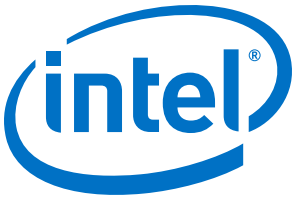


Intel® FPGA SDK for OpenCL™

Intel® Stratix® 10 GX FPGA Development Kit Reference Platform Porting Guide

Updated for Intel® Quartus® Prime Design Suite: **19.1**

UG-20102 | 2019.04.01
Latest document on the web: **PDF** | **HTML**

Subscribe
Send Feedback

Contents

UG-20102 | 2019.04.01

Send Feedback

1. Intel® FPGA SDK for OpenCL™ Intel® Stratix® 10 GX FPGA Development Kit Reference Platform Porting Guide

The *Intel® Stratix® 10 GX FPGA Development Kit Reference Platform Porting Guide* describes procedures and design considerations for modifying the Intel Stratix 10 GX FPGA Development Kit Reference Platform (s10_ref) into your own Custom Platform for use with the Intel FPGA Software Development Kit (SDK) for OpenCL™ (1) (2).

1.1. Introduction to Intel Reference Platform

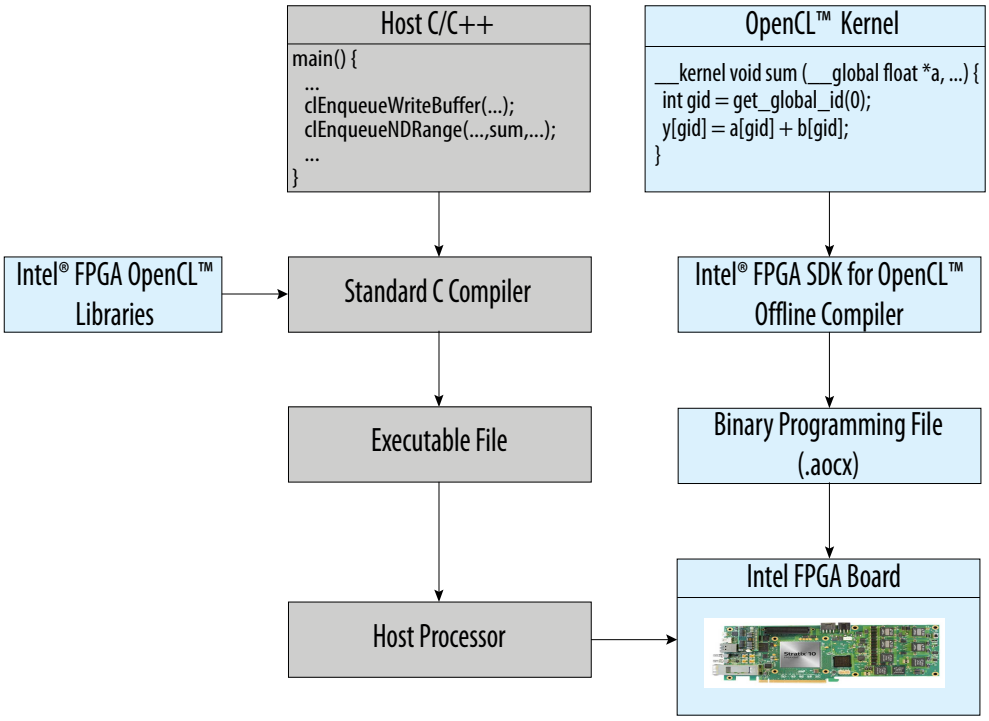
The Intel FPGA SDK for OpenCL provides you an environment to target FPGAs while abstracting FPGA details.

(1) OpenCL and the OpenCL logo are trademarks of Apple Inc. used by permission of the Khronos Group™.

(2) The Intel FPGA SDK for OpenCL is based on a published Khronos specification, and has passed the Khronos Conformance Testing Process. Current conformance status can be found at www.khronos.org/conformance.

It allows you to target Intel FPGA devices either on reference platforms provided by Intel or Intel board partners, or on your own custom platforms. A typical setup for using the SDK is illustrated in the following image:

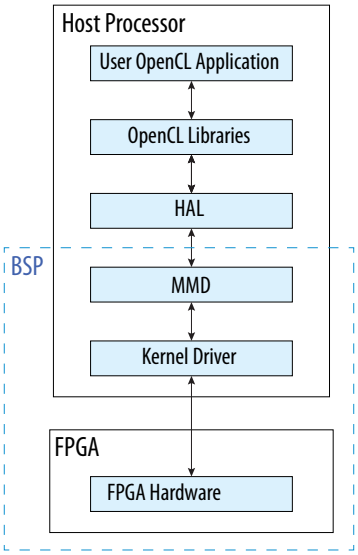
Figure 1. Setup for Using Intel FPGA SDK for OpenCL



The setup consists of a host application running on the host processor and offloading kernel tasks to the FPGA. The OpenCL kernel is converted to a hardware circuit by the SDK compiler. Leveraging this capability for your FPGA platform requires an Intel FPGA SDK for OpenCL-compatible Board Support Package (BSP). The BSP describes the reference platform to the SDK.

The following illustration depicts segments of the Intel FPGA SDK for OpenCL solution:

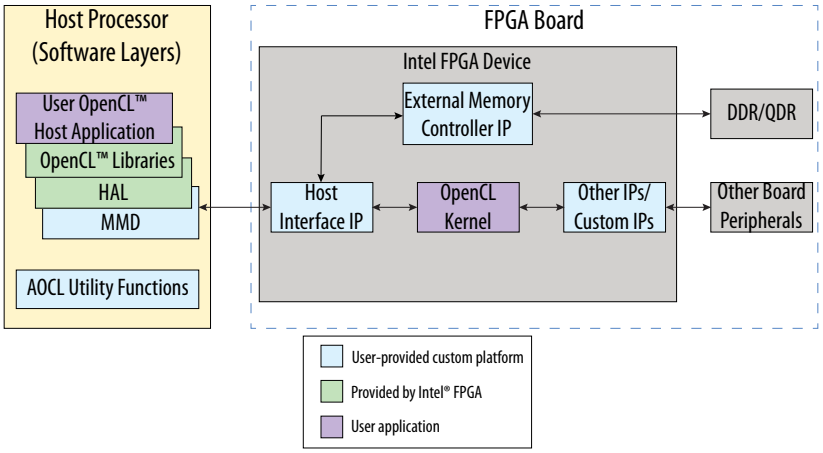
Figure 2. Layers of Intel FPGA SDK for OpenCL



Your host application communicates with the BSP layers through the Hardware Abstraction Layer (HAL). A typical Intel BSP consists of software layers and a hardware project created using the Intel Quartus® Prime Pro Edition software. The hardware project consists of FPGA board peripheral IPs and custom IPs.

The following illustration depicts the hardware project components, and how these components communicate with software layers:

Figure 3. Intel FPGA SDK for OpenCL Complete Solution



In Figure 3 on page 6, left side depicts the host application running on host processor while right side depicts the FPGA hardware acceleration board. If you are developing a custom platform to run your software applications, then you must create a custom BSP for your platform. In that case, everything that appears in blue in the image must be included in your custom BSP as follows:

- On the FPGA side, your custom BSP must include all hardware necessary to communicate with the host and the memory, that is, DDR and/or QDR memory interfaces, the DMA host interface (which can be PCIe), and any streaming interfaces to be implemented as channels. The Intel FPGA SDK for OpenCL compiles your OpenCL kernel into a data flow circuit, connects to the BSP hardware components, and generates an FPGA image for this combined circuit, which is used to configure the FPGA.
- On the host side, your custom BSP must provide the Memory Mapped Device layer (MMD) (in the form of a library) to facilitate communication between OpenCL libraries and your hardware. When you compile your host application, the host application links with both the Intel FPGA SDK for OpenCL and MMD libraries to form a host executable.

Intel provides reference BSPs for Intel FPGA development kits. Most of these reference BSPs are included in the installed directory for Intel FPGA SDK for OpenCL:

Table 1. Reference BSPs for Intel FPGA development kits

Reference BSP	Install Path
Intel Arria® 10 GX FPGA Reference Platform	Installed with the Intel FPGA SDK for OpenCL in *INTELFPGAOCLSDKROOT*/board/a10_ref.
Intel Stratix 10 GX FPGA Reference Platform	Installed with the Intel FPGA SDK for OpenCL in *INTELFPGAOCLSDKROOT*/board/s10_ref.
Intel Arria 10 SoC FPGA Reference Platform	Installed with the Intel FPGA SDK for OpenCL in *INTELFPGAOCLSDKROOT*/board/a10soc.

Attention: Cyclone® V SoC and Stratix V GX reference platforms are also available within the Intel FPGA SDK for OpenCL installation of version 18.0 and earlier.

You can use one of the above BSPs as a reference to get started with custom BSP development. You can also use reference platforms from one of the following Intel FPGA's preferred board partners and download BSP from their website if it matches your hardware requirements:

- Terasic Inc.
- REFLEX CES

Related Information

Building Custom Platforms for Intel® FPGA SDK for OpenCL™: BSP Basics

1.2. Intel Stratix 10 GX FPGA Development Kit Reference Platform: Prerequisites

The *Intel Stratix 10 GX FPGA Development Kit Reference Platform Porting Guide* assumes that you are an experienced FPGA designer familiar with Intel's FPGA design tools and concepts.

Prerequisites for the `s10_ref` Reference Platform:

- An Intel Stratix 10-based accelerator card with working PCI Express* (PCIe*) and memory interfaces

  Attention: The native Stratix 10 GX FPGA Development Kit does not automatically work with the SDK. Before using the Stratix 10 GX FPGA Development Kit with the SDK, you must setup the board by following the steps provided in the bring-up guide (included in the *INTELFPGAOCLSDKROOT*`/board/s10_ref/bringup` directory) or contact your field applications engineer or regional support center representative to configure the development kit for you.

  Alternatively, contact Intel Premier Support for assistance.
- Intel Quartus Prime Pro Edition software
- Designing with Logic Lock regions

General prerequisites:

- FPGA architecture, including clocking, global routing, and I/Os
- High-speed design
- Timing analysis
- Platform Designer design and Avalon® interfaces
- Tcl scripting
- PCIe
- DDR4 external memory

This document also assumes that you are familiar with the following Intel FPGA SDK for OpenCL-specific tools and documentation:

- Custom Platform Toolkit and the *Intel FPGA SDK for OpenCL Custom Platform Toolkit User Guide*
- Intel Arria 10 Reference Platform (a10_ref) and the *Intel FPGA SDK for OpenCL Intel Arria 10 GX FPGA Development Kit Reference Platform Porting Guide*

  The whole software stack in the `s10_ref` is derived from the `a10_ref` Reference Platform.

Related Information

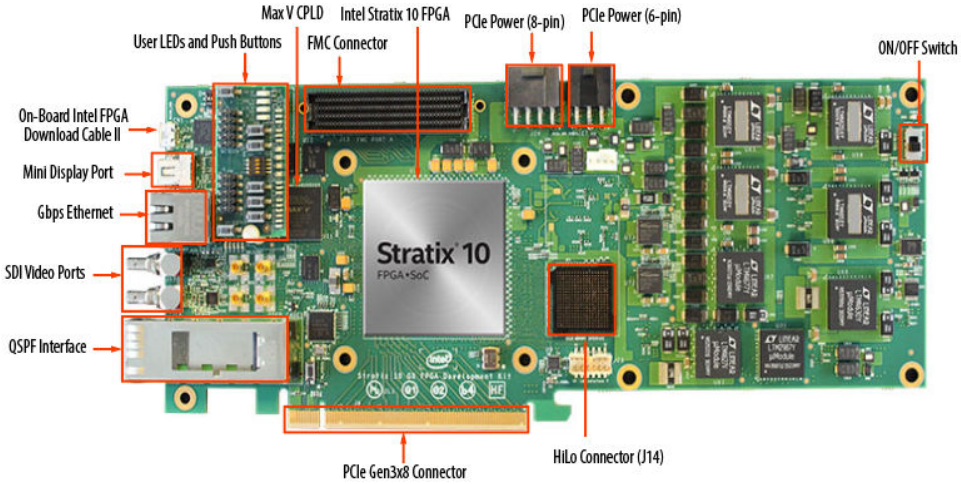
- Intel FPGA SDK for OpenCL Custom Platform Toolkit User Guide
- Intel FPGA SDK for OpenCL Intel Arria 10 GX FPGA Development Kit Reference Platform Porting Guide
- Intel FPGA SDK for OpenCL Intel Arria 10 SoC Development Kit Reference Platform Porting Guide
- Intel FPGA SDK for OpenCL Intel Cyclone V SoC Development Kit Reference Platform Porting Guide
- Intel FPGA SDK for OpenCL Intel Stratix V Network Reference Platform Porting Guide

1.3. Features of the Intel Stratix 10 GX FPGA Development Kit Reference Platform

Prior to designing an Intel FPGA SDK for OpenCL Custom Platform, decide on design considerations that allow you to fully utilize the available hardware on your computing card.

The Intel Stratix 10 GX FPGA Development Kit Reference Platform targets a subset of the hardware features available in the Intel Stratix 10 GX FPGA Development Kit.

Figure 4. Hardware Features of the Intel Stratix 10 GX FPGA Development Kit



Features of the s10_ref Reference Platform:

- OpenCL Host

  The s10_ref Reference Platform uses a PCIe-based host that connects to the Intel Stratix 10 PCIe Gen3x8 hard IP core.

- OpenCL Global Memory

  The hardware provides one 2-gigabyte (GB) DDR4 SDRAM daughtercard that is mounted on the HiLo connector and instantiated in the design using Intel Stratix 10 External Memory Interface IP (J14 in Figure 4 on page 9).

- FPGA Programming

  Via external cable and the Intel Stratix 10 GX FPGA Development Kit's on-board Intel FPGA Download Cable II interface.

- Guaranteed Timing

  The s10_ref Reference Platform relies on the Intel Quartus Prime Pro Edition compilation flow to provide guaranteed timing closure. The timing-clean s10_ref Reference Platform is preserved in the form of a precompiled post-fit netlist (that is, the `base.qdb` Intel Quartus Prime Database Export File). The Intel FPGA SDK for OpenCL Offline Compiler imports this preserved post-fit netlist into each OpenCL kernel compilation.

1.3.1. Intel Stratix 10 GX FPGA Development Kit Reference Platform Board Variants

The Intel Stratix 10 GX FPGA Development Kit Reference Platform has one board variant (that is, s10gx) that targets the Intel Stratix 10 GX FPGA Development Kit containing the Intel Stratix 10 DDR4-1866 SDRAM.

To compile your OpenCL kernel for a specific board variant, include the `-board=<board_name>` option in your `aoc` command (for example, `aoc -board=s10gx myKernel.cl`).

Related Information

Compiling a Kernel for a Specific FPGA Board and Custom Platform (-board=<board_name>) and (-board-package=<board_package_path>)

1.4. Contents of the Intel Stratix 10 GX FPGA Development Kit Reference Platform

Familiarize yourself with directories and files within the Intel Stratix 10 GX FPGA Development Kit Reference Platform (*INTELFPGAOCLSDKROOT*`/board/s10_ref` because they are referenced throughout this document.

Table 2. Highlights of the Intel Stratix 10 GX FPGA Development Kit Reference Platform Directory

Windows File or Folder	Linux File or Directory	Description
`board_env.xml`	`board_env.xml`	eXtensible Markup Language (XML) file that describes the Reference Platform to the Intel FPGA SDK for OpenCL.
`bringup`	`bringup`	Contains initialization binaries and Intel Stratix 10 Development Kit Initialization guide (`S10_DevKit_Initialization`).
`hardware`	`hardware`	Contains the Intel Quartus Prime project templates for the s10gx board variant. See Table 3 on page 11 for a list of files in this directory.
`windows64`	`linux64`	Contains the MMD library, kernel mode driver, and executable files of the SDK utilities (that is, `install`, `uninstall`, `flash`, `program`, `diagnose`) for your 64-bit operating system.
`source`	`source`	For Windows, the `source` folder contains source codes for the MMD library and SDK utilities. The MMD library and the SDK utilities are in the `windows64` folder. For Linux, the `source` directory contains source codes for the MMD library and SDK utilities. The MMD library and the SDK utilities are in the `linux64` directory.
`scripts`	`scripts`	Contains the `find_jtag_cable.tcl` script that is useful in identifying the cable and index number required for FPGA programming.

Table 3. Contents of the s10gx Directory

The following table lists the files in the *INTELFPGAOCLSDKROOT*/board/s10_ref/hardware/s10gx directory, where *INTELFPGAOCLSDKROOT* points to the location of the SDK installation.

