

Implementing Linux on the Zynq™-7000 AP SoC

Open Source Linux
Development on ZedBoard™



Accelerating Your Success™

Course Objectives

- Build critical software components of an embedded Linux system
- Explore Linux software development with Xilinx SDK
- Connect SDK to hardware for execution and debug
- Understand the advantages of running Linux on Zynq

Agenda

Lecture	Description	Labs
Chapter 1	Course and Overview of Open Source Linux on ZedBoard	Pre-Lab
Chapter 1.1	Booting ZedBoard	Lab 1.1
Chapter 1.2	Introduction to U-Boot	Lab 1.2
Chapter 1.3	Zynq Boot Image	Lab 1.3
Chapter 2.1	Linux Kernel Basics	Lab 2.1
Chapter 2.2	Linux Root File System	Lab 2.2
Chapter 2.3	Options for Building Root File Systems	Lab 2.3
Chapter 3.1	Linux Device Drivers	Lab 3.1
Chapter 3.2	Device Drivers and the Device Tree	Lab 3.2
Chapter 4.1	Linux Application Development	Lab 4.1
Chapter 4.2	Linux Application Debugging	Lab 4.2
Chapter 5	What's Next, Where to Learn More	

Lab Equipment

- **Hardware**

- PC with at least 4 GB of RAM (8 GB recommended)
 - Windows 7 64-bit
 - Processor virtualization capability
- Avnet ZedBoard™ (AES-Z7EV-7Z020-G)
 - USB cable (Type A to Micro-USB Type B) – *one included in kit*
 - SD Card – *included in kit*
 - AC/DC 12V supply – *included in kit*

- **Software:**

- Xilinx ISE WebPack™ 14.2 - *included in ZedBoard kit*
 - Check Xilinx AR51895 for required installation work-arounds
- Cypress CY7C64225 USB-to-UART Bridge Driver
 - See the [Cypress USB-to-UART Setup Guide](#) on [www.zedboard.org](#)
- Terminal emulator such as Tera Term
- PDF Viewer such as Adobe Reader
- VMware Player 5.0 or later

Xilinx Embedded Tool Flow

- **ISE WebPACK Design Suite**
 - **FREE!**
 - Supports Zynq 7010, 7020 and 7030 devices
- **Included in ISE WebPACK for Zynq**
 - PlanAhead
 - Central Design Cockpit
 - HDL Development
 - Design Implementation
 - Xilinx Platform Studio (XPS)
 - Processor Configuration and Customization
 - Software Development Kit (SDK)
 - C/C++ Integrated Development Environment
- **ZedBoard Kit Includes a ChipScope™ Pro License**



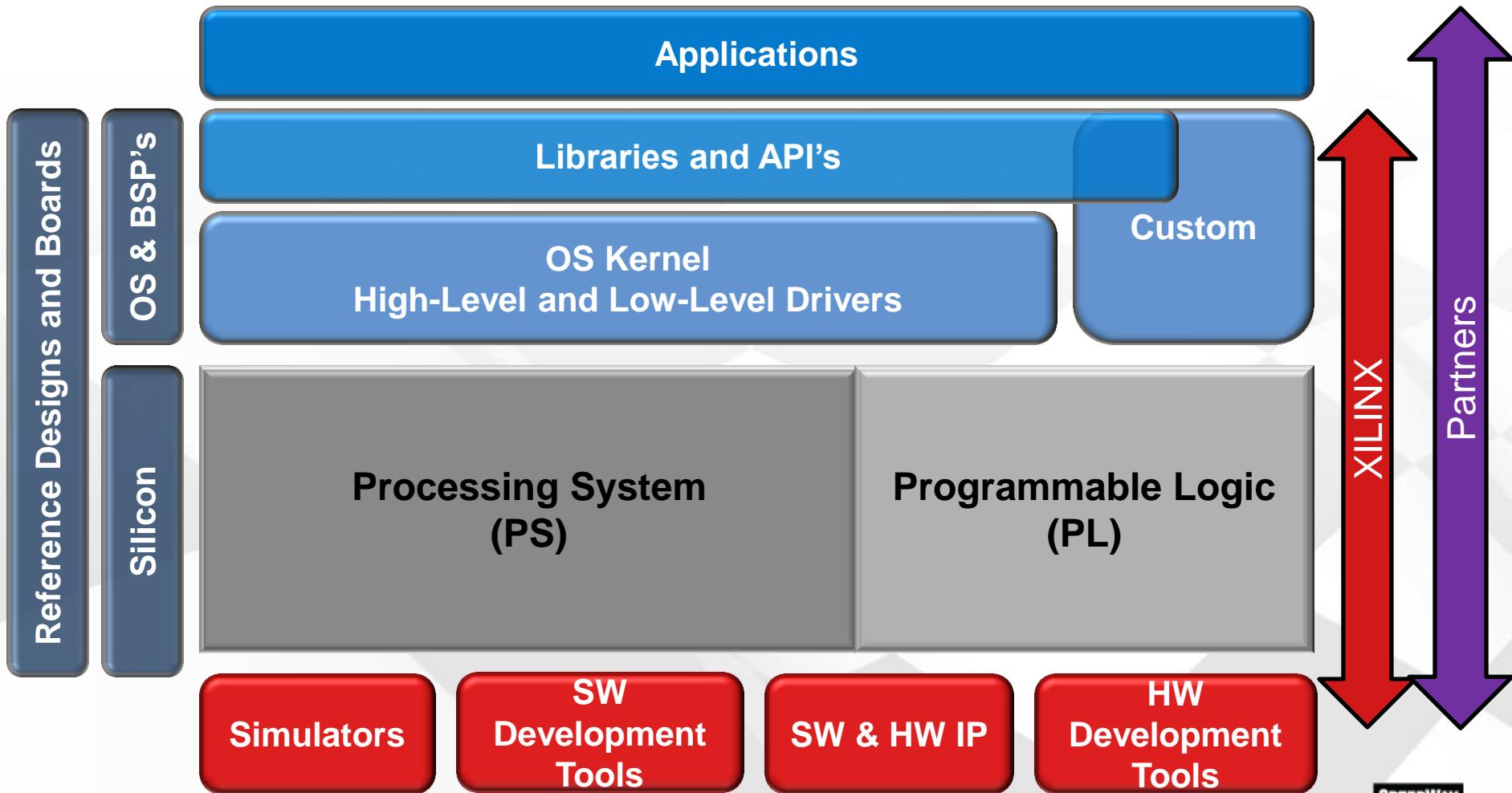
ZedBoard Overview

- Zynq Evaluation & Development (ZED) Board
- \$395
- Key features
 - Stand-alone Linux platform
 - High-speed communication interfaces
 - Support audio and 1080p video
 - Support expansion for analog and digital I/O



Buy it at <http://www.zedboard.org>

Zynq-7000 AP SoC Programming Model



Zynq-7000 OS Support

- Zynq-7000 supports a comprehensive collection of operating systems to suit your system needs

- Open Source Linux
 - Xilinx provides a freely downloadable Linux solution
 - Xilinx Git Repository
 - Xilinx acquired PetaLogix

- Open Source Android
 - Freely downloadable Android 2.3 solution

- Open Source FreeRTOS
 - Light-weight real-time OS
 - Used in applications that demand deterministic and real-time responsiveness to events in the system

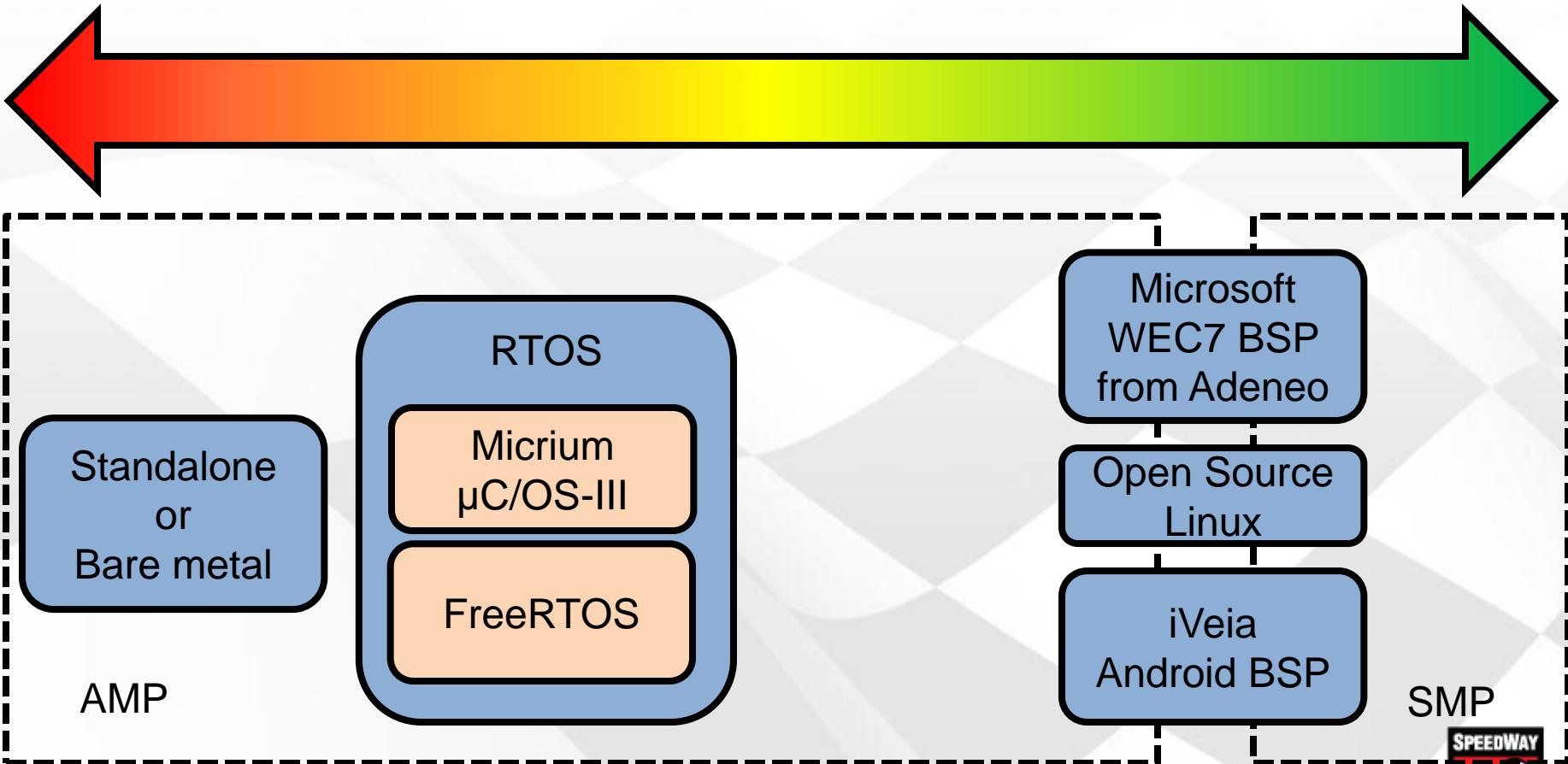


Choosing the Optimal Operating System

- What are your software application requirements?

Real-Time Performance

High System Performance



Zynq-7000 AP SoC Linux Software Platform

Applications



Libraries and API's

Custom

OS Kernel
High-Level and Low-Level Drivers



Processing System
(PS)

Programmable Logic
(PL)



If it works on Linux,
it works on Zynq!

Legal Disclaimer

- This course and related lab materials use source code made available and licensed with an open source license.
- It is up to you and your legal team to make sure your product distribution complies with the licensing requirements of whatever software package that you include in your product.
- We are not lawyers. We cannot and will not provide legal assistance or legal advice about any software licensing or patent issue. We won't be able to answer questions about the legal significance of the facts, licensing issues, conduct legal research, or provide information about the legal deadlines that might apply to your situation. You or your legal representative need to determine these issues for yourself before you distribute your product. A good place to start might be to see how not to do things, at the gpl-violations.org project.
- No one on ZedBoard.org or any Avnet.com web site, whether they are a volunteer contributor, end user, Avnet employee, or Avnet contractor can or should offer legal advice or speculation, about the Intellectual Property (IP), or any other legal topic, that may or may not be in the Linux kernel or derivative distribution. Any such questions or responses may be removed from the site without notice.



Implementing Linux on the Zynq™-7000 AP SoC

Chapter 1.1 – Booting ZedBoard

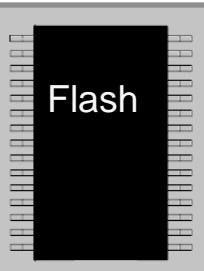


Accelerating Your Success™

Roadmap - Linux in Four Parts

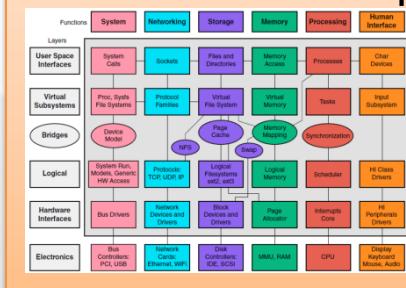
① Boot Loading – 3 stages

- Basic hardware initialization
- Loads Linux kernel
- Passing boot parameters



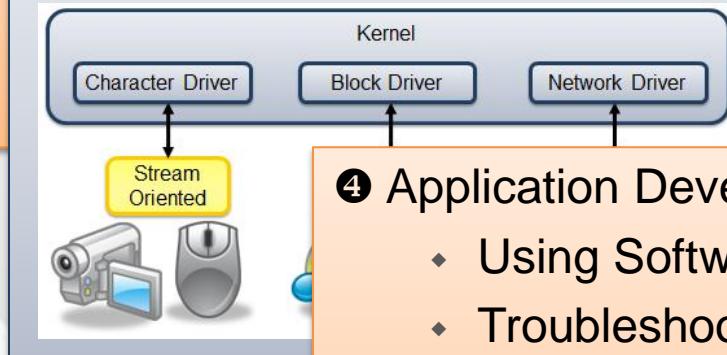
② Kernel

- Manages system resources
- Provides application services



③ Device Drivers

- Abstracts hardware details from software



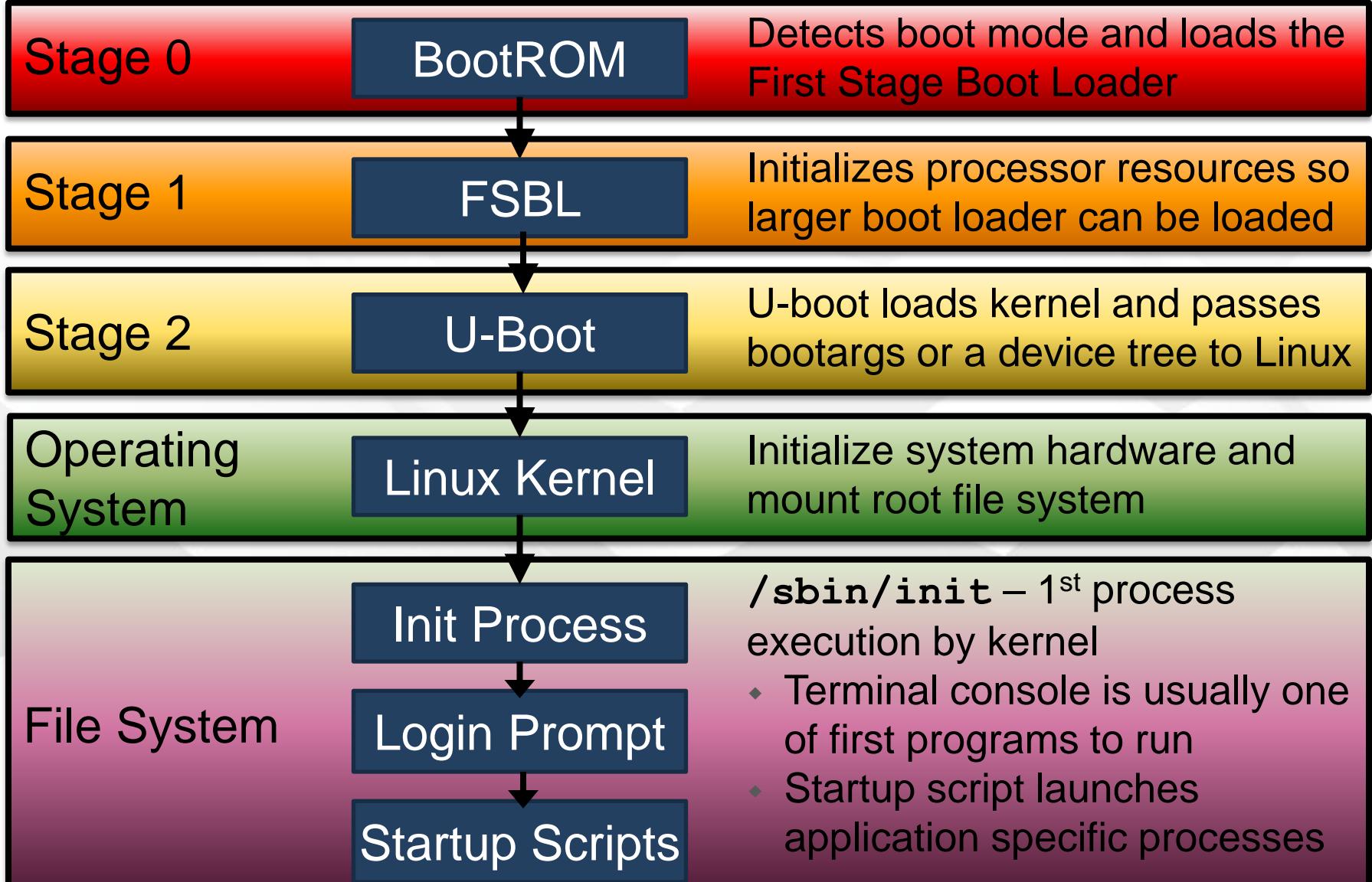
④ Application Development and Debug

- Using Software Development Kit
- Troubleshoot user applications

The terminal window shows the following boot log:

```
GEM: physdev defmap000, phydev->phy_id 0x1410dd1, phydev->addr 0x0
eth0: phy_id 0x0, phy_id 0x01410dd1
eth0: attach [Generic PHY] phy driver
IP-Config: Getting netmask 255.255.255.0
IP-Config: Getting broadcast 192.168.1.255
device: eth0, addr: 192.168.1.10, mask=255.255.255.0, gw=255.255.255.255,
host=192.168.1.10, domain=, nis-domain=none),
bootproto=255.255.255.254, rootserver=255.255.255.255, rootpath=
RAMDISK: gpt1: index 1, size 104857600 bytes
mmcblk0: new SD card at address e624
mmcblk1b: mmc0:e624 SU02G 1.84 GiB
mmcblk1b: mmcblk1b: mmcblk1b: mmcblk1b:
UPS: Mounted root (ext2 filesystem) on device 1:0.
devtcpfs: mounted
Freeing init memory: 144K
Starting rcu...
** Mounting filesystem
** Setting up imdev
** Starting httpd daemon
** Starting http daemon
** Starting ftp daemon
** Starting dropbear <ssh> daemon
rcs: complete
zyng>
```

Typical Linux Boot Sequence Overview



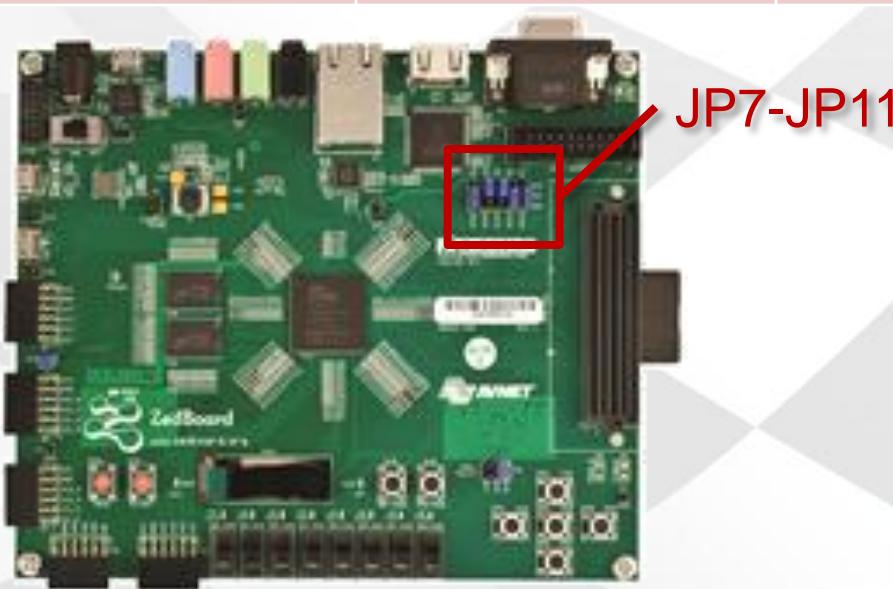
Zynq Boot Loader Components

Boot Stage	Operations	User Configurable	Zynq-7000 Terminology
Stage 0	ROM code detects desired boot mode (NAND, QSPI, JTAG) and loads executable code of First Stage Boot Loader (FSBL) from selected peripheral/interface	No	BootROM
Stage 1	Initializes external memory and system clocks so that a larger, more advanced boot loader (in our case U-boot) can be loaded	Yes	FSBL
Stage 2	Used to locate, load, and execute the Linux kernel and is also responsible for passing a device tree to the kernel	Yes	U-Boot

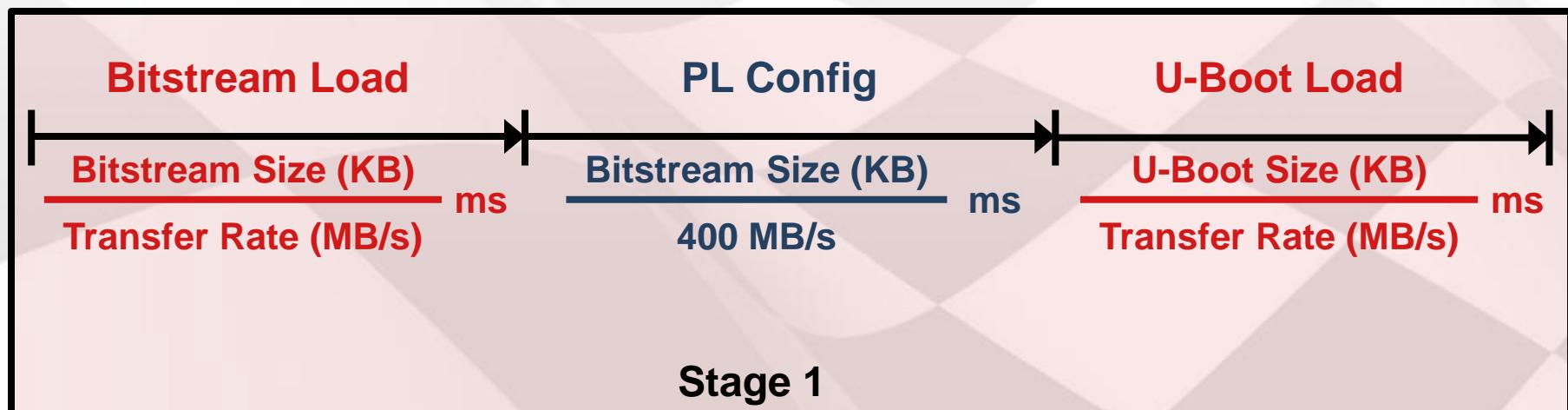
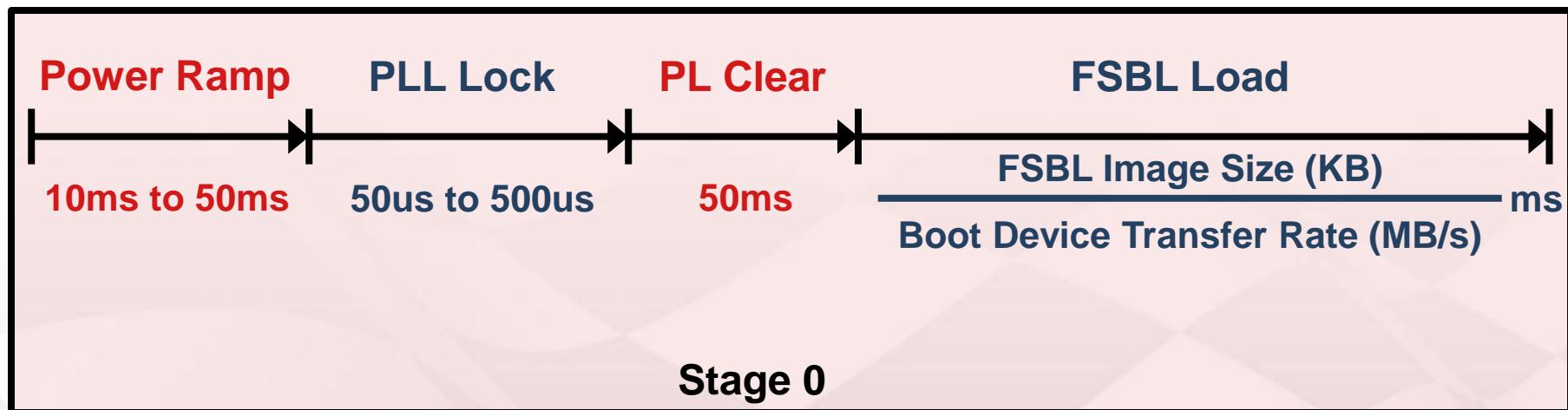
Zynq Boot Modes

- Boot mode latched at POR using MIO pins by BootROM

Mode	MIO[5] (JP10)	MIO[4] (JP9)	MIO[3] (JP8)
JTAG	0	0	0
NOR	0	0	1
NAND	0	1	0
Quad-SPI	1	0	0
SD Card	1	1	0



Factors Determining Boot Time



First Stage Boot Loader

- **Initialize Zynq PS**
 - PLLs
 - DDR memory controller
 - MIO
 - UART
- **Optionally configure PL with bitstream**
- **Load application code from boot medium to memory**
- **Transfer execution to application code (U-Boot)**

Create FSBL in SDK

- Generate directly from the SDK project template
- Initializes PS with XPS platform configuration
- Source code user editable
- Other Linux platforms refer to this code as *user boot code*

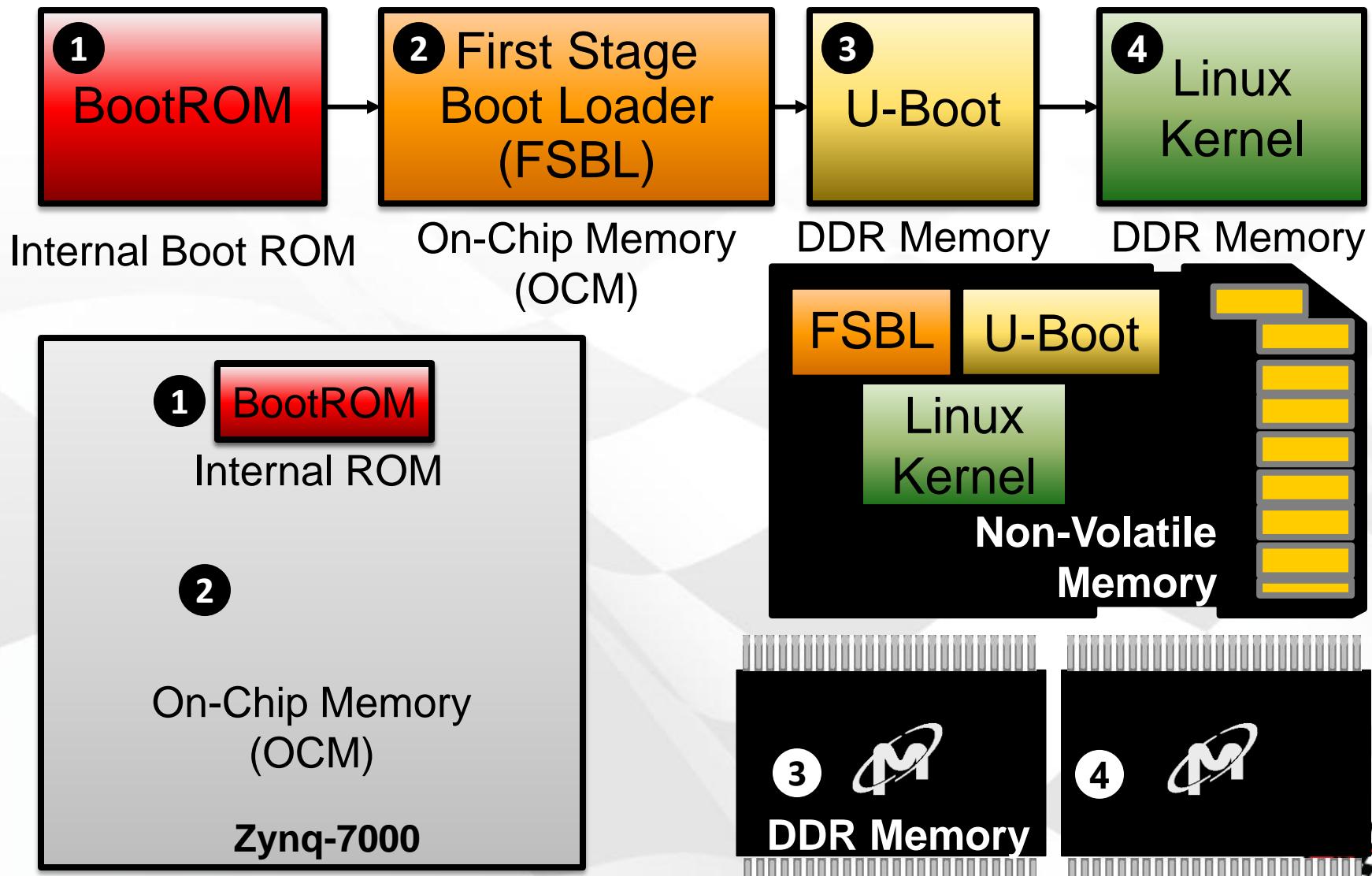
Select Project Template

Dhrystone
Empty Application
Hello World
lwIP Echo Server
Memory Tests
Peripheral Tests
SREC Bootloader
Xilkernel POSIX Threads Demo
Zynq FSBL

Description

First Stage Bootloader (FSBL) for Zynq. The FSBL configures the FPGA with HW bit stream (if it exists) and loads the Operating System (OS) Image or Standalone (SA) Image or 2nd Stage Boot Loader image from the non-volatile memory (NAND/NOR/QSPI) to RAM (DDR) and starts executing it. It supports multiple partitions, and each partition can be a code image or a bit stream.

