XAPP1107 (v1.0) January 16, 2009

Getting Started Using Git

Author: Kris Chaplin

Summary

Xilinx provides many offerings for customers to use Linux with Xilinx processor architectures. In addition to downloading kernel sources from supported Xilinx third party partners, it is also possible to download kernel sources from the Xililnx Linux Git Tree, which is a distributed version control system commonly used for distributing Linux kernel sources.

This application note describes a method for setting up a user environment for building Linux kernels using the Xilinx Git tree.

Hardware and Software Requirements

The user must be running a Linux operating system, which is compatible with the Xilinx ISE® and EDK tools. The processes in this documentation are based on running Linux Redhat 4 with EDK and ISE 10.1. Xilinx EDK and ISE are required for generating custom board hardware images.

Proxy Servers and Network Connectivity

Usually, in a corporate environment, proxy server access is given to the internet. Many UNIX command line tools support the traversal of a proxy server with the correct environment variables. Set the following variables from a bash shell to allow proxy traversal:

```
[bash]$ export http_proxy=http://xir-proxy:6900
[bash]$ export HTTP_PROXY=http://xir-proxy:6900
[bash]$ export ftp_proxy=http://xir-proxy:6900
```

The name and port of the local http proxy server should be available from your IT department. The rest of this application note assumes that the proxy environment variables have been correctly defined, if required. This is *not* required if there is a direct connection to the internet.

Downloading and Installing the Git Source Control Utilities

To gain the maximum benefit from the Xilinx Git repository, it is recommended that the Git tools be installed so that the user can check out the latest version of the `git.xilinx.com` Linux source tree. Because Git is a rapidly evolving tool-chain, it is recommended that the latest version be installed.

It is possible to download the Xilinx-provided kernel source code without using Git, using the *snapshot* feature. For this alternative approach, go directly to the "Downloading the Kernel Source Files and MLD," page 2 section.

The Git source files are available for download via `http://git.or.cz`. The home page should publish the latest version of the Git source package. Version v1.5.6.1, is used in this example. The files may be downloaded using a web browser or by typing the following command in a bash window:

```
[bash]$ wget http://kernel.org/pub/software/scm/git/git-<version>.tar.gz
```

For example, the following steps will get the tar file, extract the tar file, change directories into the directory containing the source files, confirm the Git version, and build the source.

1. Retrieve the tar file by using the following command:

```
[bash]$ wget http://kernel.org/pub/software/scm/git/git-1.5.6.1.tar.gz
```

2. Extract the files using the tar:

```
[bash]$ tar zxf git-1.5.6.1.tar.gz
```

3. Change to the directory containing the source and scripts, and build the source files by using the following commands:

```
[bash]$ cd git-1.5.6.1
[bash]$ make prefix=/usr all
```

Note: The **prefix=/usr** command can be changed to compile and install the tools into a different location. If the location is not writable, then root access is required. For example, the user may want to install the tools to /home/username/bin. Be sure to change the prefix line for both the **make all** and the **make install** commands.

4. Once the compilation is complete, the tools can be installed into the /usr directory. While logged in as root, install the tools using the following commands:

```
[bash]$ su
Password:<user_password>
[bash]# make prefix=/usr install
```

5. Enter **git --version** to confirm correct installation of the tools:

```
[bash]$ git --version
```

The git version displays as follows:

```
git version <version_number>
```

Downloading the Kernel Source Files and MLD

To download the kernel files, the name of the Git server and the name of the project to be downloaded is needed.

