



2IMC05 Prepration Graduation Project

JUNE 20,2019

KEYSHAV SURESH MOR(1237978)



Master Thesis : Integration of Embedded Systems in the Network, System Administration and Application Frameworks of the CMS DAQ at CERN

Supervisors : Dr. Majid Najafabadi(TU/e) and Dr. Frans Meijers(CMS,CERN)

**Acknowledgements: Dr. Petr Zejdl(CMS,CERN), Dr. Marc Dobson(CMS,CERN),
Awais bin Zahid(COMSATS University, Pakistan)**

Overview

- The main accelerator of CERN, the LHC will soon be upgraded to the High Luminosity Phase
- This will increase the number of particle collisions occurring at the centre of particle detectors like CMS
- To get maximum useful data out of this increase in collisions, the CMS detector and its sub-detectors would need to be upgraded
- This upgrade would also result in upgrade of front-end electronics, the data transport links and the back-end electronics
- This phase would also see introduction of SoC FPGA into the network
- These devices, with an ARM core, would run a Linux distribution and their numbers would run in thousands
- Managing such devices running Linux, speaking over the network is a challenge for SysAdmins, CERN IT and computer security teams
- Thus, it is necessary to devise a strategy of building, booting and updating Linux on these devices once they are installed in the network

Contents

- Introduction to CMS
- Introduction to Embedded Systems for CMS DAQ
- Introduction to Linux for Embedded Systems
- Challenges and Research Goals
- Project Status and Workplan

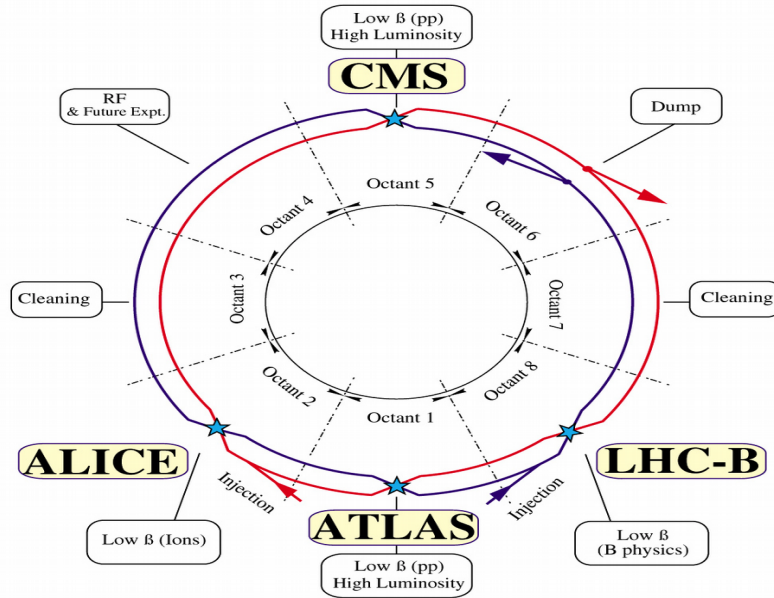
About CERN

- CERN(European Organisation for Nuclear Research) is one of the biggest particle physics laboratory in the world. Established in 1954.
- The research at CERN focuses on particle physics and answering questions on the fundamentals of physical matter.
- The LHC(Large Hadron Collider) : Main particle accelerator. Started operating in 2009.
- Consists of multiple sub-experiments like ALICE,LHCb,CMS and ATLAS.
- Detectors of experiments located along the LHC. Last run(Run 2) from 2015-2018.
- Every run followed by long shutdown(LS) of 2-3 years for maintenance and upgrade.
- Run 3 scheduled from 2020 till the end of 2022
- The next LHC upgrade(High Luminosity LHC) scheduled for 2024.



Experiments around the LHC

LHC LAYOUT



CERN AC _ EI2-4A _ V18/9/1997

Image Source : CERN

The Compact Muon Solenoid Experiment(CMS)



- One of the main experiments at CERN
- Controlled particle collisions occur at the centre of its detector
- Trackers for tracking trajectory and measuring momentum of particles
- Calorimeters for measuring energy of the particles
- Customised front-end electronics for sensor technology
- 10 Gigabit front-end links
- Customised back-end electronics installed in MicroTCA crates



