PetaLinux SDK User Guide

QEMU System Simulation Guide

UG982 (v2013.10) November 25, 2013

XILINX®

Revision History

Date	Version	Notes
2009-11-27	1.1	Initial version for SDK 1.1 release
2010-11-26	1.3	Updated for PetaLinux SDK 1.3 release and PowerPC support
2011-04-03	2.1	Updated for PetaLinux SDK 2.1 release (AXI device and CPU models
2012-08-03	3.1	Updated for PetaLinux SDK 3.1 release
2012-09-03	12.9	Updated for PetaLinux SDK 12.9 release
2012-12-17	2012.12	Updated for PetaLinux SDK 2012.12 release
2013-04-29	2013.04	Updated for PetaLinux SDK 2013.04 release
2013-11-25	2013.10	Updated for PetaLinux SDK 2013.10 release

Online Updates

Please refer to the PetaLinux v2013.10 Master Answer Record (Xilinx Answer Record #55776) for the latest updates on PetaLinux SDK usage and documentation.

XILINX

Table of Contents

About this Guide

This guide describes how to use the QEMU System Simulator included with PetaLinux SDK.

The ability to test and develop your software stack in a simulated environment allows your software and hardware design efforts to greatly improve parallelism, reducing overall development time. Please note: readers of this document are assumed to working knowledge of Linux and the PetaLinux tools. Refer to the PetaLinux Getting Started Guide for more details.

Software Simulation with QEMU

This document outlines a few scenarios for booting system images using the QEMU system simulator. The `petalinux-boot` tool, using the `--qemu` option, is used to boot the system simulator. Usage information can be obtained with `petalinux-boot --qemu --help`. The `petalinux-boot` tool must be run from within a project directory (`"<project-root>"`).

Prerequisites

This guide assumes that the following prerequisites are satisfied:

- You know how to build a PetaLinux software image (Please refer to the *PetaLinux SDK Getting Started Guide (UG977)*).
- The PetaLinux working environment has been set up . Run the following command to check whether the PetaLinux environment has been set up on the command console:

```
$ echo $PETALINUX
```

  If the PetaLinux working environment has been set up, it should show the path to the installed PetaLinux. If it shows nothing, please refer to the section Environment Setup in the *PetaLinux SDK Getting Started Guide (UG977)* .

IMPORTANT: *All command in this section are to be run from within a PetaLinux project directory.*

Exiting QEMU Simulator

Once QEMU is running, it can be exited by pressing `Ctrl + A` keys, then X.

Boot Built Linux Image

The `--kernel` option is used to boot the projects most recently built Linux image. For Microblaze this is `"<project-root>/images/linux/image.elf"`. For Zynq this is `"<project-root>/images/linux/zImage"`.

1. Build the system image
2. After the image has been built, change into the `"<project-root>"` directory if not already and run:

```
$ petalinux-boot --qemu --kernel
```

3. Watch the console, you will see the Linux boot process, ending with a login prompt:

```
INIT: version 2.88 booting
Starting Bootlog daemon: bootlogd.
Creating /dev/flash/* device nodes
Configuring network interfaces... udhcpc (v1.20.2) started
Sending discover...
Sending select for 10.0.2.15...
Lease of 10.0.2.15 obtained, lease time 86400
/etc/udhcpc.d/50default: Adding DNS 10.0.2.3
done.
starting Busybox inet Daemon: inetd... xemacps e000b000.ps7-ethernet: Set clk to 124999998 Hz
xemacps e000b000.ps7-ethernet: link up (1000/FULL)
done.
Starting uWeb server:
INIT: Entering runlevel: 5
Stopping Bootlog daemon: bootlogd.


 _____       _            _      _
| ___ \     | |          | |    (_)
| |_/ / ___ | |_   __ _ | |     _  _ __    _   _ __  __
|  __/ / _ \| __| / _‘ || |    | || ’_ \ | | | |\ \/ /
| |   |  __/| |_ | (_| || |____| || | | || |_| | >  <
\_|    \___| \__| \__,_|\_____/|_||_| |_| \__,_|/_/\_\

PetaLinux v2013.10 (Yocto 1.4) Xilinx-ZC702-2013_3 ttyPS0

Xilinx-ZC702-2013_3 login:
```

Figure 1: Boot PetaLinux Linux Image with QEMU

You may see slightly different output from the above example, it depends on the Linux image you test.

