Xelerated Sharpnose Network Processors

Xelerated NPU Software Overview

Product Description

Software versions:
Xelerated Platform Software 4.5
Xelerated Development Suite 6.7

For CUSTOMER Use Only

Doc. No. MV-S400400-00, Rev. A
July 9, 2015, Released
CONFIDENTIAL
Document Classification: Proprietary Information

Document Conventions

	**Note:** Provides related information or information of special importance.
	**Caution:** Indicates potential damage to hardware or software, or loss of data.
	**Warning:** Indicates a risk of personal injury.

Document Status

Doc Status: Released	Technical Publication: 1.1

For more information, visit our website at: http://www.marvell.com

Table of Contents

List of Tables

List of Figures

Preface: About This Document

Audience

This document is primarily intended for:

- Software engineers
- System architects
- Project managers
- Engineering managers
- CTO staff members

Prerequisites

It is assumed that the reader has a basic understanding of network processors for the development of networking equipment. It is furthermore assumed that the reader is familiar with common protocols like Ethernet, IP and MPLS.

Document Purpose

The document provides an overview of Marvell's Xelerated software for developing data plane software for the Xelerated HX4100, AX3100 and AX2100 families of network processors, collectively referred to as Sharpnose network processors (NPUs). After reading this document, the user will be able to assess the software architecture and qualities.

Document Scope

This document gives an introduction to:

- the design characteristics of the Marvell hardware (Network Processors) and software
- the software architecture and its relation to the hardware

The text has been written from the perspective of a developer as the reader and with the intention of explaining what he is able to do with the software and how.

Document Organization

The following is a summary and brief description of the major section of this guide:

- Chapter 1, *Introduction*, briefly explains the features and benefits of systems based on Xelerated Technology
- Chapter 2, *Software Architecture*, describes the software components and how they relate to one another and to the hardware.

Table 1: Acronyms Used in the Document

Acronym	Explanation
BOM	Bill of Materials
CESR	Carrier Ethernet Switch/Routers
CLI	Command Line Interface
CPU	Central Processing Unit
CTO	Chief Technical Officer
DRAM	Dynamic Random Access Memory
EAP	Engine Access Point
FPA	Forwarding Plane Application

Table 1: Acronyms Used in the Document

Acronym	Explanation
GMPLS	General Multiprotocol Label Switching
HAL	Hardware Adaptation Layer
IPRAN	IP Radio Access Network
MAC	Media Access Control address
NPU	Network Processors are referred to as NPUs, that is, Network Processor Units, in this document.
OAM	Operation, Administration and Management (as a combined network function)
OTN	Optical Transport Network
PHY	Physical Layer
PISC	Packet Instruction Set Computer
PTN	Packet Transport Network
RDK	Reference Design Kit
SerDes	Serializer/Deserializer
SMS	Shared Memory Switch
SNMP	Simple Network Management Protocol
SRAM	Static Random Access Memory
TCAM	Ternary Content Addressable Memory
XCM	Xelerated Control Message
XDS	Xelerated Development Suite
XEX	Xelerated Executable file *(pronounced "kecks")*
XPL	Xelerated Programming Language

1 Introduction

This chapter describes the benefits of using Marvell's Xelerated Network Processors for carrier and cloud infrastructure applications and explains their general software characteristics.

1.1 Benefits of the Xelerated Network Processors

The Xelerated technology is designed to carry out packet and flow processing (layer 2 to layer 5) in embedded high-speed carrier and cloud infrastructure systems. The Xelerated Network Processors (NPUs) suit both line-card and pizza-box systems.

The NPUs support a wide range of networking applications, protocols, and system designs. In addition, system vendors may develop the data plane for their specific requirements.

1.1.1 Programming Benefits

The Xelerated Sharpnose families of NPUs possess a dataflow architecture that enables extreme parallel processing of packets while addressing some of the programming challenges commonly associated with NPU designs. The architecture provides the following benefits:

- *Deterministic performance.* Resource contention is resolved at systemization and compile time, and gets verified at compile time. A compiled Forwarding Plane always performs according to its designed packet rate. The programmer does not need to consider resource contention at run-time. No further performance optimization is required.
- *Single-threaded programming model.* The developer writes packet service programs as single-threaded programs for a uni-processor, that is, the massive parallelism of the NPU is hidden from the developer.
- *Run-to-completion programming model.* The programs execute to completion inside the programmable homogeneous pipeline. In addition, programs can be extended in a flexible manner through looping.
- *Common architecture.* The Sharpnose families share instruction sets for the programmable pipeline, types of engines and tool chains. A design can therefore easily be ported between members of one family and between families.

