Real-Time Systems

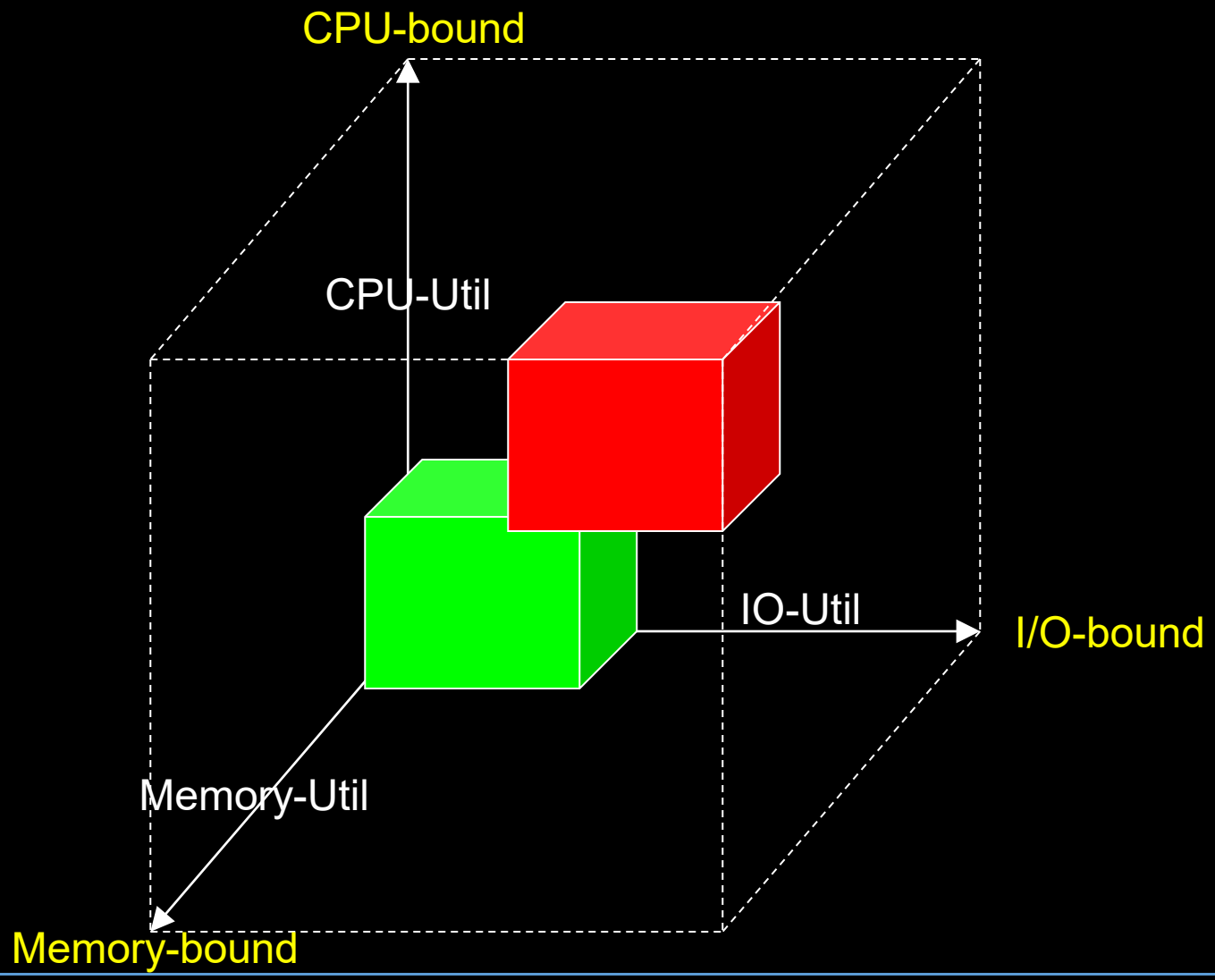
Lecture Topic - Device Interface Drivers and MMIO

Dr. Sam Siewert

Electrical, Computer and Energy Engineering

Embedded Systems Engineering Program

Resource View of HW/SW Interface



- Three-Space View of Utilization Requirements
  - CPU Margin – 30% for LUB
  - IO Latency (Bandwidth) Margin?
  - Memory Capacity (Latency) Margin?
- Upper Right Front Corner – Low-Margin
- Origin – High-Margin

Camera Demo - Linux Devices

- Computer Vision Application
  - OpenCV API – User Space
  - V4L2 API – Kernel, User Space Interface
  - UVC – Driver, Kernel Space Module
  - USB – Driver, Kernel Space Module

- sudo cat /proc [PIDs, meminfo, cpuinfo,interrupts, slabinfo, devices, bus/pci, bus/input, …]

- sudo lshw, lsusb, lspci, lsscsi, lsmod, lscpu, slabtop [more]

- dmesg, htop, iostat from sysstat [more];
- free, lsof, netstat, strace, vmstat [more]

- Trace, Profile, Debug Tools (coming up …) –
  - KernelShark (`sudo apt-get install kernelshark`),
  - wireshark (`sudo apt-get install wireshark`),
  - ftrace, sysprof, systemtap, etc.

