the 1074K CPU is configured to use shadow registers, the VI and EIC interrupt modes can specify which shadow register to use on 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 1074K CPU 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 provides 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 21 for more information on debug mode.

Memory Management Unit (MMU)

The 1074K CPU 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 from the following options:

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

The following sections explain the MMU options in more detail.

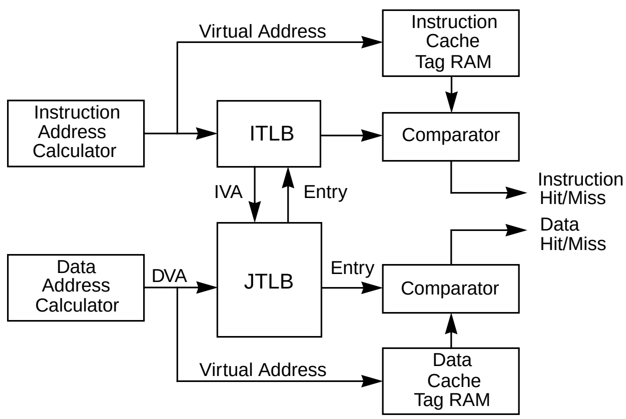
Translation Lookaside Buffer (TLB)

The basic TLB functionality is specified by the MIPS32 Privileged Resource Architecture. A TLB provides mapping and protection capability with per-page granularity. The 1074K CPU implementation allows a wide range of page sizes to be present simultaneously.

The TLB contains a fully associative dual-ported Joint TLB (JTLB). To enable higher clock speeds, a smaller instruction micro-TLB (ITLB) is also implemented. When an instruction address is calculated, the virtual address is compared to the contents of the appropriate ITLB. If the address is not found in the ITLB, the JTLB is accessed. If the entry is found in the JTLB, that entry is then written into the ITLB; if the address is not found in the JTLB, a TLB exception is taken. For data accesses, the virtual address is looked up in the JTLB only, and a miss causes a TLB exception.

Figure 5 shows how the ITLB and JTLB are implemented in the 1074K CPU.

Figure 5 Cache Access for Address Translation



Joint TLB (JTLB)

The JTLB is a dual-ported fully associative TLB cache containing 16, 32, 48, 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) with each of the entries in the *tag* portion of the joint TLB structure.

The JTLB is organized as pairs of even and odd entries that map pages ranging in size from 4 KB to 256 MB, in factors of four, to 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. Because 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 4-entry structure dedicated to performing translations for the instruction stream. The ITLB maps only 4 KB or 16 KB pages/subpages.

For 4 KB or 16 KB pages, the entire page is mapped in the ITLB. If the main TLB page size is between 4 KB and 16 KB, only the current 4 KB subpage is mapped. Similarly, for page sizes larger than 16 KB, the current 16 KB subpage is mapped.

The ITLB is managed by hardware and is transparent to software. The larger JTLB is used as a backup structure for the ITLB. If a fetch address cannot be translated by the ITLB, the JTLB attempts to translate it in the following clock cycle or when available. If successful, the translation information is copied into the ITLB for future use. The JTLB port used for ITLB miss access is shared with other MMU management activities.

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 are 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 0/16/32/64 KB, with 4-way associativity. All size references made will assume a default size of 32 KB. Because the instruction cache is virtually indexed, the virtual-to-physical address translation occurs in parallel with the tag access, rather than having to wait for the physical address translation. A tag entry holds 21 bits of physical address, a valid bit, a lock bit, and an optional parity bit. There are 7 precode bits per instruction pair, making a total of 28 bits per tag entry. The data array line consists of 256 bits (8 MIPS32 instructions) of data. Each instruction doubleword (64 bits) has 8 bits of byte parity. The IFU interface consists of 128 bits (4 MIPS32 instructions) with 16 bits of parity. The LRU replacement bits (6 bits) are shared among the 4 ways of the data and tag array and are stored in a separate array.

The instruction cache block also contains and manages the two instruction line fill buffers. 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 by data coming from the external interface. The instruction cache control logic controls the bypass function.

The 1074K CPU supports instruction-cache locking. Cache locking allows critical code 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 16/32/64 KB, with 4-way associativity. Because the data cache is virtually indexed, the virtual-to-physical address translation occurs in parallel with the cache access. A tag entry holds 21 bits of physical address, a valid bit, a lock bit, and an optional parity bit. At each tag entry there is also a corresponding 21 bit virtual tag. 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 (6 bits LRU, 4 bits dirty, and optionally 4 bits dirty parity).

When using 4 KB pages in the TLB and 32 or 64 KB cache sizes, virtual aliasing can occur, in which a single physical address can exist in multiple cache locations if it was accessed via different virtual addresses. 1074K hardware implementation eliminates virtual aliasing.

