seL4 Infrastructure: USB and Beyond

Capgemini engineering

Introductions

Capgemini Engineering, Bath, UK

Our focus is projects with High Integrity and Embedded Software aspects

A small team (2 to 3 engineers) have been working with seL4, since late 2021

Team has changed over time. Contributors, in chronological order:

- Stephen Williams
- Mark Jenkinson
- Josh Felmeden
- Daniel Storer
- Tom Harvey
- Bill Ellis (me!)

Mission

Overall Objective:

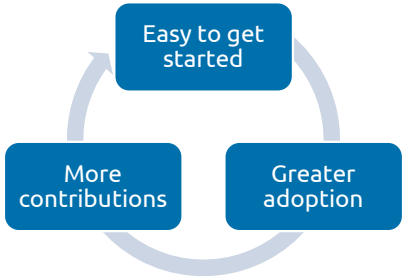
- Make it easy to get started with the seL4 Microkernel
- To encourage greater adoption, and benefit from that larger community!

Our Contributions:

- Public (open-source) contributions to the seL4 infrastructure
- Published as: The seL4 Developer Kit: https://sel4devkit.github.io

Today:

- Consolidation of the seL4 Developer Kit
- Explore its approach, and major contributions
- Reflect on its capabilities



1 - seL4 Developer Kit

Capgemini engineering

seL4 Developer Kit (overview)

Audience:

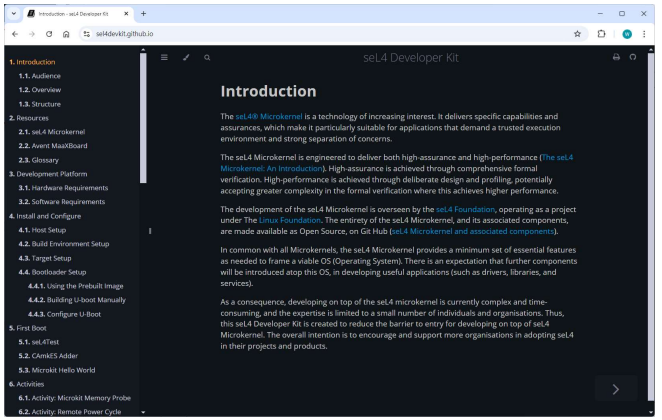
- Experienced application developer (Windows, macOS, Linux)
- Wanting to begin development with seL4

Format:

- Book style website (written: Markdown, publish: mdBook)
- Covers all the available material
- Everything on Git Hub (Git Hub Pages, Git Hub Packages, Git Hub Repositories)

Strategy:

- Be specific
- Small incremental steps
- Show expected outcomes



seL4 Developer Kit (up to first boot…)

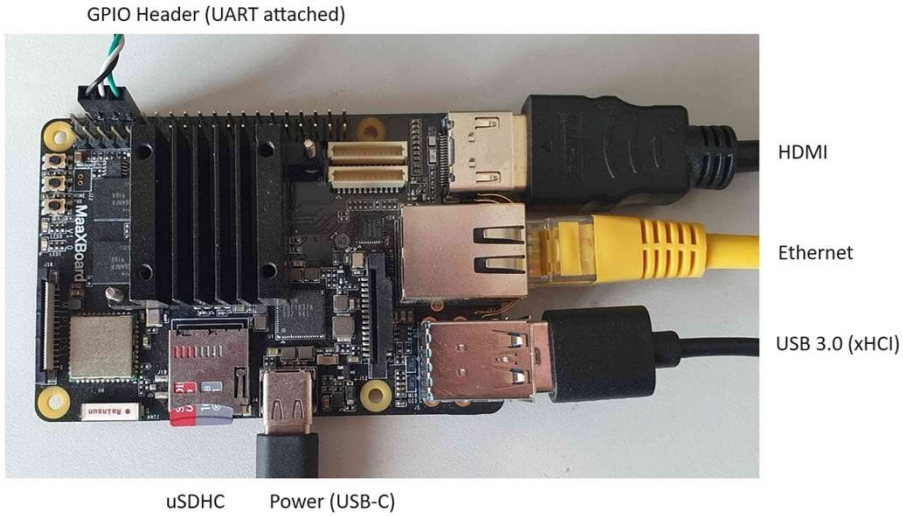
Key Features:

- Target hardware: The Avnet MaaXBoard (SBC, i.MX 8M SoC, Quad ARM Cortex-A53)
  - *Readily available, inexpensive, various devices, extensive public documentation*

- Identify minimum Development Platform (Hardware and Software requirements)
  - *Hardware part list, software dependencies, docker images*

- Detail configuration of Development Platform
  - *UART wiring, bootloader, image building, image loading*

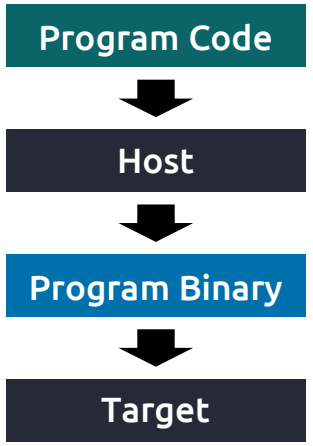
- Small set of first boot examples
  - *Step by step instructions, with sample expected outputs*



seL4 Developer Kit (Development Platform: Host)

Development Platform:

- From program code to program running on the target



Host is where we build the program:

Features:

- Separate Docker for each framework: CAmkES and Microkit
- Minimal needed content (and Vim)
- Prefer build from source (except for stock Debian dependencies)

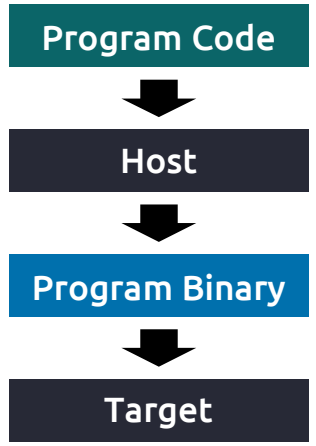
CAmkES:

- More features, constraints, complexity: *The monolithic framework?*
- Use its libraries, inherit its build system (git-repo, CMake, Ninja)

Microkit:

- Fewer features, constraints, complexity: *The micro framework?*
- Incorporate libraries as needed, build system context specific (or Make)

seL4 Developer Kit (Development Platform: Target)



Target is where we run the program:

Features:

- Select U-Boot bootloader (feature rich, vendor support (NXP))
- Open configuration (no secure boot, bootloader can interrupt into console)
- Prefer build from source (Concession: DDR PHY and HDMI firmware in binary only)

Customizable:

- Same U-Boot bootloader binary for all booting scenarios
- Custom boot script (uEnv.txt) (program binary ; location: USB/SD/TFTP, format: ELF/IMG)

Initialisation:

- Early SoC bootloader loads firmware binaries (DDR PHY, HDMI, ATF) then starts our U-Boot bootloader
- Bootloader: Enables Clocks and Power Domains
- Bootloader: Locate program binary ; loads into RAM ; execute from RAM

seL4 Developer Kit (…beyond first boot)

Select content to help kick-start seL4 development:

- Use conventional seL4 components (e.g. from seL4 Foundation)
- Prioritise capabilities needed in starting seL4 development (drivers, libraries)
- Illustrate the development process

Present as collection of Activities:

- Self-contained to ease understanding
- Flexible in style to best fit each Activity

seL4 Developer Kit (summary of Activities)

Drivers

- Harness U-Boot Driver Library: Suite of Drivers
- xHCI (USB 3.0) Driver
- Display Driver (HDMI output)

Examples

- Simple Memory Probe
- U-Boot Ethernet Driver plus Pico Server
- Security themed Case Study

Virtualization

- Single Linux Guest
- Dual Linux Guests
- Leverage Linux for USB Routing

Target Hardware

- Adding a Pressure Sensor

Host Hardware

- Using PPPS USB Hub for remote power cycle

Extensions

- Harness U-Boot Driver Library: Add new Driver
- Harness U-Boot Driver Library: Port to new Platform
- Guide on Porting seL4 to a new Platform

2 – Device Drivers

Capgemini engineering

Device Drivers

Device Drivers are a priority for the seL4 Developer Kit:

- The compromise of all Microkernels: Device Drivers are a user-mode concern
- Demand a complex mix of understanding hardware, software, and large technical specifications

Strategy:

- Select different Device Drivers and adopt different reuse strategies
- Aim is that developers find the Driver they need, or a pattern which helps them develop what they need

Previous reuse patterns:

- **Port Driver Framework:** *Port the U-Boot Driver Framework to seL4, to access its suite of Device Drivers*
  - Details: “Porting U-Boot Drivers to seL4” (Stephen Williams / seL4 Summit 2022)
- **Port Driver:** *Port NetBSD xHCI Driver (USB 3.0), and related Drivers, to seL4*
  - Details: “From Zero to a Native xHCI Driver” (Josh Felmeden / seL4 Summit 2023)

Additional reuse pattern:

- **Blended Driver:** *Blend resources to develop Display Driver*

Device Drivers (Blended Driver, design)

Activity: Blend resources to develop Display Driver

Premise:

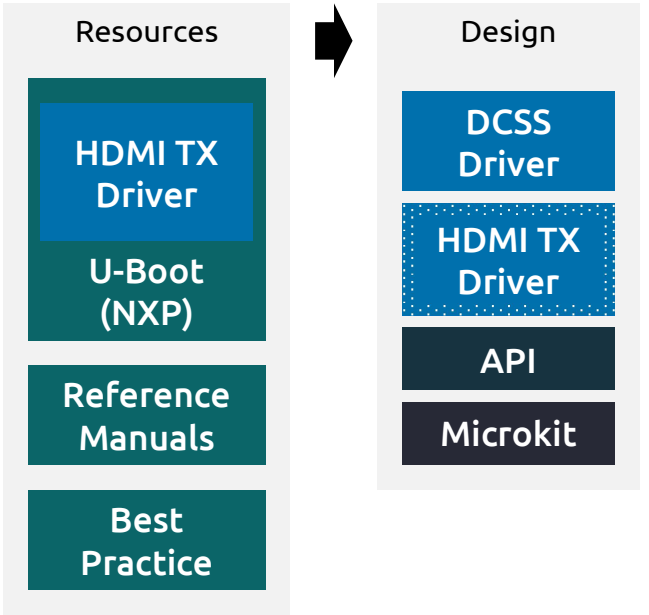
- Various resources available related to MaaXBoard display output
- Blend these resources together to prepare a Device Driver

Resources:

- Vendor (NXP) fork of U-Boot has permissive HDMI TX Driver
- Vendor (NXP) Reference Manuals
- Best practice, code and documentation (Linux, NetBSD, Internet, …)

Design:

- MaaXBoard has display pipeline options, including:
  - DCSS (Display Controller Subsystem) ➔ HDMI TX (HD Display Transmitter) ➔Display
- Position Display Driver as controlling both DCSS and HDMI TX



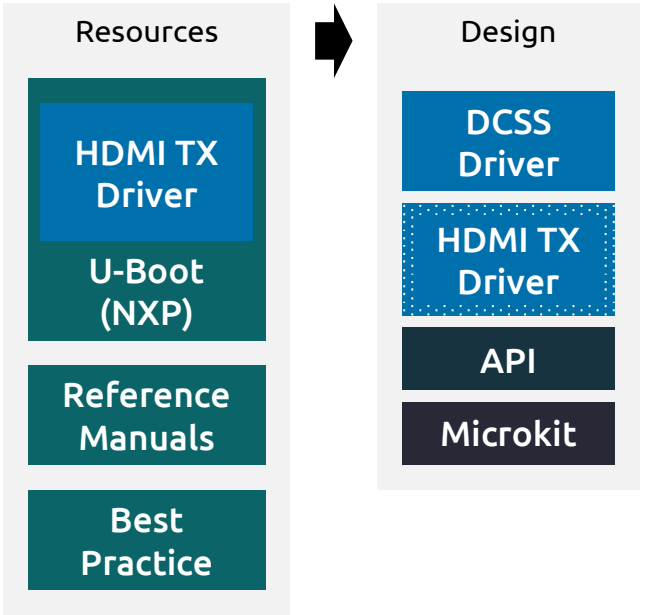
Device Drivers (Blended Driver, approach)

