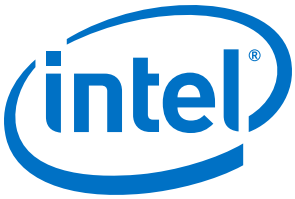

Intel® FPGA SDK for OpenCL™

Intel® Cyclone® V SoC Development Kit Reference Platform Porting Guide

Updated for Intel® Quartus® Prime Design Suite: **17.1**

UG-OCL009 | 2017.11.06
Latest document on the web: **PDF** | **HTML**

Subscribe
Send Feedback

Contents

1 Intel® FPGA SDK for OpenCL™ Intel® Cyclone® V SoC Development Kit Reference Platform Porting Guide

The *Intel® FPGA SDK for OpenCL™ Intel Cyclone® V SoC Development Kit Reference Platform Porting Guide* describes the hardware and software design of the Intel Cyclone V SoC Development Kit Reference Platform (c5soc) for use with the Intel Software Development Kit (SDK) for OpenCL [(1)] [(2)] .

Before you begin, Intel strongly recommends that you familiarize yourself with the contents of the following documents:

1. *Intel FPGA SDK for OpenCLIntel Cyclone V SoC Getting Started Guide*
2. *Intel FPGA SDK for OpenCL Custom Platform Toolkit User Guide*
3. *Cyclone V Hard Processor System Technical Reference Manual*

In addition, refer to the Cyclone V SoC Development Kit and SoC Embedded Design Suite page for more information.

Attention: Intel assumes that you have an in-depth understanding of the *Intel FPGA SDK for OpenCL Custom Platform Toolkit User Guide*. The *Cyclone V SoC Development Kit Reference Platform Porting Guide* does not describe the usage of the SDK's Custom Platform Toolkit to implement a Custom Platform for the Cyclone V SoC Development Kit. It only describes the differences between the SDK support on the Cyclone V SoC Development Kit and a generic Intel FPGA SDK for OpenCL Custom Platform.

Related Links

- Intel FPGA SDK for OpenCL Cyclone V SoC Getting Started Guide
- Intel FPGA SDK for OpenCL Custom Platform Toolkit User Guide
- Cyclone V Hard Processor System Technical Reference Manual
- Cyclone V SoC Development Kit and Intel SoC Embedded Design Suite

1.1 Overview of the Cyclone V SoC Development Kit Reference Platform

The Cyclone V SoC Development Kit Reference Platform is available with the Intel FPGA SDK for OpenCL

---

(1) OpenCL and the OpenCL logo are trademarks 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.

Intel FPGA SDK for OpenCL support for the Cyclone V SoC Development Kit takes advantage of the following board features to maximize the performance of the Cyclone V SoC FPGA:

1. FPGA device that contains the FPGA core logic.
2. Hard processor system (HPS) with dual core ARM® Cortex®-A9 CPU.
3. Shared physical memory between the CPU and the FPGA core fabric.

1.1.1 Cyclone V SoC Development Kit Reference Platform Board Variants

The Intel FPGA SDK for OpenCL Cyclone V SoC Development Kit Reference Platform includes two board variants.

- c5soc board

  This default board provides access to two DDR memory banks. The HPS DDR is accessible by both the FPGA and the CPU. The FPGA DDR is only accessible by the FPGA.

- c5soc_sharedonly board

  This board variant contains only HPS DDR connectivity. The FPGA DDR is not accessible. This board variant is more area efficient because less hardware is necessary to support one DDR memory bank. The c5soc_sharedonly board is also a good prototyping platform for a final production board with a single DDR memory bank.

  To target this board variant when compiling your OpenCL kernel, include the `-board=c5soc_sharedonly` option in your `aoc` command.

  For more information about the `-board=<board_name>` option of the `aoc` command, refer to the *Intel FPGA SDK for OpenCL Programming Guide*.

Related Links

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

1.1.2 Content of the Cyclone V SoC Development Kit Reference Platform

The Cyclone V SoC Development Kit Reference Platform consists of the following files and directories:

File or Directory	Description
`board_env.xml`	eXtensible Markup Language (XML) file that describes c5soc to the Intel FPGA SDK for OpenCL.
`linux_sd_card_image.tgz`	Compressed SD flash card image file that contains everything an SDK user needs to use the Cyclone V SoC Development Kit with the SDK.
`arm32`	Directory that contains the following: 1. A `bin` subdirectory containing SDK utilities that are specific to the Cyclone V SoC Development Kit (that is `program` and `diagnose`). 2. A `lib` subdirectory containing the memory-mapped device (MMD) library that is precompiled to 32-bit Linux on ARM Cortex-A9 environment.

continued...

File or Directory	Description
`c5soc`	Directory that contains the hardware template for the board variant that includes two DDR SDRAM.
`c5soc_sharedonly`	Directory that contains the hardware template for the board variant that includes one DDR SDRAM.
`driver`	Directory that contains the source codes for the Linux kernel driver, and the `program` and `diagnose` utilities.