Demo Camera Applications On Linux Laptop

For OpenCV hit “q” in imshow window to properly shutdown

Reset USB bus and device
If you happen to exit badly – e.g. with Ctrl-C or run camorama
[sigint could catch and handle]

Use syslog instead of printf
- Buffers for output in slack time
- Avoids immediate blocking I/O that can delay a service
- Similar features
- Add custom time-stamp

Linux kernel modules
- Must use “printk”
- Kernel Modules, Drivers

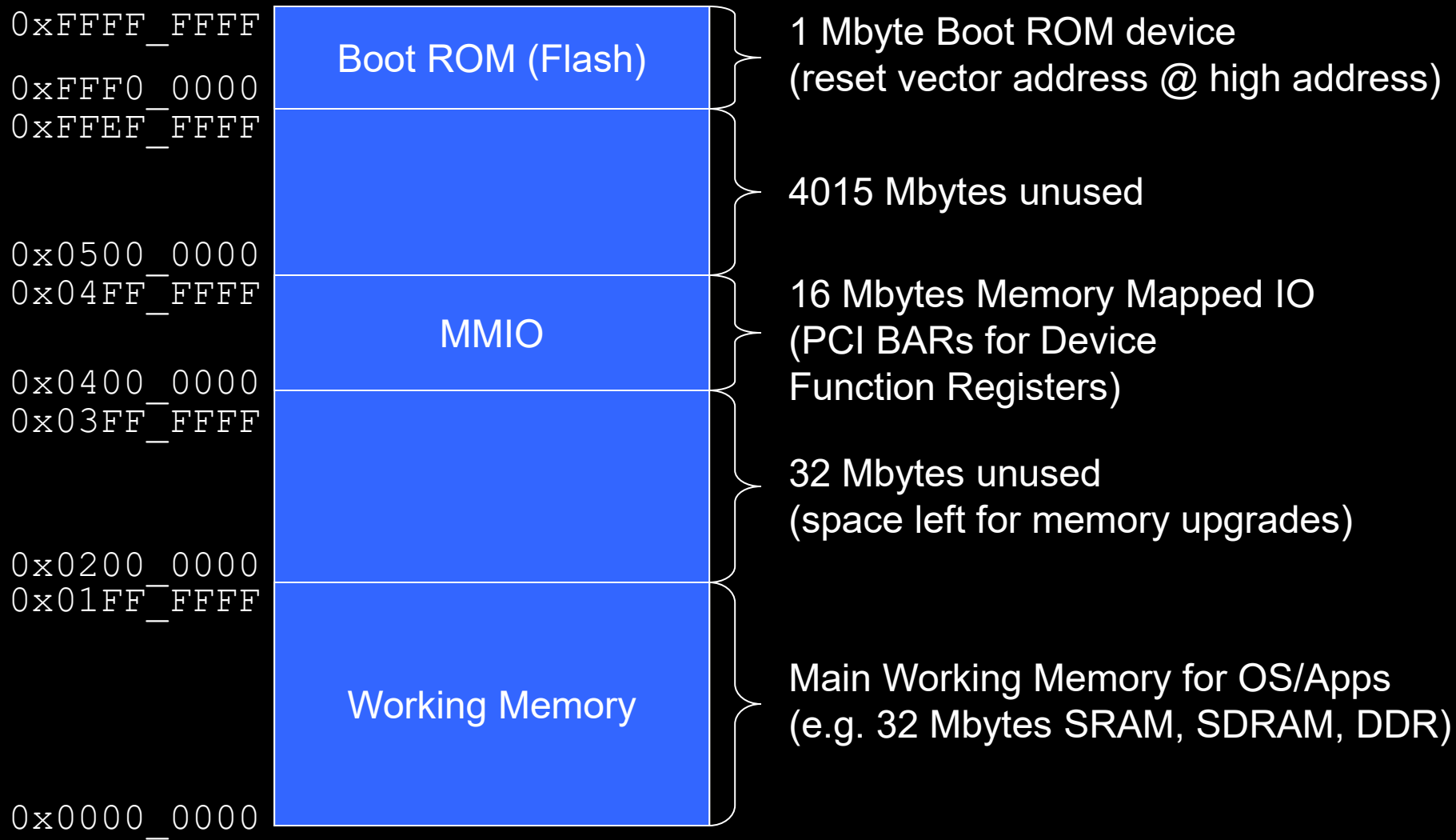
Embedded I/O (HW View)