1.1.2 Software Benefits

The Xelerated NPU software products support the design of embedded systems based on the Sharpnose families of NPUs. There are five general design attributes:

1. *High level of user-friendliness.* Integration and configuration software libraries provide powerful functions for boot, configuration, management and communication between the NPU and the host CPU. The developer can therefore operate on system object attributes without possessing (or needing) full knowledge of hardware implementations on hardware registers (see Section 2.1.7).
2. *Easy portability of Control Plane integration software.* All support and configuration functions are implemented in user space. This makes the software easy to port and integrate to the target system. Xelerated device driver is a Linux kernel module (see Section 2.1.5).
3. *Powerful programming language.* The Xelerated Programming Language (XPL) provides abstracted programming with C variables and table abstraction for all memory types (for example TCAM, SRAM, DRAM). This simplifies development and porting of Forwarding Plane software between target systems (see Section 2.2.2).

4. *Common tools.* The Xelerated Development Suite (see Section 2.3) provides feature-rich tools for editing, building, simulating and debugging the data plane software.
5. *Feature-rich reference data plane software.* Marvell's MetroCat data plane software is a fully-featured software set for the HX4100 NPU family. It provides state-of-the art features for carrier Ethernet, MPLS and IPv4/IPv6 services.

1.2 Target Applications for the Sharpnose Families of Network Processors

Sharpnose comprises the HX4100, AX3100 and AX2100 families. They have different performance and functionalities, but are programmed and configured with common tools and methods. The Sharpnose Network Processors (NPUs) are primarily intended for line cards and pizza boxes for:

- Mobile Backhaul Routers for IPRAN
- Packet Transport Network systems, PTN
- Packet-optical Transport, P-OTS
- Carrier Ethernet Switch-routers
- OTN and MSPP Platforms
- Evolved Packet Core Platforms
- Mobile Serving Gateways
- Cloud Computing Platforms
- SDN Systems
- OpenFlow Systems

1.3 What is new in Sharpnose?

The programming concepts have been maintained across several generations of Xelerated network processors. A number of hardware improvements are supported by the new software. The main additions to the Sharpnose families compared to previous generation Xelerated HX300 and AX300/AX200 families are:

- Fourth-generation modular PISC® programmable pipeline for advanced packet processing:
  - Match engines for large-scale IPv6 and IPv4 longest-prefix match and Ethernet exact-match tables
  - Dual ALUs, new instructions
  - Extensive internal TCAM and SRAM engines
  - Local EAP engines
  - Three simultaneous engine calls per EAP
  - Enhanced parallel interfaces to off-chip external DRAM for TM buffering and table extension
  - Pre- and post-formatters to support optional checksum calculation over entire IP packets, for example for UDP checksum.
- Traffic interfaces
  - 1G and 10G Ethernet MACs and PHYs
  - 40GE and 100GE MACs
  - Interlaken for connection to external fabric or framer
- Enhanced capacity of the integrated Traffic Manager with advanced hierarchical scheduling and deep-packet buffer in external DRAM
- Shared memory switch for flexible traffic configuration and intelligent oversubscription
- Multicast Copier

Please consult the HX4100, AX3100 and AX2100 product descriptions for more information about the functional specifications and use cases.

New software features for Sharpnose include:

- Xelerated Development Suite, XDS
  - IDE in Eclipse with improved graphical user interface
  - Editor enhancements to support Xelerated Programming Language
  - Enhanced Table System
  - Builder support for fourth-generation PISC programmable pipeline of Sharpnose modular pipeline additional ALU
  - Engine Resource Model supporting hard and soft rate programs
  - Simulator/Debugger extended to include debug information from the following subsystems: TI, SMS, MC, TM.
- Xelerated Platform Software, XPS
  - LPM manager for managing longest prefix match entries of embedded LPM
  - Configuration libraries to develop the Traffic Configuration through APIs
- Simplified information structure, enabling the user to declare target system data in one place.

Marvell also provides the Xelerated Build Environment, XBE, which is a Linux ISO image based on Ubuntu 12.04. It is available to customers that wish to evaluate or design data plane software for the Sharpnose NPUs. It includes all components used by Marvell to cross-compile the control and the forwarding planes for the Sharpnose NPUs.

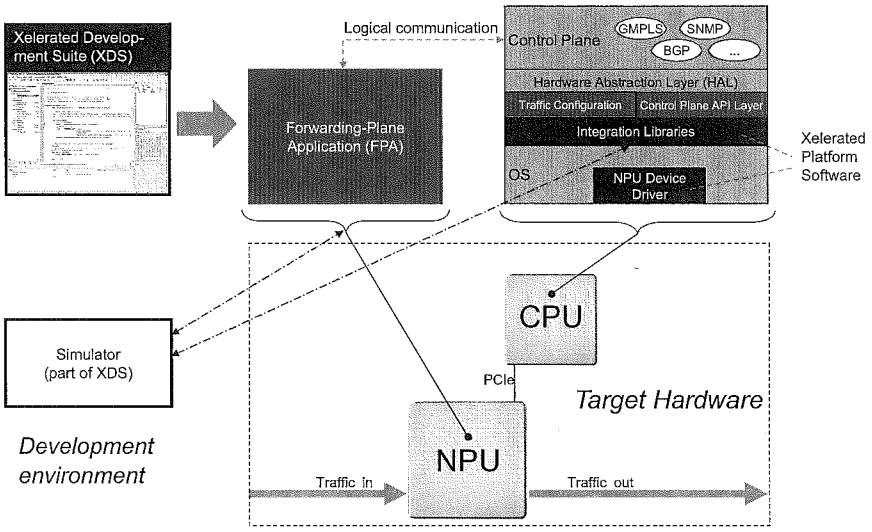
2 Software Architecture

This chapter provides an overview of the Xelerated software products, their purposes, and in what environments they function.