1.1.3 Relevant Features of the Cyclone V SoC Development Kit

The following list highlights the Cyclone V SoC Development Kit components and features that are relevant to the Intel FPGA SDK for OpenCL:

- Dual-core ARM Cortex-A9 CPU running 32-bit Linux.
- Advanced eXtensible Interface (AXI) bus between the HPS and the FPGA core fabric.
- Two hardened DDR memory controllers, each connecting to a 1 gigabyte (GB) DDR3 SDRAM.
  - One DDR controller is accessible to the FPGA core only (that is, FPGA DDR).
  - The other DDR controller is accessible to both the HPS and the FPGA (that is, HPS DDR). This shared controller allows free memory sharing between the CPU and the FPGA core.
- The CPU can reconfigure the FPGA core fabric.

1.1.3.1 Cyclone V SoC Development Kit Reference Platform Design Goals and Decisions

Intel bases the implementation of the Cyclone V SoC Development Kit Reference Platform on several design goals and decisions. Intel recommends that you consider these goals and decisions when you port this Reference Platform to your SoC FPGA board.

Below are the c5soc design goals:

1. Provide the highest possible bandwidth between kernels on the FPGA and the DDR memory system(s).
2. Ensure that computations on the FPGA (that is, OpenCL kernels) do not interfere with other CPU tasks that might include servicing peripherals.
3. Leave as much FPGA resources as possible for kernel computations instead of interface components.

Below are the high-level design decisions that are the direct consequences of Intel's design goals:

1. The Reference Platform only uses hard DDR memory controllers with the widest-possible configuration (256 bits).
2. The FPGA communicates with the HPS DDR memory controller directly, without involving the AXI bus and the L3 switch inside the HPS. The direct communication provides the best possible bandwidth to DDR, and keeps FPGA computations from interfering with communications between the CPU and its periphery.
3. Scatter-gather direct memory access (SG-DMA) is not part of the FPGA interface logic. Instead of transferring large amounts of data between DDR memory systems, store the data in the shared HPS DDR. Direct access to CPU memory by the FPGA is more efficient than DMA. It saves hardware resources (that is, FPGA area) and simplifies the Linux kernel driver.

   Warning: Memory transfer between the shared HPS DDR system and the DDR system that is accessible only to the FPGA is very slow. If you choose to transfer memory in this manner, use it for very small amounts of data only.
4. The host and the device perform non-DMA data transfer between each other via the HPS-to-FPGA (H2F) bridge, using only a single 32-bit port. The reason is, without DMA, the Linux kernel can only issue a single 32-bit read or write request, so it is unnecessary to have a wider connection.
5. The host sends control signals to the device via a lightweight H2F (LH2F) bridge. Because control signals from the host to the device are low-bandwidth signals, an LH2F bridge is ideal for the task.

1.2 Porting the Reference Platform to Your SoC FPGA Board

To port the Cyclone V SoC Development Kit Reference Platform to your SoC FPGA board, perform the following tasks:

1. Select the one DDR memory or the two DDR memories version of the c5soc Reference Platform as the starting point of your design.
2. Update the pin locations in the *INTELFPGAOCLSDKROOT*`/board/c5soc/`<*board_variant*>`/top.qsf` file,

   where *INTELFPGAOCLSDKROOT* is the path to the location of the Intel FPGA SDK for OpenCL installation, and

   <*board_variant*> is the directory name of the board variant. The `c5soc_sharedonly` directory is for the board variant with one DDR memory system. The `c5soc` directory is for the board variant with two DDR memory systems.
3. Update the DDR settings for the HPS and/or FPGA SDRAM blocks in the *INTELFPGAOCLSDKROOT*`/board/c5soc/`<*board_variant*>`/system.qsys` file.
4. All Intel FPGA SDK for OpenCL preferred board designs must achieve guaranteed timing closure. As such, the placement of the design must be timing clean. To port the c5soc board partition (`acl_iface_partition.qxp`) to your SoC FPGA board, perform the following tasks:

   For detailed instructions on modifying and preserving the board partition, refer to the *Intel Quartus® Prime Incremental Compilation for Hierarchical and Team-Based Design* chapter of the *Intel Quartus Prime Standard Edition Handbook*.

a. Remove the `acl_iface_partition.qxp` from the *INTELFPGAOCLSDKROOT*`/board/c5soc/c5soc` directory.

b. Enable the acl_iface_region Logic Lock region by changing the Tcl command

```
set_global_assignment -name LL_ENABLED OFF -section_id
acl_iface_region
```

to

```
set_global_assignment -name LL_ENABLED ON -section_id
acl_iface_region
```

c. Compile an OpenCL kernel for your board.

d. If necessary, adjust the size and location of the Logic Lock region.

e. When you are satisfied that the placement of your design is timing clean, export that partition as the `acl_iface_partition.qxp` Intel Quartus Prime Exported Partition File.

