Cyclone V SoC
PCI-Express Root Port Example Design

Application Note

7/1/2013
Table of Contents

1 Revision History .................................................................................................................. 4
2 Overview ............................................................................................................................... 5
  2.1 GSRD ............................................................................................................................. 5
3 Hardware and Software Packages ....................................................................................... 6
  3.1 GSRD ............................................................................................................................. 6
  3.2 Release package .............................................................................................................. 6
  3.3 Hardware ....................................................................................................................... 7
  3.4 Software ....................................................................................................................... 7
4 Setting up the example design platform ............................................................................. 8
  4.1 Prepare SD card .............................................................................................................. 8
  4.2 Demo: Setup board and boot ....................................................................................... 8
5 Hardware Development Flow ............................................................................................. 11
  5.1 Qsys Design and Generation ....................................................................................... 11
  5.2 Quartus Compilation .................................................................................................... 11
  5.3 Hardware Validation using System Console ................................................................. 13
    5.3.1 Reading System ID ................................................................................................. 13
    5.3.2 On chip Memory Data Filling ............................................................................... 14
    5.3.3 Setup mSGDMA for Transfer .............................................................................. 15
6 Software Development Flow ............................................................................................. 17
  Figure 8: Overview Diagram .............................................................................................. 17
  6.1 Environment setup based on GSRD ............................................................................ 17
  6.2 Yocto build ................................................................................................................... 18
    6.2.1 Build components with yocto ............................................................................. 18
    6.2.2 Build sd_image .................................................................................................... 18
  6.3 Individual components build ........................................................................................ 18
    6.3.1 Build kernel from Rocketboards.org .................................................................. 18
    6.3.2 Device Tree .......................................................................................................... 19
    6.3.3 Host system driver and End point device driver .................................................. 19
    6.3.4 Throughput Linux application ............................................................................. 19
7 Open Issues ....................................................................................................................... 20
  7.1 FB 118697 – PCIe RP Subsystem Vendor and Device ID do not show up in PCI config 20
  7.2 FB 130072 – PCIe RP TXS reading at very low speed ................................................ 20
8 References ......................................................................................................................... 21
## Revision History

<table>
<thead>
<tr>
<th>Revision</th>
<th>Date</th>
<th>Summary of Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>07/01/13</td>
<td>Initial Release</td>
</tr>
</tbody>
</table>
2 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, 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.

Figure1: Altera SoC RootPort to Endpoint connectivity through a PCIe Link.

2.1 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.

SoC EDS: https://www.altera.com/download/software/soc-eds
3 Hardware and Software Packages

3.1 GSRD

GSRD used is based on version 13.0.0.0 of the “kit installation”. The example design is built and tested with QuartusII 13.0sp1.

Use the following link to install the Cyclone V SoC development kit, “Kit installation”.


The following is a link to the “SoC Golden System Reference Design User Guide”


- Step through “Prerequisites” and “Setting Up the SoC FPGA CV Development Kit”
- Understand the “Hardware Development Flow” and “Software Development Flow”

3.2 Release package

The release package (compiled with QuartusII 13.0sp1) includes

- Cyclone V SoC PCIe RootPort User Guide
- FPGA Qsys design

These FPGA Qsys designs are located on the Altera Wiki page:

http://www.alterawiki.com/wiki/CV_SoC_PCIe_Root_Port

  - ARM Root Port system
    - Three RootPort FPGA designs are provided depending on utilization and/or performance requirements.
      - The “small” design uses the least logic utilization and is good for applications where the highest throughput is not a priority.
      - The "simplified" design enhances the “small” design and allows for more outstanding transactions thus increasing throughput but also increasing logic utilization.
      - The “full” design contains a DMA to showcase throughput to and from RootPort and Endpoints.

- cv_soc_rp_simplefied_design.tar.gz

  This design provides Jtag Master access to FPGA logic. It supports a maximum burst count of 64 to the PCIe Txs data path. An Endpoint DMA is needed to initiate bulk data transfers. A DMA in the HPS can also be used to initiate transfers to PCIe Endpoint. Logic utilization is about 12,000 ALMs.

- cv_soc_rp_small_design.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 6,000 ALMs.