4. Login to the virtual system when you see the login prompt on the simulator console with username `root` and password `root`
5. Try some Linux commands such as `ls`, `ifconfig`, `cat /proc/cpuinfo` and so on. They behave the same as on real hardware.
6. To exit the simulator, press `Ctrl + A`, release and then `X`.

Direct Boot a Specific Linux Image

`petalinux-boot` can also boot a specific Linux image, using the `image` option (`-i` or `--image`):

```
$ petalinux-boot --qemu --image <path-to-Linux-image-file>
```

e.g.:

```
$ petalinux-boot --qemu --image ./images/linux/zImage
```

Direct Boot a Linux Image with Specific DTB

Device Trees (DTB files) are used to describe the hardware architecture and address map to the Linux kernel. The PetaLinux system simulator also uses DTB files to dynamically configure the simulation environment to match your hardware platform.

If no DTB file option is provided, `petalinux-boot` extracts the DTB file from the given `image.elf` for Microblaze and from "`<project-root>/images/linux/system.dtb`" for Zynq. Alternatively, you can use the `--dtb` option as follows:

```
$ petalinux-boot --qemu --image ./images/linux/zImage --dtb ./images/linux/system.dtb
```

Virtual IO

WARNING: *The steps described in this section can cause changes to physical storage media attached to your host PC, including SD cards and USB-attached hard disks. Double-check before performing any host tunneling of storage media to avoid unintended data loss.*

Serial Ports

By default, `petalinux-boot --qemu` searches for the `stdout` serial device in the DTS file and passes the `--serial mon:stdio` option to QEMU for this serial device. It uses `--serial /dev/null` for all other serial devices described in the DTS.

Storage

IMPORTANT: *The features described in this section apply only to Zynq.*

By default, `petalinux-boot --qemu` does not use any backing files for any storage devices (e.g. Flash).

SD Card

The following section describes how to attach a block device for non-volatile SD card storage in QEMU. That is, a physical SD card, inserted in your host PC, will be "tunneled" through to the QEMU virtual machine.

Ensure that the SD device tree node in the platform's DTS file is valid. See the example below.

```
ps7_sd_0: ps7-sdio@e0100000 {
    clock-frequency = <50000000>;
    clock-names = "ref_clk", "aper_clk";
    clocks = <&clkc 21>, <&clkc 32>;
    compatible = "xlnx,ps7-sdio-1.00.a", "generic-sdhci", "arasan,sdhci";
    interrupt-parent = <&ps7_scugic_0>;
    interrupts = <0 24 4>;
    reg = <0xe0100000 0x1000>;
    xlnx,has-cd = <0x1>;
    xlnx,has-power = <0x0>;
    xlnx,has-wp = <0x1>;
} ;
```

You need to create a backing file for the virtual SD card, then you can use the `-sd` option to tell the QEMU the SD card backing file. E.g.

```
$ # Create a 512MB backing file of zeros
$ dd if=/dev/zero of=sdimage.bin bs=1M count=512
$ petalinux-boot --qemu --kernel --qemu-args "-sd sdimage.bin"
```

TIP: *If you specify a real SD card, such as `/dev/mmcblock0` as the `-sd` option, then QEMU will boot as though that SD card is connected to the virtual machine. Be aware that this will change the contents of that physical device.*

USB

IMPORTANT: *Only USB Host mode emulation is supported in QEMU.*

1. To access your USB device (e.g. USB Flash Drive) you will need to make sure the USB device tree node in the platform's DTS is valid. See the example below. Make sure the `dr_mode` is set to `host`. You will need to run QEMU if your USB devices require root access.

```
ps7_usb_0: ps7-usb@e0002000 {
    clocks = <&clkc 28>;
    compatible = "xlnx,ps7-usb-1.00.a";
    dr_mode = "host";
    interrupt-parent = <&ps7_scugic_0>;
    interrupts = <0 21 4>;
    phy_type = "ulpi";
    reg = <0xe0002000 0x1000>;
    usb-reset = <&ps7_gpio_0 7 0>;
} ;
```