As described in the *Establishing Guaranteed Timing Flow* section of the *AIntel FPGA SDK for OpenCL Custom Platform Toolkit User Guide*, by importing this `.qxp` file into the top-level design, you fulfill the requirement of providing a board design with a guaranteed timing closure flow.

For factors that might impact the quality of results (QoR) of your exported partition, refer to the *General Quality of Results Considerations for the Exported Board Partition* section in the *Intel FPGA SDK for OpenCL Custom Platform Toolkit User Guide*.

f. Disable the acl_iface_region Logic Lock region by reverting the command in Step 2 back to `set_global_assignment -name LL_ENABLED OFF -section_id acl_iface_region`.

5. If your SoC FPGA board uses different pins and peripheries of the HPS block, regenerate the preloader and the device tree source (DTS) file. If you change the HPS DDR memory controller settings, regenerate the preloader.
6. Create the SD flash card image.
7. Create your Custom Platform, which includes the SD flash card image.

   Consider creating a runtime environment version of your Custom Platform for use with the Intel FPGA Runtime Environment (RTE) for OpenCL. The RTE version of your Custom Platform does not include hardware directories and the SD flash card image. This Custom Platform loads onto the SoC FPGA system to allow host applications to run. In contrast, the SDK version of the Custom Platform is necessary for the SDK to compile OpenCL kernels.

   Tip: You may use the SDK version of your Custom Platform for the RTE. To save space, remove the SD flash card image from the RTE version of your Custom Platform.
8. Test your Custom Platform.

   Refer to the *Testing the Hardware Design* section of the *Intel FPGA SDK for OpenCL Custom Platform Toolkit User Guide* for more information.

Related Links

- Testing the Hardware Design
- Intel Quartus Prime Incremental Compilation for Hierarchical and Team-Based Design

- Establishing Guaranteed Timing Flow
- General Quality of Results Considerations for the Exported Board Partition

1.2.1 Updating a Ported Reference Platform

In the current version of the Cyclone V SoC Development Kit Reference Platform, the HPS block is inside the partition that defines all nonkernel logic. However, you cannot export the HPS as part of the `.qxp` file. To update an existing Custom Platform that you modified from a previous version of c5soc, implement the QXP preservation flow, update the SD flash card image to obtain the latest runtime environment, and update the `board_spec.xml` file to enable automigration.

The Altera® SDK for OpenCL version 14.1 and beyond probes the `board_spec.xml` file for board information, and implements automatic updates. Because you modify the design by implementing the QXP preservation flow, you must update the `board_spec.xml` file to its format in the current version. Updating the file allows the SDK to distinguish between unpreserved Custom Platforms and the current QXP-based Custom Platforms. Refer to *Custom Platform Automigration for Forward Compatibility* in the *Intel FPGA SDK for OpenCL Custom Platform Toolkit User Guide* for more information.

1. To implement the QXP preservation flow in a Cyclone V SoC FPGA hardware design that is ported from a previous version of c5soc, perform the following steps to create a subpartition to exclude the HPS from the `.qxp` file:
   a. Before creating a partition around the nonkernel logic, create a partition around the HPS in the `.qsf` Intel Quartus Prime Settings File.

      For example:

      ```
        # Manually partition the instance that models the HPS-dedicated I/O
      set_instance_assignment -name PARTITION_HIERARCHY borde_18261 -to
      "system:the_system|system_acl_iface:acl_iface|
      system_acl_iface_hps_0:hps_0|system_acl_iface_hps_0_hps_io:hps_io|
      system_acl_iface_hps_0_hps_io_border:border" -section_id
      "system_acl_iface_hps_0_hps_io_border:border"

        # Set partition to be an HPS_PARTITION type to be processed
      correctly by the rest of Quartus
        set_global_assignment -name PARTITION_TYPE HPS_PARTITION -section_id
      "system_acl_iface_hps_0_hps_io_border:border"
      ```

      Modify the setting accordingly because your design hierarchy might be different from the example.

   b. When exporting the partition for `acl_iface_partition`, include the `--incremental_compilation_export_flatten=off` option to leave the HPS partition as a blackbox.

      ```
      quartus_cdb top -c top
      --incremental_compilation_export=acl_iface_partition.qxp
      --incremental_compilation_export_partition_name=acl_iface_partition
      --incremental_compilation_export_post_synth=on
      --incremental_compilation_export_post_fit=on
      --incremental_compilation_export_routing=on
      --incremental_compilation_export_flatten=off
      ```

After you exclude the HPS from the partition, you may import the `.qxp` file and compile your design.