The software consists of three main parts (see Figure 1):

1. The *Xelerated Development Suite* (XDS): tools for development of NPU data plane software (see Section 2.3).
2. The *Xelerated Platform Software*: Libraries and driver providing APIs for boot, init, configration and management of the NPU (see Section 2.1.5).
3. The *Data Plane software*: software implementing for example Ethernet, MPLS and IP functionality on the NPU and associated Control Plane API layer (see Section 2.1.3). The data plane software consists of two main parts: the *Forwarding Plane Application* (FPA) executing on the NPU, and the *Traffic Configuration and Control Plane API layer* executing on the host CPU.

Figure 1: Schematic Representation of the Xelerated Software Architecture



Color Legend

- *Black fields*: Xelerated standard software products provided by Marvell.
- *Red fields*: Data plane software for the customer's target system. Data plane software is also available as reference software by Marvell.
- *Gray fields*: The customer's software components for the target system.
- *Black dotted arrows* indicate the parts of the software that will run equally well in the simulator (part of the Xelerated Development Suite, see Section 2.3.4) and on the target hardware.

2.1 Overview of the Control Plane

The purpose of the Control Plane is to provide features for configuration and control of the system's resources. It controls the Forwarding Plane Application (FPA) running on the NPU via updating table content. In addition, the Control Plane manages the physical resources of the board and the configuration of the NPU.

In most cases the software that integrates to the Control Plane in Xelerated NPU-based embedded networking systems has the structure outlined in this section.

2.1.1 Control Plane Applications

The customer's Control Plane Applications use routing and signaling protocols such as BGP, Spanning Tree and GMPLS to build routing and forwarding tables. They control the forwarding plane application by updating its tables, for example IP Routing Tables, Access Control Lists, and VLAN tables.

A network operator uses the Control Plane's management function to connect to its operational support system (OSS), and as a human-to-machine interface when configuring the system.

2.1.2 Hardware Abstraction Layer (HAL)

The Hardware Abstraction Layer (HAL) integrates the control plane functionality to the underlying data plane as defined by the Control Plane API. The customer is responsible for developing the HAL in his system, and it will be specific for the functionality of the target system.

2.1.3 Control Plane API Layer

The Control Plane API layer implements the interface through which the Control Plane Applications control the functionality of the FPA. The API layer controls tables that define the information model of the FPA, for example for classification, forwarding, and encapsulation.

The Control Plane API layer provides abstracted functions. One example is *add_ipv4_route* which sets a new route entry in the Forwarding Information Base (FIB). While the FIB may be implemented using several hardware tables in both internal or external memory, the HAL only needs to call the abstracted API.

2.1.4 Traffic Configuration

The purpose of the traffic configuration is to configure the traffic paths to the different subsystems of the chip, namely:

- Traffic Interfaces (TI)
- Shared Memory Switch (SMS)
- Multicast Copier (MC)
- Traffic Manager (TM)
- Programmable Pipeline (PP)

The traffic configuration is developed in C using the Configuration APIs of the Platform Software. The traffic configuration is called by the HAL after boot to set the NPU in a predefined state before user traffic is enabled.

2.1.5 Platform Software

The Xelerated Platform Software consists of Integration Libraries and the NPU Device Driver. The libraries contain C functions for boot, configuration and management of the NPU. The software also includes functions for reliable and efficient communication to the NPU. The software is provided in source code for integration to the customer's control plane application and operating system (OS) of the target hardware. The libraries are developed in Linux user space to be easily ported to any OS.

Figure 2: Platform Software Overview

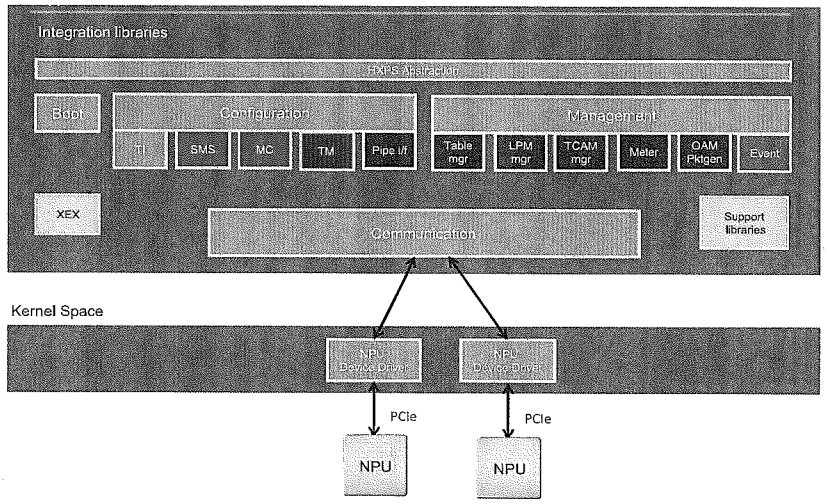


Figure 2 provides an overview of the main integration libraries available in the Xelerated platform software. They are divided into blocks to facilitate boot, configuration and management of the NPU:

- The *Boot block* includes the libraries for boot, *hxboot,* and initialization of the NPU.
- The *Configuration block* has a two layered structure of public APIs:
  - The top layer provides one library, *hxcfg*, which is an abstraction API for configuration of the connectivity, buffers and back pressure chain for the path between subsystems of the NPU. This library in turn, calls the bottom layer APIs
  - The bottom layer is divided into libraries, one for each configurable subsystem of the NPU: *hxti*, *hxsms*, *hxmc* (for multicast), hxtm (for the Traffic manager), and the *hxpp* (the Pipe Interface). These functions are generally called for configuration of the chip after boot, but may also be called at run time. Certain limitations apply for run-time changes.
- The *Management block* includes the following libraries:
  - *hxtbl* is a table manager which is used for accessing tables of the FPA from the Control Plane API layer.
  - *hxlpm is the* LPM manager is used for managing, FOR EXAMPLE, IPv4 and IPv6 tables using the Longest Prefix Match engine of the Sharpnose NPUs.
  - *hxtcam* is a TCAM manager for managing entries and priorities of the TCAM tables.
  - *hxmeter* is used for configuration of the meters of the internal SRAM engines.

- *hxpg* is a library used to configure the packet generator used e.g. for OAM services.
- *hxevnt* is an event library to configure mask settings and enable event management for the hardware exceptions. The user is recommended to subscribe to events using call back functions of the *hxps_services* API. This allows multiple users to subscribe to the same events.

All communication to the NPU for configuration, table updates and events is managed by the Communication block. Its main library is the *hxmsg,* which manages the protocol/messaging and communication aspects to and from the NPU, including DMA transfers.

The Xelerated Platform software also contains a library for parsing the Xelerated Executable file, *hxxex*, which includes the Forwarding Plane Application, which is built by the XDS builder. There are also a number of Support libraries, for example for endian conversion.

The included Debug CLI is a light-weight application for debugging purposes.

Note: Read more in the *Xelerated Platform Software User's Guide* for a complete introduction to the libraries. There is also Sample code available which describes how to use the APIs to different engines.

2.1.6 Operating System and NPU Device Driver

The Xelerated Platform Software is designed for maximum portability. The NPU device driver is the only part of the software that executes in kernel space and it controls the CPU-to-NPU communication. This is a simple character device driver implemented as a kernel object.

The NPU device driver provides messaging over PCIe and DMA handling. Multiple NPUs can be managed on a line card by the same CPU using multiple instances of the driver.

2.1.7 CPU to NPU Communication

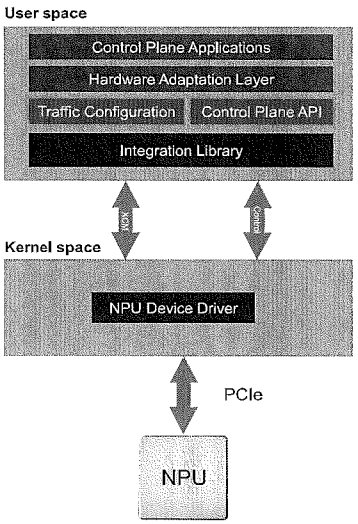
The Control Plane Applications have no direct access to the NPU or the FPA. The Control Plane Applications call the Control Plane API (through the HAL), which in turn calls the integration library to set and get information of the tables of the FPA.

All communication between the NPU and CPU is packet-based. There are two packet types:

1. *Xelerated Control Messages* (XCM), used to write and read to NPU registers.
2. *Control packets*: Slow-path traffic, for example Internet protocols re-directed to the control plane applications, and control traffic from the CPU to start management applications in the Programmable Pipeline.

Slow-path traffic may also be re-directed to CPU via standard traffic interfaces, for example XAUI. In those cases, the traffic is not managed by the Platform Software.

Figure 3: Message Exchange Between the CPU and the NPU Over PCIe



2.2 Overview of the Forwarding Plane

The programmable pipeline is structured into blocks of PISC (Packet Instruction Set Computer) cores and engine access points (EAP). PISCs executes PISC instructions. EAPs provide access to memories storing program data such as classification and forwarding tables.