Approach:

- Reuse: HDMI TX Driver (HD Display Transmitter Controller)
- Create: DCSS Driver (Display Controller Subsystem)
- Display some dots
- Control resolution and display static image
- Display double-buffered images (smooth movement)
- API: Prepare image data in memory, notification to trigger draw

Outcomes:

- Self-contained code base
- Specific to i.MX 8M (as used by MaaXBoard)
- Intended for simplicity and understanding, not performance
- Momentary screen transition when switching between double-buffered images



3 – Virtualisation

Capgemini engineering

Virtualisation

Virtualisation offers a quick and flexible alternative for enhancing capability:

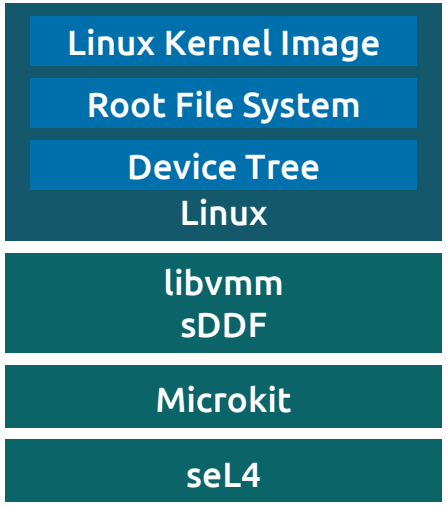
- Place an entire OS, executing some needed capability, in an (seL4) isolated user-mode process

Focus on Microkit solution:

- seL4 + Microkit + libvmm + sDDF + Linux Guests

Our contribution is packaging:

- Demonstrate virtualisation deployment
- Reuse capabilities from the Microkit solution
- Tailored for our target hardware (MaaXBoard) and audience



Virtualisation (Linux Guest For MaaXBoard)

Build Linux Guest for MaaXBoard:

- Buildroot (https://www.buildroot.org)

Configuration:

- Open configuration (root login shell, no hardening techniques)
- Select: Mainline Linux kernel, and target architecture (ARM)
- Remove all architectures, except those for target (CONFIG_ARCH_NXP / CONFIG_ARCH_MXC)
- Remove all Kernel Module Drivers

Environment:

- ARM SMC (Secure Monitor Call) includes SiP (Silicon Partner) Service Calls
- i.MX 8M SoC provides a version information SiP, which is required by Linux Driver “soc-imx8m”
- The libvmm Hypervisor emulates SMC, but is unaware of this i.MX 8M specific SiP
- Accommodated via small patch to libvmm, to emulate this single version information SiP

Virtualisation (direct and indirect patterns)

Virtualisation design choice:

- Directly or Indirectly pass device to Guest?

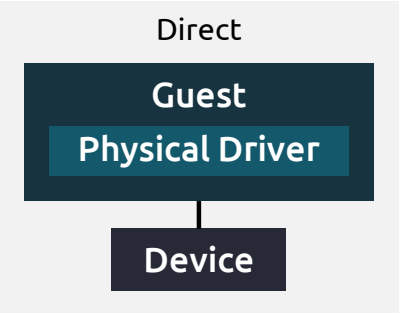
Implications:

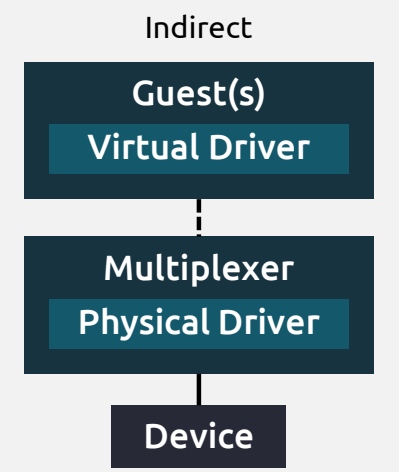
1. **Direct:** Physical device passed into Guest
   - No additional infrastructure ; inflexible ; cannot device share
2. **Indirect:** Physical device passed into Multiplexor, then virtual device passed into Guest(s)
   - Needs additional infrastructure ; flexible ; can device share