2. Update the SD flash card image with the current version of the Intel FPGA RTE for OpenCL by performing the following tasks:
   a. Mount the file allocation table (fat32) and extended file system (ext3) partitions in the existing image as loop-back devices. For detailed instructions, refer to Step 2 in *Building an SD Flash Card Image*.
   b. In the `/home/root/opencl_arm32_rte` directory, remove the files from the previous version of the RTE.
   c. Download and unpack the current verison of the RTE into the `/home/root/opencl_arm32_rte` directory.
   d. In the *<path_Custom_Platform>*`/driver/version.h` file of your Custom Platform, update the *ACL_DRIVER_VERSION* assignment to *<SDK_version>*.*<driver_version>* (for example, 16.1.*x*, where 16.1 is the SDK verison, and *x* is the driver version that you set).
   e. Rebuild the driver.
   f. Delete the hardware folder(s) of your Custom Platform. Copy the Custom Platform, along with the updated driver, to the `/home/root/opencl_arm_rte/board` directory.
   g. Copy the `Altera.icd` file from the `/home/root/opencl_arm32_rte` directory and add it to the `/etc/OpenCL/vendors` directory.
   h. Unmount and test the new image. For detailed instructions, refer to Steps 8 to 11 in *Building an SD Flash Card Image*.

Related Links

- Creating an SD Flash Card Image on page 13
- Custom Platform Automigration for Forward Compatibility

1.3 Software Support for Shared Memory

Shared physical memory between FPGA and CPU is the preferred memory for OpenCL kernels running on SoC FPGAs. Because the FPGA accesses shared physical memory, as opposed to shared virtual memory, it does not have access to the CPU's page tables that map user virtual addresses to physical page addresses.

With respect to the hardware, OpenCL kernels access shared physical memory through direct connection to the HPS DDR hard memory controller. With respect to the software, support for shared physical memory involves the following considerations:

1. Typical software implementations for allocating memory on the CPU (for example, the `malloc()` function) cannot allocate a memory region that the FPGA may use. Memory that the `malloc()` function allocates is contiguous in the virtual memory address space, but any underlying physical pages are unlikely to be contiguous physically. As such, the host must be able to allocate physically-contiguous memory regions. However, this ability does not exist in user-space applications on Linux. Therefore, the Linux kernel driver must perform the allocation.

2. The OpenCL SoC FPGA Linux kernel driver includes the `mmap()` function to allocate shared physical memory and map it into the user space. The `mmap()` function uses the standard Linux kernel call `dma_alloc_coherent()` to request physically-contiguous memory regions for sharing with a device.

3. In the default Linux kernel, `dma_alloc_coherent()` does not allocate physically-contiguous memory more than 0.5 megabytes (MB) in size. To allow `dma_alloc_coherent()` to allocate large amounts of physically-contiguous memory, enable the contiguous memory allocator (CMA) feature of the Linux kernel and then recompile the Linux kernel.

   For the Cyclone V SoC Development Kit Reference Platform, CMA manages 512 MB out of 1 GB of physical memory. You may increase or decrease this value, depending on the amount of shared memory that the application requires. The `dma_alloc_coherent()` call might not be able to allocate the full 512 MB of physically-contiguous memory; however, it can routinely obtain approximately 450 MB of memory.

4. The CPU can cache memory that the `dma_alloc_coherent()` call allocates. In particular, write operations from the host application are not visible to the OpenCL kernels. The `mmap()` function in the OpenCL SoC FPGA Linux kernel driver also contains calls to the `pgprot_noncached()` or `remap_pf_range()` function to disable caching for this region of memory explicitly.

5. After the `dma_alloc_coherent()` function allocates the physically-contiguous memory, the `mmap()` function returns the virtual address to the beginning of the range, which is the address span of the memory you allocate. The host application requires this virtual address to access the memory. However, the OpenCL kernels require physical addresses. The Linux kernel driver keeps track of the virtual-to-physical address mapping. You can map the physical addresses that `mmap()` returns to actual physical addresses by adding a query to the driver.

   The `aocl_mmd_shared_mem_alloc()` MMD application programming interface (API) call incorporates the following queries:

   a. The `mmap()` function that allocates memory and returns the virtual address.

   b. The extra query that maps the returned virtual address to physical address.

   The `aocl_mmd_shared_mem_alloc()` MMD API call then returns two addresses —the actual returned address is the virtual address, and the physical address goes to `device_ptr_out`.

   Note: The driver can only map the virtual addresses that the `mmap()` function returns to physical addresses. If you request for the physical address of any other virtual pointer, the driver returns a NULL value.

Warning: The Intel FPGA SDK for OpenCL runtime libraries assume that the shared memory is the first memory listed in the `board_spec.xml` file. In other words, the physical address that the Linux kernel driver obtains becomes the Avalon® address that the OpenCL kernel passes to the HPS SDRAM.

