GSRD v13.1 - Programming FPGA from HPS

Last modified by RaduBacrau on 26 Jun 2014 - 21:43 - r23
Tagged FPGA, GSRD, Linux, Programming

Table of Contents

Overview

The FPGA can be configured (also known as 'programmed') in several ways:

- From an external configuration flash memory,
- With the Quartus Programmer tool,
- From HPS software.

This page presents the different FPGA configuration options from HPS software:

- From Preloader
- From U-boot
- From Linux.

For the GSRD the selected method of programming the FPGA is from U-boot, with the image stored on the SD card.

This page uses the Cyclone V GHRD (Golden Hardware Reference Design) as an example, but similar instructions can be used for the Arria V GHRD.

Note: Before re-programming the FPGA fabric, make sure that the FPGA2HPS bridges (f2sdram, axi) are disabled, and that there is no software on HPS that may access the FPGA. This includes shutting down applications that access soft IP and also unloading any soft IP Linux kernel modules. Failure to do so will cause the system to behave in a non-deterministic way and most likely it will crash.

MSEL Settings

The FPGA configuration depends on the MSEL switch settings. The GSRD suports the following settings:

MSEL[4..0]	SW3	Setting	Compression	POR Delay
00000	1:ON-2:ON-3:ON-4:ON-5:ON	FPPx16	No	Fast

Note that, at least as of the 13.1 release, the SoC EDS produces compressed files. The ghrd Makefile is responsible for this, and recommends the following MSEL settings:

MSEL[4..0]	SW3	Setting	Compression	POR Delay
01010	1:ON-2:OFF-3:ON-4:OFF-5:ON	FPPx32	Yes	Fast

Note that the limitation on the number of supported MSEL settings is derived from the fact that the MAX V device on the board shares the same MSEL pin settings as the Cyclone V FPGA device. This limitation is not applicable if MAX V is not needed, or if other board designs are used.

See the following documents for more details about MSEL settings:

- Cyclone V Booting and Configuration
- Arria V Booting and Configuration
- Cyclone V FPGA Manager
- Arria V FPGA Manager

See Compile FPGA Design for details on how to convert the .sof file to the .rbf file format required when HPS configures the FPGA.

FPGA Configuration from Preloader

The FPGA can be configured from the Preloader, currently with the following limitations:

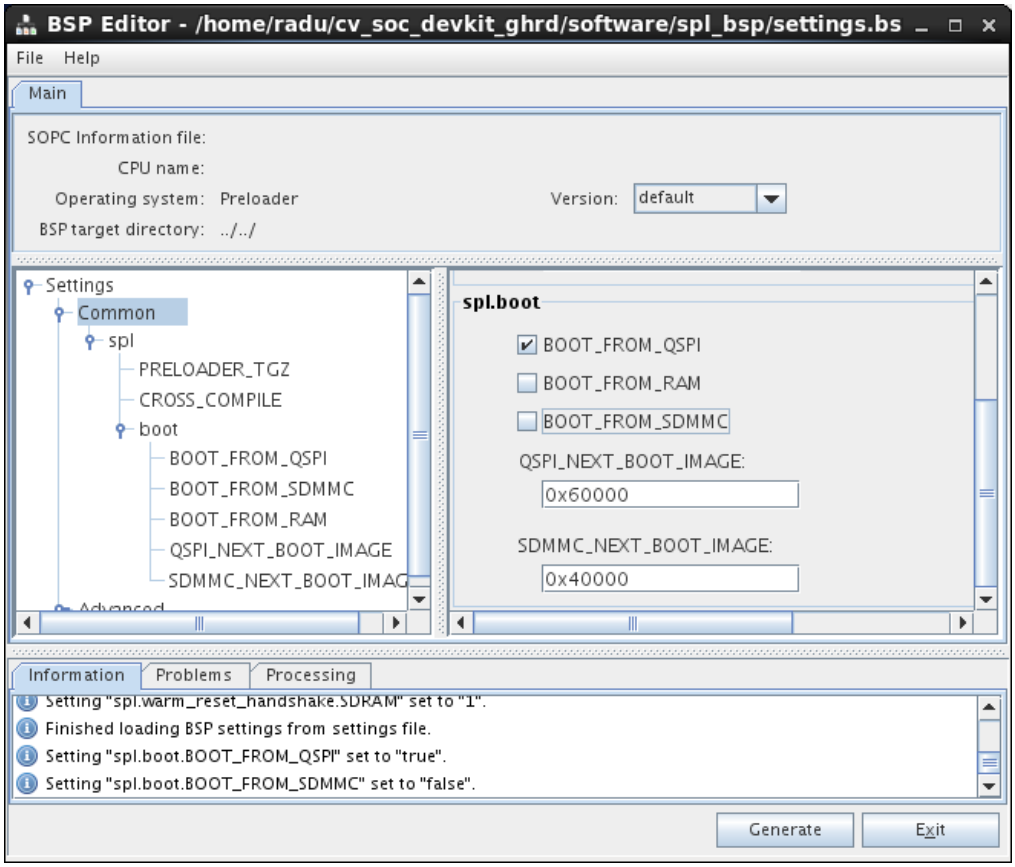
- The next boot stage (U-boot) needs to be located in QSPI,
- The FPGA image is located in QSPI too.

