interAptiv™ Multiprocessing System Datasheet

May 13, 2016

The interAptiv™ multiprocessing system (MPS) is a high performance device containing between 1 and 4 coherent processors with best-in-class power efficiency for use in system-on-chip (SoC) applications. The interAptiv architecture combines a multi-threading pipeline with a highly intelligent coherence manager to deliver best-in-class computational throughput and power efficiency. The interAptiv MPS is fully configurable and synthesizable and can contain one to four MIPS32® interAptiv CPU cores, system-level coherence manager with L2 cache, and optional coherent I/O port and floating point unit.

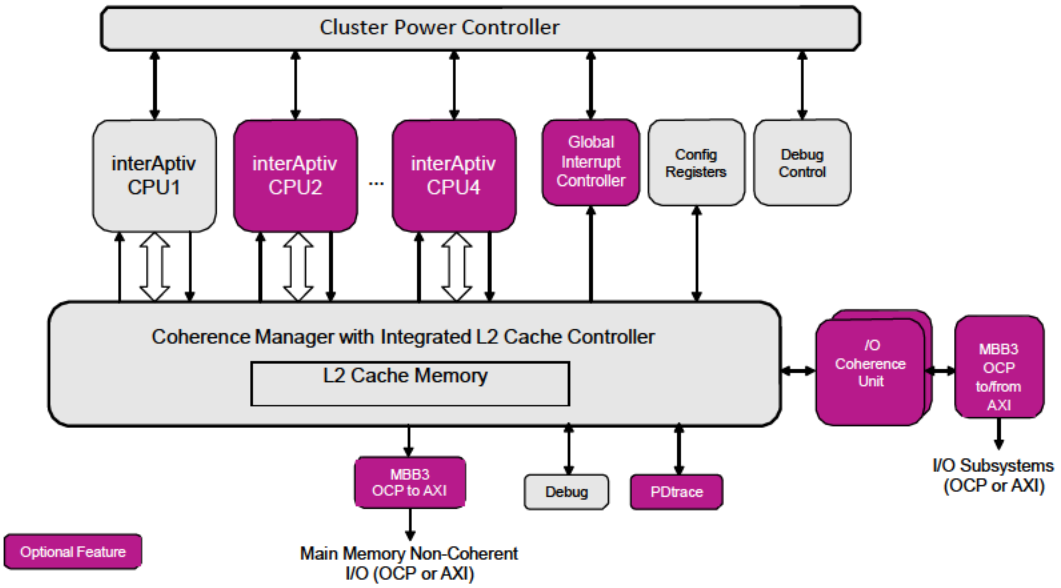
The interAptiv multiprocessing system is available in the following configurations. These configurations include the second generation Coherence Manager with an integrated L2 cache (CM2).

- Dual-core, configurable as either 1 or 2 cores
- Quad-core, configurable as 1, 2, 3, or 4 cores

The MIPS32 interAptiv multiprocessing system contains the following logic blocks:

- interAptiv Cores (1 - 4)
- Cluster Power Controller (CPC)
- Coherence Manager (2nd generation) with integrated L2 cache (CM2)
- Zero - Two I/O Coherence Units (IOCU) (optional)
- Multi-threaded Floating Point Unit (FPU) (optional)
- Global Interrupt Controller (GIC)
- Global Configuration Registers (GCR)
- Coherent Processing System (CPS) Debug Unit
- PDTrace in-system trace debugger (optional)

Figure 1 interAptiv Multiprocessing System Level Block Diagram



In the interAptiv multiprocessing system, multi-CPU coherence is handled in hardware by the Coherence Manager. The optional I/O Coherence Unit (IOCU) supports hardware I/O coherence by bridging a non-coherent OCP I/O interconnect to the CM2 and handling ordering requirements. The Global Interrupt Controller (GIC) handles the distribution of interrupts between and

among the CPUs. Under software controlled power management, the Cluster Power Controller (CPC) can gate off the clocks, voltage supply, or both to idle cores.

Features

System-level Features:

- 1- 4 coherent MIPS32® interAptiv CPU cores
- Second generation system-wide Coherence Manager (CM2) providing L2 cache, I/O and interrupt coherence across all CPU cores
- Integrated 8-way set associative L2 cache controller supporting 0 KB to 8 MB cache sizes with variable wait state control for 1:1 clock and optimal SRAM speed
- Supports cache-to-cache data transfers
- Speculative memory reads and Out-of-order data return to reduce latency
- User-defined global configuration registers
- Hardware I/O coherence port (optional) with up to 2 IOCU's per system
- SOC system interface bus supports
  - OCP version 2.1 protocol with 32-bit address and 64-bit, 128-bit, or 256-bit data paths
  - AXI version 3.0 protocol with 32-bit address and 64-bit, 128-bit, or 256-bit data path
- Power Control
  - Minimum frequency: 0 MHz
  - Software controlled power-down mode
  - Software-controlled clock divider
  - Cluster-level dynamic clocking
  - Cluster Power Controller (CPC) controlling shut down of idle CPU cores
- Core Power Reduction by
  - Turning off core clock during outstanding bus requests
  - Implementing intelligent way selection in the L1 instruction cache
  - Enabling 32-bit accesses of the L1 data cache RAMs
- EJTAG Debug 5.0 port supporting multi-CPU debug, System-level trace, and performance analysis
- MIPS PDtrace debug version 6 (optional)
  - PC, data address and data value tracing with trace compression
  - Includes features for correlation with CM trace
  - Support for on-chip and off-chip trace memory

CPU Core-level Features:

- 8/9-stage pipeline with integer, floating point and optional CorExtend execution units shared amongst issue pipes
- IEEE-754 compliant multi-threaded Floating Point Unit (FPU) (optional)
- Enhanced virtual addressing (EVA) mode allows for up to 3.0 GB of user or kernel virtual address space
- Integrated integer Multiply/Divide Unit (MDU)
- Programmable Memory Management Unit (MMU)
  - L1 Cache sizes of 4/8/16/32/64 KB, 4-Way Set Associative
  - 16/32/48/64 dual-entry JTLB per VPE
  - 8-entry DTLB
  - 4 - 12 entry ITLB
  - L1 MESI coherent cache states
  - 32-byte cache line size
  - 64-bit data an 32-bit address paths to caches
  - Virtually indexed, physically tagged
  - Parity or ECC support on L1 Dcache and DSPRAM
  - Parity support on L1 Icache and ISPRAM
  - Supports up to 1MB of instruction and data Scratchpad RAM (optional)
- Memory Protection Unit (MPU) (optional)
  - Read/Write registers with reset values set at build time
  - Configurable number of regions: 8-32 per core, in multiples of 4
  - Region size: 32B-4GB
- MIPS32 Release3 Instruction Set and Privileged Resource Architecture
- MIPS16e™ Code Compression (optional)
- MIPS16e2 Application-Specific Extension — instructions to improve code density and performance
- MIPS MT Application Specific Extension (ASE)
  - Support for 1 or 2 Virtual Processing Elements (VPEs)
  - Supports 1 - 9 thread contexts per core
  - Inter-Thread Communication (ITC) memory for efficient communication & data transfer
- MIPS DSP ASE (optional)
  - 3 additional pairs of accumulator registers
  - Fractional data types, Saturating arithmetic
  - SIMD instructions operate on 2x16b or 4x8b simultaneously

