## Contents

1 Stratix 10 SoC Virtual Platform User Guide ................................................................. 3
   1.1 Stratix 10 SoC Virtual Platform Features ............................................................ 3
   1.2 Stratix 10 SoC Virtual Platform Block Diagram .................................................. 4
   1.3 Recommended PC Requirements ....................................................................... 5
   1.4 Installing the Stratix 10 SoC Virtual Platform ...................................................... 6
   1.5 Installing and Booting a Pre-Built Linux Kernel for the Stratix 10 SoC Virtual Platform ........................................................................................................ 7
   1.6 Debugging Linux Applications with the GNU Debugger (GDB) ............................ 8
      1.6.1 Installing Host Packages ........................................................................... 8
      1.6.2 Installing Target Packages ....................................................................... 8
      1.6.3 Creating and Cross-Compiling an Application on Host .............................. 8
      1.6.4 Moving the Application to the Target ......................................................... 9
      1.6.5 Starting the gdbserver on the Target ......................................................... 9
      1.6.6 Debugging Using the GDB Client on the Host .......................................... 9
   1.7 Network Connectivity for Stratix 10 SoC Virtual Platform ..................................... 11
      1.7.1 Determining the Target IP Address .......................................................... 12
      1.7.2 Configuring the DNS Server ..................................................................... 12
      1.7.3 Initiating SSH from the Host to the Target .............................................. 13
      1.7.4 Initiating SSH from the Target to the Host ............................................. 13
      1.7.5 Transferring Files from the Host to the Target ....................................... 13
   1.8 Building a Custom Linux Kernel Using Angstrom ................................................. 14
      1.8.1 Prerequisites for Building Linux ............................................................. 14
      1.8.2 Building Linux ....................................................................................... 14
      1.8.3 Updating and Booting Linux with the Stratix 10 SoC Virtual Platform ........ 16
   1.9 Setting Up the Platform for Regression Testing .................................................... 19
      1.9.1 Running Multiple Virtual Platforms on the Same Host ............................ 19
      1.9.2 Using More than One Parameters File .................................................... 20
   1.10 Extending the Platform with a Dynamic Library ............................................... 21
      1.10.1 Building a Dynamic Library ................................................................. 21
      1.10.2 Connecting the Dynamic Library to the Platform ................................. 21
   1.11 Appendix A: Modifying the parameters.txt File ............................................... 22
   1.12 Appendix B: Memory and Interrupt Map ......................................................... 23
      1.12.1 Stratix 10 SoC Memory Map ................................................................. 23
      1.12.2 Stratix 10 SoC Virtual Platform Host Interrupts ..................................... 25
   1.13 Revision History of Stratix 10 SoC Virtual Platform User Guide ....................... 27
The Stratix® 10 SoC Virtual Platform is based on Mentor Graphics® Embedded technology and enables early software development and verification for Intel® customers.

By allowing teams to work within a virtual platform framework, software developers can gain system visibility without buying multiple development boards. You can use the virtual platform to rapidly develop software before the actual silicon or board becomes available. You can use the virtual platform for:

- Early hardware driver development and partial validation
- Non-real time algorithm creation
- Application development

For Linux debug support, the GNU debugger (GDB) can be used with the Stratix 10 SoC Virtual Platform.

**Related Links**

RocketBoards.org Forum
For support questions regarding the Stratix 10 SoC Virtual Platform, refer to the Rocketboards.org community portal.

### 1.1 Stratix 10 SoC Virtual Platform Features

The Stratix 10 SoC Virtual Platform provides the following:

- A model of the Stratix 10 SoC device
- Partial modeling of the memory map and interrupt map for Stratix 10 SoC
- Early hardware driver development and validation
- Application debug with the GNU debugger

Intel provides a pre-built Linux kernel for use with the virtual platform. You can download the kernel by visiting the Stratix 10 SoC VP Linux page on the RocketBoards.org.

**Related Links**

- Installing and Booting a Pre-Built Linux Kernel for the Stratix 10 SoC Virtual Platform on page 7
- RocketBoards.org Forum
  For support questions regarding the Stratix 10 SoC Virtual Platform, refer to the Rocketboards.org community portal.

© 2016 Intel Corporation. All rights reserved. Intel, the Intel logo, Altera, Arria, Cyclone, Enpirion, MAX, Megacore, NIOS, Quartus and Stratix words and logos are trademarks of Intel Corporation in the US and/or other countries. Other marks and brands may be claimed as the property of others. Intel warrants performance of its FPGA and semiconductor products to current specifications in accordance with Intel’s standard warranty, but reserves the right to make changes to any products and services at any time without notice. Intel assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Intel. Intel customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.
1.2 Stratix 10 SoC Virtual Platform Block Diagram

The figure below details the Stratix 10 SoC device.

Note: Modules that are modeled in the Stratix 10 SoC Virtual Platform are highlighted in yellow in the diagram. The light blue blocks represent the MPU subsystem and FPGA portion of the SoC.

Note: See the Stratix 10 SoC Virtual Platform Release Notes for information on which modeled modules have undergone functional verification testing.

Figure 1. Stratix 10 SoC Virtual Platform Block Diagram
Table 1. Modules Modeled in Stratix 10 SoC Virtual Platform

