



Intel® FPGA SDK for OpenCL™

Intel® Arria® 10 SoC Development Kit Reference Platform Porting Guide

Updated for Intel® Quartus® Prime Design Suite: **18.0**



Subscribe

Send Feedback

UG-20052 | 2018.09.17

Latest document on the web: [PDF](#) | [HTML](#)



Contents

1. Intel® FPGA SDK for OpenCL™ Intel® Arria® 10 SoC Development Kit Reference Platform Porting Guide.....	3
1.1. Intel Arria 10 SoC Development Kit Reference Platform: Prerequisites.....	3
1.2. Features of the Intel Arria 10 SoC Development Kit Reference Platform.....	4
1.3. Intel Arria 10 SoC Development Kit Reference Platform Board Variants.....	6
1.4. Contents of the Intel Arria 10 SoC Development Kit Reference Platform.....	7
1.5. Changes in Intel Arria 10 SoC Development Kit Reference Platform from 17.0 to 17.1.....	9
1.6. Changes in Intel Arria 10 SoC Development Kit Reference Platform from 17.1.2 to 18.0.....	10
2. Developing an Intel Arria 10 SoC Custom Platform.....	12
2.1. Initializing an Intel Arria 10 SoC Custom Platform.....	12
2.2. Modifying Your Intel Arria 10 SoC Custom Platform.....	13
2.3. Integrating Your Intel Arria 10 SoC Custom Platform with the Intel FPGA SDK for OpenCL.....	14
2.4. Changing the Device Part Number.....	14
2.5. Modifying the Kernel PLL Reference Clock.....	15
2.6. Modifying the Hard Processor System.....	15
2.7. Guaranteeing Timing Closure in the Intel Arria 10 SoC Custom Platform.....	16
2.8. Generating the base.qar Post-Fit Netlist for Your Intel Arria 10 SoC FPGA Custom Platform.....	16
3. Building the Software and SD Card Image for the Intel Arria 10 SoC Custom Platform.....	18
3.1. Compiling the Device Tree Blob.....	18
3.2. Recompiling the Linux Kernel for the Intel Arria 10 SoC Development Kit.....	19
3.3. Compiling and Installing the OpenCL Linux Kernel Driver.....	20
3.4. Generating Full-Chip Programming File for SD Card Image.....	21
3.5. Building the SD Card Image.....	21
3.5.1. Layout of the OpenCL Micro SD Card.....	21
3.5.2. Guidelines on Imaging the Micro SD Card.....	23
3.6. Known Issues.....	23
4. Document Revision History.....	25



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

The *Intel® Arria® 10 SoC Development Kit Reference Platform Porting Guide* describes the procedures and design considerations for modifying the Intel Arria 10 SoC Development Kit Reference Platform (a10soc) into your own Custom Platform for use with the Intel FPGA SDK for OpenCL™ (1) (2) SDK.

1.1. Intel Arria 10 SoC Development Kit Reference Platform: Prerequisites

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

Prerequisites for the a10soc Reference Platform:

- An Intel Arria 10 SoC-based accelerator card with working memory interfaces
Test these interfaces together in the same design using the same version of the Intel Quartus® Prime Pro Edition software that you will use to develop your Custom Platform.
- Intel Quartus Prime Pro Edition software version 17.1
- Designing with Logic Lock Plus regions
- Intel SoC Embedded Design Suite version 17.1

General knowledge prerequisites:

- FPGA architecture, including clocking, global routing, and I/Os
- High-speed design
- Timing analysis
- Platform Designer design, and Avalon® and AXI interfaces
- Tcl scripting
- Hard processor systems (HPS)
- DDR4 external memory
- Embedded Linux development

(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.



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 FPGA SDK for OpenCL Intel Arria 10 GX FPGA Development Kit Reference Platform Porting Guide*
- *Intel FPGA SDK for OpenCL Cyclone V SoC Development Kit Reference Platform Porting Guide*

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 Cyclone V SoC Development Kit Reference Platform Porting Guide](#)
- [Intel FPGA SDK for OpenCL Intel Stratix V Network Reference Platform Porting Guide](#)

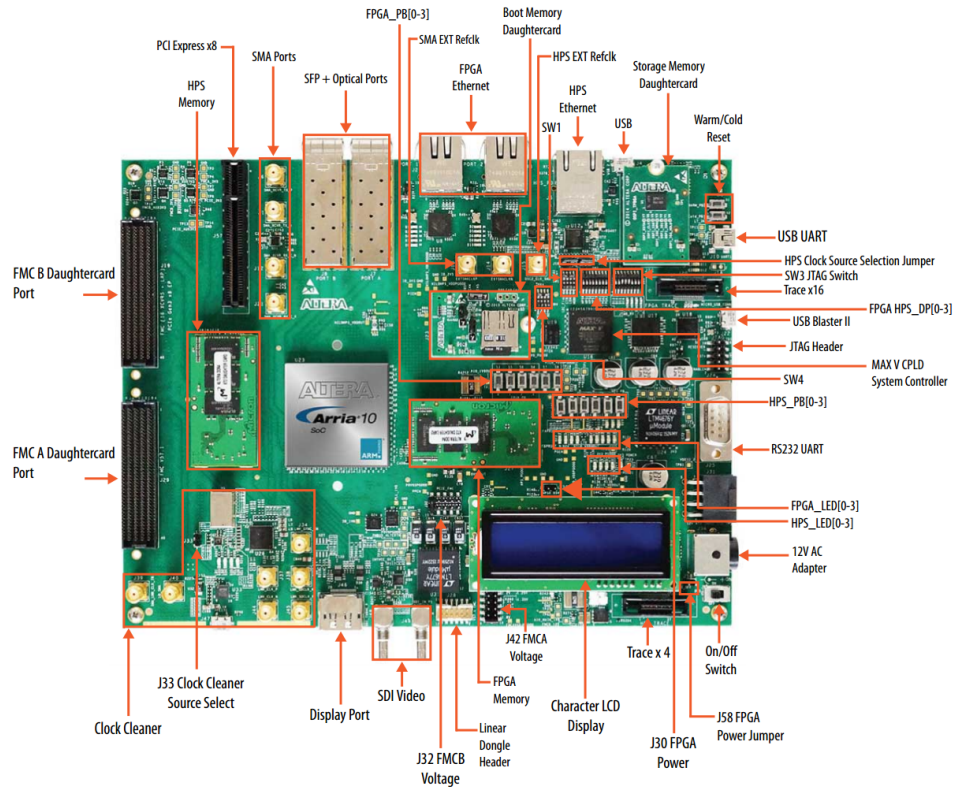
1.2. Features of the Intel Arria 10 SoC Development Kit Reference Platform

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 Arria 10 SoC Development Kit Reference Platform targets a subset of the hardware features available in the Intel Arria 10 SoC Development Kit.