- CorExtend® MIPS32 compatible User Defined Instruction Set Extensions allows user to define and add instructions to the core at build time

interAptiv™ CPU Core

Figure 2 interAptiv™ CPU Core Block Diagram

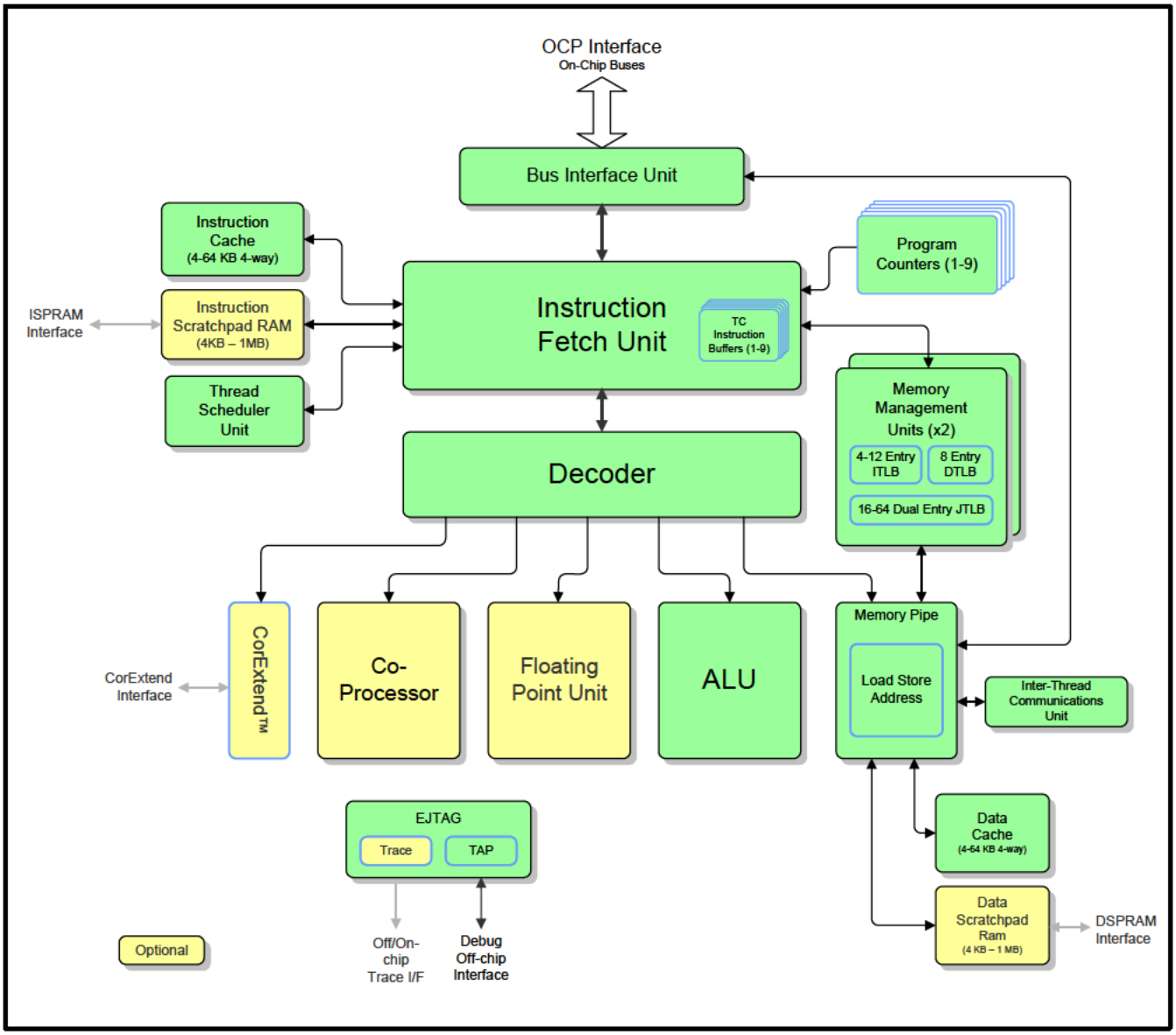


Figure 2 shows a block diagram of a single interAptiv core. The following subsections describe the logic blocks in this diagram. For more information on the interAptiv core in a multiprocessing environment, refer to the section "Multiprocessing System" on page 10.

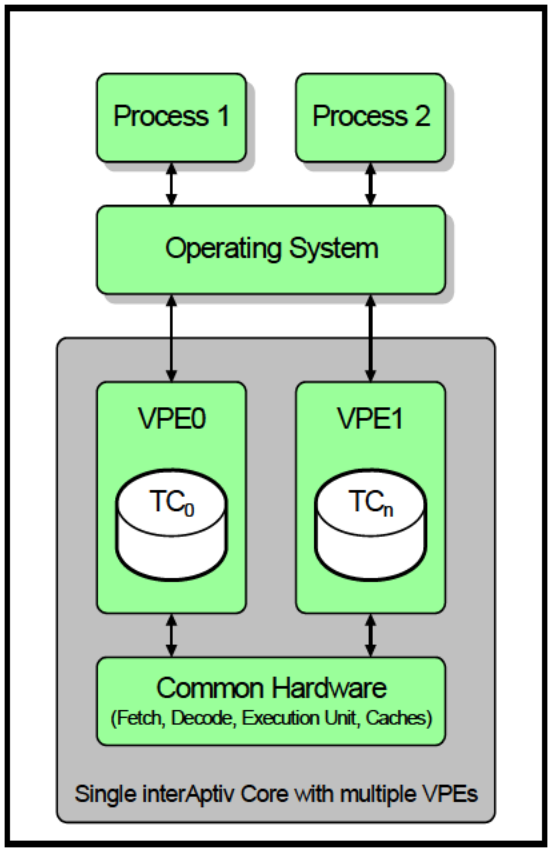
MIPS Release 3 Architecture

In addition to the base architecture, the interAptiv core supports the MIPS MT ASE that defines multi-threaded operation, the MIPS16e2 ASE for code compression, and the MIPS DSPr2 ASE for accelerating integer SIMD codes.

MIPS Multi-Thread Technology

Building on the prior generation of MIPS multi-threaded (MT) processors, the interAptiv core also implements the same multi-threaded architecture and supports the Application Specific Extensions (MT ASE) which are based on a two-layered framework involving Virtual Processing Elements (VPEs) and Thread Contexts (TCs). Each interAptiv core can support up to two VPEs which share a single pipeline among other hardware resources. Because each VPE includes a complete copy of the processor state as seen by the software system, each VPE appears as a complete standalone processor to an SMP Linux operating system. For more fine-grained thread processing applications, each VPE is capable of supporting multiple TCs. The TCs share a common execution unit but each has its own program counter and core register files so that each can handle a thread from the software. The interAptiv core can support up to nine TCs allocated across two VPEs, optimized and partitioned at run time. Figure 3 shows the relationship of the OS, VPEs, TCs, and the common hardware in the interAptiv core.

Figure 3 Single interAptiv Core with VPE's



In addition to supporting the MT ASE, the core also supports the MIPS16e2 ASE for code compression, and the MIPS DSPr2 ASE for accelerating integer SIMD codes.

Instruction Fetch Unit

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

In order to fetch instructions without intervention from the execution unit, the fetch unit contains branch prediction logic. A 512-entry Branch History Table (BHT) is used to predict the direction of branch instructions, a 4-entry Return Prediction Stack (RPS) holds 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. The address is popped when a JR instruction occurs.