Linux Boot Process Overview for Zynq



Checkpoint – Pop Quiz

Which of the following Zynq-7000 boot components are **NOT** user configurable?

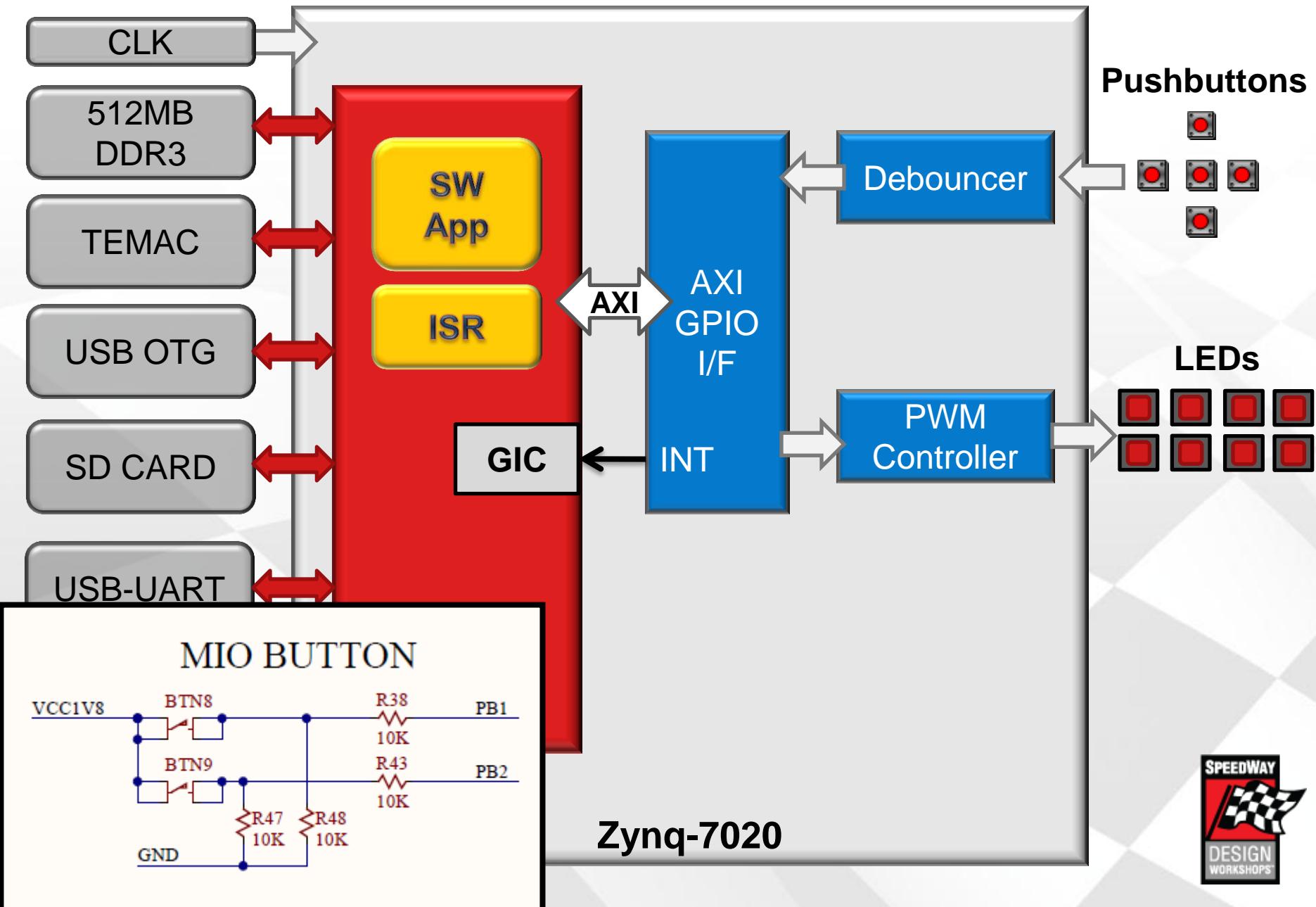
- 1) Linux kernel
- 2) BootROM
- 3) First Stage Boot Loader
- 4) Second Stage Boot Loader

Checkpoint – Pop Quiz

Which of the following Zynq-7000 Processing System hardware peripherals can be configured by the First Stage Boot Loader?

- 1) MIO
- 2) UART
- 3) DDR memory controller
- 4) PLLs
- 5) All of the above

Zynq Intro SpeedWay Platform with PL Peripherals



Lab 1.1

- **Creating the Zynq First Stage Boot Loader (FSBL)**
 - Begin with the hardware platform built during the Introduction to Zynq SpeedWay
 - Create a FSBL based upon PlanAhead hardware platform
- **Note: All SpeedWay lab materials, including lab instructions and solutions, can be downloaded from ZedBoard.org**

Implementing Linux on the Zynq™-7000 AP SoC

Chapter 1.2 – Introduction to U-Boot



Accelerating Your Success™

U-Boot Introduction

- **U-Boot is an open source software project**
- **U-Boot performs important boot functions**
 - Initializes the platform hardware needed to load kernel
 - Loads Linux kernel from boot medium to main memory (DDR)
 - Starts the Linux kernel with specified boot parameters
- **ZedBoard out of box demo runs the open-source boot loader U-Boot**

U-Boot Features

- **U-boot also provides some convenient features that help during development**
 - Provide network access
 - Ping IP addresses
 - Download binary images via TFTP
 - Memory test, copy, and comparison
 - Reads and writes arbitrary memory locations
 - Copies binary images from one location in memory to another
 - Configure and access hardware peripheral devices directly (e.g. Quad-SPI Flash, I2C, Ethernet)
 - Detect boot mode and run related boot macros

U-Boot Commands

- U-boot provides console interface to execute commands

Command	Function	Example
setenv	Sets an environment variable	setenv ipaddr 192.168.1.10
printenv	Lists environment variables and their contents	printenv printenv ipaddr
run	Runs a macro	run sdboot
dhcp	Attempts to set the IP address using DHCP	dhcp
tftp	Downloads a file into memory using TFTP	tftp 0x8000 zImage
ping	Send ICMP ECHO_REQUEST to remote network host	ping 192.168.1.50
boot	Runs the macro in the bootcmd variable	boot
fatload	Loads a file from a mounted FAT filesystem	fatload mmc 0 0x8000 zImage

U-Boot

- U-boot is configured using named text strings called environment variables
- The ‘setenv’ command is used to create, modify, and delete environment variables

	Example command	Explanation
Create	setenv myvar foo	Creates a new variable named ‘myvar’ with value ‘foo’
Modify	setenv ipaddr 192.168.0.10	Changes the value of the ‘ipaddr’ environment variable to 192.168.0.10
Delete	setenv myvar	Deletes the variable ‘myvar’ when you run setenv with no contents to set into the variable

- ‘saveenv’ command stores environment to non-volatile memory – **NOT Supported on ZedBoard**

- U-boot macros are environment variables that contain commands
 - Evaluated and executed using the 'run' command

Here is an example of the variable 'fetch_kernel' created as a macro, containing a command 'tftp'

```
> setenv serverip 192.168.1.10  
> setenv bootimage zImage  
> setenv fetch_kernel 'tftp ${serverip}:${bootimage}'
```

The variable 'fetch_kernel' exists now as a macro, containing command 'tftp' and a variable sequence

```
> run fetch_kernel
```

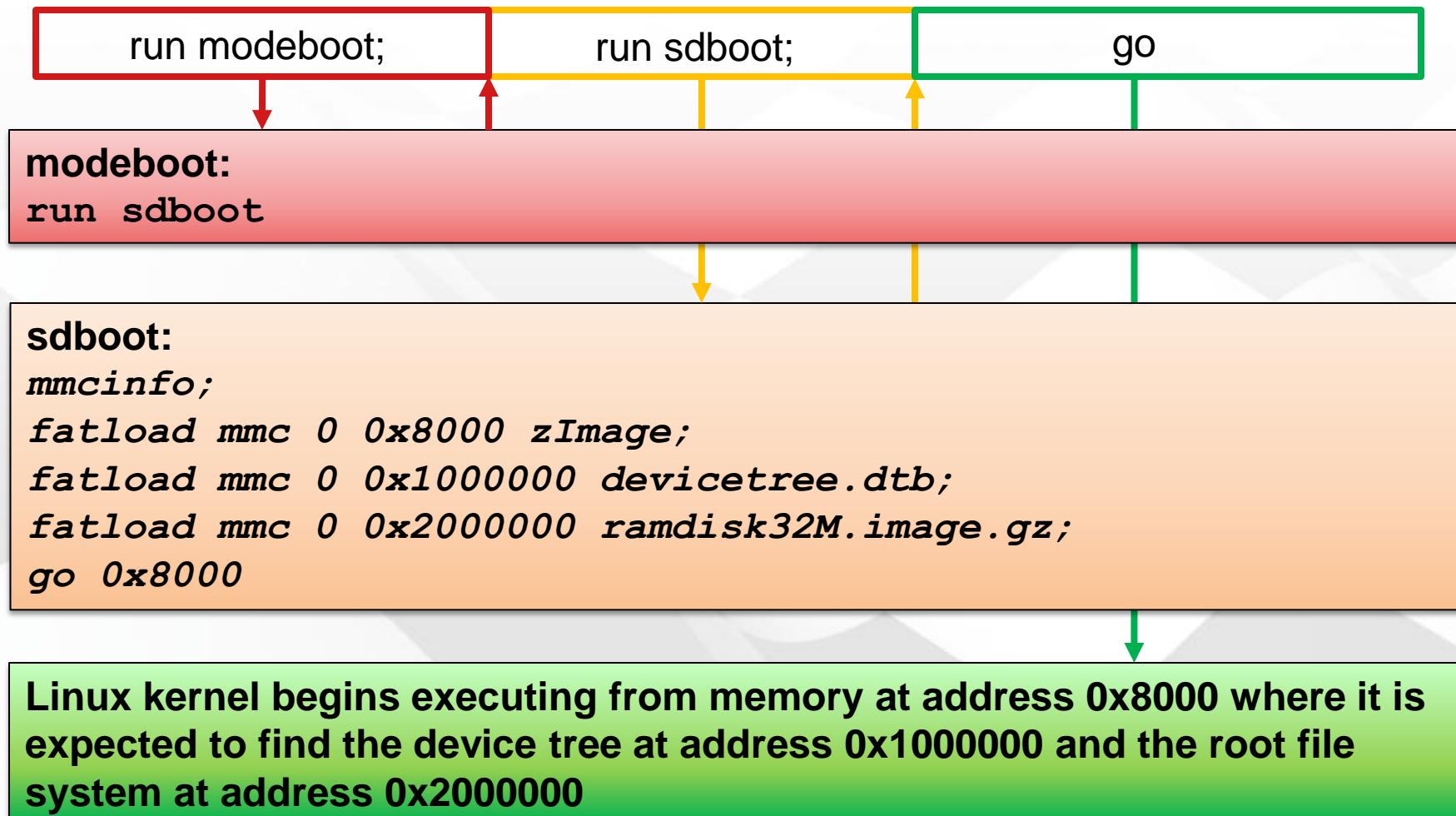
U-Boot Environment Variables

- **Certain environment variables are ‘reserved’**
 - Important standard variables used for specific U-Boot functionality

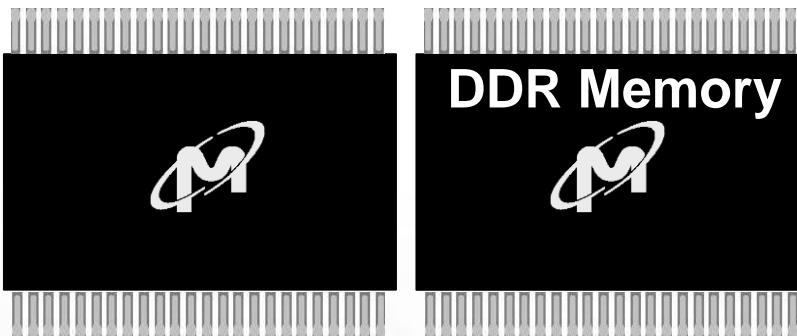
Variable	Function
bootcmd	Stores the macro that is run when the ‘boot’ command is executed
bootdelay	Delay in seconds until U-boot runs the ‘boot’ command (Set to -1 to disable autoboot)
ethaddr	Ethernet MAC address (read-only in our system)
ipaddr	The IP address of the board when running U-boot
serverip	The server IP address (IP address of the TFTP server used for <code>tftpboot</code>)
modeboot	Determines which method will be used to boot Linux (based upon BOOT_MODE_REG)

U-Boot Macros

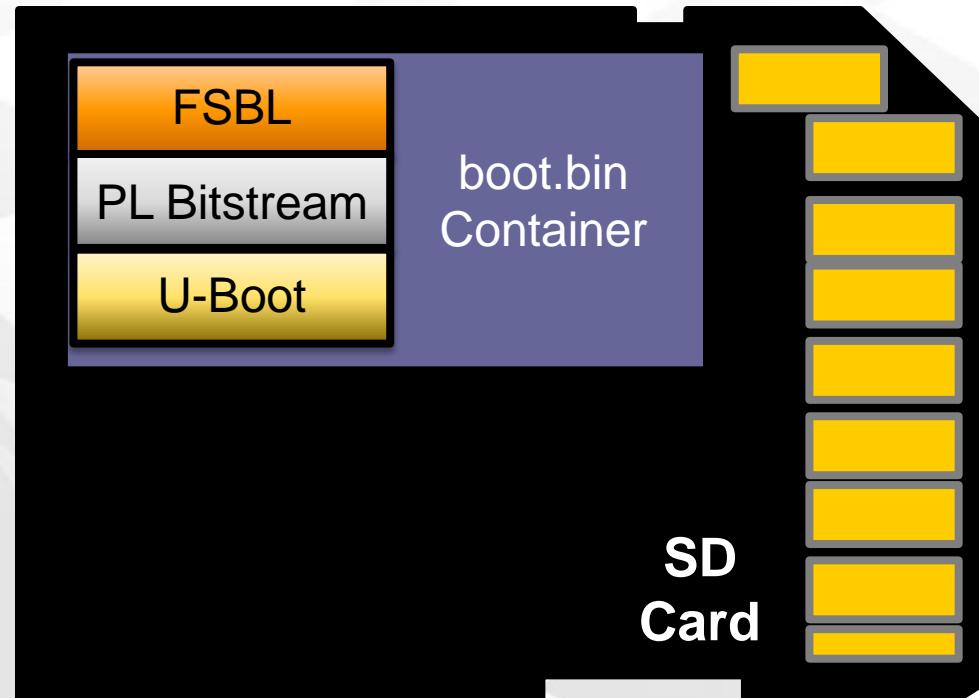
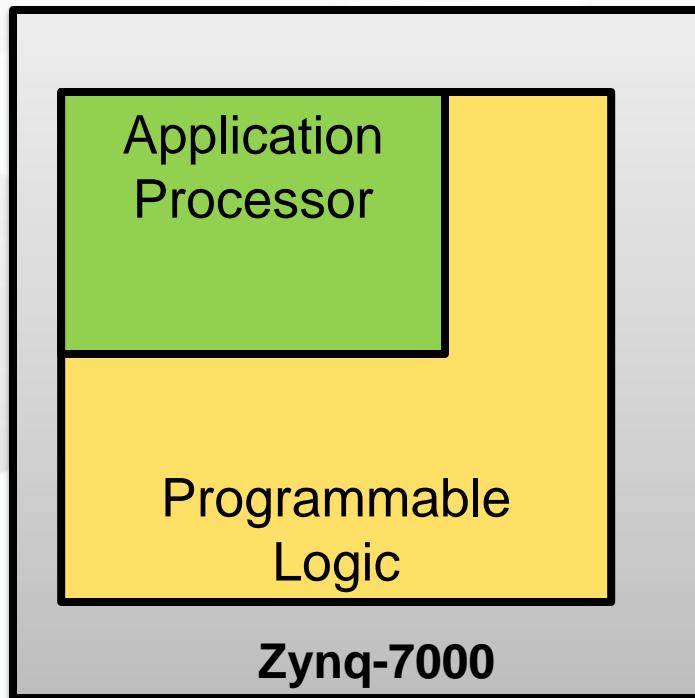
- Example use of macros: run modeboot



Booting U-Boot



- Copy this file to SD Card
 - boot.bin

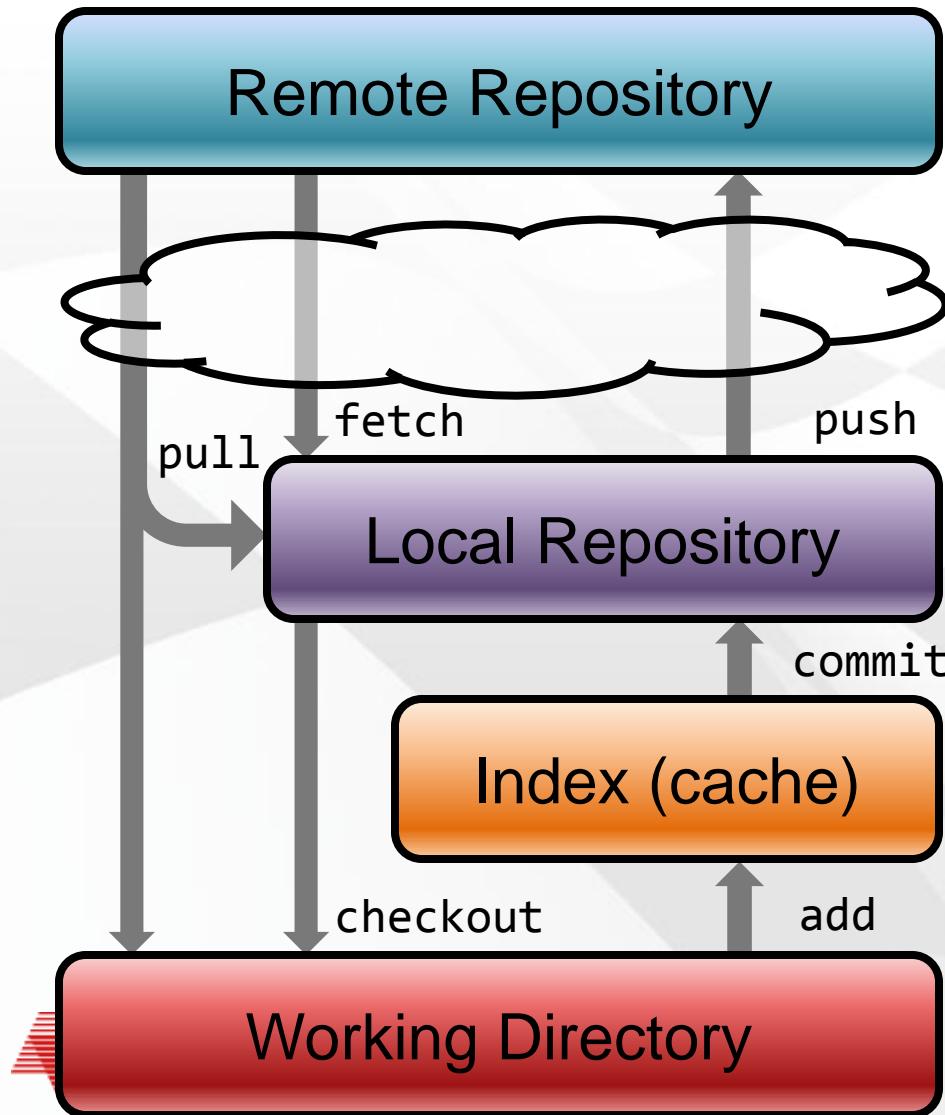


Where does the U-Boot source code come from?

- **U-boot source for Zynq hosted on the Xilinx git repository**
 - Based on source code from <git://git.denx.de>
 - Xilinx git repository can be cloned using the **git clone** command

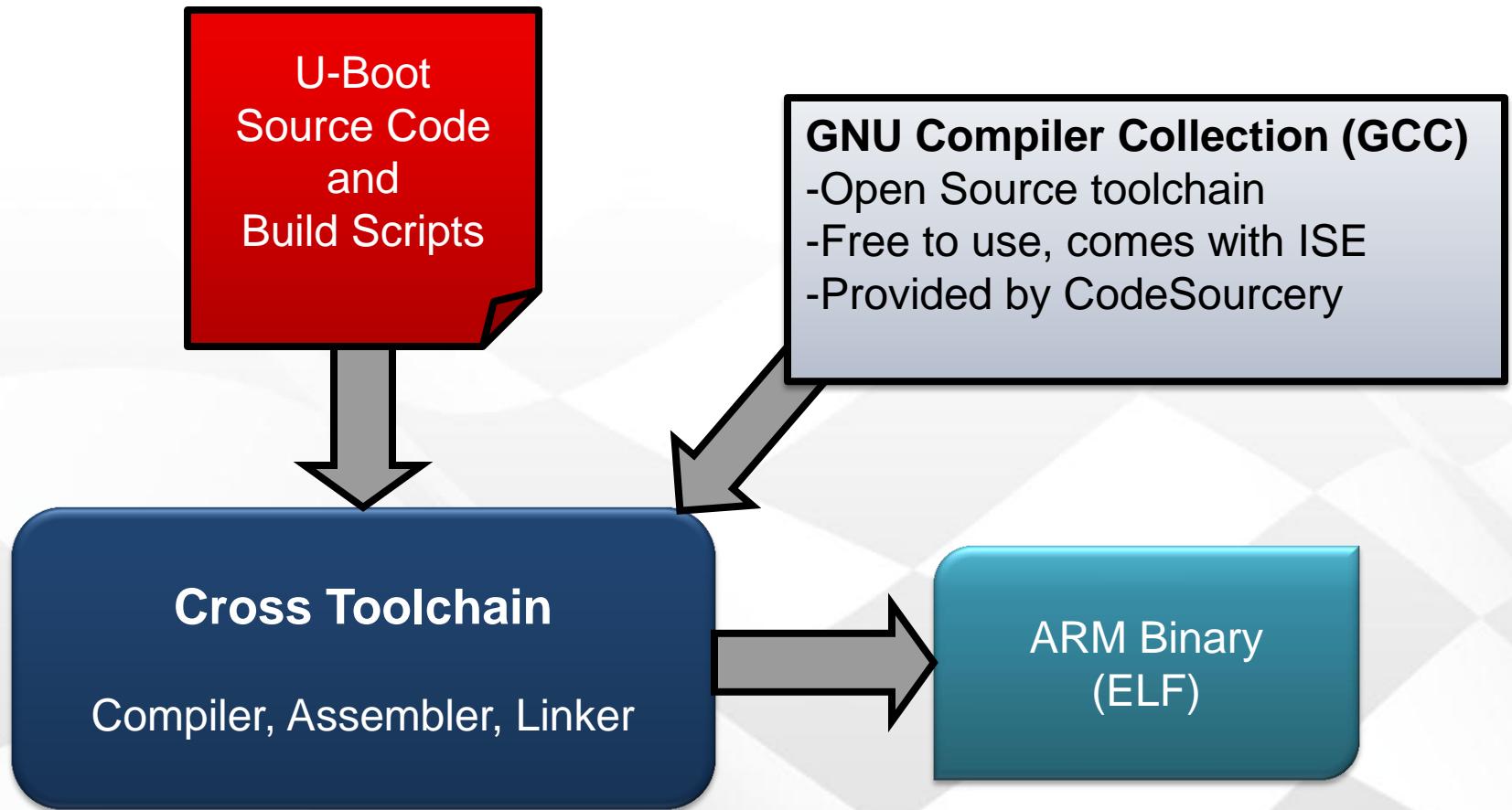
```
git clone git://git.xilinx.com/u-boot-xlnx.git
```

- What is Git and why is it used?



- **Source code management system**
- **Distributed revision control system**
- **Free software distributed under the terms of the GPL**
- **Intended to replace CVS and SVN**
- **For more info visit <http://git-scm.com/>**

U-Boot Build Process



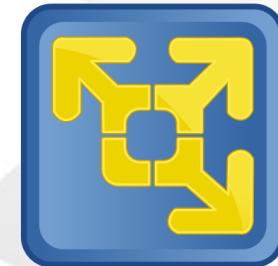
CodeSourcery cross-compiler for Zynq

- Runs on Linux host PC (x86 executable)
- Compiles executable code for Zynq (ARM executable)

Cross Build Platform

- Virtual machines can run desktop Linux
- SpeedWay labs use VMware Player to run CentOS Workstation
- Latest VMware Player can be downloaded from the VMware website:

<http://www.vmware.com/products/player/overview.html>



- CentOS can be downloaded from mirrors listed on CentOS website:

<https://www.centos.org/>



Checkpoint – Pop Quiz

What distribution of the ARM cross toolchain is used in this course?

- 1) CodeSourcery**
- 2) U-Boot
- 3) Git
- 4) CentOS

Lab 1.2

- **Building U-Boot from source code**
 - Configure and build U-Boot source tree for ZedBoard
- **Note: All SpeedWay lab materials, including lab instructions and solutions, can be downloaded from ZedBoard.org**

Implementing Linux on the Zynq™-7000 AP SoC

Chapter 1.3 – Zynq Boot Image



Accelerating Your Success™

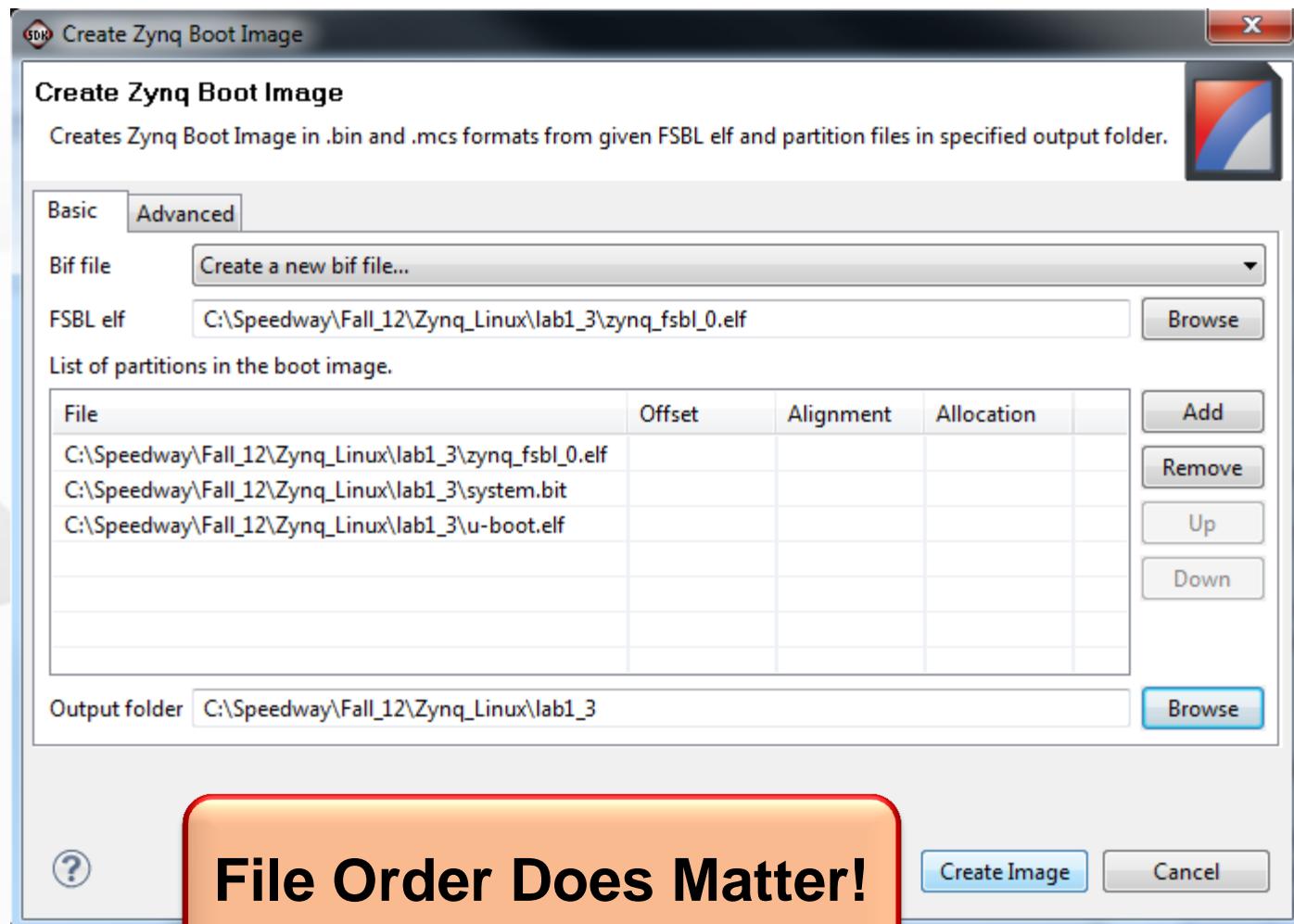
Boot Image Format File

- Zynq boot image can contain FSBL, bitstream, and application code or second stage boot loader
- Command line tool bootgen is used for constructing boot images for Zynq configuration
 - Example command line usage of bootgen
bootgen -image myDesign.bif -o i boot.bin
- Merges .BIT and .ELF files into a single boot image, using the format of a Boot Image Format (.BIF) file
 - Example BIF file entry

