Intel® FPGA SDK for OpenCL™

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

UG-20052
2017.05.08

Last updated for Intel® Quartus® Prime Design Suite: 17.0
# Contents

1 Intel® FPGA SDK for OpenCL™ Intel® Arria® 10 SoC Development Kit Reference Platform Porting Guide................................. 3  
1.1 Arria 10 SoC Development Kit Reference Platform: Prerequisites ................................................................. 3  
1.2 Features of the Arria 10 SoC Development Kit Reference Platform ................................................................. 4  
1.3 Arria 10 SoC Development Kit Reference Platform Board Variants .................................................................................. 6  
1.4 Contents of the Arria 10 SoC Development Kit Reference Platform .............................................................................. 6  
 
2 Developing an Arria 10 SoC Custom Platform ........................................................................................................... 10  
2.1 Initializing an Arria 10 SoC Custom Platform .................................................................................................................. 10  
2.2 Modifying Your Arria 10 SoC Custom Platform .............................................................................................................. 11  
2.3 Integrating Your Arria 10 SoC Custom Platform with the Intel FPGA SDK for OpenCL .................................................... 11  
2.4 Changing the Device Part Number ................................................................................................................................. 12  
2.5 Modifying the Kernel PLL Reference Clock .................................................................................................................. 12  
2.6 Modifying the Hard Processor System ........................................................................................................................... 13  
2.7 Guaranteeing Timing Closure in the Arria 10 SoC Custom Platform .................................................................................. 13  
2.8 Generating the base.qar Post-Fit Netlist for Your Arria 10 SoC FPGA Custom Platform ......................................................... 14  
 
3 Building the Software for the A10 SoC Custom Platform ............................................................................................ 15  
3.1 Compiling the Device Tree Blob ........................................................................................................................................ 15  
3.2 Recompiling the Linux Kernel for the Arria 10 SoC Development Kit .................................................................................. 16  
3.3 Compiling and Installing the OpenCL Linux Kernel Driver ............................................................................................. 17  
3.4 Building the SD Card Image ............................................................................................................................................... 18  
3.4.1 Layout of the OpenCL Linux Kernel Driver .................................................................................................................. 18  
3.4.2 Guidelines on Imaging the SD Card ............................................................................................................................. 19  
3.5 Known Issues ..................................................................................................................................................................... 20  
 
4 Document Revision History ........................................................................................................................................... 22
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 (altera_a10socdk) into your own Custom Platform for use with the Intel FPGA SDK for OpenCL™ 1 2 SDK.

1.1 Arria 10 SoC Development Kit Reference Platform: Prerequisites

The 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 altera_a10socdk Reference Platform:

• An Arria 10 SoC-based accelerator card with working memory interfaces
  Test these interfaces together in the same design using the same version of the Quartus® Prime Pro Edition software that you will use to develop your Custom Platform.

• Quartus Prime Pro Edition software version 17.0

• Designing with LogicLock™ Plus regions

• Intel SoC Embedded Design Suite version 17.0

General knowledge prerequisites:

• FPGA architecture, including clocking, global routing, and I/Os

• High-speed design

• Timing analysis

• Qsys Pro 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 Cyclone V SoC Development Kit Reference Platform Porting Guide

Related Links

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

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

Features of the altera_a10socdk Reference Platform:

- **Silicon**
  
  The current version of the altera_a10socdk Reference Platform targets an Arria 10 SoC Development Kit containing the production silicon for Arria 10 FPGA (-2 speed grade) and DDR4-2133 SDRAM.

- **OpenCL Host**
  
  The altera_a10socdk 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 altera_a10socdk Reference Platform relies on the Quartus Prime Pro Edition compiler to provide guaranteed timing closure. The timing-clean altera_a10socdk Reference Platform is preserved in the form of a precompiled post-fit netlist (that is, the base.qdb Quartus Prime Partition Database File that is part of the base.qar Quartus Prime Archive File). The Intel FPGA SDK for OpenCL Offline Compiler imports this preserved post-fit netlist into each OpenCL kernel compilation.