With respect to the runtime library, use the `clCreateBuffer()` call to allocate the shared memory as a device buffer in the following manner:

- For the two-DDR board variant with both shared and nonshared memory, `clCreateBuffer()` allocates shared memory if you specify the `CL_MEM_USE_HOST_PTR` flag. Using other flags causes `clCreateBuffer()` to allocate buffer in the nonshared memory.
- For the one-DDR board variant with only shared memory, `clCreateBuffer()` allocates shared memory regardless of which flag you specify.

Currently, 32-bit Linux support on ARM CPU governs the extent of shared memory support in the SDK runtime libraries. In other words, runtime libraries compiled to other environments (for example, x86_64 Linux or 64-bit Windows) do not support shared memory.

C5soc did not implement heterogeneous memory to distinguish between shared and nonshared memory for the following reasons:

1. **History**—Heterogeneous memory support was not available when shared memory support was originally created.
2. **Uniform interface**—Because OpenCL is an open standard, Intel maintains consistency between heterogeneous computing platform vendors. Therefore, the same interface as other board vendors' architectures is used to allocate and use shared memory.

1.4 FPGA Reconfiguration

For SoC FPGAs, the CPU can reconfigure the FPGA core fabric without interrupting the CPU's operation. The FPGA Manager hardware block that straddles the HPS and the core FPGA performs the reconfiguration. The Linux kernel includes a driver that enables easy access to the FPGA Manager.

- To view the status of the FPGA core, invoke the `cat /sys/class/fpga/fpga0/status` command.

The Intel FPGA SDK for OpenCL `program` utility available with the Cyclone V SoC Development Kit Reference Platform uses this interface to program the FPGA. When reprogramming an FPGA core with a running CPU, the `program` utility performs all of the following tasks:

1. Prior to reprogramming, disable all communication bridges between the FPGA and the HPS, both H2F and LH2F bridges.

   Reenable these bridges after reprogramming completes.

   Attention: The OpenCL system does not use the FPGA-to-HPS (F2H) bridge. Refer to the *HPS-FPGA Memory-Mapped Interfaces* section in the *Cyclone V Hard Processor System Technical Reference Manual* for more information.
2. Ensure that the link between the FPGA and the HPS DDR controller is disabled during reprogramming.
3. Ensure that the FPGA interrupts on the FPGA are disabled during reprogramming. Also, notify the driver to reject any interrupts from the FPGA during reprogramming.

Consult the source code of the `program` utility for details on the actual implementation.

Warning: Do not change the configuration of the HPS DDR controller when the CPU is running. Doing so might cause a fatal system error because you might change the DDR controller configuration when there are outstanding memory transactions from the CPU. This means that when the CPU is running, you may not reprogram the FPGA core with an image that uses HPS DDR in a different configuration.

Remember that the OpenCL system, and the Golden Hardware reference design available with the Intel SoC FPGA Embedded Design Suite (EDS), sets the HPS DDR into a single 256-bit mode.

CPU system parts such as the branch predictor or the page table prefetcher might issue DDR commands even when it appears that nothing is running on the CPU. Therefore, boot time is the only safe time to set the HPS DDR controller configuration. This also implies that U-boot must have a raw binary file (`.rbf`) image to load into memory. Otherwise, you might be enabling the HPS DDR with unused ports on the FPGA and then potentially changing the port configurations afterwards. For this reason, the OpenCL Linux kernel driver no longer includes the logic necessary to set the HPS DDR controller configuration.

The SW3 dual in-line package (DIP) switches on the Cylone V SoC Development Kit control the expected form of the `.rbf` image (that is, whether the file is compressed and/or encrypted). C5soc, and the Golden Hardware Reference Design available with the SoC EDS, include compressed but unencrypted `.rbf` images. The SW3 DIP switch settings described in the *Intel FPGA SDK for OpenCL Cyclone V SoC Getting Started Guide* match this `.rbf` image configuration.

Related Links

- HPS-FPGA Memory-Mapped Interfaces
- Configuring the SW3 Switches

1.4.1 FPGA System Architecture Details

Support for the Cyclone V SoC Development Kit Reference Platform is based on the Stratix V Reference Platform (s5_ref), available with the Intel FPGA SDK for OpenCL. The overall organization of the c5soc Platform Designer (Standard) system and the kernel driver are very similar to those in s5_ref.

The following FPGA core components are the same in both c5soc and s5_ref:

- VERSION_ID block
- Rest mechanism
- Memory bank divider
- Cache snoop interface
- Kernel clock
- Control register access (CRA) blocks

1.5 Building an SD Flash Card Image

Because the Cyclone V SoC FPGA is a full system on a chip, you are responsible for delivering the full definition of the system. Intel recommends that you deliver it in the form of an SD flash card image. The Intel FPGA SDK for OpenCL user can simply write the image to the micro SD flash card and the SoC FPGA board is ready for use.