The following steps are required in order to program the FPGA from Preloader:

1. Create the FPGA Configuration file in the .rbf (Raw Binary File) format as described in the Compiling FPGA Design.

2. Generate the Preloader based on the Quartus handoff folder as described in Generating and Compiling the Preloader. Make sure to enable booting from QSPI instead of SD/MMC before clicking **Generate** as shown in the figure below:

- Check **BOOT_FROM_QSPI**
- Unckeck **BOOT_FROM_SDMMC**



3. Compile the generated Preloader as described in Generating and Compiling the Preloader. This will ensure that all the Preloader source code is available (some it is extracted from an archive the first time the Preloader is built).

4. Edit the file **~/cv_soc_devkit_ghrd/software/spl-bsp/uboot-socfpga/include/configs/socfpga_common.h** to:

- Define the macro **CONFIG_SPL_FPGA_LOAD** (undefined by default)
- If necessary, update the macro **CONFIG_SPL_FPGA_QSPI_ADDR** to point to a different QSPI address for the FPGA image (the default is 0x800000):

```
#define CONFIG_SPL_FPGA_LOAD
#define CONFIG_SPL_FPGA_QSPI_ADDR   (0x800000)
```

5. Recompile the Preloader

```
$ ~/altera/13.1/embedded/embedded_command_shell.sh
$ cd ~/cv_soc_devkit_ghrd/software/spl_bsp
$ make
```

6. Write the Preloader, FPGA image and U-boot image to QSPI as shown in the table below:

File	QSPI Address

preloader-mkpimage.bin	0x000000
u-boot-socfpga_cyclone5.img	0x060000
soc_system.rbf	0x800000

Use one or a combination of methods as described in QSPI Programming.

7. Configure board to boot from QSPI:

Jumper	Setting
J28	right shorted
J29	left shorted
J30	left shorted

8. Power cycle or reset the board. The following will happen:

- BootROM will the load Preloader from QSPI and run it,
- Preloader will load the FPGA image from QSPI and confiugre the FPGA,
- Preloader will load U-boot from QSPI and run it,
- U-boot will load Linux from SD card

See Booting from QSPI for instructions on how to boot Linux from QSPI too. You will need to move the FPGA QSPI location from 0x800000 to 0x1800000 to accommodate the Linux layout.

FPGA Configuration from U-boot

U-boot can configure the FPGA by using an .rbf image that is first loaded into the SDRAM.See Compiling FPGA Design for instructions on how to obtain the rbf file.

The following sample commands show how to load the FPGA image from the SD card into memory, then use it to program the FPGA:

```
# fatload mmc 0:1 $fpgadata soc_system.rbf
# fpga load 0 $fpgadata $filesize
```

Notes:

- **soc_system.rbf** is the name of the FPGA configuration file, in Raw Binary File format.
- **$fpgadata** is an U-boot environment variable pointing to an empty SDRAM area to be used to store the FPGA image
- **$filesize** is an U-boot environment variable that is automatically set by the fatload command to the size of the file that was just loaded

If the hardware design is utilizing the bridges between HPS and FPGA, and additional command needs to be run in order to enable the bridges:

```
# run bridge_enable_handoff
```

Note that in order for the bridges to be correctly enabled, the Preloader needs to have been generated based on the handoff information associated to the hardware design.

