I6500 Multiprocessing System Datasheet

March 31, 2017

The I6500 Multiprocessing System (MPS) is a high performance multi-core microprocessor system that provides a best in class power efficiency for use in system-on-chip (SoC) applications. Each I6500 CPU core combines multithreading and an efficient dual-issue pipeline to deliver outstanding computational throughput. The I6500 Coherence Manager maintains Level 2 (L2) cache and system level coherency between all cores, main memory, and I/O devices. The I6500 MPS is a configurable and a synthesizable solution. The collection of clusters of cores can be configured with a variable number of cores, I/O coherent interfaces, and L2 cache size. Each of the cores can be configured with Level 1 (L1) cache sizes, number of threads, and single instruction multiple data (SIMD) functionality.

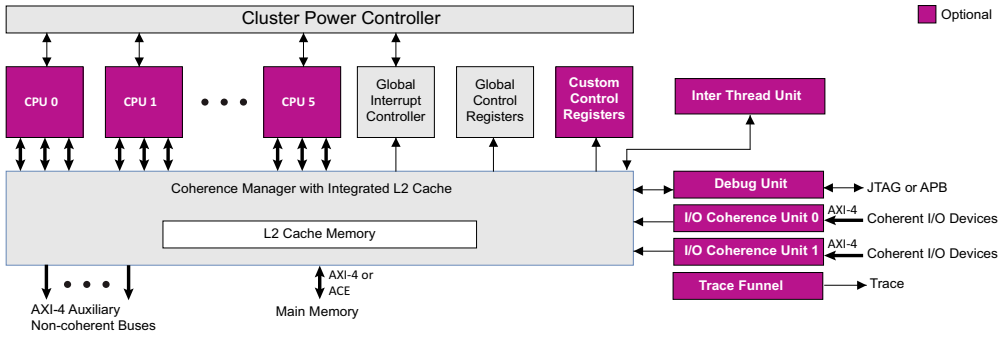
Each I6500 core implements the Release 6 of the MIPS64 Instruction Set Architecture (ISA) with full hardware multithreading and hardware virtualization support. In addition, the core can be configured with a SIMD engine supporting integer, single and double precision, and floating and fixed point operations.

Highlights of the I6500 MPS include:

- Multi-Cluster support
- Up to 6 CPU cores per cluster
- PDtrace support
- Coherence Manager (CM3.5) with integrated L2-cache:
  - Up to 8 I/O Coherence Units (total of cores + IOCUs must be no greater than 8)
  - Cluster Power Controller (CPC)
  - Global Interrupt Controller (GIC)
  - Global Configuration Registers (GCR)
  - Multiprocessor debug via in-system Debug Unit (DBU)
  - Trace Funnel (TRF)
  - Cluster Inter-thread communication unit (ITU)

Figure 1.1 shows a block diagram of a single cluster I6500 Multiprocessing System (MPS).

Figure 1.1 Block Diagram of Single Cluster I6500 Multiprocessing System



I6500 Features at a Glance

The I6500 MPS is feature rich with the current MIPS64 architecture, new CPU and system level features designed for compelling performance, power, and area form factors. The MPS flexibility and features are well suited for a broad range of markets and applications, from deeply embedded to automotive to consumer/mobile and on up to enterprise class storage, server, and dataplane solutions.

MIPS Architecture

The I6500 Multiprocessing System has four key architecture features that set the core's foundation.

MIPS64® Release 6 Architecture

MIPS64® architecture, an industry standard, is the groundwork of the I6500 product offering. The MIPS64 architecture provides a solid, high-performance base by incorporating powerful features, standardizing privileged mode instructions, and supporting past ISAs. It also provides a seamless upgrade path from the MIPS32 architecture. MIPS64 is based on a fixed-length, regularly encoded instruction set, and it uses a load/store data model. It is streamlined to support optimized execution of high-level languages. MIPS64 also has both compact and delayed branches. This helps the compiler generate dense code while still maintaining backward compatibility. Availability of 31 general-purpose registers enables compilers to further optimize code generation by keeping frequently accessed data in registers.

MIPS64 provides memory management and its information through the configuration registers. The MIPS64 architecture enables real-time operating systems and application code to be implemented once and reused.

MIPS® SIMD Architecture

SIMD (Single Instruction Multiple Data) is an important technology for modern CPU designs because it improves performance by allowing efficient parallel processing of vector operations. The MIPS® SIMD Architecture (MSA) technology incorporates a software-programmable solution into the CPU to handle emerging codecs or potentially eliminate dedicated hardware functions in some cases. This programmable solution allows for increased system flexibility. In addition, the MSA is designed to accelerate many compute-intensive applications by enabling generic compiler support, which can automatically vectorize code to enhance performance.

MIPS® Virtualization

To address security, privacy and reliability concerns in a wide range of devices, MIPS has added hardware supported virtualization technology into the I6500 core. The hardware virtualization support enables processors to be OmniShield-ready. OmniShield is security technology which ensures that applications that need to be secure are effectively and reliably isolated from each other, as well as protected from non-secure applications.

System Control Coprocessor (CP0) Architecture

In the MIPS architecture, CP0 implements the Privileged Resource Architecture (PRA), which includes:

- System configuration registers
- Virtual to physical address translation (MMU)
- Exception control system (including interrupt control)
- Processor's diagnostic capability
- Operating modes (kernel, user, supervisor, and debug)

Configuration information, such as cache size and associativity, and the presence of optional features like 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. Refer to MIPS64 Release 6 specifications for further details.

