ETH zürich

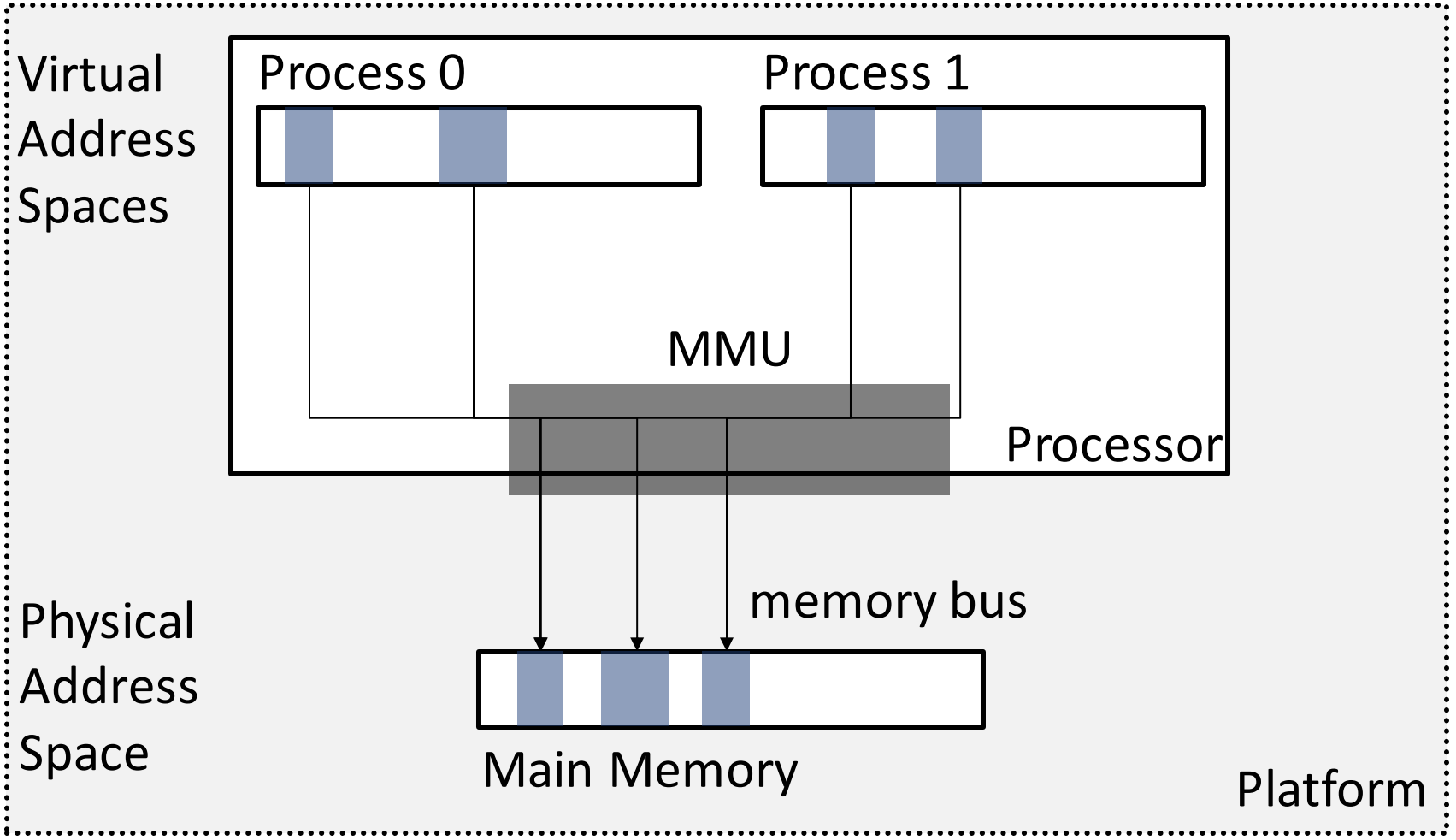
Reto Achermann

Memory Topology Models and Their Application in Operating Systems

Data 61, February 25, 2020

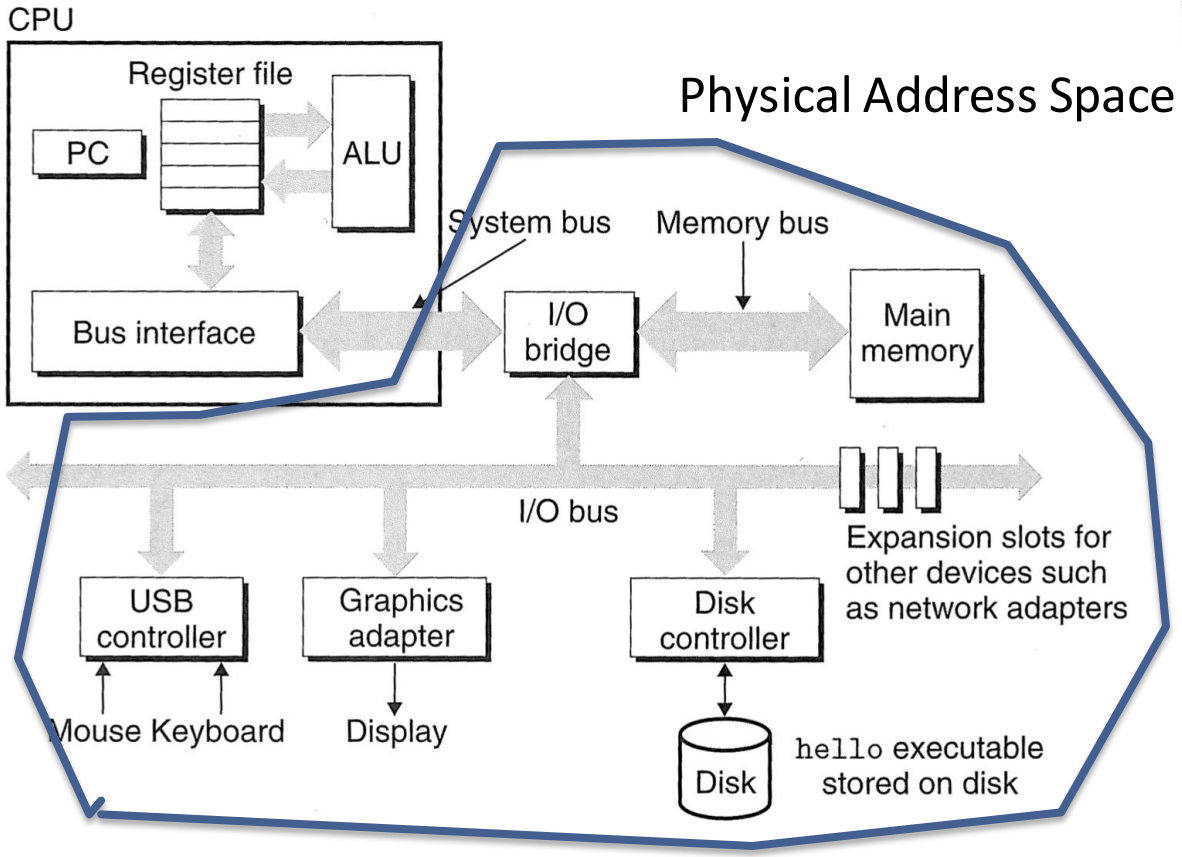
Systems@ETH zürich

Virtual Memory Abstraction



Computer Architecture 101 - All Platforms are Similar

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



Assumptions:

- A single, flat physical address space
- Physical address = resource identifier
- Physical address means the same for all CPUs and devices

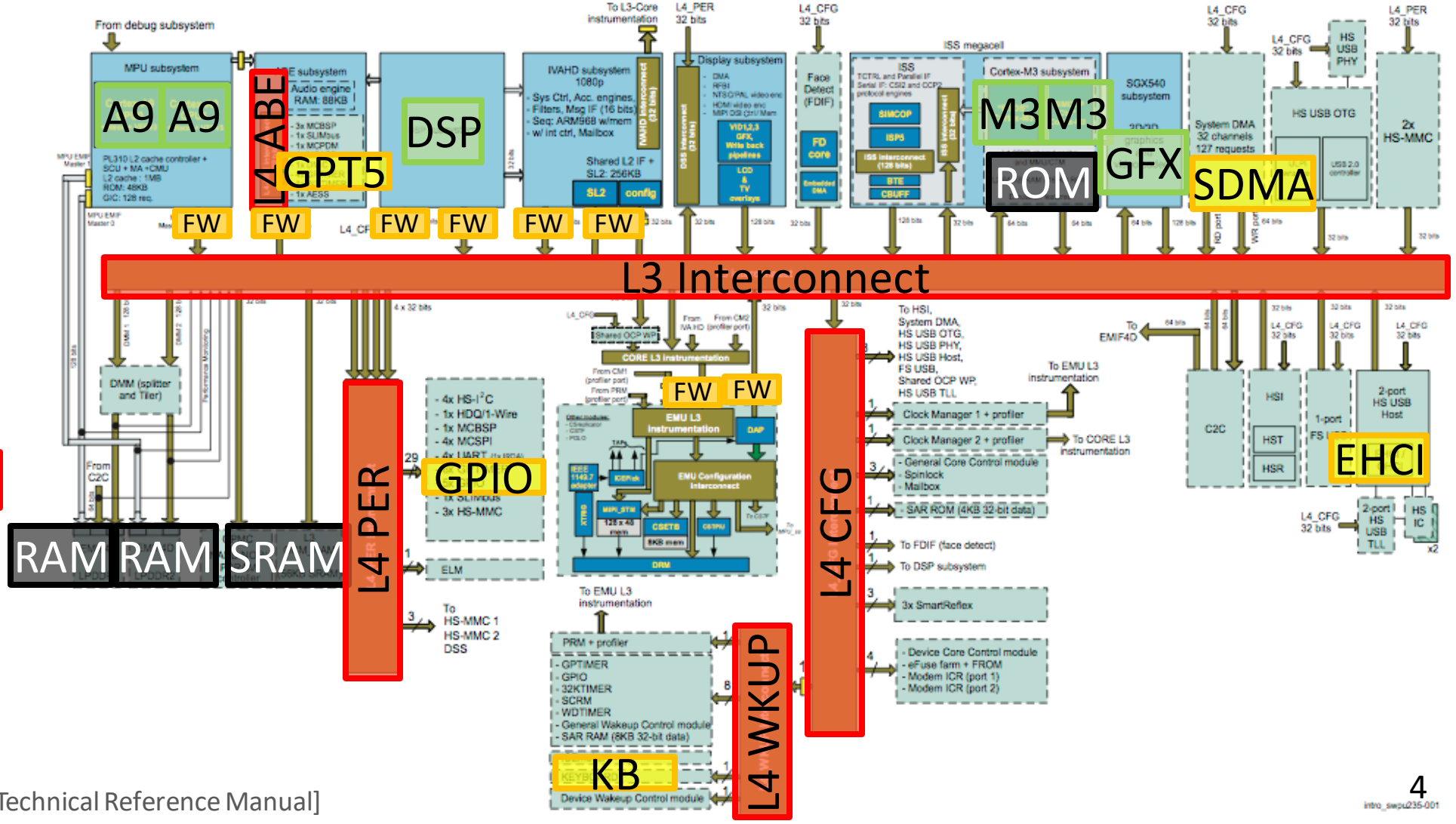
systems, but all systems have a similar look and feel. Don't worry about the complexity of this figure just now. We will get to its various details in stages throughout the course of the book.