Figure 1. Hardware Features of the Intel Arria 10 SoC Development Kit

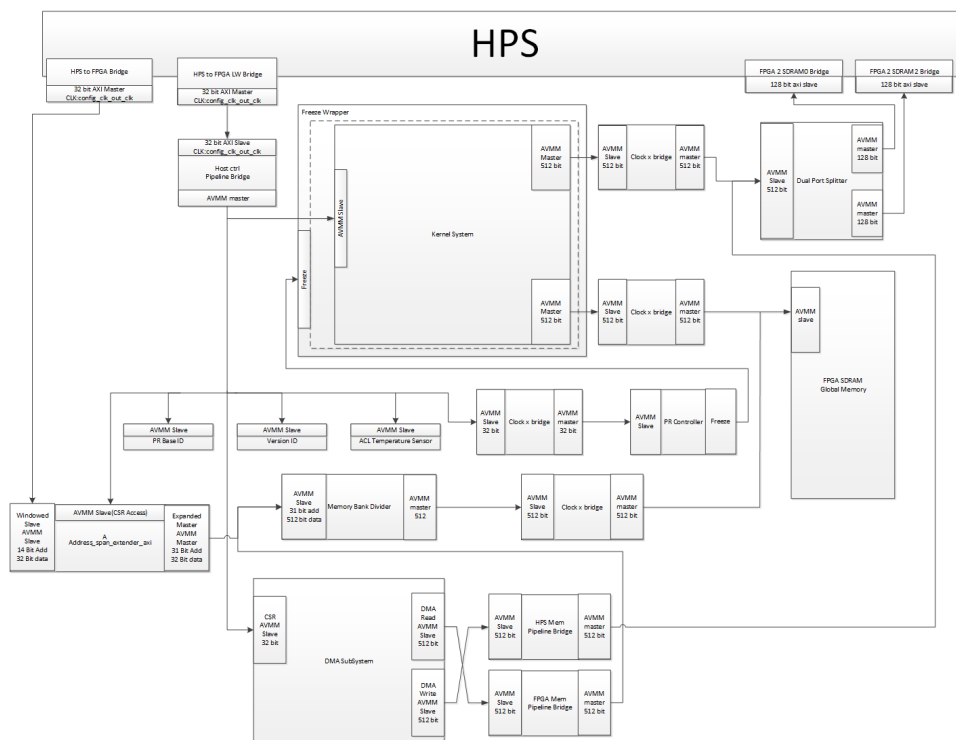


Features of the a10soc Reference Platform:

- OpenCL Host
The a10soc Reference Platform uses the Intel SoC HPS as the host that connects to the FPGA fabric via HPS-to-FPGA (H2F) and FPGA-to-HPS (F2H) bridges.
- OpenCL Global Memory
The hardware provides two 1-gigabyte (GB) DDR4 SDRAM daughtercards that are mounted on the HiLo connectors (HPS Memory and FPGA Memory in [Figure 1](#) on page 5).
- FPGA Programming via Partial Reconfiguration (PR) over HPS lightweight bridge (Lw-bridge)
- Guaranteed Timing

The a10soc Reference Platform relies on the Intel Quartus Prime Pro Edition compiler to provide guaranteed timing closure. The timing-clean a10soc Reference Platform is preserved in the form of a precompiled post-fit netlist (that is, the `base.qdb` Intel Quartus Prime Partition Database File that is part of the `base.qar` Intel Quartus Prime Archive File). The Intel FPGA SDK for OpenCL Offline Compiler imports this preserved post-fit netlist into each OpenCL kernel compilation.

Figure 2. Intel Arria 10 SoC Architecture



1.3. Intel Arria 10 SoC Development Kit Reference Platform Board Variants

The Intel Arria 10 SoC Development Kit Reference Platform includes two board variants.

- `a10soc`—targets the Intel Arria 10 SoC Development Kit with one DDR4 memory. The DDR4 memory is shared between the HPS host and the FPGA.
- `a10soc_2ddr`—targets the Intel Arria 10 SoC Development Kit with two DDR4 memories. One DDR4 memory is an added FPGA memory, and the other DDR4 memory is shared between the HPS host and the FPGA.

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

For example: `aoc -board=a10soc 2ddr myKernel.cl`

Related Information

Compiling a Kernel for a Specific FPGA Board (-board=<board_name>)



1.4. Contents of the Intel Arria 10 SoC Development Kit Reference Platform

Familiarize yourself with the directories and files within the Intel Arria 10 SoC Development Kit Reference Platform because they are referenced throughout this document.

Table 1. Highlights of the Intel Arria 10 SoC Development Kit Reference Platform Directory

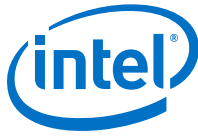
File or Directory	Description
board_env.xml	eXtensible Markup Language (XML) file that describes the Reference Platform to the Intel FPGA SDK for OpenCL.
hardware	Contains the Intel Quartus Prime project templates for the two board variants. Each a10soc Reference Platform board variant implements the entire OpenCL hardware system on a given Intel Arria 10 SoC Development Kit. See Table 2 on page 7 for a list of files in this directory.
arm32	Directory that contains the following: <ul style="list-style-type: none"> A <code>bin</code> subdirectory containing the SDK utilities that are specific to the Intel Arria 10 SoC Development Kit (that is, <code>program</code> and <code>diagnose</code>). A <code>driver</code> subdirectory containing the OpenCL Linux driver source code. A <code>lib</code> subdirectory containing the memory-mapped device (MMD) library that is precompiled to 32-bit Linux on ARM® Cortex®-A9 environment.
a10soc_linux_sd_card_image.tgz a10_2ddr_linux_sd_card_image.tgz	These SD card images include the following: <ul style="list-style-type: none"> Angstrom Linux with kernel version 4.1.22ltsi OpenCL drivers for SoC Intel FPGA Runtime Environment (RTE) for OpenCL version 17.0 Precompiled OpenCL examples

Table 2. Contents of the Board Variant Directory

The following table lists the files in the `<path_to_a10soc>/hardware/<board_name>` directory, where `<board_name>` is the name of the board variant.

File	Description
acl_kernel_interface_soc_pr.qsys	Platform Designer system that implements interface to kernel system in board system.
base.qsf	Intel Quartus Prime Settings File for the base project revision. To compile to base revision, add the <code>-bsp-flow=base</code> argument to <code>aoc</code> command (for example, <code>aoc -bsp-flow=base myKernel.cl</code>). Use this revision when porting the a10soc Reference Platform to your own Custom Platform. The Intel Quartus Prime Pro Edition software compiles this base project revision from source code.
base.qar	Intel Quartus Prime Archive File containing <code>base.qdb</code> , <code>pr_base.id</code> , and <code>base.sdc</code> . This file is generated by the <code>scripts/post_flow_pr.tcl</code> file during base revision compile, and is used during import revision compilation. <code>base.qdb</code> Intel Quartus Prime Database Export File the contains the precompiled netlist of the static regions of the design.

continued...



