Arria 10 SoC Virtual Platform User Guide

2015.09.16

UG-01168

Subscribe Send Feedback

The Arria® 10 SoC Virtual Platform is based on Mentor Embedded technology and provides early software development and verification for Altera® customers.

By allowing teams to work within a virtual platform framework, software developers can gain system visibility without the high costs associated with buying multiple development boards. The virtual platform can be used to rapidly develop software in advance of actual silicon or board availability. In addition, the virtual platform allows you to port an OS from your previous architecture to the Arria 10 SoC. This development allows for early hardware driver development and partial validation in addition to creation of non-real time algorithm and application development. For Linux debug support, the GNU debugger (gdb) can be used with the Arria 10 SoC Virtual Platform.

Related Information

- **Arria 10 SoC Virtual Platform User Guide**
  For the latest updates to this document, refer to this link.
- **Arria 10 SoC Virtual Platform Version 1.0 Release Notes**
  For information regarding enhancements and known issues in the Arria 10 SoC Virtual Platform Version 1.0, refer to the release notes.
- **RocketBoards.org Forum**
  For support questions regarding the Arria 10 SoC Virtual Platform, refer to the Rocketboards.org forum.

Arria 10 SoC Virtual Platform Features

The Arria 10 SoC Virtual Platform provides the following:

- A model of the Arria 10 SoC Device
- Partial modeling of the memory map and interrupt map for Arria 10 SoC
- Simulation speed that is close to actual Arria 10 SoC speed
- Early hardware driver development and validation
- Debug with the GNU debugger

A pre-built Linux kernel is provided for use with the virtual platform and can be downloaded from **RocketBoards.org**. Refer to the "Boot Using Pre-Built Linux Kernel" sub-section in the "Booting an Operating System" section.

Related Information

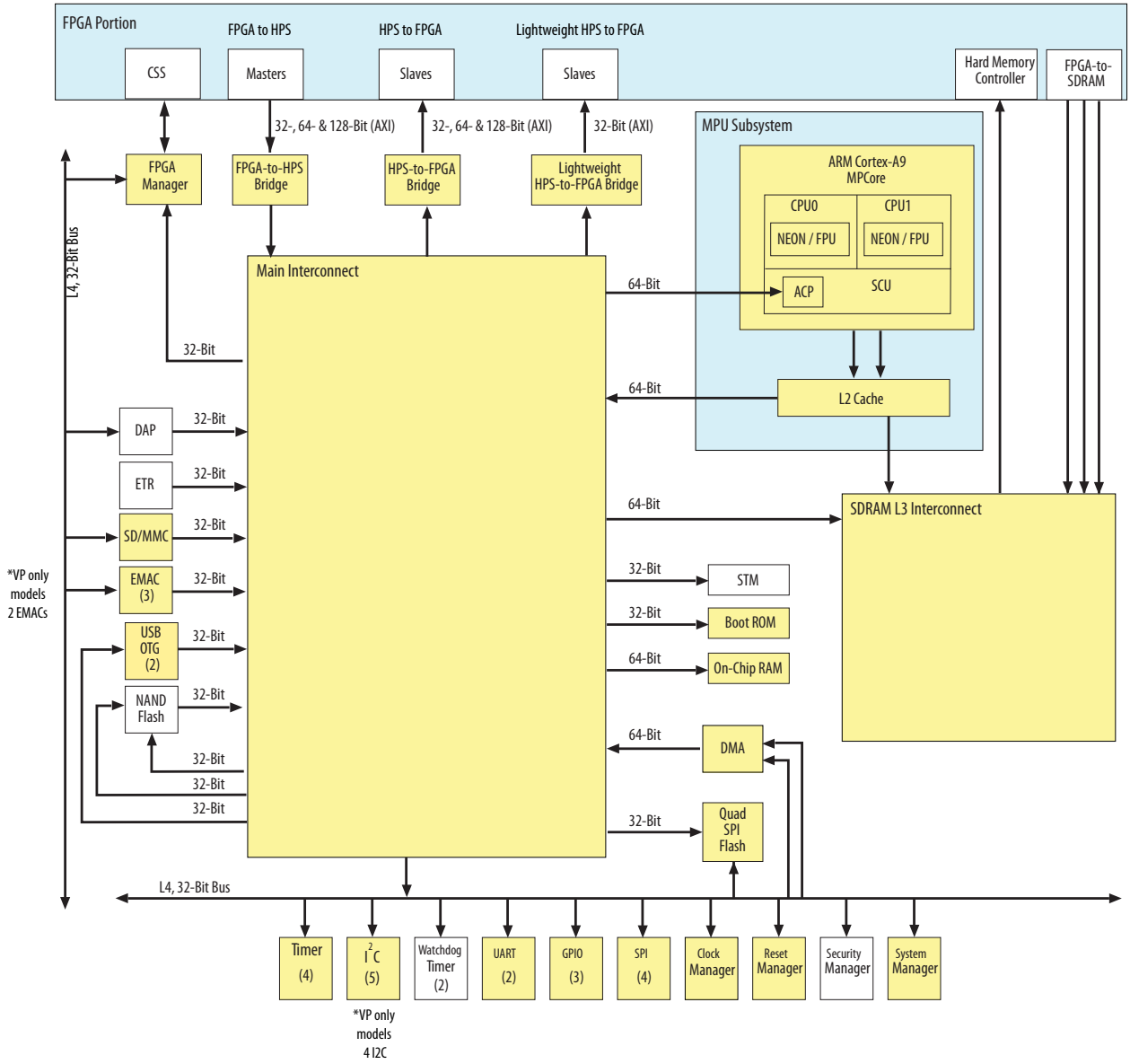
Installing and Booting a Pre-Built Linux Kernel on page 5

