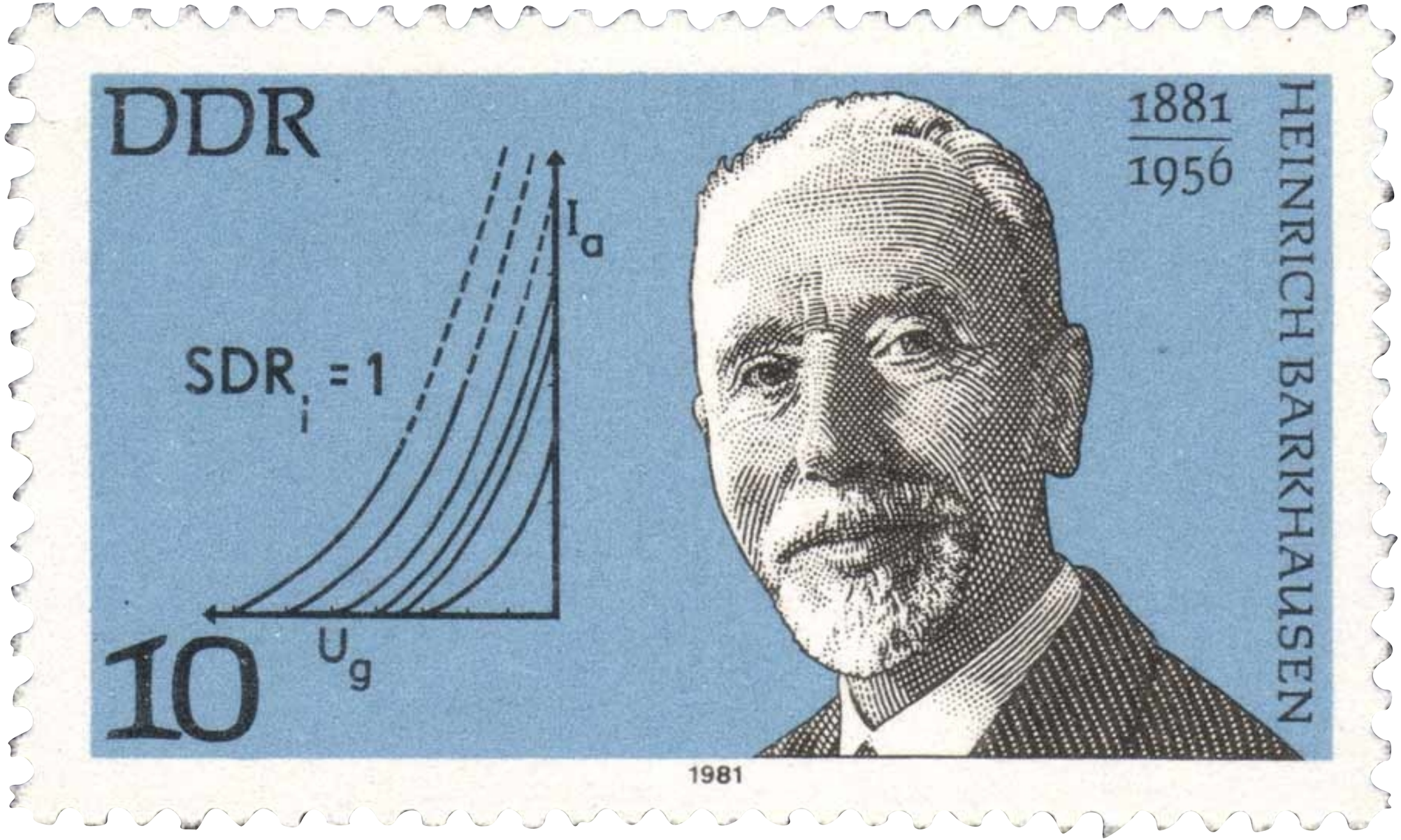
barkhauseninstitut.org

barkhausen institut

M^3 and Beyond: Secure Systems with Fast Enforcement

Michael Roitzsch


DDR
SDR i = 1
Ia
Ug
10
1881
1956
HEINRICH BARKHAUSEN
1981

Trustworthiness for the Tactile Internet of Things

Composability = Isolation + Cooperation

Mission

Composability =

Isolation +

Cooperation

Mission

Composability =

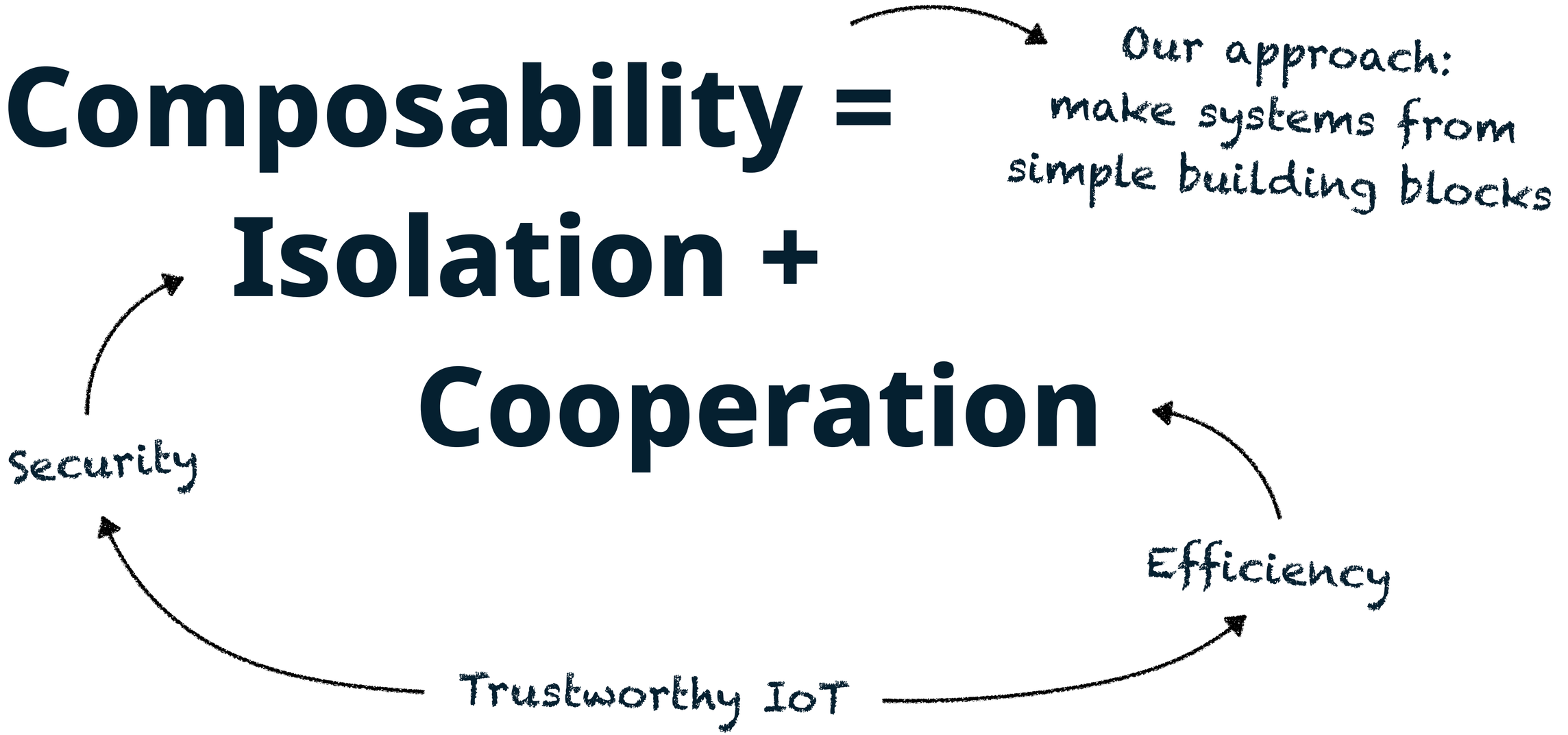
Isolation +

Cooperation

Mission

Composability = Isolation + Cooperation

Mission


Composability =
Isolation +
Cooperation
Our approach:
make systems from
simple building blocks
Security
Efficiency
Trustworthy IoT

Agenda

The M^3 Journey in Three Papers

Lessons Learned from M^3

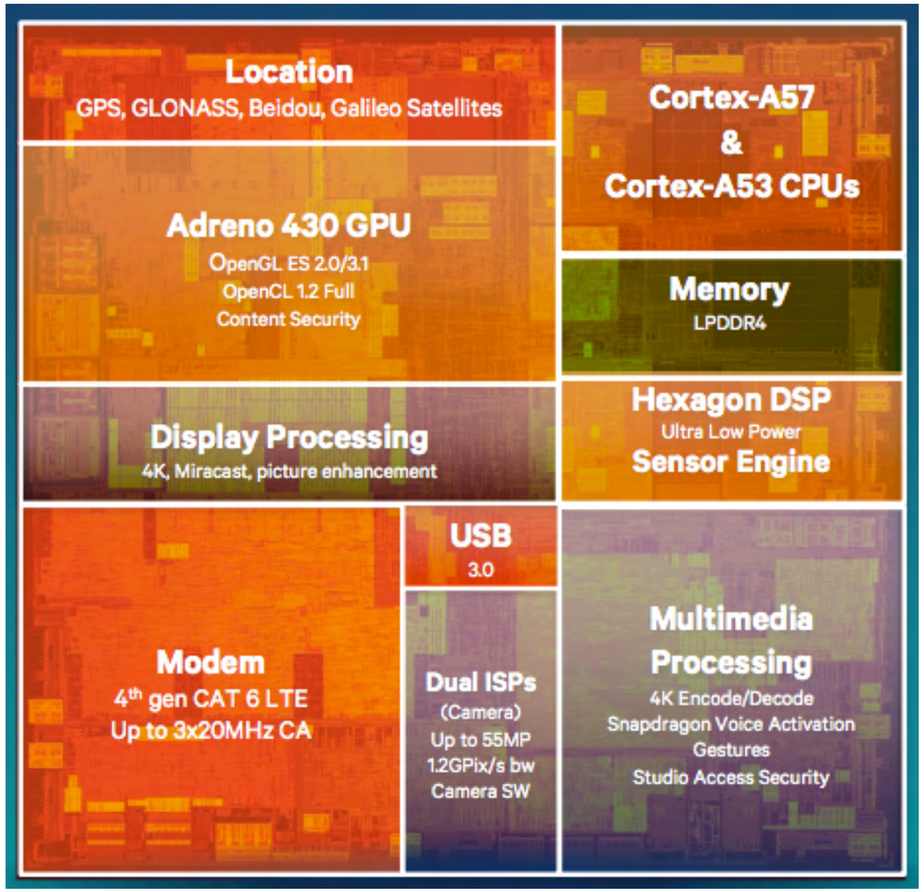
Future Directions

barkhausen institut

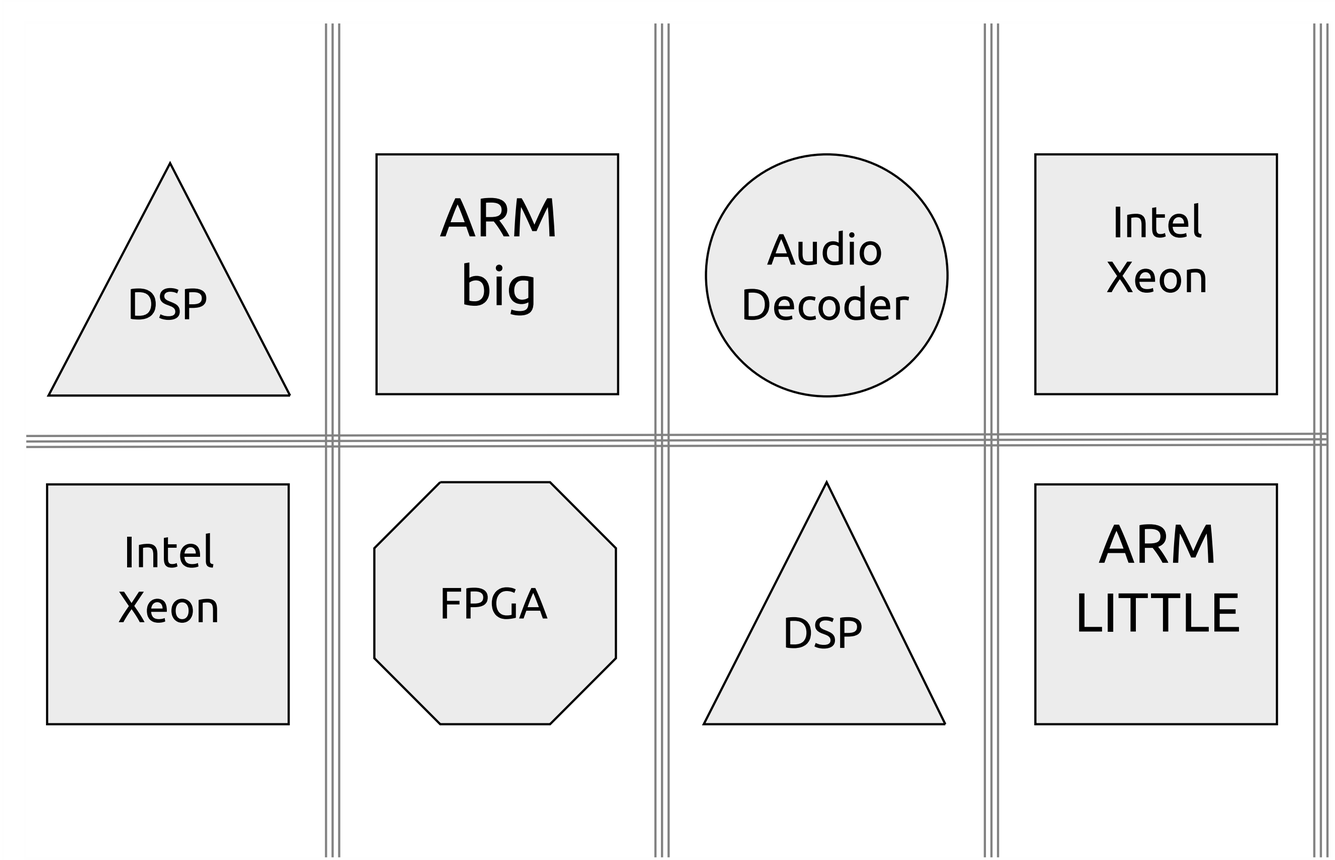
M^3 — Trustworthy SoCs for IoT and Edge

ASPLOS '16

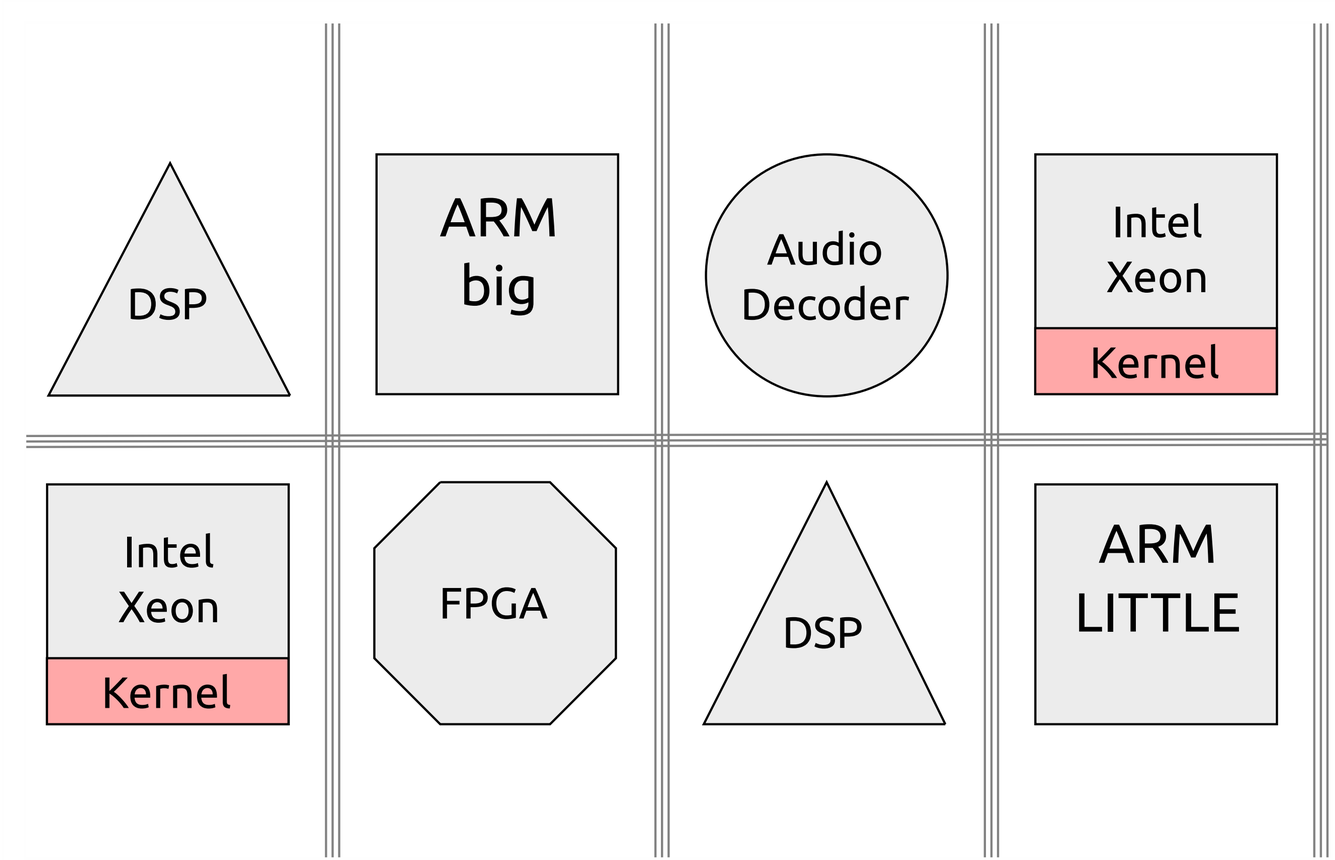
Motivation: Heterogeneous SoCs



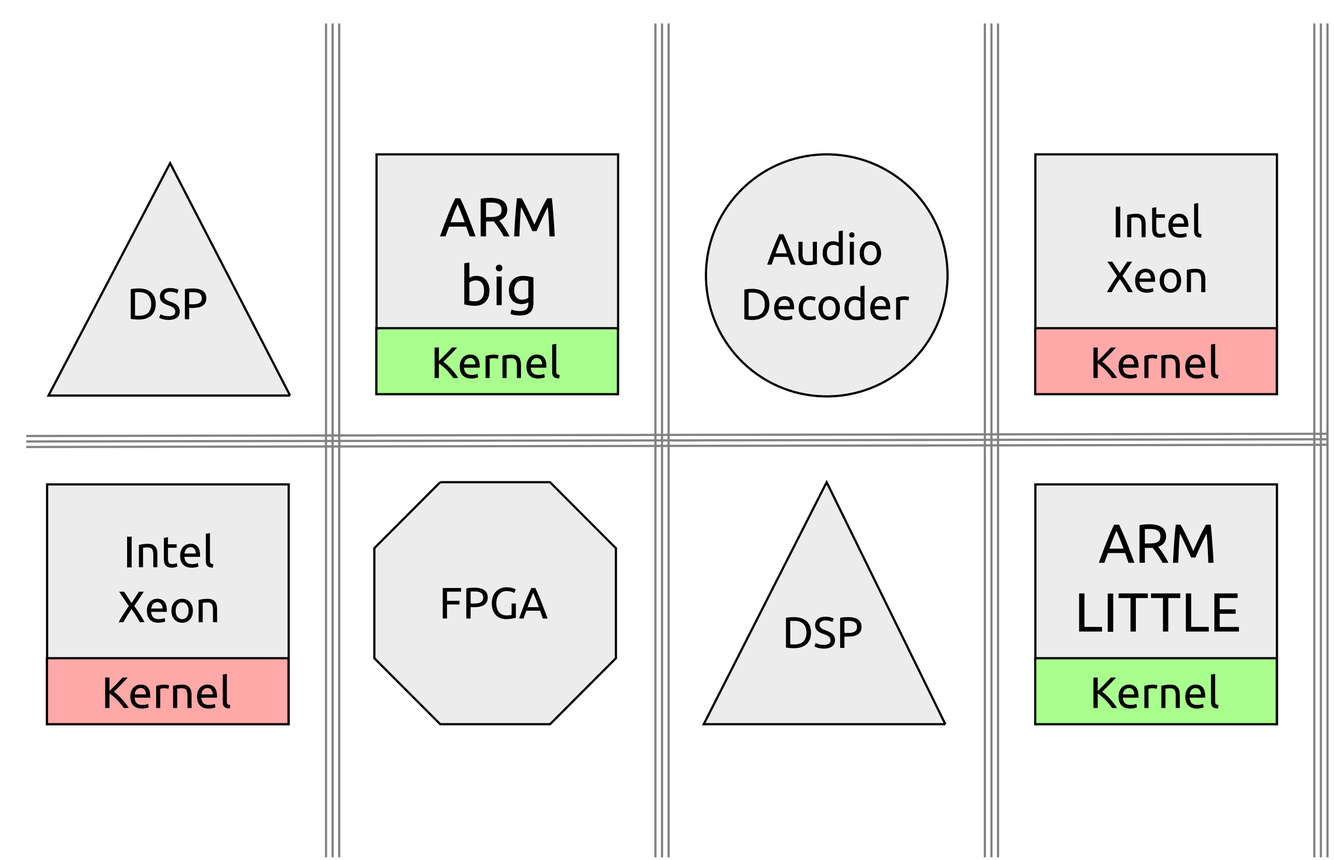
The Problem


DSP
ARM
big
Audio
Decoder
Intel
Xeon
Intel
Xeon
FPGA
DSP
ARM
LITTLE

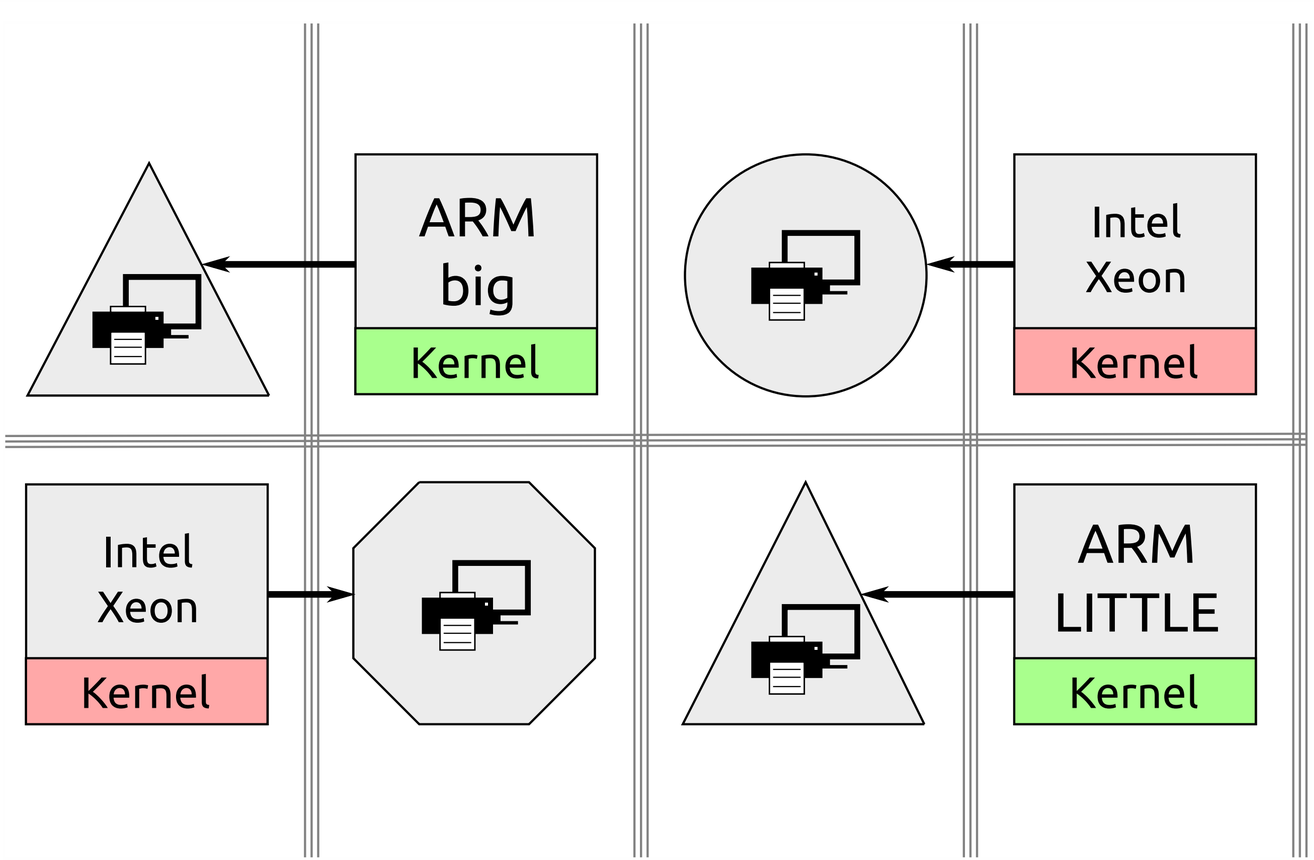
The Problem


DSP
ARM
big
Audio
Decoder
Intel
Xeon
Kernel
Intel
Xeon
Kernel
FPGA
DSP
ARM
LITTLE

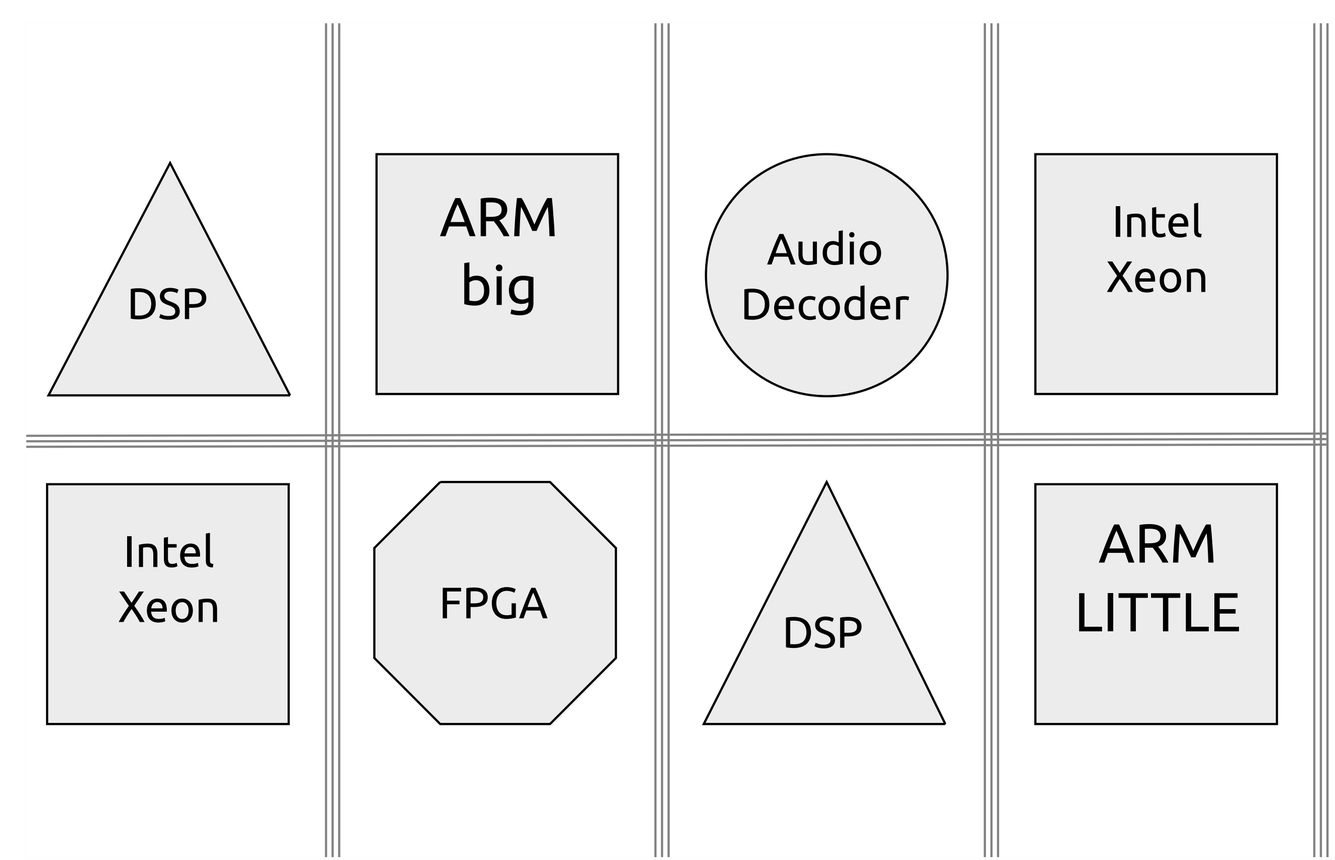
The Problem


DSP
ARM
big
Kernel
Audio
Decoder
Intel
Xeon
Kernel
Intel
Xeon
Kernel
FPGA
DSP
ARM
LITTLE
Kernel

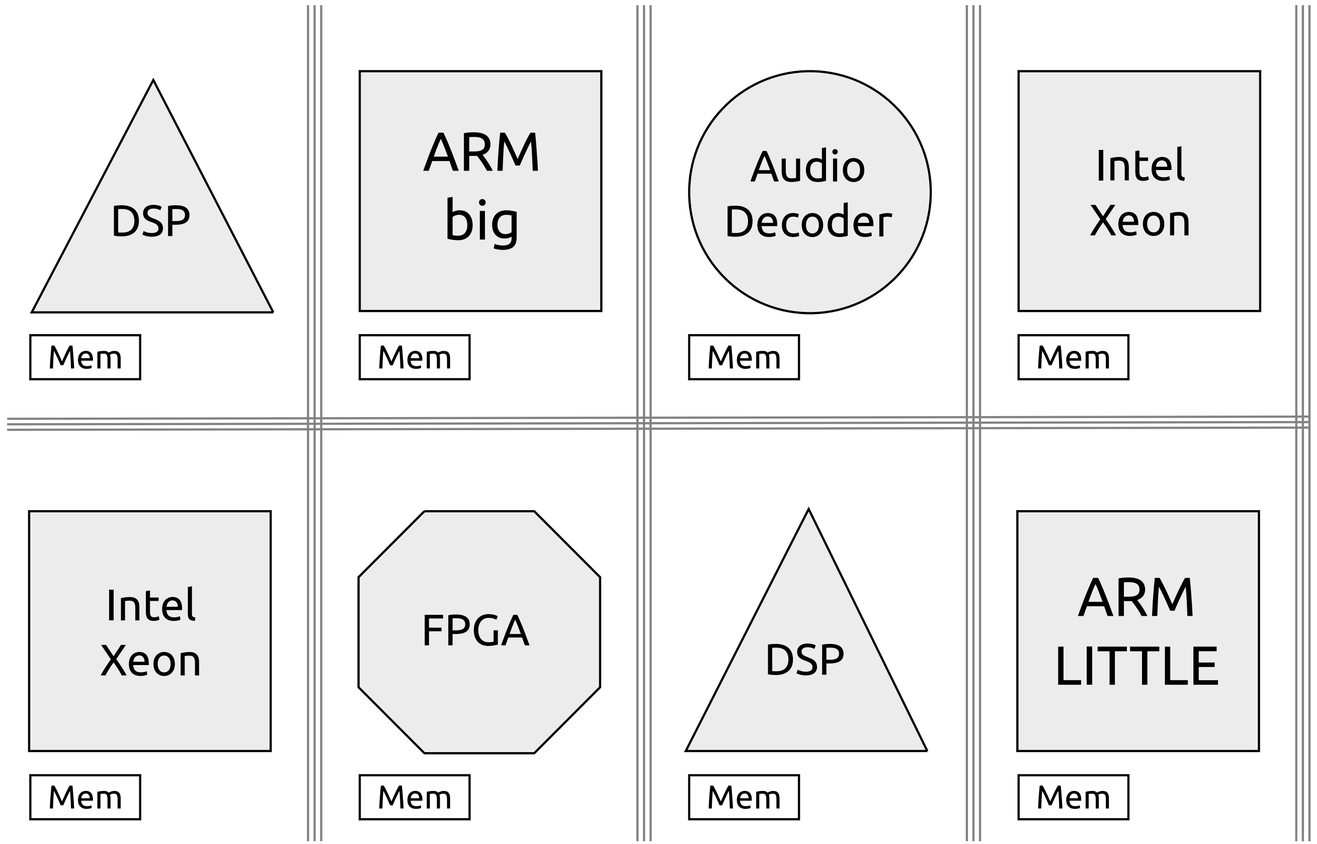
The Problem


ARM
big
Kernel
Intel
Xeon
Kernel
Intel
Xeon
Kernel
ARM
LITTLE
Kernel

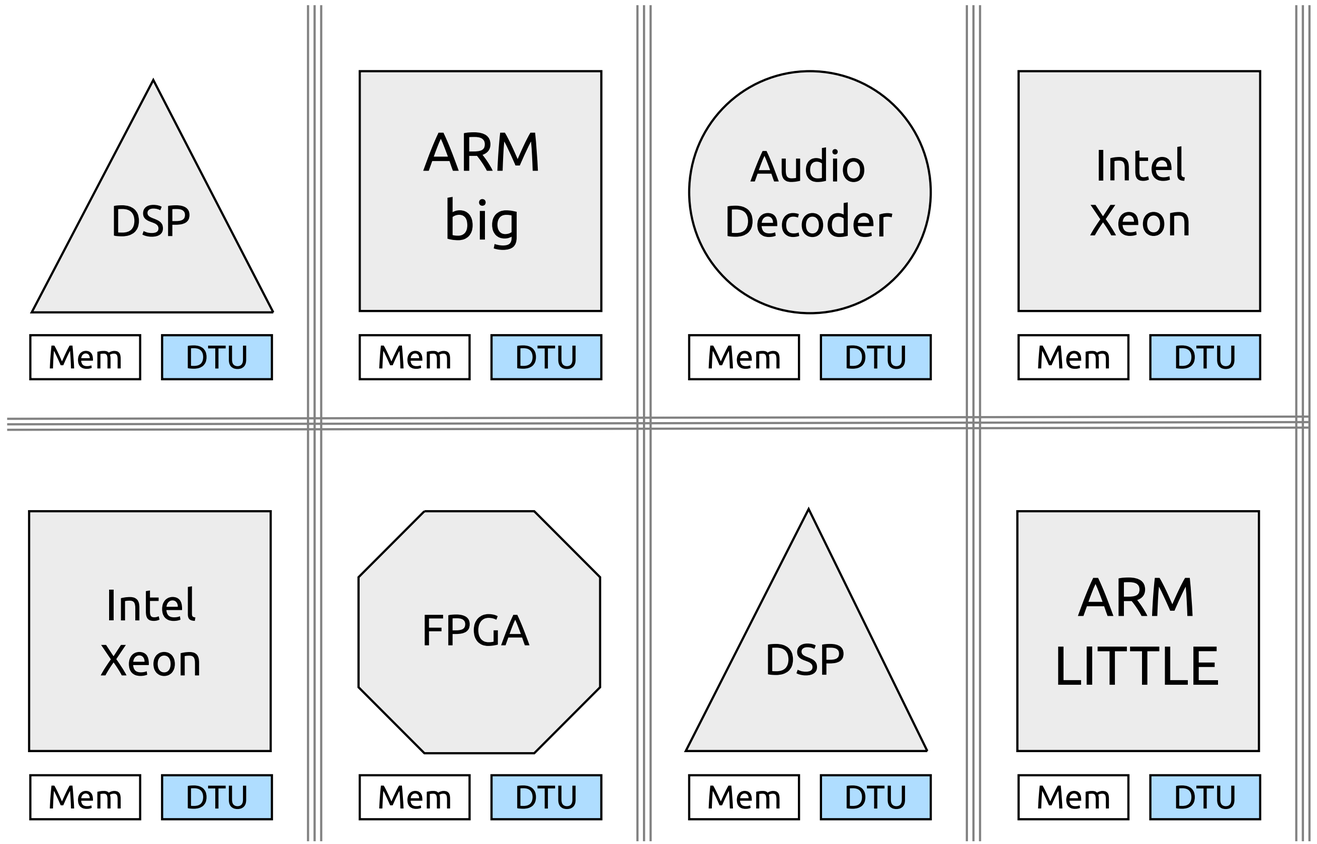
Our Approach



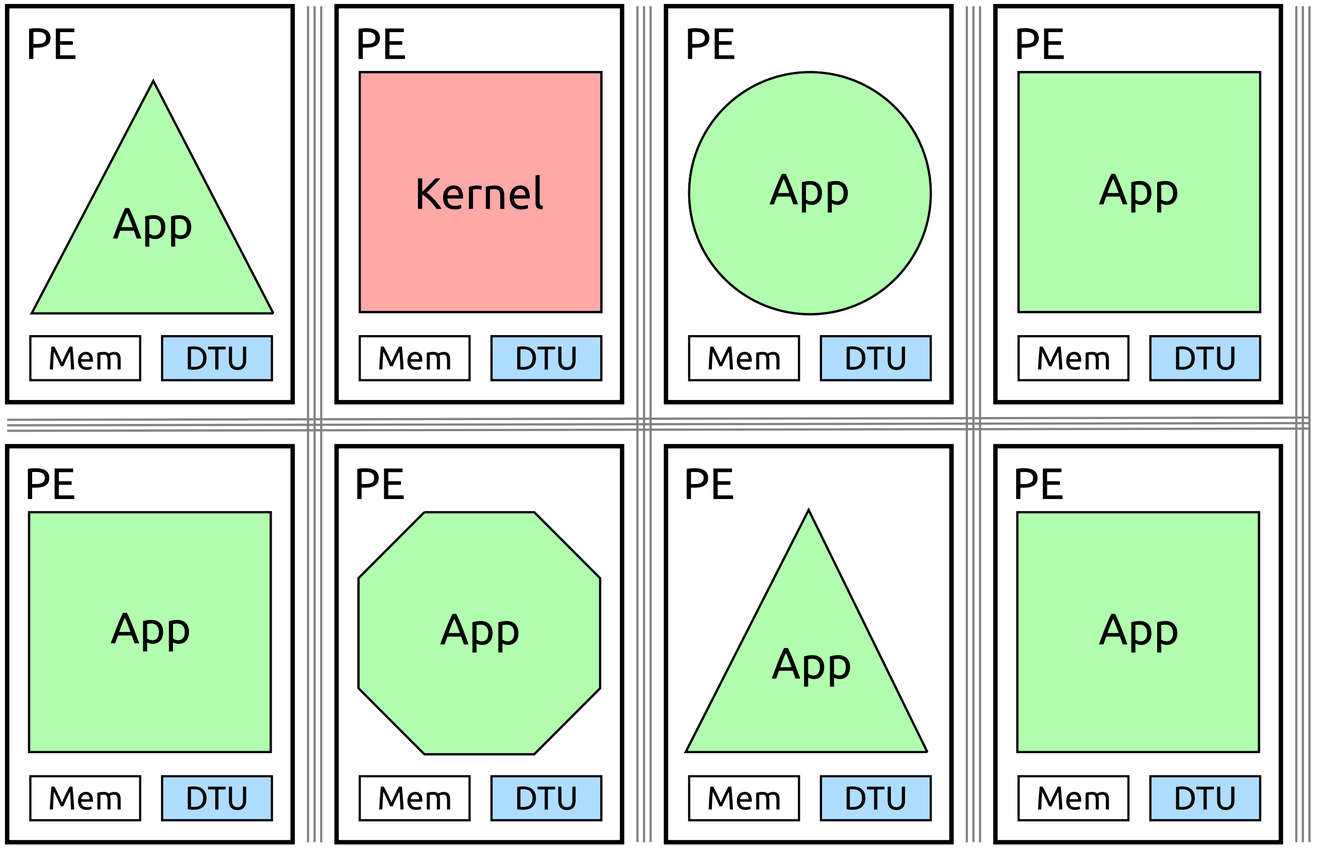
Our Approach



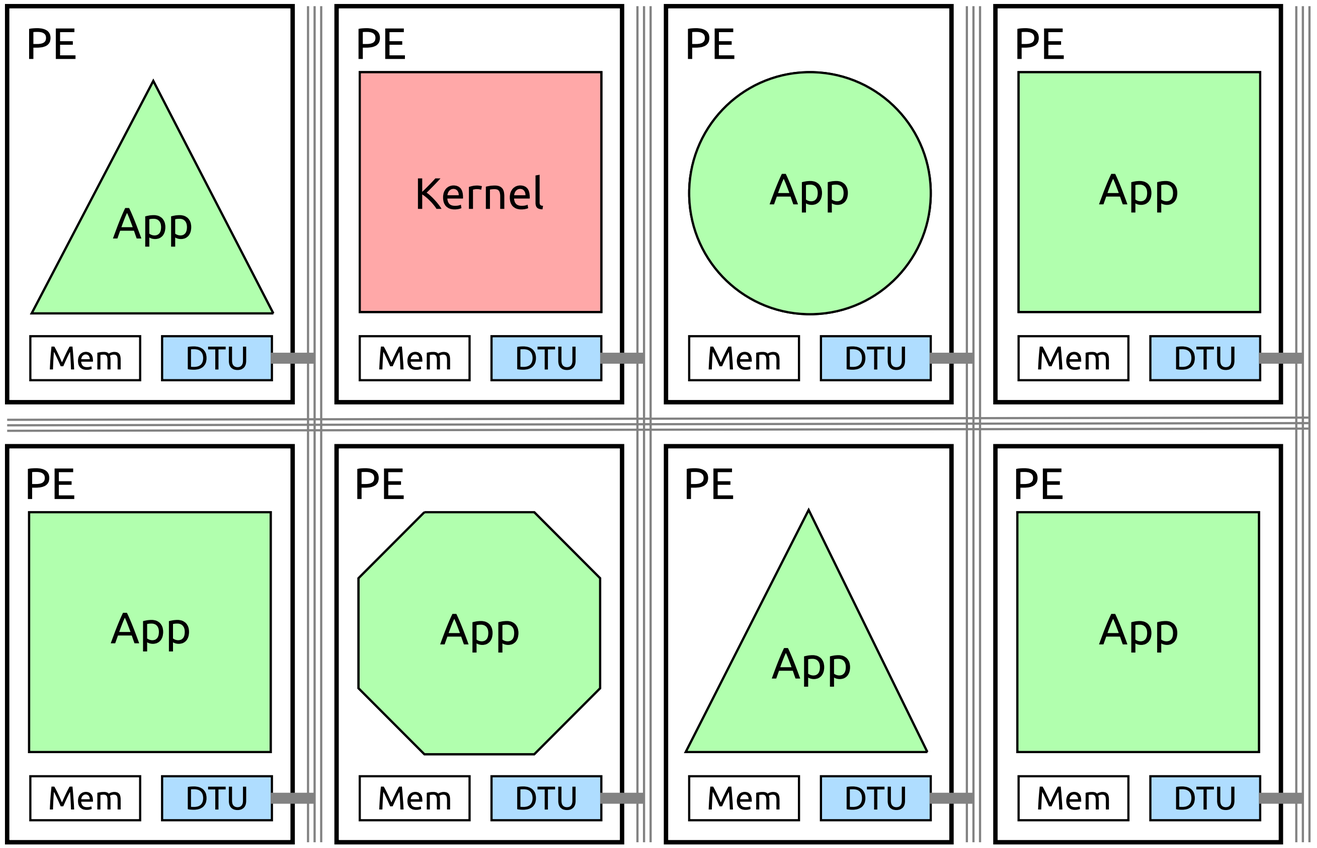
Our Approach



Our Approach



Our Approach



Data Transfer Unit

Data Transfer Unit

- supports **memory access** and **message passing**

Data Transfer Unit

- supports **memory access** and **message passing**
- provides a number of **endpoints**

Data Transfer Unit

- supports **memory access** and **message passing**
- provides a number of **endpoints**
- each endpoint can be configured for:

Data Transfer Unit

- supports **memory access** and **message passing**
- provides a number of **endpoints**
- each endpoint can be configured for:
  - accessing a contiguous piece of memory

Data Transfer Unit

- supports **memory access** and **message passing**
- provides a number of **endpoints**
- each endpoint can be configured for:
  - accessing a contiguous piece of memory
  - receiving messages into a ring buffer

Data Transfer Unit

- supports **memory access** and **message passing**
- provides a number of **endpoints**
- each endpoint can be configured for:
  - accessing a contiguous piece of memory
  - receiving messages into a ring buffer
  - sending messages to a receive endpoint

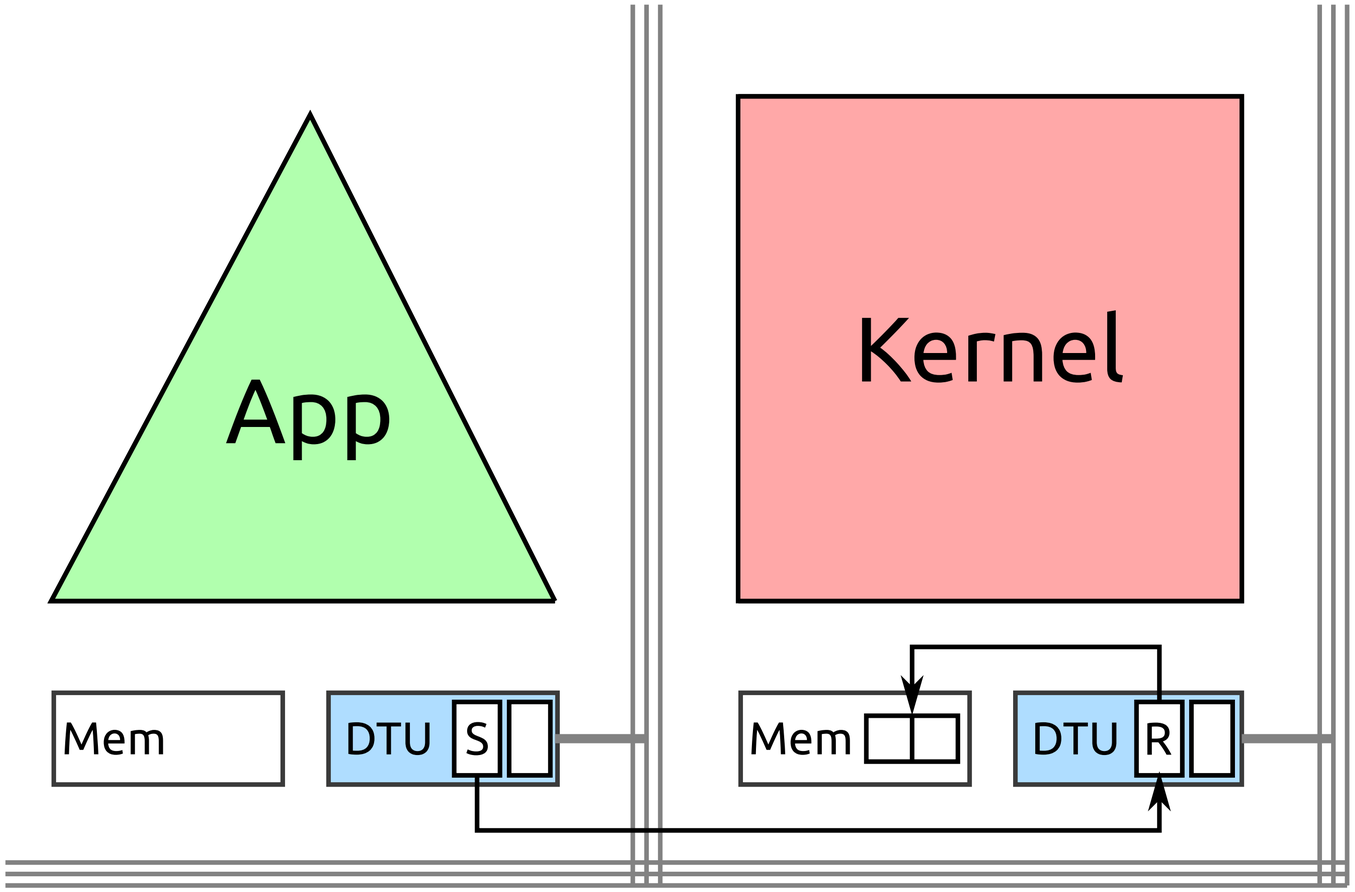
Data Transfer Unit

- supports **memory access** and **message passing**
- provides a number of **endpoints**
- each endpoint can be configured for:
  - accessing a contiguous piece of memory
  - receiving messages into a ring buffer
  - sending messages to a receive endpoint
- configuration **only by kernel**

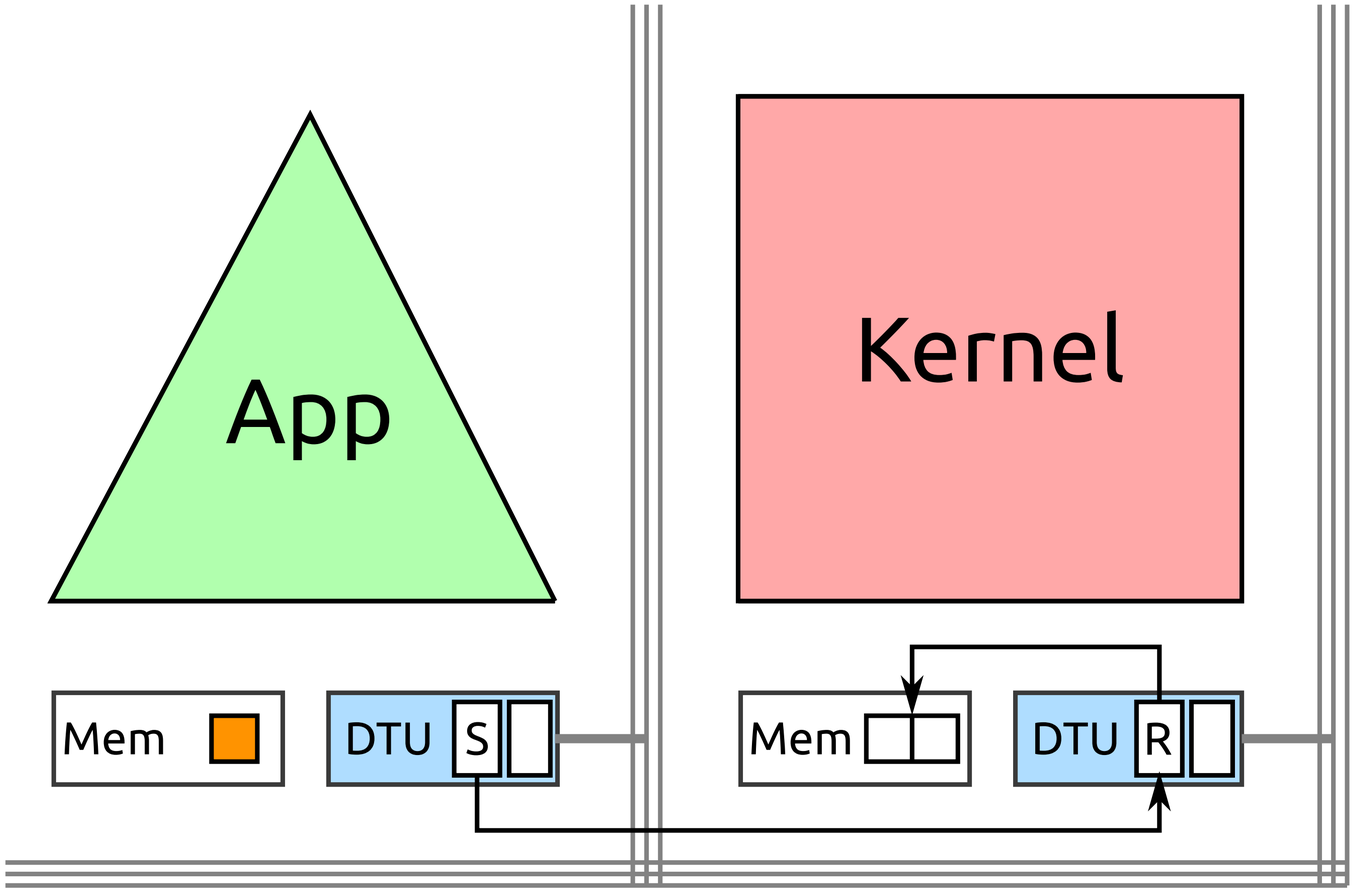
Data Transfer Unit

- supports **memory access** and **message passing**
- provides a number of **endpoints**
- each endpoint can be configured for:
  - accessing a contiguous piece of memory
  - receiving messages into a ring buffer
  - sending messages to a receive endpoint
- configuration **only by kernel**
- used **directly** by application

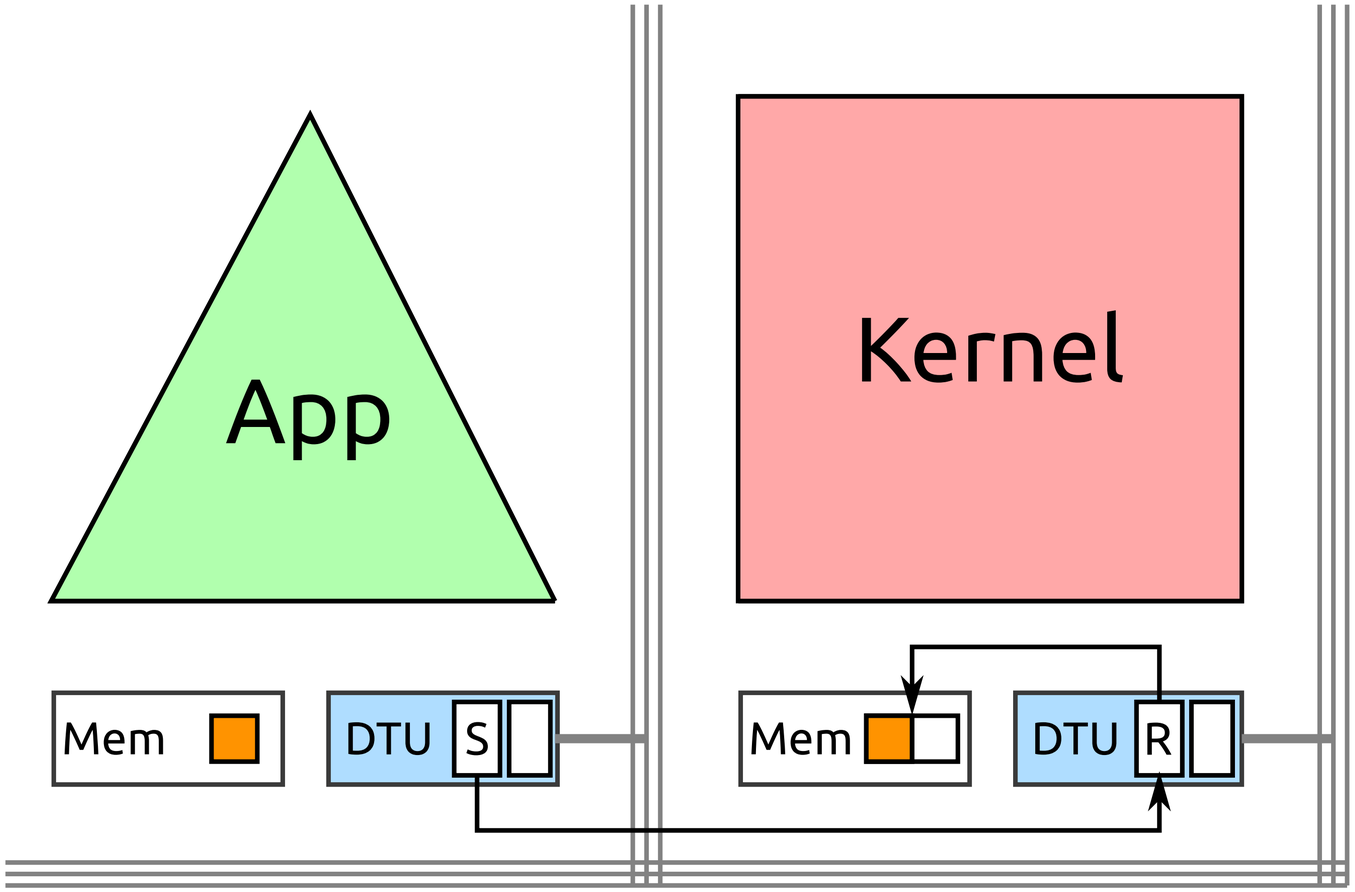
Data Transfer Example



Data Transfer Example



Data Transfer Example



$$M^3 = L4 \pm 1$$

$M^3 = L4 \pm 1$

- microkernel-based system for heterogeneous many-cores

M^3 = L4 ± 1

- microkernel-based system for heterogeneous many-cores
- implemented originally in C++, since 2019 in **Rust**

$M^3 = L4 \pm 1$

- microkernel-based system for heterogeneous many-cores
- implemented originally in C++, since 2019 in **Rust**
- mechanisms for running code on tiles, memory, and communication

$M^3 = L4 \pm 1$

- microkernel-based system for heterogeneous many-cores
- implemented originally in C++, since 2019 in **Rust**
- mechanisms for running code on tiles, memory, and communication
- **system services** implemented outside the kernel

M^3 = L4 ± 1

- microkernel-based system for heterogeneous many-cores
- implemented originally in C++, since 2019 in **Rust**
- mechanisms for running code on tiles, memory, and communication
- **system services** implemented outside the kernel
- kernel manages permissions using **capabilities**

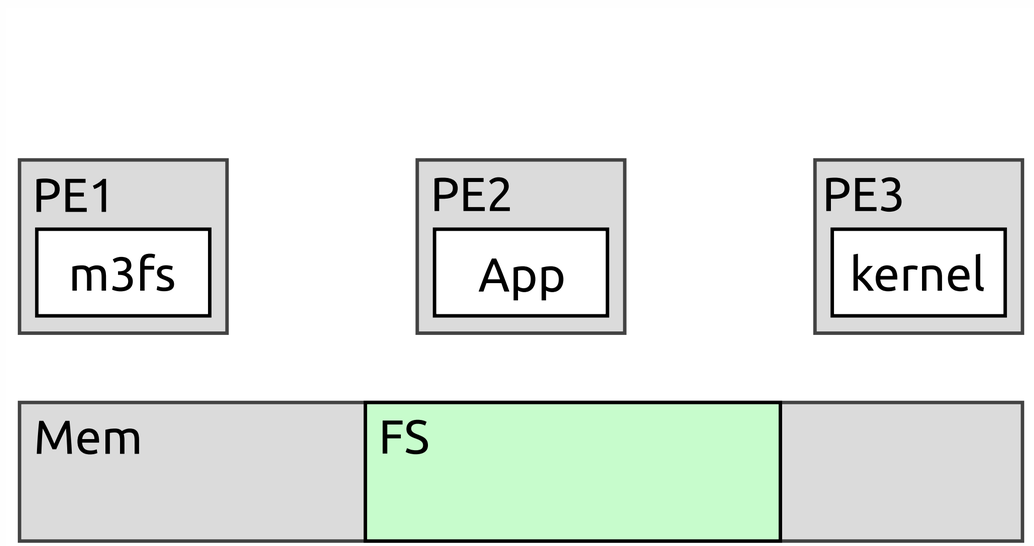
M^3 = L4 ± 1

- microkernel-based system for heterogeneous many-cores
- implemented originally in C++, since 2019 in **Rust**
- mechanisms for running code on tiles, memory, and communication
- **system services** implemented outside the kernel
- kernel manages permissions using **capabilities**
- DTU enforces permissions using endpoints

M^3 = L4 ± 1

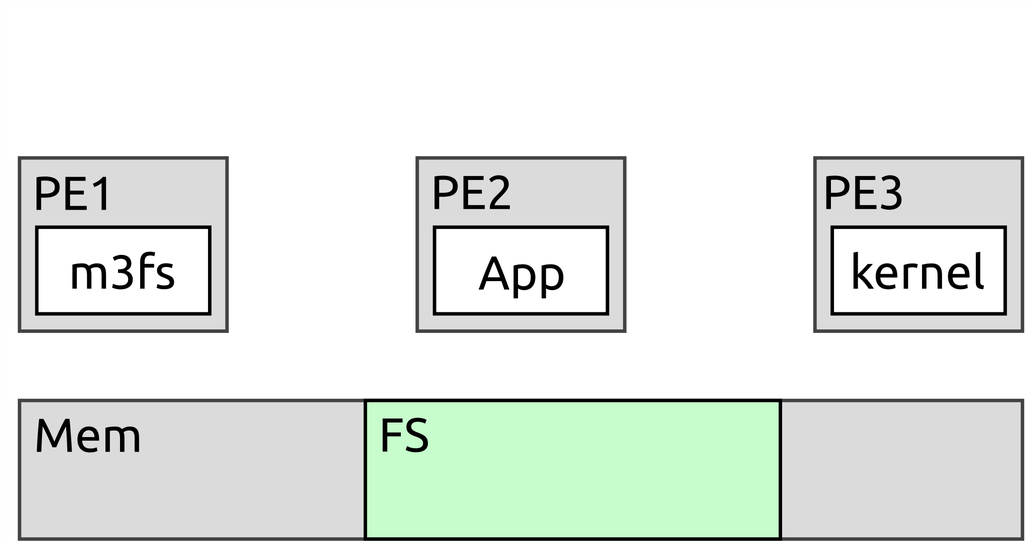
- microkernel-based system for heterogeneous many-cores
- implemented originally in C++, since 2019 in **Rust**
- mechanisms for running code on tiles, memory, and communication
- **system services** implemented outside the kernel
- kernel manages permissions using **capabilities**
- DTU enforces permissions using endpoints
- kernel has no ISA-specific code

File System



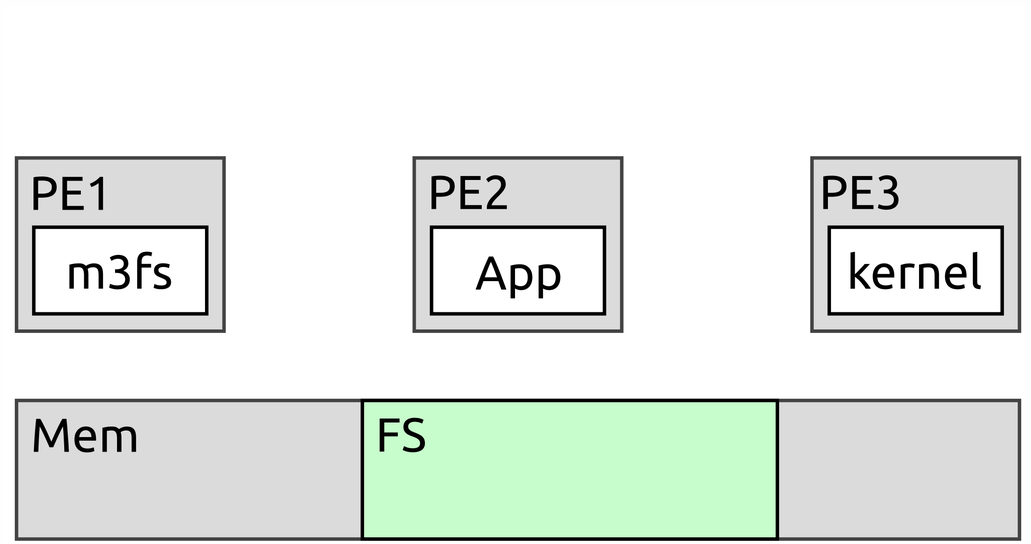
File System

- file system service implemented outside the kernel



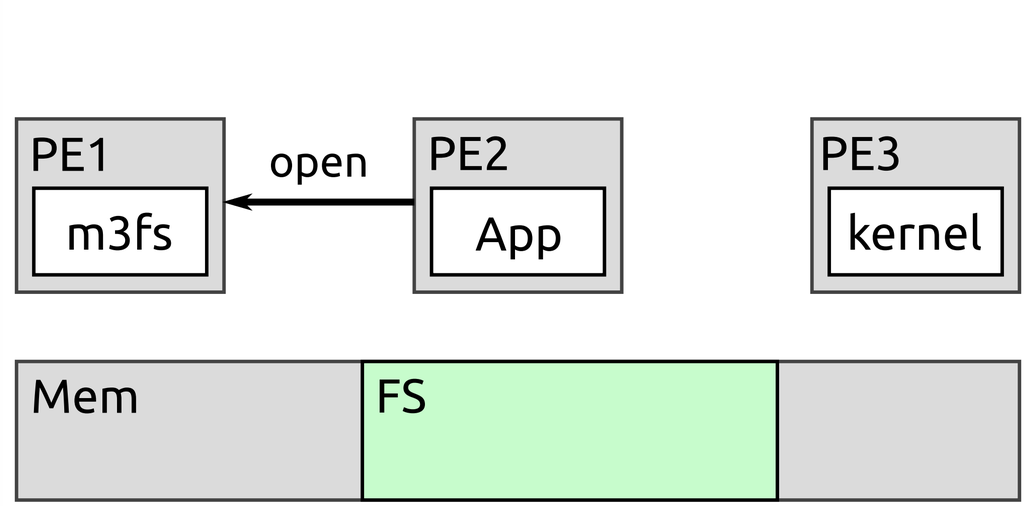
File System

- file system service implemented outside the kernel
- m3fs is currently backed by memory



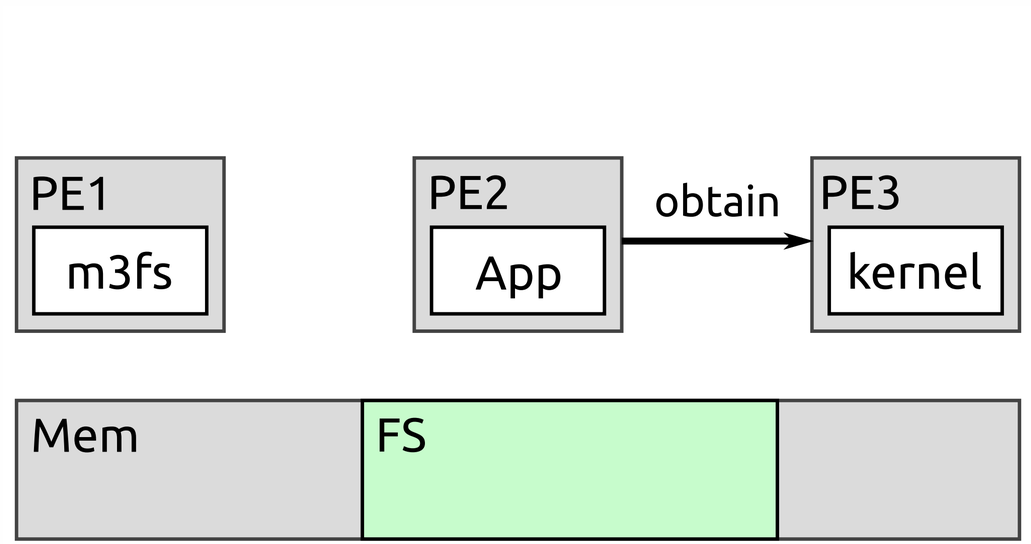
File System

- file system service implemented outside the kernel
- m3fs is currently backed by memory



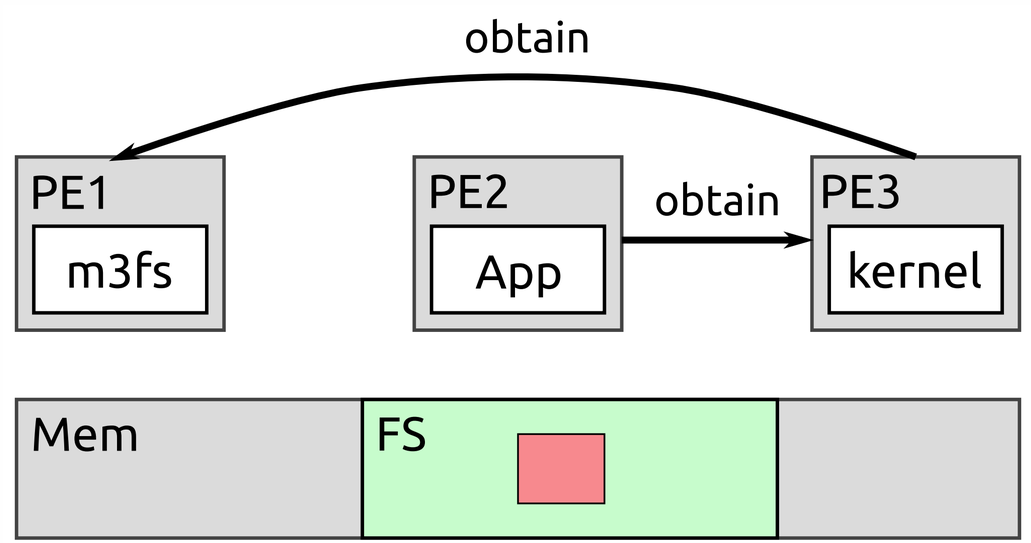
File System

- file system service implemented outside the kernel
- m3fs is currently backed by memory



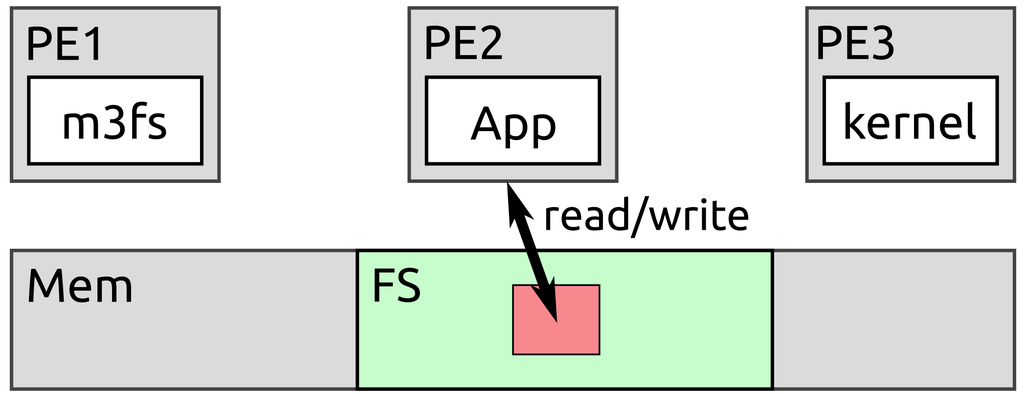
File System

- file system service implemented outside the kernel
- m3fs is currently backed by memory



File System

- file system service implemented outside the kernel
- m3fs is currently backed by memory



Evaluation

Evaluation

- performance comparison of **M³ and Linux 3.18**

Evaluation

- performance comparison of **M³ and Linux 3.18**
- based on Cadence’s cycle-accurate **Xtensa simulator**

Evaluation

- performance comparison of **M³ and Linux 3.18**
- based on Cadence’s cycle-accurate **Xtensa simulator**
  - M³ with scratch pad of 64KiB each for instructions and data

Evaluation

- performance comparison of **M³ and Linux 3.18**
- based on Cadence's cycle-accurate **Xtensa simulator**
  - M³ with scratch pad of 64KiB each for instructions and data
  - Linux with instruction and data caches of 64KiB

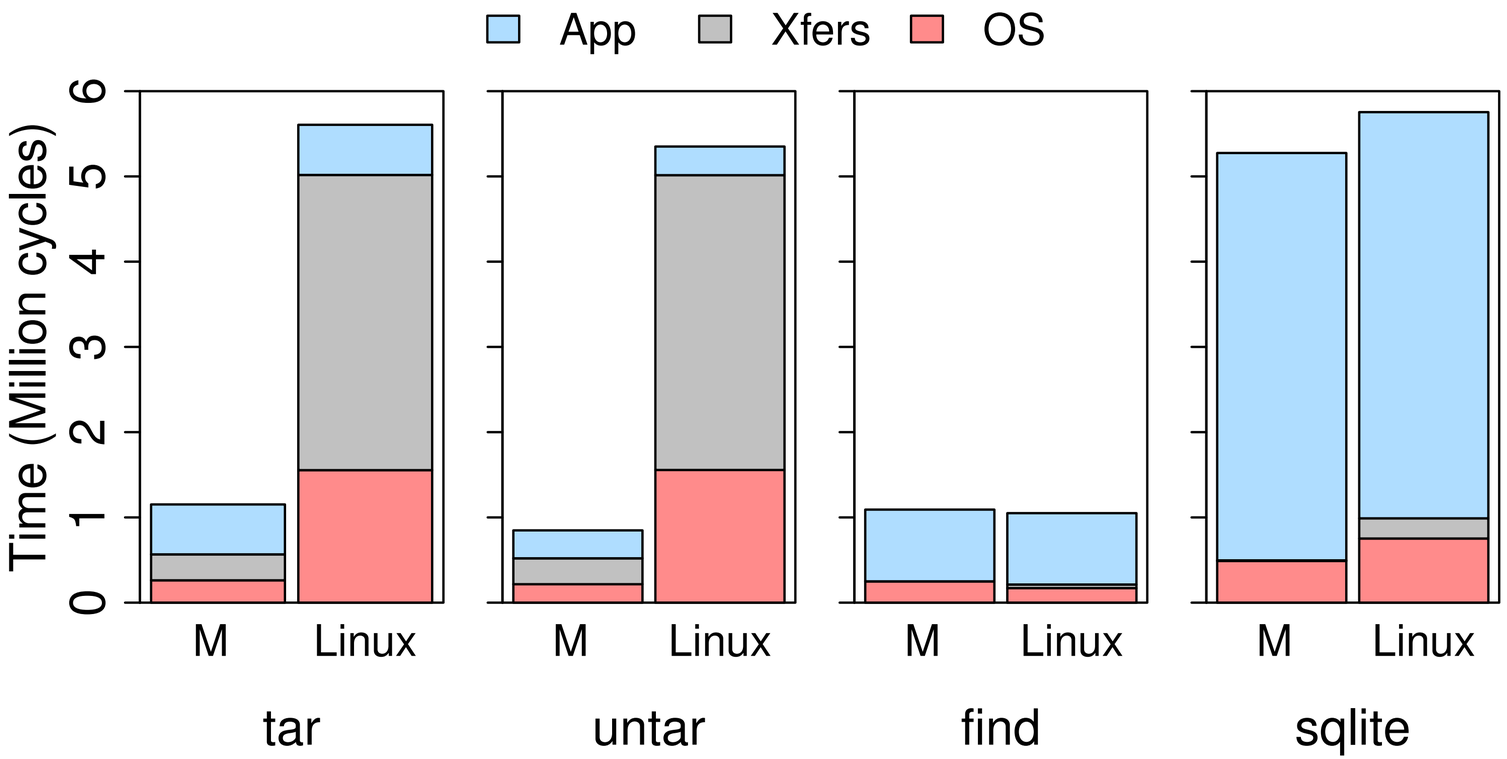
Evaluation

- performance comparison of **M³ and Linux 3.18**
- based on Cadence's cycle-accurate **Xtensa simulator**
  - M³ with scratch pad of 64KiB each for instructions and data
  - Linux with instruction and data caches of 64KiB
- time for cache-line fetch == time for read with DTU

Evaluation

- performance comparison of **M³ and Linux 3.18**
- based on Cadence's cycle-accurate **Xtensa simulator**
  - M³ with scratch pad of 64KiB each for instructions and data
  - Linux with instruction and data caches of 64KiB
- time for cache-line fetch == time for read with DTU
- **equal compute performance** of the cores

Application-Level Benchmarks



Summary

Summary

A new design point for trustworthy and heterogeneous SoCs

Summary

A new design point for trustworthy and heterogeneous SoCs

- DTU **abstracts heterogeneity** by providing a common interface

Summary

A new design point for trustworthy and heterogeneous SoCs

- DTU **abstracts heterogeneity** by providing a common interface
- M^3 kernel programs DTUs remotely

Summary

A new design point for trustworthy and heterogeneous SoCs

- DTU **abstracts heterogeneity** by providing a common interface
- M³ kernel programs DTUs remotely
- supports **untrusted code** and **untrusted cores** within the tiles

Summary

A new design point for trustworthy and heterogeneous SoCs

- DTU **abstracts heterogeneity** by providing a common interface
- M³ kernel programs DTUs remotely
- supports **untrusted code** and **untrusted cores** within the tiles
- application performance comparable to Linux

Summary

A new design point for trustworthy and heterogeneous SoCs

- DTU **abstracts heterogeneity** by providing a common interface
- M³ kernel programs DTUs remotely
- supports **untrusted code** and **untrusted cores** within the tiles
- application performance comparable to Linux

Limitations

Summary

A new design point for trustworthy and heterogeneous SoCs

- DTU **abstracts heterogeneity** by providing a common interface
- M³ kernel programs DTUs remotely
- supports **untrusted code** and **untrusted cores** within the tiles
- application performance comparable to Linux

Limitations

- rudimentary support for accelerators

Summary

A new design point for trustworthy and heterogeneous SoCs

- DTU **abstracts heterogeneity** by providing a common interface
- M³ kernel programs DTUs remotely
- supports **untrusted code** and **untrusted cores** within the tiles
- application performance comparable to Linux

Limitations

- rudimentary support for accelerators
- no support for caches and virtual memory

Summary

A new design point for trustworthy and heterogeneous SoCs

- DTU **abstracts heterogeneity** by providing a common interface
- M³ kernel programs DTUs remotely
- supports **untrusted code** and **untrusted cores** within the tiles
- application performance comparable to Linux

Limitations

- rudimentary support for accelerators
- no support for caches and virtual memory
- no multiplexing within tiles

Summary

A new design point for trustworthy and heterogeneous SoCs

- DTU **abstracts heterogeneity** by providing a common interface
- M³ kernel programs DTUs remotely
- supports **untrusted code** and **untrusted cores** within the tiles
- application performance comparable to Linux

Limitations

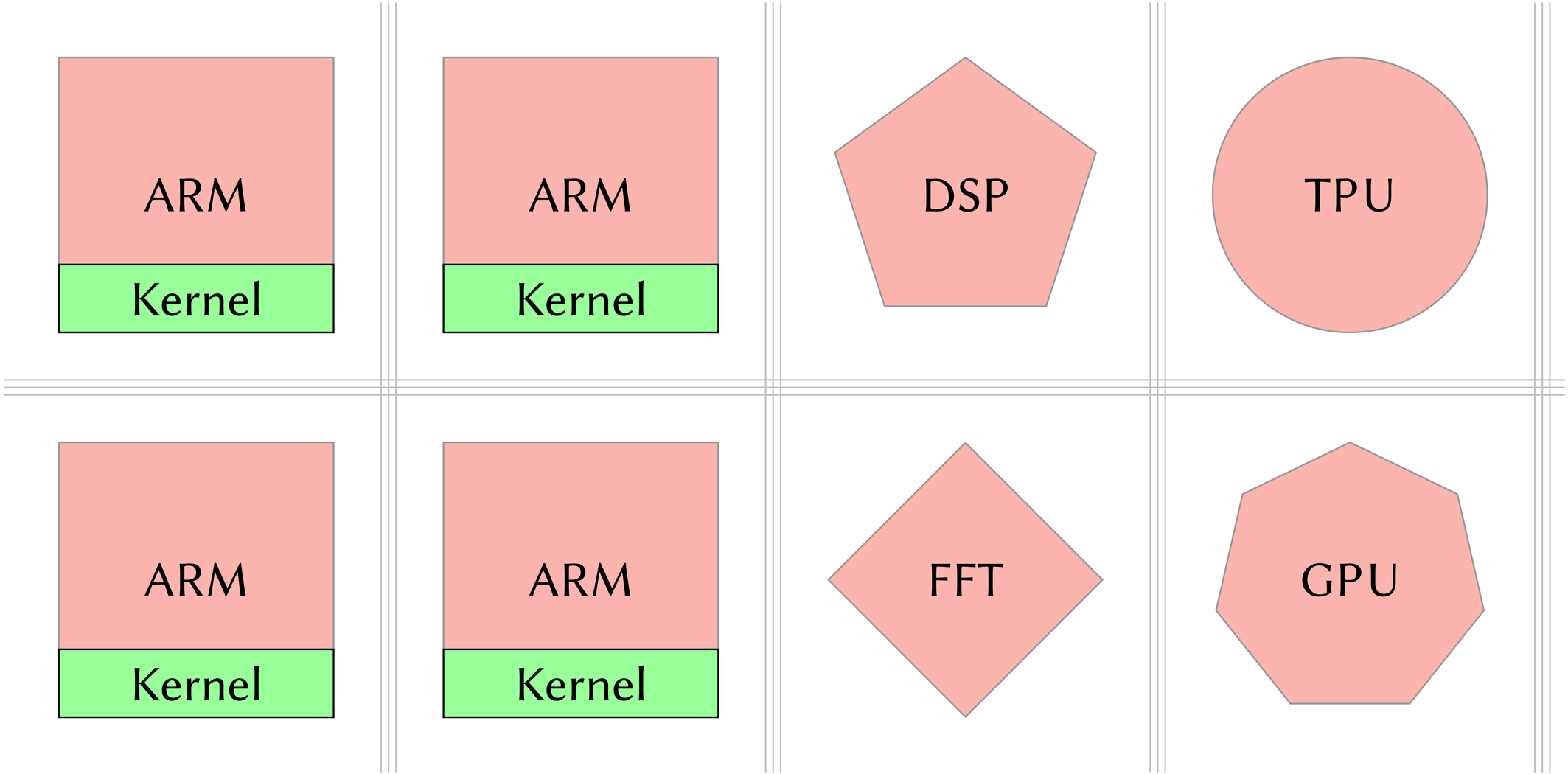
- rudimentary support for accelerators
- no support for caches and virtual memory
- no multiplexing within tiles
- only a single kernel instance

barkhausen institut

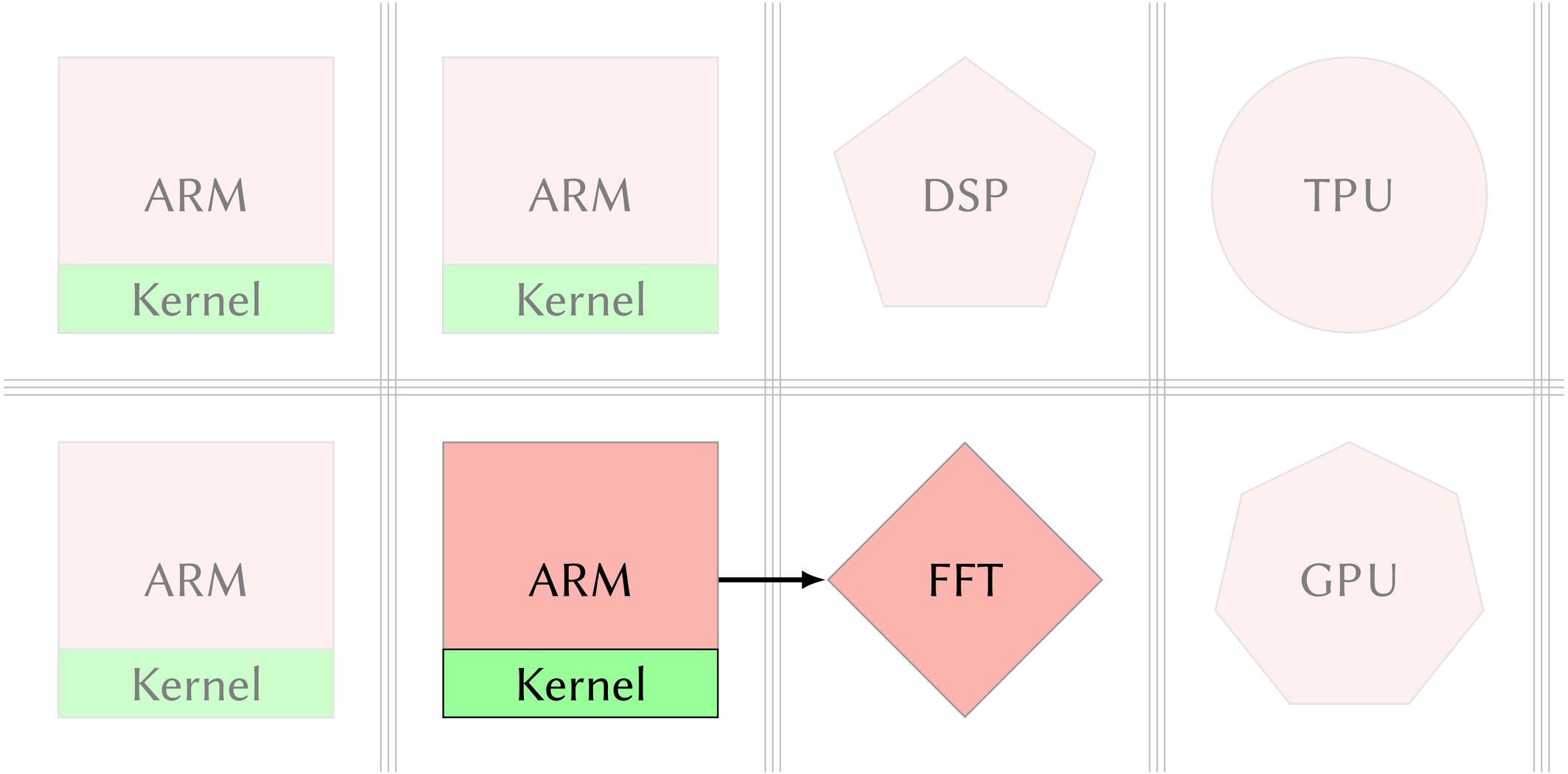
M^3x — Autonomous Accelerator Chains

ATC ’19

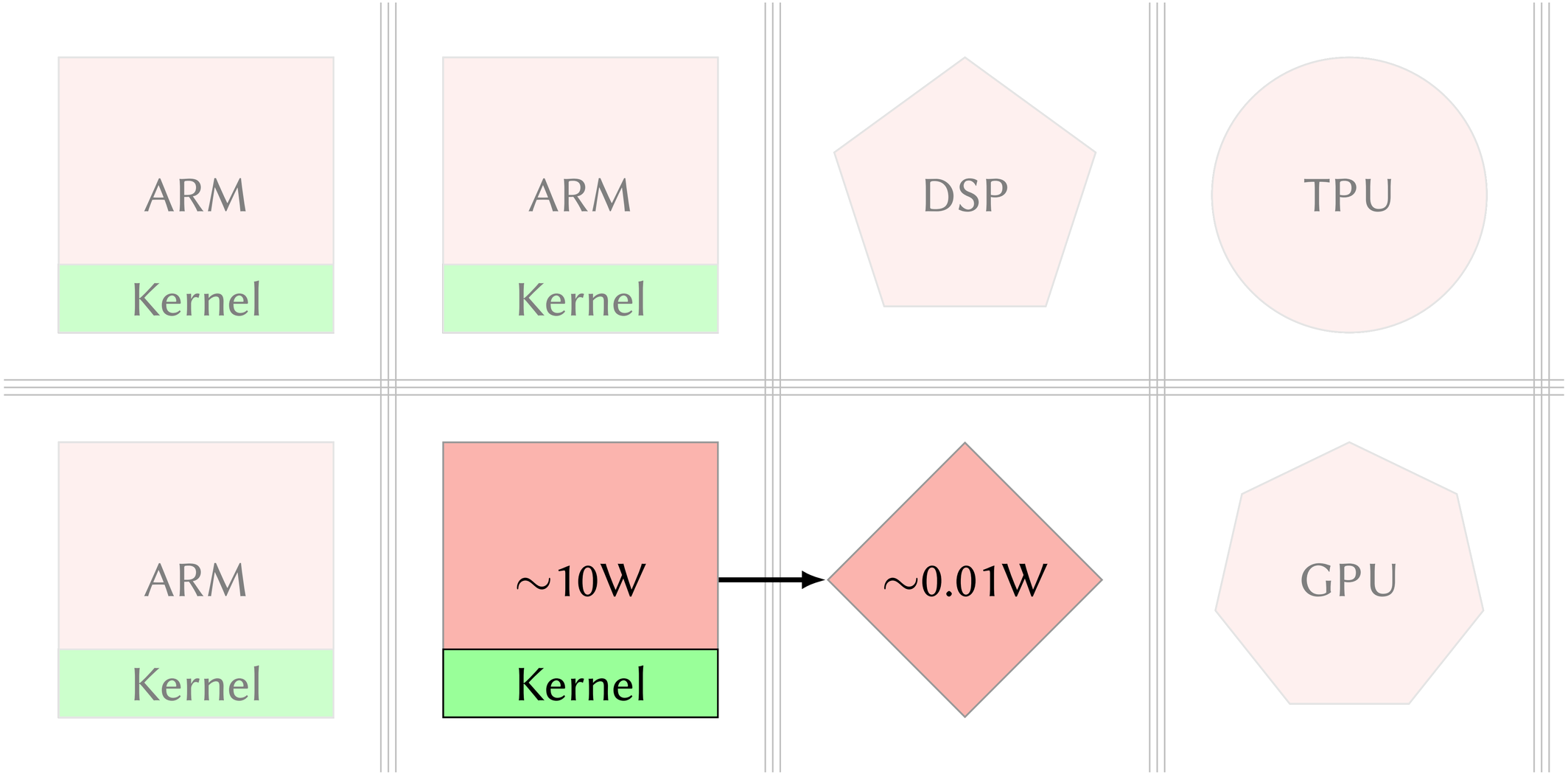
Accelerators and Energy Efficiency



Accelerators and Energy Efficiency



Accelerators and Energy Efficiency



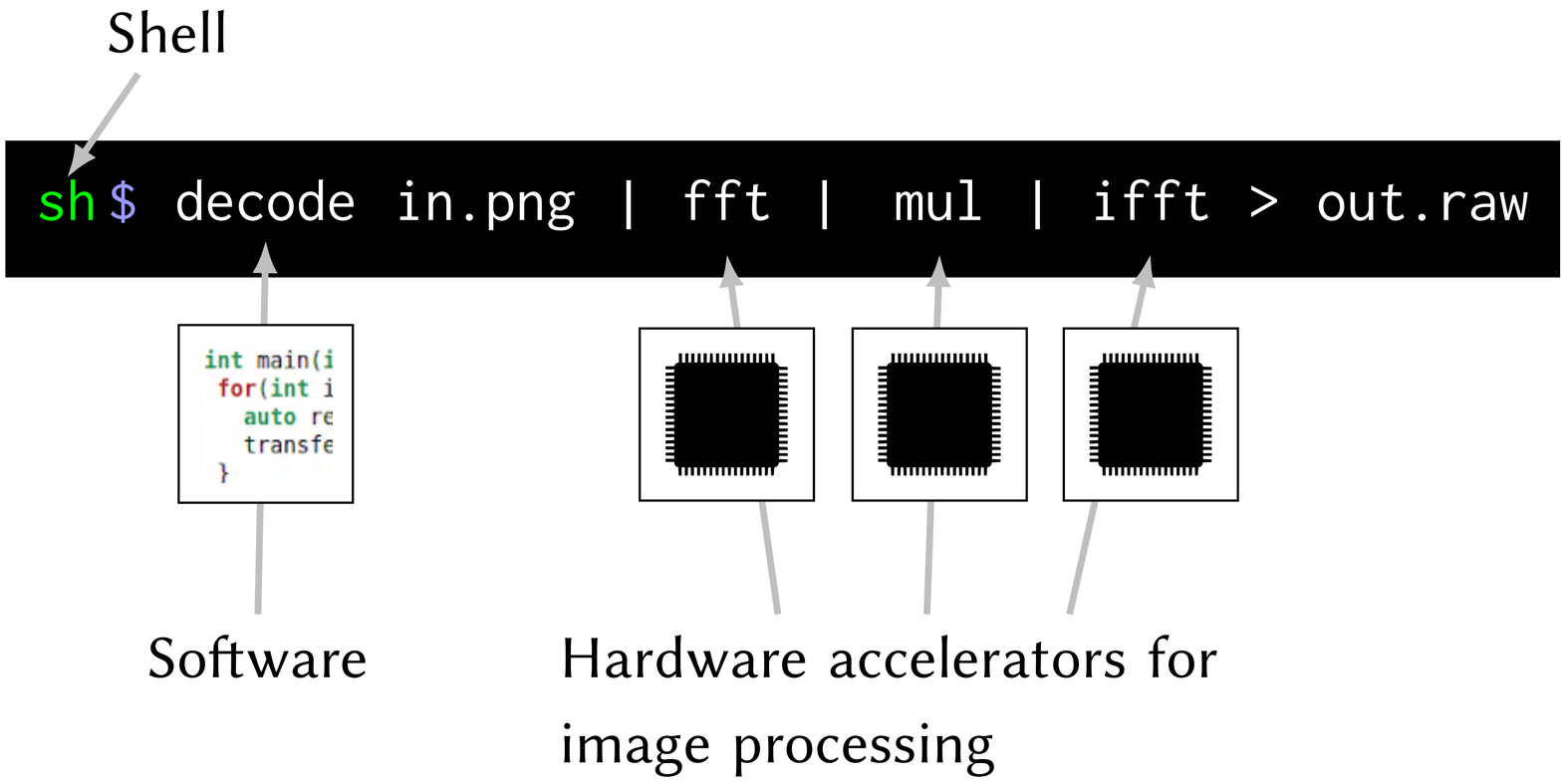
Accelerators and OS Services

```
sh$ decode in.png | fft |  mul | ifft > out.raw
```

