Celoxica

Platform Developer's Kit

Introduction to PDK

Authors: RG

Document number: 1

Customer Support at http://www.celoxica.com/support/

Celoxica in Europe	Celoxica in Japan	Celoxica in the Americas
T: +44 (0) 1235 863 656	T: +81 (0) 45 331 0218	T: +1 800 570 7004
E: sales.emea@celoxica.com	E: sales.japan@celoxica.com	E: sales.america@celoxica.com

Contents

Conventions

The following conventions are used in this document.

Warning Message. These messages warn you that actions may damage your hardware.

Handy Note. These messages draw your attention to crucial pieces of information.

Hexadecimal numbers will appear throughout this document. The convention used is that of prefixing the number with '0x' in common with standard C syntax.

Sections of code or commands that you must type are given in typewriter font like this:

```
void main();
```

Information about a type of object you must specify is given in italics like this:

```
copy SourceFileName DestinationFileName
```

Optional elements are enclosed in square brackets like this:

```
struct [type_Name]
```

Curly brackets around an element show that it is optional but it may be repeated any number of times.

```
string ::= "{character}"
```

Assumptions & Omissions

This manual assumes that you:

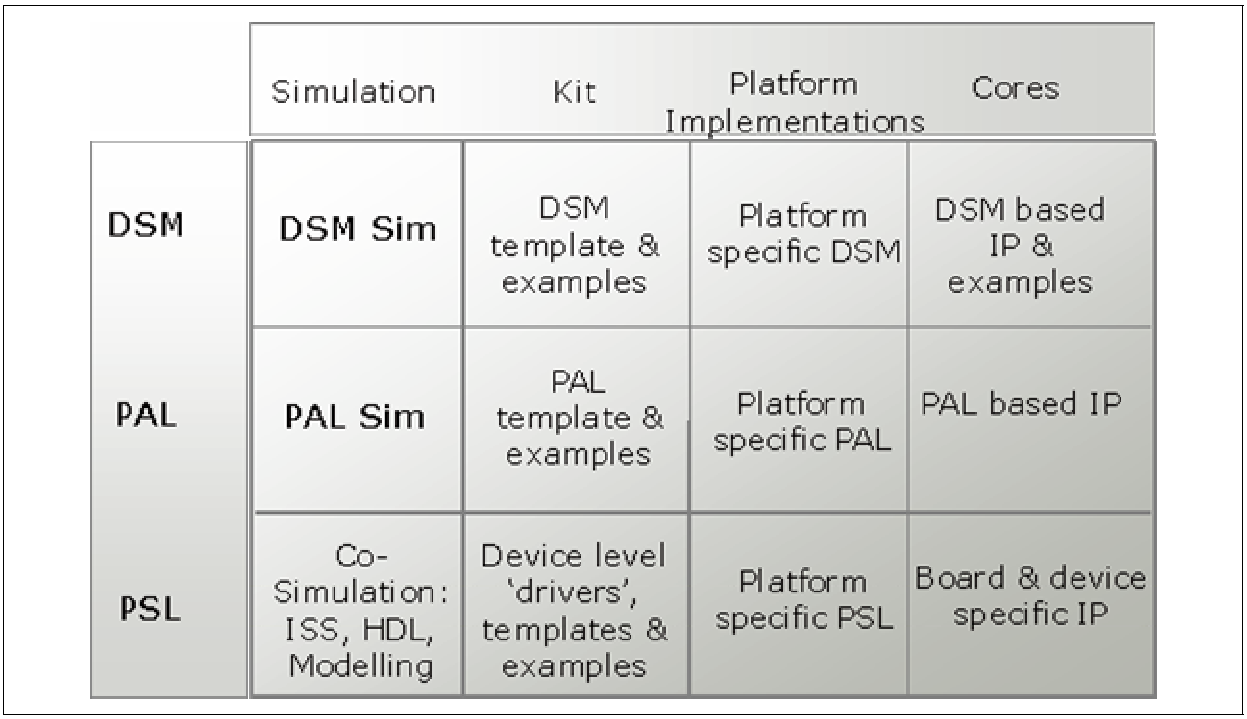
- have used Handel-C or have the Handel-C Language Reference Manual
- are familiar with common programming terms (e.g. functions)
- are familiar with your operating system (Linux or MS Windows)

