

PCIe Root Port with MSI

An example which implements a PCIe root port on an Altera Cyclone V SoC development board and a PCIe end point on a Cyclone V development board

Board: [AlteraSoCDevelopmentBoard](#)
Tools Version: 15.0
State: running
Members: [LFTAN](#)

Overview.....	
GSRD.....	
Hardware and Software Packages.....	
Getting Started.....	
Release Contents.....	
Hardware.....	
Software.....	
Setting up the example design platform.....	
Prepare SD card.....	
Demo: Setup board and boot.....	
Hardware Development Flow.....	
Qsys Design and Generation.....	
Quartus Compilation.....	
Hardware Validation using System Console.....	
Reading System ID.....	
On chip Memory Data Filling.....	
Setup mSGDMA for Transfer.....	
Convert .sof to .rbf.....	
Software Development Flow.....	
Environment setup based on GSRD.....	
Yocto build.....	
Building components with yocto.....	
Creating SD Image.....	
Individual components build.....	
Building kernel from Rocketboards.org.....	
Building Device Tree Blob.....	
Building Host system driver and End point device driver.....	
Building Throughput Linux application.....	
Kernel Configuration.....	
Disable MSI.....	
Enable MSI.....	
PCI Related Linux Commands.....	
Using PCIe RP Simple Design.....	
Open Issues.....	
References.....	

Update(19 October 2015)

- cv_soc_rp_full_design.tar.gz and cv_soc_rp_simple_design.tar.gz project updated from the Q13.1 to Q15.0.1.
- Rootfs(update from yocto to angstrom).
- The DTB updation and linux kernel update from socfpga-3.9.0 to socfpga-3.10-ltsi.
- 0001-Add-PCIe-RP-and-MSI-drivers.patch, hps_common_board_info.xml and pcie_rp_ed_5csx6c6_board_info.xml are updated.
- preloader and the u-boot are regenerated on Q15.0.1.

Update (8 July 2014)

- Hardware: Enhance data path from DMA transfer into [SoC](#) SDRAM. This enhancement bring up the throughput of PCIe DMA transfer into SDRAM to 700MB/s.
- Software: Enhance PCIe host driver to configure Retrain bit to retrain link to Gen2 speed if hardware support Gen2.

Overview

This document describes how to use Cyclone V SoC with PCIe Root Port design example release package. The FPGA design is based on the Golden System Reference Design(GSRD). Newly added modules include: PCIe RootPort(RP) IP, MSI-toGIC generator IP, MSGDMA and throughput measurement modules. The design demonstrates the Altera PCIe HIP Root Port ability to enumerate a Gen1x4 PCIe Endpoint and measure the link throughput. The Endpoint consists of an Intel® Gigabit CT Desktop Adapter or Cyclone V FPGA with PCIe HIP. This example design is provided as a starting point for PCIe system designs. It consists of both hardware designs and software packages.

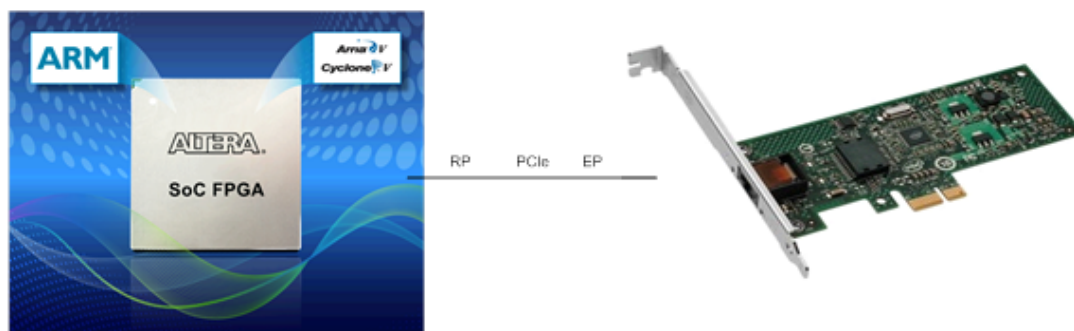


Figure 1: Altera SoC RootPort to Endpoint connectivity through a PCIe Link.

GSRD

The hardware and software designs are based on the GSRD release. Keeping hardware and software design releases in sync is important to ensure compatibility and functionality. This example design is based on Cyclone V SoC GSRD, below are the links for information and kit installation.

GSRD: <http://www.rocketboards.org/foswiki/Documentation/GSRD>

Hardware and Software Packages

Getting Started

This example design is based on Cyclone V SoC GSRD. Keeping hardware and software design releases in sync is important to ensure compatibility and functionality.

The example design is built and tested with Quartus II 15.0.1.

More information for Cyclone V SoC development kit:

<http://www.altera.com/products/devkits/altera/kit-cyclone-v-soc.html>

The following is a link to the "GSRD User Manual".