Transverse Section of the CMS Detector

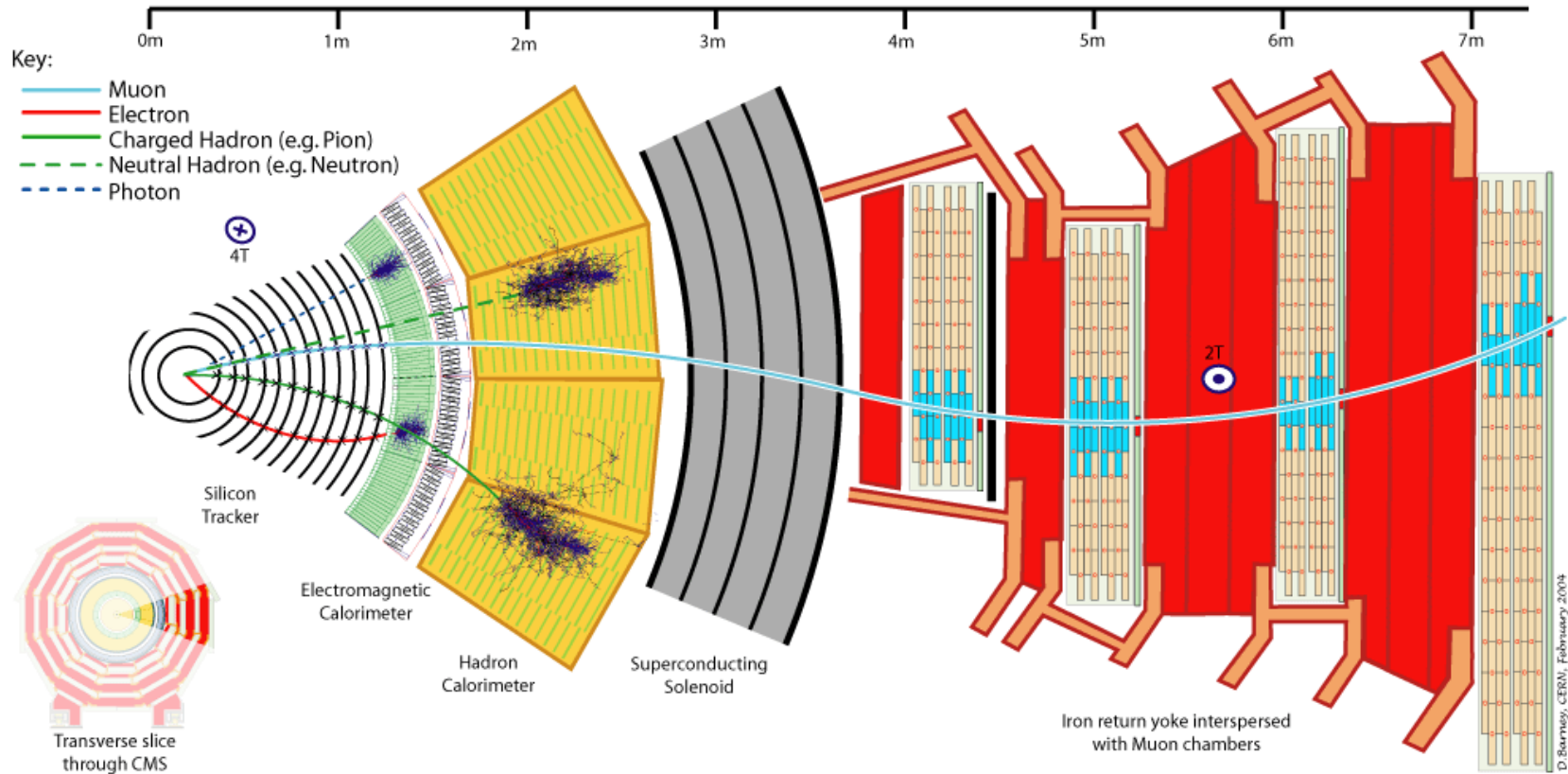


Image Source : CERN

Cross-section view of the CMS Detector

CMS DETECTOR

Total weight : 14,000 tonnes
Overall diameter : 15.0 m
Overall length : 28.7 m
Magnetic field : 3.8 T

STEEL RETURN YOKE
12,500 tonnes

SILICON TRACKERS
Pixel ($100 \times 150 \mu\text{m}$) $\sim 16\text{m}^2 \sim 66\text{M}$ channels
Microstrips ($80 \times 180 \mu\text{m}$) $\sim 200\text{m}^2 \sim 9.6\text{M}$ channels

SUPERCONDUCTING SOLENOID
Niobium titanium coil carrying $\sim 18,000\text{A}$

MUON CHAMBERS
Barrel: 250 Drift Tube, 480 Resistive Plate Chambers
Endcaps: 468 Cathode Strip, 432 Resistive Plate Chambers

PRESHOWER
Silicon strips $\sim 16\text{m}^2 \sim 137,000$ channels

FORWARD CALORIMETER
Steel + Quartz fibres $\sim 2,000$ Channels

CRYSTAL
ELECTROMAGNETIC
CALORIMETER (ECAL)
 $\sim 76,000$ scintillating PbWO_4 crystals

HADRON CALORIMETER (HCAL)
Brass + Plastic scintillator $\sim 7,000$ channels

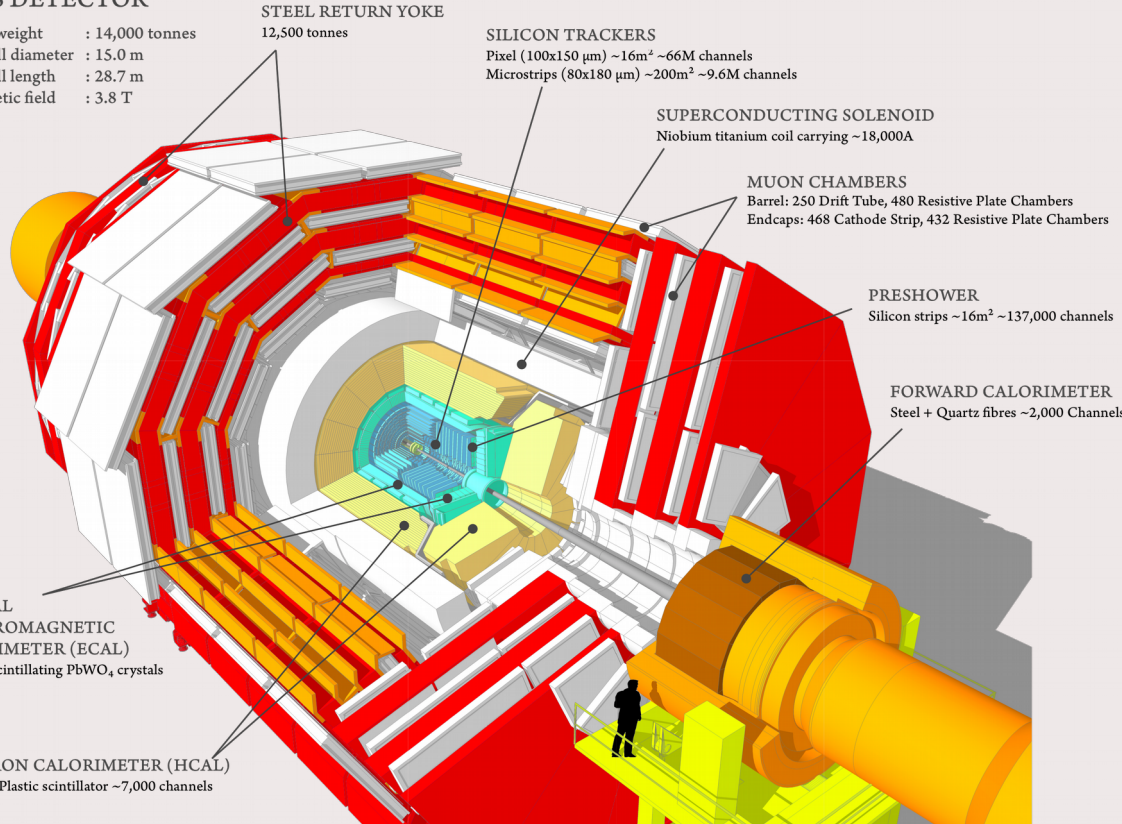
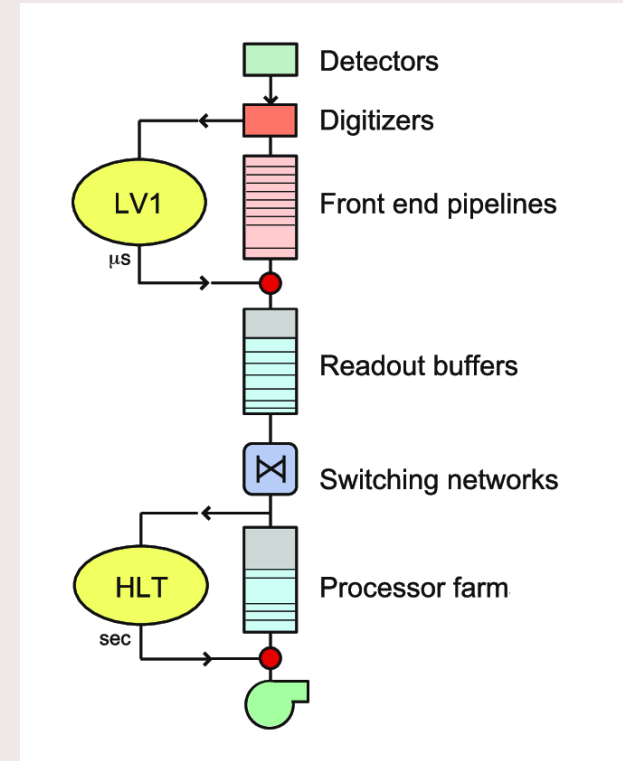