Computer Systems, A Programmer's Perspective by Randal E. Bryant and David R. O'Hallaron

Reality: Same Resource, Multiple (Local) Physical Addresses

Simplified Block Diagram of TexasInstruments OMAP4460, Q4 2011

- 6+ heterogeneous cores
- Private and shared memory
- Devices attached to different interconnects
- 5+ Interconnects
- Firewalls

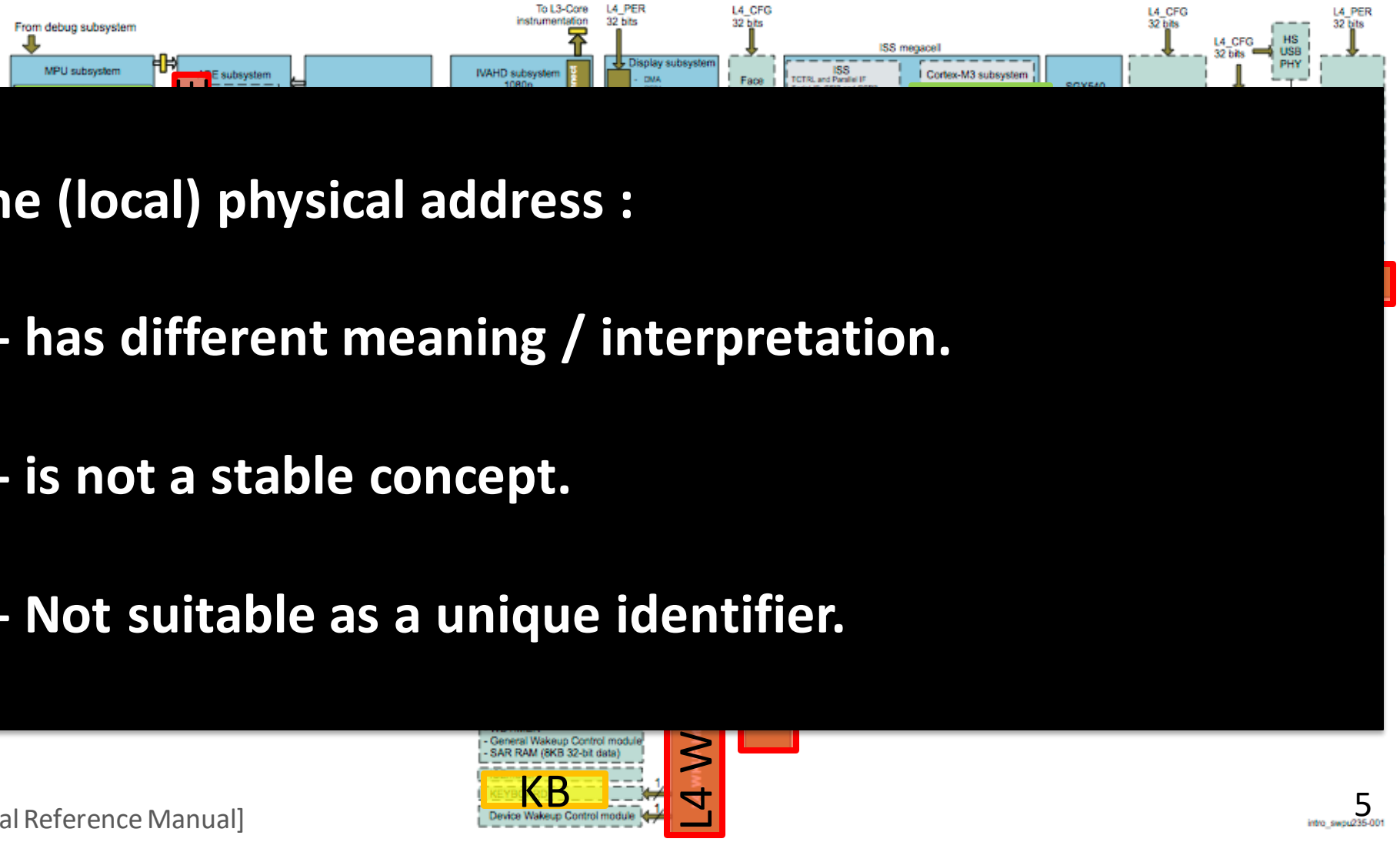


[Texas Instruments OMAP 4460 SoC, Technical Reference Manual]

Reality: Same Resource, Multiple (Local) Physical Addresses

Simplified Block Diagram of TexasInstruments OMAP4460, Q4 2011

6+ heterogeneous



The (local) physical address :

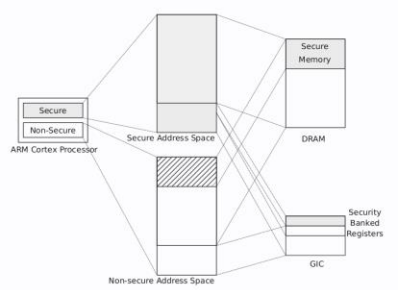
- has different meaning / interpretation.

- is not a stable concept.

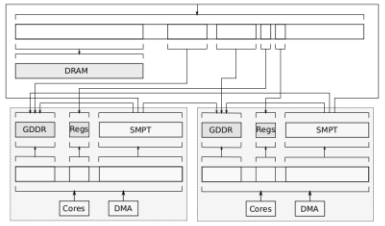
- Not suitable as a unique identifier.

[Texas Instruments OMAP 4460 SoC, Technical Reference Manual]

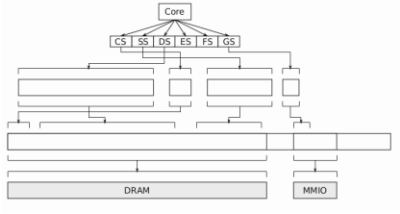
Complicated Memory Topologies are Ubiquitous



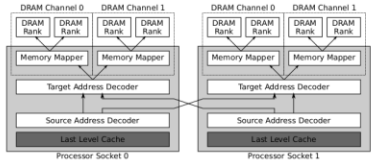
Secure and non-secure
(ARM TrustZone / Intel SGX)



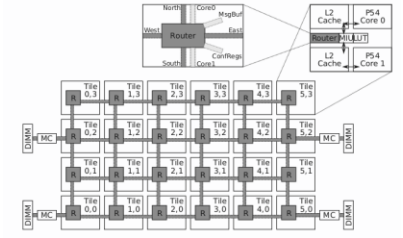
Co-Processors
(Intel Xeon Phi)



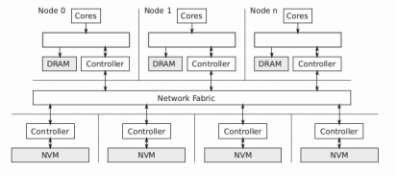
Segmentation
(x86 / Power)



Memory Controller
remappings

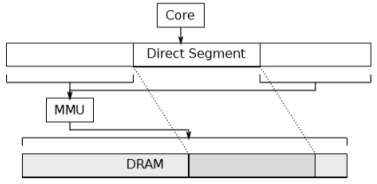


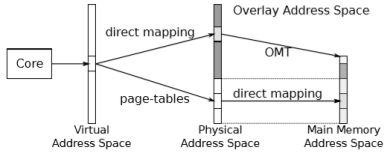
Intel Single Chip Cloud
Computer



Fabric Attached Memory
(GenZ / The Machine)

Direct Segments





Page Overlays

Platforms are highly diverse.

Address translations are configurable.

System software needs to correctly handle all of these...

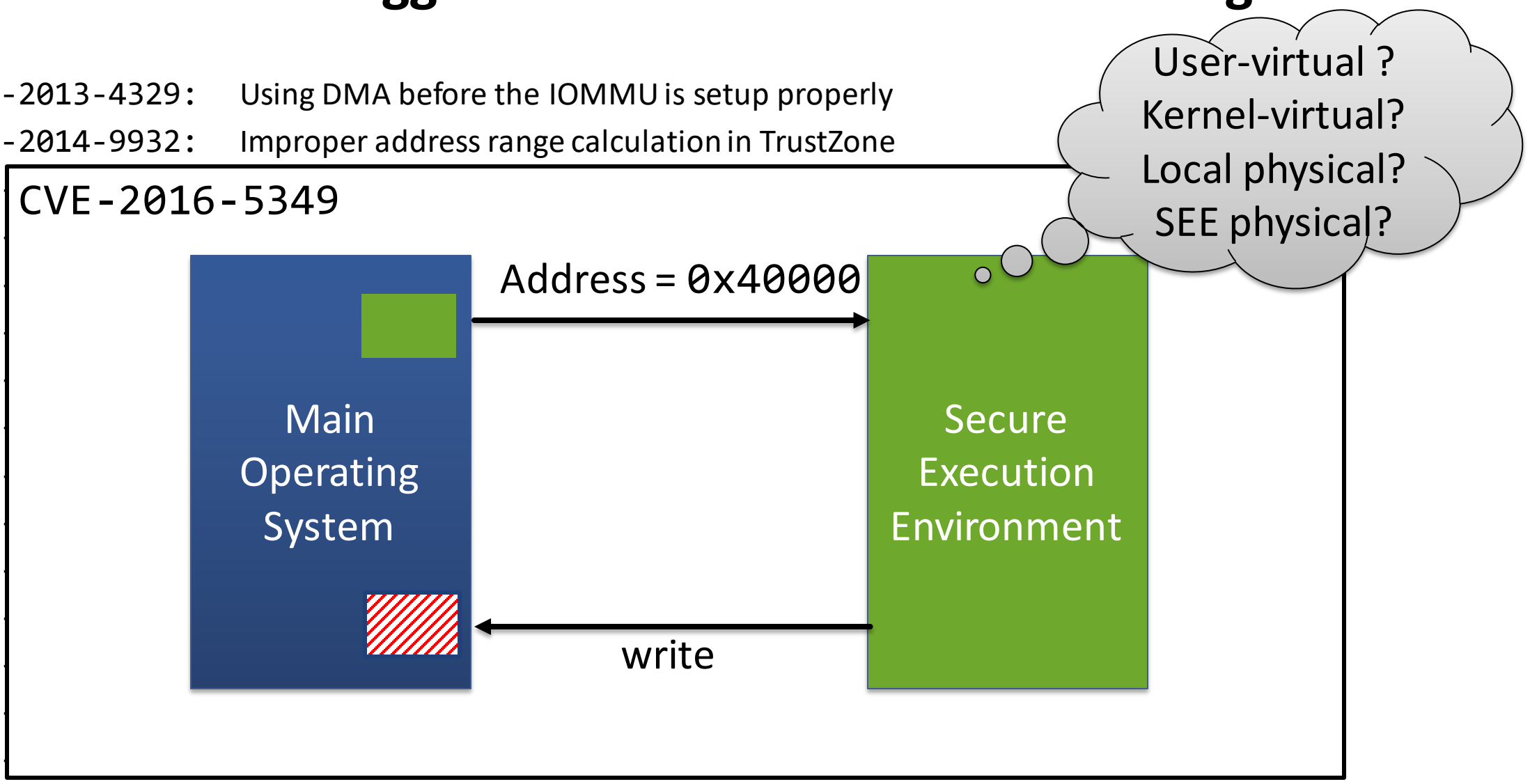
System Software Struggles with Correct Hardware Configuration

Wrong Addresses

- CVE-2013-4329: Using DMA before the IOMMU is setup properly
- CVE-2014-9932: Improper address range calculation in TrustZone
- CVE-
- CVE-
- CVE-
- CVE-
- CVE-
- CVE-

Misconfigurations

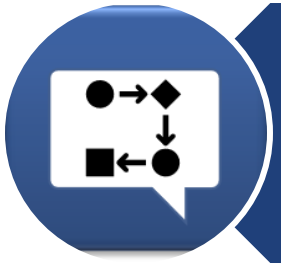
- CVE-
- CVE-
- CVE-
- CVE-
- CVE-
- CVE-
- CVE-
- CVE-
- Trusting co-processor requests, and GPU exploits bypassing protection, cross SoC attacks, …



My Research Focus

My goal: Specify the software-visible semantics of hardware to provide a sound basis for system software.

Design and Implementation of Operating Systems



Domain Specific Languages for Systems Engineering

Applying Formal Methods to Operating Systems and Hardware Specification

Research Questions

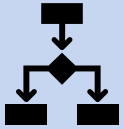
- How can system software **accurately represent** and **reason** about the complex **memory topology** as it is seen by software?