<http://www.rocketboards.org/foswiki/Documentation/GSRD>

- Step through "Prerequisites", "GHRD Overview" and "Getting Started Guide"
- Understand the "Development Flow Overview"

Release Contents

The PCIe RP example design sources and prebuilt binaries can be downloaded from <https://releases.rocketboards.org/2015.10/pcie-ed>

Folder	File	Description
bin	linux-socfga-pcierp-15.0.1-cv-bin.tar.gz	Cyclone V binaries archive. All the prebuilt images needed for demo and building the SD Image. sd_card_image_cyclone5.bin is a 2GB SD card image tar file contains all the required components for demo, including Preloader, U-boot, rootfs, kernel image and device tree blob.
hw	cv_ep_ram_design.tar.gz	DMA End Point device
^	cv_soc_rp_full_design.tar.gz	This design contains an additional modular SGDMA in the Root Port design. This DMA machine connects directly from on chip memory to the PCIe Tx data path. This connection is used to perform PCIe throughput measurements. Logic usage is about 17,500 ALMs.

Folder	File	Description
^	cv_soc_rp_simple_desi gn.tar.gz	This design provides minimum logic usage by removing the Jtag Masters. Maximum burst count to PCIe Txs data path is 2. An Endpoint DMA is needed to initiate bulk data transfer. A DMA in the HPS can also be used to initiate transfers to the PCIe Endpoint. Logic usage is less than 7,300 ALMs.
src		
^	0001-Add-P Cle-RP-and -MSI-driver s.patch	Patch if building kernel with branch "socfpga-3.10-ltsi"
^	boot.script	U-boot script

Hardware

Hardware require for this example design

- Root Port: Cyclone V SoC Development Kit (5CSXFC6)
- Endpoint: Cyclone V FPGA Development Kit (5CGXFC7)
- Endpoint: Intel® Gigabit CT Desktop Adapter (Intel® 82574L Gigabit Ethernet Controller)
- 4GB microSDHC flash card

Software

Software required for this example design

- sd_card_image_cyclone5.bin to be programmed into SD card

[Back to Top](#)

Setting up the example design platform

Hardware jumper, USB-Blaster, Serial terminal, etc should be set the same as GSRD, please refer to [GSRD Getting Started Guide](#).

Prepare SD card

The release package includes an sd_card_image_cyclone5.bin, which combined all the software require to boot and demo the example design. There are number of ways to program the SD card. For Windows, "Win32 Disk

Imager" (<http://sourceforge.net/projects/win32diskimager/>) can be used. Follow the [GSRD User Manual](#), untar sd_image.bin.tar.zip and program sd_card_image_cyclone5.bin into SD card. For Linux, dd utility is used.

Demo: Setup board and boot

Once you have SD card ready, connect the platform as shown in figures below. Again, the base setup must follow [GSRD Getting Started Guides](#).

1. Insert the SD card programed from Section [Prepare SD card](#).
2. Power up the system.
3. If setting up the Cyclone V FPGA Endpoint (Figure 3), configure the end point FPGA image (pcie_ep_ram_5cgxfc7.sof) with Quartus Programmer.
4. Start a serial terminal (PuTTY or minicom) on the host PC to communicate with the Linux target.
5. Press the warm reset button.
6. Observe Preloader → U-Boot → Linux booting on the serial terminal. Note, U-Boot will auto program the Cyclone V SoC Rootport example design (soc_system.rbf in SD card).
7. At the Linux Kernel, login as root:
socfpga login: root
8. Execute the following commands:
mount /dev/mmcblk0p1 /mnt
cd /mnt
9. mnt# insmod altera_epde.ko
10. mnt# insmod altera_rpde.ko
11. mnt# ./dmaxfer

Note: If just performing the CycloneV Soc PCIe RootPort demo, sections [Hardware Development Flow](#) and [Software Development Flow](#) can be skipped. Both section will step through the Hardware and Software development flows. These flows list how to test the hardware as well as building the software kernel.