System-level Features

- Up to six coherent MIPS64 Release 6 CPU cores
- Multi-Cluster support: Cluster composed of up to 0 - 6 CPUs and 0 - 2 IOCUs (sum being no more than 8 agents) and a Level 2 cache connection to a coherent interconnect. Support for up to 4 clusters.
- Integrated L2 cache controller supporting a 8-way and 16-way set-associativity
  - Inclusive of the L1 data caches
  - 256 KB to 8 MB cache sizes
  - Single bit correction and double bit detection
- CPC to shut down idle cores for power efficiency
- Up to 8 I/O Coherence Units (total of cores + IOCUs must be no greater than 8)

- Virtualization Module Support
- Cache-to-cache data transfers
- Out-of-order data return
- Hardware L2 cache prefetch controller significantly improves performance of work-loads such as memory to memory data transfer/copy (memcpy)
- Independent clock ratios on core, memory, and IOCU ports
- SoC system interface supports AXI-4 (Advanced eXtensible Interface rev. 4, also known as AMBA 4 AXI) or ACE (AXI Coherency Extensions) protocol with 48-bit address and 256-bit data paths. This interface can be configured to support up to 96 outstanding requests.
- High bandwidth 128-bit data paths between each core and the Coherence Manager
- Software controlled core level and cluster level power management
- Debug port supporting multi-core debug (JTAG/APB)
- Program and Data trace (PDtrace) mechanism to debug software

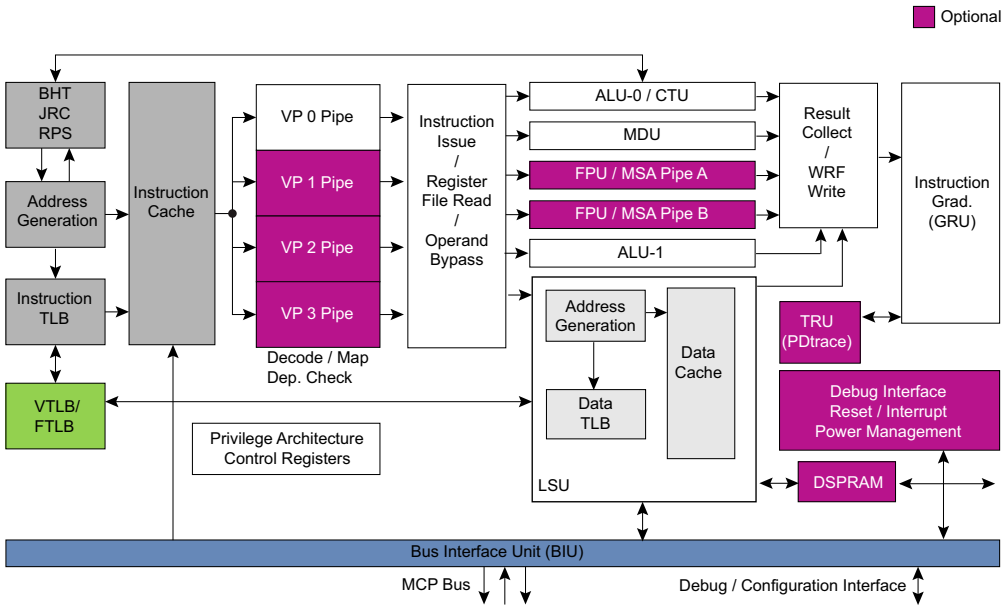
CPU Core-Level Features

- Full 64-bit Instruction Set Architecture via MIPS64 Release 6
- 48-bit virtual and physical addresses
- Power efficient design
- Dual issue instruction fetch, decode, issue, and graduate
- Hardware multithreading
- Virtualization support
- L1 caches with Error Correction Code (ECC) protection
- L2 cache support — Implemented as shared L2 in the Coherence Manager
- Programmable Memory Management Unit with large first-level ITLB/DTLB backed by fast on-core second-level variable page size TLB (VTLB) and fixed page size TLB (FTLB)
    - Shared FTLB across all virtual processors (VPs) in a CPU
    - MIPS DVM support through Global Instruction cache and TLB invalidation
- Load and store bonding support
- Unaligned load / store support in hardware
- Program and Data Trace (PDtrace) support for Instructions and Data (Virtual Addresses and Data Values)
- Optional Data Scratch Pad (DSPRAM)
- Optional InterThread Communication Unit (ITU)

I6500 CPU Core Features

Figure 1.2 shows a block diagram of a single I6500 core. The logic blocks in this diagram are described in the following sections.

Figure 1.2 I6500 Core Block Diagram



For more information on the I6500 core in a multiprocessing environment, refer to “Multiprocessing System Features” on page 16.

Instruction Fetch Unit (IFU)

The Instruction Fetch Unit (IFU) fetches instructions from the L1 instruction cache and supplies them to the Execution Unit (EXU). The IFU can fetch up to two instructions at a time from the L1 cache and fill the instruction buffers, which decouple the instruction fetch unit from the issue and execution of the instructions.

Branch Prediction

The IFU employs sophisticated branch prediction that anticipates the branch direction to improve performance and efficiency. The prediction is based on both local and global history of the branch captured in the Branch History Table (BHT) with majority voting. The predictor adapts to the program by self learning. The prediction stops on certain types of instructions, giving software control of the code execution.

Jump Prediction

The IFU has a hardware-based Jump Register Cache (JRC) and Return Prediction Stack (RPS) to predict jump target addresses. This results in faster throughput during subroutine calls and returns.

Level 1 Instruction Cache

The I6500 L1 instruction cache is configurable as 32 KB or 64 KB in size and is organized as 4-way set-associative. The instruction cache is virtually indexed and physically tagged to parallelize the data access and the virtual-to-physical address translation.

This cache is used to fetch two instructions per cycle. To conserve power, a way-prediction mechanism enables only the expected way. The cache is protected by single- and double-bit error detection logic.

Each cache line holds 64 bytes of instructions and the coherency of the cache is maintained by software with hardware assistance.

