OCTEON®

Programmer's Guide

The Fundamentals

Introduction Version for
Cavium Networks University Program

www.caviumnetworks.com

CAVIUM NETWORKS

CAVIUM NETWORKS

OCTEON®
Programmer's Guide

The Fundamentals

Part Number: CN_OCTEON_PRG_GUIDE_Vol1A-EDU

July 2010

OCTEON®
Programmer's Guide

The Fundamentals

June Curtis

Published by Cavium Networks

PUBLISHED BY
Cavium Networks
805 East Middlefield Road
Mountain View, CA 94043
Phone: 650-623-7000
Fax: 650-625-9751
Email: sales@caviumnetworks.com
Web: http://www.caviumnetworks.com/

Preface

About This Book

This volume of the *OCTEON® Programmer's Guide* provides fundamental information needed by software engineers to develop code to run on the OCTEON processor.

This volume contains the following chapters:

1. *Introduction*
2. *Packet Flow*
3. *Software Overview*
4. *SDK Tutorial*
5. *Software Debugging Tutorial*
6. *OCTEON Application Performance Tuning Whitepaper*
7. *Glossary*

The appendices relevant to a chapter are not collected at the end of the book, but instead are included with the corresponding chapter.

The chapters should be read in order, except for the appendices. Each chapter builds on information provided in the previous chapter.

If you have any suggestions for improvements or amendments, or believe you have found errors in this publication, please notify us at university@caviumnetworks.com.

The following sections briefly introduce the contents of the *OCTEON Programmer's Guide, Volume 1*.

Chapter 1: Introduction

This chapter provides an overview of Cavium Network's OCTEON® processor family, and introduces key features provided by its members.

This chapter also provides a brief discussion of the advantages of some of the OCTEON processor's key features:

- Integrated hardware accelerators
- Per-core Security Coprocessors
- On-chip interconnects
- Special Cavium Networks-specific instructions
- Cache hierarchy

Chapter 2: Packet Flow

This chapter provides a detailed description of how the packet-management units offload and cooperate with the cores to accelerate packet flow through the OCTEON processor. This chapter includes details on how the Scheduling/Synchronization/and Order (SSO) Unit off loads the cores.

This chapter is divided the following sections:

- Packet Flow Overview: A typical packet flow (packet received, processed, transmitted) will be discussed, showing how the different functional blocks work together to create reliable, fast packet processing.
- Special Hardware Features to Accelerate Packet Processing: The key packet-management hardware acceleration features are discussed:
  1. management of packet classification and priority
  2. buffer management
  3. packet-linked locks
  4. management of packet order
- The Schedule / Synchronization / Order (SSO) Unit: The Schedule / Synchronization / Order Unit (SSO) is introduced. The SSO provides essential capabilities which are unique to the OCTEON processor; the SSO is the "heart" of OCTEON. Because of the SSO's importance, the rest of this chapter introduces its hardware acceleration features, and describes how software may take advantage of them.

Chapter 3: Software Overview

This chapter provides an overview of the OCTEON processor's software-related topics, including software architecture, multicore issues, scaling, and memory management.

The chapter introduces the following topics:

- cnMIPS® cores
- Simple Executive API (HAL)
- Different runtime environment choices
- Software architecture issues
- Application Binary Interfaces (ABIs) supported
- Tools: Cross-Development and Native Toolchains
- Physical Address Map and Caching on the OCTEON processor
- Virtual Memory
- Bootmem Global Memory
- Shared memory
- Bootloader
- Software Development Kit (SDK) code conventions

Throughout the chapter relevant SDK documents are referenced to help the reader find more detailed information.

Chapter 4: SDK Tutorial

This chapter introduces the Software Development Kit (SDK) from a hands-on perspective, provides a tour of the installed SDK, and also contains useful information for users new to embedded software development or new to Linux®.

This chapter is designed to augment the SDK documentation by providing a high-level view of the SDK and step-by-step instructions from installing the SDK to running example code on an evaluation board.

The hands-on sections in this tutorial are:

- System Administration Tasks
- Connect the Development Target
- Viewing the Target Board Console Output
- Gather Key Hardware Information
- Install the SDK
- Tour the Installed SDK
- Build and Run a SE-S Application (`hello`)
- Run hello on Multiple Cores
- Build and Run Linux
- Run a SE-UM Example (`named-block`)
- Run `linux-filter` as a SE-S Application (Hybrid System)
- Run `linux-filter` as a Linux SE-UM Application
- Run `linux-filter` as a SE-UM Application on Multiple Cores
- Creating a Custom Application

Chapter 5: Software Debugging Tutorial

This chapter provides information and hands-on steps to help users get started with embedded software debugging using GDB, including:

- hardware configuration
- debugging both standalone and PCI development targets
- multicore debugging

The hands-on sections in this tutorial are:

- Debug a SE-S Application: `hello`
- Debug the Linux Kernel
- Debug a SE-UM Application: `named-block`
- Debugging a SE-S Application on the OCTEON Simulator
- Debugging Linux on the OCTEON Simulator
- Building `vmlinux` to Run on the Simulator
- Running Linux User-Mode Applications on the Simulator

About OCTEON Application Performance Tuning Whitepaper

This whitepaper provides information on how to optimize software performance by taking advantage of the OCTEON processor's unique features.

This whitepaper describes common areas where changes can bring big performance improvements. Many of the performance improvement techniques presented in this document are industry-standard; others take advantage of Cavium Networks-specific hardware acceleration.

This whitepaper addresses both designing for high performance, and post-development performance tuning. Both single core and multiple-core (scaling) issues are discussed.

Performance tuning issues are separated into four sections:

1. Software Architecture for High Performance
2. Tuning the Minimum Set of Cores
3. Tuning Multicore Applications (Scaling)
4. Linux-Specific Tuning

Within each section, performance tuning choices are presented from the easiest to most difficult to implement.