1.3 Arria 10 SoC Development Kit Reference Platform Board Variants

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

- **a10soc**—targets the 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 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 Links**

Compiling a Kernel for a Specific FPGA Board (`--board <board_name>`)
### Contents of the Board Variant Directory

The following table lists the files in the `<path_to_altera_a10socdk>/hardware/〈board_name>` directory, where `<board_name>` is the name of the board variant.

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>acl_kernel_interface_soc_pr.qsys</td>
<td>Qsys Pro system that implements interface to kernel system in board system.</td>
</tr>
<tr>
<td>base.qsf</td>
<td>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. Use this revision when porting the altera_a10socdk Reference Platform to your own Custom Platform. The Quartus Prime Pro Edition software compiles this base project revision from source code.</td>
</tr>
<tr>
<td>base.qar</td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td><strong>base.qdb</strong> Quartus Prime Database Export File the contains the precompiled netlist of the static regions of the design.</td>
</tr>
<tr>
<td></td>
<td><strong>pr_base.id</strong> 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.</td>
</tr>
<tr>
<td></td>
<td><strong>base.sdc</strong> Synopsys Design Constraints File that the Quartus Prime software autogenerates during a base compilation. The <code>base.sdc</code> file is used in the top revision compilation to import all the timing constraints from the static region.</td>
</tr>
<tr>
<td>board.qsys</td>
<td>Qsys Pro system that implements the board interfaces (that is, the static region) of the OpenCL hardware system.</td>
</tr>
<tr>
<td>board_spec.xml</td>
<td>XML file that provides the definition of the board hardware interfaces to the SDK.</td>
</tr>
<tr>
<td>DMA_system.qsys</td>
<td>Qsys Pro system that implements DMA between HPS memory and FPGA memory in the a10soc_2ddr board variant.</td>
</tr>
<tr>
<td>dual_port_splitter.qsys</td>
<td>Qsys Pro system that splits requests on single slave to two channels. Used for utilizing two FPGA2SDRAM ports on HPS.</td>
</tr>
<tr>
<td>flat.qsf</td>
<td>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 <code>base.qsf</code>, <code>top.qsf</code>, and <code>top_synth.qsf</code> files include, by reference, all the settings in the <code>flat.qsf</code> file. The Quartus Prime software compiles the flat revision with minimal location constraints. The flat revision compilation does not generate a <code>base.qar</code> file that you can use for future import compilations and does not implement the guaranteed timing flow.</td>
</tr>
<tr>
<td>import_compile.tcl</td>
<td>Tcl script for the SDK-user compilation flow (that is, import revision compilation).</td>
</tr>
</tbody>
</table>