Xilinx provides a browser-enabled Git server at http://git.xilinx.com. Figure 1 shows the display from the Git server:

Project	Description	Owner	Last Change	
device-tree.git	Device tree generator (MLD), ...	wre	15 hours ago	summary | shortlog | log | tree
gen-mhs-devtree.git	deprecated, use device-tree.git	wre	4 weeks ago	summary | shortlog | log | tree
linux-2.6-xlnx.git	Xilinx Linux Kernel, based ...	wre	3 weeks ago	summary | shortlog | log | tree
u-boot-xlnx.git	u-boot bootloader with Xilinx ...	wre	13 days ago	summary | shortlog | log | tree

X1107_01_120308

Figure 1: **Git Server**

Git Cloning Download

As shown in Figure 1, the Git server displays the time of publication and a Linux kernel Git project called linux-2.6-xlnx.git. Make a directory to contain the project, then check out the source files with the following commands:

```
[bash]$ mkdir ~/git_master
[bash]$ cd ~/git_master
[bash]$ git clone git://git.xilinx.com/linux-2.6-xlnx.git
```

As the download initializes, the following messages will be displayed:

```
Initialized empty Git repository in /<user_directory>/git_master/linux-
2.6-xlnx/.git/
remote: Generating pack...
remote: Done counting remote: 634439remote:  objects.
remote: Deltifying remote: 634439remote:  objects.
remote:  100remote: %remote:  (remote: 634439remote: /remote: 634439remote:
) done
remote: Total 634439 (delta 516391), reused 618049 (delta 500168)
Receiving objects: 100% (634439/634439), 196.72 MiB | 192 KiB/s, done.
```

```
Resolving deltas: 100% (516391/516391), done.
Checking out files: 100% (23358/23358), done.
```

The downloaded kernel source tree in the `linux-2.6-xlnx` directory is shown in Figure 2.

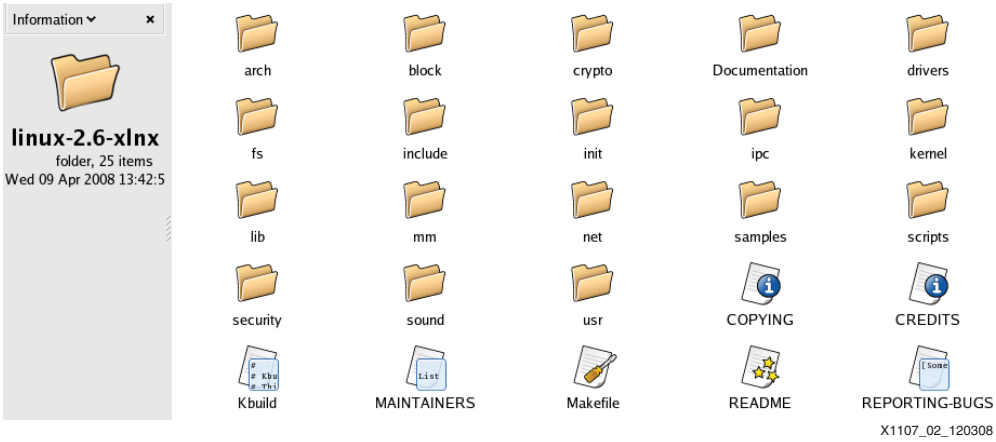

Figure 2: **linux-2.6-xlnx Directory**

Setup

To maintain the latest version of the source tree from `git.xilinx.com`, check periodically for updates to the kernel source files. Use the following command to download the updated kernel source files from the kernel root directory:

```
[bash]$ git pull
```

In addition to the Git source options, Xilinx provides bitstream and RAM disk images for the ML405 and ML507 evaluation platforms which can be downloaded from the wiki on `git.xilinx.com`. The bitstreams work with default kernel configurations within the provided Git sources to provide a quick-start for users with an ML405 or an ML507 board. This application note will assume that you have downloaded the `ramdisk.image.gz` file from the files link on the wiki page to `~/git_master/linux_support`.

Snapshot Download

If the user did not install the GIT tools, it is still possible to download the kernel sources using the snapshot feature. To download a snapshot of the source tree, select the **tree** link for each of linux-2.6-xlnx.git and device-tree, then click on **snapshot**. This will prepare and download compressed tar files (gzipped tarballs) of the most current sources. The rest of this document will assume that these files have been extracted into `/<user_directory>/git_master/linux-2.6-xlnx` and `<user_directory>/git_master/device-tree` directories respectively.

