ETH zürich

Reto Achermann

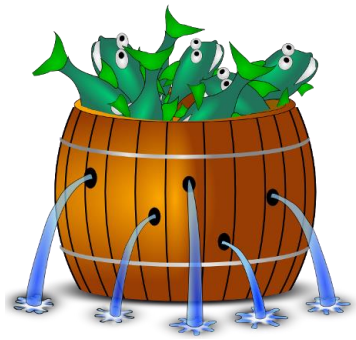
Faithful Hardware Representation and Least-Privilege Memory Management in Operating Systems

University of Texas at Austin, December 6, 2019

PhD Student, Systems Group, Department of Computer Science, ETH Zurich

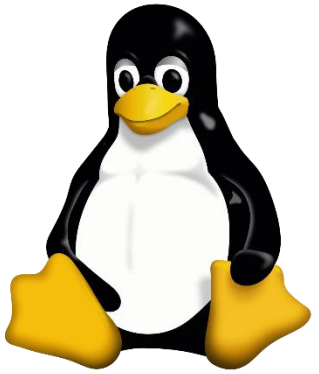
Systems@ETH zürich

My Operating Systems Coding Experiences



- Platform support: Cavium ThunderX / Xeon Phi co-processor / ARM FastModels
- Drivers: USB Stack / DMA Engines / Xeon Phi / CPU / IOMMU
- Runtimes: OpenMP / libnuma / message-passing
- Extensions to the capability system
- Domain Specific Languages

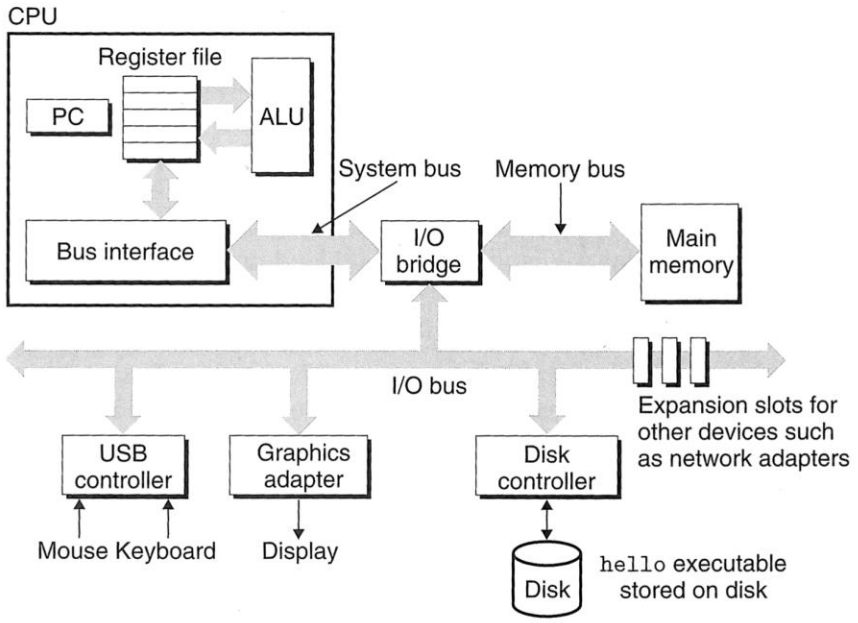
- Memory management
- Message-passing system
- Memory allocation policies
- Page-table replication
- Kernel modules

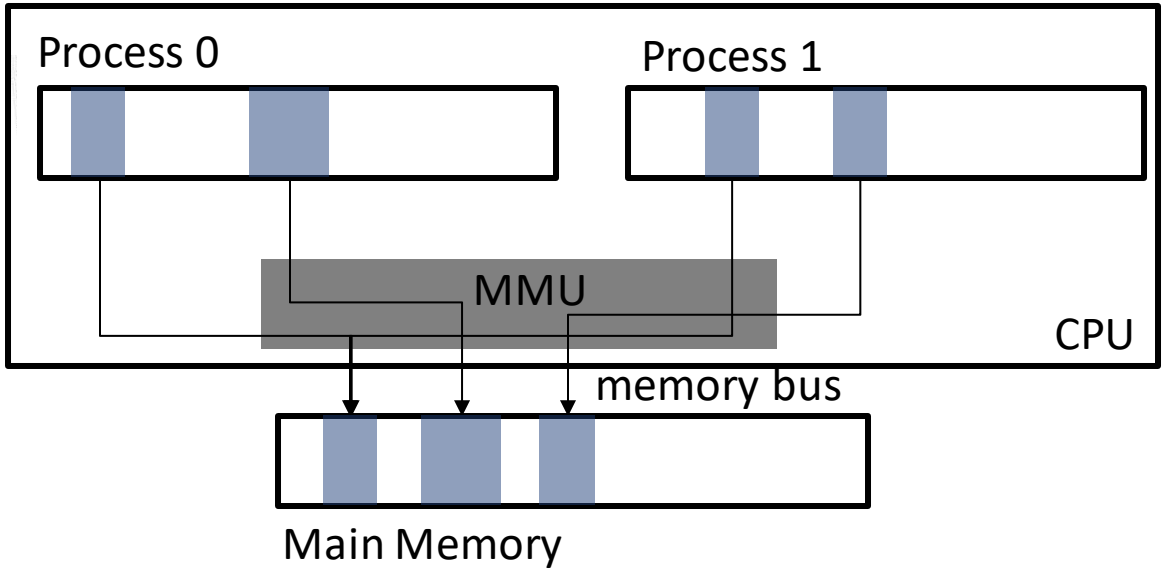


A deep **dissatisfaction** about the way operating systems abstract and represent hardware.

Computer Architecture 101

Figure 1.4
Hardware organization of a typical system. CPU: Central Processing Unit, ALU: Arithmetic/Logic Unit, PC: Program counter, USB: Universal Serial Bus.





systems, but all systems have a similar look and feel. Don't worry about the complexity of this figure just now. We will get to its various details in stages throughout the course of the book.

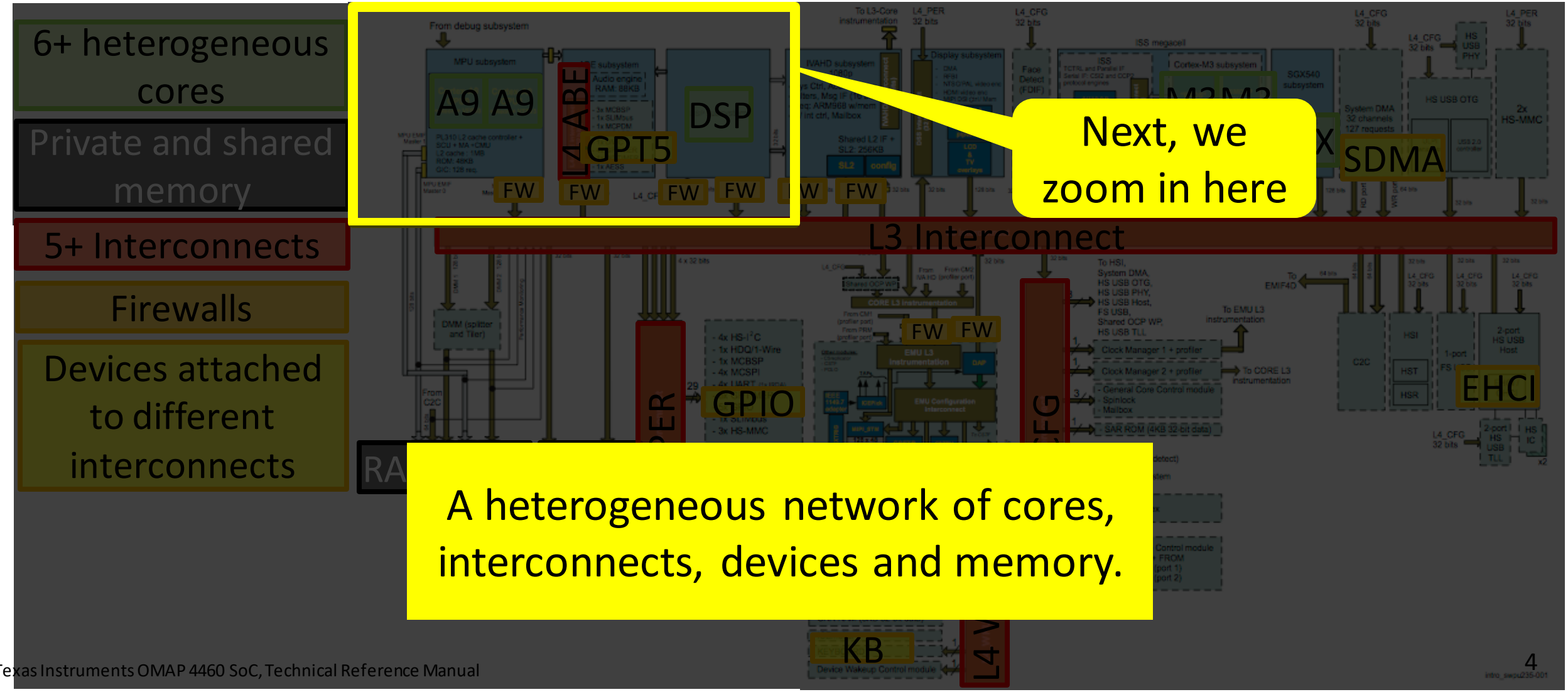
The OS just runs on a new platform

Operating System Assumptions:

- A single, flat physical address space
- physical address as a unique identifier
- Physical address means the same for all CPUs and devices