Execution Unit (EXU)

In the I6500 core, the Execution Unit (EXU) implements the logic for:

- Instruction Buffer Management
- Instruction Selection and Issue
- Source Operand Read and Bypass
- Integer Execution Units
- FPU / MIPS® SIMD Architecture (MSA) Execution Units
- Result Collection & Instruction Graduation

Instruction Buffer Management & Issue

The fetch unit delivers up to two instructions per cycle to the EXU. The EXU keeps these instructions in a deep instruction buffer. The EXU maintains a separate instruction buffer for each thread (VP).

Up to two instructions may be issued for execution during a clock cycle. The instructions can be issued from the same thread or from different threads. A round-robin priority scheme is used to arbitrate among threads.

Instructions can be concurrently issued to any two of the following EXU functional units:

- 2 Integer Units
- 1 Multiply / Divide Unit

- 1 Branch Unit
- 1 Load Store Unit
- 1 Short Floating Point Pipe
- 1 Long Floating Point Pipe

Source Operand Read and Bypass

The EXU can simultaneously read source operands from the Architectural Register File (ARF) or Working Register File (WRF) for each of the instructions (regardless of thread context). In addition, the EXU implements a fully symmetric operand bypass network to bypass a result from a preceding execution stage.

Integer Execution Units

The EXU has two complete ALUs that perform single-cycle operations including add, subtract, shifts, rotates, bit-wise logical, and several other operations. One of the ALUs assists in resolving conditional branches.

The EXU also contains a dedicated 64x64 integer multiplier and radix 4 SRT divider to speed up compute intensive applications and implements cyclic redundancy code (CRC and its variants) in hardware.

Floating Point / MSA Pipelines

The I6500 core features an optional 128-bit SIMD engine that implements the MIPS SIMD Architecture (MSA). The engine handles scalar floating point as well as SIMD integer and floating point datatypes. Floating point operations are IEEE 754-2008 compliant.

The EXU implements two separate pipelines (1 short, 1 long) to execute both floating point and MSA instructions. These two pipelines allow the execution of simple floating point instructions to bypass and execute in parallel with less frequently used complex and iterative instructions. One pipeline executes SIMD logical ops, SIMD integer adds, and FP compares and FP/SIMD stores. The other pipeline executes SIMD integer multiplies, SIMD vector shuffles, FP adds, FP multiplies, and FP divides.

The SIMD unit contains thirty-two 128-bit vector registers shared between SIMD and FPU instructions. Single-precision floating-point instructions use the lower 32 bits of the 128-bit register. Double-precision floating point instructions use the lower 64 bits of the 128-bit register. SIMD instructions use the entire 128-bit register interpreted as multiple vector elements: 16 x 8-bit, 8 x 16-bit, 4 x 32-bit, or 2 x 64 bit vector elements.

SIMD instructions enable:

- Efficient vector parallel arithmetic operations on integer, fixed-point, and floating-point data

- Operations on absolute value operands
- Rounding and saturation options
- Full precision multiply and multiply-add
- Conversions between integer, floating-point, and fixed-point data
- Complete set of vector-level compare and branch instructions with no condition flag
- Vector (1D) and array (2D) shuffle operations
- Typed load and store instructions for endian-independent operation

The SIMD unit is fully synthesizable and operates at the same clock speed as the core.

The exception model is ‘precise’ at all times.

The SIMD unit supports fused floating point multiply-adds as defined by the IEEE Standard for Floating-Point Arithmetic 754-2008. Most FPU and SIMD instructions have one cycle throughput. All floating point denormalized input operands and results are fully supported in hardware.

Result Collection & Graduation

The EXU collects all results from single-cycle, fixed-latency, and variable-latency instructions and pairs them up with associated completion status (such as exceptions and interrupts), and commits the results into the Architectural Register File (ARF). This committing of final results is called the *graduation* of the instruction.

Load Store Unit (LSU)

The Load Store Unit (LSU) moves data between the core and the system memory. It also maintains an L1 data cache to accelerate access to commonly used data by the core. The LSU accepts a single operation per cycle and maintains several buffers to keep the data moving between the EXU and L1 cache and between the L1 cache and the Bus Interface Unit (BIU) at optimal rate.

Level 1 Data Cache

The I6500 L1 data cache is configurable as 32 KB or 64 KB in size and is organized as 4-way set-associative. The data cache is physically indexed and physically tagged to avoid virtual aliasing.

The L1 data cache is capable of fetching data on both aligned and unaligned memory accesses. In addition, it can combine multiple loads and stores into a single operation using a feature called “instruction bonding” to maximize memory bandwidth.

To conserve power, a way-prediction mechanism enables only the expected way. The cache is protected by single-bit error correction and double-bit error detection logic.

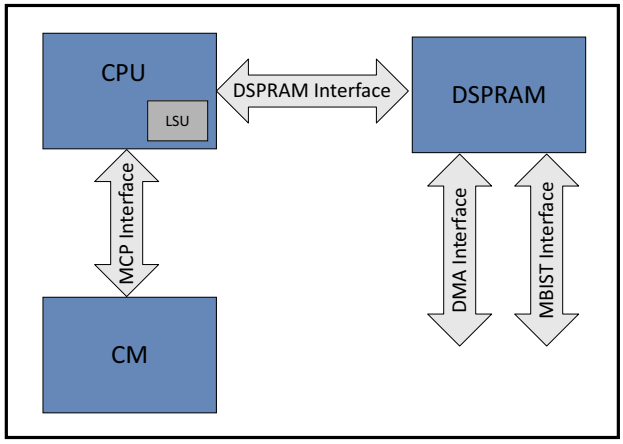
Each cache line holds 64 bytes of data as well as the associated tag and replacement information.

DSPRAM Interface

The I6500 data scratchpad RAM (DSPRAM) interface provides a connection to on-chip memory or memory mapped registers, which are accessed in parallel to the L1 data cache to minimize access latency. The DSPRAM interface connects the CPU to an external user designed DSPRAM module (a reference design is provided with the I6500 CPU).