File	Description
	<p><i>pr_base.id</i> 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.</p> <p><i>base.sdc</i> Synopsys Design Constraints File that the Intel Quartus Prime software autogenerates during a base compilation. The <i>base.sdc</i> file is used in the top revision compilation to import all the timing constraints from the static region.</p>
<i>board.qsys</i>	Platform Designer system that implements the board interfaces (that is, the static region) of the OpenCL hardware system.
<i>board_spec.xml</i>	XML file that provides the definition of the board hardware interfaces to the SDK.
<i>DMA_system.qsys</i>	Platform Designer system that implements DMA between HPS memory and FPGA memory in the <i>a10soc_2ddr</i> board variant
<i>dual_port_splitter.qsys</i>	Platform Designer system that splits requests on single slave to two channels. Used for utilizing two FPGA2SDRAM ports on HPS.
<i>flat.qsf</i>	Intel Quartus Prime Settings File for the flat project revision. This file includes all the common settings, such as pin location assignments, that are used in the other revisions of the project (that is, base, top, and top_synth). The <i>base.qsf</i> , <i>top.qsf</i> , and <i>top_synth.qsf</i> files include, by reference, all the settings in the <i>flat.qsf</i> file. The Intel Quartus Prime software compiles the flat revision with minimal location constraints. The flat revision compilation does not generate a <i>base.qar</i> file that you can use for future import compilations and does not implement the guaranteed timing flow.
<i>import_compile.tcl</i>	Tcl script for the SDK-user compilation flow (that is, import revision compilation).
<i>opencl_bsp_ip.qsf</i>	Intel Quartus Prime Settings File that collects all the required <i>.ip</i> files in a unique location. During flat and base revision compilations, the <i>board.qsys</i> , <i>acl_ddr4_a10.qsys</i> and <i>acl_ddr4_a10_core.qsys</i> Platform Designer files are added to the <i>opencl_bsp_ip.qsf</i> file.
<i>quartus_ini</i>	Contains any special Intel Quartus Prime software options that you need when compiling OpenCL kernels for the <i>a10soc</i> Reference Platform.
<i>top.qpf</i>	Intel Quartus Prime Project File for the OpenCL hardware system.
<i>top.qsf</i>	Intel Quartus Prime Settings File for the SDK-user compilation flow.
<i>top.sdc</i>	Synopsys® Design Constraints File that contains board-specific timing constraints.
<i>top.v</i>	Top-level Verilog Design File for the OpenCL hardware system.
<i>top_post.sdc</i>	Platform Designer and Intel FPGA SDK for OpenCL IP-specific timing constraints.
<i>top_synth.qsf</i>	Intel Quartus Prime Settings File for the Intel Quartus Prime revision in which the OpenCL kernel system is synthesized.
<i>ip/freeze_wrapper.v</i>	Verilog Design File that implements the <i>freeze</i> logic placed at outputs of the PR region.
<i>ip/acl_kernel_interface_soc_pr/</i> <i><file_name></i>	Directory containing the <i>.ip</i> files that the Intel Quartus Prime Pro Edition software needs to parameterize the <i>acl_kernel_interface_soc_pr</i> component.
continued...	



File	Description
	You must provide both the <code>acl_kernel_interface_soc_pr.qsys</code> file and the corresponding <code>.ip</code> files in this directory to the Intel Quartus Prime Pro Edition software.
<code>ip/board/<file_name></code>	Directory containing the <code>.ip</code> files that the Intel Quartus Prime Pro Edition software needs to parameterize the board system. You must provide both the <code>board.qsys</code> file and the corresponding <code>.ip</code> files in this directory to the Intel Quartus Prime Pro Edition software.
<code>ip/DMA_system/<file_name></code>	Directory containing the <code>.ip</code> files that the Intel Quartus Prime Pro Edition software needs to parameterize the DMA_system component in <code>a10soc_2ddr</code> board variant. You must provide both the <code>DMA_system.qsys</code> file and the corresponding <code>.ip</code> files in this directory to the Intel Quartus Prime Pro Edition software.
<code>ip/dual_port_splitter/<file_name></code>	Directory containing the <code>.ip</code> files that the Intel Quartus Prime Pro Edition software needs to parameterize the <code>dual_port_splitter</code> component. You must provide both the <code>dual_port_splitter.qsys</code> file and the corresponding <code>.ip</code> files in this directory to the Intel Quartus Prime Pro Edition software.
<code>ip/irq_controller/<file_name></code>	IP that receives interrupts from the OpenCL kernel system and DMA_system, and sends single IRQ to the host.
<code>ip/mem_splitter_port/<file_name></code>	IP that splits requests across multiple channels on burst word boundary.
<code>scripts/base_write_sdc.tcl</code>	Tcl script that the base revision compilation uses to generate the <code>base.sdc</code> file that contains all the constraints collected in the base revision compilation. The Intel Quartus Prime Pro Edition software uses the <code>base.sdc</code> file when compiling the import (top) revision.
<code>scripts/create_fpga_bin_pr.tcl</code>	Tcl script that generates the <code>fpga.bin</code> file. The <code>fpga.bin</code> file contains all the necessary files for configuring the FPGA.
<code>scripts/post_flow_pr.tcl</code>	Tcl script that implements the guaranteed timing closure flow.
<code>scripts/pre_flow_pr.tcl</code>	Tcl script that executes before the invocation of the Intel Quartus Prime software compilation. Running the script generates the Platform Designer HDL for <code>board.qsys</code> and <code>kernel_system.qsys</code> . It also creates a unique ID for the PR base revision (that is, static region). This unique ID is stored in the <code>pr_base.id</code> file.
<code>scripts/qar_ip_files.tcl</code>	Tcl script that packages up <code>base.qdb</code> , <code>pr_base.id</code> and <code>base.sdc</code> during base revision compile.
<code>scripts/regenerate_cache.tcl</code>	Tcl script that regenerates the BAK cache file in your temporary directory.

1.5. Changes in Intel Arria 10 SoC Development Kit Reference Platform from 17.0 to 17.1

Following is a list of what has changed in a10soc Reference Platform from 17.0 to 17.1 release:

Table 3. Changes in a10soc Reference Platform from 17.0 to 17.1

File	Change
<code>import_compiles.tcl</code>	Updated the file for incremental and fast compile features.
<code>board_spec.xml</code>	Updated the version from 17.0 to 17.1.
continued...	