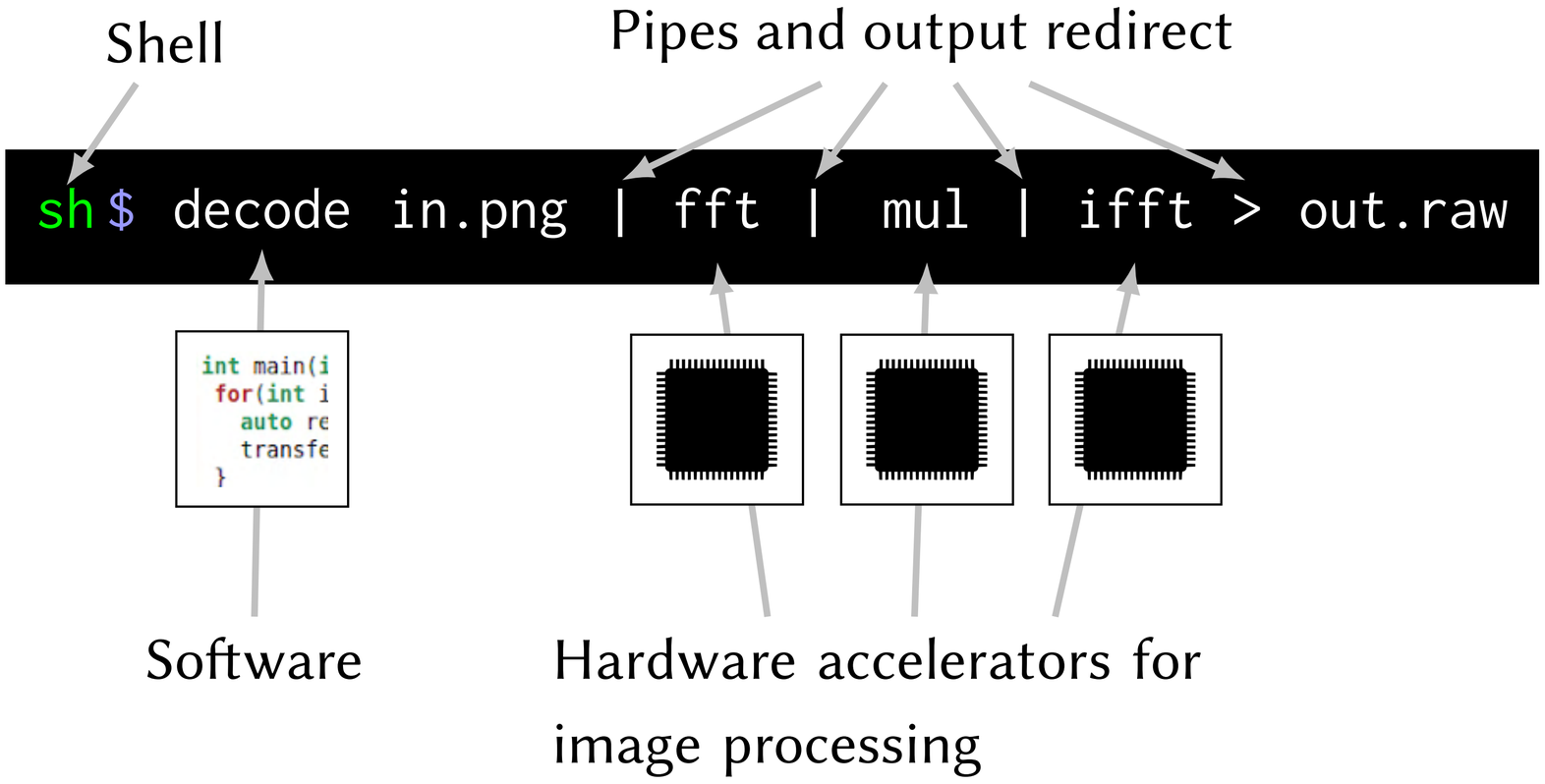
Accelerators and OS Services

Accelerators and OS Services

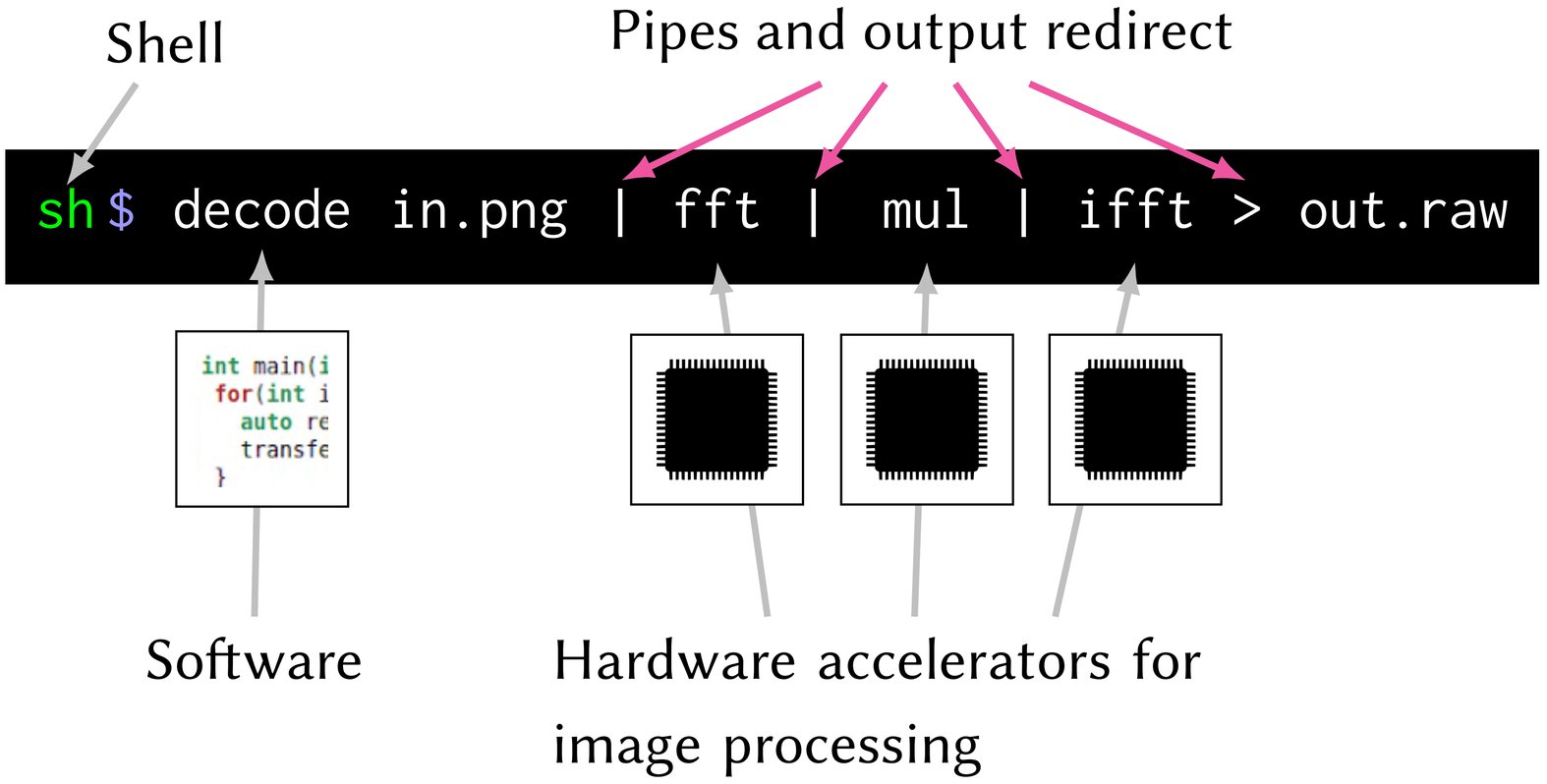
Accelerators and OS Services



Accelerators and OS Services



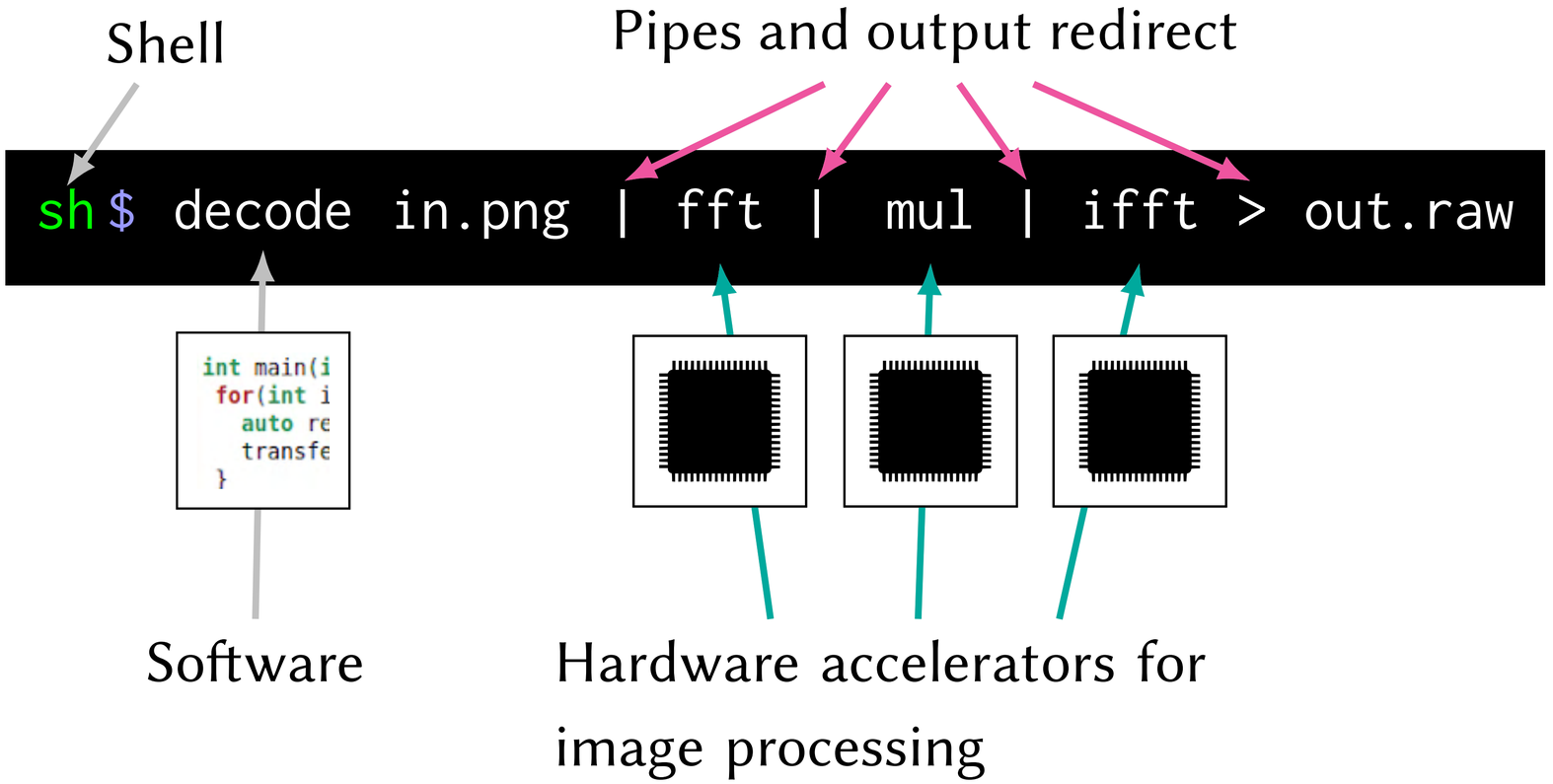
Accelerators and OS Services



Challenges:

- OS must provide generic protocols

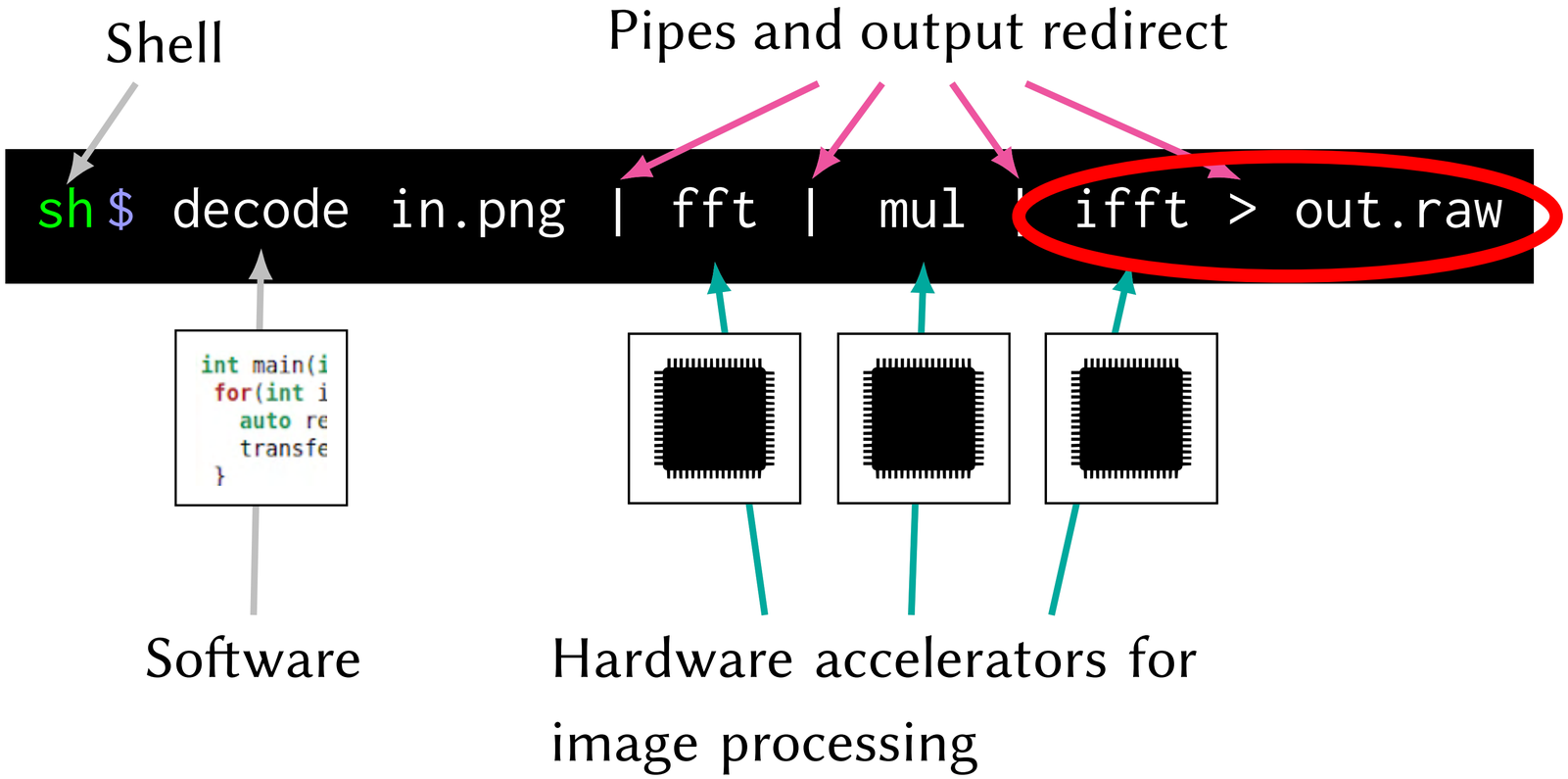
Accelerators and OS Services



Challenges:

- OS must provide generic protocols
- Accelerators need support for protocols

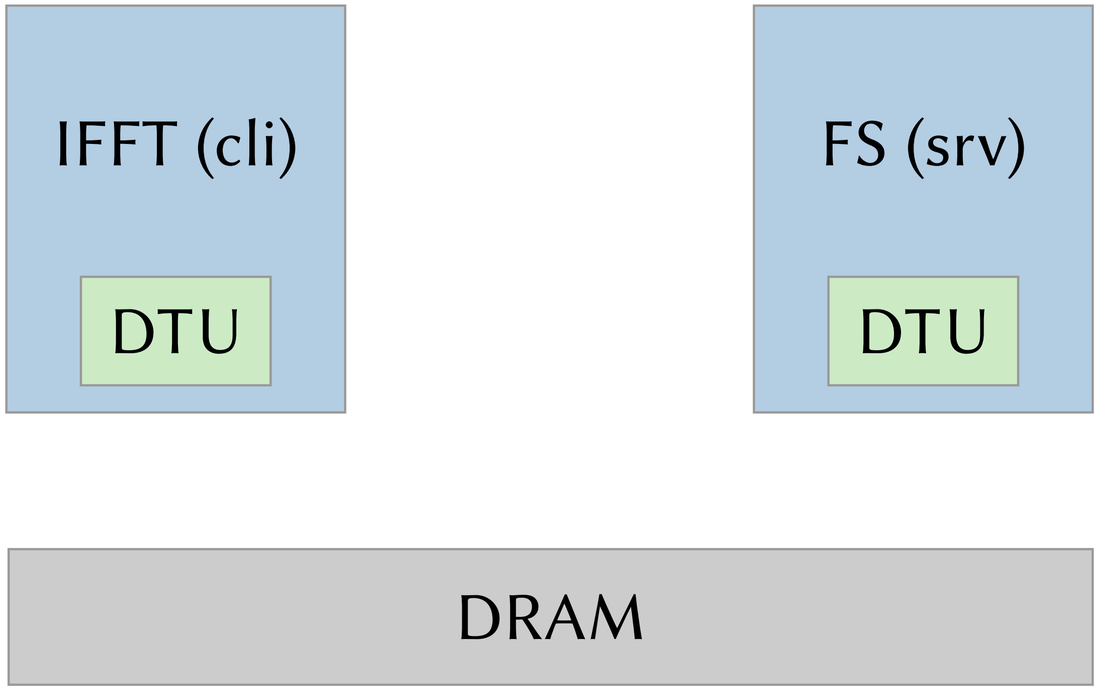
Accelerators and OS Services



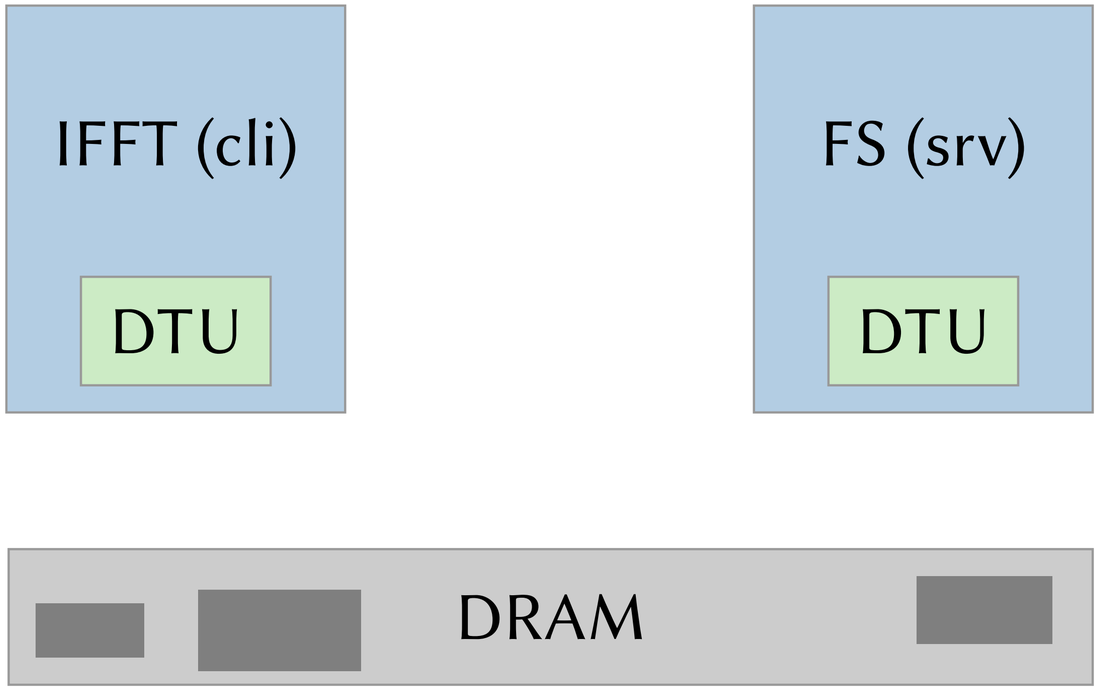
Challenges:

- OS must provide generic protocols
- Accelerators need support for protocols

Generic Protocol for Data Streams



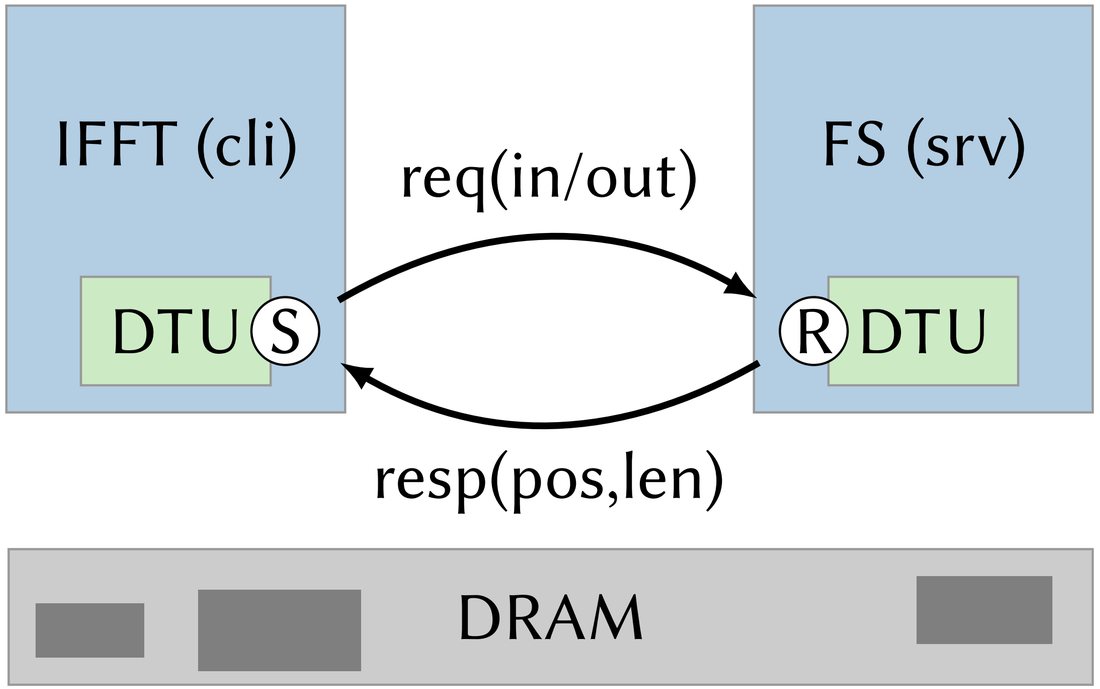
Generic Protocol for Data Streams



File protocol:

- Data in memory

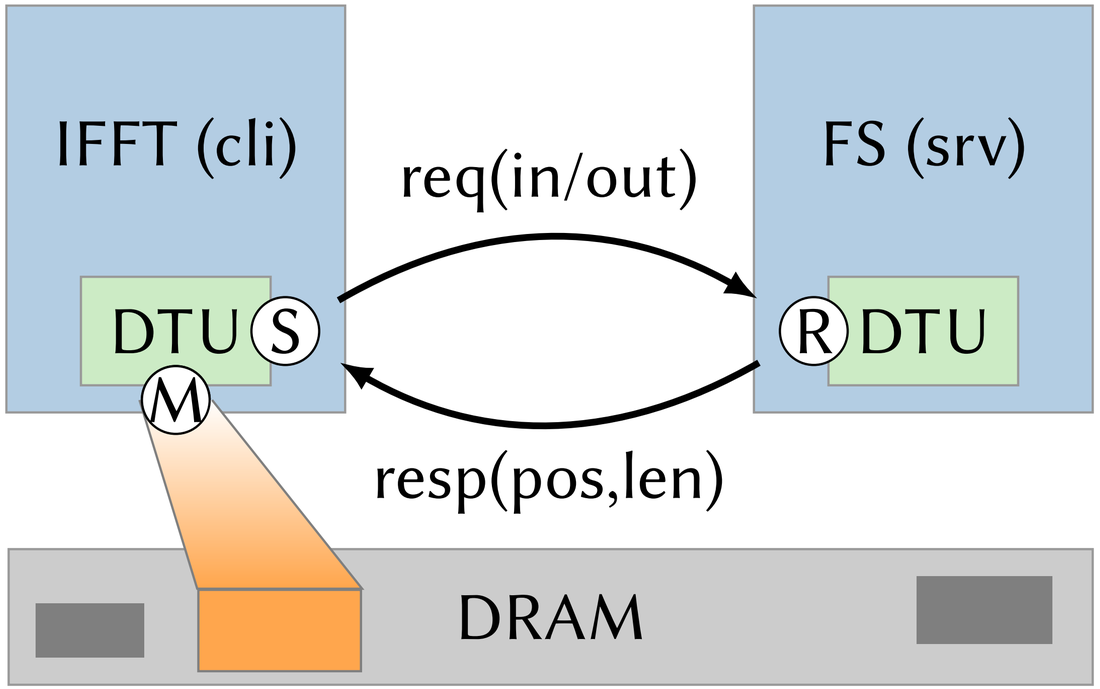
Generic Protocol for Data Streams



File protocol:

- Data in memory
- Msg channel between client and server
  - `req(in)` for next input piece
  - `req(out)` for next output piece

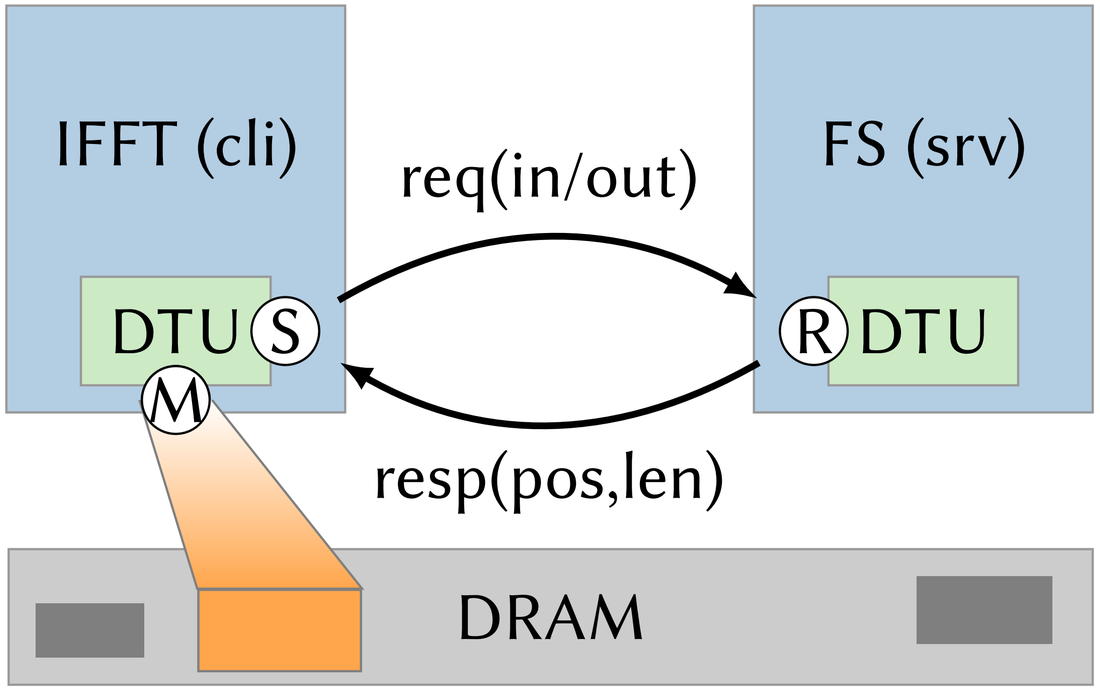
Generic Protocol for Data Streams



File protocol:

- Data in memory
- Msg channel between client and server
  - `req(in)` for next input piece
  - `req(out)` for next output piece
- Server configures client's memory EP

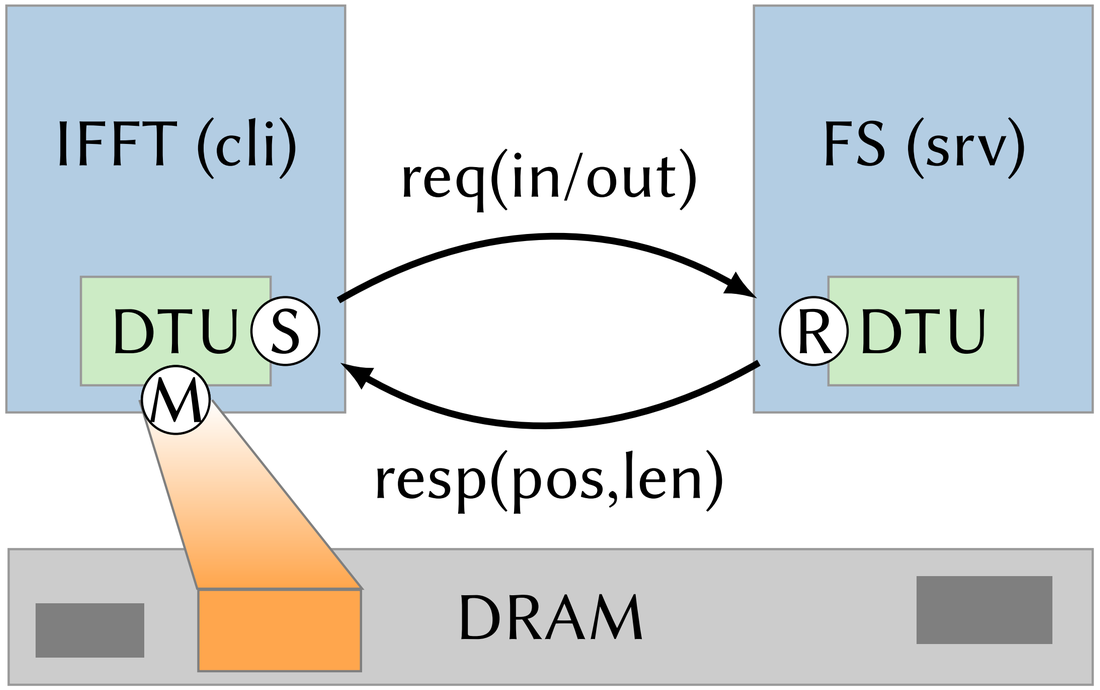
Generic Protocol for Data Streams



File protocol:

- Data in memory
- Msg channel between client and server
  - `req(in)` for next input piece
  - `req(out)` for next output piece
- Server configures client's memory EP
- Client accesses data via DTU

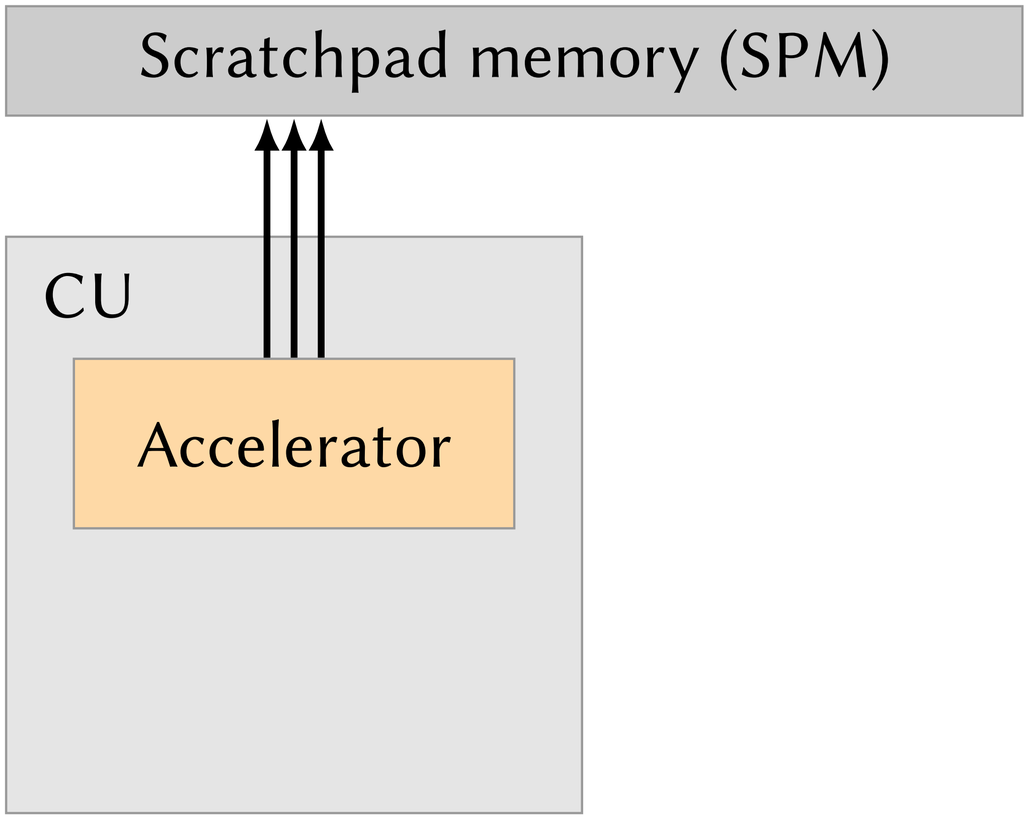
Generic Protocol for Data Streams



File protocol:

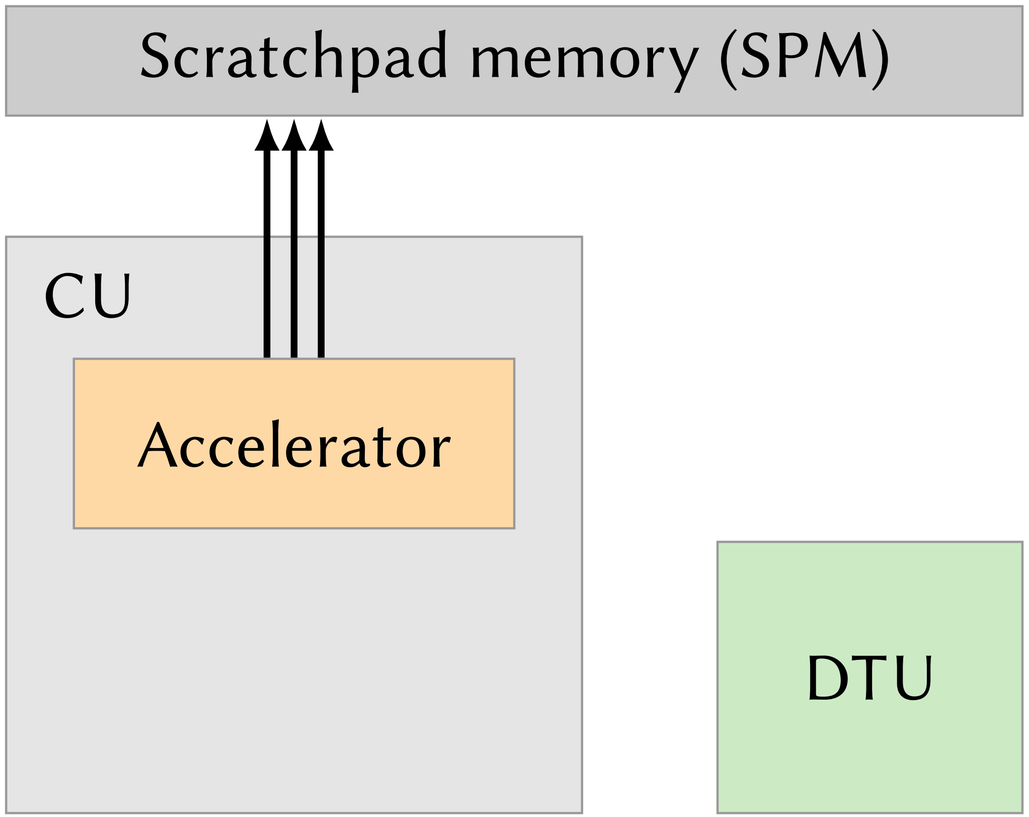
- Data in memory
- Msg channel between client and server
  - `req(in)` for next input piece
  - `req(out)` for next output piece
- Server configures client's memory EP
- Client accesses data via DTU
- Used by *all* CUs