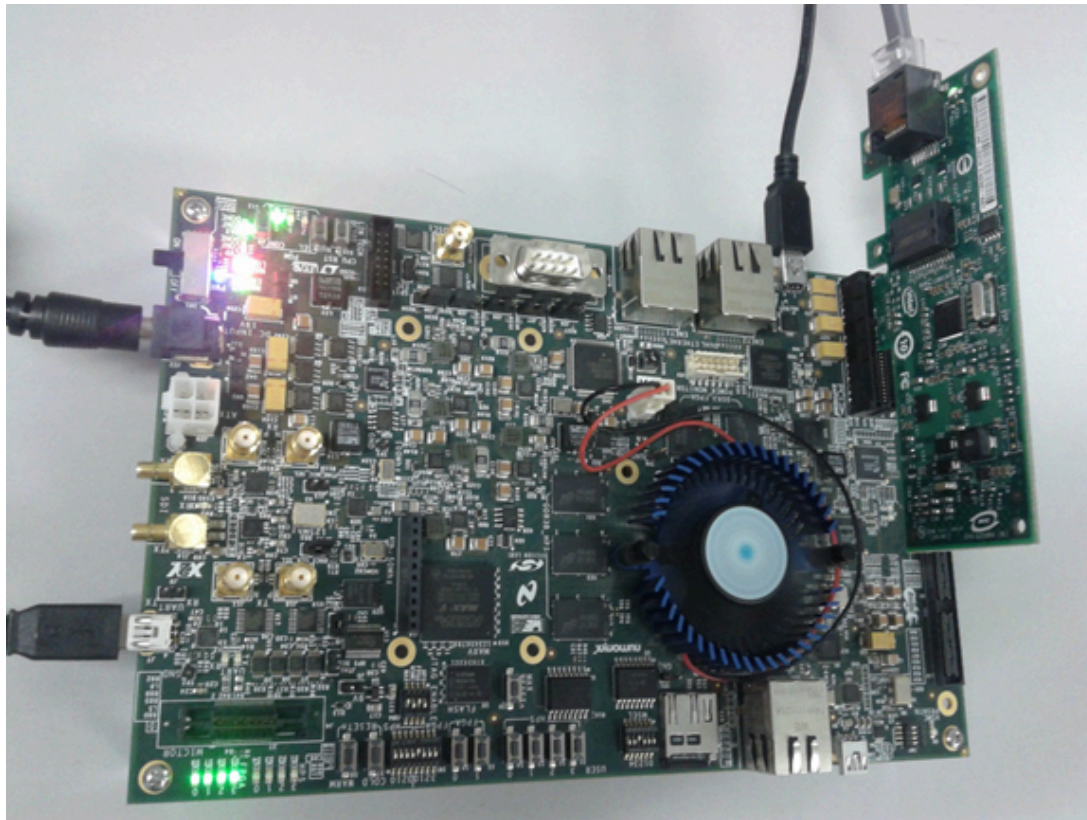


Figure 2: Intel® Gigabit CT Desktop Adapter

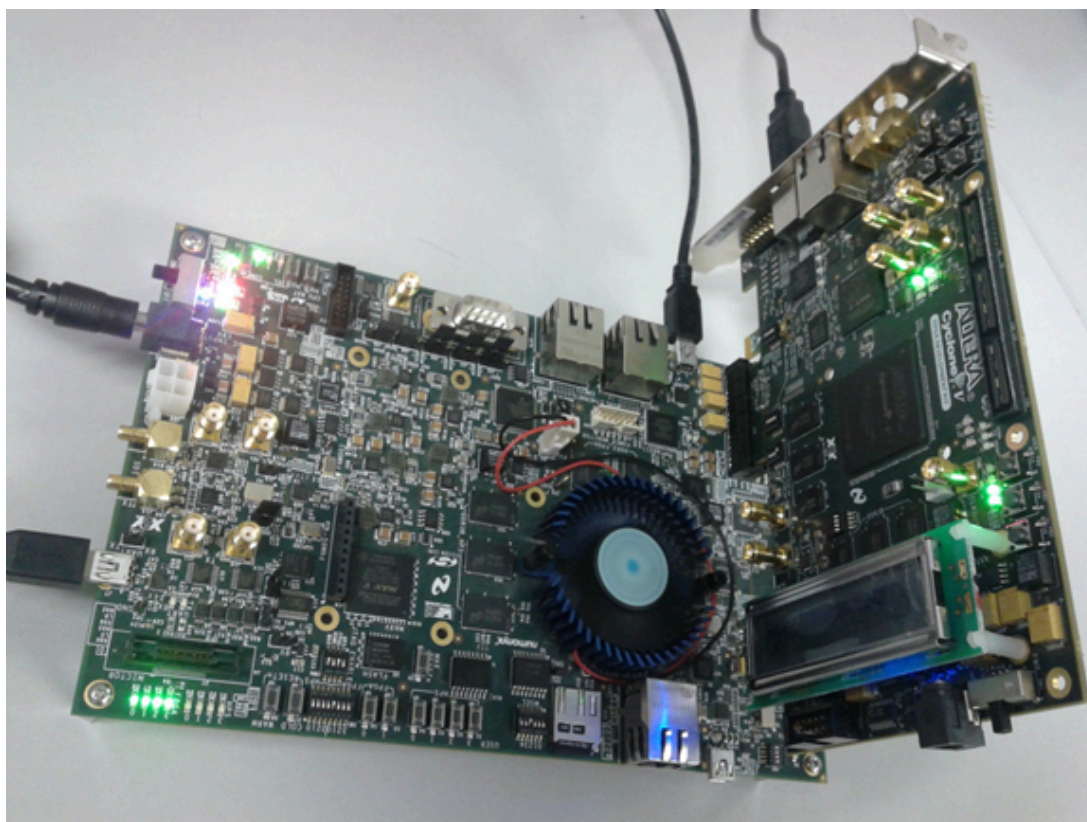


Figure 3: Cyclone V FPGA Development Kit (5CGXFC7) as End Point device

[Back to Top](#)

Hardware Development Flow

Note: Unless the Qsys design is required to be modified, please use the prebuilt image in release package. The following steps are only required if modification to the Qsys design is needed.

Altera Complete Design Suite (ACDS) flow is used for developing the hardware design of PCIe RootPort-EndPoint example, from Qsys generation to Quartus full compilation. However, some components, such as the Modular SGDMA and Interrupt Capture Module, are not currently available in the Qsys 15.0.1 Component Library. These components are included in this example design's, IP folder.