The 1074K CPU supports a data-cache locking mechanism identical to that used in 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 1074K CPU’s on-chip instruction and data caches are usually implemented from readily available single-port synchronous SRAMs.

The instruction tag array is accessed in one cycle, and the corresponding instruction data array is accessed in the following cycle. While the instruction data is being accessed, the tag data is compared to the translated address to determine a hit. The result of this hit is used to select the way of the instruction data in the following cycle, thus completing the 3-cycle sequence.

The data cache and tag arrays are accessed in the same cycle. The JTLB is also accessed at the same time for virtual to physical address translation. The virtual tag match with the virtual address is used to select the data cache way in order to bypass data as soon as possible. The result of the JTLB compare is used to further determine a match with the physical tag in the tag array to validate the virtual tag match. If the two comparisons do not agree, the data cache access is deemed to be a miss. The data cache refill can be done via a 128-bit interface and is a synthesis-time configuration option.

Table 4 lists the attributes of the 1074K CPU instruction and data caches.

Table 4 1074K™ CPU Instruction and Data Cache Attributes

Parameter	Instruction	Data
Size	0, 16, 32, or 64 KB[1]	16, 32, or 64 KB
Organization	4-way set associative	4-way set associative
Line Size	32 Bytes[1]	32 Bytes
Read Unit	128 bits[1]	128 bits
Write Unit	128 bits	128 bits
Write Policies	N/A	coherent and non-coherent writeback with write allocate
Cache Locking	per line	per line

1. Logical size of instruction cache. The cache contains some extra bits used for precoding the instruction type.

Cache Protocols

The 1074K CPU supports the following non-coherent cache protocols:

- **Non-Coherent Uncached:** Addresses in a memory area specified as uncached are not read from the cache. Stores to uncached addresses are written directly to main memory, without changing the contents of the cache.
- **Non-Coherent Writeback, write allocate:** Stores that miss in the cache will cause a cache refill. Store data, however, is only written to the cache. The memory read request will be marked as non-coherent. However, once the data is written to the cache, there is no distinction between coherent and non-coherent data. Cache 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.
- **Non-Coherent Uncached accelerated:** As with the uncached protocol, data is never loaded into the cache. In this mode, 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 separate individual writes, as is done in uncached mode.
- **Coherent, writeback, write allocate, exclusive on write:** Use coherent data. Load misses will bring the data into the cache in a shared state. Multiple caches can contain data in the shared state. Stores will bring data into the cache in an exclusive state - no other caches can contain that same line. If a store hits on a shared line in the cache, a request will be made to upgrade to exclusive.
- **Coherent, writeback, write allocate, exclusive:** Similar to the above, but load misses will bring data into the cache in an exclusive state rather than shared. This can be used if data is not shared and will eventually be written. This can reduce bus traffic because the line does not have to be refetched in an exclusive state when a store is done.

Intervention Processing

The CPU includes a duplicate set of tags for the data cache as well as the fill-store buffer and writeback buffer. Duplicate tags allow intervention lookups to be done in parallel with regular cache accesses for loads and stores. Intervention processing is fully pipelined.

If an intervention hits and needs to change the cache state or read the cache data array, it is allocated an FSB entry. The FSB entry opportunistically accesses the cache to perform the required action. Load and store accesses from the main pipeline to non-overlapping cache lines can continue being processed. Processing of interventions that miss is complete after the “state” response is returned. Intervention “state” and “data” responses are split to allow fast turnaround.

If a read-type intervention hits in the cache on a line that is Exclusive or Modified, the CPU reads the data out of the cache and returns it on the intervention port. This data is staged through the writeback buffer, and at least one entry of the buffer will always be available for interventions to avoid deadlock conditions.

Scratchpad RAM

The 1074K CPU allows blocks of scratchpad RAM to be attached to the load/store and/or instruction units. These allow low-latency access to a fixed block of memory.

These blocks can be modified by the user. A reference design is provided that includes an SRAM array and an external DMA port that allows the system to directly access the array.

L2 Cache Support

The 1074K CPU supports building a Level 2 cache on the front side bus inline with the memory access. This L2 cache is unified and contains both instruction and data segments. The L2 cache can be configured to be by-passable, i.e., memory accesses from the 1074K CPU can bypass the L2 cache directly access the main memory.

The L2 cache configuration and functional details are provided in the document *MIPS® SOC-it® L2 Cache Controller Datasheet*, MD00502.

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.

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.

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 in order to increase the efficiency of the bus. Uncached accelerated gathering is supported for word or doubleword.

