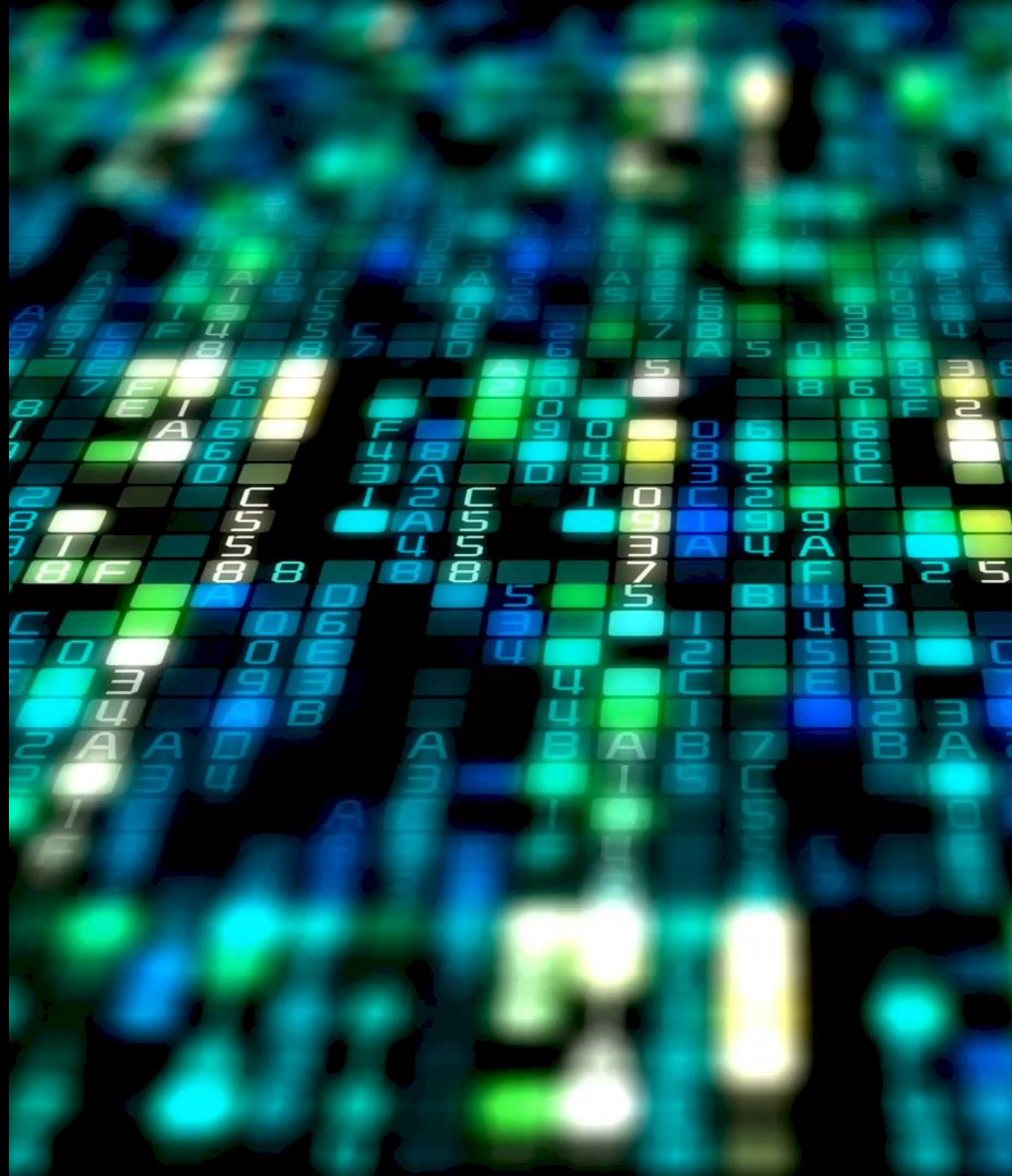
Transitioning from CAmkES VMM to Microkit VMM