<table>
<thead>
<tr>
<th>Module</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Quad-core ARM® Cortex®-A53 MPCore™ processor</td>
<td>Contains four Cortex-A53 with FPU support and a snoop control unit (SCU)</td>
</tr>
<tr>
<td>ARM L2 Cache</td>
<td>1 megabyte (MB) of shared, unified cache memory</td>
</tr>
<tr>
<td>General Interrupt controller</td>
<td>Provides partial support for the interrupt map</td>
</tr>
<tr>
<td>Memory module</td>
<td>On-chip RAM</td>
</tr>
<tr>
<td>Two 16550-compatible UARTs</td>
<td>Provides console input and output</td>
</tr>
<tr>
<td>Five I²C controllers</td>
<td>Inter-Integrated Circuit (I²C) serial communication bus controllers</td>
</tr>
<tr>
<td>Three Ethernet controllers</td>
<td>Provides network connectivity for the virtual platform</td>
</tr>
<tr>
<td>Two USB 2.0 OTG controllers</td>
<td>Supports device and host functions</td>
</tr>
<tr>
<td>Two SPI master controllers and two slave controllers</td>
<td>Supports full and half-duplex mode</td>
</tr>
<tr>
<td>SD/MMC controller</td>
<td>Interfaces to external SD and MMC flash cards and secure digital I/O devices</td>
</tr>
<tr>
<td>Clock Manager</td>
<td>Provides software-programmable clock control to configure all clocks generated in the hard processor system (HPS)</td>
</tr>
<tr>
<td>System Manager</td>
<td>Contains logic and registers to control system functions and other modules that need external control signals as part of their system integration</td>
</tr>
<tr>
<td>Reset Manager</td>
<td>Generates module reset signals based on:</td>
</tr>
<tr>
<td></td>
<td>• Reset requests from the various sources in the HPS and FPGA fabric</td>
</tr>
<tr>
<td></td>
<td>• Software writing to the reset module control registers</td>
</tr>
<tr>
<td>DMA</td>
<td>Provides high-bandwidth data transfers for modules without integrated DMA controllers. The DMA controller is based on the ARM Corelink DMA Controller (DMA-330).</td>
</tr>
<tr>
<td>Four Timers</td>
<td>General purpose timers connected to the level 4 (L4) peripheral bus</td>
</tr>
<tr>
<td>Four watchdog Timers</td>
<td>Programmable watchdog timers to recover from system lockup</td>
</tr>
<tr>
<td>Two GPIO Modules</td>
<td>General purpose I/O interfaces</td>
</tr>
</tbody>
</table>

Related Links
- Appendix B: Memory and Interrupt Map on page 23
  For more details on modeling aspects in the Stratix 10 SoC Virtual Platform
- Stratix 10 SoC Virtual Platform Release Notes

1.3 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. Intel has verified that the Ubuntu 12.04 Linux distribution and the Red Hat® Enterprise 5.10 Linux distribution support the Stratix 10 SoC Virtual Platform.
- A minimum of 8 GB RAM, but 32 GB is recommended for optimal performance.
1.4 Installing the Stratix 10 SoC Virtual Platform

You must have an account to gain access to the Download Center. You must also be logged in to continue.

You can download and install the Stratix 10 SoC Virtual Platform from Intel's software site.

1. Download the Stratix 10 SoC Virtual Platform tar file (Stratix10_vp.tgz) from the Stratix 10 SoC Virtual Platform Download page to your chosen directory.
2. Uncompress the virtual platform by typing the following command:

   ```bash
tar zxvf Stratix10_vp.tgz
   ``

   This command creates a directory named Stratix10_vp which contains an installer named install.sh.

   install.sh is a simple script that runs install_s10socvp.exe for you. install_s10socvp.exe installs the virtual platform to your directory.

3. Install the Stratix 10 SoC Virtual Platform, using one of two methods:

   Note: The directory ./altera should not exist prior to installation because the installer will not overwrite an existing directory.

   - Method 1: Type the following command if you would like to read the End User License Agreement before installation:

     ```bash
     ./Stratix10_vp/install.sh ./altera
     ``

     The installer displays the following message:

     Please press return to display the End User License Agreement

     After the license agreement is displayed, answer y or n to the prompt:

     Do you accept this End User License Agreement? [y/n]:

   - Method 2: Type the following command if you would like to accept the End User License Agreement in advance:

     ```bash
     ./Stratix10_vp/install.sh -a ./altera
     ``

     The Stratix 10 SoC Virtual Platform is now installed in ./altera.
1.5 Installing and Booting a Pre-Built Linux Kernel for the Stratix 10 SoC Virtual Platform

You can use the pre-built Linux kernel provided on RocketBoards.org to boot the Stratix 10 SoC Virtual Platform.

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 Stratix 10 SoC Virtual Platform Linux images:

   wget --no-cache http://rocketboards.org/foswiki/pub/Documentation/Stratix10SoCVPLinux/linux-stratix10swvp-socfpga-4.5-angstrom-v2014.12-swvp-1.3.tgz

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

   tar xvzf ./linux-stratix10swvp-socfpga-4.5-angstrom-v2014.12-swvp-1.3.tgz

   This command creates the following files under <PATH_TO_VP_INSTALL_DIR>:
   • ./sd-angstrom-v2014.12-stratix10swvp.img: contains the SD/MMC root file system image
   • ./linux-system-sd.elf: contains the Linux kernel image

   Note: If the .img file is not writeable, you will see this error message:

   Cannot insert disk, file not found: ./sd-angstrom-v2014.12-stratix10swvp.img

4. To run the Stratix 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 completes, an Ångström prompt appears and you are at the login prompt. Login with a username of "root" and leave the password field blank to enter the root directory.
1.6 Debugging Linux Applications with the GNU Debugger (GDB)

You can use GDB on the host to debug an application running on the Stratix 10 SoC Virtual Platform target.

1.6.1 Installing Host Packages

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

This installation requires the GDB debugger and GNU C Compiler (GCC) cross compiler packages.

Note: Ensure that you have downloaded the required package from Linaro™.

If Ubuntu is used as the host, the installation commands are:

```
sudo apt-get install gcc-aarch64-linux-gnu
sudo apt-get install gdb-multiarch
```

Related Links
Linaro Release page

1.6.2 Installing Target Packages

The target requires that only the gdbserver package is installed.

1. Boot the virtual platform as described in the "Installing and Booting a Pre-Built Linux Kernel for the Stratix 10 SoC Virtual Platform" section.

2. Edit `/etc/resolv.conf` to contain only one entry:

```
nameserver 192.168.0.3
```

3. Install the package by typing the following command:

```
opkg update
```

Related Links
• Installing and Booting a Pre-Built Linux Kernel for the Stratix 10 SoC Virtual Platform on page 7
• Configuring the DNS Server on page 12

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

1.6.3 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:

    aarch64-linux-gnu-gcc factorial.c -ggdb -o factorial.out

1.6.4 Moving the Application to the Target

To move the application to the target, run the commands on the target:

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

Note: Be sure to replace host_user, host_name and host_path with the actual values for your host Linux PC.

1.6.5 Starting the gdbserver on the Target

Start the gdbserver on the target by typing the following commands:

    cd ~
    gdbserver :8080 ./factorial.out

1.6.6 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 described in the VLAN port mapping 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
   - **dis 2**: Disable breakpoint 2
   - **l**: List code (short form of list)

The host console looks similar to the view below:

```plaintext
user@yorick:~/$ gdb-multiarch./factorial.out
GNU gdb (Ubuntu 7.10-1ubuntu2) 7.10
Copyright (C) 2015 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
...
<more messages from GNU>
...  
(gdb) target remote localhost:3624
Remote debugging using localhost:3624
Reading /lib/ld-linux-aarch64.so.1 from remote target...
warning: File transfers from remote targets can be slow. Use "set sysroot" to access files
locally instead.
Reading /lib/id-linux-aarch64.so.1 from remote target...
Reading symbols from target:/lib/id-linux-aarch64.so.1...
Reading /lib/id-2.20.so from remote target...
Reading /lib/.debug/id-2.20.so from remote target...
(no debugging symbols found)...done.
0x0000007fb7fd3d00 in ?? () from target:/lib/id-linux-aarch64.so.1
(gdb) b main
Breakpoint 1 at 0x400608: file factorial.c, line 12.
(gdb) c
Continuing.
(Reading /lib/libc.so.6 from remote target...)
(Reading /lib/libc-2.20.so from remote target...)
(Reading /lib/.debug/libc-2.20.so from remote target...)
Breakpoint 1, main () at factorial.c:12
12    for (i = 0; i < 10; ++i) {
(gdb) list
 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 0x40061c: file factorial.c, line 14.
(gdb) c
Continuing.
(gdb) s
```

The host console looks similar to the view below:
for (i = 0; i < 10; ++i) {
    printf ("factorial(%d) = %d\n", i, n);
}

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 Links
Network Connectivity for Stratix 10 SoC Virtual Platform on page 11
You can configure the Ethernet interface of the virtual platform to connect to the Internet.

1.7 Network Connectivity for Stratix 10 SoC Virtual Platform

You can configure the Ethernet interface of the virtual platform to connect to the Internet.

Network connectivity is configured in the parameters.txt file. The following code is an example of the virtual LAN (VLAN) parameter settings.

```
# ----------------------------------
# Virtual LAN settings
# ----------------------------------
vlan:net = 192.168.0.0/24
vlan:host = 192.168.0.1
vlan:hostname = host
vlan:macstart = 52:54:00:12:34:90

# TCP NAT Translations
vlan:tcp_napt = .3624 => .8080 ;
            .5684 => .23 ;
            .5247 => .69 ;
```
Note the following parameters within the "VLAN" section of the file:

- **vlan:dns** - This parameter defines the Domain Name Server that is used by the target to resolve IP addresses. You must update the target file, `/etc/resolv.conf`, 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.

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

### 1.7.1 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.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
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:10 Base address:0x8000 DMA chan:FF
```

### 1.7.2 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:* Linux automatically creates the `/etc/resolv.conf` file and the file is overwritten each time Linux is booted. Intel recommends that you incorporate the edits to the `resolv.conf` file in an initialization script that can run automatically each time Linux boots.

### 1.7.3 Initiating SSH from the Host to the Target

To initiate a secure shell (SSH) connection from the host to the target, you must use the port mapping in the default `parameters.txt` file. In the default `parameters.txt` file, SSH port(22) is mapped to port 9547. Because of this mapping, the command to initiate an SSH from the host to the target must be:

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

### 1.7.4 Initiating SSH from the Target to the Host

To run on the target, connect to outside servers by using this standard call:

```bash
ssh host_user@host_name
```

### 1.7.5 Transferring Files from the Host to the Target

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

```bash
scp -P 9547 filename root@localhost:~
```
1.8 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 Stratix 10 SoC device Ångström recipes.

1.8.1 Prerequisites for Building Linux

To build Linux on Ubuntu 12.04 LTS using Ångström, you must install certain packages. To install these packages, type the following commands:

```bash
sudo apt-get install gawk wget git-core diffstat texinfo gcc-multilib 
build-essential chrpath libstdc++.dev xterm device-tree-compiler
```

Before building Linux, choose locations for the Stratix 10 SoC Virtual Platform, the Ångström distribution and the Linaro GNU toolchain. Assign variables to these locations to be used with the tasks in this section. Customize the following variables to your own path:

```bash
export SOCVP_PATH=~/altera/socvp/stratix10/1.3
export ANGSTROM_PATH=~/angstrom
export TOOLCHAIN_PATH=~/toolchain
```

Related Links

Yocto Project Documentation

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

1.8.2 Building Linux

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

```bash
sudo dpkg-reconfigure dash
```
2. To compile Linux using Ångström, type the following commands:

```bash
mkdir ${ANGSTROM_PATH}
cd ${ANGSTROM_PATH}
git clone git://github.com/altera-opensource/angstrom-socfpga.git
cd angstrom-socfpga
git checkout angstrom-v2014.12-socfpga
```

3. Configure your environment by typing the following commands:

```bash
export KERNEL_PROVIDER="linux-altera"
MACHINE=stratix10swvp ./oebb.sh config stratix10swvp
```

If you want to build a specific version of Linux, add the kernel tag to the commands:

```bash
export KERNEL_PROVIDER="linux-altera"
echo KERNEL_TAG="89b2d9f15aed586571bf900fc041202b6c1085a9" 
MACHINE=stratix10swvp ./oebb.sh config stratix10swvp
```

You can access a list of valid kernel tags in the git repository found [here](#). In the **Code** tab, select the socfpga-4.5 branch from the **Branch:master** drop down menu. Then click the **commits** link above the **Branch:master** pull down to see the individual commits. You can copy the full kernel tag to the clipboard by selecting the **Copy the full SHA** clipboard icon to the left of the individual kernel number.

You can access commits on branches other than the socfpga-4.5 in a similar manner.