Downloading and Installing the DENX ELDK 4.1 Tool Chain

To build the kernel image for the PowerPC® 405 or the Power PC 440 processor present in Xilinx FPGAs, a GNU tool chain is needed. To create a standalone kernel build environment, the following example steps use the DENX ELDK 4.1 tool chain, which is available from `http://www.denx.de/`.

To install the DENX ELDK toolkit:

1. Go to the ELDK homepage at `http://www.denx.de/wiki/DULG/ELDK` and follow the instructions provided to download and install the development kit.
2. Download the ELDK 4.1 ISO image:

   ```
   [bash]$ wget ftp://ftp.sunet.se/pub/Linux/distributions/eldk/4.1/ppc-linux-x86/iso/ppc-2007-01-19.iso
   ```

3. As root user, create a mount folder, and mount the ISO for reading, then create a target directory for the installation of the ELDK files:

```
[bash]$ mkdir /mnt/iso
[bash]$ chmod ag+r /mnt/iso
[bash]$ mount -o loop ppc-2007-01-19.iso /mnt/iso
[bash]$ mkdir /opt/eldk
[bash]$ chown <your user_name> /opt/eldk
```

4. Exit from being the root user and install the ELDK:

```
[bash]$ cd /mnt/iso
[bash]$ ./install -d /opt/eldk
```

The system requests verification that the user wants to install the toolkit. Once the action is verified, the ELDK is installed to the specified directory.

Building the Default Kernel Image

To build a PowerPC processor kernel image for use with the reference bitstreams for the ML405 and ML507 boards:

1. Create a working directory for the project, and check out a copy of the downloaded kernel sources to that directory. For example, if the user downloaded the kernel sources to `~/git_master/linux-2.6-xlnx`, enter the following:

```
[BASH]$ cd $HOME

(ppc440/ml507) [bash]$ mkdir test_ml507
(ppc440/ml507) [bash]$ cd test_ml507
```

or

```
(ppc405/ml405) [bash]$ mkdir test_ml405
(ppc405/ml405) [bash]$ cd test_ml405

[bash]$ git clone ~/git_master/linux-2.6-xlnx
[bash]$ cd linux-2.6-xlnx
```

To perform any kernel operations using the make file, it is necessary to specify the processor architecture, and the cross compiler name. For the PowerPC processor, the architecture is `powerpc`. The cross-compiler provided by ELDK for the PowerPC processor is `ppc_4xx-`.

2. To ensure that the ELDK tools are in the user's path, the configuration script within the ELDK root directory should be executed as follows:

```
[bash]$ source /opt/eldk/eldk_init ppc_4xx
```

This sets up the path and the cross compiler environment variables, however, by default, ELDK defines the `ARCH` variable as `ppc`. PowerPC kernel development has moved to the powerpc kernel folder. Therefore, the user will build using the `powerpc`, rather than `ppc` build. This is achieved by overwriting the ELDK default by using the following command:

```
[bash]$ export ARCH=powerpc
```

3. The kernel sources are provided with default configuration settings for the ML405 and ML507 reference designs. Configure the kernel with the correct configuration for the board that is being use by using one of the following commands:

```
(ppc405/ml405) [bash]$ make 40x/virtex4_defconfig
```

or

```
(ppc440/ml507) [bash]$ make 44x/virtex5_defconfig
```

4. The default kernel images require a RAM disk to be present in the kernel source tree to act as the root file system. Copy the RAM disk image acquired from Xilinx.wikidot.com into the source tree by using the following command:

```
[bash]$ cp ~/git_master/linux_support/ramdisk.image.gz arch/powerpc/boot/
```

5. Then build the kernel image using the following command:

```
(ppc405/ml405)[bash]$ make simpleImage.initrd.virtex405-ml405
```

or

```
(ppc440/ml507) [bash]$ make simpleImage.initrd.virtex440-ml507
```

The kernel image elf file is generated as `arch/powerpc/boot/simpleImage.initrd.virtex405-ml405.elf`

or

`arch/powerpc/boot/simpleImage.initrd.virtex440-ml507.elf`.

