proAptiv™ Multiprocessing System Datasheet

March 26, 2013

The proAptiv™ Multiprocessing System (MPS) is a high performance multi-core microprocessor with best in class power efficiency for use in system-on-chip (SoC) applications. The proAptiv MPS combines a deep pipeline with multi-issue out of order execution to deliver outstanding computational throughput. The proAptiv MPS is fully configurable/synthesizable and can contain up to six MIPS32® proAptiv CPU cores, system level Coherence Manager with integrated L2 cache, optional coherent I/O port, and optional floating point unit.

The proAptiv Multiprocessing System is available with up to six cores in the following configurations. All of these configurations include a second generation Coherence Manager with integrated L2 cache (CM2).

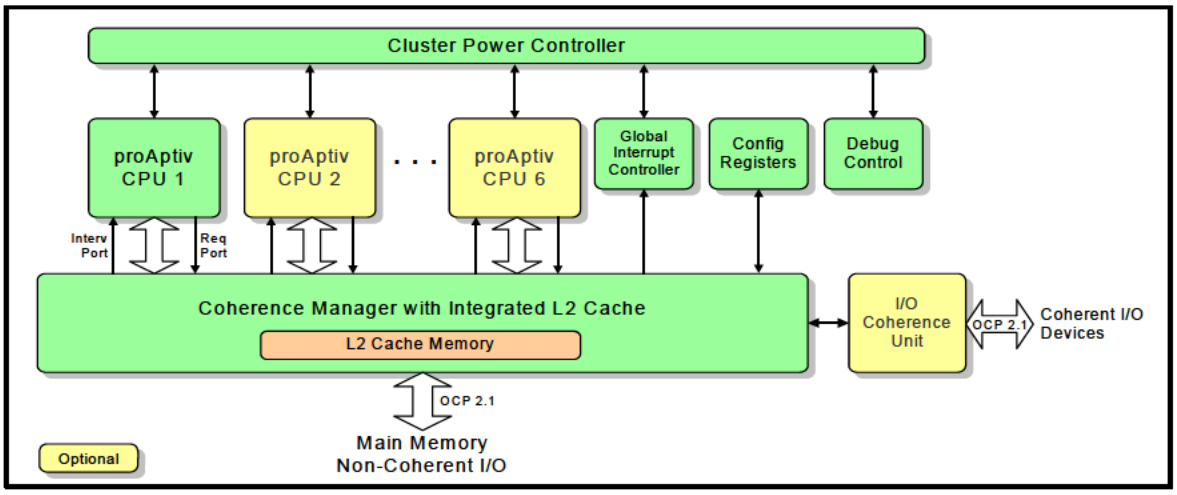
- Single core
- Dual core
- Three, four, or six cores

The MIPS32 proAptiv Multiprocessing System contains the following logic blocks.

- Cluster Power Controller (CPC)
- Coherence Manager (2nd generation) with integrated L2 cache (CM2)
- I/O Coherence Unit (IOCU)
- proAptiv Cores (1, 2, 3, 4, or 6)
- Optional 2nd generation Floating Point Unit (FPU2)
- Global Interrupt Controller (GIC)
- Global Configuration Registers (GCR)
- Multiprocessing System (MPS) Debug Unit
- Optional PDTrace in-system trace debugger

Figure 1 shows a block diagram of the proAptiv Multiprocessing System (MPS).

Figure 1. proAptiv™ Multiprocessing System Block Diagram



In the proAptiv 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 Coherence Manager (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 and/or voltage supply to idle cores.

1. Features

System level features:

- 1 - 6 coherent MIPS32 proAptiv CPU cores
- Cluster Power Controller (CPC) to shut down idle CPU cores
- Hardware I/O coherence port (optional)
- L1 data cache supporting the MESI coherence states
- Cache to cache data transfers
- Speculative memory reads to reduce latency
- Out-of-order data return
- Integrated 8-way set associative L2 cache controller with supporting 256 KB to 8 MB cache sizes
- Separate clock ratios on memory and IOCU OCP ports
- Clock ratio of 1:1 between Core, CM2, and L2 cache
- SOC system interface supports OCP version 2.1 protocol with 32-bit address and 64-bit or 256-bit data paths
- Software controlled core level and cluster level power management
- EJTAG Debug 5.0 port supporting multi-CPU debug
- MIPS PDtrace version 6
- Full scan design achieves test coverage in excess of 99% with optional memory BIST for internal SRAM arrays

CPU core level features:

- 5 integer, 2 floating-point and an optional CorExtend execution unit shared amongst multiple issue pipes
- Optional 2nd generation Floating Point Unit (FP2)
- Instruction Fetch Unit (IFU) with 4 instructions fetched per cycle
- Quad integer Out-of-Order issue with dedicated integer completion buffers that hold execution results until instructions are graduated in program order
- Dual floating-point issue with dedicated completion buffers that hold execution results until instructions are graduated in program order
- 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):
  - 16-entry Instruction TLB (ITLB) with page sizes of 4 KB or 16 KB per entry
  - 32-entry Data TLB (DTLB) with page sizes of 4 KB or 16 KB per entry
  - 64-dual-entry VTLB with page sizes up to 256 MB per entry
  - 512 dual-entry 4-way set associative FTLB with page sizes of 4 KB or 16 KB per entry (optional)
  - VTLB and FTLB can be accessed simultaneously on lookups
- L1 Instruction and Data Caches can be configured as 32 or 64 KB per cache
- Data and Instruction Scratchpad RAM can be configured from 4 KB to 1 MB (optional).
- Enhanced virtual addressing (EVA) mode allows for up to 3.0 GB of user or kernel virtual address space
- Write merging for uncached accelerated (UCA) operations
- Integrated integer Multiply/Divide Unit (MDU)
- CorExtend® MIPS32® compatible User Defined Instruction Set Extension allows user to define and add instructions to the core at build time

2. proAptiv CPU Core

Figure 2 shows a block diagram of a single proAptiv core. The following subsections describe the logic blocks in this diagram.

For more information on the proAptiv core in a multiprocessing environment, refer to Section 3. "Multiprocessing System"

2.1 MIPS Release 3 Architecture

The proAptiv core implements the MIPS32™ Release 3 Architecture in a superscalar, out-of-order execution pipeline. In addition, the proAptiv core also supports the MIPS16e™ ASE for code compression, and the DSP ASE Revision 2 for accelerating integer SIMD codes.