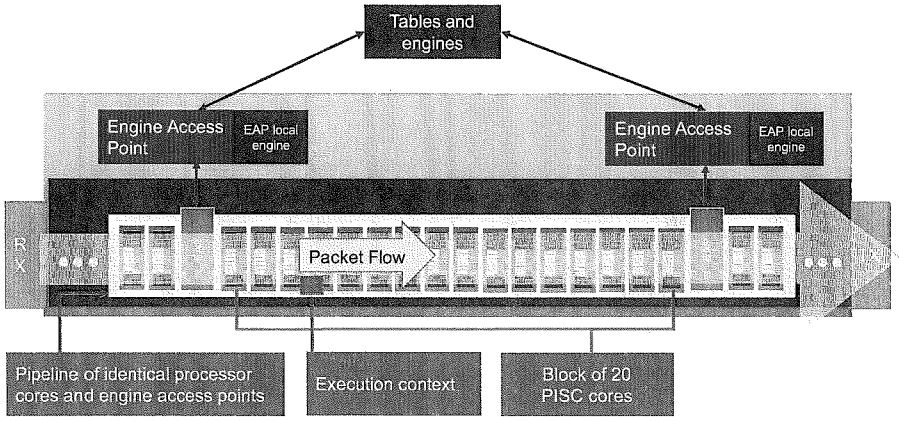
The Forwarding Plane Application is made up of one or several programs. Each program contains sequences of PISC instructions and table lookups

2.2.1 Dataflow Architecture and Programming Model

The FPA executes in the programmable pipeline, which has a unique dataflow architecture. It is designed to enable powerful packet processing at high rates, low power consumption, and with deterministic performance. The dataflow architecture (see Figure 4) includes the following components:

1. The Packet Instruction Set Computer (PISC) is a processor core specifically designed for packet processing. A pipeline can include several hundreds of PISCs.
2. The Engine Access Point (EAP) is a specialized I/O unit for classification tasks. EAPs unify access to tables stored in embedded or external memory (TCAM, SRAM, DRAM) and include resource engines for metering, counting, hashing, formatting, traffic management and table search.
3. Execution context is packet-specific data available to the programmer. It includes the first 256 bytes of the packet, general purpose registers, device registers and condition flags. An execution context is uniquely associated with each packet and accompanies it through the pipeline.

Figure 4: Dataflow Architecture



All tables and engines can be accessed from any EAP. Sharpnose NPUs also adds EAP local engines for access to small tables. All processor cores are identical, allowing for consistent programming and fully flexible use of resources for inspection, classification and modification of packets.

Thanks to the packet-specific execution content, which encapsulates the complete program state of each packet, the data flow architecture is presented to the programmer as a standard sequential uni-processor. The compiled code is loaded to the instruction memory in each PISC, mapping the first instruction to the first PISC, the second instruction to the second PISC, and so on.

The software for preparing classification, analyzing lookup results, and modifying packet headers is written as conventional sequential programs. C data types are used to simplify data structure management and be consistent with control plane programming models. They also provide a high level of data abstraction, thereby relieving the developer from the burden of keeping track of the memory allocation and the exact location of data.

2.2.2 Programming Language

The FPA is expressed in Xelerated Programming Language (XPL).

XPL files (.s, .xpl) include:

- Type and variable declarations in C
- Engine calls (lookups to tables, engine operations)
- PISC instructions (parsing packets, preparing table lookup, and modifying packets)

The language uses ANSI-C data types and variables, an arrangement that enables a high level of data abstraction and data-sharing with the control plane. Variables are auto-allocated.

Several programs execute in the programmable pipeline. For instance, a FPA may be divided into an ingress program, an egress program and an Ethernet learning program. What program to start is declared by using information about the origin of the packet and its type.

2.2.3 Tables

Tables are fundamental to the design of all forwarding plane applications. They are used for storing data for, for example, classification, ACL filtering, and packet forwarding. They are managed by the Control Plane through the Control Plane API layer.

Table data are declared as C data types that are shared between the forwarding and the control plane.

The Xelerated Builder (see Section 2.3.3) simplifies the use of tables by abstracting them from the physical memory. As a consequence the programmer will not need to determine in which physical memory location he should put a specific table. The Builder warns the programmer if data requests in engine calls are inconsistent with the declaration of the table.

Tables are defined in XCL by:

- memory type (SRAM, DRAM, TCAM)
- content (data type)
- table size

  Note: The *Xelerated Programming Reference Manual* contains a description of the programming model, the syntax, and the use of table abstraction.

2.2.4 Engine Microcode

Marvell provides microcode for the interfaces to the internal SRAM Engines, and external DRAM and Network Search Engines (TCAM). Its purpose is to hide the physical memory by providing abstracted operations such as read, write, count, and meter, independent of memory type (for example SRAM or DRAM). In addition, the microcode manages engine calls, which in turn relieves the programmer of the need to understand the data exchange and buffering aspects of memory I/O communication.

2.3 Xelerated Development Suite

The Xelerated Development Suite, XDS, is a tool chain used for developing and debugging Data Plane Software for the NPU. It consists of an Integrated Development Environment (IDE) with a graphical user interface, editor, builder, simulator and debugger.