Information is also provided on performance evaluation tools.

Performance tuning is both an art and a science. This whitepaper does not attempt to cover all the possibilities, only some of the more common ones.

Glossary

The glossary contains terms defined in this volume. Some common industry terms are also provided for convenience.

Softcopy of Chapters

OCTEON Programmer's Guide chapters are also available at the Cavium Networks support site at https:/support.caviumnetworks.com/.

Where to Get More Information

Other resources include the extensive documentation supplied with the SDK, the *Hardware Reference Manual* (*HRM*), whitepapers and application notes. All of these are available at the support site at https://support.caviumnetworks.com/. As new *OCTEON Programmer's Guide* chapters are published, they are made available at the support site.

MIPS® Architecture manuals are available from: http://www.mips.com/. The key manuals are:

- *MIPS64® Architecture for Programmers Volume I: Introduction to the MIPS64® Architecture*
- *MIPS64® Architecture for Programmers Volume II: The MIPS64® Instruction Set*
- *MIPS64® Architecture for Programmers Volume III: The MIPS64® Privileged Resource Architecture*

There are two excellent MIPS guides written by Dominic Sweetman. See *MIPS Run Linux* is currently the newer of the two books.

Sweetman, Dominic, *See MIPS Run*, ISBN-10: 1558604103; ISBN-13: 978-1558604100
Sweetman, Dominic, *See MIPS Run Linux*, ISBN-10: 0120884216; ISBN-13: 978-0120884216

Conventions Used in This Book

The following typographical conventions are used in this book:

Italic	is used for document names, new terms, special terms such as *root*, comments in code, notes in tables and figures, and for emphasis
`Constant Width`	is used for code, commands, contents of files, file names, and directory names
Italic Bold	is used for notes in text
`Constant Bold`	is used to indicate what to type on the command line, and for commands used in figures
System	is used for keyboard keys

Acknowledgments

Several key people from the Cavium Networks team contributed to this book, including representatives from hardware engineering, software engineering, architecture, technical marketing engineering, applications engineering, marketing, and sales.

Because of their efforts, each chapter provides experience, depth, and perspective which are not available to one person alone.

Summary Table of Contents

Chapter 4: Software Development Kit (SDK) Tutorial

Chapter 5: Software Debugging Tutorial

Chapter 6: OCTEON Application Performance Tuning Whitepaper

Introduction

TABLE OF CONTENTS

LIST OF FIGURES

1 Introduction

This chapter provides an overview of Cavium Network's OCTEON® processor family, and introduces key features provided by its members.

This chapter also provides a brief discussion of the advantages of some of the OCTEON processor's key features:

- Integrated hardware accelerators
- Per-core Security Coprocessors
- On-chip interconnects
- Special Cavium Networks-specific instructions
- Cache hierarchy

Note: Throughout this chapter, OCTEON model-specific hardware components are marked with an asterisk (*). See the *Hardware Reference Manual* (*HRM*) for your specific part number for more details.

2 Introducing the OCTEON Processor Family

The OCTEON family consists of three generations of software-compatible, highly integrated multicore products: OCTEON®, OCTEON® Plus and OCTEON® II (CN3XXX, CN5XXX, and CN6XXX, respectively). These processors are optimized to provide high-performance, high-bandwidth, and low power consumption.

The OCTEON processor can be used for control-plane applications, data-plane applications, or a hybrid of both. The OCTEON processor is an ideal solution for intelligent networking, wireless, and storage applications from 100 Mbps to 40 Gbps.

All OCTEON products share the same architecture, which enables software compatibility across the entire family. Individual OCTEON products vary and scale based on the:

- Number of integrated cnMIPS® cores
- Frequency of the cores and the internal interconnects
- Type and number of integrated I/O interfaces
- Type and number of integrated hardware acceleration units
- Size and associativity of the caches

Each OCTEON product offers a feature set which is optimized for the specific functionality and performance needs of the target applications. Software written for one OCTEON model will run on another OCTEON model as long as the required features are available.

Cavium Networks is actively engaged in adding new features and enhancements to the OCTEON family. For more information on the latest additions and the future roadmap, contact your Cavium Networks sales representative.

2.1 Target Applications

The OCTEON processors are used in a wide variety of OEM equipment. Some examples include routers, switches, unified threat management (UTM) appliances, content-aware switches, application-aware gateways, triple-play broadband gateways, WLAN access and aggregation devices, 3G, WiMAX and LTE basestation and core network equipments, storage networking equipment, storage systems, servers, and intelligent network adapters.

2.2 Key Features

This section provides a brief overview of key OCTEON features. Because the OCTEON II models are in development as this chapter is being written, some of the features listed are subject to change.

Note: Features which are optional are marked with an asterisk (*).

Up to 32 cores, up to 1.5 GHz: The OCTEON family of multicore processors supports up to 32 cnMIPS cores with speeds ranging from 300 MHz to 1.5 GHz.

- The OCTEON and OCTEON Plus models have from 1 to 16 cores, at speeds ranging from 300 MHz to 800 MHz.
- The OCTEON II models have up to 32 cores, with speeds up to 1.5 GHz.

Hardware Acceleration Units: Multiple hardware acceleration units are integrated into each OCTEON processor. These hardware acceleration units offload the cores, reducing software overhead and complexity. These acceleration units include:

- Packet-management accelerators
- Security accelerators
- Application accelerators
- Specialized accelerators

Dedicated DMA Engines: Dedicated DMA Engines are provided for each hardware unit which accesses memory. Additional memory-to-PCI PCIe®/PCI/PCI-X DMA Engines are present in some models.