This manual does not include:

- instruction in VHDL or Verilog
- instruction in the use of place and route tools
- tutorial example programs. These are provided in the Handel-C User Manual

1 Introduction to PDK

Celoxica's DK design suite and Handel-C language allow applications to be developed on, or migrated to, FPGA/PLDs more quickly and easily than other approaches. However, engineers developing such solutions are looking for further support – an expectation set by the development support and hardware independence provided by modern operating system environments offered on microprocessors. Introducing this support and hardware independence into an FPGA/PLD environment requires the availability of a support library providing a layered and consistent approach to hardware interfacing, simulation, development methodologies, standard libraries and added-value functionality.



OVERVIEW OF PDK

Celoxica's PDK (Platform Developer's Kit) is an integrated library offering three layers of functionality, each composed of 4 inter-connected areas of support. An overview of PDK is shown above.

The three layers of functionality provided by PDK are PSL (Platform Support Library), PAL (Platform Abstraction Layer) and DSM (Data Stream Manager). These support different aspects of application development but are mutually integrated and together support the development of either stand-alone or integrated Handel-C applications.

- DSM provides integration between processors/DSPs and FPGAs/PLDs, either on an integrated platform device or on discrete Processor and FPGA/PLD platforms.
- PAL provides a consistent API for portable board-level Handel-C implementations.

- PSL provides board, hardware or development tool specific support for DK and Handel-C.

Each of these three areas of functionality provides:

- Simulation – hardware independent simulation of DSM and PAL APIs and co-simulation with external tools, simulators and design modelling environments
- Kit – key components and/or templates to allow development of new platform specific implementations
- Platform – Implementations of DSM, PAL and PSL components to suit a specific platform or environment
- Cores – Implementations of added-value functionality, demonstrations or examples.

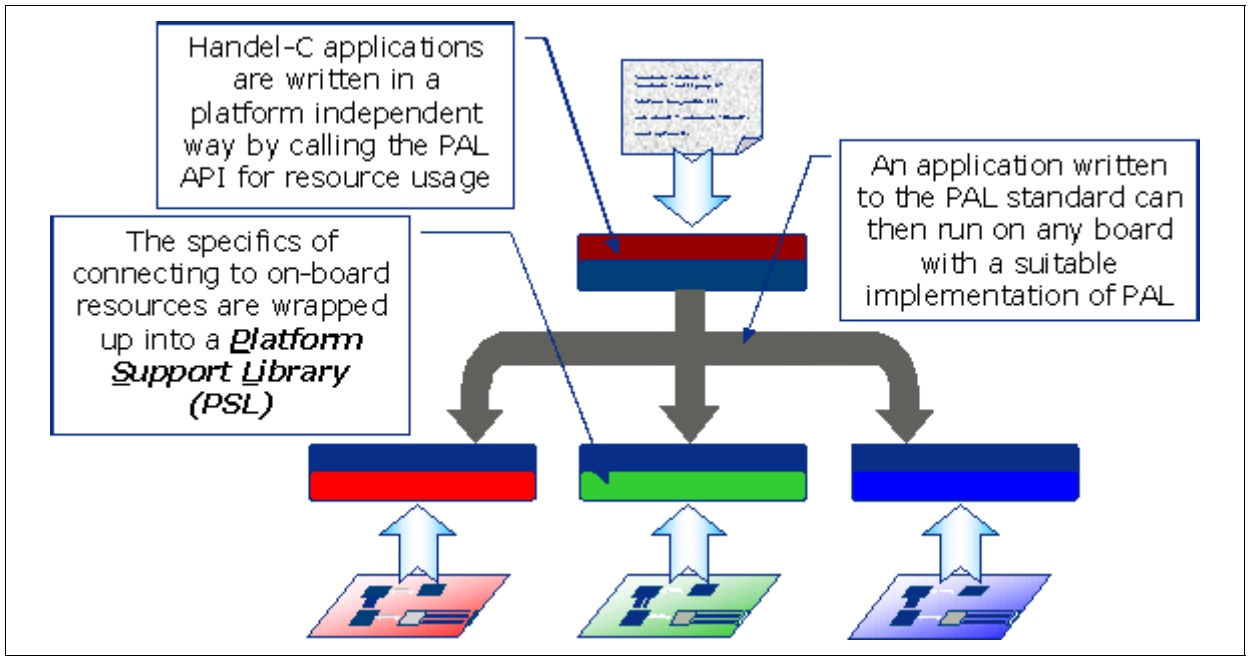
Benefits from using the PDK include:

- More rapid development by reducing or removing the need for the developer to be concerned with hardware detail
- Easier migration of designs between different platforms
- Reduced cost of maintenance and support
- Increased focus on application added value
- Improved opportunity for design space exploration before final implementation decisions are made

2 Platform Abstraction Layer

Celoxica's Platform Abstraction Layer (PAL) provides hardware independence through the use of a consistent API which a Handel-C program uses to access hardware resources.

PAL implementations are written once for an embedded system's hardware architecture. Applications using the PAL API to access platform resources are then portable between systems utilizing the PAL concept. This approach can completely eliminate porting work. Development engineers can rapidly port designs from their prototype platform to their final system, or perform design reuse from one generation of a product to the next. So, at the same time as accelerating the development cycle for products utilizing FPGA/PLDs, the PAL approach allows the more effective use of design resources.

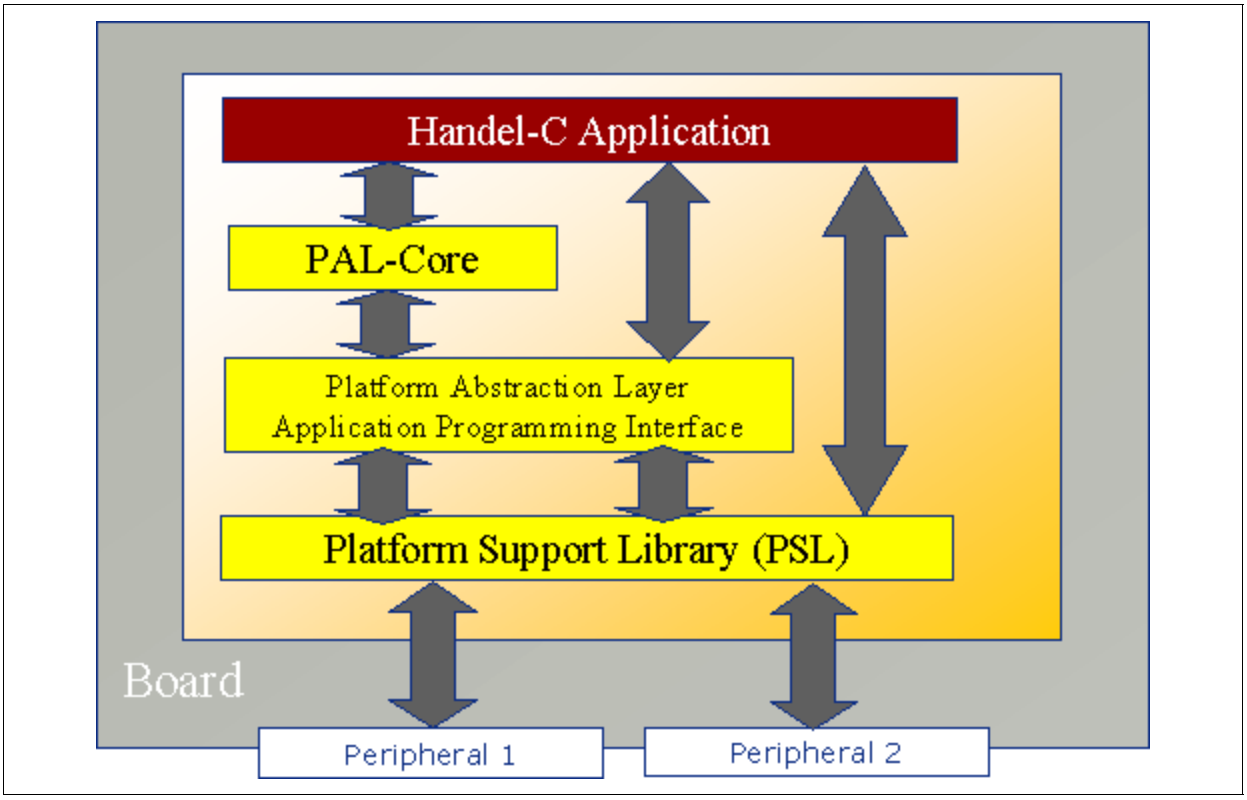


PAL OVERVIEW

PAL shields development engineers from low-level hardware interfaces in order to ease the integration of FPGA/PLDs with physical resources. This is done by developing a library of low-level interfaces to specific platform resources, such as I/O or memory. This library, called the Platform Support Library (PSL), is then accessed from the Handel-C applications on the FPGA/PLD using a simple and consistent Application Programming Interface - the PAL API.

Without the level of abstraction and support provided by Celoxica's Platform Abstraction Layer (PAL), a Handel-C application has to include not only the application functionality, but also the code necessary to interface to all of the required external devices. This can require significant additional effort and makes porting applications from one device environment to another more complex than necessary.

By providing a layer of abstraction that insulates the Handel-C developer from the low-level hardware interfaces used between the FPGA/PLD and external logic, connectors or systems, Celoxica's Platform Abstraction Layer (PAL) facilitates the development of Handel-C applications and the migration of software from microprocessor implementations to FPGAs or PLDs. (see figure above).



PAL ARCHITECTURE

PAL is composed of four main elements:

- PAL API implementations for specific environments