An experience report

Leigha VanderKlok, Embedded Engineer II

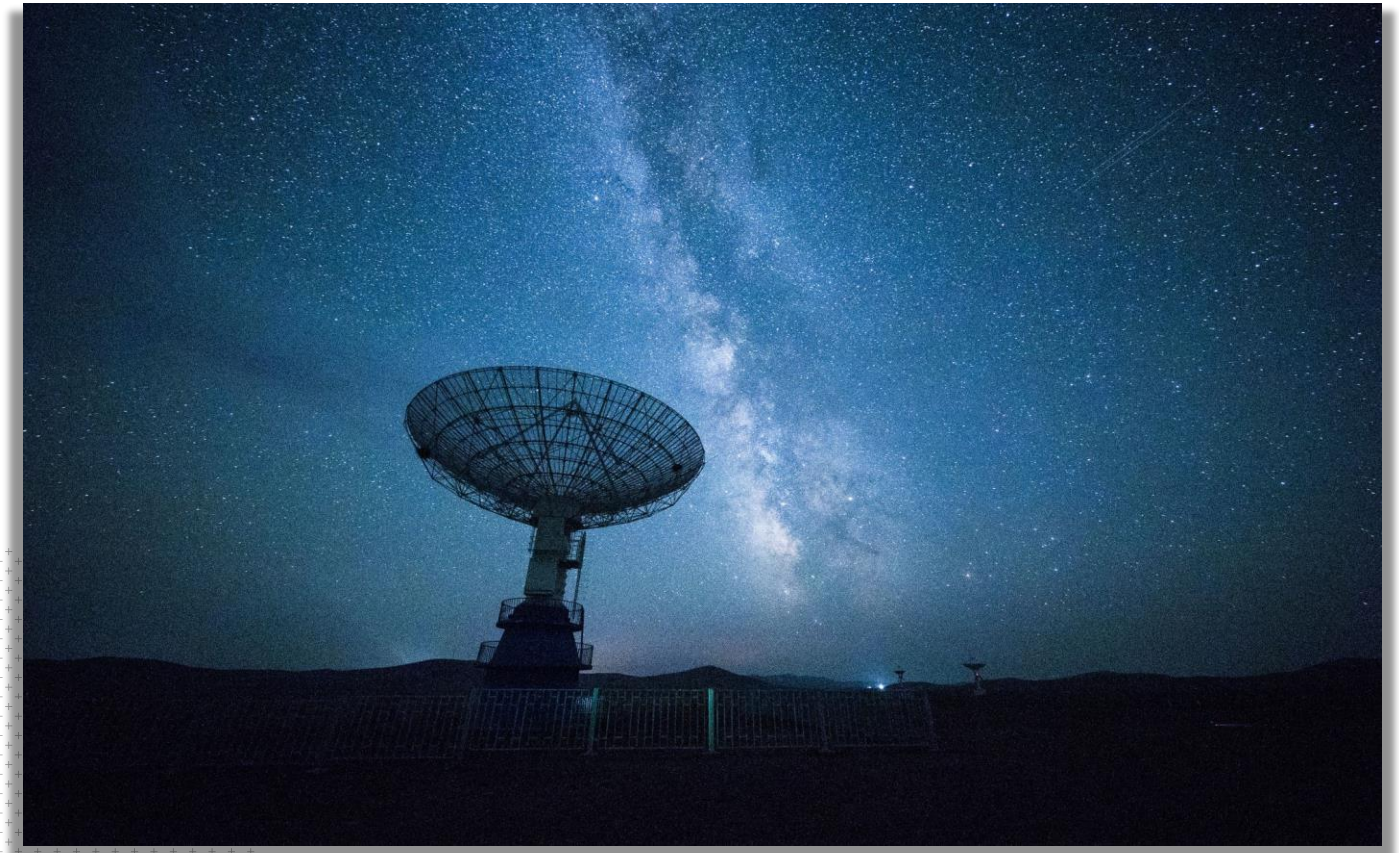
leigha.vanderklok@dornerworks.com

DORNERWORKS Michigan, USA



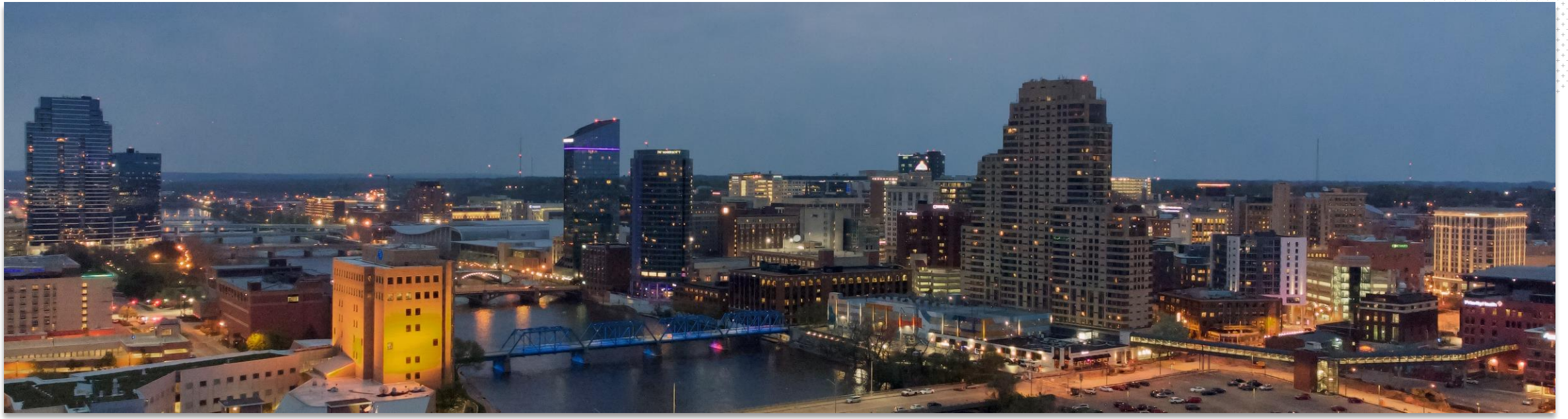
Agenda

- Reasons to use Microkit
- My experience
- Developing real world products
- How DornerWorks can help



Background

- DornerWorks has a tremendous amount of experience working with seL4 CAmkES VMM technology
  - Have done work for multiple aerospace, defense firms (NASA, Collins Aerospace, US Army, and others)
- Tested several simple Microkit virtual machine configurations
  - ZCU102 – not yet fully supported, but close
- Compared Microkit VMM to CAmkES VMM and assessed against a list of requirements



Grand Rapids, MI

Why use to Microkit VMM?

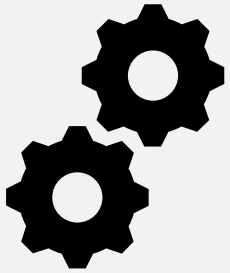
- Technical perspective
  - Integration with seL4 Device Driver Framework (sDDF)
  - Easier development process and less intimidating
    - Simpler abstractions and fewer abstractions could lower the barrier to entry
  - Stay up to date
  - Write applications in non-C languages
- Business perspective
  - Quicker on-boarding time is a plus for our customers and new engineers

- DornerWorks can provide insight for Microkit as a business use case

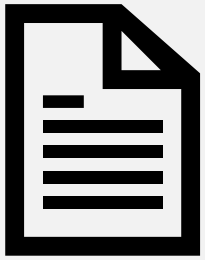
High-Level Microkit vs CAmkES

Difference	Microkit	CAmkES
Distribution	Distributed as an SDK	Source code and project files managed using repo tool.
High-level application configuration	System description file – written in xml.	Editing CAmkES attributes and settings in multiple application files.
Build system	No formal system.	CMake for file generation. Ninja is the build automation tool.
Language: Tooling and Libraries	Primarily Rust	Primarily C, parts of the toolchain are written in Python