Indirect via VIRTIO (Virtual I/O Device) (https://docs.oasis-open.org/virtio):

- Protocol defines a suite of virtual devices
- Linux has extensive support for virtual devices
- Microkit solution has growing collection of virtual devices capabilities

Use Microkit solution, to package Activities that illustrate Direct and Indirect…





Virtualisation (Single and Dual Linux Guests)

Activity: Direct (Single Linux Guest)

Console Interface:

- Pass-through physical UART to Linux, use physical Linux UART Driver

Features:

- Physical Linux UART Driver has onward chain of eight device dependencies

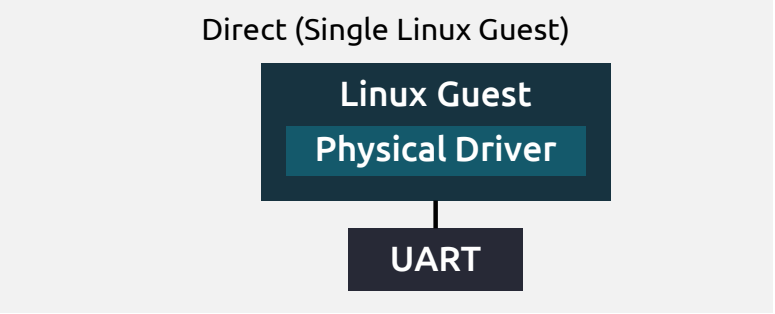
Activity: Indirect (Dual Linux Guests)

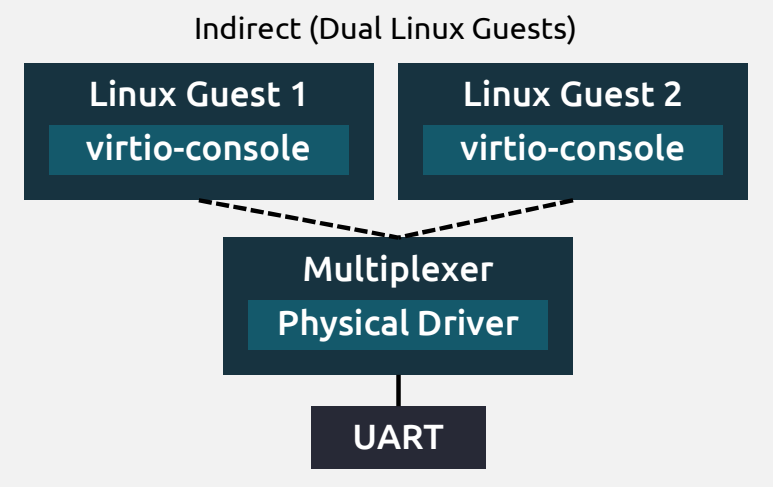
Console Interface:

- Pass-through physical UART to Multiplexer, use native physical UART Driver
- Pass-through virtual UART to Linux, use virtual Linux UART Driver (virtio-console)

Features:

- Linux virtual UART Driver has zero onward device dependencies
- kernel boot parameter: clk_ignore_unused (do not disable unused clocks)





Virtualisation (USB Routing, design)

Premise: Improved isolation of USB Devices sharing same physical port

- Multiplexer manages physical USB port, with USB Driver
- Multiplexer routes selected device traffic to a selected process

Design – Plan A: *Use xHCI specification hardware support for Virtualisation*

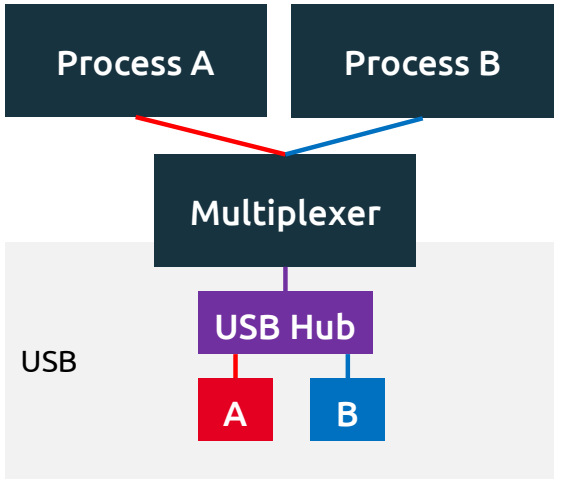
- "An optional normative xHCI feature…"
- …Not supported on MaaXBoard (Not supported *anywhere!*)

