PAVE Framework (PLD API for VxWorks Embedded Systems)

DS084 (v1.0) September 17, 2001

Product Specification

Features

- C++ API for configuring Xilinx FPGAs via SelectMAP™ or IEEE-1149.1 JTAG
- System Integration Framework (SIF):
  - Creates Wind River Systems Tornado® project and workspace files
  - Provides a structure to easily evolve new product applications from existing ones.
  - Makefiles create C++ source and header files from register maps
- PAVE defined payload upgrades FPGAs and system software
- PAVE provides baseline communication from host to target for lab development
- Supports Virtex™ Series and Spartan™-II devices

Introduction

The PAVE Framework is a development framework for creating and maintaining hardware field upgradable systems. It consists of an API for device configuration and a host based System Integration Framework (SIF) for use with Wind River's Tornado IDE. PAVE is used, like the FPGA device, as an element in Internet Reconfigurable Logic (IRL™) applications (Figure 1).

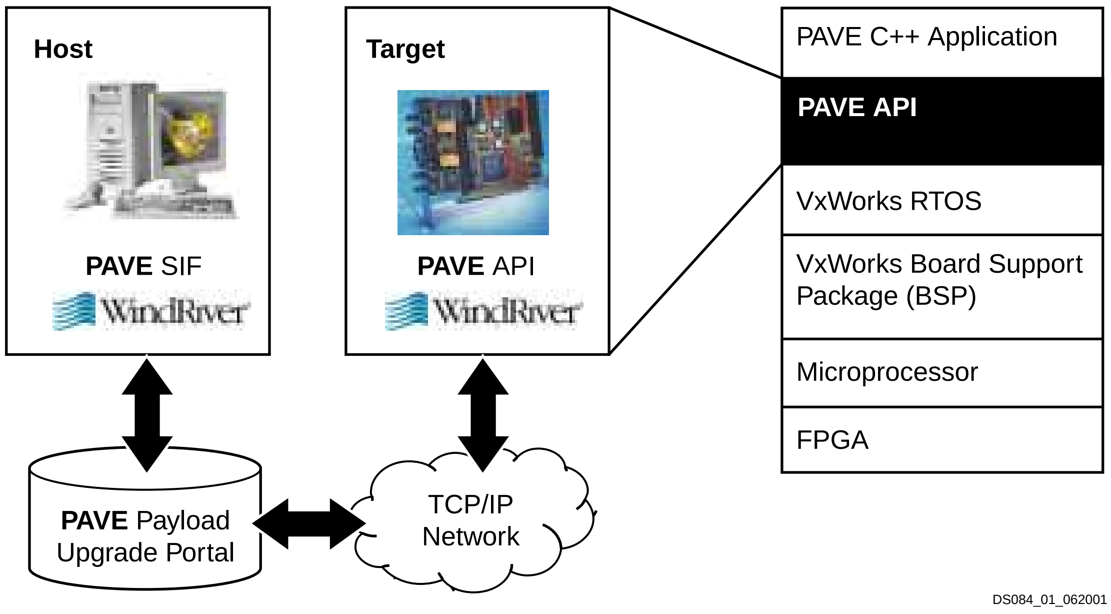

Figure 1: **Xilinx PAVE in IRL Enabled System**

Functional Description Overview

PAVE consists of two major components: the System Integration Framework and the configuration API. The SIF provides a powerful and easy to use framework for creating and managing upgradable applications. The API provides a C++ developer an off the shelf, standard tool for configuring Xilinx FPGAs. Used together, they provide the capability for quickly and efficiently upgrading deployed products over a network.

System Integration Framework (SIF) Description

Figure 2 shows the interaction of the PAVE SIF with other design tools as well as its input and output. The base level input into the SIF is a register map of the IRL enabled device(s) in the system. Once the register maps are created a reference to each map is used in the board level specification which describes the devices on a board. Reference to the board level specification is then used to create the system level specification. These simple text files are then used as input for the SIF generator which then outputs C++ header and source files that are used for the PAVE enabled application. The SIF directory structure reflects the view of an upgradable system into its Host and Target components. A directory for each target board in the system can be created using a PAVE SIF utility. The target board directory contains three subdirectories, host, server, and common that contain the code relevant to their respective functions.]