- Analog I/O
  - DAC analog output: servos, motors, heaters, ...
  - ADC analog input: photodiodes, thermistors, ...
- Digital I/O
  - Direct TTL I/O or GPIO
  - Digital Serial (I2C, SPI, ... - Chip-to-Chip)
  - Digital Interface to UART/Line Driver, Serializer-deserializer (Serdes), or Physical I/F
    - Digital I/O to Line Driver Interface (e.g. RS232)
    - 10/100 MII, 1G RGMII, 10G XGMII interfaces to Serdes to Phy

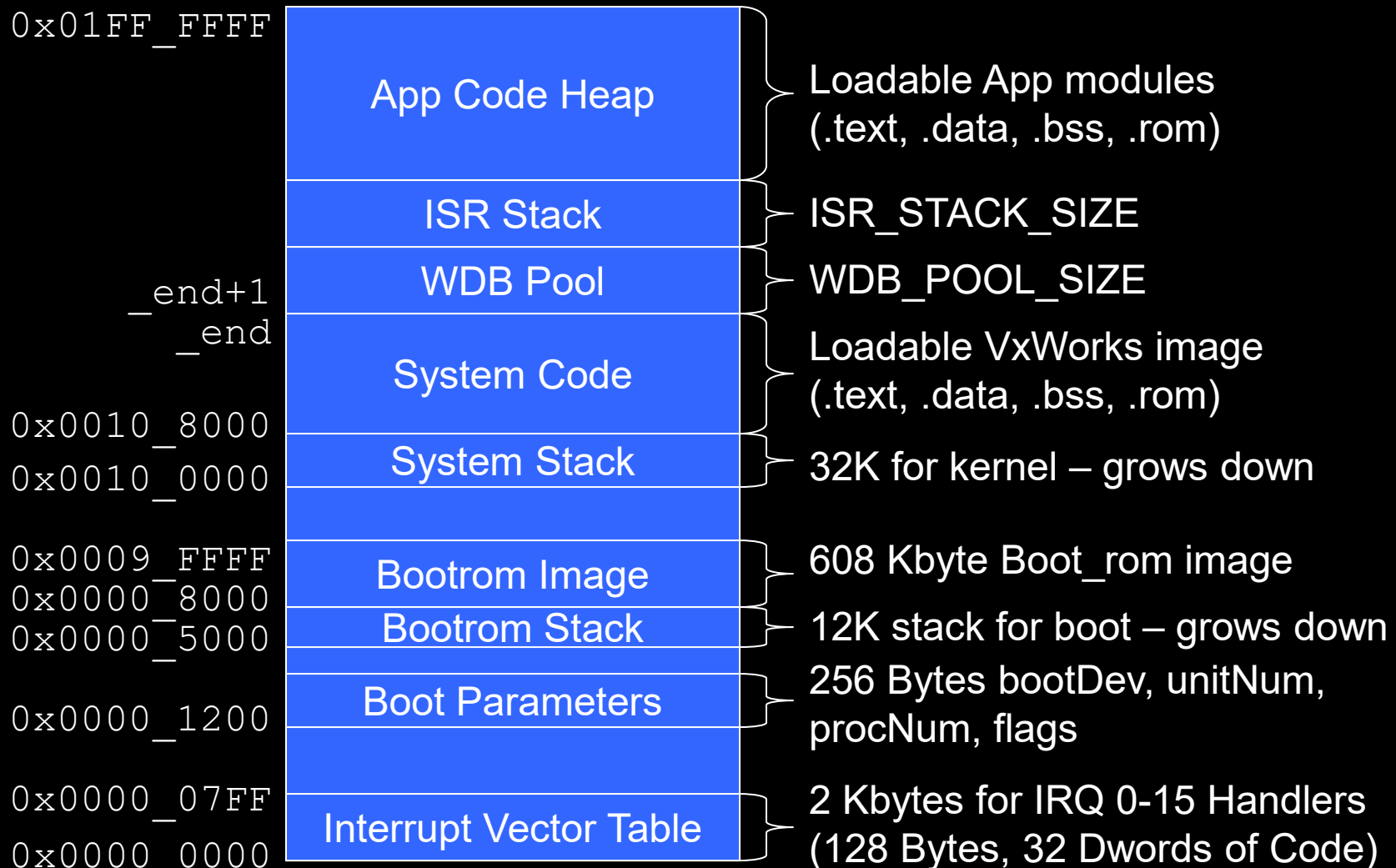
CPU Core I/O (SW View)