Follow the instruction in the "Downloading Images to a Target Board," page 8 to configure the FPGA and download the kernel elf file to the board.

Building a Custom Kernel Image

The MLD files provide XPS with the necessary interface to patch the downloaded Linux kernel with the address map and interrupt connectivity of a custom system.

1. Download the MLD from the Git server using the following command:

```
[bash]$ cd ~/git_master
[bash]$ git clone git://git.xilinx.com/device-tree.git
```

This pulls down the required files for customizing the operating system and integrating it into XPS. Later, this MLD will be copied into the custom board project.

Using XPS to Patch the Kernel for a Custom Board – ML405 example

Once the `device-tree` MLD files are downloaded and installed, it is possible to generate a custom device-tree file for inclusion in a kernel build.

Creating a suitable custom XPS project for running a Linux kernel is outside the scope of this application note. However, the Base System Builder (BSB) can be used to generate an example target platform. Ensure that interrupts are enabled for key IP cores such as UART and Ethernet, as this is a kernel requirement.

1. In XPS, invoke the Base System Builder wizard. See Figure 3.

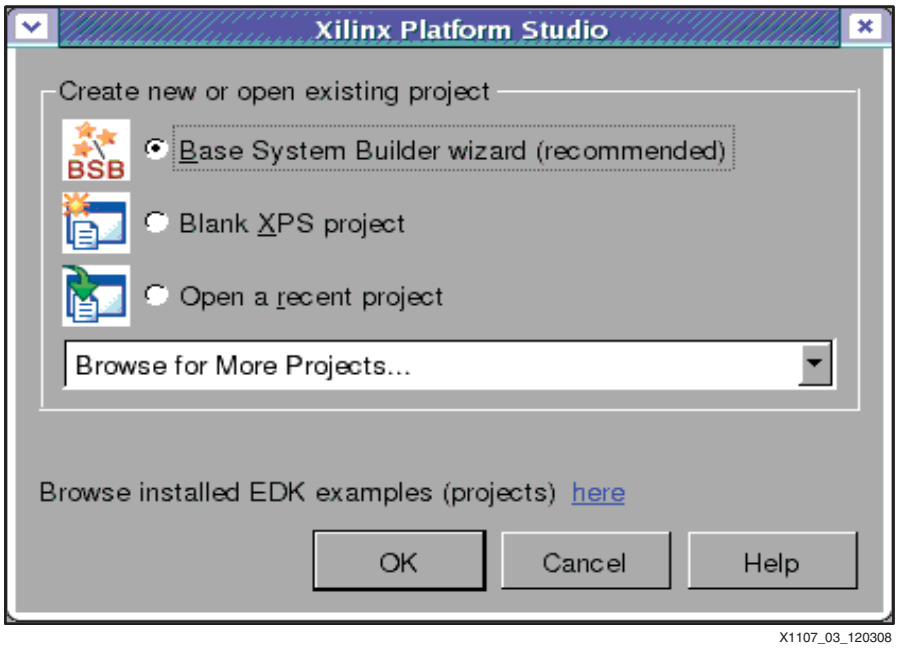

Figure 3: **Base System Builder Wizard**

2. Specify the directory in which the project should reside. Create a new design, click **Next** and select the desired board, for example, the ML405. The option and board selections are shown in Figure 4 and in Figure 5.

