NETWORK BOOTING.

Flashless, Network-Based Booting of MicroTCA AMCs

Patrick Huesmann
Software Engineer, DESY MSK
14th MicroTCA Workshop for Industry and Research

2025-12-03

HELMHOLTZ

DESY.

Network Booting: Motivation

Why can network booting be useful?

Flashless Operation

- No degradation / bit corruption
  (SW/FW storage is not subject to radiation)
- Stateless operation
  (Device always boots up in the same state)

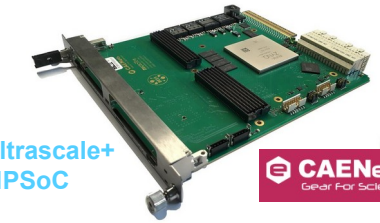
Easier deployment & fleet maintenance

- Each board only needs a bootloader, the application SW/FW doesn't have to be programmed in advance
- SW/FW updates can be conducted by changing just one single file on a central server

Network Booting: Status

- Implemented and lab-tested on our SoC-based AMCs (Zynq-7000, Zynq Ultrascale+)
- Not yet rolled out in our facilities
  → No "real-world" experiences yet

DAMC-FMC2ZUP (Supercarrier)



DAMC-FMC1Z7IO

DAMC-DS812ZUP

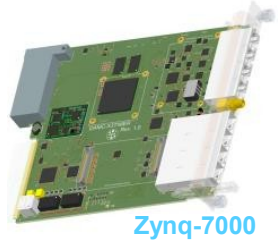
DAMC-MOTCTRL

Ultrascale+
MPSoC

DAMC-UNIZUP

DAMC-DS5014DR
(RFSoC-based)

DAMC-X3Timer



2020 2021 2022 2023 2024 2025

„Flavors“ of network booting

- U-Boot supports NFS and TFTP to load data over network.
- Once the kernel and device tree blob (DTB) is loaded from U-Boot,
  - the kernel can mount the rootfs on a remote server (via NFS)
  - or mount a RAM disk image (loaded via NFS or TFTP from U-Boot).

Root filesystem on NFS

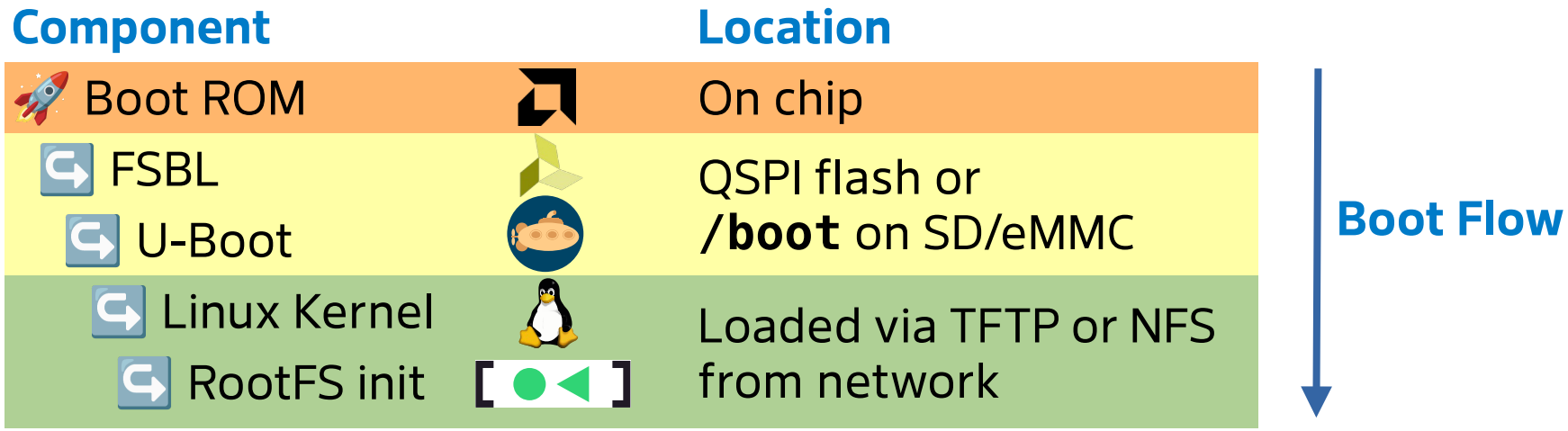
❌ Requires a dedicated rootfs directory for each board (if mounted R/W)
❌ Stateful / persistent (if mounted R/W)
❌ Provisioning has to be split up (kernel/DTB and rootfs)
❌ Runtime lag for every file access
❌ Additional burden on the network at runtime