File	Change
quartus.ini	Added qhd_skip_pr_revision_type_check=on INI to the file.
scripts/post_flow_pr.tcl	Updated the file to: <ul style="list-style-type: none">• Enable the fast compile feature.• Remove manual call to quartus_cpf for creating PR programming file since it now done automatically in the flow.
scripts/create_fpga_bin_pr.tcl	Added the Quartus version as part of fpga.bin.
scripts/qar_ip_files.tcl	Updated the file to include: <ul style="list-style-type: none">• Changes required for renaming .qsys files.• Changes required for moving other tcl scripts into Intel FPGA SDK for OpenCL.
scripts/regenerate_cache.tcl	Updated the file to include changes required to move bak_flow.tcl into Intel FPGA SDK for OpenCL.
scripts/bak_flow.tcl	Moved the file into Intel FPGA SDK for OpenCL.
scripts/helpers.tcl	Moved the file into Intel FPGA SDK for OpenCL.
board.qsys	<ul style="list-style-type: none">• Moved the base address of PR IP from 0xcfb0 to 0xcf00.• Increased the ACL_VERSIONID to 0xA0C7C1E2 due to the PR IP address change.• Synced all IPs.
hw_mmd_constants.h	Increased the ACL_VERSIONID to 0xA0C7C1E2 due to the PR IP address change.
base.qar	Updated the file with ACDS 17.1 static region.

1.6. Changes in Intel Arria 10 SoC Development Kit Reference Platform from 17.1.2 to 18.0

Following is a list of what has changed in a10soc Reference Platform from 17.1.2 to 18.0 release:

High-level changes include:

- Various clean up of import_compile.tcl, pre_flow_pr, post_flow_pr
- Clean up of old top_synth revision (migrated to new simplified PR flow)
- Fix for write .sdc, in base_write_sdc.tcl
- AOC no longer emits .qsys files, it emits just Verilog HDL (kernel_system.v) and injects a .qip file into project. Changes needed:
 - Instantiate module kernel_system from kernel_system.v (in pr_region.v now)
 - If you use add_pipe in board_spec.xml, instantiate pipeline registers in the BSP but within the PR region (see ip/kernel_mem and pr_region.v)

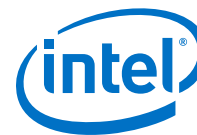


Table 4. Changes in a10soc Reference Platform from 17.1.2 to 18.0

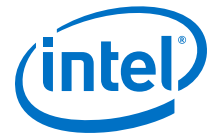
File	Change
base.gar	Updated with 18.0 static region.
board_spec.xml	Updated version, static region used resources, removed addpipe parameter, which is no longer needed.
top.qpf	Removed top_synth revision.
flat.qsf	Removal of obsolete assignments.
base.qsf	Removed Qsys flow.
kernel_system_inst	pr_region_inst
top.qsf	Simplified PR flow.
opencl_bsp_ip.qsf	Removed Qsys flow.
top.sdc	Removed Qsys flow.
import_compile.tcl	Removed Qsys flow, simplified PR flow.
ip/freeze_wrapper.v	Removed Qsys flow.
scripts/base_write_sdc.tcl	Updated to correct base.sdc ordering.
scripts/post_flow_pr.tcl	Updated for fast-compilation
scripts/pre_flow_pr.tcl	Updated for fast-compile, removed Qsys flow, and cleaned up (moved some functions into OpenCL SDK).
scripts/qar_ip_files.tcl	Do not package up opencl_bsp_ip.qsf.
base.gar	Updated with 18.0 static region.

Files added:

- kernel_mem.qsys
- ip/pr_region.v
- ip/kernel_mem/kernel_mem_mm_bridge_0.ip

Removed files:

- top_synth.qsf, which is obsolete because of simplified PR flow.



2. Developing an Intel Arria 10 SoC Custom Platform

Use the tools available in Intel Arria 10 SoC Development Kit Reference Platform (a10soc) 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*
- *Intel FPGA SDK for OpenCL Intel Arria 10 GX FPGA Development Kit Reference Platform Porting Guide*
- Contents of the SDK Custom Platform Toolkit
- *Cyclone V SoC Development Kit Reference Platform Porting Guide*
- Documentation for all the Intel 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, 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 Cyclone V SoC Development Kit Reference Platform Porting Guide](#)
- [Intel FPGA SDK for OpenCL Intel Stratix V Network Reference Platform Porting Guide](#)
- [Intel FPGA SDK for OpenCL Getting Started Guide](#)
- [Intel FPGA SDK for OpenCL Programming Guide](#)

2.1. Initializing an Intel Arria 10 SoC Custom Platform

To initialize your Intel FPGA SDK for OpenCL Custom Platform, first copy the Intel Arria 10 SoC Development Kit Reference Platform to your own directory and rename it.



1. Copy the contents of the `INTELFPGAOCCLSDKROOT/board/a10soc` directory (where `INTELFPGAOCCLSDKROOT` points to the location of your Intel FPGA SDK for OpenCL installation) to a directory that you own and rename the directory (`<your_custom_platform>`).
2. Choose one of the board variants in the `<your_custom_platform>/hardware` directory as the basis of your design.
The `INTELFPGAOCCLSDKROOT/board/a10soc` directory includes the following board variants:
 - `a10soc`—includes one DDR4 shared memory between the HPS host and the FPGA
 - `a10soc_2ddr`—includes one DDR4 shared memory and one DDR4 memory for the FPGA
3. Rename the directory of the chosen board variant to match the name of your FPGA board (`<your_custom_platform>/hardware/<board_name>`). Delete the other `a10socdk` board variant that you do not need.
4. Modify the `<your_custom_platform>/board_env.xml` file so that the `name` and `default` fields match the changes you made in 1 on page 13 and 3 on page 13, respectively.
5. Set the environment variable `AOCL_BOARD_PACKAGE_ROOT` variable to point to the location of your Custom Platform.
6. Invoke the command `aoc -list-boards` to confirm that the Intel FPGA SDK for OpenCL Offline Compiler displays the board name in your Custom Platform.

```
> aoc -list-boards
Board list:
my_board
```

2.2. Modifying Your Intel Arria 10 SoC Custom Platform

After initializing your Intel Arria 10 SoC Custom Platform, modify the existing Intel Quartus Prime design in `<your_custom_platform>` to fit your design needs.

1. Instantiate or edit the HPS IP parameters.
2. Instantiate any controllers required (for example, memory controllers, PR controllers and so on) and I/O channels, if required. 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 `<your_custom_platform>/hardware/<board_name>/flat.qsf` file to use only the pin-outs and settings for your system.
4. Update the offset addresses of controllers in the respective header files in `<your_custom_platform>/arm32/drivers` directory, if you modified any controllers in your design.