101 Innovation Drive, San Jose, CA 95134

Arria 10 SoC Virtual Platform Block Diagram

The figure below details the Arria 10 SoC device. Modules that are modeled in the Arria 10 SoC Virtual Platform are highlighted in yellow in the diagram.

Figure 1: Arria 10 SoC Virtual Platform Block Diagram



Note: The System Manager and Clock Manager module are currently modeled after the Arria V SoC device System Manager and Clock Manager register sets, respectively.

Table 1: Modules Modeled in Arria 10 SoC Virtual Platform

Module	Description
Dual ARM® Cortex®-A9 MPCore™ processor	Contains two Cortex-A9 with FPU support and a snoop control unit (SCU)
ARM L2 Cache (PL310)	512 KB of shared, unified cache memory
General Interrupt Controller (GIC)	Provides partial support for the interrupt map
System Interconnect (Arteris™ FlexNOC® network-on-chip (NoC))	Consists of the following: • Main level 3 (L3) interconnect • SDRAM L3 interconnect • Level 4 (L4) buses
Memory modules	• Boot ROM • On-Chip RAM
Two 16550-compatible UARTs	Each UART can interact with separate consoles
Four of the I^2C controllers	Four of the five I^2C controllers in the Arria 10 SoC are modeled and two of the controllers can optionally be used for Ethernet PHY communication
Two Ethernet controllers	Two of the three Ethernet controllers are modeled in the virtual platform
Two USB 2.0 OTG controllers	Both USBs provide support for mass storage **Note:** The USB controllers have been verified to work with the following USB storage devices: • PQI Cool Drive 350, up to 2GB • Apacer 4GB • Silicon Power 4GB • Pretec I-Disk 512MB
Two SPI master controllers and two slave controllers	Supports full and half-duplex mode
Quad SPI flash controller	Used for access to serial NOR flash devices
SD/MMC controller	Used for interfacing to external SD and MMC flash cards and secure digital I/O devices
Clock Manager(1)	Provides software-programmable clock control to configure all clocks generated in the HPS
System Manager(1)	Contains logic and registers to control system functions and other modules that need external control signals as part of their system integration
Reset Manager	Generates module reset signals based on reset requests from the various sources in the HPS and FPGA fabric, and software writing to the module-reset control registers

(1) This module's register set is modeled after the Arria V SoC device.

Module	Description
DMA	Provides high-bandwidth data transfers for modules without integrated DMA controllers. The DMA controller is based on the ARM Corelink DMA Controller (DMA-330)
FPGA Manager	Manages and monitors the FPGA portion of the System on a Chip (SoC) FPGA device
Four Timers	General purpose timers are connected to the level 4 (L4) peripheral bus
Three GPIO Modules	General purpose I/O interfaces
FPGA-to-HPS Bridge	Provides access to the peripherals and memory in the HPS
HPS-to- FPGA Bridge	Provides a configurable-width, high-performance master interface to the FPGA fabric
Lightweight HPS-to-FPGA Bridge	Provides a lower-performance interface to the FPGA fabric

For more details on modeling aspects of the Arria 10 SoC Virtual Platform, refer to "Appendix B: Memory and Interrupt Maps."

Related Information

Appendix B: Memory and Interrupt Map on page 16

Recommended PC Requirements

To run the virtual platform environment, your PC must meet the following minimum requirements:

- Any 64-bit version of the Linux operating system. The Ubuntu 12.04 Linux distribution and Red Hat® Enterprise 5.10 Linux distribution have been verified as supported.

  Note: For building Linux, only Ubuntu 12.04 has been tested and is supported.

- A minimum of 8 GB of RAM, but 32 GB is recommended for optimal performance.

Installing the Arria 10 SoC Virtual Platform

The following steps describe how to download and install the Arria 10 SoC Virtual Platform from Altera's Software site.

1. Download the Arria 10 SoC Virtual Platform **tar** file from the **Mentor Graphics® website** to your chosen directory. The file downloaded to your directory is named **Arria10_vp.tgz**.
2. Uncompress the virtual platform by typing the following command:

   ```
   tar zxvf Arria10_vp.tgz
   ```

   A directory named **Arria10_vp** is created with an installer named **install.sh** and an executable file named **install_a10socvp.exe**.

install.sh is a simple script that runs **install_a10socvp.exe** for you. **install_a10socvp.exe** installs the virtual platform to your directory.

3. To install the Arria 10 SoC Virtual Platform, change to directory **Arria10_vp** and type the following command:

```
./install.sh
```

The installer displays the following message:

```
Running: <chosen_directory>/Arria10_vp/install_a10socvp.exe  -install
<home_directory>/altera/socvp/arria10/<version_number>.  Please wait...
```

Note: The installation directory should not exist before running the install command. The virtual platform executable automatically creates this directory.