**continued...**
<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>opencl_bsp_ip.qsf</td>
<td>Quartus Prime Settings File that collects all the required .ip files in a unique location. During flat and base revision compilations, the board.Qsys Pro, acl_ddr4_a10.Qsys Pro and acl_ddr4_a10_core.Qsys Pro files are added to the opencl_bsp_ip.qsf file.</td>
</tr>
<tr>
<td>quartus_ini</td>
<td>Contains any special Quartus Prime software options that you need when compiling OpenCL kernels for the altera_a10socdk Reference Platform.</td>
</tr>
<tr>
<td>top.qpf</td>
<td>Quartus Prime Project File for the OpenCL hardware system.</td>
</tr>
<tr>
<td>top.qsf</td>
<td>Quartus Prime Settings File for the SDK-user compilation flow.</td>
</tr>
<tr>
<td>top.sdc</td>
<td>Synopsys® Design Constraints File that contains board-specific timing constraints.</td>
</tr>
<tr>
<td>top.v</td>
<td>Top-level Verilog Design File for the OpenCL hardware system.</td>
</tr>
<tr>
<td>top_post.sdc</td>
<td>Qsys Pro and Intel FPGA SDK for OpenCL IP-specific timing constraints.</td>
</tr>
<tr>
<td>top_synth.qsf</td>
<td>Quartus Prime Settings File for the Quartus Prime revision in which the OpenCL kernel system is synthesized.</td>
</tr>
<tr>
<td>ip/freeze_wrapper.v</td>
<td>Verilog Design File that implements the freeze logic placed at outputs of the PR region.</td>
</tr>
<tr>
<td>ip/acl_kernel_interface_soc_pr/&lt;file_name&gt;</td>
<td>Directory containing the .ip files that the Quartus Prime Pro Edition software needs to parameterize the acl_kernel_interface_soc_pr component. You must provide both the acl_kernel_interface_soc_pr.qsys file and the corresponding .ip files in this directory to the Quartus Prime Pro Edition software.</td>
</tr>
<tr>
<td>ip/board/&lt;file_name&gt;</td>
<td>Directory containing the .ip files that the Quartus Prime Pro Edition software needs to parameterize the board system. You must provide both the board.qsys file and the corresponding .ip files in this directory to the Quartus Prime Pro Edition software.</td>
</tr>
<tr>
<td>ip/DMA_system/&lt;file_name&gt;</td>
<td>Directory containing the .ip files that the Quartus Prime Pro Edition software needs to parameterize the DMA_system component in a10soc_2ddr board variant. You must provide both the DMA_system.qsys file and the corresponding .ip files in this directory to the Quartus Prime Pro Edition software.</td>
</tr>
<tr>
<td>ip/dual_port_splitter/&lt;file_name&gt;</td>
<td>Directory containing the .ip files that the Quartus Prime Pro Edition software needs to parameterize the dual_port_splitter component. You must provide both the dual_port_splitter.qsys file and the corresponding .ip files in this directory to the Quartus Prime Pro Edition software.</td>
</tr>
<tr>
<td>ip/irq_controller/&lt;file_name&gt;</td>
<td>IP that receives interrupts from the OpenCL kernel system and DMA_system, and sends single IRQ to the host.</td>
</tr>
<tr>
<td>ip/mem_splitter_port/&lt;file_name&gt;</td>
<td>IP that splits requests across multiple channels on burst word boundary.</td>
</tr>
<tr>
<td>scripts/bak_flow.tcl</td>
<td>Tcl script that executes the BAK flow. If a cache of the previous run exists in your temporary directory, it uses that cache.</td>
</tr>
<tr>
<td>scripts/base_write_sdc.tcl</td>
<td>Tcl script that the base revision compilation uses to generate the base.sdc file that contains all the constraints collected in the base revision compilation. The Quartus Prime Pro Edition software uses the base.sdc file when compiling the import (top) revision.</td>
</tr>
</tbody>
</table>

continued...
<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>scripts/create_fpga_bin_pr.tcl</td>
<td>Tcl script that generates the fpga.bin file. The fpga.bin file contains all the necessary files for configuring the FPGA.</td>
</tr>
<tr>
<td>scripts/helpers.tcl</td>
<td>Tcl script with helper functions used by qar_ip_files.tcl</td>
</tr>
<tr>
<td>scripts/post_flow_pr.tcl</td>
<td>Tcl script that implements the guaranteed timing closure flow.</td>
</tr>
<tr>
<td>scripts/pre_flow_pr.tcl</td>
<td>Tcl script that executes before the invocation of the Quartus Prime software compilation. Running the script generates the Qsys Pro HDL for board.qsys and kernel_system.qsys. It also creates a unique ID for the PR base revision (that is, static region). This unique ID is stored in the pr_base.id file.</td>
</tr>
<tr>
<td>scripts/qar_ip_files.tcl</td>
<td>Tcl script that packages up base.qdb, pr_base.id and base.sdc during base revision compile.</td>
</tr>
<tr>
<td>scripts/regenerate_cache.tcl</td>
<td>Tcl script that regenerates the BAK cache file in your temporary directory.</td>
</tr>
</tbody>
</table>
Use the tools available in Arria 10 SoC Development Kit Reference Platform (altera_a10socdk) 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 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 Links
- 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 Arria 10 SoC Custom Platform

To initialize your Intel FPGA SDK for OpenCL Custom Platform, first copy the Arria 10 SoC Development Kit Reference Platform to your own directory and rename it.
1. Copy the contents of the ALTERAOCLSDKROOT/board/a10soc directory (where ALTERAOCLSDKROOT 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 ALTERAOCLSDKROOT/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 altera_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 11 and 3 on page 11, 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.