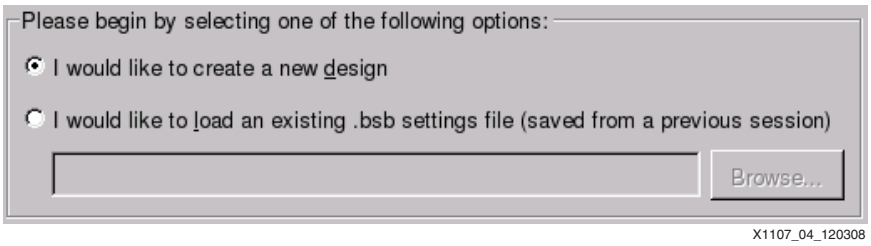

Figure 4: **Selecting an Option**

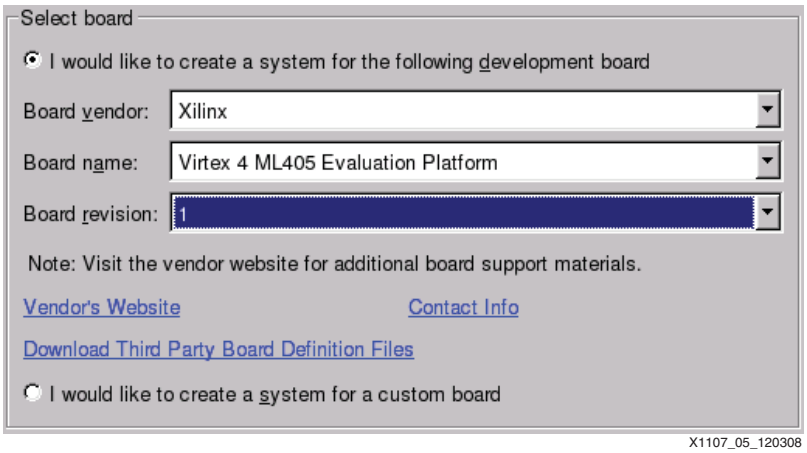

Figure 5: **Selecting a Board**

3. Select the PowerPC processor as the architecture, click **next**, then specify the required operating frequencies as shown in Figure 6.

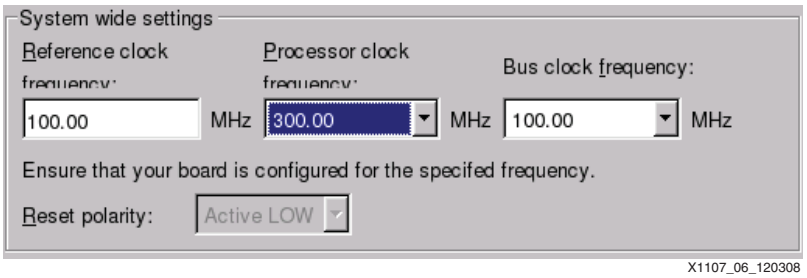

Figure 6: **Specifying the Operating Frequencies**

4. Click **Next**, then begin adding or removing peripherals as required. For cores such as UART, IIC, and Ethernet, ensure that interrupts are checked for use as shown in Figure 7.