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

When 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 doubleword stores to the same 32-bit 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 1074K CPU to structures that 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 in which 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 1074K CPU includes an Enhanced JTAG (EJTAG) block for use in software debugging of application and kernel code. For this purpose, in addition to standard user/supervisor/kernel modes of operation, the 1074K CPU provides a Debug mode. Debug mode is entered when a debug exception occurs (resulting from a hardware breakpoint, single-step exception, etc.) 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 1074K CPU. In addition to the standard JTAG instructions, special instructions defined in the EJTAG specification define which 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 1074K CPU: Instruction breakpoints and Data breakpoints.

During synthesis, the 1074K CPU can be configured to support the following breakpoint options:

- Zero instruction, zero data breakpoints
- Four instruction, 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.

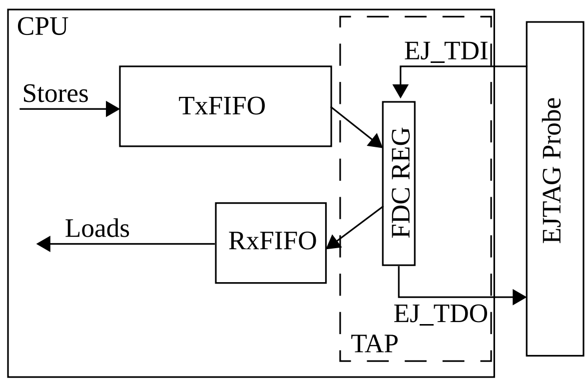
Data breakpoints occur on load and/or store transactions. Breakpoints are set on virtual address and ASID values, similar to the Instruction breakpoint. Data breakpoints can also be set based on the value of the load/store operation. Finally, masks can be applied to the virtual address, ASID value, and 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. This reset pin has no effect inside the CPU.

Fast Debug Channel

The 1074K CPU includes the EJTAG Fast Debug Channel (FDC) as a mechanism for efficient bidirectional data transfer between the CPU and the debug probe. Data is transferred serially via the TAP interface. A pair of memory- mapped FIFOs buffer the data, isolating software running on the CPU from the actual data transfer. Software can configure the FDC block to generate an interrupt based on the FIFO occupancy or can poll the status.

Figure 6 Fast Debug Channel



MIPS Trace

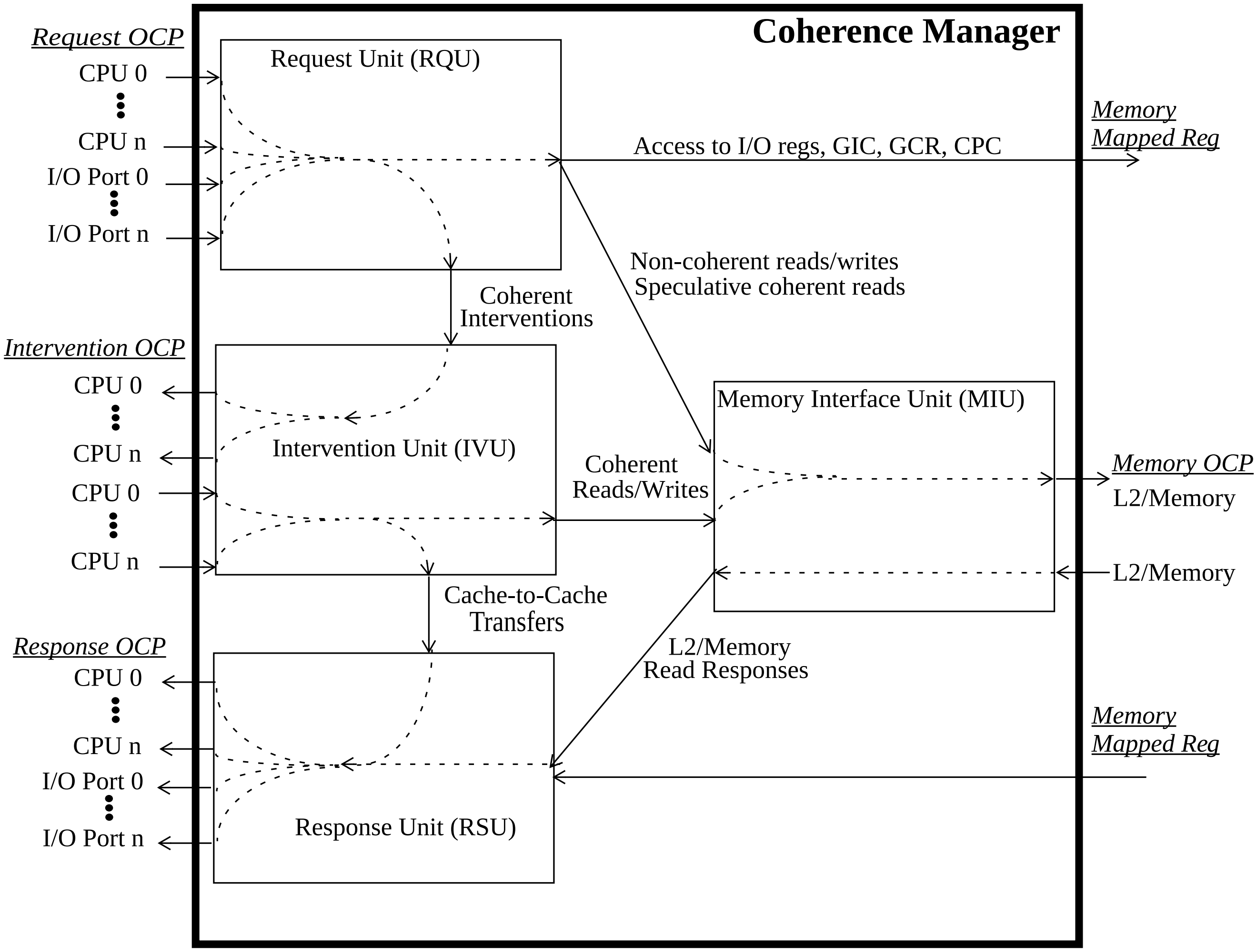
The 1074K CPU includes optional MIPS Trace support for real-time tracing of instruction addresses, data addresses, data values, performance counters, and processor pipeline inefficiencies. The trace information is sent out of the CPU to a trace funnel where it is interleaved with trace data from the other CPUs and Coherence Manager. The trace information is collected in an on-chip or off-chip memory, for post-capture processing by trace regeneration software. Software-only control of trace is possible in addition to probe-based control.