4. Build the kernel:

```bash
source environment-angstrom-v2014.12
export BB_ENV_EXTRAWHITE="$BB_ENV_EXTRAWHITE" KERNEL_PROVIDER"
MACHINE=stratix10swvp bitbake virtual/kernel console-image
```
If you exported a KERNEL_TAG in the previous step, use these commands:

```bash
source environment-angstrom-v2014.12
export BB_ENV_EXTRAWHITE="$(BB_ENV_EXTRAWHITE) KERNEL_PROVIDER KERNEL_TAG"
MACHINE=stratix10svfvp bitbake virtual/kernel console-image
```

**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 issue occurs, the build command may fail with an error similar to this example case for gumstix source code:

```bash
IOError: file ${ANGSTROM_PATH}/angstrom-socfpga/sources/meta-gumstix-community/conf/layer.conf not found
ERROR: Unable to parse ${ANGSTROM_PATH}/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
```

To correct this error, you must edit the `$(ANGSTROM_PATH)/angstrom-socfpga/conf/bblayers.conf` file and remove the source of the error. In the example above, removing the following line from `bblayers.conf` allows the build to complete:

```bash
$(TOPDIR)/sources/meta-gumstix-community
```

### 1.8.3 Updating and Booting Linux with the Stratix 10 SoC Virtual Platform

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

#### 1.8.3.1 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:

   ```bash
   mkdir ${TOOLCHAIN_PATH}
   cd ${TOOLCHAIN_PATH}
   wget https://releases.linaro.org/15.06/components/toolchain/binaries/\4.8/aarch64-linux-gnu/gcc-linaro-4.8-2015.06-x86_64_aarch64-linux-gnu.tar.xz
   tar xf gcc-linaro-4.8-2015.06-x86_64_aarch64-linux-gnu.tar.xz
   ```
1.8.3.2 Updating the Stratix 10 SoC Virtual Platform Linux Kernel Image

1. Get the build tools from Rocketboards:

   cd ${SOCVP_PATH}/Software
   wget --no-cache http://rocketboards.org/foswiki/pub/Documentation/
   Stratix10SoCVPLinux/build_tools.tgz
   tar xf build_tools.tgz

2. Compress the console image:

   cd ${ANGSTROM_PATH}/angstrom-socfpga/deploy/glibc/images/stratix10swvp/
   gzip -c console-image-stratix10swvp.cpio > 
   console-imagestratix10swvp.cpio.gz

3. Build the Linux ELF file:

   export PATH= 
   ${TOOLCHAIN_PATH}/gcc-linaro-4.8-2015.06-x86_64_aarch64-linux-gnu/bin:\n
   export CROSS_COMPILE=aarch64-linux-gnu-
   cd ${SOCVP_PATH}/Software/stratix10/sw
   ./build.sh 
   ${ANGSTROM_PATH}/angstrom-socfpga/deploy/glibc/images/stratix10swvp/ 
   Image \n   ${ANGSTROM_PATH}/angstrom-socfpga/deploy/glibc/images/stratix10swvp/\n   stratix10_swvp.dtb \n   ${ANGSTROM_PATH}/angstrom-socfpga/deploy/glibc/images/stratix10swvp/ 
   consoleimage-\n   stratix10swvp.cpio.gz \n   linux-system-sd.elf \n   aarch64-linux-gnu-

   These steps create the linux-system-sd.elf file. Copy this file to the $ 
   (SOCVP_PATH) directory.

   When the build.sh script has completed, you can run the virtual platform using the 
   instructions in the Executing the Stratix 10 SoC Virtual Platform section.

   Related Links
   Executing the Stratix 10 SoC Virtual Platform on page 18

1.8.3.3 Creating an SD Card Image for Stratix 10 SoC Virtual Platform

1. To create a file named sd-angstrom-v2014.12-stratix10swvp.img that can 
   be used as the SD card image, type the following:

   cd ${SOCVP_PATH}/Software/stratix10/linux/buildsocvpsd
   ./buildsd.sh ${ANGSTROM_PATH} \n   stratix10swvp stratix10_swvp

   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-stratix10swvp.img

2. Move the new file to the ${SOCVP_PATH} directory.
1.8.3.4 Executing the Stratix 10 SoC Virtual Platform

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

```
  cd ${SOCVP_PATH}
  ./run.exe
```

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

1.8.3.5 Exiting the Stratix 10 SoC Virtual Platform

Similar to all Linux systems, you must gracefully shut down the Stratix 10 SoC Virtual Platform.

Because the SD card image is writable, you can corrupt the image if you select `Ctrl-C` to exit the platform. The next time you boot the platform, you may encounter errors like this:

```
[FAILED] Failed to start File System Check on Root Device.
See 'systemctl status systemd-fsck-root.service' for details.
...<more>...
systemd-shutdown[1]: Sending SIGKILL to remaining processes...
systemd-shutdown[1]: Unmounting file systems.
systemd-shutdown[1]: All filesystems unmounted.
...<more>...
systems-shutdown[1]: Rebooting.
reboot: Restarting system
Reboot failed -- System halted
```

For a graceful shutdown, use any of these commands:

- `halt`
- `poweroff`
- `init 0`
- `reboot -p`
- `shutdown`
1.9 Setting Up the Platform for Regression Testing

The Software Virtual Platform can be configured for regression testing, running without user interaction in an automated environment.

To do this, you must change the `parameters.txt` file to redirect input and output. Normally, the `console_type` is set to `xterm` for user interaction, but it can be set to `batch` to run on automated systems. Adding these lines to the parameters file configures the platform for automation, take input from a script called `testinput`, and send output to a log file called `testoutput`.

```plaintext
console_type = batch
console_script_path = ./testinput
console_log_path = ./testoutput
```

The `testinput` file uses a very simple set of commands that receives certain input and then sends a response. An example `testinput` file that logs into the system and runs the telnet daemon is shown below:

```plaintext
receive "host login:"
send "root\n"
receive "root@host:~#"
send "telnetd\n"
```

The `testinput` file is optional, but can be used to setup or start a test. Output from console is saved to the `testoutput` file.