Figure 1.3 shows a block diagram of the I6500 DSPRAM and interface.

Figure 1.3 I6500 DSPRAM Block Diagram



Features:

- 16-Byte wide datapath for both read and write.
- Data can be protected (parity/ECC/none on 32 bit granularity), in DSPRAM.
- Multi-threaded design, if one thread is blocked the other threads may continue to access the DSPRAM.

Store and Write Buffer

The LSU contains store buffers that decouple the main pipeline from the memory subsystem, allowing the LSU to efficiently schedule cache writes and coherence operations while the main pipeline continues to execute subsequent instructions. After a store instruction graduates in the main pipeline, the LSU takes control and forwards the store data from the store buffer to subsequent load instructions until the data is committed to the cache or main memory.

The store buffers can merge multiple cacheable stores into a single larger write operation, which can take advantage of the 512-bit cache write datapath. This store buffer improves performance by avoiding contention at the cache RAM ports and saves power

by reducing the number of RAM accesses. When data from the cache is written back to main memory, an entire cache line is transferred from the cache RAM to the evict buffer in the BIU in a single clock cycle. This frees up the LSU cache pipeline to proceed with subsequent operations, while the BIU streams the write-back data to the CM3.5 as a burst write transaction.

The store buffer also merges multiple uncached-accelerated stores into a single burst-write transaction, to increase the efficiency of the bus and avoid stalling the main pipeline. Gathering of uncached accelerated stores can start on any arbitrary address and can be combined in any order within a 64-byte aligned block of memory.

Memory Management Unit (MMU)

The Memory Management Unit (MMU) translates virtual addresses to physical addresses and provides attribute information for different segments of memory. The I6500 MMU contains the following Translation Lookaside Buffer (TLB) structures:

- Instruction TLB (ITLB)
- Data TLB (DTLB)
- Variable Page Size Translation Lookaside Buffer (VTLB) per VP
- Fixed Page Size Translation Lookaside Buffer (FTLB) per core

Instruction and Data TLB (ITLB and DTLB)

The ITLB and DTLB (micro TLBs) are fully associative. The micro TLBs are used by the IFU and LSU to perform high speed virtual to physical memory address translation for instruction fetch and data movements respectively.

The ITLB is implemented in the IFU with support for 4 KB, 16 KB, or 64 KB page sizes per entry. The DTLB is implemented in the LSU with support for 4 KB, 16 KB, or 64 KB page sizes per entry. The micro TLB arrays are shared between VPs.

The number of entries varies with the number of VPs present, as listed in Table 1.1.

Table 1.1 Entries per VP

	VPs	Entries
ITLB	1	6
	2	12
	4	18
DTLB	1	8
	2	14
	4	20

The micro TLBs are managed completely by hardware and are transparent to the software. The micro TLBs are backed up by larger VTLB and FTLB structures. If a virtual address cannot be translated by the micro TLB, the VTLB / FTLB attempts to translate the address in the following clock cycle or when available. If successful, the translation information is copied into the appropriate micro TLB for future use. When Virtualization is in use, the micro TLBs store the full two-level translation from the Guest Virtual Address to Root Physical Address to maintain high performance.

Variable Page Size TLB (VTLB)

The VTLB is a fully associative translation lookaside buffer with 16 dual entries per thread that can map variable page sizes from 4 KB to 1 GB.

Fixed Page Size TLB (FTLB)

The FTLB contains 512 dual entries organized as 128 sets and 4-way set-associative. The FTLB page size is configurable at run-time to either 4 KB, 16 KB, or 64 KB.

FTLB translations are shared for all VPs with the same GID (Guest ID) + MMID (Memory Map ID) when the MMID is enabled. Using the 16-bit MMID creates a global address space, allowing the MMU translations to be shared across VPs on a core and global invalidates to be performed across cores. Optionally, the legacy 10-bit ASID can still be used, in which case FTLB translations are not shared across the VPs.

Virtualization Support

The Virtualization Module is a set of extensions to the MIPS64 Architecture for efficient implementation of virtualized systems. This feature provides privileged (root) and unprivileged (guest) operating modes. It supports up to 31 guests.

The guest mode can be enabled by software. The key element is a control program known as a Virtual Machine Monitor (VMM) or hypervisor. The hypervisor is in full control of machine resources at all times.

When an operating system (OS) kernel runs within a virtual machine (VM), it becomes a “guest” of the hypervisor. All operations performed by a guest must be explicitly permitted by the hypervisor. To ensure that it remains in control, the hypervisor always runs at a higher level of privilege than a guest operating system kernel.

The hypervisor manages access to sensitive resources, maintains the expected behavior for each VM, and shares resources between multiple VMs.

In a traditional operating system, the kernel (or supervisor) runs at a higher level of privilege than user applications. The kernel provides a protected virtual-memory environment for each user application, inter-process communications, I/O device sharing, and transparent context switching. The hypervisor performs these same basic functions in a virtualized system, except that the hypervisor’s clients are full operating systems rather than user applications.

The virtual machine execution environment created and managed by the hypervisor consists of the full Instruction Set Architecture (ISA), including all Privileged Resource

Architecture (PRA) facilities, and any device-specific or board-specific peripherals and associated registers. It appears to each guest operating system as if it is running on a real machine with full and exclusive control.

Bus Interface (BIU)

The BIU interfaces the instruction and data caches with the CM3.5. This interface implements MIPS Coherence Protocol (MCP) and has three channels that support 128-bit data transfers. The transaction size can vary from 1 byte to 16 bytes for single uncached access or the full 64 bytes for a cache line. BIU supports full memory coherency including interventions.

Interrupt Handling

Each I6500 core supports six hardware interrupts including a timer interrupt and a performance counter interrupt. In addition, it support two software interrupts. These interrupts can be used in any of three interrupt modes, as defined by the MIPS64 Architecture:

- *Interrupt compatibility* mode.
- *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.