The following message displays when the virtual platform has successfully installed:

```
Virtual Prototype installed successfully to <chosen_directory>/altera/socvp/
arria10/<version_number>.
```

Installing and Booting a Pre-Built Linux Kernel

The following steps describe how to boot using the pre-built Linux kernel for the Arria 10 SoC Virtual Platform provided on RocketBoards.org.

1. Open a console and go to the directory where your virtual platform is installed:

```
cd <PATH_TO_VP_INSTALL_DIR>
```

2. At the Linux prompt, type the following command to download the Arria 10 SoC Virtual Platform Linux images:

```
wget --no-cache http://rocketboards.org/foswiki/pub/Documentation/\
Arria10SoCVPLinux/linux-arria10swvp-socfpga-3.10-ltsi-angstrom-v2014.12.tgz
```

3. Uncompress the **tgz** file by typing the following command:

```
tar xvzf ./linux-arria10swvp-socfpga-3.10-ltsi-angstrom-v2014.12.tgz
```

Note: This command creates the following files in the **./Software/arria10/linux/** folder:

- **linux-system-sd.elf**: contains the Linux kernel image
- **sd-angstrom-v2014.12-arria10swvp.img**: contains the root file system

4. To run the Arria 10 SoC Virtual Platform with the default pre-built Linux binaries, type the following command:

```
cd <PATH_TO_VP_INSTALL_DIR>
./run.exe
```

After the command runs, an Ångström prompt appears and you are in the root directory.

Debugging Linux Applications with the GNU Debugger (gdb)

This section describes how to use gdb on the host to debug an application running on the target.

Installing Host Packages

This section assumes Ubuntu is used on the host. Other operating systems require different commands.

For this installation, the gdb debugger and gcc cross compiler packages are required. If Ubuntu is used as the host, the installation commands are:

```
sudo apt-get install gcc-arm-linux-gnueabihf
sudo apt-get install gdb-multiarch
```

Installing Target Packages

The target requires that only the gdbserver package is installed.

1. Boot the virtual platform as shown in the "Boot Using Pre-Built Linux Kernel" section.
2. Edit the **/etc/resolv.conf** to contain only one entry with the selected DNS server as shown in the "Configuring the DNS Server" section.
3. Install the package by typing the following command:

```
opkg update
opkg install gdbserver
```

Related Information

- **Installing and Booting a Pre-Built Linux Kernel** on page 5
- **Configuring the DNS Server** on page 10

Creating and Cross-Compiling an Application on Host

1. Go to your home folder on the host and create a file named **factorial.c** that contains the following source code:

```
#include <stdio.h>

int factorial(int n) {
      if (n == 0)
            return 1;
      return n * factorial (n - 1);
}

int main () {
      int i;
      int n;
      for (i = 0; i < 10; ++i) {
        n = factorial (i);
        printf ("factorial(%d) = %d\n", i, n);
      }
      return 0;
}
```

2. Cross-compile the **factorial.c** file by typing the following command:

```
arm-linux-gnueabihf-gcc factorial.c -ggdb -o factorial.out
```

Moving the Application to the Target

To move the application to the target, run the commands listed below on the target. Be sure to replace `host_user`, `host_name` and `host_path` with the actual values for your host Linux PC:

```
cd ~
scp host_user@host_name:host_path/factorial.out .
scp host_user@host_name:host_path/factorial.c .
```

Starting the gdb Server on the Target

Start the gdb server on the target by typing the following commands:

```
cd ~
gdbserver :8080 ./factorial.out
```

Debugging Using the gdb Client on the Host

1. Run the gdb client on the host:

   ```
   gdb-multiarch ./factorial.out
   Process ./factorial.out created; pid = 229
   Listening on port 8080
   ```

2. Connect gdb to the target.

   Note: This example uses `localhost:3624` instead of `192.168.0.9:8080` as shown in the VLAN port mapping found in the "Network Connectivity" section.

   ```
   (gdb) target remote localhost:3624
   Remote debugging using localhost:3624
   warning: Unable to find dynamic linker breakpoint function.
   GDB will be unable to debug shared library initializers
   and track explicitly loaded dynamic code.
   0x76fcfb00 in ?? ()
   ```

3. Use the following gdb sample commands to debug the code:

   - `b main`: Set a breakpoint at the `main` function
   - `c`: Continue until the breakpoint is hit
   - `s`: Step one instruction
   - `b 14`: Insert a breakpoint at line 14
   - `c` typed multiple times: Run through iterations of the loop
   - `l`: List code

   The host console looks similar to the view below:

   ```
   <name>@ubuntu-12:~$ gdb-multiarch ./factorial.out
   GNU gdb (Ubuntu/Linaro 7.4-2012.04-0ubuntu2.1)7.4-2012.04
   Copyright (C) 2012 Free Software Foundation, Inc.
   License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
   This is free software: you are free to change and redistribute it.
   There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
   and "show warranty" for details.
   This GDB was configured as "x86_64-linux-gnu".
   For bug reporting instructions, please see:
   <http://bugs.launchpad.net/gdb-linaro/>...
   Reading symbols from /home/<name>/factorial.out...done.
   ```

```
(gdb) target remote localhost:3624
Remote debugging using localhost:3624
warning: Unable to find dynamic linker breakpoint function.
GDB will be unable to debug shared library initializers
and track explicitly loaded dynamic code.
0x76fcfb00 in ?? ()
(gdb) b main
Cannot access memory at address 0x0
Breakpoint 1 at 0x83ce: file factorial.c, line 12.
(gdb) c
Continuing.
warning: Could not load shared library symbols for 2 libraries, e.g.
/lib/libc.so.6.
Use the "info sharedlibrary" command to see the complete listing.
Do you need "set solib-search-path" or "set sysroot"?

Breakpoint 1, main () at factorial.c:12
12          for (i = 0; i < 10; ++i) {
(gdb) l
7     }
8
9     int main () {
10          int i;
11          int n;
12          for (i = 0; i < 10; ++i) {
13                n = factorial (i);
14                printf ("factorial(%d) = %d\n", i, n);
15          }
16          return 0;
(gdb) b 14
Breakpoint 2 at 0x83de: file factorial.c, line 14.
(gdb) c
Continuing.
Breakpoint 2, main () at factorial.c:14
14                printf ("factorial(%d) = %d\n", i, n);
(gdb) s
Cannot access memory at address 0x0
Breakpoint 2, main () at factorial.c:14
14                printf ("factorial(%d) = %d\n", i, n);
(gdb) c
Continuing.
Breakpoint 2, main () at factorial.c:14
14                printf ("factorial(%d) = %d\n", i, n);
(gdb) c
Continuing.
Breakpoint 2, main () at factorial.c:14
14                printf ("factorial(%d) = %d\n", i, n);
(gdb) c
Continuing. Breakpoint 2, main () at factorial.c:14
14                printf ("factorial(%d) = %d\n", i, n);
(gdb) c
Continuing.
Breakpoint 2, main () at factorial.c:14
14                printf ("factorial(%d) = %d\n", i, n);
(gdb) c
```

The target console looks similar to the view below:

```
root@host:~# gdbserver :8080 ./factorial.out
Process ./factorial.out created; pid = 229
Listening on port 8080
Remote debugging from host 192.168.0.1
factorial(0) = 1
factorial(1) = 1
factorial(2) = 2
factorial(3) = 6
```

```
factorial(4) = 24
factorial(5) = 120
factorial(6) = 720
factorial(7) = 5040
factorial(8) = 40320
factorial(9) = 362880

Child exited with status 0
```

Related Information

Network Connectivity on page 10

Network Connectivity

The Ethernet interface of the virtual platform can be configured to connect to the internet.

Network connectivity is configured in the **parameters_Arria10.txt** file. The default configuration found in the file is:

```
##################################
##            VLAN              ###
##################################
vlan:macstart = 52:54:00:12:34:90
vlan:net = 192.168.0.0/24
vlan:host = 192.168.0.1
vlan:hostname = host
vlan:dns = 192.168.0.3
vlan:dhcpstart = 192.168.0.9
vlan:tcp_napt = :3624 => :8080 ; :5684 => :23 ;:5247 => :69; :9547 => :22 ; :8524
=> :21 ; :6527 => :53
pmr_reset_value = 0xFF

Arria10_top.EMAC0.eth_name = eth1
Arria10_top.EMAC0.eth_mac = 52:54:00:12:34:57
```

The parameters within the "VLAN" section of the file to be aware of are:

- `vlan:dns` - This parameter defines the Domain Name Server that is used by the target to resolve IP addresses. The target file, **/etc/resolv.conf**, must be updated to direct it to use this DNS.
- `vlan:dhcpstart` - This parameter is the starting address provided by DHCP server. It is the target that the operating system IP addresses.
- `vlan:tcp_napt` - This parameter maps the ports on the target to the ports on the host. This mapping is required so that applications on the host can connect to applications on the target. Applications on the target can connect to the outside world (including the host) with the actual IP addresses and ports.
- `Arria10_top.EMAC0.eth_mac` – This parameter defines the MAC address of the target network adapter.

The following sections describe how to use the networking feature of the virtual platform.

Determining the Target IP Address

To determine the target IP address, you must run the `ifconfig` command on the target. The target IP address is returned by the VLAN DHCP that is configured as `192.168.0.9` in the **parameters_Arria10.txt** file. An example of determining the target IP address is shown below.

```
root@host:~# ifconfig
eth0 Link encap:Ethernet  HWaddr 52:54:00:12:34:57
     inet addr:192.168.0.9  Bcast:192.168.0.255 Mask:255.255.255.0
     inet6 addr: fe80::5054:ff:fe12:3457/64 Scope:Link
     UP BROADCAST RUNNING MULTICAST DYNAMIC  MTU:1500  Metric:1
     RX packets:221 errors:0 dropped:0 overruns:0 frame:0
     TX packets:378 errors:0 dropped:0 overruns:0 carrier:0
     collisions:0 txqueuelen:1000
     RX bytes:43814 (42.7 KiB)  TX bytes:37491 (36.6 KiB)
     Interrupt:124 Base address:0xc000
```

Configuring the DNS Server

To enable the target operating system to resolve internet addresses, you must configure the DNS server.