1.9.1 Running Multiple Virtual Platforms on the Same Host

If you choose to run more than one virtual platform on the same host, modify the vlan settings in the parameters file so that no host port is mapped twice. For example, one invocation might use these settings:

```plaintext
vlan:tcp_napt = :3624 => :8080 ;
    :5684 => :23   ;
    :5247 => :69   ;
    :9547 => :22   ;
    :8524 => :21   ;
    :6527 => :53
vlan:udp_napt = :5248 => :69
```

And another invocation might use these settings:

```plaintext
vlan:tcp_napt = :3625 => :8080 ;
    :5685 => :23   ;
    :5248 => :69   ;
    :9548 => :22   ;
    :8525 => :21   ;
    :6528 => :53
vlan:udp_napt = :5249 => :69
```
1.9.2 Using More than One Parameters File

You may choose to put all of the parameters that do not change in one file, and include other files specific to a test. To do this, the specific parameters should be put into a separate file. For example, if you create a `testparams.txt` file, your top level `parameters.txt` file should be modified to include both the base file and the specific file as shown:

```
@include $DIRNAME/parameters_$env(VISTA_PLATFORM_NAME).txt
@include $DIRNAME/testparams.txt
```
1.10 Extending the Platform with a Dynamic Library

The virtual platform can be extended to add new models of hardware in the memory space. This can be used to simulate new hardware before the hardware is actually developed. This could be used to simulate the programmers’ view of an FPGA, allowing the programmer to develop code for an FPGA model before the FPGA is available.

1.10.1 Building a Dynamic Library

The distribution contains a sample dynamic library in the `Software/External_function` directory. To build it, change to that directory on your host system and run `make`. You must have a GCC compiler in your path. The `External_function` directory contains a `Readme.txt` file that explains how to use the simple example.

1.10.2 Connecting the Dynamic Library to the Platform

Once built, the dynamic library is connected to the platform by specifying the location of the dynamic library in the `parameters.txt` file. These values in the file should point to the dynamic library:

```bash
# ----------------------------------
# External function library
# ----------------------------------
Stratix10_top_ext_inst.dynamic_library = "\n$DIRNAME/Software/External_function/example.so"
```

In this case, the dynamic library in use is the sample `External_function` library included with the platform.
1.11 Appendix A: Modifying the parameters.txt File

You must modify the parameters.txt file to fit the requirements of your design before executing the virtual platform with Linux.

The Stratix 10 SoC Virtual Platform uses two parameter files.
- parameters.txt contains the most commonly modified parameters.
  - File pointers
  - Console setup
  - Warnings and message controls
  - Network configuration
- extended_parameters.txt contains system parameters that are not commonly modified. This file includes parameters that align the memory map and interconnect to the physical Stratix 10 SoC device.
  - SystemC settings
  - Processor subsystem settings
  - Interconnect address ranges
  - Port settings
  - Clocks

Note: Modifying extended_parameters.txt can have adverse effects on software stacks and platform compatibility with the physical Stratix 10 SoC device.

Refer to the "Network Connectivity for Stratix 10 SoC Virtual Platform" section for more information about configuring parameters.txt for Internet communication.

Related Links
Network Connectivity for Stratix 10 SoC Virtual Platform on page 11
You can configure the Ethernet interface of the virtual platform to connect to the Internet.
### 1.12 Appendix B: Memory and Interrupt Map

#### 1.12.1 Stratix 10 SoC Memory Map

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

<table>
<thead>
<tr>
<th>Module</th>
<th>Description</th>
<th>Start Address</th>
<th>End Address</th>
<th>Modeled in Virtual Platform?</th>
</tr>
</thead>
<tbody>
<tr>
<td>MEMORY</td>
<td>DDR Memory</td>
<td>0x00000000</td>
<td>0x7FFFFFFF</td>
<td>Yes</td>
</tr>
<tr>
<td>FPGASLAVES</td>
<td>FPGA slaves via HPS2FPGA bridge</td>
<td>0x80000000</td>
<td>0xDFFFFFFF</td>
<td>No</td>
</tr>
<tr>
<td>CCU_NOC</td>
<td>Cache Controller Unit</td>
<td>0x70000000</td>
<td>0xF7FFFFFF</td>
<td>No</td>
</tr>
<tr>
<td>DDR_REGS</td>
<td>Multiport front end module</td>
<td>0xF8000000</td>
<td>0xF80240FF</td>
<td>No</td>
</tr>
<tr>
<td>LWFPGASLAVES</td>
<td>FPGA slaves accessed via lightweight HPS2FPGA bridge module</td>
<td>0xF9000000</td>
<td>0xF91FFFFF</td>
<td>No</td>
</tr>
<tr>
<td>MMU</td>
<td>System MMU</td>
<td>0xFA000000</td>
<td>0xFA03FFFF</td>
<td>No</td>
</tr>
<tr>
<td>EMAC0</td>
<td>EMAC0 module</td>
<td>0xFF800000</td>
<td>0xFF80105B</td>
<td>Yes</td>
</tr>
<tr>
<td>EMAC1</td>
<td>EMAC1 module</td>
<td>0xFF802000</td>
<td>0xFF80305B</td>
<td>Yes</td>
</tr>
<tr>
<td>EMAC2</td>
<td>EMAC2 module</td>
<td>0xFF804000</td>
<td>0xFF80505B</td>
<td>Yes</td>
</tr>
<tr>
<td>SDMMC</td>
<td>SD/MMC module</td>
<td>0xFF808000</td>
<td>0xFF8083FF</td>
<td>Yes</td>
</tr>
<tr>
<td>EMAC0ECC_RX</td>
<td>Receive ECC, Ethernet MAC0</td>
<td>0xFF8C0000</td>
<td>0xFF8C03FF</td>
<td>No</td>
</tr>
<tr>
<td>EMAC0ECC_TX</td>
<td>Transmit ECC, Ethernet MAC0</td>
<td>0xFF8C0400</td>
<td>0xFF8C07FF</td>
<td>No</td>
</tr>
<tr>
<td>EMAC1ECC_RX</td>
<td>Receive ECC, Ethernet MAC1</td>
<td>0xFF8C0800</td>
<td>0xFF8C0BFF</td>
<td>No</td>
</tr>
<tr>
<td>EMAC1ECC_TX</td>
<td>Transmit ECC, Ethernet MAC1</td>
<td>0xFF8C0C00</td>
<td>0xFF8C0FFF</td>
<td>No</td>
</tr>
<tr>
<td>EMAC2ECC_RX</td>
<td>Receive ECC, Ethernet MAC2</td>
<td>0xFF8C1000</td>
<td>0xFF8C13FF</td>
<td>No</td>
</tr>
<tr>
<td>EMAC2ECC_TX</td>
<td>Transmit ECC, Ethernet MAC2</td>
<td>0xFF8C1400</td>
<td>0xFF8C17FF</td>
<td>No</td>
</tr>
<tr>
<td>USB0ECC</td>
<td>USB 2.0 OTG 0 ECC</td>
<td>0xFF8C4000</td>
<td>0xFF8C43FF</td>
<td>No</td>
</tr>
<tr>
<td>USB1ECC</td>
<td>USB 2.0 OTG 1 ECC</td>
<td>0xFF8C4400</td>
<td>0xFF8C47FF</td>
<td>No</td>
</tr>
<tr>
<td>NANDECC</td>
<td>NAND ECC</td>
<td>0xFF8C8000</td>
<td>0xFF8C83FF</td>
<td>No</td>
</tr>
<tr>
<td>NANDREADECC</td>
<td>NAND read ECC</td>
<td>0xFF8C8400</td>
<td>0xFF8C87FF</td>
<td>No</td>
</tr>
<tr>
<td>NANDWRITEECC</td>
<td>NAND write ECC</td>
<td>0xFF8C8800</td>
<td>0xFF8C8BFF</td>
<td>No</td>
</tr>
<tr>
<td>SDMMC_ECC</td>
<td>SD/MMC ECC</td>
<td>0xFF8C8C00</td>
<td>0xFF8C8FFF</td>
<td>No</td>
</tr>
<tr>
<td>DMAECC</td>
<td>DMA ECC</td>
<td>0xFF8C9000</td>
<td>0xFF8C90D0</td>
<td>No</td>
</tr>
<tr>
<td>APSRAMECC</td>
<td>APS RAM ECC</td>
<td>0xFF8CC000</td>
<td>0xFF8CC3FF</td>
<td>No</td>
</tr>
</tbody>
</table>