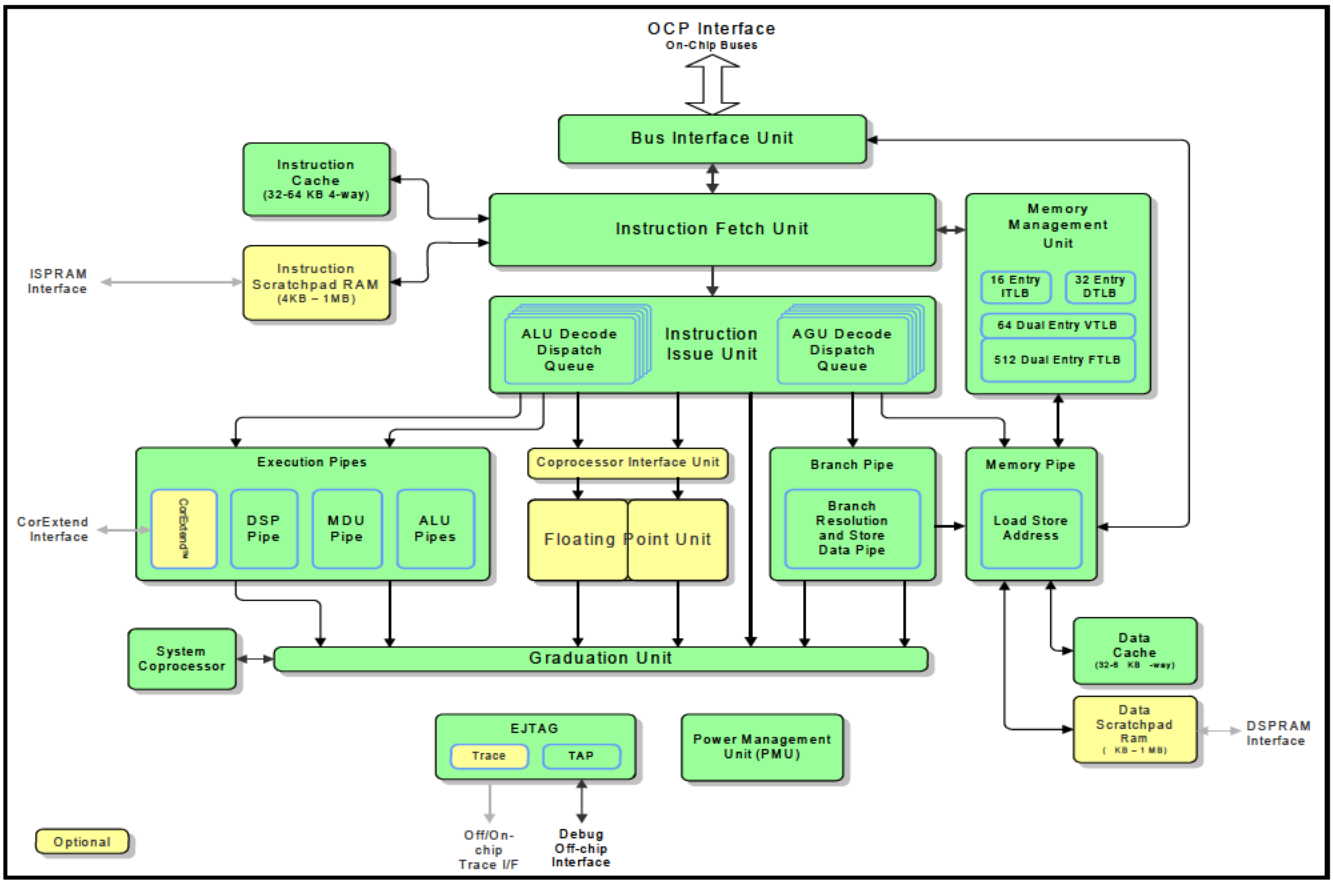
2.2 Instruction Fetch Unit

The Instruction Fetch Unit (IFU) is responsible for fetching instructions from the instruction cache, instruction scratchpad or memory, supplying them to the Instruction Issue Unit (IIU). The IFU can fetch up to four MIPS32 instructions at a time from the 4-way associative instruction cache. Instructions can also be fetched immediately from refill buffers in the event of an instruction cache miss.

The IFU employs sophisticated branch prediction and instruction supply strategies. The main predictor consists of three 2048-entry global branch history tables (BHT) that are indexed by different combinations of instruction PC and global history. A proprietary scheme is used to combine information from the three arrays to make a branch direction prediction.

Branch target prediction is provided by a hierarchy of multiple arrays. The fully-associative Level 1 BTB (Branch Target Buffer) is used for fast target re-steers on predicted taken branches, including returns. A large 4-way associative Level 2 BTB backs up the Level 1 BTB and also predicts indirect branches, even those with multiple target addresses.

Figure 2. proAptiv™ Core Block Diagram



The IFU also has a hardware-based return prediction stack to predict subroutine return addresses.The main predictor contains a BTAC (Branch Target Address Calculator) that can correct target mispredicts from lower-level predictors without paying a full branch resolution penalty. The IFU supports fully out-of-order branch resolution.

The IFU has a 16-entry micro-Instruction TLB (ITLB) used to translate the virtual address into a physical address. This translated physical address is used to compare against tags in the instruction cache to determine a hit. Refer to Section 2.6 "Memory Management Unit (MMU)" for more information.

A 12-entry instruction buffer decouples the instruction fetch from the execution. Up to 4 instructions can be written into this buffer, and a maximum of 2 instructions can be read from this buffer. To maximize performance, some 'fusing' (or concatenation) of instructions is done at this stage while other types of instruction 'fusion' are performed downstream.

The IFU can also be configured to allow for hardware prefetching of cache lines on a miss. When an instruction cache miss is detected, the IFU can prefetch the next 0, 1, or 2 lines (in addition to the missed line) to reduce average miss latency. This mechanism provides excellent performance without incurring the area, power and latency costs of more overly complicated branch or instruction prefetch strategies.

The Global History register is internal to the IFU block and supports a novel history computation scheme that factors different information into the history for different kinds of control transfer instructions. These novel hashing schemes enable significantly lower mispredict rates than other competing processors, directly translating to real world performance in many different applications.

The proAptiv level 1 (L1) instruction cache incorporates 'next fetch way' hit prediction logic. This allows the IFU to power on only those cache tag and data arrays that will provide the final instruction bytes and contributes to low power consumption.

2.3 Instruction Issue Unit (IIU)

The Instruction Issue Unit (IIU) is responsible for receiving instructions from the IFU and dispatching them to the out-of-order instruction scheduling windows and global instruction tracking window at a rate of 4 instructions per cycle.

The IIU tracks dynamic data flow dependencies between operations and issues them to the various pipes as efficiently as possible. Two schedulers, called the ALU DDQ and the AGU DDQ, service the various integer pipes.

The schedulers employ multiple dependency wake-up and pick schemes to enable age-based scheduling at high frequency. Having only two schedulers, rather than a low-frequency centralized scheduler or a large number of distributed reservation stations, is key to providing superior performance and power characteristics.

The IIU helps to 'fuse' load and store operations whereby two 32-bit loads or stores to adjacent locations are 'fused' or concatenated into one 64-bit memory access. This allows a factor of two improvement in certain memory intensive codes. The IIU also enables instruction 'Fission', whereby certain operations, like cacheable stores, are split into multiple micro-ops such as store-address and store-data operations.

Instructions are first renamed using a rename map, replacing the architectural register names with microarchitectural names from a global rename pool. The IIU also keeps track of the progress of each instruction through the pipeline, updating the availability of operands in the 'rename map' and in all dependent instructions. Renamed instructions are steered to the most appropriate schedulers, taking opcode and other information into account.

The IIU also keeps track of global pipeline flushes, adjusting the rename map and other control structures to deal with interrupts, exceptions and other unexpected changes of control.

2.4 Level 1 Instruction Cache

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

Each instruction cache entry contains a tag portion, a data portion, and a way select portion.

An instruction tag entry holds 21 bits of physical address, a valid bit, a lock bit, and a parity bit. There are 7 precode bits per instruction pair, making a total of 55 bits per tag entry. The data entry consists of 256 bits (8 MIPS32 instructions) of data and 32 bits of parity for a total of 288 bits. The way-select entry contains a 6 bit least-recently-used (LRU) field.