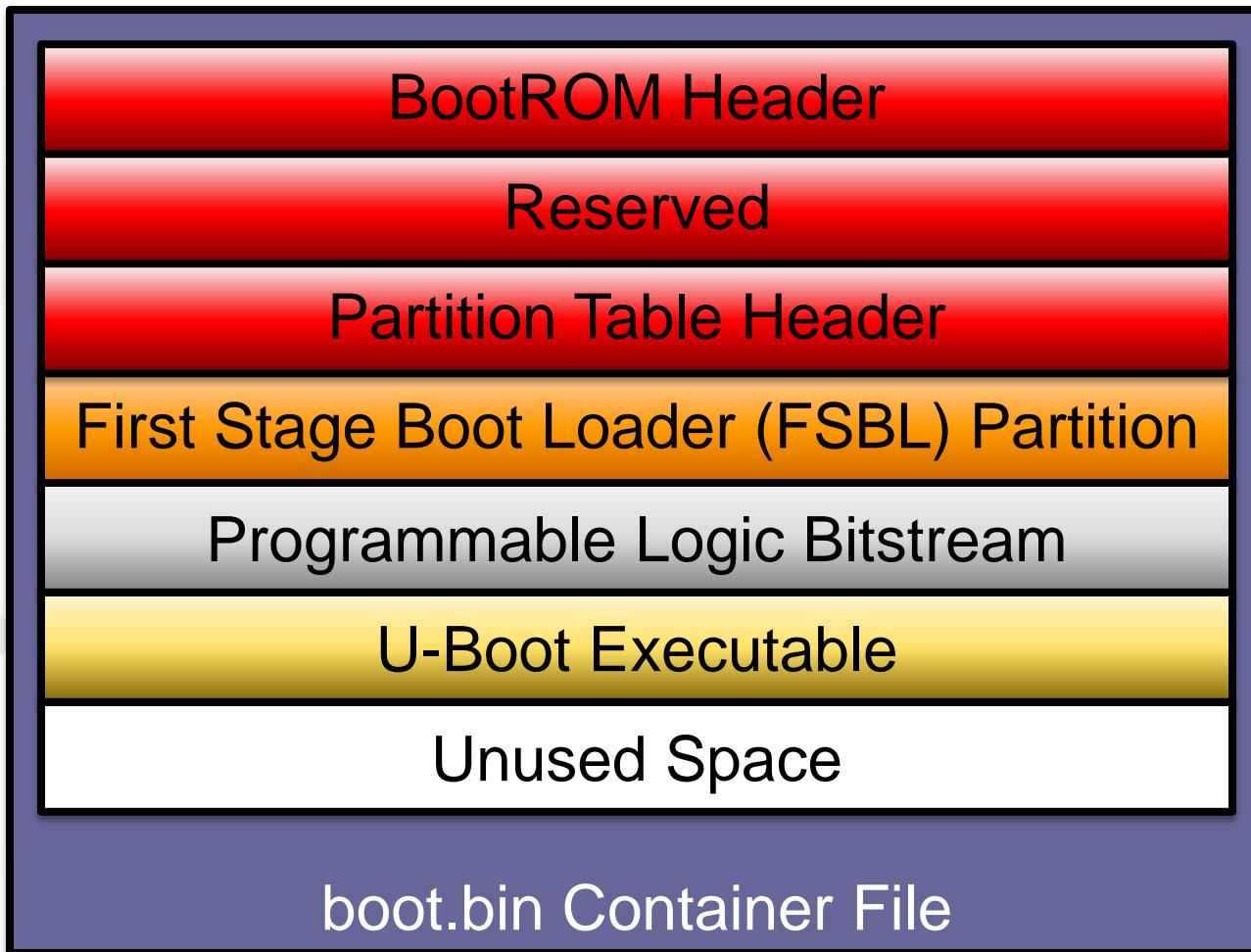
```
the_ROM_image:  
{  
    [bootloader] zynq_fsbl_0.elf  
    system.bit  
    u-boot.elf  
}
```

Create Zynq Boot Image in SDK

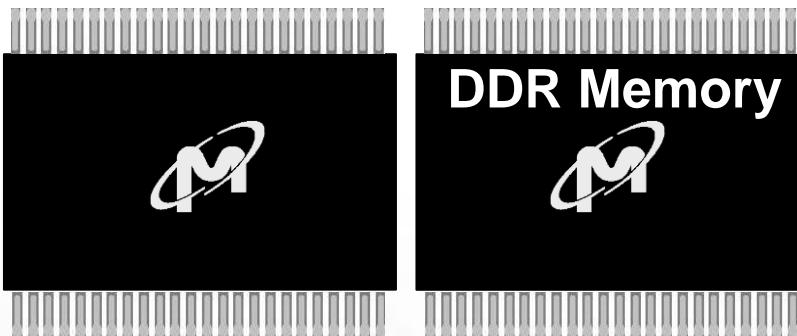
- Graphical front end to command line bootgen



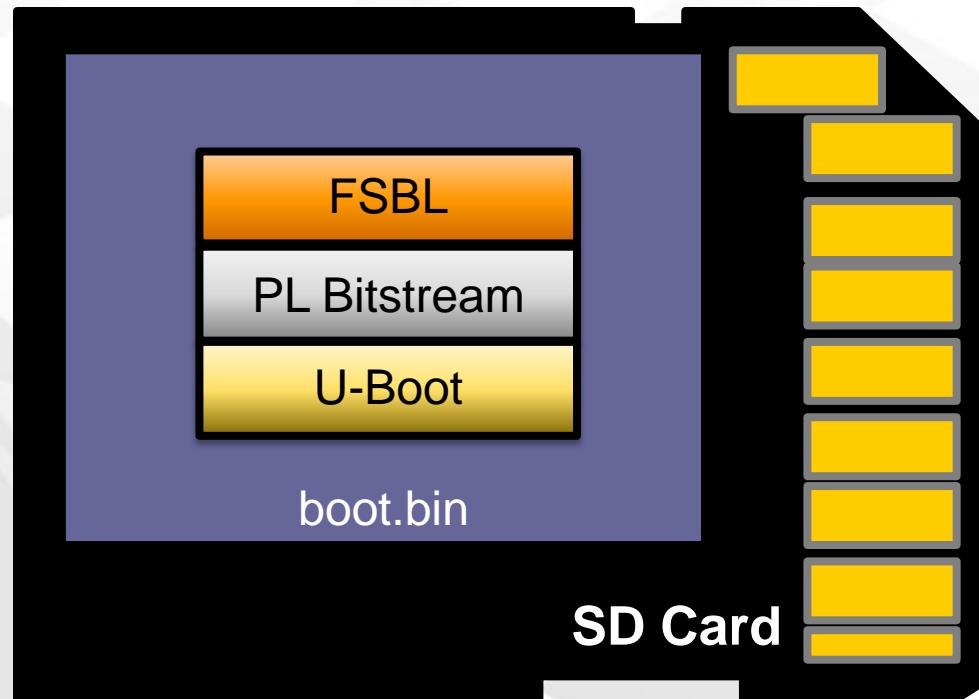
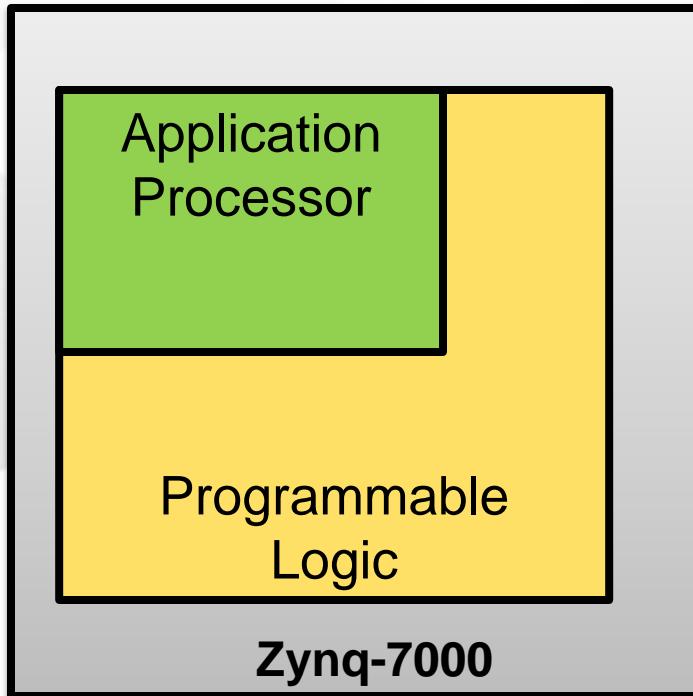
Example Of The Boot Image Format



Boot Image Mechanics



- Copy boot.bin to SD Card
- Set Boot jumpers to SD mode
- Power on the ZedBoard



Checkpoint – Pop Quiz

Which of the following tools is used to create a boot image which can be used to boot Zynq-7000?

- 1) makeboot**
- 2) bitgen**
- 3) bootgen**
- 4) libgen**

Lab 1.3

- Create a boot image with SDK
 - Boot to U-Boot on ZedBoard
-
- Note: All SpeedWay lab materials, including lab instructions and solutions, can be downloaded from ZedBoard.org

Implementing Linux on the Zynq™-7000 AP SoC

Chapter 2.1 – Linux Kernel Basics

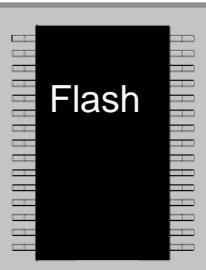


Accelerating Your Success™

Roadmap - Linux in Four Parts

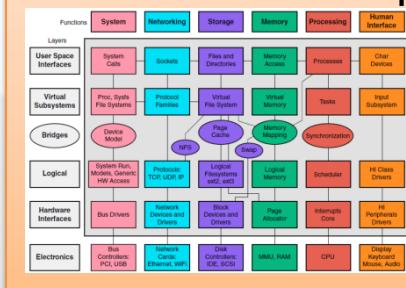
① Boot Loading – 3 stages

- Basic hardware initialization
- Loads Linux kernel
- Passing boot parameters



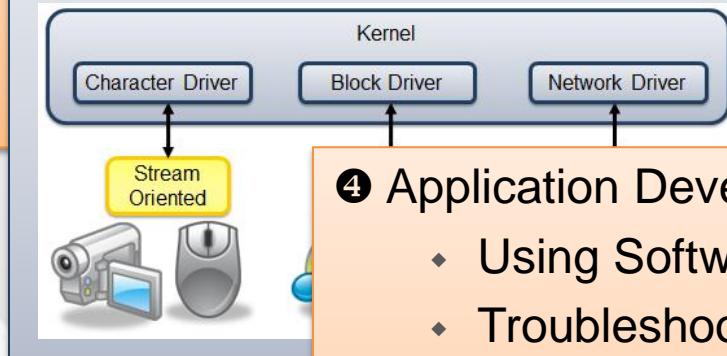
② Kernel

- Manages system resources
- Provides application services



③ Device Drivers

- Abstracts hardware details from software



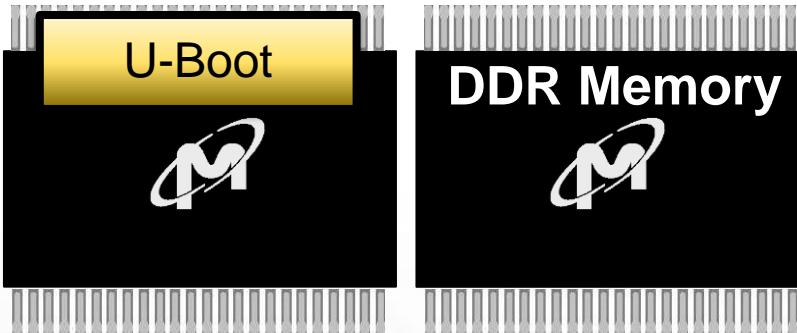
④ Application Development and Debug

- Using Software Development Kit
- Troubleshoot user applications

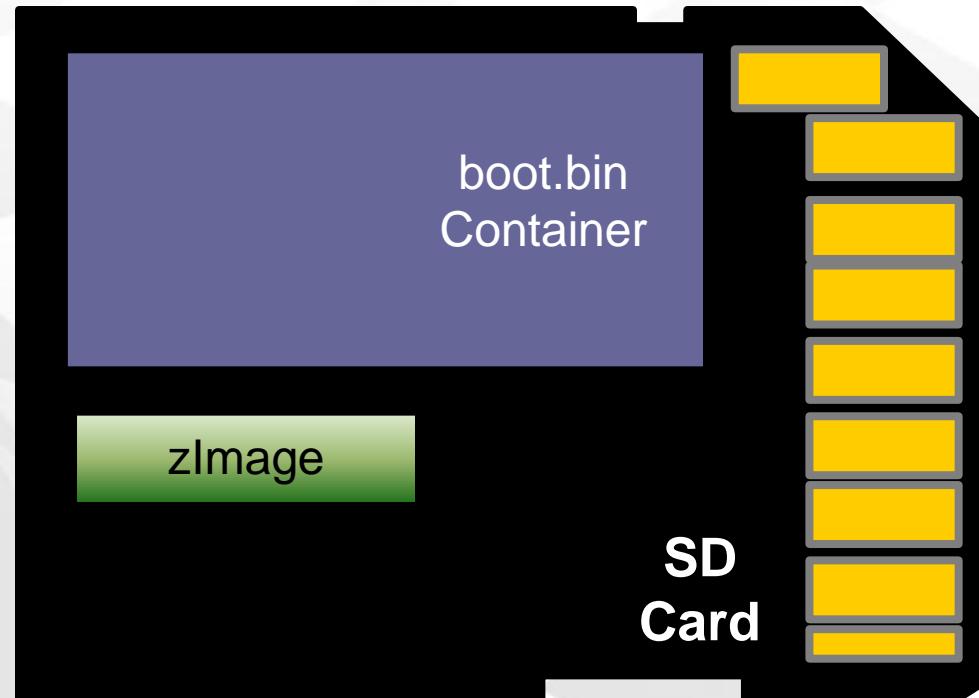
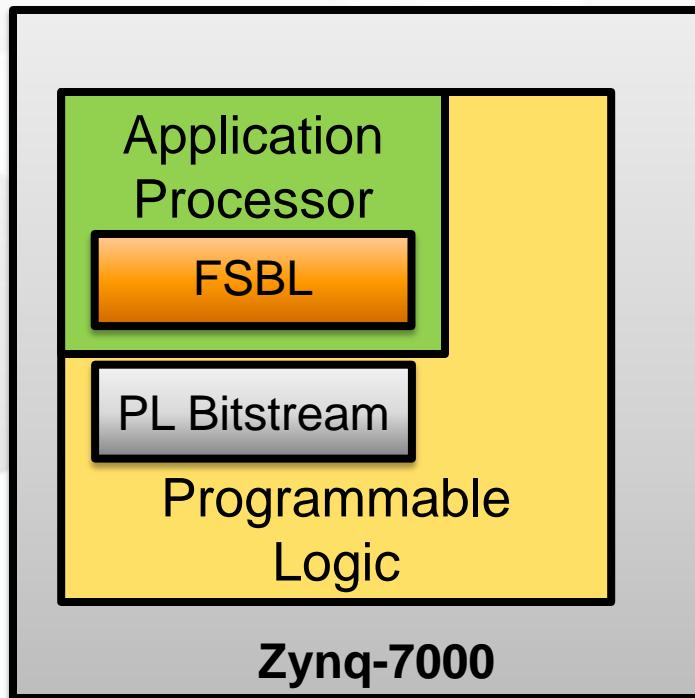
The terminal window shows the following boot log:

```
GEM: physdev defmap000, phydev->phy_id 0x1410dd1, phydev->addr 0x0
eth0: phy_id 0x0, phy_id 0x01410dd1
eth0: attach [Generic PHY] phy driver
IP-Config: Getting netmask 255.255.255.0
IP-Config: Getting broadcast 192.168.1.255
device: eth0, addr: 192.168.1.10, mask=255.255.255.0, gw=255.255.255.255,
host=192.168.1.10, domain=
MAC address: 255.255.255.255 (none)
rootserver=255.255.255.255, rootpath=
RAMDISK: gsize: 1024000, max_size: 1024000, block 0
mmcblk0: new SD card at address e624
mmcblk1b: mmc0:e624 SU02G 1.84 GiB
mmcblk1b: mmcblk1b: mmcblk1b: mmcblk1b:
UPS: Mounted root (ext2 filesystem) on device 1:0.
devtcpfs: mounted
Freeing init memory: 144K
Starting rcu...
** Mounting filesystem
** Setting up imdev
** Starting httpd daemon
** Starting http daemon
** Starting ftp daemon
** Starting dropbear <ssh> daemon
rcs: complete
zyng>
```

Booting Linux



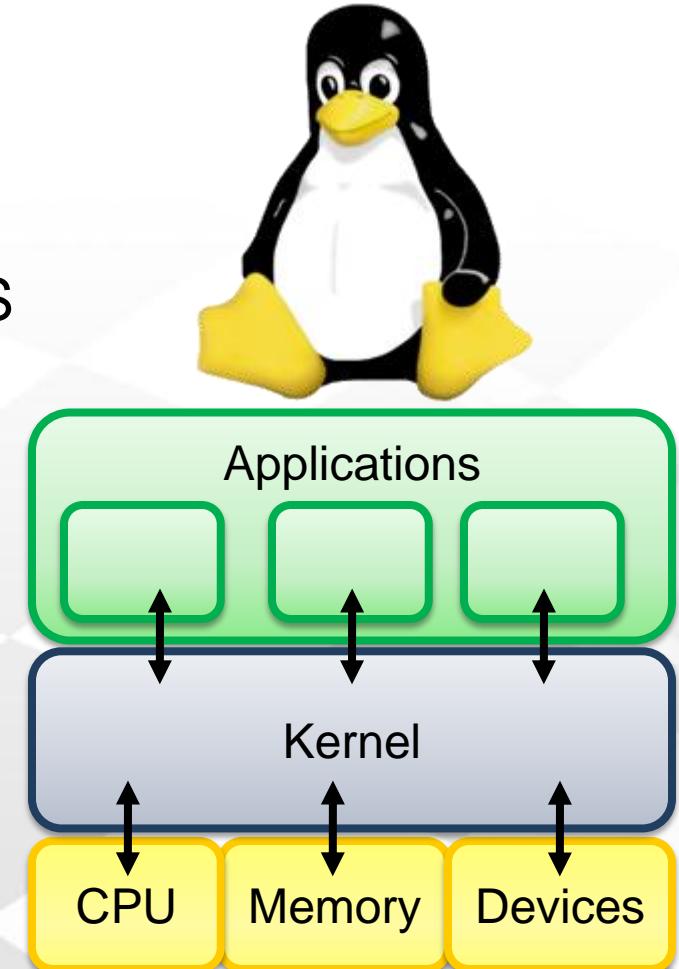
- Copy this file to SD Card
 - zImage



Linux Kernel Basics

- **Advantages of Linux**

- Broad use as an open source desktop, server, and embedded OS
- Feature-rich
 - Symmetric multiprocessing
 - Preemptive multitasking
 - Shared libraries
 - Device drivers
 - Memory management
 - IP networking
 - Support for multiple file systems



- **Linux kernel is free and open-source software**

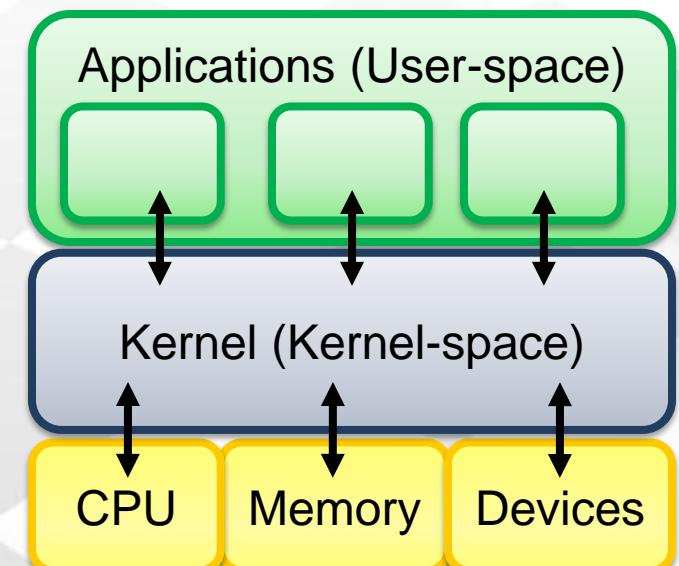
- Licensed under the GNU Public License, v2

Kernel Space Vs. User Space

- **Critical operations must be performed in Kernel-space**
 - Device drivers and kernel code run in kernel-space with full access to the underlying hardware
 - Context switches and interrupt handling performed by kernel

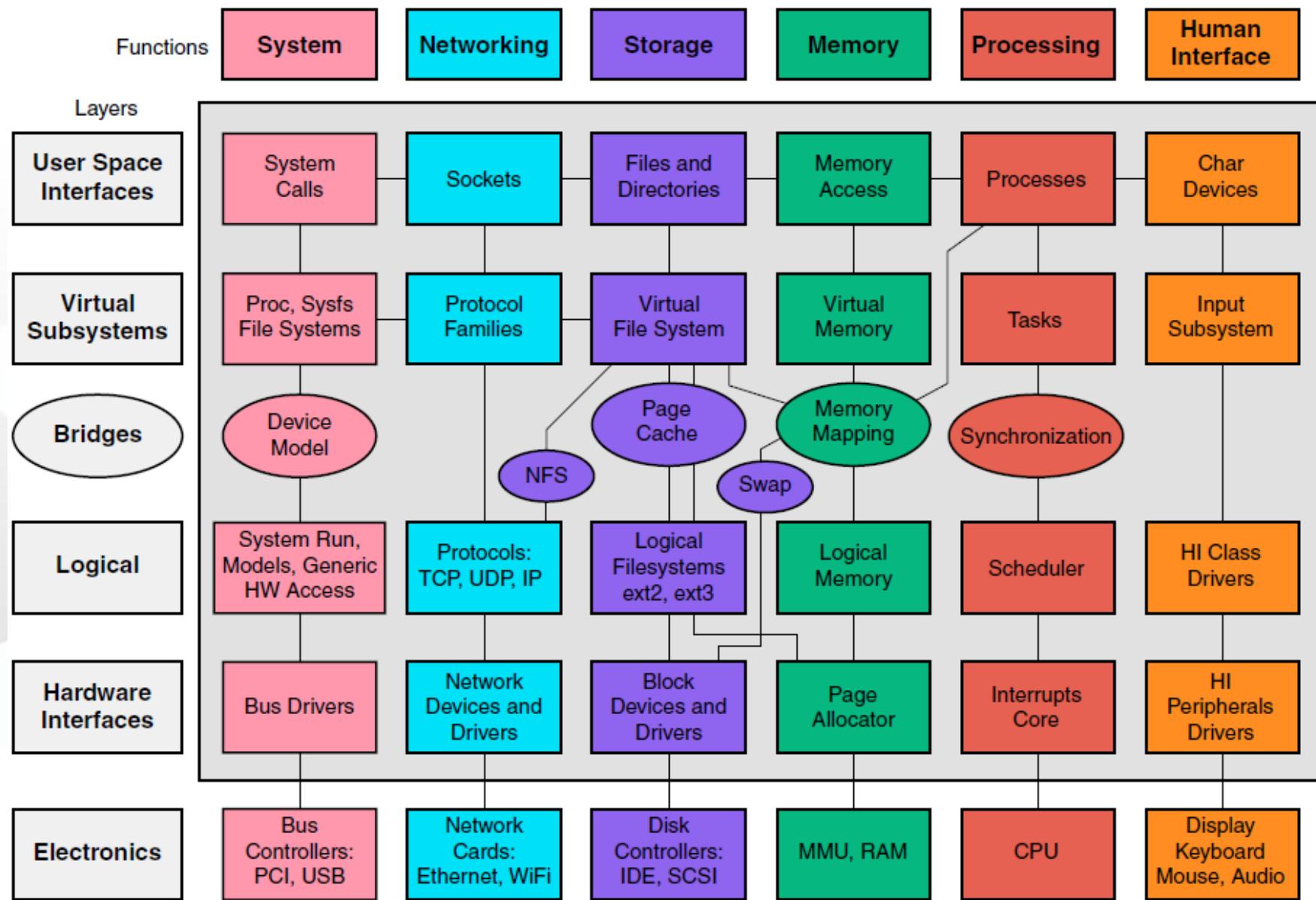
- **Applications run in User-space**

- Allocated their own memory space using a virtual memory system
- Access to physical memory space is prohibited except via kernel APIs



- **Linux replaces standalone BSP within SDK**

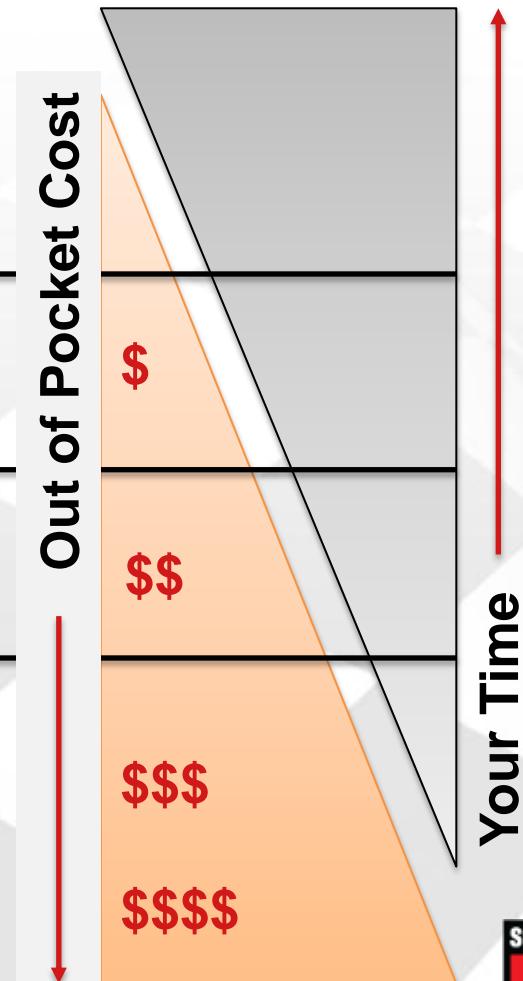
Linux Device Drivers



Linux Kernel Support Model

Support can come from many sources...

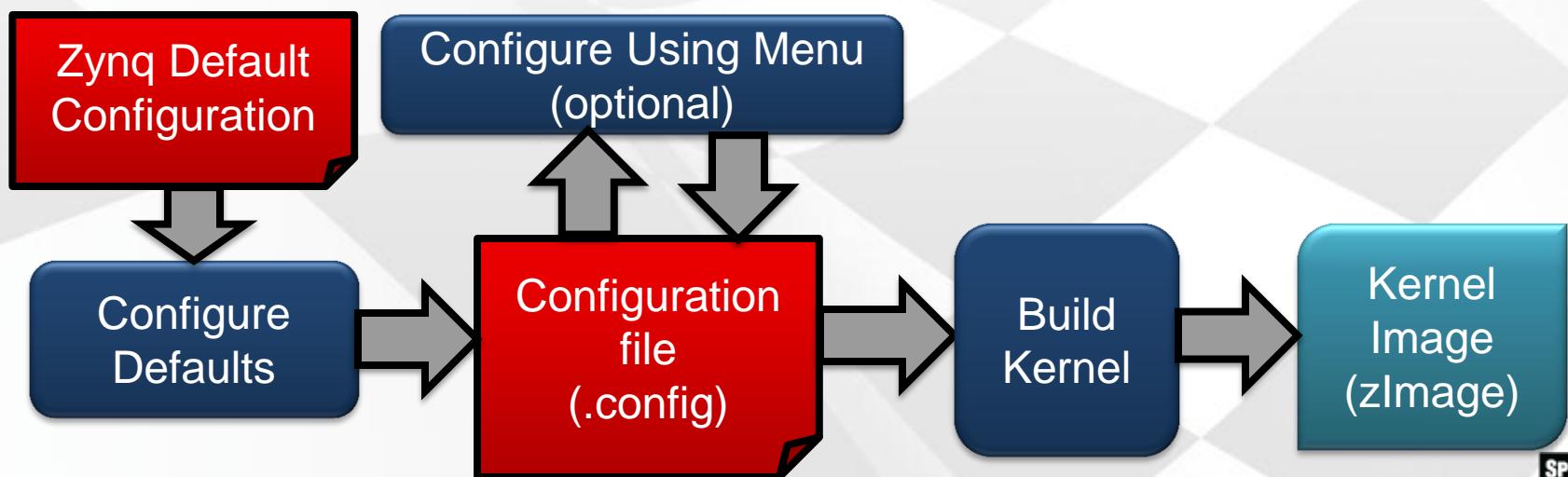
- **Community-based support**
 - Websites, forums, and mailing lists
- **Silicon vendors and distributors**
- **Board vendors**
- **Paid support options:**
 - Consultants
 - Commercially-available support



Linux Kernel Configuration and Build

- **Kernel is highly configurable**

- Add / remove support for
 - Debugging
 - Specific device drivers
 - Types of file systems
 - Boot options



Linux Kernel Basics

- **Default configuration available for Zynq platforms**

- `xilinx_zynq_defconfig` – Used for ZC702 and ZedBoard
- Located in `linux-xlnx/arch/arm/configs/` folder

The image shows two windows side-by-side. On the left is a text editor window titled "xilinx_zynq_defconfig (~/linux-xlnx/arch/arm/configs)". It displays the contents of the configuration file, which includes various kernel configuration options for the Zynq platform. On the right is a terminal window titled "Terminal" showing the "Linux/arm 2.6.39 Kernel Configuration" menu. The menu lists several kernel modules and features, with "Patch physical to virtual translations at runtime (EXPERIMENT)" currently selected.

```
# Xilinx Specific Options
#
CONFIG_ZYNQ_EARLY_UART1=y
# CONFIG_ZYNQ_EARLY_UART_EP107 is not set
CONFIG_XILINX_FIXED_DEVTREE_ADDR=y
CONFIG_XILINX_L1_PREFETCH=y
CONFIG_XILINX_L2_PREFETCH=y
# CONFIG_XILINX_TEST is not set
CONFIG_ZYNQ_DEFAULT_KERNEL=y
# CONFIG_ZYNQ_AMP_CPU0_MASTER is not set
# CONFIG_ZYNQ_AMP_CPU1_SLAVE is not set
# CONFIG_ZYNQ_CPU1_TEST is not set

#
# Processor Type
#
CONFIG_CPU_V7=y
CONFIG_CPU_32v6K=y
CONFIG_CPU_32v7=y
CONFIG_CPU_ABRT_EV7=y
CONFIG_CPU_PABRT_V7=y
CONFIG_CPU_CACHE_V7=y
```

File Edit View Terminal Help
.config - Linux/arm 2.6.39 Kernel Configuration

Linux/arm 2.6.39 Kernel Configuration

Arrow keys navigate the menu. <Enter> selects 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 [] excluded <M> module <>

[] Patch physical to virtual translations at runtime (EXPERIMENT)
General setup --->
[*] Enable loadable module support --->
[*] Enable the block layer --->
System Type --->
Bus support --->
Kernel Features --->
Boot options --->
CPU Power Management --->
Floating point emulation --->
v(+)

<Select> < Exit > < Help >

Plain Text ▾ Tab Width: 8 ▾ Ln 298, Col 1 INS

DESIGN WORKSHOPS™

Building the Linux Kernel

- **Kbuild System**

- Utilizes GNU make tool and a highly specialized set of rules to build the kernel

- **Required environment variables**

- ARCH=<target architecture, e.g. arm, powerpc, i386>
- CROSS_COMPILE=<prefix to gcc for cross toolchain>

```
> make ARCH=arm CROSS_COMPILE=arm-xilinx-linux-gnueabi-
```

Checkpoint – Pop Quiz

In Linux terminology, where must all critical software operations be performed?