*continued...*
<table>
<thead>
<tr>
<th>Module</th>
<th>Description</th>
<th>Start Address</th>
<th>End Address</th>
<th>Modeled in Virtual Platform?</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSI</td>
<td>Protocol specific interface (PSI) debug</td>
<td>0xFFA44400</td>
<td>0xFFA47FFF</td>
<td>No</td>
</tr>
<tr>
<td>USB0</td>
<td>USB 2.0 OTG 0 controller module registers</td>
<td>0xFFF00000</td>
<td>0xFFF3FFFF</td>
<td>Yes¹</td>
</tr>
<tr>
<td>USB1</td>
<td>USB 2.0 OTG 1 controller module register</td>
<td>0xFFF40000</td>
<td>0xFFF7CFFF</td>
<td>Yes¹</td>
</tr>
<tr>
<td>NANDREGS</td>
<td>NAND controller module registers</td>
<td>0xFFF80000</td>
<td>0xFFF807D3</td>
<td>No</td>
</tr>
<tr>
<td>NANDDATA</td>
<td>NAND controller module data</td>
<td>0xFFF90000</td>
<td>0xFFF9FFFF</td>
<td>No</td>
</tr>
<tr>
<td>UART0</td>
<td>UART0 module</td>
<td>0xFFC02000</td>
<td>0xFFC020FF</td>
<td>Yes</td>
</tr>
<tr>
<td>UART1</td>
<td>UART1 module</td>
<td>0xFFC02100</td>
<td>0xFFC021FF</td>
<td>Yes</td>
</tr>
<tr>
<td>I2C0</td>
<td>I²C0 module</td>
<td>0xFFC02800</td>
<td>0xFFC028FF</td>
<td>Yes</td>
</tr>
<tr>
<td>I2C1</td>
<td>I²C1 module</td>
<td>0xFFC02900</td>
<td>0xFFC029FF</td>
<td>Yes</td>
</tr>
<tr>
<td>I2C2</td>
<td>I²C2 module</td>
<td>0xFFC02A00</td>
<td>0xFFC02AFF</td>
<td>Yes</td>
</tr>
<tr>
<td>I2C3</td>
<td>I²C3 module</td>
<td>0xFFC02B00</td>
<td>0xFFC02BFF</td>
<td>Yes</td>
</tr>
<tr>
<td>I2C4</td>
<td>I²C4 module</td>
<td>0xFFC02C00</td>
<td>0xFFC02CFF</td>
<td>Yes</td>
</tr>
<tr>
<td>SPTIMER0</td>
<td>SP Timer0 module</td>
<td>0xFFC03000</td>
<td>0xFFC030FF</td>
<td>Yes</td>
</tr>
<tr>
<td>SPTIMER1</td>
<td>SP Timer1 module</td>
<td>0xFFC03100</td>
<td>0xFFC031FF</td>
<td>Yes</td>
</tr>
<tr>
<td>GPIO0</td>
<td>GPIO0 module</td>
<td>0xFFC03200</td>
<td>0xFFC0327F</td>
<td>Yes</td>
</tr>
<tr>
<td>GPIO1</td>
<td>GPIO1 module</td>
<td>0xFFC03300</td>
<td>0xFFC0337F</td>
<td>Yes</td>
</tr>
<tr>
<td>TMR_SYS0</td>
<td>System Timer0 module</td>
<td>0xFFD00000</td>
<td>0xFFD000FF</td>
<td>Yes</td>
</tr>
<tr>
<td>TMR_SYS1</td>
<td>System Timer1 module</td>
<td>0xFFD00100</td>
<td>0xFFD001FF</td>
<td>Yes</td>
</tr>
<tr>
<td>WDT0</td>
<td>Watchdog0 module</td>
<td>0xFFD00200</td>
<td>0xFFD002FF</td>
<td>Yes</td>
</tr>
<tr>
<td>WDT1</td>
<td>Watchdog 1 module</td>
<td>0xFFD00300</td>
<td>0xFFD003FF</td>
<td>Yes</td>
</tr>
<tr>
<td>WDT2</td>
<td>Watchdog2 module</td>
<td>0xFFD00400</td>
<td>0xFFD004FF</td>
<td>Yes</td>
</tr>
<tr>
<td>WDT3</td>
<td>Watchdog3 module</td>
<td>0xFFD00500</td>
<td>0xFFD005FF</td>
<td>Yes</td>
</tr>
<tr>
<td>CLKMGR</td>
<td>Clock manager module</td>
<td>0xFFD10000</td>
<td>0xFFD10147</td>
<td>Yes</td>
</tr>
<tr>
<td>RSTMGR</td>
<td>Reset manager module</td>
<td>0xFFD11000</td>
<td>0xFFD110FF</td>
<td>Yes</td>
</tr>
<tr>
<td>SYSMGR</td>
<td>System manager module</td>
<td>0xFFD12000</td>
<td>0xFFD124FF</td>
<td>Yes</td>
</tr>
<tr>
<td>PINMUX</td>
<td>I/O manager</td>
<td>0xFFD13000</td>
<td>0xFFD13FFF</td>
<td>Yes</td>
</tr>
<tr>
<td>MAINPRB</td>
<td>NoC main probe</td>
<td>0xFFD20000</td>
<td>0xFFD223FF</td>
<td>No</td>
</tr>
<tr>
<td>MAINATB</td>
<td>NoC Advanced Trace Bus (ATB) Endpoint</td>
<td>0xFFD22800</td>
<td>0xFFD2287F</td>
<td>No</td>
</tr>
<tr>
<td>QOSMAIN</td>
<td>NoC quality-of-service (QoS) main</td>
<td>0xFFD24000</td>
<td>0xFFD2427F</td>
<td>No</td>
</tr>
</tbody>
</table>