The proAptiv core supports instruction-cache locking. Cache locking allows critical code segments to be locked into the cache on a “per-line” basis, enabling the system programmer to maximize the performance of the system cache.

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

The proAptiv core implements virtual aliasing for the instruction cache, although this function can be disabled by the user.

2.5 Level 1 Data Cache

The Level 1 (L1) data cache is configurable at 32 or 64 KB in size. It is also organized as 4-way set associative. Data cache misses are non-blocking and up to nine misses may be outstanding. The data cache is virtually indexed and physically tagged to make the data access independent of virtual-to-physical address translation. To achieve the highest possible frequencies using commercially available SRAM generators, cache access and hit determination is spread across three pipeline stages, dedicating an entire cycle for the SRAM access.

Each instruction cache entry contains a tag portion, a data portion, a way select portion, and a dirty status portion.

A data tag entry holds 21 bits of physical address, a valid bit, a state bit, and a parity bit, making a total of 24 bits per tag entry. The data entry consists of 256 bits consisting of 32 bytes of data of data and 32 bits of parity for a total of 288 bits. The way-select entry contains a 6 bit least-recently-used (LRU) field, a 4-bit lock field, and a 4-bit lock parity field for a total of 14 bits. The Dirty state entry contains a 4-bit dirty field and a 4-bit dirty parity field.

The proAptiv core supports a data-cache locking mechanism identical to that used in the instruction cache. Critical data segments are locked into the cache on a “per-line” basis. The locked contents can be updated on a store hit, but are not selected for replacement on a cache miss.

The proAptiv core implements virtual aliasing for the data cache. This function is managed in hardware and is transparent to the user.

2.6 Memory Management Unit (MMU)

The proAptiv core contains a Memory Management Unit (MMU) that is primarily responsible for converting virtual addresses to physical addresses and providing attribute information for different segments of memory. The proAptiv MMU contains the following Translation Lookaside Buffer (TLB) types:

- 16-entry Instruction TLB (ITLB) with 4 KB or 16 KB per entry
- 32-entry Data TLB (DTLB) with up to 4 KB or 16 KB per entry
- 64 dual-entry Variable Page Size Translation Lookaside Buffer (VTLB) with up to 256 MB per entry
- 512 dual-entry 4-way set associative Fixed Page Size Translation Lookaside Buffer (FTLB) with up to 16 KB per entry

2.6.1 Instruction TLB (ITLB)

The ITLB is a 16-entry high speed TLB dedicated to performing translations for the instruction stream. The ITLB maps only 4 KB or 16 KB pages. Larger pages are split into smaller pages of one of these two sizes and installed in the ITLB.

The ITLB is managed by hardware and is transparent to software. The larger VTLB and FTLB structures are used as a backup structure for the ITLB. If a fetch address cannot be translated by the ITLB, the VTLB/FTLB attempts to translate it in the following clock cycle or when available. If successful, the translation information is copied into the ITLB for future use.

2.6.2 Data TLB (DTLB)

The DTLB is a 32-entry high speed TLB dedicated to performing translations for the data stream. The DTLB maps only 4 KB or 16 KB pages. Larger pages are split into one of these configured sizes and installed in the DTLB.

The DTLB is managed by hardware and is transparent to software. The larger VTLB and FTLB structures are used as a backup structure for the DTLB. If a fetch address cannot be translated by the DTLB, the VTLB/FTLB attempts to translate it in the following clock cycle or when available. If successful, the translation information is copied into the DTLB for future use.

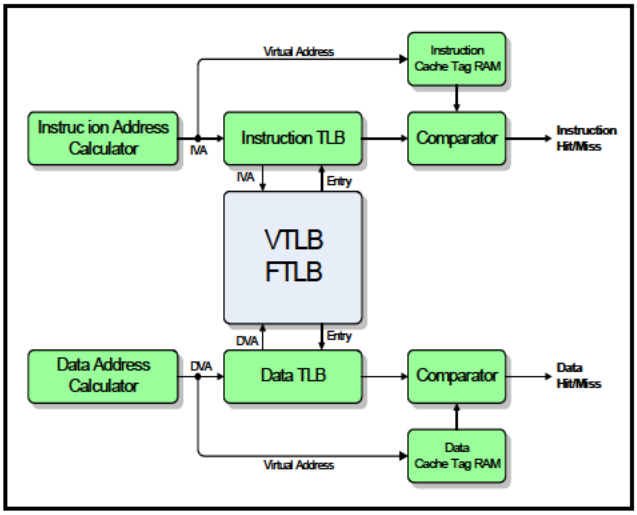
2.6.3 Variable Page Size TLB (VTLB)

The VTLB is a fully associative variable translation lookaside buffer with 64 dual entries that can map variable size pages from 4KB to 256MB. When an instruction address is calculated, the virtual address is first compared to the contents of the ITLB and DTLB. If the address is not found in either the ITLB or DTLB, the VTLB/FTLB is accessed. If the entry is found in the VTLB, that entry is then written into the ITLB or DTLB. If the address is not found in the VTLB,

a software TLB exception is taken. For data accesses, the virtual address is looked up in the VTLB only, and a miss causes a TLB exception.

Figure 3 shows how the ITLB, DTLB, and VTLB/FTLB are implemented in the proAptiv core.

Figure 3 Address Translation



2.6.4 Fixed Page Size TLB (FTLB)

The FTLB is 512 dual entries organized as 128 sets and 4-ways. Each set of each way contains dual data RAM entries and one tag RAM entry. If the tag RAM contents match the requested address, either the low or high RAM location of the dual data RAM is accessed depending on the state of the most-significant-bit (MSB) of the offset portion of the virtual address (VPN2). Each RAM location can only map a fixed page size, which is configurable to 4KB or 16KB.

The FTLB resides at the top of the VTLB range as shown in Figure 4.

Figure 4 VTLB and FTLB

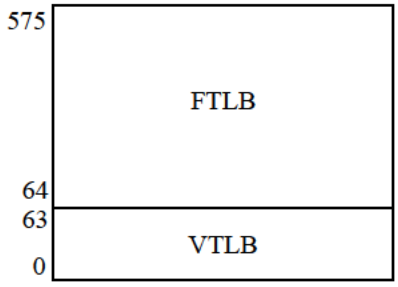
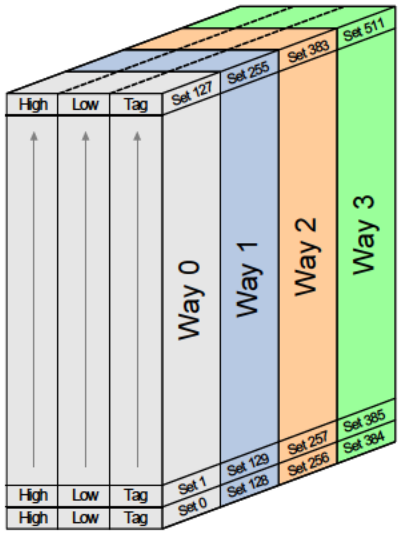


Figure 5 shows a block diagram of the 512-entry FTLB.

Figure 5 FTLB Organization



2.6.5 Enhanced Virtual Address

The proAptiv 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 6.