- Word
  - Register Control/Config, Status, Data
  - Typical of Low-Rate I/O Interfaces (RS232)
- Block
  - FIFOs (2-32K), Dual-Port RAM and DMA
  - Typical of High-Rate I/O Interfaces
- System Memory Map (32 or 64 bit space)
  - Physical Memory (SDRAM, DDR, SRAM)
  - BootROM (Typically Flash)
  - Configuration and Control Registers

Example 4Gb System Memory Map



OS and App Use of 32 Mb Memory



Platform/RTOS I/O (FW View)

- Driver Interfaces Used By Applications
  - Top-Half: Application Interface
  - Bottom-Half: FW Interface to HW
- Application Interface Examples
  - Word-at-a-Time I/O Drivers
    - Serial byte streams
    - Device Configuration, Control, and Status
  - Block I/O Drivers
    - Disk, Flash, High-Rate Network (Typically DMA)
  - Stack I/O Drivers
    - Filesystem, TFFS/DOC, TCP/IP

Network Stack – Driver Interface

L7	**Application**
L6	Data Presentation
L5	Sockets API - Session
L4	**TCP**
L3	IP
L2	IEEE 802.3z - GigE
L1	Cat-6 UTP

Layer 1 - Physical (Port)
Layer 2 - Link (Hub)
Layer 3 - Network (Switch)
Layer 4 - Transport (Router)

Platform/RTOS I/O (FW View) – Device Interfaces

Word-at-a-Time - RS-232/422 UART, GPIO, RELAY, ADC

Block I/O - Common Flash Interface, Ethernet Link Layer

Stack I/O - E.g. Flash Filesystem and Network Stack

```
[    0.871116] mmc0: sdhost-bcm2835 loaded - DMA enabled (>1)
[    0.873353] Waiting for root device /dev/mmcblk0p7...
[    0.930047] mmc0: host does not support reading read-only switch, assuming write-enable
[    0.932946] mmc0: new high speed SDHC card at address 0001
[    0.934278] mmcblk0: mmc0:0001 EB1QT 29.8 GiB
[    0.938366]  mmcblk0: p1 p2 < p5 p6 p7 >
[    0.949017] mmc1: new high speed SDIO card at address 0001
[    0.979784] EXT4-fs (mmcblk0p7): mounted filesystem with ordered data mode. Opts: (null)
[    2.761221] EXT4-fs (mmcblk0p7): re-mounted. Opts: (null)
[   25.240038] EXT4-fs (mmcblk0p5): mounted filesystem with ordered data mode. Opts: (null)
pi@raspberrypi:~/Downloads $
```