Image Source : CERN

The CMS Trigger and Data Acquisition System(TDAQ)

- Particle collisions occur at the frequency of 40 MHz. Front-end detectors collect data at same frequency.
- Constraints in data transfer to the surface and in data storage
- Not all collisions and their sub-events are of interest for physics research.
- Hence, CMS data acquisition mechanism divided in 2 parts :
- **1. Trigger** : Does selection of interesting data and brings down data sampling rate or event-rate
- **2. Data Acquisition System** : Acquires event-data selected by the trigger and sends it upstream for analysis
- Trigger divided in 2 levels :
- **1. Level-1 Trigger** : Hardware trigger based on custom electronics
Selects interesting event data. Brings event rate down to 100 kHz
- **2. High-level Trigger(HLT)** : Software based trigger running on a processor farm. Further selection. Brings down event rate to 1kHz. Stores data.



The CMS Trigger and Data Acquisition System(TDAQ)

- Data Acquisition System(DAQ) sits between L1 and High-level Trigger
- DAQ is not a monolithic structure but a modular one
- Data acquired only after L1 trigger selection
- The DAQ read-out units receives data fragment over 500, 10 Gigabit links.
- 56 Gigabit optical links used to send these fragments to event-building processors(Builder Units)
- Builder units build an event by aggregating fragments of data related to an interesting event
- Builders units send these events to HLT for further classification

The CMS Trigger and Data Acquisition System(TDAQ) IN PHASE -2

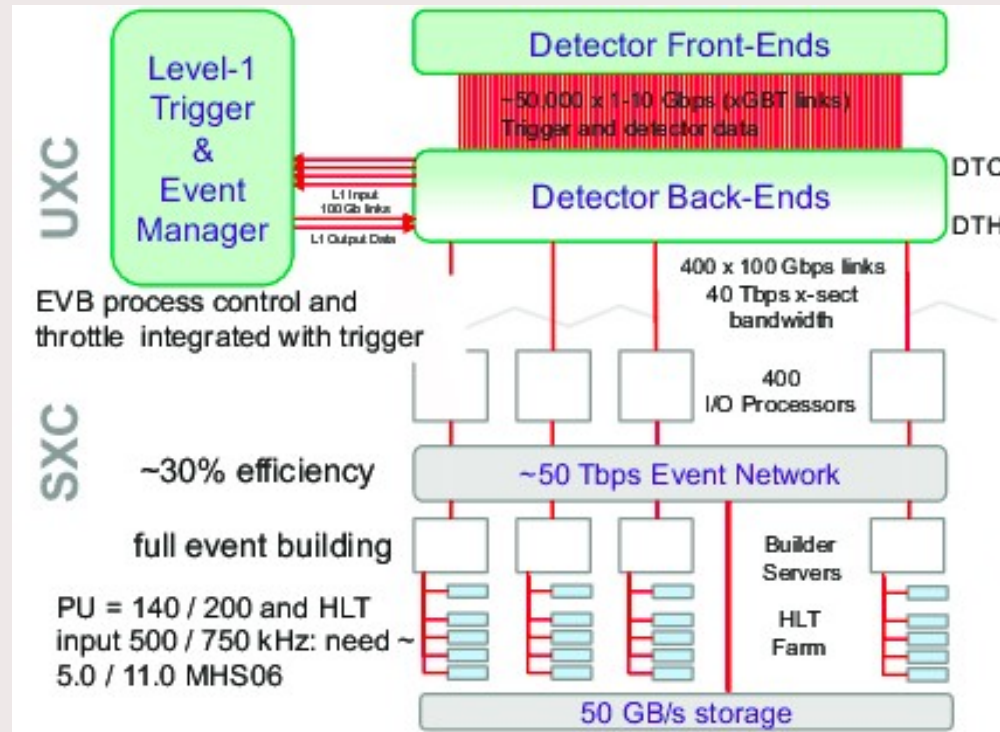


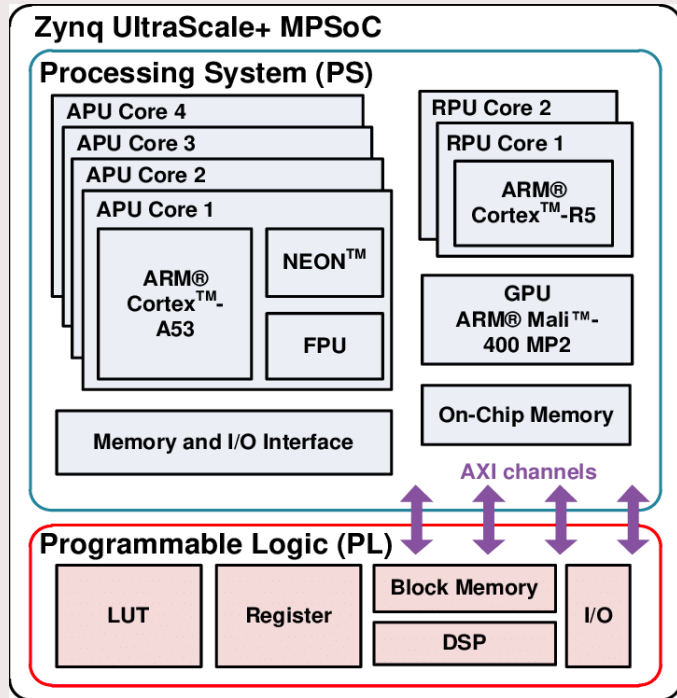
Image Source : ResearchGate

The CMS DAQ Phase-2 Upgrade

- The High-Luminosity LHC phase in 2024 would result in high rate of particle collision
- Thus, increased rate of interesting event occurrence. Trigger rate would go from 100 kHz to 750 kHz
- Upgrade of front-end electronics. Around 50,000 front-end 5-10 Gigabit links
- Upgrade of back-end read-out electronics
- Around 700 optical links of 100 Gigabit/second per sub-detector
- Xilinx Zynq FPGA SoC in the read-out electronics
- Migration from microTCA to ATCA crates for read-out electronics