- How can the accurate representation be **efficiently implemented** in an operating system and what is the resulting overhead?

Talk Outline

- How can system software **accurately represent** and **reason** about the complex **memory topology** as it is seen by software?

➔ The *Decoding Net* - A Formal Model for Memory Addressing



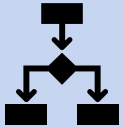
- How can the accurate representation be **efficiently implemented** in an operating system and what is the resulting overhead?

➔ Operating System Support for Dynamic Decoding Networks

Talk Outline

- How can system software **accurately represent** and **reason** about the complex **memory topology** as it is seen by software?

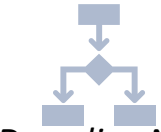
➔ The *Decoding Net* - A Formal Model for Memory Addressing



- How can the accurate representation be **efficiently implemented** in an operating system and what is the resulting overhead?

➔ Operating System Support for Dynamic Decoding Networks

Specifying and Modeling Hardware and Memory Topology



- Industry Standards: DeviceTrees / UEFI / ACPI / USB / PCI Express

➔ Insufficient for accurate hardware representation with well-defined semantics

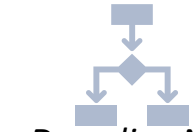
- Memory & Processor Models: ARM's ASL, VAMP, and Sewell et al.

➔ Focusing on the processor core. Complimentary problem to address routing.

- Verified Operating Systems: seL4, CertiKOS

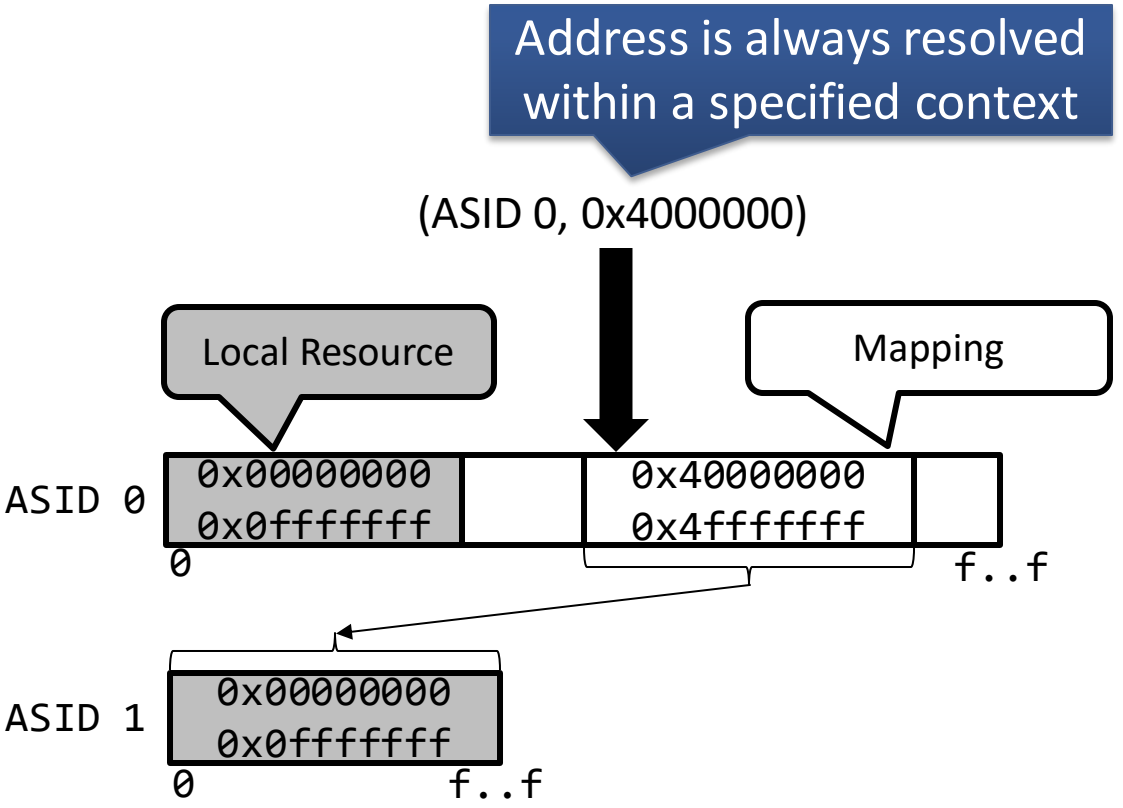
➔ Proofs need an accurate platform abstraction. A customer of my work.

Explicit Address Spaces – A More Accurate Representation



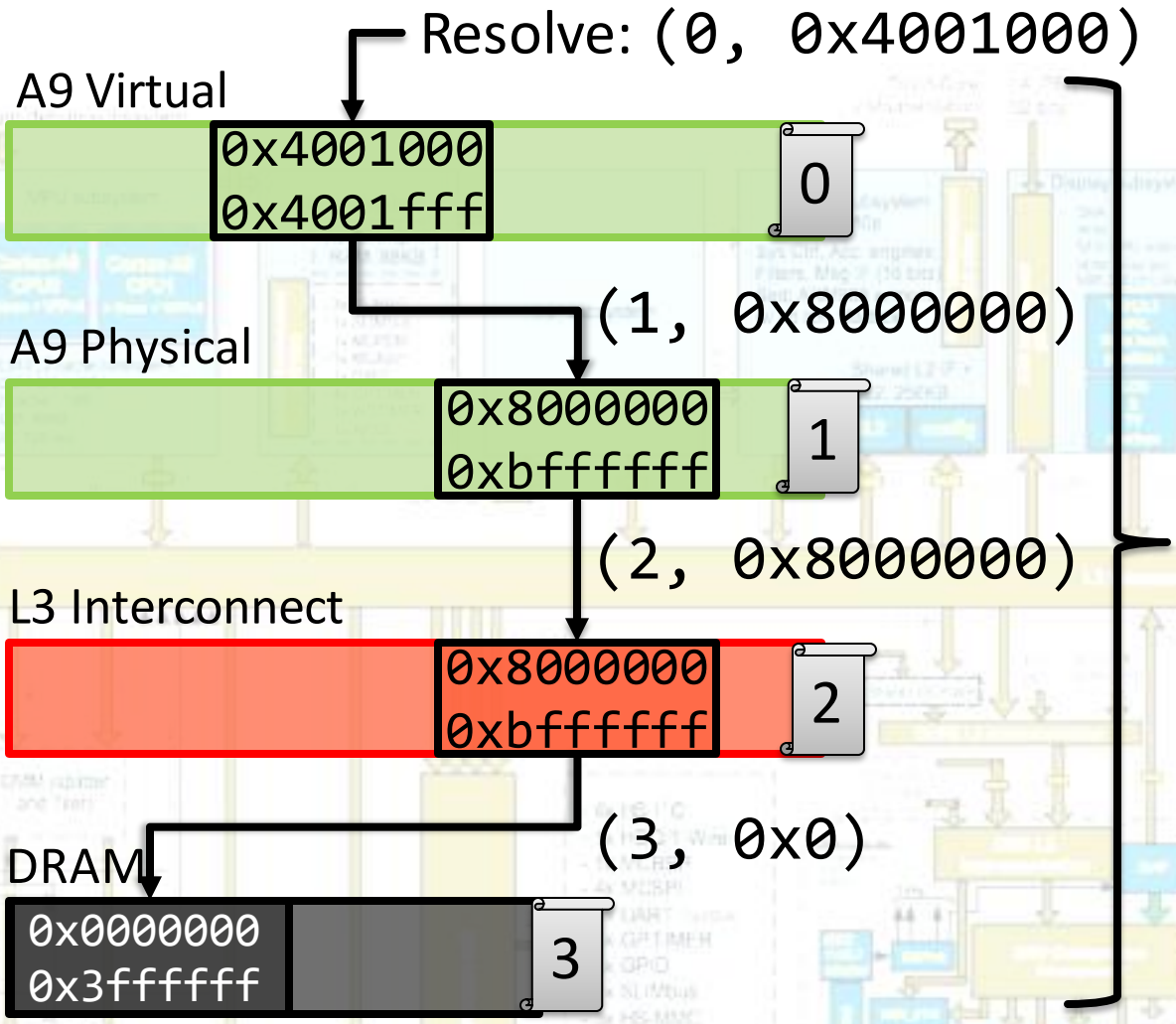
- Memory management requires **unambiguous references** to resources.
  ➔ **explicit context** for addresses (**Naming Problem**)

- Address Space
  - **Context** for resolving addresses
  - Set of address values e.g. range [0, 2^b)
  - Regions **mappings or local resources**



Example: Accessing DRAM from a Cortex A9 core on the OMAP

Decoding Net



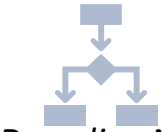
Write down the address spaces as seen from software:

L3 **translates** [0x8000000-0xbfffffff] to DRAM at 0x0

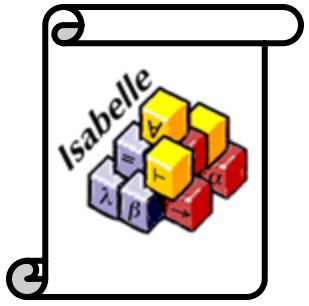
DRAM **local** [0x00000000-0x3fffffff]

This gives an *intuition* of the address decoding

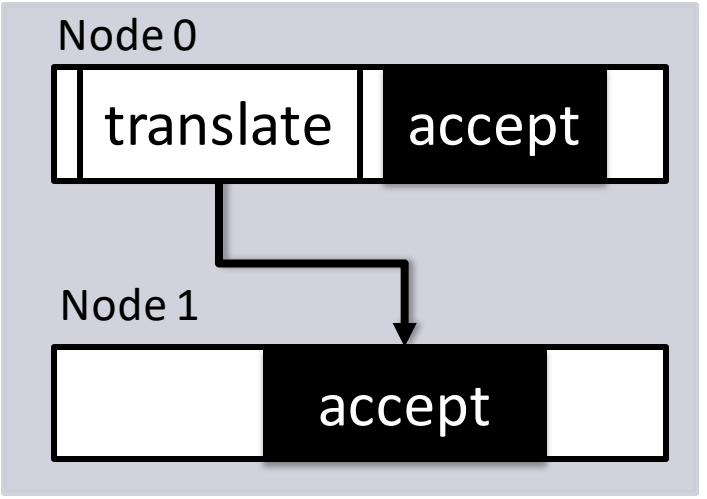
Modelling Memory Accesses as a *Decoding Net*



- Directed graph: node ↔ address space
- Name: **qualified address**
- Nodes have **two properties**:
  - **Accept**: local resources
  - **Translate**: mapped resources