- 1) I/O-space
- 2) User-space
- 3) **Kernel-space**
- 4) File-space

Lab 2.1

- **Kernel configuration and build from source code**
 - Configure and build Linux kernel source tree for Zynq
- **Note: All SpeedWay lab materials, including lab instructions and solutions, can be downloaded from ZedBoard.org**

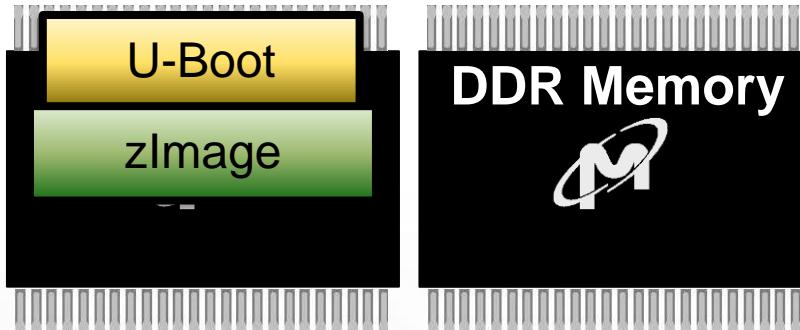
Implementing Linux on the Zynq™-7000 AP SoC

Chapter 2.2 – Linux Root File System



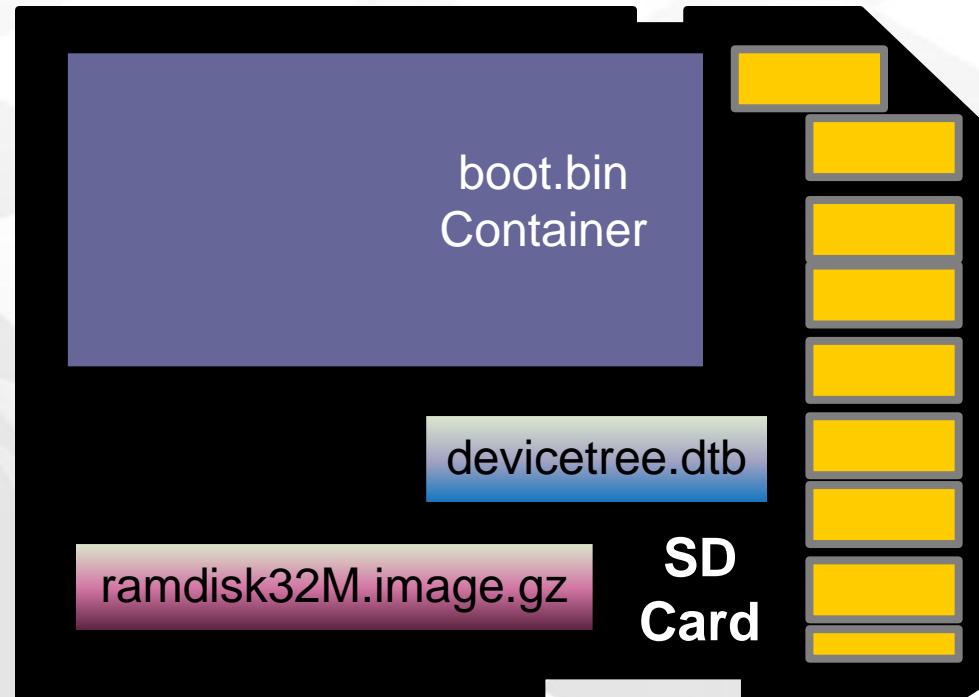
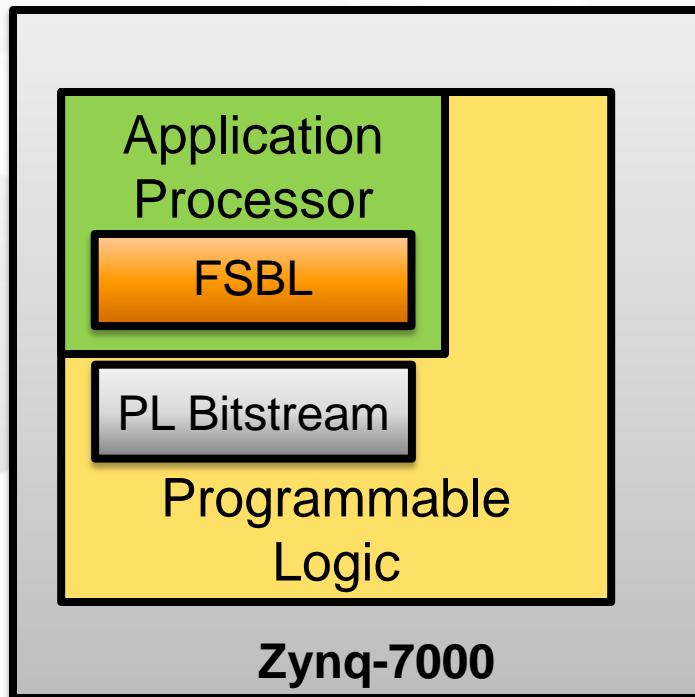
Accelerating Your Success™

Booting Linux



- Copy these files to SD Card

- devicetree.dtb
- ramdisk32M.image.gz

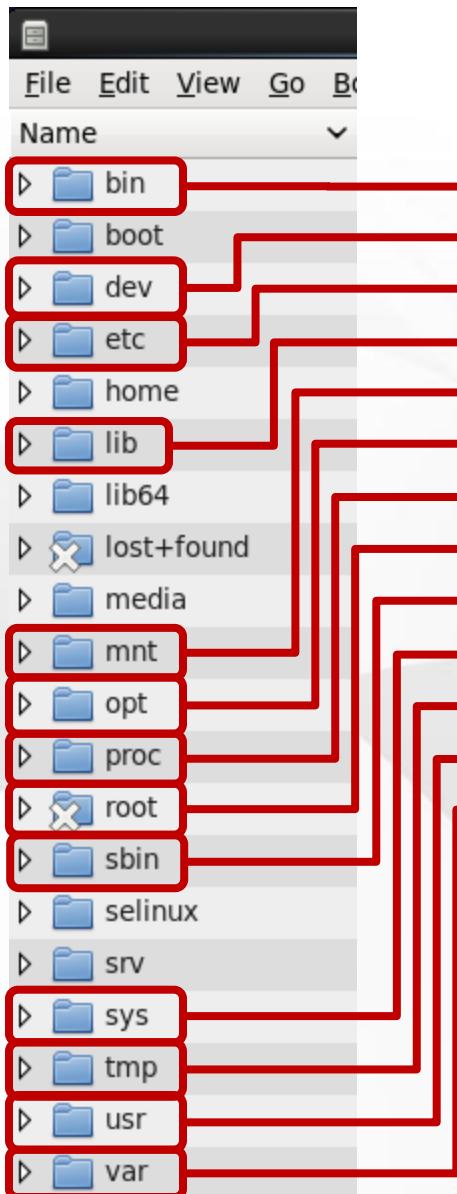


Linux Root File System

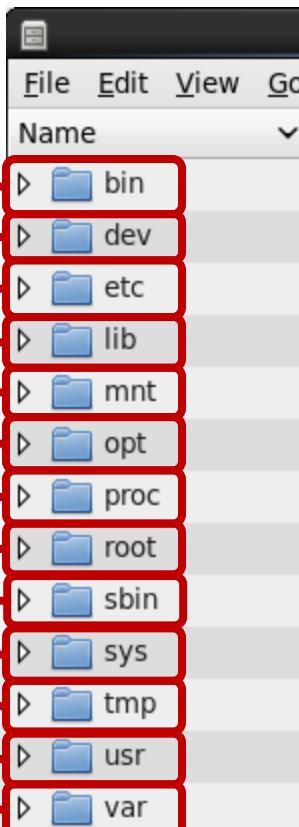
- **Root file system is an essential component of any Linux system and contains many critical system components**
 - Applications
 - Configuration files
 - Shared libraries
 - Data files
- **Different file system types used for different media**
 - RAM
 - SD Card
 - NAND/NOR Flash
 - Network
- **Mounted immediately after kernel initialization completes**
- **Contains first application run by the init process**

Desktop vs. Embedded Root File Systems

Desktop (CentOS)



Target (Embedded)



Folders common to Desktop Linux:

/dev – System devices (Chapter 3.1 topic)

/root - Storage for super user files

- Each user gets their own folder (e.g. /home/user)
- Similar to “My Documents” in Windows
- “root” user is different, that user’s folder is at /root

/mnt - Mount point for other file systems

- Linux only allows one root file system but other disks can be added by mounting them to a directory in the root file system
- Similar to mapping a drive under Windows

/lib - System libraries

- Location of system shared object libraries
- Similar to Windows “C:\Windows\System”

/sys and /proc - Virtual file systems location

- Exposes kernel parameters (kobjects) as files
- Similar to Windows Registry

/usr - Storage for user binaries

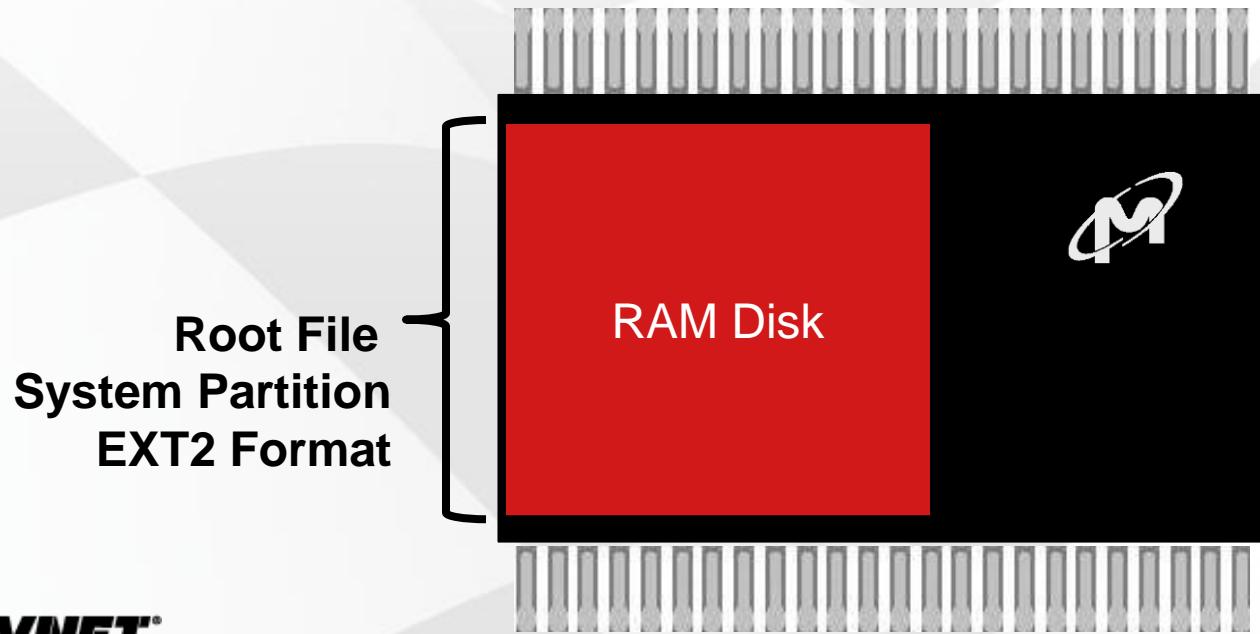
- Similar to “Program Files” in Windows
- Linux system programs are stored in here

File System Types

Type	Location	Features
FAT32	HDD, SSD, SD Card, USB Drive	File system commonly used with Windows operating system
EXT2, EXT3, EXT4 (EXTended File System)	HDD, SSD, SD Card, RAM	Commonly used file systems rotating and solid-state drives or other devices with drive-like logical interfaces (such as USB, SD Card)
JFFS and JFFS2 (Journaling Flash File System)	Non-volatile memory (Flash)	One of the first flash file systems supported in Linux. Good for smaller file systems (under 64MB)
NFS (Network File-System)	Network Server	Small footprint, read/write, good for development purposes

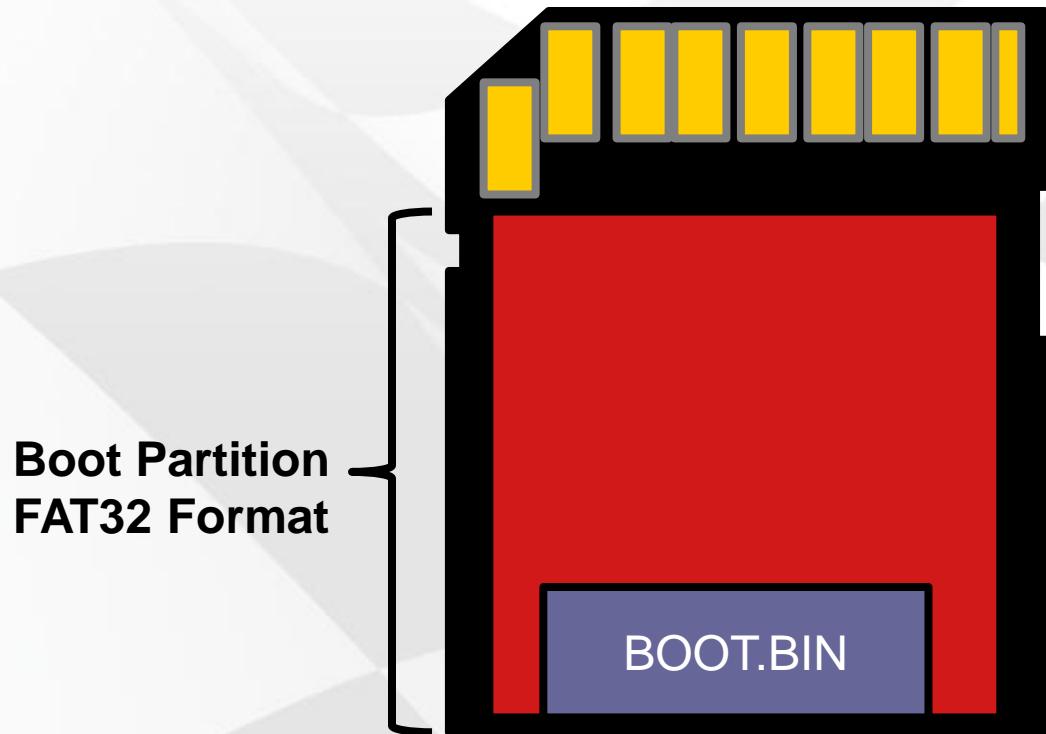
File Systems Used in Labs

- Kernel can mount a block device image as a RAM Disk
 - Partition must be formatted to a supported file system type
 - In this case, we're using EXT2 (EXTended File System 2)
 - Read/write, volatile file contents
 - Limited by amount of RAM available on system



File Systems Used in Labs

- Kernel can mount a partition of the SD Card
 - Partition must be formatted to a supported file system type
 - In this case, we're using FAT32 (32-bit File Allocation Table)
 - Read/write, non-volatile file contents
 - Can also be mounted under the root file system



Checkpoint – Pop Quiz

Which of the following file systems are supported by the Zynq BootROM for loading the boot image file from the SD card boot device?

1) FAT32

2) NTFS

3) EXT3

4) NFS

Lab 2.2

- **Build Busybox**
 - Used as command interpreter and toolset for Linux
- **Build Dropbear**
 - Contains SSH server application for remote control of system
- **Creating the basic root file system**
 - Create directory structure
 - Add critical libraries and user space applications
- **Note: All SpeedWay lab materials, including lab instructions and solutions, can be downloaded from ZedBoard.org**

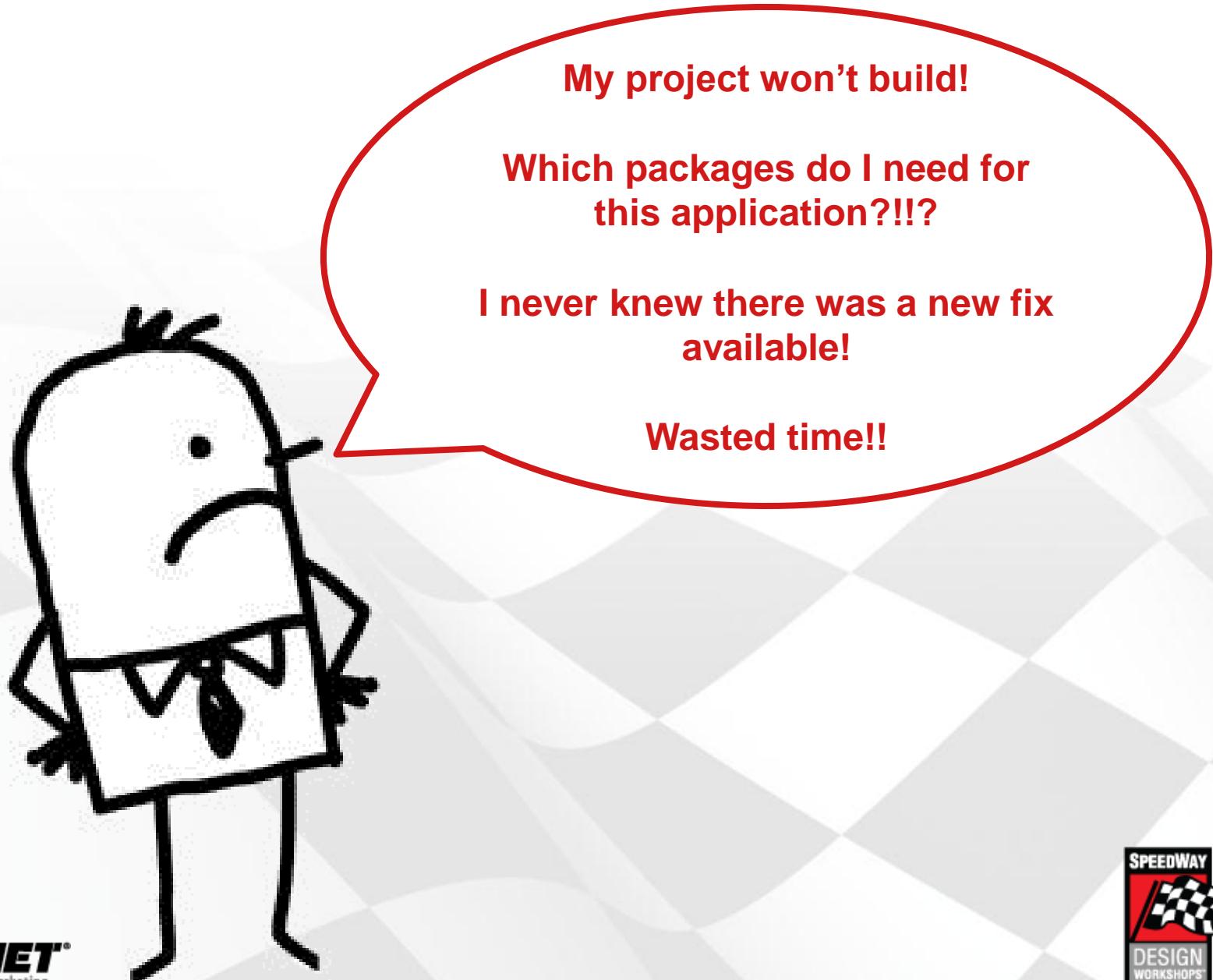
Implementing Linux on the Zynq™-7000 AP SoC

**Chapter 2.3 – Options for Building Root
File Systems**



Accelerating Your Success™

Timesys LinuxLink Automates BSP Creation



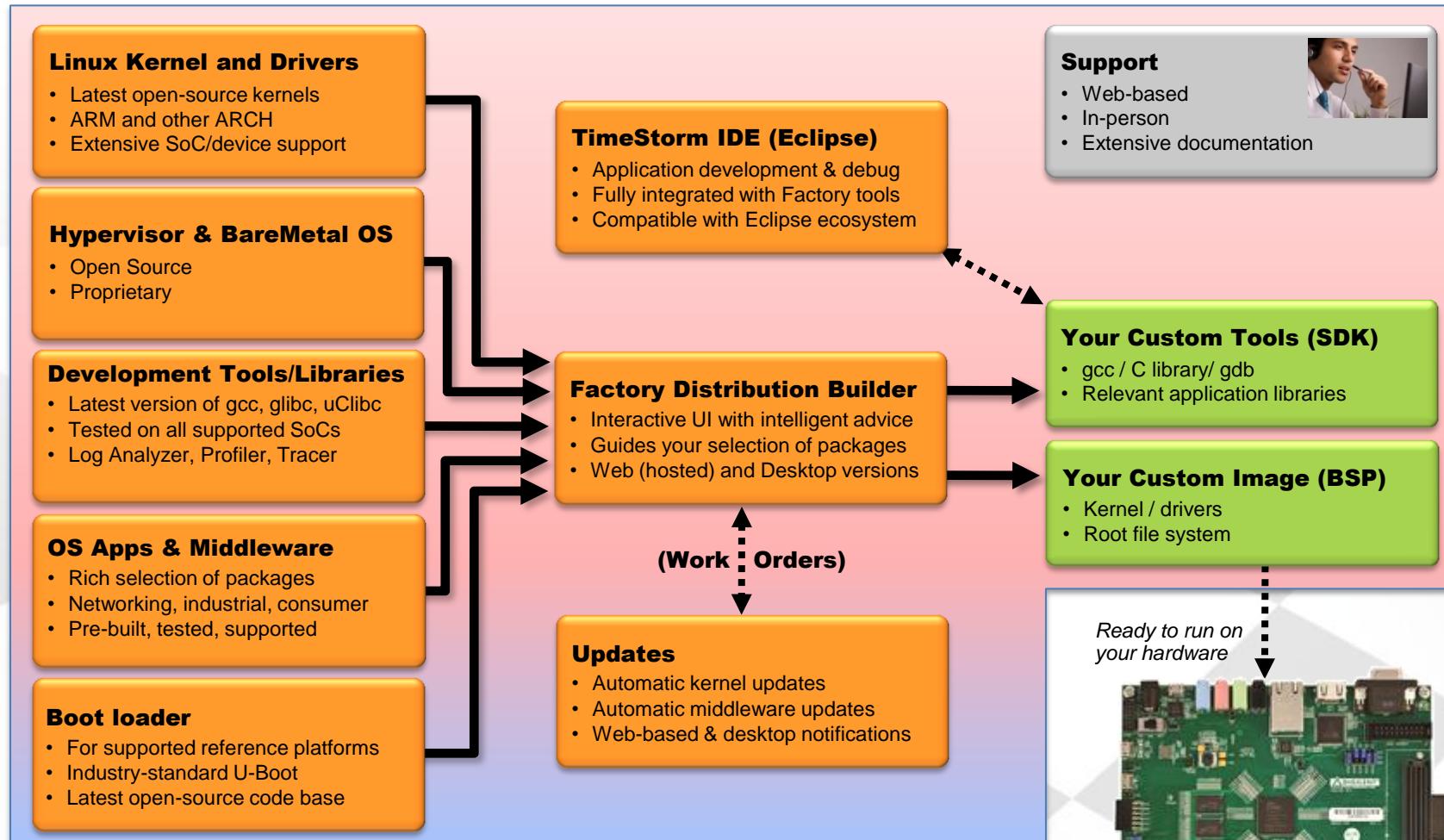
Timesys LinuxLink Automates BSP Creation

Design freedom through operating system and architecture choice

CHOOSE

BUILD

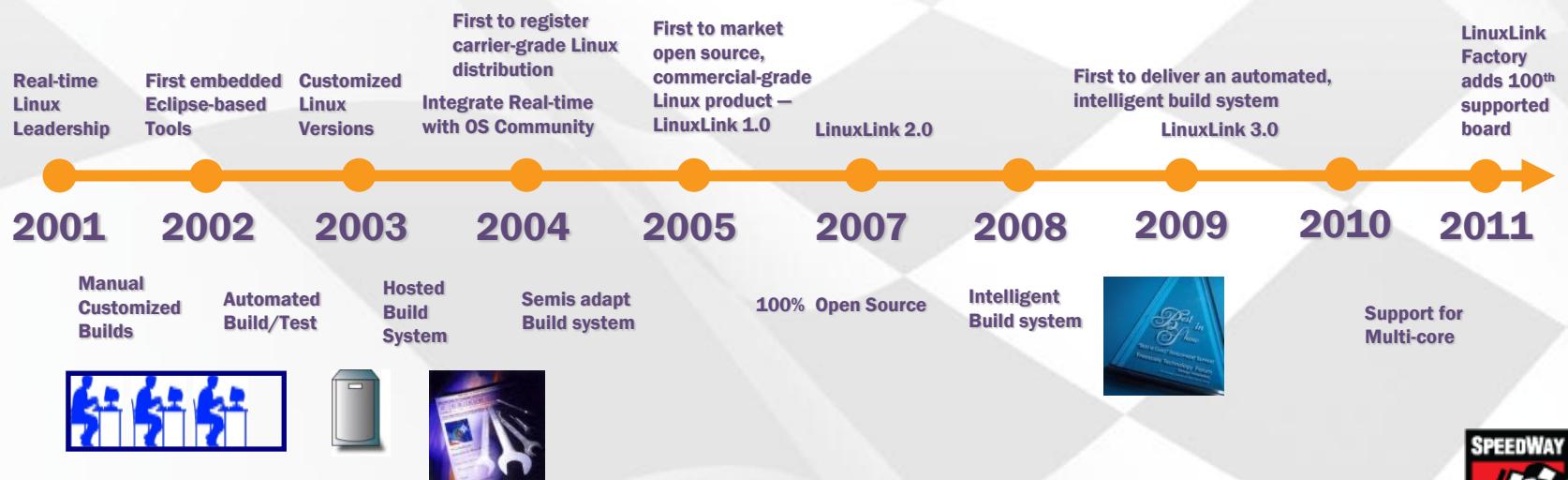
DEPLOY



LinuxLink Software Development Framework

Timesys: Linux Industry Pioneer

- Over 1000 projects, 200+ customers
- Applications for industrial automation, networking, mil/aero, medical, consumer and automotive segments
- Supports 8 architectures, 100+ processors/SoCs including Zynq
- Support for multi-core environments
- First to develop and deliver an award-winning, automated, intelligent, embedded Linux build system (LinuxLink 3.0)



LinuxLink Suite of Tools



Web Factory

- RFS Assembler
- Build Engine
- Advice Engine
- Update Engine
- BSP/SDK



Desktop Factory

- Toolchain builder
- Kernel Customization
- Busybox Customization
- Bootloader Customization
- Multicore Hypervisor Customization
- RFS Customization
- Build Engine
- Advice Engine
- Update Engine



Timestorm IDE

- Target Management Plugin
- Toolchain Management Plugin
- Factory plugin
 - Build Engine
 - Advice Engine
 - Update Engine
- Code Fragments
- Demos
- Qt plugin
- Community Tools
- LTT plugin
- Oprofile plugin
- Memory plugin
- RSE plugin
- Egit plugin

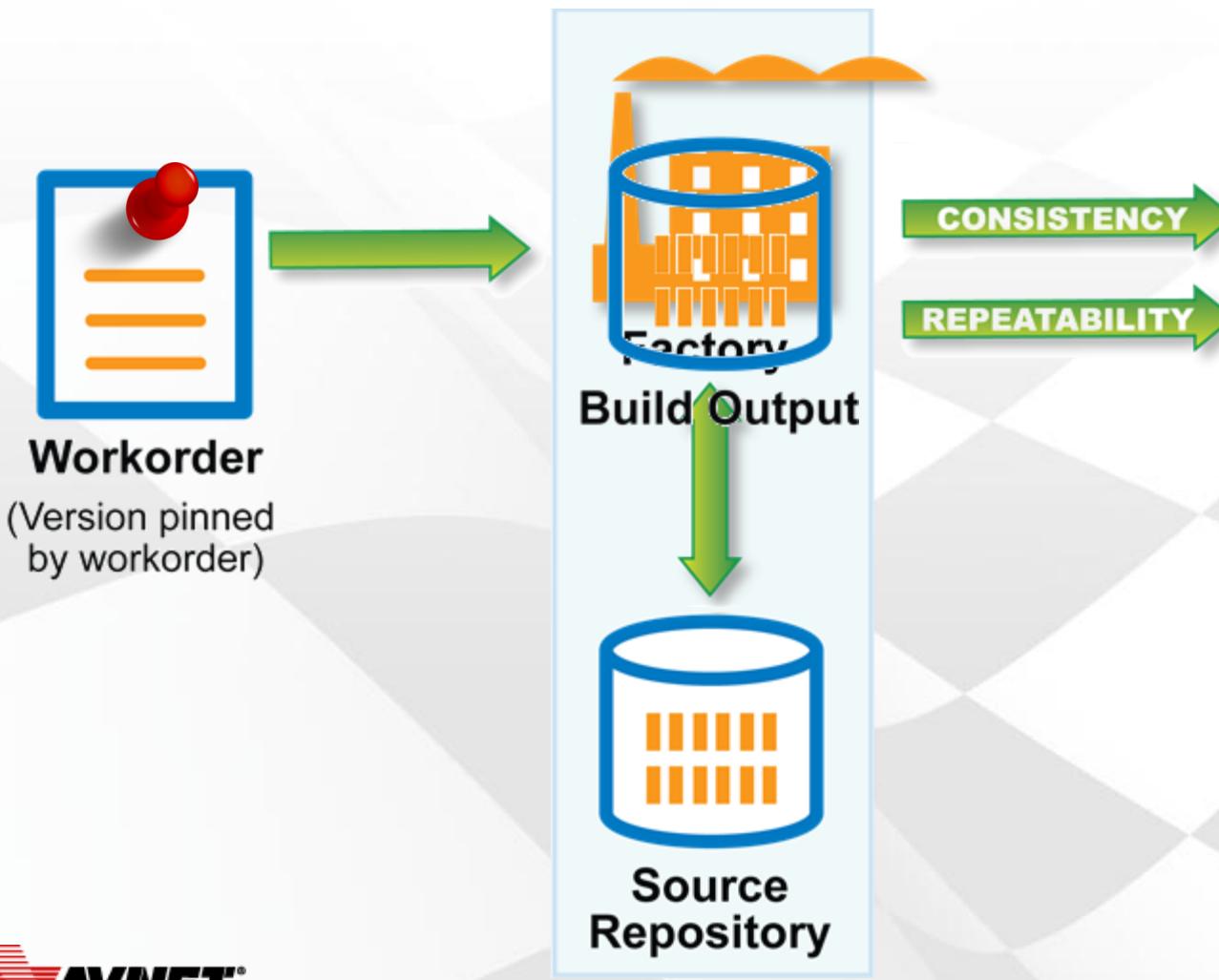
Command Line Tools



- Libraries & Packages
- Bootloaders
- Kernels
- Dev Tools
- Compliance - Publish OpenSource and build instructions

Timesys LinuxLink Factory Process

- Ensure build repeatability for certification and maintenance



LinuxLink Web Factory

Logout | My Account

Download BSP/SDK Build BSP/SDK Get IDE Docs Learning Center Support

Project Workorder Kernel Toolchain Packages Build Options Advice Summary

Which packages to select?
Just select the packages required for your board and application development. We will handle the package dependencies. If you are looking for a particular feature / package, use the search below to find the package.