The Xilinx Zynq System-on-Chip



- **System-on-Chip(SoC)** : tightly integrated programmable logic (PL) with processing system (PS)
- CPU and FPGA on one chip. Customised hardware design with powerful processing. Saves space.
- Previously, two separate systems communicating with each other.
- ARM core CPU with its own memory and I/O interfaces
- Older Zynq families come with low-speed, single/dual core processors
- Newer Zynq families come with faster, multi-core processors and multiple processing units
- Modern Zynq capable of running a desktop OS with GUI

Xilinx Zynq-7000 and Zynq Ultrascale +

Zynq-7000 SoC features

- Single/Dual ARM Cortex A-9
- 1 Ghz
- **32-bit** processor



Image Source : Xilinx

Zynq Ultrascale+ SoC features

- Dual/Quad Arm Cortex-A53
- Upto 1.5 Ghz
- **64-bit** processor
- L1 Cache 32KB
- L2 Cache 1MB
- Dual ARM Cortex-R5 Real-time Unit
- ARM Mali-400 MP2 GPU

Linux for DAQ Embedded Systems

- Embedded systems capable of running tasks through bare-metal applications and real-time operating systems(RTOS)
- Such devices allow little or no possibility for accessing and configuring the embedded systems without removing them from the setup.
- Devices like Zynq-7000 and Zynq Ultrascale+ powerful enough to run a full Linux distribution
- Scientists, engineers, system administrators familiar with Linux operating systems
- Linux supports networking and security features required at CMS and CERN
- Linux distributions available for Zynq ARM architectures
- Linux distributions need to be ported to CMS DAQ embedded systems to support specific hardware requirements

A Linux based Operating System

An operating system based on the Linux kernel has the following important layers :

- **Userspace/Application Layer** : Accessible to the user, contains applications, software package and other utilities
- **Root Filesystem** : It contains all important libraries, binaries, applications, system logs etc. that are required by the user or the OS
- **The Linux kernel** :
 - ➔ Manages the memory and time used by processes,
 - ➔ Manages inter-process communication,
 - ➔ Ensures integrity of the system in multi-user environment
 - ➔ Controls access to the hardware through driver

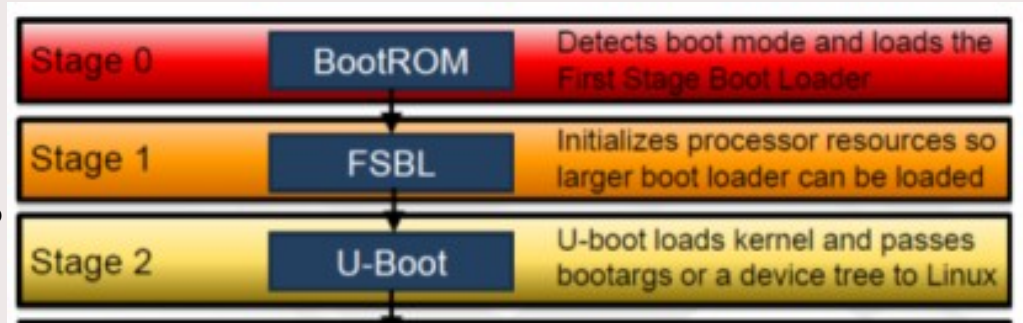
Components of a Linux Boot

The booting of a Linux based operating system on any hardware depends on the following components :

- **Bootloader** : First software to execute when the system is booted-on. Initializes the memory and some hardware to load the kernel
- **Device Tree** : Describes underlying hardware, peripheral registers and memory locations to the kernel.(like BIOS)
- **Init** : Executes with PID(1) and executes a few initial services including user-login service to allow user to log in. **Systemd** and **init** are commonly used as init

The Bootloader

- The booting process can have multiple stages of bootloaders
- Multi stage bootloaders help users in having more control over boot process
- **FSBL(First Stage Bootloader)** used to **initialise some hardware and memory** before loading a **SSBL(Second Stage Boot Loader)** in some embedded devices
- Helps in networking aspects such as booting and configuring FPGA over the network
- **U-Boot (Das U-Boot)** is a SSBL for Xilinx Zynq devices, similar to Grub for x86 processors
 - Open-Source
 - Supports ARM architectures
 - Dynamically configurable
 - Supports networking features like TFTP
 - Has its own command terminal