High-Speed Interconnects: The hardware units and the cores are connected by high-speed interconnects. These interconnects run at the same frequency as the cores. Each interconnect is a collection of multiple buses with extensive pipelining and sophisticated hardware arbitration logic. The width and placement of the buses are optimized to streamline packet data flow, eliminating potential bottlenecks. Some of the OCTEON II models include a cross-bar hyper-connect to scale up to 32 cores and higher.

Industry-Standard Toolchains and Operating Systems: Industry-standard toolchains (GCC, GDB) and operating systems (including SMP Linux) have been modified to utilize the OCTEON processor's multiple cores, hardware acceleration units, and special Cavium Networks-specific instructions. Users can easily write C/C++ code, and can re-use legacy software. Programs written for MIPS64 and MIPS32 ISA are inherently supported.

Flexible Software Architecture: The hardware architecture allows flexible software architecture design, including the ability to group cores as desired to add more performance where it is needed. A common configuration is data-plane plus control-plane. In this configuration, one group of cores runs a data-plane application; another group of cores runs SMP Linux or another general-purpose operating system to provide the control-plane functionality. If needed, cores can be added to the data-plane, resulting in linear performance scaling.

Streamlined Software Development: Software development complexity is minimized by hardware acceleration units, flexible software architecture, standard MIPS64 ISA, and industry-standard toolchains and operating systems. A Cavium Networks Software Development Kit (SDK) is provided. The SDK includes the GNU C/C++ compiler and other development tools, C-language Application Programmer's Interfaces (APIs) to the hardware units, a simple executive which can run code on the cores without any operating system, and Cavium Networks SMP Linux. Optional software packages are available to support more complex features.

Packet-management Acceleration: Packet receive/transmit is automated by software-configurable packet-management accelerators. Accelerations include:

- Packet data buffers are automatically allocated and freed
- Layer-2 through layer-4 packet header parsing, exception checks and checksum calculation
- Up to 7-tuple flow classification with VLAN stacking support
- Packet data is automatically stored in L2/DRAM on ingress
- Packet ordering and scheduling is automatically managed by hardware without requiring explicit software locking
- Packet data transmission is managed by a hardware accelerator

TCP/UDP Acceleration: TCP/UDP acceleration features include:

- Packet-management accelerations
- Automated packet header checking on receive
- Automated TCP/UDP checksum generation on transmit
- Timer Unit supports efficient implementation of TCP retransmission

Per-Core Security Hardware Acceleration*: Common security algorithms are accelerated by optional per-core Security Engines. (See the *HRM* for a complete list of hardware accelerated algorithms.) In the following list, a double asterisk (**) denotes security algorithms which are not present in all Security Engines. The hardware accelerated algorithms include:

- Large multiply-and-accumulate unit for fast modular exponentiation needed for RSA and Diffie-Hellman operations
- Security hash algorithms:
  - MD5, SHA-1, SHA-224, SHA-256, SHA-384, and SHA-512, AES XCBC HMAC
- Symmetric cryptographic algorithms:
  - 3DES and DES in ECB and CBC modes
  - AES in CBC, ECB, CTR, LRW, ICM, XTS, GCM, and CCM modes
  - RC4
  - KASUMI** (OCTEON Plus, OCTEON II)
  - SNOW 3G** (OCTEON II)
  - SMS4** (OCTEON II)
- Asymmetric key operations:
  - RSA, DSA, DH
- TKIP Operations: TKIP
- Galois field multiplication (used in both security, such as SNOW 3G, and RAID calculations)

Per-Core CRC Engines: CRC generation is accelerated by the per-core CRC Engines:

- Hardware CRC calculation (up to 32 bits): For example, CRC10 accelerates AAL2 and CRC32 accelerates AAL5 protocol processing.
- CRC hardware also accelerates ROHC (Robust Header Compression) protocol processing and iSCSI checksum calculation/verification.

FIPS Certification Support: Other features which facilitate high-level FIPS (Federal Information Processing Standards) certification include:

- NIST-certified algorithms
- A cryptographically secure Random Number Generator (RNG) hardware unit. This unit has been designed to handle upcoming FIPS standards.
- Secure on-chip memory* for security keys which cannot be accessed through I/O interfaces
- A pin for zeroing out all the stored keys
- Restricted PCI host access

Storage Application Acceleration: Storage applications are accelerated by:

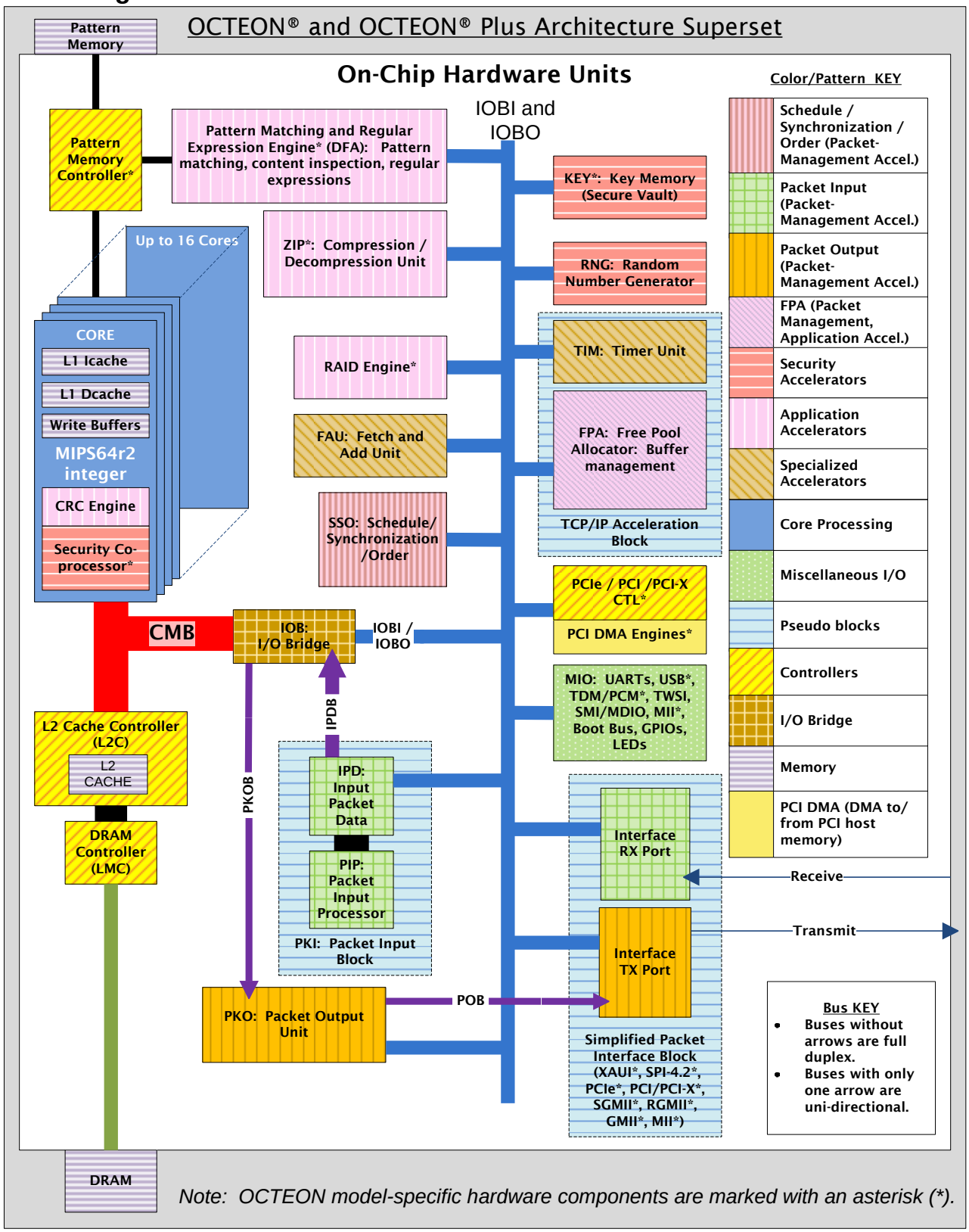
- RAID Engine*: RAID/XOR Acceleration for RAID 5 and RAID 6
- Per-core Security Engines*
- Galois field multiplication (per-core security acceleration) can also be used for RAID calculations.
- De-duplication acceleration

Other architectural features include the following list (optional features are marked with an asterisk (*)):

- MIPS64 release 2 integer Instruction Set Architecture (ISA)
- Additional Cavium Networks-specific instructions, enhancing the MIPS core to create the cnMIPS core. Most of these instructions are automatically generated by the C/C++ compiler.
- Dual-issue ALU with additional security acceleration coprocessor units. Combining two instruction issues together with additional security units, it is possible for more than two operations to be simultaneously executing in a given cycle.
- A cache hierarchy including:
  - L2 cache with ECC protection, ranging from 256 KB to 2 MB (OCTEON II: up to 4 MB), shared by the cores and I/O subsystem
  - L1 instruction cache (Icache) with parity protection, 32 KB (OCTEON II: up to 37 KB), per core
  - L1 data cache (Dcache) with parity protection, 8 KB to 16 KB (OCTEON II: up to 32 KB), per core
- A 32-entry to 64-entry TLB (OCTEON II: up to 128-entry) which supports:
  - variable page sizes from 4K to 256 MB
  - read and execute inhibit per-page options (used to protect against overflow attacks and malicious code)
- Memory options include:
  - DDR2 from DDR2-400 up to DDR2-800 for OCTEON and OCTEON Plus
  - DDR3 up to DDR3-1600 for OCTEON II
- Per-core Write Buffer with aggressive write combining, reducing unnecessary traffic on the buses by limiting the number of writes to memory
- Industry-standard I/O Interfaces: XAUI*, SPI-4.2*, PCIe*, PCI/PCI-X*, SGMII*, RGMII*, GMII*, MII*, (OCTEON II: serial RapidIO®* (sRIO), and Interlaken*)
- Support for NOR and NAND* flash
- Boot from NOR flash (CN52XX supports boot from NAND)
- Misc I/O Including: UARTs, USB 2.0* (including PHY), TDM/PCM*, TWSI, SMI/MDIO, MII*, Boot Bus, GPIOs, LEDs
- PCIe/PCI/PCI-X DMA Engines* to DMA to/from PCI host memory or from memory to memory
- Pattern Memory Controller*: used to connect pattern memory to the cores and to the Pattern Matching and Regular Expression Engine.

The following figure shows the OCTEON processor's on-chip hardware units in an idealized way because it shows all of the model-specific hardware components in one superset. The OCTEON II processor features are not included in this diagram. The OCTEON II processors will contain many of the features shown in the figure below, and will also contain new and enhanced features.

Figure 1: The OCTEON and OCTEON Plus Processor Overview



Note: OCTEON model-specific hardware components are marked with an asterisk (*).

3 Hardware-Acceleration Units

Hardware-acceleration units are divided into groups for this discussion:

- Packet-management accelerators
- Security accelerators
- Application accelerators
- Specialized accelerators

Packet-Management Accelerators:

- SSO Unit – Schedule/Synchronization and Order Unit: This unit manages packet scheduling and ordering.
- FPA Unit – Free Pool Allocator Unit: This unit manages pools of free buffers, including Packet Data buffers.
- PIP Unit – Packet Input Processor Unit: This unit works with IPD to manage packet input.
- IPD Unit – Input Packet Data Unit: This unit works with PIP to manage packet input.
- PKO Unit – Packet Output Unit: This unit manages packet output.