Modifying an Existing SD Flash Card Image on page 13
Creating an SD Flash Card Image on page 13

1.5.1 Modifying an Existing SD Flash Card Image

Intel recommends that you simply modify the image available with the Cyclone V SoC Development Kit Reference Platform. You also have the option to create a new SD flash card image.

The c5soc `linux_sd_card_image.tgz` image file is available in the *INTELFPGAOCLSDKROOT*`/board/c5soc` directory, where *INTELFPGAOCLSDKROOT* points to the path of the Intel FPGA SDK for OpenCL's installation directory.

Attention: To modify the SD flash card image, you must have root or sudo privileges.

1. To decompress the `$INTELFPGAOCLSDKROOT/board/c5soc/linux_sd_card_image.tgz` file, run the `tar xvfz linux_sd_card_image.tgz` command.
2. Compile the hello_world OpenCL example design using your Custom Platform support. Rename the `.rbf` file that the Intel FPGA SDK for OpenCL Offline Compiler generates as `opencl.rbf`, and place it on the fat32 partition within the SD flash card image.

   You can download the hello_world example design from the OpenCL Design Examples page on the Altera website.
3. 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.
4. After you create the SD card image, write it to a micro SD card by invoking the following command:

   ```
   sudo dd if=/path/to/sdcard/image.bin of=/dev/sdcard
   ```
5. To test your SD flash card image, perform the following tasks:
   a. Insert the micro SD flash card into the SoC FPGA board.
   b. Power up the board.
   c. Invoke the `aocl diagnose` utility command.

1.5.2 Creating an SD Flash Card Image

You also have the option to create a new SD flash card image. Generic instructions on building a new SD flash card image and rebuilding an existing SD flash card image are available on the GSRD v17.1 - SD Card page of the RocketBoards.org website.

The steps below describe the procedure for creating the `linux_sd_card_image.tgz` image from the Golden System Reference Design (GSRD) SD flash card image:

Note: To create the image from the c5soc image, perform all applicable tasks outlined in this procedure.

1. Download and unpack the GSRD SD flash card image version 17.0 from Rocketboards.org.
2. Mount the file allocation table (fat32) and extended file system (ext3) partitions in this image as loop-back devices. To mount a partition, perform the following steps:
   a. Determine the byte start of the partition within the image by invoking the `/sbin/fdisk -lu image_file` command.
      For example, partition number 1 of type W95 FAT has a block offset of 2121728. With 512 bytes per block, the byte offset is 512 bytes x 2121728 = 1086324736 bytes.
   b. Identify a free loop device (for example, `/dev/loop0`) by typing the `losetup -f` command.
   c. Assuming `/dev/loop0` is the free loop device, assign your flash card image to the loop block device by invoking the `losetup /dev/loop0 image_file -0 1086324736` command.
   d. Mount the loop device by invoking the mount `/dev/loop0 /media/disk1` command.
      Within the image file, `/media/disk1` is now a mounted fat32 partition.
   e. Repeat steps a to d for the ext3 partition.
3. Download the Cyclone V SoC FPGA version of the Intel FPGA Runtime Environment for OpenCL package from the Download Center on the Altera website.
   a. Click the **Download** button beside Intel Quartus Prime software edition.
   b. Specify the release version, the operating system, and the download method.
   c. Click the **Additional Software** tab, and select to download **Intel FPGA Runtime Environment for OpenCL Linux Cyclone V SoC TGZ**.
   d. After you download the `aocl-rte-<version>.arm32.tgz` file, unpack it to a directory that you own.
4. Place the unpacked `aocl-rte-<version>.arm32` directory into the `/home/root/opencl_arm32_rte` directory on the ext3 partition of the image file.
5. Delete the hardware folder(s) of your Custom Platform, and then place the Custom Platform into the `board` subdirectory of `/home/root/opencl_arm32_rte`.
6. Create the `init_opencl.sh` file in the `/home/root` directory with the following content:

```
export INTELFPGAOCLSDKROOT=/home/root/opencl_arm32_rte
export AOCL_BOARD_PACKAGE_ROOT=$INTELFPGAOCLSDKROOT/board/<board_name>
export PATH=$INTELFPGAOCLSDKROOT/bin:$PATH
export LD_LIBRARY_PATH=$INTELFPGAOCLSDKROOT/host/arm32/lib:$LD_LIBRARY_PATH
insmod $AOCL_BOARD_PACKAGE_ROOT/driver/aclsoc_drv.ko
```

The SDK user runs the `source ./init_opencl.sh` command to load the environment variables and the OpenCL Linux kernel driver.

7. If you need to update the preloader, the DTS files, or the Linux kernel, you need the arm-linux-gnueabihf-gcc compiler from the SoC EDS. Follow the instructions outlined in the *Intel SoC FPGA Embedded Design Suite User Guide* to acquire the software, recompile them, and update the relevant files on the mounted fat32 partition.