Figure 6. Remapping Kernel and User Virtual Address Space Using EVA

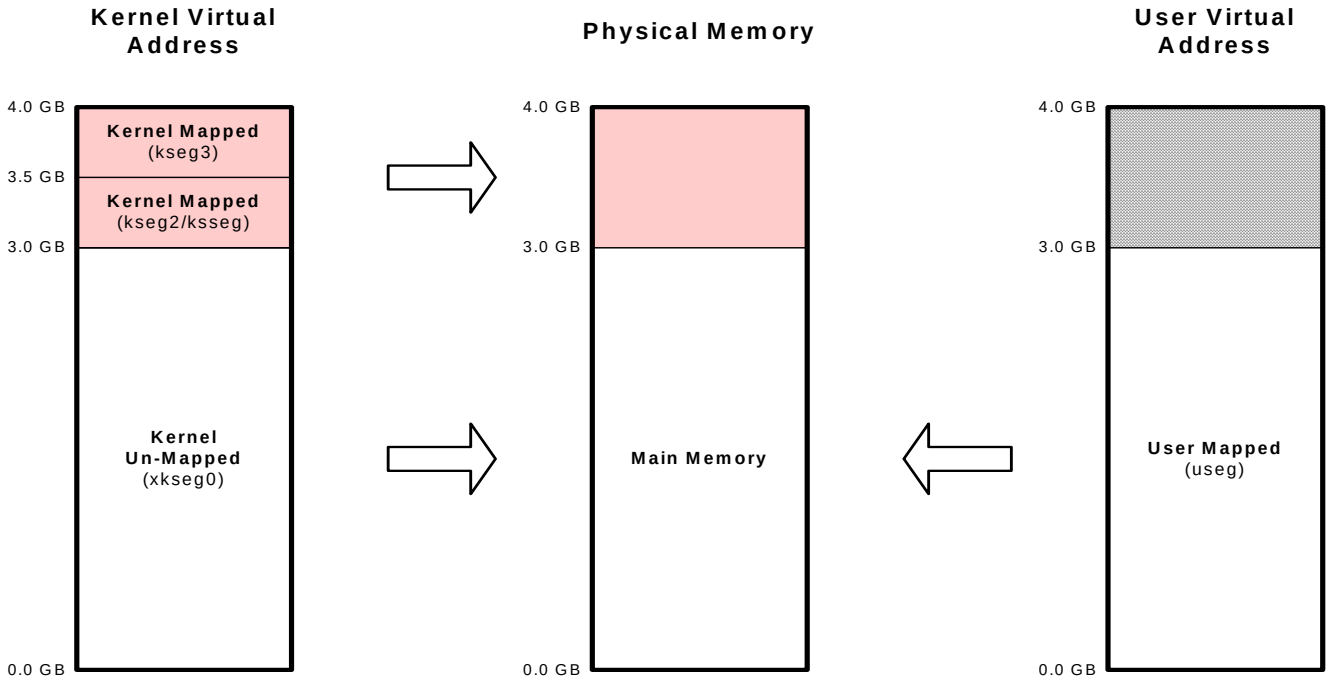


Figure 6 shows an example of how the traditional MIPS kernel virtual address space can be remapped using programmable memory segmentation to facilitate an extended virtual address space. As a result of defining the larger kernel segment as *xkseg0*, the kernel has unmapped access to the lower 3GB of 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.

2.7 Execution Pipelines

The proAptiv core contains the following execution pipelines;

- Arithmetic Logic Pipeline
- DSP Pipeline
- Multiply-Divide Pipeline
- Memory Pipeline
- Branch Pipeline
- Two FPU Pipelines (optional)

Each of these execution units is described in the following subsections. Instruction intended for arithmetic logic pipeline are driven by the out-of-order ALU Decode and Dispatch queue inside the Instruction Issue Unit (IIU) as shown in Figure 2. The other four pipelines are driven by the out-of-order Address Generation unit (AGU) Decode and Dispatch queue also located in the IIU.

2.7.1 Arithmetic Logic Pipeline

The arithmetic unit pipeline consists of one execution unit, called the ALU (Arithmetic Logic Unit), which performs integer instructions such as adds, shifts and bitwise logical operations with a single cycle latency.

If the IIU decodes a single cycle instruction, it is usually sent to the ALU dispatch queue that feeds the arithmetic unit pipeline. This pipeline also contributes to performing ‘fused’ loads. Refer to Section 2.3 for a definition of instruction ‘fusion’.

2.7.2 Digital Signal Processing Pipeline

The DSP pipeline executes a subset of the DSP instructions, including shifts. It can also execute certain arithmetic operations, large shifts and special operations such as counting leading zeroes or ones. Most operations in this unit execute with a two cycle latency.

2.7.3 Multiply/Divide Pipeline

The multiply/divide pipeline executes integer multiplies, integer divides, integer multiply-accumulates and some DSP instructions. The multiply/divide pipeline incorporates a new very high-speed integer divider.

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

The MDU supports execution of one multiply or multiply-accumulate operation every clock cycle whereas divides can be executed as fast as one every six cycles.

2.7.4 Memory Pipeline

The memory pipeline primarily contains the LSU (Load Store Unit), which is responsible for interfacing with the AGU dispatch queue (see Figure 2) and processing load/store instructions to read/write data from data caches and downstream memory.

This unit is capable of handling loads and stores issued out-of-order. The LSU also supports the fission of store instructions by allowing a store's data and address to reach it in any order. This ability to receive loads and stores in almost any order enables very high performance, compared to competing out-of-order machines that do not allow such concurrency. Such instruction-level parallelism allows maximum utilization of the memory pipe resources with minimal area and power.

The LSU can execute loads and stores at twice the rate of regular operations by concatenating data from two 32-bit memory to form a single 64-bit entity. This 'fusion' of instructions allows the LSU to provide almost all the benefits of dual memory access pipes without incurring the area and power costs of multiple tag, data and TLB structures.

The Memory Pipe receives instructions from the Instruction Issue Unit (IIU) and interfaces to the L1 data cache and data scratchpad RAM (DSPRAM). Loads are non-blocking in the proAptiv core. Loads that miss in the data cache are allowed to proceed with their destination register marked unavailable. Consumers of this destination register are held back and replayed as needed once the cache miss has been serviced by the downstream memory subsystem, which includes the high performance L2 cache.

Graduated load misses and store hits and misses are sent in order to the Load/Store Graduation Buffer (LSGB). The LSGB has corresponding data and address buffers to hold all relevant attributes.

An 8-entry Fill Store Buffer (FSB) tracks outstanding fill or copy-back requests. It fills the data cache at the rate of 128-bits per cycle when an incoming line is completely received. Each FSB entry can hold an entire cache line.

The Load Data Queue (LDQ) keeps track of outstanding load misses and forwards the critical data to the main pipe as soon as it becomes available.

Hardware anti-aliasing allows using the core with operating systems that do not support software page coloring. The fully-associative DTLB operates a clock earlier in the LSU pipeline, making use of fast add-and-compare logic to enable virtual address to physical address translations that do not require the area and power expense of virtual tagging. All of this is done completely transparent to software.

2.7.5 Branch Pipeline

The Branch pipeline performs the following functions:

- Executes Branch and Jump instructions
- Performs Branch resolution
- Performs Jump resolution
- Sends the redirect to the Instruction Fetch Unit (IFU)
- Performs a write-back to the Link registers

2.7.6 Floating Point Pipelines

The optional Floating Point Unit (FPU) contains two pipelines; one for arithmetic operations and one for data transfer operations. The arithmetic pipeline executes operations such as multiply, divide, and square root.

The data transfer pipeline executes floating point loads, stores, move operations, and register-to-register transfers between the FPU and the integer unit.