1. Run Linux on the target.
2. From the target Linux console, edit the **/etc/resolv.conf** file so that the only entry that exists is:
   `nameserver 192.168.0.3`
3. Test the network connectivity by typing:
   `wget www.google.com`

Note: The file **/etc/resolv.conf** is automatically created by Linux so it is overwritten each time Linux is booted. It is recommended that you incorporate the edits to the **resolv.conf** file in an initialization script that can run automatically each time Linux is booted.

Initiating ssh from the Host to the Target

To initiate ssh from host to target, you must use the port mapping found in the default **parameters_Arria10.txt** file. In the default **parameters_Arria10.txt** file, ssh port(22) is mapped to port 9547. Because of this mapping, the command to initiate ssh from host to target must be:

```
ssh root@localhost -p 9547
```

Initiating ssh from the Target to the Host

To run on the target, connect to outside servers using standard calls:

```
ssh host_user@host_name
```

Transferring Files from Host to Target

Use scp to move files from the host to the target:

```
scp host_user@host_name:host_path target_path
```

Building a Custom Linux Kernel Using Angstrom

If you want to customize your own Linux kernel, you can build a version of Linux using Ångström. The following sections provide an example of how to compile the Linux kernel and root file system using the Arria 10 SoC device Ångström recipes.

Note: For building Linux, only Ubuntu 12.04 has been tested and is supported.

Prerequisites for Building Linux

To build Linux on Ubuntu 12.04 LTS using Ångström, there are certain packages that must be pre-installed. To install these packages, type the following commands:

```
sudo apt-get install gawk wget git-core diffstat unzip texinfo gcc-multilib \
build-essential chrpath libsdl1.2-dev xterm
```

Related Information

Yocto Project Documentation

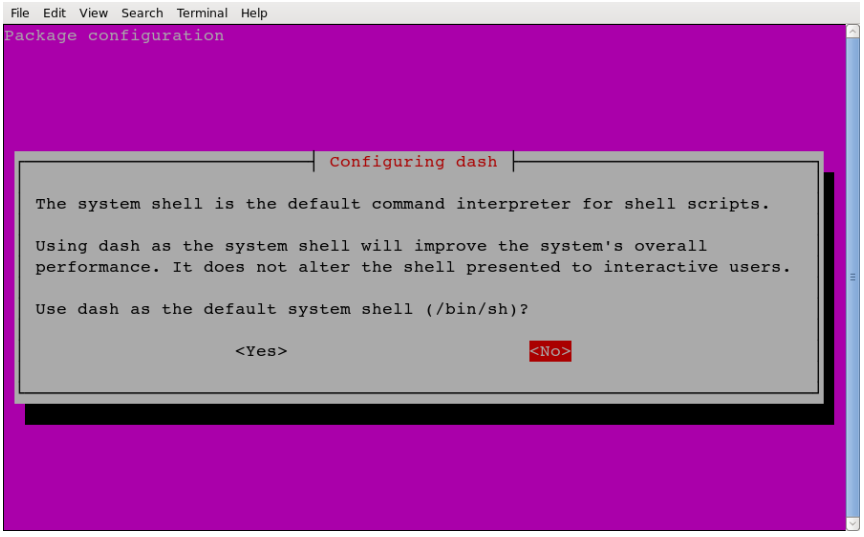
For more information about the packages that are pre-installed, refer to the *Yocto Project Documentation* website.

Building Linux

1. Set the default shell to bash. Select **No** when you are prompted to use dash as the system shell:

```
sudo dpkg-reconfigure dash
```

Figure 2: Bash Shell Configuration



2. To compile Linux using Ångström, type the following commands:

```
mkdir <PATH_TO_ANGSTROM_DIR>
cd <PATH_TO_ANGSTROM_DIR>
git clone git://github.com/altera-opensource/angstrom-socfpga.git
```

```
cd angstrom-socfpga
git checkout angstrom-v2014.12-socfpga
```

3. Setup your environment by typing the following commands:

```
export KERNEL_PROVIDER="linux-altera-ltsi"
export KERNEL_TAG="51a839398fdcdda7b283b7ac27425011c436525d"
MACHINE=arria10swvp ./oebb.sh config arria10swvp
```

Note: The current Linux kernel configuration can be found on **RocketBoards.org**.

4. Build the kernel:

```
source environment-angstrom-v2014.12
export BB_ENV_EXTRAWHITE="${BB_ENV_EXTRAWHITE} KERNEL_PROVIDER KERNEL_TAG"
MACHINE=arria10swvp bitbake virtual/kernel console-image virtual/bootloader
```

Note: Some third-party sources may not be present during the build process at remote repositories and may prevent this command from working. If this happens, the build command may fail with an error similar to this example case for gumstix source code:

```
IOError: file <PATH_TO_ANGSTROM_DIR>/angstrom-socfpga/sources/meta-gumstix-
community/conf/layer.conf not found

ERROR: Unable to parse <PATH_TO_ANGSTROM_DIR>/angstrom-socfpga/sources/meta-
gumstix-community/conf/layer.conf: file <PATH_TO_ANGSTROM_DIR>/angstrom-
socfpga/sources/meta-gumstix-community/conf/layer.conf not found
```

If this happens, you must edit the ***<PATH_TO_ANGSTROM_DIR>*/angstrom-socfpga/conf/bblayers.conf** file and remove the source causing problems. In the example above, removing the line shown below from **bblayers.conf** allows the build to complete:

```
${TOPDIR}/sources/meta-gumstix-community
```

Updating and Booting Linux with the Arria 10 SoC Virtual Platform

The following sections list the instructions for updating and replacing the default Linux images created by the Arria 10 SoC Virtual Platform with your Ångström build images.