Thread Schedule Unit (TSU)

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

Policy Manager

The policy manager is a configurable block. Simple round-robin or fixed priority policies can be selected during design. The interAptiv includes a reference policy manager that implements a weighted round-robin algorithm for long-term distribution of execution bandwidth.

Execution Unit

The interAptiv CPU execution unit implements a load/store architecture with single-cycle ALU operations (logical, shift, add, subtract) and an autonomous multiply/divide unit. Each TC on a interAptiv CPU contains thirty-one 32-bit general-purpose registers used for integer operations and address calculation. Additional sets of shadow register files can be added to be dedicated for interrupt and 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 bit-wise logical operations
- Shifter and Store Aligner

MIPS16e™ Application Specific Extension

The interAptiv CPU includes support for the MIPS16e and MIPS16e2 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/tear-down for efficient subroutine entry/exit. MIPS16e2 adds more instructions to MIPS16e to further enhance the compression as well as the performance of compressed code.

Multiply/Divide Unit (MDU)

The multiply/divide unit (MDU) contains a separate pipeline for integer multiply and divide operations. This pipeline operates in parallel with the integer unit pipeline.

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 or accumulate operation every clock cycle. Divide can be executed as fast as one every six cycles.

Floating Point Unit (FPU) (optional)

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

The FPU can be configured at build time to run at either the same or one-half the clock rate of the integer CPU. The FPU is not as deeply pipelined as the integer CPU, so the maximum CPU frequency will only be attained with the FPU running at one-half the CPU frequency.

FPU Performance

FPU performance is optimized for single precision formats. Most instructions have one FPU cycle throughput. 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 greatly 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 pipeline. 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 dispatched and completed in order, loads and stores can complete out of order.

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.

System Control Coprocessor (CP0)

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

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

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

Interrupt Handling

Each interAptiv VPE includes support for six hardware interrupt pins, two software interrupts, a timer interrupt, and a performance counter interrupt. These interrupts can be used in the following interrupt modes:

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

Modes of Operation

The interAptiv 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 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 16 for more information on debug mode.

Memory Management Options

The interAptiv uses a TLB-based MMU, or a simpler Memory Protection Unit (MPU). If the TLB option is selected you can choose the number of dual entries in the main TLB array.

Each interAptiv VPE contains a Memory Management Unit (MMU) that primarily converts virtual addresses to physical addresses and provides attribute information for different segments of memory. In a dual-TLB configuration, each VPE contains a separate JTLB, so that translations for each VPE are independent from the other VPE.

The interAptiv core also can be configured with an optional Memory Protection Unit (MPU). The MPU provides for finer-grain protection of memory pages relative to an MMU and also eliminates the address translation mechanism for space-sensitive applications. The MPU provides memory access control at both the segment and the region level.

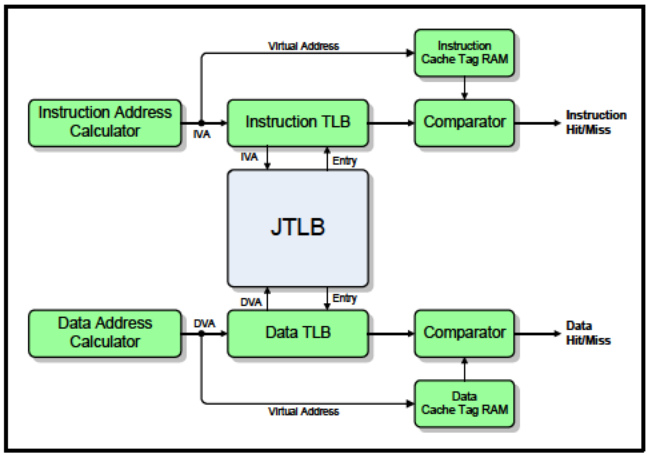
Translation Lookaside Buffer (TLB)

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

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 interAptiv CPU.

Figure 4 interAptiv Core Address Translation



Joint TLB (JTLB)

The JTLB is a 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) 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. Because page sizes 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 contains between 4 and 12 entries and is dedicated to performing translations for the instruction stream. The ITLB is a hybrid structure having 3 entries that are shared by all TCs plus an additional entry dedicated to each TC. Therefore, a single core system with one VPE and one TC would have a 4-entry TLB with all entries dedicated to one core. Conversely, a single core system with 1 VPE and 9 TC's would have a three shared entries, plus one entry per TC, for a total of 12 entries.

The ITLB 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 translate it.

Data TLB (DTLB)

The DTLB is an 8-entry, fully associative TLB dedicated to performing translations for loads and stores. All entries are shared by all TCs. Similar to the ITLB, the DTLB 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. The JTLB translation information is copied into the DTLB for future use.

Enhanced Virtual Address

The interAptiv core contains a programmable memory segmentation scheme called Enhanced Virtual Address (EVA), which allows for more efficient use of 32-bit address space. Traditional MIPS virtual memory support divides up the virtual address space into fixed segments, each with fixed attributes and access privileges. Such a scheme limits the amount of physical memory available to 0.5GB, the size of kernel segment 0 (kseg0).

In EVA, the size of virtual address space segments can be programmed, as can their attributes and privilege access. With this ability to overlap access modes, kseg0 can now be extended up to 3.0GB, leaving at least one 1.0GB segment for mapped kernel accesses. This extended kseg0 is called xkseg0. This space overlaps with useg, because segments in xkseg0 are programmed to support mapped user accesses and unmapped kernel accesses. Consequently, user space is equal to the size of xkseg0, which can be up to 3.0GB. This concept is shown in Figure 5.

Figure 5 Enhanced Virtual Addres

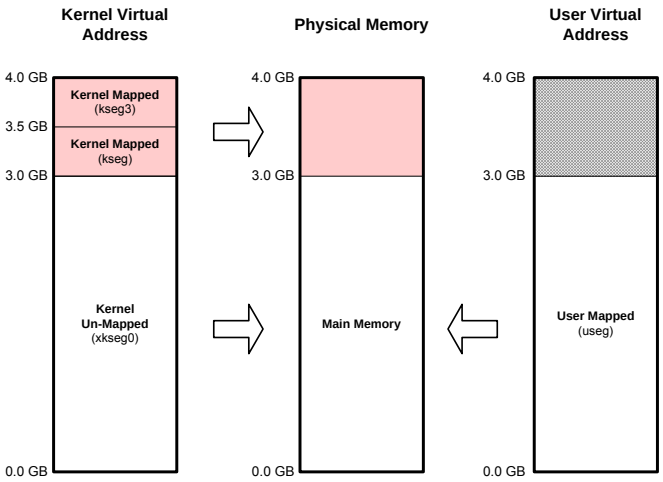


Figure 5 shows an example of how the traditional MIPS kernel virtual address space can be remapped using programmable memory segmentation to facilitate an extended virtual address space. As a result of defining the larger kernel segment as xkseg0, the kernel has unmapped access to the lower 3GB of the virtual address space. This allows for a total of 3GB of DRAM to be supported in the system.

To allow for efficient kernel access to user space, new load and store instructions have been defined which allow kernel mapped access to useg.

Note that the attributes of xkseg0 are the same as the previous kseg0 space in that it is a kernel unmapped, uncached region.

Level 1 Data Cache

The Level 1 (L1) data cache is an on-chip memory block of 4/8/16/32/64 KB, with 4-way associativity. A tag entry holds 22 bits of physical address, two cache state bits, and an optional parity bit. The data entry holds 64 bits of data per way, with optional parity or ECC per byte. There are 4 data entries for each tag entry. The tag and data entries exist for each way of the cache. The way-select array holds the dirty and LRU replacement algorithm bits for all 4 ways (6-bit LRU, 4-bit dirty, and optionally 4-bit dirty parity or 12 bits of ECC).