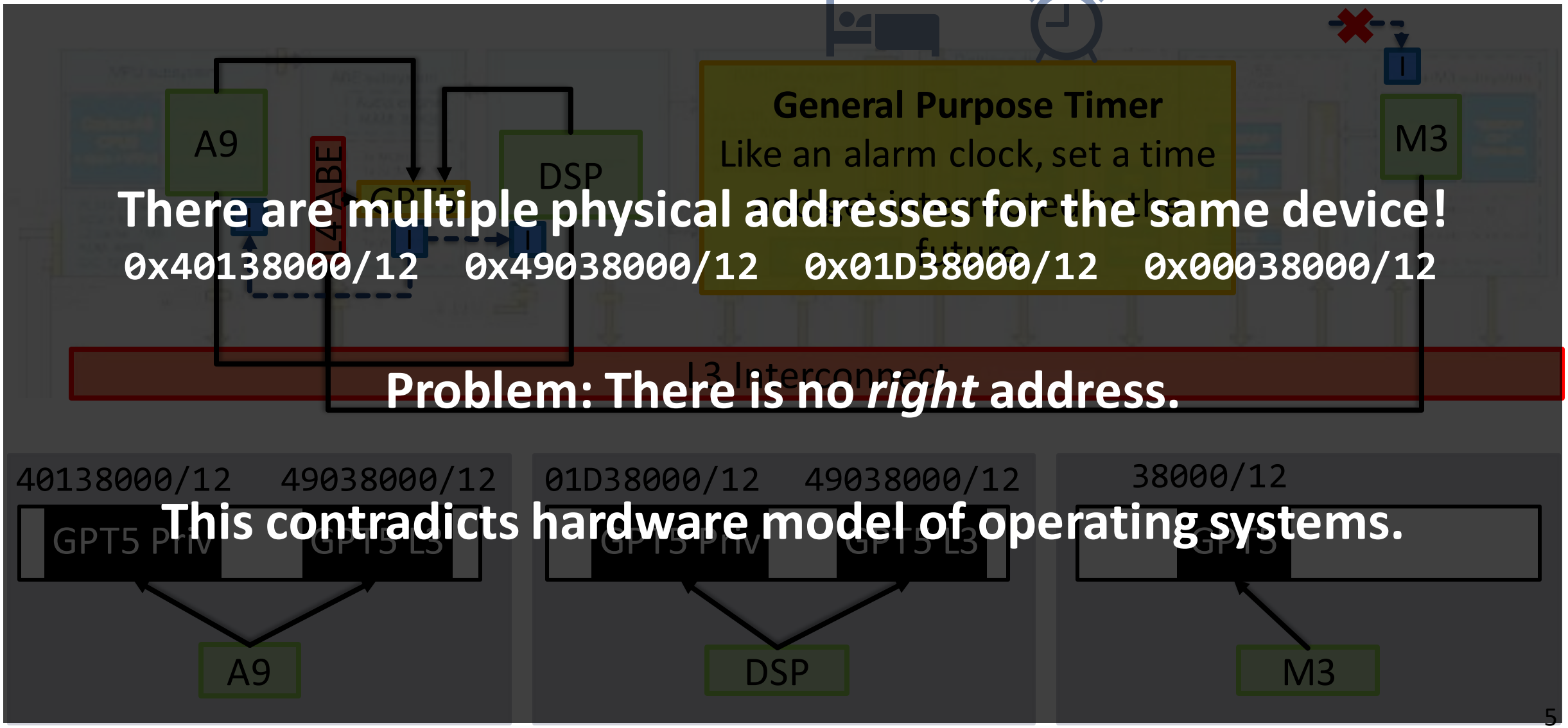
Reality: Hardware Violates the Assumptions Made by the OS

TexasInstruments OMAP4460, Q4 2011



Ambiguous Physical Addresses and Non-Uniform Views

OMAP 4460 SoC

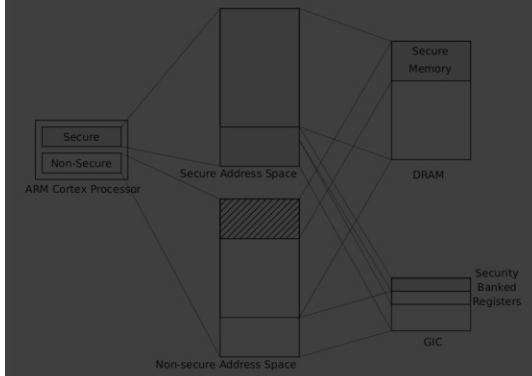


There are multiple physical addresses for the same device!

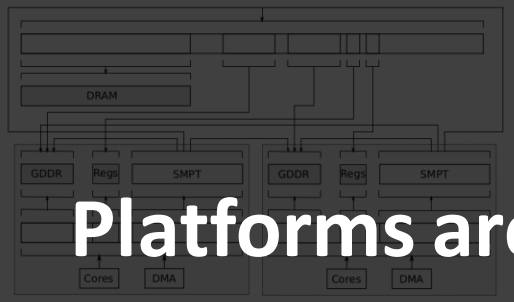
Problem: There is no *right* address.

This contradicts hardware model of operating systems.

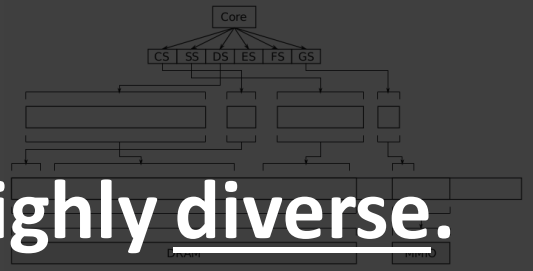
Complicated Memory Topologies are a Universal Problem



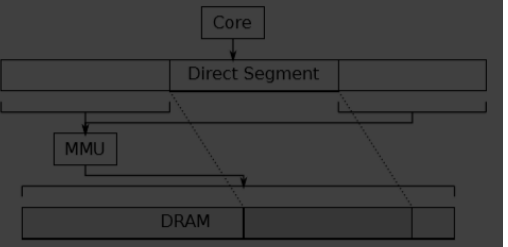
Secure and non-secure (ARM TrustZone / Intel SGX)



Co-Processors (GPU / TPU)



Segmentation (x86 / S/390)

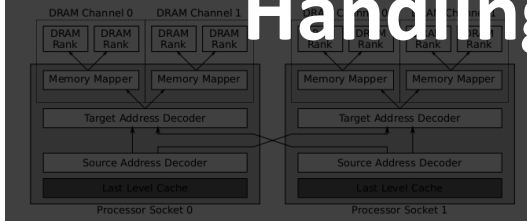


Direct Segments

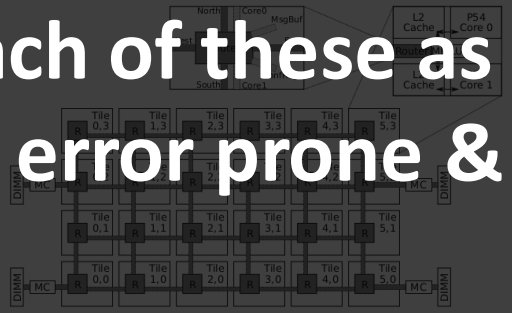
Platforms are highly diverse.

Address translations are configurable.

Handling each of these as a special case does not scale: error prone & time consuming.

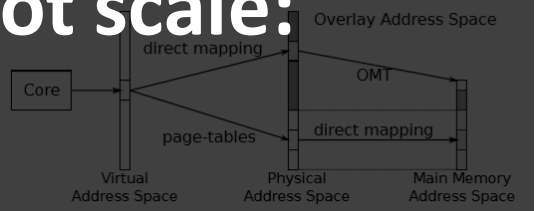


Memory Controller remappings



Intel Single Chip Cloud Computer

Fabric Attached Memory (GenZ / The Machine)



Page Overlays

Mismatch: Hardware Abstraction in OS⇔ Real Hardware

This mismatch is a **problem** – the OS does not seem to get it right

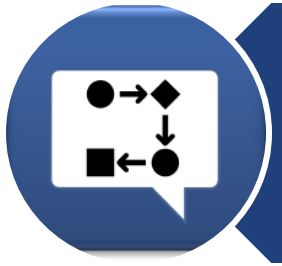
Bugs and vulnerabilities in systems software:

- CVE-2014-3601: Miscalculation of affected pages
- CVE-2016-5349: Not enough memory address information provided
- CVE-2017-8061: Wrong DMA addresses
- CVE-2017-16994: Ignoring holes in huge-pages
- CVE-2014-9888: wrong access rights for data pages
- CVE-2019-2250: authorization bug allows writing to memory locations
- CVE-2018-11994: SMMU misconfiguration allows access to memory
- 30% of patches to Linux memory manager are bugfixes

My Research Focus

My goal:
A sound basis for reasoning about reliable operating system on any hardware platform.

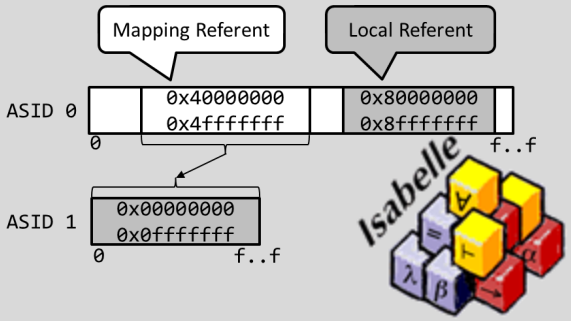
Design and Implementation of Operating Systems



Domain Specific Languages for Systems Engineering

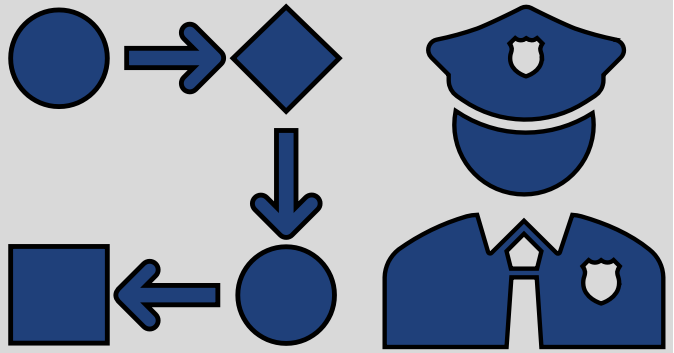
Applying Formal Methods to Operating Systems and Hardware Specification

Faithful Hardware Representation and Least-Privilege Memory Management in Operating Systems



A new model to express memory addressing on modern machines

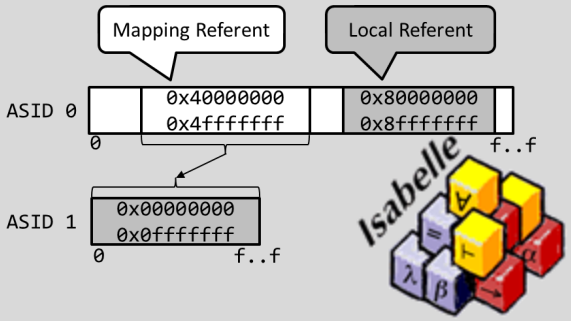
Generate OS code and hardware configuration



Express changes of the configuration and the required authority

Efficient Implementation in an Operating System

Faithful Hardware Representation and Least-Privilege Memory Management in Operating Systems



A new model to express memory addressing on modern machines

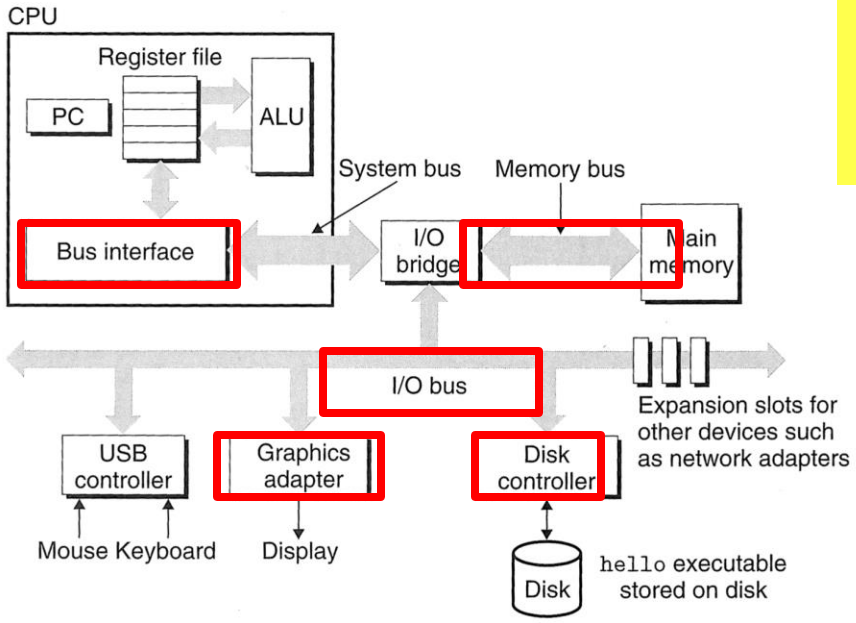
Generate OS code and hardware configuration

Express changes of the configuration and the required authority

Efficient Implementation in an Operating System

A More Realistic View of the Platform

Figure 1.4
Hardware organization of a typical system. CPU: Central Processing Unit, ALU: Arithmetic/Logic Unit, PC: Program counter, USB: Universal Serial Bus.