DecodingNet.thy

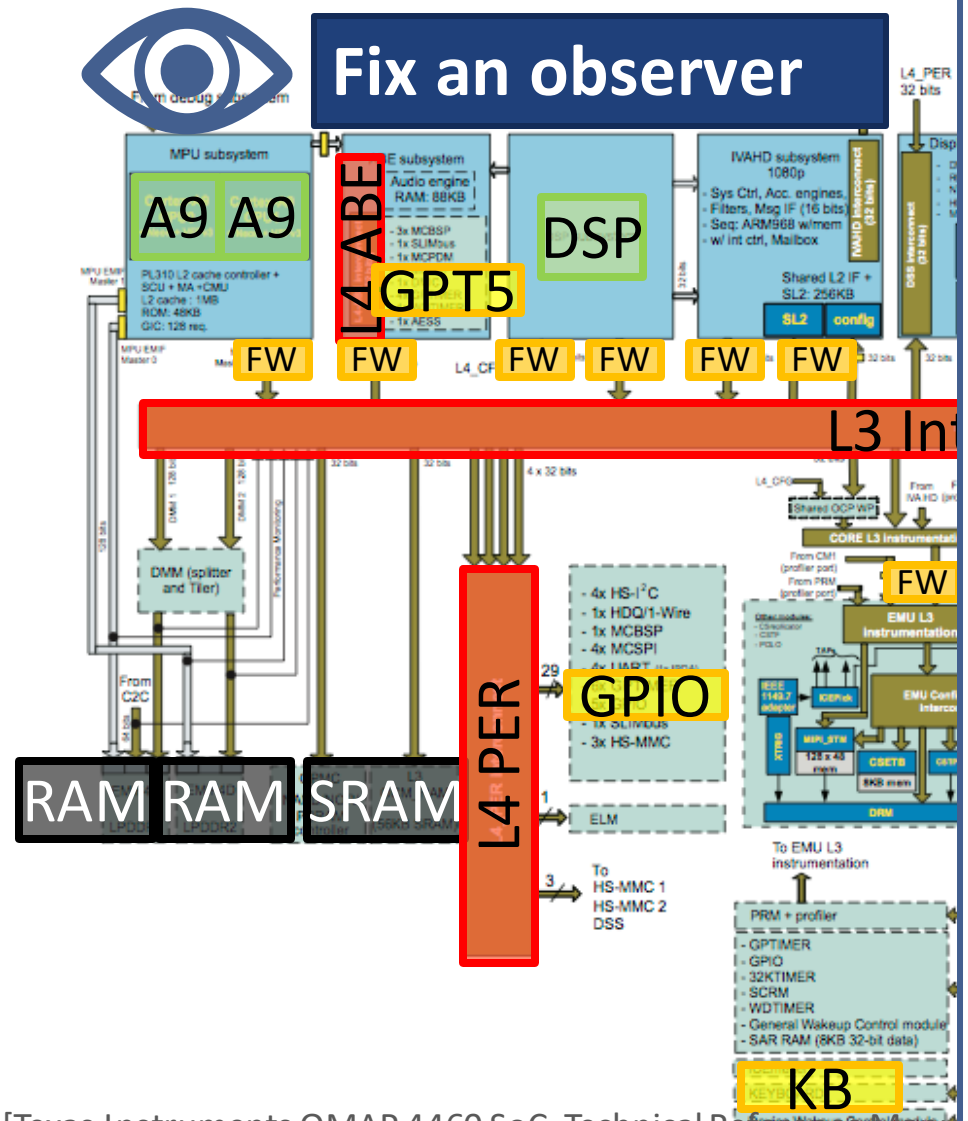


$$\text{name} :: (\mathbb{N}, \mathbb{N})$$

$$\text{node} ::$$
$$accept :: \{\mathbb{N}\}$$
$$translate :: \mathbb{N} \to \{name\}$$

$$\text{net} :: \mathbb{N} \to node$$

Example: View-Preserving Transformations



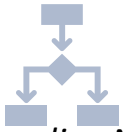
[Texas Instruments OMAP 4460 SoC, Technical Reference Manual]

OMAP.thy

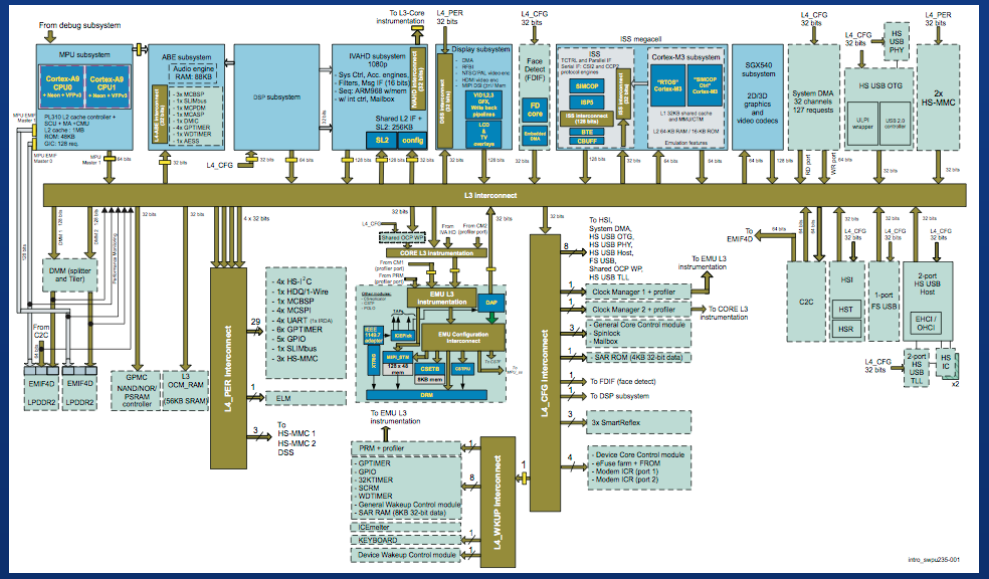
```
definition "a9virt_0 =  empty_spec ⦇
  map_blocks := [block_map (blockn 0x4001000 0x4001fff)  1 0x80000000 ]
⦈"

definition "a9phys_0 =  empty_spec ⦇
  map_blocks := [direct_map (blockn 0x80000000 0xBFFFFFFF) 2]
⦈"

definition "l3_interconnect = empty_spec ⦇
  map_blocks := [block_map (blockn 0x80000000 0xBFFFFFFF) 3 0x00000000]
⦈"

definition "ram = empty_spec ⦇
  acc_blocks := [blockn 0x00000000  0x3FFFFFFF],
⦈"

definition "sys = [(3,ram), (2, l3_interconnect),
                   (1, a9phys_0), (0,a9virt_0)]"

definition "OMAP44xx_Net = mk_net sys"
```

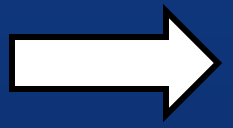
Example: View-Preserving Transformations

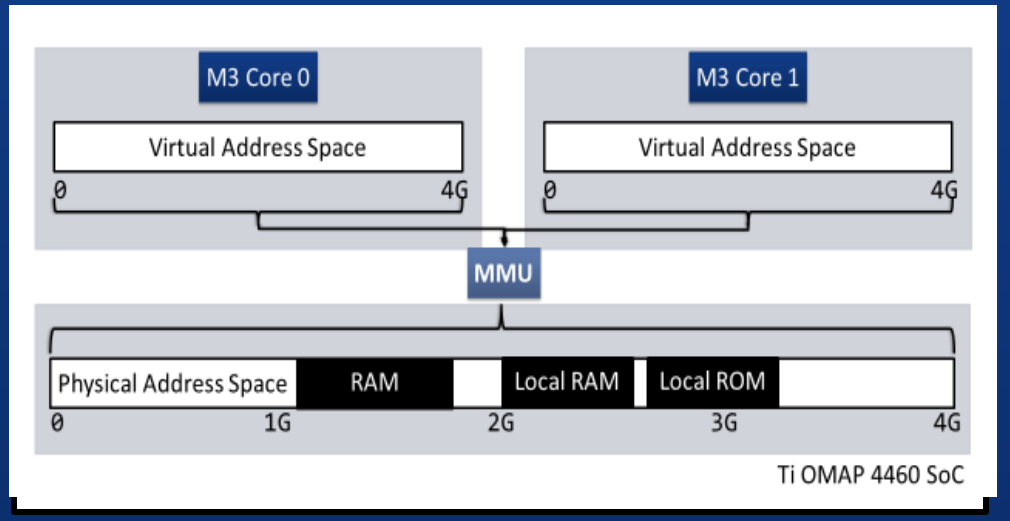
Fixed observer



Decoding Net





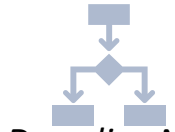


For ONE observer the flattened representation is equivalent.

0x4000000 DRAM

Accept 0x0

Can The *Decoding Net* Represent Real Hardware?



Case Study: Software-Loaded Translation Lookaside Buffer (TLB)

[MIPS R4600 Manual]

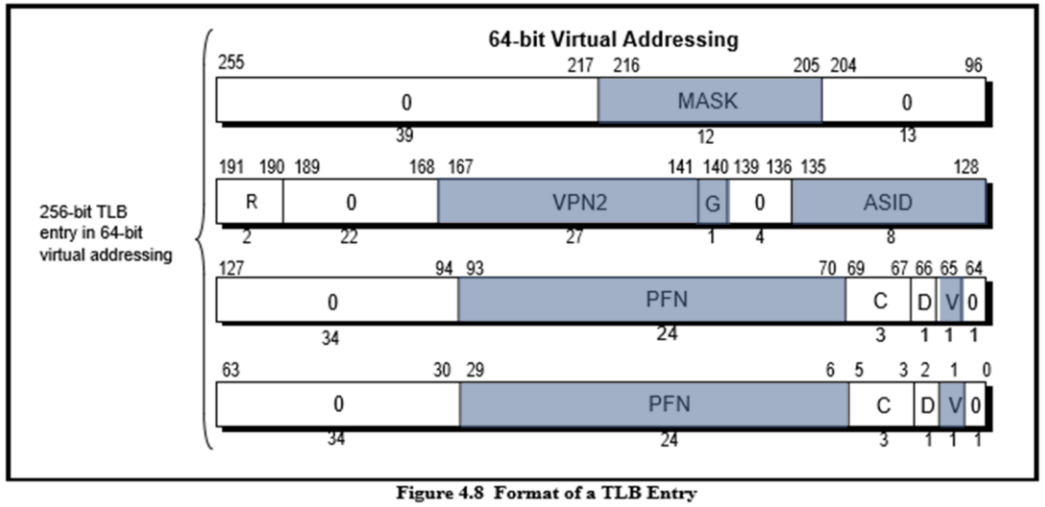
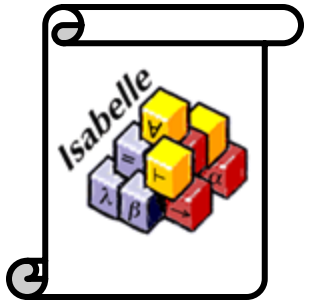


Figure 4.8 Format of a TLB Entry



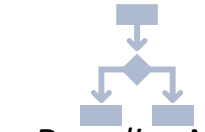
MIPS_R4600_TLB.thy

Definition of an **operational model** of the MIPS R4600 TLB.

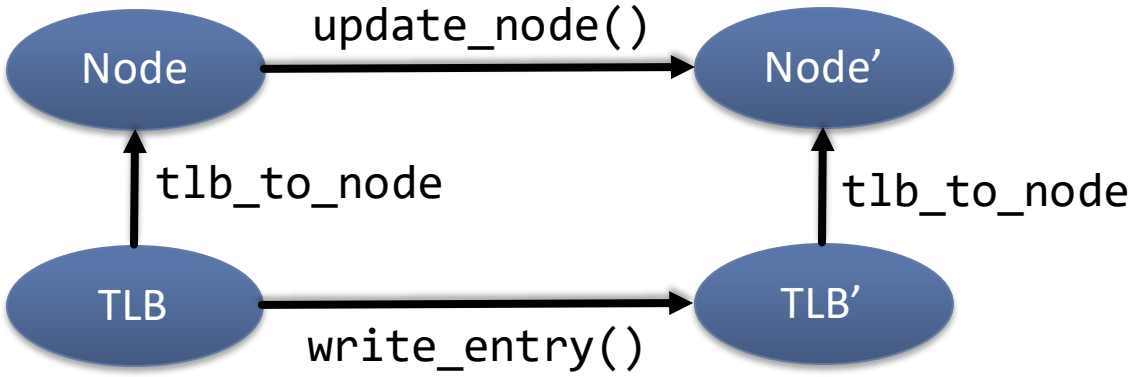
software-loaded:
explicit control on TLB updates

48 entry-pairs, multiple page sizes, address space identifiers, …

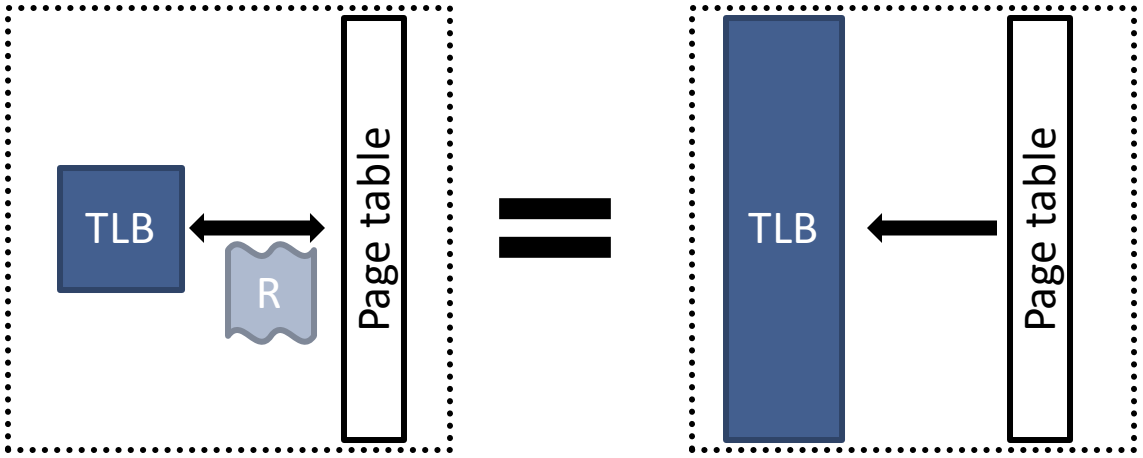
Results of the Translation Lookaside Buffer Model



Refinement:



Equivalence:



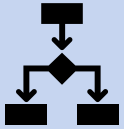
The Decoding Net captures the state of a real TLB hardware as seen by software.