Virtual aliasing can occur when using 4KB pages in the TLB and 32 or 64KB cache sizes. Because it is quite challenging for software to manage virtual aliases across multiple devices, these larger cache arrays are banked on the aliased 1 or 2 physical address bits to eliminate the virtual aliases.

The interAptiv CPU supports data-cache locking. Cache locking allows critical code or data segments to be locked into the cache on a “per-line” basis, enabling the system programmer to maximize the efficiency of the system cache. The locked contents can be updated on a store hit, but are not selected for replacement on a cache miss. Locked lines do not participate in the coherence scheme so processes which lock lines into a particular cache should be locked to that processor and prevented from migrating.

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.

The data cache supports ECC dual-bit error detection and single-bit error correction.

Level 1 Instruction Cache

The Level 1 (L1) instruction cache is an on-chip memory block of 4/8/16/32/64 KB**,** with 4-way associativity. A tag entry holds 22 bits of physical address, a valid bit, a lock bit, and an optional parity bit. The instruction data entry holds two instructions (64 bits), 6 bits of pre-decode information to speed the decode of branch and jump instructions, and 9 optional parity bits (one per data byte plus one more for the pre-decode information). The instruction cache does not support ECC. There are four data entries for each tag entry. The tag and data entries exist for each way of the cache. The LRU replacement bits (6-bit) are shared among the 4 ways and are stored in a separate array.

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

The interAptiv CPU 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.

Level 1 Cache Memory Configuration

The interAptiv CPU incorporates on-chip L1 instruction and data caches that are typically 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 1 lists the interAptiv CPU instruction and data cache attributes.

Table 1 interAptiv™ CPU L1 Instruction and Data Cache Attributes

Parameter	Instruction	Data
Size[1]	4, 8, 16, 32, or 64 KB	4, 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	coherent and non-coherent write-back with write allocate
Miss restart after transfer of	miss word	miss word
Cache Locking	per line	per line

1. For Linux based applications, MIPS recommends a 64 KB L1 cache size, with a minimum size of 32 KB.

Instruction and Data Scratchpad RAM

The interAptiv core 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. The size of both the instruction scratch pad RAM (ISPRAM) and data scratch pad RAM (DSPRAM) can be configured from a range of 4 KB to 1 MB. These RAM's are used for the temporary storage of information and can be modified by the user at any time. The ISPRAM supports ECC error detection/correction and parity bit.

InterThread Communication Unit (ITU)

This block provides a mechanism for efficient communication between TCs. This block has a number of locations that can be used as mailboxes, FIFO mailboxes, mutexes, and semaphores.

Bus Interface (BIU)

The Bus Interface Unit (BIU) controls a 64-bit interface to the CM2. The interface implements the industry standard "Open Core Protocol" (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 eight, 32-byte buffers. Each buffer can contain data from a single 32-byte aligned block of memory. When using the write-through cache policy or performing uncached accelerated writes, the write buffer significantly reduces the number of write transactions on the external interface and reduces the amount of stalling in the core caused by the issuance of multiple writes in a short period of time.

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

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

Gathering of uncached accelerated stores can start on any arbitrary address and can be combined in any order within a cache line. Uncached accelerated stores that do not meet the conditions required to start gathering are treated like regular uncached stores.

SimpleBE Mode

To aid in attaching the interAptiv core to structures that cannot easily handle arbitrary byte-enable patterns, there is a mode that generates only "simple" byte enables. In this mode, only byte enables representing naturally aligned byte, halfword, word, and doubleword transactions is generated.

In SimpleBE mode, the SI_SimpleBE input pin only controls the byte enables generated by the interAptiv core(s). It has no effect on byte enables produced by the IOCU. To achieve the effect of setting SI_SimpleBE to 'one' in systems with an IOCU, the I/O sub-system must only issue requests to the IOCU with naturally aligned byte enables.

When the SI_SimpleBE input signal to the interAptiv core is asserted, hardware sets bit 21 of the config register (Config.SB) to indicate the device is in simple byte enable mode.

Interrupt Handling

The interAptiv core supports six hardware interrupts, two software interrupts, a timer interrupt, and a performance counter interrupt. These interrupts can be used in any of three interrupt modes.

- *Interrupt compatibility mode*: acts identically to that in an implementation of Release 1 of the Architecture
- *Vectored Interrupt (VI) mode*: adds the ability to prioritize and vector interrupts to a handler dedicated to that interrupt
- *External Interrupt Controller (EIC) mode*: provides support for an external interrupt controller that handles prioritization and vectoring of interrupts

Modes of Operation

The interAptiv core supports four modes of operation:

- *User mode*: most often used for application programs
- *Supervisor mode*: provides an intermediate privilege level with access to the ksseg (kernel supervisor segment) address space
- *Kernel mode*: used for handling exceptions and operating system kernel functions, including CP0 management and I/O device accesses
- *Debug mode*: used during system bring-up and software development. Refer to section "EJTAG Debug Support" on page 16 for more information on debug mode

Multiprocessing System

The interAptiv Multiprocessing System (MPS) consists of the logic modules shown on page 1. Each of these blocks is described throughout this section.

Cluster Power Controller (CPC)

The Cluster Power Controller (CPC) has two versions: Standard CPC and CPC Basic. The CPC Basic has less functionality, but instantiates all signals and each design only uses the signals that it requires.

CPC Standard — 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 CPC. The CPC handles the power shutdown and ramp-up of all CPUs in the cluster. Any interAptiv CPU that supports power-gating features is managed by the CPC.

The CPC Standard also organizes power-cycling of the CM2 dependent on the individual core status and shut-down policy. Reset and root-level clock gating of individual CPUs are considered part of this sequencing.

The CPC standard allows power up to be controlled via HW signals and SW commands. After power-up the power state transition is controlled via SW commands.

CPC-basic—the primary operating mode of the CPC-basic allows powering up/down via HW signals. However, the CPC can be placed in a special debug mode, which allows the power states to be controlled via SW (similar to the CPC standard).

Also, the CPC-Standard operates at the same clock rate as the Cores and CM2, but the CPC-basic uses its own clock input, allowing it to run at any frequency.

Coherence Manager (CM2)

The Coherence Manager with integrated L2 cache (CM2) 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 CM2 is shown in Figure 6. Each of the blocks is described in more detail in the following subsections.

Request Unit (RQU)

The Request Unit (RQU) receives OCP bus transactions from multiple CPU cores and/or I/O ports, serializes the transactions and routes them to the Intervention Unit (IVU), Transaction Routing Unit (TRU), or an auxiliary port used to access a configuration registers or memory-mapped IO. The routing is based on the transaction type, the transaction address, and the CM2's programmable address map.