The OS needs to be ported:
- Requires engineering effort
- Manual process: source of bugs (more engineering effort)

Configurable I/O bus:
- PCI bridge programming
- PCI hot-plug
- I/O MMUs / System MMUs
- Virtualization

Interconnects:
- Configurable, multi-stage translations
- Configurable Firewalls
- Private access ports

Processors
- Core-Local resources
- Cache Hierarchy, operation modes
- Configurable Multi-Stage Translation
- Virtualization

Devices
- Memory access restrictions
- Configurable Translations
- Self-virtualization

The OS must **correctly** configure and manage these

Specification and Modeling Hardware

- Industry Standards: DeviceTrees / UEFI / ACPI / USB / PCI Express
  - Limited topology information, not available everywhere
  - Assumptions: a uniform view & unique physical addresses

➔ Insufficient for accurate hardware representation

- Memory & Processor Models: ARM's ASL and Sewell et al.
  - Model the behavior of instructions and memory requests
  - Stop at processor boundaries

➔ Complimentary problem orthogonal to address translation

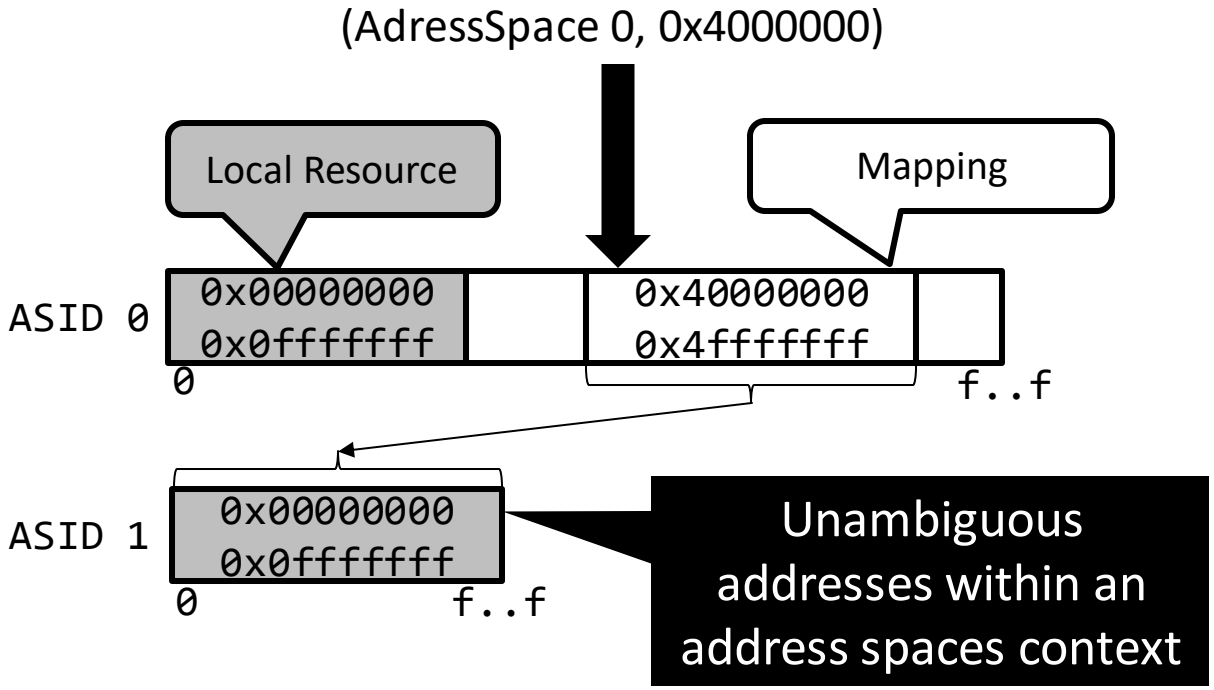
- Verified Operating Systems: seL4, CertiKOS
  - Proofs based on a linear flat array to physical memory

➔ Proofs need an accurate hardware representation

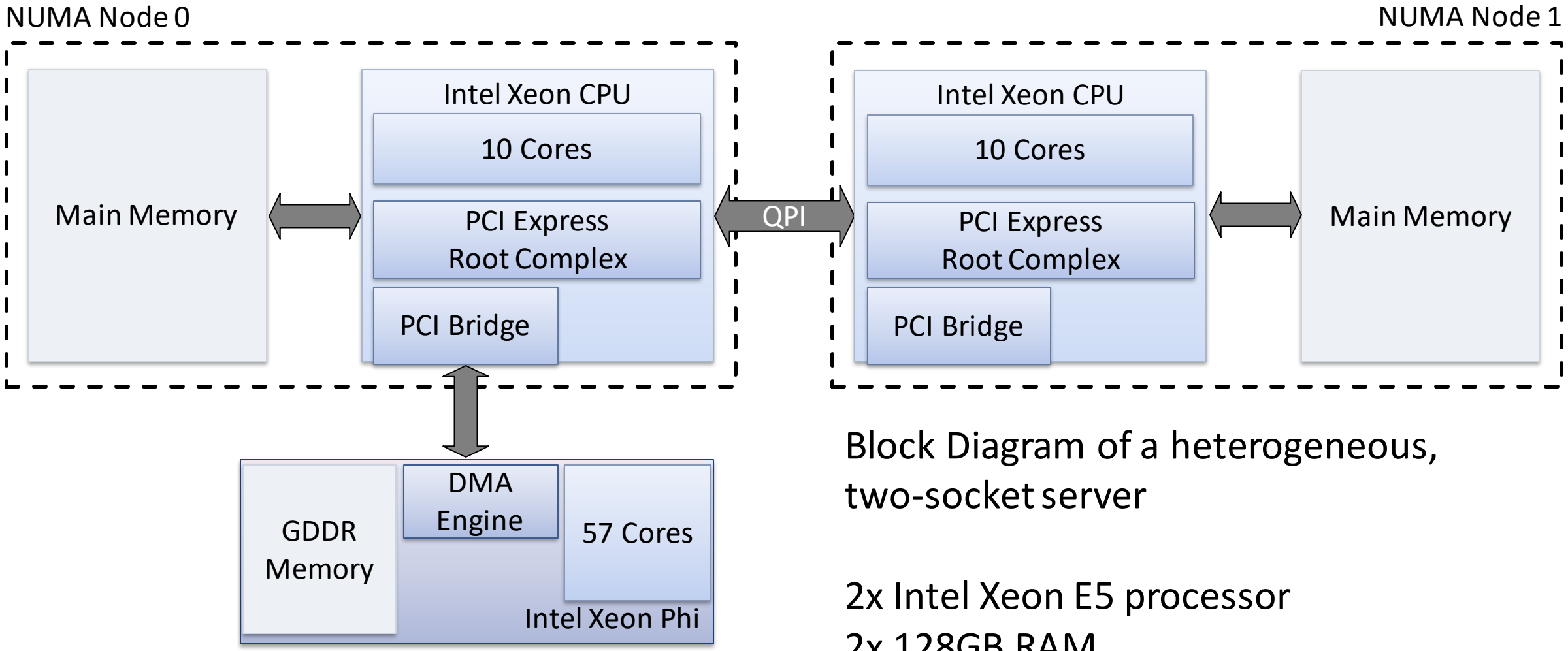
The Address Space Abstraction – A More Faithful Representation

- Memory management needs an unambiguous reference to physical resources.

- Address Space (**Naming Problem**)
  - Context for resolving addresses
  - Range of address values e.g. [0, 2^b)
  - Regions of address mappings
  - Regions of local resources



New Address Space:
- Ambiguous addresses
- different interpretation of addresses

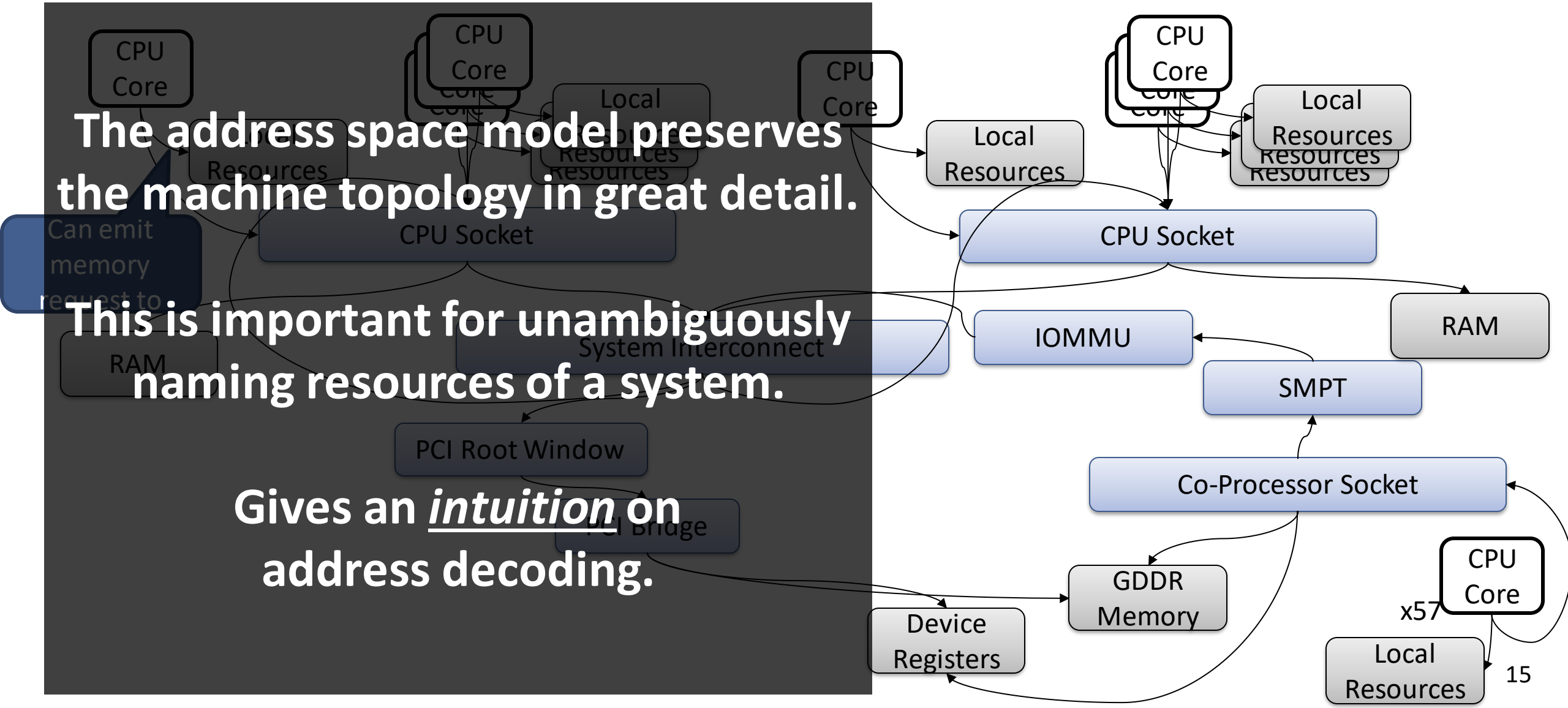


Block Diagram of a heterogeneous, two-socket server

2x Intel Xeon E5 processor
2x 128GB RAM

Intel Xeon Phi Co-Processor attached to PCI Express

Decomposing a Heterogeneous Server into Address Spaces



The address space model preserves the machine topology in great detail.