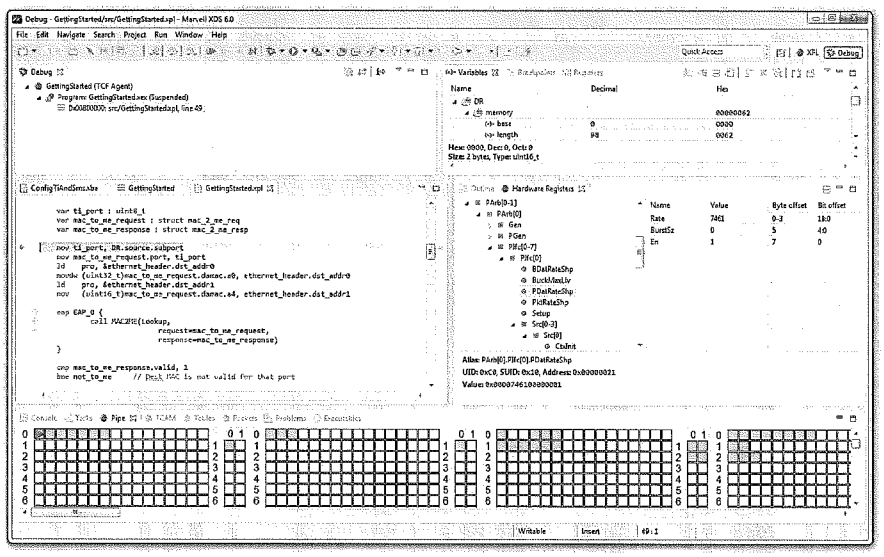
The developer uses the XDS for:

- Editing code
- Building suites
- Simulations and debugging
- Generating packets for tests

2.3.1 Integrated Development Environment (IDE)

The Integrated Development Environment (IDE) is based on Eclipse. It provides a graphical user interface to the tools, and a number of views to organize the development of data plane software. The tools can also be accessed through a CLI for scripting.

Figure 5: Eclipse-based IDE



2.3.2 Editor

The Editor automatically identifies correct XPL syntax to enable the programmer to complete the coding without typos and to ease reading of the source code.

2.3.3 Builder

The Builder is used to generate the Xelerated executable file (XEX, pronounced "kecks"). This file is an XML representation of the FPA, including configuration and instructions of the forwarding plane. The developer uses the Builder to compile and link the XPL source files into a XEX file. During boot the XEX file is parsed and loaded onto the NPU (see Figure 6).

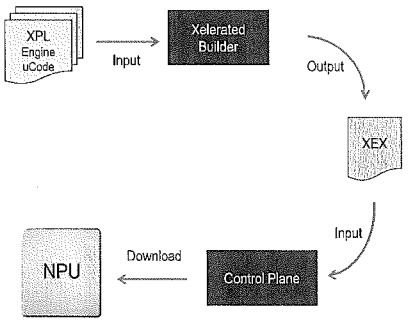
The builder is an integrated tool that consists of:

- A compiler for XPL with an integrated C preprocessor
- A linker

The developer may run the Builder either as a stand-alone program or via the IDE from the Build menu.

Note: Read more about the Builder in the *Xelerated Programmer's Reference Manual* .

Figure 6: Overview of the Build Process



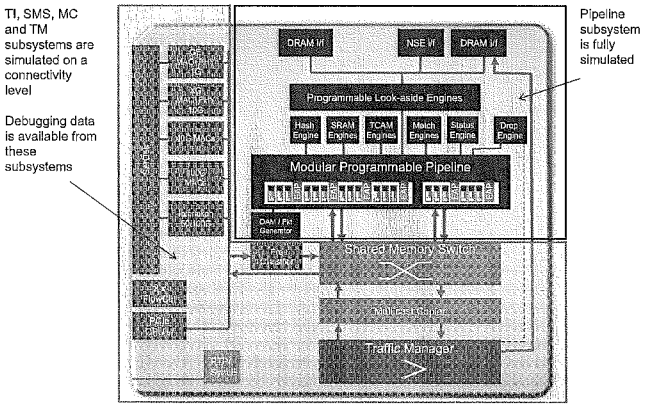
2.3.4 Simulator and Debugger

The simulator/debugger is a clock-cycle correct software model of the programmable pipeline subsystem of the NPU. It simulates the pipeline, including engines, and the connectivity to and from pipeline subsystems of packet paths across the NPU (see Figure 7). The developer uses it for functional simulations of the Forwarding Plane Application.

The XDS includes strong debugging capabilities. The user can single-step through the code and set execution break points. Such a point stops execution of a test packet at a given instruction and exposes the packet context data to the programmer.

The simulator/debugger may either operate in a stand-alone mode, or be connected to the target hardware, allowing the use of external test equipment, and reading out the table content, and packet header data and execution context from the target hardware while single-stepping through the source code.

Figure 7: Programmable Pipeline Subsystem Is Fully Simulated for Debugging



The debugger/simulator is used in the complete development phase to develop and debug the FPA and associated control plane API layer. It also enables the developer to start development prior to having target hardware available.

The developer runs the simulator on Windows 7/XP either as a stand-alone operating environment, without connection to the control plane software, or in a control plane co-simulation.

2.3.5 Packets for test

The XDS also includes support for defining and sending test packets to verify functional behavior of the FPA without dependency of external test equipment. A number of standard packet formats are provided as templates.