File	Description
mem.qsys	Platform Designer system that, together with the .ip files in the ip/mem/ sub-directory, implements the mem component. It can be recognized as the memory subsystem instantiated in the board.qsys. It contains EMIF hard IP, AVMM S10 CCBs (Clock Crossing Bridge), ACL Uniphy Status and ACL SW Reset (Calibration) components.
board.qsys	Platform Designer system that implements the board interfaces (that is, the static region) of the OpenCL hardware system.
device.tcl	Tcl file that is included in all revisions and contains all device-specific information (for example, device family, ordering part number (OPN), voltage settings, pin assignments and so on). It is sourced in other QSF files, such as opencl_bsp_ip.qsf and flat.qsf.
opencl_bsp_ip.qsf	Intel Quartus Prime Settings File that collects all the required .ip files in a unique location. During flat and base revision compilations, the board.qsys and mem.qsys related IP files are added to the opencl_bsp_ip.qsf file.
flat.qsf	Intel Quartus Prime Settings File for the flat project revision. This file includes all common settings, such as VID and global signal settings, that are used in other revisions of the project (that is, base and top). The base.qsf and top.qsf files include, by reference, all settings in the flat.qsf file. The Intel Quartus Prime software compiles the flat revision with minimal constraints. The flat revision compilation does not generate a base.qar file that you can use for future import compilations and does not implement the guaranteed timing flow. It is used to make edits and check functionality of the design.
base.qsf	Intel Quartus Prime Settings File for the base project revision. This file includes, by reference, all the settings in the flat.qsf file. The Intel Quartus Prime Pro Edition software compiles this base project revision from source code unlike the top compile that uses the base.qar output.
top.qsf	Intel Quartus Prime Settings File for the SDK-user compilation flow (import compilation flow).
top.v	Top-level Verilog Design File for the OpenCL hardware system.
top.sdc	Synopsys Design Constraints File that contains board-specific timing constraints.
top_post.sdc	Platform Designer and Intel FPGA SDK for OpenCL IP-specific timing constraints.
ip/mem/<*file_name*>	Directory containing the .ip files that the Intel Quartus Prime Pro Edition software needs to parameterize the mem component. Along with mem.qsys, files in this directory are required for flat and base revision compiles. These are added to the flow by pre_flow_pr.tcl.
ip/board/<*file_name*>	Directory containing the .ip files that the Intel Quartus Prime Pro Edition software needs to parameterize the board instance. Along with board.qsys, files in this directory are required for flat and base revision compiles. These are added to the flow by pre_flow_pr.tcl

continued...

File	Description
ip/freeze_wrapper.v	Verilog Design File that implements the *freeze* logic. Freeze logic allows user to construct the building blocks for a design that is suitable for Partial Reconfiguration.
ip/pr_region.v	Verilog Design File that contains the Partial Reconfiguration (PR) region logic.
ip/temperature/<file_name>	A wrapper to the actual temperature IP that has an Avalon streaming interface. The wrapper sets the necessary default values and converts the interface to an AVMM interface so that it can be used by other IPs.
ip/irq_controller/*<file_name>*	IP that receives interrupts from the OpenCL kernel system and sends message signaled interrupts (MSI) to the host. Refer to the *Message Signaled Interrupts* section for more information.
compile_script.tcl	Tcl script for SDK compilation flows.
scripts/create_fpga_bin_pr.tcl	Tcl script that generates the ELF binary file, fpga.bin from .sof, .rbf, and pr_base.id files. The fpga.bin file contains all files necessary for configuring the FPGA.
scripts/qar_ip_files.tcl	Tcl script that packages up base.qdb, pr_base.id, base.sdc, board and mem Platform Designer system generation output during base revision compile.
scripts/helpers.tcl	Tcl script with helper functions used by qar_ip_files.tcl.
scripts/post_flow_pr.tcl	Tcl script that runs after every Intel Quartus Prime Pro Edition software compilation. It facilitates the guaranteed timing flow by setting the kernel clock PLL, generating a small report in the acl_quartus_report.txt file, and rerunning STA with the modified kernel clock settings.
scripts/pre_flow_pr.tcl	Tcl script that executes before the invocation of the Intel Quartus Prime software compilation. Running the script generates the Platform Designer HDL for board.qsys.
scripts/get_static_region_kernel_fmax.tcl	Tcl script to generate reports for kernel clk worst paths in static region. *Note:* The region where pre-compiled BSP hardware design is placed is called Static region.
scripts/regenerate_cache.tcl	Helper scripts for bak flow.
scripts/base_write_sdc.tcl	Tcl script to save the SDC from a base revision compile.
scripts/create_acds_ver_hex.tcl	Tcl script to burn Intel Quartus Prime software version to ROM during compile.
adjust_plls.tcl	Tcl script that is not part of the scripts directory, but it is an important script to know about. This PLL adjustment script for the kernel clock PLL guarantees timing closure on the kernel clock by setting it to the maximum allowed frequency.
base.qar	Intel Quartus Prime Archive File that contains base.qdb, pr_base.id, base.sdc, board and mem Platform Designer generation output from base revision compile. This Intel Quartus Prime Archive file is generated by the scripts/post_flow_pr.tcl file during base revision compile and is used during top revision compilation. *base.qdb* — Intel Quartus Prime Database Export File that contains the precompiled netlist of the static regions of the design. *pr_base.id* — Text file containing a unique number for a given base compilation that the runtime uses to determine whether it is safe to use PR programming.

continued...

File	Description
	base.sdc Synopsis Design Constraint file for base revision that is generated by `base_write_sdc.tcl` during base revision compile.
`top.qpf`	Intel Quartus Prime Project File for the OpenCL hardware system.
`quartus.ini`	Contains any special Intel Quartus Prime software options that you need to compile OpenCL kernels for the s10_ref Reference Platform.
`iface.ipx`	Specifies the relative path of directories to search for IP cores. In general, `.ipx` (that is, IP Index Files) files facilitate faster searches. `iface.ipx` is a top-level `.ipx` file that references `hw_iface.iipx` and `sw_iface.iipx` (intermediate-`ipx`) files.
`hw_iface.ipx`	Intermediate IP Index file. It is an XML file that consists of `<component>` elements with attributes to define some of the BSP components.
`sw_iface.ipx`	Intermediate IP Index file.
`board_spec.xml`	XML file that provides the definition of the board hardware interfaces to the SDK.

Related Information

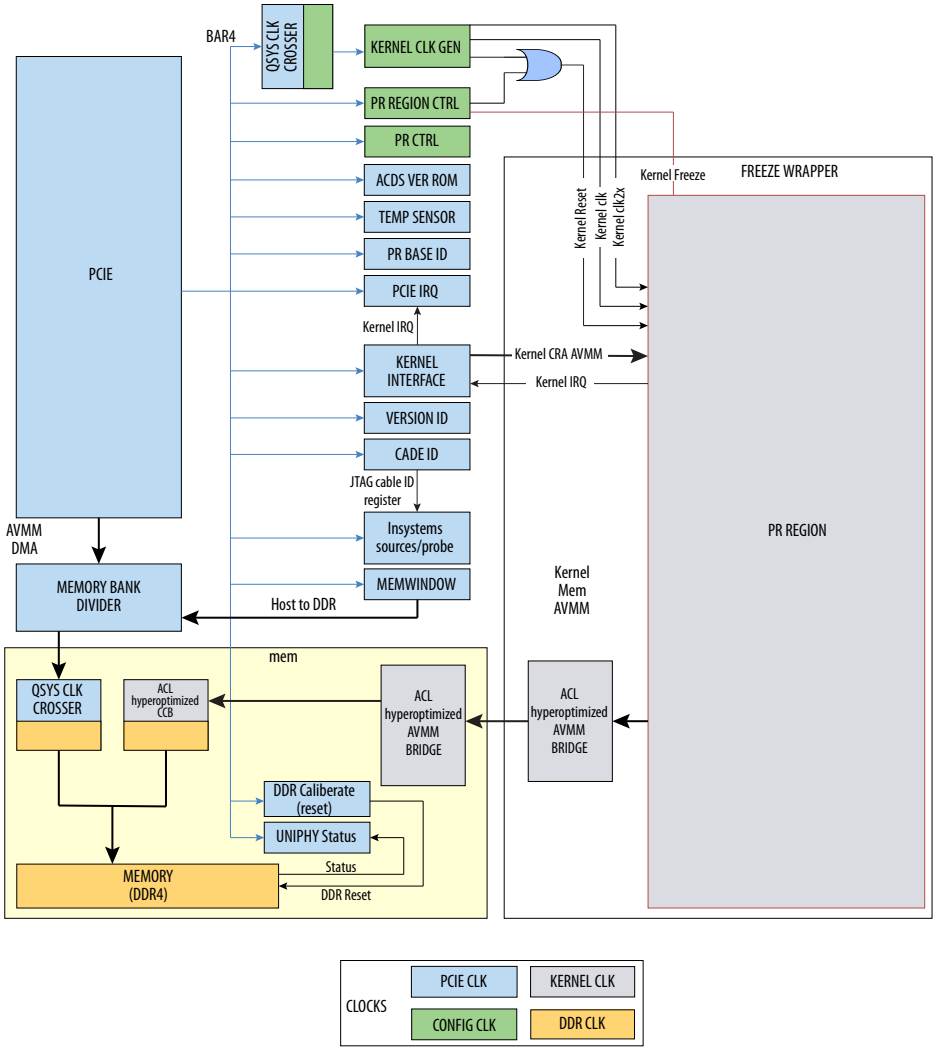
- Guaranteed Timing Closure of the Intel Stratix 10 GX FPGA Development Kit Reference Platform Design on page 37
- Intel FPGA SDK for OpenCL Compilation Flows on page 40

2. Intel Stratix 10 GX FPGA Development Kit Reference Platform Design Architecture

Intel created the Intel Stratix 10 GX FPGA Development Kit Reference Platform (s10_ref) based on various design considerations. Familiarize yourself with these design considerations. Having a thorough understanding of the design decision-making process might help in the design of your own Intel FPGA SDK for OpenCL Custom Platform.

Figure 5. System Design of Intel Stratix 10 Reference Platform

The following image provides an overview of s10_ref platform hardware project. The interface and IPs are explained in detail in the rest of this guide:



Host-to-Intel Stratix 10 FPGA Communication over PCIe on page 16
DDR4 as Global Memory for OpenCL Applications on page 27
Host Connection to OpenCL Kernels on page 30
Partial Reconfiguration on page 30
Other Components in the Reference Design on page 31
Intel Stratix 10 FPGA System Design on page 31
Guaranteed Timing Closure of the Intel Stratix 10 GX FPGA Development Kit Reference Platform Design on page 37
Intel FPGA SDK for OpenCL Compilation Flows on page 40
Addition of Timing Constraints on page 43

Connection of the Intel Reference Platform to the Intel FPGA SDK for OpenCL on page 43

Intel Stratix 10 FPGA Programming Flow on page 45

Implementation of Intel FPGA SDK for OpenCL Utilities on page 45

Considerations in Intel Stratix 10 GX FPGA Development Kit Reference Platform Implementation on page 47

2.1. Host-to-Intel Stratix 10 FPGA Communication over PCIe

The Intel Stratix 10 GX FPGA Development Kit Reference Platform instantiates the Intel Stratix 10 PCIe hard IP in `board.qsys` file to implement a host-to-device connection over PCIe.

2.1.1. Instantiation of Intel Stratix 10 PCIe Hard IP with Direct Memory Access

The Intel Stratix 10 GX FPGA Development Kit Reference Platform instantiates the Intel Stratix 10 PCIe hard IP with direct memory access (DMA) to implement a host-to-device connection over PCIe.

Dependencies

- Intel Stratix 10 PCIe hard IP core
- *Parameter Settings* section of the *Intel Stratix 10 Avalon-MM DMA Interface for PCIe Solutions User Guide*

Table 4. Highlights of Intel Stratix 10 PCIe Hard IP Parameter Settings

Set the parameters for the Intel Stratix 10 PCIe hard IP in the parameter editor within the Intel Quartus Prime Pro Edition software.

Parameter(s)	Setting
System Settings	
Application interface type	**Avalon-MM with DMA** This Avalon Memory-Mapped (Avalon-MM) interface instantiates the embedded DMA of the PCIe hard IP core. Check the **Enable Avalon-MM DMA** option under Avalon-MM settings.
Hard IP mode	**Gen3x8, Interface: 256-bit, 250 MHz** Number of Lanes: x8 Lane Rate: Gen3 (8.0 Gbps)
Avalon-MM Settings	
Export MSI/MSI-X conduit interfaces	**Enabled** Export the MSI interface in order to connect the interrupt sent from the kernel interface to the MSI.
Instantiate Internal Descriptor Controller	**Enabled** Instantiates the descriptor controller in the Avalon-MM DMA bridge. Use the 128-entry descriptor controller that the PCIe hard IP core provides.
Address width of accessible PCIe memory space	**64 bits**

continued...

Parameter(s)	Setting
	This value is machine dependent. To avoid truncation of the MSI memory address, 64-bit machines should allot 64 bits to access the PCIe address space.
Base Address Register (BAR) Settings	
Base Address Registers (BARs)	This design uses two BARs. For BAR 0, set **Type** to **64-bit prefetchable memory**. The **Size** parameter setting is disabled because the **Instantiate Internal Descriptor Controller** parameter is enabled in the Avalon-MM system settings. BAR 0 is only used to access the DMA Descriptor Controller, as described in the *Intel Stratix 10 Avalon-MM DMA for PCI Express* section of the *Intel Stratix 10 Avalon-MM DMA Interface for PCIe Solutions User Guide*. For Bar 4, set **Type** to **64-bit prefetchable memory**, and set **Size** to **18 bits (256 KBytes)**. BAR 4 is used to connect PCIe to the OpenCL kernel systems and other board modules.

Related Information

- Avalon-MM Settings
- Intel Stratix 10 Avalon-MM DMA for PCI Express Design Example

2.1.2. Device Identification Registers for Intel Stratix 10 PCIe Hard IP

To build PCIe hardware, you must set PCIe IDs related to the device hardware under **Device Identification Registers** in **PCIe IP** settings and must match software MMD since MMD uses them to identify the board.

You can find these PCIe ID definitions in the PCIe controller instantiated in the *INTELFPGAOCLSDKROOT*/board/s10_ref/hardware/s10gx/board.qsys Platform Designer System File. These IDs are necessary in the driver and the SDK's programming flow.

Table 5. Device Hardware-Related PCIe ID Registers

ID Register Name	ID Provider	Description	Parameter Name in PCIe IP Core
Vendor ID	PCI-SIG®	Identifies the FPGA manufacturer. Always set this register to **0x1172**, which is the Intel vendor ID.	`Vendor ID`
Device ID	Intel	Describes the PCIe configuration on the FPGA according to Intel's internal guideline. Set the device ID to the device code of the FPGA on your accelerator board. For the Intel Stratix 10 GX FPGA Development Kit Reference Platform, set the **Device ID** register to **0x5170**, which signifies Gen 3 speed, 8 lanes, Intel Stratix 10 device family, and Avalon-MM interface, respectively. Refer to Table 6 on page 18 for more information.	`Device ID`
Revision ID		When setting this ID, ensure that it matches the following revision IDs:	`Revision ID`

continued...

ID Register Name	ID Provider	Description	Parameter Name in PCIe IP Core
		• For Windows, the revision ID is specified in <*your_custom_platform*>\windows64\driver\Shim.inf file. • For Linux, the revision ID specified for the ACL_PCI_REVISION variable in the <*your_custom_platform*>/linux64/driver/hw_pcie_constants.h file.	
Class Code	Intel	The Intel FPGA SDK for OpenCL utility checks the base class value to verify whether the board is an OpenCL device. Do not modify the class code settings. • Base class: **0x12** for processing accelerator • Sub class: **0x00** • Programming interface: **0x01**	Class Code
Subsystem Vendor ID	Board vendor	Identifies the manufacturer of the accelerator board. Set this register to the vendor ID of manufacturer of your accelerator board. For the s10_ref Reference Platform, the subsystem vendor ID is **0x1172**. If you are a board vendor, set this register to your vendor ID.	Subsystem Vendor ID
Subsystem Device ID	Board vendor	Identifies the accelerator board. The SDK uses this ID to identify the board because the software might perform differently on different boards. If you create a Custom Platform that supports multiple boards, use this ID to distinguish between the boards. Alternatively, if you have multiple Custom Platforms, each supporting a single board, you can use this ID to distinguish between the Custom Platforms. *Important:* Make this ID unique to your Custom Platform. For example, for the s10_ref Reference Platform, the ID is **0x5170**.	Subsystem Device ID

The kernel driver uses the **Vendor ID**, **Subsystem Vendor ID** and the **Subsystem Device ID** to identify the boards it supports. The SDK's programming flow checks the **Device ID** to ensure that it programs a device with a `.aocx` Intel FPGA SDK for OpenCL Offline Compiler executable file targeting that specific device.

Table 6. Intel FPGA SDK for OpenCL's Numbering Convention for PCIe Hard IP Device ID

Location in ID	Definition
15:14	RESERVED
13:12	Speed • 0 — Gen 1 • 1 — Gen 2 • 2 — Gen 3 • 3 — Gen 4
11	RESERVED
10:8	Number of lanes

continued...