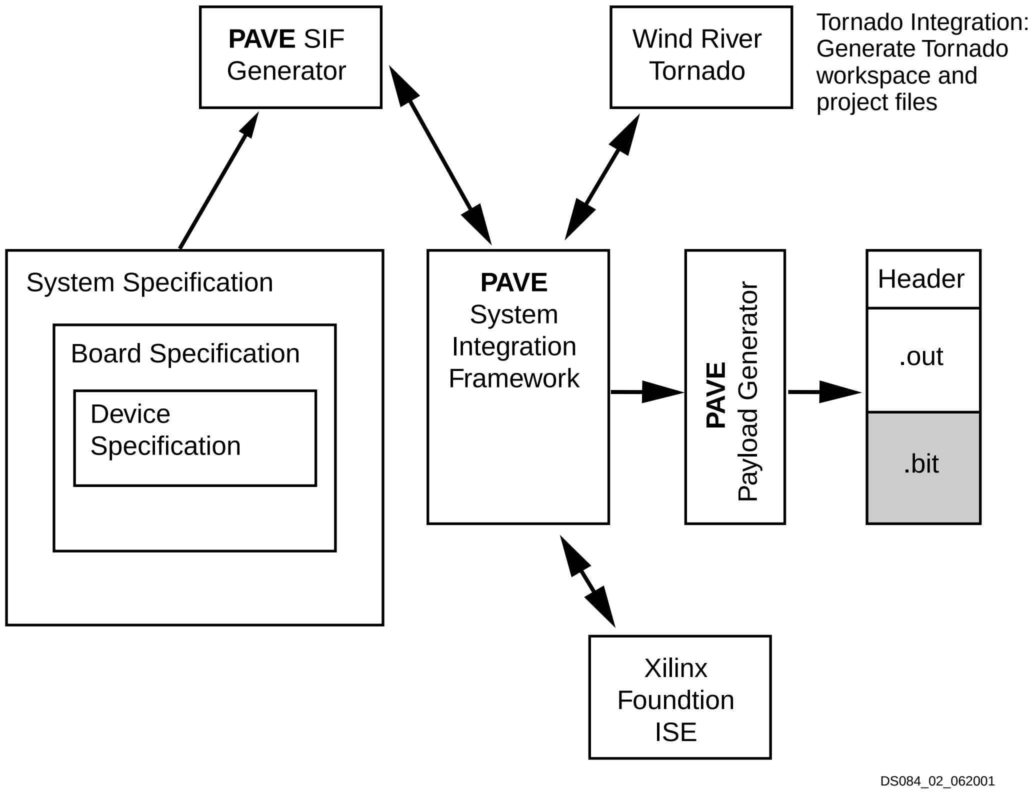

Figure 2: **System Integration Framework Flow Diagram**

PAVE API Description

The PAVE API is a software based methodology for configuring Xilinx FPGAs using SelectMAP or IEEE 1149 JTAG. It provides the facilities for a user written C++ application for writing to, reading from, and programming the FPGA. The API consist of several C++ classes that form an object model (Table 1). The hardware requirements for its use are a 32-bit Configuration Register (Figure 3 and Table 2) and a VxWorks compatible Board Support Package (BSP). The PAVE API abstracts the underlying VxWorks hardware platform through the use of a BSP and the PAVE object model. This allows the developer to use the same C++ code for PAVE across many different hardware platforms.

The PAVE API uses its methods to construct a ***classSignalBuffer*** object which is then sequenced to the hardware via the methods of the ***classRegister*** and ***classPlatform*** objects (Figure 4).

Table 1: **PAVE API Classes** (see the PAVE User Guide for complete listing)

API Class Object	Description
classRegister	Provides functionality to read from and write to the control registers of the upgradable device. This class abstracts the details of the masking and shifting operations that normally accompany read, write, and modify operations on a register.
classStateMachine	Abstracts the functionality of a finite state machine. This object is contained in the classJTAGComponent object and is used to setup the TAP controller state machine for JTAG configuration mode. The classStateMachine object is very generic and can also be used to facilitate the implementation of very complex real-time state machines in embedded applications.
classSignalBuffer	Encapsulates the functionality required to setup and sequence a set of control signals to hardware registers via software control. This class also contains a number of utility methods that allow the developer to generate arbitrary test vectors that can be used in device test benches.
classPlatform	Facilitates porting efforts by forming a thin abstraction layer between the platform specific code and the application code.