Design – Plan B: *Use VIRTIO*

- …The VIRTIO protocol does not cover xHCI (or USB)

Design – Plan C: *Blended from resources*

- Pass-through physical xHCI to Multiplexer, use thin native physical xHCI Driver
- Pass-through virtual xHCI to Process, use virtual xHCI Driver
- …Really rather difficult: Simple virtual xHCI to annex? What is the virtual protocol?



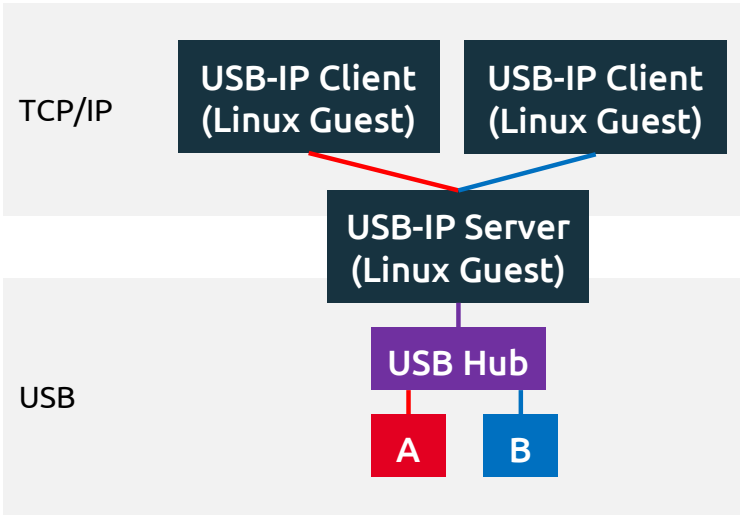
Virtualisation (USB Routing, design and pivot)

Design – Plan D: *Use Linux Kernel USB-IP*

- USB-IP supports selection and routing of USB traffic over TCP/IP
- Dependency: Virtual Networking Support (virtio-net, and virtual switch)
- …virtual switch is a work in progress

Design – Plan E: *Pivot as exercise for developer!* ☺

- Describe Linux Kernel USB-IP as means to USB Routing
- Demonstration of novel deployments that virtualisation supports



4 - Conclusions

Capgemini engineering

Conclusions

Make it easy to get started with the seL4 Microkernel:

- The seL4 Developer Kit: https://sel4devkit.github.io

Outcomes:

- Accessible seL4 Development Platform
- Suite of self-contained Activities:
  - Directly usable contributions
  - Indirectly useful development stories
- Another little pack of engineers with seL4 experience!

5 - Questions?

Capgemini engineering

About Capgemini Engineering

World leader in engineering and R&D services, Capgemini Engineering combines its broad industry knowledge and cutting-edge technologies in digital and software to support the convergence of the physical and digital worlds. Coupled with the capabilities of the rest of the Group, it helps clients to accelerate their journey towards Intelligent Industry. Capgemini Engineering has 65,000 engineer and scientist team members in over 30 countries across sectors including Aeronautics, Space, Defense, Naval, Automotive, Rail, Infrastructure & Transportation, Energy, Utilities & Chemicals, Life Sciences, Communications, Semiconductor & Electronics, Industrial & Consumer, Software & Internet.

Capgemini Engineering is an integral part of the Capgemini Group, a global business and technology transformation partner, helping organizations to accelerate their dual transition to a digital and sustainable world, while creating tangible impact for enterprises and society. It is a responsible and diverse group of 340,000 team members in more than 50 countries. With its strong over 55-year heritage, Capgemini is trusted by its clients to unlock the value of technology to address the entire breadth of their business needs. It delivers end-to-end services and solutions leveraging strengths from strategy and design to engineering, all fueled by its market leading capabilities in AI, cloud and data, combined with its deep industry expertise and partner ecosystem. The Group reported 2023 global revenues of €22.5 billion.

Get the future you want | www.capgemini.com

Capgemini engineering