|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | | |  |
|  | | |
|  | | |
| ½Ô²穙k1 | | | |
|  | **FACS**  User Development Guide | | ¸½¼þ1-16K |
|  | |
| **Issue** | **01** |
| **Release Date** | **2017-01-17** |
| **Huawei Technologies Co., Ltd.** | |

|  |
| --- |
| Copyright © Huawei Technologies Co., Ltd. 2017. All rights reserved.  No part of this document may be reproduced or transmitted in any form or by any means without prior written consent of Huawei Technologies Co., Ltd.  Trademarks and Permissions  附件3-图 and other Huawei trademarks are trademarks of Huawei Technologies Co., Ltd.  All other trademarks and trade names mentioned in this document are the property of their respective holders.  Notice  The purchased products, services and features are stipulated by the contract made between Huawei and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.  The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied. |

|  |  |
| --- | --- |
| Huawei Technologies Co., Ltd. | |
| Address: | Huawei Industrial Base  Bantian, Longgang  Shenzhen 518129  People's Republic of China |
| Website: | <http://e.huawei.com> |
| Email: | [support@huawei.com](mailto:Support@huawei.com) |

Contents

[1 Introduction to FACS Interfaces 3](#_Toc534810060)

[1.1 Overview 3](#_Toc534810061)

[1.2 Terminology 3](#_Toc534810062)

[1.3 Specifications 4](#_Toc534810063)

[1.3.1 PCIe Interfaces 4](#_Toc534810064)

[1.3.2 DDR Interfaces 4](#_Toc534810065)

[1.3.3 Queues 4](#_Toc534810066)

[1.3.4 BAR Space 4](#_Toc534810067)

[1.4 Hardware Description 4](#_Toc534810068)

[1.4.1 Hardware Resources 4](#_Toc534810069)

[2 FACS General-Purpose Interface (SDAccel) Description 6](#_Toc534810070)

[2.1 FACS SDAccel Overview 6](#_Toc534810071)

[2.2 Terminology 6](#_Toc534810072)

[2.3 FACS SDAccel Implementation Scheme Description 6](#_Toc534810073)

[2.4 FACS SDAccel Development Description 9](#_Toc534810074)

[2.4.1 Host Code Development 9](#_Toc534810075)

[2.4.2 Kernel Code Development 9](#_Toc534810076)

[2.5 Example Design Description 9](#_Toc534810077)

# Introduction to FACS Interfaces

## Overview

This chapter describes acceleration design, including interfaces, syntax, usage examples, and hardware resources and interfaces available for Huawei FACS users.

## Terminology

FACS: indicates FPGA Accelerated Cloud Server, and FPGA is short for Field Programmable Gate Array.

HDK: indicates hardware develop kit, consisting of coding environment, simulation platform, automated compilation tools, and code encryption and debugging tools.

SDK: indicates software develop kit, consisting of application examples, hardware abstract interfaces, accelerator abstract interfaces, an accelerator driver, runtime, and a version management tool.

Shell: indicates static logic, which is provided by Huawei, including external interfaces such as PCIe and DDR4.

UL: indicates user logic, which is provided by a developer.

PF: indicates physical function.

VF: indicates virtual function.

DMA: indicates direct memory access.

HPI: indicates high performance interface.

BD: indicates buffer description.

PR: indicates partial reconfiguration.

AE: indicates acceleration engine.

BAR: indicates base address register.

AXI4: indicates ARM Advanced eXtensible Interface.

AXI4-Stream: indicates ARM Advanced eXtensible Stream Interface.

AXI4-Lite: indicates ARM Advanced eXtensible Lite Interface.

M-: indicates the master end of the AXI bus.

S-: indicates the slave end of the AXI bus.

PMD: indicates poll mode drivers.

## Specifications

### PCIe Interfaces

Each FPGA supports a PCIe x16 interface, with a maximum bandwidth of 110 Gbit/s.

### DDR Interfaces

Each FPGA supports four DDR4 channels. The single-channel capacity of a DDR4 is 16 GB, and the total capacity is 64 GB.

### Queues

One PF.

One VF.

Each VF supports eight queues.

### BAR Space

UL VF BAR space specifications:

BAR2 is 64-bit, and the space is 16 Mbytes.

BAR4 is 64-bit, and the space is 512 Mbytes.

## Hardware Description

### Hardware Resources



The following resources are available:

* One PCIe Gen3 x16 interface;
* Four DDR4 DIMM interfaces;
* Nine ETH300G interfaces (100 Gbit/s).

DDR partitioning:

* One DDR controller is placed in the static logic partition.
* Three DDR controllers are placed in the UL partition.
* A maximum of four DDR controllers can be used.

# FACS General-Purpose Interface (SDAccel) Description

## FACS SDAccel Overview

Developed based on Xilinx SDAccel 2017.4.op, FACS SDAccel supports OpenCL development and application on VMs and is compatible with Xilinx SDAccel 2017.4.op. This chapter describes the implementation scheme of FACS SDAccel and how to develop host and kernel code.

## Terminology

HAL: indicates hardware abstraction layer.

XDMA: indicates Xilinx direct memory access.

Mgmt.: indicates management.

Drv.: indicates driver.

PF: indicates physical function.

OpenCL: indicates open computing language.

## FACS SDAccel Implementation Scheme Description

The SDAccel implementation scheme can be deployed on VMs, and you can develop, debug, and apply OpenCL host and kernel code on VMs. For security, the FPGA management is implemented by the host. Therefore, some kernel management initiated on VMs is implemented on the host, for example, loading an xclbin file.

FACS SDAccel implementation scheme



Figure description:

OpenCL runtime: indicates the runtime of Xilinx OpenCL and displays OpenCL APIs to users.

HAL: adapts OpenCL runtime and kernel driver and manages global memory addresses.