Root filesystem on RAM disk

✅ All boards in a fleet can boot from a single file
✅ Stateless / non-persistent
→ board always boots up in the same state
✅ All parts of the provisioning (kernel, DTB, rootfs, FPGA…) can be bundled in a single file
❌ One-time startup lag from remote-loading the image
✅ Instant file access, because rootfs lives in RAM
❌ Memory penalty due to rootfs size

→ For MicroTCA AMC boards, the „rootfs in RAM“ approach looks appropriate

Stages of Zynq/ZynqUP network boot sequence



FPGA bitstream can be loaded by U-Boot (before Linux starts) or by fpgautil (from the running Linux system).

- FSBL & U-Boot are the only parts of the application that need to live in flash memory
  - They only exist to load the later stages in the boot process
  - After handoff to Linux Kernel they're not running anymore
- So we can call it "flashless" operation, even though some components live in flash

MicroTCA specific patches & adjustments

Backplane ethernet requires some tweaks due to differences with „regular ethernet"

PICMG AMC.2 R1.0 "Ethernet Advanced Mezzanine Card Specification"

- 1000BASE-BX (SerDES Gigabit) Ethernet link
- "One gigabit-per-second Ethernet data link over two pairs of copper: one pair for transmit and one for receive."
- From the PHY point of view, it is the same as talking to a SFP module.

Device tree based PHY configuration

- Use **`marvell,reg-init`** property to initialize PHY correctly for backplane ethernet.
- Advertise only 1000BASE-X Full-Duplex.
- Make PHY translate from RGMII (SoC side) to 1000BX (AMC backplane side).

```
phy0: phy@0 {
    // Marvell 88E1512
    marvell,reg-init =
        // Set Fiber Auto-Negotiation Advertisement Register
        // to 1000BASE-X Full-Duplex *only*
        < 1  4 0x0000 0x0020>,
        // Set PHY Mode to "RGMII --> 1000BASE-X"
        <18 20 0x0000 0x0002>,
        // Reset PHY
        <18 20 0x7fff 0x8000>;
};
```

DT-based config for backplane ethernet

MicroTCA specific patches & adjustments

U-Boot patches required to bring the Backplane Ethernet link up

Backport `marvell-reg-init`

- Support **custom PHY init** from device tree (see slide above).
- Only required for very old U-Boot (as in Petalinux 2020.2). Recent U-Boot can do this out of the box.

Patch: `https://gitlab.desy.de/techlab/software/yocto-layers/meta-techlab-bsp/-/blob/rel-v2020.2/recipes-bsp/u-boot/files/eth-backplane/marvell-reg-init.patch`

Use Fiber page of PHY

- Marvell PHY has different **pages** depending on selected **network standard**.
- Switch to the **Fiber page** to read back the correct values for autonegotiation and link status.

Patch: `https://gitlab.desy.de/fpgafw/yocto/meta-desy-bsp/-/blob/rel-v2023.2/recipes-bsp/u-boot/files/eth-backplane/Use-PHY-Fiber-interface.patch`

Add support for PHY reset GPIO

- Use `phy-reset-gpios` from device tree to take the **PHY out of reset**.

Patch: `https://gitlab.desy.de/fpgafw/yocto/meta-desy-bsp/-/blob/rel-v2023.2/recipes-bsp/u-boot/files/eth-backplane/zynq_gem-add-support-for-phy-reset-gpios.patch`

NFS-related issues & workarounds

(Only relevant if U-Boot NFS client is used)

U-Boot only supports UDP, which has been disabled in recent kernels

- Linux kernel disables NFS over UDP via **`CONFIG_NFS_DISABLE_UDP_SUPPORT`** since 5.6.
- Can be worked around by either compiling a custom kernel or setting up a **userspace NFS server**.
- Commonly available package **`nfs-ganesha`** is a userspace server with UDP support.

```
EXPORT
{
  # Export Id (mandatory)
  Export_Id = 1;
  # Exported path (mandatory)
  Path = /nfsroot;
  # Use NFSv3
  Protocols = 3;
  # Force UDP support
  Transports = UDP;
  ...
}
```