Location in ID	Definition
	• 0 — 1 lane • 1 — 2 lanes • 3 — 4 lanes • 4 — 8 lanes • 5 — 16 lanes • 6 — 32 lanes
7:4	Device family • 0 — Altera Stratix IV GX • 1 — Altera Arria II GX • 2 — Stratix II GX • 3 — Arria GX • 4 — Cyclone IV GX • 5 — External • 6 — Stratix V • 7 — Arria V • 8 — Cyclone V • 9 — Arria 10 • 10 — Stratix 10
3	1 — Soft IP (SIP) This ID indicates that the PCIe protocol stack is implemented in soft logic. If unspecified, the IP is considered a hard IP.
2:0	Platform Designer PCIe interface type • 0 — 64 bits • 1 — 128 bits • 2 — 256 bits • 3 — Desc/Data (that is, Avalon-Streaming (Avalon-ST) interface) • 4 — Avalon-MM interface

2.1.3. Instantiation of the version_id Component

Intel specifies an additional version ID IP and uses it to verify the address map of the system. The host verifies the version ID of the Intel Stratix 10 GX FPGA Development Kit Reference Platform when instantiating the version_id component that connects to the PCIe Avalon master.

The version ID for the s10_ref Reference Platform is 0xA0C7C1E5 (decimal from signed two's compliment is 1597521435).

Before communicating with any part of the FPGA system, the host first reads from this `version_id` register to confirm the following:

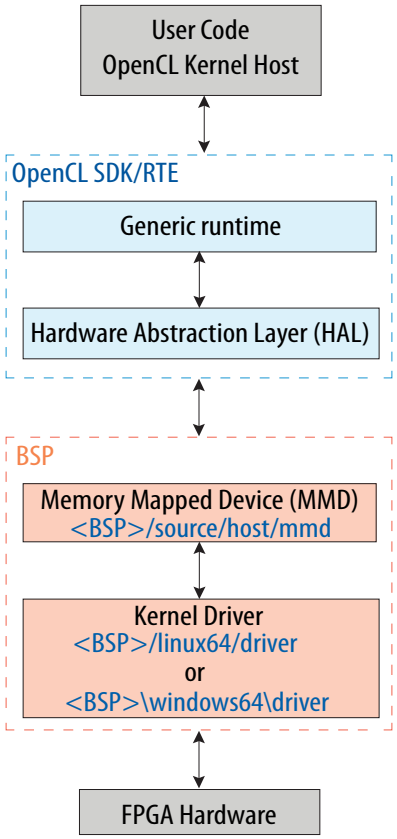
- The PCIe can access the FPGA fabric successfully
- The address map matches the map in the MMD software

Update the `VERSION_ID` parameter in the version_id component to a new value with every slave addition or removal from the PCIe BAR 4 bus, or whenever the address map changes.

2.1.4. Board Support Package Software Layer

The following image illustrates the platform software stack:

Figure 6. High Level Software Flow



The runtime and Hardware Abstraction Layer (HAL) are part of the Intel FPGA SDK for OpenCL or Intel FPGA RTE for OpenCL. The Host-to-Device Memory Mapped Device (MMD) and PCIe Kernel Driver are delivered as a part of the BSP. Hence, apart from rebranding the BSP, you might need to update some code in the MMD or driver based on changes in the hardware project.

Common Hardware Constants in Software Headers Files on page 20

PCIe Kernel Driver on page 22

Host-to-Device MMD Software Implementation on page 23

2.1.4.1. Common Hardware Constants in Software Headers Files

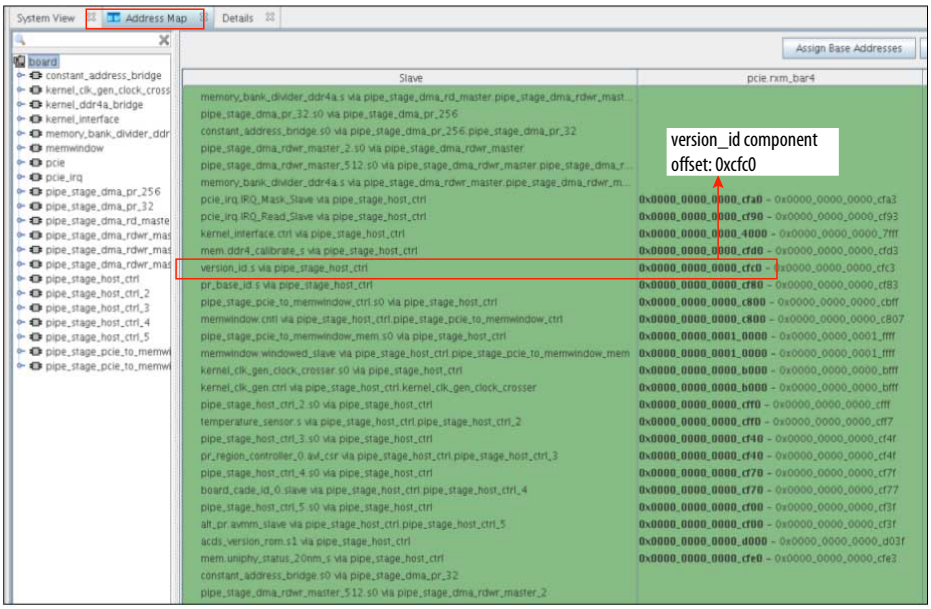
After you build the PCIe component in your hardware design, you need a software layer to communicate with the board via PCIe. To enable communication between the board and the host interface, define the hardware constants for the software in header files.

The two header files that describe the hardware design to the software are in the following locations:

- For Windows systems, the header files are in the *INTELFPGAOCLSDKROOT*\board\s10_ref\source\include folder, where *INTELFPGAOCLSDKROOT* is the path to the SDK installation.
- For Linux systems, the header files are in the *INTELFPGAOCLSDKROOT*/board/s10_ref/linux64/driver directory.

Important: Ensure that hw_pcie_constants.h and hw_pcie_dma.h header files match the MMD offsets in the address map of board.qsys. Use the **Address Map** viewer in Platform Designer to view the PCIe BAR4 offsets and match this information with hw_pcie_constants.h file as shown in the following image. Refer to the following table for more information on hw_pcie_dma.h.

Figure 7. Address Map for board.qsys from s10_ref Reference Platform



Following is a snapshot from *INTELFPGAOCLSDKROOT*/board/s10_ref/linux64/driver/hw_pcie_constants.h

```
//Version ID and Uniphy Status
#define ACL_VERSIONID_BAR                    4
#define ACL_VERSIONID_OFFSET              0xcfc0
```

Table 7. Intel Stratix 10 GX FPGA Development Kit Reference Platform Header Files

Header File Name	Description
hw_pcie_constants.h	Header file that defines most of the hardware constants for the board design specially in board.qsys. This file includes constants such as the IDs described in *PCIe Device Identification Registers*, BAR number, and offset for different components in your design. In addition, this header file also defines the name strings of ACL_BOARD_PKG_NAME, ACL_VENDOR_NAME, and ACL_BOARD_NAME.

continued...

Header File Name	Description
	Update the information in this file whenever you change the board design.
hw_pcie_dma.h	Header file that defines DMA-related hardware constants. • ACL_PCIE_DMA_ONCHIP_RD_FIFO_BASE refers to the Platform Designer address of rd_dts_slave on the PCIe IP's dma_rd_master. • ACL_PCIE_DMA_ONCHIP_WR_FIFO_BASE refers to the Platform Designer address of wr_dts_slave on the PCIe IP's dma_rd_master. Update these addresses whenever you change the board design. Refer to the *Direct Memory Access* section for more information. • ACL_PCIE_DMA_TABLE_SIZE refers to the DMA descriptor FIFO depth connected to the DMA. When using the internal descriptor controller, refer to the *DMA Descriptor Controller Registers* section in the *Intel Stratix 10 Avalon-MM DMA Interface for PCIe Solutions User Guide* for the required size. • ACL_PCIE_DMA_PAGES_LOCKED specifies the maximum pages you can lock. You may modify this constant to improve performance. • ACL_PCIE_DMA_NON_ALIGNED_TRANS_LOG specifies the starting and ending power-of-two values that non-aligned DMA transfers should have. You may modify this constant to improve performance.

Related Information

- Direct Memory Access on page 24
- Device Identification Registers for Intel Stratix 10 PCIe Hard IP on page 17
- DMA Descriptor Controller Registers

2.1.4.2. PCIe Kernel Driver

A PCIe kernel driver is necessary for the OpenCL runtime library to access your board design via a PCIe bus.

This driver is installed using the Intel FPGA SDK for OpenCL `install` utility.

The s10_ref Reference Platform

- For Windows systems, the driver is in the *<path_to_s10_ref>*`\windows64\driver` folder.
- For Linux, an open-source MMD-compatible kernel driver is in the *<path_to_s10_ref>*`/linux64/driver` directory. The table below highlights some of the files that are available in this directory.

Table 8. Highlights of the Intel Stratix 10 GX FPGA Development Kit Reference Platform's Linux PCIe Kernel Driver Directory

File	Description
pcie_linux_driver_exports.h	Header file that defines the special commands that the kernel driver supports. The installed kernel driver works as a character device. The basic operations to the driver are `open()`, `close()`, `read()`, and `write()`.

continued...

File	Description
	To execute a complicated command, create a variable as an `acl_cmd` struct type, specify the command with the proper parameters, and then send the command through a `read()` or `write()` operation. This header file defines the interface of the kernel driver, which the MMD layer uses to communicate with the device.
`aclpci.c`	File that implements the Linux kernel driver's basic structures and functions, such as the `init`, `remove`, and `probe` functions, as well as hardware design-specific functions that handle interrupts. For more information on the interrupt handler, refer to the *Message Signaled Interrupts* section.
`aclpci fileio.c`	File that implements the kernel driver's file I/O operations. The kernel driver that is available with the `s10_ref` Reference Platform supports four file I/O operations: `open()`, `close()`, `read()`, and `write()`. Implementing these file I/O operations allows the OpenCL user program to access the kernel driver through the file I/O system calls (that is, `open`, `read`, `write`, or `close`).
`aclpci cmd.c`	File that implements the specific commands defined in the `pcie_linux_driver_exports.h` file. These special commands include `SAVE_PCI_CONTROL_REGS`, `LOAD_PCI_CONTROL_REGS`, and `GET_PCI_SLOT_INFO`.
`aclpci dma.c`	File that implements DMA-related routines in the kernel driver. Refer to the *Direct Memory Access* section for more information.
`aclpci queue.c`	File that implements a queue structure for use in the kernel driver to simplify programming.

Related Information

- aocl install on page 46
- Message Signaled Interrupt on page 25
- Direct Memory Access on page 24

2.1.4.3. Host-to-Device MMD Software Implementation

The Intel Stratix 10 GX FPGA Development Kit Reference Platform's MMD layer is a thin software layer that is essential for communication between the host and the board. A full implementation of the MMD library is necessary for every Custom Platform for the proper functioning of the OpenCL host applications and board utilities. Details of the API functions, their arguments, and return values for MMD layer are specified in the *INTELFPGAOCLSDKROOT*`/board/s10_ref/source/include/aocl_mmd.h` file.

The source codes of an MMD library that demonstrates good performance are available in the *INTELFPGAOCLSDKROOT*`/board/s10_ref/source/host/mmd` directory. Refer to the *Host-to-Device MMD Software Implementation* section in the *Stratix V Network Reference Platform Porting Guide* for more information.

For more information on the MMD API functions, refer to the *MMD API Descriptions* section of the *Intel FPGA SDK for OpenCL Custom Platform Toolkit User Guide*.

Related Information

- Host-to-Device MMD Software Implementation
- MMD API Descriptions

- Intel FPGA SDK for OpenCL Stratix V Network Reference Platform Porting Guide

2.1.5. Direct Memory Access

The Intel Stratix 10 GX FPGA Development Kit Reference Platform relies on the PCIe hard IP core's soft DMA engine to transfer data. The Intel Stratix 10 PCIe hard IP core's DMA interface is instantiated as a soft IP inside the PCIe hardware when the **Avalon-MM with DMA** application interface type is selected in the IP parameter editor.

Note: The DMA interface is capable of full duplex data transfers. However, the driver handles one read or write transfer at a time.

Hardware Considerations

The instantiation process exports the DMA controller slave ports (that is, `rd_dts_slave` and `wr_dts_slave`) and master ports (that is, `rd_dcm_master` and `wr_dcm_master`) into the PCIe module. Two additional master ports, `dma_rd_master` and `dma_wr_master`, are exported for DMA read and write operations, respectively. For the DMA interface to function properly, all these ports must be connected correctly in the `board.qsys` Platform Designer system, where the PCIe hard IP is instantiated.

At the start of DMA transfer, the DMA Descriptor Controller reads from the DMA descriptor table in user memory, and stores the status and the descriptor table into a FIFO address. There are two FIFO addresses: `Read Descriptor FIFO` address and `Write Descriptor FIFO` address. After storing the descriptor table into a FIFO address, DMA transfers into the FIFO address can occur. The `dma_rd_master` port, which moves data from user memory to the device, must connect to the `rd_dts_slave` and `wr_dts_slave` ports. Because the `dma_rd_master` port connects to DDR4 memory also, the locations of the `rd_dts_slave` and `wr_dts_slave` ports in the address space must be defined in the `hw_pcie_dma.h` file.

The `rd_dcm_master` and `wr_dcm_master` ports must connect to the `txs` port. At the end of the DMA transfer, the DMA controller writes the MSI data and the `done` status into the user memory via the `txs` slave. The `txs` slave is part of the PCIe hard IP in `board.qsys`.

All modules that use DMA must connect to the `dma_rd_master` and `dma_wr_master` ports. For DDR4 memory connection, Intel recommends implementing an additional pipeline to connect the two 256-bit PCIe DMA ports to the 512-bit memory slave. For more information, refer to the *DDR4 Connection to PCIe Host* section.

Software Considerations

The MMD layer uses DMA to transfer data if it receives a data transfer request that satisfies both of the following conditions:

- A transfer size that is greater than 1024 bytes
- The starting addresses for both the host buffer and the device offset are aligned to 64 bytes

Related Information

- Common Hardware Constants in Software Headers Files on page 20
- Intel Stratix 10 DMA Avalon-MM DMA Interface to the Application Layer
- DMA Descriptor Controller Registers
- Implementing a DMA Transfer on page 25
- DDR4 Connection to PCIe Host on page 28

2.1.5.1. Implementing a DMA Transfer

Implement a DMA transfer in the MMD on Windows (*INTELFPGAOCLSDKROOT*\board\s10_ref\source\host\mmd\acl_pcie_dma_windows.cpp) or in the kernel driver on Linux (*INTELFPGAOCLSDKROOT*/board/s10_ref/linux64/driver/aclpci_dma.c).

Note: On Windows, polling is the default method for maximizing PCIe DMA bandwidth at the expense of CPU run time. To use interrupts instead of polling, assign a non-NULL value to the *ACL_PCIE_DMA_USE_MSI* environment variable.

To implement a DMA transfer:

1. Verify that the previous DMA transfer sent all the requested bytes of data.
2. Map the virtual memories that are requested for DMA transfer to physical addresses.

   Note: The amount of virtual memory that can be mapped at a time is system dependent. Large DMA transfers will require multiple mapping or unmapping operations. For a higher bandwidth, map the virtual memory ahead in a separate thread that is in parallel to the transfer.
3. Set up the DMA descriptor table on local memory.
4. Write the location of the DMA descriptor table, which is in user memory, to the DMA control registers (that is, `RC Read Status and Descriptor Base` and `RC Write Status and Descriptor Base`).
5. Write the Platform Designer address of descriptor FIFOs to the DMA control registers (that is `EP Read Descriptor FIFO Base` and `EP Write Status and Descriptor FIFO Base`).
6. Write the start signal to the RD_DMA_LAST_PTR and WR_DMA_LAST_PTR DMA control registers.
7. After the current DMA transfer finishes, repeat the procedure to implement the next DMA transfer.

Related Information

Direct Memory Access on page 24

2.1.6. Message Signaled Interrupt

The Intel Stratix 10 GX FPGA Development Kit Reference Platform uses one MSI line for both DMA and the kernel interface.

Two different modules generate the signal for the MSI line. The DMA controller in the PCIe hard IP core generates the DMA's MSI. The PCI Express interrupt request (IRQ) module (that is, the *INTELFPGAOCLSDKROOT*/board/s10_ref/hardware/s10gx/ip/irq_controller directory) generates the kernel interface's MSI.

For more information on the PCI Express IRQ module, refer to *Handling PCIe Interrupts* webpage.