The example design package zip file, *cv_soc_rp_full_design.tar.gz*, consists of the following:

- Qsys design files
- Quartus files
- RTL files (including PCIe IP patch)
- Qsys component library files
- RBF binary

Qsys Design and Generation

pcie_rp_ed_5csx6c6.qsys is the Qsys top level design file. The design consists of a HPS subsystem, PCIe HIP, Modular SGDMA subsystem, and some peripherals designed for PCIe RP example.

The user is only required to perform Qsys generation for *pcie_rp_ed_5csx6c6.qsys*. Generated RTL will be contained in *pcie_rp_ed_5csx6c6* folder.

Quartus Compilation

With HPS SoC SDRAM Controller being enabled in this example design, two step compilation will be required in order to implement the SDRAM pin assignments.

After opening the *pcie_rp_ed.qpf*, a simple Analysis & Synthesis run is required followed by running a TCL script for SDRAM pin assignments. The *hps_sdram_p0_pin_assignments.tcl* TCL script will be run.

The following screen shows illustrate how to use QuartusII to select the TCL script to be run.

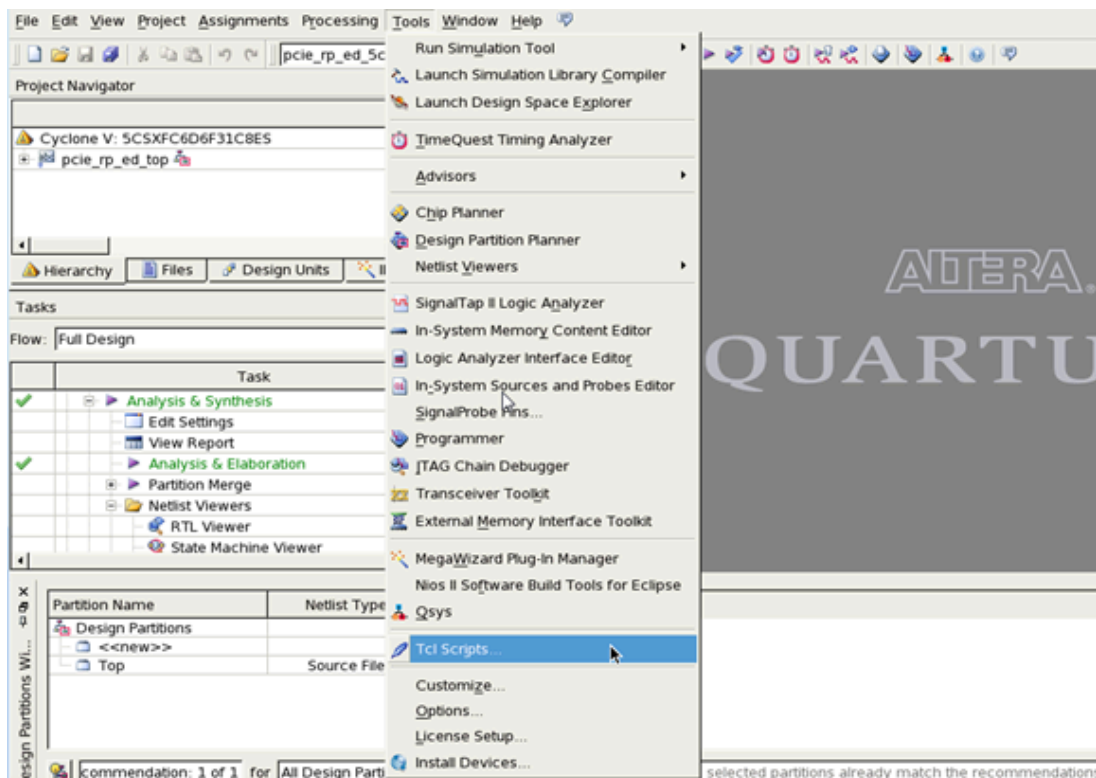


Figure 4: Location of Tcl Scripts option in Quartus II

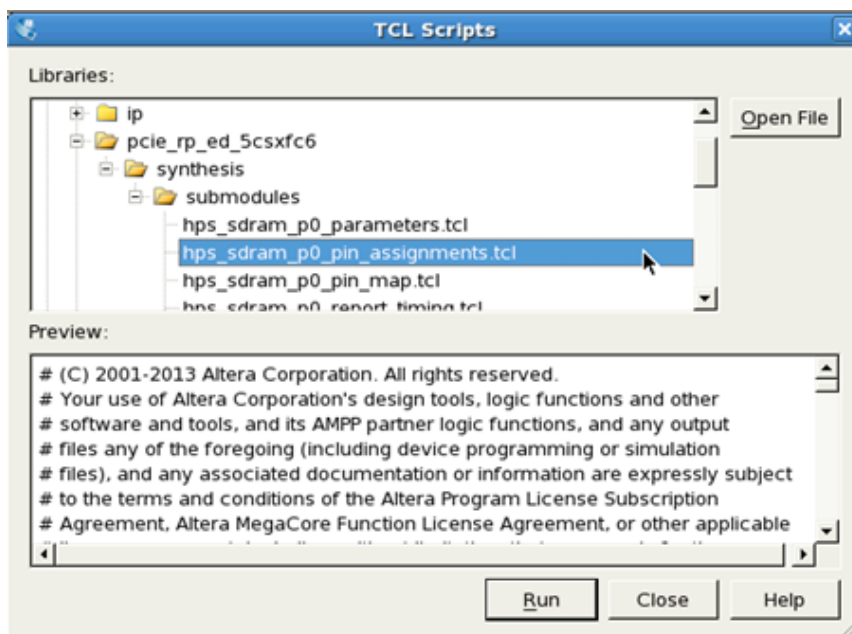


Figure 5: Selecting the tcl script to run

Finally, a full compilation is required to generate a SOF binary. You may refer [Cyclone V Hard IP for PCI Express User Guide](#) for timing optimization.

Hardware Validation using System Console