Operating Modes

The I6500 core supports seven modes of operation:

- Two user modes (guest and root) are used for application programs.
- Two supervisor modes (guest and root)
- Two kernel modes (guest and root) are used to handle exceptions and operate system kernel functions, including CP0 management and I/O device accesses.
- Debug mode is used during system bring-up and software development. Refer to Section ""“Core Debug Support” on page 15” for more information on debug mode.

I6500 Core Power Management

The I6500 core offers several power-management features. It supports low-power design, such as active power management and power-down modes of operation. The I6500 core is a static design that supports slowing or halting the clocks to reduce system power consumption during idle periods.

Instruction-Controlled Power Management

The Instruction Controlled power-down mode is invoked through execution of the WAIT instruction.

The WAIT instruction puts the processor in a quiescent mode where no instructions are running. When the WAIT instruction is seen by the Instruction Fetch Unit (IFU), subsequent instruction fetches are stopped. However, the internal timer and some of the input pins continue to run. Any interrupt, NMI, or reset condition causes the CPU to exit this mode and resume normal operation.

Core Debug Support

The I6500 core includes a debug block available for use in software debugging of application and kernel code. For this purpose, in addition to standard user, supervisor, and kernel modes of operation, the I6500 core provides a Debug mode.

Debug mode is entered when a debug exception occurs and continues until a debug exception return instruction is executed or the CPU is reset. The Debug features include:

- Up to 8 instruction breakpoints
- Up to 4 data breakpoints
- Single-step execution
- Memory and register access
- Program and data trace (PDtrace).

Multiprocessing System Features

The I6500 Multiprocessing System (MPS) provides multi-cluster support where each cluster consists of up to six I6500 cores, a Coherence Manager (CM3.5) with integrated L2 cache, up to eight IOCUs, a cluster power controller (CPC), global interrupt controller (GIC), debug unit (DBU), and global configuration registers (GCR). The CM3.5 maintains coherence with the cores' L1 caches by implementing a directory-based coherence protocol that enables both low power and high performance.

The I6500 extends capability from a single coherent six-core cluster with support I/O coherency to a new set of capabilities that enable more complex systems, such as:

- Multiple coherent clusters of CPUs
- Heterogeneous Multi-processing (CPU + GPU or other coherently designed processing elements)
- Groups of coherent I/O or co-processing functions or clusters

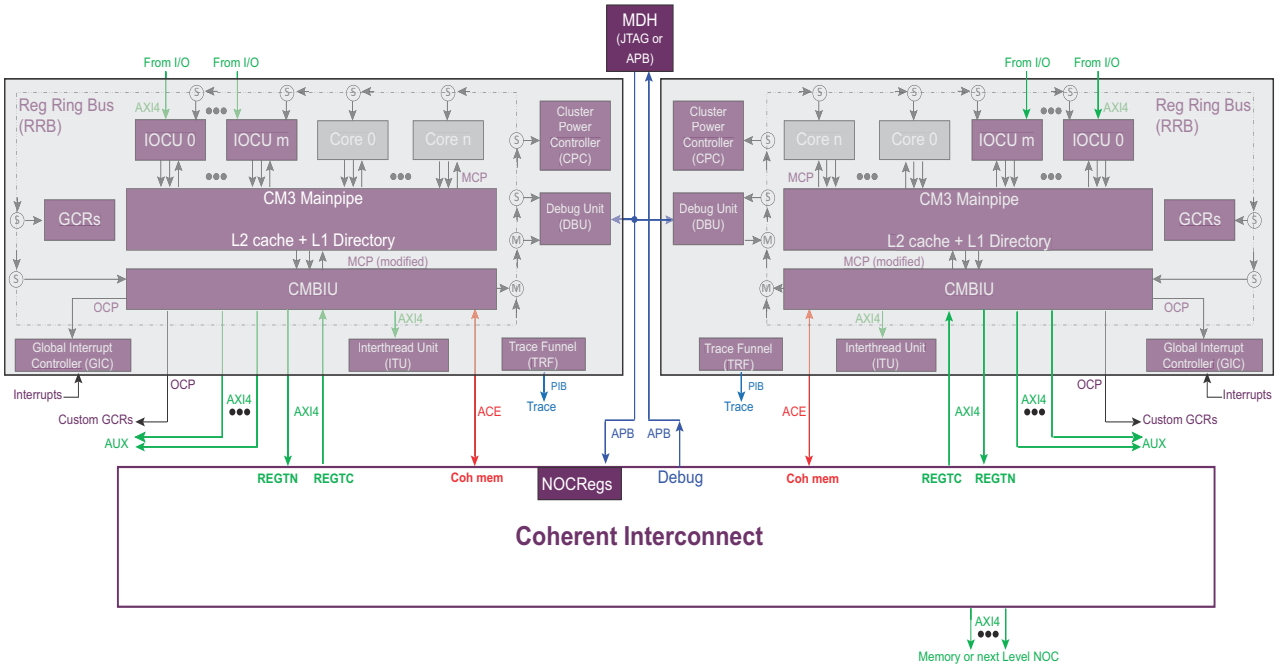
A cluster is composed of up to 0 - 6 CPUs and 0 - 8 IOCUs (sum being no more than 8 agents) and a Level 2 cache connection to a coherent interconnect. An agent is either a CPU, which is included in the cluster, or an external I/O device. The initial I6500 implementation support is 2 - 4 clusters.

The I6500 cluster can be configured in one of two modes:

1. It can be configured as a single non-coherent cluster, similar to the I6400. In this case the main memory bus interface from the cluster is AXI-4 (same as the I6400).
2. It can be configured to support multiple coherent clusters. In this case, the main memory bus interface is ACE.

Figure 1.4 shows a reference design of a cluster integrated with a network.