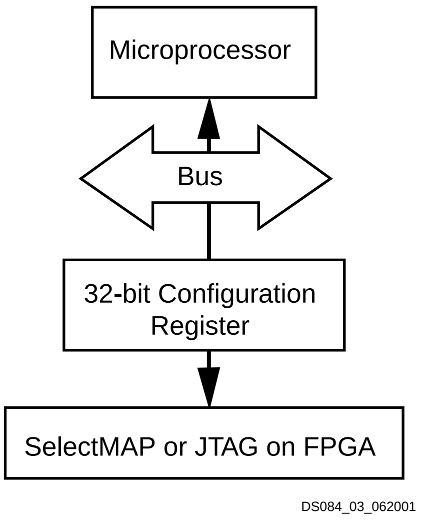

Figure 3: **PAVE API Configuration Register Requirement** (refer to Table 2)

Table 2: **32-Bit Configuration Register**

Name	Start Bit	N bits
JTAG_BUFF_OE	0	1
JTAG_TCK	1	1
JTAG_TMS	2	1
JTAG_TDI	3	1
JTAG_TDO	4	1
SMAP_BUFF_OE	5	1
SMAP_CCLK	6	1
SMAP_RW	7	1
SMAP_CS	8	1
SMAP_PROG	9	1
SMAP_INIT	10	1
SMAP_DONE	11	1
SMAP_BUSY	12	1
SMAP_D[7:0]*	13	8
MODE_M[2:0]*	21	3
MODE_HSWAP_EN	24	1
Reserved	25	7

* Start bit is 0

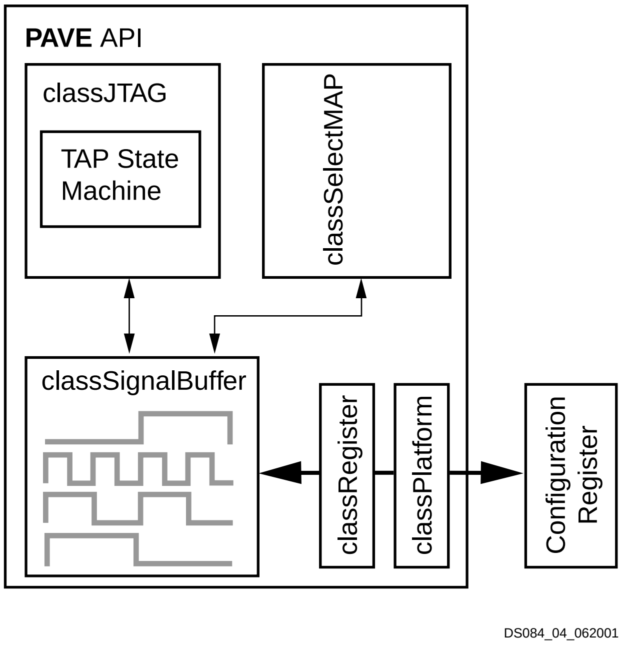

Figure 4: **PAVE API Block Diagram**

Host System Requirements

- Tornado II
- Windows NT/Win 2000
- 64 MB RAM
- 300 MB disk space
- Pentium II class or better
- Network interface card
- TCP/IP installed
- Netscape or IE 4.0 or better

Target System Requirements

- VxWorks 5.4 or higher

PAVE Ordering Information

The PAVE Framework software and IRL Design guidelines can be download by following the links at this location:

http://www.xilinx.com/irl

Documentation

- PAVE User's Guide - A detailed manual with information on
  - System Integration Framework
  - API
  - JTAG and SelectMAP Configuration
  - Using PAVE with the Durango and ADMXRC boards

Related Information

- Xilinx Application Note **XAPP412: "Architecting Systems for Upgradability with Internet Reconfigurable Logic (IRL)"**
- Wind River Systems VxWorks Programmers Guide
- Wind River Systems Tornado Users Guide
- Alpha Data Ltd. **ADM-XRC**
- Xilinx Foundation Series 3.1i Quick Start Guide
- Xilinx Application Note **XAPP058: "Xilinx In-System Programming Using an Embedded Microcontroller"**
- Xilinx Application Note **XAPP138: "Virtex Configuration and Readback"**
- Xilinx Application Note **XAPP139: "Configuration and Readback of Virtex FPGAs Using (JTAG) Boundary-Scan"**
- IEEE P1149.1/D2000.5 Draft Standard Test Access Port and Boundary Scan Architecture

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
09/17/01	1.0	Initial Xilinx release.