Address translation defined by an in-memory data structure.

Talk Outline

- How can system software **accurately represent** and **reason** about the complex **memory topology** as it is seen by software?

➔ The *Decoding Net* - A Formal Model for Memory Addressing



- How can the accurate representation be **efficiently implemented** in an operating system and what is the resulting overhead?

➔ Operating System Support for Dynamic Decoding Networks

From the Static *Decoding Net* Model to an OS Implementation

OS Support

Dynamic Address Spaces

Authorization Model

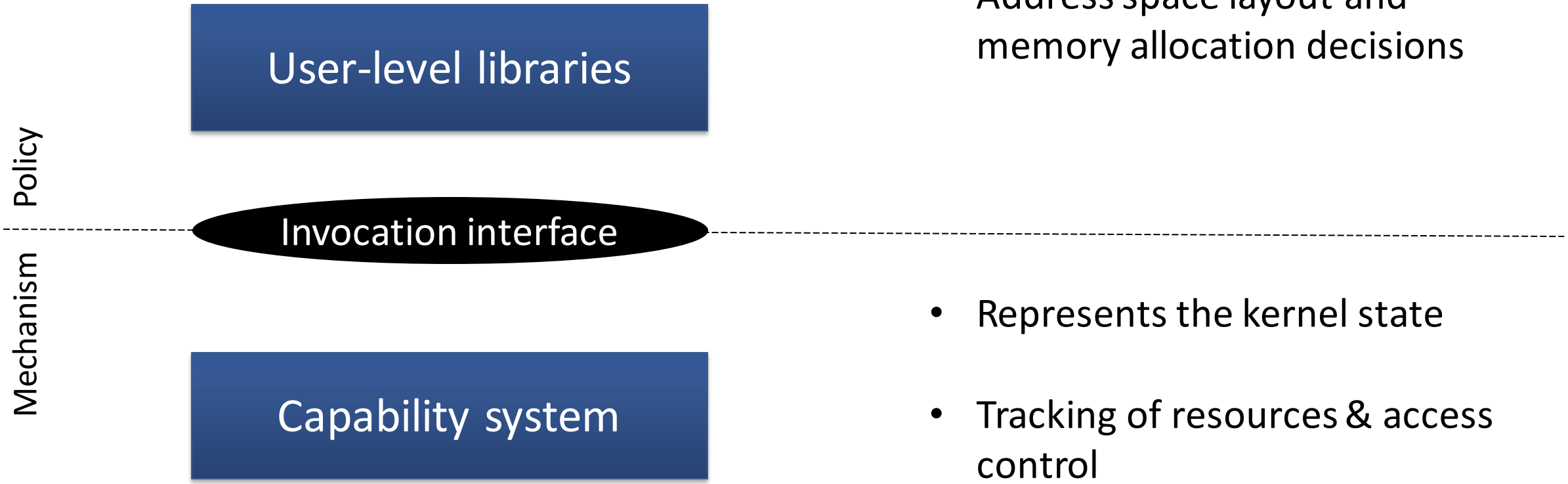
Executable Specification

Operating System Implementation

- **Barrelfish/MAS:** extension to Barrelfish
- Support for heterogeneous platforms

Background: Policy-Mechanism Separation in Barrelfish

OS Support

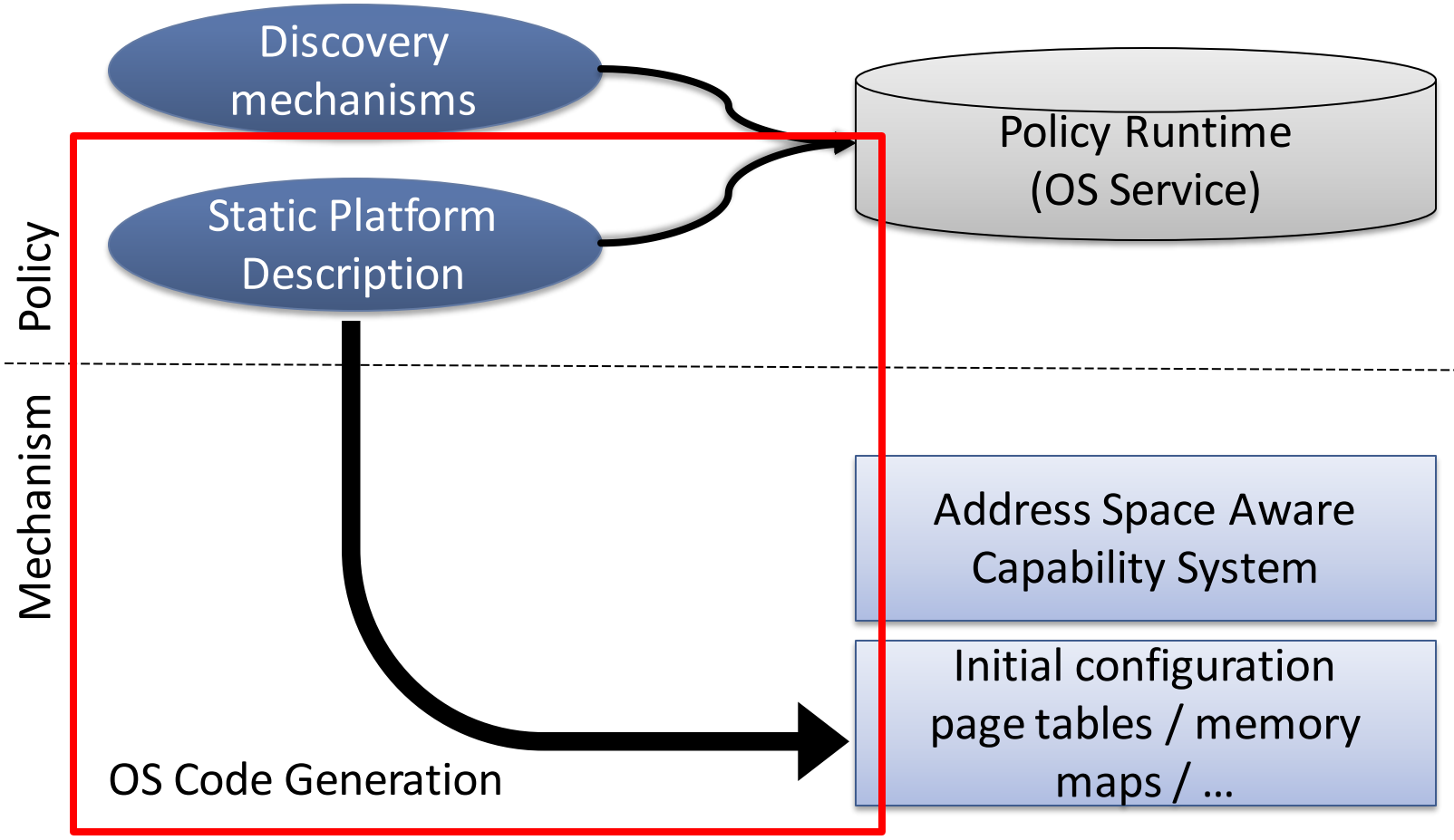


- Address space layout and memory allocation decisions
- Represents the kernel state
- Tracking of resources & access control

Capability = unforgeable token of authority

Barrelfish/MAS Architecture

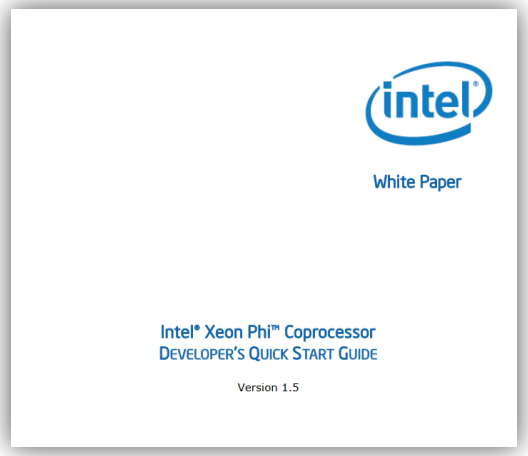
Input to the model state



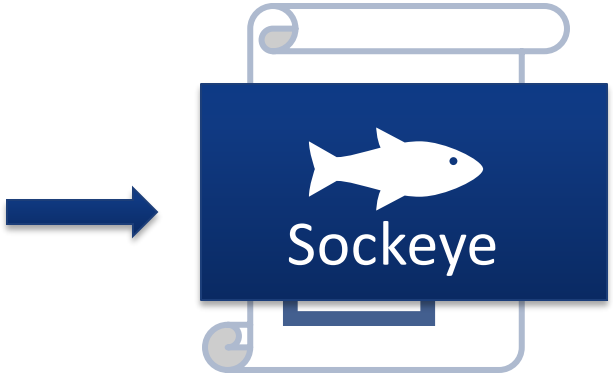
- Memory topology representation
- Allocation and configuration policies

- Address space configuration

From Platform Description to Operating System Code



Vendor supplied data (e.g. Hardware manual, XML files, …)



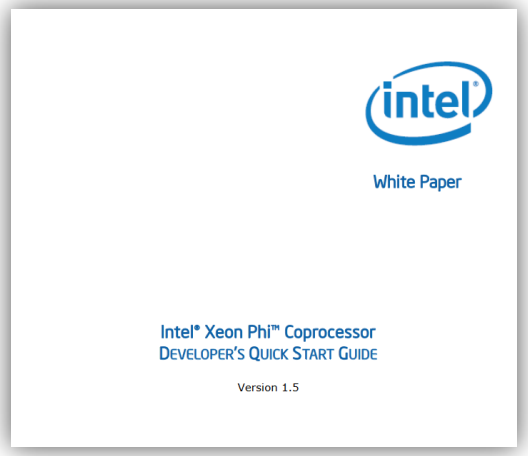
Machine readable description of the platform