   Attention: It is most likely that you need to update the preloader if your Custom Platform has different pin usages than those in c5soc.

   Remember: If you recompile the Linux kernel, recompile the Linux kernel driver with the same Linux kernel source files. If there is a mismatch between the Linux kernel driver and the Linux kernel, the driver will not load. Also, you must enable the CMA.

   Refer to *Recompiling the Linux Kernel* for more information.

8. Compile the hello_world OpenCL example design using your Custom Platform support. Rename the `.rbf` file that the Intel FPGA SDK for OpenCL Offline Compiler generates as `opencl.rbf`, and place it on the fat32 partition within the SD flash card image.

   You can download the hello_world example design from the OpenCL Design Examples page on the Altera website.

9. After you store all the necessary files onto the flash card image, invoke the following commands:

   a. `sync`

   b. `unmount /media/disk1`

   c. `unmount <ext3_partition_directory>`

      where *<ext3_partition_directory>* is the directory name you use for mounting the ext3 partition in 2 on page 14 (for example, `/media/disk2`).

   d. `losetup -d /dev/loop0`

   e. `losetup -d /dev/loop1`

10. Compress the SD flash card image by invoking the following command:

    `tar cvfz <my_linux_sd_card_image>.tgz linux_sd_card_image`

11. Deliver the *<my_linux_sd_card_image>*`.tgz` file inside the root directory of your Custom Platform.

12. To test your SD flash card image, perform the following tasks:

    a. Write the resulting uncompressed image onto a micro SD flash card.

    b. Insert the micro SD flash card into the SoC FPGA board.

    c. Power up the board.

    d. Invoke the `aocl diagnose` utility command.

Related Links

- Intel SoC FPGA Embedded Development Suite User Guide
- OpenCL Design Examples page on the Altera website
- Recompiling the Linux Kernel on page 16

- Querying the Device Name of Your FPGA Board (diagnose)

1.6 Compiling the Linux Kernel for Cyclone V SoC FPGA

Before running OpenCL applications on the Cyclone V SoC FPGA board, you must compile the Linux kernel source, and compile and install the OpenCL Linux kernel driver.

1. Recompiling the Linux Kernel on page 16
2. Compiling and Installing the OpenCL Linux Kernel Driver on page 17

1.6.1 Recompiling the Linux Kernel

To enable the CMA, you must first recompile the Linux kernel.

1. Click the GSRD v14.0 - 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-3.13-rel14.0` as the *<test_branch_name>*.

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.

   Attention: The total amount of physical memory available to the ARM processor on the SoC FPGA board is 1 GB. Intel does not recommend that you set the CMA manager close to 1 GB.

3. Run the `make mrproper` command to clean the current configuration.
4. Run the `make ARCH=arm socfpga_deconfig` 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 Cyclone V SoC FPGA-specific GSRD User Manual on Rocketboards.org.
8. *Note:* To correctly insert the OpenCL Linux kernel driver, first load an SDK-generated `.rbf` file onto the FPGA.

   To create the `.rbf` file, compile an SDK design example with the Cyclone V 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 Cyclone V SoC Development Kit and then power up the SoC FPGA 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 FPGA board powered up, run the `grep init_cma /proc/kallsyms` command. CMA is enabled if the output is non-empty.
13. To use the recompiled Linux kernel with the SDK, compile and install the Linux kernel driver.

Related Links

- Golden System Reference Design (GSRD) User Manuals
- Building an SD Flash Card Image on page 13

1.6.2 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 Cyclone V SoC FPGA version of the Intel FPGA Runtime Environment for OpenCL. In addition, ensure that you have loaded an Intel FPGA SDK for OpenCL-generated `.rbf` file into the FPGA to prevent incorrect installation of the Linux kernel module.

1. Download the Cyclone V SoC FPGA version of the Intel FPGA Runtime Environment for OpenCL package from the Download Center on the Altera website.
   a. Click the **Download** button beside Intel Quartus Prime software edition.
   b. Specify the release version, the operating system, and the download method.
   c. Click the **Additional Software** tab, and select to download **Intel FPGA Runtime Environment for OpenCL Linux Cyclone V SoC TGZ**.
   d. After you download the `aocl-rte-<version>.arm32.tgz` file, unpack it to a directory that you own.

The driver source is in the `aocl-rte-<version>.arm32/board/c5soc/driver` directory.

2. 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.
3. Run the `export CROSS_COMPILE=arm-linux-gnueabihf-` command to indicate the prefix of your tool chain.
4. Run the `make clean` command.
5. Run the `make` command to create the `aclsoc_drv.ko` file.
6. Transfer the `opencl_arm32_rte` directory to the Cyclone V SoC FPGA board.