2.3. Integrating Your Intel Arria 10 SoC Custom Platform with the Intel FPGA SDK for OpenCL

After modifying the Intel Quartus Prime design files, integrate your Custom Platform with the Intel FPGA SDK for OpenCL.

1. Update the `<your_custom_platform>/hardware/<board_name>/board_spec.xml` file. Ensure that there is at least one global memory interface, and all the global memory interfaces correspond to the exported interfaces from the `board.qsys` Platform Designer System File.

2. Set the environment variable `ACL_DEFAULT_FLOW` to `flat`.

Setting this environment variable instructs the SDK to compile the flat revision corresponding to `<your_custom_platform>/hardware/<board_name>/flat.qsf` file without the partitions or Logic Locks.

Tip: Intel recommends to get a timing clean flat revision compiled before proceeding to the base revision compiles. You can also invoke the following command with the `-bsp-flow=<revision_type>` attribute to run different revisions of your project (for example, flat or base compiles).

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

3. Set the environment variable `ACL_DEFAULT_FLOW` to `base`.

Setting this environment variable instructs the SDK to compile the base revision corresponding to the `<your_custom_platform>/hardware/<board_name>/base.qsf` file.

4. Perform the steps outlined in the `INTELFPGAOCCLSDKROOT/board/custom_platform_toolkit/tests/README.txt` file to compile the `INTELFPGAOCCLSDKROOT/board/custom_platform_toolkit/tests/boardtest/boardtest.cl` OpenCL kernel source file.

The environment variable `INTELFPGAOCCLSDKROOT` points to the location of the Intel FPGA SDK for OpenCL installation.

5. If compilation fails because of timing failures, fix the errors, or compile `INTELFPGAOCCLSDKROOT/board/custom_platform_toolkit/tests/boardtest.cl` with different seeds. To compile the kernel with a different seed, include the `-seed=<N>` option in the `aoc` command (for example, `aoc -seed=2 boardtest.cl`).

2.4. Changing the Device Part Number

When porting the Intel Arria 10 SoC 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 `flat.qsf` file:
 - Change the device part number in the set global assignment `-name DEVICE 10AS066N3F40E2SG` QSF assignment.
 - Update the necessary pin assignment changes.
- The updated device number will appear in the `base.qsf`, `top.qsf`, and `top_synth.qsf` files.
- In the `board.qsys` file, change all occurrences of `10AS066N3F40E2SG`.

2.5. Modifying the Kernel PLL Reference Clock

The Intel Arria 10 SoC Development Kit Reference Platform uses an external 100 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 a10soc 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 `kernel_pll_refclk`.
3. [Optional] In the `<your_custom_platform>/hardware/<board_name>/top.v` file, update the comment for the `kernel_pll_refclk` input port.

After you update the `board.qsys` and the `top.sdc` files, the `post_flow_pr.tcl` script will automatically determine the I/O PLL reference frequency and compute the correct PLL settings.

2.6. Modifying the Hard Processor System

The Intel Arria 10 SoC Development Kit Reference Platform uses HPS as the host system. You can modify the HPS settings in the `<your_custom_platform>/hardware/<board_name>/board.qsys` file. Regenerate the uboot bootloader after you change the HPS settings.

In the reference design, the HPS IP was instantiated with FPGA-to-HPS interface width set to "128-bit AXI", F2SDRAM port configuration set to "Port Configuration 3" and F2SDRAM0 and F2SDRAM2 enabled.

This instantiation was done to maximize kernel to HPS memory bandwidth. A custom IP module was instantiated between kernel memory interface and the two SDRAM ports to split kernel memory access across the ports.

Attention: **You must regenerate the uboot bootloader after you change any HPS settings.** For instructions on regenerating the bootloader, refer the *Intel SoC Embedded Design Suite User Guide*.

Related Information

[Intel FPGA SoC Embedded Design Suite User Guide: Boot Tools User Guide](#)

2.7. Guaranteeing Timing Closure in the Intel Arria 10 SoC Custom Platform

When modifying the Intel Arria 10 SoC Development Kit Reference Platform into your own Custom Platform, ensure that guaranteed timing closure holds true for your Custom Platform.

1. Establish the floorplan of your design.

Important: Consider all design criteria outlined in the *FPGA System Design* section of the *Intel FPGA SDK for OpenCL Custom Platform Toolkit User Guide*.

2. Compile several seeds of the `INTELFPGAOCCLSDKROOT/board/custom_platform_toolkit/tests/boardtest/boardtest.cl` file until you generate a design that closes timing cleanly.

To specify the seed number, include the `-seed=<N>` option in your `aoc` command.

3. Copy the `base.qar` file from the `INTELFPGAOCCLSDKROOT/board/a10soc/` directory into your Custom Platform.
4. Use the `flat.qsf` file in the a10soc Reference Platform as references to determine the type of information you must include in the `flat.qsf` file for your Custom Platform.

The `base.qsf`, `top.qsf`, and `top_synth.qsf` files automatically inherit all the settings in the `flat.qsf` file. However, if you need to modify Logic Lock Plus region or PR assignments, only make these changes in the `base.qsf` file.

5. Remove the `ACL_DEFAULT_FLOW` environment variable that you added when integrating your Custom Platform with the Intel FPGA SDK for OpenCL.
6. Ensure that the environment variable `CL_CONTEXT_COMPILER_MODE_INTELFPGA=3` is not set.
7. Run the `boardtest_host` executable.

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](#)

2.8. Generating the base.qar Post-Fit Netlist for Your Intel Arria 10 SoC FPGA Custom Platform

To implement a compilation flow, you must generate a `base.qar` Intel Quartus Prime Archive File for your Intel Arria 10 SoC Custom Platform.

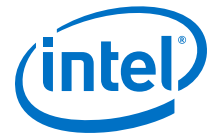


The steps below represent a general procedure for regenerating the `base.qar` file:

1. Port the system design and the `flat.qsf` file to your computing card.
2. Compile the `INTELFPGAOCCLSDKROOT/board/custom_platform_toolkit/tests/boardtest/boardtest.cl` kernel source file using the base revision. Fix any timing failures and recompile the kernel until timing is clean.

Attention: Add the `-bsp-flow=base` argument to the `aoc` command to generate a `base.qar` file during the kernel compilation.

3. Copy the generated `base.qar` file (which contains the `base.qdb` and `pr_base.id` files) into your Custom Platform.
4. Using the default compilation flow, test the `base.qdb` 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} .
 - The OpenCL design examples function on the accelerator board.



3. Building the Software and SD Card Image for the Intel Arria 10 SoC Custom Platform

To build the software for the Intel Arria 10 SoC Custom Platform, compile the device tree blob, Linux kernel, and OpenCL Linux kernel driver. You also need to prepare the micro SD card image.