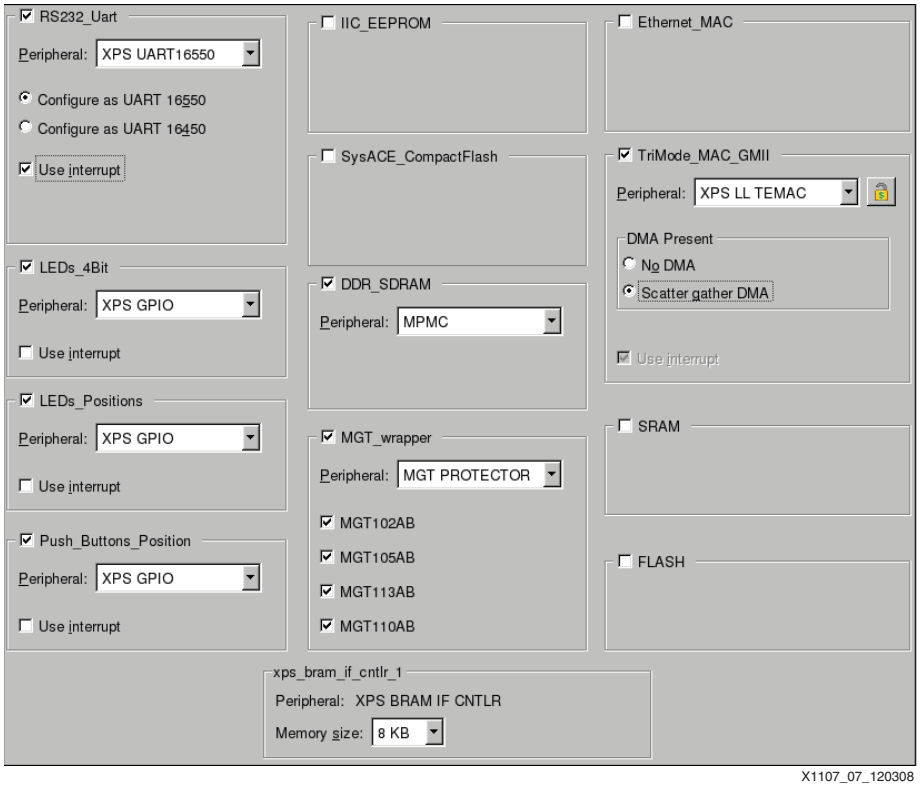

Figure 7: **Selecting Peripherals**

5. Generate the test application projects as required, and finish the wizard.

   To enable the device-tree MLD for the project, add a copy of the files that were downloaded in "Building a Custom Kernel Image," page 5 by using the following command:

   ```
   [bash]$ cp -r ~/git_master/device-tree/bsp <project_directory>/
   ```

   For XPS to notice this MLD, either close and reopen the XPS project, or click on **Project→ Rescan User Repositories**.

6. From the **Software→Software Platforms Settings→Software Platform→OS & Library Settings** menu, pull down the OS setting box, then select **device-tree** as shown in Figure 8. If device-tree is not available as an option, check the MLD installation instructions in the "Building a Custom Kernel Image," page 5.

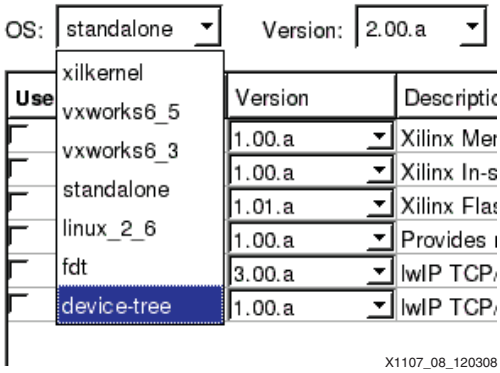

Figure 8: **Selecting the OS Options**

7. Click on the OS and Libraries option to expose the device-tree configuration as shown in Figure 9. Overwrite the kernel boot arguments as necessary. Type in the uart peripheral instance name for the XPS peripheral to act as console output.

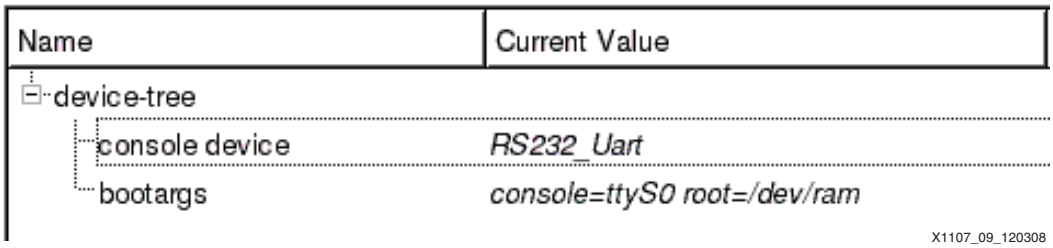

Figure 9: **device-tree Configuration**

8. On launching **Software→Generate Libraries and BSPs**, a `xilinx.dts` file is generated in the `<processor>/libsrc/device-tree/` directory.
9. Set up the environments for the Git, ELDK, and ARCH environment build as directed in "Building the Default Kernel Image," page 4, and clone a copy of the kernel source tree into the working directory.

```
[bash]$ git clone ~/git_master/linux-2.6-xlnx
```