Select Packages for:
Project: Coffee maker J1
Board: TI AM3517 EVM

Viewing search results for: ssh

Name	Version	License	Size (MB)	Build Options	Rationale
Desktop					
Development					
Graphics					
Multimedia					
Networking					
dropbear	2011.54-1	MIT	0.48		
libssh	0.5.2-1	LGPLv2.1	0.47		
libssh2	1.3.0-1	BSD	0.71		
openssh	5.9p1-1	BSD	1.83		
Runtimes					
System					
Utilities					

ssh
7 packages found
You Selected 2
Dependencies Selected 2
Total Selected 4
Size (MB) 18

Menu allows for selecting the packages needed for your BSP

Timesys LinuxLink Options

- **Free Edition allows for building BSPs against supported reference boards including ZedBoard**
- **Pro Edition offers capability of building for custom board**
 - Dedicated support engineers
 - 24-hour turn-around time
 - Unlimited number of tickets for build related issues
 - Every subscriber (seat) can submit their own tickets
- **Both Free and Pro Editions for ZedBoard available from the Avnet Embedded Software Store**

<http://www.embeddedsoftwarestore.com/timesyszedboard>

- An online information and e-Commerce based website
- Dedicated to the embedded design community
- The Embedded Software Store aims to accelerate software innovation for embedded applications by establishing an industry marketplace
- This will allow users to easily locate available software supporting the ARM architecture
- The site seeks to consolidate a large number of high value software options within a single site

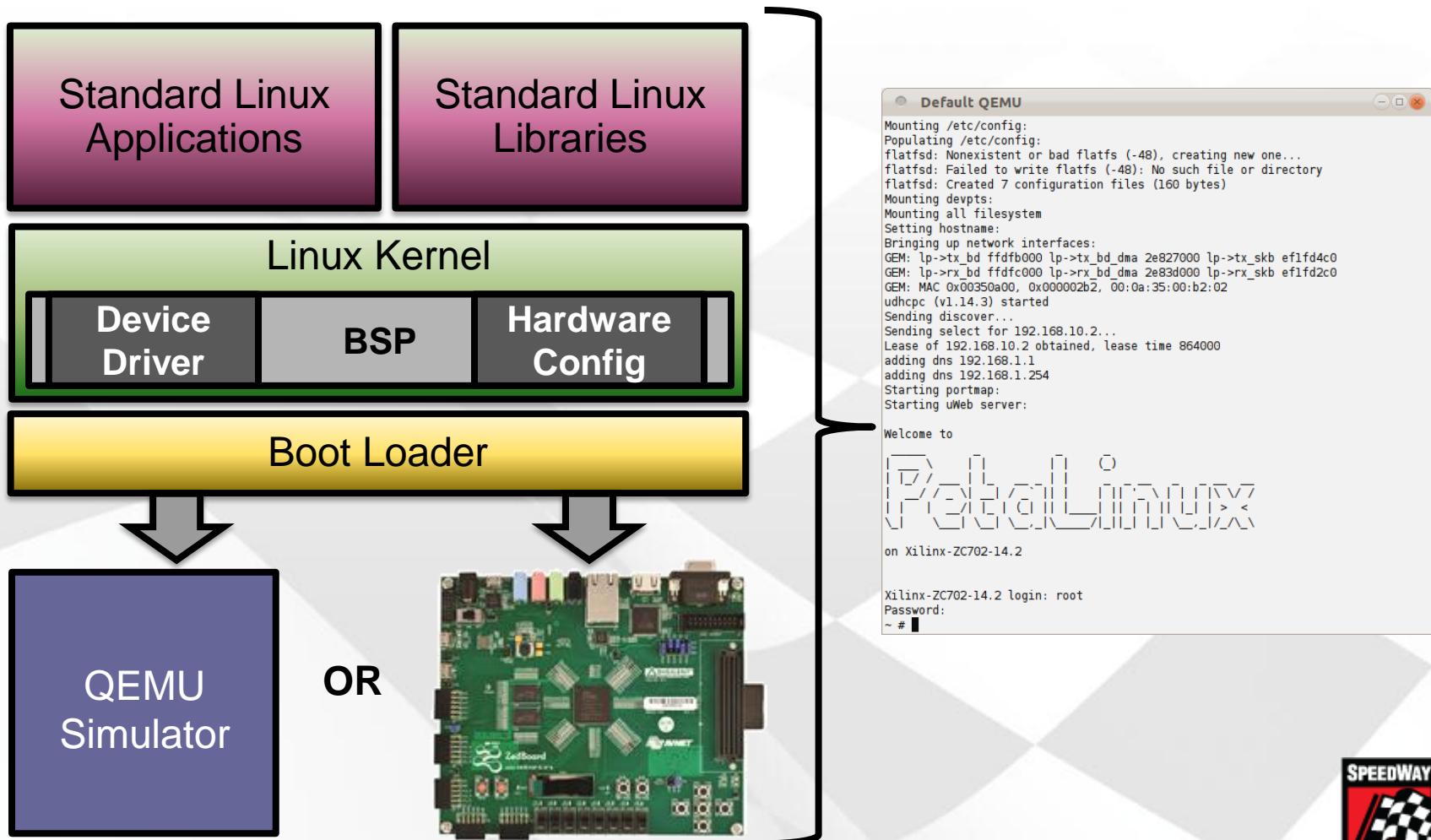
<http://www.embeddedsoftwarestore.com/>

Xilinx PetaLinux Overview

- **Board support package**
 - Boot loader
 - CPU-optimized kernel
 - Device drivers and driver configuration files
 - Drivers for many Xilinx embedded IP cores
 - Libraries
 - Thread and FPU
 - C & C++ application support
- **Development, Debug and Platform Management Utilities**
 - Trace & Debug agents and utilities
 - Integrated web server for easy network, firmware and configuration management

PetaLinux ZedBoard Support

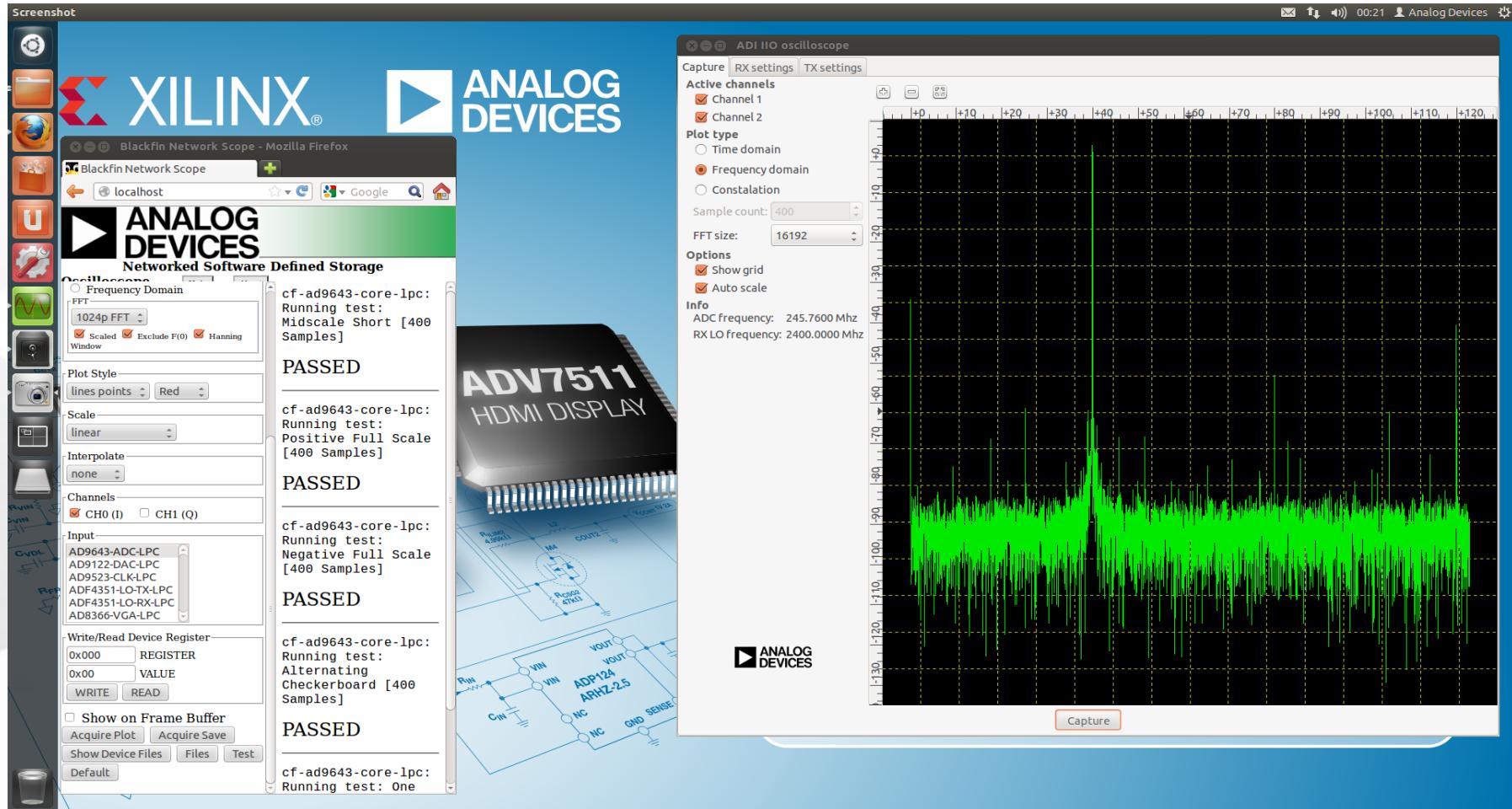
- Linux operating system for Xilinx devices



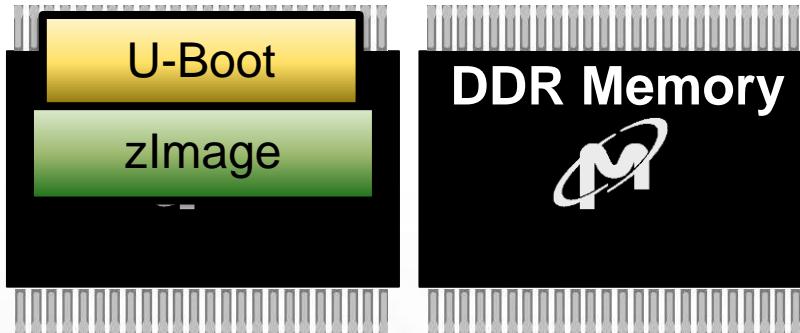
Try PetaLinux SDK Today

- **No charge 12 month evaluation license**
 - Download available on Xilinx site
<http://www.xilinx.com/download>
- **Out of the box validated and bootable software images**
 - Once installed and SDK running tools are ready for application, library, driver development within minutes
 - Boot on system simulator or physical hardware
 - Example platform Bitstream included

Ubuntu Linux Desktop

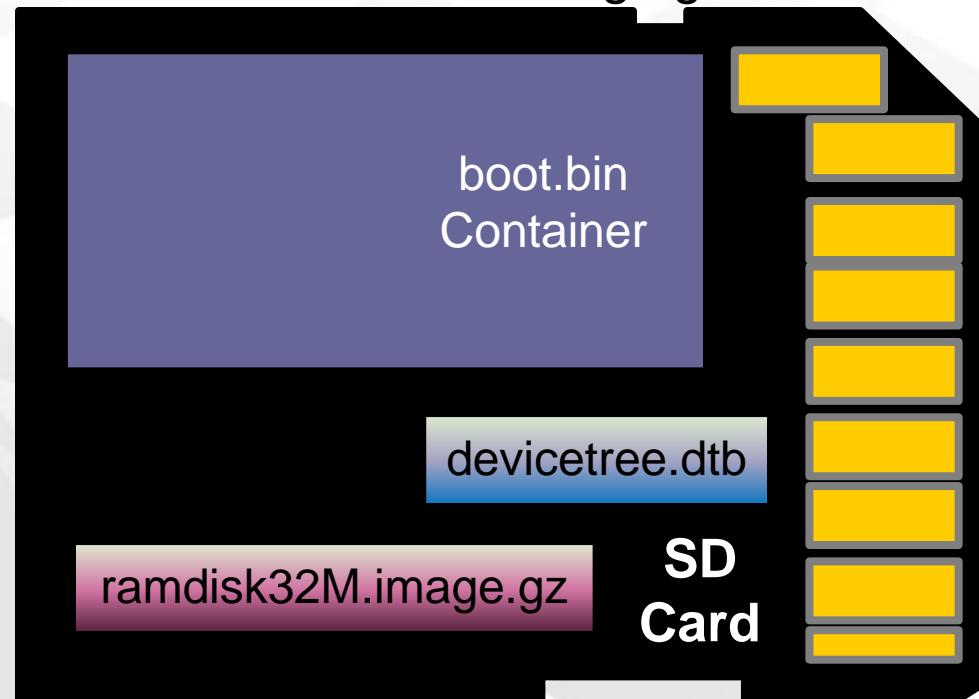
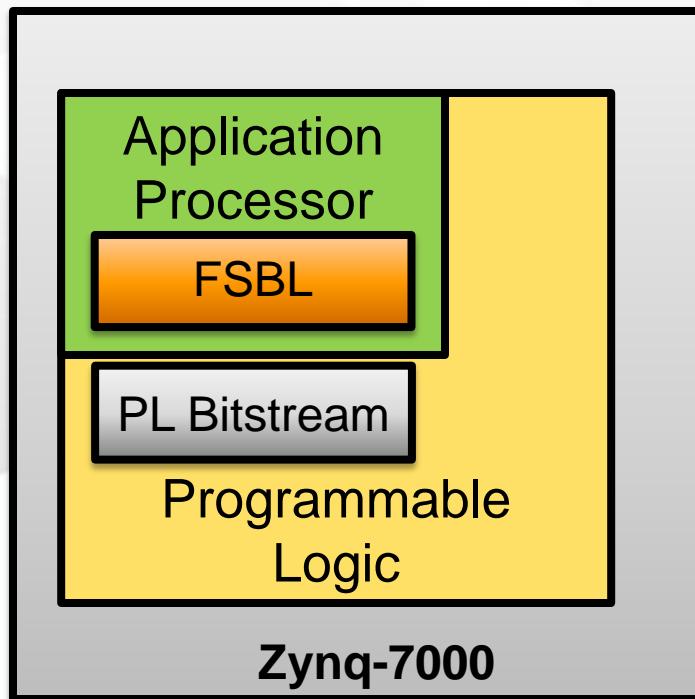


Booting Linux



- Copy these files to SD Card

- boot.bin
- devicetree.dtb
- zImage
- ramdisk32M.image.gz



Checkpoint – Pop Quiz

Which of the following provide a compelling alternative to creating a root file system by hand?

- 1) Linaro Ubuntu Linux Distribution**
- 2) Xilinx PetaLinux SDK**
- 3) Timesys LinuxLink**
- 4) All of the above**

Lab 2.3

- **Booting Linux**
 - Combine the root file system with the results of previous labs and boot ZedBoard to the Linux command prompt
- **Note: All SpeedWay lab materials, including lab instructions and solutions, can be downloaded from ZedBoard.org**

Implementing Linux on the Zynq™-7000 AP SoC

Chapter 3.1 – Linux Device Drivers

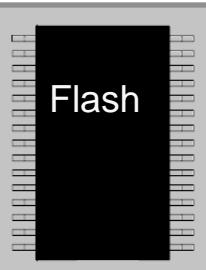


Accelerating Your Success™

Roadmap - Linux in Four Parts

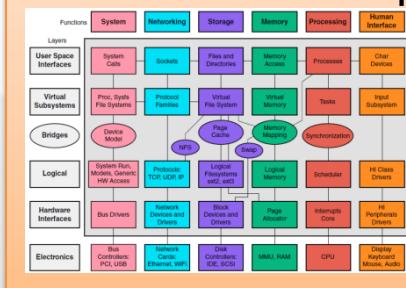
① Boot Loading – 3 stages

- Basic hardware initialization
- Loads Linux kernel
- Passing boot parameters



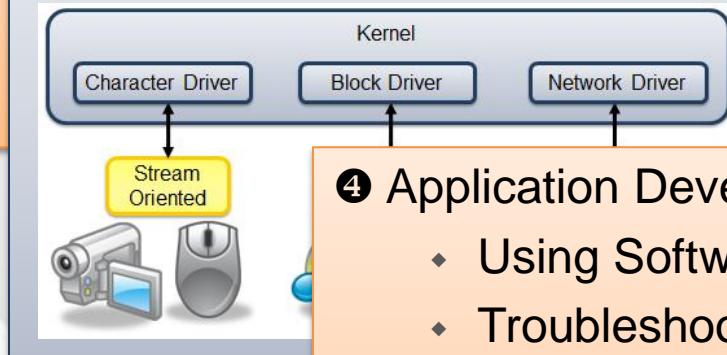
② Kernel

- Manages system resources
- Provides application services



③ Device Drivers

- Abstracts hardware details from software



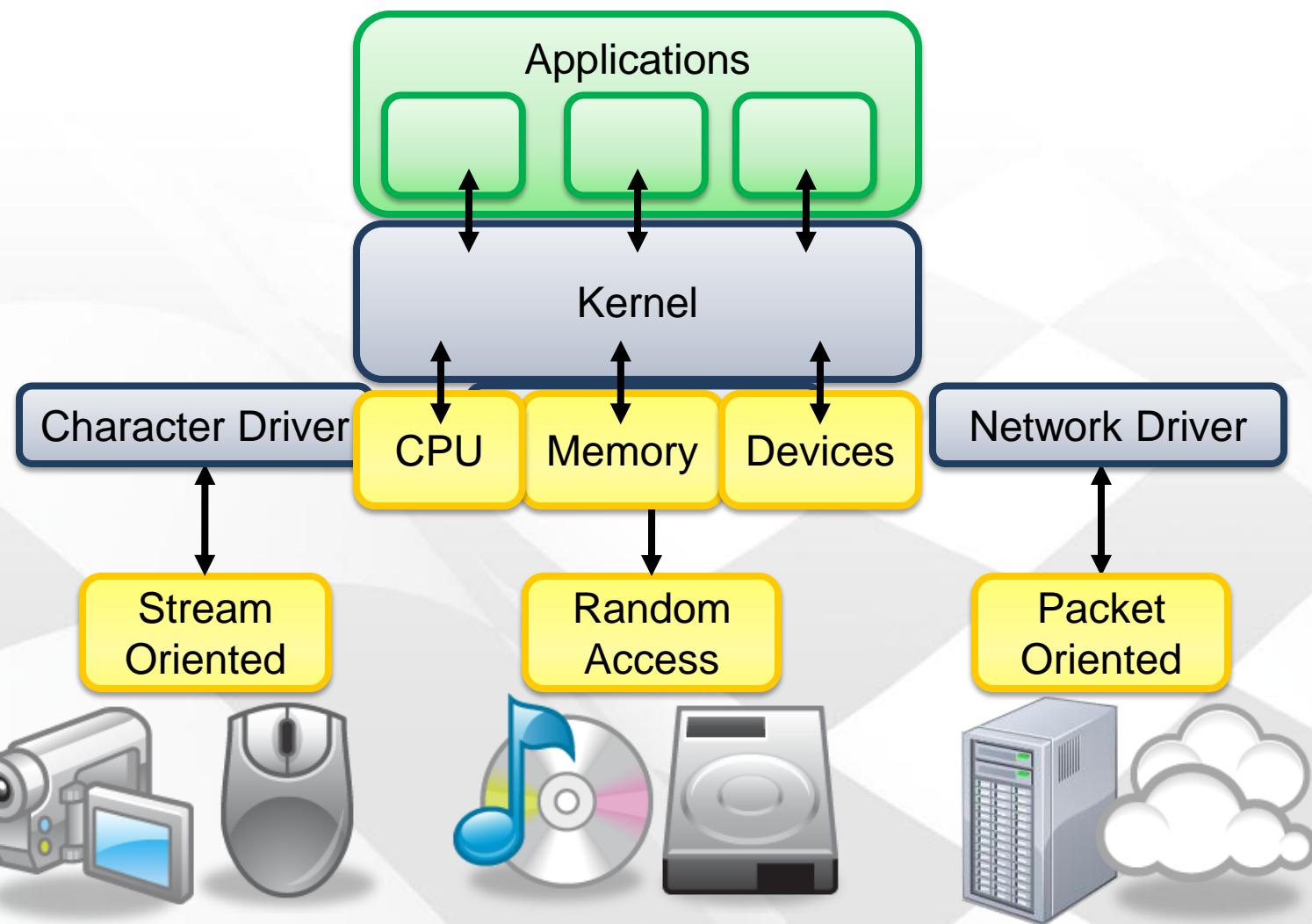
④ Application Development and Debug

- Using Software Development Kit
- Troubleshoot user applications

The terminal window shows the following boot log:

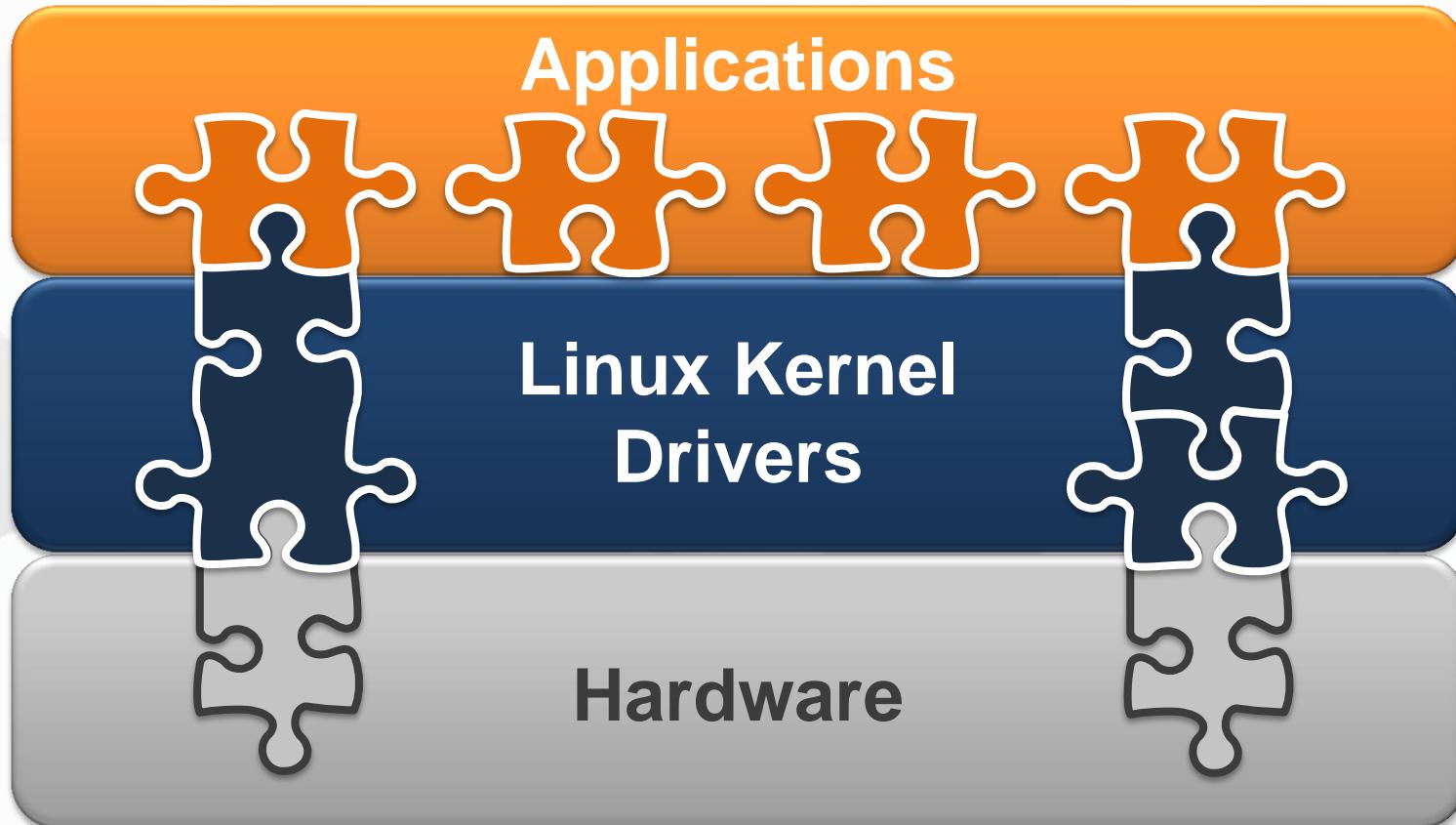
```
GEM: physdev defmap000, phydev->phy_id 0x1410dd1, phydev->addr 0x0
eth0: phy_id 0x0, phy_id 0x01410dd1
eth0: attach [Generic PHY] phy driver
IP-Config: Getting netmask 255.255.255.0
IP-Config: Getting broadcast 192.168.1.255
device: eth0, addr: 192.168.1.10, mask=255.255.255.0, gw=255.255.255.255,
host=192.168.1.10, domain=, nis-domain=none),
bootproto=255.255.255.254, rootserver=255.255.255.255, rootpath=
RAMDISK: gpt1: index 1, size 104857600 bytes
mmcblk0: new SD card at address e624
mmcblk1b: mmc0:e624 SU02G 1.84 GiB
mmcblk1b: mmcblk1b: mmcblk1b: mmcblk1b:
UPS: Mounted root (ext2 filesystem) on device 1:0.
devtcpfs: mounted
Freeing init memory: 144K
Starting rcu...
** Mounting filesystem
** Setting up imdev
** Starting httpd daemon
** Starting http daemon
** Starting ftp daemon
** Starting dropbear <ssh> daemon
rcs: complete
zyng>
```

Linux Device Driver Types

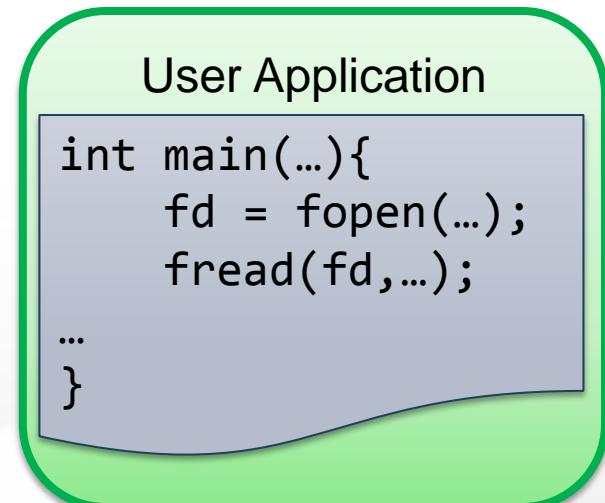


Linux Device Driver Modularity

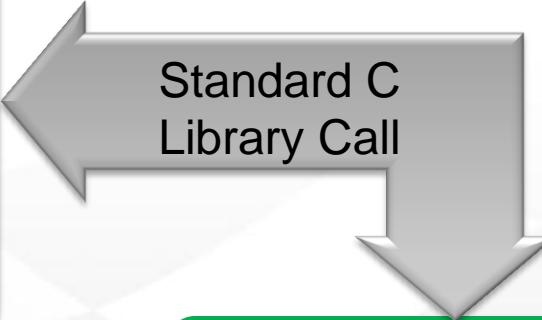
- Provide access to physical hardware resources
- Built-in or loaded at run-time (loadable modules)
- Can be multi-layered subsystems (USB, I2C, Ethernet)