Protocol Implementation for Accelerators



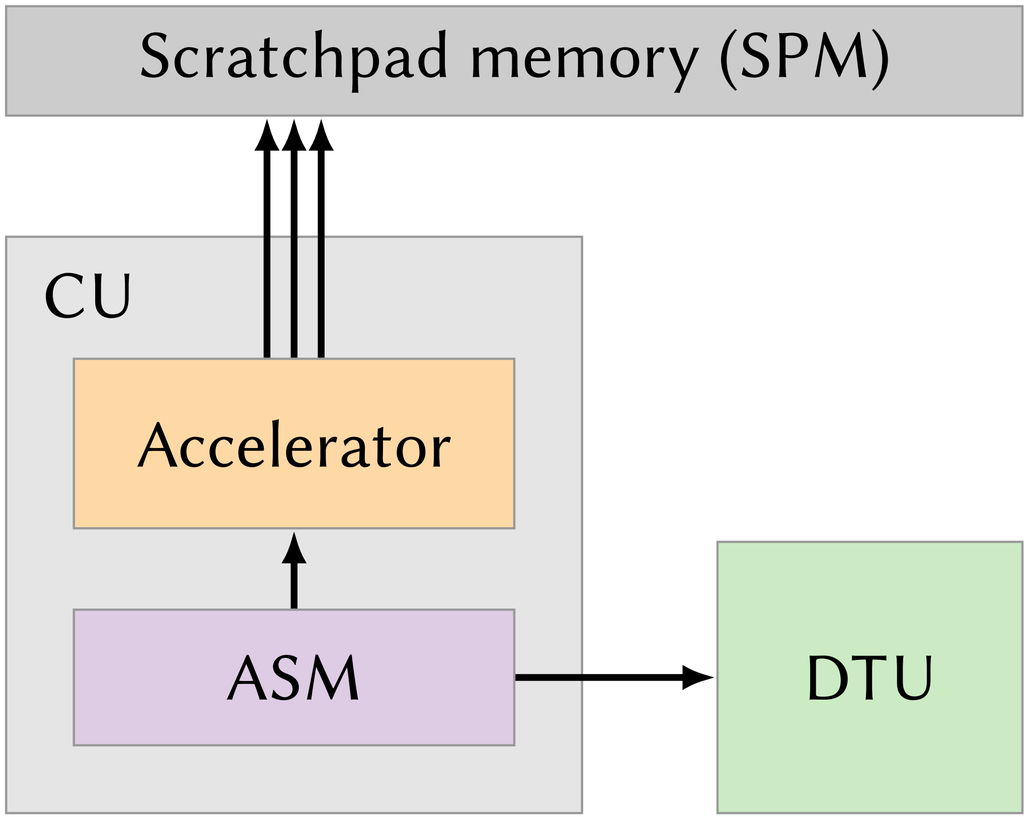
Off-the-shelf accelerators

Protocol Implementation for Accelerators



Off-the-shelf accelerators

Protocol Implementation for Accelerators

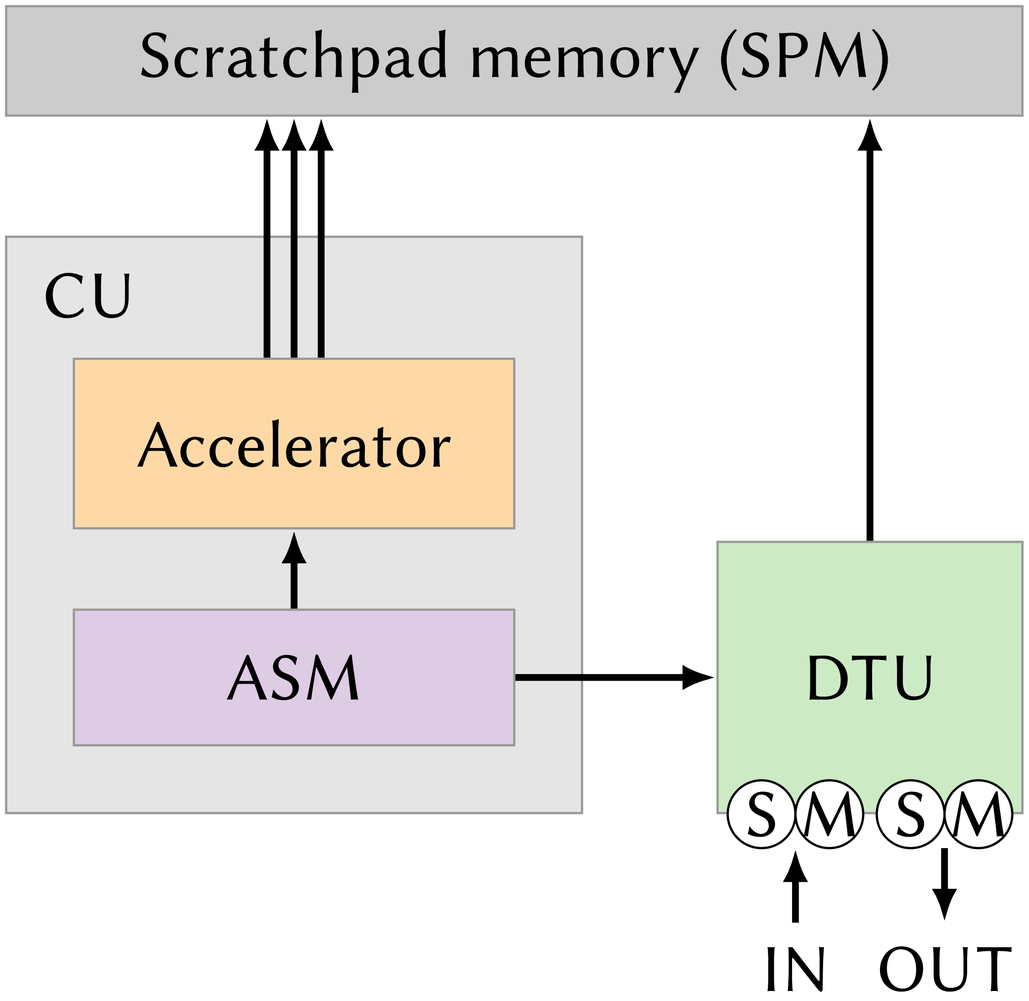


Off-the-shelf accelerators

Accelerator Support Module (ASM):

- Interacts with DTU and accelerator

Protocol Implementation for Accelerators

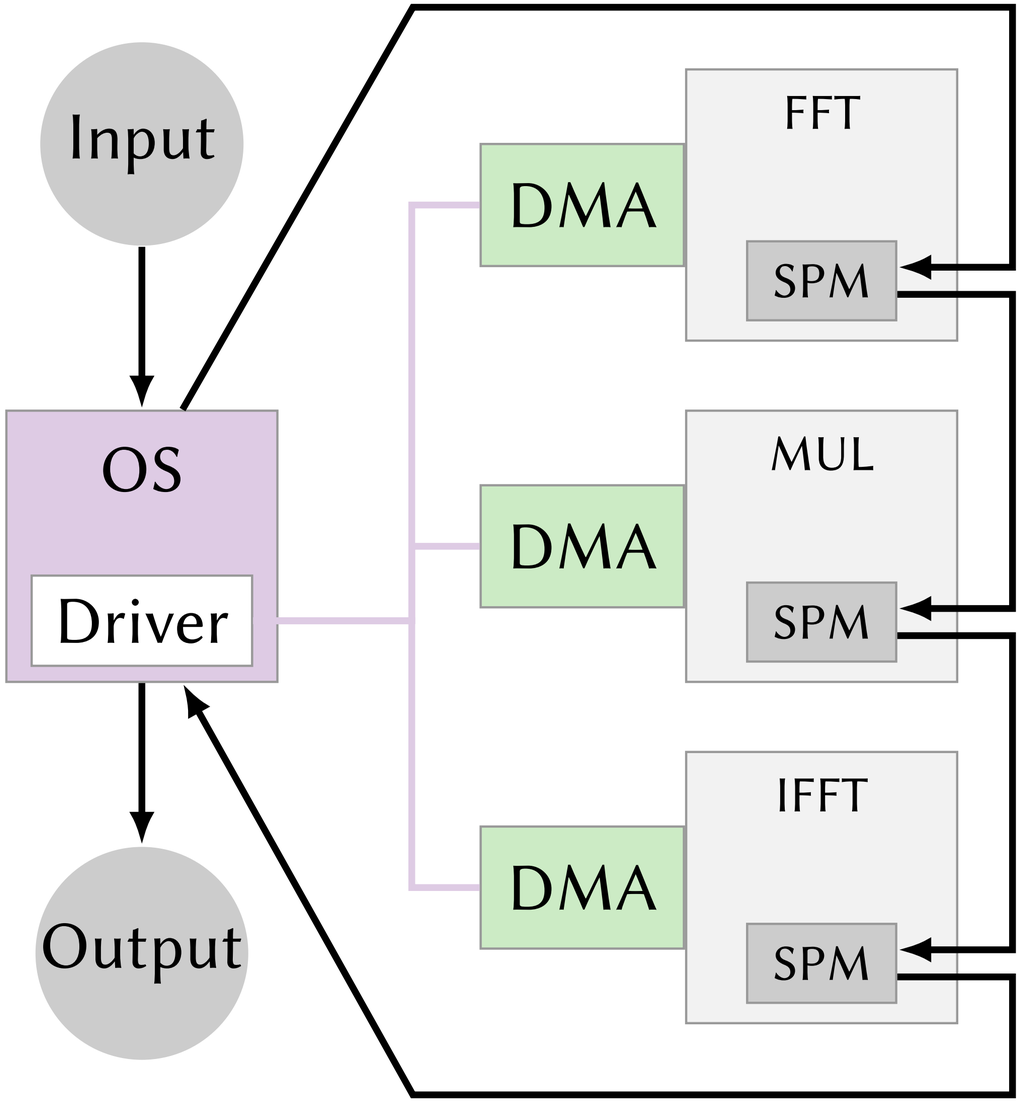


Off-the-shelf accelerators

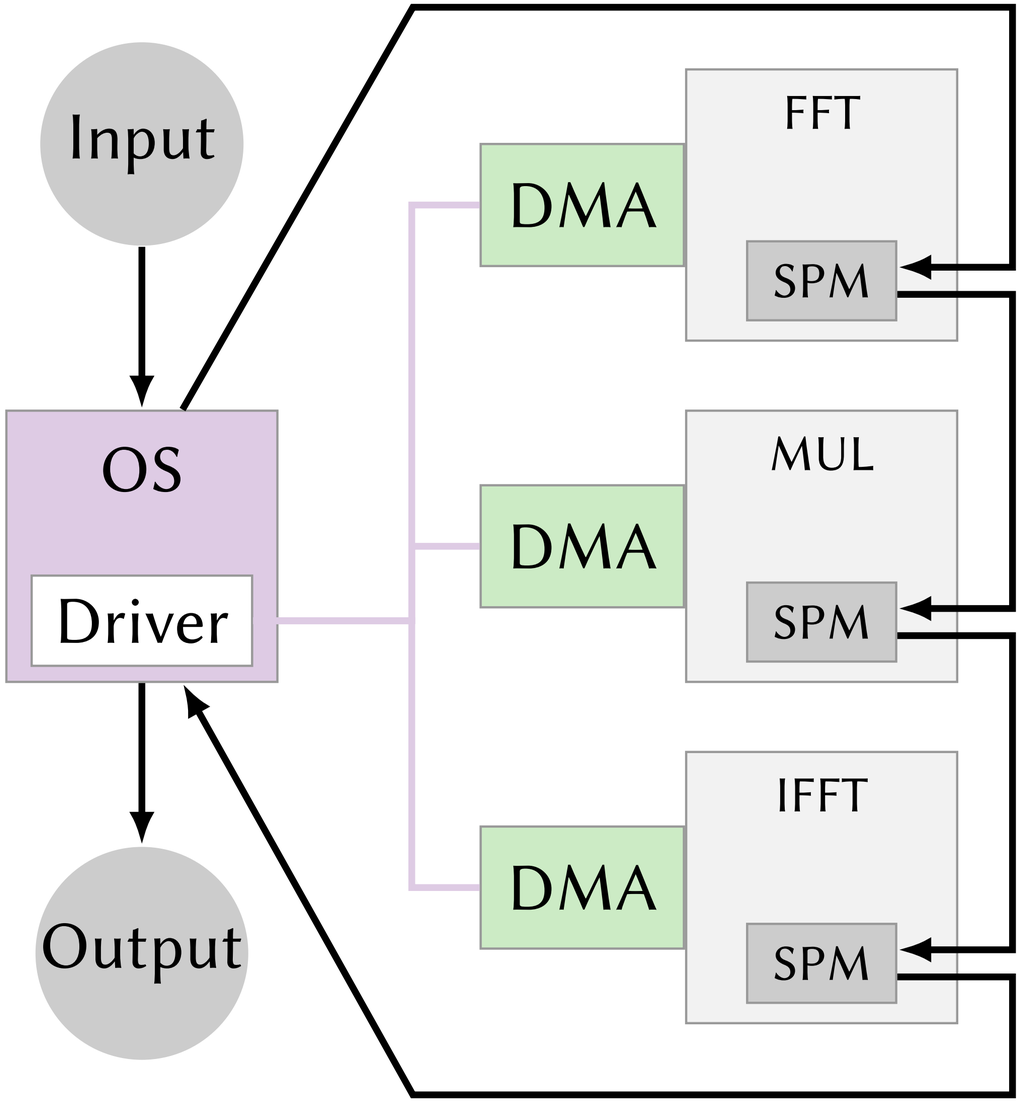
Accelerator Support Module (ASM):

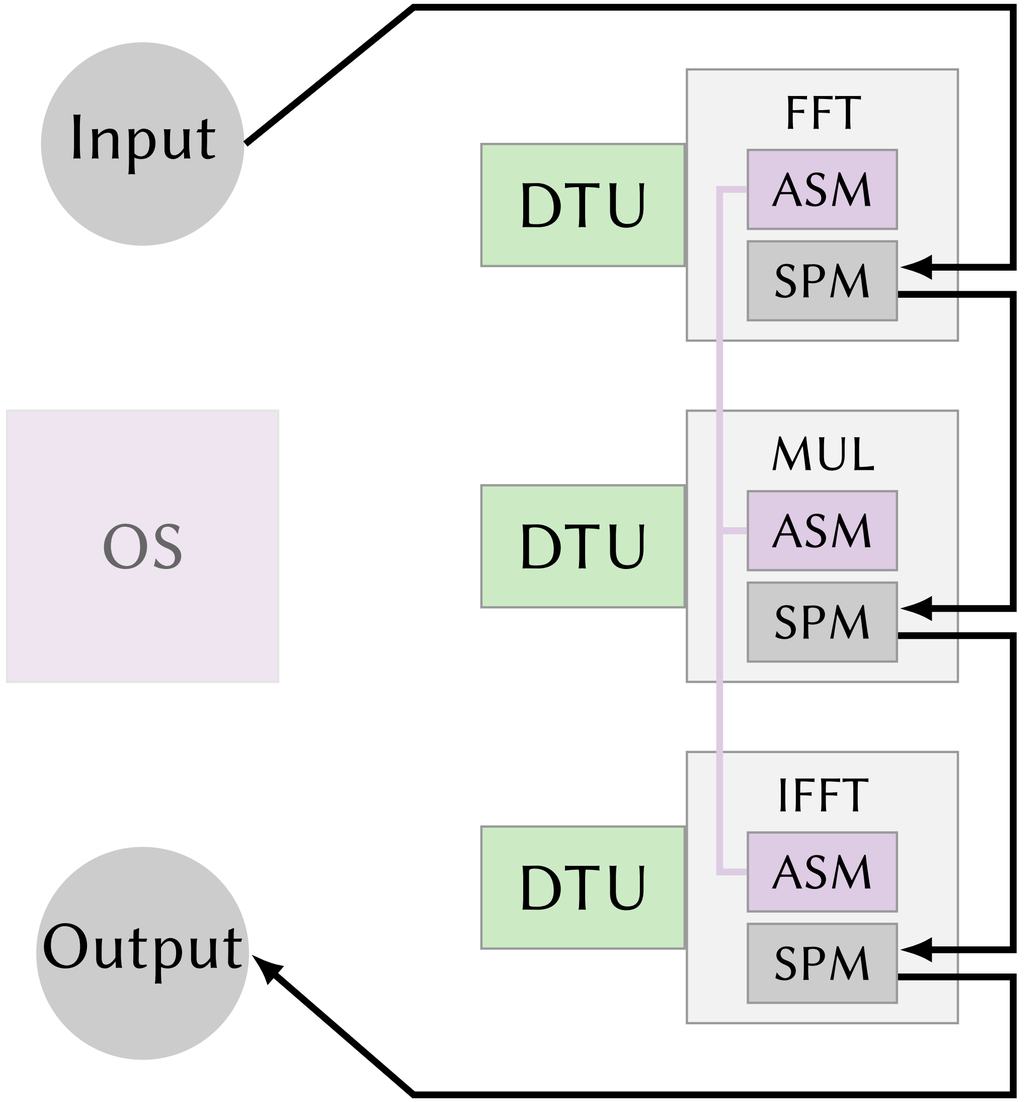
- Interacts with DTU and accelerator
- Implements file protocol for input and output channel

Assisted vs. Autonomous

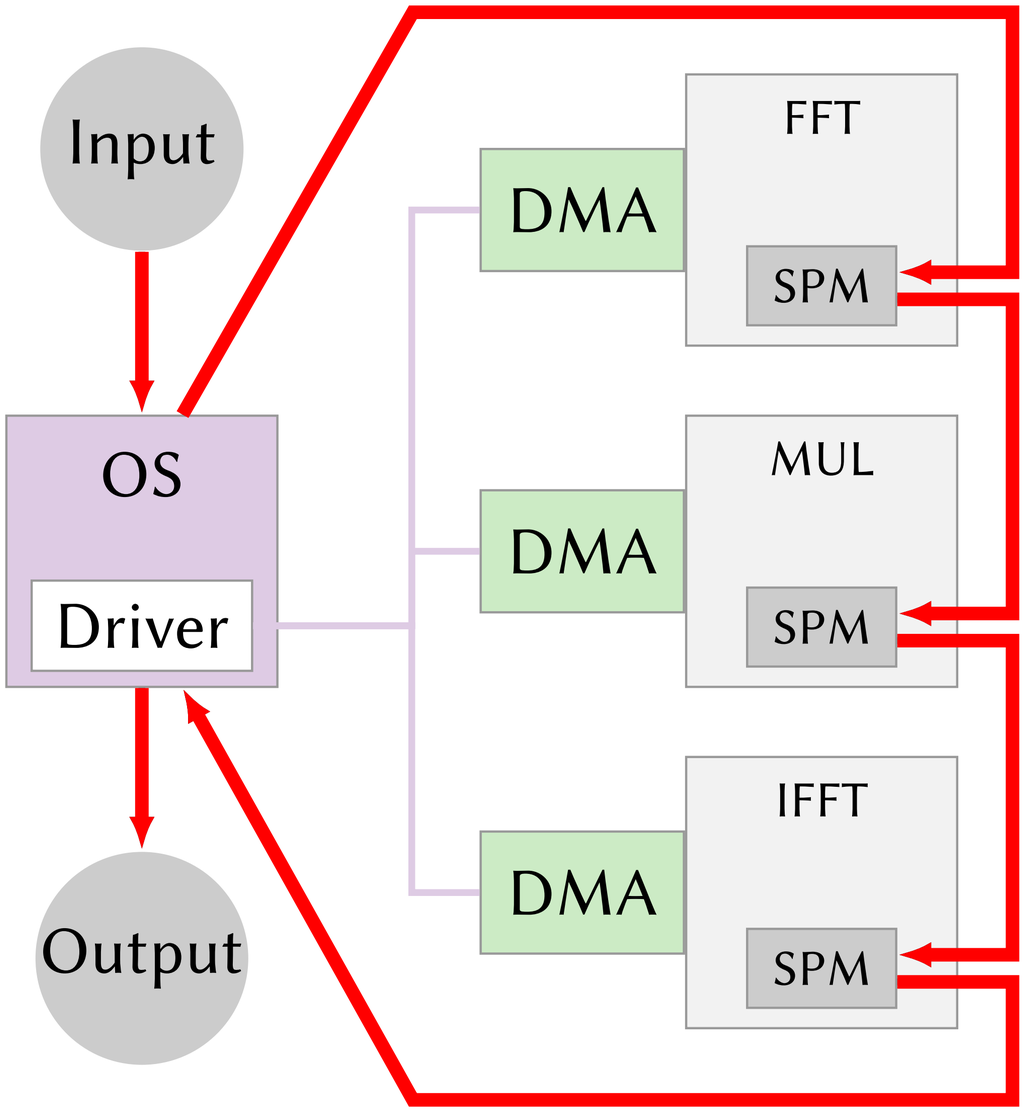


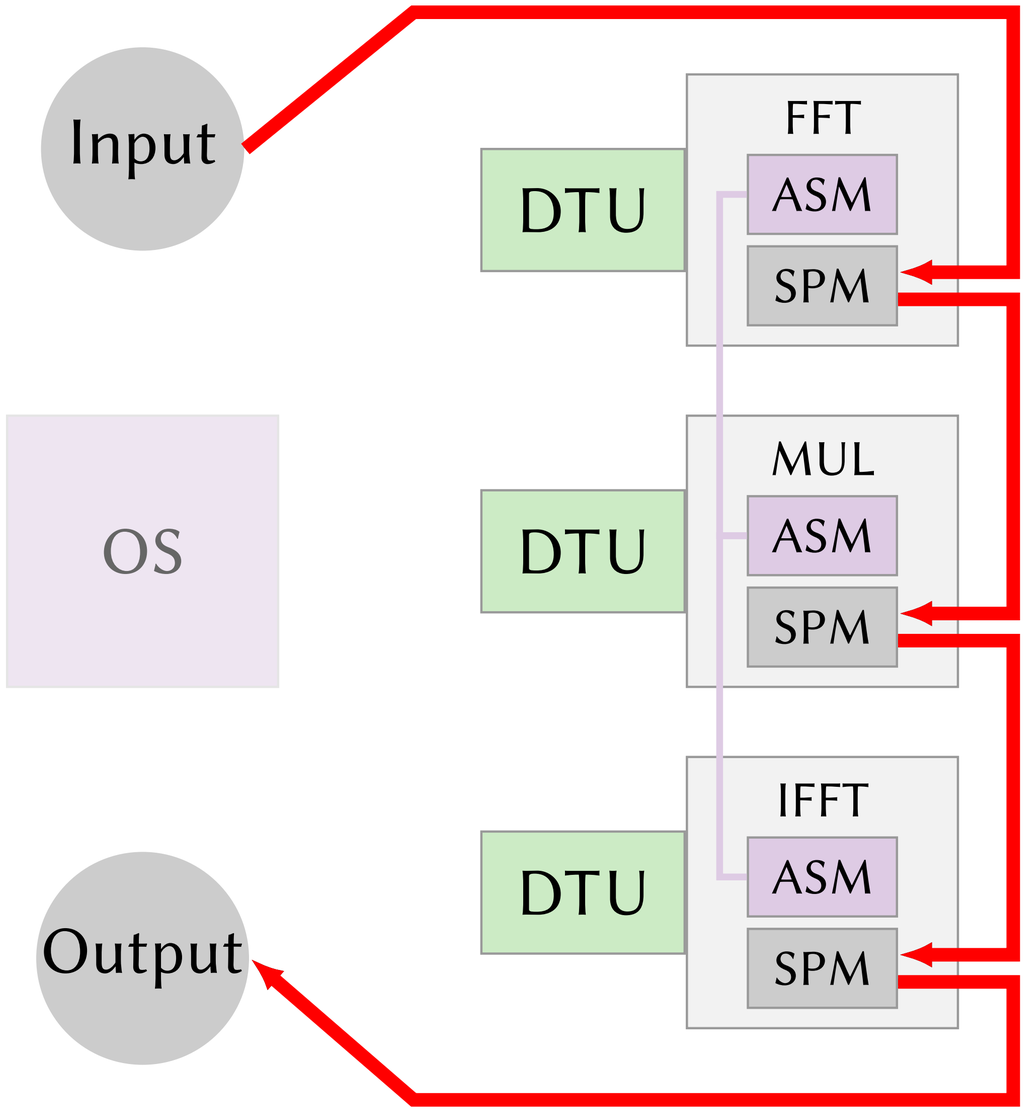
Assisted vs. Autonomous



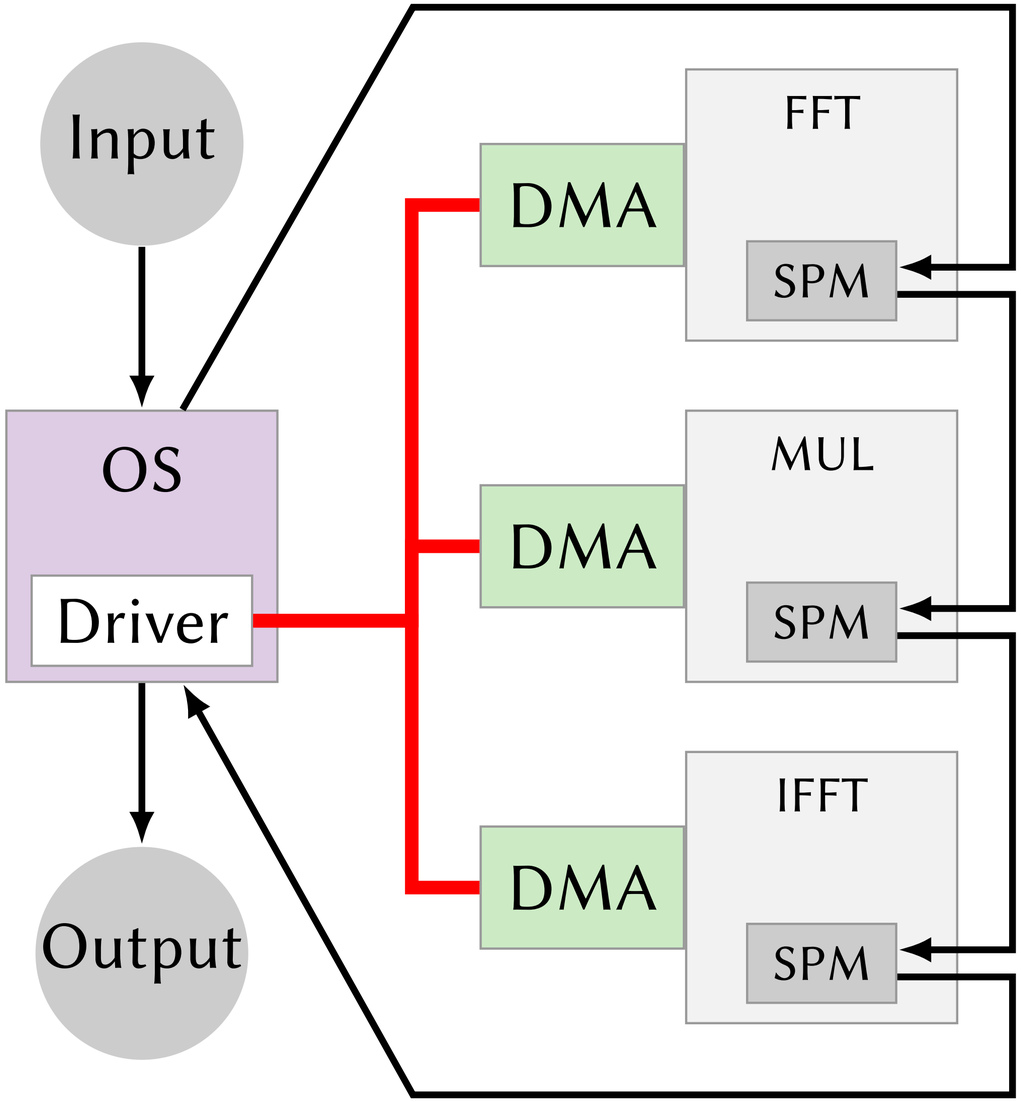


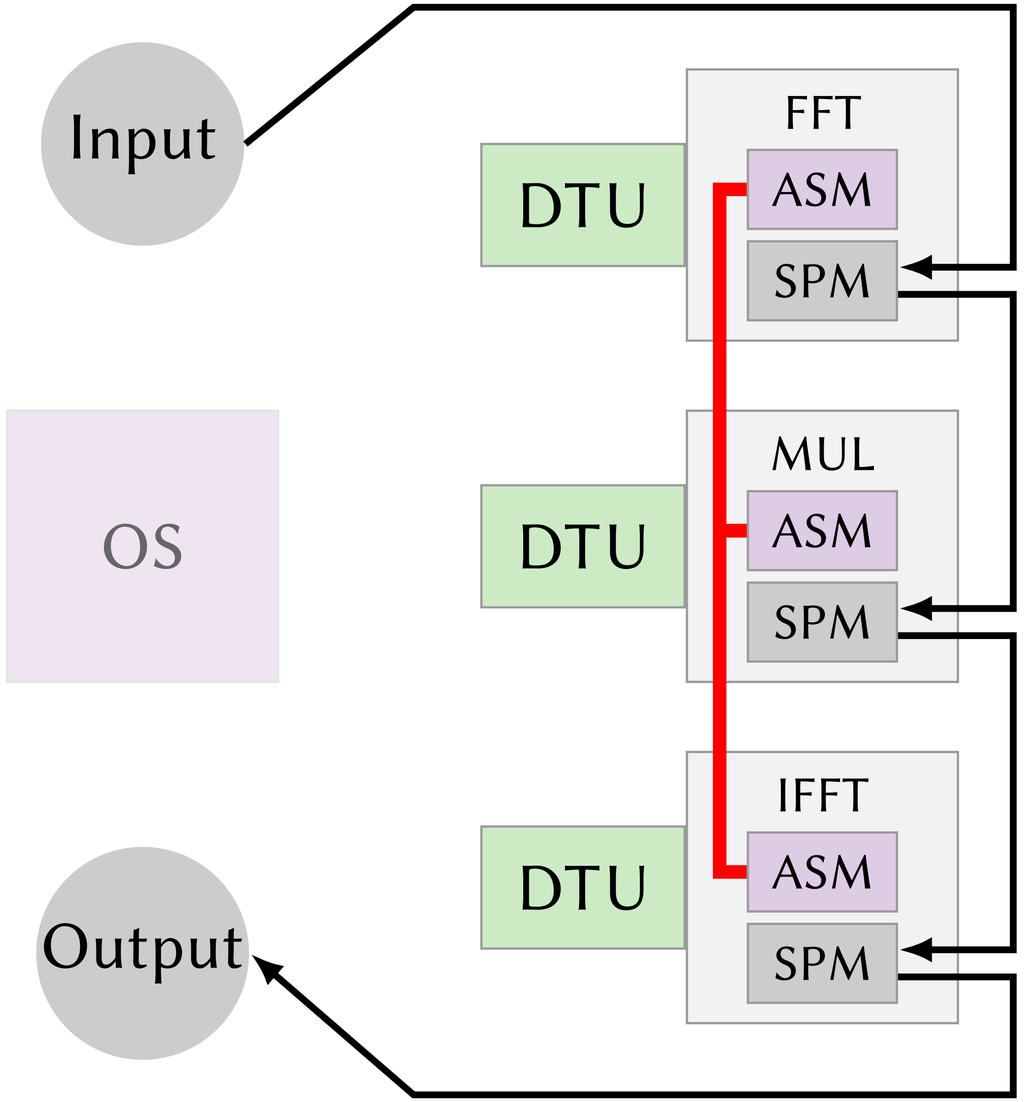
Assisted vs. Autonomous





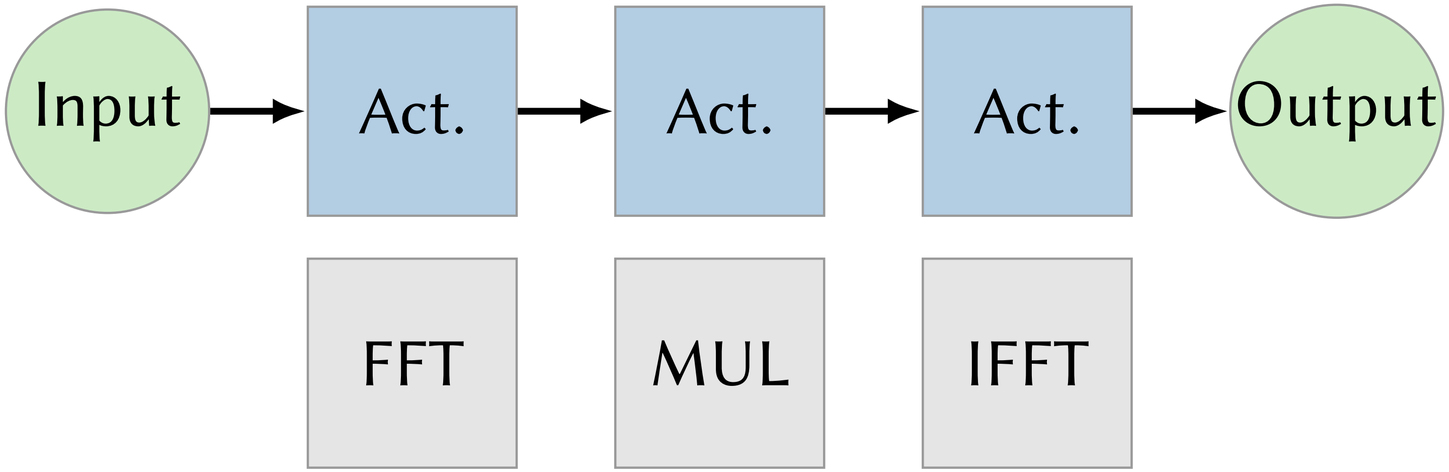
Assisted vs. Autonomous



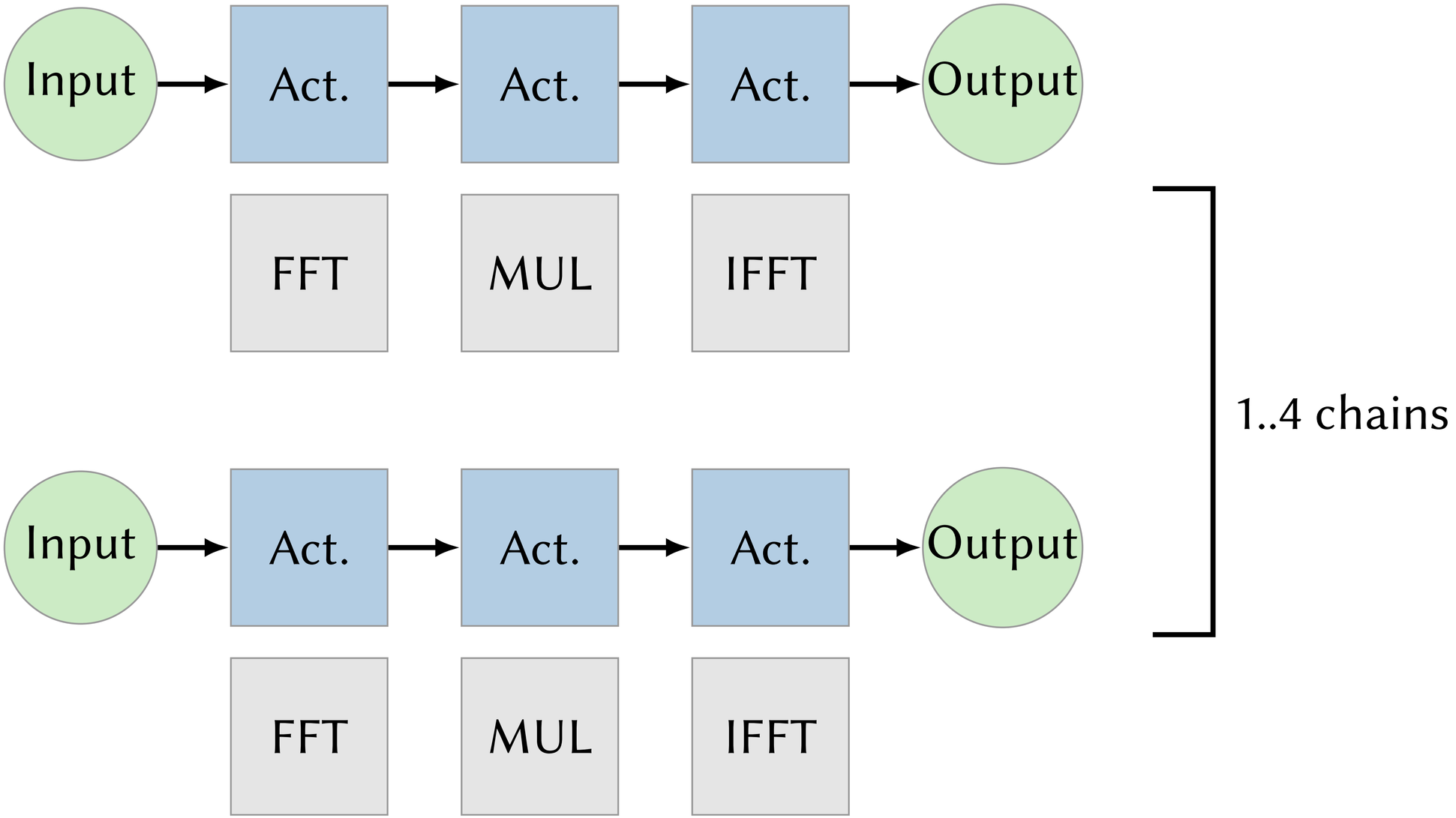


Evaluation

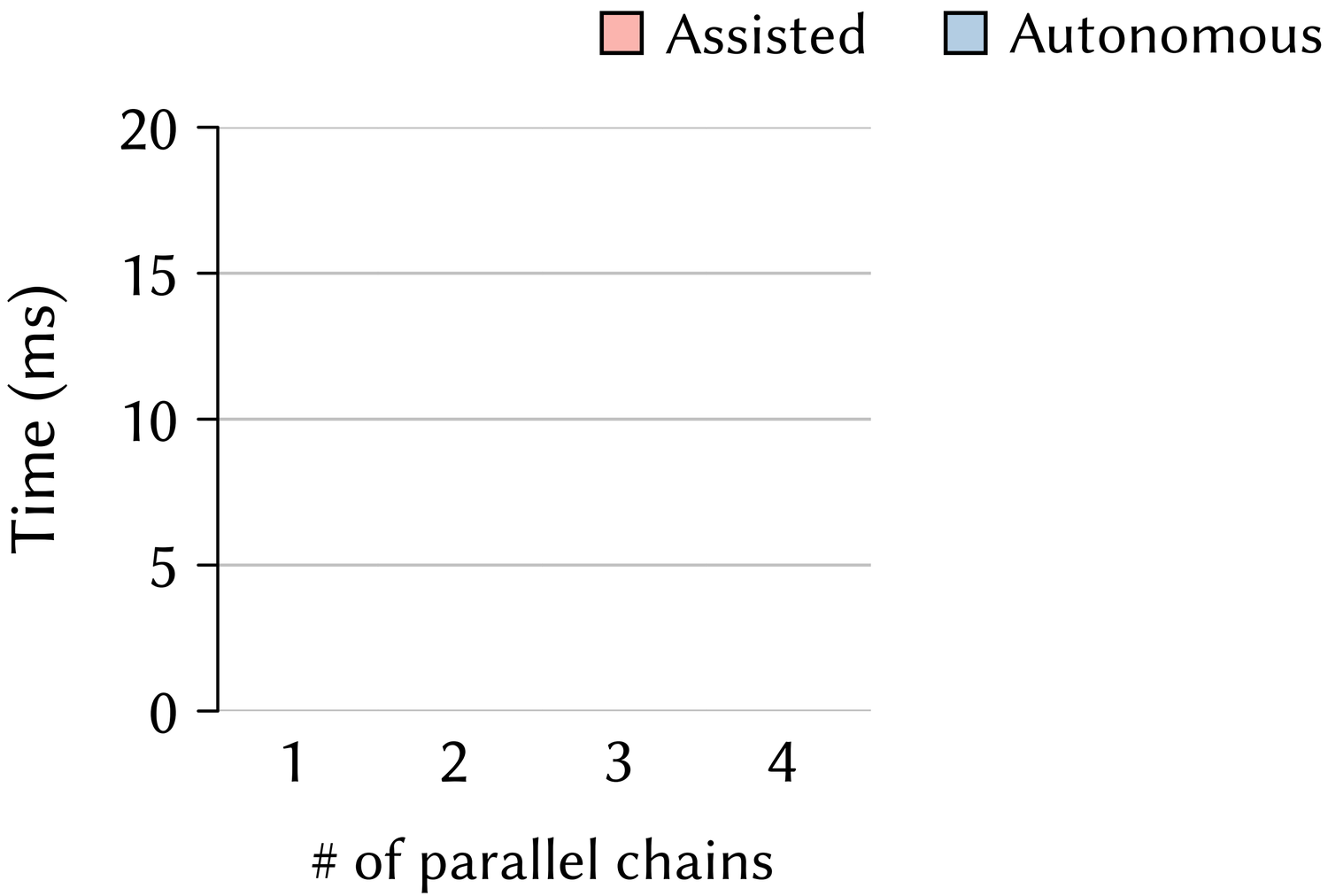
Evaluation



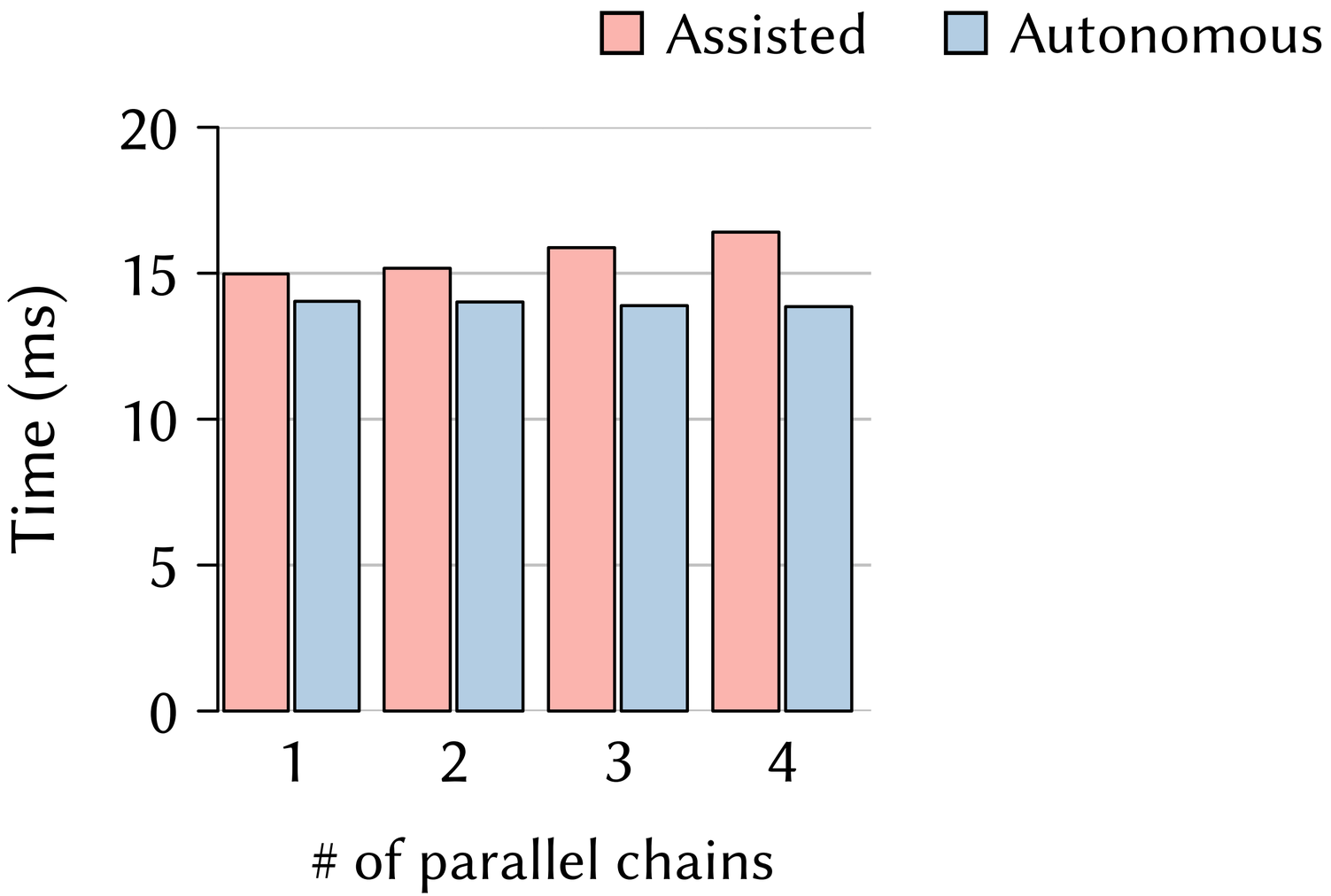
Evaluation


Input
Act.
Act.
Act.
Output
FFT
MUL
IFFT
Input
Act.
Act.
Act.
Output
FFT
MUL
IFFT
1..4 chains