Hardware Considerations

In *INTELFPGAOCLSDKROOT*/board/s10_ref/hardware/s10gx/board.qsys, the DMA MSI is connected internally; however, you must connect the kernel interface interrupt manually. For the kernel interface interrupt, the PCI Express IRQ module is instantiated as pcie_irq in board.qsys. The kernel interface interrupts connections are as follows:

- The kernel_irq_to_host port from the OpenCL Kernel Interface (kernel_interface) connects to the interrupt receiver, which allows the OpenCL kernels to signal the PCI Express IRQ module to send an MSI.
- The PCIe hard IP's msi_intfc port connects to the MSI_Interface port in the PCI Express IRQ module. The kernel interface interrupt receives the MSI address and the data necessary to generate the interrupt via msi_intfc.
- The IRQ_Gen_Master port on the PCI Express IRQ module, which is used to write the MSI, connects to the txs port on the PCIe hard IP.
- The IRQ_Read_Slave and IRQ_Mask_Slave ports connect to the pipe_stage_host_ctrl module on Bar 4. After receiving an MSI, the user driver can read the IRQ_Read_Slave port to check the status of the kernel interface interrupt, and read the IRQ_Mask_Slave port to mask the interrupt.

Software Considerations

The interrupt service routine in the Linux driver checks which module generates the interrupt. For the DMA's MSI, the driver reads the DMA descriptor table's status bit in local memory, as specified in the *Read DMA Example* section of the *Intel Stratix 10 Avalon-MM DMA Interface for PCIe Solutions User Guide*. For kernel interface's MSI, the driver reads the interrupt line sent by the kernel interface.

The interrupt service routine involves the following tasks:

1. Check DMA status on the DMA descriptor table.
2. Read the kernel status from the IRQ_READ_SLAVE port on the PCI Express IRQ module.
3. If a kernel interrupt was triggered, mask the interrupt by writing to the IRQ_MASK_SLAVE port on the PCI Express IRQ module. Then, execute the kernel interrupt service routine.
4. If a DMA interrupt was triggered, reset the DMA descriptor table and execute the DMA interrupt service routine.
5. If applicable, unmask a masked kernel interrupt.

Related Information

- Handling PCIe Interrupts
- Read DMA Example

2.1.7. Instantiation of board_cade_id_0 Component – JTAG Cable Autodetect Feature

While using the Intel FPGA SDK for OpenCL Offline Compiler program flow, the Intel Stratix 10 GX FPGA Development Kit Reference Platform automatically tries to detect the cable by default when programming the FPGA via the Intel FPGA Download Cable. This flow is useful when loading a new base image to the FPGA, where partial reconfiguration cannot be used.

You can set the ACL_PCIE_JTAG_CABLE or ACL_PCIE_JTAG_DEVICE_INDEX environment variables to disable the auto-detect feature and use values that you define.

Cable autodetect is useful when you have multiple devices connected to a single host.

The memory-mapped device (MMD) uses in-system sources and probes to identify the cable connected to the target board. You must instantiate the board_cade_id_0 register block and connect it to Bar 4 with the correct address map. You must also instantiate board_in_system_sources_probes_0, which is an in-system sources and probe component, and connect it to board_cade_id_0 register.

The MMD must be updated to take in the relevant changes. Add the scripts/find_jtag_cable.tcl script to your custom platform.

When the FPGA is being programmed via the Intel FPGA Download Cable, the MMD invokes quartus_stp to execute the find_jtag_cable.tcl script. The script identifies the cable and index number which is then used to program the FPGA through the quartus_pgm command.

2.2. DDR4 as Global Memory for OpenCL Applications

The Intel Stratix 10 GX FPGA Development Kit has one bank of 2GB x72 DDR4-1866 SDRAM. The DDR4 SDRAM is a daughtercard that is mounted to the development kit's HiLo connector.

In the current version of the s10_ref Reference Platform, all Platform Designer components related to the DDR4 global memory are now part of the *INTELFPGAOCLSDKROOT*/board/s10_ref/hardware/s10gx/mem.qsys Platform Designer subsystem within board.qsys.

Dependencies

DDR4 external memory interfaces

For more information on the DDR4 external memory interface IP, refer to the *DDR3 Board Design Guidelines* and *DDR4 Board Design Guidelines* sections in *Intel Stratix 10 External Memory Interfaces IP User Guide*.

To use the DDR4 SDRAM as global memory for Intel FPGA SDK for OpenCL designs, you must instantiate the memory controller IP, connect the memory IP to the host, and connect the memory IP to the kernel.

Related Information

- DDR3 Board Design Guidelines

- DDR4 Board Design Guidelines

2.2.1. DDR4 IP Instantiation

The Intel Stratix 10 GX FPGA Development Kit Reference Platform uses one DDR4 Controller IP to communicate with the physical memory.

Table 9. DDR4 SDRAM Controller IP Configuration Highlights

Configuration Setting	Description
Timing Parameters	As per the computing card's data specifications.
Avalon Width Power of 2	
EMIF S10 IP ➤ Memory ➤ DQ Width	Currently, OpenCL does not support non-power-of-2 bus widths. As a result, the s10_ref Reference Platform uses the option that forces the DDR4 controller to power of 2. Use the additional pins of this x72 core for error checking between the memory controller and the physical module.
Byte Enable Support	
EMIF S10 IP ➤ Memory ➤ Data Mask	Enabled. Check the **Data Mask** option in the **Memory** tab of the EMIF Intel Stratix 10 IP. Byte enable support is necessary in the core because the Intel FPGA SDK for OpenCL requires byte-level granularity to all memories.
Performance	
EMIF S10 IP ➤ Controller ➤ Enable Reordering	Enabling the reordering of DDR4 memory accesses and a deeper command queue look-ahead depth might provide increased bandwidth for some OpenCL kernels. For a target application, adjust these and other parameters as necessary. *Note:* Increasing the command queue look-ahead depth allows the DDR4 memory controller to reorder more memory accesses to increase efficiency, which improves overall memory throughput.
Debug	Disabled for production.

Important: Follow limitations, such as no support for non-power of two bus widths and need for byte-level granularity, while designing your custom boards. Develop schematics to achieve byte-granularity by the external memory components used. Failure to do so may result in poor performance of your OpenCL applications.

Related Information

External Memory Interfaces Intel Stratix 10 FPGA IP User Guide

2.2.2. DDR4 Connection to PCIe Host

Connect all global memory systems in the Intel Stratix 10 GX FPGA Development Kit Reference Platform to the host via the **OpenCL Memory Bank Divider** component (`memory_band_divider_ddr4a`) component in `board.qsys`.

The **OpenCL Memory Bank Divider** component sits in the datapath of host and FPGA memory. It accepts input from the DMA engine or BAR4 of PCIE and outputs to the FPGA memory. It is mainly useful in OpenCL BSPs with multiple memory banks, where it creates a larger memory space. Implementations of appropriate clock crossing and pipelining are based on the design floorplan and the clock domains specific to the computing card. The *OpenCL Memory Bank Divider* section in the *Intel FPGA SDK for OpenCL Custom Platform Toolkit User Guide* specifies the connection details of `acl_bsp_snoop` and `acl_bsp_memorg_host` ports.

The DDR4 IP core has one bank where its width and address configurations match those of the DDR4 SDRAM. Intel tunes the other parameters such as burst size, pending reads, and pipelining. These parameters are customizable for an end application or board design.

Important:

- When designing a multi-bank OpenCL BSP (two DDR banks), you must change *number of banks* to 2 at the instantiation of the OpenCL Memory Bank divider. Furthermore, the `acl_bsp_memorg_host` wire must be connected to the `kernel_interface` block. The Memory Bank Divider has options of being used in two modes. The **DEFAULT** option uses banks in an interleaved manner to obtain more aggregated bandwidth. It can also be used in a non-interleaved mode, which creates a contiguous address space. This flag can be set during offline compilation using `-no-interleave` flag. The **Address Span Extender** (`memwindow`) component in the host to DDR path is used to transfer the unaligned portion of the data using BAR4 during DMA transfer or when DMA is disabled.
- Instruct the host to verify the successful calibration of the memory controller.

The *INTELFPGAOCLSDKROOT*`/board/s10_ref/hardware/s10gx/board.qsys` Platform Designer system uses a custom UniPHY Status to AVS IP component to aggregate different UniPHY status conduits into a single Avalon slave port named `s`. This slave port connects to the `pipe_stage_host_ctrl` component so that the PCIe host can access it.

Related Information

OpenCL Memory Bank Divider

2.2.3. DDR4 Connection to the OpenCL Kernel

The OpenCL kernel needs to connect directly to the memory controller in the Intel Stratix 10 GX FPGA Development Kit Reference Platform via a FIFO-based clock crosser.

A clock crosser is necessary because the kernel interface for the compiler must be clocked in the kernel clock domain. In addition, the width, address width, and burst size characteristics of the kernel interface must match those specified in the OpenCL Memory Bank Divider connecting to the host. Appropriate pipelining also exists between the clock crosser and the memory controller.

To get maximum kernel clock speed for Intel Stratix 10 devices, custom hyper-optimized CCB and AVMM Bridge are used in this path. The CCB is in `mem.qsys` and the bridge is stall-free that helps in f_{max}. The two end points, clock crosser and logic in the kernel supports AVMM stall latency.

Important: Ensure that the `WAITREQUEST_ALLOWANCE` in ACL hyper-optimized CCB matches that value in `board_spec.xml` for that interface. The reference design BSP has one ACL hyper-optimized bridge in the static region and one in the kernel generated logic. Hence, a `WAITREQUEST_ALLOWANCE` value of 6 is used. Intel recommends that you leave this number as is for best performance. If you have timing closure problems, add standard Platform Designer bridges after the ACL hyper-optimized CCB in the DDR clock domain. For an additional bridge that you add, add 2 to the `WAITREQUEST_ALLOWANCE`.

2.3. Host Connection to OpenCL Kernels

The PCIe host needs to pass commands and arguments to the OpenCL kernels via the control register access (CRA) Avalon slave port that each OpenCL kernel generates.

The **OpenCL Kernel Interface for S10** (`kernel_interface`) component exports an Avalon master interface (`kernel_cra`) that connects to this slave port. The OpenCL Kernel Interface for S10 component also generates the kernel reset (`kernel_reset`) that resets all logic in the kernel clock domain. The `kernel_interface` component also bridges the interrupt signal generated by the kernel to the `pcie_irq` block.

The Intel Stratix 10 FPGA Development Kit Reference Platform has one DDR4 memory bank. As a result, the Reference Platform instantiates the OpenCL Kernel Interface component and sets the **Number of global memory systems** parameter to 1.

2.4. Partial Reconfiguration

The Intel Stratix 10 GX FPGA Development Kit Reference Platform uses partial reconfiguration (PR) as a default mechanism to reconfigure the OpenCL kernel-related partition of the design without altering the static board interface that is in a running state.

Partial Reconfiguration Controller S10 (`alt_pr`) IP	Used to help support PR in this reference platform. During the PR process, the bitstream from host is transferred to this IP.
Partial Reconfiguration Region Controller (`pr_region_controller_0`) IP	Helps with the freeze and reset logic during the PR process. The interface of the `pr_region.v` is gated with the freeze signal from this IP to keep the static region in a known state during PR. For more information, refer to *Intel Quartus Prime Pro Edition User Guide: Partial Reconfiguration*.
Register (`pr_base_id`)	Stores the unique PR BASE ID that is generated on every base compile. During reprogramming of the device, this unique ID is used to identify the top compiles that are generated from the same base build. During the Intel FPGA SDK for OpenCL Offline Compiler program flow, if the design being loaded and the existing design in the FPGA have the same PR BASE ID, then partial reconfiguration is used for reprogramming. Otherwise, full chip reprogramming is performed via JTAG interface.

Related Information

- Partial Reconfiguration IP Core
- Intel Quartus Prime Pro Edition User Guide: Partial Reconfiguration

2.4.1. Constant Address Bridge (constant_address_bridge)

The constant address bridge is the simplest block of the BSP design.

It is an IP with AVMM slave and AVMM master, with all wires in between. This means a direct feed-through except that the address is ignored and the AVMM master always outputs address 0 while also outputting a constant `0x1` burst-count. This is important for the PR IP to ensure that the entire PR bitstream is written to the same target address of the PR IP.

2.5. Other Components in the Reference Design

The following are other components in the Intel Stratix 10 GX FPGA Development Kit Reference Platform (s10_ref) that you should familiarize yourself with:

- **On-Chip Memory ROM (`acds_version_rom`)**: This read-only memory is a placeholder to burn Intel Quartus Prime software versions during bitstream compile. The version is required for checks during Intel FPGA SDK for OpenCL Offline Compiler programming flow.
- **ACL temperature sensor for S10 (`temperature_sensor`)**: It is used to read and report the temperature sensed when using `aocl diagnose` utility.

2.6. Intel Stratix 10 FPGA System Design

To integrate all components, close timing, and deliver a post-fit netlist that functions in the hardware, you must first address several additional FPGA design complexities.

Examples of design complexities:

- Designing a robust reset sequence
- Establishing a design floorplan
- Managing global routing
- Pipelining

Optimizations of these design complexities occur in tandem with one another to meet timing and board hardware optimization requirements.

2.6.1. Clocks

Several clock domains affect the Platform Designer hardware system of the Intel Stratix 10 GX FPGA Development Kit Reference Platform.

These clock domains include:

- 100 MHz PCIe clock
- 116 MHz DDR4 clock
- 50 MHz general clock (`config_clk`)
- Kernel clock that can have any clock frequency

With the exception of the kernel clock, the s10_ref Reference Platform is responsible for the timing closure of these clocks. However, because the board design must clock cross all interfaces in the kernel clock domain, the board design also has logic in the kernel clock domain. It is crucial that this logic is minimal and achieves an F_{max} higher than typical kernel performance.

Note: OpenCL S10 reconfigurable kernel clock generator (`kernel_clk_gen`) is the reconfigurable PLL that generates `kernel_clk` and `kernel_clk2x` for the kernel. You must specify the REF_CLK_RATE of the clock that you are passing to this configurable PLL. Leave the value of KERNEL_TARGET_CLOCK_RATE to 500.

Related Information

Guaranteed Timing Closure of the Intel Stratix 10 GX FPGA Development Kit Reference Platform Design on page 37

2.6.2. Resets

The Intel Stratix 10 GX FPGA Development Kit Reference Platform design includes the implementation of reset drivers.

These reset drivers include:

- The `por_reset_counter` in the *INTELFPGAOCLSDKROOT*`/board/s10_ref/hardware/s10gx/board.qsys` Platform Designer system implements the power-on-reset. The power-on-reset resets all the hardware on the device by issuing a reset for a number of cycles after the FPGA completes configuration.
- The PCIe bus issues a `perst` reset that resets all hardware on the device.
- The OpenCL Kernel Interface component issues the `kernel_reset` that resets all logic in the kernel clock domain.

The power-on-reset and the `perst` reset are combined into a single `global_reset`; therefore, there are only two reset sources in the system (that is, `global_reset` and `kernel_reset`). However, these resets are explicitly synchronized across the various clock domains, resulting in several reset interfaces.

Important Considerations Regarding Resets

- Synchronizing resets to different clock domains might cause several high fan-out resets.

  Platform Designer automatically synchronizes resets to the clock domain of each connected component. In doing so, the Platform Designer instantiates new reset controllers with derived names that might change when the design changes. This name change makes it difficult to make and maintain global clock assignments to some of the resets. As a result, for each clock domain, there are explicit reset controllers. For example, `global_reset` drives `reset_controller_pcie` and `reset_controller_ddr4`. However, they are synchronized to the PCIe and DDR4 clock domains, respectively.

- Resets and clocks must work together to propagate reset to all logic.

  Resetting a circuit in a given clock domain involves asserting the reset over a number of clock cycles. However, your design may apply resets to the PLLs that generate the clocks for a given clock domain. This means a clock domain can hold in reset without receiving the clock edge that is necessary for synchronous resets. In addition, a clock holding in reset might prevent the propagation of a reset signal because it is synchronized to and from that clock domain. Avoid such situations by ensuring that your design satisfies the following criteria:

  — Generate the `global_reset` signal off the free-running `config_clk`.

  — The ddr4_calibrate IP resets the External Memory Interface controller separately.

- Apply resets to both reset interfaces of a clock-crossing bridge or FIFO component.

  FIFO content corruption might occur if only part of a clock-crossing bridge or a dual-clock FIFO component is reset. These components typically provide a reset input for each clock domain; therefore, reset both interfaces or none at all. For example, in the `s10_ref` Reference Platform, `kernel_reset` resets all the kernel clock-crossing bridges between DDR on both the `m0_reset` and `s0_reset` interfaces.

2.6.3. Floorplan

Intel establishes the floorplan of the Intel Stratix 10 GX FPGA Development Kit Reference Platform by iterating on the design and IP placements.