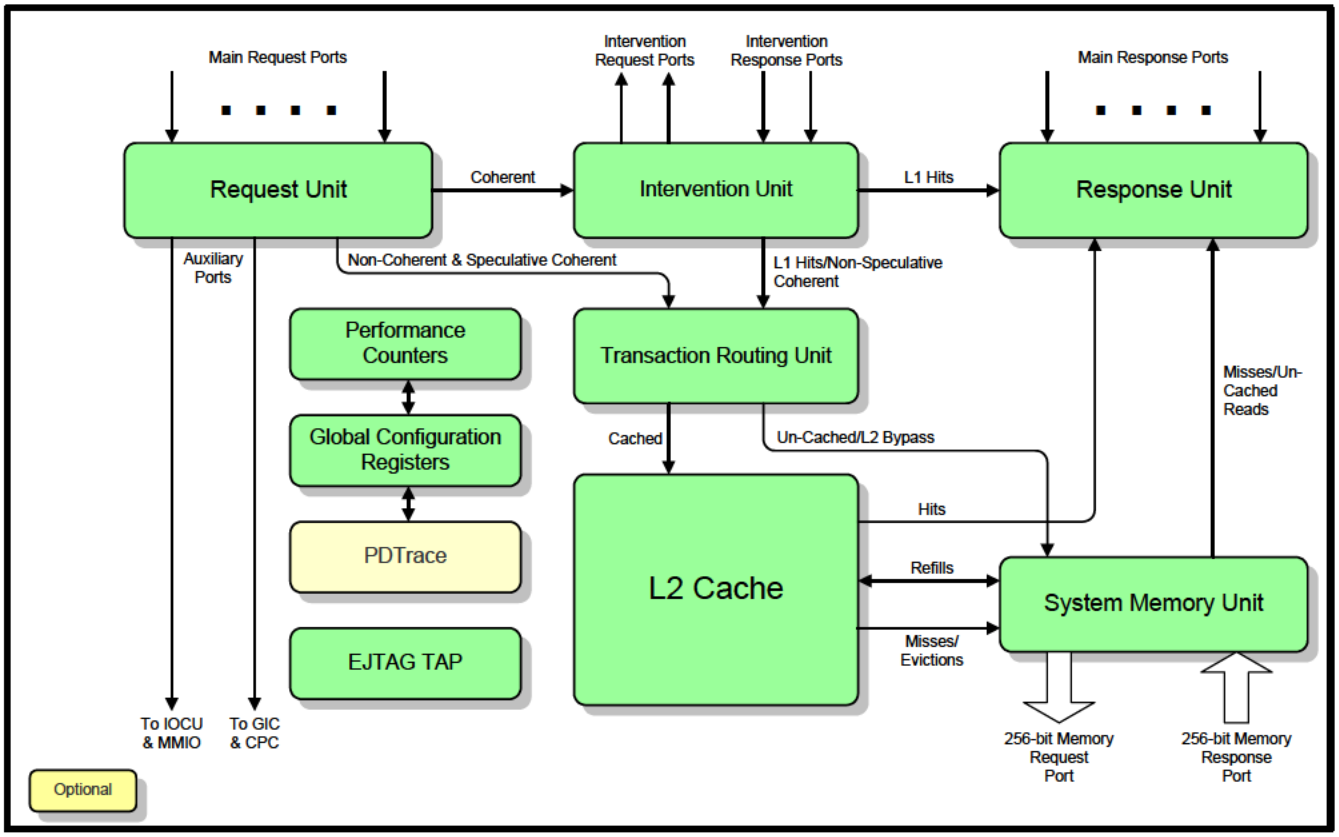
Intervention Unit (IVU)

The Intervention Unit (IVU) interrogates the L1 caches by placing requests on the intervention OCP interfaces. Each processor responds with the state of the corresponding cache line. For most transactions, if a CPU core has the line in the MODIFIED or EXCLUSIVE states, it provides the data with its response. If the original request was a read, the IVU routes the data to the original requestor via the Response Unit (RSU). For the MESI protocol, intervention data may also be routed to the L2/Memory via the TRU (implicit writeback).

Priority Arbiter

The CM2 may be configured with a priority arbiter, which allows IOCUs and Cores to be assigned high or low priority via a programmable GCR. Requests from high-priority agents are generally serialized ahead of requests from low-priority agents. However, a timeout value can be programmed to ensure that low-priority agents are provided some minimum bandwidth.

Figure 6 Coherence Manager with Integrated L2 Cache (CM2) Block Diagram



The IVU 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 speculative are issued to the Memory Interface Unit (MIU)
- Modified data returned from the CPU is sent to the MIU to be written back to memory
- Data returned from the CPU is forwarded to the Response Unit (RSU) 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

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. Note that the IVU is not included in non-coherent configurations, such as a single core without an IOCU.

System Memory Unit (SMU)

The System Memory Unit (SMU) provides the interface to the memory OCP port. For an L2 refill, the SMU reads the data from an internal buffer and issues the refill request to the L2 pipeline.

Note that the external interface may operate at a lower frequency than the Coherence Manager (CM2), and the external block may not be able to accept as many requests as multiple CPUs can generate, so some buffering of requests may be required.

Response Unit (RSU)

The RSU takes responses from the SMU, L2, IVU, or auxiliary port and places them on the appropriate OCP interface. Data from the L2 or SMU is buffered inside a buffer associated with each RSU port, which is an enhancement over the previous generation Coherence Manager.

When a coherent read receives an intervention hit in the MODIFIED or EXCLUSIVE state, the Intervention Unit (IVU) provides the data to the RSU. The RSU then returns the data to the requesting core.

Transaction Routing Unit

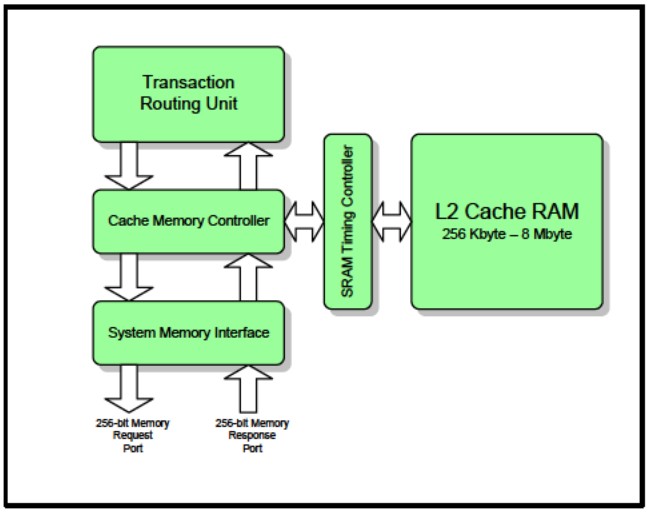
The Transaction Routing Unit (TRU) arbitrates between requests from the RQU and IVU, and routes requests to either the L2 or the SMU. The TRU also contains the request and intervention data buffers which are written directly from the RQU and IVU, respectively. The TRU reads the appropriate write buffer when it processes the corresponding write request.

Level 2 Cache

The unified Level 2 (L2) cache holds both instruction and data references and contains a 7-stage pipeline to achieve high frequencies with low power while using commercially available SRAM generators.

Cache read misses are non-blocking; that is, the L2 can continue to process cache accesses while up to 15 misses are outstanding. The cache is physically indexed and physical tagged. shows a block diagram of the L2 cache.