2. Get the Vendor ID and the Product ID of the USB device using the `lsusb` command on the host.
3. Use `-usb -usbdevice host:<VendorID>:<ProductID>` QEMU option to enable QEMU USB host support and attach the specified device. See the following example (to be entered all on one command line):

```
$ petalinux-boot --root --qemu --kernel --qemu-args \
          "-usb -usbdevice host:<VENDORID>:<PRODUCTID>"
```

Going Further

QEMU Virtual Networking Modes

There are two execution modes in QEMU: non-root (the default) and root (requires sudo or root permission). The difference in the modes relates to virtual network configuration.

In non-root mode QEMU sets up an internal virtual network which restricts network traffic passing from the host and the guest. This works similar to a NAT router. You can not access this network unless you redirect tcp ports.

In root mode QEMU creates a subnet on a virtual ethernet adapter, and relies on a DHCP server on the host system.

The following sections detail how to use the modes, including redirecting the non-root mode so it is accessible from your local host.

Redirecting ports in non-root mode

If running QEMU in the default non-root mode, and you wish to access the internal (virtual) network from your host machine (e.g.to debug with either GDB or TCF Agent), you will need to forward the emulated system ports from inside the QEMU virtual machine to the local machine.
The `petalinux-boot --qemu` command utilises the `--qemu-args` option to perform this redirection.
The following table outlines some examples redirection arguments. This is standard QEMU functionality, please refer to the QEMU documentation for more details.

QEMU options switch	Purpose	Accessing guest from host
`-tftp <path-to-directory>`	Sets up a TFTP server at the specified directory, the server is available on the QEMU internal IP address of 10.0.2.2.	
`-redir tcp:10021:10.0.2.15:21`	Redirects port 10021 on the host to port 21 (ftp) in the guest	`host> ftp localhost 10021`
`-redir tcp:10023:10.0.2.15:23`	Redirects port 10023 on the host to port 23 (telnet) in the guest	`host> telnet localhost 10023`
`-redir tcp:10080:10.0.2.15:80`	Redirects port 10080 on the host to port 80 (http) in the guest	Type http://localhost:10080 in the web broswer
`-redir tcp:10022:10.0.2.15:22`	Redirects port 10022 on the host to port 22 (ssh) in the guest	Run `ssh -P 10022 localhost` on the host to open a SSH session to the target

The following example shows the command line used to redirect ports:

```
$ petalinux-boot --qemu --kernel --qemu-args "-redir tcp:1534::1534"
```

The Application Development Guide assumes the use of port 1534 for gdbserver and tcf-agent, but it is possible to redirect to any free port. The internal emulated port can also be different from the port on the local machine:

```
$ petalinux-boot --qemu --kernel --qemu-args "-redir tcp:1444::1534"
```

Specifying the QEMU Virtual Subnet

By default, PetaLinux uses `192.168.10.*` as the QEMU virtual subnet in `--root` mode. If it has been used by your local network or other virtual subnet, you may wish to use another subnet. You can configure PetaLinux to use other subnet settings for QEMU by running `petalinux-boot` as follows on the command console:

WARNING: *This feature requires sudo access on the local machine, and must be used with the `--root` option.*

```
$ petalinux-boot --qemu --root --uboot --subnet <subnet gateway IP>/
    <number of the bits of the subnet mask>
```

e.g., to use subnet `192.168.20.*`:

```
$ petalinux-boot --qemu --root --uboot --subnet 192.168.20.0/24
```

Debugging the Linux Kernel in QEMU

You can debug the MicroBlaze Linux Kernel inside QEMU, using the GDB debugger. The default TCP port for MicroBlaze QEMU is `9000`:

1. Launch QEMU with the currently built Linux by running the following command:

   ```
   $ petalinux-boot --qemu --kernel
   ```

2. Open another command console (ensuring PetaLinux settings script has been sourced), and change to the Linux directory:

   ```
   $ cd "<project-root>/images/linux"
   ```

3. Start GDB on the `vmlinux` kernel image in command mode:

   ```
   $ petalinux-util --gdb vmlinux
   ```

   You should see the `gdb` prompt. e.g.:

```
GNU gdb (crosstool-NG 1.18.0) 7.6.0.20130721-cvs
Copyright (C) 2013 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 "--host=x86_64-build_unknown-linux-gnu
        --target=microblazeel-xilinx-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
```

4. Attach to the QEMU target in GDB by running the following GDB command:

```
(gdb) target remote :9000
```

5. To let QEMU continue execution:

```
(gdb) continue
```

6. You can use `Ctrl+C` to interrupt the kernel and get back the GDB prompt.
7. You can set break points and run other GDB commands to debug the kernel.