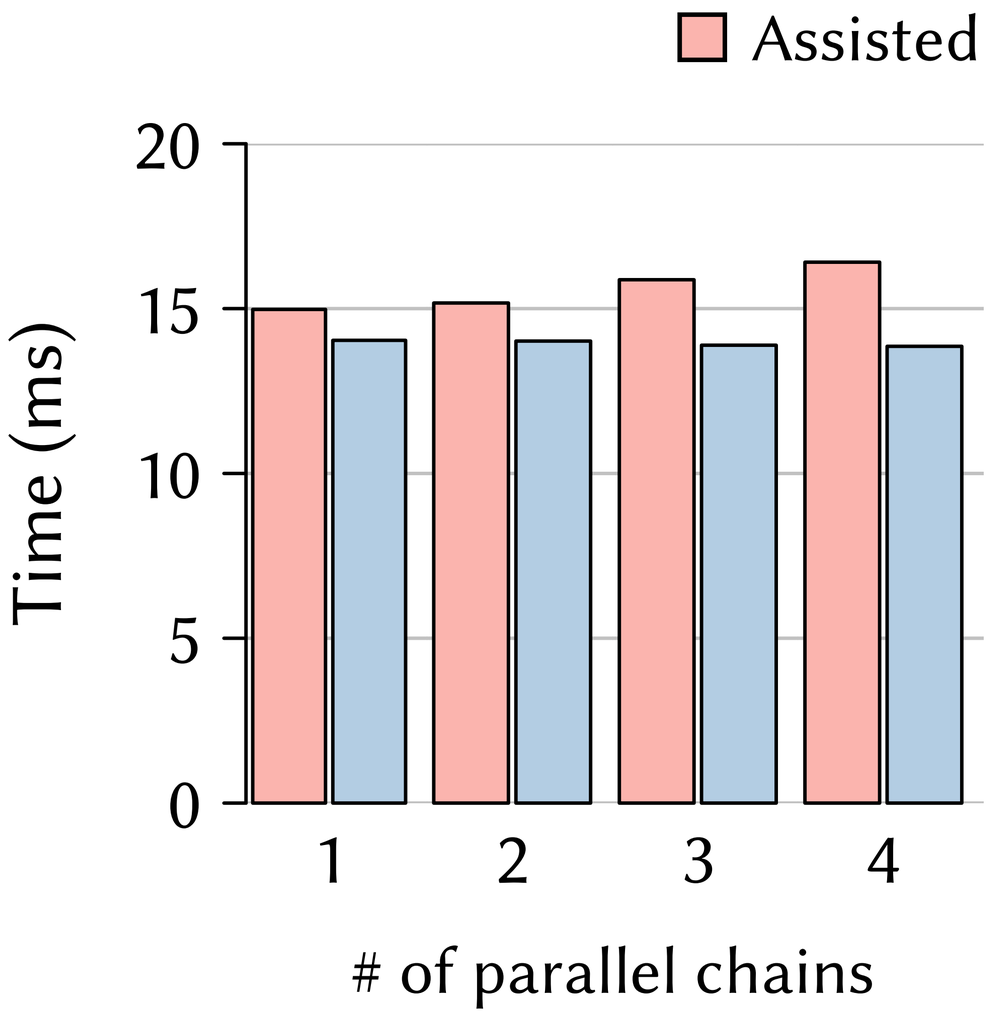
Evaluation

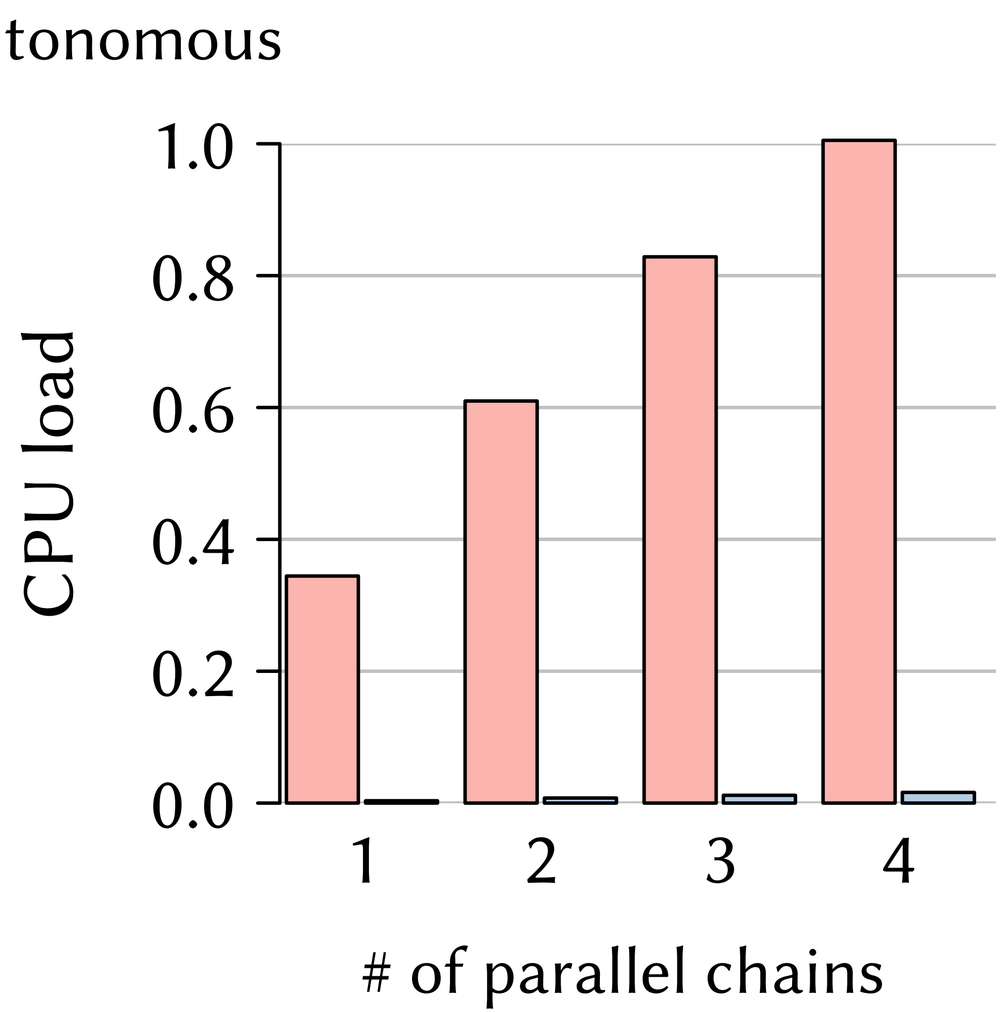

Assisted
Autonomous
20
15
10
5
0
Time (ms)
1
2
3
4
# of parallel chains

Evaluation

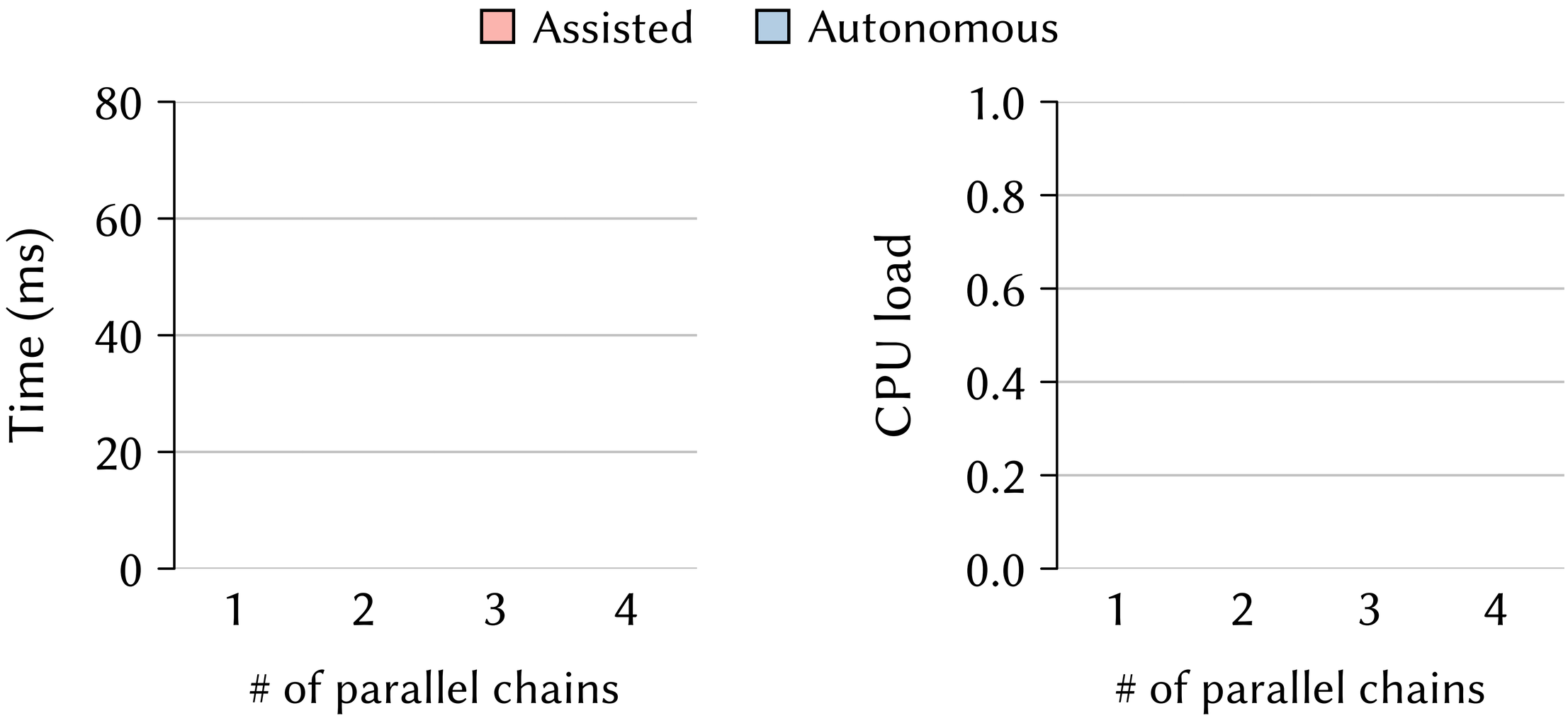

Assisted
Autonomous
20
15
10
5
0
Time (ms)
1
2
3
4
# of parallel chains

Evaluation

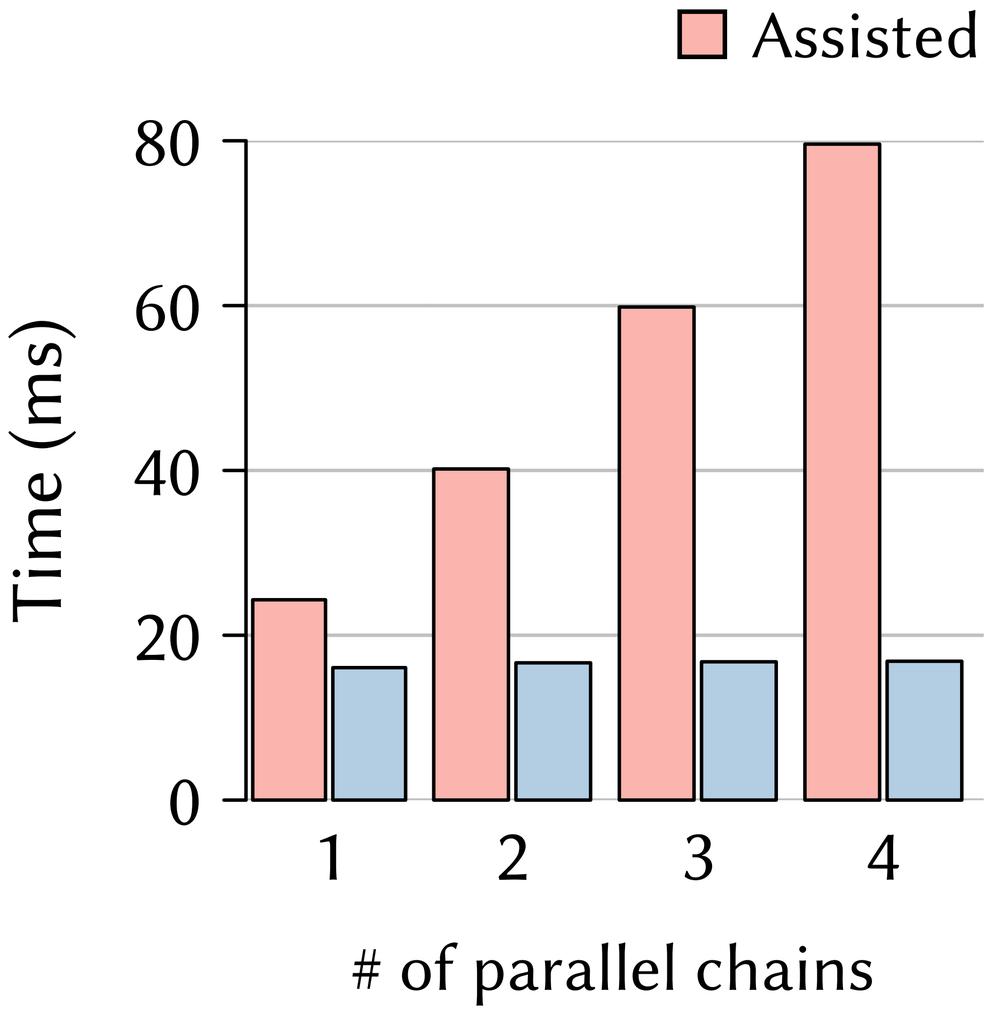

Assisted
Autonomous
Time (ms)
20
15
10
5
0
1
2
3
4
# of parallel chains

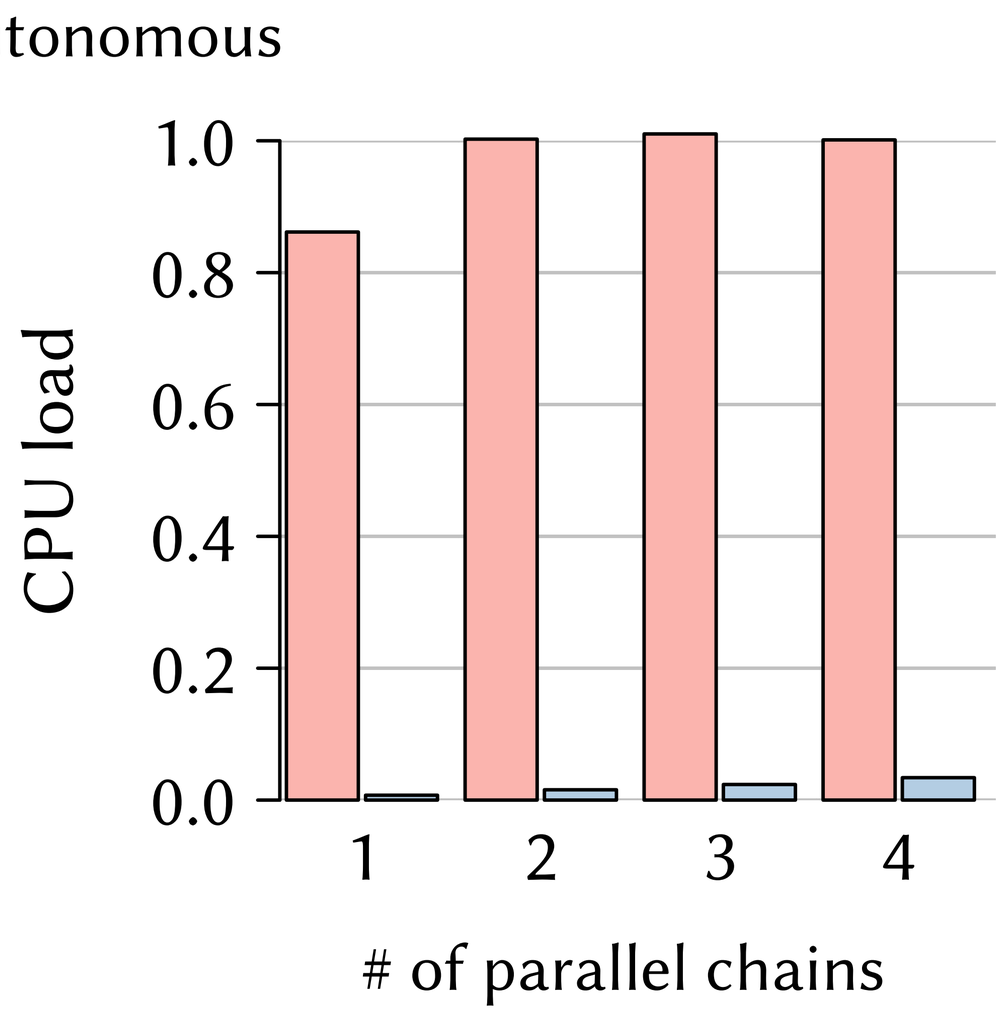

CPU load
1.0
0.8
0.6
0.4
0.2
0.0
1
2
3
4
# of parallel chains

Evaluation: PCIe Latency

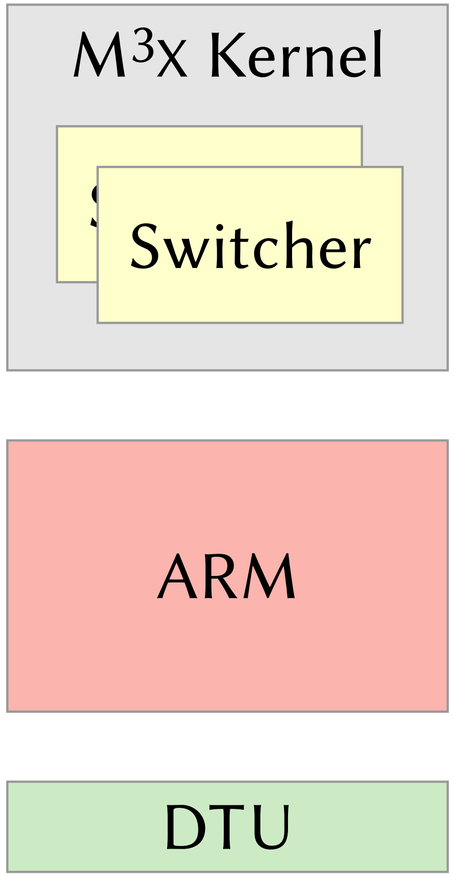


Evaluation: PCIe Latency



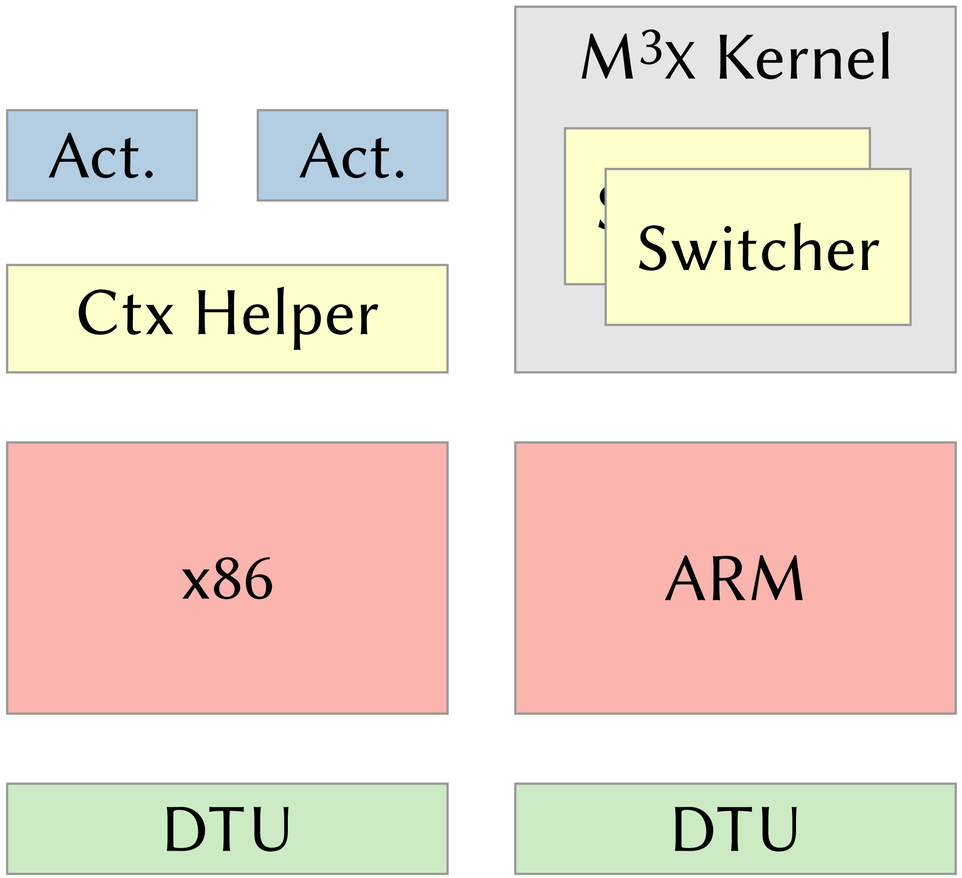


Addendum: Context Switching



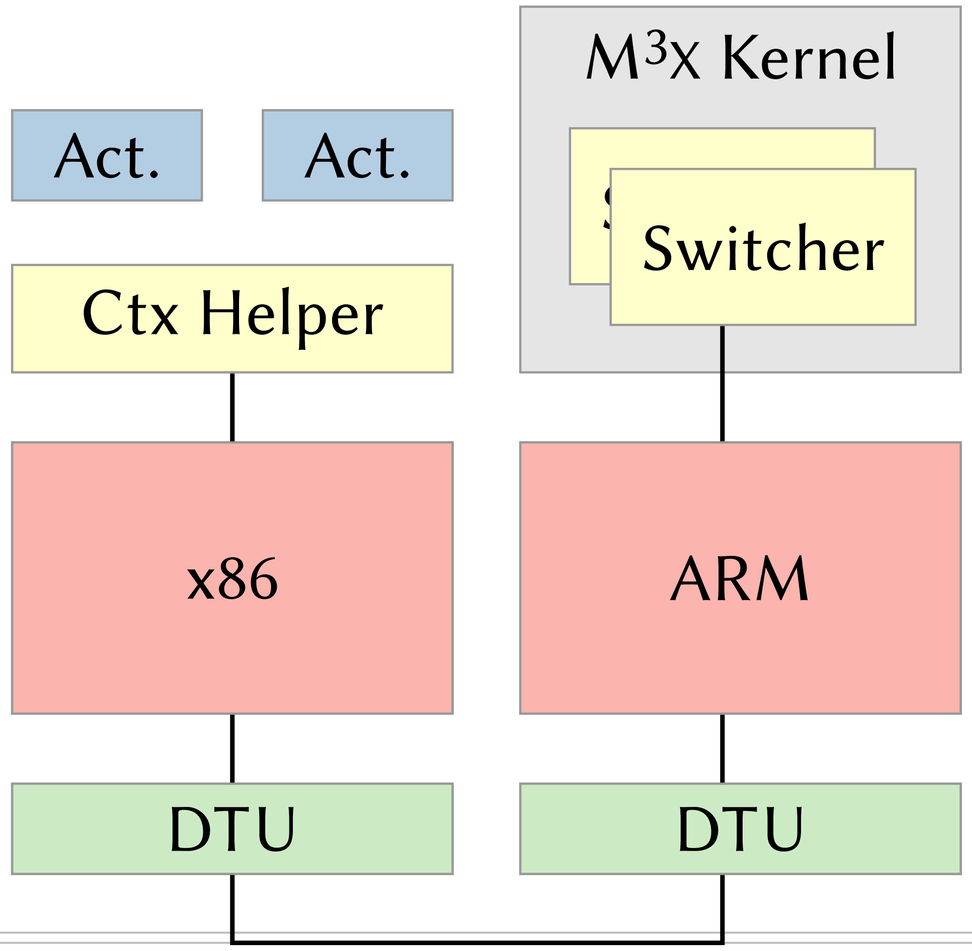
- Kernel handles complex part
  - Schedules/migrates activities
  - Initiates context switches

Addendum: Context Switching



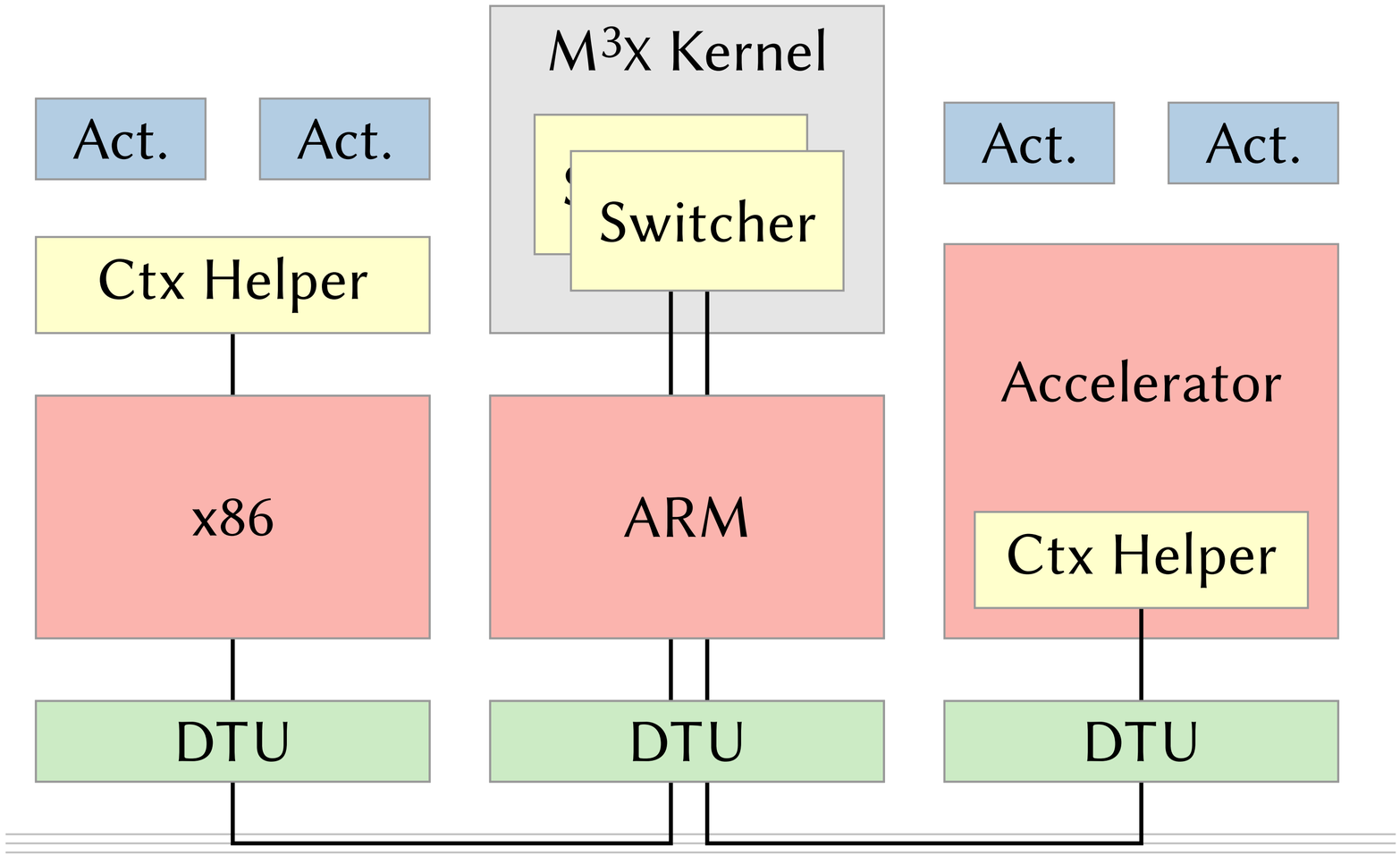
- Kernel handles complex part
  - Schedules/migrates activities
  - Initiates context switches
- Helper on user tiles implements save/restore
  - General purpose tiles: Software helper

Addendum: Context Switching



- Kernel handles complex part
  - Schedules/migrates activities
  - Initiates context switches
- Helper on user tiles implements save/restore
  - General purpose tiles: Software helper

Addendum: Context Switching



- Kernel handles complex part
  - Schedules/migrates activities
  - Initiates context switches
- Helper on user tiles implements save/restore
  - General purpose tiles: Software helper
  - Accelerator tiles: Helper implemented in hardware as part of ASM

Summary

Summary

Autonomous accelerator chains

Summary

Autonomous accelerator chains

- communication without involving the kernel or an orchestration core

Summary

Autonomous accelerator chains

- communication without involving the kernel or an orchestration core
- all tiles implement a uniform data stream protocol

Summary

Autonomous accelerator chains

- communication without involving the kernel or an orchestration core
- all tiles implement a uniform data stream protocol
- lightweight hardware implementation for fixed-function accelerators

Summary

Autonomous accelerator chains

- communication without involving the kernel or an orchestration core
- all tiles implement a uniform data stream protocol
- lightweight hardware implementation for fixed-function accelerators
- autonomous accelerator chains **reduce CPU load by a factor of 30**

Summary

Autonomous accelerator chains

- communication without involving the kernel or an orchestration core
- all tiles implement a uniform data stream protocol
- lightweight hardware implementation for fixed-function accelerators
- autonomous accelerator chains **reduce CPU load by a factor of 30**

Limitations

Summary

Autonomous accelerator chains

- communication without involving the kernel or an orchestration core
- all tiles implement a uniform data stream protocol
- lightweight hardware implementation for fixed-function accelerators
- autonomous accelerator chains **reduce CPU load by a factor of 30**

Limitations

- remote context switching

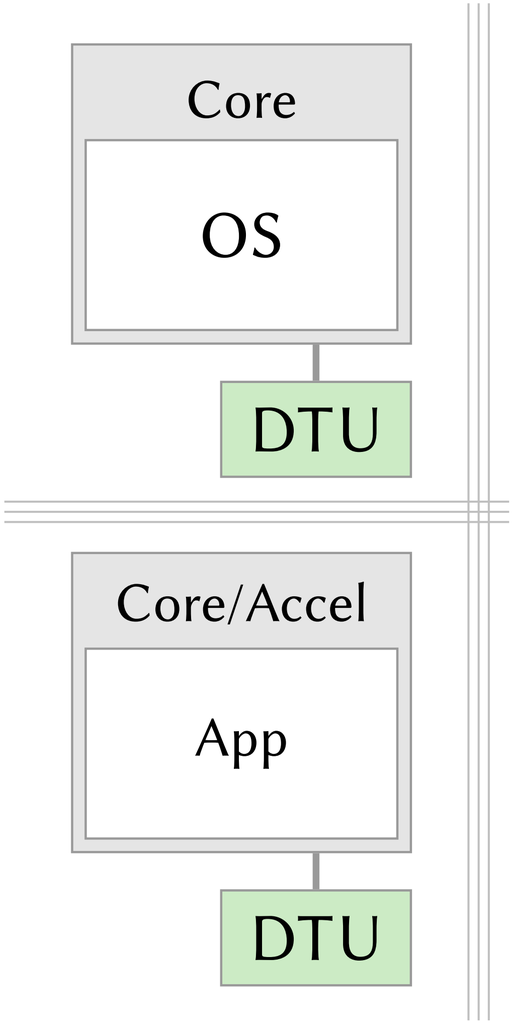
barkhausen institut

M³v — Virtualizing the DTU for Fast Context Switching

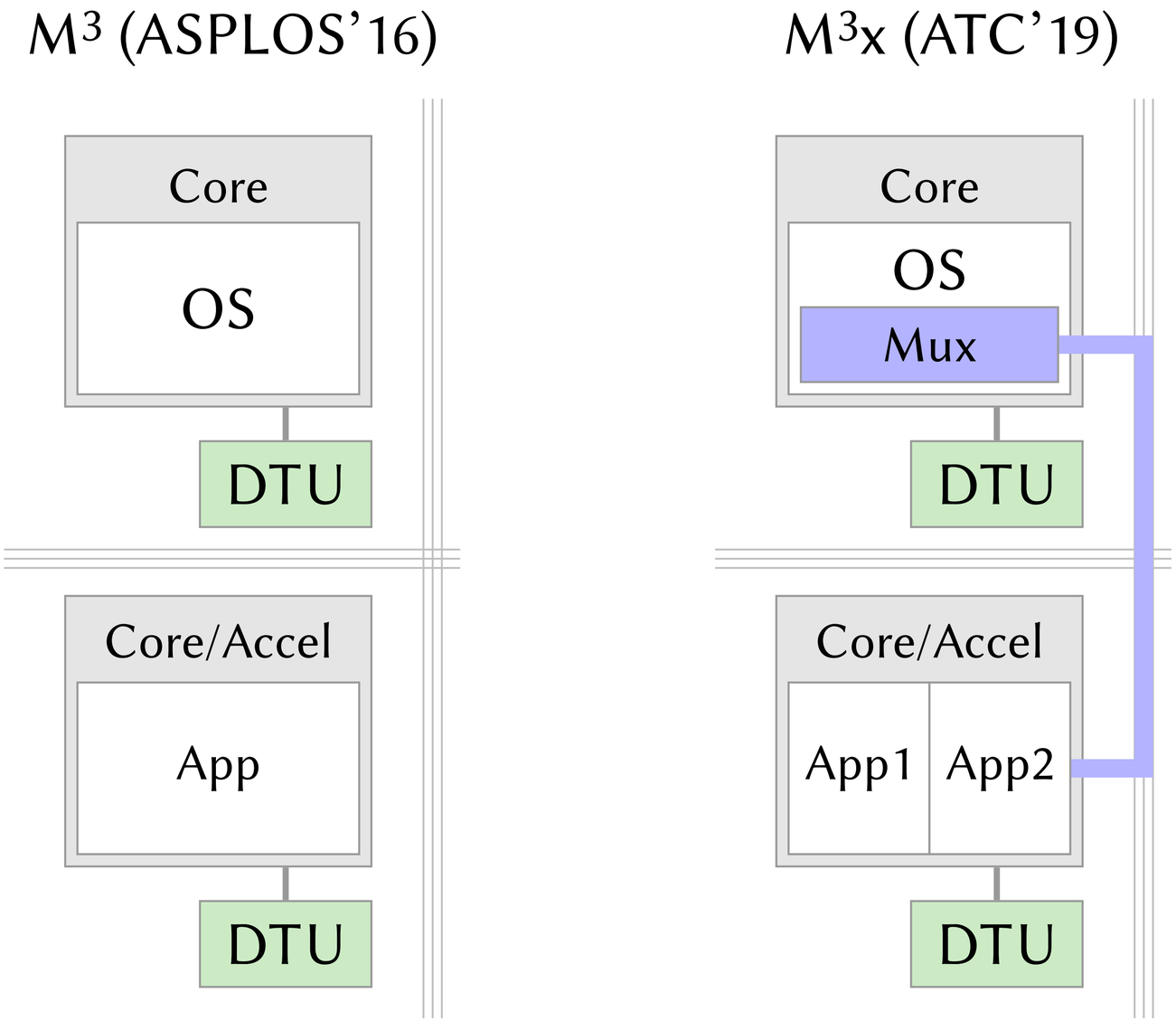
ASPLOS '22

Multiplexing Approaches

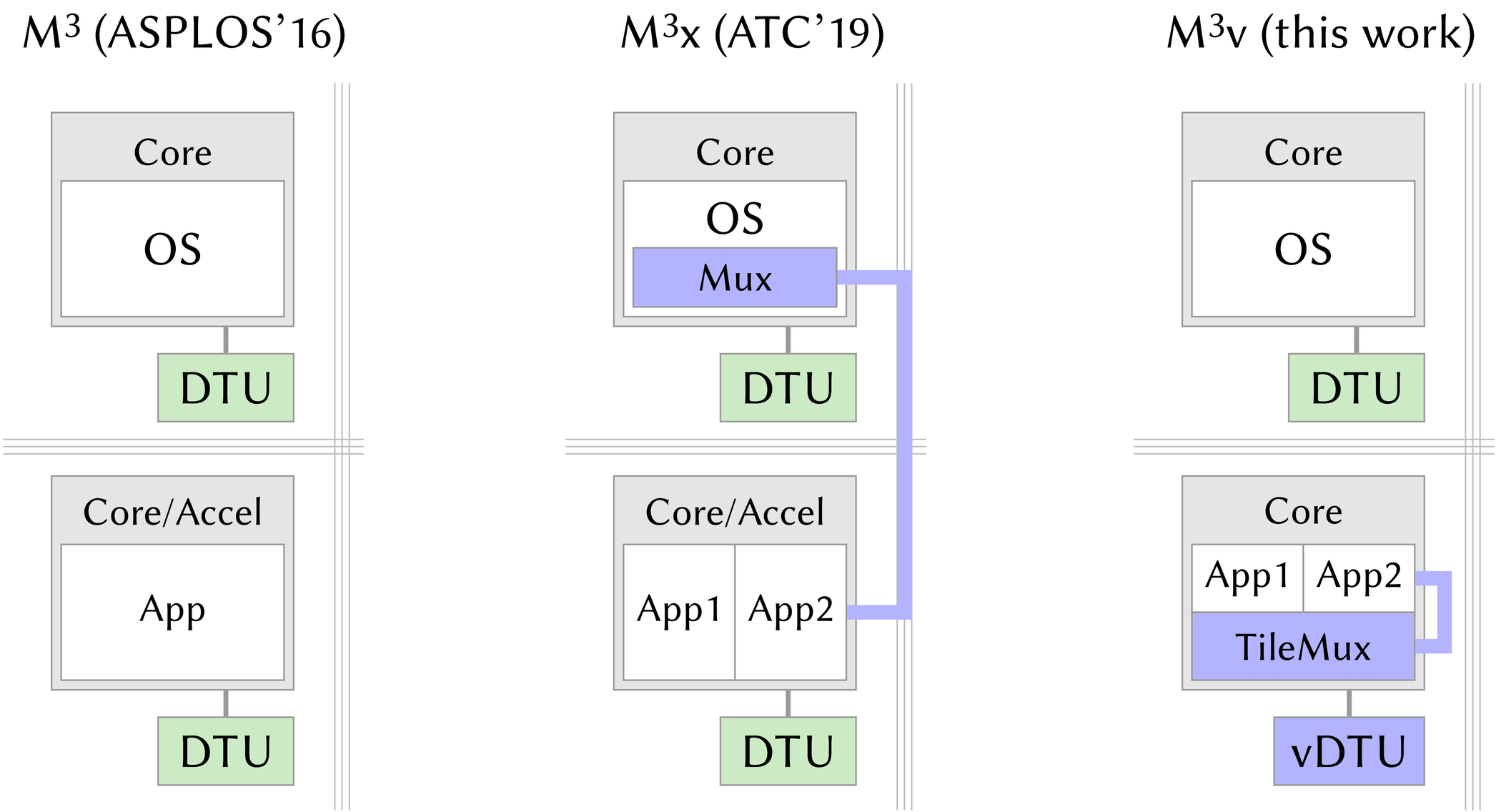
M³ (ASPLOS'16)



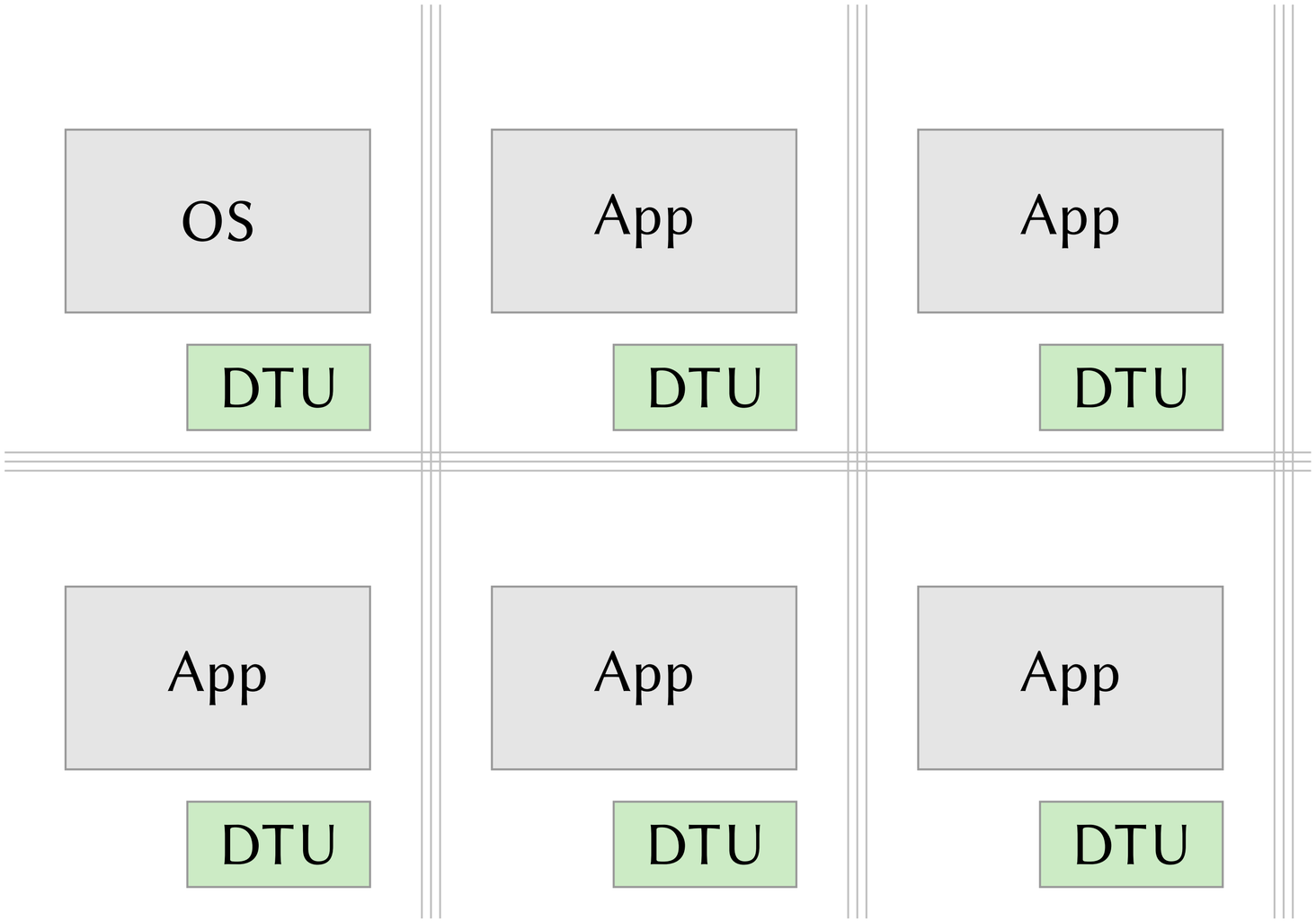
Multiplexing Approaches



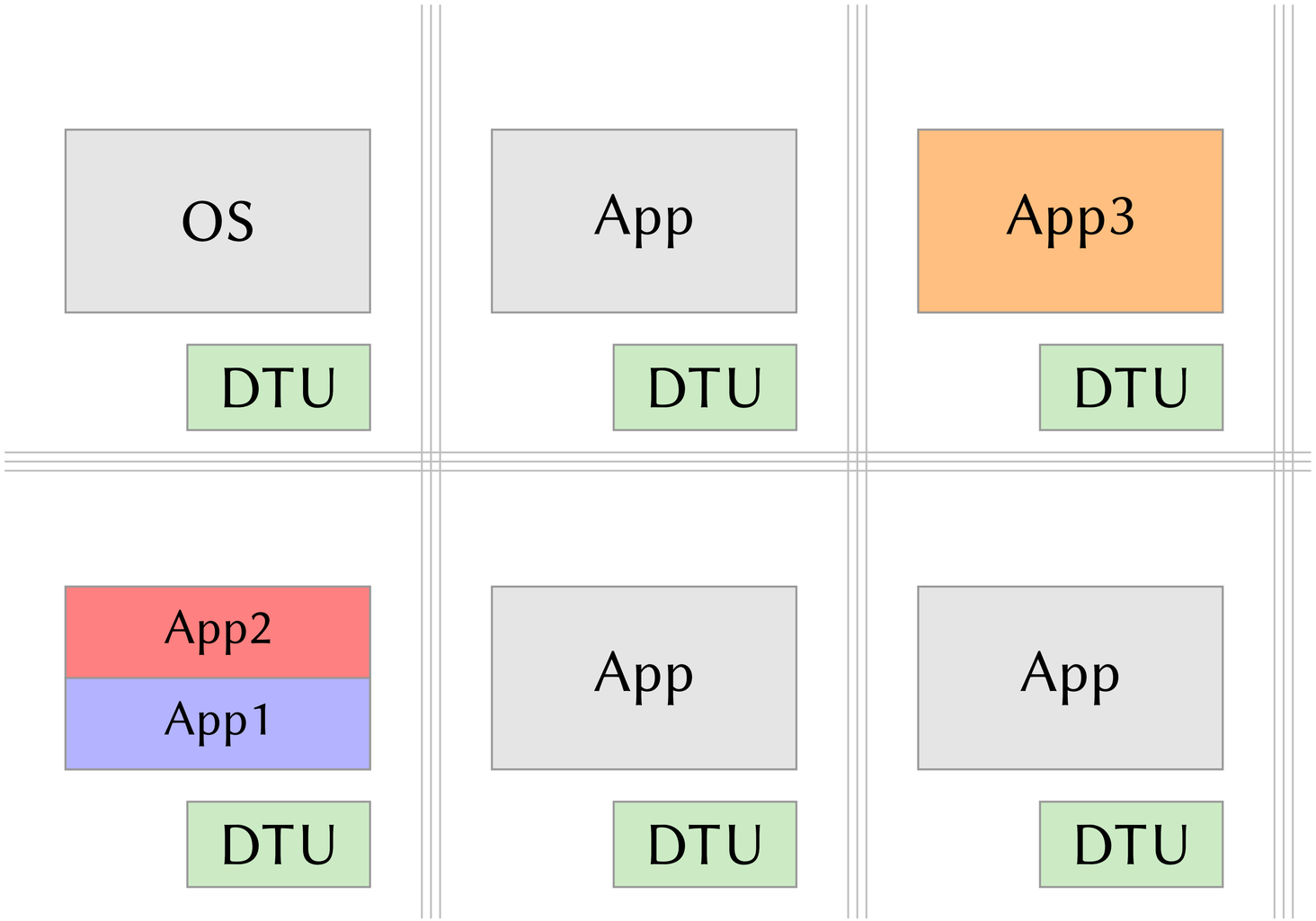
Multiplexing Approaches



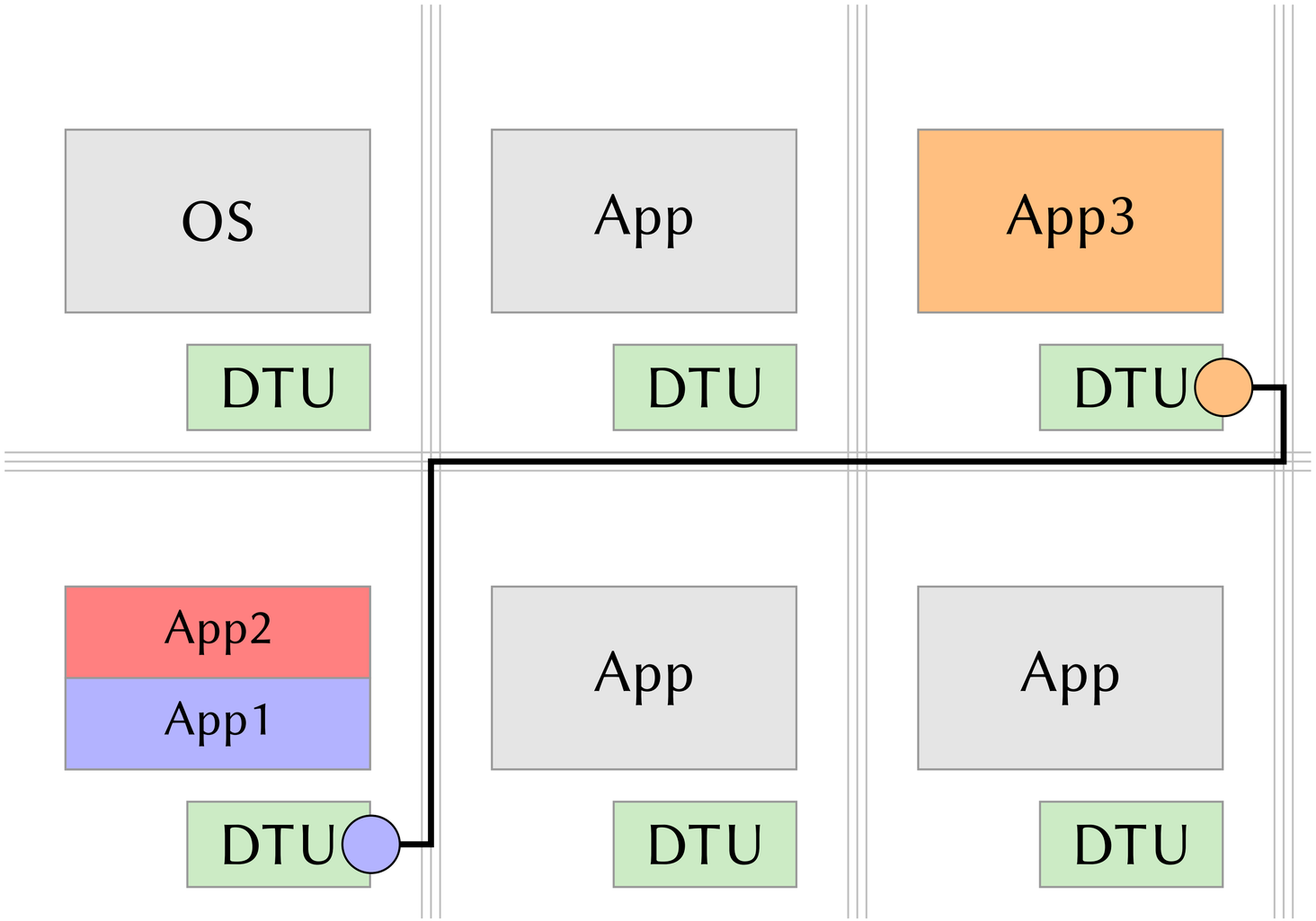
Multiplexing vs. Fast-Path Communication