For more information, refer to Section 2.15 "Floating Point Unit (FP2)".

2.7.7 Graduation Unit (GRU)

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

The GRU reads the next set of completed instructions from the global instruction window every cycle and then reads the

corresponding completion buffers and associated information. After processing the exception conditions, the GRU performs the following functions:

- Destination register(s) are updated and the completion buffers are released.
- Graduation information is sent to the IIU so it can update the rename maps to reflect the state of execution results (i.e., GPRs, Accumulators, etc.).
- Resolved branch information is sent to the IFU so that branch history tables can be updated and if needed, a pipeline redirect can be initiated. If sequential control flow is aborted for any reason, the GRU signals all core units to flush and recover microarchitectural state. After recovery is complete, it allows the IIU to resume dispatching instructions.

2.8 Instruction and Data Scratch Pad RAM

The proAptiv 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.

2.9 Bus Interface (BIU)

The Bus Interface Unit (BIU) controls a 64-bit interface to the CM2. The interface implements the Open Core Protocol (OCP).

2.9.1 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 writes, the write buffer can gather multiple writes together and then perform a bursted write in order to increase the efficiency of the bus. Uncached accelerated gathering is supported for any size less than a doubleword.

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.

2.9.2 SimpleBE Mode

To aid in attaching the proAptiv 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 will be generated.

In SimpleBE mode, the *SI_SimpleBE* input pin only controls the byte enables generated by the proAptiv 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 proAptiv core is asserted, hardware sets bit 21 of the *Config* register (*Config*.*SB*) to indicate the device is in simple byte enable mode.

2.10 System Control Coprocessor (CP0)

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

CP0 also contains the state used for identifying and managing exceptions. Exceptions can be caused by a variety of sources, including boundary cases in data, external events, or program errors.

2.11 Interrupt Handling

The proAptiv 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, as defined by Release 3 of the MIPS32 Architecture:

- Interrupt compatibility mode, which acts identically to that in an implementation of Release 1 of the Architecture.
- Vectored Interrupt (VI) mode, which adds the ability to prioritize and vector interrupts to a handler dedicated to that interrupt. The presence of this mode is denoted by the *VInt* bit in the *Config3* register. This mode is architecturally optional. As it is always present on the proAptiv core, the *VInt* bit will always read 1.
- External Interrupt Controller (EIC) mode, which provides support for an external interrupt controller that handles prioritization and vectoring of interrupts. This mode is optional in the Release 2 architecture. The presence of this mode is denoted by the *VEIC* bit in the *Config3* register.

2.12 Modes of Operation

The proAptiv 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, typically used for handling exceptions and operating 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 2.17 "EJTAG Debug Support" for more information on debug mode.

2.13 CorExtend® Unit

The CorExtend unit is a custom block that allows the user to connect to the proAptiv core pipeline with access to all programmer-visible general purpose registers and accumulator state.

MIPS provides a template to define the operand format and latency for the new instruction(s) to be added. Each instruction may select up to 2 source GPRs and/or 1 Accumulator from a set of 32 GPRs and 4 accumulators. The instruction may have a destination of either a GPR, an accumulator, or a private state.

2.14 Coprocessor Interface Unit (CIU)

The CIU provides an interface between the main integer core and the Floating Point Unit (FPU2). The CIU contains a number of queues used to pass data to and from the coprocessors.

Coprocessor1 Load/Store instructions are forwarded to the FPU2. Even though some Coprocessor instructions do not go through the main integer pipeline, they are assigned an instruction identifier. This identifier is tracked in the Graduation Unit to generate a synchronization signal that is used to indicate to the CP1 coprocessor that the instruction has been cleared of all speculation and exception conditions in the integer pipe. Only coprocessor instructions that have reached such a state are allowed to commit results in the Coprocessor.

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

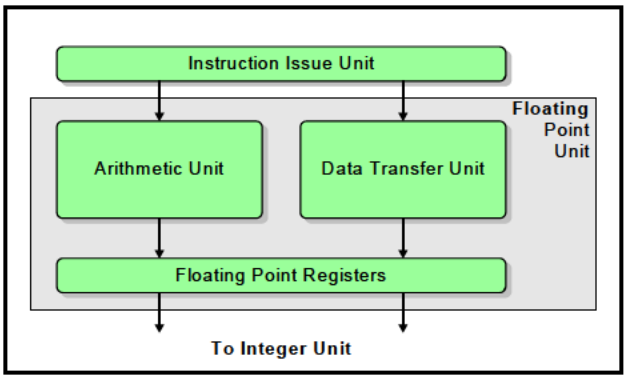
2.15 Floating Point Unit (FP2)

The proAptiv core features an optional IEEE 754 compliant 2nd generation Floating Point Unit (FPU2)[1]. The FP2 contains thirty-two, 64-bit floating point registers used for floating point operations. The FP2 is fully synthesizable and operates at the same clock speed as the CPU.

The FP2 supports both single- and double-precision instructions. It connects to the main processor through the CP1 coprocessor interface. The FPU can read up to 2 instructions from this queue and issue into its execution pipes.

Figure 7 shows a simplified block diagram of the FP2 floating point unit.

Figure 7 Floating Point Unit Block Diagram



2.15.1 FPU Performance

The performance of the FPU is optimized for double-precision formats. Most instructions have a one 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

1. Requires separate MIPS license.

to be the same as executing a MUL and an ADD instruction separately, but the instruction latency, instruction fetch, dispatch bandwidth, and the total number of register accesses required are greatly improved.

IEEE denormalized input operands and results are supported by hardware for many instructions. IEEE denormalized output results are not supported by hardware in general, but a fast flush-to-zero mode is provided to optimize performance. The fast flush-to-zero mode is enabled through the *FCSR* register, and use of this non-standard mode is recommended for best performance when denormalized results are generated. This situation occurs most often in GPU driver code or multimedia CODECS handling real-time data streams.

The FPU has two separate pipelines for floating point instruction execution—one for load/store instructions and another for all other compute instructions. These pipelines operate in parallel with the integer core pipeline and do not stall when the integer pipeline stalls. This allows long-running FPU operations, such as divide or square root, to be partially masked by system stalls and/or other integer unit instructions.

Arithmetic instructions are always dispatched and graduated in order, but loads and stores can complete out-of-order. The integer *core* will perform the data access for load/store operations and transfer data to and from the FPU using the CIU. Load data may arrive in the FPU out-of-order relative to program order. The exception model is ‘precise’ at all times.

2.16 proAptiv Core Power Management

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

2.16.1 Instruction-Controlled Power Management

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

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

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

2.16.2 Register Controlled Power Management

The *RP* (Reduced Power) bit in the CP0 *Status* register enables a standard software mechanism for placing the system into a low-power state. The state of the *RP* bit is available externally on the *SI_RP* output signal. Three additional pins— *SI_EXL*, *SI_ERL*, and *EJ_DebugM*—support the power-management functions by allowing the user to change the power state if an exception or error occurs while the core is in a low-power state.

Setting the *RP* bit of the CP0 *Status* register causes the core to assert the *SI_RP* signal. The external agent can then decide to reduce the clock frequency and place the core into power-down mode.

If an interrupt occurs while the device is in power-down mode, that interrupt may need to be serviced, depending on the needs of the application. The interrupt causes an exception, which in turn causes the *EXL* bit to be set. Setting the *EXL* bit causes the assertion of the *SI_EXL* signal on the external bus, indicating to the external agent that an interrupt has occurred. When *SI_EXL* is asserted, the external agent can choose to either speed-up the clocks and service the interrupt or let it be serviced at the lower clock speed.