Prerequisites for Updating Linux for the Virtual Platform

The Linaro GNU toolchain (cross-compiler for ARM) must be installed to update the virtual platform Linux binaries.

1. Please run the following commands to install the Linaro GNU toolchain:

```
mkdir <PATH_TO_TOOLCHAIN_INSTALL>
cd <PATH_TO_TOOLCHAIN_INSTALL>
sudo apt-get install ia32-libs
wget https://releases.linaro.org/13.12/components/toolchain/binaries/\
gcc-linaro-arm-linux-gnueabihf-4.8-2013.12_linux.tar.bz2
tar xjvf gcc-linaro-arm-linux-gnueabihf-4.8-2013.12_linux.tar.bz2
export PATH=$PWD/gcc-linaro-arm-linux-gnueabihf-4.8-2013.12_linux/bin:$PATH
export CROSS_COMPILE=arm-linux-gnueabihf-
```

Updating the Arria 10 SoC Virtual Platform Linux Kernel Image

1. Run the build script by typing the following:

```
cd <PATH_TO_VP_INSTALL>/Software/arria10/linux/buildsocvpelf

./build.sh <PATH_TO_ANGSTROM_DIR>/angstrom_socfpga/deploy/glibc/images/\
arria10swvp/zImage \
<PATH_TO_ANGSTROM_DIR>/angstrom-socfpga/deploy/glibc/images/arria10swvp/\
socfpga_arria10_swvp.dtb \
linux-system-sd.elf \
arm-linux-gnueabihf-
```

   This script creates the **linux-system-sd.elf** file. You can replace a similar file contained in the Rocketboards.org Linux binary package in one of two ways:

   - Move the new file to ***<PATH_TO_VP_INSTALL>*/Software/arria10/linux.**
   - Update ***<PATH_TO_VP_INSTALL>*/parameters_Arria10.txt** to point to the new file.

   Once the **build.sh** script has completed, you can run the virtual platform using the instructions in the "Executing the Arria 10 SoC Virtual Platform" section.

Related Information

Executing the Arria 10 SoC Virtual Platform on page 14

Creating an SD Card Image

1. To create a file named **sd-angstrom-v2014.12-arria10swvp.img** that can be used as the SD card image, type the following:

```
cd <PATH_TO_VP_INSTALL>/Software/arria10/linux/buildsocvpsd
./buildsd.sh <PATH_TO_ANGSTROM> \
arria10swvp arria10_swvp
```

   This script creates the **sd-angstrom-v2014.12-arria10swvp.img** file. This file must be writeable by the user who runs the virtual platform. To give the user write permissions type:

```
sudo chown $USER:$USER sd-angstrom-v2014.12-arria10swvp.img
```

   Move this image file to the Linux binary directory in one of two ways:

   - Move the new file to ***<PATH_TO_VP_INSTALL>*/Software/arria10/linux.**
   - Update ***<PATH_TO_VP_INSTALL>*/parameters_Arria10.txt** to point to this file.

Executing the Arria 10 SoC Virtual Platform

1. To run the Arria 10 SoC Virtual Platform, type the following command:

```
cd <PATH_TO_VP_INSTALL_DIR>
./run.exe
```

   After the command executes, an Ångström prompt displays and you are in the root directory.

Appendix A: Modifying the parameters_Arria10.txt File

The **parameters_Arria10.txt** file must be modified to fit the requirements of your design before executing the virtual platform with Linux.

The **parameters_Arria10.txt** file contains editable parameters for the Arria 10 SoC Virtual Platform model. The following list identifies the types of parameters that are contained with the file:

- Clock Frequencies
- Debugging features
- Cortex-A9 and L2 cache attributes
- L3 interconnect bus sizes and base addresses
- Peripheral memory map information
- QSPI flash sizes and image file location
- SD/MMC image file location
- Ethernet connection configuration

Refer to the "Network Connectivity" section for more information about configuring **parameters_Arria10.txt** for internet communication.

Related Information

Network Connectivity on page 10

Appendix B: Memory and Interrupt Map

Arria 10 SoC Memory Map

The following table details the Arria 10 SoC memory map and identifies which parts of the memory map are available on the Arria 10 SoC Virtual Platform.