Figure 7 L2 Cache Block Diagram



L2 Cache Configuration

The L2 cache in the CM2 can be configured as follows:

- 0, 32, 64, 128, 256, 512, 1024, 2048, 4096, or 8192 KBytes. The 32K and 64K L2 caches are only supported for configurations with 64B cache lines, and no ECC.
- 32- or 64-byte line size
- 8 ways
- 512 to 16384 sets per way (in powers of two)

L2 Pipeline Tasks

The L2 pipeline manages the flow of data to and from the L2 cache. The L2 pipeline performs the following tasks:

- Accesses the tags and data RAMs located in the memory block (MEM)
- Returns data to the RSU for cache hits
- Issues L2 miss requests
- Issues L2 write and eviction requests
- Returns L2 write data to the SMU. The SMU issues refill requests to the L2 for installation of data for L2 allocations

L2 Cache Features

- Supports write-back operation
- Pseudo-LRU replacement algorithm
- Programmable wait state generator to accommodate a wide variety of SRAMs
- Operates at same clock frequency as CPU
- Cache line locking support
- Optional ECC support for resilience to soft errors
- Single bit error correction and 2 bit error detection support for Tag and Data arrays
- Single bit detection only for WS array
- Bypass mode
- Fully static design: minimum frequency is 0MHz
- Sleep mode
- Support for extensive use of fine-grained clock gating
- Optional memory BIST for internal SRAM arrays, with support for integrated (March C+, IFA-13) or custom BIST controller

CM2 Configuration Registers

The Registers block (GCR) contains the control and status registers for the CM2. It also contains the Trace Funnel, EJTAG TAP state machine, and other multi-core features.

PDTrace Unit

The CM2 PDTrace Unit (PDT) is an optional unit used to collect, pack and send out CM2 debug information.

Performance Counter Unit

The CM Performance Counter Unit (PERF) implements the performance counter logic.

Coherence Manager Performance

The CM2 has a number of high performance features:

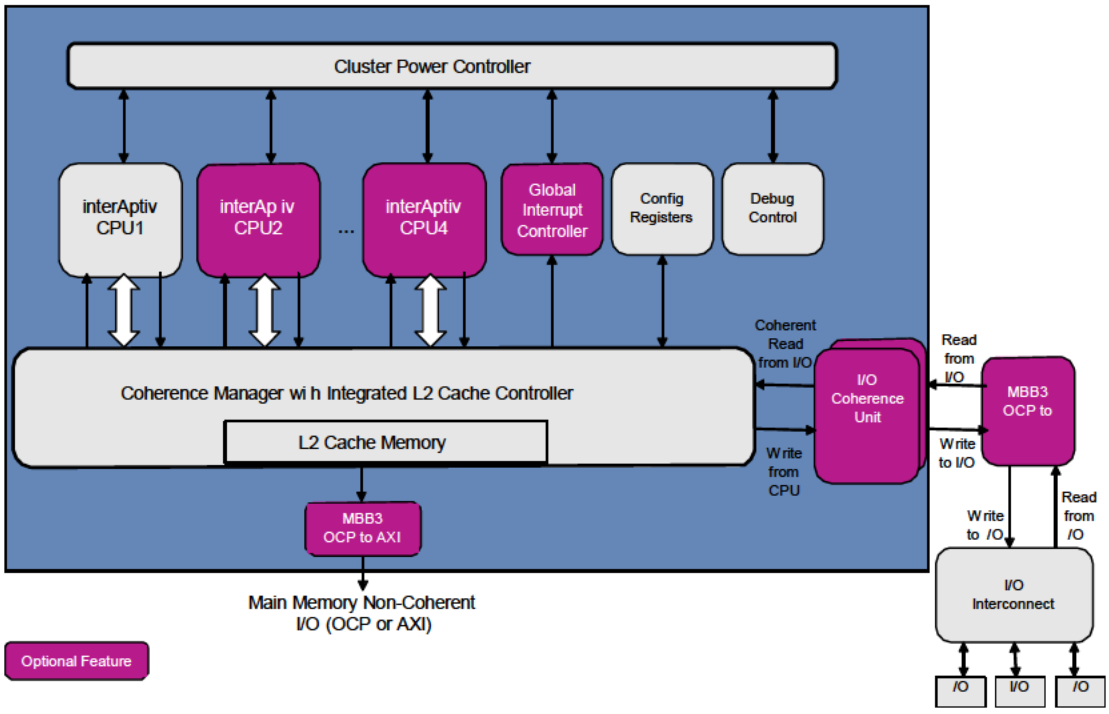
- 64-bit wide internal data paths throughout the CM2
- 64-, 128-, or 256-bit wide system OCP interface
- Cache to Cache transfers: If a read request hits in another L1 cache in the EXCLUSIVE or MODIFIED state, it will return the data to the CM and it will be forwarded to the requesting CPU, thus reducing 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 CM2 cancels the speculative request—dropping the request if it has not been issued, or dropping the memory response if it has.

I/O Coherence Unit (IOCU) (optional)

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. Up to 2 IOCUs are supported in an interAptiv MPS.

The IOCU acts as an interface block between the Coherence Manager 2 (CM2) 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.

Figure 8 Role of a Single IOCU in a Two-Core Multiprocessing System



The IOCU also provides a legacy (without coherent extensions) OCP or AXI slave interface to the I/O interconnect for I/O devices to read and write system memory. The reference design also includes an OCP or AXI 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
- The standard mapping units allow these cache attributes to be determined by signals driven with the request.
- Cacheable or Coherent Reads or Writes of any size may be allocated in the L2 cache.
- Supports incremental bursts up to 16 beats (64 bits) on the 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 IOCU contains the following features used to enforce transaction ordering.

- Set-aside buffer: This buffer can delay read responses from the I/O device until Previous I/O writes have completed
- Writes are issued to the CM in the order they were received
- The CM provides an acknowledge (ACK) signal to the IOCU when writes are “visible” (guaranteed that a subsequent CPU read will receive that data)
  - Non-coherent write is acknowledged after serialization
  - Coherent write is acknowledged 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

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 CM2 sends 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.

Global Interrupt Controller

The Global Interrupt Controller (GIC) handles the distribution of interrupts between and among the CPUs in the cluster. This block has the following features:

- Software interface through can be relocated throughout the 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
- Support for NMI routing
- Standardized mechanism for sending inter-processor interrupts

No-Global Interrupt Controller

At configuration time, you can also select the option to not instantiate the GIC within your cluster. When this option is selected, the interrupt-related signals appear as pins on the mips_soc module so you can use these pins or connect your own external Interrupt Controller.

Global Configuration Registers (GCR)

The Global Configuration Registers (GCR) are 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 of the system resets the Cluster Power Controller (CPC). Reset sideband signals are required to qualify a reset as system cold, or warm start. Register setting determine the course of action:

- Remain in powered-down
- Go into clock-off mode
- Power-up and start execution

This prevents random power up of power domains before the CPC is properly initialized. In case of a system cold start, after reset is released, the CPC powers up the interAptiv CPUs as directed in the CPC cold start configuration. If at least one CPU has been chosen to be powered up on system cold start, the CM2 is also powered up.

When supply rail conditions of power gated CPUs have reached a nominal level, the CPC will enable clocks and schedule reset sequences for those CPUs and the coherence manager.

At a warm start reset, the CPC brings all power domains into their cold start configuration. However, to ensure power integrity for all domains, the CPC ensures that domain isolation is raised before power is gated off. Domains that were previously powered and are configured to power up at cold start remain powered and go through a reset sequence.

Within a warm start reset, sideband signals are also used to qualify if coherence manager status registers and GIC watch dog timers are to be reset or remain unchanged. The CPC, after power up of any CPU, provides a test logic reset sequence per domain to initialize TAP and PDTrace logic.

In addition to controlling the deassertion of the CPC reset signal, there are memory-mapped registers that can set the reset exception address for each CPU. This allows different boot vectors to be specified for each of the cores so they can execute unique code if required. Each of the cores will have a unique CPU number, so it is also possible to use the same boot vector and branch based on that.