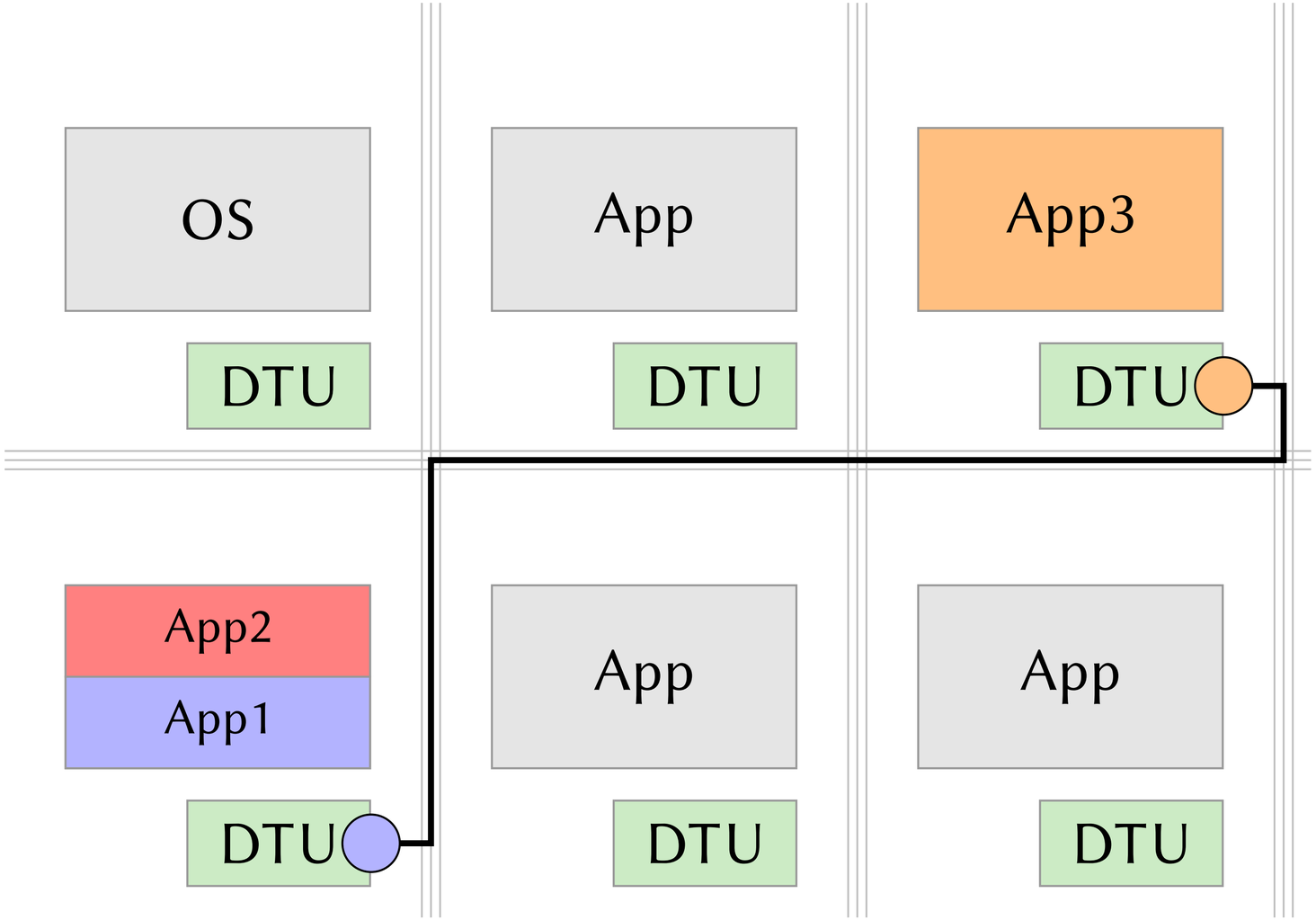
Multiplexing vs. Fast-Path Communication



Multiplexing vs. Fast-Path Communication

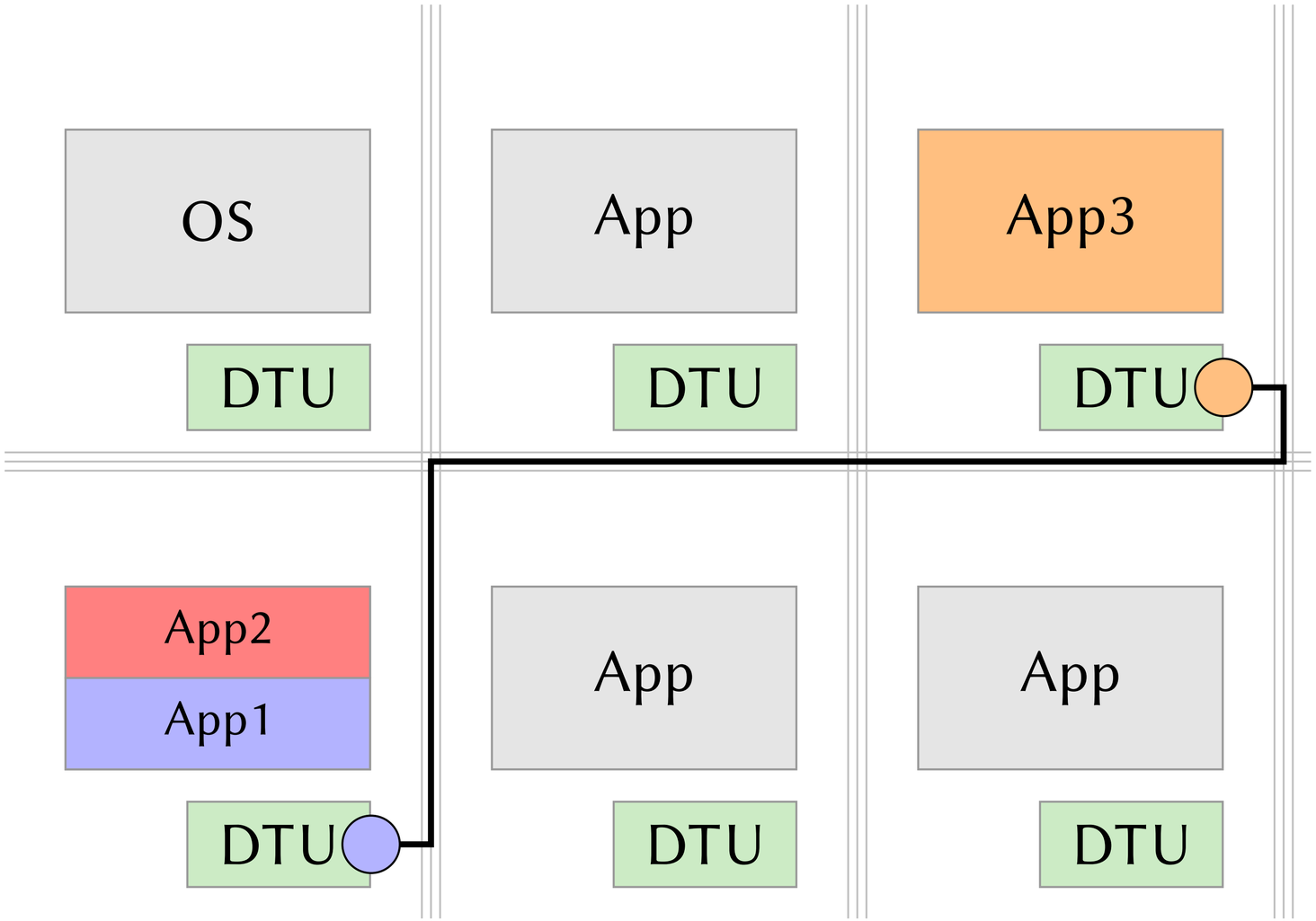


Multiplexing vs. Fast-Path Communication



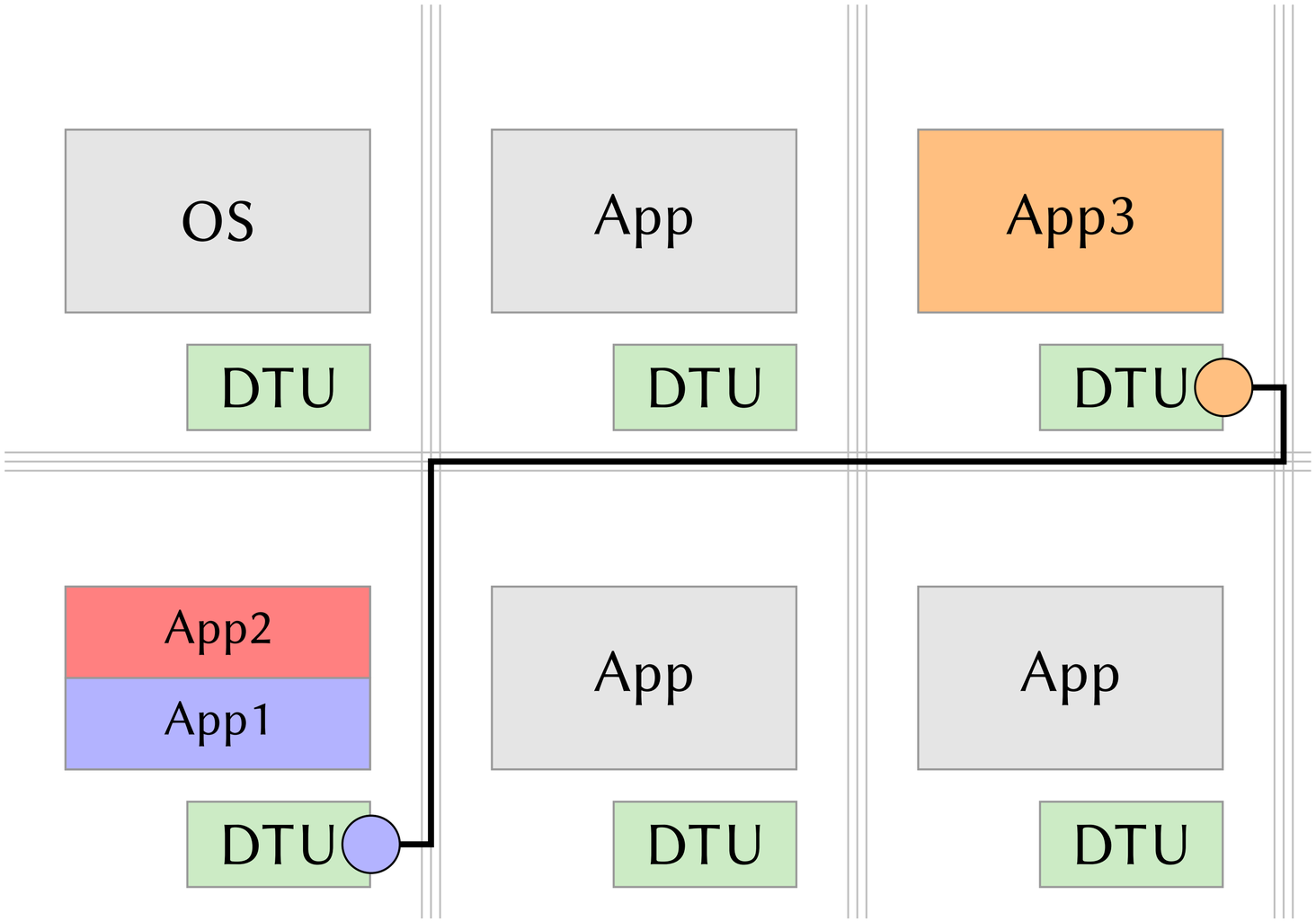
- Suspend App1 until new message, schedule App2

Multiplexing vs. Fast-Path Communication



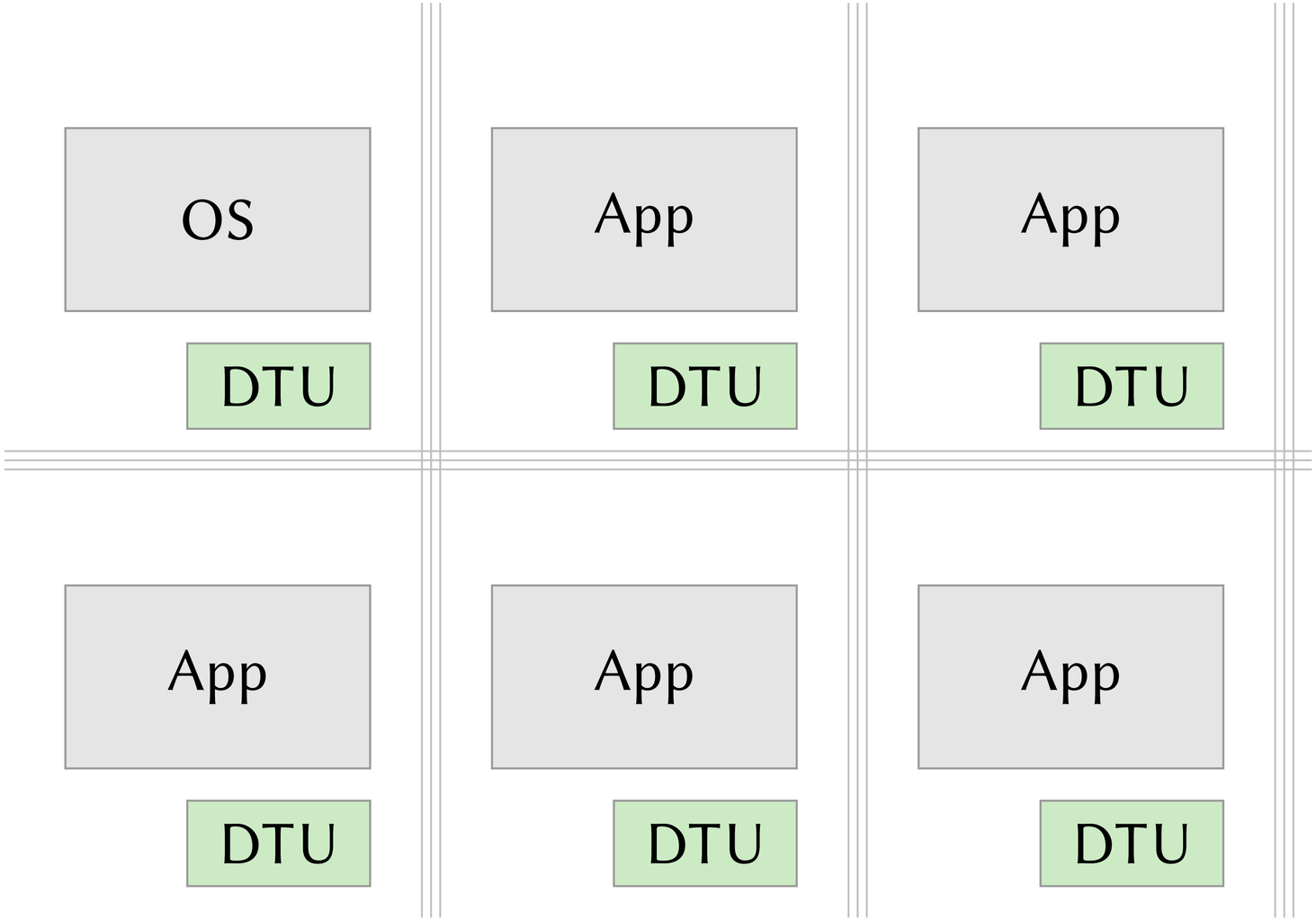
- Suspend App1 until new message, schedule App2
- Resume App1 upon new message

Multiplexing vs. Fast-Path Communication

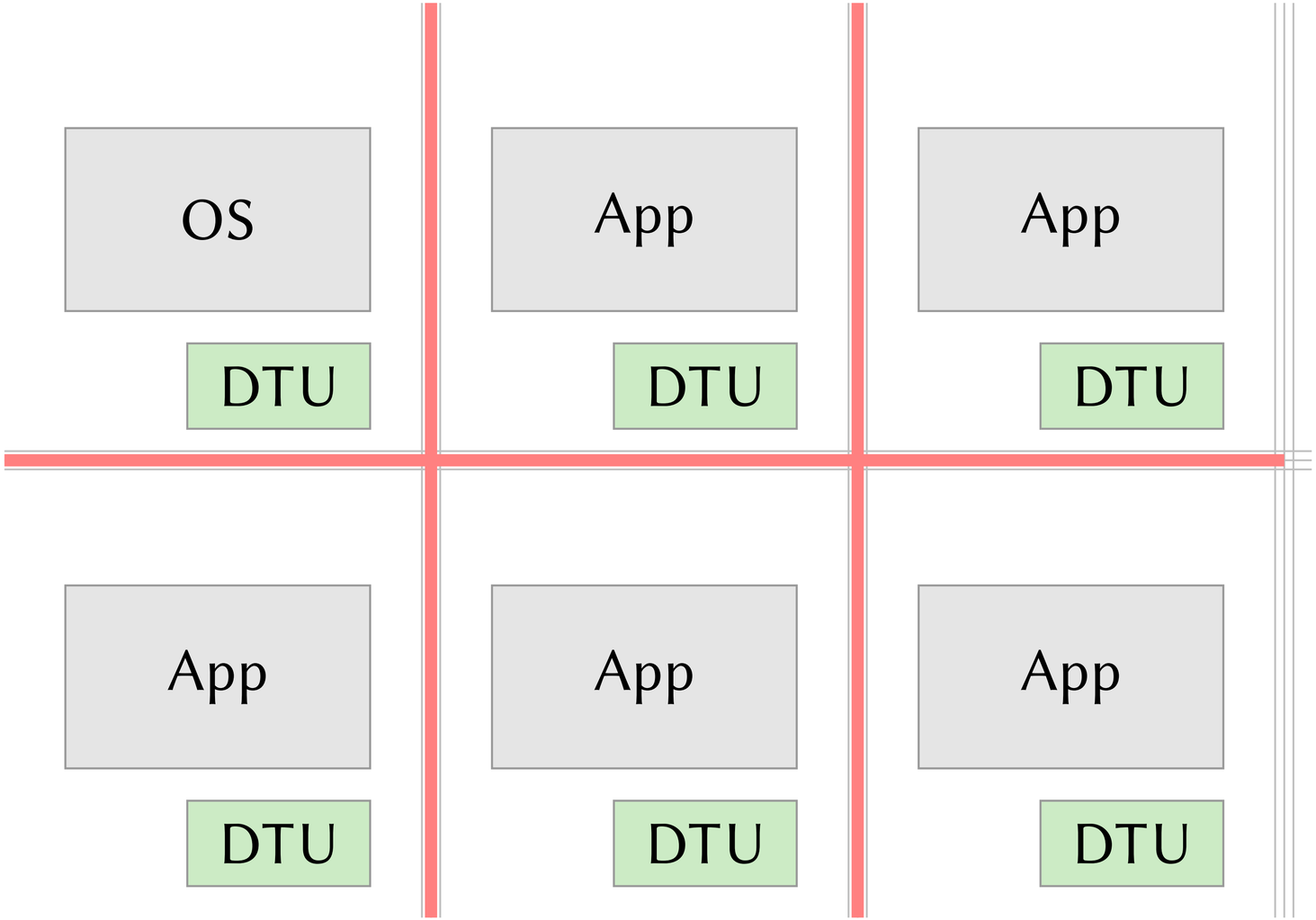


- Suspend App1 until new message, schedule App2
- Resume App1 upon new message
- Multiplexing conflicts with fast-path communication

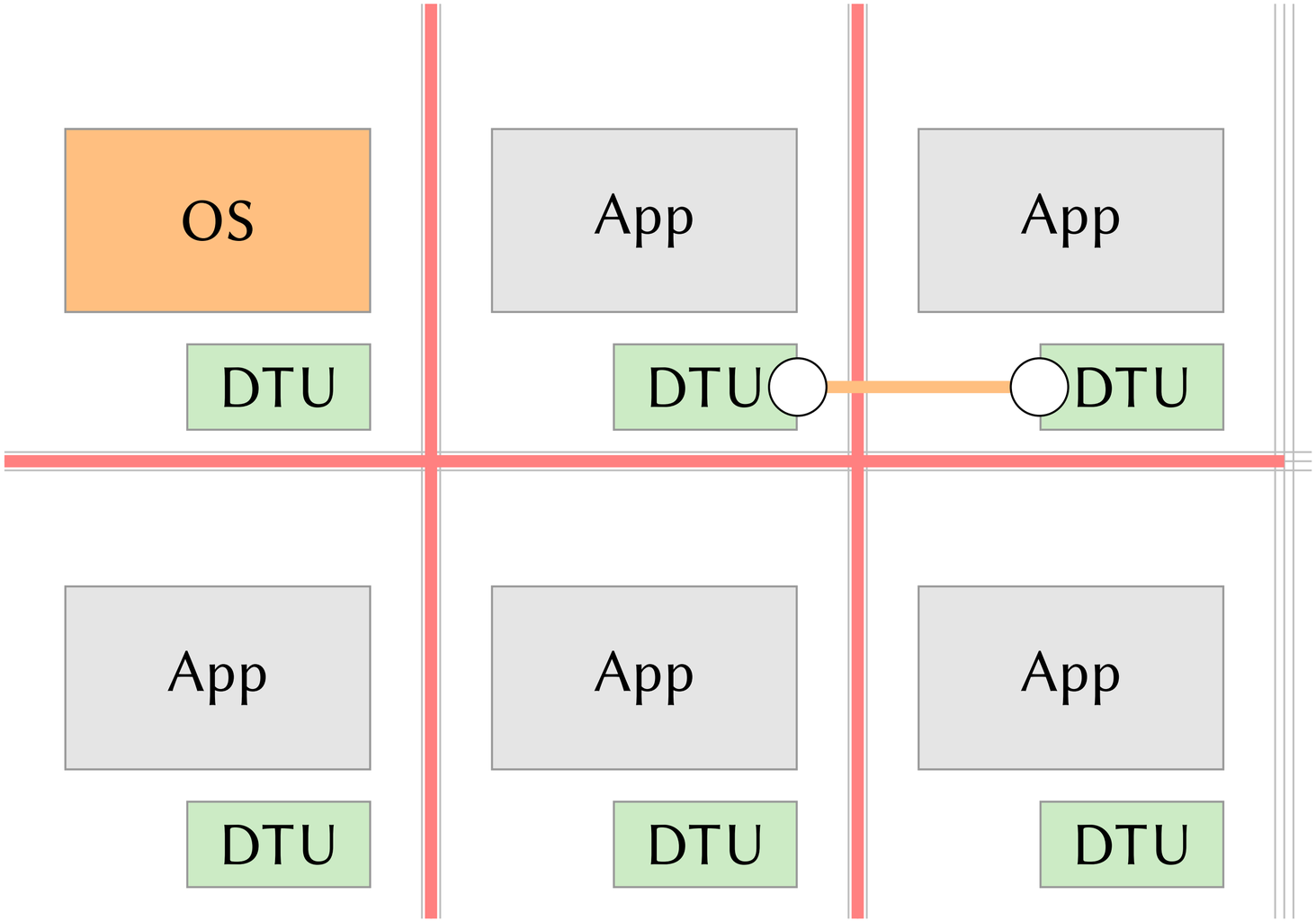
Multiplexing vs. Tile Isolation



Multiplexing vs. Tile Isolation

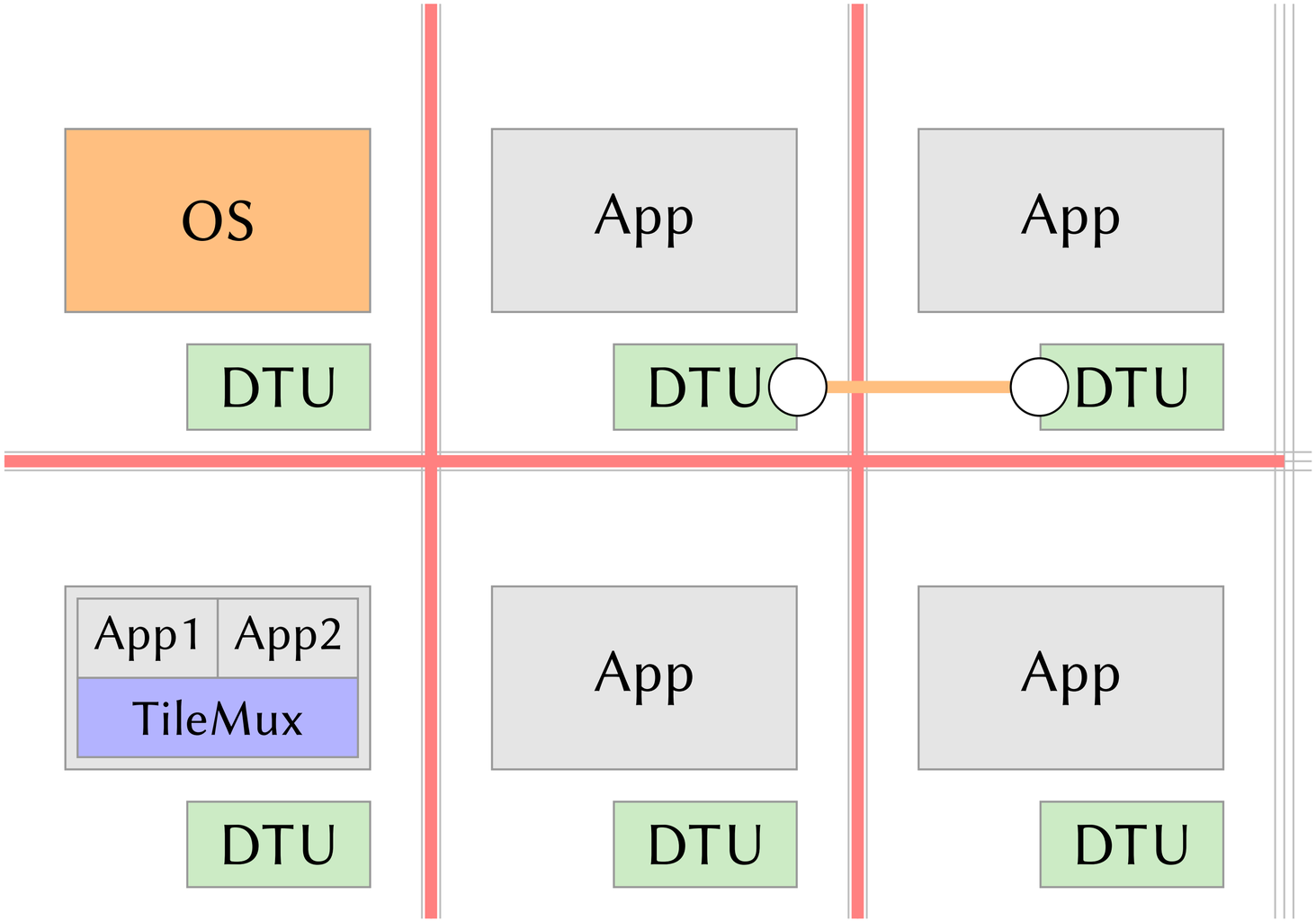


Multiplexing vs. Tile Isolation



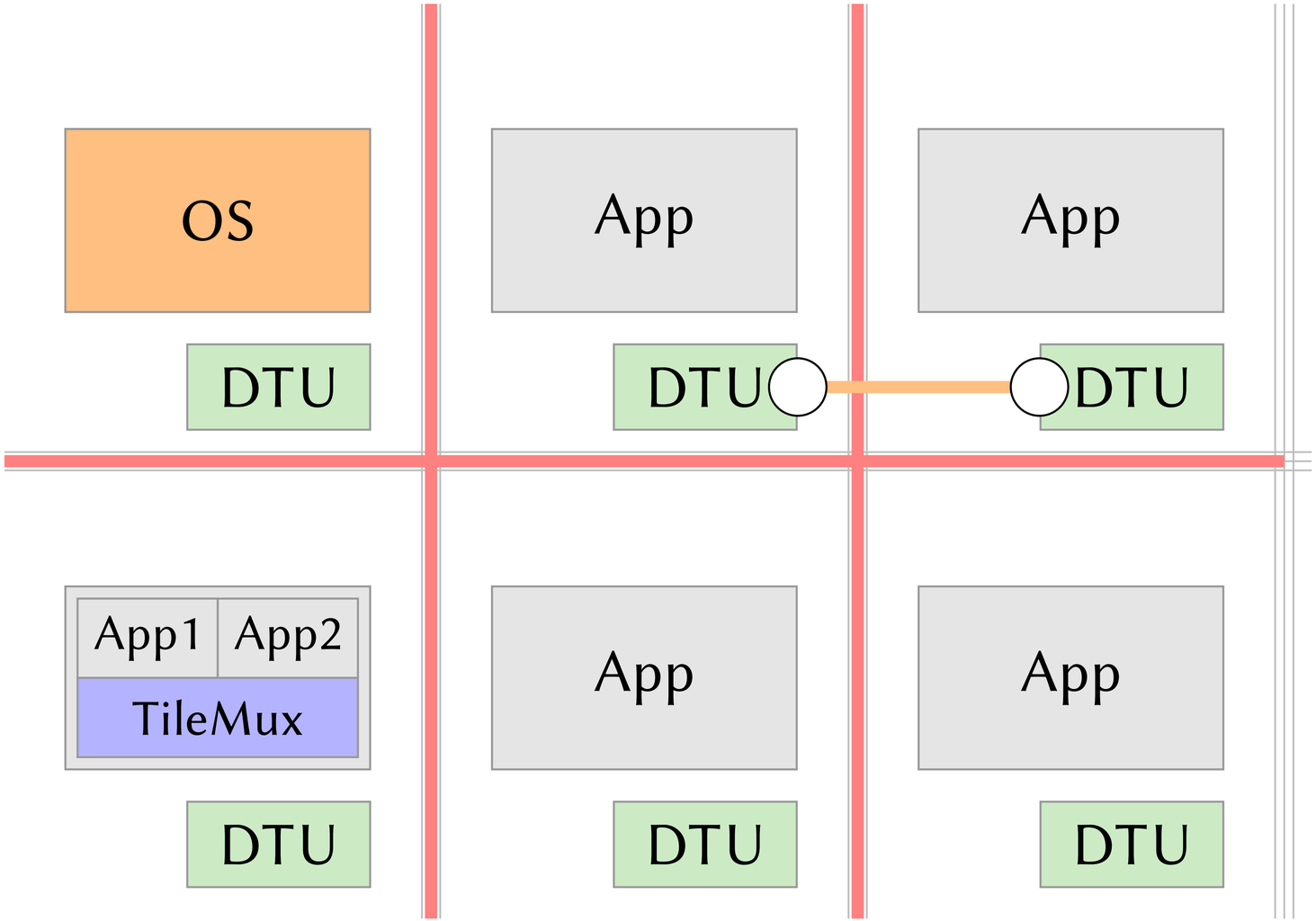
- Only the OS can provide access to tile-external resources

Multiplexing vs. Tile Isolation



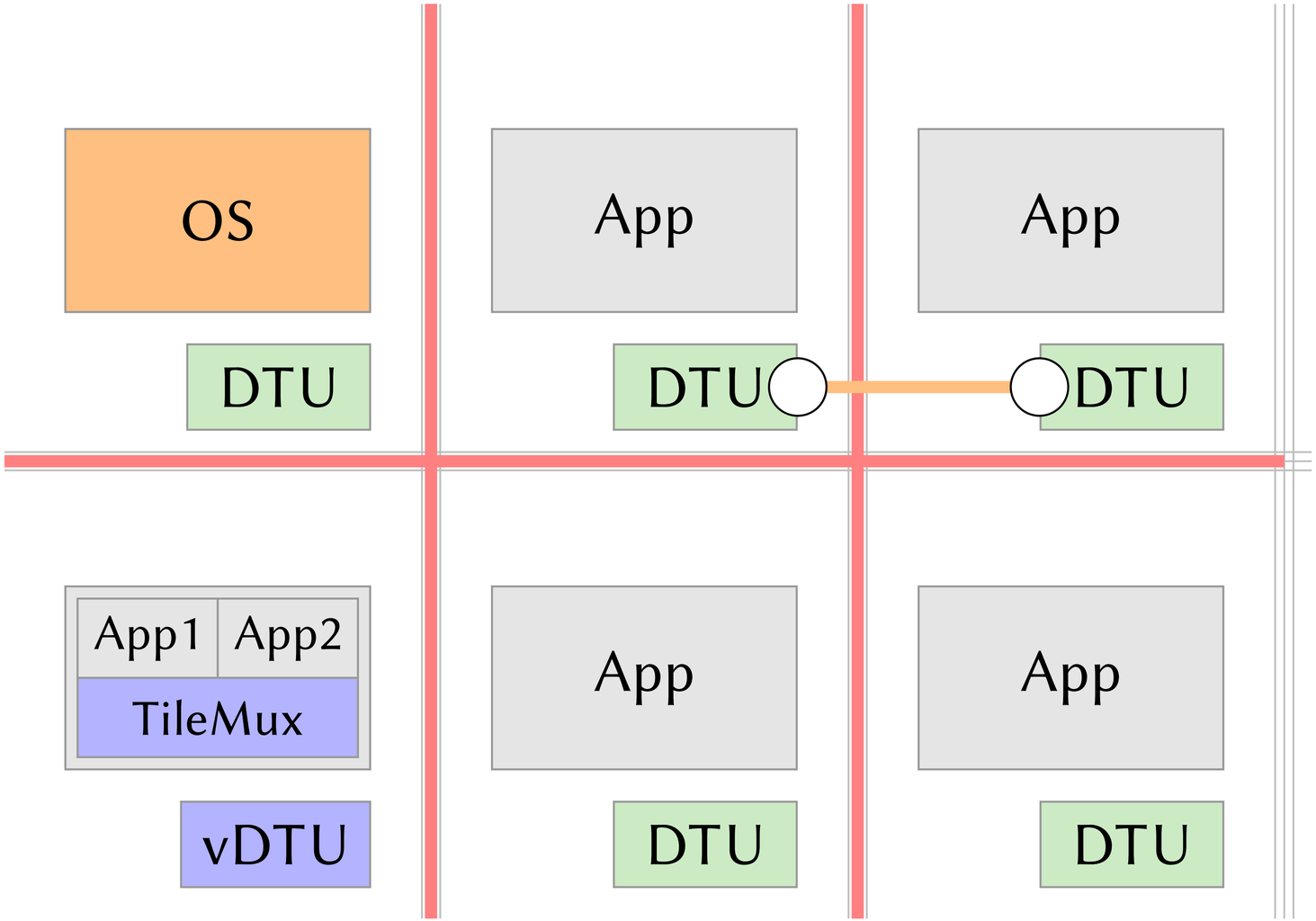
- Only the OS can provide access to tile-external resources
- Restoring DTU state provides access to **all** resources

Multiplexing vs. Tile Isolation



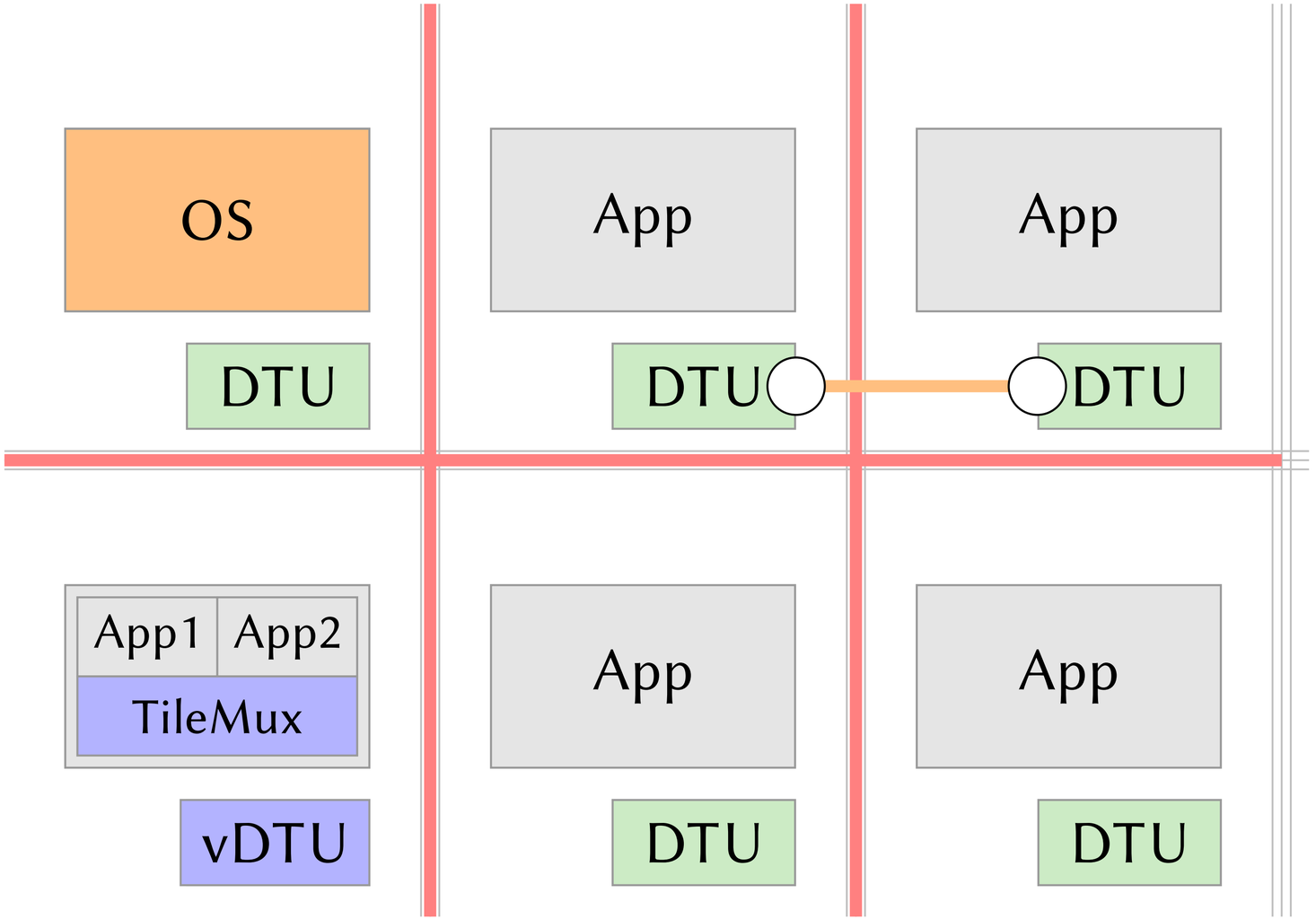
- Only the OS can provide access to tile-external resources
- Restoring DTU state provides access to **all** resources
- TileMux **must not** restore DTU state!