Figure 1.4 I6500 Integrated Cluster with Network



Directory Based Level 1 Cache Coherence

The Coherence Manager (CM3.5) keeps all the L1 data caches coherent with each other by maintaining a directory that tracks the state of each L1 data cache line for each core. The directory uses the same address tags as the Level 2 cache, reducing the power and area required to maintain coherence. All Level 1 data and instruction cache misses are looked up in the directory to determine the state of the line in the L1 data caches as well as the L2 cache. Depending on the request attributes and directory state, the CM3.5 sends intervention requests to cores that have the line in their L1 data cache, reads the data from the L2 Data RAMs, or issues a request to the memory subsystem. The CM3.5 immediately updates the directory state and routes the corresponding data to the requesting core.

With a directory-based coherence architecture, each of the cores do not need to maintain a second copy of the L1 cache tags to “watch” the memory transactions and compare them against its internal cache contents. Instead, that information is maintained by the directory, which shares the L2 address tags.

L1 Instruction Cache Coherence

The Level 1 instruction caches are not coherent, in that the CM3.5 directory does not track their contents. However, L1 instruction cache misses will be looked-up in the CM3.5 directory, and depending on the state, may receive its data from a core’s L1 data

cache. This feature reduces the overhead of the software required to maintain L1 instruction cache coherence.

CM3.5 Main Pipeline

The CM3.5 Main Pipeline manages all the data and control flows throughout the CM3.5 and the I6500 Multiprocessing System.

The main pipeline implements the directory-based coherence architecture and manages a unified and shared L2 cache. Some key features of the L2 cache are:

- 64-byte cache line size
- 8- or 16-way set-associative
- 256 KB, 512 KB, 1 MB, 2 MB, 4 MB, and 8 MB cache-size options
- 1 or 2 cycle tag RAM access
- 2 or 4 cycle data RAM access
- 1 or 2 L2 cache pipelines, each with two memory banks
- Pseudo LRU line-replacement algorithm
- Writeback architecture
- L2 is inclusive of the L1 data caches, that is, it is always a superset of all L1 Data Caches
- Physically Indexed and Physically Tagged
- Non-Blocking architecture (Fully Pipelined)
- 48-Bit Physical Address
- L2 Hardware Prefetcher automatically recognizes workloads, such as memcopy, and efficiently prefetches data into the L2 cache
- Hardware can automatically initialize L2 cache upon reset. Hardware can also be programmed to initialize/flush all or part of the L2 cache.
- Cache line locking support
- ECC support (single-bit error correction and double-bit error detection) for Tag and Data arrays
- Parity support on data buses

The CM3.5 main pipeline arbitrates among the requests received from the cores, IOCUs, and L2 hardware prefetcher. It accesses and updates the directory and L2 cache tags, performs reads or writes to the L2 data RAMs as necessary, and issues interventions to manage each core's L1 data caches.

Uncached requests are also handled by the CM3.5 main pipeline, but neither the directory nor L2 cache is accessed. Uncached accesses are decoded based on a programmable address map and routed to the CM's Bus Interface Unit (CMBIU). The programmable address map determines the final target of the request, such as uncached memory or a configuration register in the interrupt controller, power controller, etc.

The CM3.5 main pipeline identifies and resolves conflicting accesses as required.

The CM3.5 includes high performance features for data movement:

- 512-bit wide internal data paths throughout the CM3.5
- Three channel (two of 128-bit wide) system MCP interface to each of the CPU cores and IOCUs
- When configured as multi-cluster, ACE interface to inter-cluster network; AXI4 interface when configured as single cluster
- Support for up to 4 non-coherent Auxiliary AXI4 ports.

Cluster Power Controller (CPC)

Individual CPUs within the cluster can have their clock, power, or both 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 cores in the cluster. The CPC can be controlled via software by accessing and changing values in the registers and by hardware through a signal interface.

The CPC also organizes power-cycling of the CM3.5, 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 CPC also controls the clock ratios of the cores, CM3.5, I/O buses, and main memory bus. The CPC allows for the clock ratio of each component to be controlled independently, programmed by means of software commands or hardware signals. The clock ratio can be changed dynamically while the system is fully operating.

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. Signal settings determine the course of action at deassertion of reset:

- Remain powered down
- Go into clock-off mode

- Power-up and start execution

In case of a system cold start and after reset is released, the CPC powers up the I6500 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 CM3.5 is also powered up.

At a warm start reset, the CPC brings all power domains into their cold start configuration. 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.

The CM includes memory-mapped registers that can override the default exception vector location. This allows different boot vectors to be specified for each of the VPs, so they can execute unique code if required. Furthermore, signals by the system also determine which VPs on each core start execution up. The CPC implements the capability to bring a core out of reset with no VPs running, letting the system hardware start one or more VPs at a later time.

I/O Coherence Unit (IOCU)

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 (CM3.5) and I/O devices. Reads and writes from I/O devices may access the L1 and L2 caches by passing through the IOCU and the CM3.5. Each request from an I/O device may be marked as coherent, or uncached. Coherent requests access the L1 and L2 caches. Uncached requests bypass both the L1 and L2 caches and are routed to main memory.

The IOCU provides an AXI slave interface to the I/O interconnect for I/O devices to read and write system memory.

The IOCU provides several features for easier integration:

- Supports incremental bursts up to 256 beats (128 bits per beat) on I/O side. These requests are split into cache-line sized requests on the CM3.5 side
- Coherent writes are issued to the CM3.5 in the order they were received

Global Interrupt Controller (GIC)

The Global Interrupt Controller handles the distribution of interrupts between and among the CPUs in the cluster. 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
- Support for NMI routing
- Standardized mechanism for sending inter-processor interrupts
- Support for Virtualization of interrupts: each interrupt to be mapped to Guest or Root

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.