The command consists of the following instructions:

```
mw $fpgaintf $fpgaintf_handoff
mw $fpga2sdram $fpga2sdram_handoff
go $fpga2sdram_apply
mw $axibridge $axibridge_handoff
mw $l3remap $l3remap_handoff
md $fpgaintf
md $fpga2sdram
md $axibridge
```

The following environment variables are set to the corresponding HPS registers as shown in the table below:

Variable	Register

fpgaintf	SYSMGR.FPGAINTF.MODULE
fpga2sdram	SDR.CTRLGRP.FPGAPORTRST
axibridge	RSTMGR.BRGMODRST
l3remap	L3REGS.REMAP

If the new FPGA image has a different set of bridges enabled than the one that was used to generate the current Preloader, please use the following commands instead:

1. For HPS peripheral controller to FPGA interface:

```
mw $fpgaintf <my_fpgaintf_new_value>
```

2. For FPGA to SDRAM bridge:

```
mw $fpga2sdram <my_fpga2sdram_new_value>
go $fpga2sdram_apply
```

3. For AXI bridges (HPS2FPGA, LWHPS2FPGA, FPGA2HPS)

```
mw $axibridge <my_axibridge_new_value>
mw $l3remap <my_l3remap_new_value>
```

To figure out the required values, please refer to address map info pages at

- Cyclone V Register Map
- Arria V Register Map

To disable all bridges, run the following command:

```
run bridge_disable
```

U-Boot FPGA Programming Error Messages

If any failure during FPGA programming in U-Boot, it will return error message with value.
These value will represent the stages where the programming failed. Here are the numbers and error stages.

Error No	Description
-1	The FPGA is not able to enter reset phase after FPGA reset request being made at FPGA Manager
-2	The FPGA is not able to enter configuration phase after FPGA reset release request being made at FPGA Manager
-3	The FPGA is having config error after enable AXI configuration at FPGA Manager
-4	The FPGA is having timeout from entering config done phase after enable AXI configuration at FPGA Manager
-5	The FPGA is having timeout from getting dclkcnt done signal after enable the dclkcnt delay.
-6	The FPGA is having timeout from entering init phase or user mode after disable AXI configuration at FPGA Manager
-7	The FPGA is having timeout from getting dclkcnt done signal after enable the dclkcnt delay. This is happen after FPGA reaching init phase or user mode
-8	The FPGA is having timeout from getting into final stage which is user mode

See the following known issue affecting designs that have the FPGA2HPS SDRAM Interface enabled:

Issue	Headline	Description
158537	Updating U-Boot FPGA2SDRAM driver	The FPGA2HPS SDRAM interface (if enabled in hardware design) is not correctly configured by U-Boot which may cause HPS to freeze. Use U-Boot from branch socfpga_v2013.01.01-rel to fix this issue. The FPGA2HPS SDRAM interface can only be safely enabled from U-Boot, not from Preloader or Linux. (The issue is fixed in the current GSRD release, but not in the Linux 3.9)

FPGA Configuration from Linux

Linux can configure the FPGA by using an .rbf image file.See Compiling FPGA Design for instructions on how to obtain the rbf file.

The image file must be available to Linux, such as in the root filesystem or loaded up through networking or it could be in a mounted partition on the SD card.