An optional on-chip trace memory may be configured in size from 256B to 8 MB; it is accessed either through load instructions or the existing EJTAG TAP interface, which requires no additional chip pins. Off-chip trace memory is accessed through a special trace probe and can be configured to use 4, 8, 16, or 64 data pins plus a clock.

Coherence Manager

The Coherence Manager (CM) is responsible for establishing the global ordering of requests and for collecting the intervention responses and sending the correct data back to the requester. A high-level view of the request/response flow through the CM is shown in Figure 7. Each of the sub-units is described in more detail in the following subsections.

Figure 7 Coherence Manager Transaction Flow



Request Unit (RQU)

This block receives requests from the coherent devices and serializes them. Non-coherent requests are forwarded to the Memory Interface Unit. Coherent requests are sent to the Intervention Unit.

The CM supports speculative reads. For coherent read requests, the memory request is started assuming that the line will not be available from another L1 cache. This avoids increasing the memory latency for the common case when it is not. In order to avoid a read-after-write hazard, the read address is compared with all pending coherent requests that can generate writes; if a match is detected, the speculative read will not be started.

Intervention Unit (IVU)

This block receives the serialized stream of coherent requests from the Request Unit. These requests are sent as interventions to each of the coherent caching agents. The caching agent updates its cache state appropriately for the intervention and gives a response. If the cache has the line in an Exclusive or Modified state, it returns the data with its response on a read type intervention. The Intervention Unit gathers the responses from each of the agents and manages the following actions:

- Speculative reads are resolved (confirmed or cancelled).
- Memory reads that are required because they were not speculated are issued to the Memory Interface Unit.
- Modified data returned from the CPU is sent to the Memory Interface Unit to be written back to memory.
- Data returned from the CPU is forwarded to the Response Unit to be sent to the requester.
- The MESI state in which the line is installed by the requesting CPU is determined (the "install state"). If there are no other CPUs with the data, a Shared request is upgraded to Exclusive.

Memory Interface Unit (MIU)

This block handles the interface to the L2 cache or memory. Non-coherent reads and writes as well as speculative coherent reads are sent to the Memory Interface Unit from the Request Unit. Coherent writes and late reads are generated from the Intervention Unit.

The external interface may operate at a lower frequency than the CM, and the external block may not be able to accept as many requests as multiple CPUs can generate, so some buffering of requests is performed.

This block is responsible for staging the read data back to each of the agents. There are independent staging registers for each agent, which allows concurrent data return to different agents. This block buffers the read data returned from the system if the read is still speculative, or if the response unit is busy.

Response Unit (RSU)

This block is responsible for returning data to the requesting agent. It can send data from the Intervention Unit, the Memory Interface Unit, or from memory-mapped accesses to the GCR/GIC/MMIO.