This is important for unambiguously naming resources of a system.

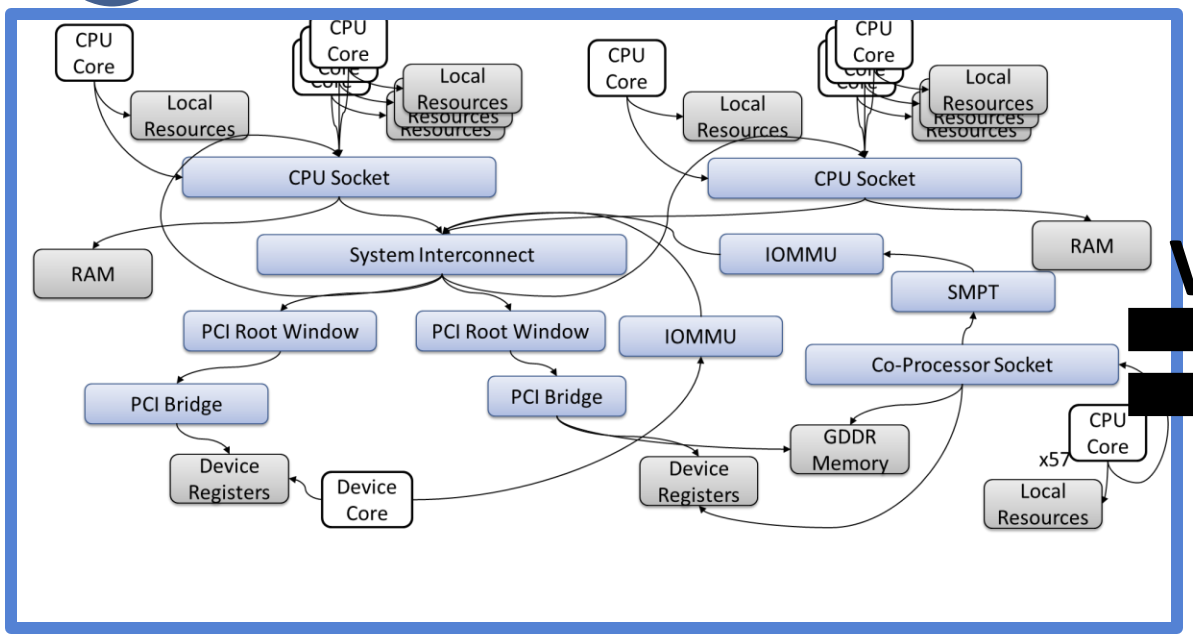
Gives an *intuition* on address decoding.

Applying Formal Methods to the Address Space Model

- Formalization of the model in Isabelle/HOL
  [Decoding Net, MARS'17 / ITP'18]
- Well-defined **semantics** of address resolution termination proofs, …
- Verification of **algorithms** on top of the model
- Capture the **current, static** state of the system

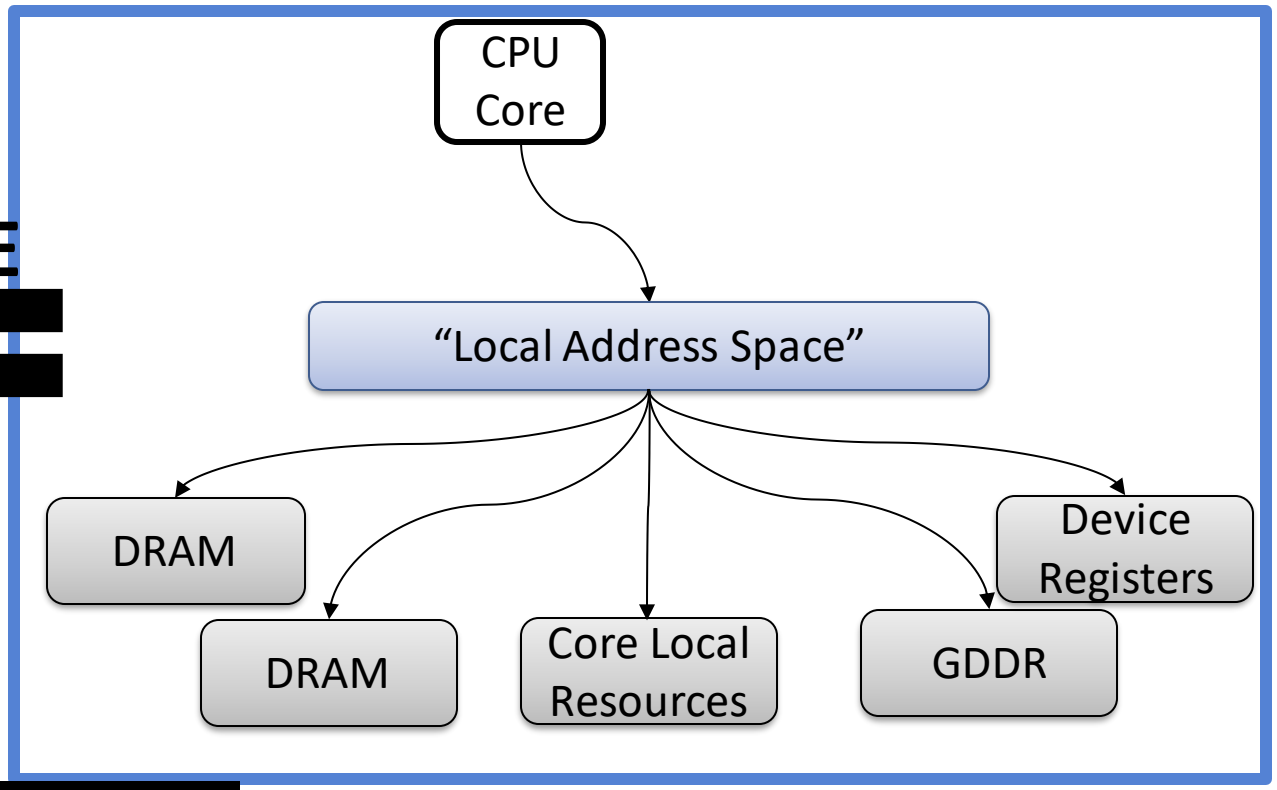
Sound foundation to express address translation of real hardware (e.g. TLB models)

“What is reachable from a core and at which address?”



VE

Normal Form

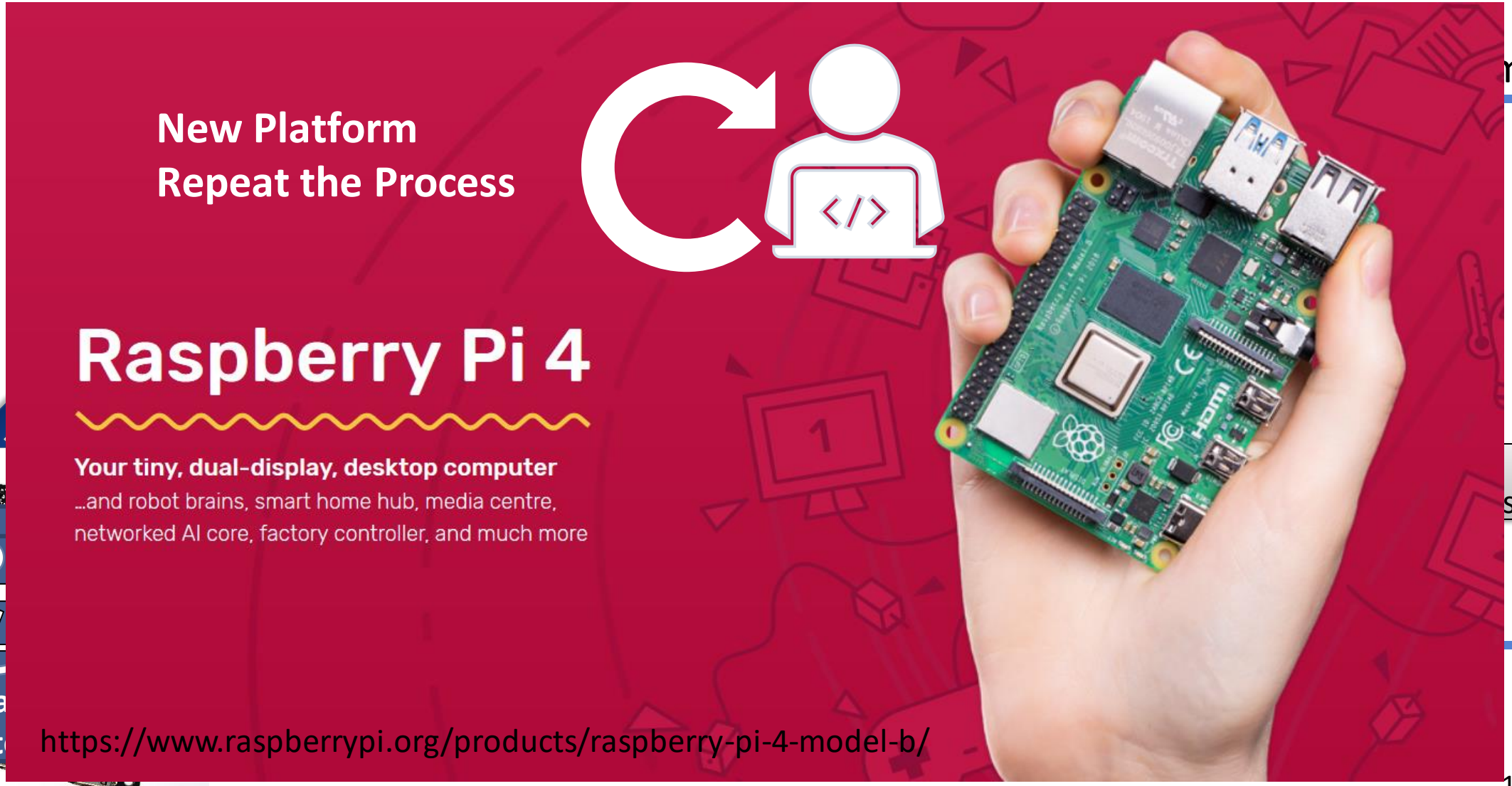


Proof of view-equivalence

Algorithms produce provably to the correct values. Fewer bugs in the OS.