- 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 Txs data path. This connection is used to perform PCIe throughput measurements. Logic usage is about 19,000 ALMs.

- DMA End Point device
  - c5gx_pci_gen1x4.tar.gz

Software package
The software packages are located on rocketboards.org:
http://www.rocketboards.org/foswiki/Projects/PCIeRootPort

Please refer to the README for more details.

- Example Design Patch
  - linux-socfpga
    - Patch if building kernel with branch "socfpga-3.7-ghrd"
      http://rocketboards.org/gitweb/?p=linux-socfpga.git;a=summary
  - yocto
    - Patch to yocto build extracted from GSRD kit installation
      linux-socfpga-cv-soc-gsrd-13.0-cb-RC10-src.bsx

- Prebuilt Image
  - components
    - All the prebuilt components needed for testing and building the
      sd_image
  - sd_image
    - SD 4G image that combines all the require components for
testing, including Pre-loader, U-Boot, rfs, drivers and application.

- Source Code

3.3 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
- Other setup, refer “SoC Golden System Reference Design User Guide”

3.4 Software
Software required for this example design

- sd_image.bin to be programmed into SD card
- Other setup, refer “SoC Golden System Reference Design User Guide”
4 Setting up the example design platform


4.1 Prepare SD card

The release package includes an sd_image.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/](http://sourceforge.net/projects/win32diskimager/)) is used. Follow the user guide and program sd_image.bin into SD card.

4.2 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 “Golden System Reference Design User Guide”.

1. Insert the SD card programed from Section 4.1.
2. Power up the system.
3. Use Quartus Programmer to configure Cyclone V SoC with pcie_rp_ed_5csxfc6.sof.
4. If setting up the Cyclone V FPGA Endpoint, configure using c5gx_pcie.sof.
5. Start a serial terminal on the host PC to communicate with the Linux target.
6. Press the warm reset button.
7. Observe Preloader -> U-Boot -> Linux booting on the serial terminal.
8. At the Linux Kernel, you should see:
   - socfpga_cyclone5 login: root
   - root@socfpga_cyclone5:~# modprobe altera_rpde
   - root@socfpga_cyclone5:~# modprobe altera_epde
   - root@socfpga_cyclone5:~# ./altera/dmaxfer

Note: If just performing the CycloneV Soc PCIe RootPort demo, sections 5 and 6 can be skipped. Sections 5 and 6 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
5 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 13.0sp1 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
- HPS software handoff files
- SOF binary

5.1 Qsys Design and Generation

pcie_rp_ed_5csxc6.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.

sub_msgdma.qsys is the Qsys subsystem design file that consists of the modular SGDMA. The sub modules of SGDMA is fully parameterizable into any combinational of Avalon interfaces. This subsystem is instantiated by the Qsys top level design file.

The user is only required to perform Qsys generation for pcie_rp_ed_5csxc6.qsys. Generated RTL will be contained in pcie_rp_ed_5csxc6 folder.

5.2 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_5csxc6.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.

5.3 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.

![System Console: Sourcing script_pcie_ed.tcl](image)

5.3.1 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.
5.3.2 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.

```tcl
% 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
```