Performance

The CM has a number of features that improve performance:

- Cache to Cache transfers: If a read request hits in another L1 cache in the Exclusive or Modified state, it will return the data to the CM and it will be forwarded to the requesting CPU, thus reducing latency on the miss.
- Speculative Reads: Coherent read requests are forwarded to the memory interface before they are looked up in the other caches. This is speculating that the cache line will not be found in another CPU's L1 cache. If another cache was able to provide the data, the memory request is not needed, and the CM will cancel the speculative request—dropping the request if it has not been issued, or dropping the memory response if it has.

Table 5 provides a cycle-by-cycle description of the latency added by the CM to a read request, assuming the internal queues are all empty and the CM/Core clock mode is Synchronous-Only (always 1:1). In this case, the CM adds 6 cycles to the round-trip latency of the request. When the CM/Core Clock Mode is Semi-Synchronous (allows for non-1:1 clock ratios), additional latency occurs on Cycles 0 and 6+N. The amount of the additional latency is dependent upon the clock ratio selected.

Table 5 Read Request Timing

Cycle	Description
0	CPU sends out request, captured by input buffers in CM.RQU.
1	Serialization of requests in CM.RQU. One request is selected in Round-robin fashion from all sources.
2	Request is sent from CM.RQU to CM.MIU (bypassing internal queues).
3	Request is presented on L2/memory interface.
3+N	Response from L2 on bus. Captured by input flops in CM.MIU.
4+N	Look-up in table to determine what to do with response (drop, wait for intervention, etc.). Response data forwarded to CM.MIU output queue.

Table 5 Read Request Timing

Cycle	Description
5+N	Response is forwarded to the CM.RSU.
6+N	Response is sent back to the CPU.

I/O Coherence Unit (IOCU)

Optional support for hardware I/O coherence is provided by the I/O Coherence Unit (IOCU), which maintains I/O coherence of the caches in all coherent CPUs in the Cluster.

The IOCU acts as an interface block between the Coherence Manager (CM) and coherent I/O devices. Coherent reads and writes of I/O devices generate interventions in other coherent CPUs that query the L1 cache. I/O reads access the latest data in caches or in memory, and I/O writes invalidate stale cache data and merge newer write data with existing data as required. An example system topology is shown in Figure 8.

A reference design is provided for the IOCU. This block provides a legacy (without coherent extensions) OCP slave interface to the I/O interconnect for I/O devices to read and write system memory. The reference design also includes an OCP Master port to the I/O interconnect that allows the CPUs to access registers and memory on the I/O devices.

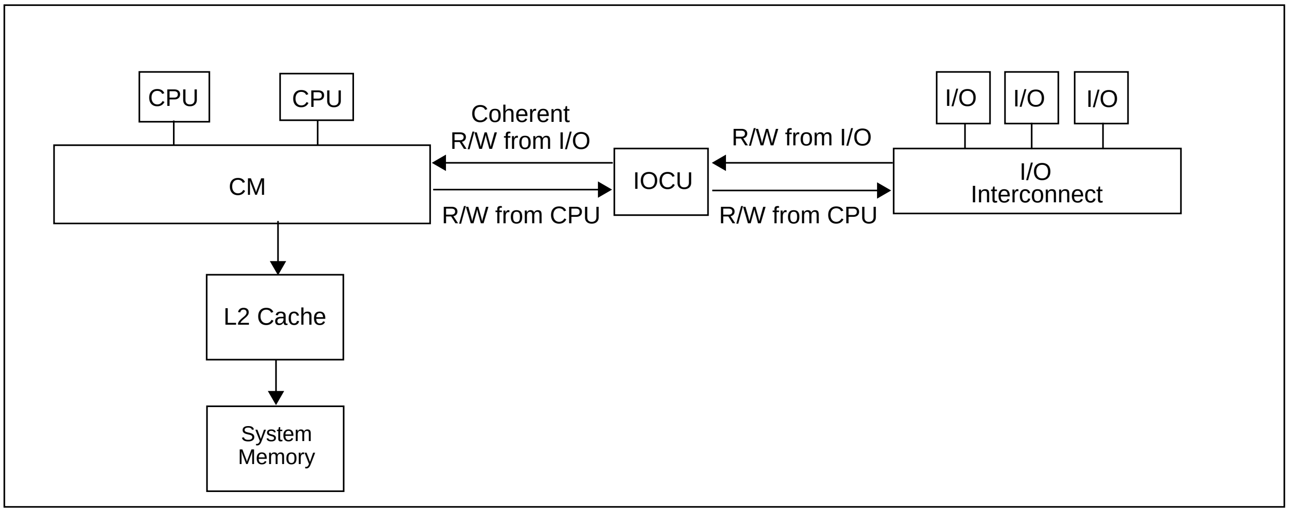
The reference IOCU design provides several features for easier integration:

- A user-defined mapping unit can define cache attributes for each request—coherent or not, cacheable (in L2) or not, and L2 allocation policy.
- Supports incremental bursts up to 16 beats on I/O side. These requests are split into cache-line- sized requests on the CM side.
- Ensures proper ordering of responses for the split requests and tagged requests.