- PAL Kit – the elements required to implement PAL for a specific environment
- PAL Cores – added value functionality layered on the PAL API
- PAL Sim – a simulation environment for PAL compliant applications

PAL provides an API (Applications Programming Interface) specifying generic function or macro calls that provide the Handel-C application with access to and communication with the hardware environment. Applications using the PAL API are portable across any platform with a suitable PAL implementation available.

2.1 PAL API

The PAL API consists of calls appropriate for different classes of device. Classes are defined for:

- Memory
- Data ports (including RS232, PS2 and Parallel)
- LEDs and Seven Segment displays
- Video capture
- Video output
- Block storage
- Switches and Buttons
- Audio input/output
- Ethernet
- Audio
- Touchscreen

For applications to make use of the PAL API, a PAL layer must be implemented for each hardware platform using an appropriate Platform Support Library (PSL). Celoxica provides PAL API implementations for the Celoxica RC10, RC100, RC200, RC203, RC250, RC300, RC1000 and RC2000 platforms, the Altera Nios and Nios II Development board and the Memec V2Pro board.

2.2 Platform Support Libraries (PSL)

Introduction to PDK. Next: PAL Cores

For each supported hardware environment - effectively a board design, such as Celoxica's RC300, a set of drivers customized to that design called a Platform Support Library (PSL) is required. The PSL is analogous to the Board Support Package provided by embedded operating system vendors to target specific processor boards. Once a PSL has been implemented, a Platform Abstraction Layer can be developed to provide access to the PSL using the PAL standard API calls.

Celoxica provides PSL implementations for the boards and features listed in the table below.

In addition there is PSL support for the following processor core interfaces :

- MicroBlaze processor : OPB bus
- Xilinx VII Pro : OPB bus and PLB bus
- Altera NIOS II : Avalon bus through SoPC builder

PSL Device I/O	RC10	RC100	RC200 / RC203	RC250	RC300	RC2000	RC2000Pro	Altera NIOS development board	Altera NIOS II Development kit
Synchronous Static RAM		Y	Y	Y	Y	Y			Y
Asynchronous Static RAM								Y	
LED output	Y	Y	Y	Y	Y				Y
Seven Segment Display	Y	Y	Y	Y				Y	Y
Parallel Port		Y	Y						
Character Display					Y				
RS232	Y		Y	Y	Y				Y
Ethernet			Y	Y	Y				Y
PS/2 Port	Y	Y	Y	Y	Y				
Flash memory	Y	Y	SM	SD	SM			Y	
Composite Video In		Y	Y	Y	Y				
S-Video In		Y	Y	Y	Y				
S-Video out			Y						
VGA Output	Y	Y	Y	Y	Y			Y (1)	
DVI Input				Y	Y				