Some of the control options include:

- *Address map* — The base address for the various peripheral blocks, such as the CPC, GCR, User GCRs, and GIC address ranges can be specified
- *Error reporting and control* — Logs information about errors detected by the CM3.5 and controls how errors are handled (ignored, interrupt, etc.)
- *Control Options* — Various features of the CM3.5 can be disabled or configured
- *L2 Cache operations* — Registers used during L2 cache maintenance instructions
- *Mapping registers* — Route requests to one of the non-coherent Auxiliary (AUX) AXI-4 ports
- *Multi-cluster register access* — Allows a CPU of one cluster to access a register on a remote cluster via the REGTC/REGTN AXI4 buses

Custom GCRs

The CM3.5 provides the ability to implement a 64 KB block of custom registers that can be used to control system level functions. These registers are user defined and then instantiated into the design. Two global registers are provided by the CM3.5 to implementation custom registers: the Global Custom Base register, and the Global Custom Status register.

Interthread Communication Unit (ITU)

The I6500 MPS includes an integrated cluster ITU, which provides a gating storage capability for synchronization between threads on all I6500 CPUs.

The I6500 ITU includes the following features:

- Memory Mapped ITU Control Registers (ICR) within the ITU Addressable Region
- Entry Cell storage: 64-bit Dword
- Multi Entry Cell storage: 64-bit wide FIFO with build time configurable depth of 2 - 8

Clocking Options

The I6500 Multiprocessing System has the following clock domains:

- *Reference Clock* — This clock is created by the SOC and used by the I6500 Multiprocessing System. The reference clock is controlled and scaled by the input clock. This clock drives the CPC.
- *Prescaled clock* — The reference clock can be prescaled by a programmable value of 1:1 (no prescale) to 1:255. This prescaled clock is used a base clock for all cluster components, except the CPC.
- *Cluster clock domain* — This clock drives the CM3.5 (including Coherence Manager, Global Interrupt Controller, IOCU, and L2 cache). This clock can be configured to be the same as Prescale Clock or Prescale / 2.
- *Core-N clock domain* — Each core in the cluster can operate at independent frequency. This clock can be controlled at run time (via CPC).
    - When the CM3.5 is operating at 1:1, the cores can run at 1:1, 1:2, 1:3, 1:4, 1:5, 1:6, 1:7 or 1:8 of the prescale clock.
    - When the CM3.5 is operating at 1:2, the cores can run at 1:1, 1:2, 1:4, 1:6, or 1:8 of the prescale clock.
- *System clock domain* — The AXI-4 or ACE port connecting to the SOC and the rest of the memory subsystem may operate at a ratio of the cluster clock domain. The system clock domain can be configured to use an internal clock or an external clock. When configured to use an internal clock, the rate is a ratio of the prescale clock. Supported ratios are 1:1, 1:2, 1:3, 1:4, 1:5, 1:6, 1:7, 1:8.
- AUX AXI clock domains — The optional non-coherent AXI-4 port connecting to the SOC may operate at a ratio of the cluster clock domain. Each auxiliary AXI clock domain can be independently configured to use an internal clock or an external clock. When configured to use an internal clock, the rate is a ratio of the pre-scale clock. Supported ratios are 1:1, 1:2, 1:3, 1:4, 1:5, 1:6, 1:7, 1:8.
- When configured to use an external asynchronous clock, the AXI interface captures/drives on that clock and there is an asynchronous boundary crossing implemented internal to the cluster.
- *TAP clock domain* — This is a low-speed clock domain for the JTAG TAP controller, controlled by the EJ_TCK pin. It is asynchronous to the Reference Clock.

- *I/O clock domains* — Each port connects the IOCU to the I/O Subsystem. Each IOCU clock may operate at a ratio of the prescale clock domain. Supported ratios are the same as the system clock domain. Similar to the System clock domain, each I/O clock domain can be configured to operate at a ratio of the prescale clock or an asynchronous external clock.

Debug Unit

The Debug Unit (DBU) is an optional component that enables debug using a probe connected through a JTAG scan chain. Alternatively, the DBU can be connected to the system through an APB transactor port. The DBU contains the single TAP controller in the cluster, which can access registers through the cluster. The DBU also contains a RAM to hold instructions and data accessed by the cores while in debug mode.

Features of the debug unit include Hardware Breakpoints and a Fast Debug Channel:

Hardware breakpoints stop the normal operation of the CPU and force the system into debug mode. There are two types of hardware breakpoints implemented in the I6500 CPU: Instruction breakpoints and Data breakpoints.

Instruction breaks occur on virtual instruction execution addresses and may be qualified by ASID or MMID, VP, GuestID, and Context. Addresses may be single, masked, or ranges.

Data breaks occur on load and store operations based on virtual address, ASID or MMID, VP, GuestID, Context, and data value. Addresses may be single, masked, or ranges. Loads and stores may be aligned or misaligned.

The Fast Debug Channel is a mechanism for efficient bidirectional transfer between a CPU and the debug probe. Data is transferred serially via the TAP interface. 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 level or can operate in a polled mode. Up to 16 virtual channels can travel in each direction.

Inter-CPU Debug Breaks

The MPS includes registers that enable cooperative debugging across all VPs. Programmable registers allow VPs to be placed into debug groups such that whenever one VP within the group enters debug mode, a debug interrupt (DINT) is sent to all VPs within the group, causing them to also enter debug mode and stop executing non-debug mode instructions. This same mechanism can be used to have multiple VPs exit debug mode simultaneously.

PC Sampling

Each VP has hardware to provide periodic sampling of the program counter. Through the DBU, a probe can read addresses that have been executed. The host software can accumulate these executed addresses and provide views of program hot spots, from the module and function level down to the source line and individual instruction levels.