Security Accelerators:

- RNG Unit – Random Number Generator
- KEY* Unit: This unit provides and manages secure on-chip memory which can be used to store a hardware key, and can be reset using an external pin.
- Per-Core Security Coprocessor* (Security Engine): This unit is a special coprocessor used to accelerate security algorithms. There is one Security Coprocessor per core.

Application Accelerators:

- Pattern Matching and Regular Expression Engine*: This unit is used to perform string matching. The unit has different names on different OCTEON models, for example Deterministic Finite Automata (DFA). Users store rules in the attached pattern memory. In the OCTEON II and the NITROX® deep packet inspection (dpi) products, the pattern matching accelerators have been enhanced to include non-deterministic finite automata (NFA) functionality. Using NFA results in up to 4 times higher performance, support for very complex expressions, and significantly lower pattern memory requirements.
- ZIP Engine*: This unit is a compression/decompression engine which provides DEFLATE compression/decompression as defined in RFC 1951, ALDER32 checksum for ZLIB as defined in RFC 1950, CRC32 checksum for GZIP as defined in RFC 1952.
- RAID Engine*: This unit provides RAID/XOR Acceleration for RAID 5 and RAID 6.
- Per-Core CRC Engine: This unit is used to accelerate CRC generation.
- FPA Unit – Free Pool Allocator Unit: This unit manages pools of free buffers, including Packet Data buffers. It can be used as a general buffer manager, not only as a Packet Data buffer manager.

Specialized Accelerators:

- FAU – Fetch and Add Unit: This unit is used to add a number to a memory location, and can be used to manage counters.
- TIM – Timer Unit: This unit provides timers, which can be used for TCP timeouts as well as other more general purposes.

All of these hardware units offload work, freeing the cores to focus on essential packet processing.

Note: Hardware versus software CRC and hashing are discussed in more detail in the *OCTEON Application Performance Tuning Whitepaper*.

4 Packet-Management Accelerators

The packet-management accelerators automatically handle an enormous amount of packet processing, offloading the cores from many time-consuming responsibilities. These hardware units are responsible for packet receive, buffering, buffer management, flow classification, QoS (Quality of Service), and transmit processing. All of these functions are highly configurable (at per virtual port granularity), and can be customized using software to access the configuration registers.

Packet-management accelerators manage the following functions, without assistance from the cores.

- Manage list of free packet data and other buffers. (FPA)
- Packet receive and transmit through networking or PCI type interfaces
- Automatic packet data buffer allocation and freeing, and buffer management
- Layer-2 through layer-4 packet header parsing, exception checks and checksum calculation
- Up to 7-tuple flow classification with VLAN stacking support
- Congestion avoidance option, based on multi-priority RED algorithm
- Packet ordering and scheduling is automatically managed by hardware without requiring explicit software locking
- Traffic management with strict priority and/or weighted round-robin scheduling for packet transmit
- 8 levels of hardware-managed QoS for the input ports
- Up to 16 levels of hardware-managed QoS for each output port

4.1 Packet Flow, Summarized

The PIP and IPD units work together to receive a packet and perform early processing on it. Each packet is represented as "work" for the cores to do, and is represented by a Work Queue Entry (WQE) data structure.

The PIP and IPD are responsible for many layer-2 through layer-7 processing requirements such as exception checks and TCP/UDP checksum verification.

The PIP/IPD hardware units:

1. Verify the IPV4 checksum and payload checksums for TCP and UDP (if applicable).
2. Classify the flow. The QoS and group values are set.
3. Obtain packet data buffers and WQE buffers from the FPA.
4. DMA the packet data into memory using a dedicated bus.
5. Send the WQE to the correct QoS queue in the SSO. The WQE includes the first 96 bytes of the packet, so the core can begin to work on it immediately after receiving the WQE.

The PIP/IPD hardware also implements a per-QoS-queue random early discard (RED) algorithm, and a per-port threshold algorithm. These algorithms allow the PIP/IPD to discard input packets if necessary.

The SSO is responsible for scheduling the work to cores, and for maintaining the packet ingress order.

Cores may request work from the SSO either asynchronously (the core continues to do other work while the instruction completes), or synchronously (the core waits for the instruction to complete). Typically, cores minimize idle time by requesting the work before it is actually needed.

The PKO is responsible for packet transmission. When packet processing is complete, the core notifies the PKO that the packet is ready for transmission. The PKO manages transmission priority.

The PKO:

1. DMAs the packet data from memory into its internal memory over a dedicated bus.
2. Optionally computes TCP and UDP payload checksum and inserts it into the packet's header on egress.
3. When the packet is ready to transmit, the PKO sends the packet data from its internal memory to the output port over a dedicated bus.
4. Optionally frees the packet data buffer back to the FPA after the transmission is complete.

Details on the packet-management accelerators are provided in the *Packet Flow* chapter, including a discussion of how they work together to offload the cores.

4.2 The Scheduling/Synchronization and Order Unit (SSO)

The SSO unit enables scalable use of the multiple cores, maximizing parallel processing. It schedules work for the cores to do based on QoS priority, and work group. The cores are completely freed of this responsibility. In addition to scheduling, the SSO maintains packet order. The SSO also provides a locking mechanism to protect critical regions. The principal advantage of this locking mechanism over spin locks is that the cores continue to work while the SSO manages locking constraints. It would take considerable software coding and runtime cycles to implement the SSO's sophisticated scheduling mechanism in software.

The SSO's design supports the goal of flexible software architecture. The SSO's features work effectively with pipelining, modified pipelining, and run-to-completion software architectures.

Scheduling:

- When cores are ready for more work (packets to process), they request work from the SSO. The SSO is responsible for scheduling the highest priority work to each core.
- The scheduling algorithm is highly flexible and software tunable, allowing the user to customize scheduling to optimize application performance.