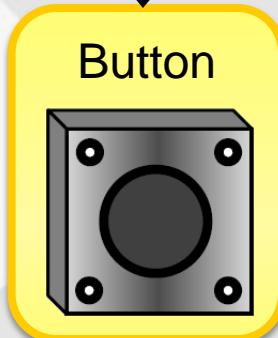
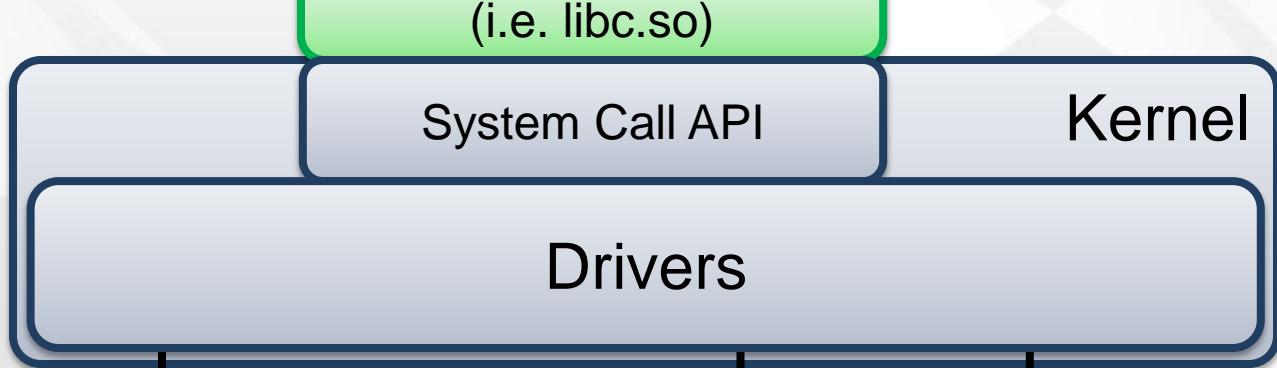
System Call API



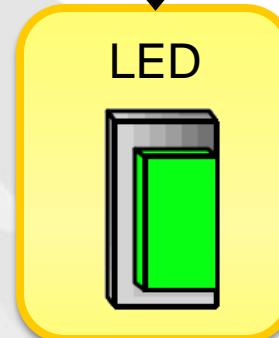
- System Call API is the primary way that applications interact with the kernel



- Example Library Functions:
 - `fopen()`
 - `fread()`
 - `fwrite()`
 - `fseek()`
 - `fclose()`



Physical
Devices



Traditional Device Drivers vs. sysfs

- **Traditional /dev devices**

- Handles streaming data (i.e. audio/video)
- Efficient exchange of binary data and structures rather than individual text strings
- Protection from simultaneous access

- **Device drivers under sysfs**

- Limited to simple single text value
- Easy access to device data via both shell scripts and user space programs



Weigh the tradeoffs to decide which solution is appropriate for your own application

System vs. Library Function Calls

Open /dev/led-brightness

Application calls write()

Kernel code

Call device driver write()
handler function

write() handler function

Logic to determine
correct action
(write register)

Write IO space register

Return number of bytes
written (numerical value)

Open /sys/class/gpio/gpioXX/value

Application calls fwrite()

Kernel code uses device
attribute to determine
function to call

Call sysfs device driver
write() handler function

sysfs write() handler function

Decode text string

Write MIO GPIO register

Return write status

Best for large data transactions

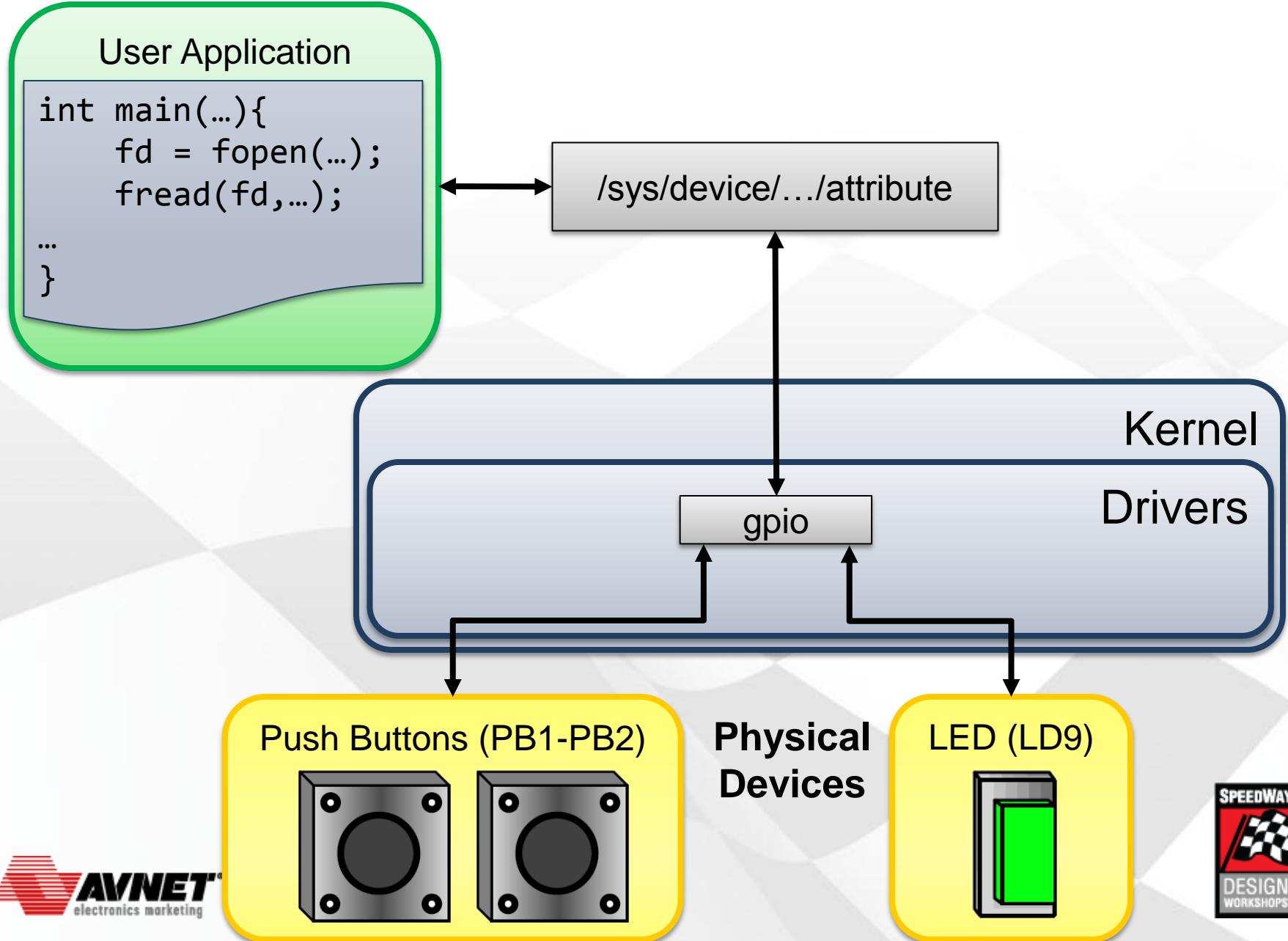
Effective for single
control/status transactions

Details on sysfs

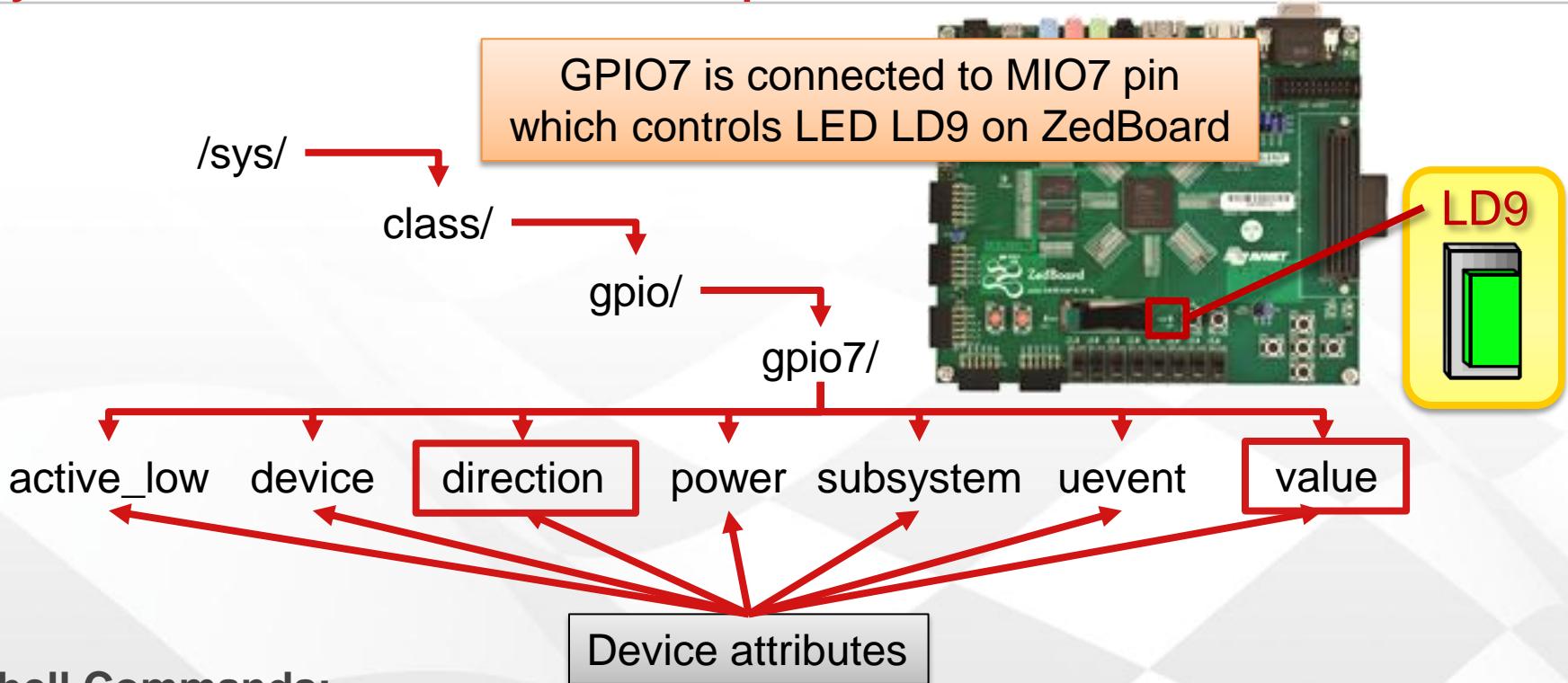
- The sysfs subsystem is a dynamic “virtual” file system
- Mounted on /sys
- Provides a representation of the device model which is visible from user space
 - Device attributes visible in the form of regular files
 - Allows get/set capability of specific parameters from user space

```
COM13:115200baud - Tera Term VT
File Edit Setup Control Window Help
/ # ls -l /sys/
total 0
drwxr-xr-x  19 root      0          0 Jan  1 00:00 block
drwxr-xr-x  16 root      0          0 Jan  1 00:00 bus
drwxr-xr-x  29 root      0          0 Jan  1 00:00 class
drwxr-xr-x   4 root      0          0 Jan  1 04:20 dev
drwxr-xr-x   9 root      0          0 Jan  1 00:00 devices
drwxr-xr-x   2 root      0          0 Jan  1 04:20 firmware
drwxr-xr-x   2 root      0          0 Jan  1 04:20 fs
drwxr-xr-x   3 root      0          0 Jan  1 04:20 kernel
drwxr-xr-x  45 root      0          0 Jan  1 04:20 module
drwxr-xr-x   2 root      0          0 Jan  1 04:20 power
/ #
```

Example Device File Mapping



Sysfs Device Driver Example



Shell Commands:

```
/sys/class/gpio/gpio7 # echo 1 > value  
/sys/class/gpio/gpio7 # cat value
```

Very convenient for configuring and
controlling devices using shell scripts

C code:

```
fprintf(file_led7, "%d", 1);           /* write */  
fscanf(file_led7, "%d", &n_ch);          /* read */
```

See kernel document **Documents/gpio.txt** in kernel source tree for further details

Checkpoint – Pop Quiz

Which device attributes are we most concerned with when using sysfs to access GPIO pins?

1) **direction**

2) **drive_strength**

3) **value**

4) **pullup**

Lab 3.1

- **Device drivers and MIO GPIO hardware**
 - Explore Sysfs and interact with ZedBoard hardware devices
- **Note:** All SpeedWay lab materials, including lab instructions and solutions, can be downloaded from ZedBoard.org

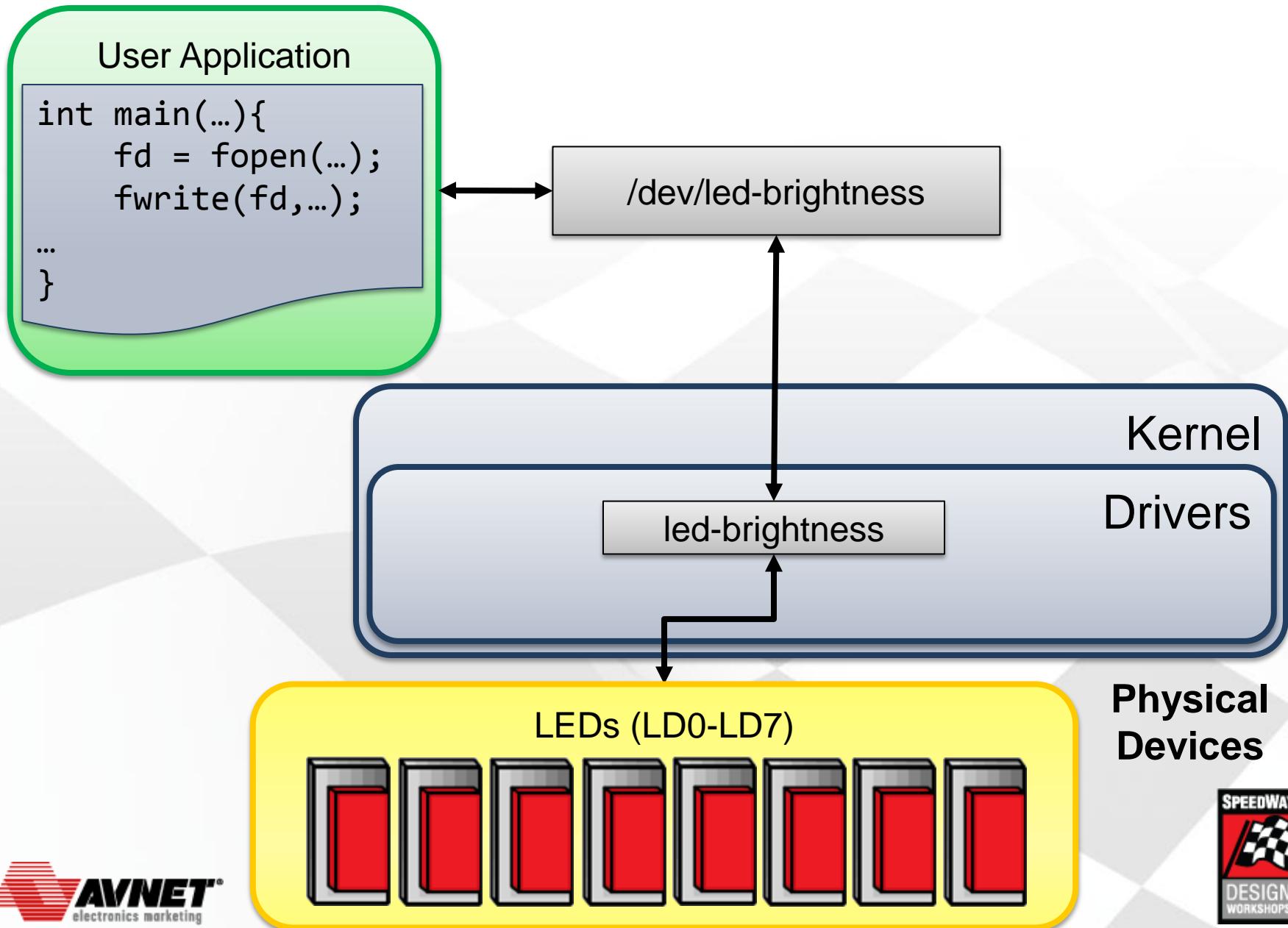
Implementing Linux on the Zynq™-7000 AP SoC

**Chapter 3.2 – Device Drivers and the
Device Tree**



Accelerating Your Success™

Example Custom Device Driver



Linux Kernel Patching

- Patches are applied to code bases to duplicate the desired modifications
- Patches are applied using:
 - patch – a utility for applying a .patch file contents to code files or source directory trees

```
$ patch -p1 < add_lcd_panel_support.patch
```

- Software version control systems such as git, svn, and cvs use internal apply mechanisms

```
$ git apply add_lcd_panel_support.patch
```

Patch Creation

- **Vast majority of modifications to open source code bases are patch files**
- **Patch files are the difference between the original and the modified code**
- **Patches are generated using:**
 - diff – a utility for analyzing the differences between two code files or source directory trees

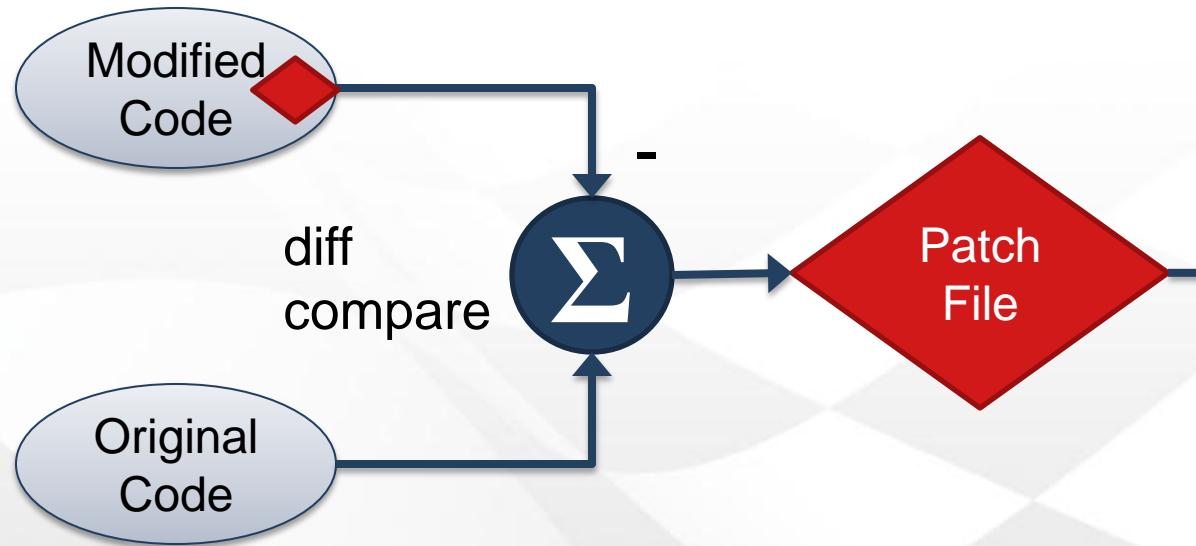
```
$ diff -u original.c new.c > add_lcd_panel_support.patch
```

- Software version control systems such as git, svn, and cvs
(Git is used by U-Boot and Linux kernel developers)

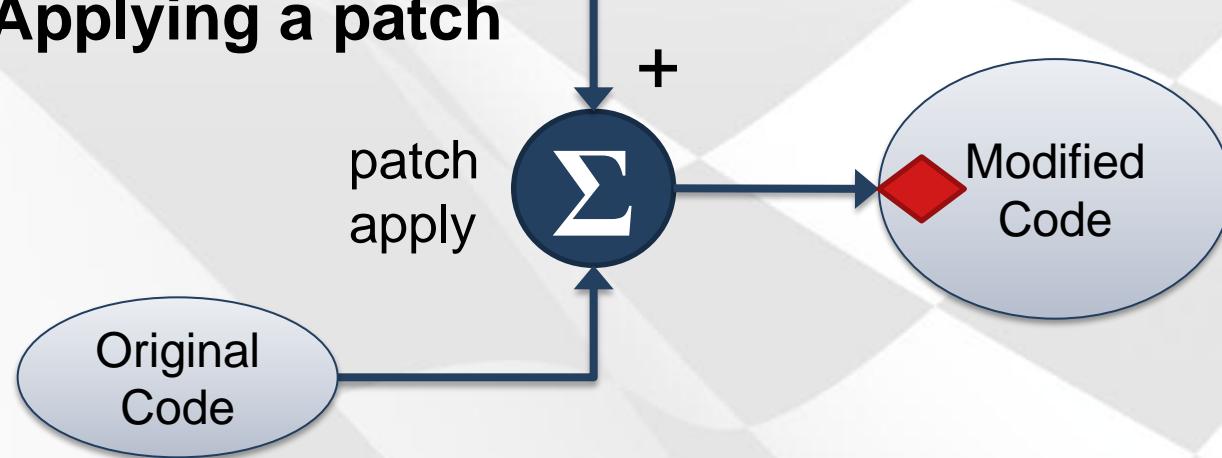
```
$ git format-patch -1
```

Patch Lifecycle

Creating a patch



Applying a patch



Kernel Device Tree

- Device Tree data structure describes the system hardware
- Data structure passed to the kernel at boot time
- In general, device nodes for IP cores take the following form:

```
(instance-name) : (generic-name) @ (base-address) {  
    compatible = "xlnx,(ip-core-name)-(HW_VER)"  
                [, (list of compatible devices), ...];  
    reg = <(baseaddr) (size)>;  
    interrupts = < ... >;  
    interrupt-parent = <&interrupt-controller-phandle>;  
    xlnx,(parameter1) = "(string-value)";  
    xlnx,(parameter2) = <(int-value)>;  
};
```

- The device tree structure reflects the AXI attachments made in the system.mhs file in XPS tool

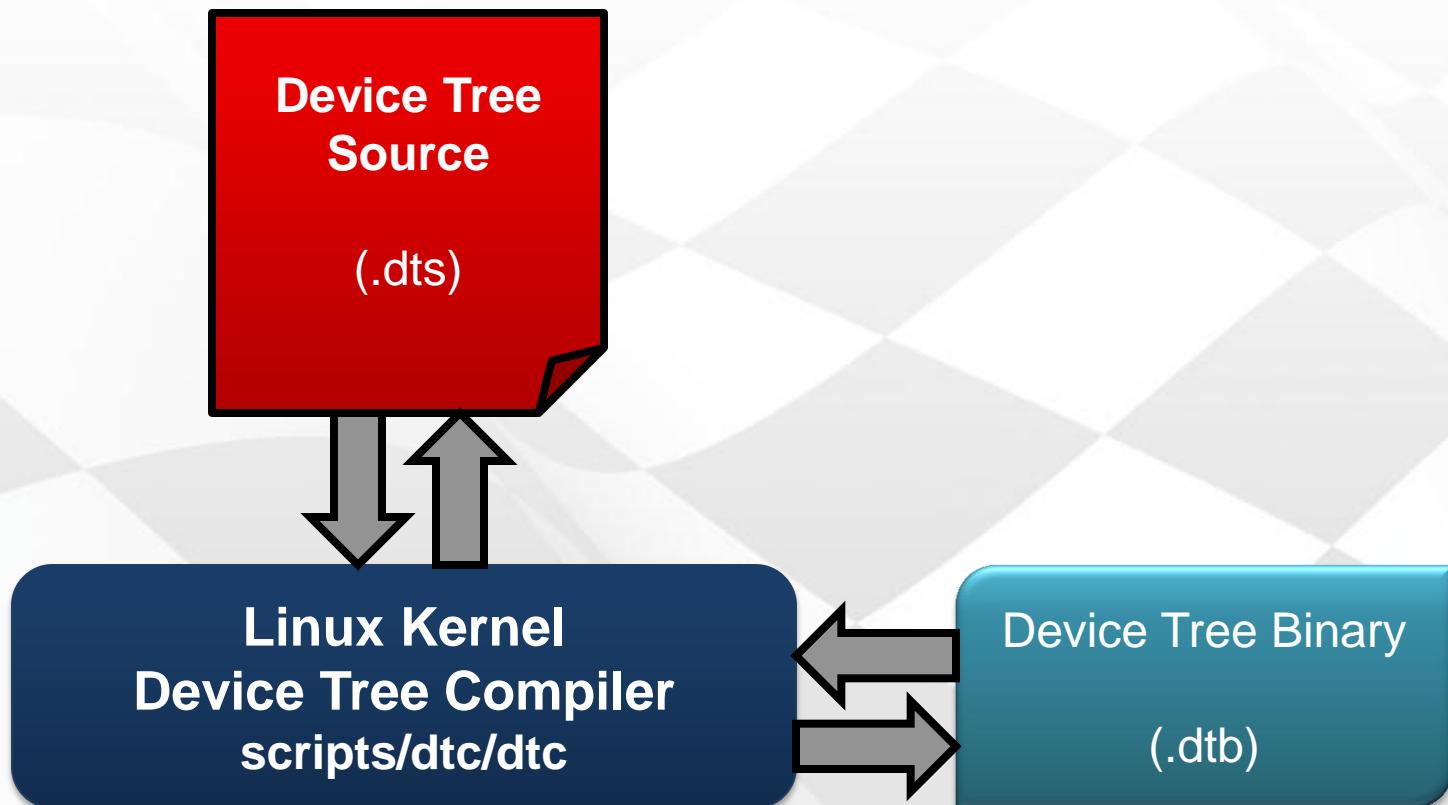
Device Tree Example

- Device trees are created in a human readable, text-based source format known as a Device Tree Source (DTS) file

```
memory {  
    device_type = "memory";  
    reg = <0x0 0x20000000>;  
};  
  
chosen {  
    bootargs = "console=ttyPS0,115200 root=/dev/ram rw initrd=0x2000000,32M  
    linux,stdout-path = "/amba@0/uart@E0001000";  
};  
  
amba@0 {  
    compatible = "simple-bus";  
    #address-cells = <0x1>;  
    #size-cells = <0x1>;  
    ranges;  
  
    led-brightness@41200000 {  
        compatible = "avnet,led-brightness";  
        reg = <0x41200000 0x20>;  
    };  
};
```

Device Tree Compiler

- DTS files are compiled with a Device Tree Compiler (DTC) to create a machine readable binary that is passed to the kernel
- The resulting binary is known as a Device Tree Binary (DTB)



Checkpoint – Pop Quiz

Which mechanism is used to inform the kernel of the hardware resources available on a platform?