   Running the `scp -r <path_to_opencl_arm32_rte> root@your-ip-address:<directory>` command places the runtime environment in the `/home/root` directory.
7. Run the `init_opencl.sh` script that you created when you built 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.

1.7 Known Issues

Currently, there are certain limitations on the usage of the Intel FPGA SDK for OpenCL with the Cyclone V SoC Development Kit Reference Platform.

1. You cannot override the vendor and board names reported by the `CL_DEVICE_VENDOR` and `CL_DEVICE_NAME` strings of the `clGetDeviceInfo()` call.
2. If the host allocates constant memory in 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.
3. 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 Cyclone V SoC Development Kit for the following reasons:

   a. The `install` utility has to compile the `aclsoc_drv` Linux kernel driver and enable it on the SoC FPGA. The development machine has to perform the compilation; however, it already contains Linux kernel sources for the SoC FPGA. The Linux kernel sources for the development machine are different

from those for the SoC FPGA. The location of the Linux kernel sources for the SoC FPGA is likely unknown to the SDK user. Similarly, the `uninstall` utility is also unavailable to the Cyclone V SoC Development Kit.

Also, delivering `aclsoc_drv` to the SoC board is challenging because the default distribution of the Cyclone V SoC Development Kit does not contain Linux kernel `include` files or the GNU Compiler Collection (GCC) compiler.

b. 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.

4. When switching between the Intel FPGA SDK for OpenCL Offline Compiler executable files (`.aocx`) that correspond to different board variants (that is, c5soc and c5soc_sharedonly), 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.
5. The `.qxp` file does not include the interface partition assignments because the Intel Quartus Prime software consistently meets timing requirements of this partition.
6. When you power up the board, its media access control (MAC) address is set to a random number. If your LAN policy does not allow this behavior, set the MAC address by performing the following tasks:
   a. During U-Boot power-up, press any key to enter the U-Boot command prompt.
   b. Type `setenv ethaddr 00:07:ed:00:00:03` at the command prompt.

      You may choose any MAC address.
   c. Type the `saveenv` command.
   d. Reboot the board.

1.8 Document Revision History

Table 1. Document Revision History of the Intel FPGA SDK for OpenCL Cyclone V SoC Development Kit Reference Platform Porting Guide

Date	Version	Changes
November 2017	2017.11.03	• Rebranded references to the following: — *ALTERAOCLSDKROOT* to *INTELFPGAOCLSDKROOT* — LogicLock to Logic Lock — Qsys to Platform Designer (Standard) — Quartus Prime to Intel Quartus Prime • Implemented single dash and -option=<value> convention in Cyclone V SoC Development Kit Reference Platform Board Variants on page 4. • Since the hyperlinks were broken, updated the hyperlinks to point to 17.0 SD card image in Creating an SD Flash Card Image on page 13.
May 2017	2017.05.08	• Maintenance release.
Octoboer 2016	2016.10.31	• Rebranded Altera SDK for OpenCL to Intel FPGA SDK for OpenCL. • Rebranded Altera Offline Compiler to Intel FPGA SDK for OpenCL Offline Compiler.

continued...

Date	Version	Changes
May 2016	2016.05.02	• Modified instructions on building and modifying an SD flash card image. • Modified instructions on recompiling the Linux kernel and the OpenCL Linux kernel driver.
November 2015	2015.11.02	• Maintenance release, and changed instances of *Quartus II* to *Quartus Prime*.
May 2015	15.0.0	• In FPGA Reconfiguration, removed instruction to reprogram the FPGA core with a `.rbf` image by invoking the `cat <image_filename>.rbf > /dev/fpga0` command because this method is not recommended.
December 2014	14.1.0	• Renamed the document as *Altera Cyclone V SoC Development Kit Reference Platform Porting Guide*. • Updated the `reprogram` utility to the `aocl program <device_name> <your_kernel_filename>.aocx` utility command. • Updated the `diagnostic` utility to the `aocl diagnose` and `aocl diagnose <device_name>` utility command. • Updated the procedure in the *Porting the Reference Platform to Your SoC Board* section to include instructions on porting and modifying the c5soc board partition to create a timing-clean partition for the guaranteed timing closure flow. • Inserted the topic *Updating a Ported Reference Platform* to outline the procedures for the following tasks: 1. Excluding the hard processor system (HPS) block in the board partition 2. Updating the SD flash card image • Updated the *Building an SD Flash Card Image* section. Recommended using version 14.0 of the Golden System Reference Design (GSRD) image as the starting point instead of the image available with SoC Embedded Design Suite (EDS). • Updated the Recompiling the Linux Kernel and the OpenCL Linux Kernel Driver section: 1. Added instruction to set the *CROSS_COMPILE* variable. 2. Changed the command you run to verify that the CMA is enabled successfully.
July 2014	14.0.0	• Initial Release.