Sockeye Domain Specific Language for Hardware Descriptions

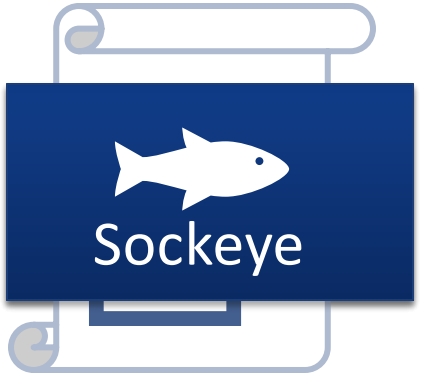
```
module KNC_225e {
  memory (0 bits 40) GDDR
  GDDR accepts [(0x000000000 to 0x3fffffff)]

  memory (0 bits 12) LAPIC[0 to 227]
  LAPIC[*] accepts [(*)]

  memory (0 bits 16) MMIO
  MMIO accepts [(*)]

  memory (0 bits 40) KNC_SOCKET
  KNC_SOCKET maps [
    (0x0000000000 to 0x3fffffff) to GDDR at (*);
    (0x08007D0000 bits 16) to MMIO at (*)
  ]

  memory (0 bits 40) K1OM_CORE[0 to 227]
  K1OM_CORE[*] maps [
    (0xfee00000 bits 12) to LAPIC at (*)
  ]

  K1OM_CORE[*] overlays KNC_SOCKET }
```

From Platform Description to Operating System Code

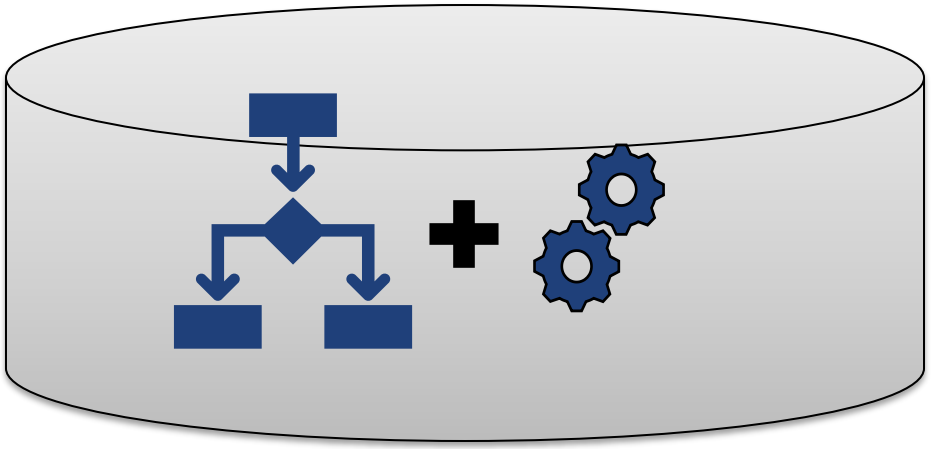
OS Support



Vendor supplied data (e.g. Hardware manual, XML files, …)



Machine readable description of the platform



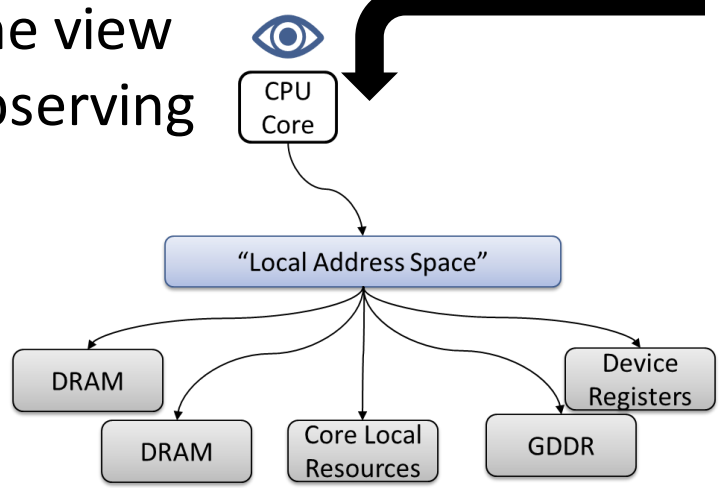
- Memory topology
- Algorithms (Allocation, …)

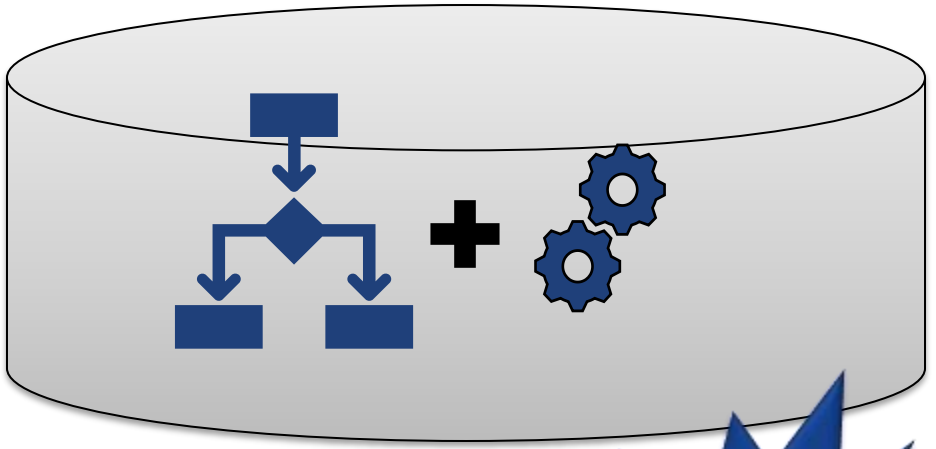
From Platform Description to Operating System Code

OS Support

Policy Runtime (OS Service)

Compute the view from the observing core [0]





ARMv7, ARMv8, x86_64, K1OM

core 0

Page Table

Address Translations

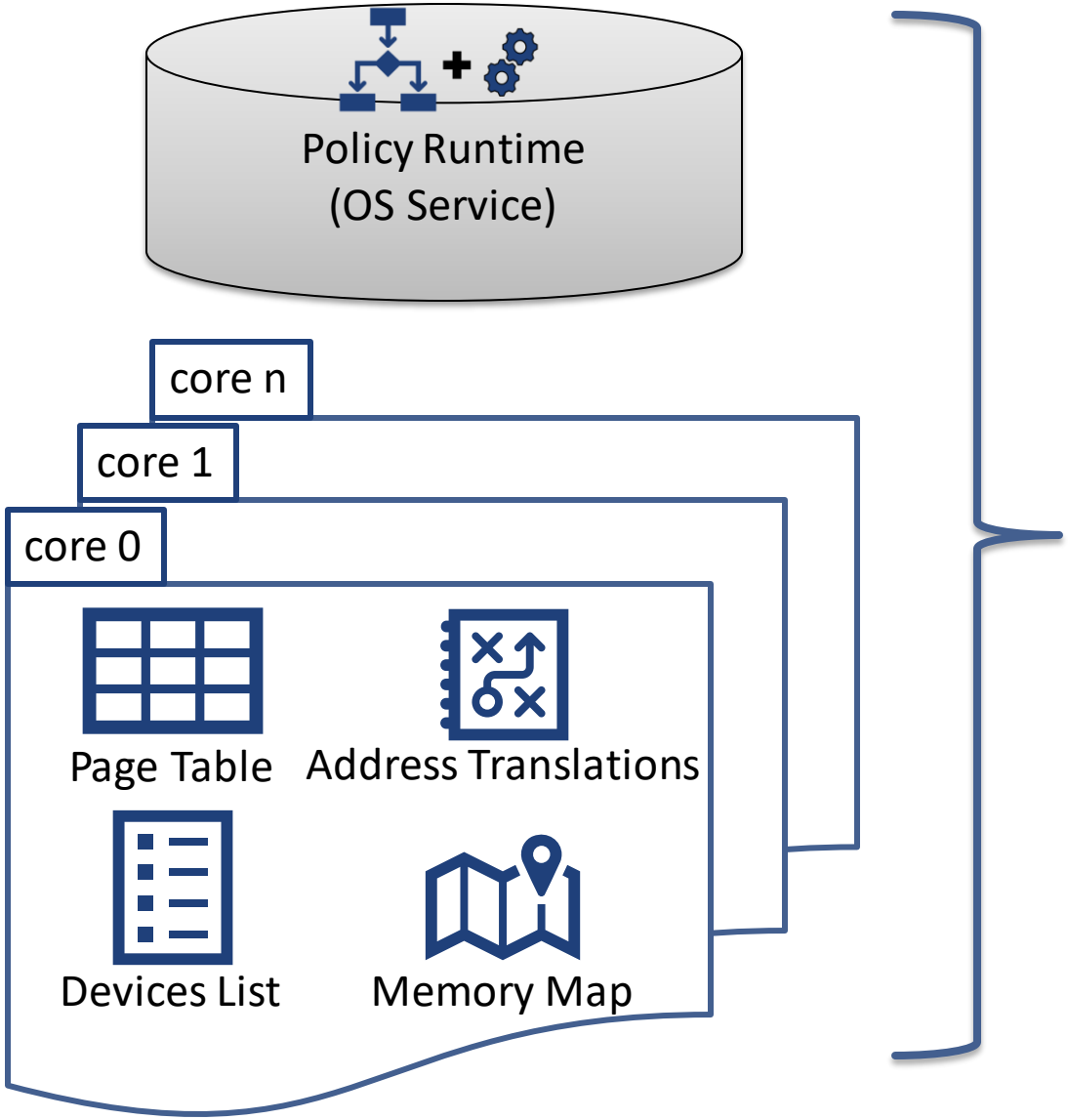
Devices List

Memory Map

Generate page-tables based on the flattened representation

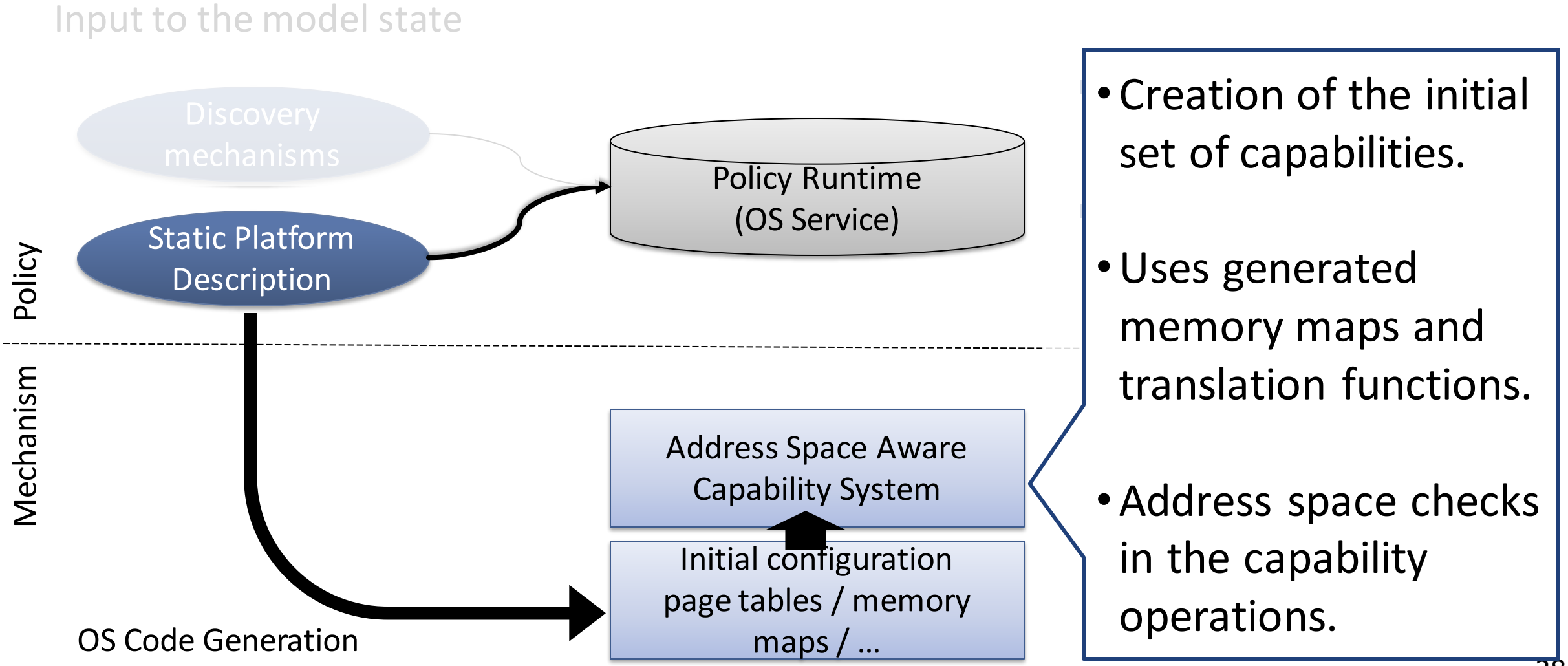
Page Table Spec [Co
Architecture = [ARMv8
Mappings = [
DRAM0 @ 0x8000000,
DevRegs @ 0x10000000
]

From Platform Description to Operating System Code



Customized operating system image built for a specific platform.