This is an additional validation process for your design using Altera System Console. Before the software driver is developed, the accessibility of system peripherals can be validated via Altera System Console with a downloaded

SOF into your actual FPGA hardware or development board.

Altera System Console provides master access to the in-system peripherals through Avalon MM JTAG Master component in the designed system. In this PCIe RP example design, there are 3 Avalon MM JTAG Masters being instantiated. Each of them is targeting either FPGA2HPS port, Avalon peripherals in FPGA, or Modular SGDMA control for data transfer to/from PCIe HIP.

After the Altera System Console is evoked, user is recommended to source for a pre-coded TCL file to setup the JTAG Masters for access control. In addition, this TCL file also provides some simple procedure to access the in-system peripherals. For example, access to the System ID, Avalon PIO, JTAG UART, on chip memory, Modular SGDMA setup, and PCIe HIP slave ports.

Following figure shows step to source the pre-coded TCL file, *script_pcie_ed.tcl*. The script also displays three JTAG Masters are being discovered and enabled for control.

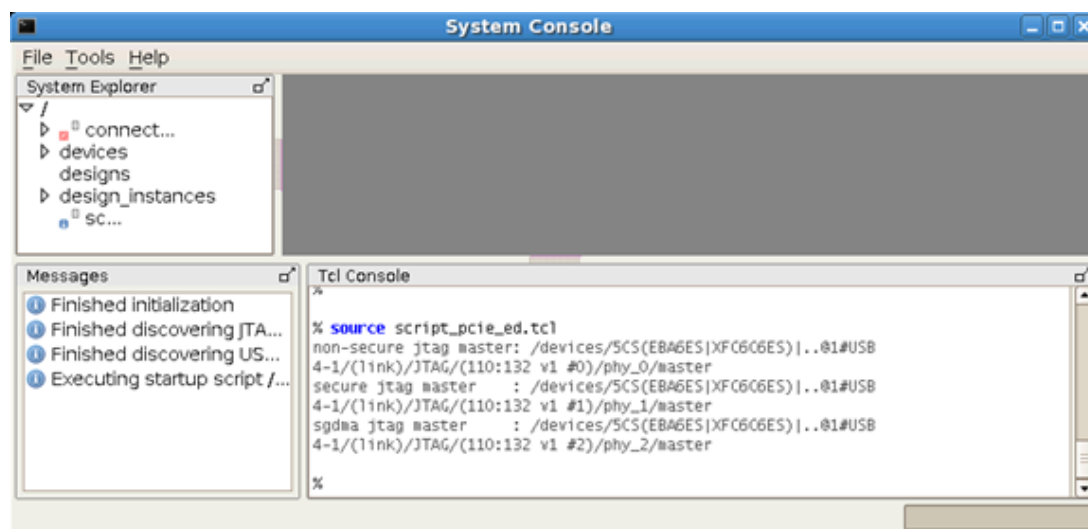


Figure 6: System Console: Sourcing script_pcie_ed.tcl

Reading System ID

Altera recommends including the System ID component for identifying the designed system. The TCL procedure of *sysid_read* will enquiry the system ID for the PCIe example design system from the actual hardware. Following figure shows the read back value of 0xacd51305 for this particular system.