Unreachable Resources

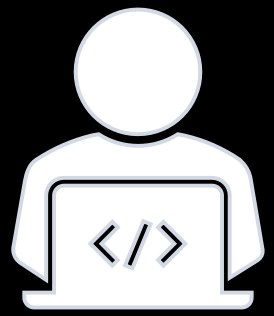
Using the Correct Output to write Platform Specific OS Code



Many new SoC-Platforms are released every year

System-on-a-Chip Platform Vendors

Actions Semiconductor, Advanced Micro Devices (AMD), Advanced Semiconductor Engineering (ASE), Aeroflex Gaisler, Agate Logic, Alchip, Allwinner Technology, Altera, Amkor Technology, Amlogic, Analog Devices, Anyka, Apple Inc., Applied Micro Circuits Corporation (AMCC), ARM Holdings, ASIX Electronics, Atheros, Atmel, Axis Communications, Broadcom, Cambridge Silicon Radio, Cavium Networks, CEVA, Inc., Cirrus Logic, Conexant, Core Logic, Coronis (Wavenis Technology), Cortina Systems, CPU Tech, Cypress Semiconductor, FameG (Fulhua Microelectronics Corp.), Freescale Semiconductor, Frontier Silicon Ltd, Fujifilm, HiSilicon, Horizon Semiconductors, Imagination Technologies, Infineon Technologies, Innova Card, Intel Corporation, InvenSense, Lattice Semiconductor, Leadcore Technology, LSI Corporation, Marvell Technology Group, MediaTek, Maxim Integrated Products, Milkymist, MIPS Technologies, Mistletoe Technologies, MosChip Semiconductor Technology, MStar Semiconductor, Naksha Technologies, Nokia, NuCORE Technology, Nufront, NVIDIA, NXP Semiconductors (formerly Philips Semiconductors), Corporation, PMC-Sier Scorpion, Redpine Sign Sequence Design, Shar Silicon Motion, Skywor SolidRun, Spreadtrum, Sunplus Technology, Sy Tensilica, Teridian Semi TranSwitch, Vimicro, RDA Microelectronics



Each platform is different.
Operating system needs to be adapted every time.
Manual porting:
- time effort
- source of bugs…

SoC Released 2018

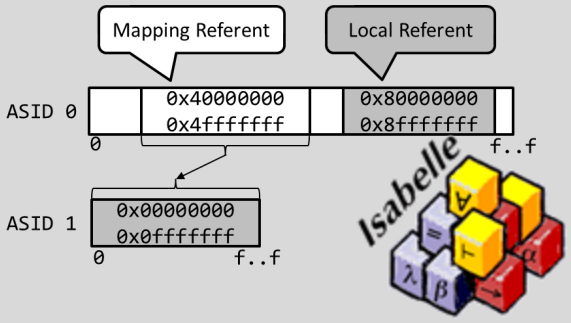
Apple
A12, S4, W3

Samsung
Exynos 9 Series, Exynos 7 Octa, Exynos 5 Hexa

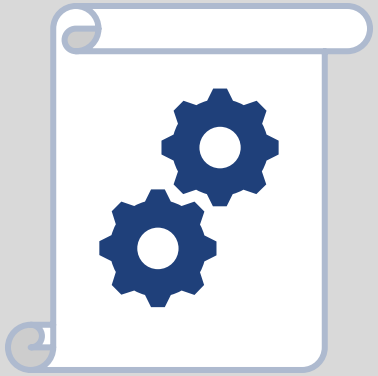
QualComm
SDM439, SDM429, SDM632, SDM670, SDM710, SDM845, QCC5120

T6755S, Helio
Helio P70

Faithful Hardware Representation and Least-Privilege Memory Management in Operating Systems



A new model to express memory addressing on modern machines

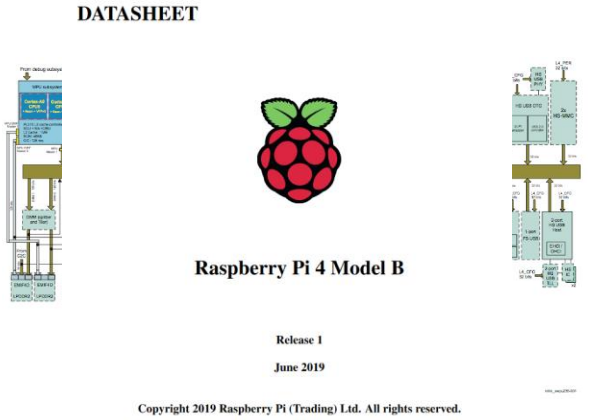


Generate OS code and hardware configuration

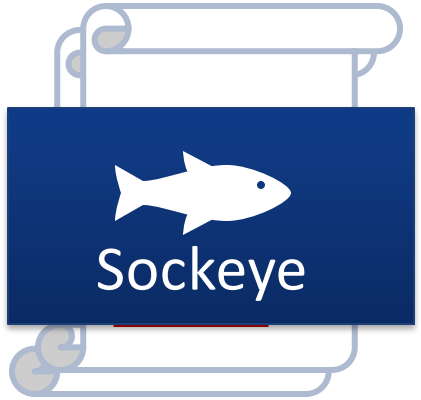
Express changes of the configuration and the required authority

Efficient Implementation in an Operating System

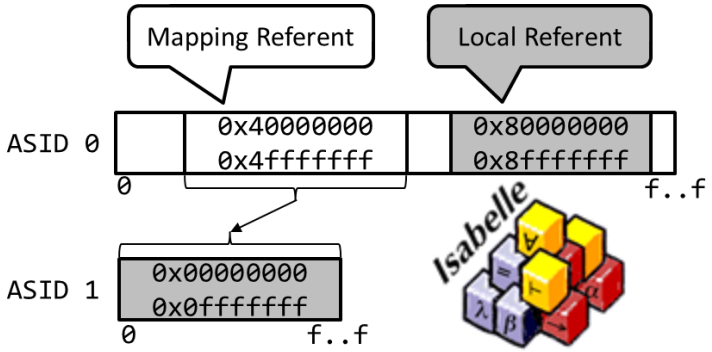
Automatic Generation of OS Code From the Model



Vendor supplied data (e.g. Hardware Manual)



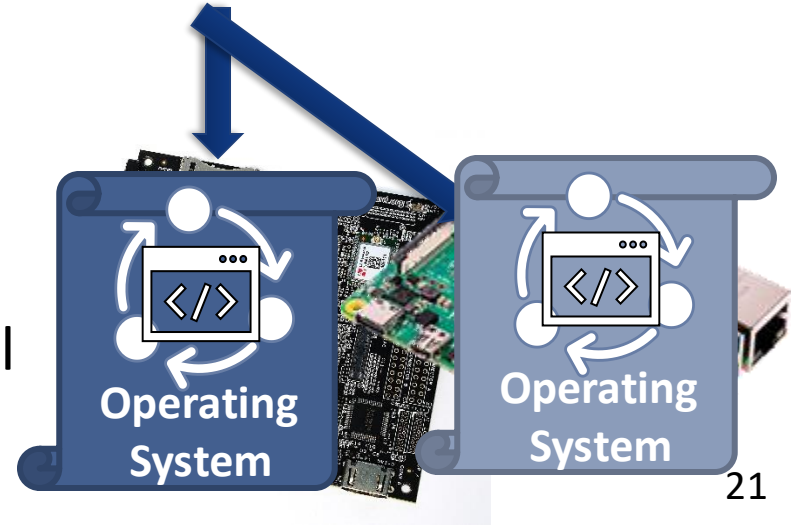
Machine readable description of the platform



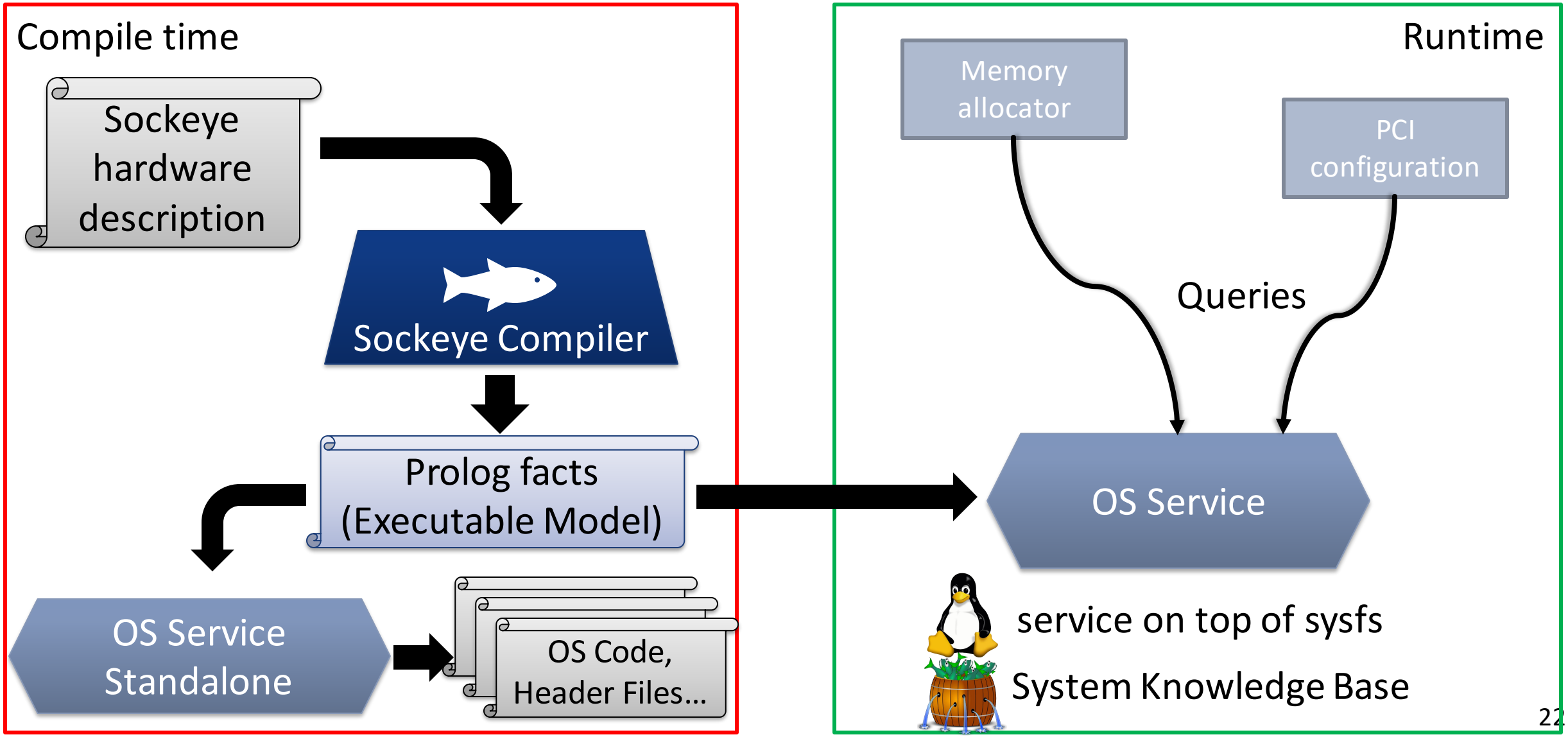
Executable representation of the model + algorithms