3.1. Compiling the Device Tree Blob

You must modify the device tree blob contained in the FAT partition to match your Intel Arria 10 SoC Custom Platform. Use the Device Tree Generator (sopc2dts) and the Device Tree Compiler (dtc) to generate the necessary device tree blob.

For detailed information on how to generate the device tree blob, refer to Rocketboards.org.

1. Run base revision compile with your a10soc custom platform.
2. Start an Embedded Command Shell and navigate to the Quartus project directory from your base revision compile.
3. Invoke the following command to generate the .dts Device Tree file, which is a text representation of the Device Tree:

```
sopc2dts --input board/board.sopcinfo \
--output a10soc.dts \
--board hps_a10_common_board_info.xml \
--board hps_a10_devkit_board_info.xml \
--board ghrd_10as066n2_board_info.xml \
--bridge-removal all --clocks
```

The board.sopcinfo file is generated during the base revision compilation of your FPGA design. You may download the XML files from the Intel Arria 10 GHRD on Rocketboards.org.

Attention: Ensure that the name of the Intel Arria 10 Hard Processor System in your board.qsys file matches the name used in Intel Arria 10 GHRD project you are downloading the XML files from. At the time this document was written, the name of the Intel Arria 10 Hard Processor System in board.qsys and in Intel Arria 10 GHRD project was arria10_hps.

4. After you generate the .dts file, modify its contents by performing the following tasks:
 - a. In the Device Tree (a10soc.dts), change the compatible field setting to altr, socfpga.

```
board_irq_ctrl_0: unknown@0x10000cfa0 {
    compatible = "altr,socfpga";

    reg = <0x00000001 0x0000cfa0 0x00000004>;
```



```
<0x00000001 0x0000cf90 0x00000004>;
reg-names = "IRQ_Mask_Slave", "IRQ_Read_Slave";
interrupt-parent = <&a10_hps_arm_gic_0>;
interrupts = <0 19 4>;
interrupt-controller;
#interrupt-cells = <1>;
clocks = <&config_clk>;
}; //end unknown@0x10000cfa0 (board_irq_ctrl_0)
```

Note: The compatible field setting in `a10soc.dts` must match the driver code in `aclsoc.c`. If the two strings do not match, the driver installation process will not allow the kernel to probe the device. As a result, the interrupt will not be registered and the host code will fail.

Code snippet in the `aclsoc.c` file:

```
static const struct of_device_id aclsoc_of_match[] = {
    { .compatible = "altr,socfpga", },
    { /* end of list */ },
};
```

5. After you modify the `.dts` file and it is ready to be probed by the platform driver, compile the device tree blob by invoking the following Device Tree Compiler command:

```
dtc -f -I dts -O dtb -o a10soc.dtb a10soc.dts
```

This command generates the `a10soc.dtb` file.

6. Copy the `a10soc.dtb` file into the FAT partition of the micro SD card and name the file `socfpga_arria10_socdk_sdmmc.dtb`. The device tree blob must match the name of the golden SD card or the system will not boot directly.
7. **If you modify any of the HPS settings in the design, you must regenerate the uboot.**

3.2. Recompiling the Linux Kernel for the Intel Arria 10 SoC Development Kit

Before running OpenCL applications on an Intel Arria 10 SoC board, compile the Linux kernel with the contiguous memory allocator (CMA) feature enabled. Before enabling CMA, recompile the Linux kernel.

1. Click the [GSRD v17.1 - Compiling Linux](#) link on the Resources page of the RocketBoards.org website to access instructions on downloading and rebuilding the Linux kernel source code.

For use with the Intel FPGA SDK for OpenCL, specify `socfpga-4.1.22-ltsi` as the test branch name (`test_branch`). You can find the commands you need to run under **Building Kernel & U-Boot Separately From Git Tree** on the GSRDv17.1 - Compiling Linux page.

2. Add the following lines to the bottom of the `arch/arm/configs/socfpga_defconfig` file:

```
CONFIG_MEMORY_ISOLATION=y
CONFIG_CMA=y
CONFIG_DMA_CMA=y
CONFIG_CMA_DEBUG=y
CONFIG_CMA_SIZE_MBYTES=512
```

```
CONFIG_CMA_SIZE_SEL_MBYTES=y
CONFIG_CMA_ALIGNMENT=8
CONFIG_CMA_AREAS=7
```

Note: The building process creates the `arch/arm/configs/socfpga_defconfig` file. This file specifies the settings for the socfpga default configuration.

The `CONFIG_CMA_SIZE_MBYTES` configuration value sets the upper limit on the total number of physically contiguous memory available. You may increase this value if you require more memory.

3. Run the `make mrproper` command to clean the current configuration.
4. Run the `make ARCH=arm socfpga_defconfig` command.
`ARCH=arm` indicates that you want to configure the ARM architecture.
`socfpga_defconfig` indicates that you want to use the default socfpga configuration.
5. Run the `export CROSS_COMPILE=arm-linux-gnueabi-` command.
This command sets the `CROSS_COMPILE` environment variable to specify the prefix of the desired tool chain.
6. Run the `make ARCH=arm zImage` command. The resulting image is available in the `arch/arm/boot/zImage` file.
7. Place the `zImage` file into the FAT32 partition of the flash card image. For detailed instructions, refer to the Arria 10 GSRD v17.1 User Manual on Rocketboards.org.
8. Insert the programmed micro SD card, which contains the SD card image you modified or created earlier, into the Intel Arria 10 SoC Development Kit and then power up the SoC board.
9. Verify the version of the installed Linux kernel by running the `uname -r` command.
10. To verify that you enable the CMA successfully in the kernel, with the SoC board powered up, run the `grep init_cma /proc/kallsyms` command.
CMA is enabled if the output is non-empty.

Related Information

[Arria 10 Golden System Reference Design \(GSRD\) v17.1 - User Manual](#)

3.3. Compiling and Installing the OpenCL Linux Kernel Driver

Compile the OpenCL Linux kernel driver against the compiled kernel source.

The driver source is available in the Intel Arria 10 SoC reference SD card image. You may compile the driver yourself on a host machine that has `sudo` and the most recent version of the SoC EDS.

1. Mount the original SD card image.
2. Copy the driver source to a host machine directory that you own.
The driver source is available in the `/home/root/opencl_arm32_rte/board/a10soc/driver` directory.
3. To recompile the OpenCL Linux kernel driver, set the `KDIR` value in the driver's `Makefile` to the directory containing the Linux kernel source files.