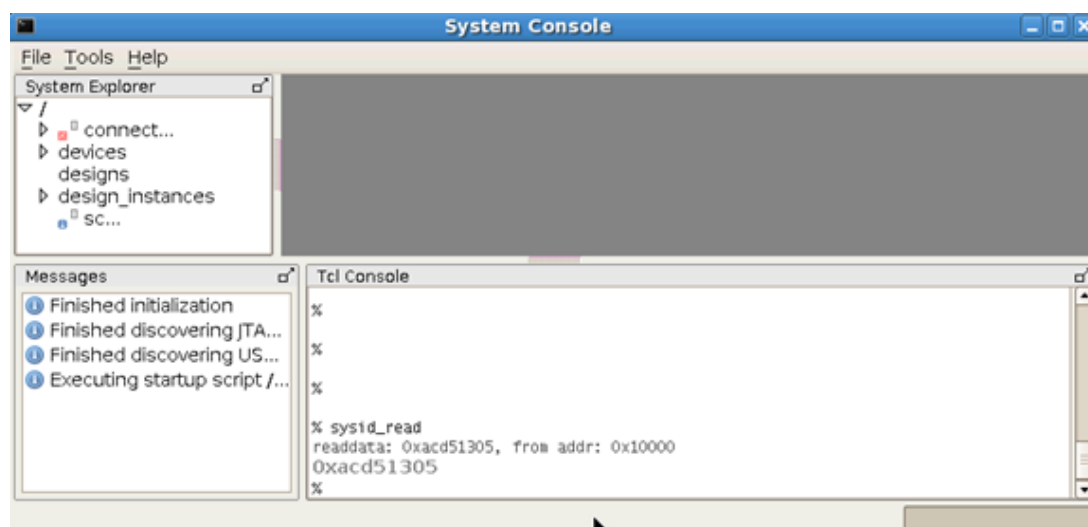


Figure 7: Successful sysid_read command

On chip Memory Data Filling

Procedure “mem_init <start addr> <number of word>” provides a mechanism to fill up on chip memory content. Insert the procedure command at TCL Console will have your on chip memory filled up for the defined words of data.

```
% mem_init 0x20000000 8
written to address: 0x20000000
written to address: 0x20000004
written to address: 0x20000008
written to address: 0x2000000c
written to address: 0x20000010
written to address: 0x20000014
written to address: 0x20000018
written to address: 0x2000001c
Done writing 8 set of data words into On chip Memory
Start reading back for comparison
readdata: 0x00000008, from addr: 0x20000000
readdata: 0x00000007, from addr: 0x20000004
readdata: 0x00000006, from addr: 0x20000008
readdata: 0x00000005, from addr: 0x2000000c
readdata: 0x00000004, from addr: 0x20000010
readdata: 0x00000003, from addr: 0x20000014
readdata: 0x00000002, from addr: 0x20000018
readdata: 0x00000001, from addr: 0x2000001c
Data mismatched: 0
```

Setup mSGDMA for Transfer

The Modular SGDMA needs some configuration before starting data transfer between source to destination location. The following procedures prepare mSGDMA by setting up its descriptors and starting the DMA.

```
% dma_sts
readdata: 0x00000002, from addr: 0x4000
read existing DMA Status reg: 0x00000002
% dma_en_irq
readdata: 0x00000000, from addr: 0x4004
read existing DMA Ctrl reg: 0x00000000
Ctrl reg value to be written: 0x00000010
written to address: 0x4004
```

```
% dma_setup 0x20000000 0x20001000 32
written to address: 0x4020
readdata: 0xdeadddead, from addr: 0x4020
read DMA Descriptor Read Addr: 0xdeadddead
written to address: 0x4024
readdata: 0x00000000, from addr: 0x4024
read DMA Descriptor Write Addr: 0x00000000
written to address: 0x4028
readdata: 0x00000000, from addr: 0x4028
read DMA Descriptor Transfer Size: 0x00000000
readdata: 0x00000000, from addr: 0x402c
read existing DMA Descriptor Ctrl field: 0x00000000
Ctrl reg value to be written: 0x00004000
written to address: 0x402c
% dma_go
Only most significant byte of control field to be written
SGDMA "Go"
% dma_sts
readdata: 0x00000202, from addr: 0x4000
read existing DMA Status reg: 0x00000202
```

At the end of transfer, procedure *dma_sts* used to check the status register of mSGDMA, it shows that the DMA transfer has been completed and descriptor FIFO is now empty.

To verify if the data transferred between source and destination addresses are correct, procedure "after_dma_compare <source addr> <destination addr> <word count>" can be used.

```
% after_dma_compare 0x20000000 0x20001000 8
readdata: 0x00000008, from addr: 0x20000000
readdata: 0x00000007, from addr: 0x20000004
readdata: 0x00000006, from addr: 0x20000008
readdata: 0x00000005, from addr: 0x2000000c
readdata: 0x00000004, from addr: 0x20000010
readdata: 0x00000003, from addr: 0x20000014
readdata: 0x00000002, from addr: 0x20000018
readdata: 0x00000001, from addr: 0x2000001c
Done reading 8 set of data from Read Addr
Start read destination value for comparison
readdata: 0x00000008, from addr: 0x20001000
readdata: 0x00000007, from addr: 0x20001004
```

```
readdata: 0x00000006, from addr: 0x20001008
readdata: 0x00000005, from addr: 0x2000100c
readdata: 0x00000004, from addr: 0x20001010
readdata: 0x00000003, from addr: 0x20001014
readdata: 0x00000002, from addr: 0x20001018
readdata: 0x00000001, from addr: 0x2000101c
Total word transferred: 8
Data mismatched: 0
Data matched : 8
```

Convert .sof to .rbf

Please refer to [GSRD User Manual](#) to convert FPGA image from .sof to .rbf.

[Back to Top](#)

Software Development Flow

The complete software flow is similar to GSRD flow, except to patch kernel and yocto before build binaries. This section describes the additional flow require to build binaries.

Note: Unless Qsys design and/or Software are required to be modified, please use the prebuilt image in the release package. Steps below are required only if the Qsys design and/or Software modification is needed.