In addition, the reference design has a number of features that support the producer-consumer ordering model and help ensure that transaction ordering can be enforced. These features include:

- Set-aside buffer: This buffer can delay read responses from the I/O device until previous writes have completed.
- Writes are issued to the CM in the order they were received.
- The CM provides ACK to the IOCU when writes are "visible" (guaranteed that a subsequent CPU read will receive that data):
  - non-coherent write is ACK'ed after serialization
  - coherent write is ACK'ed after intervention complete on all CPUs
- The IOCU can be configured to treat incoming writes as non-posted and provide a write ACK when they become visible.
- The IOCU maintains order-of-write visibility within a thread. When a thread has coherent writes followed by non-coherent writes, the IOCU will not issue the non-coherent writes until all previous coherent writes from that thread have been ACK'ed by the CM.

Figure 8 I/O Coherent System



Software I/O Coherence

For cases where system redesign to accommodate hardware I/O coherence is not feasible, the CPUs and Coherence Manager provide support for an efficient software-managed I/O coherence. This support is through the globalization of hit-type CACHE instructions. When a coherent address is used for the CACHE operations, the CPU makes a corresponding coherent request. The CM will send interventions for the request to all of the CPUs, allowing all of the L1 caches to be maintained together. The basic software coherence routines developed for single CPU systems can be reused with minimal modifications.

L2 Cache Interface

When the CM is used with the SOC-it L2 cache controller, an optimized 256b interface is used. The L2 cache controller is able to access 256b at a time, so expanding the interface allows that cache bandwidth to be better utilized.

This interface is not a true 256b OCP interface. Because the CM-CPU interface and L2-memory interfaces are 64b OCP, converting to a true 256b OCP would add two additional alignment steps and increase latency. The key differences are:

- Data can be returned either 64b or 256b at a time. On L2 misses, the data is returned from the system 64b at a time and can be passed directly back, rather than forcing the L2 to gather all 256b before responding.
- Data is aligned such that the critical dword of the original request is always in bits [63:0], which eases timing pressure when the critical dword bypasses the CM's read buffer.

This interface can be thought of as a 64b OCP interface, where all 4 beats of a burst can be returned in a single cycle.

System Features

Global Configuration Registers (GCR)

The Cluster includes a set of memory-mapped registers that are used to configure and control various aspects of the Coherence Manager and the coherence scheme.

Reset Control

The reset input from the system will reset the entire Cluster, including all 1074K CPUs, the IOCU, and the Coherence Manager. After reset is released, the Coherence Manager, IOCU, and CPU0 will be active, but the remaining CPUs will continue to be held in reset. This allows CPU0 to initialize the system resources and perform the bringup in a controlled manner. Software must explicitly enable each of the other CPUs by writing to a reset register.

In addition to controlling the deassertion of the CPU reset signals, there are memory-mapped registers that can set the value for each CPU's *SI_ExceptionBase* pins. This allows different boot vectors to be specified for each of the CPUs so they can execute unique code if required.

Following reset, the caches are uninitialized and coherent requests should not be looked up in the cache. A coherence- enable register can be set after the caches have been flushed and the CPU is ready to start participating in the coherence scheme.

Inter-CPU Debug Breaks

The 1074K Cluster includes registers that enable cooperative debugging across all CPUs. Each VPE features an *EJ_DebugM* output that indicates it has entered debug mode (possibly through a debug breakpoint). Registers are defined that allow CPUs to be placed into debug groups such that whenever one CPU within the group enters debug mode, a debug interrupt is sent to all CPUs within the group, causing them to also enter debug mode and stop executing non-debug mode instructions.

CM Control

Registers in the GCR allow software to configure and control various aspects of the operation of the Coherence Manager. Some of the control options include:

- *Address map*: the base address for the GCR and GIC address ranges can be specified. An additional four address ranges can be defined as well. These control whether non-coherent requests go to memory or to memory-mapped I/O. A default can also be selected for addresses that do not fall within any range.
- *Error reporting and control*: Logs information about errors detected by the CM and controls how errors are handled (ignored, interrupt, etc.).
- *Control Options*: Various features of the CM can be disabled or configured. Examples of this are disabling speculative reads and preventing ReadShared requests from being upgraded to Exclusive.