### 2.2 Modifying Your Arria 10 SoC Custom Platform

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

1. Instantiate or edit the HPS IP parameters.

2. Instantiate any memory controllers and I/O channels. You can add the board interface hardware either as Qsys Pro components in the board.qsys Qsys Pro 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.

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

After modifying the 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 Qsys Pro System File.

2. 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.

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

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

4. If compilation fails because of timing failures, fix the errors, or compile `ALTERAOCLSDKROOT/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 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.
    
    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 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 altera_a10socdk 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 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 u-boot 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.

For instructions on regenerating the bootloader, refer the Intel SoC Embedded Design Suite User Guide.

Related Links

2.7 Guaranteeing Timing Closure in the Arria 10 SoC Custom Platform

When modifying the 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 `ALTERAOCLSDKROOT/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 `ALTERAOCLSDKROOT/board/a10soc/` directory into your Custom Platform.

4. Use the `flat.qsf` file in the `altera_a10socdk` 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 LogicLock 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_ALTERA=3` is not set.

7. Run the `boardtest_host` executable.
Related Links

- 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 Arria 10 SoC FPGA Custom Platform

To implement a compilation flow, you must generate a base.qar Quartus Prime Archive File for your 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 ALTERAOCLESDKROOT/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.
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 Fmax.
   - The OpenCL design examples function on the accelerator board.
3 Building the Software for the A10 SoC Custom Platform

To build the software for the 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 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. Start an Embedded Command Shell and navigate to the GHRD directory.

2. Invoke the following command to generate the .dts file, which is a text representation of the Device Tree:

```
sopc2dts --input 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 Arria 10 GHRD on Rocketboards.org.

3. 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.

```
 acl_iface_kernel_interface_irq_ena_0: unknown@0x100004050 { 
    compatible = "altr,socfpga";
    reg = <0x00000001 0x00004050 0x00000004>;
    interrupt-parent = <&hps_0_arm_gic_0>;
    interrupts = <0 40 4>;
    interrupt-controller;
    #interrupt-cells = <1>;
    clocks = <&clk_0>;
}; //end unknown@0x100004050 {acl_iface_kernel_interface_irq_ena_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:

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

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

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

This command generates the a10soc.dtb file.

5. Copy the a10soc.dtb file into the FAT partition of the micro SD card and name the file socfpga.dtb. The device tree blob must match the name of the golden SD card or the system will not boot directly.

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

Before running OpenCL applications on an 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 v17.0 - 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 GSRDv17.0 - Compiling Linux page.