DVI Output			Y	Y				
Audio in		Y	Y	Y				
Audio out	Y	Y	Y	Y				
Control and Status Port								
Switches and Buttons	Y(2)	Y	Y	Y			Y	Y
BlueTooth module		Y						
Piezo buzzer	Y							
ADCs	Y							
Servos	Y							
Expansion header	Y							
USB Data	Y							
CMOS camera	Y							
TFT Screen		Y	Y	Y				
Touch Screen		Y	Y	Y				
Local BUS - PCI interface (Host to card transfers)					Y	Y		

Note1 - Via mezzanine card

Note 2 - Joystick

2.3 PAL Cores

PAL also provides a higher layer of functionality, known as PAL Cores. PAL Cores are device controllers which are layered on top of PAL and which do not need to be re-written for different boards. Celoxica currently provides PAL Cores implementing a mouse driver, a keyboard driver, a frame buffer and PAL Console – a means to display text and other information from a Handel-C application.

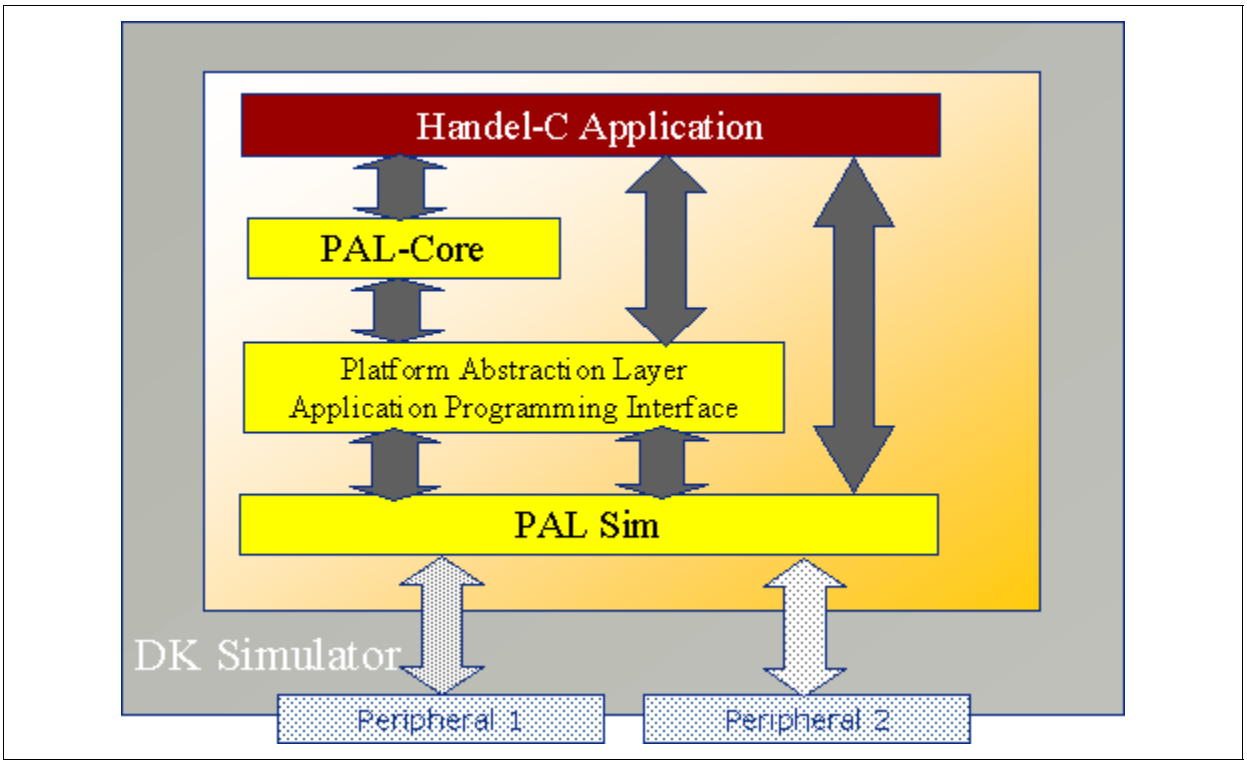
2.4 PAL Kit

PAL Kit provides information and support for the development of PAL implementations for specific boards or devices. PAL Kit includes:

- Templates
- Documentation
- Tutorials
- Examples and demonstrations

2.5 PAL Sim

PAL Sim provides a windows-based simulated platform utilizing the PAL API. This allows development of PAL compliant applications to take place before hardware or hardware support libraries have been developed or without hardware at all where the application is designed to be hardware independent.