---

¹ USB data FIFO is not included in model
<table>
<thead>
<tr>
<th>Module</th>
<th>Description</th>
<th>Start Address</th>
<th>End Address</th>
<th>Modeled in Virtual Platform?</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMA_NS</td>
<td>Non-secure DMA</td>
<td>0xFFDA0000</td>
<td>0xFFDA0FFF</td>
<td>Yes</td>
</tr>
<tr>
<td>DMA_S</td>
<td>Secure DMA</td>
<td>0xFFDA1000</td>
<td>0xFFDA1FFF</td>
<td>No</td>
</tr>
<tr>
<td>SPI0</td>
<td>SPI module 0 slave</td>
<td>0xFFDA2000</td>
<td>0xFFDA20FF</td>
<td>Yes</td>
</tr>
<tr>
<td>SPI1</td>
<td>SPI module 1 slave</td>
<td>0xFFDA3000</td>
<td>0xFFDA30FF</td>
<td>Yes</td>
</tr>
<tr>
<td>SPI2</td>
<td>SPI module 0 master</td>
<td>0xFFDA4000</td>
<td>0xFFDA40FF</td>
<td>Yes</td>
</tr>
<tr>
<td>SPI3</td>
<td>SPI module 1 master</td>
<td>0xFFDA5000</td>
<td>0xFFDA50FF</td>
<td>Yes</td>
</tr>
<tr>
<td>OCRAM</td>
<td>On-chip RAM module</td>
<td>0xFFE00000</td>
<td>0xFFE3FFFFF</td>
<td>Yes</td>
</tr>
<tr>
<td>GIC</td>
<td>General Interrupt Controller</td>
<td>0xFFFC1000</td>
<td>0xFFFC7FFF</td>
<td>Yes</td>
</tr>
</tbody>
</table>

### 1.12.2 Stratix 10 SoC Virtual Platform Host Interrupts

The table below lists the host interrupts available to the Stratix 10 SoC Virtual Platform.

**Table 3. Stratix 10 SoC Virtual Platform Interrupt Map**

<table>
<thead>
<tr>
<th>Stratix 10 SoC Virtual Platform Interrupt Request Number</th>
<th>Source</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>SYSMGR</td>
<td>System manager interrupt request</td>
</tr>
<tr>
<td>81</td>
<td>DMANS_IRQ0</td>
<td>Non-secure DMA interrupt request 0</td>
</tr>
<tr>
<td>82</td>
<td>DMANS_IRQ1</td>
<td>Non-secure DMA interrupt request 1</td>
</tr>
<tr>
<td>83</td>
<td>DMANS_IRQ2</td>
<td>Non-secure DMA interrupt request 2</td>
</tr>
<tr>
<td>84</td>
<td>DMANS_IRQ3</td>
<td>Non-secure DMA interrupt request 3</td>
</tr>
<tr>
<td>85</td>
<td>DMANS_IRQ4</td>
<td>Non-secure DMA interrupt request 4</td>
</tr>
<tr>
<td>86</td>
<td>DMANS_IRQ5</td>
<td>Non-secure DMA interrupt request 5</td>
</tr>
<tr>
<td>87</td>
<td>DMANS_IRQ6</td>
<td>Non-secure DMA interrupt request 6</td>
</tr>
<tr>
<td>88</td>
<td>DMANS_IRQ7</td>
<td>Non-secure DMA interrupt request 7</td>
</tr>
<tr>
<td>89</td>
<td>DMA_IRQ_Abort</td>
<td>Non-secure DMA abort interrupt</td>
</tr>
<tr>
<td>90</td>
<td>EMAC0</td>
<td>EMAC0 combined interrupt request</td>
</tr>
<tr>
<td>91</td>
<td>EMAC1</td>
<td>EMAC1 combined interrupt request</td>
</tr>
<tr>
<td>92</td>
<td>EMAC2</td>
<td>EMAC2 combined interrupt request</td>
</tr>
<tr>
<td>93</td>
<td>USB0</td>
<td>USB0 interrupt request</td>
</tr>
<tr>
<td>94</td>
<td>USB1</td>
<td>USB1 interrupt request</td>
</tr>
<tr>
<td>96</td>
<td>SDMMC</td>
<td>SDMMC interrupt request</td>
</tr>
<tr>
<td>99</td>
<td>SPI0</td>
<td>SPI0 interrupt request</td>
</tr>
<tr>
<td>100</td>
<td>SPI1</td>
<td>SPI1 interrupt request</td>
</tr>
<tr>
<td>101</td>
<td>SPI2</td>
<td>SPI2 interrupt request</td>
</tr>
<tr>
<td>102</td>
<td>SPI3</td>
<td>SPI3 interrupt request</td>
</tr>
</tbody>
</table>