Sockeye Domain Specific Language for Hardware Descriptions

Generation of correct low-level OS code

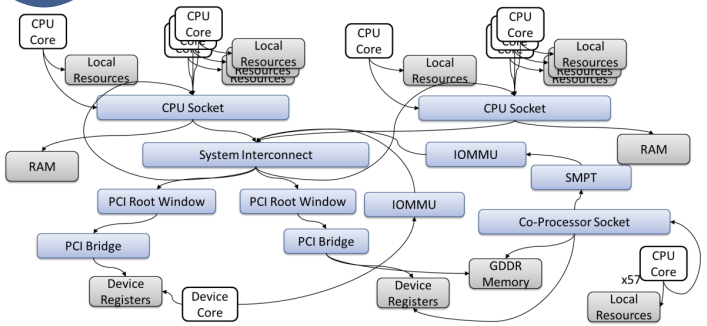


Toolchain


Compile time
Sockeye hardware description
Sockeye Compiler
Prolog facts (Executable Model)
OS Service Standalone
OS Code, Header Files…
Runtime
Memory allocator
PCI configuration
Queries
OS Service
service on top of sysfs
System Knowledge Base

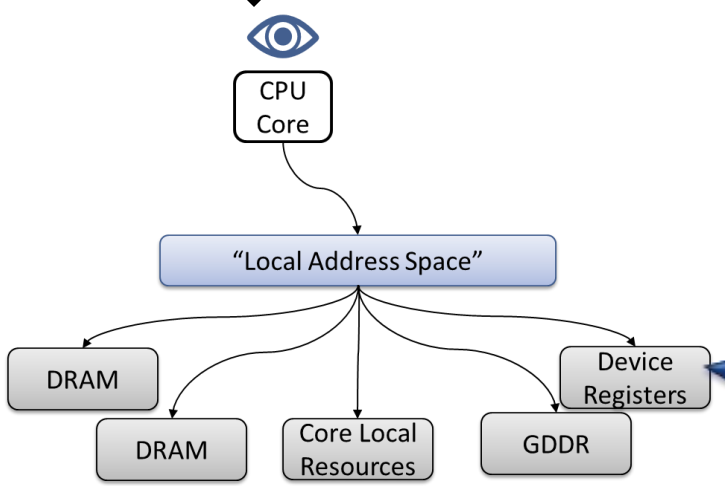
Use Case Example: Correct-by-Construction Page-Table Generation

Specify the observing core



Topology as generated by Sockeye

Flatten the decoding net using view-preserving operations

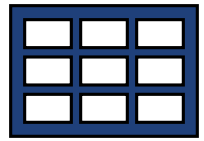


ARMv7, ARMv8, x86_64, K1OM

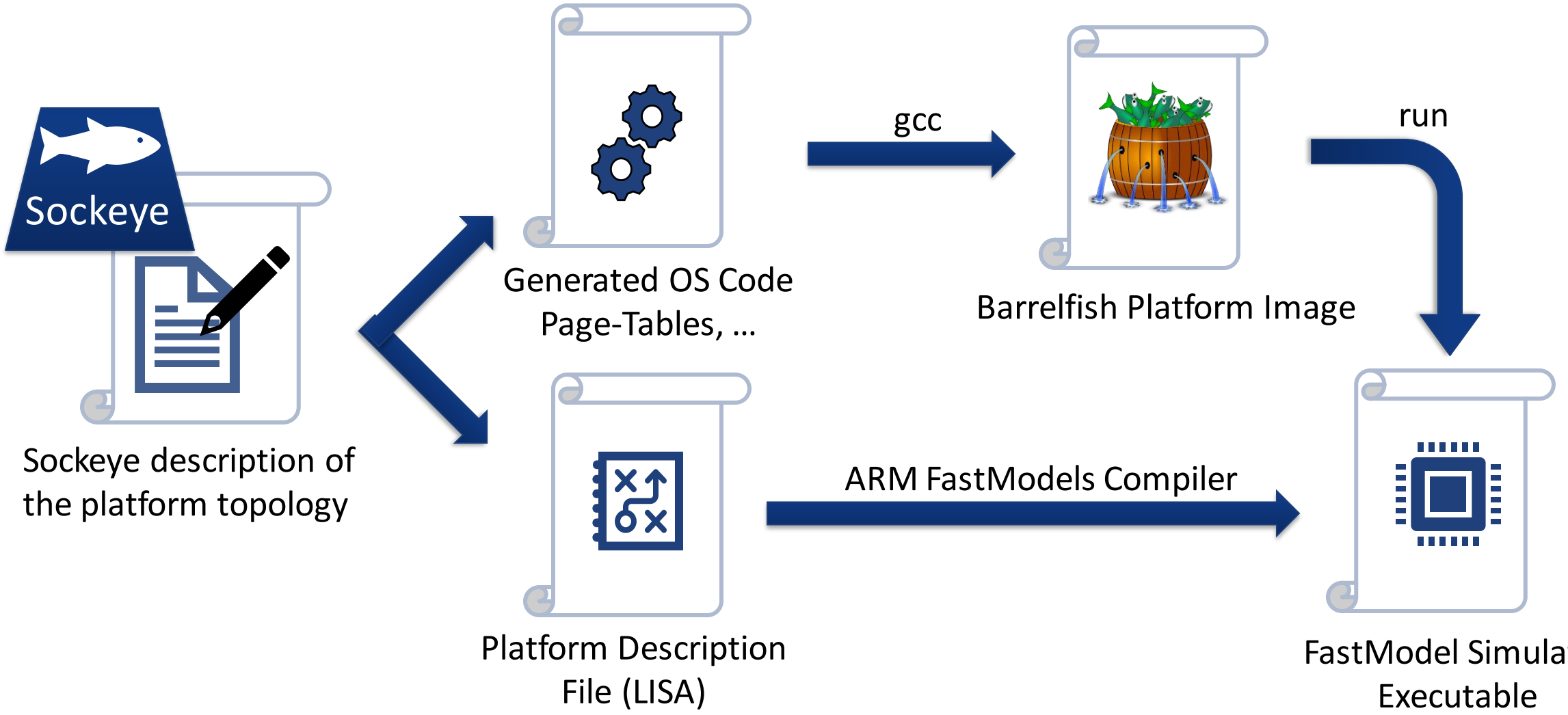
Page Table Receipt
Architecture = [ARMv8]
Mappings = [
DRAM0 @ 0x8000000,
DevRegs @ 0x10000000
]

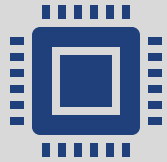
Generate page-tables based on the flattened representation

Page-Table Binary

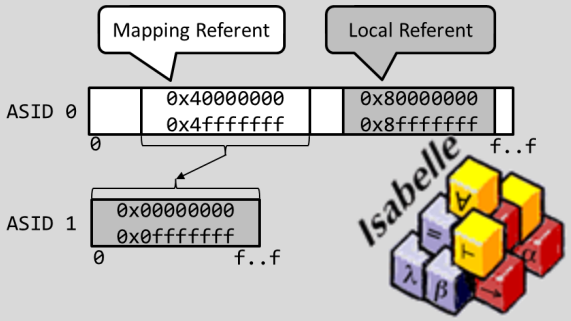


Validation: Custom Simulation Platforms for ARM FastModels

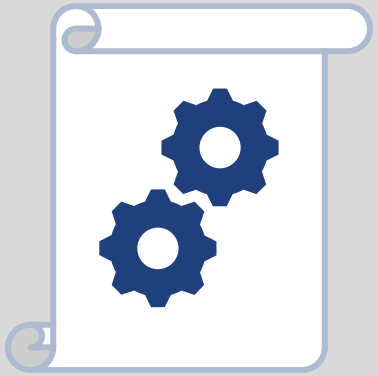




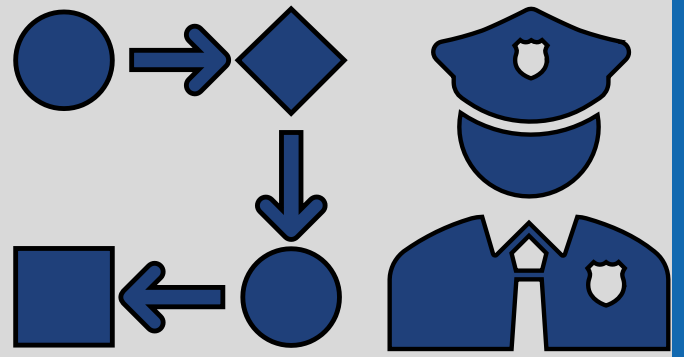
Faithful Hardware Representation and Least-Privilege Memory Management in Operating Systems



A new model to express memory addressing on modern machines



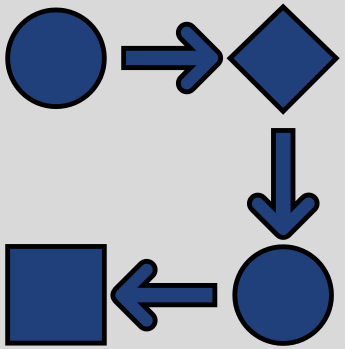
Generate OS code and hardware configuration