What we like



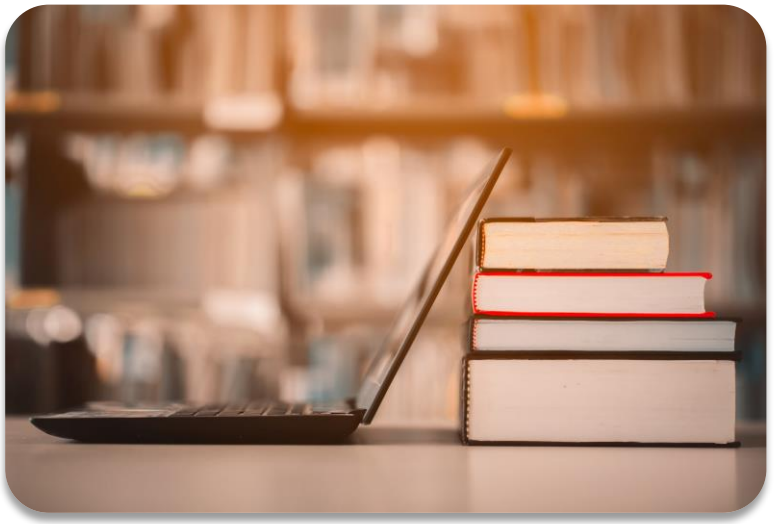
- Simpler environment setup
  - Fewer dependencies
  - Install as SDK, no need to manage over a dozen separate git repos
  - Easy to grasp abstract concepts, user friendly manual



- System reporting feature (report.txt file)
  - A snapshot of a running system, describes which entities have access to which seL4 capabilities
  - Lists physical addresses of guest RAM and ELF file regions
  - Documents that there is no overlap in guest memory between VMs

Learning experience

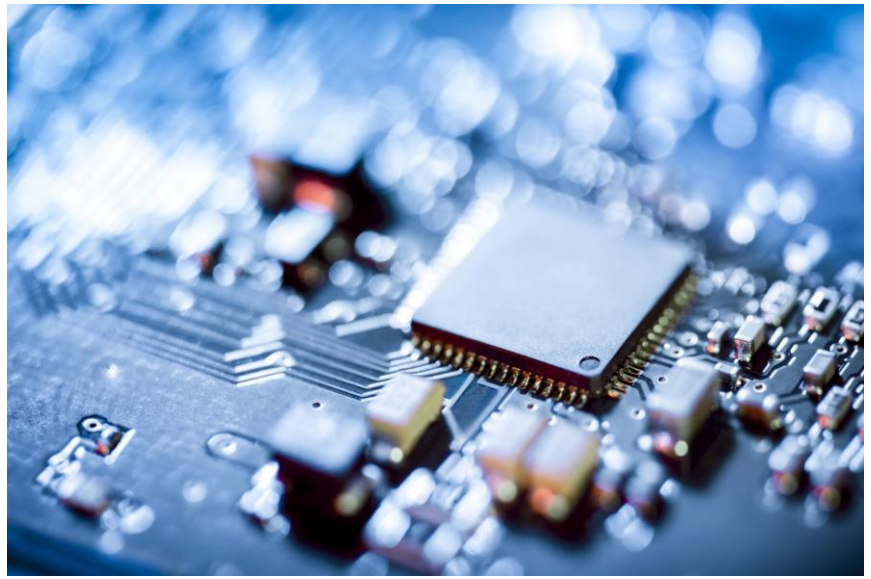
- Need to make U-Boot changes if using some Xilinx boards with Microkit
  - U-Boot platform specific for some boards (ZynqMP and Versal) override the default U-Boot behavior that drops from EL2 to EL1 when executing the image using the ‘go’ command
  - Small changes must be made to U-Boot source to work with Microkit, extra step not needed for CAmkES
- Requires more manual configuration than CAmkES VMM
  - CAmkES can have a steeper learning curve
  - But…
    - A lot more code is generated (has a build system)
    - Most configuration is done via attributes and clearly laid out settings
    - Feels like I have a menu of options to choose from and a starting place
  - With Microkit it can be cumbersome to create a new design from scratch
  - More on this later!