Table 2: Arria 10 Memory Map for Virtual Platform

Module	Description	Base Address	Range	Available in Virtual Platform? (Y/N)
STM	STM module	0xFC000000	48 MB	No
DAP	DAP module	0xFF000000	2 MB	No
LWFPGASLAVES	FPGA slaves accessed via lightweight HPS2FPGA bridge module	0xFF200000	2 MB	Yes
EMAC0	EMAC0 module	0xFF800000	8 KB	Yes
EMAC1	EMAC1 module	0xFF802000	8 KB	Yes
EMAC2	EMAC2 module	0xFF804000	8 KB	No
SDMMC	SD/MMC module	0xFF808000	4 KB	Yes
QSPIREGS	QSPI flash controller module registers	0xFF809000	4 KB	Yes
EMAC0RXECC	Receive ECC, Ethernet MAC0	0xFF8C0800	1 KB	No
EMAC0TXECC	Transmit ECC, Ethernet MAC0	0xFF8C0C00	1 KB	No
EMAC1RXECC	Receive ECC, Ethernet MAC1	0xFF8C1000	1 KB	No
EMAC1TXECC	Transmit ECC, Ethernet MAC1	0xFF8C1400	1 KB	No
EMAC2RXECC	Receive ECC, Ethernet MAC2	0xFF8C1800	1 KB	No
EMAC2TXECC	Transmit ECC, Ethernet MAC2	0xFF8C1C00	1 KB	No
NANDECC	NAND ECC	0xFF8C2000	1 KB	No
NANDREADECC	NAND read ECC	0xFF8C2400	1 KB	No
NANDWRITEECC	NAND write ECC	0xFF8C2800	1 KB	No
SDMMCECC	SD/MMC ECC	0xFF8C2C00	1 KB	No

Module	Description	Base Address	Range	Available in Virtual Platform? (Y/N)
OCRAMECC	On-chip RAM ECC	0xFF8C3000	1 KB	No
DMAECC	DMA ECC	0xFF8C8000	1 KB	No
QSPIECC	QSPI ECC	0xFF8C8400	1 KB	No
USB0ECC	USB 2.0 OTG 0 ECC	0xFF8C8800	1 KB	No
USB1ECC	USB 2.0 OTG 1 ECC	0xFF8C8C00	1 KB	No
QSPIDATA	QSPI flash module data	0xFFA00000	1 MB	Yes
USB0	USB 2.0 OTG 0 controller module registers	0xFFB00000	256 KB	No
USB1	USB 2.0 OTG 1 controller module register	0xFFB40000	256 KB	No
NANDREGS	NAND controller module registers	0xFFB80000	64 KB	No
NANDDATA	NAND controller module data	0xFFB90000	64 KB	No
UART0	UART0 module	0xFFC02000	256 B	Yes
UART1	UART1 module	0xFFC02100	256 B	Yes
I2C0	I^2C0 module	0xFFC02200	256 B	Yes
I2C1	I2C1 module	0xFFC02300	256 B	Yes
I2C2	I^2C2 module (can be used with EMAC0)	0xFFC02400	256 B	Yes
I2C3	I^2C3 module (can be used with EMAC	0xFFC02500	256 B	Yes
I2C4	I^2C4 module (can be used with EMAC2)	0xFFC02600	256 B	No
SPTIMER0	SP Timer0 module	0xFFC02700	256 B	Yes
SPTIMER1	SP Timer1 module	0xFFC02800	256 B	Yes
GPIO0	GPIO0 module	0xFFC02900	256 B	Yes
GPIO1	GPIO1 module	0xFFC02A00	256 B	Yes
GPIO2	GPIO2 module	0xFFC02B00	256 B	Yes

Module	Description	Base Address	Range	Available in Virtual Platform? (Y/N)
HMCREGS	HMC control registers	0xFFCFA000	4 KB	No
HMCAREGS	HMC adapter control registers	0xFFCFB000	4 KB	No
SECMGRDATA	Security manager module data	0xFFCFE000	1 KB	No
FPGAMGRDATA	FPGA manager module configuration data	0xFFCFE400	1 KB	Yes
OSC1TIMER0	OSC1 Timer0 module	0xFFD00000	256B	Yes
OSC1TIMER1	OSC1 Timer1 module	0xFFD00100	256B	Yes
L4WD0	Watchdog0 module	0xFFD00200	256B	No
L4WD1	Watchdog1 module	0xFFD00300	256B	No
SECMGRREGS	Security manager module control and status registers	0xFFD02000	4 KB	No
FPGAMGRREGS	FPGA manager module control and status registers	0xFFD03000	4 KB	Yes
CLKMGR	Clock manager module	0xFFD04000	4 KB	Yes
RSTMGR	Reset manager module	0xFFD05000	4 KB	Yes
SYSMGR	System manager module	0xFFD06000	4 KB	Yes
IOMGR	I/O manager module	0xFFD07000	4 KB	No
FWL4PRIV	L4 privilege firewall registers	0xFFD11000	256 B	No
MPURADAPTER	MPU rate adapter registers	0xFFD11100	3.84 KB	No
DDRPRB	DDR probe registers	0xFFD12000	1 KB	No
SCHREGS	DDR scheduler control registers	0xFFD12400	128 B	No
FWL4PER	L4 peripheral firewall registers	0xFFD13000	256 B	No