Synchronization:

- Packets can either be processed in parallel or one at a time. In typical packet processing, many operations may be done in parallel, increasing application throughput.
- When packet processing requires obtaining a shared resource (for example, a "critical region" in the code), locked access is needed. The SSO manages this access, offloading software. After the critical processing completes, the packet is processed in parallel again. These locks are referred to as *packet-linked locks,* and are discussed in detail in the *Packet Flow* chapter.
- The SSO manages the synchronization of the packets as they move through these processing phases.

Order:

- Whether the packet is processed in parallel or one at a time, the SSO maintains the packet's ingress order so that it may be transmitted in order.

4.3 Architectural Advantages of Work Groups

Each group can have from 0 to all of the cores. Cores can be in more than one group. The number of cores in each group can easily be changed by software, allowing the application to dynamically adapt to varying workload requirements. When a core requests more work to do, the SSO will only schedule work from an appropriate group to the core.

Work groups are used to balance the processing load among the different cores, providing architectural flexibility. Work groups can be used, for example, to assign a set of cores to run the data-plane application, and another set of cores to run the control-plane application. The work group value of a packet is set automatically on ingress, and can be changed by software. If the data-plane application needs to route a packet to the control-plane application, it simply changes the packet's work group value, and sends the work back to the SSO to be rescheduled.

The exact number of work groups depends on the OCTEON model. For example, in CN58XX, there are up to 16 groups, and there are 16 cores.

This feature provides the following benefits:

- Divide Data Plane and Control Plane Responsibilities: Groups can be used to divide cores into belonging to either the data plane or the control plane. The packet can be routed from the control plane to the data plane by changing the group value.
- Scaling: Applications can be easily modified to meet different performance targets by using OCTEON models with different numbers of cores. Cores are simply added to or removed from existing work groups.

- Reduce Latency: Tasks that are latency sensitive (must complete quickly) can be assigned to dedicated cores. Tasks which take longer to complete can be assigned to different cores, so they cannot cause head-of-line blocking.
- Partition Responsibilities: Users can dedicate certain cores for handling hardware interrupts, and avoid scheduling long latency tasks on these cores. As a result, interrupt response latency can be minimized.

5 Per-Core Security Coprocessors

Each core has a dedicated Security Coprocessor which can be used to accelerate security applications and hash generation. Once the core issues an instruction to the coprocessor, the core can continue to do other work while the coprocessor completes the instruction, or the core can wait for the coprocessor to complete the task.

Because each Security Coprocessor is dedicated to the core, there is no contention for this resource, increasing the determinism in performing security operations.

There are three primary advantages for integrating these Security Engines into each cnMIPS core:

- Current Industry Trends: Security processing technology is becoming an integral part of all networking applications and all kinds of networking devices. Every element in a network, end to end, will become security aware and capable. As a result, cnMIPS cores, with integrated security acceleration, offer the most flexible and scalable solution for implementing networking applications.
- Performance: The per-core Security Coprocessors provide optimal performance: there is no overhead or additional latency required to transfer data to and from the security acceleration hardware for processing. This is especially critical for smaller packets.
- Flexibility: The OCTEON security acceleration architecture can easily support novel modes for existing cryptographic algorithms, even modes that have not been defined today. This support can be added via a software upgrade without any hardware change. For example, suppose a new mode were invented for using the AES cryptographic algorithm, that used a different feedback algorithm, or that used a different operation to create the cyphertext. Such a mode could be easily supported with minor software changes.

6 On-Chip Interconnects

The on-chip interconnects join the different integrated units together. There are two key interconnects on the OCTEON and OCTEON Plus processors:

1. Coherent Memory Bus (CMB): The CMB connect all of the cnMIPS cores, the L2 cache controller, and the I/O Bridge. (Note: Although the CMB contains the word "bus", it is actually an interconnect which contains several buses, with sophisticated hardware arbitration.)
2. I/O Interconnect (IOI): The I/O Interconnect connects the remaining on-chip functional units: the I/O Bridge, hardware acceleration units, miscellaneous units such as the PCI DMA Engine, and the I/O interfaces.

The I/O Bridge is used to join the two interconnects.

The bulk of the data traveling on the interconnects is packet data, so the interconnects are optimized to handle packet data efficiently.

The interconnects are designed to:

- Maximize performance
- Support linear scaling when cores are added
- Deliver low-latency, high-determinism data transfers
- Minimize power consumption

Key features of the interconnect architecture include:

- The topology of the buses is carefully designed to align with packet processing flow, optimizing the movement of packet data among the various on-chip units.
- The split transaction and highly pipelined buses optimize bandwidth utilization, and avoid unnecessary over-provisioning.
- The data transfer latency is minimized and data transfer latency determinism is maximized as a result of focused connection of only the relevant data producers and consumers on each interconnect. (In some instances there are direct connections between the producer and consumer.)
- The bandwidth provisioning for each of the buses in the CMB or I/O interconnect is optimized based on the type of data transfers that the bus is responsible for in the packet processing flow.
- The interconnect architecture is scaled to best serve the number of cores on each OCTEON model.

The overall power consumption is minimized because:

- The optimized topology of the overall interconnect structure minimizes the paths and distance that data travels.
- The focused bandwidth provisioning avoids unnecessary over-provisioning of bandwidth.

See the *HRM* for details.

6.1 The Coherent Memory Bus Interconnect

The CMB interconnect is a collection of buses which connect all the cnMIPS cores, the L2 cache controller (L2C), and the I/O Bridge (IOB). (Note: The information in this section is for the OCTEON, OCTEON Plus, and some of the OCTEON II models. Some of the higher core count OCTEON II models will have a different interconnect architecture, which includes a cross-bar hyper-connect to scale up to 32 cores and higher.)