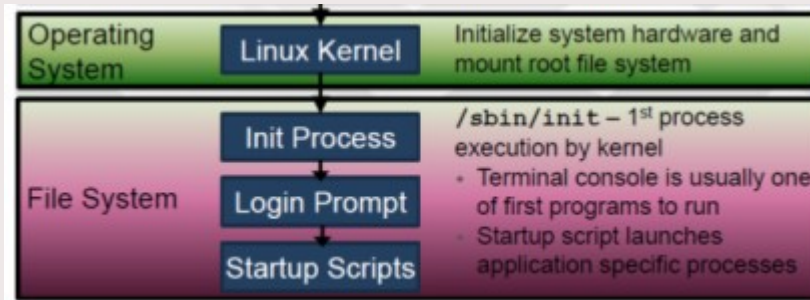


The Device Tree

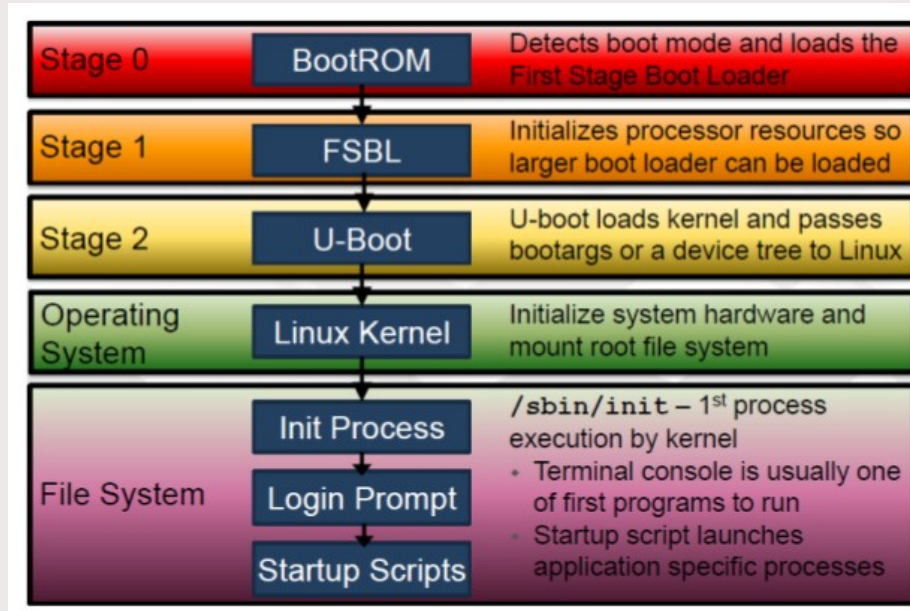
- While booting, the kernel does not have any information about underlying hardware
- In commercial products like standalone PCs, hardware is configured by BIOS and kernel reads its configuration from BIOS directly
- Zynq devices don't have BIOS
- Zynq I/O and bus interconnect are defined by board developers
- In absence of such a BIOS : a **different kernel configuration** for every new board using a Zynq SoC would be needed
- To ensure that a single kernel is used across boards with Zynq family, a different way of passing hardware configuration is needed : **a device tree**
- It describes the hardware design, the hardware configuration, bus connections, memory interfaces etc.
- The device tree is loaded by U-Boot and passed to the kernel when kernel starts execution

The Root Filesystem

- Functionality of root filesystem remains the same in general purpose and embedded Linux
- Device drivers, libraries, software packages may vary as per hardware and Linux distribution
- The root file system of an embedded device running Linux may be lighter in terms of memory compared to root file system on a desktop PC
- **NFS** is a root file system type to be used during the project
- **NFS** (Network File System) is a file system type used to mount the root file system over the network.
- The root filesystem is located on the server and is accessible to the hardware just like a local filesystem

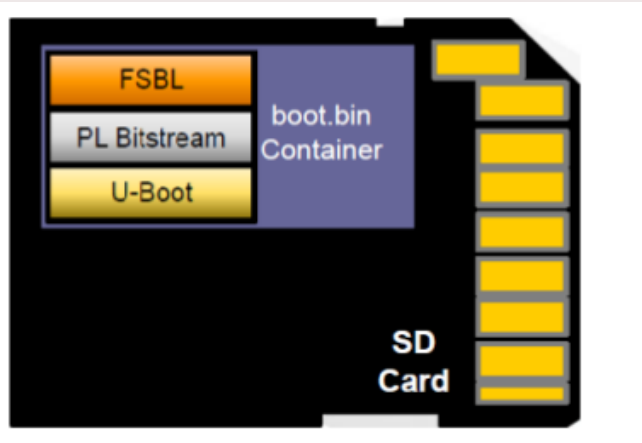


Booting the Linux on Zynq



Booting the Linux on Zynq

- For the project, we use 2 ways of booting the Zynq :



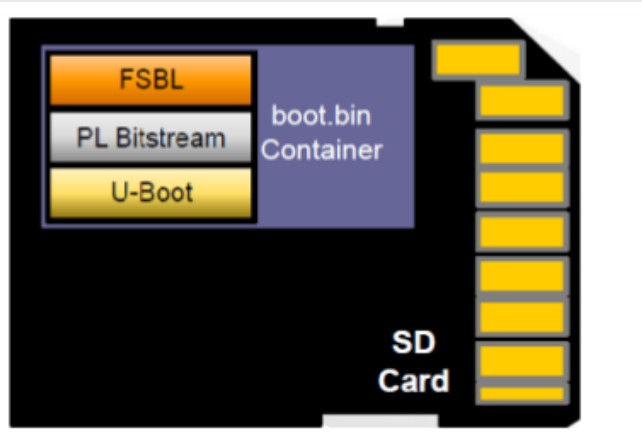
1. SD Card boot

2. TFTP boot

- Since the Zynq would be integrated in the CMS DAQ network, we choose a modular boot process :
 - 1. SD Boot till the U-Boot stage
 - 2. TFTP boot to load the kernel and the device tree
 - U-Boot loads over the network using TFTP
 - 3. Kernel mounts the root filesystem over NFS

Contents of Boot Files

- Contents of the bootable SD card are :



1. FSBL

2. Optional Bitstream

3. U-Boot

- **Image.ub** file contains the kernel and the device tree
- **Image.ub** is an image file readable by U-Boot
- Can be loaded through SD Card or TFTP

Linux Distributions



- A Linux Distribution is an OS with a customised Linux kernel, a collection of software packages and a package manager.
- Distributions can be designed by keeping a user-community in focus. (For ex. Kali Linux for security professionals)
- Distributions may be hardware specific (For ex. Raspbian)
- Distributions may have their own package manager (For ex. apt for Debian/Ubuntu, yum for CentOS/Fedora)
- Distribution may have rolling updates(Fedora/Archlinux) or may provide Long term support(LTS)
- Long term Support is interesting for system administrators



Linux Distributions for the Project

- In this project, we focus on CentOS Linux distribution and Xilinx Linux kernel
 - **Xilinx Linux**
 - Linux kernel adapted and maintained by Xilinx for the Zynq SoC
 - Contains Zynq specific modules
 - Used by Yocto Project and PetaLinux Tools to build distribution
 - Quarterly or half-yearly updates
 - **CentOS**
 - Free distribution based on RedHat Enterprise Linux commercial release
 - Regular, tested security patches and updates from its developers
 - Wide usage at CERN, familiar for System administrators, IT and cyber security teams
 - Long-Term Support



The Yocto Project

- Yocto Project is not a distribution , but a framework to build distributions
- Helps build custom distributions for custom hardware
- Allows user full freedom to customise their distribution (choice of kernel, u-boot, root file system)
- It uses different layers called “meta” layers to specify configuration of the distribution
- Many semi-conductor companies develop special meta-layers for Yocto Project
- Allows cross-compilation of software for the hardware in use



PetaLinux Tools

- Petalinux Tools are developed by Xilinx. The tool chain is based on the Yocto Project
- Download of Petalinux downloads the whole Yocto project meta-layers, with Xilinx BSP, U-Boot, Kernel etc.
- Does the same thing as Yocto but ...
 1. Easier to use, abstracts complex Yocto processes
 2. Good for beginners to understand distribution building
 3. Easy command line flow
 4. Less flexible than Yocto
 5. Challenging to integrate customised root filesystem, U-Boot, Kernel

Story till now...