The setting of the *ERL* bit causes the assertion of the *SI_ERL* signal on the external bus, indicating to the external agent that an error has occurred. The external agent can then choose to either speed up the clocks and service the error or let it be serviced at the lower clock speed.

Similarly, the *EJ_DebugM* signal indicates that the processor is in debug mode. Debug mode is entered when the processor takes a debug exception. If fast handling of this is desired, the external agent can speed up the clocks.

The core provides four power-down signals that are part of the system interface. Three of the pins change state as the corresponding bits in the CP0 *Status* register are set or cleared, and the fourth pin indicates that the processor is in debug mode:

- The *SI_RP* signal represents the state of the *RP* bit (27) in the CP0 *Status* register.
- The *SI_EXL* signal represents the state of the *EXL* bit (1) in the CP0 *Status* register.
- The *SI_ERL* signal represents the state of the *ERL* bit (2) in the CP0 *Status* register.
- The *EJ_DebugM* signal indicates that the processor has entered debug mode.

2.17 EJTAG Debug Support

The proAptiv core includes an Enhanced JTAG (EJTAG) block for use in software debugging of application and kernel code. For this purpose, in addition to standard user/supervisor/kernel modes of operation, the proAptiv core provides a Debug mode.

Debug mode is entered when a debug exception occurs (resulting from a hardware breakpoint, single-step exception, etc.) and continues until a debug exception return (DERET) instruction is executed. During this time, the processor executes the debug exception handler routine.

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

There are several types of simple hardware breakpoints defined in the EJTAG specification. These breakpoints stop the normal operation of the CPU and force the system into debug mode.

During synthesis, the proAptiv core can be configured to support the following breakpoint options:

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

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

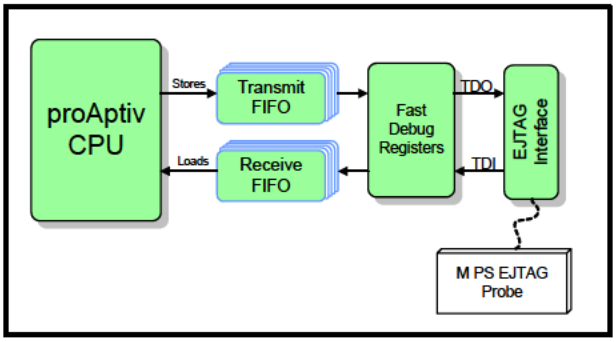
Data breakpoints occur on load and/or store transactions. Breakpoints are set on virtual address and address space identifier (ASID) values, similar to the Instruction breakpoint. Data breakpoints can also be set based on the value of the load/store operation. Finally, masks can be applied to the virtual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

In debug mode, EJTAG can request that a 'soft' reset be masked. This request is signalled via the *EJ_SRstE* pin. When this pin is deasserted, the system can choose to block some sources of soft reset. Hard resets, such as power-on reset or a reset switch, should not be blocked by this signal. This reset pin has no effect inside the core.

2.17.1 Fast Debug Channel

The proAptiv CPU includes the EJTAG Fast Debug Channel (FDC) as a mechanism for efficient bi-directional 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 8 Fast Debug Channel



2.17.2 MIPS Trace

The proAptiv core includes optional MIPS Trace support for real-time tracing of instruction addresses, data addresses, data values, performance counters, and processor pipeline inefficiencies. The trace information is collected in an on-chip or off-chip memory, for post-capture processing by trace regeneration software. Software-only control of trace is possible in addition to probe-based control.

An optional on-chip trace memory may be configured in size from 256B to 8 MB; it is accessed either through load instructions or the existing EJTAG TAP interface, which requires no additional chip pins.

Off-chip trace is managed with the PIB2 (2nd-generation Probe Interface Block) hardware that ships with the product. It provides a selectable trace port width of 4, 8, or 16 pins plus DDR clock. Trace data is streamed on these pins and captured using the MIPS Navigator™ Pro probe.

3. Multiprocessing System

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

3.1 Cluster Power Controller (CPC)

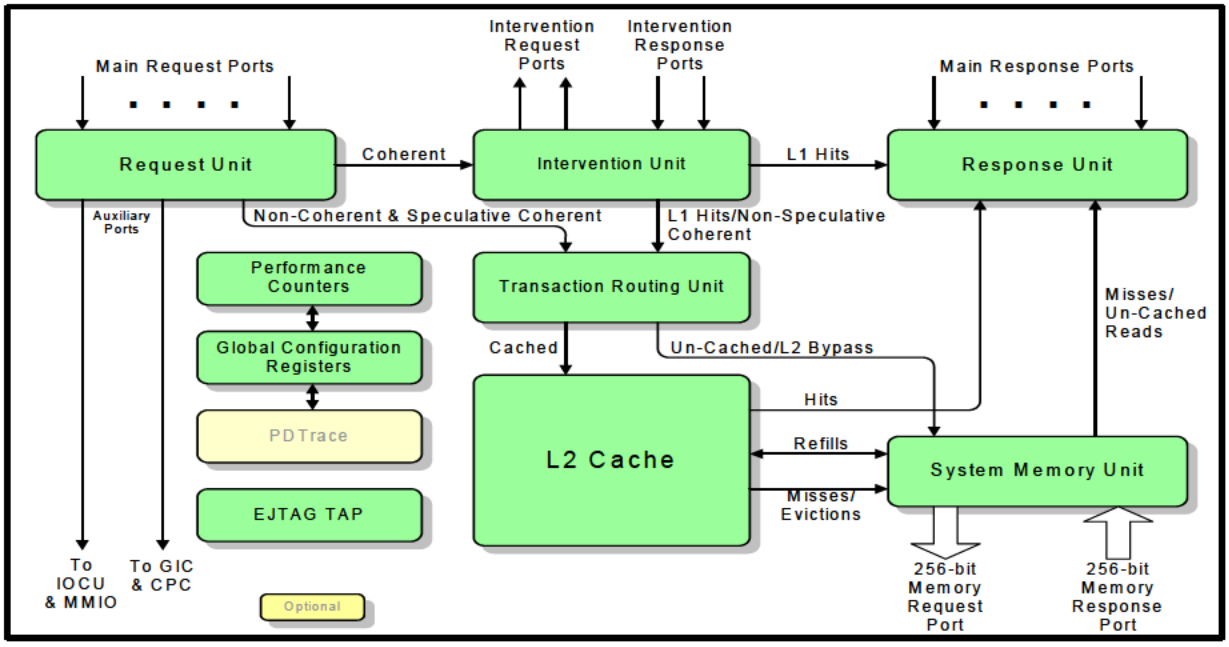
Individual CPUs within the cluster can have their clock and/or power gated off when they are not in use. This gating is managed by the Cluster Power Controller (CPC). The CPC handles the power shutdown and ramp-up of all CPUs in the cluster. Any proAptiv CPU that supports power-gating features is managed by the CPC.

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

3.2 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 9. Each of the blocks is described in more detail in the following subsections.

Figure 9. Coherence Manager with Integrated L2 Cache (CM2) Block Diagram



3.2.1 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.

3.2.2 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).

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.

3.2.3 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.

3.2.4 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.