WARNING: *If another process is using port 9000, `petalinux-boot` will attempt to use a different port. Look at the output of `petalinux-boot` to determine what port was used. In the following example port 9001 was used.*

```
INFO: TCP PORT is free 9001 ip = 11 INFO: Found 1 Ethernet devices
```

TIP: *It may be helpful to enable kernel debugging in the kernel configuration menu (`petalinux-config --kernel` > **Kernel hacking** > **Kernel debugging**), so that kernel debug symbols are present in the image.*

More about petalinux-boot

This guide has introduced the most commonly used modes and options of `petalinux-boot --qemu`. To learn more, run it with the `--help` option:

```
$ petalinux-boot --qemu --help
```

The detailed explanation of each option will be shown on the console.

Troubleshooting

This section describes some common issues you may experience when creating and testing user application, and ways to solve them.

Problem/Error Message	Description and Solution

QEMU failed to get IP address.	***Problem Description:***
	This is most likely because your firewall blocking DHCP requests.
	
	Solution:
	use `ifconfig` to assign an IP belonging to the QEMU virtual subnet to the system on the QEMU console:
	
	`# ifconfig eth0 <system IP>`
	
	e.g., to use `192.168.10.3`:
	
	`# ifconfig eth0 192.168.10.3`
	
	Alternatively, contact your system administrator to allow DHCP request to your workstation.

(gdb) target remote W.X.Y.Z:9000 :9000: Connection refused.	***Problem Description:*** GDB failed to attach the QEMU target. This is most likely because the port `9000` is not the one QEMU is using. ***Solution:*** 1. Check your QEMU console to make sure QEMU is running. 2. Try port `9001` on the GDB console: `(gdb) target remote :9001` The PetaLinux always tries to use port `9000` for QEMU GDB. If the port has been used, it will increase the port number by 1 until it can get a free port.

QEMU Supported Xilinx IP Models

The QEMU simulator supports a limited number of Xilinx IP models.

- Zynq-7000 ARM Cortex-A9 CPU
- Zynq-7000 ARM Cortex-A9 MPCore
- MicroBlaze CPU (little-endian AXI)
- Xilinx Zynq Triple Timer Counter
- Xilinx Zynq DDR Memory Controller
- Xilinx Zynq DMA Controller
- Xilinx Zynq Static Memory Controller (NAND/NOR Flash)
- Xilinx Zynq SD/SDIO Peripheral Controller
- Xilinx Zynq Gigabit Ethernet Controller
- Xilinx Zynq USB Controller (Host support only)
- Xilinx Zynq UART Controller
- Xilinx Zynq SPI Controller
- Xilinx Zynq QSPI Controller
- Xilinx Zynq I2C Controller
- Xilinx AXI Timer and Interrupt controller peripherals
- Xilinx AXI External Memory Controller connected to parallel flash
- Xilinx AXI DMA Controller
- Xilinx AXI Ethernet
- Xilinx AXI Ethernet Lite
- Xilinx AXI UART 16650 and Lite
- Xilinx AXI SPI

By default, QEMU will disable any devices for which there is no model available. For this reason it is not possible to use QEMU to test your own customized IP Cores.

Additional Resources

References

- PetaLinux SDK Application Development Guide (UG981)
- PetaLinux SDK Board Bringup Guide (UG980)
- PetaLinux SDK Firmware Upgrade Guide (UG983)
- PetaLinux SDK Getting Started Guide (UG977)
- PetaLinux SDK Installation Guide (UG976)
- PetaLinux SDK QEMU System Simulation Guide (UG982)

PetaLinux SDK Documentation is available at http://www.xilinx.com/petalinux.