Developing Real World Products

- Assessed Microkit readiness level using a 28-point test which analyzed
  - ❑ Type of supported platforms
  - ❑ Ability to passthrough devices
  - ❑ CPU pinning capabilities
  - ❑ Single and multiple virtual machine configurations
  - ❑ Architecture support
  - ❑ and more

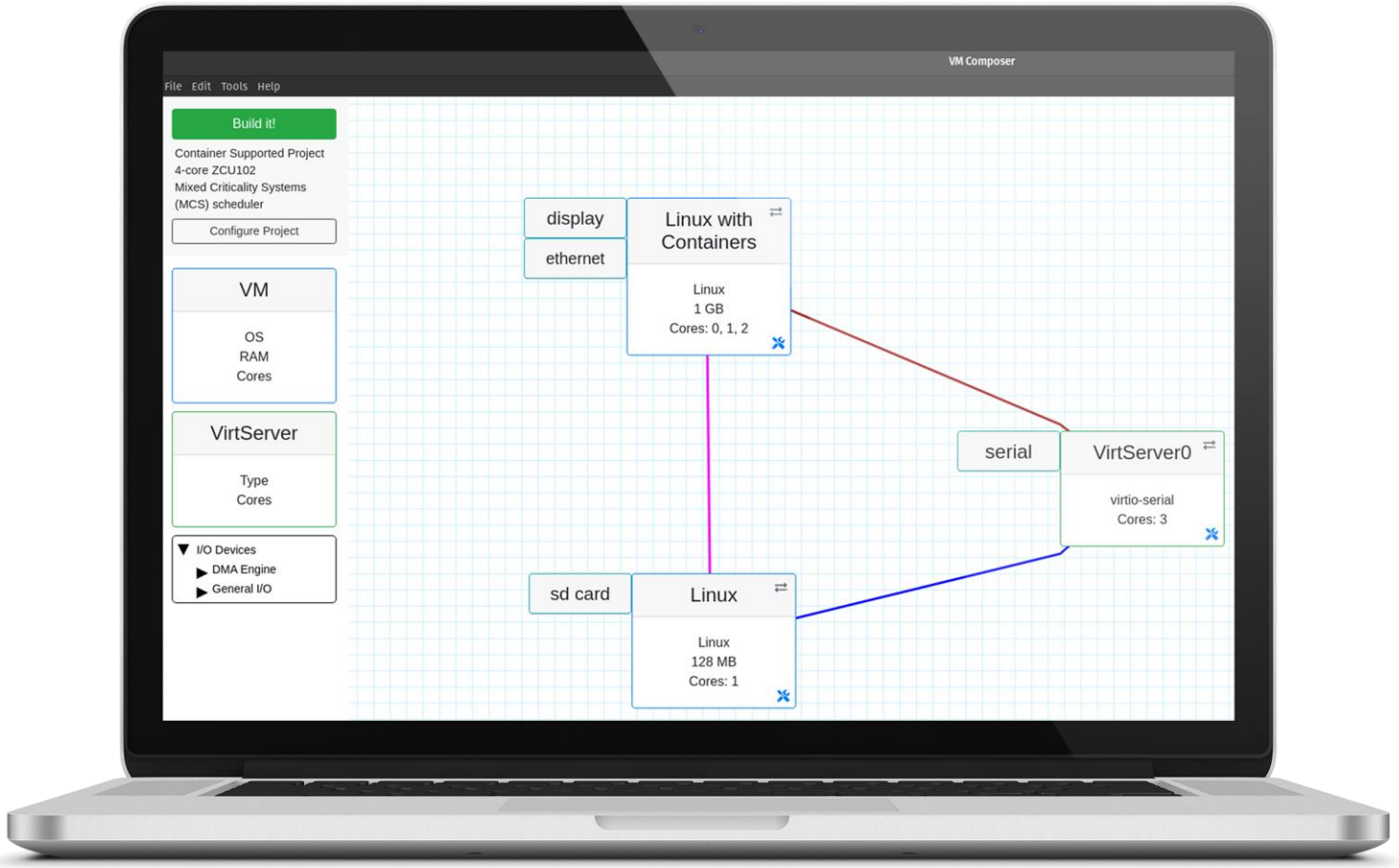
Developing Real World Products



Important to our business:

- x86 support
  - Seeing a growth of demand for seL4 on x86
  - DornerWorks has been supporting a lot of seL4 x86 development
  - Customers desire:
    - High compatibility with modern and legacy software
    - x86 is used in more high-end computing platforms, where Intel has a large market share
- Multiple vCPUs per virtual_machine instance
  - Useful to DornerWorks use case, tend to use seL4 with hypervisor support enabled
  - Need to make full use of all cores a platform has for Linux to use and run on top of seL4
  - Won't be ready to transition without this

VM Composer

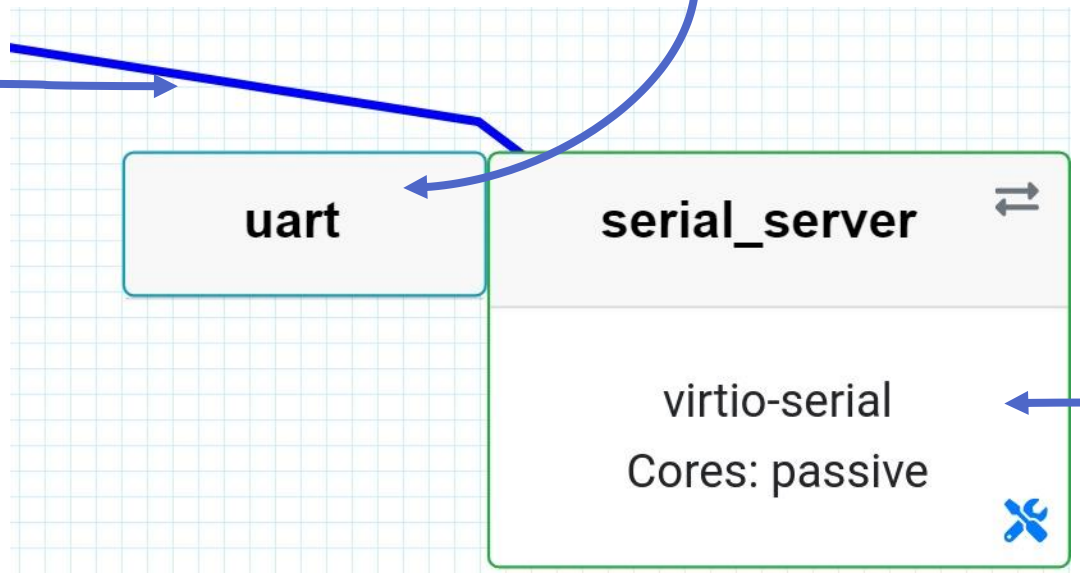


- Dilemma: Microkit can require more manual configuration than CAmkES
  - Ex; UART device passthrough: need to add memory regions, then map into a VM with the IRQ
  - CAmkES: add path to dtb attribute / hardware abstraction
  - Devs will start with known working example and then modify
- Generates all the code needed for a complete project and builds bootable images
- Currently generates CAmkES-based designs only

Theoretical Microkit VMM Example: Serial Server

```
<memory_region name="uart" size="0x1_000" phys_addr="0x9_000_000"/>
```

```
<channel>
    <end pd="client" id="1" />
    <end pd="serial_server" id="2" />
</channel>
```



- Simplify copying and pasting boilerplate code
- Learn how a project is generated
- Ease some of Microkit's inconveniences

```
<protection_domain name="serial_server" priority="254">
    <program_image path="serial_server.elf" />
    <map mr="uart" vaddr="0x2000000" perms="rw" cached="false" setvar_vaddr="uart_base_vaddr"/>
    <map mr="serial_to_client" vaddr="0x4000000" perms="wr" setvar_vaddr="serial_to_client_vaddr"/>
    <map mr="client_to_serial" vaddr="0x4001000" perms="r" setvar_vaddr="client_to_serial_vaddr"/>
    <irq irq="33" id="1" />
</protection_domain>
```

Conclusion

- DornerWorks' engineers are excited to try Microkit
- Planning to add Microkit support to VM Composer
- Still need features to make an effective business use case to our customers

- Questions?