3.2.5 Transaction Routing Unit

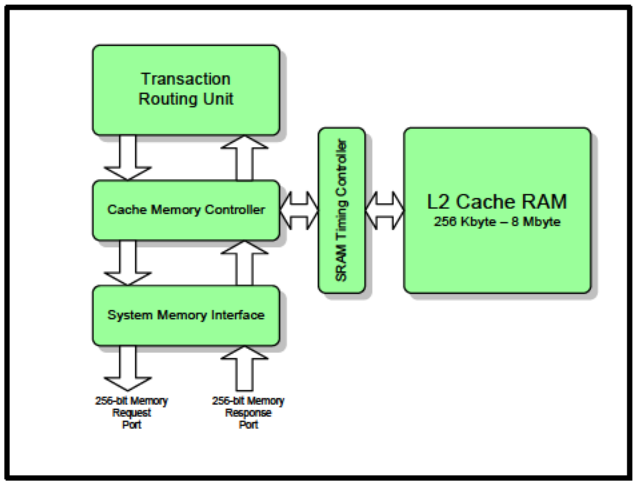
The Transaction Routing Unit (TRU) arbitrates between requests from the RQU and IVU, and routes requests to either the L2 or the SMU. The TRU also contains the request and intervention data buffers which are written directly from the RQU and IVU, respectively. The TRU reads the appropriate write buffer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

3.2.6 Level 2 Cache

The unified 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 10 L2 Cache Block Diagram



L2 Cache Configuration

For a list of L2 cache configuration options, refer to Section Table 1. "Build-Time Configuration Options" at the back of this document.

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

3.2.7 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.

3.2.8 PDTrace Unit

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

3.2.9 Performance Counter Unit

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

3.2.10 Coherence Manager Performance

The CM2 has a number of high performance features:

- 256-bit wide internal data paths throughout the CM2
- 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.

3.3 I/O Coherence Unit (IOCU)

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

The IOCU acts as an interface block between the Coherence Manager (CM2) 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 CM2. Each request from an I/O device may be marked as coherent, non-coherent cached, or uncached. Coherent requests access the L1 and L2 caches. Non-coherent cached requests access only the L2 cache. Uncached requests bypass both the L1 and L2 caches and are routed to main memory. An example system topology is shown in Figure 11.

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

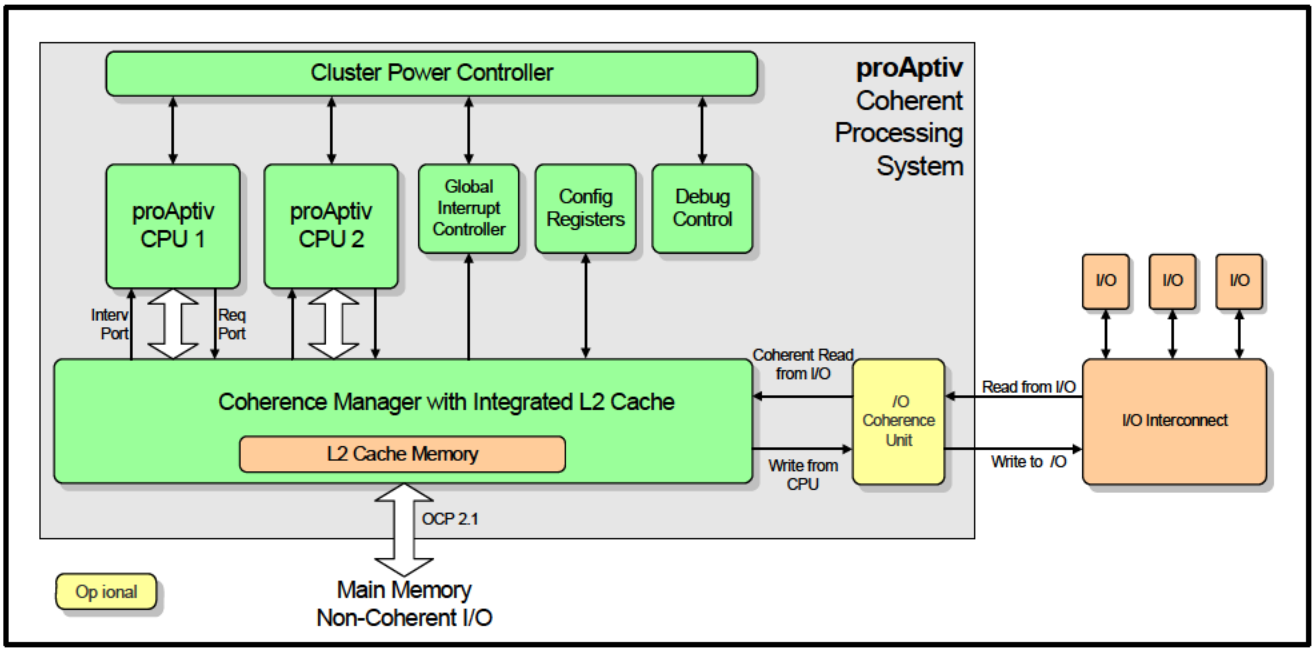
The reference IOCU design provides several features for easier integration:

- A user-defined mapping unit can define cache attributes for each request—coherent or not, cacheable (in L2) or not, and L2 allocation policy.
- Supports incremental bursts up to 16 beats (128 bits) on I/O side. These requests are split into cache-line- sized requests on the CM side.
- Ensures proper ordering of responses for the split requests and tagged requests.

In addition, the 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 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.

Figure 11. Role of the IOCU in a Two-Core Multiprocessing System



3.3.1 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.

3.4 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 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.

3.5 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.

3.5.1 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 proAptiv 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.

Note that unused CPUs are not held in reset until released by writing into the configuration registers. Rather, unused CPUs remain powered down and are held isolated towards the rest of the cluster. If power gating is not selected for a given implementation, unused CPUs are powered but receive no clock and remain isolated until activated by the CPC.

In addition to controlling the deassertion of the CPC reset signal, there are memory-mapped registers that can set the value for each CPU's *SI_ExceptionBase* pins. This allows different boot vectors to be specified for each of the 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.

3.5.2 Inter-CPU Debug Breaks

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

3.5.3 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, etc.).
- *Control Options*: Various features of the CM2 can be disabled or configured. Examples of this are disabling speculative reads and preventing ReadShared requests from being upgraded to Exclusive.

4. Clocking Options

The proAptiv core has the following clock domains:

- Cluster domain — This is the main clock domain, and includes all proAptiv 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, 1:1.5, 1:2, 1:2.5, 1:3, 1:3.5, 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 12 shows a diagram with the four clock domains.