*continued...*
<table>
<thead>
<tr>
<th>Stratix 10 SoC Virtual Platform Interrupt Request Number</th>
<th>Source</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>103</td>
<td>I2C0</td>
<td>I²C0 interrupt request</td>
</tr>
<tr>
<td>104</td>
<td>I2C1</td>
<td>I²C1 interrupt request</td>
</tr>
<tr>
<td>105</td>
<td>I2C2</td>
<td>I²C2 interrupt request</td>
</tr>
<tr>
<td>106</td>
<td>I2C3</td>
<td>I²C3 interrupt request</td>
</tr>
<tr>
<td>107</td>
<td>I2C4</td>
<td>I²C4 interrupt request</td>
</tr>
<tr>
<td>108</td>
<td>UART0</td>
<td>UART0 interrupt request</td>
</tr>
<tr>
<td>109</td>
<td>UART1</td>
<td>UART1 interrupt request</td>
</tr>
<tr>
<td>110</td>
<td>GPIO0</td>
<td>GPIO0 interrupt request</td>
</tr>
<tr>
<td>111</td>
<td>GPIO1</td>
<td>GPIO1 interrupt request</td>
</tr>
<tr>
<td>113</td>
<td>TIMER0</td>
<td>SP Timer0 interrupt request</td>
</tr>
<tr>
<td>114</td>
<td>TIMER1</td>
<td>SP Timer1 interrupt request</td>
</tr>
<tr>
<td>115</td>
<td>TIMER2</td>
<td>System Timer0 interrupt request</td>
</tr>
<tr>
<td>116</td>
<td>TIMER3</td>
<td>System Timer1 interrupt request</td>
</tr>
<tr>
<td>117</td>
<td>L4WD0</td>
<td>Watchdog timer 0 interrupt request</td>
</tr>
<tr>
<td>118</td>
<td>L4WD1</td>
<td>Watchdog timer 1 interrupt request</td>
</tr>
<tr>
<td>119</td>
<td>CLKMGR</td>
<td>Clock manager interrupt request</td>
</tr>
<tr>
<td>120</td>
<td>RSTMGR</td>
<td>Reset manager interrupt request</td>
</tr>
<tr>
<td>125</td>
<td>L4WD2</td>
<td>Watchdog timer 2 interrupt request</td>
</tr>
<tr>
<td>126</td>
<td>L4WD3</td>
<td>Watchdog timer 3 interrupt request</td>
</tr>
<tr>
<td>128-134, 136-169</td>
<td>SMMU</td>
<td>System MMU interrupt request</td>
</tr>
</tbody>
</table>
## 1.13 Revision History of Stratix 10 SoC Virtual Platform User Guide

<table>
<thead>
<tr>
<th>Date</th>
<th>Version</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>October 2016</td>
<td>2016.10.07</td>
<td>• Updated Stratix 10 SoC Virtual Platform Block Diagram&lt;br&gt;• Updated Installing the Stratix 10 SoC Virtual Platform section&lt;br&gt;• Updated Installing and Booting Pre-Built Linux for the Stratix 10 SoC Virtual Platform section&lt;br&gt;• Updated Installing Target Packages section&lt;br&gt;• Updated Building Linux section&lt;br&gt;• Updated tasks in Prerequisites for Updating Linux for the Stratix 10 SoC Virtual Platform&lt;br&gt;• Updated Updating the Stratix 10 SoC Virtual Platform Linux Kernel Image section&lt;br&gt;• Added Creating an SD Card Image for the Stratix 10 SoC Virtual Platform subsection to the Updating and Booting Linux with the Stratix 10 SoC Virtual Platform section</td>
</tr>
<tr>
<td>July 2016</td>
<td>2016.07.15</td>
<td>• Replaced parameters_Stratix 10.txt filename with parameters.txt throughout the document&lt;br&gt;• Removed Creating and SD Card Image for Stratix 10 SoC Virtual Platform section&lt;br&gt;• Updated Appendix A: Modifying the parameters.txt File</td>
</tr>
<tr>
<td>April 2016</td>
<td>2016.04.29</td>
<td>• Updated Stratix 10 SoC Virtual Platform Block Diagram&lt;br&gt;• Updated steps in &quot;Installing and Booting a Pre-Built Linux Kernel for the Stratix 10 SoC Virtual Platform.&quot;&lt;br&gt;• Modified steps in &quot;Building Linux&quot; section&lt;br&gt;• Updated &quot;Debugging Using the GDB Client on Host&quot;&lt;br&gt;• Added &quot;Creating an SD Card Image for Stratix 10 SoC Virtual Platform&quot;&lt;br&gt;• Added &quot;Executing the Stratix 10 SoC Virtual Platform&quot; section&lt;br&gt;• Modified &quot;Stratix 10 SoC Memory Map&quot;</td>
</tr>
<tr>
<td>March 2016</td>
<td>2016.03.04</td>
<td>• Updated Stratix 10 SoC Virtual Platform Block Diagram&lt;br&gt;• Modified &quot;Modules Modeled in Stratix 10 SoC Virtual Platform&quot; table in &quot;Stratix 10 SoC Virtual Platform Block Diagram&quot; section&lt;br&gt;• Modified code in &quot;Installing and Booting a Pre-Built Linux Kernel for the Stratix 10 SoC Virtual Platform&quot; section&lt;br&gt;• Updated &quot;Network Connectivity for Stratix 10 SoC Virtual Platform&quot;&lt;br&gt;• Modified code in &quot;Prerequisites for Updating Linux for the Virtual Platform&quot;&lt;br&gt;• Corrected port mapping in &quot;Initiating ssh from the Host to the Target&quot;&lt;br&gt;• Updated &quot;Appendix B: Memory and Interrupt Map&quot;&lt;br&gt;• Removed &quot;Appendix C: Known Issues&quot;</td>
</tr>
<tr>
<td>January 2016</td>
<td>2016.01.11</td>
<td>Corrected the link for the Stratix 10 SoC Virtual Platform download page.</td>
</tr>
<tr>
<td>December 2015</td>
<td>2015.12.07</td>
<td>Added the &quot;Setting up the Platform for Regression Testing&quot; and &quot;Extending the Platform with a Dynamic Library&quot; sections.</td>
</tr>
<tr>
<td>November 2015</td>
<td>2015.11.13</td>
<td>• Updated the block diagram&lt;br&gt;• Updated Appendix B&lt;br&gt;• Added Appendix C for known issues</td>
</tr>
<tr>
<td>October 2015</td>
<td>2015.10.07</td>
<td>Initial release</td>
</tr>
</tbody>
</table>