- We had a look at what does a Linux operating system mean
- We saw what is a Linux distribution and how to build it
- We saw the important components of a Linux boot process
- We saw the Linux boot process on Zynq

And the story from here...

- We would have a look the challenges related to embedded Linux at CMS DAQ
- The research goals of this project
- The work schedule of the project

Challenges

- Thousands of Xilinx Zynq devices to be installed in CMS DAQ for the Phase-2 upgrade
- Most of these devices would be running some Linux distribution
- The devices would have special hardware design, peripheral interface and special software needs
- This spells the following challenges for the Sys Admins, IT and computer security teams :
 - 1. Problem of Plenty** : Sys Admins have to provide long terms support to diverse distributions, diverse kernels, diverse root file systems for diverse hardware
 - 2. Network Integration** : Sys Admins need to worry about network management, IP address allocation, NFS and TFTP server configuration (Global or server cluster) and bandwidth requirements, especially if all devices boot at the same time.
 - 3. Security** : Sys Admins need to provide security patches and software updates for diverse distributions on a regular basis and need knowledge of these distributions

Research Goals

- **Develop a common Linux distribution such that it at least meets minimum requirements of maximum groups**
- **Define a common boot process shared by all SoC devices on the network and find a way to load board-specific details like kernel, device-tree and bitstream over the network**
- **Define a flexible and scalable common network file system, that has an additional layer of a file system containing board specific software packages and read-writable root directories**
- **Investigate an efficient method of software package and system update that ensures board-specific updates without affecting other boards**
- **Test the CentOS ARM 64-bit kernel on Xilinx Zynq Ultrascale+ and investigate how to port Xilinx specific features in the kernel**

Work done till now...

- Building a distribution for Zynq Zedboard (Zynq 7000) using Xilinx Linux and PetaLinux Tools
- Booting the Zedboard over the network using TFTP
- Mounting root file system over the network using NFS
- Prepared and presented PetaLinux tool chain tutorial and live demo with Zedboard at CMS SoC workshop on June 13

Lessons learnt...

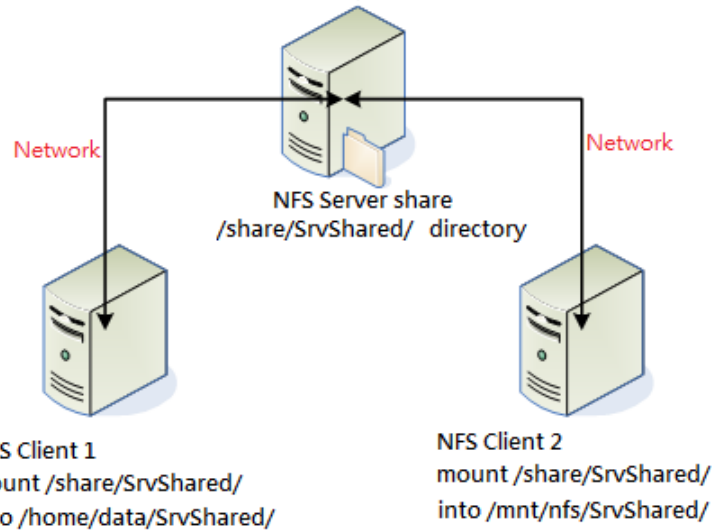
- U-Boot fails to get TFTP server IP automatically from DHCP (U-Boot code needs a fix)
- Booting Zynq Ultrascale+ is significantly complicated compared to Zynq 7000
- PetaLinux is not easy to configure using external kernel, U-Boot and root-file system

Work Plan

- Full time work at CERN CMS from July 1,2019 till January 17,2020 (24 Weeks)
- The following tasks need to be completed over the course of 24 weeks :
 - Build distributions based on Xilinx Linux kernel using Yocto and PetaLinux for Zynq Ultrascale+ and test booting over network. Test root file system mount using external root file systems (CentOS)
 - Fix problems with TFTP server configuration in U-Boot
 - Work on porting Xilinx specific features to CentOS kernel
 - Build distributions based on CentOS kernel using Yocto and PetaLinux for Zynq Ultrascale+ and test booting over network
 - Test loading bitstream, software package and updates over the network
 - Test strategies of regular system updates for boards over the network
 - Investigate flexible, scalable, board-specific root file systems. Test network configuration managers for dynamic configuration of the boards over the network
 - Coordinate with system administrators to integrate the Zynq Ultrascale+ into CMS network and run tests

Backup Slides

Mounting the Root Filesystem over NFS

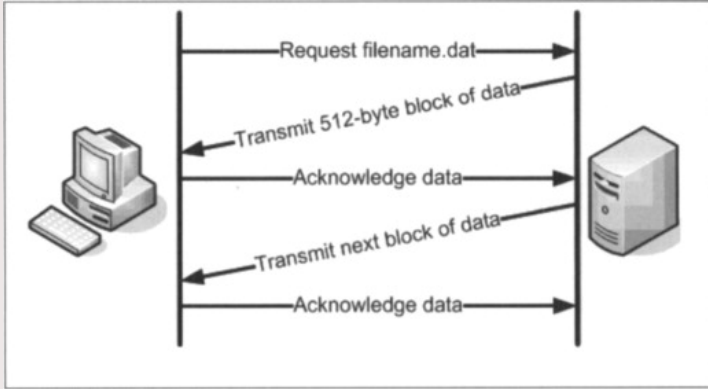


- NFS stands for Network File System
- Has multiple version. Latest version NFSv4 is specified by the RFC-3530 standard
- Can use TCP or UDP, depending upon the version.
- NFSv3 has been used till now in this project.
- Default block size of 32 KB. Can be modified during the mount request using **-rsize** and **-wsize**
- Used to mount, view, edit (based on permissions) remote file systems over network