Dependencies

- Chip Planner
- Logic Lock regions

Intel performed the following tasks iteratively to derive the floorplan of the s10_ref Reference Platform:

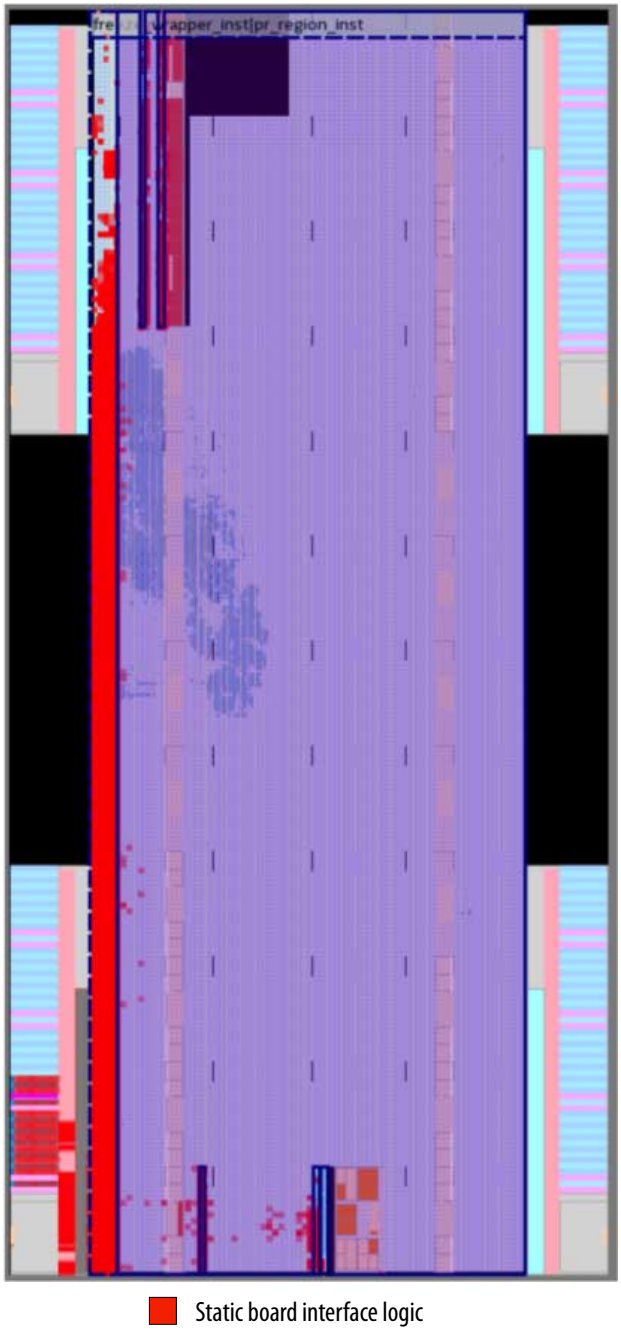
1. Compile a design without any region or floorplanning constraints in the flat revision.

Intel recommends that you compile the design with several seeds. For more information, refer to Integrating Your Intel Stratix 10 Custom Platform with the Intel FPGA SDK for OpenCL on page 51.

2. Examine the placement of the IP cores (for example, PCIe, DDR4, Avalon interconnect pipeline stages and adapters) for candidate locations, as determined by the Intel Quartus Prime Pro Edition software's Fitter. In particular, Intel recommends examining the seeds that meet or almost meet the timing constraints.

For the s10_ref Reference Platform, the PCIe I/O is located in the lower left corner of the Intel Stratix 10 FPGA. The DDR4 I/O is located on the top part of the left I/O column of the device. Because the placements of the PCIe and DDR4 IP components tend to be close to the locations of their respective I/Os, you can apply Logic Lock regions to constrain the IP components to those candidate regions.

Figure 8. Floorplan of the Intel Stratix 10 FPGA Development Kit Reference Platform



As shown in this Chip Planner view of the floorplan, the Logic Lock region (`freeze_wrapper_inst|pr_region_inst`) is spread out between the PCIe I/O and the top region of the left I/O column (that is, the DDR4 I/O area). In case of s10_ref reference platform, the Logic Lock region contains most of the kernel logic. The scatter area (shown in red) depicts the board interface (that is, static region) that is placed outside the ten Logic Lock regions assigned to kernel logic in `base.qsf`. The following figure illustrates the assignment in `base.qsf`.

Figure 9. Logic Lock Regions in `base.qsf` File in s10_ref Reference Platform

```
##########################################
#    PR region definition for the Kernel     #
##########################################
set_instance_assignment -name CORE_ONLY_PLACE_REGION ON -to freeze_wrapper_inst|pr_region_inst
set_instance_assignment -name RESERVE_PLACE_REGION ON -to freeze_wrapper_inst|pr_region_inst
set_instance_assignment -name PLACE_REGION "19 0 62 324; 63 37 66 324; 67 0 140 324; 141 37 144 324; 145 0 145 324;
146 37 150 324; 151 0 273 324; 19 325 33 432; 38 325 46 432; 51 325 273 432" -to freeze_wrapper_inst|pr_region_inst
set_instance_assignment -name ROUTE_REGION "0 0 273 432" -to freeze_wrapper_inst|pr_region_inst
```

You must create a dedicated Logic Lock region for the OpenCL kernel system for your custom platform. Furthermore, if you are logic-locking the board interface logic, ensure that you do not place kernel logic in the board's Logic Lock regions.

Intel recommends the following strategies to maximize the available FPGA resources for the OpenCL kernel system to improve kernel routability:

- The size of a Logic Lock region should be just large enough to contain the board logic and to meet timing constraints of the board clocks. Oversized Logic Lock regions consume FPGA resources unnecessarily.
- Avoid creating tightly-packed Logic Lock regions that cause very high logic utilization and high routing congestion.

  High routing congestion within the Logic Lock regions might decrease the Fitter's ability to route OpenCL kernel signals through the regions.

In the case where the board clocks are not meeting timing and the critical path is between the Logic Lock regions (that is, across region-to-region gap), insert back-to-back pipeline stages on paths that cross the gap. For example, if the critical path is between Region 1 and Region 2, lock down the first pipeline stage (an Avalon-MM Pipeline Bridge component) to Region 1, lock down the second pipeline stage to Region 2, and connect the two pipeline stages directly. This technique ensures that pipeline registers are on both sides of the region-to-region gap, thereby minimizing the delay of paths crossing the gap.

Refer to the *Pipelining* section for more information.

Related Information

- Pipelining on page 36
- Designing with Logic Lock Regions

2.6.4. Global Routing

FPGAs have dedicated clock trees that distribute high fan-out signals to various sections of the devices. In the FPGA system that the Intel Stratix 10 FPGA Development Kit Reference Platform targets, global routing can distribute high fan-out signals regionally or globally. Regional distribution applies across any quadrant of the device. Global distribution applies across the entire device.

There is no restriction on the placement location of the OpenCL kernel on the device. As a result, the kernel clocks and kernel reset must distribute high fan-out signals globally.

2.6.5. Pipelining

You must manually insert pipelines throughout the FPGA system.

In the Platform Designer, you can implement pipelines via an Avalon-MM Pipeline Bridge component by setting the following pipelining parameters within the **Avalon MM Pipeline Bridge** dialog box:

- Select **Pipeline command signals**
- Select **Pipeline response signals**
- Select both **Pipeline command signals** and **Pipeline response signals**

Examples of Pipeline Implementation

- Signals that traverse long distances because of the floorplan's shape or the region-to-region gaps require additional pipelines.

  The DMA at the bottom of the FPGA must connect to the DDR4 memory at the top of the FPGA. To achieve timing closure of the board interface logic at a DDR4 clock speed of 233 MHz, additional pipeline stages between the OpenCL Memory Bank Divider component and the DDR4 controller IP are necessary. In the Intel Stratix 10 GX FPGA Development Kit Reference Platform's `board.qsys` Platform Designer system, the pipeline stages are named `pipe_stage`.

  The middle pipeline stage, `kernel_ddra_bridge`, combines both the direct kernel DDR4 accesses and the accesses through the OpenCL Memory Bank Divider. The multistage pipeline approach ensures that the kernel entry point to the pipeline is neither geared towards the OpenCL Memory Bank Divider, which is close to the PCIe IP core, nor the DDR4 IP core, which is at the very top of the FPGA.

2.6.6. DDR4 Calibration

The Intel Stratix 10 GX FPGA Development Kit Reference Platform includes special mechanisms to ensure the functional stability of the Intel Stratix 10 silicon. For example, the DDR4 memory might not calibrate successfully after FPGA reconfiguration.

The driver within the s10_ref Reference Platform can detect a failed calibration via the Uniphy Status to AVS IP, and retrigger calibration through the ddr4_calibrate IP block.

The following two components in the `mem.qsys` design helps with the DDR calibration:

- **ACL Uniphy Status to AVS for A10 (`uniphy_status_20nm`)**: Helps to read the DDR status from the DDR IP.
- **ACL SW Reset (`ddr4_calibrate`)**: Issues reset to DDR IP.

2.7. Guaranteed Timing Closure of the Intel Stratix 10 GX FPGA Development Kit Reference Platform Design

One of the key features of the Intel FPGA SDK for OpenCL is that it abstracts away hardware details, such as timing closure, for software developers.

Both the SDK and the BSP contribute to the implementation of the SDK's guaranteed timing closure feature.

The SDK provides the IP to generate the kernel clock, and a post-flow script that ensures this clock is configured with a safe operating frequency confirmed by timing analysis. The SDK imports a post-fit netlist during a top compile that has already achieved timing closure on all non-kernel clocks.

2.7.1. Supply the Kernel Clock

In the Intel Stratix 10 GX FPGA Development Kit Reference Platform, the OpenCL Kernel Clock Generator component provides the kernel clock and its 2x variant.

The **REF_CLK_RATE** parameter specifies the frequency of the reference clock that connects to the kernel PLL (`kernel_pll_refclk`). For the s10_ref Reference Platform, the **REF_CLK_RATE** frequency is 50 MHz.

The **KERNEL_TARGET_CLOCK_RATE** parameter specifies the frequency that the Intel Quartus Prime Pro Edition software attempts to achieve during compilation. The board hardware contains some logic that the kernel clock clocks. At a minimum, the board hardware includes the clock crossing hardware. To prevent this logic from limiting the Fmax achievable by a kernel, the **KERNEL_TARGET_CLOCK_RATE** must be higher than the frequency that a simple kernel can achieve on your device. For the Intel Stratix 10 GX FPGA Development Kit that the s10_ref Reference Platform targets, the **KERNEL_TARGET_CLOCK_RATE** is 500 GHz.

2.7.2. Guarantee Kernel Clock Timing

The Intel Quartus Prime database interface executable (`quartus_cdb`) runs a script after every Intel Quartus Prime Pro Edition software compilation as a post-flow script. In the Intel Stratix 10 GX FPGA Development Kit Reference Platform, the OpenCL Kernel Clock Generator component works together with the post-flow script to guarantee kernel clock timing.

In the import (that is, top) revision compilation, the compilation script `compile_script.tcl` invokes the *`INTELFPGAOCLSDKROOT`*`/board/s10_ref/hardware/s10gx/scripts/post_flow_pr.tcl` Tcl script in the s10_ref Reference Platform after every Intel Quartus Prime Pro Edition software compilation using `quartus_cdb`.

The `post_flow_pr.tcl` script also determines the kernel clock and configures it to a functional frequency.

Important: Execute this post flow script for every Intel Quartus Prime compilation.

2.7.3. Provide a Timing-Closed Post-Fit Netlist

Each Intel FPGA SDK for OpenCL-compatible Reference and Custom Platform, such as the Intel Stratix 10 GX FPGA Development Kit Reference Platform, provides a timing-closed post-fit netlist that imports placement and routing information for all nodes clocked by non-kernel clocks.

Dependencies

Intel Quartus Prime Pro Edition compiler

Intel Quartus Prime software provides several mechanisms for preserving the placement and routing of some previously compiled logic and importing this logic into a new compilation. For Intel Stratix 10 devices, the previously compiled logic is imported into the compilation flow.

Figure 10. Custom Platform Development Flow and Hand-Off between Board Developer and SDK End User

The board developer is responsible for porting the s10_ref Reference Platform to their own board, closing timing, and locking down the static part of the board.

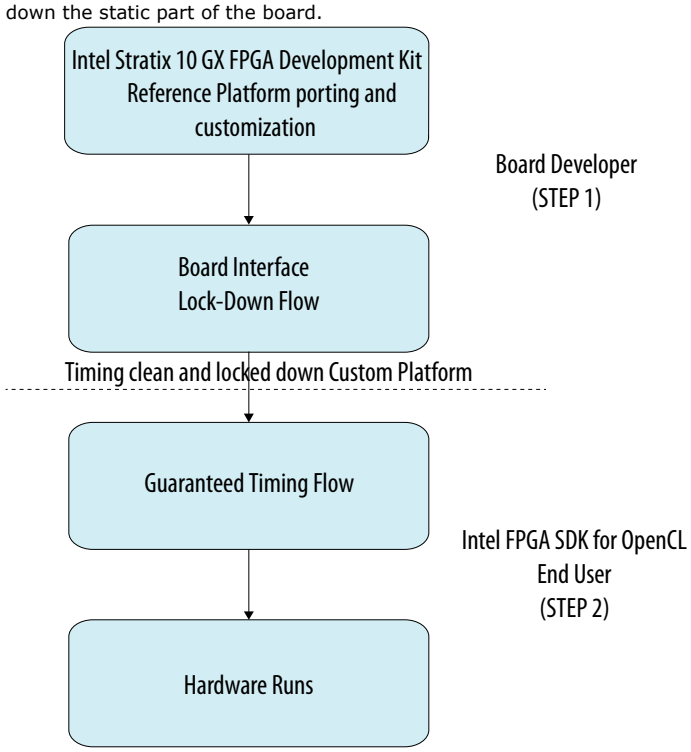
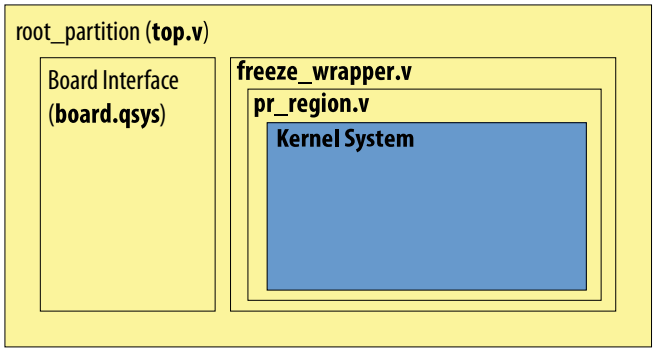


Figure 11. Structure of the Hierarchy for the OpenCL Hardware System on the Intel Stratix 10 Device

This figure illustrates that the placement and routing for everything outside the `kernel_system` partition are preserved and are imported in the top revision compilations. The `kernel_system` partition itself is not preserved and is compiled from source.



The Intel Quartus Prime Pro Edition compilation flow can preserve the placement and routing of the board interface partition via the exported Intel Quartus Prime Archive File. `base.qar` and `base.qdb` files contain all the database files for the base compilation of `root_partition`. The `s10_ref` Reference Platform is configured with the project revisions and partitioning that are necessary to implement the compilation

flow. By default, the SDK invokes the Intel Quartus Prime Pro Edition software on the top revision. This revision is configured to import and restore the `base.qdb` file, which has been precompiled and exported from a base revision compilation.

When developing your Custom Platform from the s10_ref Reference Platform, it is essential to maintain the `flat.qsf`, `base.qsf`, `top.qsf`, and `opencl_bsp_ip.qsf` Intel Quartus Prime Settings Files.

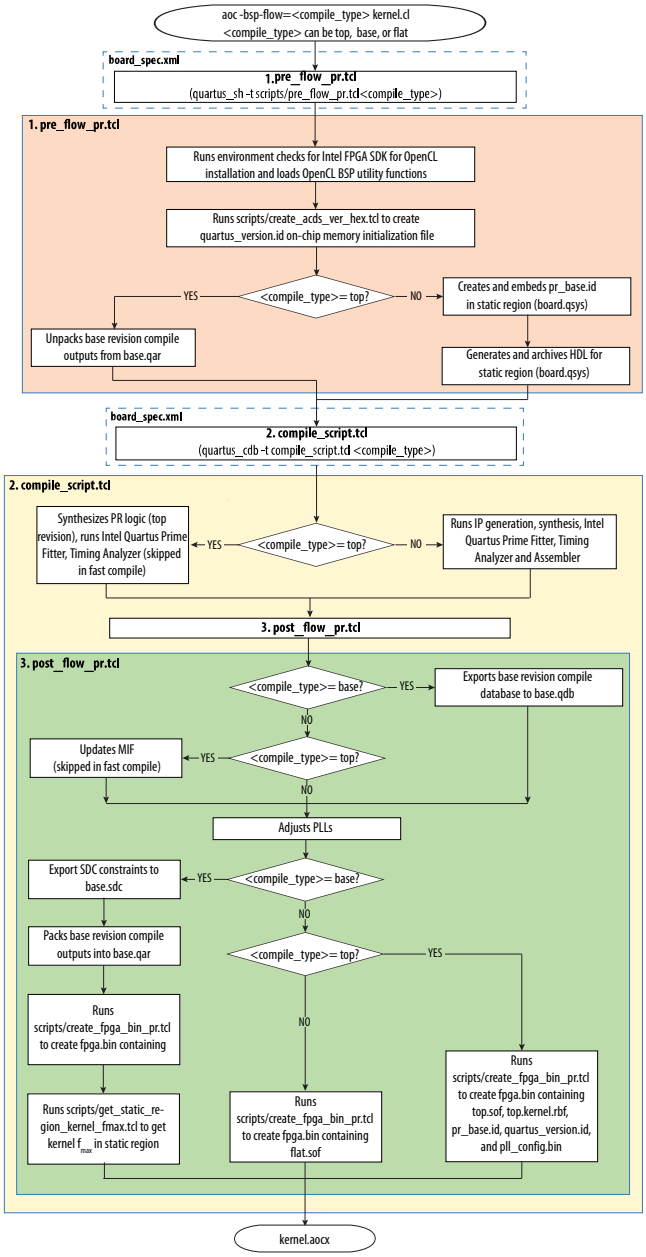
The s10_ref Reference Platform includes two additional partitions: the `Top` partition and the `kernel` partition. The `Top` partition contains all logic, and the `kernel` partition contains only the kernel logic.