Global Interrupt Controller

The Cluster includes an interrupt controller. This block has the following features:

- Software interface through relocatable memory-mapped address range.
- Configurable number of system interrupts - from 8 to 256 in multiples of 8.
- Support for different interrupt types:
  - Level-sensitive: active high or low.
  - Edge-sensitive: positive, negative, or double-edge-sensitive.
- Ability to mask and control routing of interrupts to a particular CPU.
- NMI routing is also supported.
- Standardized mechanism for sending inter-processor interrupts.

Clock and Test Considerations

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

Clocking

The 1074K CPU Cluster has various clock domains:

- Core domain - This is the main CPU clock domain, controlled by the *SI_ClkIn* clock input. All cores in the Cluster must operate at the same frequency.
- CP1 domain - This domain provides the clock for Coprocessor1 block and is controlled by the *SI_Cp1ClkIn* and *SI_Cp1Sync* inputs. Applicable to core configured with 3:2 Core to FPU clock ratio. Applicable to 1074Kf core only.
- CM Domain - This is the domain for most of the Coherence Manager, Global Interrupt Controller and Cluster Power Controller. The CM can be configured to operate in one of two Core/CM clock modes:

- In Synchronous-Only Clock Mode, the CM Domain is equivalent to the Core domain.

- In Semi-synchronous Clock Mode, the CM Domain may be a ratio of 1:1, 5:4, 3:2 or 2:1 of the Core Domain frequency. In this mode, CPU memory accesses incur additional latency.

- IO Domain - The OCP port connecting the IOCU to the IO Subsystem may also operate at a ratio of the CM Domain. Supported ratios are 1:1, 1:1.5, 1:2, 1:2.5, 1:3, 1:3.5, 1:4, 1:5, and 1:10
- L2 Domain - When an L2 Cache is configured, it may operate at a ratio of 1:1, 1:1.5, 1:2, 1:2.5, 1:3, 1:3.5, 1:4, 1:5, and 1:10.
- Memory Domain - The OCP port connecting to the Memory subsystem may operate at a ratio of the L2 Domain (if the L2 is present) or the CM Domain (if the L2 is not present). The same ratios as the L2:CM domains 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 1074K CPU Cluster offers a number of power management features, including low-power design, active power management, and power-down modes of operation. The CPU is a static design that supports slowing or halting the clocks to reduce system power consumption during idle periods.

Cluster Power Controller

Individual CPUs within the cluster can have their clock and/or power gated off when they are not in use. This gating is managed by the Cluster Power Controller (CPC). The CPC handles the power shutdown and ramp-up of all CPUs in the Cluster. Any 1074K CPU that supports power-gating features is managed by the CPC. The CPC also organizes power-cycling of the CM dependent on the individual core status and shutdown policy. Reset and root-level clock gating of individual CPUs are considered part of this sequencing.

The 1074K CPU 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 pin. 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 1074K CPU is in a low-power state. Depending on what type of exception is taken, one of these three bits will be set to 1 and be reflected in the *SI_EXL*, *SI_ERL*, and *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 four 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.

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. When the CPU is in this instruction-controlled power management mode, any interrupt, NMI, intervention or reset condition causes the CPU to exit this mode and resume normal operation.

The 1074K CPU asserts the *SI_Sleep* signal, which is part of the system interface, whenever it has entered low-power mode (sleep mode). It will enter sleep mode when all bus transactions are complete and there are no running instructions.

The WAIT instruction can put the processor in a mode where no instructions are running. When the WAIT instruction is seen by the IFU, subsequent instruction fetch is stopped. The WAIT instruction is dispatched down the pipe and graduated. Upon graduation of the WAIT, the GRU waits for the processor to reach a quiescent state and allows the processor to enter sleep mode.

Local Clock Gating

A significant portion of the power consumed by the 1074K CPU is often in the clock tree and clocking registers. The CPU has support for extensive use of local gated clocks. Clock gating can be turned on at the top level, block level, or at the register (fine-grained) level. Power-conscious implementors can use these gated clocks to significantly reduce power consumption within the CPU.

D-Cache Clock Gating

Any load instruction involves reading of four ways of the data array, though the required data may be available only in one of the four ways of the D-cache. The way information for four recently used D-cache lines are stored in a data structure, and a subsequent load to one of those lines enables the clock to only one of the data arrays, thereby saving the memory power required for a read operation on three ways of the D-cache. Also, for additional power savings, the D-cache data array clocks are disabled for store instructions and idle cycles. This optional feature significantly reduces the power consumed by the D-cache data array.

Internal Scan

The 1074K supports full mux-based scan for maximum test coverage, with a configurable number of scan chains. ATPG test coverage can exceed 99%, depending on standard cell libraries and configuration options.

Memory BIST

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

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 CPU to facilitate this approach.