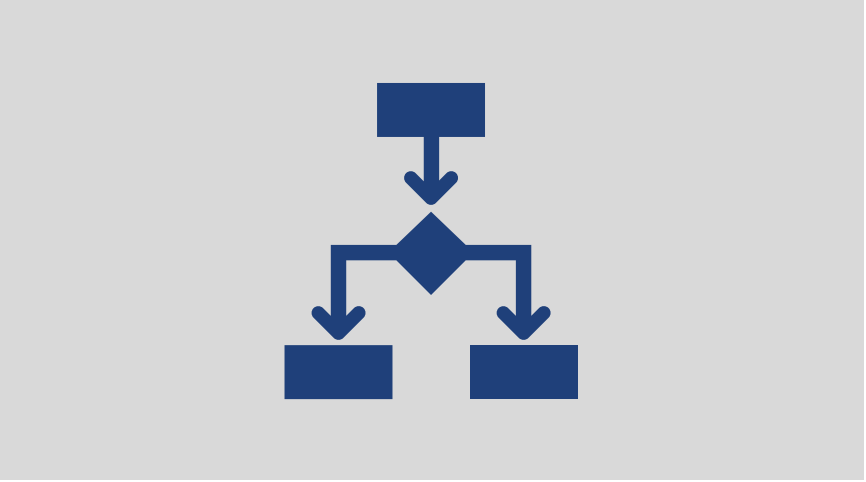
Barrelfish/MAS Architecture



- Creation of the initial set of capabilities.
- Uses generated memory maps and translation functions.
- Address space checks in the capability operations.

Evaluation

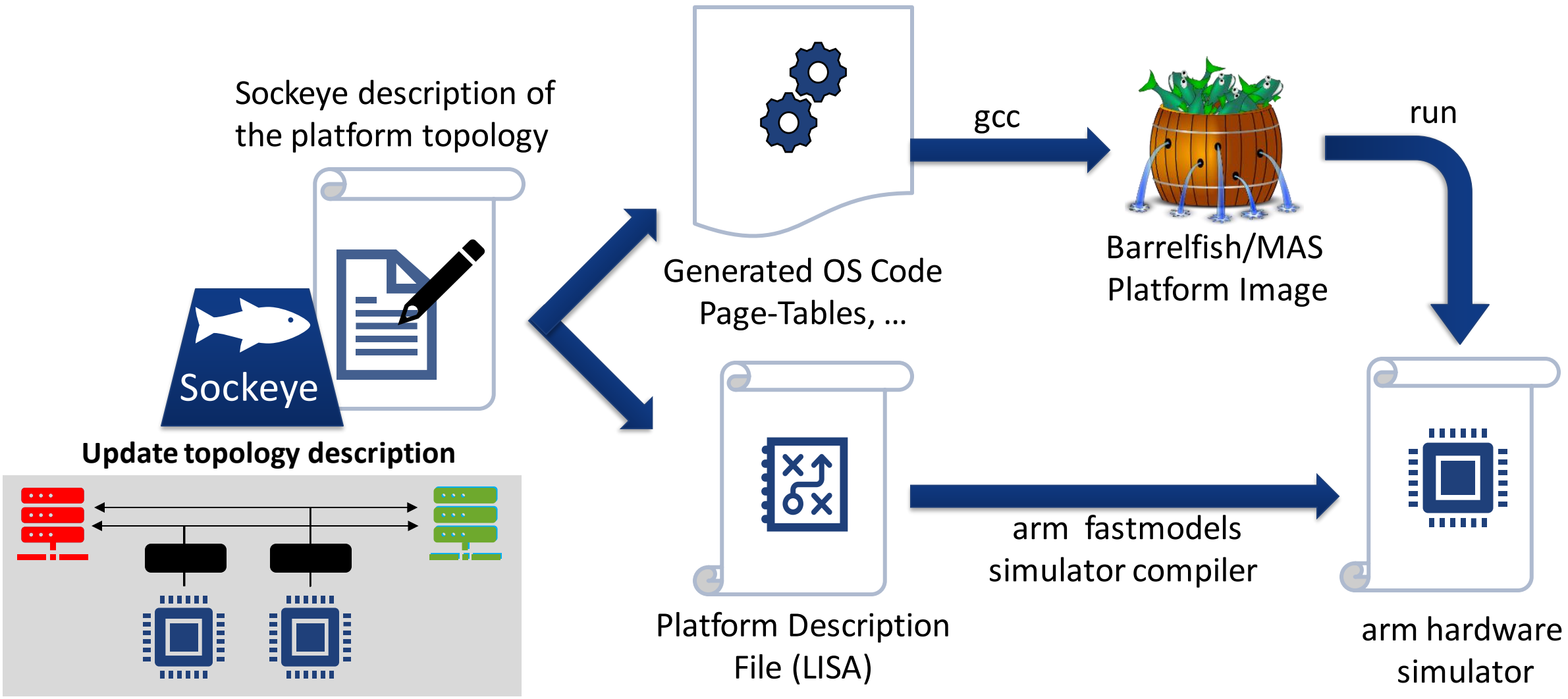
Evaluation



Can Barrelfish/MAS handle complex memory topologies ?

What is the overhead for dynamic address space configuration?

Validation: Barrelfish/MAS Handles Complex Topologies



The Cost of Dynamic Address Space Reconfiguration

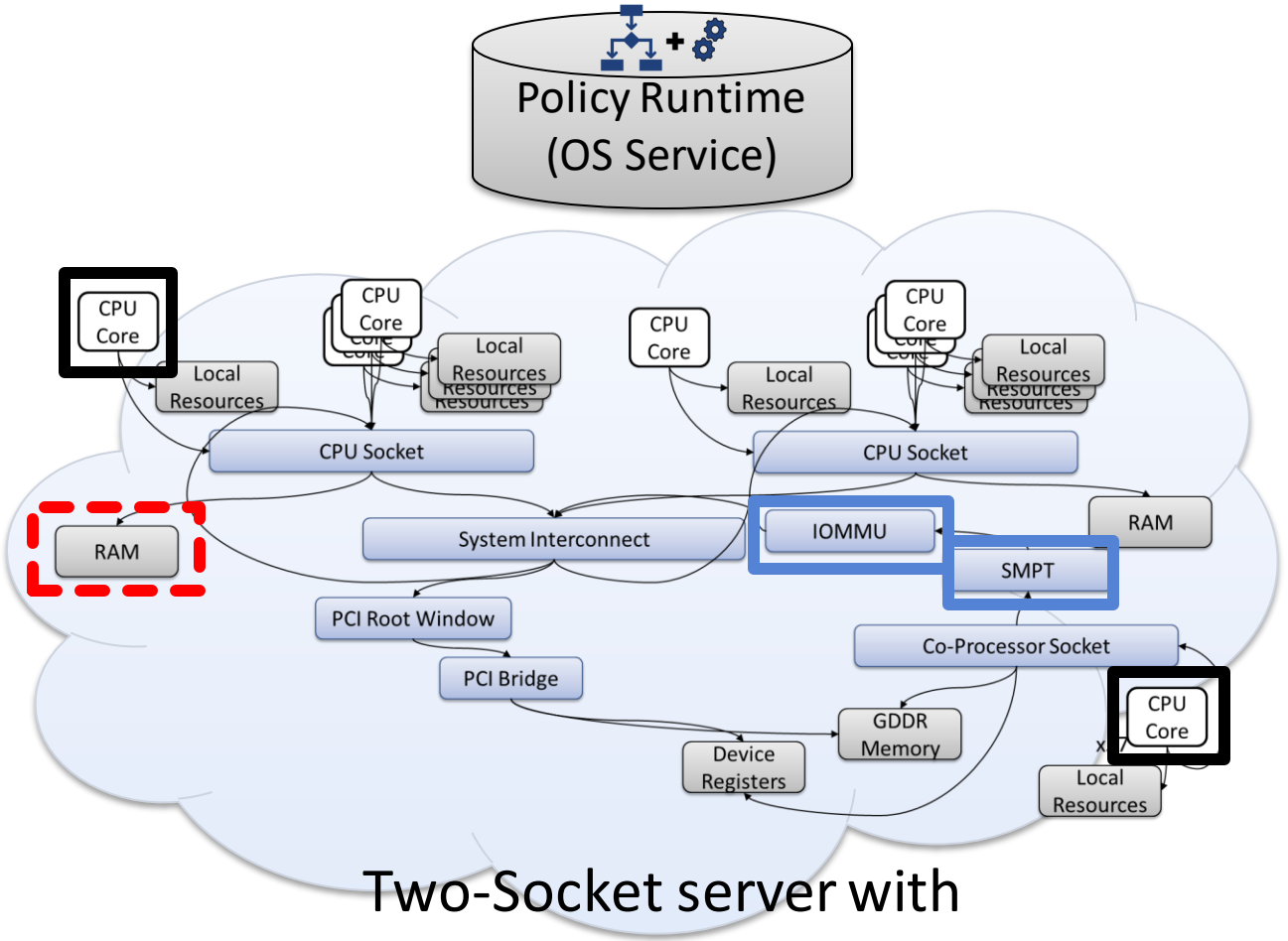
Task: Setup a shared buffer between the host CPU and the co-processor.

Buffer size: 8 MiB

1) Model Query:
   - address space to allocate memory
   - list of address spaces to configure