2.8. Intel FPGA SDK for OpenCL Compilation Flows

The BSP contains scripts that facilitate all kernel compiles based on the revision, that is, flat, base or top. These scripts are located inside the `hardware/s10gx/scripts` and `hardware/s10gx` directory together.

Figure 12. Complete Flow Diagram of Intel Stratix 10 Reference BSP

The following flowchart explains the sequence of events that takes place after you execute the `aoc` command to compile a kernel:



The image provides an overview of scripts called and respective stages executed during each phase of the compile, depending on which project revision (flat, base or top) is targeted. As soon as you execute the `aoc` command, for example, `aoc -bsp-flow=<flat/base/top> kernel.cl -o kernel.aocx;` the Intel FPGA SDK for OpenCL Offline Compiler looks at the `board_spec.xml` associated with the provided BSP and calls the `pre_flow_pr.tcl` script for the particular project revision. It then goes into some decision stages and runs `compile_script.tcl` and

`post_flow_pr.tcl` scripts underneath for a complete successful compile. For more details about the information provided in the flowchart, view scripts under the `hardware/s10gx` directory inside your board support package.

Related Information

- Platform Designer System Generation on page 42
- QAR/QDB File Generation on page 43
- Intel FPGA SDK for OpenCL Offline Compiler Kernel Compilation Flows

2.8.1. Compile Flow

Following are the types of compile flows:

Flat Compile
A flat revision uses the `flat.qsf` settings file and performs a flat compilation of the entire design (BSP along with kernel generated hardware). The `flat.qsf` has minimal location constraints, and generally has all of the pin assignments (sourced using `device.tcl`) and basic settings to compile a hardware design. To compile a flat revision of your BSP, use `-bsp-flow=flat` modifier option with the `aoc` command.

Base Compile
A base revision uses the `base.qsf` settings file to compile the board support package. The `base.qsf` uses all the `flat.qsf` settings and adds the required location constraints and Logic Lock regions on top of it. The kernel clock target is relaxed during the base compilation so that the BSP hardware has more freedom to meet timing. A `base.qar` database is created to preserve the BSP hardware, which is the static region. The revision can be compiled using `-bsp-flow=base` modifier option with the `aoc` command.

Top Compile
The top flow, also known as the import compile, is generally the default flow of kernel compiles. It uses the `top.qsf` settings file for compilation and the `base.qar` from a base revision compile to import the pre-compiled netlist of the static region. It guarantees the timing closed static region and compiles only the kernel generated hardware. It also increases the kernel clock target to obtain the best kernel maximum operating frequency (f_{max}).

2.8.2. Platform Designer System Generation

The Intel FPGA SDK for OpenCL Offline Compiler generates RTL for the `board.qsys` Platform Designer system and kernel sub-systems in the kernel compilation directory after successfully completing a first-stage compilation.

The *INTELFPGAOCLSDKROOT* environment variable points to the location of the Intel FPGA SDK for OpenCL installation directory.

The `board.qsys` Platform Designer system represents the bulk of the static region. The `pre_flow_pr.tcl` script generates the Platform Designer systems on the fly before the beginning of the Intel Quartus Prime compilation flow in both the flat and base revision compilations.

2.8.3. QAR/QDB File Generation

The `base.qdb` Intel Quartus Prime Compilation Database File contains all the necessary compilation database information for importing a timing-closed and placed-and-routed netlist of the static region.

The *INTELFPGAOCLSDKROOT*`/board/s10_ref/hardware/s10gx/scripts/post_flow_pr.tcl` script creates the `base.qdb` file. The `qar_ip_files.tcl` file invokes the `qar_ip_files proc` command to export the entire base revision compilation database to the `base.qar` file that also contains the `base.qdb`, `pr_base.id` and `base.sdc` files. For your Custom Platform, you do not need to add these files to the board directory (that is, *INTELFPGAOCLSDKROOT*`/board/<`*custom_platform*`>/hardware/<`*board_name*`>`) separately. They are generated when you do a base revision compile with your custom platform.

2.9. Addition of Timing Constraints

A Custom Platform must apply the correct timing constraints to the Intel Quartus Prime project. In the Intel Stratix 10 FPGA Development Kit Reference Platform, the `top.sdc` file contains all timing constraints applicable before IP instantiation in Platform Designer. The `top_post.sdc` file contains timing constraints applicable after Platform Designer generation is run.

The order of the application of time constraints is based on the order of appearance of the `top.sdc` and `top_post.sdc` files in the `flat.qsf` file. To ensure proper SDC ordering, the `opencl_bsp_ip.qsf` file is sourced between `top.sdc` and `top_post.sdc` files. All IPs are added to `opencl_bsp_ip.qsf` during `aoc` compile flow. This ensures that the SDC order is `top.sdc` followed by SDCs for the IP components and then `top_post.sdc` in all `aoc` compiles.

One noteworthy constraint in the `s10_ref` Reference Platform is the multicycle constraint for the kernel reset in the `top_post.sdc` file. Using global routing saves routing resources and provides more balanced skew. However, the delay across the global route might cause recovery timing issues that limit kernel clock speed. Therefore, it is necessary to include a multicycle path on the global reset signal.

```
#Make the kernel reset multicycle
#changes made to the multicycle path here need to also be reflected in the
#multicycle value in scripts/adjust_plls_s10.tcl
Set_multicycle_path -to * -setup 15 -from {freeze_wrapper_inst|
board_kernel_reset_n_reg}
Set_multicycle_path -to * -hold 14 -from {freeze_wrapper_isnt|
board_kernel_reset_reset_n_reg}
```

2.10. Connection of the Intel Reference Platform to the Intel FPGA SDK for OpenCL

A Custom Platform must include a `board_env.xml` file to describe its general contents to the Intel FPGA SDK for OpenCL Offline Compiler.

For each hardware design inside your Custom Platform, your Custom Platform requires a `board_spec.xml` that describes the hardware to the SDK.

The following sections describe the implementation of these files for the Intel Stratix 10 GX FPGA Development Kit Reference Platform.

2.10.1. Describe the Intel Stratix 10 GX FPGA Development Kit Reference Platform to the Intel FPGA SDK for OpenCL

The *INTELFPGAOCLSDKROOT*/board/s10_ref/board_env.xml file describes the Intel Stratix 10 GX FPGA Development Kit Reference Platform to the Intel FPGA SDK for OpenCL.

Details of each field in the board_env.xml file are available in the *Creating the board_env.xml File* section of the *Intel FPGA SDK for OpenCL Custom Platform Toolkit User Guide*.

Related Information

Creating the board_env.xml File

2.10.2. Describe the Intel Stratix 10 GX FPGA Development Kit Reference Platform Hardware to the Intel FPGA SDK for OpenCL

The Intel Stratix 10 GX FPGA Development Kit Reference Platform includes an *INTELFPGAOCLSDKROOT*/board/s10_ref/hardware/s10gx/board_spec.xml file that describes the hardware to the Intel FPGA SDK for OpenCL.

Device

The device section contains the name of the device model file available in the *INTELFPGAOCLSDKROOT*/share/models/dm directory of the SDK and in the board spec.xml file. The used_resources element accounts for all logic outside of the kernel partition. The value of used_resources for alms equals the difference between the total number of adaptive logic modules (ALMs) used in final placement and the total number of ALMs available to the kernel partition. You can derive this value from the Partition Statistic section of the Fitter report after a compilation. Consider the following ALM categories within an example Fitter report:

Figure 13. Fitter Partition Statistics

```
+-----------------------------------------------------------------------------------------------------------------------------------+
; Fitter Partition Statistics                                                                                                      ;
+-----------------------------------------------+---------------------+---------------------+-----------------------------------+
; Statistic                                     ; |                   ; auto_fab_0           ; freeze_wrapper_inst|pr_region_inst ;
+-----------------------------------------------+---------------------+---------------------+-----------------------------------+
; ALMs needed [=A-B+C]                          ; 80273.9 / 933120 ( 8 % ) ; 2538.3 / 933120 ( < 1 % ) ; 25596.4 / 855360 ( 2 % )  ;
```

The value of used_resources equals the total number of ALMs in l minus the total number of ALMs in freeze wrapper inst|pr_region_inst. In the example above, used_resources = 80273 - 25596 = 54677 ALMs.

You can derive used_resources for rams and dsps in the same way using M20Ks and DSP blocks, respectively. The used_resources value for ffs is four times the used_resources value for alms because there are two primary and two secondary logic registers per ALM.

Note: The Fitter Partition Statistics section can be observed in the Fitter report generated during base or top compiles only.

Global Memory

In the `board_spec.xml` file, there is one `global_mem` section for DDR memory. Assign the string `DDR` to the `name` attribute of the `global_mem` element. The **board** instance in Platform Designer provides all of these interfaces. Therefore, the string `board` is specified in the `name` attribute of all the `interface` elements within `global_mem`.

- DDR

  Because DDR memory serves as the default memory for the board that the `s10_ref` Reference Platform targets, its `address` attribute begins at zero. Its `config_addr` is `0x018` to match the `acl_bsp_memorg_host0x018` conduit used to connect to the corresponding OpenCL Memory Bank Divider for DDR.

  Attention: The width and burst sizes must match the parameters in the OpenCL Memory Bank Divider for DDR (`memory_bank_divider_ddr4a`).

Interfaces

The `interfaces` section describes kernel clocks, reset, CRA, and snoop interfaces. The OpenCL Memory Bank Divider for the default memory (in this case, `memory_bank_divider_ddr4a`) exports the snoop interface described in the `interfaces` section. The width of the snoop interface should match the width of the corresponding streaming interface.

2.11. Intel Stratix 10 FPGA Programming Flow

There are two ways to program the Intel Stratix 10 FPGA for the Intel Stratix 10 GX FPGA Development Kit Reference Platform: Flash and `quartus_pgm`

In the order from the longest to the shortest configuration time, the two FPGA programming methods are as follows:

- To maintain the previously programmed state after power cycling, use Flash programming.
- To replace both the FPGA periphery and the core, use the Intel Quartus Prime Programmer command-line executable (`quartus_pgm`) to program the device via cables such as the Intel FPGA Download Cable (formerly USB-Blaster).

For more information about Intel Stratix 10 GX FPGA programming flow and board bring-up, refer to the Intel Stratix 10 Development Kit Initialization guide in the `<S10GX_BSP_Directory>/bringup` directory.

2.12. Implementation of Intel FPGA SDK for OpenCL Utilities

The Intel Stratix 10 GX FPGA Development Kit Reference Platform includes a set of Intel FPGA SDK for OpenCL utilities for managing the FPGA board.

For more information on the implementation requirements of the AOCL utilities, refer to the *Providing Intel FPGA SDK for OpenCL Utilities Support* section of the *Intel FPGA SDK for OpenCL Custom Platform Toolkit User Guide*.

Related Information

Providing Intel FPGA SDK for OpenCL Utilities Support

2.12.1. aocl install

The `install <path_to_customplatform>` utility in the Intel Stratix 10 GX FPGA Development Kit Reference Platform installs the kernel driver on the host computer. Users of the Intel FPGA SDK for OpenCL only need to install the driver once, after which the driver should be automatically loaded each time the machine reboots.

Attention: You must have write privileges to the SDK directory to install the kernel directory.

Windows

The `install.bat` script is in the *<your_custom_platform>*`\windows64\libexec` directory, where *<your_custom_platform>* points to the top-level directory of your Custom Platform. This `install.bat` script triggers the `install` executable to install the SSG Driver on the host machine.

Linux

The `install` script is located in the *<your_custom_platform>*`/linux64/libexec` directory. This `install` script first compiles the kernel module in a temporary location and then performs the necessary setup to enable automatic driver loading after reboot.

2.12.2. aocl uninstall

The `uninstall <path_to_customplatform>` utility in the Intel Stratix 10 GX FPGA Development Kit Reference Platform removes the current host computer drivers used for communicating with the board.

Windows

The `uninstall.bat` script is located in the *<your_custom_platform>*`\windows64\libexec` directory, where *<your_custom_platform>* points to the top-level directory of your Custom Platform. This `uninstall.bat` script triggers the `uninstall` executable to uninstall the SSG Driver on the host machine.

Linux

The `uninstall` script is located in the *<your_custom_platform>*`/linux64/libexec` directory. This `uninstall` script removes the driver module from the linux kernel.

2.12.3. aocl program

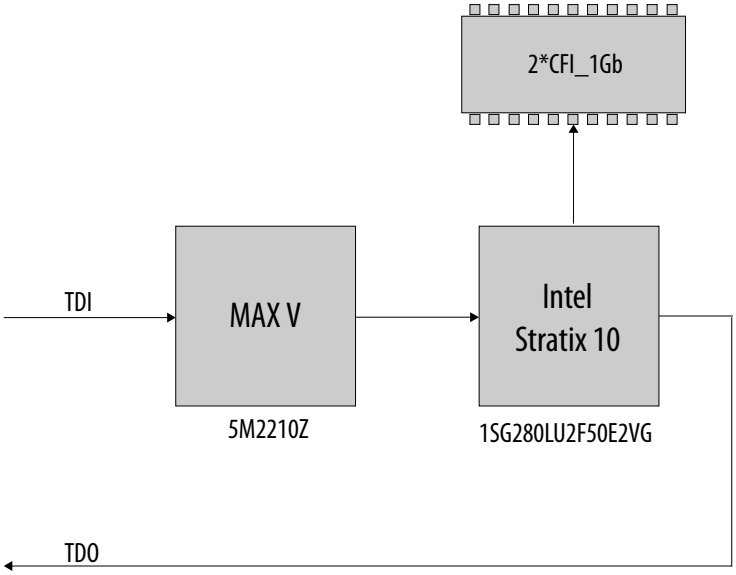
The `program` utility in the Intel Stratix 10 GX FPGA Development Kit Reference Platform programs the board with the specified `.aocx` file. Calling the `aocl_mmd_reprogram()` MMD API function implements the `program` utility.

2.12.4. aocl flash

The `flash` utility in the Intel Stratix 10 GX FPGA Development Kit Reference Platform configures the power-on image for the FPGA using the specified `.aocx` file. Calling into the MMD library implements the `flash` utility.

Figure 14. JTAG Chain with Intel Stratix 10 FPGA, MAX V CPLD, and CFI Flash Memory

This figure illustrates the JTAG chain and the location of the common flash interface (CFI) relative to the MAX® V CPLD on the Intel Stratix 10 GX FPGA Development Kit.



2.12.5. aocl diagnose

The `diagnose` utility in the Intel Stratix 10 GX FPGA Development Kit Reference Platform reports device information and identifies issues. The `diagnose` utility first verifies the installation of the kernel driver. Depending on whether an additional argument is specified in the command, the utility then performs different tasks.

Without an argument, the utility returns the overall information of all the devices installed in a host machine. If a specific device name is provided as an argument (that is, `aocl diagnose` *<device_name>*), the `diagnose` utility runs a memory transfer test and then reports the host-device transfer performance.

You can run the `diagnose` utility for multiple devices (that is, `aocl diagnose` *<device_name1> <device_name2> <device_name3>*). If you want to run the `diagnose` utility for all devices, use the `all` option (that is `aocl diagnose all`).

2.12.6. aocl list-devices

The `list-devices` utility lists all the devices installed in a host machine, grouped by board packages.

The `list-devices` utility is similar to the `diagnose` utility. It first verifies the installation of the kernel driver and then lists all the devices.

2.13. Considerations in Intel Stratix 10 GX FPGA Development Kit Reference Platform Implementation

The implementation of the Intel Stratix 10 GX FPGA Development Kit Reference Platform includes some workarounds that address certain Intel Quartus Prime Pro Edition software known issues.

- The `quartus_syn` executable reads the SDC files. However, it does not support the Tcl command `get_current_revision`. Therefore, in the `top_post.sdc` file, a check is in place to determine whether `quartus_syn` has read the file before checking the current version.