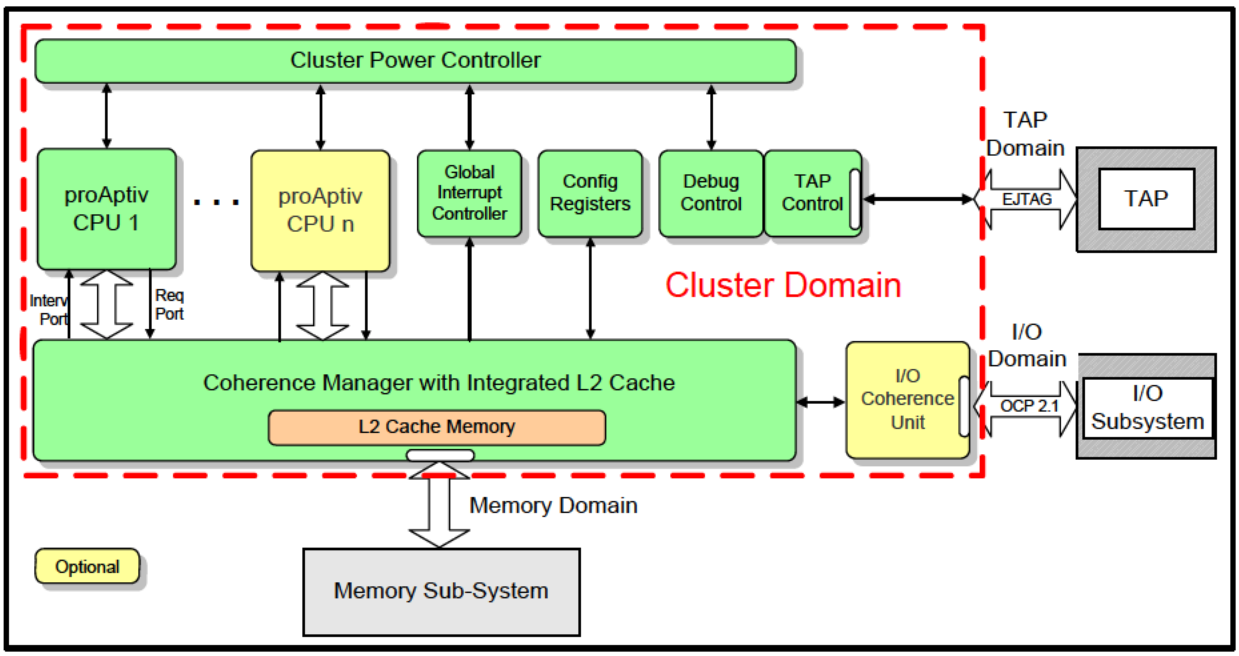
5. Design For Test (DFT) Features

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

5.1 Internal Scan

The proAptiv 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.

Figure 12 proAptiv™ MPS Clocking Domains



5.2 Memory BIST

The proAptiv core provides an integrated memory BIST solution for testing of all internal SRAMs. 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.

6. Build-Time Configuration Options

The proAptiv Multiprocessing System allows a number of features to be customized based on the intended application.

Table 1 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*® proAptiv™ *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 1. Build-Time Configuration Options

Configuration Option	Choices	Software Visibility (*Register*$_{FIELD}$)
System Options		
Number of CPUs	1, 2, 3, 4, and 6	*GCR_CONFIG*$_{PCORES}$
I/O Coherence Unit	Present or not	*GCR_CONFIG*$_{NUMIOCU}$
Memory Connection ID mask	0-8b	N/A
PDTrace support	Present or not	*CONFIG3*$_{TL}$
PDTrace memory location	On-core, off-chip, or both	*TCBCONFIG*$_{OnT}$, *TCBCONFIG*$_{OfT}$
PDTrace on-chip memory size	256B - 8MB	*TCBCONFIG*$_{SZ}$
MIPS Trace triggers[1]	0-8	*TCBCONFIG*$_{TRIG}$
Probe Interface Block - Number of data pins	4, 8, 16	N/A
Per-CPU Options		
I-cache size	32 or 64 KB	*CONFIG1*$_{IL}$, *CONFIG1*$_{IS}$, *CONFIG1*$_{IA}$
D-cache size	32 or 64 KB	*CONFIG1*$_{DL}$, *CONFIG1*$_{DS}$, *CONFIG1*$_{DA}$
D-cache hardware aliasing support	Present or not. For 32 or 64 KB only. MMU type is TLB.	*CONFIG7*$_{AR}$
Floating point unit	Present or not	*CONFIG1*$_{FP}$
FTLB Memory	Present or not	*CONFIG*$_{MT}$
Data ScratchPad RAM interface	Present or not	*CONFIG*$_{DSP}$
Instruction ScratchPad RAM interface	Present or not	*CONFIG*$_{ISP}$
Clock gating	Top-level, block-level, fine-grain, D-cache, or none	N/A
CorExtend Block	Present or not	*CONFIG*$_{UDI}$[2]

Table 1. Build-Time Configuration Options (Continued)

Configuration Option	Choices	Software Visibility ($Register_{FIELD}$)
Repeat rate for CorExtend instructions using private state	1 through 15	N/A
Number of CorExtend completion buffers	1 through 15	N/A
Sideband inputs to external CorExtend module	Bus width (in bits)	N/A
Sideband outputs to external CorExtend module	Bus width (in bits)	N/A
Memory BIST	Integrated (March C+ plus IFA-13) or custom	N/A
Global Interrupt Controller Options		
Number of system interrupts	8*[1-32]	$GIC\_SH\_CONFIG_{NUMINTERRUPTS}$
Enable routing of interrupts to each core (per core)	Yes/No	N/A
Implement External Interrupt Controller (EIC) per core	Yes/No	$GIC\_VPEi\_EICSS_{j}$
External Interrupt Controller (EIC) enabled by default at boot time	Yes/No	$GIC\_VPEj\_CTL_{EIC\_MODE}$
Coherence Manager Options		
Number of Address Regions	0, 4, 6	$GCR\_CONFIG_{NUM\_ADDR\_REGIONS}$
Default Global Configuration Registers (GCR) base address and writeability	Any 32 KByte-aligned physical address Hardwired or programmable	GCR_BASE
Default Exception Base for each CPU	Any 4 KByte-aligned physical address	$GCR\_Cx\_RESET\_BASE$
Advanced CM2 Config - fine tuning of buffer sizes, etc.	Varies	N/A
MIPS Trace source field bits in trace word	0, 2, or 4	$TCBCONTROLB_{TWSrcWidth}$
L2 Cache Options		
Cache Size	256, 512, 1024, 2048, 4096, or 8192 KBytes	$CONFIG2_{SS,}$ $CONFIG2_{SL,}$ $CONFIG2_{SA,}$
Cache Line Size	32 bytes or 64 bytes	$CONFIG2_{SL}$
ECC Protection	ECC or no ECC	Build-time choice with run-time enable via $ErrCtl_{L2EccEn}$
Number of Outstanding Read Misses	8, 12, or 15	N/A
Data SRAM Latency	1 or 2 cycles (2 cycle option requires pipelined SRAMs)	N/A
Memory BIST	Integrated (March C+ plus IFA-13) or custom	N/A
Clock Gating	Top-level/fine-grain, none	N/A
IOCU Options		
IODB implementation style	Flops or generator	N/A
Cluster Power Controller Options		
Microstep delay in cycles	1-1024	N/A
RailEnable delay	1-1024	N/A

Table 1. Build-Time Configuration Options (Continued)

Configuration Option	Choices	Software Visibility (*Register*$_{FIELD}$)
Power Gating Enabled	Enabled or not	N/A
Clock Gating Enabled	Enabled or not	N/A

1. These are not hardware execution and data triggers. They provide break and trace control from external (to the core) sources via the *TC_ChipTrigIn* and *TC_ProbeTrigin* signals.
2. These bits indicate the presence of external blocks. Bits will not be set if interface is present, but block is not.

7. Revision History

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

Revision	Date	Description
1.00	September 21, 2012	Preliminary release of proAptiv datasheet.
1.01	March 6, 2013	Initial release of proAptiv datasheet.
1.02	March 23, 2013	Made minor updates. Indicated L1 instruction and data cache parity as non-optional.