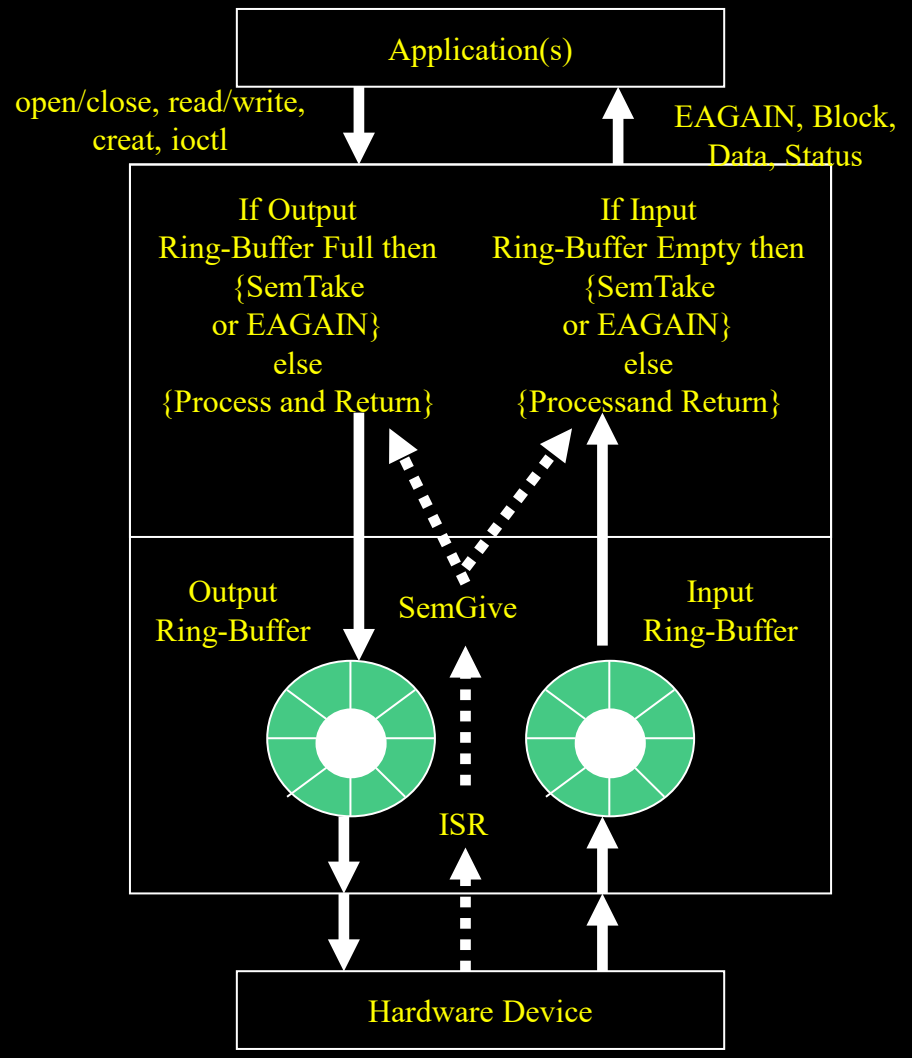
Linux Block Interfaces

mmcblk is the block interface to SD card Nand flash on Linux

Disk drivers are normally SCSI (SAS) or ATA (SATA) – including SSD

The mmcblk driver interface is critical to I/O performance for stored data

Driver Design



- Application Interface
  - Application Policy
  - Blocking/Non-Blocking
  - Multi-thread access
  - Abstraction
- Device Interface
  - SW/HW Interface
  - Immediate Buffering
  - Interrupt Service Routine
- HAL Interface

Cached Memory and DMA

- Cache Coherency
  - Making sure that cached data and memory are in sync
  - Can become out of sync due to DMAs and Multi-Processor Caches
  - Push Caches Allow for DMA into and out of Cache Directly
  - Cache Snooping by HW may Obviate Need for Invalidate
- Drivers Must Ensure Cache Coherency
  - Invalidate Memory Locations on DMA Read Completion
    - cacheInvalidate()
  - Flush Cache Prior to DMA Write Initiation
    - cacheFlush()
- IO Data Cache Line Alignment
  - Ensure that IO Data is Aligned on Cache Line Boundaries
  - Other Data That Shares Cache Line with IO Data Could Otherwise Be Errantly Invalidated

Disadvantages of Abstracted Driver

- Generally the Software Engineering Value Outweighs any Inefficiency
- PDL – Peripheral Driver Library, Focus on Device Interfaces [Devices Interfaces and HAL]
- Full Driver Abstraction Adds Overhead and Complexity
  - Function calls and table look-up
  - More Code than Single Layer API
- Adds Buffering and Copy Features
  - Can Use Zero-Copy Buffering (Pointers) to Minimize Impact
  - Zero-Copy Buffers can be Complex

Advantages of Abstracted Driver

- **Portability**
  - If SW/HW Interface changes, change Device Interface
  - If Application Interface changes, change Application Interface
- **Testability (Test HW Interface and User Space Application Interface Separately)**
- **Single Point of Access and Maintenance**
- **Enforce Multi-Thread Usage Policies**
- **Separate Buffering and ISR from Usage**
- **Common Application Entry Points**
- **Scheduled I/O (Most Work in Task Context rather than ISR Context)**

Driver Writer Resources

- Linux Device Drivers
  - Kernel Modules (Loaded After Boot)
  - Built-in Drivers (Loaded with Kernel Image)
  - Linux 2.6.x Device Drivers, by Rubini and Corbet [Web Version]
  - Jerry Cooperstein's Linux Device Drivers
  - Linux Foundation [most current]

- VxWorks RTOS Device Interfaces
  - All applications and drivers are kernel modules (E.g. Btvid analog NTSC Video Decoder)
  - Driver entry point registration table
  - Section 3.9 of VxWorks Programmer's Guide, pages 166-174
  - VxWorks Programmer's Guide, Appendix D for x86 Architecture – pages 431-478