Mounting Root File system over NFS

```
macb e000b000.ethernet eth0: link up (1000/Full)
• IPv6: ADDRCONF(NETDEV_CHANGE): eth0: link becomes ready
Sending DHCP requests ., OK
IP-Config: Got DHCP answer from 192.168.100.1, my address is 192.168.100.10
IP-Config: Complete:
    device=eth0, hwaddr=08:00:30:f4:03:35, ipaddr=192.168.100.10, mask=255.255.255.0, gw=192.168.100.1
    host=192.168.100.10, domain=, nis-domain=(none)
    bootserver=192.168.100.1, rootserver=192.168.100.1, rootpath=/home/kmor/tftpboot,tcp,v3      names1
ALSA device list:
  No soundcards found.
VFS: Mounted root (nfs filesystem) on device 0:12.
devtmpfs: mounted
Freeing unused kernel memory: 1024K
INIT: version 2.88 booting
Starting udev
udev[722]: starting version 3.2.2
udev[723]: starting eudev-3.2.2
FAT-fs (mmcblk0p1): Volume was not properly unmounted. Some data may be corrupt. Please run fsck.
Mon Jun 10 16:09:16 UTC 2019
Starting internet superserver: inetd.
```

Loading the Kernel over TFTP



- TFTP stands for Trivial File Transfer Protocol
- Defined by the RFC1350 standard
- Uses UDP as transport protocol
- Simple protocol. Also used in network booting process
- Allows clients to read and request data from the server
- Every block of data transferred from server is replied with an acknowledgement

Loading the Kernel over TFTP

```
Zynq> run netboot
Using ethernet@e000b000 device
TFTP from server 192.168.100.1; our IP address is 192.168.100.10
Filename 'image.ub'.
Load address: 0x10000000
Loading: #####
#####
#####
#####
#####
7.3 MiB/s
done
Bytes transferred = 3956632 (3c5f98 hex)
## Loading kernel from FIT Image at 10000000 ...
   Using 'conf@system-top.dtb' configuration
```

A Device Tree

- A snapshot of a device tree used to
- boot a Xilinx Linux kernel built
- with PetaLinux Tools
- This device tree specifies
 - 1. The MAC address,
 - 2. The kernel boot arguments
 - 3. Partitioning of the Quad-SPI flash memory

```
/*
 * CAUTION: This file is automatically generated by PetaLinux SDK.
 * DO NOT modify this file
 */

/ {
    chosen {
        bootargs = "console=ttyP80,115200 earlyprintk root=/dev/nfs ip=dhcp rw";
        stdout-path = "serial0:115200n8";
    };
}

&gem0 {
    local-mac-address = [00 0a 35 00 1e 53];
};

&qspi {
    #address-cells = <1>;
    #size-cells = <0>;
    flash0: flash@0 {
        compatible = "n25q512a","micron,m25p80";
        reg = <0x0>;
        #address-cells = <1>;
        #size-cells = <1>;
        spi-max-frequency = <50000000>;
        /delete-node/ partition@qspi-fsbl-uboot;
        /delete-node/ partition@qspi-linux;
        /delete-node/ partition@qspi-device-tree;
        /delete-node/ partition@qspi-rootfs;
        /delete-node/ partition@qspi-bitstream;
        partition@0x00000000 {
            label = "boot";
            reg = <0x00000000 0x00500000>;
        };
        partition@0x00500000 {
            label = "bootenv";
            reg = <0x00500000 0x00020000>;
        };
        partition@0x00520000 {
            label = "kernel";
            reg = <0x00520000 0x00a80000>;
        };
        partition@0x00fa0000 {
            label = "spare";
            reg = <0x00fa0000 0x00000000>;
        };
    };
};
```

A Root Filesystem

```
[kmor@pcepcomd64 RootFS]$ ls
bin boot dev etc home lib media mnt proc run sbin sys tmp usr var
[kmor@pcepcomd64 RootFS]$ ls -l
total 16
drwxr-xr-x. 2 kmor zh 4096 Jun 10 18:09 bin
drwxr-xr-x. 2 kmor zh 6 Jun 10 17:57 boot
drwxr-xr-x. 2 kmor zh 6 Jun 10 17:57 dev
drwxr-xr-x. 25 kmor zh 4096 Jun 10 18:09 etc
drwxr-xr-x. 3 kmor zh 18 Jun 10 18:09 home
drwxr-xr-x. 5 kmor zh 4096 Jun 10 18:09 lib
drwxr-xr-x. 2 kmor zh 6 Jun 10 17:57 media
drwxr-xr-x. 2 kmor zh 6 Jun 10 17:57 mnt
drwxr-xr-x. 2 kmor zh 6 Jun 10 17:57 proc
drwxr-xr-x. 2 kmor zh 6 Jun 10 17:57 run
drwxr-xr-x. 2 kmor zh 4096 Jun 10 18:09 sbin
drwxr-xr-x. 2 kmor zh 6 Jun 10 17:57 sys
drwxr-xr-x. 2 kmor zh 6 Jun 10 17:57 tmp
drwxr-xr-x. 10 kmor zh 107 Dec 3 2018 usr
drwxr-xr-x. 8 kmor zh 132 Dec 3 2018 var
[kmor@pcepcomd64 RootFS]$ █
```

Entering the U-Boot

```
U-Boot 2018.01 (Jun 10 2019 - 17:12:39 +0000)

Model: Zynq Zed Development Board
Board: Xilinx Zynq
Silicon: v3.1
DRAM:  ECC disabled 512 MiB
MMC:  mmc@e0100000: 0 (SD)
SF: Detected s25fl256s_64k with page size 256 Bytes, erase size 64 KiB, total 32 MiB
In:    serial@e0001000
Out:   serial@e0001000
Err:   serial@e0001000
Model: Zynq Zed Development Board
Board: Xilinx Zynq
Silicon: v3.1
Net:   ZYNQ GEM: e000b000, phyaddr 0, interface rgmii-id
eth0: ethernet@e000b000
Hit any key to stop autoboot:  0
Zynq> █
```


