RCE Diskless Node (NFS)

Starting with RCE SDK V1.7.0, an RCE may load Linux over NFS, and run as a diskless node.

At boot, the following files are copied over NFS into bootloader memory:

- uImage - RCE Linux kernel
- devicetree.dtb - RCE device tree
- fpga.bit - RCE firmware bitfile

At Linux boot, the NFS served Arch Linux ARM installation is mounted as the root filesystem.

Please note that this feature is currently experimental!

Use at your own risk!

Unresolved issues include:

- Node unique SSH key generation and storage
- Dynamic read/write access to root filesystem files and directories
- Arch Linux ARM package maintenance (pacman)
- Bootloader NFS read performance

Install the SDK

First, download and install the latest RCE SDK.

Fetch the Arch Linux ARM Installation

Download and extract the Arch Linux filesystem to an NFS served directory on your host system.

wget http://os.archlinuxarm.org/os/ArchLinuxARM-zedboard-latest.tar.gz

tar -xzf ArchLinuxARM-zedboard.latest.tar.gz

The directory your Arch Linux installation is extracted to will be referred to as <arch_linux_root>.

Prepare the Arch Linux installation

```
mkdir <arch_linux_root>/boot (if not already present)
```

```
mkdir <arch_linux_root>/var/lib/machines
```

Copy the following files from the RCE SDK to the Arch Linux filesystem:

```
cp arm-linux/tgt/linux/fstab.diskless <arch_linux_root>/etc/fstab
```

```
cp arm-linux/tgt/linux/kernel/uImage to <arch_linux_root>/boot
```

```
cp arm-linux/tgt/linux/kernel/devicetree.dtb to <arch_linux_root>/boot
```

Copy your own firmware file to <arch_linux_root>/boot/fpga.bit

Or download and copy the default firmware bitfile for your RCE to <arch_linux_root>/boot/fpga.bit

For diskless DTMs, use a different filename than DPMs, like <arch_linux_root>/boot/fpga_dtm.bit

wget http://www.slac.stanford.edu/projects/CTK/SDK/fpga.dpm.bit

wget http://www.slac.stanford.edu/projects/CTK/SDK/fpga.dtm.bit

Remove the following service file links:

```
rm <arch_linux_root>/etc/systemd/system/sockets.target.wants/systemd-networkd.socket
```

```
rm <arch_linux_root>/etc/systemd/system/multi-user.target.wants/systemd-networkd.service
```

For console or SSH login, allow read access to shadow:

```
chmod a+r <arch_linux_root>/etc/shadow
```

For SSH usage, allow root login permission by modifying the SSH daemon configuration:

edit <arch_linux_root>/etc/ssh/sshd_config - uncomment PermitRootLogin and set value to yes

For SSH usage, allow read/write access for ssh key generation:

```
chmod a+rw <arch_linux_root>/etc/ssh
```

Change back to read-only after first boot

Add RCE core software (optional)

To make use of the RCE core software on the diskless node, copy the following files from the RCE SDK:

```
cp arm-linux/bin/* <arch_linux_root>/bin
cp arm-linux/lib/* <arch_linux_root>/lib
cp arm-linux/tgt/linux/rced.service <arch_linux_root>/etc/systemd/system/multi-user.target.wants
```

COB DTM only:

```
cp arm-linux/tgt/linux/fmd.service <arch_linux_root>/etc/systemd/system/multi-user.target.wants
cp arm-linux/tgt/linux/packages/system/minirc/* <arch_linux_root>/etc
cp <arch_linux_root>/etc/minirc.bay0.0 <arch_linux_root>/etc/minirc.dfl
```

DTM DHCP server only:

```
cp arm-linux/tgt/linux/dtm_dhcp.sh <arch_linux_root>/bin
cp arm-linux/tgt/linux/dhcpd.dtm.conf <arch_linux_root>/etc
cp arm-linux/tgt/linux/dtm_dhcp.service <arch_linux_root>/etc/systemd/system/multi-user.target.wants
```

Configure the NFS server

Make sure <arch_linux_root> is exported by your NFS server

Configure the DHCP server

Set the next-server parameter to the IP address of the NFS server.

```
next-server <nfs_server_ip_address>
```

Set the root filesystem path to the Arch Linux installation

```
option root-path "<arch_linux_root>"
```

See arm-linux/tgt/linux/dhcpd.dtm.conf in the RCE SDK for example entries.

Prepare the RCE bootloader

First, update the RCE bootloader (boot.bin) with the version in the latest SDK.

Reboot the RCE with default firmware (reboot_rce -t linux -b 0)

Boot the RCE into the bootloader.

For non-eFUSE RCEs, make note of the ethaddr variable for later restoration.