- 1) driver list
- 2) device tree
- 3) device registry
- 4) driver registry

Lab 3.2

- **Device drivers for PL devices**
 - Edit device tree source and create the device tree binary file
 - Using a test bench, interact with custom logic peripherals across the AXI interconnect
- **Note: All SpeedWay lab materials, including lab instructions and solutions, can be downloaded from ZedBoard.org**

Implementing Linux on the Zynq™-7000 AP SoC

**Chapter 4.1 – Linux Application
Development**

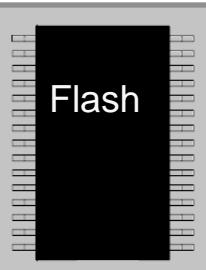


Accelerating Your Success™

Roadmap - Linux in Four Parts

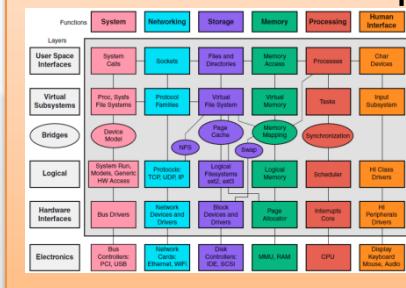
① Boot Loading – 3 stages

- Basic hardware initialization
- Loads Linux kernel
- Passing boot parameters



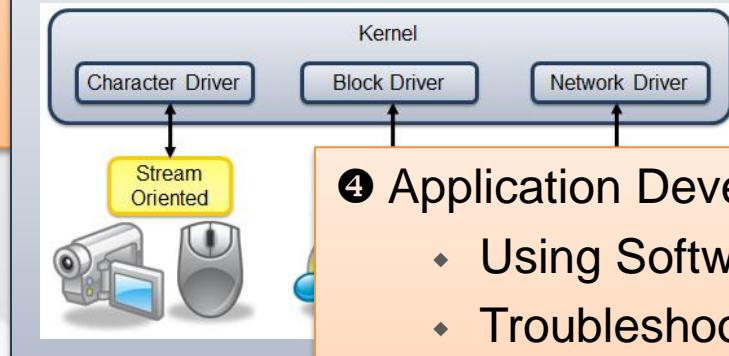
② Kernel

- Manages system resources
- Provides application services



③ Device Drivers

- Abstracts hardware details from software



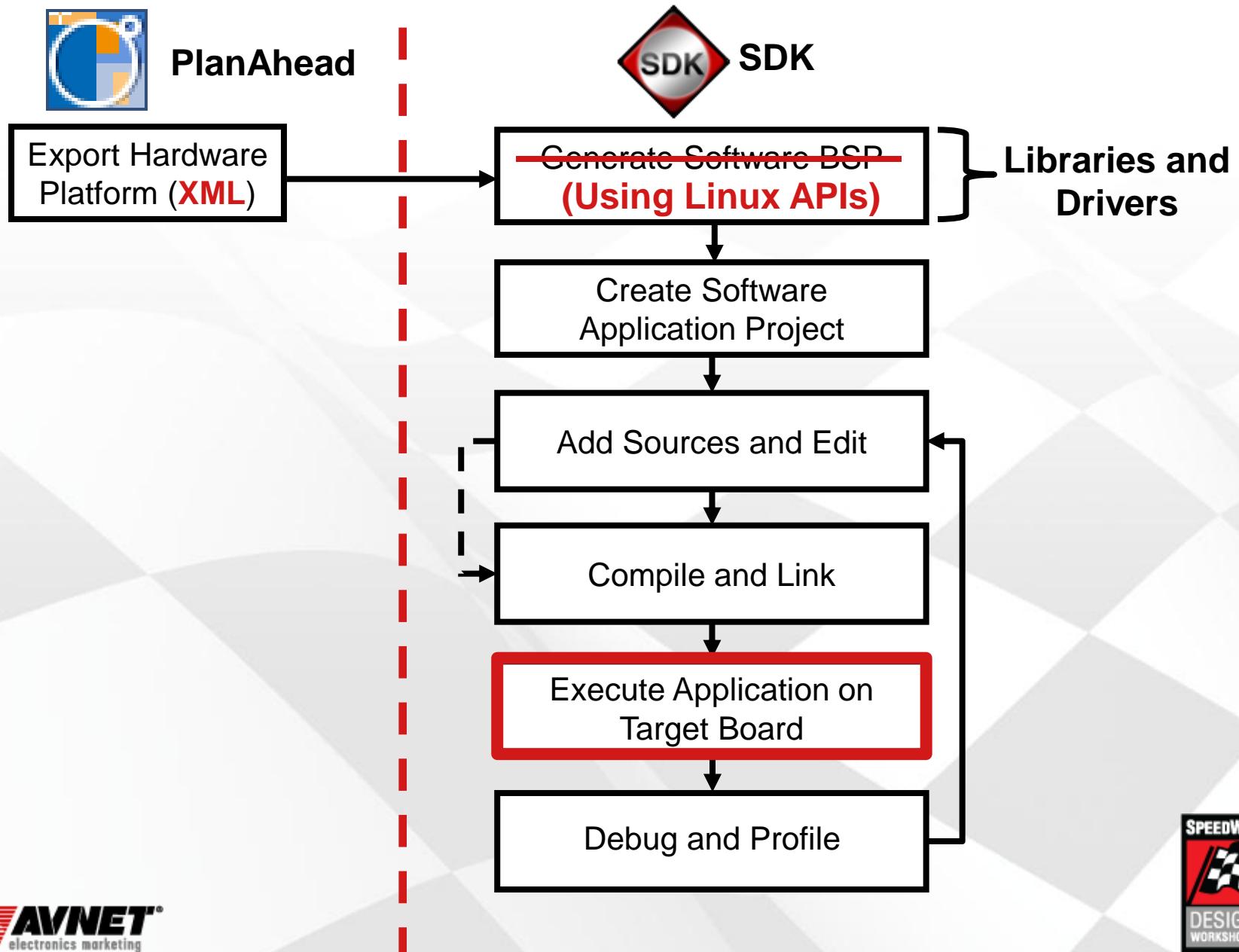
④ Application Development and Debug

- Using Software Development Kit
- Troubleshoot user applications

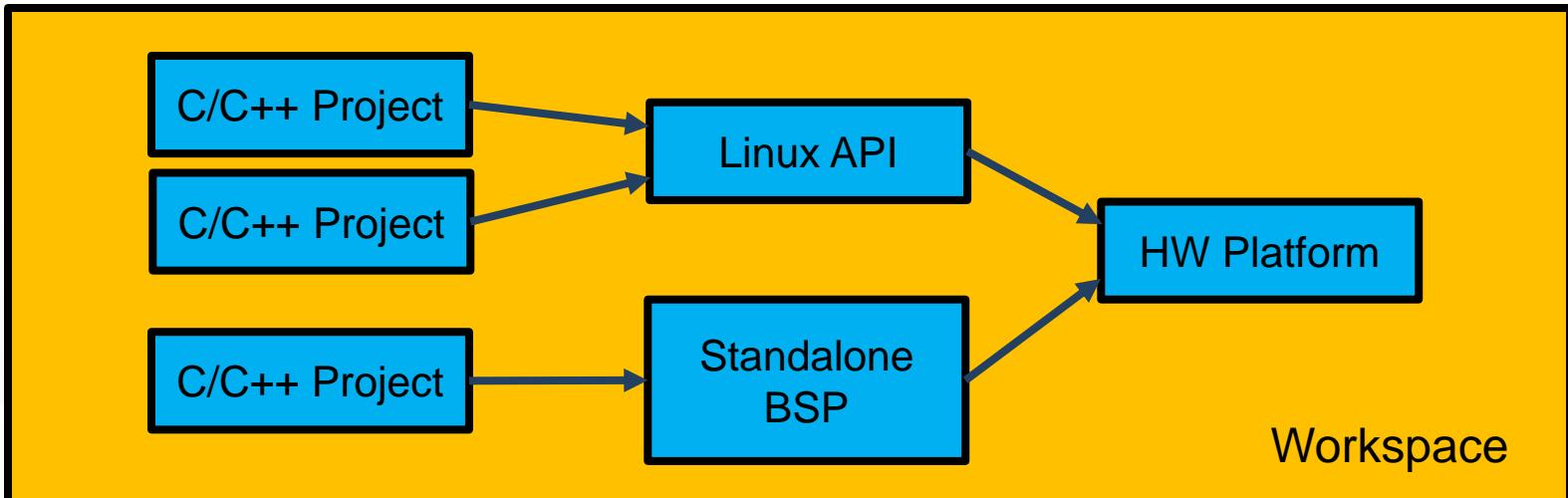
The terminal window shows the following boot log:

```
GEM: physdev defmap000, phydev->phy_id 0x1410dd1, phydev->addr 0x0
eth0: phy_id 0x0, phy_id 0x01410dd1
eth0: attach [Generic PHY] phy driver
IP-Config: Getting netmask 255.255.255.0
IP-Config: Getting broadcast 192.168.1.255
device: eth0, addr: 192.168.1.10, mask=255.255.255.0, gw=255.255.255.255,
host=192.168.1.10, domain=
MAC address: 255.255.255.255 (none)
rootserver=255.255.255.255, rootpath=
RAMDISK: gsize: 1024000, max_size: 1024000, block 0
mmcblk0: new SD card at address e624
mmcblk1b: mmc0:e624 SU02G 1.84 GiB
mmcblk1b: mmcblk1b: mmcblk1b: mmcblk1b:
UPS: Mounted root (ext2 filesystem) on device 1:0.
devtcpfs: mounted
Freeing init memory: 144K
Starting rcu...
** Mounting filesystem
** Setting up imdev
** Starting httpd daemon
** Starting http daemon
** Starting ftp daemon
** Starting dropbear <ssh> daemon
rcs: complete
zyng>
```

Linux Application Development Flow



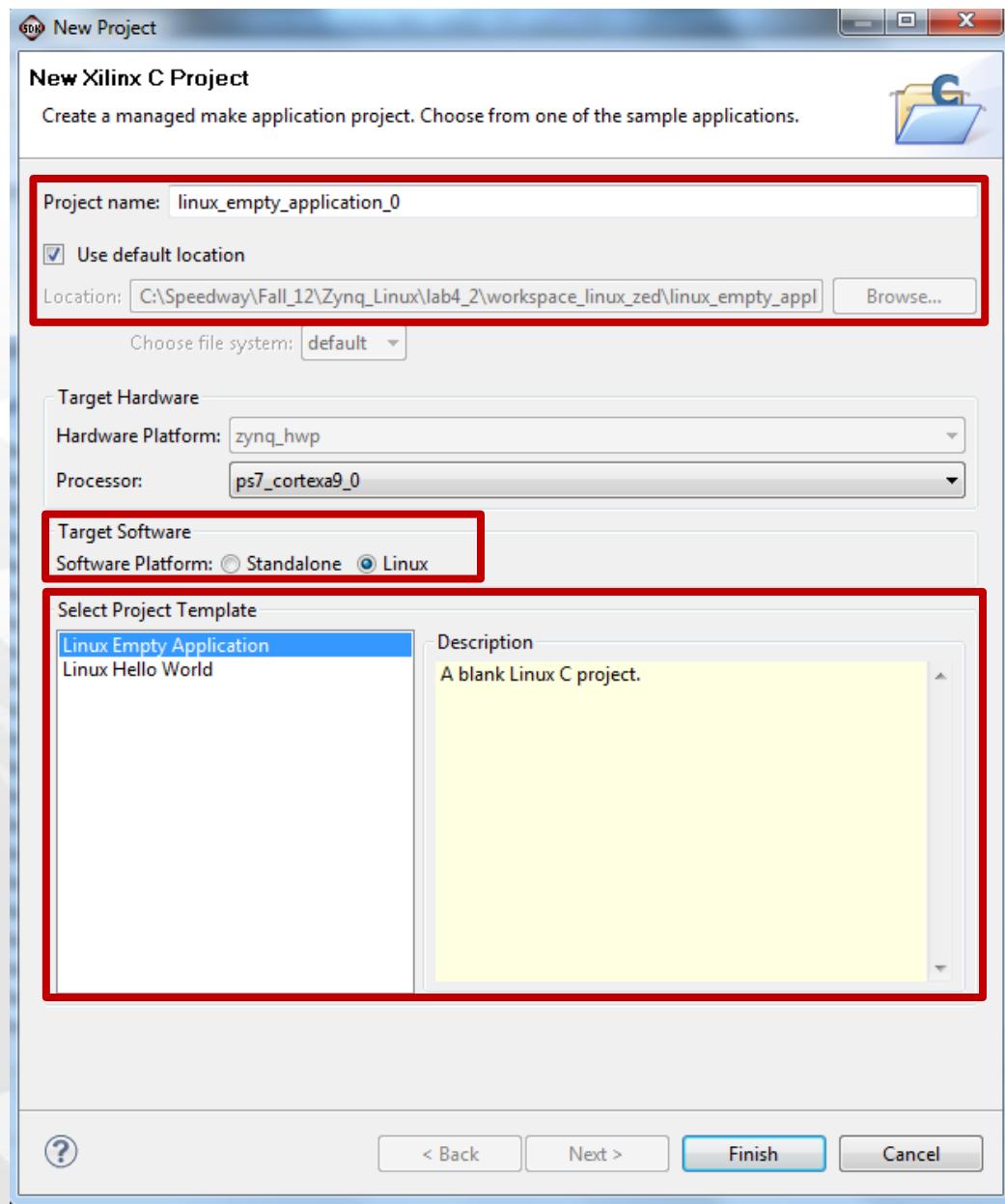
Application Dependencies



- **Every workspace has one Hardware Platform**
 - Specified through an XML file
 - XML file can be generated in PlanAhead
- **Multiple Board Support Packages (BSPs) can reference a single Hardware Platform**
 - Xilinx provides a Standalone BSP and a Linux API
- **Multiple C/C++ projects can reference a single BSP**

Xilinx C Project Wizard

- Target applications to a Linux Software Platform



Writing Code

- **The Eclipse Editor has many useful features**

- Remote System Explorer (Linux Target Only)
- Code completion
- C/C++ content assist
- Hovering
- Code folding
- Refactoring
- Highlighting inactive code blocks
- Quick keys for most actions

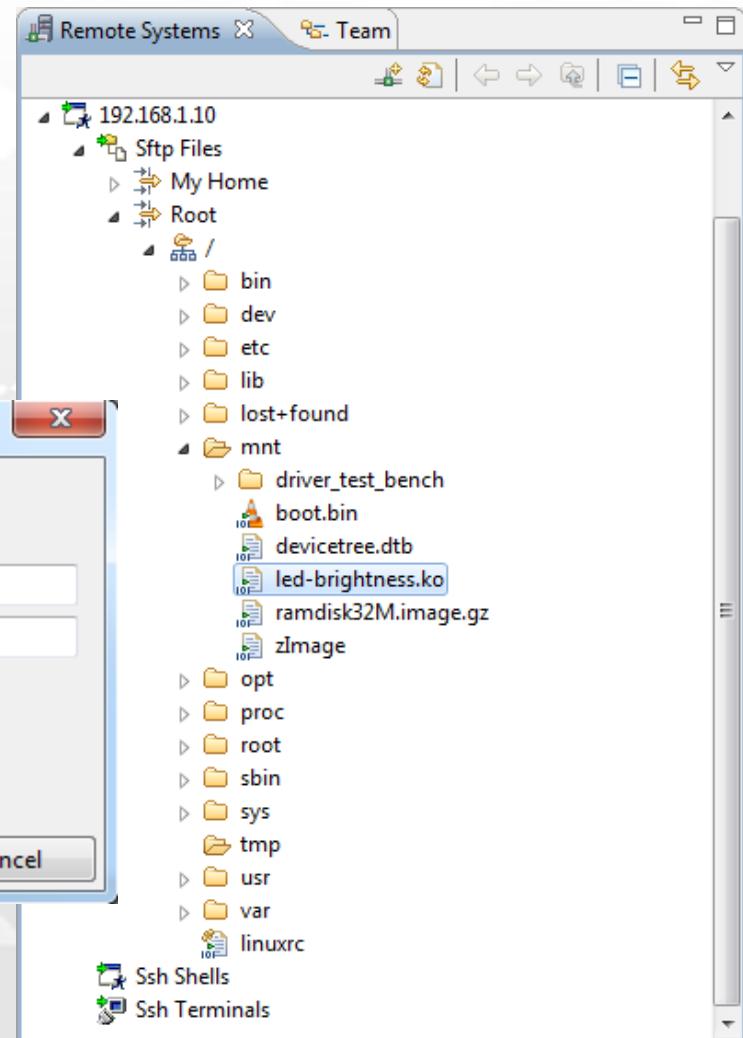
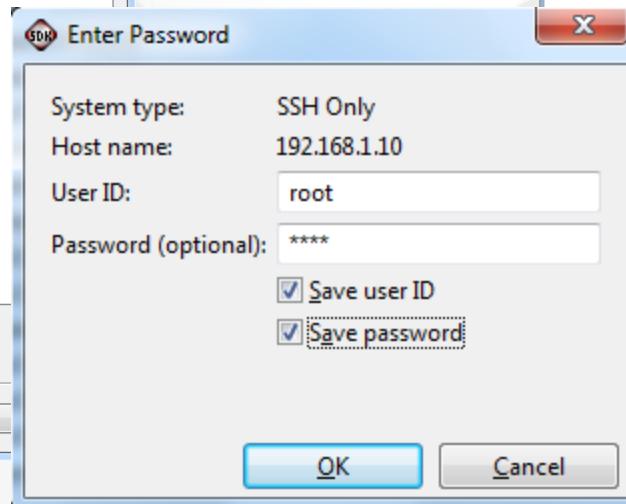
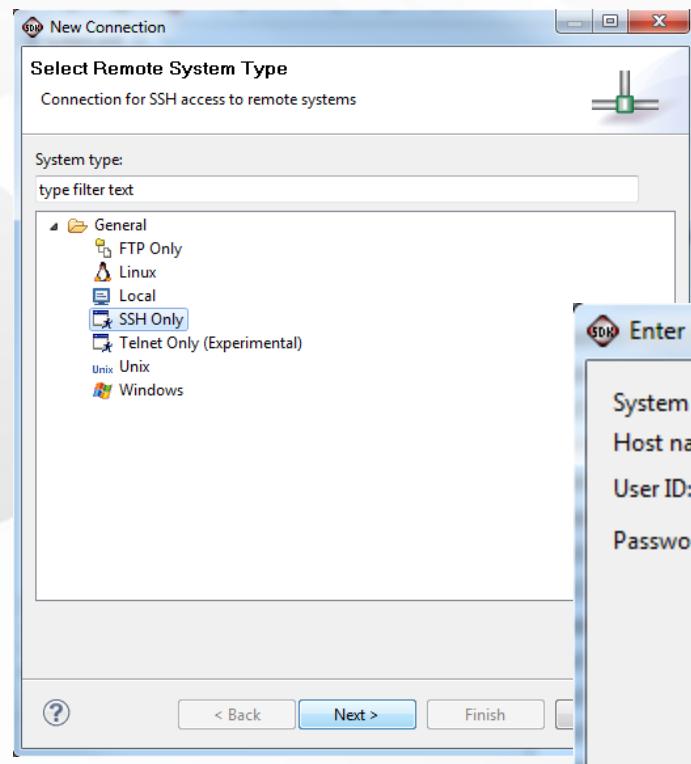
- **The Editor can be customized in the Eclipse preferences**

- Numerous options to help facilitate a particular coding style

Remote System Explorer

- **Secure remote network connection to target**

- Transfer files between PC and target Zynq file system
- Explore target root file system



Checkpoint – Pop Quiz

Which file is exported from PlanAhead into SDK to specify the hardware platform configuration?

- 1) system.c
- 2) system.h
- 3) system.xml
- 4) system.bin

Lab 4.1

- **Using SDK for Linux application development**
 - Create an application which interacts with ZedBoard hardware
 - Utilize both push buttons using PS GPIO via sysfs driver
 - Utilize custom logic to control LEDs using Programmable Logic via custom device driver
- **Note: All SpeedWay lab materials, including lab instructions and solutions, can be downloaded from ZedBoard.org**

Implementing Linux on the Zynq™-7000 AP SoC

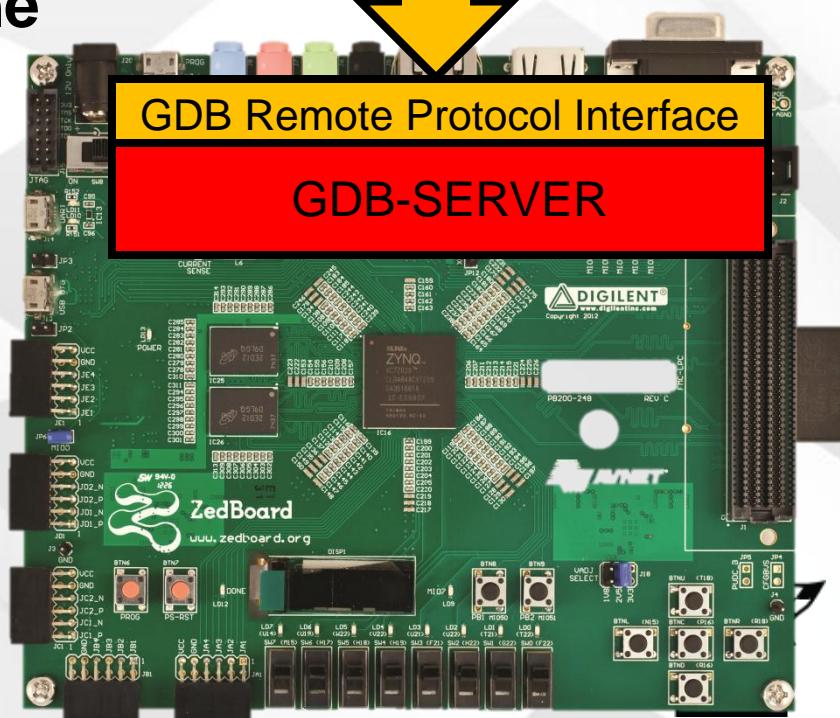
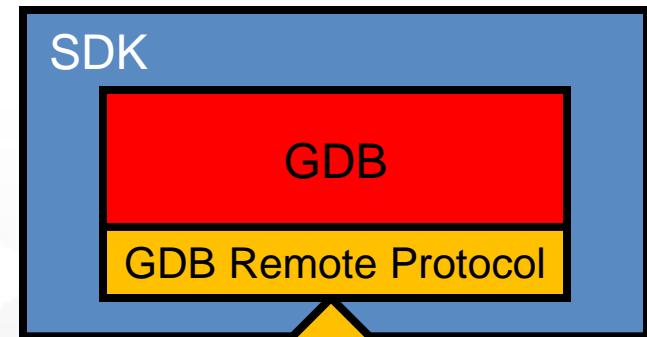
Chapter 4.2 – Linux Application Debug



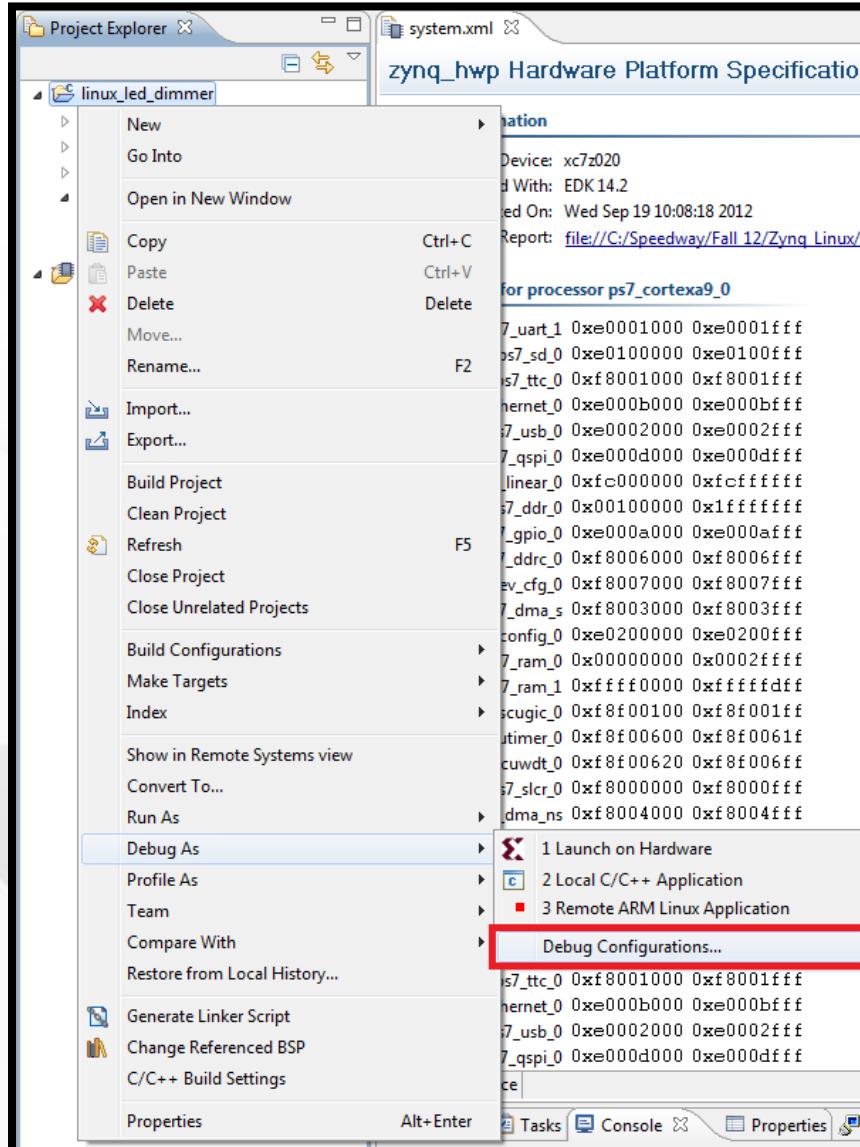
Accelerating Your Success™

Remote Application Debugging

- GDB is a source and assembly-level debugger
- Communicates with GDB-SERVER
- GDB is tightly integrated into the Eclipse Debug Perspective



Creating A New Debug Configuration



Debug Configuration Detail

SDK Debug Configurations

Create, manage, and run configurations

type filter text

- C/C++ Application
- C/C++ Attach to Application
- C/C++ Postmortem Debugger
- C/C++ Remote Application
- Launch Group
- Remote ARM Linux Application
 - linux_led_dimmer Debug
- Xilinx C/C++ ELF

Name: linux_led_dimmer Debug

Main (x= Arguments Common)

Connection: 192.168.1.10 New... Properties...

Project: linux_led_dimmer Browse...

Build configuration: Debug

C/C++ Application:

Debug\linux_led_dimmer.elf Search Project... Browse...

Remote Absolute File Path for C/C++ Application:
/tmp/linux_led_dimmer.elf Browse...

Commands to execute before application

Skip download to target path.

Apply Revert

Debug Close

Launching A Remote Debug Session



1. Create a debug configuration to store settings for attaching to remote host
2. Remote System Explorer uses SSH to control remote system
3. Remote System Explorer uses SFTP to browse remote file system
4. Remote System Explorer launches **gdb-server** automatically
5. SDK GDB attaches to remote **gdb-server** session for application debug

More Debug Options



ARM®

The Architecture for the Digital World®

www.em.avnet.com/zeddebugadapter

Checkpoint – Pop Quiz

Which communications mechanisms are used by the Remote System Explorer to communicate with the target hardware?

- 1) UART**
- 2) SSH**
- 3) TCP/IP**
- 4) SFTP**

Lab 4.2

- **Using SDK for Linux Application Debug**
 - Use the SDK debugger to track down root cause for unexpected application behavior
- **Note: All SpeedWay lab materials, including lab instructions and solutions, can be downloaded from ZedBoard.org**

Implementing Linux on the Zynq™-7000 AP SoC

**Chapter 5 – What's Next, Where to Learn
More**



Accelerating Your Success™

Course Objectives Review

- Built critical software components of an embedded Linux system
- Explored Linux software development with Xilinx SDK
- Connected SDK to hardware for application execution and debug
- Learned the advantages of running Linux on Zynq



Next Steps



Accelerating Your Success™



Home > Product Support & Documentation

Product Support & Documentation

Search Support

Search all Documentation, Answers, and Forums

Search Tips

Design Advisory Answer Records

- Watch this video to learn about Design Advisories
- Be notified of solutions to issues affecting current designs
- Understand how to sign up for weekly alerts

1 2 3



[Watch Overview Video](#)

Additional Resources

- Browse or Ask a Question on Xilinx Forums
- View Solution Centers
- View Most Recent Answers
- View Online Training Videos
- Subscribe to Email Alerts
- See All Customer Notices
- Open a WebCase

Silicon Devices

Boards & Kits

IP

Design Tools

App Notes & White Papers

Filter by Product Name

All Programmable SoCs

Zynq-7000

XA Zynq-7000

FPGA Device Families

7 Series

Virtex-6

Virtex-6Q

Virtex-5

Virtex-5Q

Virtex-5QV

Virtex-4

Virtex-4Q

Virtex-4QV

Spartan-6

Spartan-6Q

Extended Spartan-3A

Spartan-3A DSP

Spartan-3AN

Spartan-3A

Spartan-3E

➤ [Zynq-7000](#) | Product Page

Documentation

- [Zynq-7000 Data Sheets](#)
- [Zynq-7000 User Guides](#)
- [Zynq-7000 Application Notes](#)
- [Zynq-7000 Errata](#)
- [Zynq-7000 White Papers](#)

[See All Zynq-7000 Documentation](#)

Answers

- [Zynq-7000 AP SoC Devices - Silicon Revision Differences](#)
- [See All Zynq-7000 Answers](#)

Document Alert / Changes

- [Zynq-7000 All Programmable SoC Technical Reference Manual](#) : See Revision History in document.
- [XAPP744 - Hardware In The Loop \(HIL\) Simulation for the Zynq-7000 All Programmable SoC](#) : Placed tutorial ZIP file behind click-through license.
- [XAPP794 - 1080p60 Camera Image Processing Reference Design](#) : New application note.
- [Zynq-7000 AP SoC XC7Z020 CES Errata](#) : See revision history.
- [Zynq-7000 AP SoC Devices - Silicon Revision Differences](#) : This Master Answer Record contains a list of the Answer Records referenced in the Zynq-7000 EPP XC7Z020 CES Errata. Updated on November 2, 2012.

[Home](#) : [Support](#) : [Training](#)

Training

Xilinx hands-on training programs provide you the foundational knowledge necessary to begin designing right away. These programs target both engineers new to FPGA technology and experienced engineers developing complex connectivity, digital signal processing, or embedded solutions. Contact your local [Sales Rep](#) or [Authorized Training Provider](#) to see if your company has any [Training Credits](#) available.

Instructor-led Training

Language Training:

- VHDL
- Verilog
- Tcl Scripting
- C Programming
- C to Gates

Tools Training:

- Vivado™ Design Suite
- Vivado High-Level Synthesis
- ISE® Design Suite
- ChipScope™ Pro
- PlanAhead™
- System Generator

Device Training:

- Zynq™-7000 All Programmable SoC
- 7-Series FPGA Families
- 6-Series FPGA Families
- Virtex®-5 FPGA Family
- Contact [ATP](#) for Legacy Product Training

Point Solutions:

- Analog Mixed Signal
- Ethernet MAC Controllers
- Memory Interfaces
- Multi-Gigabit Serial I/O
- Partial Reconfiguration
- PCI Express
- Power & Debug
- Signal Integrity

Training by Design Speciality:

- Embedded Software Design
- Embedded Hardware Design
- Embedded Firmware Design
- System Architect
- FPGA Design
- Advanced FPGA Design
- DSP - Digital Signal Processing Design
- Connectivity Design

One Day FPGA Design Workshops:

- Embedded Design Workshop
- DSP Design Workshop
- Connectivity Design Workshop

Quick Links

-  [Register for a Course](#)
-  [Training Schedule](#)
-  [Training Providers](#)
-  [Training Promotions](#)

Getting Help and Support

- Visit the ZedBoard Community Site <http://www.zedboard.org>

The screenshot shows the homepage of ZedBoard.org. At the top, there is a navigation bar with links for Home, Overview, Buy, Support, Projects, Accessories, and Partners. A search bar is also present. The main banner features the text "Put Your Imagination to Work" and "Order Your ZedBoard™ Today". Below the banner are logos for Digilent, Avnet Electronics Marketing, and Xilinx. The page is divided into three main sections: "Getting Started", "Community Projects", and "Support", each with a list of links.