Express changes of the configuration and the required authority

Efficient Implementation in an Operating System

Dynamic Configuration of the Decoding Net



Semantics of **dynamic** configuration of the **translate** function

Rights and authority required to **change** the translate function

Principle of Least Privilege

From the Static Model to a Dynamic Implementation in an OS

1. Identification of all relevant **objects** and **subjects** and their relationship

2. Development of an **executable specification** for rapid prototyping

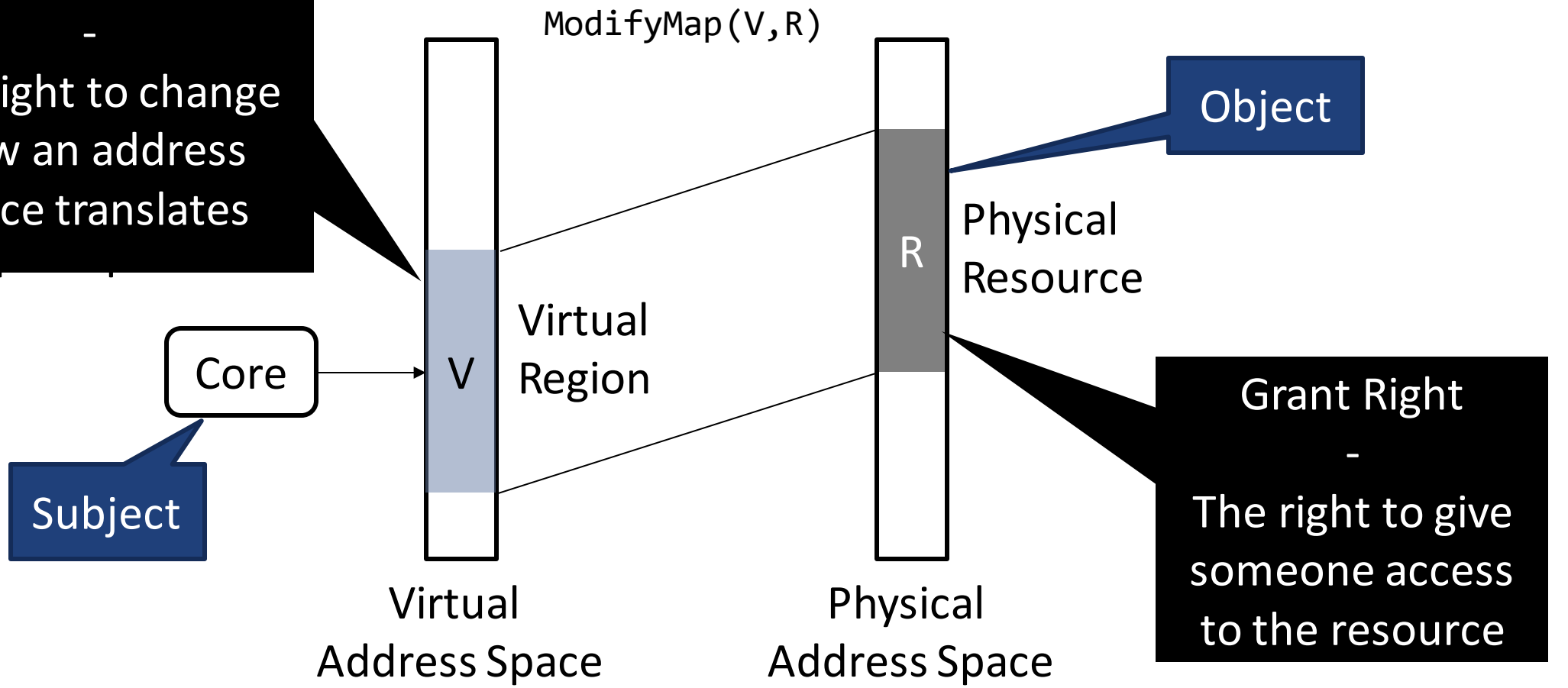
3. Executable specification as a guide for an efficient **OS implementation**

Fine-Grained Decomposition of Rights and Operations

Linux:
mmap()
Kernel h

What ha
least pri

Map Right
-
The right to change how an address space translates

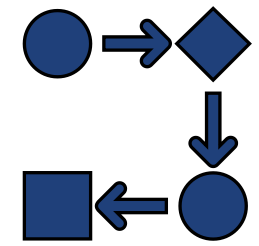


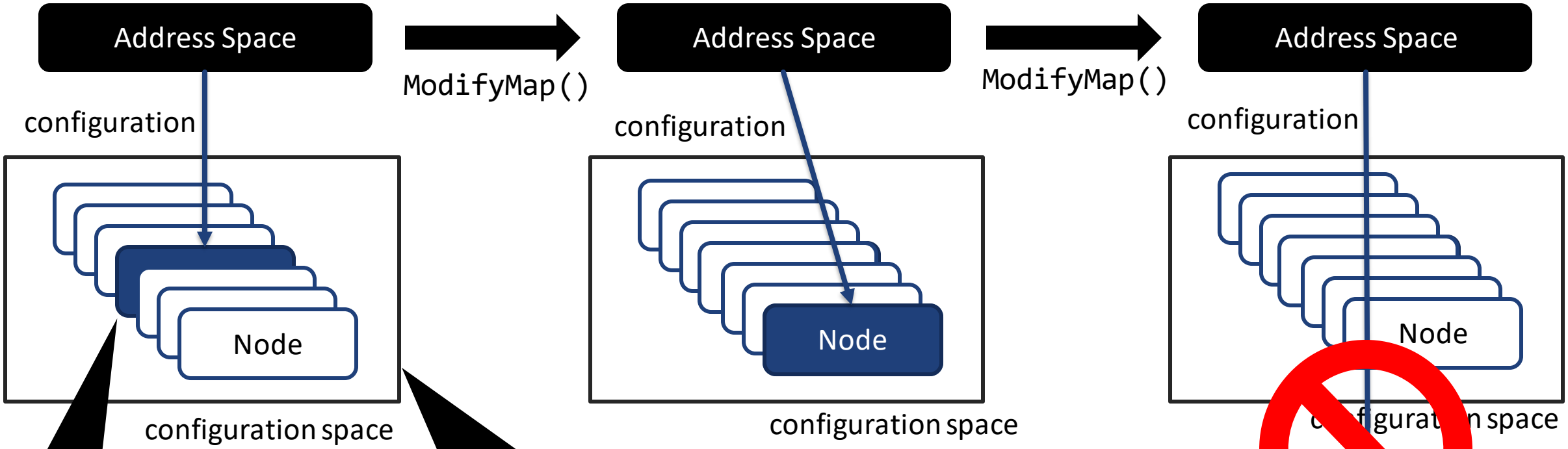
Expressing Fine-Grained Authority in a System

Objects / Subjects	DRAM Region	Core Address Space	…
…			
Process	Grant	Map	
…			

The access control matrix defines what **address space configurations** and **transitions** are **valid**.

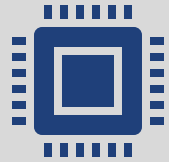
Dynamic Address Space Configuration



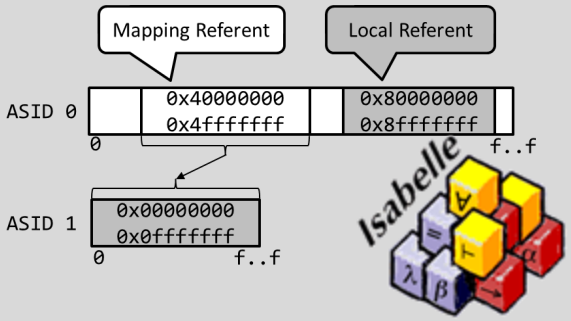


From the Static Model to a Dynamic Implementation in an OS

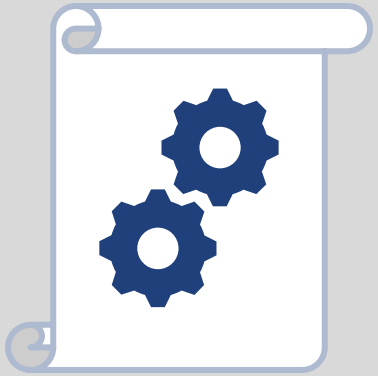
- ✓ Identification of all relevant **objects** and **subjects** and their relationship
- Development of an **executable specification** for rapid prototyping
- Executable specification as a guide for an efficient **OS implementation**
  - Barrelfish with Multiple Address Spaces + Least-Privilege
  - Access control with Capabilities as a natural match for least privilege
  - Support for heterogeneous platforms



Faithful Hardware Representation and Least-Privilege Memory Management in Operating Systems



A new model to express memory addressing on modern machines



Generate OS code and hardware configuration

Express changes of the configuration and the required authority

Efficient Implementation in an Operating System

Evaluation

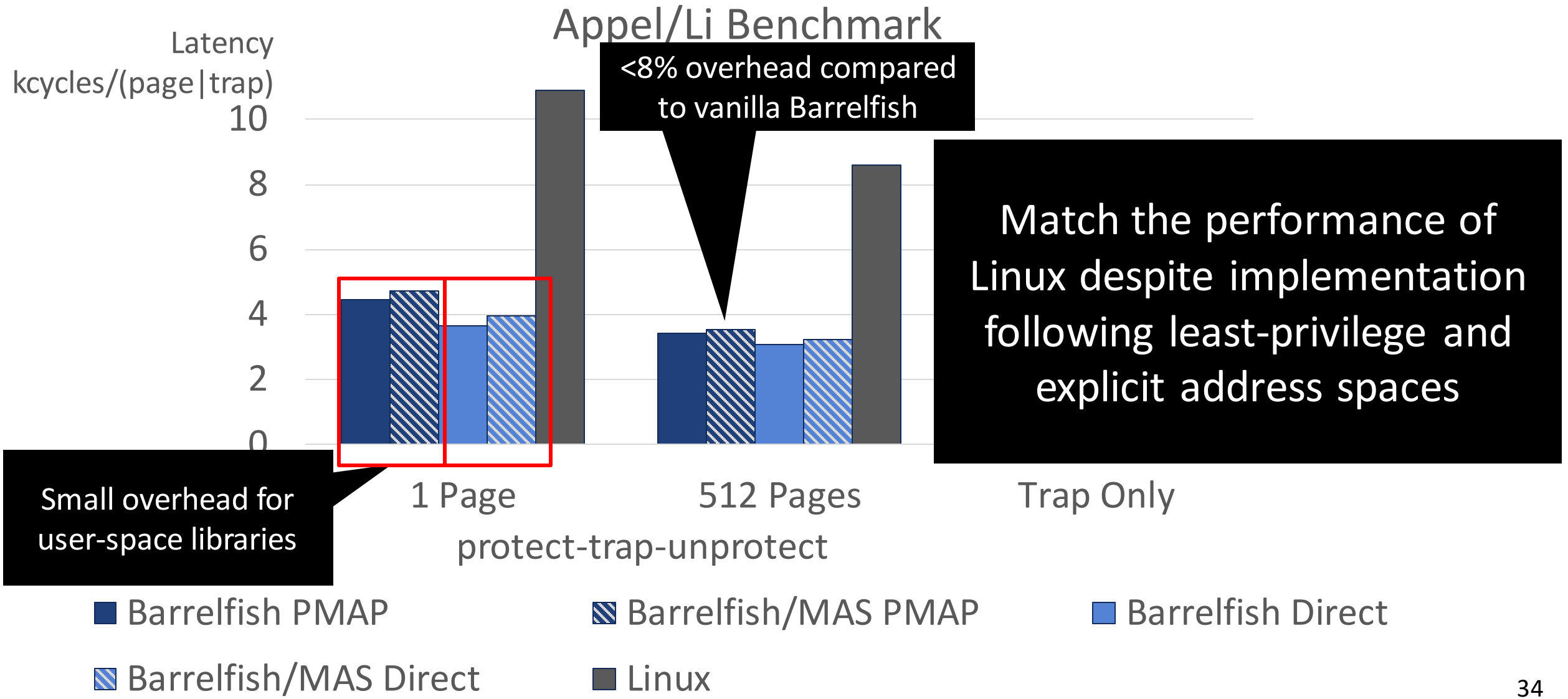
Multiple address spaces & least-privilege access control.

What's the cost of implementing this in an operating system?

1) What is the cost of memory management operations?

2) What is the overhead for dynamic address space configuration following the least-privilege principle?