The CMB consists of several split-transaction, and highly pipelined buses. The data buses in the CMB are designed to meet the needs of the most demanding applications, and are over-provisioned for scalability to 16-cores and higher. The details of the CMB vary with the OCTEON model.

6.2 I/O Interconnect

The I/O interconnect connects the various on-chip functional units: the I/O Bridge, hardware acceleration units, miscellaneous units such as the PCI DMA Engine, and the I/O interfaces.

The I/O subsystem is connected together through a number of split-transaction and highly pipelined buses that run at core frequency. These buses together comprise a vast amount of bandwidth provisioned for internal data transfer. The exact buses and width vary depending on the OCTEON model.

The following figure shows an example of packet input and output using buses in the I/O interconnect. The following buses are used to transfer packet data:

Receive:

- IOBI – I/O Bus for Input: The IOBI carries the packet data from the input port to the IPD.
- IPDB – IPD Bus: Dedicated for DMA transfers from the IPD to L2 cache/DRAM.

Transmit:

- PKOB – PKO Bus: Dedicated for DMA transfers from L2 cache/DRAM to the PKO's internal memory.
- POB – Packet Output Bus: Dedicated for transfers from the PKO's internal memory to the output port(s).

Figure 2: Packet Data Movement Over I/O Buses

Packet Data Movement Over I/O Buses

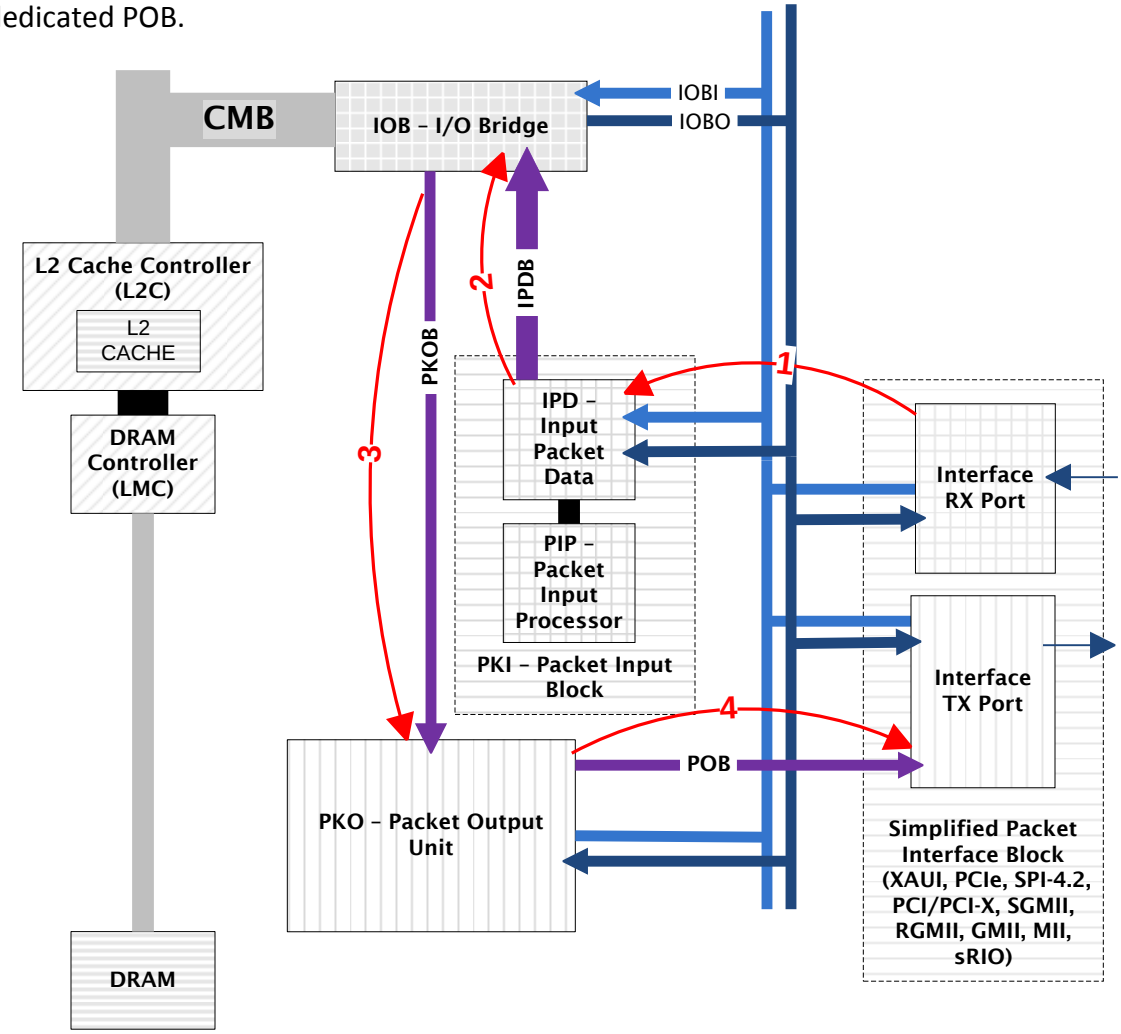
Most of the data traffic on the I/O interconnects is packet data. The I/O interconnect topology is carefully designed to optimize the flow of packet data:

Receive:

1. Receive packet data goes directly from the input interface to the IPD on the IOBI (the IPD is a second sink on the bus).
2. The IPD DMAs the packet data to L2 Cache/Memory via a dedicated IPDB.

Transmit:

3. The PKO DMAs the packet data from L2Cache/Memory to its internal memory on the dedicated PKOB.
4. The PKO sends the packet data from its internal memory to the output port on the dedicated POB.



