



PetaLinux SDK Guide to QEMU System Simulation

v2.1 April 3rd, 2011



Table of Contents

Table of Contents	2
About This Guide	3
Related PetaLinux Documents	3
PetaLinux Software Simulation with QEMU	3
Prerequisites.....	3
Boot Recently Built Linux Image.....	4
Direct Boot a Specific Linux Image.....	5
Direct Boot a Linux Image with Specified DTB.....	6
QEMU boot Linux Image via U-boot.....	6
Going Further	8
Specifying the QEMU Virtual Subnet.....	8
Access Internet with QEMU.....	9
Debugging the Linux Kernel in QEMU.....	9
More about petalinux-qemu-boot.....	11
Trouble Shooting	12
QEMU Supported Xilinx IP Models	13
Revision History	13

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 be more greatly parallelised, reducing overall development time.

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:

- PetaLinux SDK Installation Guide
- Getting Started with PetaLinux SDK
- Creating and Debugging User Applications in PetaLinux SDK

You can also find PetaLinux documentation online at

- <http://www.petalogix.com/resources/documentation>

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 have 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:

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 match exactly 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!

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

```
$ petalinux-qemu-boot -i <path-to-u-boot-ELF-file>
```

Be sure to enter the command on one line

e.g.:

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

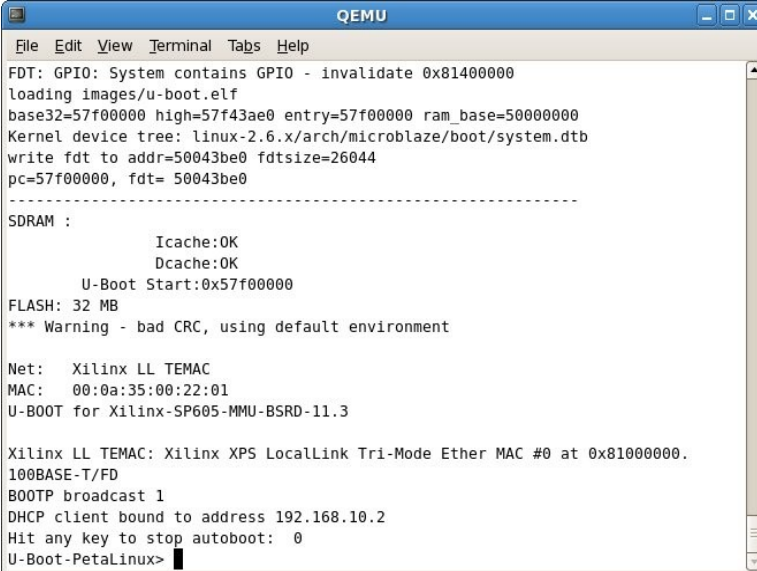
If no DTB file is specified with the `--dtb` option, and there is no DTB file embedded in the ELF file (such as with the `u-boot.elf` image), then the most recently compiled DTB is used instead. For MicroBlaze this is

```
$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:

A screenshot of a QEMU terminal window. The window title is "QEMU" and it has a menu bar with "File", "Edit", "View", "Terminal", "Tabs", and "Help". The terminal output shows the following text:

```
FDT: GPIO: System contains GPIO - invalidate 0x81400000  
loading images/u-boot.elf  
base32=57f00000 high=57f43ae0 entry=57f00000 ram_base=50000000  
Kernel device tree: linux-2.6.x/arch/microblaze/boot/system.dtb  
write fdt to addr=50043be0 fdtsize=26044  
pc=57f00000, fdt= 50043be0  
-----  
SDRAM :  
          Icache:OK  
          Dcache:OK  
          U-Boot Start:0x57f00000  
FLASH: 32 MB  
*** Warning - bad CRC, using default environment  
  
Net:   Xilinx LL TEMAC  
MAC:   00:0a:35:00:22:01  
U-BOOT for Xilinx-SP605-MMU-B5RD-11.3  
  
Xilinx LL TEMAC: Xilinx XPS LocalLink Tri-Mode Ether MAC #0 at 0x81000000.  
100BASE-T/FD  
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 MicroBlaze or PowerPC 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 MicroBlaze or PowerPC440 system.

Going Further

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:

```
$ 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. `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, and while every effort is made to restore that state, PetaLogix does not accept or assume any responsibility for any loss or damage that may occur through inadvertant changes to firewall 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 -nw 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.

Trouble Shooting

Issue/Error Message	Solution
<p>QEMU failed to get IP address.</p> <p>When you run <code>ifconfig</code> on the QEMU console, you may get this output if QEMU failed to get IP address:</p> <pre> ~ # ifconfig eth0 Link encap:Ethernet HWaddr 00:0A:35:00:22:01 UP BROADCAST RUNNING MTU:1500 Metric:1 RX packets:10 errors:0 dropped:0 overruns:0 frame:0 TX packets:11 errors:0 dropped:0 overruns:0 carrier:0 collisions:0 txqueuelen:1000 RX bytes:3200 (3.1 KiB) TX bytes:5394 (5.2 KiB) Interrupt:2 </pre>	<p>This is most likely because your firewall blocks DHCP request.</p> <p>Steps of the solution:</p> <ol style="list-style-type: none"> 1. use <code>ifconfig</code> to assign an IP belonging to the QEMU virtual subnet to the system on the QEMU console: <pre> # ifconfig eth0 <system IP> </pre> <p>E.g., to use 192.168.10.3:</p> <pre> # ifconfig eth0 192.168.10.3 </pre> <p>Alternatively, contact your system administrator to allow DHCP request to your workstation.</p>
<p>(gdb) target remote W.X.Y.Z:9000 :9000: Connection refused.</p>	<p>GDB failed to attach the QEMU target. This is most likely because the port “9000” is not the one QEMU is using.</p> <p>Steps of the solution:</p> <ol style="list-style-type: none"> 1. Check your QEMU console to make sure QEMU is running. 2. Try port “9001” on the GDB console: <pre> (gdb) target remote :9001 </pre> <p>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.</p>

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
- UART 16650 (PLB and AXI)
- Xilinx UART lite (PLB and AXI)
- Xilinx Ethernet lite (PLB and AXI)
- Xilinx LL_TEMAC (PLB)
- Xilinx AXI Ethernet
- Xilinx AXI DMA
- Timer and Interrupt controller peripherals (PLB and AXI)
- Xilinx External Memory Controller connected to parallel flash (PLB and AXI)

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. If you want to use QEMU to test your customised IP Core, you should contact PetaLogix at enquiries@petalogix.com to discuss the development and integration of a custom IP model.

Revision History

Date	Version	Notes
11/27/2009	1.00	Initial version for SDK 1.1 release
11/26/2010	1.3	Updated for PetaLinux 1.3 release and PowerPC support.
04/03/11	2.1	Updated for PetaLinux 2.1 release (AXI device and CPU models)