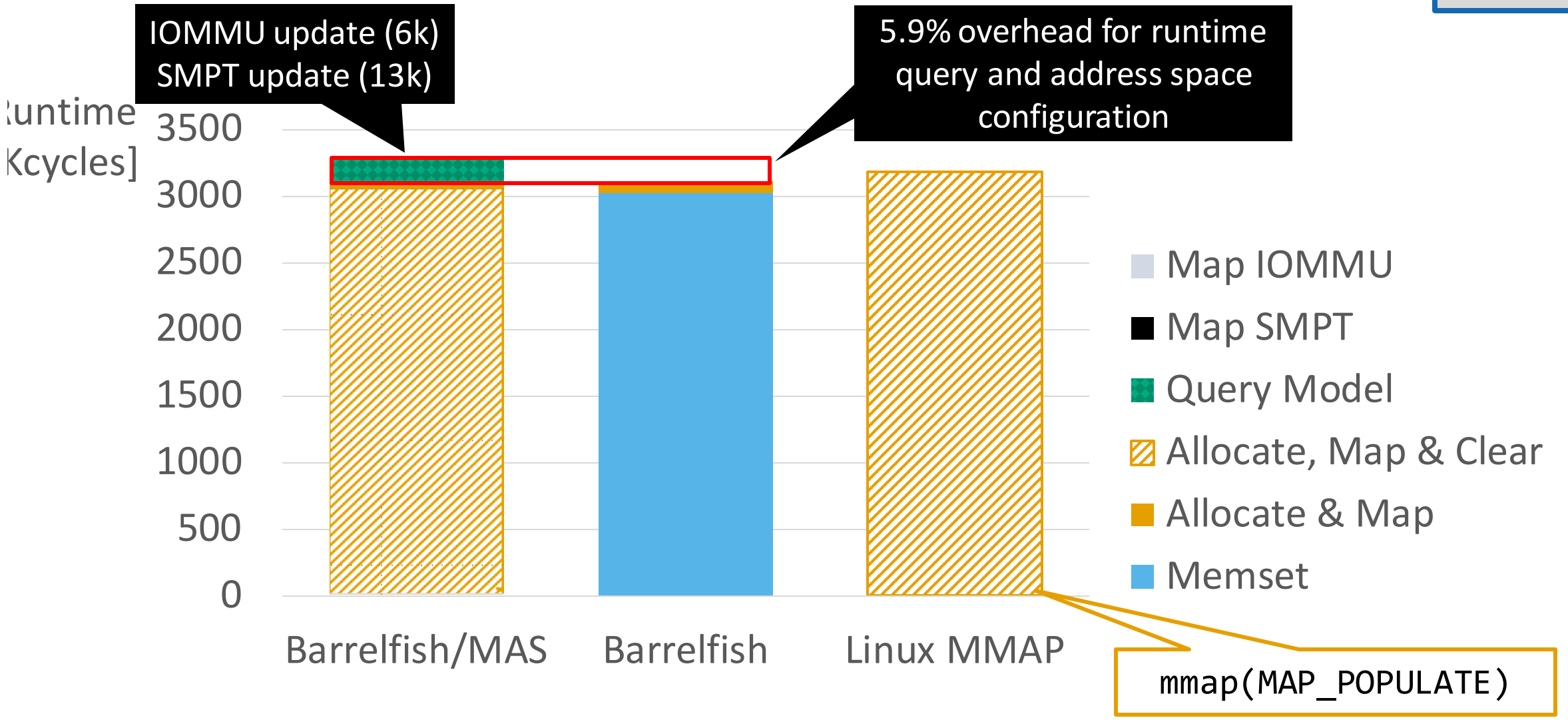
2) Allocation & mapping of memory

3) Configuration of address translation



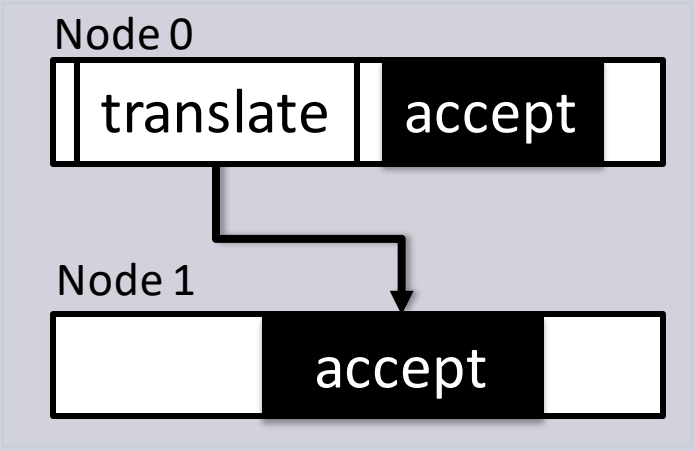
Two-Socket server with
Xeon Phi Co-Processor

The Cost of Dynamic Address Space Reconfiguration

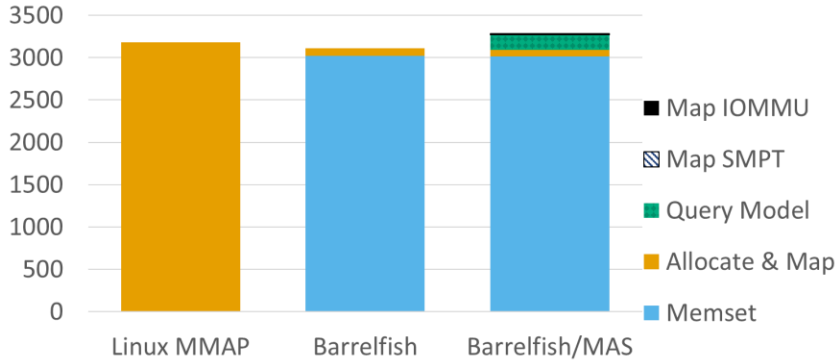


uffer Size: 8 MiB

Summary




Barrelfish/MAS

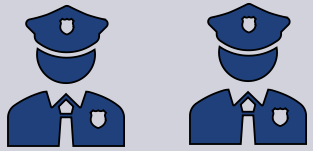


Decoding Net
Accurate representation of the memory subsystem of a platform.

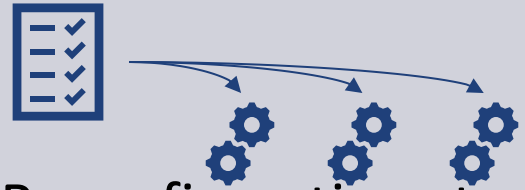
OS Implementation
Address space aware capability system and OS code generation

Efficient Implementation
Detailed memory topology model, at low overhead.

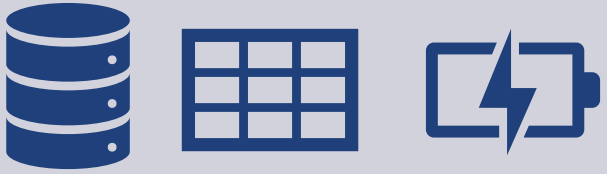
Performance and memory type properties



Multiple, distrusting reference monitors

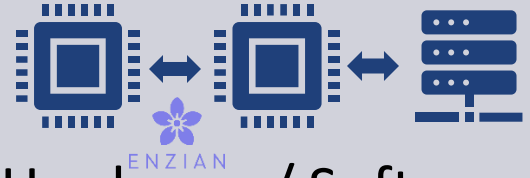


Reconfiguration steps and code generation



Dynamic caches, clocks, and power

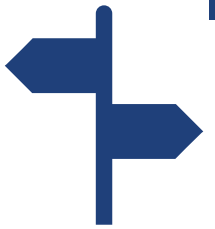
Verification of the executable model



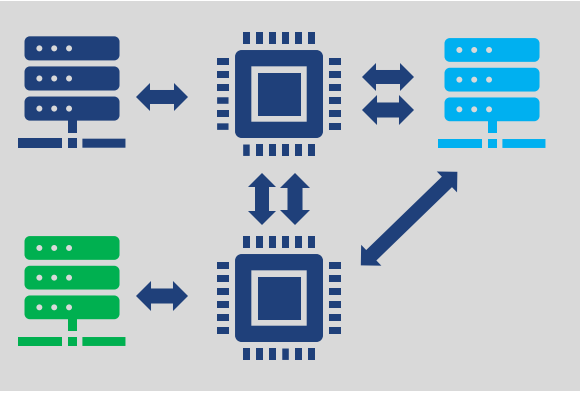
Hardware / Software co-design

Future Directions

Future Directions: Memory Access Characteristics

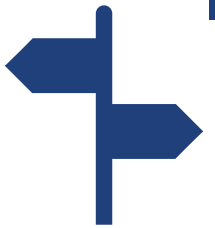


- Observation: Not all memory is equal
  - Persistent / volatile memory / high-bandwidth memory / DRAM /…
- Different access characteristics
  - latencies / bandwidth / coherency



- “Smart Memories” with data processing capabilities
- **libnuma++** Extend the model with characteristics and integrate this in memory allocation policies

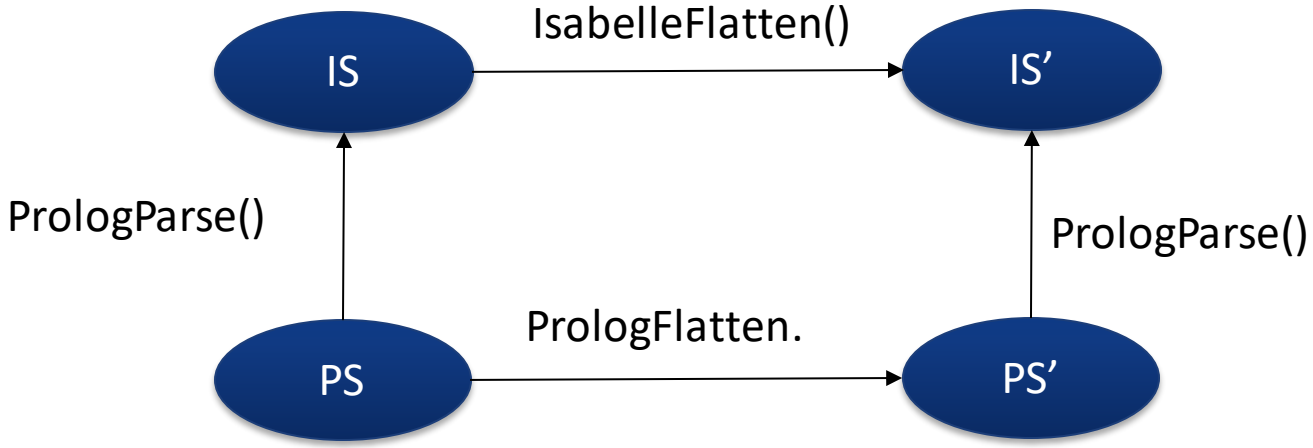
Future Directions: Verification of the Runtime Model



- Current state:

- Open: Is the Prolog representation correct?

- Proof Framework for Prolog programs.



Decoding Net / Sockeye for seL4?

- Well-defined description of target platform
- Correct-by-construction initial state
- Reasoning about
  - multi-level translation schemes
  - Memory accesses from devices / co-processors / …

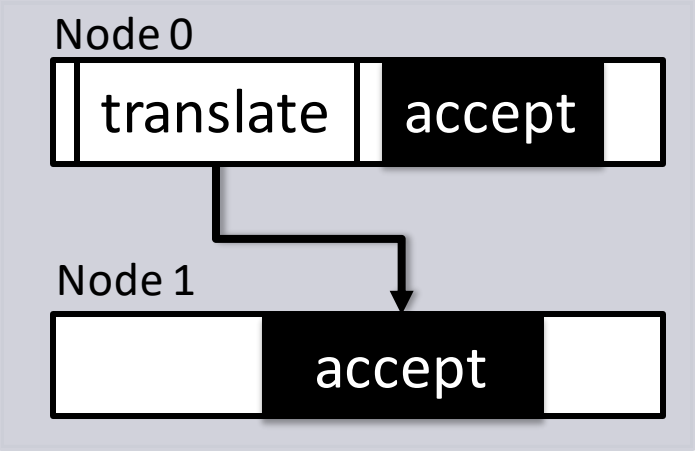
Thanks to my collaborators

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	Ashish Panwar	Many contributors to the Barrelfish OS	…

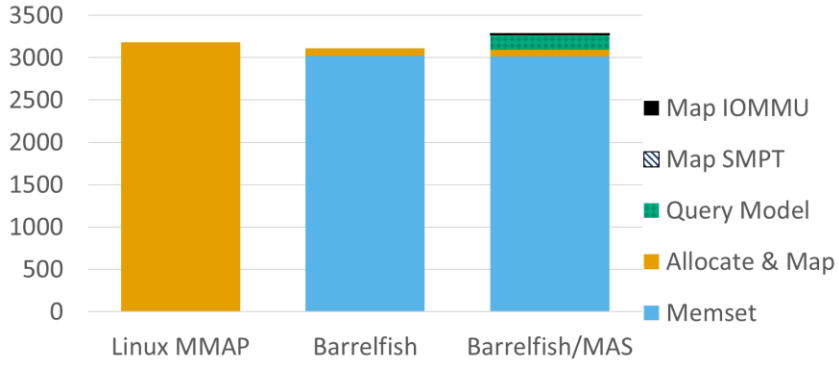
List of Related Publications

- **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).

Summary



Barrelfish/MAS



Decoding Net
Accurate representation of the memory subsystem of a platform.

OS Implementation
Address space aware capability system and OS code generation

Efficient Implementation
Detailed memory topology model, at low overhead.