Example nfs-ganesha config

Enable or fix NFSv3 support in U-Boot

- `nfs-ganesha` only supports **NFSv3** or higher; U-Boot requires patches for it to work (depending on version)
- older U-Boot (PLNX 2023.2): Backport NFSv3 support
  Patch: `https://gitlab.desy.de/fpgafw/yocto/meta-desy-bsp/-/blob/rel-v2023.2/recipes-bsp/u-boot/files/backport-NFSv3-support-from-upstream.patch`
- more recent U-Boot (PLNX 2024.2): Fix NFSv3 file handle
  Patch: `https://gitlab.desy.de/fpgafw/yocto/meta-desy-bsp/-/blob/rel-v2024.2/recipes-bsp/u-boot/files/net-nfs-fix-file-handle-length-in-NFSv3.patch`

The fitImage format

Universal image format to bundle components of a system into a single file

- Supported out of the box by U-Boot and Yocto buildsystem
- Overview: `https://www.thegoodpenguin.co.uk/blog/u-boot-fit-image-overview`
- Specification: `https://fitspec.osfw.foundation`

Building a fitImage with generic Yocto Linux

Add following to local.conf:

```
KERNEL_CLASSES=kernel-fitimage
KERNEL_IMAGETYPES=fitImage
INITRAMFS_IMAGE=<yocto-image-name>
```

Building a fitImage with fwklinux

Invoke build with:

```
make cfg=yocto yocto_fitimage
```

Loading/booting a fitImage in U-Boot

```
# load via NFS
nfs 0x60000000 \
  192.168.1.138:/nfsroot/fitimage

# ...or TFTP
tftpboot 0x60000000 \
  192.168.1.138:fitimage

# boot the fitImage
bootm 0x60000000
```

The server IP address can also be provided by DHCP - see RFC2132 (BOOTP).

```
$ mkimage -l fitImage-fwklinux-image-minimal-damc-unizup-damc-unizup

FIT description: Kernel fitImage for FwkLinux/6.1.60-xilinx-v2023.2+gitAUTOINC+9c55be2d8a/damc-unizup
Created:         Tue Oct 31 04:35:22 2023
 Image 0 (kernel-1)
  Description:  Linux kernel
  Created:      Tue Oct 31 04:35:22 2023
  Type:         Kernel Image
  Compression:  gzip compressed
  Data Size:    9943482 Bytes = 9710.43 KiB = 9.48 MiB
  Architecture: AArch64
  OS:           Linux
  Load Address: 0x00200000
  Entry Point:  0x00200000
  Hash algo:    sha256
  Hash value:   cca9ef2ad27dfd9c0c4b12c3c04cc473b732856ae1e72
 Image 1 (fdt-system-top.dtb)
 …
  Type:         Flat Device Tree
  Hash algo:    sha256
  Hash value:   72db57bf6d8143dcb7f50cc1ef6418866304b5c97a612
 Image 2 (ramdisk-1)
 …
  Type:         RAMDisk Image
  Hash algo:    sha256
  Hash value:   ee381997bf7e38390026146eada79c167458aeec61dc0
 Default Configuration: 'conf-system-top.dtb'
 Configuration 0 (conf-system-top.dtb)
  Description:  1 Linux kernel, FDT blob, ramdisk
  Kernel:       kernel-1
  Init Ramdisk: ramdisk-1
  FDT:          fdt-system-top.dtb
  Hash algo:    sha256
  Hash value:   unavailable
```

Image Type

Load and Entry Addresses

Image Checksum

Metadata in a example fitImage

Network Booting: Outlook

Generic setup for a whole fleet of boards

- With **MMC mailbox**, U-Boot already has information about the board setup
  - e.g.: AMC slot, FMC mounted, RTM mounted
- U-Boot can send a specific request to the server depending on those parameters
  - such as: download fitImage **for a DAMC-UNIZUP** in **slot 2** with a **DRTM-SFP4+** mounted
- Then the board can load different applications depending on those parameters and doesn't even have to know it in advance.
- A fleet of boards can then have a common, generic U-Boot setup and all specific data can be setup on the server side!

```
root@ZUP-0634:~# mmcinfo mmc
MMC information
---------------
App version     : 2.10
Lib version     : 2.25
CPLD board ver. : 1.12
CPLD lib ver.   : 1.12
STAMP revision  : Rev. D
AMC slot        : 6
IPMB addr       : 0x7c
Board name      : DAMC-FMC2ZUP
IANA Vendor ID  : 0x053f
IANA Product ID : 0x200b
Uptime          : 11 minutes, 32 seconds
```

Thank you!

Contact

Patrick Huesmann (Firmware)
<patrick.huesmann@desy.de>

Michael Fenner (Digital Hardware)
<michael.fenner@desy.de>

Deutsches Elektronen-Synchrotron DESY

MSK Department
Notkestraße 85
22607 Hamburg
www.desy.de