XDMA Drv.: indicates Xilinx DMA kernel driver.

fpga\_tool: provides basic FPGA management tools, such as the FPGA kernel loading command tool and loading status querying tool.

Host Mgmt. Drv.: indicates the management driver running on the host and implements FPGA kernel loading.

User PF: indicates the PF interface on the user plane and is directly connected to a VM, providing FPGA access channels for users.

Mgmt. PF: indicates the PF interface on the management plane. This interface is a channel for the host FPGA to access FPGA.

Based Region: indicates the area of FPGA static logic.

Expanded Region: indicates the area of FPGA dynamic logic.

User PF and Mgmt. PF are bound to user VM and server host respectively. Therefore, limited management commands can be executed directly on VMs. FACS SDAccel provides management tool fpga\_tool on VMs, and users need to use fpga\_tool for management operations on VMs. In addition, compared with Xilinx xbsak, fpga\_tool supports limited commands and does not support all xbsak commands. Therefore, some API users at the HAL layer cannot use fpga\_tool on VMs. The following table describes HAL APIs supported by FACS SDAccel.

| No. | API | Remarks |
| --- | --- | --- |
| 1 | xclOpen( ) |  |
| 2 | xclClose( ) |  |
| 3 | xclGetDeviceInfo2( ) | Support after modification. |
| 4 | xclGetAXIErrorStatus( ) |  |
| 5 | xclLoadXclBin( ) | Support after modification. |
| 6 | xclAllocDeviceBuffer( ) |  |
| 7 | xclAllocDeviceBuffer2( ) |  |
| 8 | xclFreeDeviceBuffer( ) |  |
| 9 | xclCopyBufferHost2Device( ) |  |
| 10 | xclCopyBufferDevice2Host( ) |  |
| 11 | xclWrite( ) |  |
| 12 | xclRead( ) |  |
| 13 | xclReClock2( ) | Support after modification. |
| 14 | xclProbe( ) |  |
| 15 | xclLockDevice( ) |  |
| 16 | xclWriteHostEvent( ) |  |
| 17 | xclGetDeviceTimestamp( ) |  |
| 18 | xclGetDeviceClockFreqMHz( ) |  |
| 19 | xclGetReadMaxBandwidthMBps( ) |  |
| 20 | xclGetWriteMaxBandwidthMBps( ) |  |
| 21 | xclSetOclRegionProfilingNumberSlots( ) |  |
| 22 | xclPerfMonClockTraining( ) |  |
| 23 | xclPerfMonStartCounters( ) |  |
| 24 | xclPerfMonStopCounters( ) |  |
| 25 | xclPerfMonReadCounters( ) |  |
| 26 | xclPerfMonStartTrace( ) |  |
| 27 | xclPerfMonStopTrace( ) |  |
| 28 | xclPerfMonGetTraceCount( ) |  |
| 29 | xclPerfMonReadTrace( ) |  |

## FACS SDAccel Development Description

### Host Code Development

Host code development supports all OpenCL APIs that are supported by Xilinx SDAccel 2017.4.op. For details, see Xilinx *UG1023*. You can develop host code based on the interfaces supported by Xilinx SDx 2017.4.op.

The following is the link of *UG1023*:

<https://www.xilinx.com/support/documentation/sw_manuals/xilinx2017_4/ug1023-sdaccel-user-guide.pdf>

### Kernel Code Development

Kernel code development supports OpenCL C syntax that is supported by Xilinx SDAccel 2017.4.op. For details, see *UG1023* and *UG1207*. Users can develop kernel code based on the syntax supported by Xilinx SDx 2017.4.op.

The following is the link of *UG1207*:

<https://www.xilinx.com/support/documentation/sw_manuals/xilinx2017_4/ug1207-sdaccel-optimization-guide.pdf>

## Example Design Description

FACS SDAccel provides three examples, which describe the process of development and debugging in three different kernel development modes, including RTL, HLS, and OpenCL C. Each example consists of two parts:

* The development and simulation paths of kernel are stored in the **/huaweicloud-fpga/fp1/hardware/sdaccel\_design/examples** directory, and corresponding execution scripts are provided.
* The development and on-board debugging paths of the host code are stored in the **/huaweicloud-fpga/fp1/software/app/sdaccel\_app** directory, and corresponding execution scripts are provided.
* example vadd\_cl

This example shows the kernel development process using OpenCL C. The kernel source code comes from **<SDx\_install\_area>SDx/2017.4.op /examples/vadd** of the Xilinx SDx. In addition, the example provides kernel compilation and simulation scripts. If you choose the OpenCL C development mode, replace the kernel source code with your own code, and then run the scripts provided by this example to perform compilation and simulation debugging, or on-board debugging.

* example mmult\_hls

This example shows the kernel development process using HLS. The kernel source code comes from **<SDx\_install\_area>SDx/2017.4.op/ samples/mmult** of the Xilinx SDx installation tool. In addition, the example provides kernel compilation and simulation scripts. If you choose the HLS development mode, replace the kernel source code with your own code, and then run the scripts provided by this example to perform compilation and simulation debugging, or on-board debugging.

* example vadd\_rtl

The example shows the kernel development process using HLS. The kernel source code comes from an open-source example on Xilinx github. For details, visit <https://github.com/Xilinx/SDAccel_Examples/tree/master/getting_started/rtl_kernel>. In addition, this example provides kernel compilation and simulation scripts. If you choose the RTL development mode, prepare the RTL kernel source code and XML description files, and then run the scripts provided by this example to perform compilation and simulation debugging, or on-board debugging.

For details about the SDAccel development and debugging process, visit <https://www.xilinx.com/support/documentation/sw_manuals/xilinx2017_4/ug1021-sdaccel-intro-tutorial.pdf>.