PetaLinux SDK User Guide

QEMU System Simulation Guide

UG982 (v2012.12) December 17, 2012

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

Table of Contents

About this Guide

This guide describes how to use the QEMU System Simulator included with PetaLinux SDK. The system simulator in PetaLinux SDK has the unique capability to automatically configure the simulation model to match the architecture of your target hardware system.

The ability to test and develop your software stack in a simulation environment allows your software and hardware design efforts to greatly improve parallelism, reducing overall development time.

- As of PetaLinux SDK v3.1, Zynq target architecture is supported.
- As of PetaLinux SDK v1.3, both MicroBlaze and PowerPC440 target architectures are supported.

Please note: readers of this document are assumed to have basic Linux knowledge such as how to run simple Linux commands.

Related PetaLinux Documents

The following other documents exist to help you to make the most of your PetaLinux experience:

- Application Development Guide
- Getting Started Guide
- Installation Guide

PetaLinux Software Simulation with QEMU

The `petalinux-qemu-boot` tool is used to boot a PetaLinux kernel image in the system simulator.

Prerequisites

This guide assumes that the following prerequisites have been satisfied:

- PetaLinux SDK and at least one PetaLinux BSP has been installed.
- Your user has sudo permission (required for QEMU virtual networking). For assistance with sudo configuration please contact your local system or network administrator.
- You know how to build PetaLinux software image (Please refer to the PetaLinux Getting Started Guide).
- Your workstation has `tftpd` server running.
- There exists a `/tftpboot` directory on your workstation and all users have read/write permissions to it.
- PetaLinux working environment has been setup in each command console with which you work with PetaLinux. Run the following command to check whether the PetaLinux environment has been setup on the command console:

  ```
  $ echo $PETALINUX
  ```

  If the PetaLinux working environment has been setup, it should show the path to the installed PetaLinux. If it shows nothing, please refer to section `"Environment Setup"` in the `"Getting Started with PetaLinux SDK"` document to setup the environment.

Boot Recently Built Linux Image

If no argument is given, `petalinux-qemu-boot` boots the most recently built Linux image (from the `"$PETALINUX/software/petalinux-dist/image.elf"` system image).

1. Build the PetaLinux software image
2. After the software image has been successfully built, run `petalinux-qemu-boot` on the command console:

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

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

```
Mounting devpts:
Mounting all filesystem
Setting hostname:
Bringing up network interfaces:
GEM: lp->tx_bd ffdfb000 lp->tx_bd_dma 2e833000 lp->tx_skb ef1cb4c0
GEM: lp->rx_bd ffdfc000 lp->rx_bd_dma 2e83b000 lp->rx_skb ef1cb2c0
GEM: MAC 0x00350a00, 0x000002b2, 00:0a:35:00:b2:02
udhcpc (v1.14.3) started
Sending discover...
Sending select for 192.168.10.2...
Lease of 192.168.10.2 obtained, lease time 864000
adding dns 192.168.1.1
adding dns 192.168.1.254
Starting portmap:
Starting uWeb server:

Welcome to
 _____       _            _       _
| ___ \     | |          | |     (_)
| |_/ / ___ | |_   __ _ | |      _  _ __    _   _ __  __
|  __/ / _ \| __| / _‘ || |     | || '_ \ | | | |\ \/ /
| |   |  __/| |_ | (_| || |____ | || | | || |_| | >  <
\_|    \___| \__| \__,_|\_____/|_||_| |_| \__,_|/_/\_\


on Xilinx-ZC702-14.4


Xilinx-ZC702-14.4 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.
5. Try some Linux commands such as `"ls"`, `"ifconfig"`, `"cat /proc/cpuinfo"` and so on. The same as playing with Linux on hardware.
6. To exit the simulator, press `"Ctrl+A"` then `"X"`.

Direct Boot a Specific Linux Image

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

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

e.g.:

```
$ petalinux-qemu-boot -i images/image.elf
```

IMPORTANT: *petalinux-qemu-boot can only boot ELF images*

Direct Boot a Linux Image with Specified 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 exactly match your hardware model.

If no DTB file option is provided, as in the previous two examples, petalinux-qemu-boot extracts the DTB file from the given ELF image and use it. Alternatively, you specify a particular DTB file by giving the DTB command option(`-d` or `--dtb`) as follows:

```
$ petalinux-qemu-boot -d <path-to-DTB-file>
```

e.g. for a MicroBlaze system:

```
$ petalinux-qemu-boot -d linux-2.6.x/arch/microblaze/boot/system.dtb
```

or PowerPC:

```
$ petalinux-qemu-boot -d linux-2.6.x/target.dtb
```

Note that the Linux PowerPC build system stores the device tree in a slightly different location from the MicroBlaze build, however in most cases you can ignore this fact.

QEMU boot Linux Image via U-boot

In addition to the kernel, it is also possible to boot the u-boot bootloader in QEMU. This is useful for example to prepare and test custom u-boot command scripts or other settings. It is even possible to replicate the u-boot / tftpboot / Linux kernel boot process, all inside QEMU!

IMPORTANT: *This feature requires "sudo" access on the local machine to setup the QEMU virtual network, and does not work with the "--noroot" option.*

1. Boot u-boot with QEMU by running the `petalinux-qemu-boot` as follows on the command console:

   ```
   $ petalinux-qemu-boot -u
   ```

   If no DTB file is specified with the `--dtb` option, and there is no DTB file embedded in the ELF file (such as with the `u-boot.elf` image), then the most recently compiled DTB is used instead. For MicroBlaze this is:

   ```
   $PETALINUX/software/linux-2.6.x/arch/microblaze/boot/system.dtb
   ```

   While for PowerPC it is:

```
$PETALINUX/software/linux-2.6.x/target.dtb
```

2. Watch the console, you will see the u-boot booting messages.
3. Hit any key to stop auto boot when you see `"Hit any key to stop autoboot:"` on the console. e.g:

```
U-Boot 2011.06
Xilinx Zynq Platform

DRAM:  1 GiB
MMC:   SDHCI: 0
Using default environment

In:    serial
Out:   serial
Err:   serial

Net:   XGem.e000b000

U-BOOT for Xilinx-ZC702-14.4

Resetting PHY...