4. Run the `make clean` command.
5. Run the `make` command to create the `aclsoc_drv.ko` file.
6. Transfer the `aclsoc_drv.ko` directory to the Arria 10 SoC board.
Running the `scp -r <path_to_openc1_arm32_rte> root@your-ip-address:/home/root` command places the runtime environment in the `/home/root` directory.
7. Run the `init_openc1.sh` script that is included in the SD card image.
8. Invoke the `aocl diagnose` utility command. The `diagnose` utility will return a passing result after you run `init_openc1.sh` successfully
9. Run a few design examples to ensure that the you have completed the installation correctly.

3.4. Generating Full-Chip Programming File for SD Card Image

The full-chip programming file, `socfpga.rbf`, is in RBF (Raw Binary File) format, and stored in Partition 1 of the SD Card Image.

This `.rbf` file is used to program the Intel Arria 10 SoC FPGA during power up.

To generate `socfpga.rbf` file, execute the following commands in a directory containing an `aocx` file compiled for your Intel Arria 10 SoC custom platform:

```
1. aocl binedit <.aocx> get .acl.fpga.bin .temp.fpga.bin
2. aocl binedit .temp.fpga.bin get .acl.sof .temp.sof
3. sof2flash --offset=0 --input="./.temp.sof" --output="./.temp_sof2rbf.flash"
4. nios2-elf-objcopy -I srec -O binary "./.temp_sof2rbf.flash" "./socfpga.rbf"
5. rm .temp.fpga.bin .temp.sof .temp_sof2rbf.flash
```

After generating the `socfpga.rbf` file, place it in FAT32 partition of the flash card image.

Remember: The full-chip programming RBF file is different from the partial reconfiguration RBF file generated during PR import compile.

3.5. Building the SD Card Image

The [Creating and Updating the SD Card](#) section in the *Arria 10 GSRD v17.1 User Manual* on Rocketboards.org provides detailed instructions on creating an SD card image for your Intel Arria 10 SoC board.

3.5.1. Layout of the OpenCL Micro SD Card

The micro SD card that the Arria 10 GSRD uses has three partitions, each containing different parts of the OpenCL SD card image.



Table 5.

Location	File Name	Description
Partition 1	socfpga_arria10_socdk_sdmmc.dtb	The device tree blob that describes the peripherals available to the system. Refer to <i>Compiling the Device Tree Blob</i> for more information.
	socfpga.rbf	The full-chip .rbf (Raw Binary file) file generated from Quartus compile. This is different from the PR .rbf file. Refer to Generating Full-Chip Programming File for SD Card Image on page 21 for more details.
	zImage	The compressed kernel image. Refer to <i>Recompiling the Linux Kernel for the Intel Arria 10 SoC Development Kit</i> for more information.
Partition 2	Various rootfs files	Partition 2 is a Linux partition that contains the uncompressed root file system (rootfs). The OpenCL reference SD card rootfs is very similar to the Arria 10 GHRD image. Refer to the <i>Compiling Linux Kernel and Root Filesystem</i> section in the <i>Arria 10 GSRD v17.1 User Manual</i> for more information.
Partition 3	uboot_w_dtb-mkpimage.bin	Partition 3 must be of type a2. The Master Boot Record recognizes the partition and then loads the uboot_w_dtb-mkpimage.bin bootloader from it. Note: The uboot_w_dtb-mkpimage.bin file is written into the a3 partition. Generating U-boot and device tree section in Arria 10 GSRD user manual or the <i>Boot Tools User Guide</i> chapter of the <i>Intel SoC FPGA Embedded Design Suite User Guide</i> describes how to create the uboot_w_dtb-mkpimage.bin file and provides information on the relevant tools.

Related Information

- [Compiling the Device Tree Blob](#) on page 18
- [Intel FPGA SoC Embedded Design Suite User Guide: Boot Tools User Guide](#)
- [Recompiling the Linux Kernel for the Intel Arria 10 SoC Development Kit](#) on page 19
- [Compiling Linux Kernel and Root Filesystem](#)



3.5.2. Guidelines on Imaging the Micro SD Card

After creating all the files in the SD card image, you have several options to image the micro SD card.

General recommendations and resources for imaging the micro SD card:

- Use the Linux `fdisk` command to create, delete, or modify existing partitions on the GSRD SD card image. Use the Linux `dd` command to write file systems to the existing partitions on the GSRD SD card image.
Important: To use these commands, you must have extensive Linux knowledge and have `sudo` on your machine.
- Rocketboards.org provides a python script that generates a `.bin` file. You can write this `.bin` file to the micro SD card. Refer to the *Creating and Updating the SD Card* section of the *Arria 10 GSRD v17.1 User Manual* for more information.
- Intel provides the `alt-boot-disk-util` SD card boot utility to create SD boot images. For more information, refer to the *SD Card Boot Utility* chapter of the *Intel SoC FPGA Embedded Design Suite User Guide*.

The different partitions in the micro SD card are related to each other. For example, if the OpenCL `.rbf` file is not programmed onto the FPGA, the driver will not load. In addition, the `socfpga.rbf` file will not program the board when you boot it up if the `.rbf` file name does not match the label in the bootloader library. Before modifying the SD card image, consider whether the modification is necessary for your design.

Related Information

- [Creating and Updating the SD Card](#)
- [Intel FPGA SoC Embedded Design Suite User Guide: SD Card Boot Utility](#)

3.6. Known Issues

Currently, there are several limitations on the usage of the Intel FPGA SDK for OpenCL with the Intel Arria 10 SoC Development Kit Reference Platform.



- You cannot override the vendor and board names that the `CL_DEVICE_VENDOR` and `CL_DEVICE_NAME` strings of the `clGetDeviceInfo()` call reports, respectively.
- If the host allocates constant memory in the shared DDR system (that is, HPS DDR) and it modifies the constant memory after kernel execution, the data in memory might become outdated. This issue arises because the FPGA core cannot snoop on CPU-to-HPS DDR transactions.

To prevent subsequent kernel executions from accessing outdated data, implement one of the following workarounds:

- Do not modify constant memory after its initialization.
- If you require multiple `__constant` data sets, create multiple constant memory buffers.
- If available, allocate constant memory in the FPGA DDR on your accelerator board.
- The SDK utility on ARM only supports the `program` and `diagnose` utility commands. The `flash`, `install`, and `uninstall` utility commands are not applicable to the Intel Arria 10 SoC Development Kit for the following reasons:
 - The `install` utility has to compile the `aclsoc_drv` Linux kernel driver and enable it on the SoC. The development machine has to perform the compilation; however, it already contains Linux kernel sources for the SoC. The Linux kernel sources for the development machine are different from those for the SoC. The location of the Linux kernel sources for the SoC is likely unknown to the SDK user. Similarly, the `uninstall` utility is also unavailable to the Intel Arria 10 SoC Development Kit.