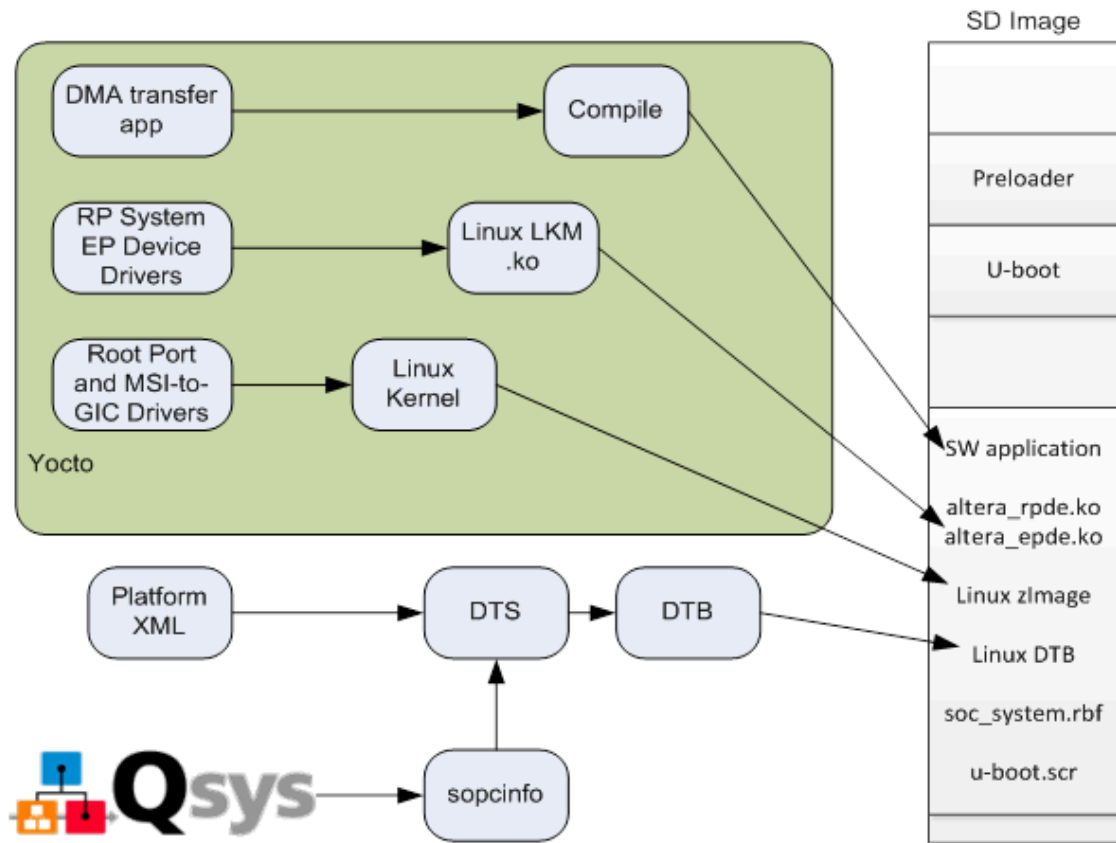


Figure 8: Overview Diagram

Environment setup based on GSRD

Please follow the setup document in the link below, as these steps are required before starting this section.

http://rocketboards.org/foswiki/view/Documentation/AngstromOnSoCFPGA_1

Get Linux BSP of

- Linux kernel
- u-boot
- The tool chain

Export variable

- export CROSS_COMPILE=arm-linux-gnueabi-
- export ARCH=arm
- PATH=<u-boot directory>/tools:\$PATH

Yocto build

This is a complete build using yocto recipes from the GSRD kit installation and patch that build with PCIe RP components. There is no need for step [Individual components build](#) if following steps [Building components with yocto](#) and [Creating SD Image](#).

Building components with yocto

<http://rocketboards.org/foswiki/view/Documentation/AVGSRD150CompilingLinux>

```
1. $ git clone https://github.com/altera-opensource/angstrom-socfpga
2. $ cd angstrom-socfpga/
3. $ git checkout -b test_branch ACDS15.0.1_REL_GSRD_UPDATE1_PR
4. $ ./oebb.sh config cyclone5
5. $ export KERNEL_TAG=refs/tags/ACDS15.0.1_REL_GSRD_UPDATE1_PR
6. $ export BB_ENV_EXTRAWHITE="$BB_ENV_EXTRAWHITE KERNEL_PROT"
7. $ export KERNEL_PROT=http
8. $ source ./environment-angstrom-v2014.12
9. $ bitbake gsr-console-image (generate gsr-console-image-cyclone5.tar.xz)
10. Build preloader image file as described in GSRD
11. Replace all required components into SD Card, or build and replace the whole image as described in Creating SD Image
```

Creating SD Image

For more information about creating SD card, please refer [GSRD User Manual - AV CV GSRD 15.0.1 User manual- Creating and Upating SD Card](#).

```
1. $ sudo tar xzf ./yocto/build/tmp/deploy/images/altera-pcie-image-socfpga_cyclone5.tar.gz -C /tmp/rootfs
2. $ sudo ./make_sdimage.py \ -f \ -P preloader-mkpimage.bin,u-boot-
cyclone5.img,num=3,format=raw,size=10M,type=A2 \ -P rootfs/*,num=2,format=ext3,size=1500M \ -P
zImage,u-
boot.scr,soc_system.rbf,socfpga.dtb,altera_epde.ko,altera_rpde.ko,dmaxfer,num=1,format=vfat,size=500M
\ -s 2G \ -n sd_card_image_cyclone5.bin
3. Follow section Prepare SD card to program SD card
```

Individual components build

Note: Steps for quickly performing a component build for testing and debugging.