PHY reset complete.

Waiting for PHY to complete autonegotiation.
PHY claims autonegotiation complete...
GEM link speed is 100Mbps
BOOTP broadcast 1
DHCP client bound to address 192.168.10.2
Hit any key to stop autoboot:  0
U-Boot-PetaLinux>
```

Figure 2: QEMU U-boot

4. Run u-boot command print on the QEMU console to check whether the u-boot variable serverip has been set to the right TFTP server from which to download the Linux image:

```
U-Boot-PetaLinux> print serverip
```

By default, this should be `192.168.10.1` - the address of your Linux workstation in the QEMU virtual subnet. If this value is not correct, run u-boot command set to change it on the QEMU console:

```
U-Boot-PetaLinux> set serverip <TFTP serverip IP>
```

5. Run the netboot command to boot the virtual processor via TFTP over the virtual network:

```
U-Boot-PetaLinux> run netboot
```

6. Watch the QEMU console, you should see the Linux image booting messages.
7. When you see the Linux login prompt on the QEMU console, login and now, you are using Linux on a virtual processor.

Going Further

Running QEMU without sudo access

By default QEMU requires "sudo" access to the local machine. This access is required for setting up the virtual networking between the local machine and the emulated system.

In order to be able to use the virtual networking features of QEMU you must have local "sudo" access.

If the virtual networking is not required you can run QEMU with "--noroot". This will allow QEMU to execute without "sudo" access, and emulate the target system without networking.

```
$ petalinux-qemu-boot --noroot
```

In this mode the network device will appear however it will not be able to communicate with the host.

Specifying the QEMU Virtual Subnet

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

IMPORTANT: *This feature requires "sudo" access on the local machine, and does not work with the "--noroot" option.*

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

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

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

Access Internet with QEMU

By default, the Linux boot via QEMU cannot access the Internet or other network addresses outside the virtual QEMU network.

IMPORTANT: *This feature requires "sudo" access on the local machine, and does not work with the "--noroot" option.*

petalinux-qemu-boot provides an option to allow QEMU access the Internet:

```
$ petalinux-qemu-boot --iptables-allowed
```

This command will add a rule to the iptables FORWARD chain to allow traffic from the QEMU virtual subnet to eth0 to any destination. This rule will be deleted when the QEMU is terminated.

If you test the pre-built MicroBlaze software image with QEMU, you can run `petalinux-boot-prebuilt` as follows to allow QEMU access the Internet:

```
$ petalinux-boot-prebuilt -p <reference platform> -q --qemu-args
    --iptables-allowed
```

Please note that the use of the `--iptables-allowed` option does modify your PC firewall rules and state. While every effort is made to restore any changes to the state, it is not guaranteed to restore the state. If you have any doubt, or you are in a secure networking environment, **DO NOT USE THIS OPTION**.

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-qemu-boot
   ```

2. Open another command console and go to the Linux directory:

   ```
   $ cd $PETALINUX/software/petalinux-dist/linux-2.6.x
   ```

3. Start GDB on the vmlinux kernel image in command mode inside `linux-2.6.x` directory:

   ```
   $ microblaze-unknown-linux-gnu-gdb vmlinux
   ```

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

   ```
   GNU gdb 6.5.0.20060626-cvs
   Copyright (C) 2006 Free Software Foundation, Inc.
   GDB is free software, covered by the GNU General Public License, and you
   are welcome to change it and/or distribute copies of it under certain
   conditions. Type "show copying" to see the conditions.
   There is absolutely no warranty for GDB.  Type "show warranty" for details.
   This GDB was configured as
   "--host=i686-pc-linux-gnu --target=microblaze-unknown-linux-gnu"...
   (gdb)
   ```

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.

It may be helpful to enable kernel debugging in the kernel configuration menu (`make kconfig`), so that kernel debug symbols are present in the image.

More about petalinux-qemu-boot

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

```
$ petalinux-qemu-boot --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. If you cannot solve your problem after referring to the table below please open a support ticket.

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.

- MicroBlaze CPU (big-endian PLB and little-endian AXI)
- PowerPC440 CPU and processor block, including DMA engines
- Zynq-7000 ARM Cortex-A9 CPU
- Zynq-7000 ARM Cortex-A9 MPCore
- Xilinx Timer and Interrupt controller peripherals (PLB and AXI)
- Xilinx External Memory Controller connected to parallel flash (PLB and AXI)
- Xilinx UART 16650 (PLB and AXI)
- Xilinx UART Lite (PLB and AXI)
- Xilinx Ethernet Lite (PLB and AXI)
- Xilinx SPI (PLB and AXI)
- Xilinx LL TEMAC (PLB)
- Xilinx AXI DMA Controller
- Xilinx AXI Ethernet
- 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

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.