Apply the default u-boot environment:

```
zynq-uboot> env default -a
## Resetting to default environment
```

You may set nfs boot as the default boot option prior to saving the environment

```
zynq-uboot> setenv modeboot nfsboot
```

You may need to restore the MAC address (for non-eFUSE RCEs)

```
zynq-uboot> setenv ethaddr <mac_address>
```

You may need to restore the boot delay parameter (for autoboot)

```
zynq-uboot> setenv bootdelay 3
```

DTM Firmware FPGA Bitfile

DTMs use different firmware bit files than DPMs.

The default filename of fpga.bit must be updated to reflect the name of the DTM file located in <arch_linux_root>/boot.

In this example, the DTM firmware bitfile is named fpga_dtm.bit

```
zynq-uboot> setenv nfsfpga 'nfs 0x1000000 ${rootpath}/boot/fpga_dtm.bit && fpga loadb 0 0x1000000 ${filesize}'
```

Save the environment and reset:

```
zynq-uboot> saveenv
Saving Environment to FAT...
writing uboot.env
done
zynq-uboot> reset
resetting ...
```

Boot Using NFS

Boot the RCE into the bootloader.

Execute the nfs boot command:

```
zynq-uboot> run nfsboot
```

Sample output:

```
zynq-uboot> run nfsboot
PHY not detected, assuming PHY at address 0
BOOTP broadcast 1
DHCP client bound to address 192.168.204.38
PHY not detected, assuming PHY at address 0
Gem.e000b000:0 is connected to Gem.e000b000. Reconnecting to Gem.e000b000
Using Gem.e000b000 device
File transfer via NFS from server 192.168.204.12; our IP address is 192.168.204.38
Filename '/nfsexport/users/smaldona/dat/arch-linux-arm/arch-linux/boot/uImage'.
Load address: 0x3000000
Loading: ##################################################################
 #########################
done
Bytes transferred = 3784144 (39bdd0 hex)
PHY not detected, assuming PHY at address 0
Gem.e000b000:0 is connected to Gem.e000b000. Reconnecting to Gem.e000b000
Using Gem.e000b000 device
File transfer via NFS from server 192.168.204.12; our IP address is 192.168.204.38
Filename '/nfsexport/users/smaldona/dat/arch-linux-arm/arch-linux/boot/devicetree.dtb'.
Load address: 0x2a00000
Loading: ##
done
Bytes transferred = 9984 (2700 hex)
PHY not detected, assuming PHY at address 0
Gem.e000b000:0 is connected to Gem.e000b000. Reconnecting to Gem.e000b000
Using Gem.e000b000 device
File transfer via NFS from server 192.168.204.12; our IP address is 192.168.204.38
Filename '/nfsexport/users/smaldona/dat/arch-linux-arm/arch-linux/boot/fpga.bit'.
Load address: 0x1000000
Loading: ##################################################################
 ##
done
Bytes transferred = 13321492 (cb4514 hex)
 design filename = "DpmTest;UserID=0XFFFFFFFF"
 part number = "7z045ffg900"
 date = "2014/10/22"
 time = "15:38:09"
 bytes in bitstream = 13321404
zynq_load: Bitstream is not swapped(1) - swap it
Net: mac 08:00:56:00:44:3f
## Booting kernel from Legacy Image at 03000000 ...
 Image Name: Linux-4.4.0-xilinx-00025-g96ce8f
 Image Type: ARM Linux Kernel Image (uncompressed)
 Data Size: 3784080 Bytes = 3.6 MiB
 Load Address: 00008000
 Entry Point: 00008000
 Verifying Checksum ... OK
## Flattened Device Tree blob at 02a00000
 Booting using the fdt blob at 0x02a00000
 Loading Kernel Image ... OK
OK
 Loading Device Tree to 1fb4a000, end 1fb4f6ff ... OK

Starting kernel ...
```

Read-only ArchL inux Filesystem

In order to make the NFS Arch Linux filesystem read-only to the RCEs, execute this on the NFS host:

```
chmod -R a-w <arch_linux_root>
```

Note, command this may need to be performed with sudo privileges.

Once an RCE writes to the filesystem, the root user has ownership of any new files.

Adding Packages to the Arch Linux NFS Filesystem

In order to add packages using pacman, the NFS Arch Linux filesystem must have global read-write permissions.

On the NFS host, execute:

```
chmod -R a+w <arch_linux_root>
```

New packages are added using an RCE that is already running as a diskless node.

Package files must reside in a directory of <arch_linux_root> where RCEs have visibility.

Log into an RCE, and install the package(s) as follows:

```
pacman -U package_file.xz
```

Be sure to restore the filesystem to read-only permissions after package maintenance is complete.