3 Information Structure

This chapter outlines the information structure for development of data plane software for the Sharpnose families of NPUs. It also provides an overview of the data exchange at compile, boot and run-time.

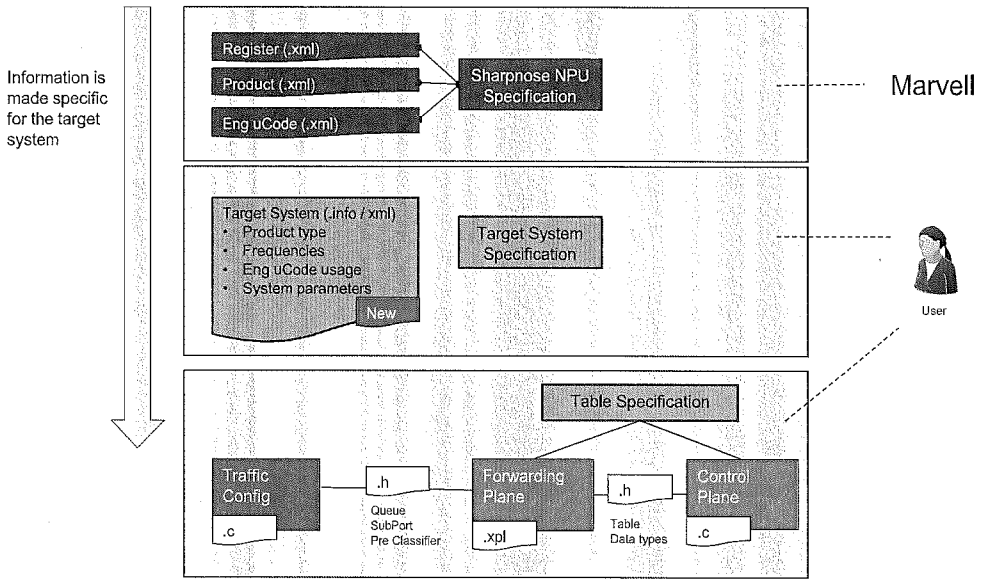
The information structure is simplified for the Sharpnose NPUs compared to previous generations and some new concepts have been introduced. In general, the user declares system data in one place only, and all functions that are dependent on this data will use it. That is, they all use one common source.

3.1 Information Structure View

Figure 8 provides an introduction to how the information is made system-specific in the development of the target system. The Sharpnose Product Specification is provided by Marvell. This data is included in the software for XDS, engine microcode and XPS platform.

The Sharpnose Product Specification includes information about the NPU registers (Register XML), the product definitions (Product XML), and Engine microcode (Eng uCode XML).

Figure 8: Information Structure



The user makes the product data system-specific by declaring information about the target system in the Target System XML file, *target.info*. This is referred to as Target System Specification.

The Traffic Configuration is programmed by the user to configure the infrastructure inside the NPU (TI, SMS, MC, and TM subsystems). The Traffic Configuration uses the information declared in the Target System Specification and calls functions of the libraries of the Configuration block of the XPS.

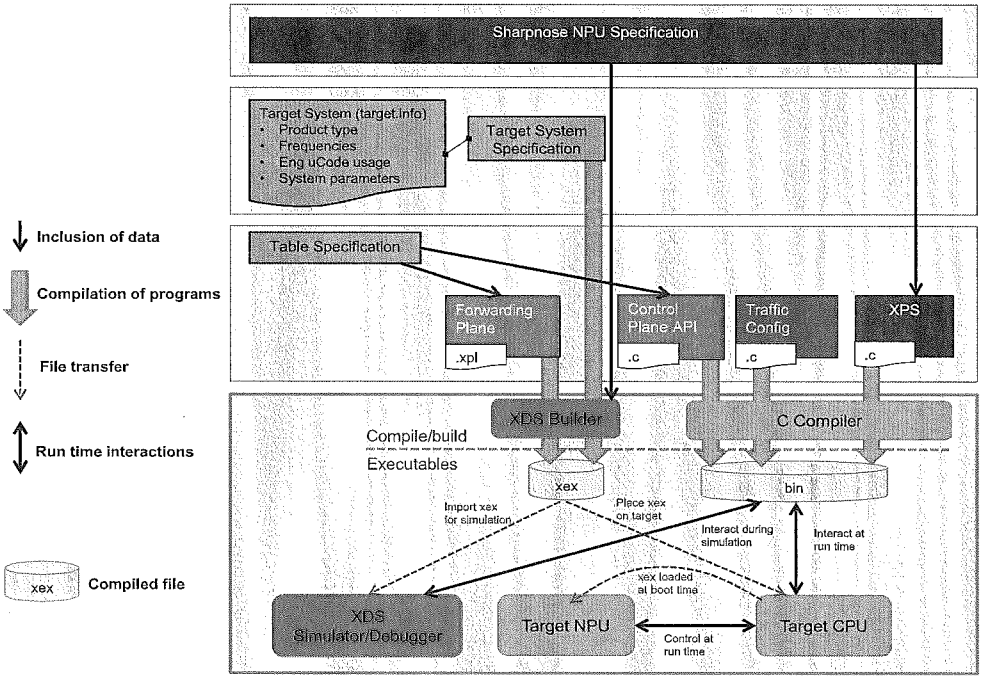
The user also declares the information model of the Forwarding Plane by defining the Table Specification. The tables are declared in one place, and the data structures are shared between the Forwarding and Control Planes.