Multiplexing vs. Tile Isolation



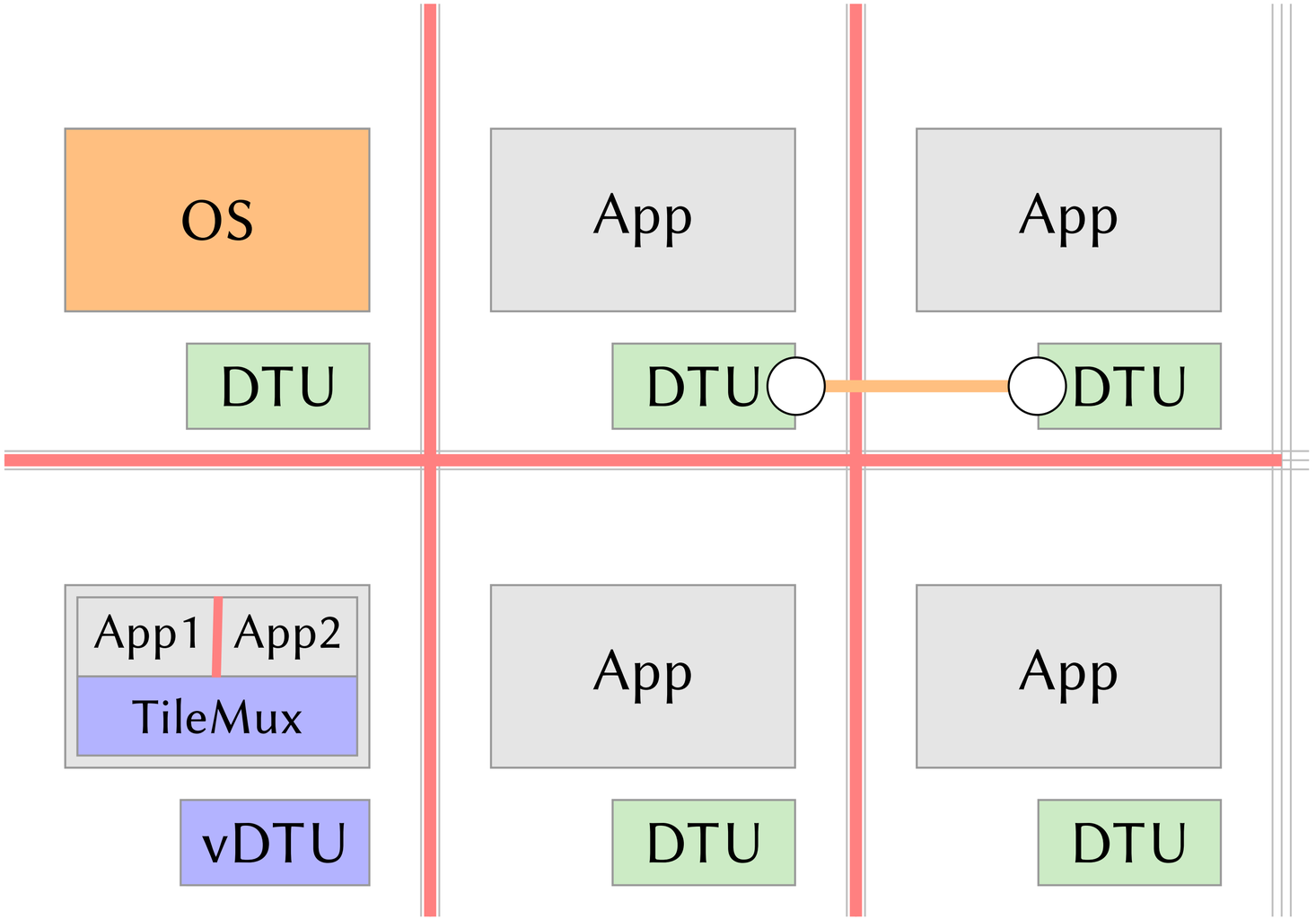
- Only the OS can provide access to tile-external resources
- Restoring DTU state provides access to **all** resources
- TileMux **must not** restore DTU state!

Multiplexing vs. Tile Isolation



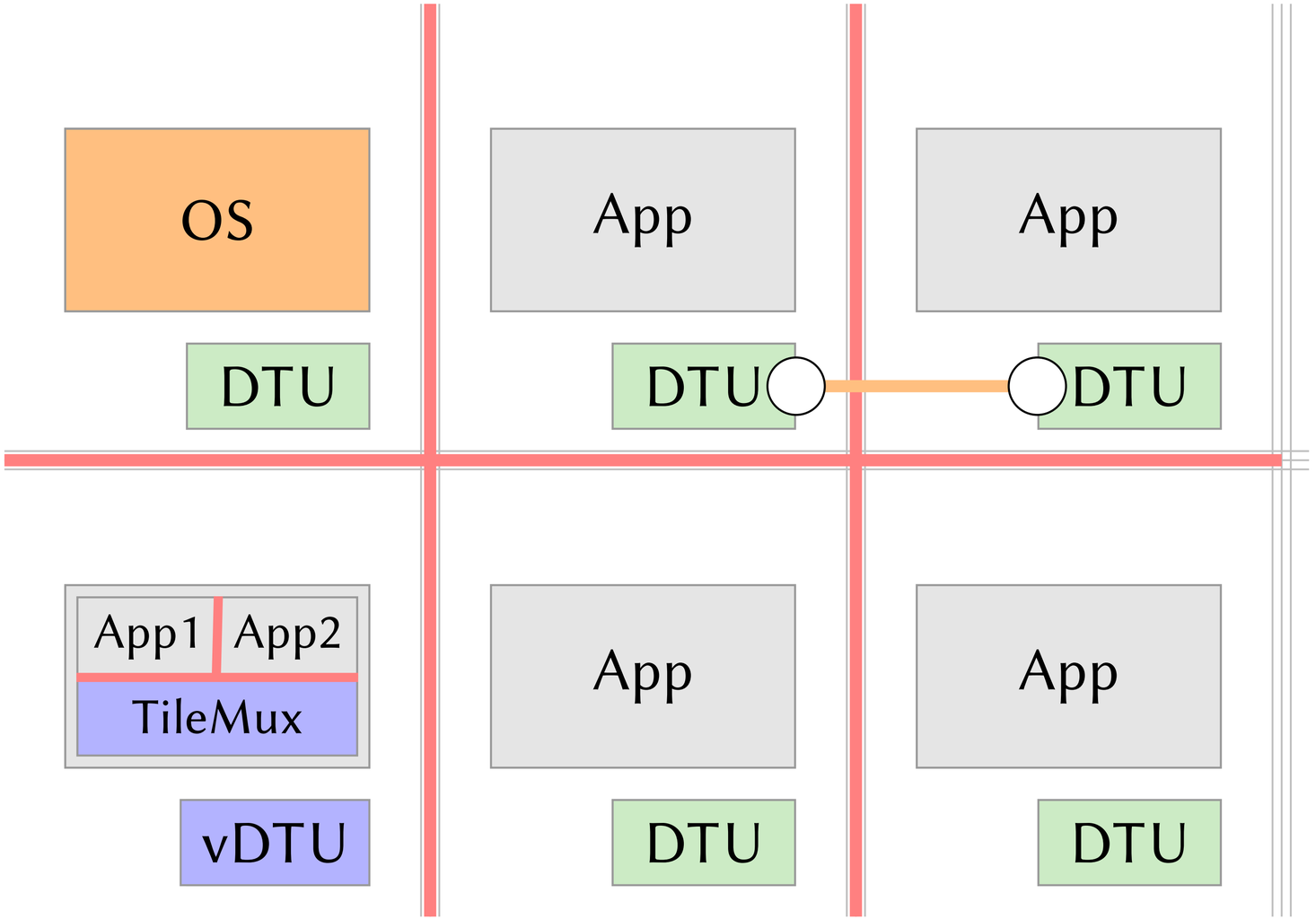
- M³* provides better isolation than conventional architectures

Multiplexing vs. Tile Isolation



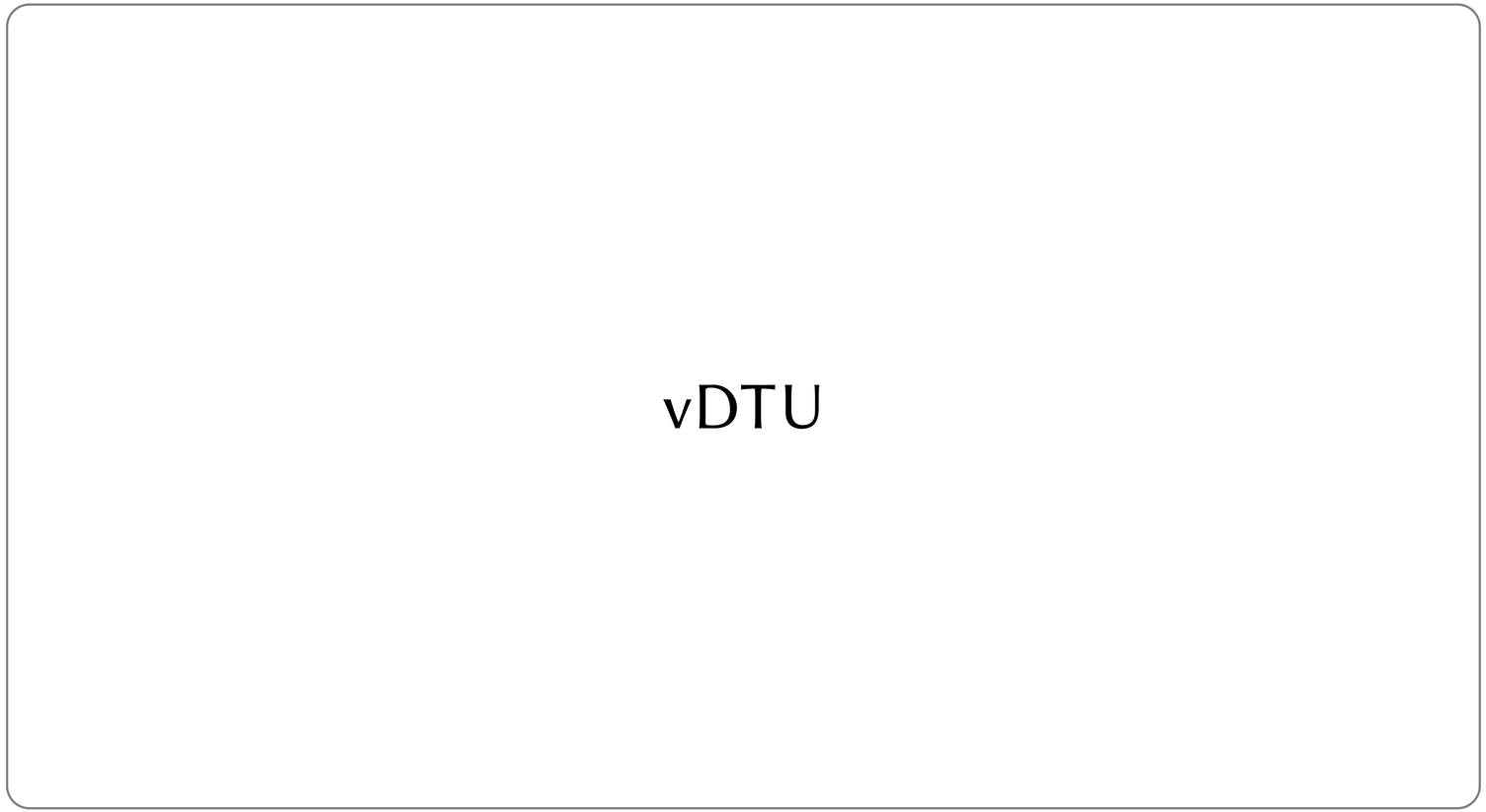
- M³* provides better isolation than conventional architectures
- M³x and M³v trade some isolation for better resource utilization

Multiplexing vs. Tile Isolation

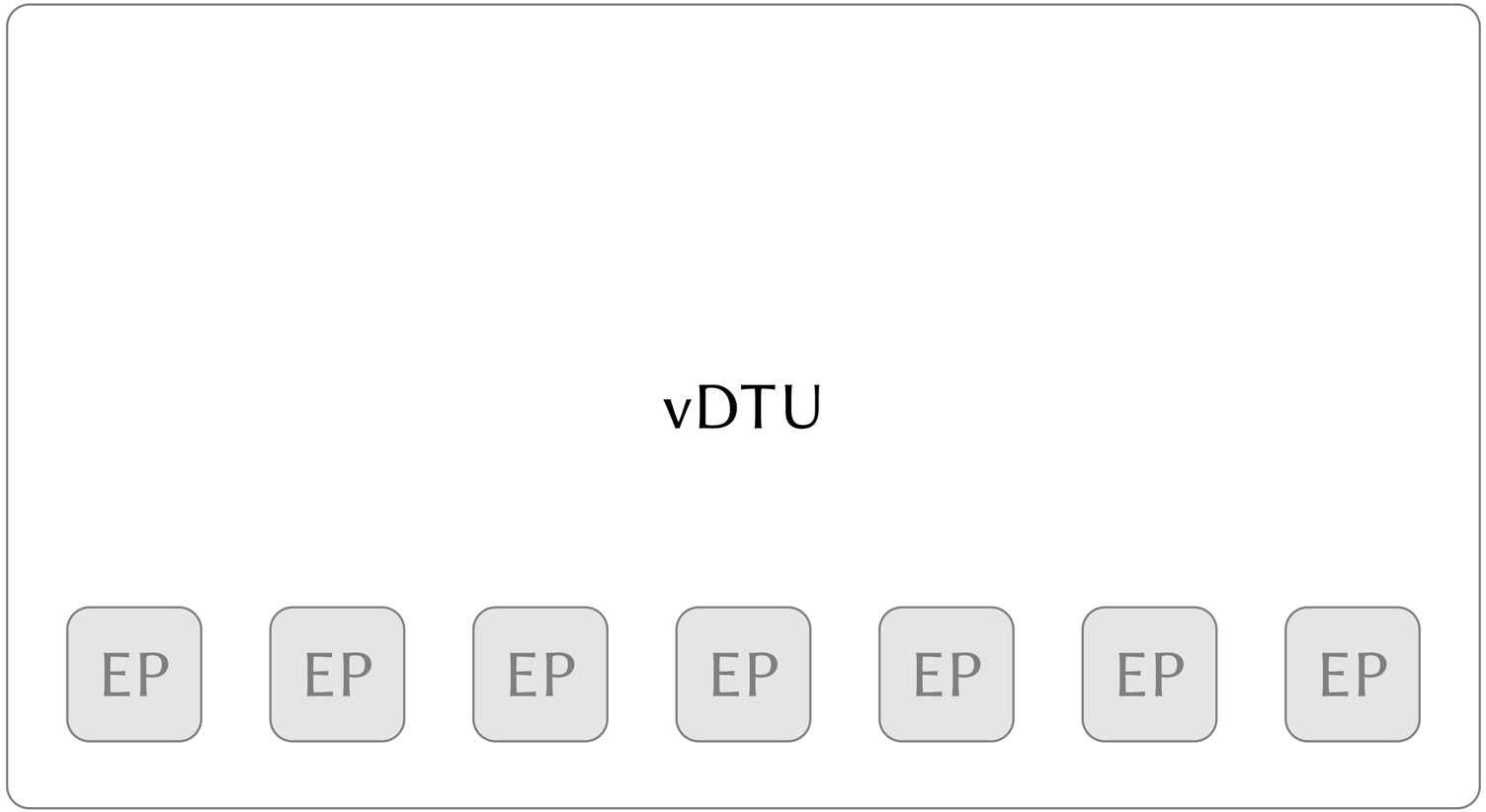


- M³* provides better isolation than conventional architectures
- M³x and M³v trade some isolation for better resource utilization
- M³v trades some more isolation for better efficiency

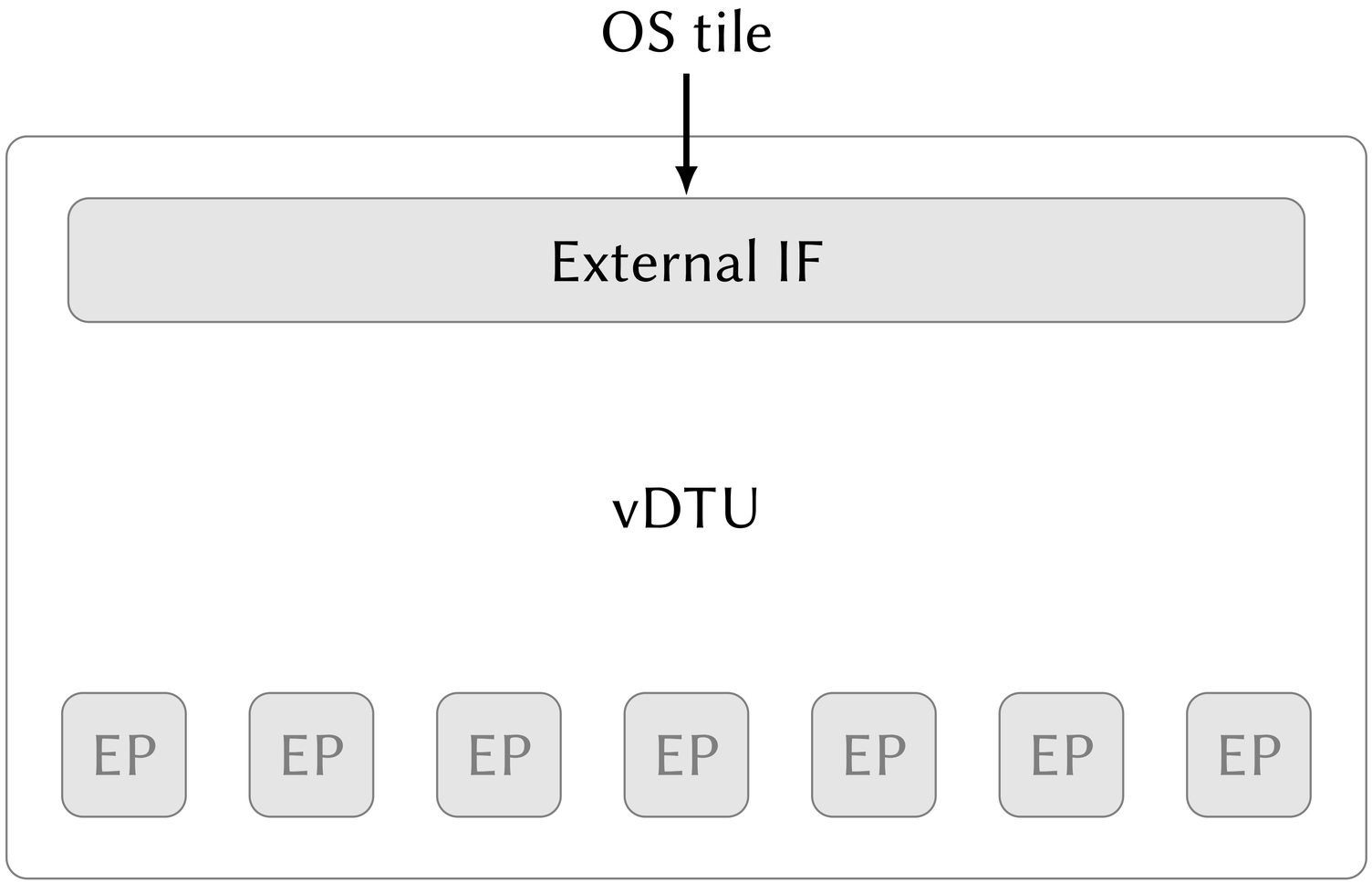
Virtualizing the DTU



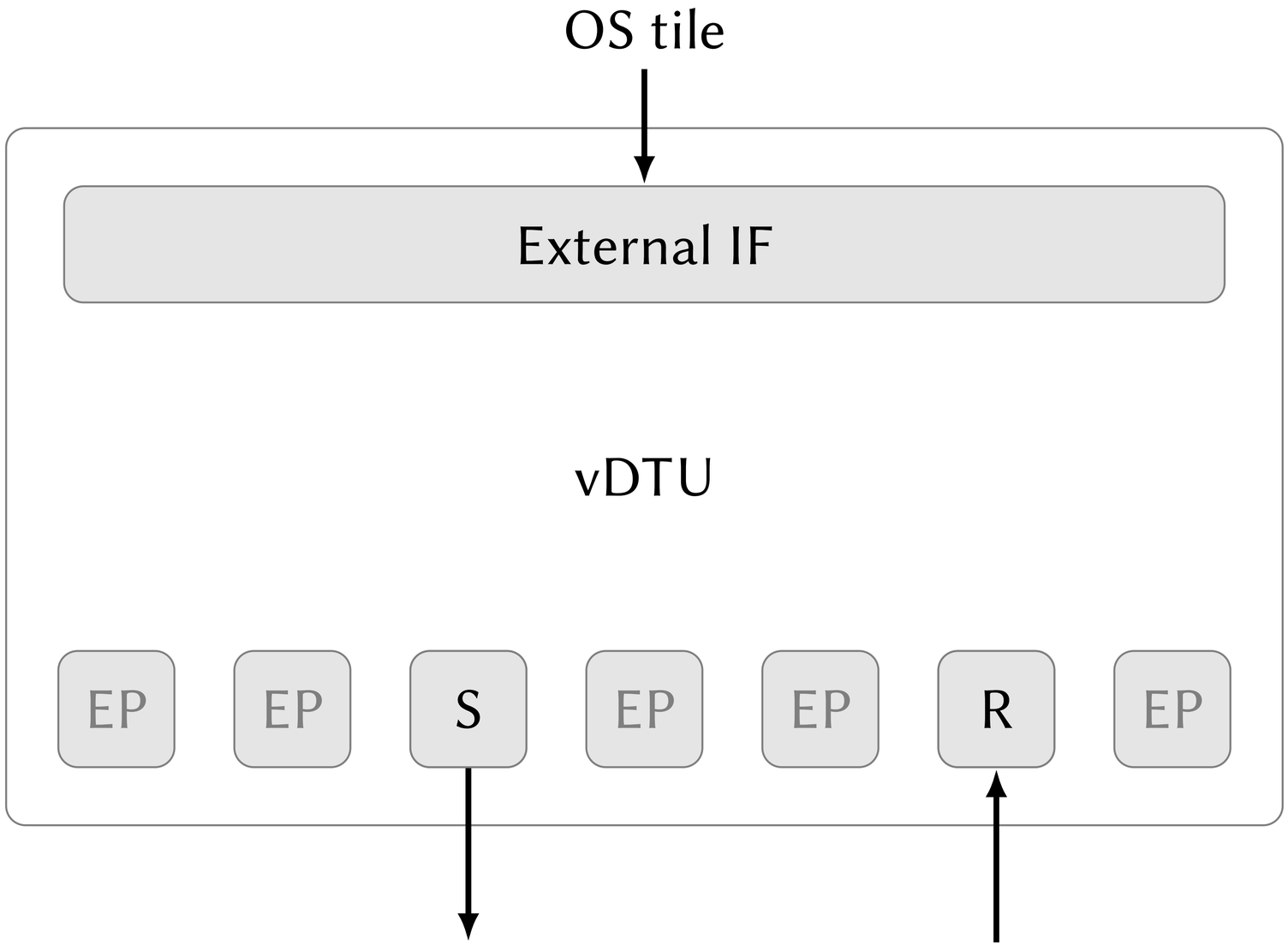
Virtualizing the DTU



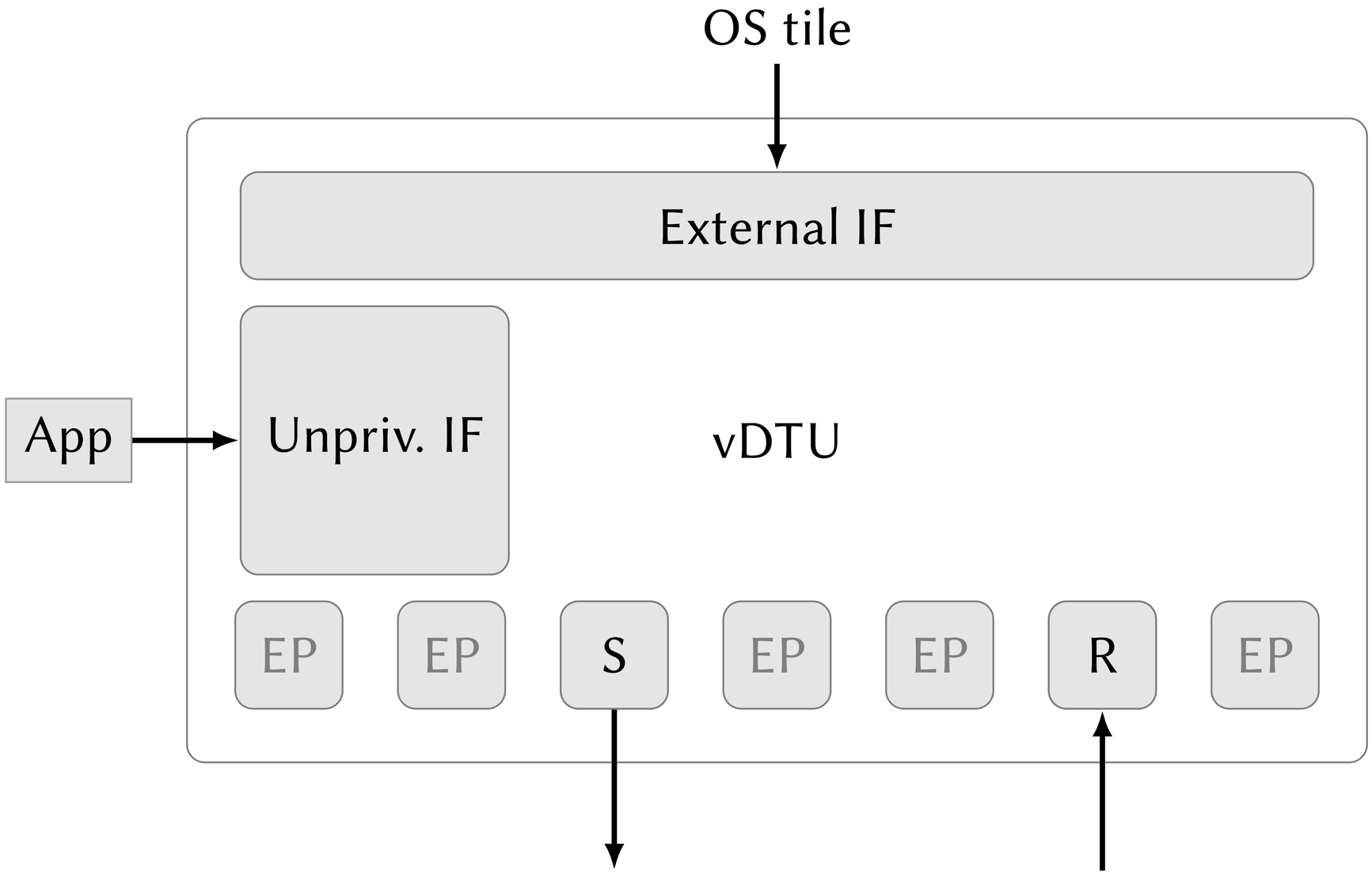
Virtualizing the DTU



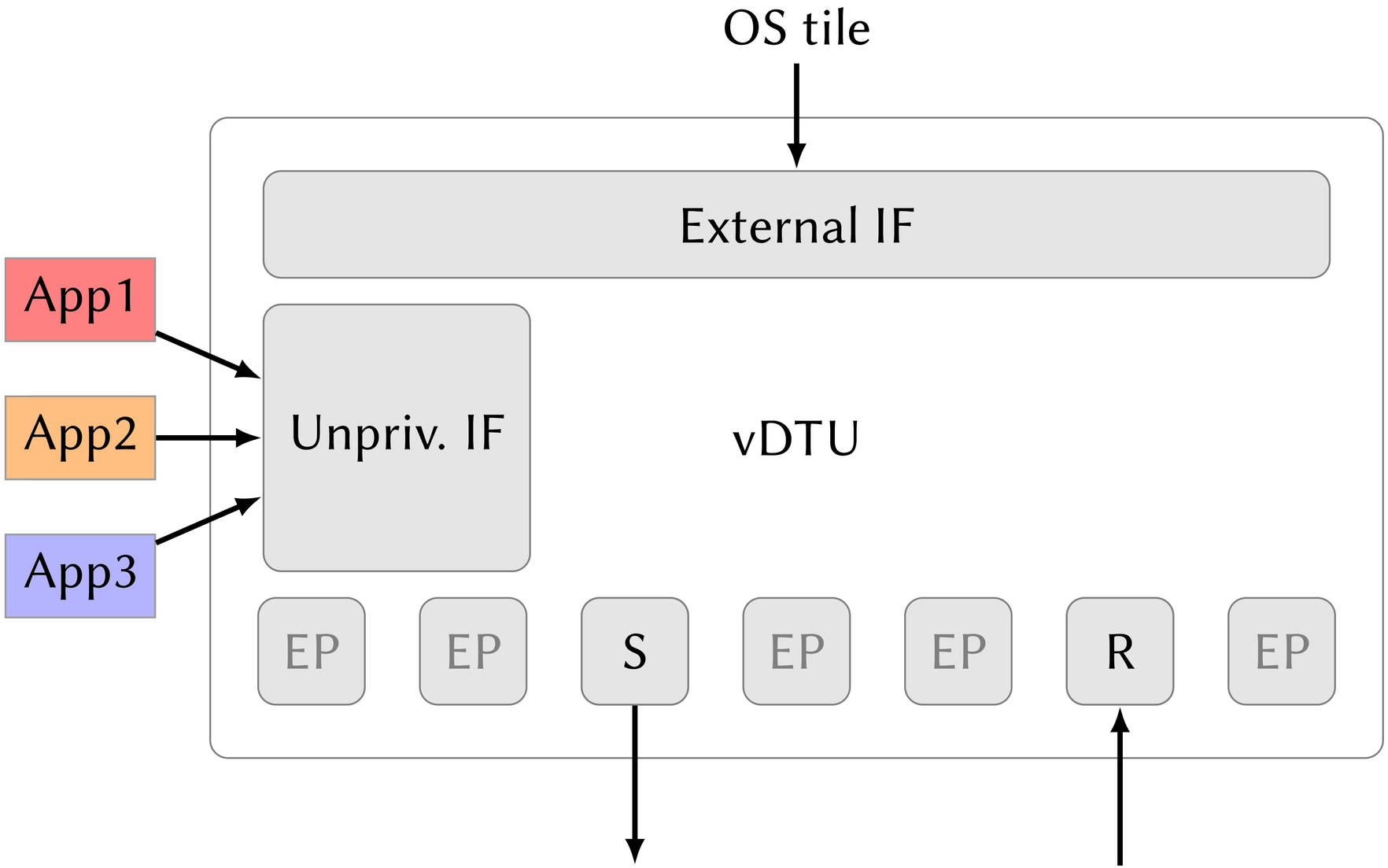
Virtualizing the DTU



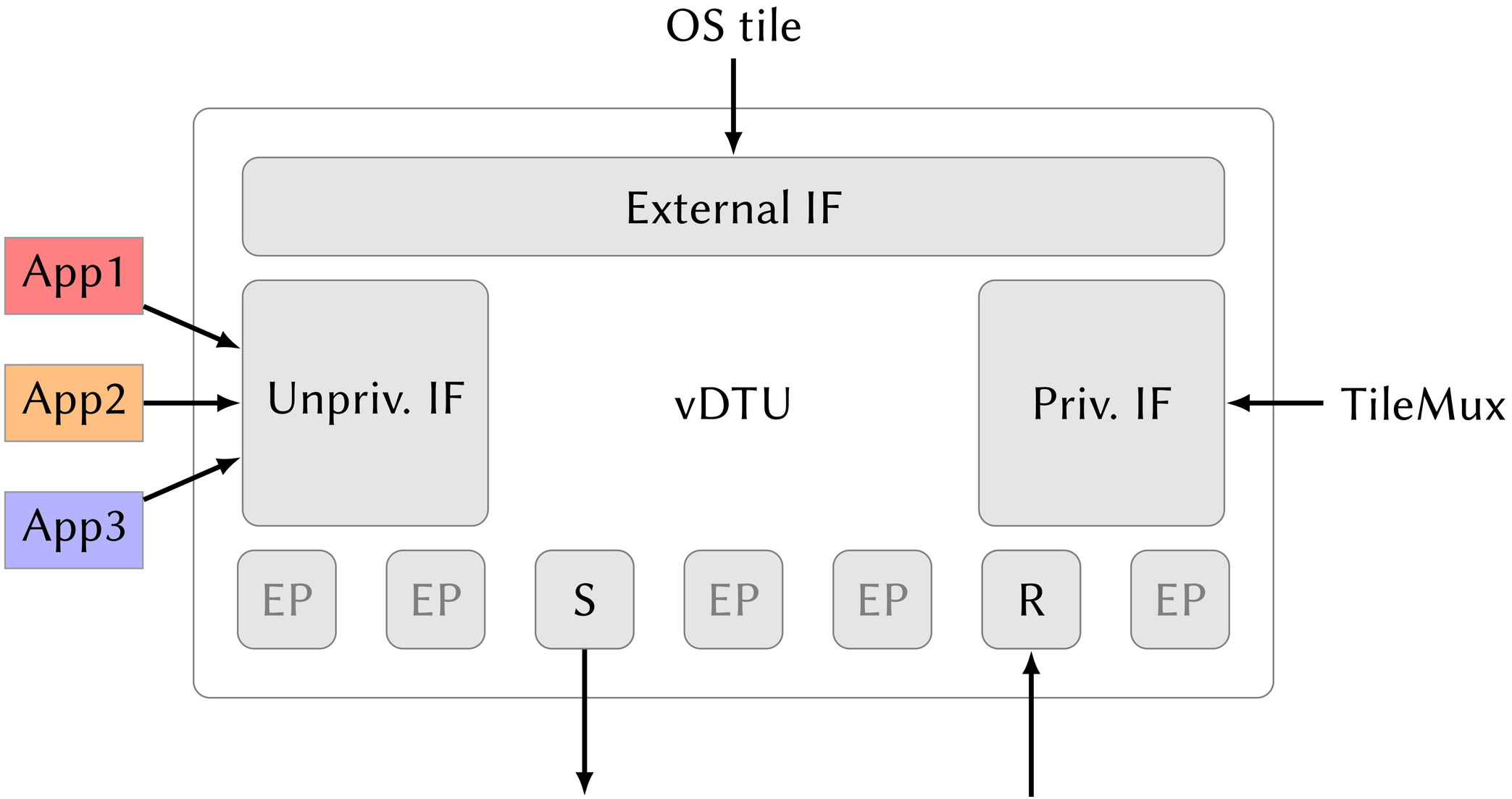
Virtualizing the DTU



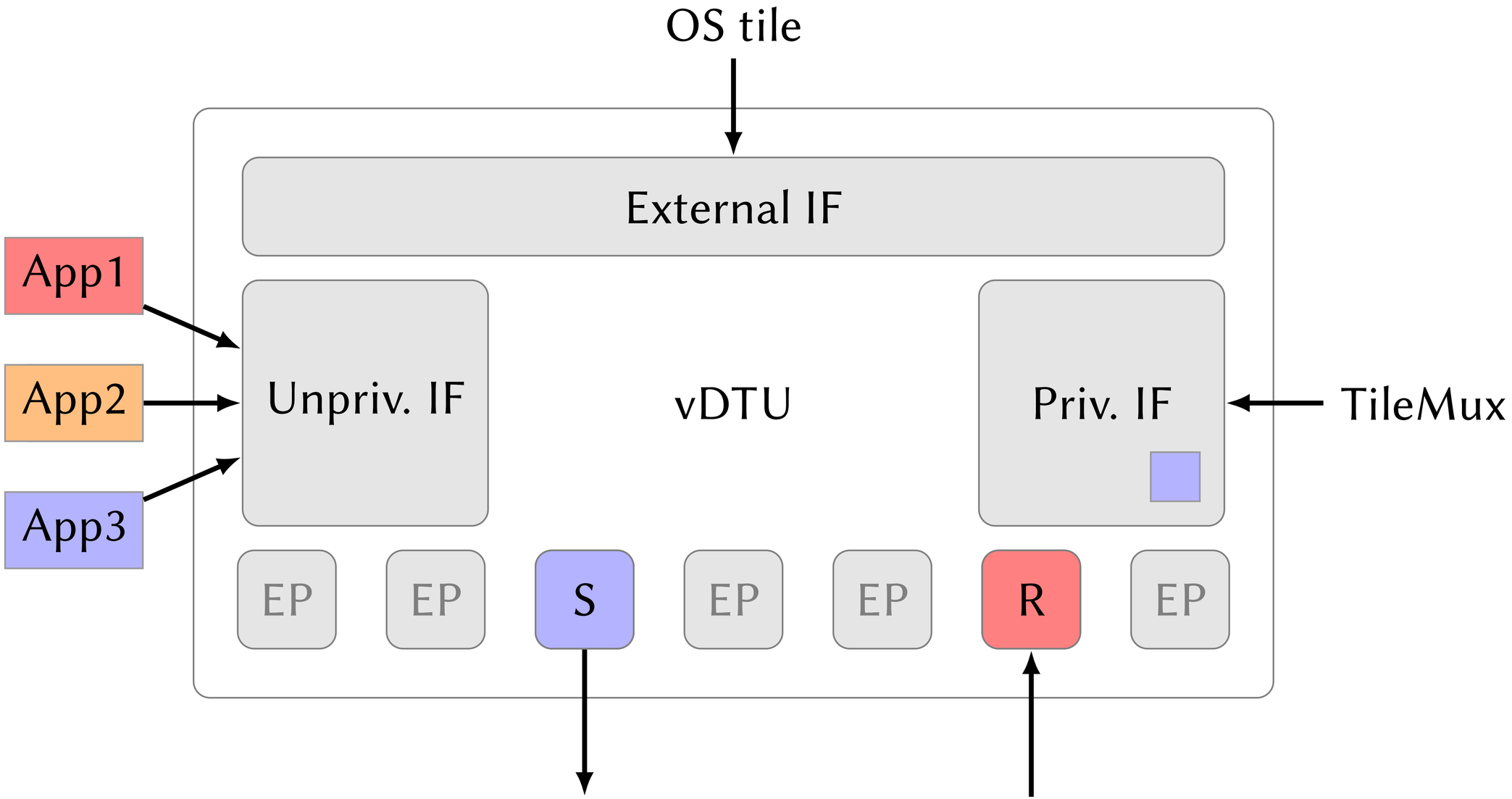
Virtualizing the DTU



Virtualizing the DTU



Virtualizing the DTU



Blocking Applications

Blocking Applications

Problem

Blocking Applications

Problem

- if the current app waits for a message, other apps should make progress

Blocking Applications

Problem

- if the current app waits for a message, other apps should make progress
- applications fetch new messages directly from the vDTU

Blocking Applications

Problem

- if the current app waits for a message, other apps should make progress
- applications fetch new messages directly from the vDTU
- if there are none *and* other apps are ready, TileMux is invoked to block

Blocking Applications

Problem

- if the current app waits for a message, other apps should make progress
- applications fetch new messages directly from the vDTU
- if there are none *and* other apps are ready, TileMux is invoked to block
- race condition: check for new message and blocking (like lost wakeup)

Blocking Applications

Problem

- if the current app waits for a message, other apps should make progress
- applications fetch new messages directly from the vDTU
- if there are none *and* other apps are ready, TileMux is invoked to block
- race condition: check for new message and blocking (like lost wakeup)

Solution

Blocking Applications

Problem

- if the current app waits for a message, other apps should make progress
- applications fetch new messages directly from the vDTU
- if there are none *and* other apps are ready, TileMux is invoked to block
- race condition: check for new message and blocking (like lost wakeup)

Solution

- vDTU tracks number of messages for current application

Blocking Applications

Problem

- if the current app waits for a message, other apps should make progress
- applications fetch new messages directly from the vDTU
- if there are none *and* other apps are ready, TileMux is invoked to block
- race condition: check for new message and blocking (like lost wakeup)

Solution

- vDTU tracks number of messages for current application
- privileged interface can atomically switch to a new application

Unblocking Applications

Unblocking Applications

M^3x – Original DTU

Unblocking Applications

M^3x – Original DTU

- if receiver is blocked, incoming messages cannot be stored in memory

Unblocking Applications

M^3x – Original DTU

- if receiver is blocked, incoming messages cannot be stored in memory
- because the receive endpoint is not available in the DTU

Unblocking Applications

M³x – Original DTU

- if receiver is blocked, incoming messages cannot be stored in memory
- because the receive endpoint is not available in the DTU
- fall back to slow path by forwarding message via the OS tile

Unblocking Applications

M³x – Original DTU

- if receiver is blocked, incoming messages cannot be stored in memory
- because the receive endpoint is not available in the DTU
- fall back to slow path by forwarding message via the OS tile

M³v – Virtualized DTU

Unblocking Applications

M³x – Original DTU

- if receiver is blocked, incoming messages cannot be stored in memory
- because the receive endpoint is not available in the DTU
- fall back to slow path by forwarding message via the OS tile

M³v – Virtualized DTU

- the vDTU knows all endpoints at all times, can always store the message

Unblocking Applications

M³x – Original DTU

- if receiver is blocked, incoming messages cannot be stored in memory
- because the receive endpoint is not available in the DTU
- fall back to slow path by forwarding message via the OS tile

M³v – Virtualized DTU

- the vDTU knows all endpoints at all times, can always store the message
- if owner of receive endpoint is blocked, the vDTU injects an interrupt

Unblocking Applications

M³x – Original DTU

- if receiver is blocked, incoming messages cannot be stored in memory
- because the receive endpoint is not available in the DTU
- fall back to slow path by forwarding message via the OS tile

M³v – Virtualized DTU

- the vDTU knows all endpoints at all times, can always store the message
- if owner of receive endpoint is blocked, the vDTU injects an interrupt
- TileMux marks the receiver as ready

Unblocking Applications

M³x – Original DTU

- if receiver is blocked, incoming messages cannot be stored in memory
- because the receive endpoint is not available in the DTU
- fall back to slow path by forwarding message via the OS tile

M³v – Virtualized DTU

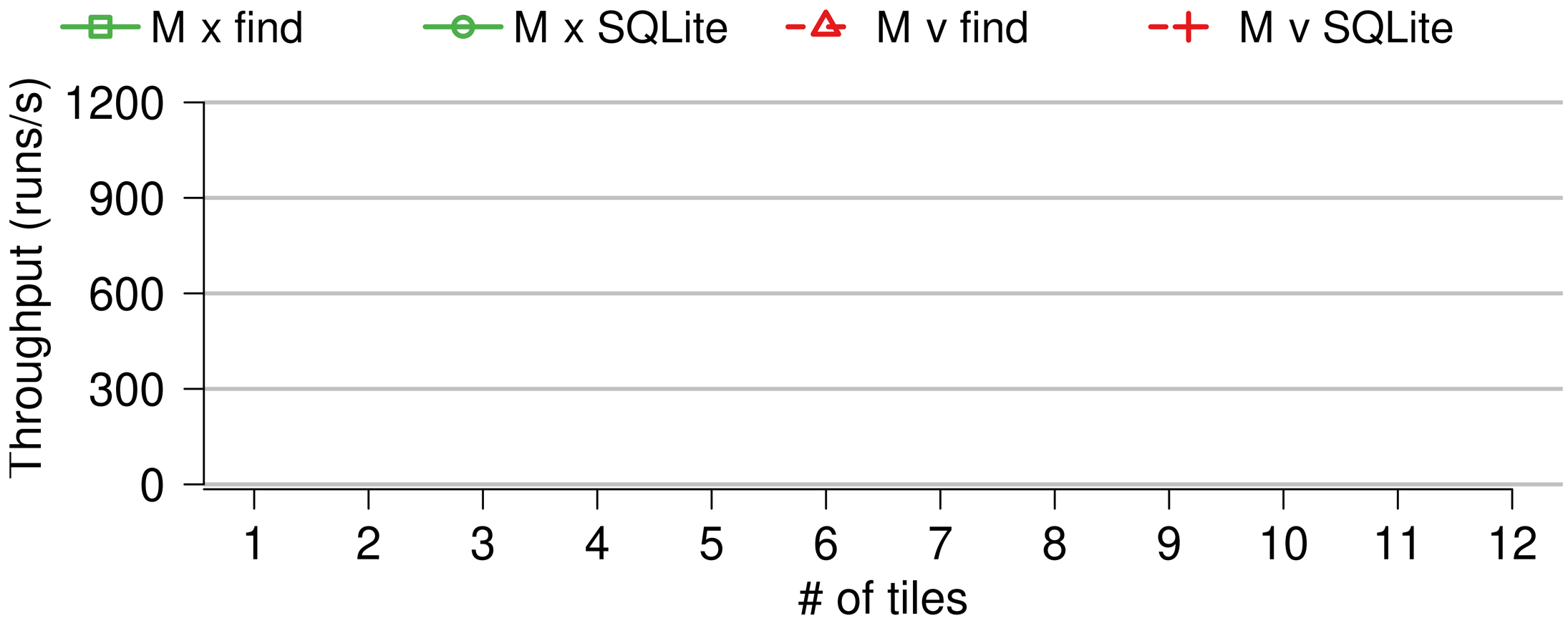
- the vDTU knows all endpoints at all times, can always store the message
- if owner of receive endpoint is blocked, the vDTU injects an interrupt
- TileMux marks the receiver as ready
- OS tile never involved, TileMux only involved if receiver is not running