In addition to these workarounds, take into account the following considerations:

- Intel Quartus Prime compilation is only ever performed after the Intel FPGA SDK for OpenCL Offline Compiler embeds an OpenCL kernel inside the system.
- Perform Intel Quartus Prime compilation after you install the Intel FPGA SDK for OpenCL and set the *INTELFPGAOCLSDKROOT* environment variable to point to the SDK installation.
- The name of the directory where the Intel Quartus Prime project resides must match the `name` field in the `board_spec.xml` file within the Custom Platform. The name is case sensitive.
- The *PATH* or *LD_LIBRARY_PATH* environment variable must point to the MMD library in the Custom Platform.

3. Developing Your Intel Stratix 10 Custom Platform

Use the tools available in Intel Stratix 10 GX FPGA Development Kit Reference Platform (s10_ref) and the Intel FPGA SDK for OpenCL Custom Platform Toolkit together to create your own Custom Platform.

Developing your Custom Platform requires in-depth knowledge of the contents in the following documents and tools:

- *Intel FPGA SDK for OpenCL Custom Platform User Guide*
- Contents of the SDK Custom Platform Toolkit
- *Intel FPGA SDK for OpenCL Intel Arria 10 GX FPGA Development Kit Reference Platform Porting Guide*
- Documentation for all the Intel FPGA IP in your Custom Platform
- *Intel FPGA SDK for OpenCL Getting Started Guide*
- *Intel FPGA SDK for OpenCL Programming Guide*

In addition, you must independently verify all IP on your computing card (for example, PCIe controllers and DDR4 external memory).

Related Information

- Intel FPGA SDK for OpenCL Custom Platform Toolkit User Guide
- Intel FPGA SDK for OpenCL Intel Arria 10 GX FPGA Development Kit Reference Platform Porting Guide
- Intel FPGA SDK for OpenCL Intel Arria 10 SoC Development Kit Reference Platform Porting Guide
- Intel FPGA SDK for OpenCL Intel Cyclone V SoC Development Kit Reference Platform Porting Guide
- Intel FPGA SDK for OpenCL Getting Started Guide
- Intel FPGA SDK for OpenCL Programming Guide

3.1. Initializing Your Intel Stratix 10 Custom Platform

To initialize your Intel FPGA SDK for OpenCL Custom Platform, copy the Intel Stratix 10 GX FPGA Development Kit Reference Platform to another directory and rename it.

1. Copy the *INTELFPGAOCLSDKROOT*/board/s10_ref directory, where *INTELFPGAOCLSDKROOT* is the location of the SDK installation.
2. Paste the s10_ref directory into a directory that you own (that is, not a system directory) and then rename it (<*your_custom_platform*>).
3. Choose the s10gx board variant in the <*your_custom_platform*>/hardware directory that matches the production silicon for the Intel Stratix 10 FPGA as the basis of your design.
4. Rename s10gx board variant to match the name of your FPGA board (<*your_custom_platform*>/hardware/<*board_name*>).
5. Modify the <*your_custom_platform*>/board_env.xml file so that the name and default fields match the changes you made in step 2 on page 50 and step 4 on page 50, respectively.
6. Modify the my_board name in the inside <*your_custom_platform*>/hardware/<*board_name*>/board_spec.xml file to match the change you made in step 2 on page 50.

Related Information

- Setting the Intel FPGA SDK for OpenCL User Environment Variables for Windows
- Setting the Intel FPGA SDK for OpenCL User Environment Variables for Linux
- Describe the Intel Stratix 10 GX FPGA Development Kit Reference Platform to the Intel FPGA SDK for OpenCL on page 44

3.2. Modifying the Intel Stratix 10 GX FPGA Development Kit Reference Platform Design

Modify the Intel Quartus Prime design for the Intel Stratix 10 GX FPGA Development Kit Reference Platform to fit your design needs.

You can add a component in Platform Designer and connect it to the existing system, or add a Verilog file to the available system. After adding the custom components, connect those components in Platform Designer.

1. Instantiate your PCIe controller, as described in *Host-to-Intel Stratix 10 Communication over PCIe* section.
2. Instantiate any memory controllers and I/O channels. You can add the board interface hardware either as Platform Designer components in the board.qsys Platform Designer system or as HDL in the top.v file.

   The board.qsys file and the top.v file are in the <*your_custom_platform*>/hardware/<*board_name*> directory.
3. Modify the device.tcl file to match all the correct settings for the device on your board. The device.tcl file is sourced into opencl_bsp_ip.qsf and flat.qsf files.
4. Modify the <*your_custom_platform*>/hardware/<*board_name*>/flat.qsf file to change settings for your system. The base.qsf and top.qsf files will include all settings from the flat.qsf file.

All `.qsf` files are in the *<your_custom_platform>*`/hardware/`*<board_name>* directory. Ensure that the `flat.qsf` file does not have any `IP_FILE` assignments after the assignment that adds `top_post.sdc` to the project since this changes the order in which SDC files are read during compile. Refer to Addition of Timing Constraints on page 43 for more information about SDC ordering.

Related Information

Host-to-Intel Stratix 10 FPGA Communication over PCIe on page 16

3.3. Integrating Your Intel Stratix 10 Custom Platform with the Intel FPGA SDK for OpenCL

When modifying the Intel Stratix 10 GX FPGA Development Kit Reference Platform into your own Custom Platform, ensure that guaranteed timing closure holds true for your Custom Platform. Follow the steps below to ensure your design meets timing, to check your BSP with multiple kernels and to implement top revision compile flow.

1. Set *AOCL_BOARD_PACKAGE_ROOT* to point to your custom platform. Use `flat.qsf` file in *INTELFPGAOCLSDKROOT*`/board/s10_ref` reference platform to determine the type of information you must include in the `flat.qsf` file for your Custom Platform.
2. Update the *<your_custom_platform>*`/hardware/`*<board_name>*`/board_spec.xml` file. Ensure that there is at least one global memory interface, and all global memory interfaces correspond to the exported interfaces from the `board.qsys` Platform Designer system file.
3. After all your hardware design changes are finalized, compile flat revision with several seeds of the *INTELFPGAOCLSDKROOT*`/board/custom_platform_toolkit/tests/boardtest/boardtest.cl` kernel until you generate a design that closes timing cleanly.

   To specify the seed number during compile, include the *-seed=<N>* option in your `aoc` command. Use *-bsp-flow=flat* option in your `aoc` command for flat compile.

   ```
   aoc -bsp-flow=flat boardtest.cl -o=bin/boardtest.aocx
   ```

4. Based on the output of your flat revision compiles, establish the floorplan of your design in base revision. Add Logic Lock regions in `base.qsf` only. `base.qsf` and `top.qsf` files automatically inherit all the settings in the `flat.qsf` file.

   Important: Consider all design criteria outlined in the Intel Stratix 10 FPGA System Design section in this guide. Compile base revision with several seeds of the *INTELFPGAOCLSDKROOT*`/board/custom_platform_toolkit/tests/boardtest/boardtest.cl` kernel until you generate a design that closes timing cleanly. This flow is used to create the timing closed base database for the static region which is needed for guaranteed timing support. Use *-bsp-flow=base* option in your `aoc` command for base compile.

   ```
   aoc -bsp-flow=base boardtest.cl -o=bin/boardtest.aocx
   ```

   Attention: In a typical development of a custom platform, designers generally validate the functionality of boardtest using the flat flow before starting to add guaranteed timing functionality to their BSP.

5. From the compiled output directory for base revision compile, copy `base.qar` file into your Custom Platform hardware directory to replace old `base.qar` with new `base.qar` containing post-fit netlist for your reference platform.
6. Make sure *AOCL_BOARD_PACKAGE_ROOT* is set to your reference platform with new `base.qar` and compile top revision, that is, default compilation flow. Confirm that you can use the `.aocx` file to reprogram the FPGA by invoking the `aocl program acl0 boardtest.aocx` command.
7. Using the default compilation flow, test your `base.qar` file across several OpenCL design examples and confirm that the following criteria are satisfied:
   - All compilations close timing.
   - The OpenCL design examples achieve satisfactory f_{max} (Check the `acl quartus_report.txt` for achieved f_{max}).

Related Information

- Describe the Intel Stratix 10 GX FPGA Development Kit Reference Platform Hardware to the Intel FPGA SDK for OpenCL on page 44
- Intel Stratix 10 FPGA System Design on page 31
- Provide a Timing-Closed Post-Fit Netlist on page 38

3.4. Setting up the Intel Stratix 10 Custom Platform Software Development Environment

Prior to building the software layer for your Intel FPGA SDK for OpenCL Custom Platform, set up the software development environment.

- To compile the MMD layer for Windows, perform the following tasks:
  a. Install the GNU `make` utility on your development machine.
  b. Install a version of Microsoft Visual Studio that has the ability to compile 64-bit software (for example, Microsoft Visual Studio version 2010 Professional).
  c. Set the environment for using Microsoft Visual Studio.
  d. Set the development environment so that SDK users can invoke commands and utilities at the command prompt.
  e. Modify the <*your_custom_platform_name*>`/source/Makefile.common` file so that *TOP_DEST_DIR* points to the top-level directory of your Custom Platform.
  f. To check that you have set up the software development environment properly, invoke the `make` or `make clean` command.
- To compile the MMD layer for Linux, perform the following tasks:
  a. Ensure that you use a Linux distribution that Intel supports (for example, GNU Compiler Collection (GCC) version 4.47).
  b. Modify the <*your_custom_platform*>`/source/Makefile.common` file so that *TOP_DEST_DIR* points to the top-level directory of your Custom Platform.
- To check that you have set up the software environment properly, invoke the `make` or `make clean` command.

3.5. Establishing Intel Stratix 10 Custom Platform Host Communication

After modifying and rebranding the Intel Stratix 10 GX FPGA Development Kit Reference Platform to your own Custom Platform, use the tools and utilities in your Custom Platform to establish communication between your FPGA accelerator board and your host application.

1. Program your FPGA device with the `<your_custom_platform>/hardware/<board_name>/base.sof` file and then reboot your system.

   The `base.sof` file is generated during base revision compile when integrating your Custom Platform with the Intel FPGA SDK for OpenCL. Refer to the *Integrating Your Intel Stratix 10 Custom Platform with the Intel FPGA SDK for OpenCL* section for more information.
2. Confirm that your operating system recognizes a PCIe device with your vendor and device IDs.
   - For Windows, open the **Device Manager** and verify that the correct device and IDs appear in the listed information.
   - For Linux, invoke the `lspci` command and verify that the correct device and IDs appear in the listed information.
3. Set the environment variable *AOCL_BOARD_PACKAGE_ROOT* to point to your custom platform.
4. Run the `aocl install <path_to_customplatform>` utility command to install the kernel driver on your machine.
5. For Windows, set the *PATH* environment variable. For Linux, set the *LD_LIBRARY_PATH* environment variable.

   For more information about the settings for *PATH* and *LD_LIBRARY_PATH*, refer to *Setting the Intel FPGA SDK for OpenCL User Environment Variables* in the *Intel FPGA SDK for OpenCL Getting Started Guide*.
6. Modify the `version_id_test` function in your `<your_custom_platform>/source/host/mmd/acl_pcie_device.cpp` MMD source code file to exit after reading from the `version ID` register. Rebuild the MMD software.
7. Run the `aocl diagnose` utility command and confirm that the `version ID` register reads back the ID successfully. You may set the environment variables *ACL_HAL_DEBUG* and *ACL_PCIE_DEBUG* to a value of 1 to visualize the result of the diagnostic test on your terminal.

Related Information

- Integrating Your Intel Stratix 10 Custom Platform with the Intel FPGA SDK for OpenCL on page 51
- Setting the Intel FPGA SDK for OpenCL Environment Variables for Linux
- Setting the Intel FPGA SDK for OpenCL User Environment Variables for Windows

3.6. Branding Your Intel Stratix 10 Custom Platform

Modify the library, driver, and source files in the Intel Stratix 10 GX FPGA Development Kit Reference Platform to reference your Intel FPGA SDK for OpenCL Custom Platform.

1. In the software development environment available with the s10_ref Reference Platform, replace all references of "s10_ref" with the name of your Custom Platform.
2. Modify the PACKAGE_NAME and MMD_LIB_NAME fields in the <*your_custom_platform*>/source/Makefile.common file.
3. Modify the name, linklib, and mmlibs elements in <*your_custom_platform*>/board_env.xml file to your custom MMD library name.
4. In your Custom Platform, modify the following lines of code in the hw_pcie_constants.h file to include information of your Custom Platform:

   ```
   #define ACL_BOARD_PKG_NAME "s10_ref"
   #define ACL_VENDOR_NAME "Intel Corporation"
   #define ACL_BOARD_NAME "Stratix 10 Reference Platform"
   ```

   For Windows, the hw_pcie_constants.h file is in the <*your_custom_platform*>\source\include folder. For Linux, the hw_pcie_constants.h file is in the <*your_custom_platform*>/linux64/driver directory.

   Note: The *ACL_BOARD_PKG_NAME* variable setting must match the name attribute of the board_env element that you specified in the board_env.xml file.
5. Define the Device ID, Subsystem Vendor ID, Subsystem Device ID, and Revision ID, as defined in the *Device Identification Registers for Intel Stratix 10 PCIe Hard IP* section.

   Note: The PCIe IDs in the hw_pcie_constants.h file must match the parameters in the PCIe controller hardware.
6. Update your Custom Platform's board.qsys Platform Designer system and the hw_pcie_constants.h file with the IDs defined in step 5 on page 54.
7. For Windows, update the <*your_custom_platform*>\windows64\driver\Shim.inf file.
8. Run make in the<*your_custom_platform*>/source directory to generate the driver.

Related Information

Device Identification Registers for Intel Stratix 10 PCIe Hard IP on page 17

3.7. Changing the Device Part Number

When porting the Intel Stratix 10 GX FPGA Development Kit Reference Platform to your own board, change the device part number, where applicable, to the part number of the device on your board.

Update the device part number in the following files within the <*your_custom_platform*>/hardware/<*board_name*> directory:

- In the `device.tcl` file, change the device part number in the `set global assignment -name DEVICE 1SG280LU2F50E2VG` QSF assignment. The updated device number will appear in the `base.qsf`, `top.qsf`, `flat.qsf` and `opencl_bsp_ip.qsf` files.
- In the `board.qsys` and `mem.qsys` files, change all occurrences of `1SG280LU2F50E2VG`.

3.8. Connecting the Memory in the Intel Stratix 10 Custom Platform

Calibrate the external memory IP and controllers in your Custom Platform, and connect them to the host.

1. In your Custom Platform, instantiate your external memory IP based on the information in the *DDR4 as Global Memory for OpenCL Applications* section. Update the information pertaining to the `global_mem` element in the <*your_custom_platform*>/hardware/<*board_name*>/`board_spec.xml` file.
2. Remove the `boardtest` hardware configuration file (that is, `aocx`) that you created during the integration of your Custom Platform with the Intel FPGA SDK for OpenCL.
3. Recompile the *INTELFPGAOCLSDKROOT*/board/custom_platform_toolkit/tests/boardtest/`boardtest.cl` kernel source file.

   The environment variable *INTELFPGAOCLSDKROOT* points to the location of the SDK installation.
4. Reprogram the FPGA with the new `boardtest` hardware configuration file and then reboot your machine.
5. Modify the `wait_for_uniphy` function in the `acl_pcie_device.cpp` MMD source code file to exit after checking the UniPHY status register. Rebuild the MMD software.

   For Windows/Linux, the `acl_pcie_device.cpp` file is in the <*your_custom_platform*>\source\host\mmd folder.
6. Run the `aocl diagnose` SDK utility and confirm that the host reads back both the version ID and the value 0 from the uniphy_status component. The utility should return the message `Uniphy are calibrated`.
7. Consider analyzing your design in the Signal Tap logic analyzer to confirm the successful calibration of all memory controllers.

   Note: For more information on Signal Tap logic analyzer, download the Signal Tap II Logic Analyzer tutorial from the University Program Tutorial page.

Related Information

- DDR4 as Global Memory for OpenCL Applications on page 27
- Integrating Your Intel Stratix 10 Custom Platform with the Intel FPGA SDK for OpenCL on page 51
- Signal Tap II with Verilog Designs

3.9. Modifying the Kernel PLL Reference Clock

The Intel Stratix 10 GX FPGA Reference Platform uses an external 50 MHz clock as a reference for the I/O PLL. The I/O PLL relies on this reference clock to generate the internal `kernel_clk` clock, and the `kernel_clk2x` clock that runs at twice the frequency of `kernel_clk`.

When porting the `s10_ref` Reference Platform to your own board using a different reference clock, update the `board.qsys` and `top.sdc` files with the new reference clock speed.

1. In the <*your_custom_platform*>/hardware/<*board_name*>/`board.qsys` file, update the REF_CLK_RATE parameter value on the kernel_clk_gen IP module.
2. In the <*your_custom_platform*>/hardware/<*board_name*>/`top.sdc` file, update the `create_clock` assignment for `config_clk` in `top.sdc`.
3. In the <*your_custom_platform*>/hardware/<*board_name*>/`top.v` file, update the comment for the `config_clk` input port, which is connected to `kernel_pll_refclk` in `board.qsys`.