Figure 7: Successful sysid_read command
5.3.3 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.

```plaintext
% 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: 0xdeaddead, from addr: 0x4020
read DMA Descriptor Read Addr: 0xdeaddead
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: 0x00000000
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
6 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.

![Diagram](image)

Figure 8: Overview Diagram

6.1 Environment setup based on GSRD

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


Get Linux BSP of
- Linux kernel
- u-boot
- The tool chain

Export variable

- export CROSS_COMPILE=arm-none-linux-gnueabi-
- export LANG=C
- export ARCH=arm
- PATH=</u-boot directory>/tools:/$PATH

### 6.2 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 6.3 if following steps 6.2.1 and 6.2.2.

#### 6.2.1 Build components with yocto

1. Extract source from GSRD kit installation
4. $ ./altera-linux/bin/install_altera_socfpga_src.sh ./yocto
5. $ patch –p1 –i altera_pcie_yocto.patch
6. $ source ./altera-linux/init ./build
7. $ bitbake altera-ed-pcierp
8. Copy and replace all required components into SD Card

#### 6.2.2 Build sd_image

1. Copy preloader-mkpimage.bin and u-boot.img to temporary folder
2. $ sudo tar xzf ./yocto/build/tmp/deploy/images/altera-ed-pcierp-socfpga_cyclone5.tar.gz -C /tmp/rootfs
3. $ sudo sh ./altera-linux/bin/make_sdimage.sh -k uImage,socfpga.dtb -p preloader-mkpimage.bin -b u-boot.img -r /tmp/rootfs -o /tmp/sd_image.bin -g 3400M
4. Follow section 4.1 to program SD card

### 6.3 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.

#### 6.3.1 Build kernel from Rocketboards.org

1. Download linux-socfpga.git from rocketboards.org branch "socfpga-3.7-ghrd"
3. $ git clone [http://git.rocketboards.org/linux-socfpga.git](http://git.rocketboards.org/linux-socfpga.git) linux-socfpga
4. $ cd linux-socfpga
5. $ git checkout –t –b socfpga-3.7-ghrd origin/socfpga-3.7-ghrd
6. Copy altera_pcierp_driver.patch and altera_pciede_driver.patch to the root source tree
7. $ patch –p1 –i altera_pcierp_driver.patch
8. $ patch –p1 –i altera_pciediag_driver.patch
9. `$ make socfpga_defconfig` 
10. `$ make menuconfig` 
11. (i) Enable PCI with "PCI support" 
12. (ii) Enable ALTERA_PCIE_RP with "Altera PCIe Root Port Driver" 
13. (iii) Enable E1000E with "Intel(R) PRO/1000 PCI-Express Gigabit Ethernet support" 
14. save and exit 
15. `$ make uImage LOADADDR=0x8000` 
16. Copy and replace ulmage into SD Card

6.3.2 Device Tree
Since Device Tree Generator (DTG) is not ready in Quartus II 13.0, a patch to GSRD DTS is provided to build DTS with RP device. Not all GSRD DTS devices are used, therefore replace the DTS from ZIP, which has minimum parameters require for PCIe RP.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td><code>$ sopc2dts –gui &amp;</code></td>
</tr>
<tr>
<td>2.</td>
<td>Load pcie_rp_ed_5csxc6.sopcinfo and pcie_rp_ed_5csxc6_board_info.xml, then generate DTS socfpga.dts file.</td>
</tr>
<tr>
<td>3.</td>
<td>Copy socfpga.dts to <code>\linux-socfpga\arch\arm\boot\dts\</code></td>
</tr>
<tr>
<td>4.</td>
<td><code>$ make socfpga.dtb</code></td>
</tr>
<tr>
<td>5.</td>
<td>Copy and replace socfpga.dtb into SD card.</td>
</tr>
</tbody>
</table>

6.3.3 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.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Copy altera_rpde.* and altera_epde.* to <code>\linux-socfpga\drivers\misc\</code></td>
</tr>
<tr>
<td>2.</td>
<td>Update Makefile to include both drivers</td>
</tr>
<tr>
<td>3.</td>
<td><code>$ make modules</code></td>
</tr>
</tbody>
</table>

6.3.4 Throughput Linux application
The application is compiled with ARM cross-compiler.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td><code>$ arm-none-linux-gnueabi-gcc –o dmaxfer dmaxfer.c –static</code></td>
</tr>
</tbody>
</table>
7 Open Issues

7.1 FB 118697 – PCIe RP Subsystem Vendor and Device ID do not show up in PCI config

Problem: RP Subsystem Vendor and Device ID does not show up in PCI configuration space.
Implication: Customer/OEM designed RP system will not able to load driver that come with Subsystem Vendor and Device ID.
Workaround: Alternatively use Vendor and Device ID only
Status: Quartus 13.1

7.2 FB 130072 – PCIe RP TXS reading at very low speed

Problem: Reading TXS port is very low at ~200MB/s compare writing at ~700MB/s.
Implication: Customer will experience low read speed especially when using DMA read.
Workaround: None
Status: In progress
8 References

1. Cyclone V Hard IP for PCI Express User Guide