Quickly building and debugging individual components is preferable in order to save time. Steps below allow components to be built individually.

Building kernel from Rocketboards.org

```
1. Download linux-socfpga.git from rocketboards.org branch "socfpga-3.10-ltsi" http://rocketboards.org/foswiki/Documentation/GitWeb
```

```

2. $ git clone https://github.com/altera-opensource/linux-socfpga.git
3. $ cd linux-socfpga
4. $ git checkout -b kernel_3.10LTSi origin/socfpga-3.10-ltsi
5. Copy 001-Add-PCIe-RP-and-MSI-drivers.patch to the kernel root source tree
6. Check if patch can be applied

$ git apply --check 001-Add-PCIe-RP-and-MSI-drivers.patch

1. $ git am 001-Add-PCIe-RP-and-MSI-drivers.patch
2. $ make socfpga_defconfig
3. $ make zImage
4. Copy and replace zImage into SD Card

```

Building Device Tree Blob

For more details about device tree generation, please refer to [GSRD User Manual - Generating the Device Tree](#).

```

1. $ socp2dts --input <pcie_design>.sopcinfo --output socfpga.dts -b <pcie_design>_board_info.xml --board
   hps_common_board_info.xml --clocks --bridge-removal all
2. You can find *.sopcinfo and *.xml files from PCIe example design hardware directory.
3. $ dtc -I dts -O dtb -o socfpga.dtb socfpga.dts
4. Copy and replace socfpga.dtb into SD card.

```

Building Host system driver and End point device driver

Both device drivers are built as Loadable Kernel Module (LKM), and load into Kernel at run time after system booted.

```

1. download the modules.tar.gz to <path-to-your-Linux-kernel-directory>
2. untar modules.tar.gz to <path-to-your-Linux-kernel-directory>
3. cd <path-to-your-Linux-kernel-directory>/modules
4. $ make KERNEL_SRC=<path-to-your-Linux-kernel-directory>
5. Copy altera_rpde.ko and altera_epde.ko into SD card.
6. Use insmod command to load these 2 kernel modules on SOCFPGA terminal.

```

Building Throughput Linux application

The application is compiled with ARM cross-compiler.

```

1. $ arm-linux-gnueabi-gcc -o dmaxfer dmaxfer.c

```

Kernel Configuration

Disable MSI

The following steps illustrate how to disable MSI.

```

1. Go to Linux kernel top directory and type "make menuconfig"

```

2. Disable Bus support→Message Signaled Interrupts (MSI and MSI-X)
3. Save and exit
4. Recompile kernel

Enable MSI

The following steps illustrate how to enable MSI.

1. Go to Linux kernel top directory and type "make menuconfig"
2. Enable Bus support→Message Signaled Interrupts (MSI and MSI-X)
3. Enable Bus support→PCI host controller drivers→Altera PCIe controller→Altera MSI-to-GIC support
4. Save and exit
5. Recompile kernel

PCI Related Linux Commands

Here is a list of PCI related Linux commands.

1. lspci is the standard tool to query the devices connected to any PCI compatible bus. Type "lspci --help" for more details.
\$ lspci -v
\$ lspci -x
2. To list vendor and device IDs
\$ cat /proc/bus/pci/devices
3. To access the PCI resources on platform
\$ cat /sys/devices/pci0000:00

Using PCIe RP Simple Design

The prebuilt SD image is using full design by default. User needs to replace the *.rbf and *.dtb in the SD card.

1. Untar cv_soc_rp_simple_design.tar.gz tar xzf cv_soc_rp_simple_design.tar.gz
2. Copy cv_soc_rp_simple_design/output_files/pcie_rp_ed_5csxfc6.rbf to SD card and rename it to soc_system.rbf. Note, the default RBF filename is soc_system.rbf (refer to uboot.script file)
3. Copy cv_soc_rp_simple_design/ pcie_rp_ed_5csxfc6.dtb to SD card and rename it to socfpga.dtb. Note, default DTB filename is socfpga.dtb

[Back to Top](#)

Open Issues

When we encounter the "altera_hps2fpga_bridge fpgabridge.3: regmap for altr,rst-mgr lookup failed." we can open the dts files and add the syscon on the line of hps_0_rstmgr compatible = "altr,rst-mgr-15.0", "altr,rst-

mgr";**syscon**";**syscon**";to fix the warning message.

[Back to Top](#)

References

1. [Cyclone V Hard IP for PCI Express User Guide](#)
2. [GSRD User Manual](#)
3. [Arria V PCIe Root Port with MSI Example Design](#)

[Back to Top](#)