PDtrace

The I6500 core includes trace support for real-time tracing of instructions, data addresses, data values, and performance counters. A Trace funnel muxes the PDTrace stream from all cores and the CM, and either stores the trace information into an on-chip trace RAM 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. The on-chip trace memory may be accessed either through load instructions or the existing JTAG TAP interface, which requires no additional chip pins.

The off-chip trace is managed with the PIB3 (3rd-generation Probe Interface Block) hardware that ships with the product. It provides a selectable trace port width of 8 or 16 pins plus DDR clock. The trace data is streamed on these pins and captured using a compatible probe such as the MIPS Sysprobe SP58ET.

Initial and Possible Configurations

The I6500 Multiprocessing System can support a variety of configurations. Initially, the following configurations listed in Table 1.2 will be supported. If a different configuration is required, contact your sales account manager for current configurations and to request a new configuration. Each of the initial configurations can include an optional SIMD engine (with integer and floating point data types).

Multi-core offerings are symmetric (same cache size, VPs, SIMD for each core) but different clock ratios per core are supported.

Table 1.2 I6500 Configurations

Config Type	Feature Name	Description	Allowed Values
Defined in cm3_config.vh			
Cores	num_cores	Number of cores per cluster	0, 1, 2, 3, 4, 5, 6
L2 cache	l2_cache_size	L2 cache size	256, 512, 1024, 2048 KB
L2 cache misses	l2_missq_override	L2 cache misses in flight (to override default)	8 - 96
L2 data buffer	l2_mem_sdb_override	L2 Store data buffer size (to override default)	8 - 64
L2 prefetch	l2_pref	L2 Prefetch	0, 1 (for absent or present)
Pipes	num_pipes	Number of pipes or scheduler paths in the Coherence Manager	1, 2
ACE requests	l2_num_intv_override	Number of incoming ACE snoop requests	1 - 16
Interrupts	num_irqs	Number of interrupts. Interrupts are as a number of 'slices' where a 'slice' is 8 interrupts.	8 - 256 (in increments of 8)
IOCUs	num_iocus	Number of I/O Coherence Units (IOCUs)	0 - 8
IOCU size	iocu_size	IOCU size information. Chooses the size of the IOCU implementation for reads/writes and SDB IDs. Small=4 RDs, 4 WRs, 4 SDB IDs Medium=8 RDs, 8 WRs, 8 SDB IDs Large=16 RDs, 12 WRs, 16 SDB IDs	Small, Medium, Large
IOCU reads	iocu_num_reads	Number of IOCU reads in flight. N.B: overrides iocu_size.	4, 8, 12, 16
IOCU writes	iocu_num_writes	Number of IOCU writes in flight. N.B: overrides iocu_size.	4, 8, 12
IOCU buffer IDs	iocu_sdb_ids	Number of IOCU Store Data Buffer IDs. N.B: overrides iocu_size.	8 - 32

Table 1.2 I6500 Configurations

Config Type	Feature Name	Description	Allowed Values
IOCU width	iocu_user_width	IOCU AxUSER width, routed to MEM port, default=8	0 - 9
GCRs	ugcr	User Global Configuration Registers (GCRs)	0, 1 (for absent or present)
Relay stages	num_relay_stages_core_mcp_cm	Number of relay stages between cores and CM	0, 1, 2 (0 default)
ITU	cluster_itu	Interthread Communication Unit	0, 1 (for absent or present)
External clock	c_mem_ext_clk_en	External Clock on ACE memory port (only used if ace=1 and not integrated NoC)	0, 1 (for absent or present)
Aux ports	num_aux	Number of auxiliary ports	0 - 4
Aux port 0	aux0_data_width	Auxiliary Port 0 data width	32, 64, 128, 256, 512
Aux port 1	aux1_data_width	Auxiliary Port 1 data width	32, 64, 128, 256, 512
Aux port 2	aux2_data_width	Auxiliary Port 2 data width	32, 64, 128, 256, 512
Aux port 3	aux3_data_width	Auxiliary Port 3 data width	32, 64, 128, 256, 512
defined in mips_config.vh			
Clusters	num_clusters	Number of coherent clusters. NOTE: With an integrated NoC, this refers to the number of clusters instantiated in mips_soc. However, without an integrated NoC this refers to the number of clusters in the example mips_soc. It does NOT refer to the number of clusters that you may instantiate in your design.	1-4 (for integrated NoC) Generally set to 2 for non-integrated NoC because a dual-cluster mips_soc example is provided.
ACE	ace	MEM port includes AXI Coherency Extensions	0, 1 (for absent, or present)
Virtual Processors	num_vps	Number of VPs per core	1, 2, 4
PDtrace output bus	tru_ext_bus_type	Type of external bus for Pdtrace. PIB: output of Probe Interface Block. TC: 256-bit wide output of Trace Funnel.	PIB or TC
AXI parity	axi_addr_parity	AXI address parity supported.	0, 1 (for absent or present)
AXI parity/byte	axi_addr_perbyte_parity	AXI address parity per-byte	0 - single parity bit, 1 - per byte parity
AXI data parity	axi_data_parity	AXI data parity supported.	0, 1 (for absent or present)
PDtrace	pdtrace	PDtrace unit	0, 1 (for absent or present)

Table 1.2 I6500 Configurations

Config Type	Feature Name	Description	Allowed Values
PDtrace memory	tru_mem	PDtrace internal memory size	6, 7, 8
Pdtrace PIB	tru_PIB	PDtrace PIB size (width)	8, 16 bits
defined in sam_core_config.vh			
L1 Instr cache	l1_icache_size	L1 Instruction cache size	32, 64 KB
L1 Data cache	l1_dcache_size	L1 Data cache size	32, 64 KB
FPU	fpu_present	FPU and MSA support	Yes, No
DSPRAM	dspram_size	Core Data Scratchpad RAM (DSPRAM) size	0, 64 KB

Revision History

Revision	Date	Description
01.00	March 31, 2017	• RC 1.00 Initial production release of I6500