Inter-CPU Debug Breaks

The CPS includes registers that enable cooperative debugging across all CPUs. Each core features an 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.

CM2 Control Registers

Control registers in the CM2 allow software to configure and control various aspects of the operation of the CM2. 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 CM2 and controls how errors are handled (ignored, interrupt, and so on)
- *Control Options*: Various features of the CM2 can be disabled or configured. Examples of this are disabling speculative reads and preventing Read/Shared requests from being upgraded to Exclusive

MIPS® BusBridge™ 3 with AXI

The MIPS® BusBridge™ 3 Module (MBB3) provides high-performance OCP 2.1 interfaces between a MIPS core and other subsystems in an SOC. The MBB3 includes the following blocks:

- OCP-to-AXI Bridge (OCP-AXI) provides a bridge between an OCP 2.1 master an AMBA AXI slave
- AXI-to-OCP Bridge (AXI-OCP) provides a bridge between AMBA AXI master and OCP 2.1 compliant slave. It facilitates connection of an AMBA AXI based subsystem to the OCP slave interfaces of the I/O Coherence Unit (IOCU)

For additional details, refer to the “MIPS BusBridge3 Modules User’s Manual”.

EJTAG Debug Support

The interAptiv CPU 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 interAptiv CPU provides a Debug mode that is entered after a debug exception (derived from a hardware breakpoint, single-step exception, and so on.) 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 interAptiv CPU. In addition to the standard JTAG instructions, special instructions defined in the EJTAG specification define what registers are selected and how they are used.

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 interAptiv CPU: Instruction breakpoints and Data breakpoints.

During synthesis, the interAptiv CPU can be configured to support the following breakpoint options per VPE:

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

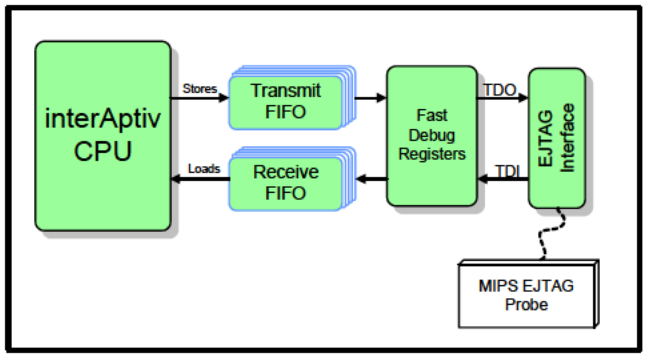
Instruction breaks occur on instruction fetch operations, and the break is set on the virtual address. Instruction breaks can also be made on the ASID value used by the MMU. A mask can be applied to the virtual address to set breakpoints on a range of instructions.

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

Fast Debug Channel

The interAptiv 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 9 Fast Debug Channel



MIPS Trace (optional)

The interAptiv CPU includes MIPS Trace support for real-time tracing of instruction addresses, data addresses and data values. 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. On-chip trace memory may be configured in size from 0 to 1MB; it is accessed through the existing EJTAG TAP interface and requires no additional chip pins. Off-chip trace memory is accessed through a special trace probe and can be configured to use 4, 8, or 16 data pins and a clock.

Clocking Options

The interAptiv core has the following clock domains:

- *Cluster domain*: This is the main clock domain, and includes all interAptiv cores (including optional FP2) and the CM2 (including Coherence Manager, Global Interrupt Controller, Cluster Power Controller, trace funnel, IOCU, and L2 cache)
- *System Domain*: The OCP port connecting to the SOC and the rest of the memory subsystem may operate at a ratio of the cluster domain. Supported ratios are 1:1, 1:2, 1:3, 1:4, 1:5, and 1:10
- *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*
- *IO Domain*: This is the OCP port connecting the IOCU to the I/O Subsystem. This clock may operate at a ratio of the CM2 domain. Supported ratios are the same as the system domain

Figure 10 shows a diagram with the four clock domains.

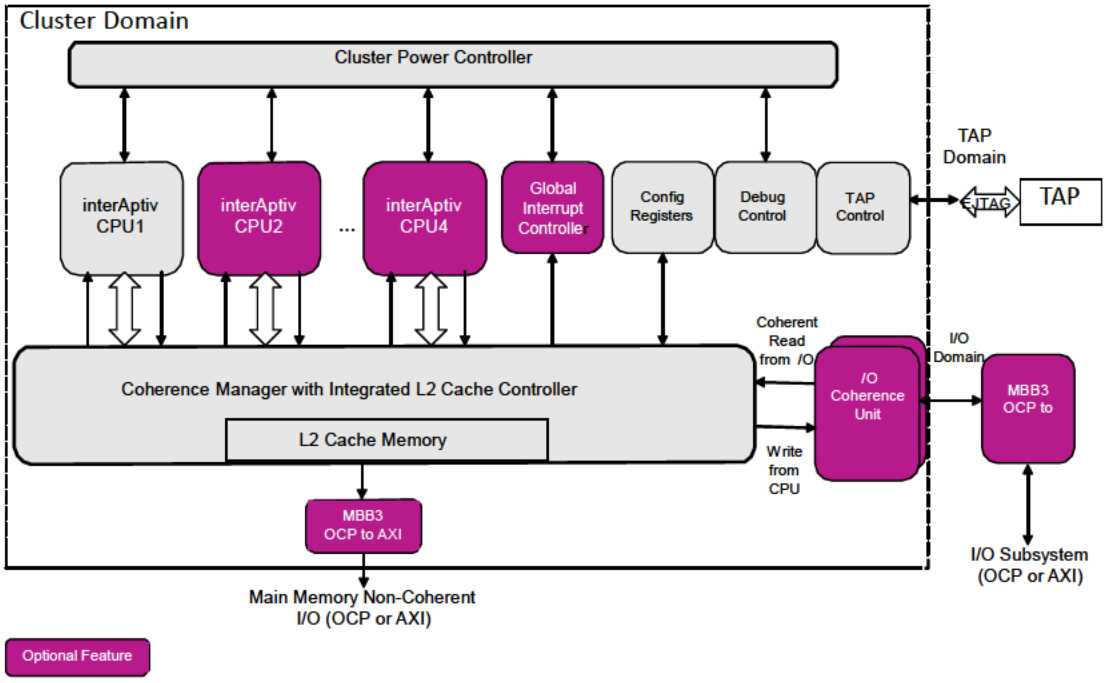
Design For Test (DFT) Features

The interAptiv core provides the following test for determining the integrity of the core.

- *Internal Scan*: The interAptiv core 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 interAptiv core 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 core to facilitate this approach.