10. Patch the kernel sources with the new device-tree file. Note that the filename is different if the user is targeting the PowerPC 405 processor or the PowerPC 440 processor (the ML405 targets the PowerPC 405 processor, whereas the ML507 targets the PowerPC processor). Use one of the following commands:

```
(ppc405/ml405) [bash]$ cp ppc405_0/libsrc/device-tree/<filename.dts>
linux-2.6 xlnx/arch/powerpc/boot/dts/virtex405-ml405.dts
(ppc405) [bash]$ cd linux-2.6-xlnx
(ppc405) [bash]$ make 40x/virtex4_defconfig
```

or

```
(ppc440) [bash]$ cp ppc440_0/libsrc/device-tree/<filename.dts> linux-2.6-
xlnx/arch/powerpc/boot/dts/virtex440-ml507.dts
(ppc440) [bash]$ cd linux-2.6-xlnx
(ppc440) [bash]$ make 44x/virtex5_defconfig
```

Kernel patching of the address map is now complete.

11. The user can continue to customize the kernel configuration and add a RAMdisk as required before building the kernel image by using one of the following commands:

```
[bash]$ cp ~/git_master/linux_support/ramdisk.image.gz arch/powerpc/boot/
(ppc405/ml405 [bash]$ make simpleImage.initrd.virtex405-ml405
```

or

```
(ppc440/ml507 [bash] make simpleImage.initrd.virtex440-ml507
```

Follow the instructions in "Downloading Images to a Target Board," page 8 to configure the FPGA and to download the kernel elf file to the board.

Downloading Images to a Target Board

Downloading Reference Bitstreams

To run a Linux kernel on a Xilinx evaluation platform or custom board:

1. Ensure that the following files and directories are available.
   - The FPGA configuration bitstream BIT file for the board.

     This file is the FPGA hardware configuration, and is used by the iMPACT programming tool. In an XPS project, this file is called `download.bit` in the `./implementation` directory of the project. Also, there are pre-built bitstreams via

`http://git.xilinx.com` that correspond to the default configurations for the ML405 and ML507 evaluation platforms.

- The kernel image elf file, `simpleImage.[initrd.]virtex405-ml405.elf` or `simpleImage.[initrd.]virtex440-ml507.elf`

  This file is the Linux Kernel in an ELF format file, and is used by the XMD microprocessor debug tool to upload to external memory on the platform using JTAG.

2. Ensure that the board is powered up with a serial RS232 cable connected to the computer and a Xilinx programming cable connected between the computer and the board. A green LED will light up to indicate that the Xilinx Parallel-IV and USB cables are connected correctly to a powered board.
3. Download the pre-generated bitstreams from `xilinx.wikidot.com` for the ML405 and ML507 reference boards.
4. Run the iMPACT tool to configure these boards. The iMPACT tool can run in either command line or GUI mode. The GUI mode is shown in the following figures. Start the tool from the command line using the following command:

   `[bash]$` **impact**

   Note: For more information about iMPACT, refer to the Help topics from within the tool.
5. Create a new project, and select the default **Configure devices using Boundary Scan** and click **Finish**. The boundary scan chain for the board should appear in the main iMPACT window. Select **Configuration→Cancel All** and then **Device Programming Properties→Cancel**. Your GUI should look similar to Figure 10 or Figure 11.