PAL SIM

PAL Sim supports the following interface classes:

- Memory (including load from- and save to- disk)
- Data ports (to and from a disk file, or mouse/keyboard)
- LEDs and Seven Segment displays
- Video capture (from file)
- Video output
- Block storage
- Switches and Buttons
- Audio input/output (using PC audio hardware)
- Ethernet (disk file or live network)

3 Data Stream Manager

FPGA/PLDs are increasingly being used to provide co-processing support to microprocessors with the functions in the FPGA/PLD often being derived from pre-existing software or software prototypes using the Handel-C language to migrate these into hardware. With SoPC (System on a Programmable Chip) products now being introduced, the requirement for this type of solution is increasing rapidly, as is the requirement to design solutions using a development platform and then migrate the implementation to a different production environment.

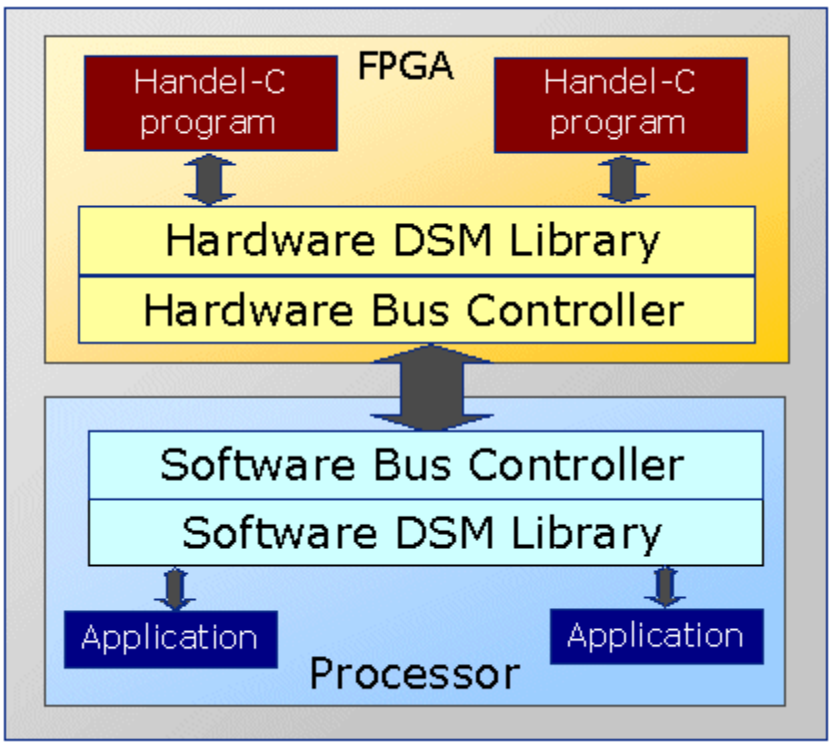
While Celoxica's PAL (Platform Abstraction Layer) simplifies the implementation of solutions requiring access to the physical resources of the embedded solution, co-processing solutions will require integration between those parts of the application running on the microprocessor and the accelerating functions provided on the FPGA/PLD. Developing integration on a case-by-case basis will be unwieldy and complex and cause short and long term migration and maintenance problems.

With the goal of reducing development time and increasing focus on added value, Celoxica has developed DSM (Data Stream Manager) to simplify and standardize platform integration across FPGA/PLD, processor and operating system platforms

3.1 DSM overview

DSM provides the key integration components and methods needed to simplify the implementation of system designs split across processors and FPGA/PLDs, whether these are discrete components or integrated platforms. The integration is achieved by providing a consistent data transfer API (Application Programming Interface) for both the software in the microprocessor and for the Handel-C program in the FPGA/PLD. DSM also provides mechanisms that standardize the integration of processor, FPGA/PLD and the data transport bus between them. The API and the underlying transport mechanism insulate the developer from the hardware detail of the physical implementation and the operating system specifics in order to facilitate migration and maintenance of the ultimate solution.