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

   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
   ```
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-gnueabihf- 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 v17.0 User Manual on Rocketboards.org.

8. **Note:** To correctly insert the OpenCL Linux kernel driver, first load an SDK-generated raw binary file (.rbf) onto the FPGA.
   To create the .rbf file, compile an SDK design example with the Arria 10 SoC Development Kit Reference Platform as the targeted Custom Platform.

9. Place the .rbf file into the FAT32 partition of the flash card image.
   **Attention:** The FAT32 partition must contain both the zImage file and the .rbf file. Without a .rbf file, a fatal error will occur when you insert the driver.

10. Insert the programmed micro SD card, which contains the SD card image you modified or created earlier, into the Arria 10 SoC Development Kit and then power up the SoC board.

11. Verify the version of the installed Linux kernel by running the `uname -r` command.

12. 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 Links**
Arria 10 Golden System Reference Design (GSRD) v17.0 - 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 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_opencl_arm32_rte> root@your-ip-address:/home/root` command places the runtime environment in the `/home/root` directory.
7. Run the `init_opencl.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_opencl.sh` successfully.
9. Run a few design examples to ensure that the you have completed the installation correctly.

### 3.4 Building the SD Card Image

The Creating and Updating the SD Card section in the *Arria 10 GSRD v17.0 User Manual* on Rocketboards.org provides detailed instructions on creating an SD card image for your Arria 10 SoC board.

#### 3.4.1 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>
<thead>
<tr>
<th>Location</th>
<th>File Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Partition 1</td>
<td>socfpga.dtb</td>
<td>The device tree blob that describes the peripherals available to the system. Refer to <em>Compiling the Device Tree Blob</em> for more information.</td>
</tr>
<tr>
<td></td>
<td>soc_system.rbf</td>
<td>The full-chip .rbf file that you generate when you run a base compilation of the Arria 10 SoC Custom Platform. This .rbf file will program the board on power-up.</td>
</tr>
<tr>
<td></td>
<td>u-boot.scr</td>
<td>The u-boot script that is called after the bootloader.</td>
</tr>
<tr>
<td></td>
<td>zImage</td>
<td>The compressed kernel image. Refer to <em>Recompiling the Linux Kernel for the Arria 10 SoC Development Kit</em> for more information.</td>
</tr>
<tr>
<td>Partition 2</td>
<td>Various rootfs files</td>
<td>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.</td>
</tr>
</tbody>
</table>

*continued*...
Partition 3 must be of type a2. The Master Boot Record recognizes the partition and then loads the \texttt{u-boot\_w\_dtb.bin} bootloader from it.

\textbf{Note:} The \texttt{u-boot\_w\_dtb.bin} file is written into the a3 partition. The Boot Tools User Guide chapter of the Intel SoC FPGA Embedded Design Suite User Guide describes how to create the \texttt{u-boot\_w\_dtb.bin} file and provides information on the relevant tools.

### Related Links
- [Compiling the Device Tree Blob](#) on page 15
  You must modify the device tree blob contained in the FAT partition to match your Arria 10 SoC Custom Platform.
- [Recompiling the Linux Kernel for the Arria 10 SoC Development Kit](#) on page 16
  Before enabling CMA, recompile the Linux kernel.
- [Compiling Linux Kernel and Root Filesystem](#)

### 3.4.2 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 \texttt{fdisk} command to create, delete, or modify existing partitions on the GSRD SD card image. Use the Linux \texttt{dd} command to write file systems to the existing partitions on the GSRD SD card image.

  \textit{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 .\texttt{bin} file. You can write this .\texttt{bin} file to the micro SD card. Refer to the \textit{Creating and Updating the SD Card} section of the Arria 10 GSRD v17.0 User Manual for more information.

- Intel provides the \texttt{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 .\texttt{rbf} file is not programmed onto the FPGA, the driver will not load. In addition, the \texttt{soc\_system.rbf} file will not program the board when you boot it up if the .\texttt{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.
3 Building the Software for the A10 SoC Custom Platform

Related Links
- Creating and Updating the SD Card
- Intel FPGA SoC Embedded Design Suite User Guide: SD Card Boot Utility

3.5 Known Issues

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

  Also, delivering `aclsoc_drv` to the SoC board is challenging because the default distribution of the 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 4. Document Revision History of the Intel FPGA SDK for OpenCL Arria 10 SoC Development Kit Reference Platform Porting Guide

<table>
<thead>
<tr>
<th>Date</th>
<th>Version</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>May 2017</td>
<td>2017.05.08</td>
<td>• Rebranded various references as follows:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>— Rebranded Altera SDK for OpenCL to Intel FPGA SDK for OpenCL.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>— Rebranded Altera Offline Compiler to Intel FPGA SDK for OpenCL Offline Compiler.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>— Rebranded Altera RTE for OpenCL to Intel FPGA RTE for OpenCL.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>— Changed references from <em>Altera SoC Embedded Design Suite User Guide</em> to <em>Intel SoC FPGA Embedded Design Suite User Guide</em></td>
</tr>
<tr>
<td>October 2016</td>
<td>2016.10.31</td>
<td>Initial release.</td>
</tr>
</tbody>
</table>