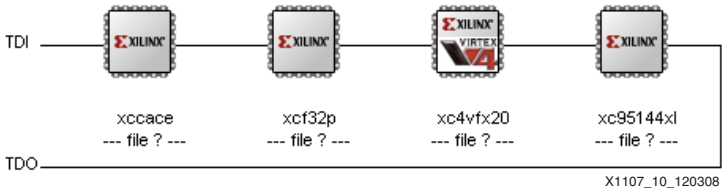

Figure 10: **ML405 Configuration Chain**

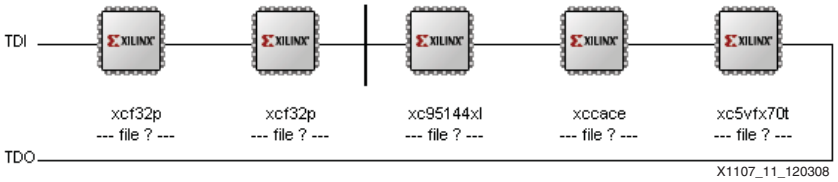

Figure 11: **ML507 Configuration Chain**

6. Double-click on the FPGA (`xc4vfx20` or `xc5vfx70t`) and select the appropriate bitstream. Click **Open**, and then **OK**. Right-click the target FPGA, and select **Program →OK** to configure the FPGA. The GUI displays that programming succeeded.

Downloading Custom XPS Project Bitstreams

From a cygwin, or bash prompt, download a custom XPS project to a connected board by using the following command:

`[bash]$` **make –f <project>.make download**

For example, with a project named `system.xmp`, the command would be:

`[bash]$` **make –f system.make download**

In addition, **Hardware >Download Bitstream** in the XPS project GUI may be selected to download an XPS project.

Downloading a Kernel Software Image

1. From a bash or EDK shell, launch the Xilinx Microprocessor Debugger (XMD) by using the following command:

   ```
   [bash]$ xmd
   ```

2. Connect to the processor by using the following command:

   ```
   XMD% connect ppc hw
   ```

   XMD will display the following message when the connection is successful:

   ```
   Connected to "ppc" target. id = 0
   Starting GDB server for "ppc" target (id = 0) at TCP port no 1234
   ```

3. Download the kernel image to the processor. For example, for the ML405, type the following command:

   ```
   XMD% dow linux-2.6-xlnx/arch/powerpc/boot/simpleImage.initrd.virtex405-ml405.elf
   ```

   XMD displays a message when the system reset is complete:

   ```
   System Reset .... DONE
   Downloading Program -- simpleImage.initrd.virtex405-ml405.elf
           section, .text: 0x00400000-0x00408b83
           section, .data: 0x00409000-0x0040ab77
           section, __builtin_cmdline: 0x0040ab78-0x0040ad77
           section, .kernel:dtb: 0x0040ad78-0x0040c868
           section, .kernel:vmlinux.strip: 0x0040d000-0x005854ef
           section, .kernel:initrd: 0x00586000-0x006f5f1f
           section, .bss: 0x006f6000-0x00702dd7
   Setting PC with Program Start Address 0x004007d8
   ```

4. Start an RS232 terminal, running at 9600 baud, and continue the processor:

   ```
   XMD% con
   ```

The kernel then outputs data on the serial port and boots accordingly.

Additional References

1. *Embedded System Tool Reference Manual*
   http://www.xilinx.com/support/documentation/sw_manuals/edk10_est_rm.pdf

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
1/16/09	1.0	Initial Xilinx release.

Notice of Disclaimer

Xilinx is disclosing this Application Note to you "**AS-IS**" with no warranty of any kind. This Application Note is one possible implementation of this feature, application, or standard, and is subject to change without further notice from Xilinx. You are responsible for obtaining any rights you may require in connection with your use or implementation of this Application Note. XILINX MAKES NO REPRESENTATIONS OR WARRANTIES, WHETHER EXPRESS OR IMPLIED, STATUTORY OR OTHERWISE, INCLUDING, WITHOUT LIMITATION, IMPLIED WARRANTIES OF MERCHANTABILITY, NONINFRINGEMENT, OR FITNESS FOR A PARTICULAR PURPOSE. IN NO EVENT WILL XILINX BE LIABLE FOR ANY LOSS OF DATA, LOST PROFITS, OR FOR ANY SPECIAL, INCIDENTAL, CONSEQUENTIAL, OR INDIRECT DAMAGES ARISING FROM YOUR USE OF THIS APPLICATION NOTE.