After you update the `board.qsys` and the `top.sdc` files, the `post_flow_pr.tcl` script automatically determines the I/O PLL reference frequency and computes the correct PLL settings for your kernel.

3.10. Integrating an OpenCL Kernel in Your Intel Stratix 10 Custom Platform

After you establish host communication and connect the external memory, test the FPGA programming process from kernel creation to program execution.

1. Perform the steps outlined in *INTELFPGAOCLSDKROOT*/`board/custom_platform_toolkit/tests/README.txt` file to build the hardware configuration file from the *INTELFPGAOCLSDKROOT*/`board/custom_platform_toolkit/tests/boardtest/boardtest.cl` kernel source file.

   The environment variable *INTELFPGAOCLSDKROOT* points to the location of the Intel FPGA SDK for OpenCL installation.
2. Program your FPGA device with the hardware configuration file you created in step 1 on page 56 and then reboot your machine.
3. Remove the early-exit modification in the `version_id_test` function in the `acl_pcie_device.cpp` file that you implemented when you established communication between the board and the host interface.

   For Windows/Linux, the `acl_pcie_device.cpp` file is in the <*your_custom_platform*>`\source\host\mmd` folder.
4. Recompile the MMD.
5. Invoke the `aocl diagnose` <*device_name*> command, where <*device_name*> is the string you define in your Custom Platform to identify each board.

In case you have only one variant, invoke the `aocl diagnose acl0` command.

6. Build the `boardtest` host application using the `.sln` file (Windows) or `Makefile.linux` (Linux) in the SDK's Custom Platform Toolkit.

   For Windows, the `.sln` file for Windows is in the *INTELFPGAOCLSDKROOT*`\board\custom_platform_toolkit\tests\boardtest\host` folder. For Linux, the `Makefile.linux` is in the *INTELFPGAOCLSDKROOT*`/board/custom_platform_toolkit/tests/boardtest/host` directory.

7. Set the environment variable *CL_CONTEXT_COMPILER_MODE_INTELFPGA* to a value of 3 and run the `boardtest` host application. The `boardtest` evaluates host to memory and kernel to memory connections. You may have to make modifications to `boardtest` host code base on your hardware design changes.

8. Using the default compilation flow, test your custom platform file across several OpenCL design examples and confirm that the OpenCL design examples function correctly on the accelerator board.

   For more information about *CL_CONTEXT_COMPILER_MODE_INTELFPGA*, refer to *Troubleshooting Intel Stratix 10 GX FPGA Development Kit Reference Platform Porting Issues*.

Related Information

- Establishing Intel Stratix 10 Custom Platform Host Communication on page 53
- Troubleshooting Intel Stratix 10 GX FPGA Development Kit Reference Platform Porting Issues on page 57

3.11. Troubleshooting Intel Stratix 10 GX FPGA Development Kit Reference Platform Porting Issues

Set Intel FPGA SDK for OpenCL-specific environment variables to help diagnose Custom Platform design problems.

Table 10. Intel FPGA SDK for OpenCL-Specific Environment Variables for Identifying Custom Platform Design Problems

Environment Variable	Description
ACL_HAL_DEBUG	Set this variable to a value of 1 to 5 to enable increasing debug output from the Hardware Abstraction Layer (HAL), which interfaces directly with the MMD layer.
ACL_PCIE_DEBUG	Set this variable to a value of 1 to 10000 to enable increasing debug output from the MMD. This variable setting is useful for confirming that the `version ID` register was read correctly and the UniPHY IP cores are calibrated.
ACL_PCIE_JTAG_CABLE	Set this variable to override the default `quartus_pgm` argument that specifies the cable number. The default is cable 1. If there are multiple Intel FPGA Download Cables, you can specify a particular one here.
ACL_PCIE_JTAG_DEVICE_INDEX	Set this variable to override the default `quartus_pgm` argument that specifies the FPGA device index. By default, this variable has a value of 2. If the FPGA is not the first device in the JTAG chain, you can customize the value.

continued...

Environment Variable	Description
ACL_PCIE_USE_JTAG_PROGRAMMING	Set this variable to force the MMD to reprogram the FPGA using the JTAG cable.
ACL_PCIE_DMA_USE_MSI	Set this variable if you want to use MSI for DMA transfers on Windows.
CL_CONTEXT_COMPILER_MODE_INTELFPGA	Unset this variable or set it to a value of 3. The OpenCL host runtime reprograms the FPGA as needed, which it does at least once during initialization. To prevent the host application from programming the FPGA, set this variable to a value of 3.

4. Document Revision History

Table 11. Document Revision History of the Intel FPGA SDK for OpenCL Intel Stratix 10 GX FPGA Development Kit Reference Platform Porting Guide

Document Version	Intel Quartus Prime Version	Changes
2019.04.01	19.1	• Intel Stratix 10 GX FPGA Development Kit Reference Platform Design Architecture on page 14 and Developing Your Intel Stratix 10 Custom Platform on page 49 chapters interchanged. • Few topics were rearranged in this guide. • Changed all occurrences of `s10gx_ea_htile` to `s10gx`. • Added the following new topics: — Introduction to Intel Reference Platform on page 4 — Board Support Package Software Layer on page 20 — Partial Reconfiguration on page 30 — Other Components in the Reference Design on page 31 — Compile Flow on page 42 — Constant Address Bridge (constant_address_bridge) on page 31 • Removed following topics: — *Intel Quartus Prime Compilation Flow and Scripts* — *Intel Quartus Prime Compilation Flow for Board Developers* — *Intel Quartus Prime Compilation Flow for Custom Platform Users* — *Intel Arria 10 FPGA Development Kit Reference Platform Scripts* and its content was merged with Contents of the Intel Stratix 10 GX FPGA Development Kit Reference Platform on page 10. — *Guaranteeing Timing Closure in the Custom Platform* — *Generating the base.qar Post-Fit Netlist for Your Custom Platform* — *Define the Contents of the fpga.bin File for the Intel Stratix 10 GX FPGA Development Kit Reference Platform* • In Intel Stratix 10 GX FPGA Development Kit Reference Platform: Prerequisites on page 7, following changes were made: — `altera_s10pciedk` changed to `s10_ref` — Removed the statement about testing interfaces together in the same design. • In Features of the Intel Stratix 10 GX FPGA Development Kit Reference Platform on page 9, following changes were made: — For OpenCL Host, PCIe Gen2x8 hard IP core updated to PCIe Gen3x8 hard IP core. — Updated the OpenCL Global Memory description. — Updated Figure 4 on page 9 to show the HiLo Connector. • In Intel Stratix 10 GX FPGA Development Kit Reference Platform Board Variants on page 10, removed reference to RevA silicon and -3 grade speed.

continued...

Document Version	Intel Quartus Prime Version	Changes
		• In Contents of the Intel Stratix 10 GX FPGA Development Kit Reference Platform on page 10, following changes were made: — In Table 2 on page 10, `bringup` and `scripts` directories were added along with their descriptions, and `source_windows64` was changed to `source`. — In Table 3 on page 11 • Rearranged the file list. • Removed `max5_116.pof`, `top_synth.qsf`, `root_partition.qdb` and `scripts/kernel_system_update.tcl`. • Added `ip/pr_region.v`, `ip/temperature/<file_name>`, `scripts/regenerate_cache.tcl`, `scripts/base_write_sdc.tcl`, `scripts/create_acds_ver_hex.tcl`, `adjust_plls_s10.tcl`, `hw_iface.ipx`, `sw_iface.ipx`, and `board_spec.xml`. • Updated the descriptions of `mem.qsys`, `scripts/create_fpga_bin_pr.tcl`, `scripts/post_flow_pr.tcl`, `opencl_bsp_ip.qsf`, `flat.qsf`, `device.tcl`, `ip/mem/<file_name>`, `scripts/pre_flow_pr.tcl`, `base.qar` and `base.qsf` descriptions updated. • Changed `acl_ddr4_s10.qsys` to `mem.qsys` • Changed `ip/acl_ddr4_s10/` to `ip/mem/` • Changed `import_compile.tcl` to `compile_script` • Changed `scripts/adjust_plls_s10.tcl` to `scripts/get_static_region_kernel_fmax.tcl` and updated its description. • Updated the description of `base.qar` `scripts/qar_ip_files.tcl`. • In Intel Stratix 10 GX FPGA Development Kit Reference Platform Design Architecture on page 14, added an image to illustrate the system design of s10_ref platform hardware project. • Updated the description in Host-to-Intel Stratix 10 FPGA Communication over PCIe on page 16. • In Table 4 on page 16, following changes were made: — For Application interface type parameter, mentioned about Enable Avalon-MM DMA. — For Hard IP mode, changed to Gen3x8 and 250 MHz, and Gen3 lane rate. Removed the note. — Removed Rx Buffer credit allocation. — For Instantiate Internal Descriptor Controller, removed the statement discussing `ip/host_channel` sub-directory. — In Base Address Registers (BARs), changed 256 KBytes - 18 bits to 18 bits (256 KBytes). • In Device Identification Registers for Intel Stratix 10 PCIe Hard IP on page 17, following changes were made: — `vendor_id_hwtcl` changed to `Vendor ID` — `device_id_hwtcl` changed to `Device ID` — — to changed `Revision ID` — — to changed `Class Code` — `subsystem_vendor_id_hwtcl` changed to `Subsystem Vendor ID` — `subsystem_device_id_hwtcl` changed to `Subsystem Device ID` — For Device ID, changed the Device ID value to `0x5170` — Changed the Revision ID path for Windows.

continued...

Document Version	Intel Quartus Prime Version	Changes
		• In Instantiation of the version_id Component on page 19, updated the version ID of s10_ref reference platform. • In Common Hardware Constants in Software Headers Files on page 20, following changes were made: — Modified the topic title. — Added an important note about matching MMD offsets in the address map. — For `hw_pcie_constants.h` header file, made minor update to the description. — Added an image depicting Address Map and a snapshot of `hw_pcie_constants.h` file. • In PCIe Kernel Driver on page 22, following changes were made: — Modified the topic title. — Changed `path_to_sl0pciedk` to `path_to_sl0_ref` — Removed Jungo driver related information. • In Host-to-Device MMD Software Implementation on page 23, replaced *<your_custom_platform>* with *INTELFPGAOCLSDKROOT*`/board/s10_ref` and removed references to *<your_custom_platform>*. • In Implementing a DMA Transfer on page 25, `aclpci_dma` changed to `aclpci_dma.c` and removed Jungo driver related information. • In Message Signaled Interrupt on page 25, changed `s10gx_ea_htile` to `s10gx` and `pcie_irq_0` to `pcie_irq`. • In Instantiation of board_cade_id_0 Component – JTAG Cable Autodetect Feature on page 27, following changes were made: — Modified the topic title. — Description was updated. — `cade_id` was changed to `board_cade_id_0` — `board_in_system_sources_probes_cade_id` changed to `board_in_system_sources_probes_0` • In DDR4 as Global Memory for OpenCL Applications on page 27, following changes were made: — `acl_ddr4_s10.qsys` changed to `mem.qsys` — Updated the description by removing outdated statements. • In DDR4 IP Instantiation on page 28, updated the Table 9 on page 28 and added an important note about limitations. • In DDR4 Connection to PCIe Host on page 28, following changes were made: — Mentioned about `memory_band_divider_ddr4a` component in `board.qsys` and added detailed description about it. — Removed paragraph about Avalon master interfaces. — Added two more bullet items to the Important note about designing multi-bank OpenCL BSP and Address Span Extender. • In DDR4 Connection to the OpenCL Kernel on page 29, added information about hyper-optimized bridges and important note about the same. • In Host Connection to OpenCL Kernels on page 30, updated the OpenCL Kernel Interface component name and mentioned about `pcie_irq`. • In Clocks on page 31, updated the PCIe clock and DDR4 clock values, and added a note about `kernel_clk_gen`. • In Floorplan on page 33, following changes were made: — Changed all occurrences of Logic Lock Plus to Logic Lock. — Updated step 1 about compiling the design. — Floorplan image updated. — Updated most parts of the description. — Added an image to depict the Logic Lock regions in `base.qsf` file.

continued...

Document Version	Intel Quartus Prime Version	Changes
		• In Pipelining on page 36, changed `pipe_stage_ddr4a_dimm_*` to `pipe_stage` and `pipe_stage_ddr4a_dimm` to `kernel_ddra_bridge` • In DDR4 Calibration on page 37, added information about two components in the `mem.qsys`. • Made minor update to the description in Guaranteed Timing Closure of the Intel Stratix 10 GX FPGA Development Kit Reference Platform Design on page 37. • In Supply the Kernel Clock on page 38, changed `pll_refclk` to `kernel_pll_refclk` and value of `KERNEL_TARGET_CLOCK_RATE` to 500. • In Guarantee Kernel Clock Timing on page 38, changed `import_compile.tcl` to `compile_script.tcl` and `post_flow.tcl` to `post_flow_pr.tcl` • In Provide a Timing-Closed Post-Fit Netlist on page 38, following changes were made: — Updated the image of Structure of the Hierarchy on the Intel Stratix 10 device. — Changed `top_synth.qsf` to `opencl_bsp_ip.qsf` — Changed `root_partition.qdb` to `base.qdb` • In Platform Designer System Generation on page 42, following changes were made: — Removed the references to `kernel_system.qsys` and custom platform directory path. — Updated the description to include a reference to kernel subsystems and compilation directory. — Replaced base and import with flat and base revision compilations. • In QAR/QDB File Generation on page 43, following changes were made: — Replaced `.tcl` with `qar_ip_files.tcl`. — Replaced `export_design` with `qar_ip_files proc` — Included `base.sdc` file. — Replaced `root_partition.qdb` with `base.qdb` — Added the last statement of the paragraph. • In Addition of Timing Constraints on page 43, added code snippets and included information about SDC ordering. • In Connection of the Intel Reference Platform to the Intel FPGA SDK for OpenCL on page 43, modified the title and slightly updated the description. • In Describe the Intel Stratix 10 GX FPGA Development Kit Reference Platform to the Intel FPGA SDK for OpenCL on page 44, removed the paragraph discussing Windows dynamic link libraries (DLLs). • In Describe the Intel Stratix 10 GX FPGA Development Kit Reference Platform Hardware to the Intel FPGA SDK for OpenCL on page 44, following changes were made: — Fitter report updated. — Changed `freeze wrapper inst|kernel_system_inst` to `freeze wrapper inst|pr_region_inst`. — Updated the `used_resources` values to match the fitter report. — Replaced `memorg` with `acl_bsp_memorg_host0x018`. — Replaced `memory_bank_divider` with `memory_bank_divider_ddr4a` • In Intel Stratix 10 FPGA Programming Flow on page 45, rewrote bullet 1 statement and added a note about bring-up directory. • In aocl install on page 46 and aocl uninstall on page 46, removed Jungo driver related information and replaced WinDriver with SSG driver. • In aocl flash on page 46, updated the part number in the image.

continued...

Document Version	Intel Quartus Prime Version	Changes
		• In Initializing Your Intel Stratix 10 Custom Platform on page 49, in step 1, replaced drop directory with the actual path and last step and mentioned about *AOCL_BOARD_PACKAGE_ROOT*. • In Modifying the Intel Stratix 10 GX FPGA Development Kit Reference Platform Design on page 50, modified the last step and removed `top_synth.qsf` and added more information about SDC ordering. • In Integrating Your Intel Stratix 10 Custom Platform with the Intel FPGA SDK for OpenCL on page 51, all steps were rewritten completely. • In Setting up the Intel Stratix 10 Custom Platform Software Development Environment on page 52, I made the following changes in MMD Layer for Windows instructions: — Added a new step 3 for Microsoft Visual Studio. — In step 6, changed `gmake` and `gmake clean` to `make` and `make clean`. — Removed Jungo driver related information. • In Establishing Intel Stratix 10 Custom Platform Host Communication on page 53, improved step1 information and added a new step about setting up *AOCL_BOARD_PACKAGE_ROOT* variable. Mentioned about MMD library in step 6. • In Branding Your Intel Stratix 10 Custom Platform on page 53, replaced `source_windows64` with `source` and modified the step 7. • In Changing the Device Part Number on page 54, following changes were made: — Updated the device part number. — Added the `flat.qsf` to the list. — Changed `top_synth.qsf` to `opencl_bsp_ip.qsf` — Changed `acl_ddr4_s10.qsys` to `mem.qsys` • In Modifying the Kernel PLL Reference Clock on page 56, in step 2, changed "`kernel_pll_refclk`" to "`config_clk` in `top.sdc`" and added more information to optional step 3. • In Integrating an OpenCL Kernel in Your Intel Stratix 10 Custom Platform on page 56, updated step 5 and in step 6, changed `Makefile` to `Makefile.linux` and updated its file path. In step 7, added additional information and step 8 was newly added.
2017.11.21	17.1	Initial release.