7 Special Cavium Networks-Specific Instructions

Cavium Networks has added to the MIPS64 release 2 instruction set. The following list highlights some of the added instructions, but is not meant to be a complete list. For a complete list, see the *HRM*.

Note that the GNU C/C++ compilers supplied with the Cavium Networks SDK automatically generate most of these instructions. By re-compiling existing applications, users can automatically take advantage of these special instructions. Many instructions are also available through provided APIs.

Added instructions include:

- Bit field processing, extraction, and bit test branch instructions: MIPS64 release 2 provides a few bit-granularity instructions. Cavium Networks provides additional bit-granularity instructions. These instructions are used extensively in packet processing.
- Multiple instructions for controlling memory order: The MIPS64 ISA has a SYNC instruction for controlling memory order. Cavium Networks added several variations which provide finer memory order control for higher performance.
- Prefetch: The MIPS64 ISA provides a prefetch instruction which is used to load data to both L2 and L1 cache before it is actually needed. When the data is needed, it is already present, preventing the core from stalling while the data is fetched. Cavium Networks added prefetch instructions which allow the cores to:
  - prefetch data to L1 cache, bypassing the L2 cache (used to save space in the L2 cache, preventing eviction of cache blocks which are still needed)
  - prefetch to L2 cache without prefetching to L1 cache (used to prefetch data which will be needed by a different core than the requesting core)
- True unaligned loads and stores: The cnMIPS architecture processes unaligned data accesses without software intervention. In a traditional RISC processor core, unaligned data accesses result in an exception that the operating system needs to handle, significantly impacting system performance. The ability to process unaligned data also eases software reuse from legacy systems where unaligned data is allowed, saving the time and effort finding and fixing unaligned accesses.
- Atomic add to memory locations*: This instruction is used to atomically add a value to the value stored in a memory location. This instruction can be used to implement of large number of statistics counters in memory without explicit software locking. This instruction is not implemented in all OCTEON models.
- 32-bit, 64-bit population count: Population count instructions are used to determine number of bits that are set in a 32-bit or 64-bit piece of data. Population counts are useful in networking applications. For example, packet queues are typically managed as bit-masks with a set bit indicating that the corresponding queue is non-empty. Population count instructions can then determine the number of non-empty queues with a single instruction.

8 Cache Hierarchy

The OCTEON processor architecture offers the optimal caching policy for multi-core solutions, where efficient and low latency data sharing is critical to the overall performance.

The OCTEON processor cache hierarchy includes:

- Per-core L1 data cache
- Per-core L1 instruction cache
- Shared L2 cache (shared by all cnMIPS cores and the I/O sub-system)

The L1 data cache implements a hybrid write-through, write-back policy (using a write-buffer mechanism). The L2 cache implements a write-back policy.

The OCTEON processor architecture also offers many innovative cache-related features, including:

- Don't Write Back (DWB): The OCTEON processor provides the option not to write back selective data to L2/DRAM, to avoid unnecessary L2 data writes to memory. For example, the packet data in L2 cache can be discarded after the packet is transmitted; there is no need to store the data to L2/DRAM. In a conventional L2 cache design, all dirty data is written back to memory. This feature allows the user to tune the system to conserve memory bandwidth and power.
- L2 Cache Way Partitioning: The L2 cache ways can be partitioned among the cnMIPS cores and the I/O sub-system. This OCTEON feature enables intelligent management of the L2 cache to minimize cache pollution and the resulting loss of performance.
- Flexible Control of Data Movement: The OCTEON processor provides flexible control of moving data among the L2 cache, main memory, application acceleration engines, and I/O sub-system. For example, the OCTEON processor can be configured to automatically send:
  - the received packet header to the L2 cache
  - the packet data to main memory, bypassing the L2 cache
- Data Prefetch Instructions: The OCTEON processor provides multiple prefetch instructions to move data into L1 and/or L2 caches prior to the application needing the data. These instructions are used to avoid cache misses.

9 Summary

The OCTEON family of processors is highly optimized to achieve the highest performance per dollar and per Watt for a wide variety of networking, security, wireless, and storage applications. The OCTEON processor can be used for control-plane applications, data-plane applications, or a hybrid of both.

The OCTEON and OCTEON Plus processors have been designed into products from a significant majority of tier-1 OEM customers. They have delivered market-leading performance, along with low cost points and dramatically reduced power consumption.

Key features of the OCTEON family include:

- Custom-designed dual issue MIPS64 release 2 cores, with additional innovative Cavium Networks instructions added for improved performance.
- Linear scalability from 1 to 32 cores to provide the widest range of performance, power, and price options, while providing full software compatibility.
- Extensive integrated hardware acceleration units to dramatically improve overall application performance and automatically load-balance and synchronize processing.
- High-speed interconnects along with powerful DMA Engines, which are designed to optimize the flow of packet data through the OCTEON processor.
- Integrated I/O interfaces and memory controllers, which enable reduced bill of materials (BOM) cost and smaller form-factor designs.
- Simple software architecture based on standard C/C++ code, GNU toolchains, industry-standard operating systems, and optimized software stacks.

Software development complexity is minimized by hardware acceleration units, flexible software architecture, standard MIPS64, industry-standard toolchains and operating systems, packet-linked locks, and built-in scaling support.

OCTEON processors provide a proven, industry-leading solution for embedded developers looking to achieve fastest time-to-market with leading product performance and features.

This mini-version of the OCTEON Programmer's Guide does not include:

- Ch 2: Packet flow
- Ch 3: Software Overview
- Ch 4: Software Development Kit (SDK) Tutorial
- Ch 5: Software Debugging Tutorial
- Ch 6: OCTEON Application Performance Tuning White Paper
- Glossary

For a complete copy of the book please fill out the form at:
http://university.caviumnetworks.com/enroll.php

or contact us via email at:
university@caviumnetworks.com