Entering the U-Boot

```
U-Boot 2018.01 (Jun 10 2019 - 17:12:39 +0000)

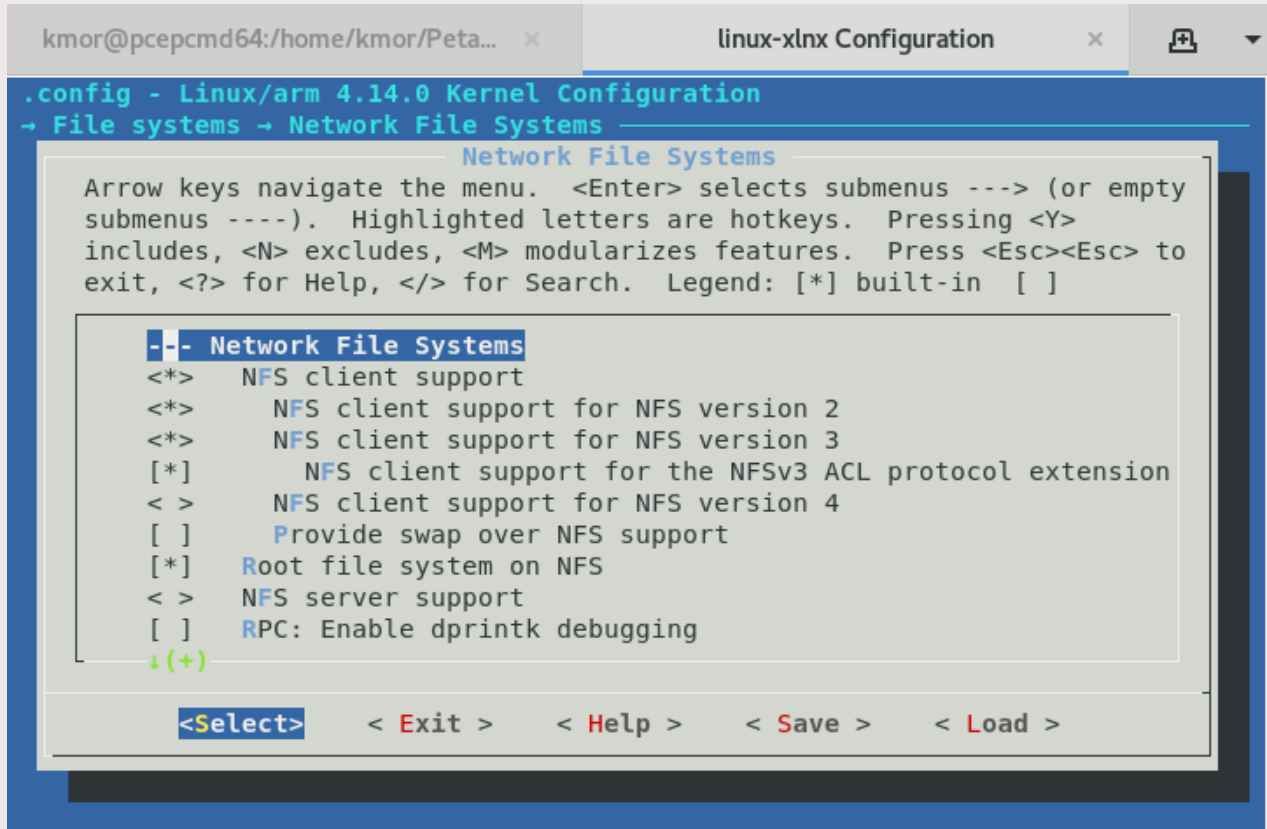
Model: Zynq Zed Development Board
Board: Xilinx Zynq
Silicon: v3.1
DRAM:  ECC disabled 512 MiB
MMC:  mmc@e0100000: 0 (SD)
SF: Detected s25fl256s_64k with page size 256 Bytes, erase size 64 KiB, total 32 MiB
In:    serial@e0001000
Out:   serial@e0001000
Err:   serial@e0001000
Model: Zynq Zed Development Board
Board: Xilinx Zynq
Silicon: v3.1
Net:   ZYNQ GEM: e000b000, phyaddr 0, interface rgmii-id
eth0: ethernet@e000b000
Hit any key to stop autoboot:  0
Zynq> █
```

Entering the U-Boot

```
Zynq> help
?      - alias for 'help'
base   - print or set address offset
bdinfo - print Board Info structure
boot   - boot default, i.e., run 'bootcmd'
bootd  - boot default, i.e., run 'bootcmd'
bootefi - Boots an EFI payload from memory
bootelf - Boot from an ELF image in memory
bootm  - boot application image from memory
bootp  - boot image via network using BOOTP/TFTP protocol
bootvx - Boot vxWorks from an ELF image
bootz  - boot Linux zImage image from memory
clk    - CLK sub-system
cmp    - memory compare
coninfo - print console devices and information
cp     - memory copy
crc32  - checksum calculation
dcache - enable or disable data cache
dfu    - Device Firmware Upgrade
dhcp   - boot image via network using DHCP/TFTP protocol
```

- The U-Boot has its own environment and terminal to execute binaries
- One can define environment variables in U-Boot which are saved in the Flash memory
- U-Boot must be configured for tftp, dhcp and other networking services before building U-Boot

Kernel Configuration



The screenshot shows a terminal window with the title bar "linux-xmlns Configuration". The terminal content displays the ".config - Linux/arm 4.14.0 Kernel Configuration" menu. The "File systems" option is selected, leading to the "Network File Systems" submenu. A detailed help message explains navigation: "Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty submenus ----). Highlighted letters are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*] built-in []". The "Network File Systems" menu is expanded, showing options for NFS client support (versions 2, 3, 4, and ACL extension), swap over NFS, root file system on NFS, NFS server support, and RPC debugging. A green arrow points to the "Select" option at the bottom of the menu.

```
.config - Linux/arm 4.14.0 Kernel Configuration
→ File systems → Network File Systems

Network File Systems

Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty
submenus ----). Highlighted letters are hotkeys. Pressing <Y>
includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to
exit, <?> for Help, </> for Search. Legend: [*] built-in [ ]

-- Network File Systems
<*> NFS client support
<*> NFS client support for NFS version 2
<*> NFS client support for NFS version 3
[*] NFS client support for the NFSv3 ACL protocol extension
< > NFS client support for NFS version 4
[ ] Provide swap over NFS support
[*] Root file system on NFS
< > NFS server support
[ ] RPC: Enable dprintk debugging
↓ (+)

<Select> < Exit > < Help > < Save > < Load >
```

Kernel Configuration

```
.config - Linux/arm 4.14.0 Kernel Configuration
→ Networking support → Networking options
    Networking options
    Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty
    submenus ----). Highlighted letters are hotkeys. Pressing <Y>
    includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to
    exit, <?> for Help, </> for Search. Legend: [*] built-in [ ]

    ↑ (-)
    [ ] Transformation statistics
    < > PF_KEY sockets
    [*] TCP/IP networking
    [*]   IP: multicasting
    [ ]   IP: advanced router
    [*]   IP: kernel level autoconfiguration
    [*]     IP: DHCP support
    [*]     IP: BOOTP support
    [*]     IP: RARP support
    <M>   IP: tunneling
    ↓ (+)

    <Select>    <Exit >    <Help >    <Save >    <Load >
```