The following FPGA programming related commands are available:

Command	Description
cat /sys/class/fpga/fpga0/status	Determine the current FPGA state
cat /sys/class/fpga-bridge/fpga2hps/enable cat /sys/class/fpga-bridge/hps2fpga/enable cat /sys/class/fpga-bridge/lwhps2fpga/enable	Determine bridge status
echo 0 > /sys/class/fpga-bridge/fpga2hps/enable echo 0 > /sys/class/fpga-bridge/hps2fpga/enable echo 0 > /sys/class/fpga-bridge/lwhps2fpga/enable	Disable enabled bridges
dd if=soc_system.rbf of=/dev/fpga0 bs=1M	Program FPGA
echo 1 > /sys/class/fpga-bridge/fpga2hps/enable echo 1 > /sys/class/fpga-bridge/hps2fpga/enable echo 1 > /sys/class/fpga-bridge/lwhps2fpga/enable	Enable needed bridges

The following sample procedure shows how to program the FPGA from Linux, when using the GSRD:

1. Boot Linux as described in Booting Linux but stop at the U-boot prompt by pressing any key when asked.

2. At U-boot console, boot Linux without configuring FPGA, with the following commands:

```
# run mmcload
# run mmcboot
```

3. Login into Linux by entering 'root' when asked for user name.

4. From Linux console, display the status of the FPGA

```
cat /sys/class/fpga/fpga0/status
configuration phase
```

5. From Linux console, display the status of the bridges:

```
# cat /sys/class/fpga-bridge/*/enable
0
0
0
```

6. Mount the SD card to have access to the GSRD RBF file

```
# mkdir -p sdcard
# mount /dev/mmcblk0p1 sdcard
# ls -la sdcard/soc_system.rbf
 -rwxr-xr-x    1 root     root       2256304 Sep 13 13:34 sdcard/soc_system.rbf
```

7. Program the FPGA

```
# dd if=sdcard/soc_system.rbf of=/dev/fpga0 bs=1M
2+1 records in
2+1 records out
```

8. Check that FPGA was configured

```
# cat /sys/class/fpga/fpga0/status
user mode
```

9. Enable the needed bridges

```
# echo 1 > /sys/class/fpga-bridge/fpga2hps/enable
# echo 1 > /sys/class/fpga-bridge/hps2fpga/enable
# echo 1 > /sys/class/fpga-bridge/lwhps2fpga/enable
```

Note that the GSRD was designed so that the FPGA is programmed from U-boot before loading Linux. Therefore, even if the FPGA is now programmed, not all the GSRD functionality is available. The above procedure was provided only to illustrate loading FPGA from Linux.

GSRD FPGA Configuration

This section provides some details about how the FPGA is configured in the GSRD.

In the GSRD case, the FPGA is configured from U-boot with the help of a script. This scheme ensures that the Linux boot succeeds even if the FPGA image is not present on the SD card. If the FPGA image is not present, Linux will still boot, but the FPGA will need to be configured using another method.

The following U-boot environment variables are used to define the U-boot procedure for programming FPGA:

```
bootcmd=run callscript; run mmcload; run mmcboot
callscript=if fatload mmc 0:1 $fpgadata $scriptfile;then source $fpgadata;  else echo Optional boot script not foun
d. Continuing to boot normally;
scriptfile=u-boot.scr
```

Note that the U-boot environment variables can be seen at the U-boot console by running the 'printenv' command.

The script file **u-boot.scr** was obtained with the following procedure:

1. On host PC, create text file **u-boot.txt**with the following contents:

```
fatload mmc 0:1 $fpgadata soc_system.rbf;
fpga load 0 $fpgadata $filesize;
run bridge_enable_handoff;
run mmcload;
run mmcboot;
```

2. Add the U-boot header to the **u-boot.txt** script file to create the **u-boot.scr** file:

```
$ ~/cv_soc_devkit_ghrd/software/spl_bsp/uboot-socfpga/tools/mkimage  -A arm -O linux -T script -C none -a 0 -e 0 -n
"My script" -d u-boot.txt u-boot.scr
```

3. Write the file on the SD card:

```
# mkdir -p sdcard
# mount /dev/mmcblk0p1 sdcard
# cp  u-boot.scr sdcard/
# umount sdcard
```

Anhänge (2)

bsp-boot-from-qspi.png (46.92K)
Version 1 uploaded by Radu Bacrau on 07 Oct 2013 - 19:18

bsp-open.png (14.29K)
Version 1 uploaded by Radu Bacrau on 07 Oct 2013 - 19:34