Build-Time Configuration Options

The 1074K CPU allows a number of features to be customized based on the intended application. Table 6 summarizes the key configuration options that can be selected when the CPU is synthesized and implemented.

For a CPU 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® 1074K™ CPU 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 CPU are not visible to software.

Table 6 Build-time Configuration Options

Configuration Option	Choices	Software Visibility
System Options		
Number of CPUs	1, 2, 3, 4, 6	*GCR_CONFIG*$_{PCORES}$

Table 6 Build-time Configuration Options (Continued)

Configuration Option	Choices	Software Visibility
I/O Coherence Unit	Present or not	*$GCR\_CONFIG_{NUMIOCU}$*
MConnID mask	0-8b	N/A
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	*$TCBCONFIG_{SZ}$*
Probe Interface Block - Number of data pins	4, 8, 16	N/A
IOCU Options		
IODB implementation style	Flops or generator	N/A
Advanced IOCU Config - fine tuning of buffer sizes, etc.	Varies	N/A
Coherence Manager Options		
Core/CM Clock Mode	Synchronous-Only or Semi-Synchronous	N/A
RWDB implementation style	Flops or generator	N/A
RDB implementation style	Flops or generator	N/A
Number of Address Regions	0, 4, 6	*$GCR\_CONFIG_{NUM\_ADDR\_REGIONS}$*
Default GCR base address & writeability	Any 32KB-aligned physical address Hardwired or programmable	*GCR_BASE*
Default Exception Base for each CPU	Any 4KB-aligned physical address	*GCR_Cx_RESET_BASE*
Advanced CM Config - fine tuning of buffer sizes, etc.	Varies	N/A
Global Interrupt Controller Options		
Number of system interrupts	8*[1-32]	*$GIC\_SH\_CONFIG_{NUMINTERRUPTS}$*
Local routing of CPU sourced interrupts (per VPE)	Present or not	N/A
Local routing of CPU Timer Interrupt	Enabled or not	*$GIC\_VPEj\_CTL_{TIMER\_ROUTABLE}$*
Local routing of CPU Performance Counter Interrupt	Enabled or not	*$GIC\_VPEj\_CTL_{PERFCOUNT\_ROUTABLE}$*
Local routing of CPU Fast Debug Channel Interrupt	Enabled or not	*$GIC\_VPEj\_CTL_{FDC\_ROUTABLE}$*
Local routing of CPU Software Interrupts	Enabled or not	*$GIC\_VPEj\_CTL_{SWINT\_ROUTABLE}$*
Cluster Power Controller Options		
Microstep delay in cycles	1-1024	
RailEnable delay	1-1024	
Power Gating Enabled	Enabled or not	
Clock Gating Enabled	Enabled or not	
CPU Options		

Table 6 Build-time Configuration Options (Continued)

Configuration Option	Choices	Software Visibility
Floating point unit (Applicable to 1074Kf core only)	Present or not	$Config1_{FP}$
Memory Management Type	TLB or FMT	$Config_{MT}$
TLB Size	16, 32, 48, or 64 dual entries	$Config1_{MMUSize}$
Integer Register File sets	1, 2, or 4	$SRSCTL_{HSS}$
{Instruction, Data} hardware breakpoints	{0,0} or {4,2}	DCR_{IB}, IBS_{BCN}
Fast Debug FIFO Sizes	Min (2Tx, 2Rx), Useful (12Tx, 4Rx)	*FDCFG*
CorExtend Block	Present or not	$Config_{UDI}$[1]
Data ScratchPad RAM interface	Present or not	$Config_{DSP}$[1]
Instruction ScratchPad RAM interface	Present or not	$Config_{ISP}$[1]
I-cache size	0, 16, 32, or 64 KB	$Config1_{IL}$, $Config1_{IS}$
D-cache size	16, 32, or 64 KB	$Config1_{DL}$, $Config1_{DS}$
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, block-level, fine-grain, D-cache, or none	N/A
Power gating	Implemented or not	N/A
Control and Observe flops	Present or not	N/A
Repeat rate for CorExtend instructions using private state	1 through 15	N/A
Number of CorExtend completion buffers	1 through 15	N/A
Sideband inputs to external CorExtend module	Bus width (in bits)	N/A
Sideband outputs to external CorExtend module	Bus width (in bits)	N/A

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

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 those sections indicate changes since the previous version of the relevant Architecture document.

Revision	Date	Description
01.00	July 30, 2010	Initial release of the document.
01.01	September 24, 2010	Reclassified the document. Config option changes. Various corrections.
01.02	March 30, 2011	Edits to reflect the maximum number of cores supported to 6.

Revision	Date	Description
01.03	June 03, 2011	Corrected FDC related information.

Template: nDb0.02, Built with tags: 2B