ZedBoard.org™

Home Overview Buy Support Projects Accessories Partners

Put Your Imagination to Work
Order Your ZedBoard™ Today

DIGILENT **AVNET** **XILINX**

Getting Started

- ▶ Documentation
- ▶ Reference Designs
- ▶ Training and Videos

Community Projects

- ▶ View Projects
- ▶ Create Projects

Support

- ▶ Forums
- ▶ Contacts
- ▶ ZynqGeek Blog

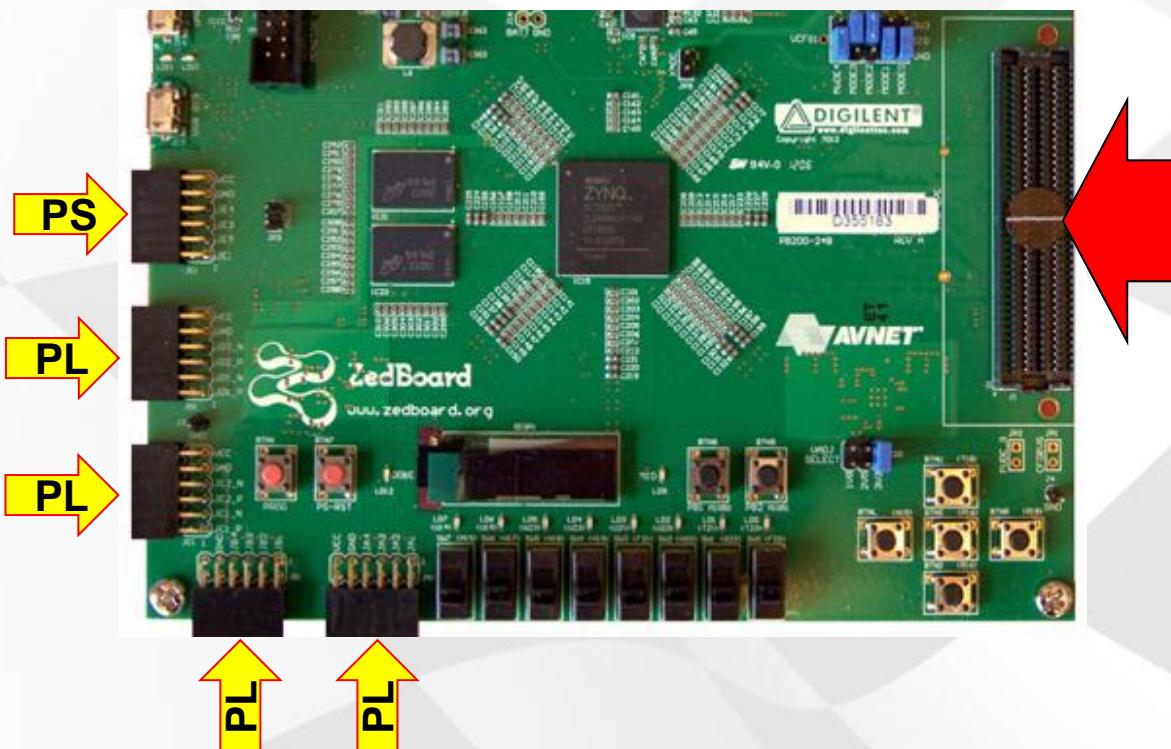
Welcome to ZedBoard.org

User login

Prototyping with Evaluation Boards



Pmods

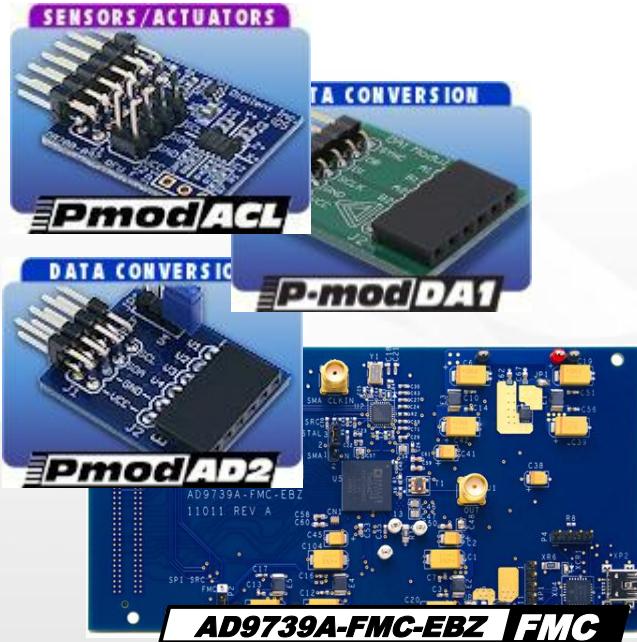


FMC



Authorized Distributor

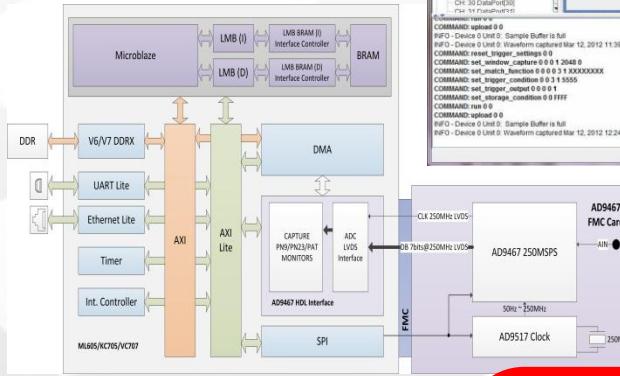
Analog Devices Evaluation Boards



Pmods

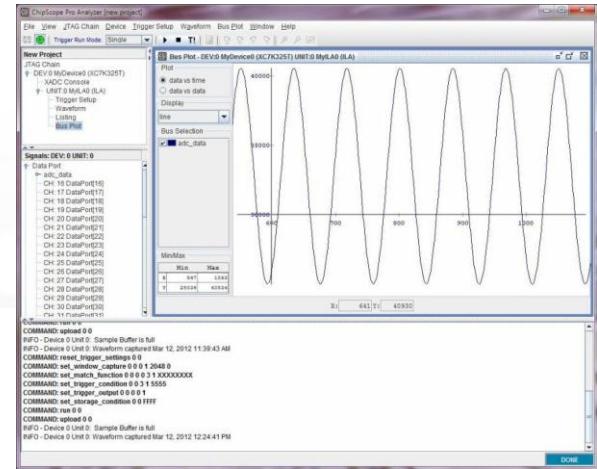
- 12-, 16-, and 24-bit ADCs
- 8-, 12-, and 16-bit DAC's
- 3-axis Accelerometer
- Gyroscope
- Digital Potentiometer
- I2C Temp Sensor

Reference Designs with Immediate Results



FMCs

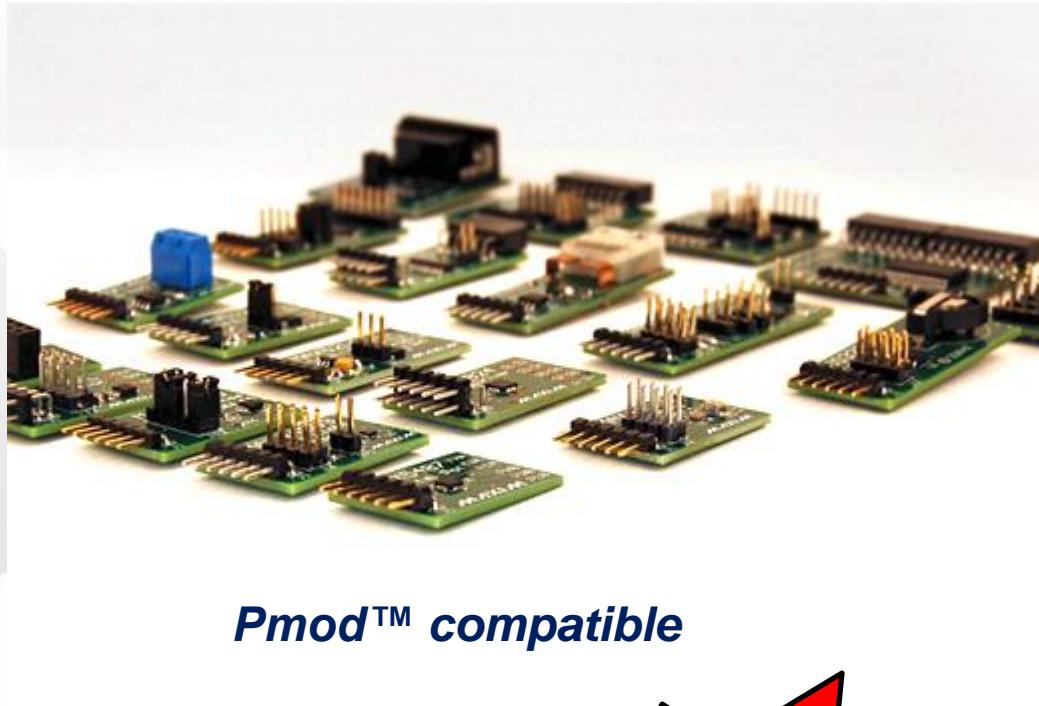
- Hi-speed 14-bit DAC @ 2.5GSPS
- High-speed 16-bit ADC @ 250MSPS
- RF Transmit and Receive channels
- Interposer for SDP-B



AMS

- Instrumentation Amplifier Board
- RF RMS Detector
- RF LogAmp Detector

MAXIM Plug-In Peripheral Modules



Pmod™ compatible

\$89.95

- Over 15 Popular Functions

- Data conversion,
Isolation,
IO Expansion, Current
Sense, Current Limiting,
Temp Sense, Light Sense,
RS232, RS485, and more

- Complete Software, FPGA
code and documentation
available on-line

- All modules follow
Digilent's Pmod pin-out
conventions

Pmod™ is a trademark of Digilent, Inc

ZedBoard Accessories

 ZedBoard.org™

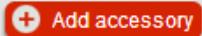
Login | Register

[Home](#) [Overview](#) [Buy](#) [Support](#) [Projects](#) **Accessories** [Partners](#) 

[Home](#)

[FMC](#)
[Pmods](#)
[Software](#)
[Other](#)

Accessories



ZedBoard was designed for expandability, with the 5 Digilent® Pmod™ compatible headers and a low-pin count (LPC) FMC slot. Check out what ZedBoard accessories and add-ons are available to help jump start your design effort. If you would like to advertise your ZedBoard compatible accessory, please complete the accessory posting request form. In posting these accessories, ZedBoard.org is in no way supporting or endorsing the listed items. ZedBoard.org assumes no responsibility or liability for any listed accessory item.

[Pmod >](#)


\$89.95

Maxim's Analog Essentials Collection of Peripheral Module

A collection of 15 peripheral modules that connect to the ZedBoard's Digilent® Pmod™ compatible headers. Functions include ADC and DAC, digital pot, real-time clock, RS232, and much more. Documentation includes example designs and full source code

[Website](#)

ZedBoard Sponsors

Sponsors



Featured partners

FOX

INTELTRONIC

MARVELL

MICRON

Texas Instruments

ZedBoard Purchase

 ZedBoard.org™

Login | Register

[Home](#) [Overview](#) **Buy** [Support](#) [Projects](#) [Accessories](#) [Partners](#) 

[Home](#) » [Buy](#)

ZedBoard is available for purchase from Avnet Electronics Marketing and Digilent.

 \$319

Digilent exclusively offers the academic edition ZedBoard world-wide to all academic qualifying purchasers.
(* Proof of student or professor status is required)
Part Number: ZEDBOARD

[Buy Item](#)

 \$395

Avnet Electronics Marketing offers the commercial version of ZedBoard world-wide via the Avnet Express web site or through a local Avnet sales office

Part Number: AES-Z7EV-7Z020-G

[Buy Item](#)

Where to go for additional training?

Xilinx Authorized Training Partner (ATP) Courses

◆ Building Linux Platform

- Use of Open-Source Components
- Environment Configurations
- Embedded Linux Development

**Embedded Open-Source Linux Design
(2 Days, Level 4)**

www.xilinx.com/training/atp.htm

◆ Details of Using Zynq Processor

- Advanced Boot Methodologies
- Cortex-A9 Processor Services
- Advanced DMA Controller

**Advanced Features and Techniques of
Embedded Systems Software Design
(1 Day, Level 4)**

www.xilinx.com/training/atp.htm

◆ C-Based Design

- High-level synthesis flow of Vivado HLS
- Optimize design performance
- Creating Custom C-to-gates Peripheral

**High-Level Synthesis with Vivado HLS
(2 Days, Level 3)**

www.xilinx.com/training/atp.htm

www.xilinx.com/training/xilinx-training-courses.pdf

Where to go for additional training?

Avnet Speedway Courses

- ◆ **Introduction to the Zynq-7000 AP SoC**
 - Understand Zynq development flow
 - Introduce the dual-core ARM® Cortex™-A9 MPCore™
 - Utilize Xilinx tools to design a Zynq system
- ◆ **Software-Defined Radio on Zynq-7000 AP SoC**
 - High-speed analog signal chain
 - Direct conversion radio architecture
 - High-speed data converter interfacing to FPGA's



www.em.avnet.com/zynqspeedways

Thank You!

Please fill out the course survey



Accelerating Your Success™

Appendix



Accelerating Your Success™

Reference

- **SDK Help:** In SDK, select *Help → Help Contents*
- **Eclipse website**
<http://www.eclipse.org/>
- **Eclipse CDT (C/C++ Development Tooling)**
<http://www.eclipse.org/cdt/>
- **Xilinx Wiki**
<http://wiki.xilinx.com>

Cross Build Platform

- Virtual machines can run desktop Linux
- SpeedWay labs use VMware Player to run CentOS Workstation
- Latest VMware Player can be downloaded from the VMware website:

<http://www.vmware.com/products/player/overview.html>



- CentOS can be downloaded from mirrors listed on CentOS website:

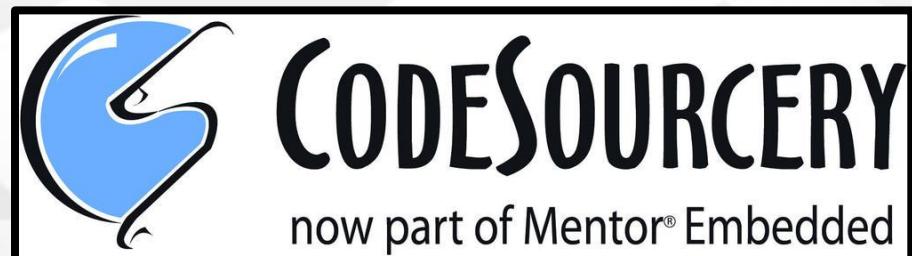
<https://www.centos.org/>



Cross Build Platform

- Virtual machines create a portable build environment where the cross toolchain can be installed
- Zynq cross toolchain from CodeSourcery can be downloaded from the Xilinx website:

http://www.xilinx.com/member/mentor_codebench/xilinx-2011.09-50-arm-xilinx-linux-gnueabi.bin



- Free and open source Git SCM can be downloaded from the Git website:

<http://git-scm.com/>



Lab 0.1

- **Creating a Cross Build Platform for Zynq**

U-Boot Commands/Variables

Common U-Boot Commands:

boot	Boot using the commands in the bootcmd environment variable
md	Memory display, used to dump memory contents to console
mw	Memory write, used to write memory values arbitrarily
setenv	Set environment variables to indicated string
printenv	Print current setting of environment variable(s)
ping	Ping host machine using Ethernet interface
tftp	Trivial File Transfer Protocol client for moving data to/from remote host

Common U-Boot Environment Variables:

baudrate	Baudrate of the terminal connection, defaults to 115200
bootcmd	Command string executed when boot delay countdown is not interrupted
bootdelay	Delay count in seconds prior to executing bootcmd automatically
ethaddr	Ethernet MAC address for Ethernet interface
ipaddr	Local IP address, needed for tftp command
serverip	TFTP server IP address, needed for tftp command

Linux Command Summary

File Management

ls	Directory listing
cd	Change directory
cp	Copy file or directory
mv	Move file or directory
rm	Delete file or directory
pwd	Show current directory
tar	Create/extract .tar and tar.gz archives
chmod	Change file or directory permissions
chown	Change file or directory ownership
mkdir	Create a directory
mount	Mount a file system to a mount point
umount	Detach file system from a mount point
touch	Update access/modification time of file
df	Report file system space usage
fdisk	Modify disks and partitions
echo	Print a string or variable to console
cat	Concatenate files and print to console
sudo	Perform command as superuser
su	Change current user to superuser
whoami	Displays current username
tail -f	Monitor messages sent to log files
more	File perusal filter for terminal viewing
less	Similar to more command, but better

Networking

ifconfig	Configure network interface
ifup	Bring network interface up
ifdown	Bring network interface down
ping	Request remote host send ICMP echo

Process Control

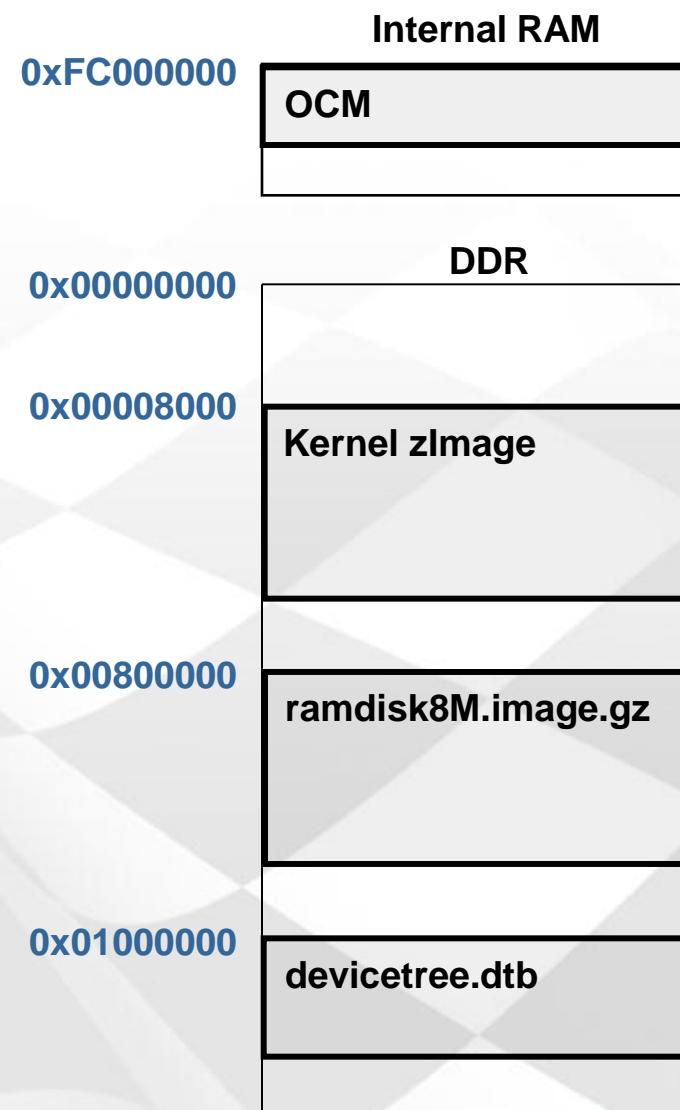
<Ctrl>-c	Terminate foreground process <SIGINT>
ps	Report snapshot of current processes
top	Display process task list
kill	Send signal to a process
renice	Alter priority of running processes

Kernel

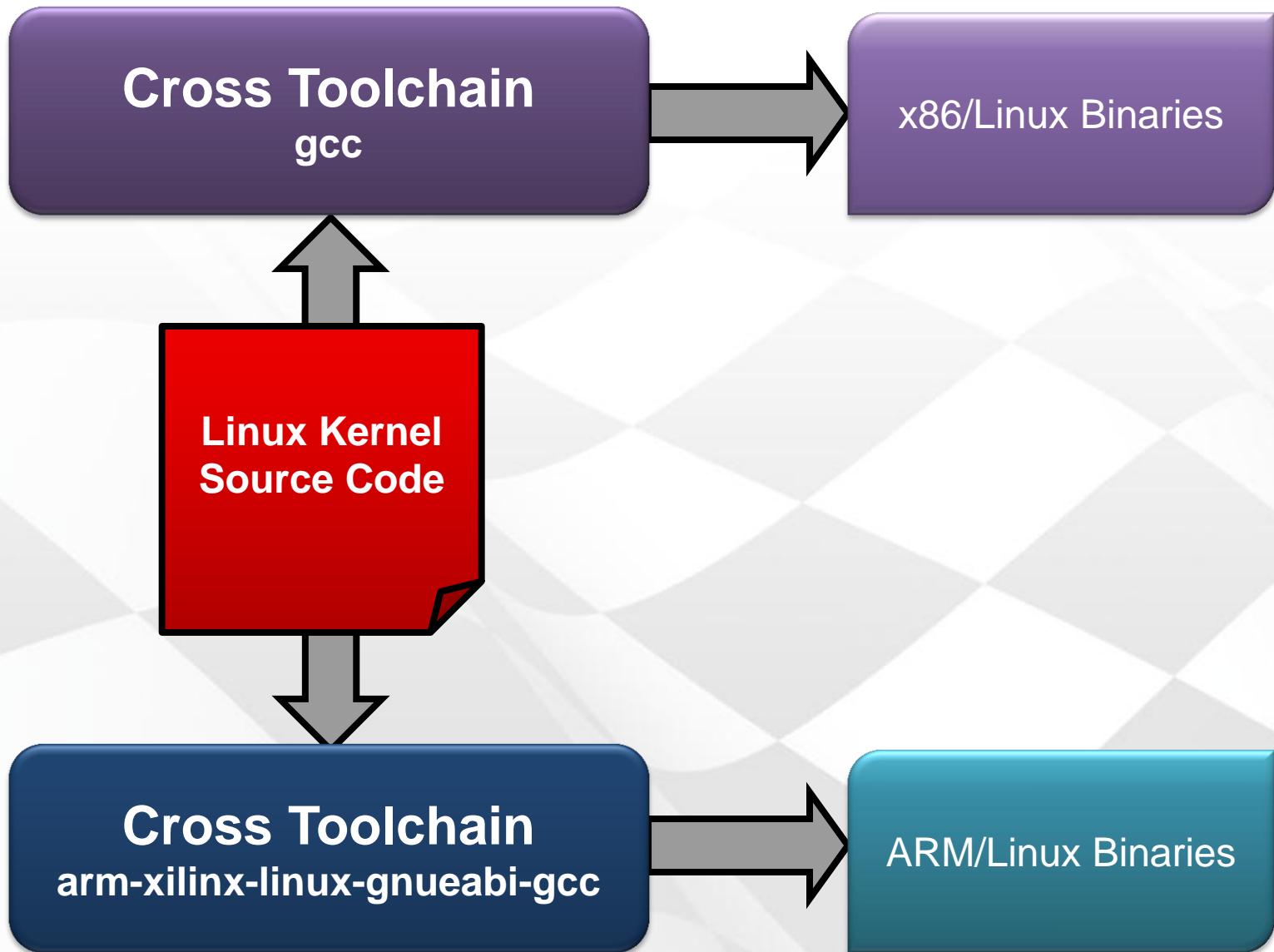
dmesg	Print the kernel ring buffer
insmod	Dynamically insert kernel module
lsmod	List status of modules in kernel
rmmmod	Dynamically remove kernel module
uname -a	Show kernel version and architecture

Memory Map of Boot Process (7ynq-7000)

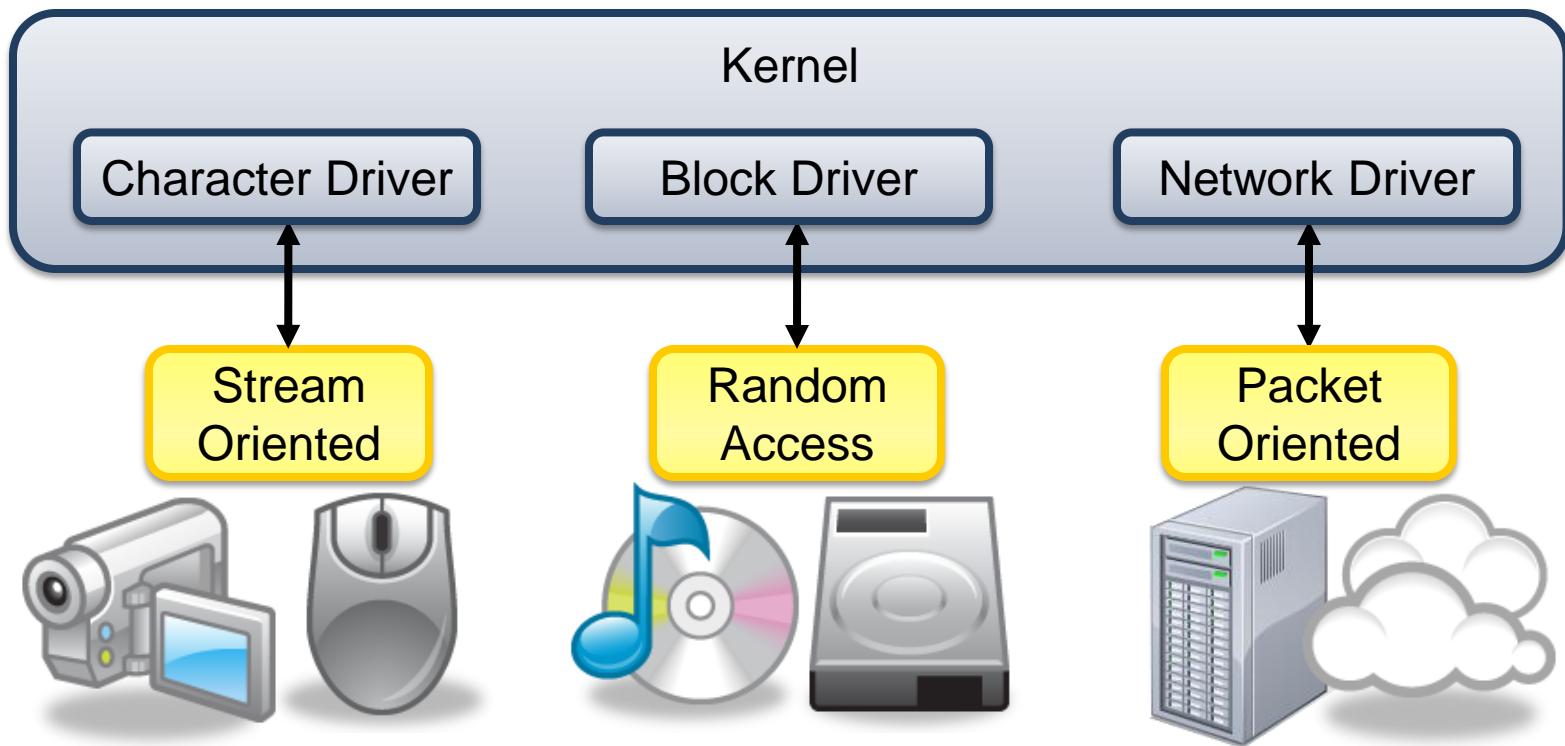
- FSBL is loaded into OCM which is remapped during handoff to U-Boot
- U-Boot is loaded first into DDR
- U-Boot then loads the zImage from boot source as directed by BOOTMODE
- U-Boot performs a checksum and then relocates (if required) the Kernel found in the zImage to the load address
- U-Boot then jumps to the entry address specified in the zImage header, linux boot starts at this point
- Note: the addresses used here are for reference only and may not apply to all devices



Native vs. Cross Compiler

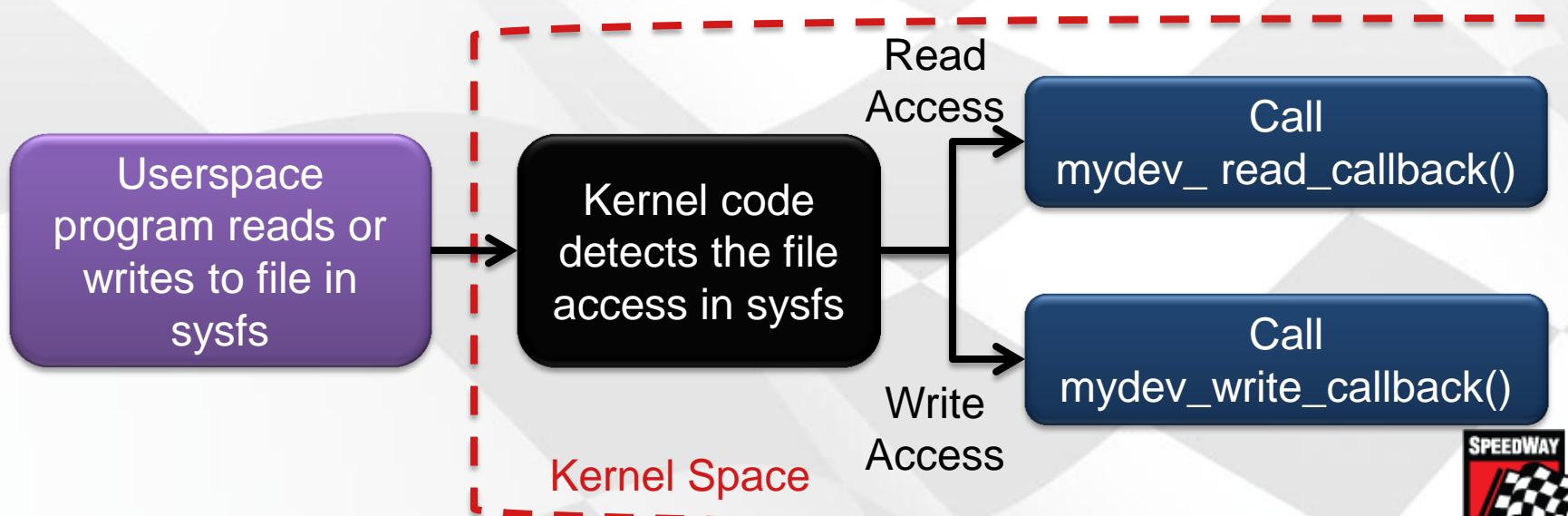


Linux Device Drivers



Linux Device Drivers

- The sysfs device driver author exports device attributes
- Attributes allow user space access to driver resources using regular files in the sysfs directory structure
- Device driver author installs specific callback functions to handle read and write actions to the file
 - Possibilities are endless: Trigger certain behavior, set internal driver parameters, access registers



Timesys Support

- **Expert, responsive team of Linux engineers**

- Dedicated support engineers
- 24-hour turn-around time
- Unlimited number of tickets for build related issues
- Every subscriber (seat) can submit their own tickets

- **Quick start training**

- **Self-help & education**

- Extensive documents, videos, demos

- **Customized training – available on request**

- **Professional services available**

- Porting/board bring-up, driver/package development
- Fast boot, low-latency, field upgradeability



Timesys Contacts

- **contact Al Feczko at:**

- Al.feczko@timesys.com
- 1-866-392-4897 or (412)-325-6390
- (412) 897-2416 (mobile)



Request a personalized,
live demo and/or quote

- **contact Technical Support at:**

- <https://linuxlink.timesys.com/support>