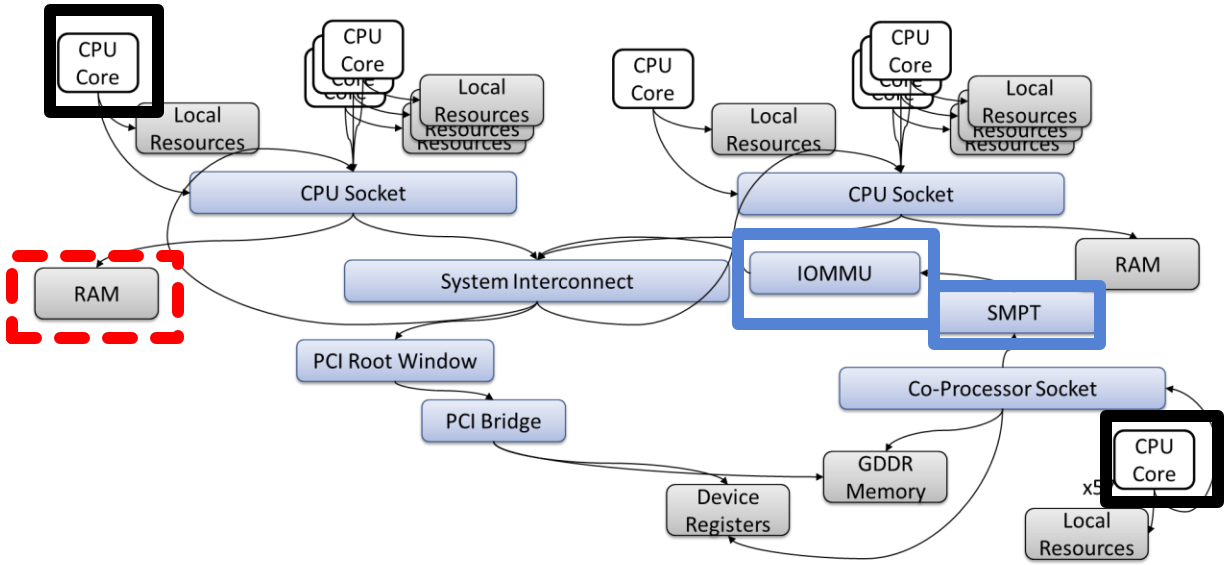
Evaluation of virtual memory management operations



The Cost of Dynamic Reconfiguration with Least-Privilege

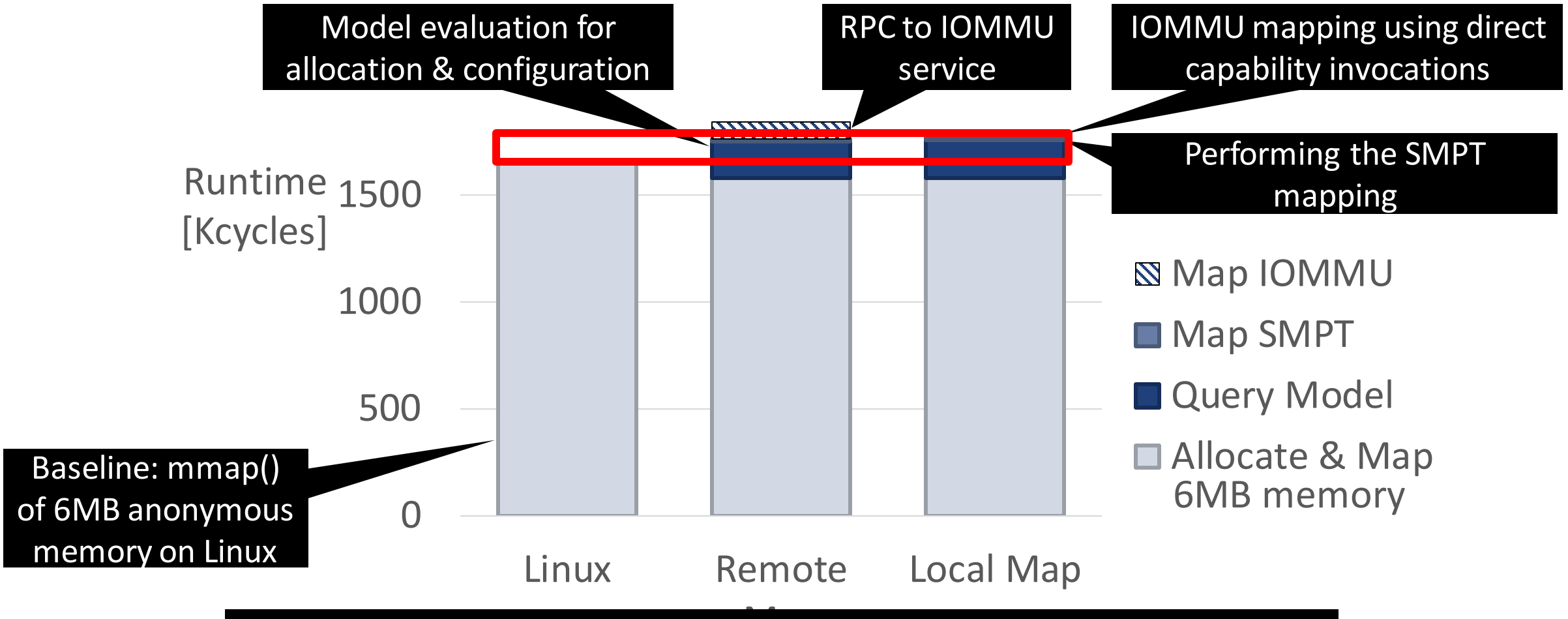
Task: Setup a shared buffer between the host CPU and the co-processor

1) Invoke model to obtain
   - memory resources to allocate
   - list of address spaces to configure
2) Allocation & mapping of memory
3) Configuration of address translation



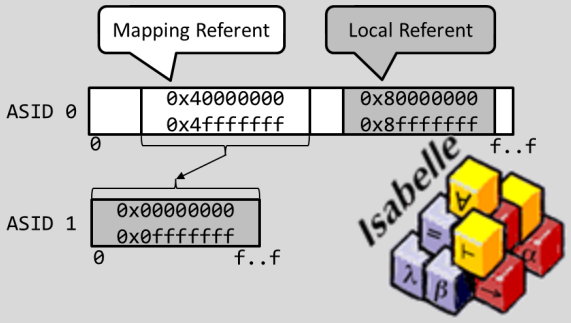
Two-Socket server with Xeon Phi Co-Processor

The Cost of Dynamic Reconfiguration with Least-Privilege

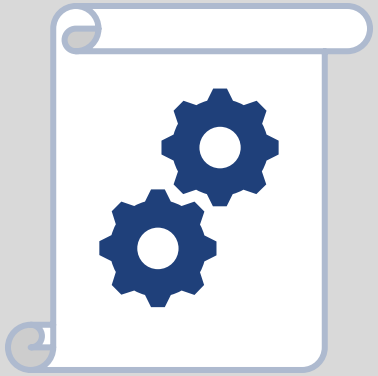


< 10% overhead for least privilege implementation plus additional mappings and model queries

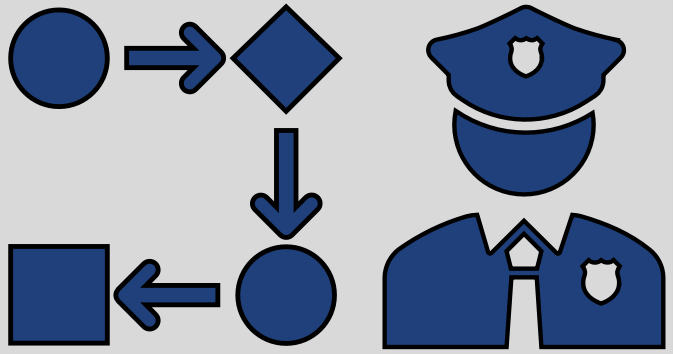
Faithful Hardware Representation and Least-Privilege Memory Management in Operating Systems



A new model to express memory addressing on modern machines



Generate OS code and hardware configuration

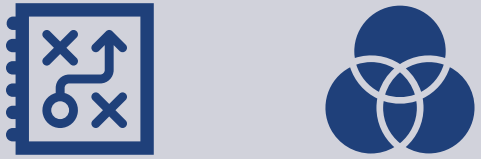


Express changes of the configuration and the required authority

Efficient Implementation in an Operating System

Future Directions

Performance Characteristics



Memory Access & Type Properties

Dynamic Caches

Vision

Apply the same approach to other areas of the operating system to obtain correct and reliable system software running on any platform.

Combining OS design & implementation with programming languages, code synthesis and Formal Methods.

Thanks to my collaborators

Timothy Roscoe	Lukas Humbel	Nora Hossle	David Cock	Daniel Schwyn
Simon Gerber	Kornilios Kourtis	Dejan Milojicic	Stefan Kaestle	Tim Harris
Gerd Zellweger	Roni Haecki	Moritz Hoffmann	Sabela Ramos	Jayneel Gandhi
Izzat El Hajj	Alexander Merritt	Contributors to the Barrelfish Operating System		

List of Related Publications

- **R. Achermann**, A. Panwar', J. Gandhi, A. Bhattacharjee, T. Roscoe. *Mitosis: Transparently Self-Replicating Page-Tables for Large-Memory Machines* (ASPLOS20)
- **R. Achermann**, N. Hossle, L. Humbel, D. Schwyn, D. Cock, T. Roscoe. *A Least-Privilege Memory Protection Model for Modern Hardware. (*ArXiv)
- L. Azriel, L. Humbel, **R. Achermann**, A. Richardson, M. Hoffmann, A. Mendelson, T. Roscoe, RN. Watson, P. Faraboschi, D. Milojicic D. *Memory-side protection with a capability enforcement co-processor*. (TACO).
- **R. Achermann**, L. Humbel, D. Cock, T. Roscoe. *Physical addressing on real hardware in Isabelle/HOL*. (ITP'18).
- L. Humbel, **R. Achermann**, D. Cock, T. Roscoe. *Towards Correct-by-Construction Interrupt Routing on Real Hardware.* (PLOS'17).
- **R. Achermann**, C. Dalton, P. Faraboschi, M. Hoffmann, D. Milojicic, G. Ndu, A. Richardson, T. Roscoe, A. L. Shaw; R. N. M. Watson. *Separating Translation from Protection in Address Spaces with Dynamic Remapping*. (HOTOS'XVI).
- **R. Achermann**, L. Humbel, D. Cock and T. Roscoe. *Formalizing Memory Accesses and Interrupts*. (MARS 2017).
- S. Kaestle, **R. Achermann**, R. Haecki, M. Hoffmann, S. Ramos, and T. Roscoe. *Machine-Aware Atomic Broadcast Trees for Multicores*. (OSDI'16).
- I. El Hajj, A. Merritt, G. Zellweger, D. Milojicic, **R. Achermann**, W. Hwu, K. Schwan, T. Roscoe, P. Faraboschi. *SpaceJMP: Programming with Multiple Virtual Address Spaces*. (ASPLOS XXI).
- S. Kaestle, **R. Achermann,** T. Roscoe, T. Harris. *Shoal: Smart Allocation and Replication of Memory For Parallel Programs* (ATC'15)
- S. Gerber, G. Zellweger, **R. Achermann**, K. Kourtis, T. Roscoe, D. Milojicic. Not Your Parents' Physical Address Space. (HotOS XV).