Module	Description	Base Address	Range	Available in Virtual Platform? (Y/N)
FWL4SYS	L4 system firewall registers	0xFFD13100	256 B	No
FWOCRAM	On-chip RAM firewall registers	0xFFD13200	256 B	No
FWFPGA2SDRAM	DDR firewall registers for FPGA-to-SDRAM	0xFFD13300	256 B	No
FWDDRL3	DDR L3 firewall registers	0xFFD13400	256 B	No
FWHPS2FPGA	HPS-to-FPGA firewall registers	0xFFD13500	256 B	No
L4PRB	L4 interconnect probe registers	0xFFD14000	4 KB	No
MPUPRB	MPU probe and test registers	0xFFD15000	4 KB	No
L4QOS	L4 interconnect QoS	0xFFD16000	4 KB (estimated)	No
EMACTSF	EMAC transaction status filter registers	0xFFD1 7080	44 B	No
DMANONSE-CURE	DMA non-secure module registers	0xFFDA0000	4 KB	No
DMASECURE	DMA secure module registers	0xFFDA1000	4 KB	Yes
SPI0	SPI module 0 slave	0xFFDA2000	4 KB	Yes
SPI1	SPI module 1 slave	0xFFDA3000	4 KB	Yes
SPI2	SPI module 0 master	0xFFDA4000	4 KB	Yes
SPI3	SPI module 1 master	0xFFDA5000	4 KB	Yes
OCRAM	On-chip RAM module	0xFFE00000	1 MB (256 KB used)	Yes
ROM	Boot ROM module	0xFFFC0000	128 KB	Yes
MPU	MPU Module registers	0xFFFFC000	8 KB	Yes
MPUL2	MPU L2 Cache Controller module registers	0xFFFFF000	4 KB	Yes

Host Interrupts

The host interrupts that are available to the virtual platform model are listed below:

Table 3: Arria 10 SoC Virtual Platform Interrupt Map

Note: The IRQ numbers used by the Arria 10 SoC Virtual Platform are different than the interrupt vector numbers assigned to sources in the general interrupt controller (GIC) of Arria 10 SoC device hardware. The differences are noted in the table below.

Virtual Platform IRQ Number	Arria 10 GIC Interrupt Number	Source	Description
18	50	L2 Cache	L2 Cache interrupt request
83	115	DMA_IRQ0	DMA interrupt request0
84	116	DMA_IRQ1	DMA interrupt request1
85	117	DMA_IRQ2	DMA interrupt request2
86	118	DMA_IRQ3	DMA interrupt request3
87	119	DMA_IRQ4	DMA interrupt request4
88	120	DMA_IRQ5	DMA interrupt request5
89	121	DMA_IRQ6	DMA interrupt request6
90	122	DMA_IRQ7	DMA interrupt request7
91	123	DMA_IRQ_Abort	DMA abort interrupt
92	124	EMAC0	EMAC0 combined interrupt request
93	125	EMAC1	EMAC1 combined interrupt request
98	130	SDMMC	SDMMC interrupt request
100	132	QSPI	QSPI combined interrupt request
103	133	SPI0	SPIM0 interrupt request
104	134	SPI1	SPIM1 interrupt request
101	135	SPI2	SPIS0 interrupt request
102	136	SPI3	SPIS1 interrupt request
105	137	I2C0	I^2C0 interrupt request
106	138	I2C1	I^2C1 interrupt request
107	139	I2C2	I^2C2 interrupt request
108	140	I2C3	I^2C3 interrupt request
110	142	UART0	UART0 interrupt request
111	143	UART1	UART1 interrupt request
112	144	GPIO0	GPIO0 interrupt request

Virtual Platform IRQ Number	Arria 10 GIC Interrupt Number	Source	Description
113	145	GPIO1	GPIO1 interrupt request
114	146	GPIO2	GPIO2 interrupt request
115	147	TIMER0	SP Timer0 interrupt request
116	148	TIMER1	SP Timer1 interrupt request
117	149	TIMER2	OSC Timer0 interrupt request
118	150	TIMER3	OSC Timer1 interrupt request
123	155	FPGAMANGR	FPGA Manager interrupt request

Revision History of Arria 10 SoC Virtual Platform User Guide

Date	Version	Changes
September 2015	2015.09.16	• Updated commands in steps 3 and 4 in the "Building Linux" section
September 2015	2015.09.04	• Updated block diagram and included the two USB 2.0 OTG modules as modeled in the Arria 10 SoC Virtual Platform in the section "Arria 10 SoC Virtual Platform Block Diagram" • Modified Linux requirements in the "Recommended PC Requirements" section • Added detail to Step 1 and clarified Step 2 and Step 4 in the "Building Linux" subsection • Updated "Prerequisites for Updating Linux for the Virtual Platform" subsection and "Updating the Arria 10 SoC Virtual Platform Linux Kernel Image" subsection • Added "Creating an SD Card Image" subsection • Moved the "Appendix C: Known Issues" section to the *Arria 10 SoC Virtual Platform Version 1.0 Release Notes* document
July 2015	2015.07.13	• Updated the "Building Linux Using Angstrom" section • Removed "Install Packages Under Fedora" subsection • Removed "Install Packages Under Ubuntu" subsection • Updated "Prerequisites for Building Linux" subsection • Updated "Building Linux" subsection • Updated "Updating the Arria 10 SoC Virtual Platform Linux Images" subsection • Moved "Modifying the parameters_Arria10.txt File" as an appendix • Added target console view to "Debugging Using the gdb Client on the Host"
July 2015	2015.07.10	Initial release