The figure below shows a system overview using the DSM model. Applications in a microprocessor or DSP communicate co-processor function requests to the Software DSM Library. These requests are routed by DSM through the Software Bus Controller and across the physical interconnect to the FPGA/PLD.



DSM ARCHITECTURE

In the FPGA/PLD, the Hardware Bus Controller delivers the data to the User's Handel-C program via the Hardware DSM Library. Returned data is passed back by the same route to the originating application. The same mechanism may also be used for inter-FPGA/PLD data transfer or function requests or for data transfers originating in the User's Handel-C program.

DSM uses a layered architecture and a consistent API in order to ensure portability of applications and implementations across various hardware and operating system platforms. The benefits offered by this portability include a reduction of development effort and time together with improved re-usability of applications and functions. The architecture also facilitates the development of new DSM implementations for new processor, operating system and FPGA/PLD environments.

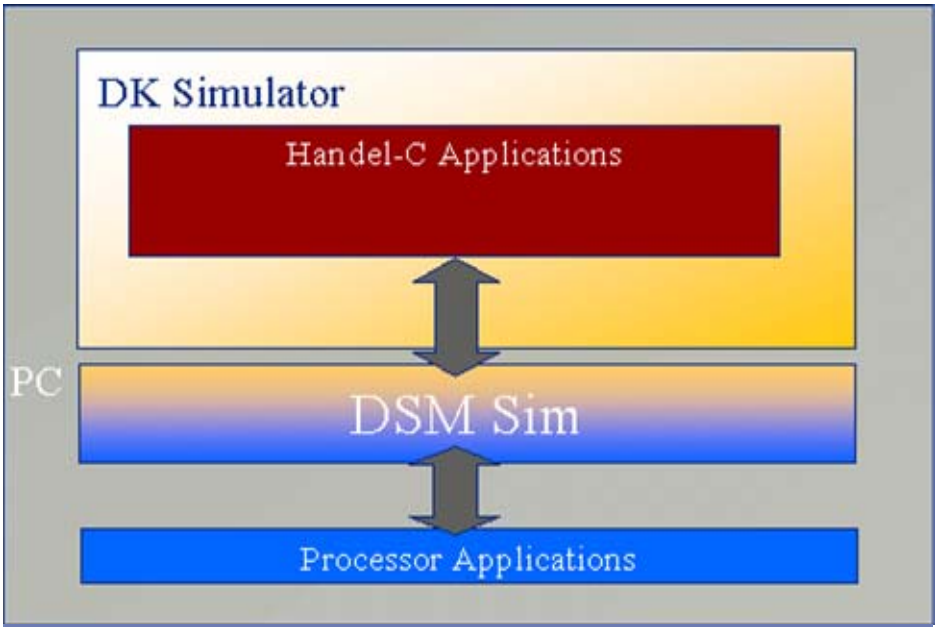
3.2 DSM API

DSM provides an API for the Hardware environment to be used with Handel-C and an API for the software environment for use with ANSI-C. These APIs provide functions for initialization, data transfer, transfer control, program control and shutdown. The DSM kit

provides header files that contain the declarations of all the types and macros together with libraries that provide the platform specific implementations of DSM.

3.3 DSM Sim

DSM Sim provides a functionally accurate simulation environment for DSM applications. On a PC host it allows ANSI-C programs and Handel-C programs to interact using the DSM APIs. The ANSI-C program is run as a native executable on the PC, the Handel-C program is run using the simulation and debugging capability of Celoxica's DK. All of the API functions are provided allowing a complete system to be designed and developed without requiring a microprocessor/FPGA development platform. Once working, the application can be easily transferred to the target platform for final testing.



DSM SIM

3.4 DSM Kit

Included with DSM are all of the components necessary for the development of platform specific DSM implementations. As well as basic header files and libraries, there are also tutorials, examples and templates that provide a starting point for development and guidance during implementation.

3.5 DSM supported platforms

Version 4.0 of DSM, including version 2.0 of the DSM API provides ready-to-use implementations for the following platforms:

- DSM Sim
- RC2000 and RC2000Pro with Windows-based PC
- Xilinx MicroBlaze on RC10, RC100, RC200, RC203 and RC300 platforms
- Xilinx Virtex-II Pro embedded PowerPC processor
- Altera NiosII on RC250 and Nios II Development board

4 Index