3.2 Compile Time View

At compile time, the Target System Specification is compiled together with the PISC code for the Forwarding Plane to the Xelerated executable, XEX. This is used at boot process to initiate the NPU and load the FPA to the chip. It is also used by the XDS simulator/debugger.

The Control Plane API and the Traffic Configuration are cross-compiled together with the platform software to execute on the target CPU. This process is shown in Figure 9.

Figure 9: Data Exchange at Compile Time

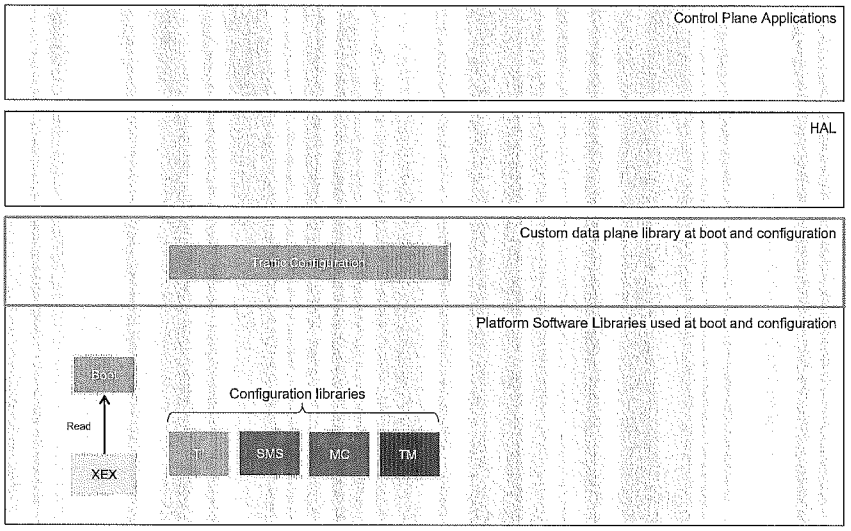


3.3 Boot and Configuration View

The data exchange for boot, initialization and configuration of the NPU is shown in Figure 10. The process starts at the left by calling the boot library for the NPU. The boot library reads the XEX file, which includes compiled information for the FPA. During the boot process, the memories are initiated, and training and BIST is executed for external interfaces.

The Traffic Configuration is then loaded onto the NPU before moving to the run-time state.

Figure 10: Data Exchange at Boot and Configuration

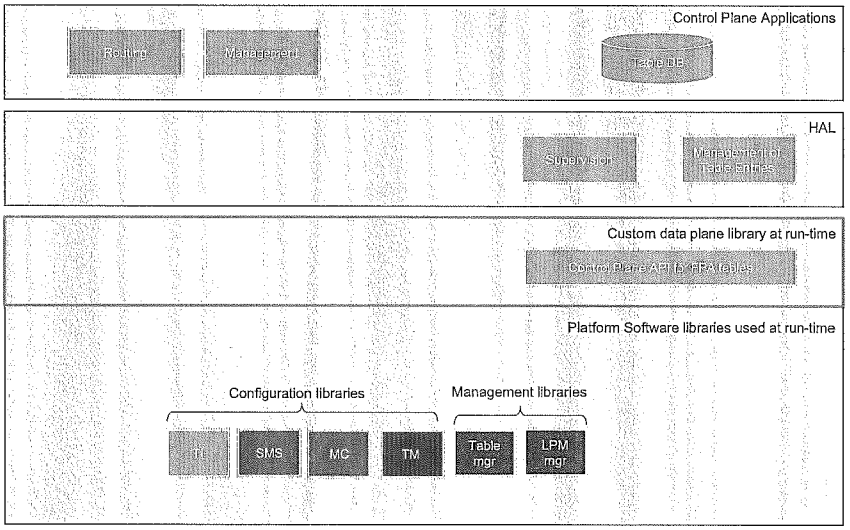


3.4 Run-time View

The information exchange at run-time is shown in Figure 11. The main exchange relates to table updates, which is managed through the Control Plane API for the FPA tables.

The Control Plane API towards all functionality of the NPU is generally stateless, except for data shadows kept for the TM scheduler, and the LPM manager.

Figure 11: Data Exchange at Run-time



A Revision History

Table 2: Revision History

Document Number	Internal Revision	Description	Date
MV-S400400-00 Rev-	1.0	Document created	July 2013
MV-S400400-00 Rev A	1.1	Updated	July 2015

Marvell Semiconductor, Inc.
5488 Marvell Lane
Santa Clara, CA 95054, USA

Tel: 1.408.222.2500
Fax: 1.408.988.8279

http://www.marvell.com

Marvell. Moving Forward Faster