Also, delivering `aclsoc_drv` to the SoC board is challenging because the default distribution of the Intel Arria 10 SoC Development Kit does not contain Linux kernel `include` files or the GNU Compiler Collection (GCC) compiler.
 - The `flash` utility requires placing a `.rbf` file of an OpenCL design onto the FAT32 partition of the micro SD flash card. Currently, this partition is not mounted when the SDK user powers up the board. Therefore, the best way to update the partition is to use a flash card reader and the development machine.
- When switching between the Intel FPGA SDK for OpenCL Offline Compiler executable files (`.aocx`) that correspond to different board variants (that is, `a10soc` and `a10soc_2ddr`), you must use the SDK's `program` utility to load the `.aocx` file for the new board variant for the first time. If you simply run the host application using a new board variant but the FPGA contains the image from another board variant, a fatal error might occur.
- When you power up the board, it does not acquire an IP address by default. Invoke the `ifup eth0` command to initiate IP address acquisition.

4. Document Revision History

Table 6. Document Revision History of the Intel FPGA SDK for OpenCL Intel Arria 10 SoC Development Kit Reference Platform Porting Guide

Date	Version	Changes
September 2018	2018.09.17	Added <i>Changes in Intel Arria 10 SoC Development Kit Reference Platform from 17.1.2 to 18.0</i>
November 2017	2017.11.03	<ul style="list-style-type: none"> Rebranded references of the following: <ul style="list-style-type: none"> altera_a10socdk to a10soc Environment variable <code>ALTERAOCLSDKROOT</code> to <code>INTELFPGAOCSDKROOT</code>. Arria 10 to Intel Arria 10. <code>CL_CONTEXT_COMPILER_MODE_ALTERA</code> to <code>CL_CONTEXT_COMPILER_MODE_INTELFPGA</code>. LogicLock to Logic Lock Qsys Pro to Platform Designer In Intel Arria 10 SoC Development Kit Reference Platform: Prerequisites on page 3 assumptions added <i>Intel FPGA SDK for OpenCL Intel Arria 10 GX FPGA Development Kit Reference Platform Porting Guide</i>. In Features of the Intel Arria 10 SoC Development Kit Reference Platform on page 4, removed the Silicon feature. In Modifying Your Intel Arria 10 SoC Custom Platform on page 13, updated step 2 and added step 4. In Recompiling the Linux Kernel for the Intel Arria 10 SoC Development Kit on page 19, updated the step 8 about creating <code>.rbf</code> file. Implemented the single dash and <code>-option=<value></code> conventions in the following topics: <ul style="list-style-type: none"> Intel Arria 10 SoC Development Kit Reference Platform Board Variants on page 6 Contents of the Intel Arria 10 SoC Development Kit Reference Platform on page 7 Initializing an Intel Arria 10 SoC Custom Platform on page 12 Integrating Your Intel Arria 10 SoC Custom Platform with the Intel FPGA SDK for OpenCL on page 14 Guaranteeing Timing Closure in the Intel Arria 10 SoC Custom Platform on page 16 In Contents of the Intel Arria 10 SoC Development Kit Reference Platform on page 7: <ul style="list-style-type: none"> Changed <code>board.Qsys Pro</code>, <code>acl_ddr4_a10.Qsys Pro</code> and <code>acl_ddr4_a10_core.Qsys Pro</code> files as <code>board.qsys</code>, <code>acl_ddr4_a10.qsys</code> and <code>acl_ddr4_a10_core.qsys</code> Platform Designer files. Removed <code>scripts/bak_flow.tcl</code> since it is now moved into OpenCL SDK. Removed <code>scripts/helpers.tcl</code> since it is now moved into OpenCL SDK. Added Changes in Intel Arria 10 SoC Development Kit Reference Platform from 17.0 to 17.1 on page 9 to list all changes in the reference platform from 17.0 to 17.1.

continued...



Date	Version	Changes
		<ul style="list-style-type: none">In Modifying the Hard Processor System on page 15, added a note to regenerate uboot loader for any changes in HPS settings.In Changing the Device Part Number on page 14, added a bullet point to update <code>flat.qsf</code> file for pin assignment changes.In Generating the base.qar Post-Fit Netlist for Your Intel Arria 10 SoC FPGA Custom Platform on page 16, added a note to add the <code>-bsp-flow=base</code> argument to the <code>aoc</code> command to generate a <code>base.qar</code> file during the kernel compilation.In Compiling the Device Tree Blob on page 18:<ul style="list-style-type: none">Added a new step 1.Updated the step 2.Updated step 3 code and added a note.Replaced the code in step 4a.In Step 6, changed <code>socfpga.dtb</code> to <code>socfpga_arria10_socdk_sdmmc.dtb</code>.Updated the title of the topic Building the Software and SD Card Image for the Intel Arria 10 SoC Custom Platform on page 18.Added a new topic Generating Full-Chip Programming File for SD Card Image on page 21.Updated all GSRD related hyperlinks to point to Arria 10 GSRD v17.1 User Manual.In Layout of the OpenCL Micro SD Card on page 21:<ul style="list-style-type: none">Changed <code>socfpga.dtb</code> to <code>socfpga_arria10_socdk_sdmmc.dtb</code>Changed <code>soc_system.rbf</code> to <code>socfpga.rbf</code>Updated the description of <code>socfpga.rbf</code> file.Changed <code>u-boot_w_dtb.bin</code> to <code>uboot_w_dtb-mkpmimage.bin</code>Added a hyperlink to <i>Generating U-boot and device tree</i> section in Arria 10 GSRD user manual.In Guidelines on Imaging the Micro SD Card on page 23, changed <code>soc_system.rbf</code> to <code>socfpga.rbf</code>.In Initializing an Intel Arria 10 SoC Custom Platform on page 12, added the output of <code>aoc -list-boards</code> to step 5.In Integrating Your Intel Arria 10 SoC Custom Platform with the Intel FPGA SDK for OpenCL on page 14, added a new step 2 about setting environment variable <code>ACL_DEFAULT_FLOW</code> to <code>flat</code>.In Compiling the Device Tree Blob on page 18, added step 7 about regenerating uboot in case of change HPS settings modification in the design.
May 2017	2017.05.08	<ul style="list-style-type: none">Rebranded various references as follows:<ul style="list-style-type: none">Rebranded Altera SDK for OpenCL to Intel FPGA SDK for OpenCL.Rebranded Altera Offline Compiler to Intel FPGA SDK for OpenCL Offline Compiler.Rebranded Altera RTE for OpenCL to Intel FPGA RTE for OpenCL.Changed references from <i>Altera SoC Embedded Design Suite User Guide</i> to <i>Intel SoC FPGA Embedded Design Suite User Guide</i>
October 2016	2016.10.31	Initial release.