Figure 10 interAptiv CPS Clocking Domains



Build-Time Configuration Options

The interAptiv Coherent Processing System allows some features to be customized based on the intended application. Table 2 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® interAptiv™ Processor 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 2 interAptiv Build-time Configuration Options

Option	Choices	Software Visibility
System Options		
Number of CPUs	interAptiv Dual: 2-core (configured as 1 or 2) interAptiv Quad: 4-core (configured as 1, 2, 3, or 4)	*GCR_CONFIG$_{PCORES}$*
MIPS Trace support	Present or not	*Config3$_{TL}$*
MIPS Trace memory location	On-chip, off-chip, or both	*TCBCONFIG$_{OnT}$*, *TCBCONFIG$_{OfT}$*
Clock Generator	• Standard clock generator (for simulation purposes, supports integer and fractional ratios) • 1:1 clock generator (integer OCP ratios, used for synthesis) • Custom clock generator	N/A

Table 2 interAptiv Build-time Configuration Options (Continued)

Option	Choices	Software Visibility
CPU Options		
Number of VPEs	1 or 2	*MVPConf0*$_{PVPE}$
Number of threads per VPE	1 - 9	*MVPConf0*$_{PTC}$
TLB size (per VPE)	16, 32, 48, or 64 dual entries	*Config1*$_{MMUSize}$
MMU type	MPU / TLB	N/A
Number of MPU regions	8 - 32, by 4	*MPU_Config.NumEntries*
TLB data array implementation style	Flops or generator	N/A
Integer register file implementation style	Flops or generator	N/A
Enable FPU	Yes/No	*Config1*$_{FP}$
Enable MT FPU support	Yes/No	*MVPConf1*$_{PCP1}$
FPU clock ratio relative to integer CPU	1:1 or 1:2	*Config7*$_{FPR}$
Number of outstanding data cache misses	4 or 8	N/A
Number of outstanding loads	4 or 9	N/A
Power gating	Enabled or not	N/A
Clock gating	Enabled or not	N/A
PrID company option	0x0 - 0x7F	PrID$_{CompanyOption}$
I-cache size	4, 8, 16, 32, 64 KB	*Config1*$_{IL}$, *Config1*$_{IS}$, *Config1*$_{IA}$
Instruction ScratchPad RAM interface	Present or not	*Config*$_{ISP}$
Instruction ScratchPad RAM size	4 - 1024 KB in powers of 2	N/A
D-cache size	4, 8, 16, 32, or 64 KB	*Config1*$_{DL}$, *Config1*$_{DS}$, *Config1*$_{DA}$
Data Scratch Pad RAM interface	Present or not	*Config*$_{DSP}$
Data Scratch Pad RAM size	4 - 1024 KB in powers of 2	N/A
L1 cache parity/ECC support	• No parity on instruction or data caches • Parity on instruction, parity on data • Parity on instruction, ECC on data	*ErrCtl*$_{PE,}$ *ErrCtl*$_{L1ECC}$
Number of breakpoints (per VPE)	• None • 2 instr, 1 data • 4 instr, 2 data	*DCR*$_{IB}$, *DCR*$_{DB}$, *DBS*$_{BCN}$, *IBS*$_{BCN}$
Breakpoint on VPE	• VPE 0 only • VPE 1 only • Both	*DCR*$_{IB}$, *DCR*$_{DB}$, *DBS*$_{BCN}$, *IBS*$_{BCN}$
Fast Debug FIFO Sizes	Minimum (2 transmit, 2 receive) Typical (12 transmit, 4 receive)	*FDCFG*

Table 2 interAptiv Build-time Configuration Options (Continued)

Option	Choices	Software Visibility
Number of Trace Control Block (TCB) Triggers	0 - 8	N/A
CorExtend	Present or not	$Status_{CCE}$
Coprocessor2 interface	Present or not	$Config1_{C2}$
Yield Manager	Standard or custom	N/A
External Policy Manager	• Round-robin • Weighted round-robin (2 types) • Custom	N/A
Clock Gating	See general options section below	N/A
Memory BIST	See general options section below	N/A
Coherence Manager Options		
Number of Address Regions	• 4 standard only • 4 standard + 2 attribute • 4 standard + 4 attribute	$GCR\_CONFIG_{NUM\_ADDR\_REGIONS}$
Default GCR base address and writeability	Any 32KB-aligned physical address Hardwired or programmable	*GCR_BASE*
Base GCR address set by software	Yes/No	N/A
Boot in EVA mode	Yes/No	$GCR\_Cx\_RESET\_EXT\_BASE_{EVAReset}$
Use legacy exception vector boot	Yes/No	$GCR\_Cx\_RESET\_EXT\_BASE_{Legacy\ UseExceptionBase}$
Default Exception Base for each CPU	Any 4KB-aligned physical address	*GCR_Cx_RESET_BASE*
Boot exception vector overlay region size	1 MB to 256 MB	$GCR\_Cx\_RESET\_EXT\_BASE_{BEV\ ExceptionBaseMask}$
Number of relay stages between cores and CM2 (per core)	0, 1, or 2	N/A
Custom GCR register block	Enabled or not	$GCR\_CUSTOM\_STATUS_{GGU\_EX}$
External interrupt controller (EIC)	Present or not	$CONFIG3_{VEIC}$
Boot in EIC mode	Yes/No	$CONFIG3_{VEIC}$
MIPS Trace on-chip memory size	256B - 1MB	$TCBCONFIG_{SZ}$
Probe Interface Block	Present, Not Present, or Custom	N/A
Probe Interface Block number of data pins	4, 8, 16	N/A
Global Interrupt Controller Options		
Include Internal GIC	Yes (default) / No (GIC not included)	*CM_GCR_GIC_STATUS.GIC_EX*
Number of system interrupts	8 - 256 in multiples of 8	$GIC\_SH\_CONFIG_{NUMINTERRUPTS}$

Table 2 interAptiv Build-time Configuration Options (Continued)

Option	Choices	Software Visibility
Local routing of CPU sourced interrupts (per VPE)	Present or not	N/A
ITU Options		
Select ITU	Yes/No	N/A
Number of single entry mailboxes	0, 1, 2, 4, 8, or 16	*ITCAddressMap1*$_{NumEntries}$
Number of 4 entry FIFOs	0, 1, 2, 4, 8, or 16	
Cluster Power Controller Options		
CPC Version	Standard (default) / CPC Basic	*CPC_REVISION_REG.MAJOR_REV[7:6]*
Microstep delay in cycles	1 - 1024	*CPC_SEQDEL_REG.MICROSTEP*
RailEnable delay	1 - 1024	*CPC_RAIL_REG.RAILDELAY*
Reset Width	5 - 1024	*CPC_RESETLEN_REG.RESETLEN*
Power Gating Enabled	Enabled or not	N/A
Clock Tree Root Gating Enabled for low power use	Enabled or not	N/A
IOCU Options		
Number of IOCU's	0, 1, or 2	*GCR_CONFIG*$_{NUMIOCU}$
IODB implementation style	Flops or generator	N/A
MConnID mask	0 - 8 bit	N/A
L2 Cache Options		
Cache Size	0, 128, 256, 512, 1024, 2048, 4096, or 8192 KBytes	*CONFIG2*$_{SS,}$ *CONFIG2*$_{SL,}$ *CONFIG2*$_{SA,}$
Cache Line Size	32 bytes or 64 bytes	*CONFIG2*$_{SL}$
Memory port data width	64-bit, 128-bit, or 256-bit	N/A
ECC Protection	ECC or no ECC	Build-time choice with run-time enable via *ErrCtl*$_{L2EccEn}$
Clock Gating	See general options section below	N/A
Memory BIST	See general options section below	N/A
General Options (applicable to multiple blocks)		
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

Revision History

Revision	Date	Description
~~01.00~~	~~July 27, 2012~~	~~• Early Access (EA) release of interAptiv data sheet.~~
01.01	September 20, 2012	• General Access (GA) release of interAptiv data sheet.
01.10	August 24, 2015	• Maintenance Release 1.
01.20	May 13, 2016	• Maintenance Release 2.

Public. This publication contains proprietary information which is subject to change without notice and is supplied ‘as is’, without any warranty of any kind.

Copyright © Wave Computing, Inc. All rights reserved.
www.wavecomp.ai