Performance Comparison to M³x

- Setup: gem5 simulator, 3 GHz out-of-order x86-64 cores
- Every tile runs: SQLite/find benchmark and in-memory filesystem

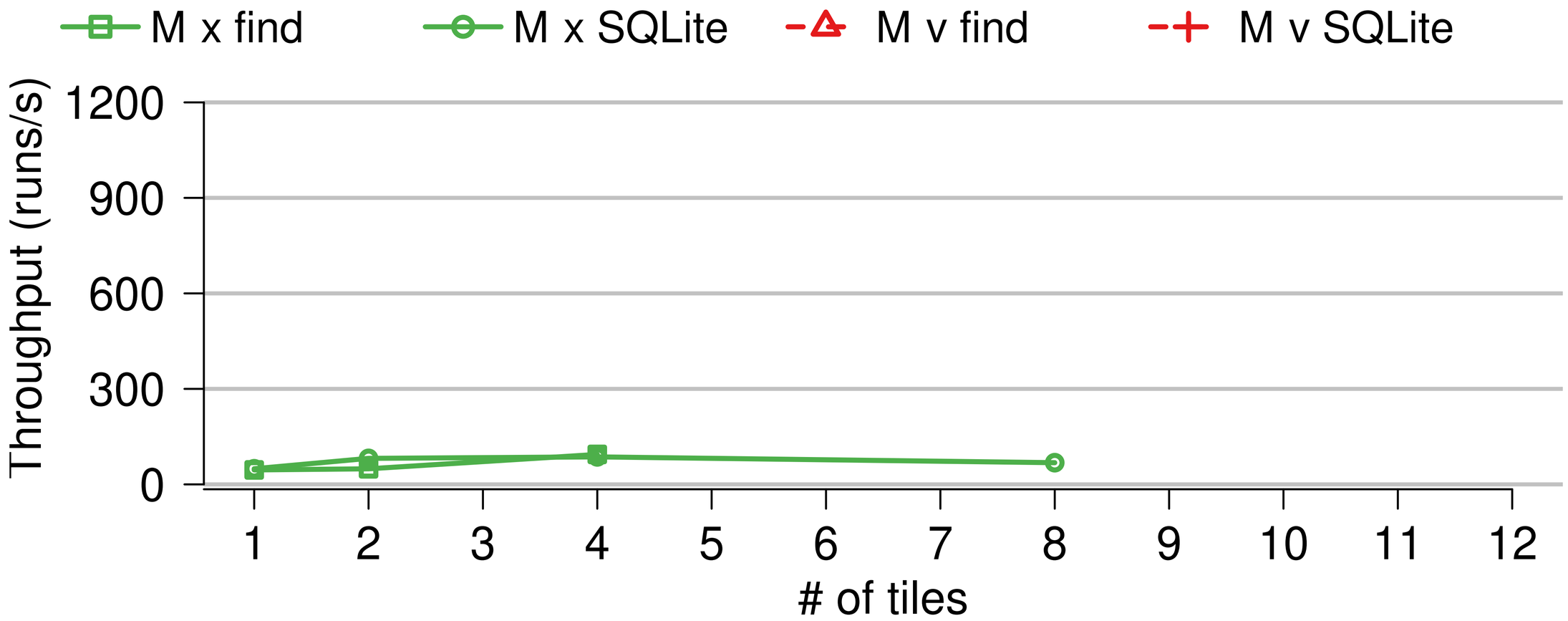
Performance Comparison to M³x

- Setup: gem5 simulator, 3 GHz out-of-order x86-64 cores
- Every tile runs: SQLite/find benchmark and in-memory filesystem



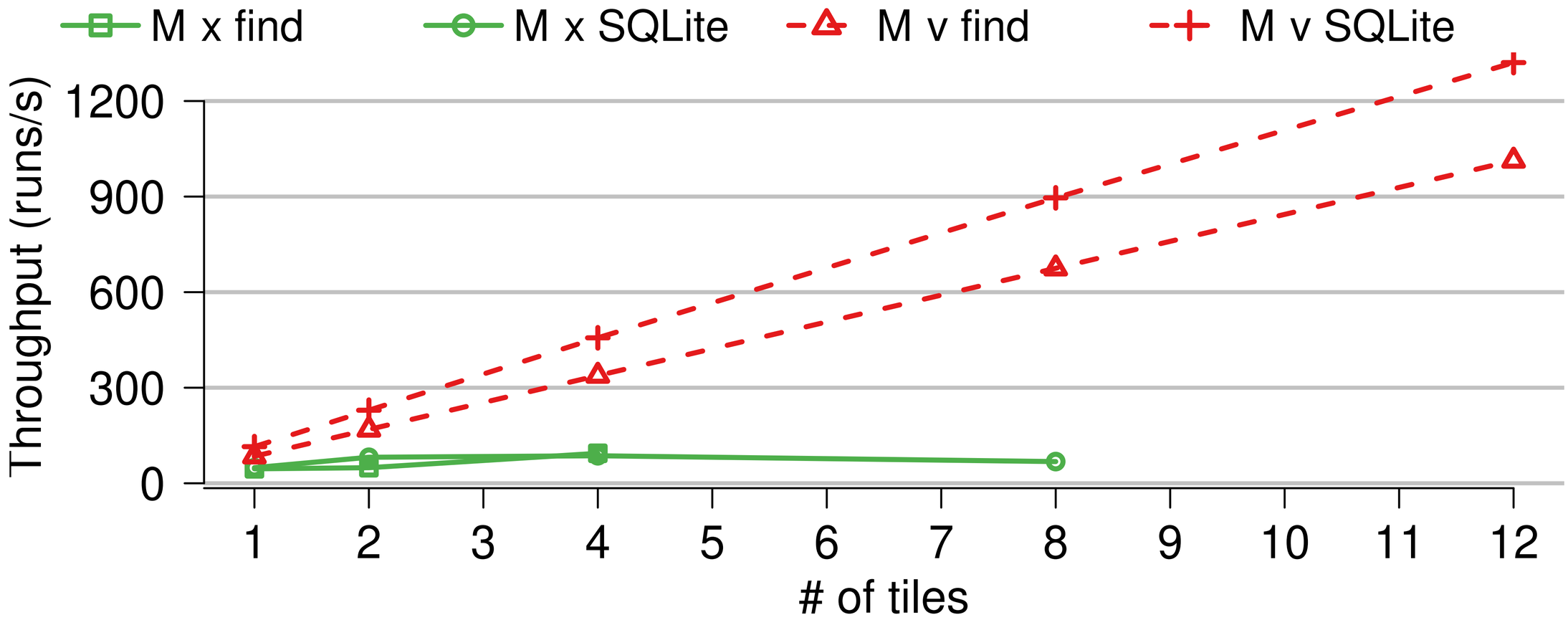
Performance Comparison to M³x

- Setup: gem5 simulator, 3 GHz out-of-order x86-64 cores
- Every tile runs: SQLite/find benchmark and in-memory filesystem



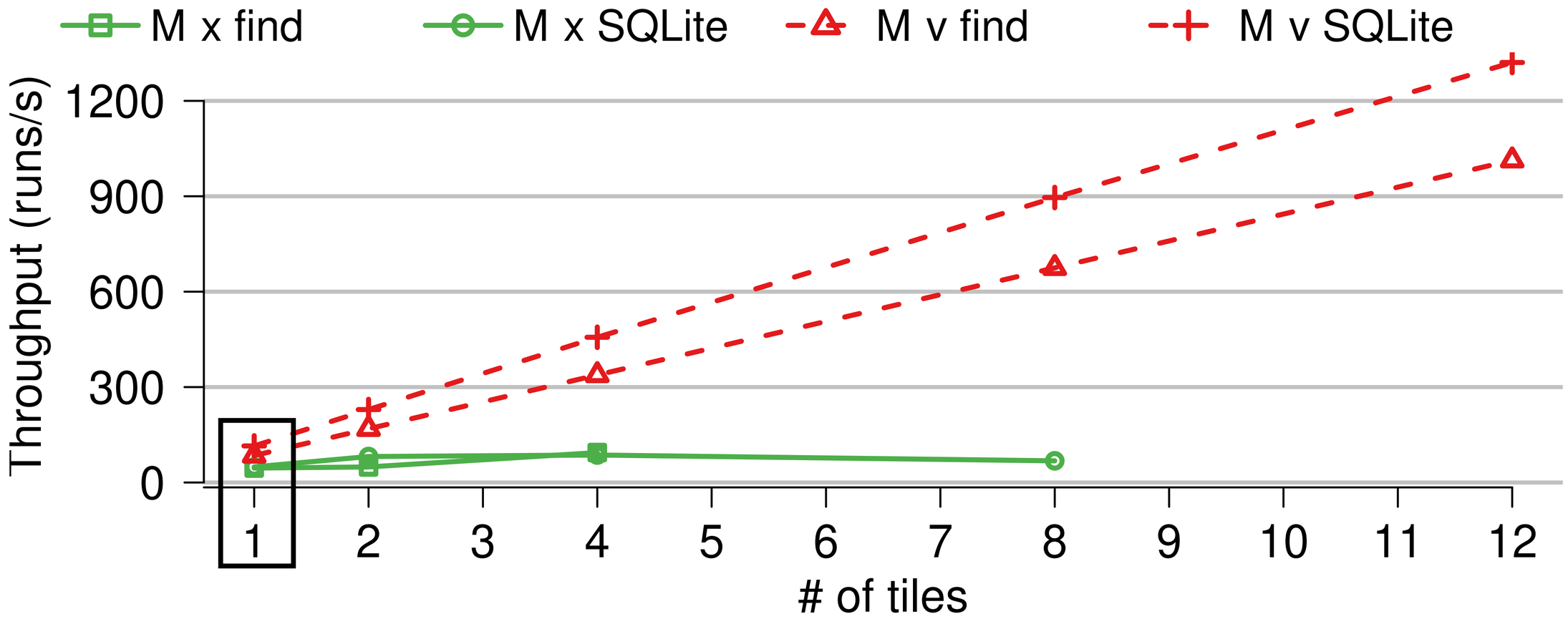
Performance Comparison to M³x

- Setup: gem5 simulator, 3 GHz out-of-order x86-64 cores
- Every tile runs: SQLite/find benchmark and in-memory filesystem



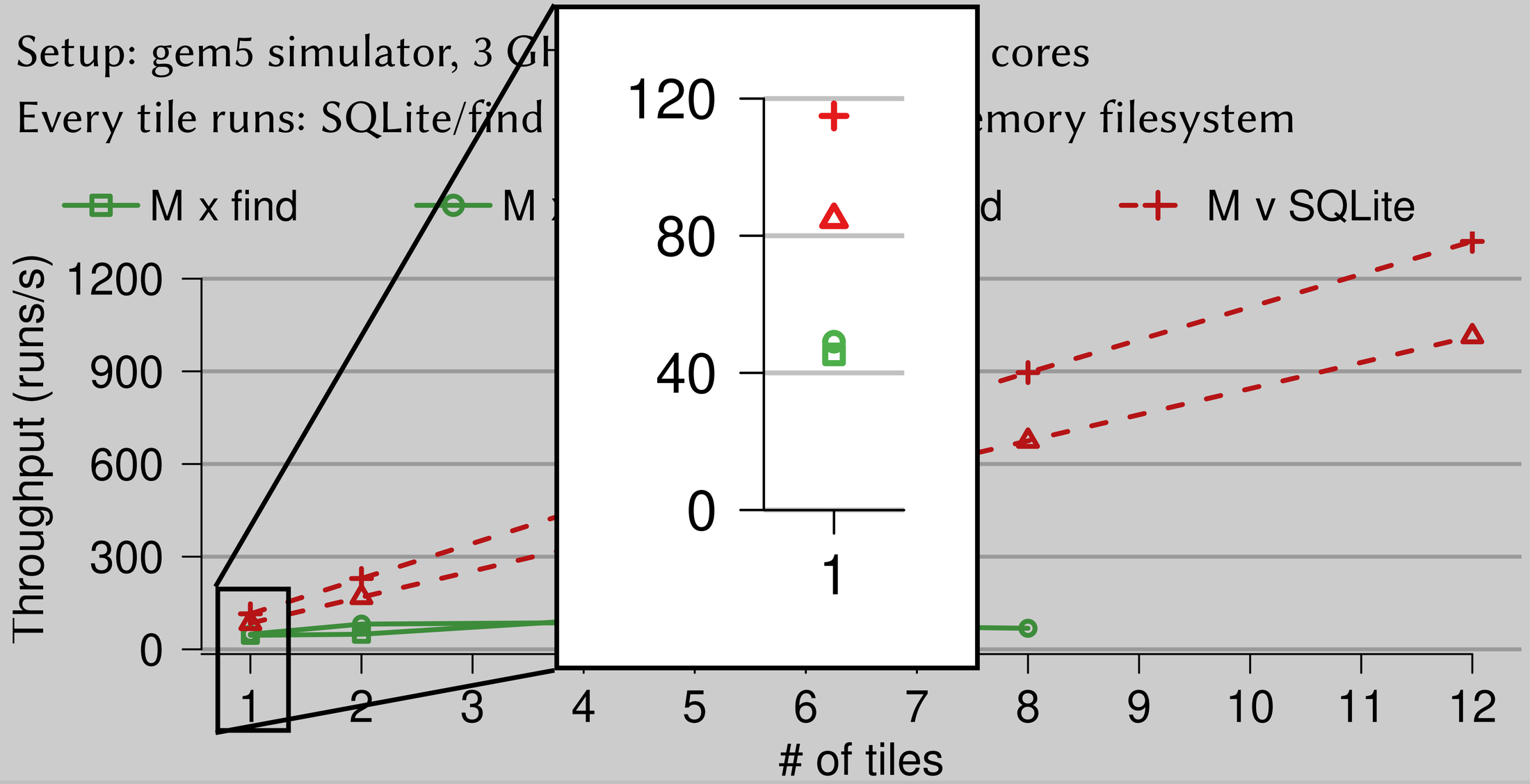
Performance Comparison to M³x

- Setup: gem5 simulator, 3 GHz out-of-order x86-64 cores
- Every tile runs: SQLite/find benchmark and in-memory filesystem

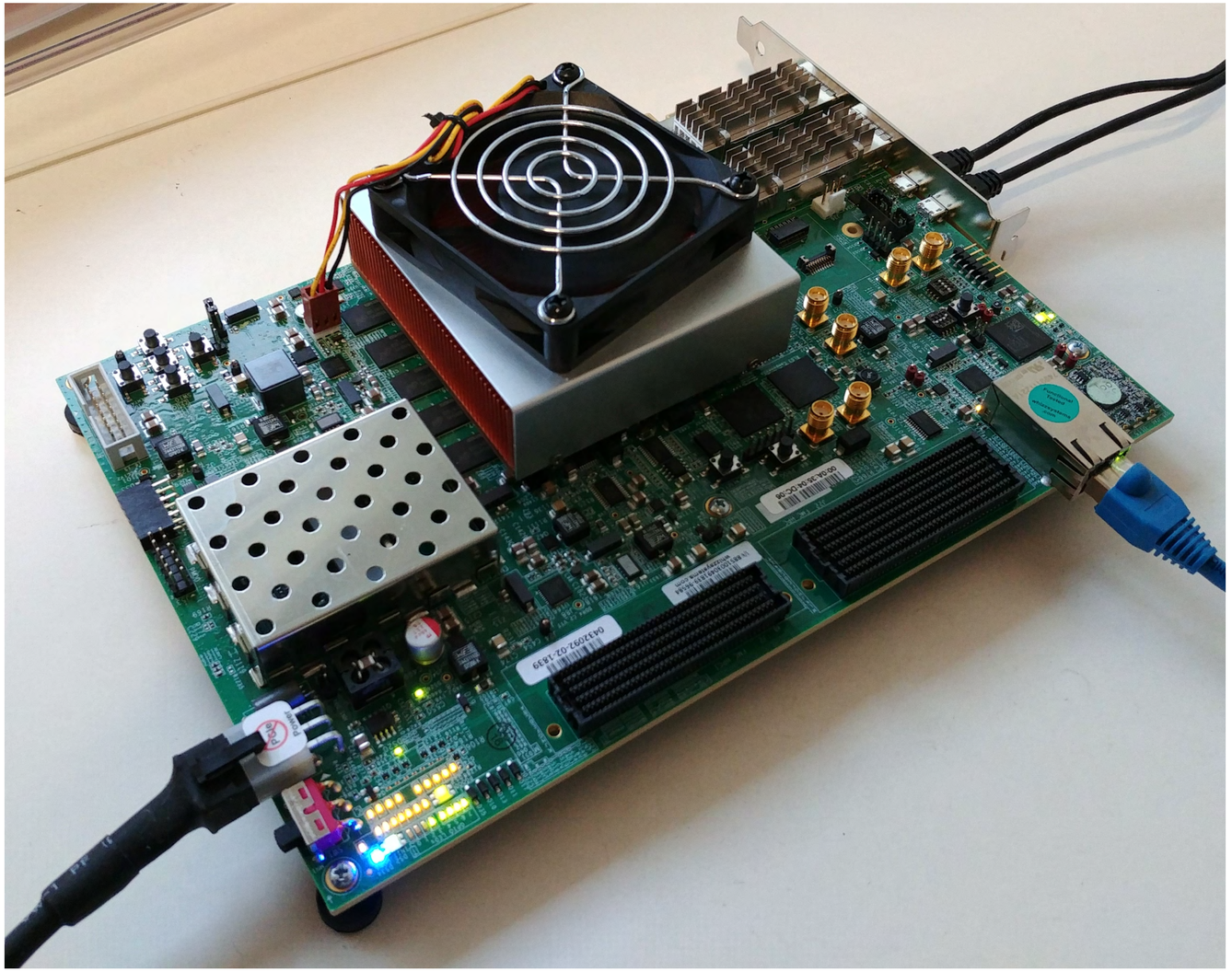


Performance Comparison to M³x

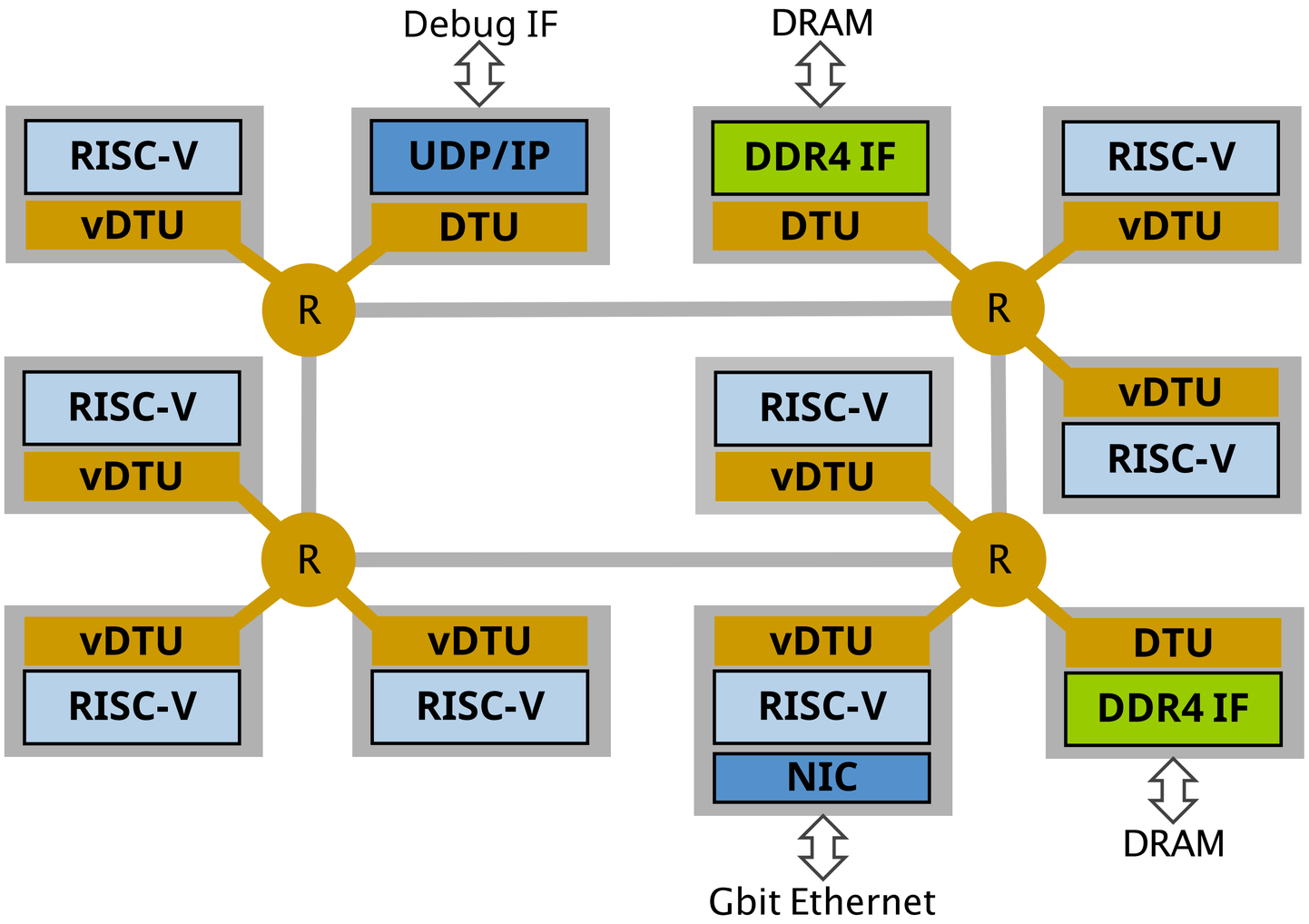
- Setup: gem5 simulator, 3 GH cores
- Every tile runs: SQLite/find emory filesystem



Hardware Implementation

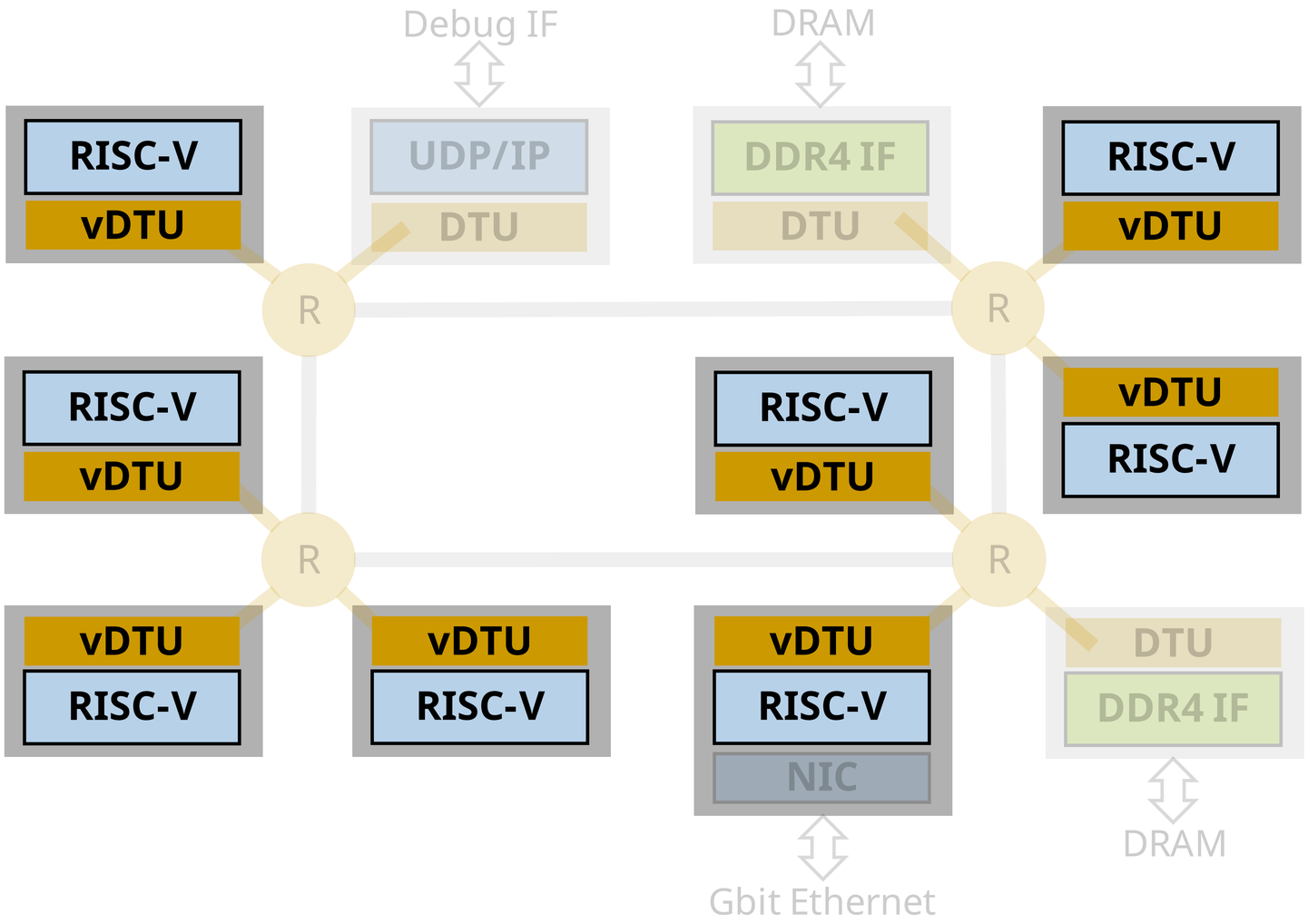


Hardware Implementation



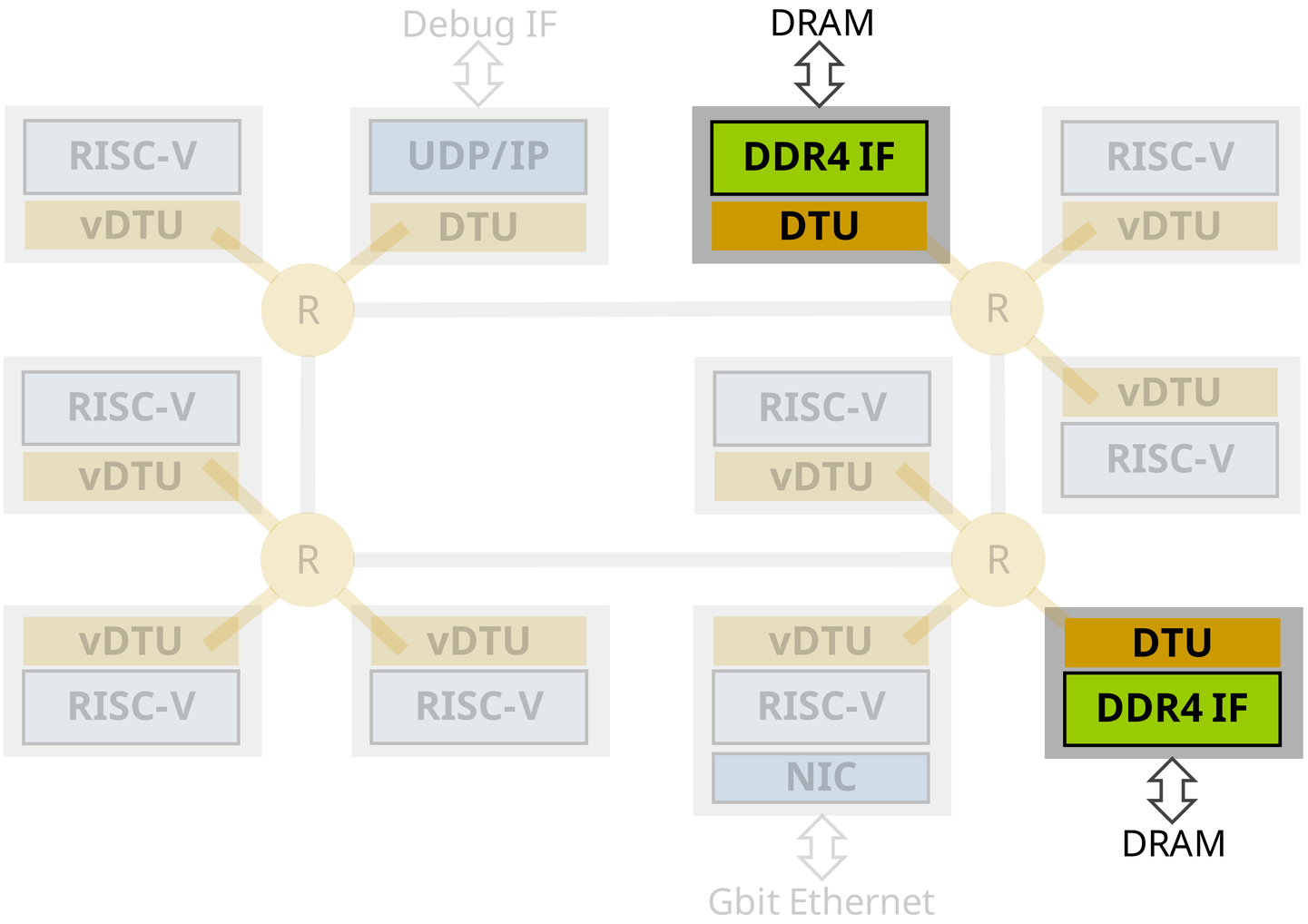
- Xilinx VCU118 FPGA
- RISC-V: in-order Rocket or out-of-order BOOM
- Rocket at 100 MHz, BOOM at 80 MHz
- 2x16 kB L1, 512 kB L2
- vDTU contains 128 EPs

Hardware Implementation



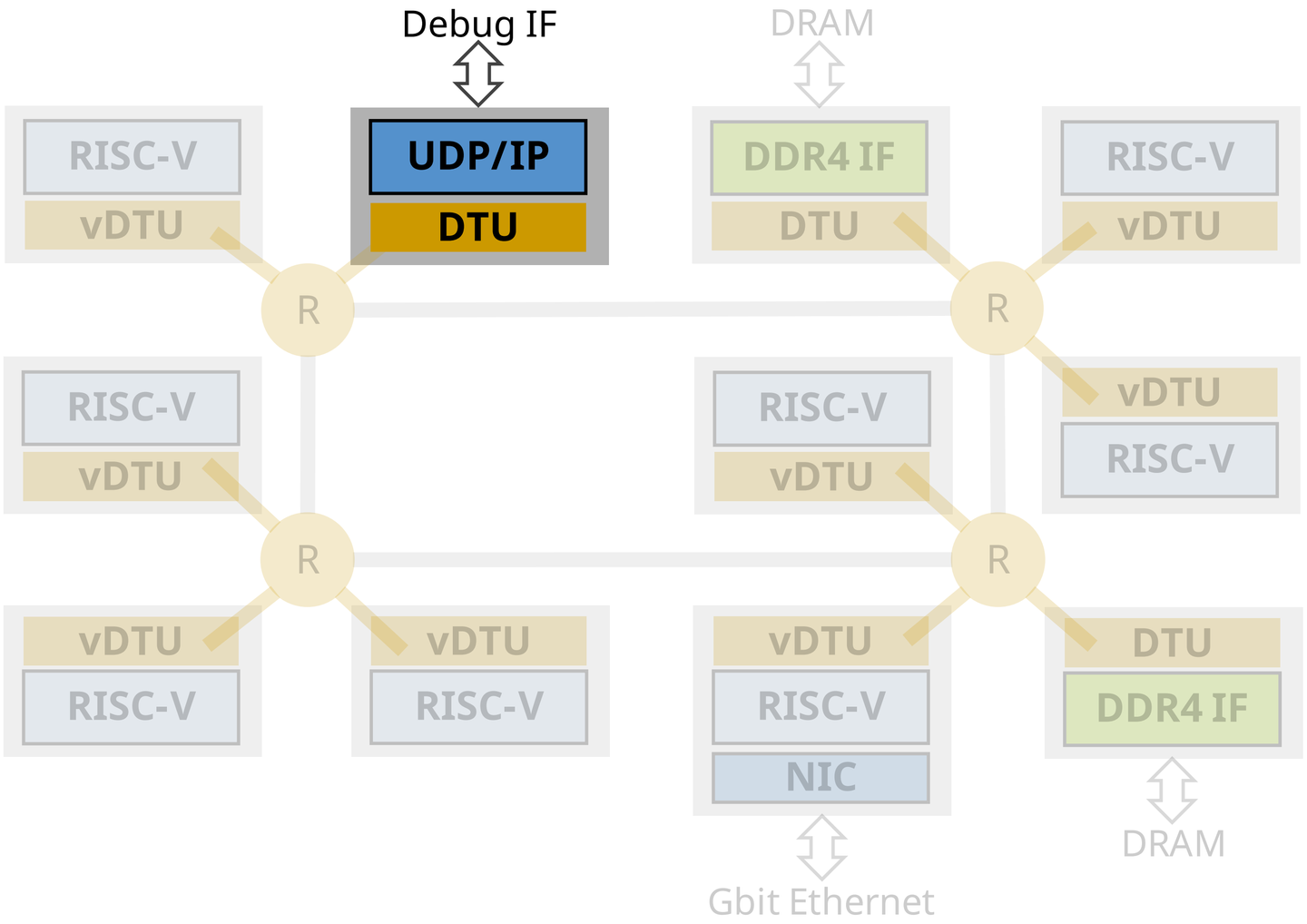
- Xilinx VCU118 FPGA
- RISC-V: in-order Rocket or out-of-order BOOM
- Rocket at 100 MHz, BOOM at 80 MHz
- 2x16 kB L1, 512 kB L2
- vDTU contains 128 EPs

Hardware Implementation



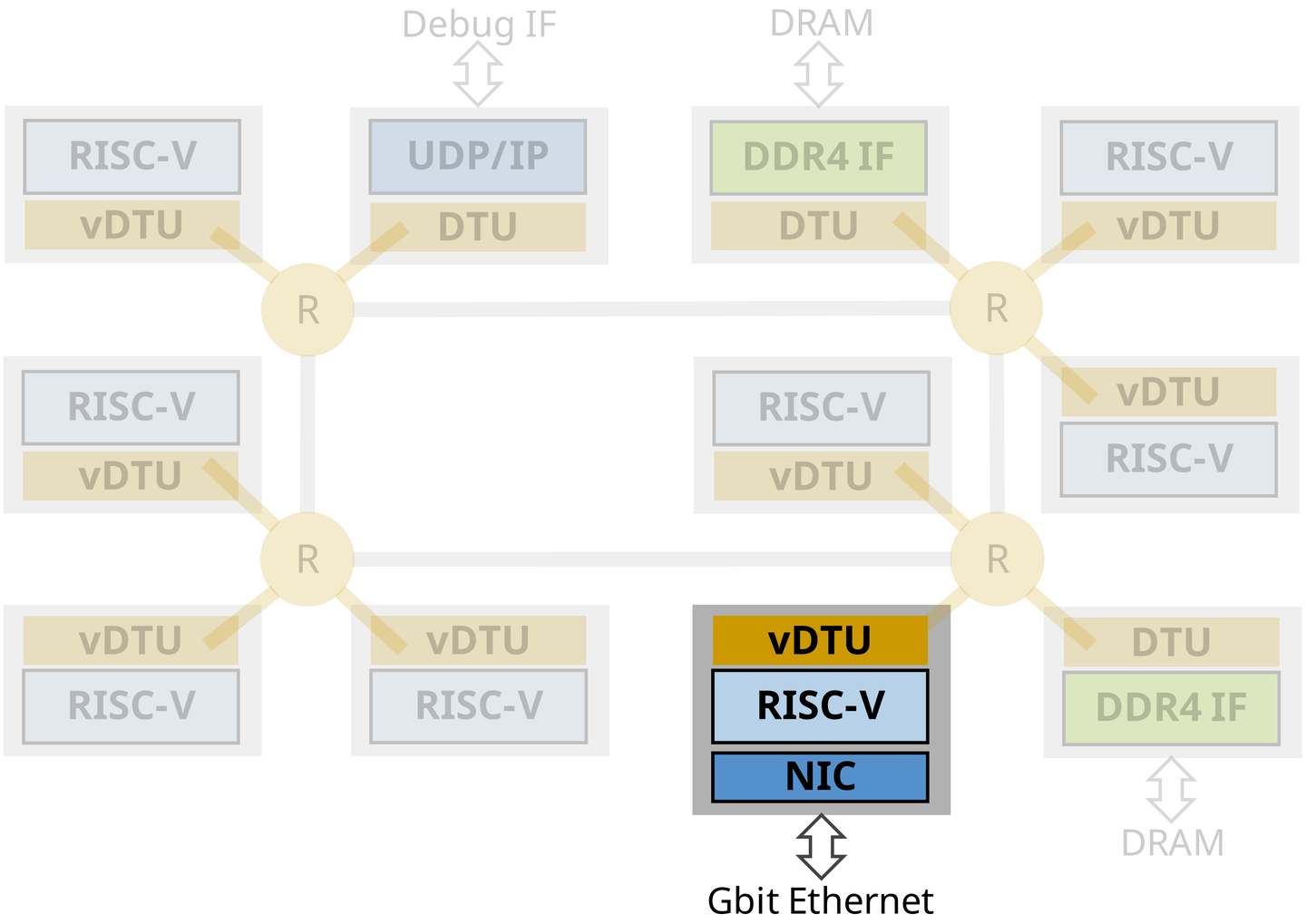
- Xilinx VCU118 FPGA
- RISC-V: in-order Rocket or out-of-order BOOM
- Rocket at 100 MHz, BOOM at 80 MHz
- 2x16 kB L1, 512 kB L2
- vDTU contains 128 EPs

Hardware Implementation



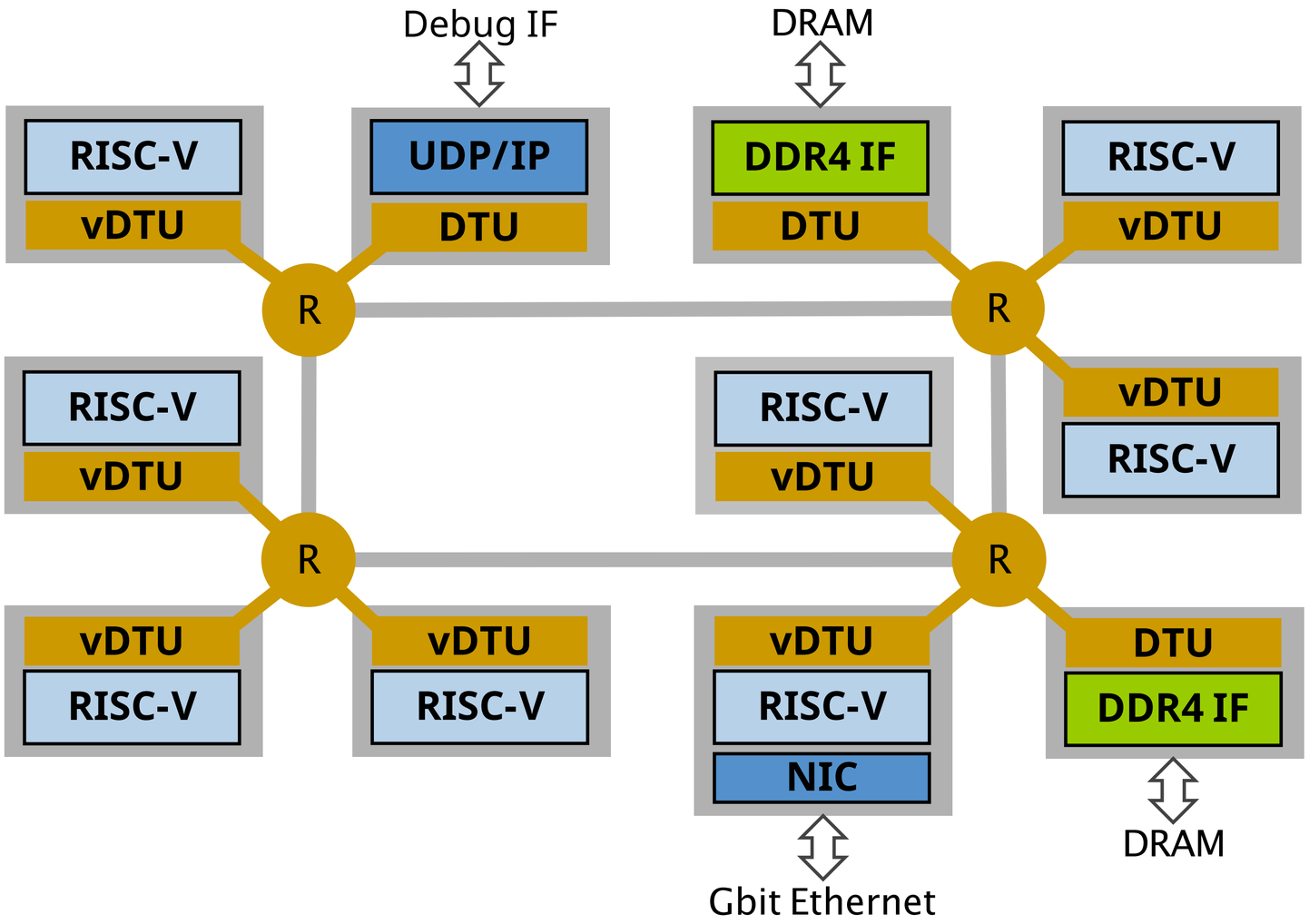
- Xilinx VCU118 FPGA
- RISC-V: in-order Rocket or out-of-order BOOM
- Rocket at 100 MHz, BOOM at 80 MHz
- 2x16 kB L1, 512 kB L2
- vDTU contains 128 EPs

Hardware Implementation



- Xilinx VCU118 FPGA
- RISC-V: in-order Rocket or out-of-order BOOM
- Rocket at 100 MHz, BOOM at 80 MHz
- 2x16 kB L1, 512 kB L2
- vDTU contains 128 EPs

Hardware Implementation



- Xilinx VCU118 FPGA
- RISC-V: in-order Rocket or out-of-order BOOM
- Rocket at 100 MHz, BOOM at 80 MHz
- 2x16 kB L1, 512 kB L2
- vDTU contains 128 EPs

DTU Size and Complexity

	LUTs [k]	FFs [k]	BRAMs
BOOM	143.8	71.8	159
Rocket	46.6	22.0	152
NoC router	3.4	2.2	0
vDTU	15.2	5.8	0.5
Control Unit	10.3	3.3	0.5
NoC CTRL	3.2	1.5	0
CMD CTRL	7.1	2.8	0.5
Unpriv. IF	6.2	2.5	0.5
Priv. IF	0.9	0.3	0
Register file	2.0	1.0	0
Memory mapper + PMP	0.6	0.2	0
I/O FIFOs	2.3	0.3	0

DTU Size and Complexity

	LUTs [k]	FFs [k]	BRAMs
BOOM	143.8	71.8	159
Rocket	46.6	22.0	152
NoC router	3.4	2.2	0
vDTU	15.2	5.8	0.5
Control Unit	10.3	3.3	0.5
NoC CTRL	3.2	1.5	0
CMD CTRL	7.1	2.8	0.5
Unpriv. IF	6.2	2.5	0.5
Priv. IF	0.9	0.3	0
Register file	2.0	1.0	0
Memory mapper + PMP	0.6	0.2	0
I/O FIFOs	2.3	0.3	0

DTU Size and Complexity

	LUTs [k]	FFs [k]	BRAMs
BOOM	143.8	71.8	159
Rocket	46.6	22.0	152
NoC router	3.4	2.2	0
vDTU	15.2	5.8	0.5
Control Unit	10.3	3.3	0.5
NoC CTRL	3.2	1.5	0
CMD CTRL	7.1	2.8	0.5
Unpriv. IF	6.2	2.5	0.5
Priv. IF	0.9	0.3	0
Register file	2.0	1.0	0
Memory mapper + PMP	0.6	0.2	0
I/O FIFOs	2.3	0.3	0

DTU Size and Complexity

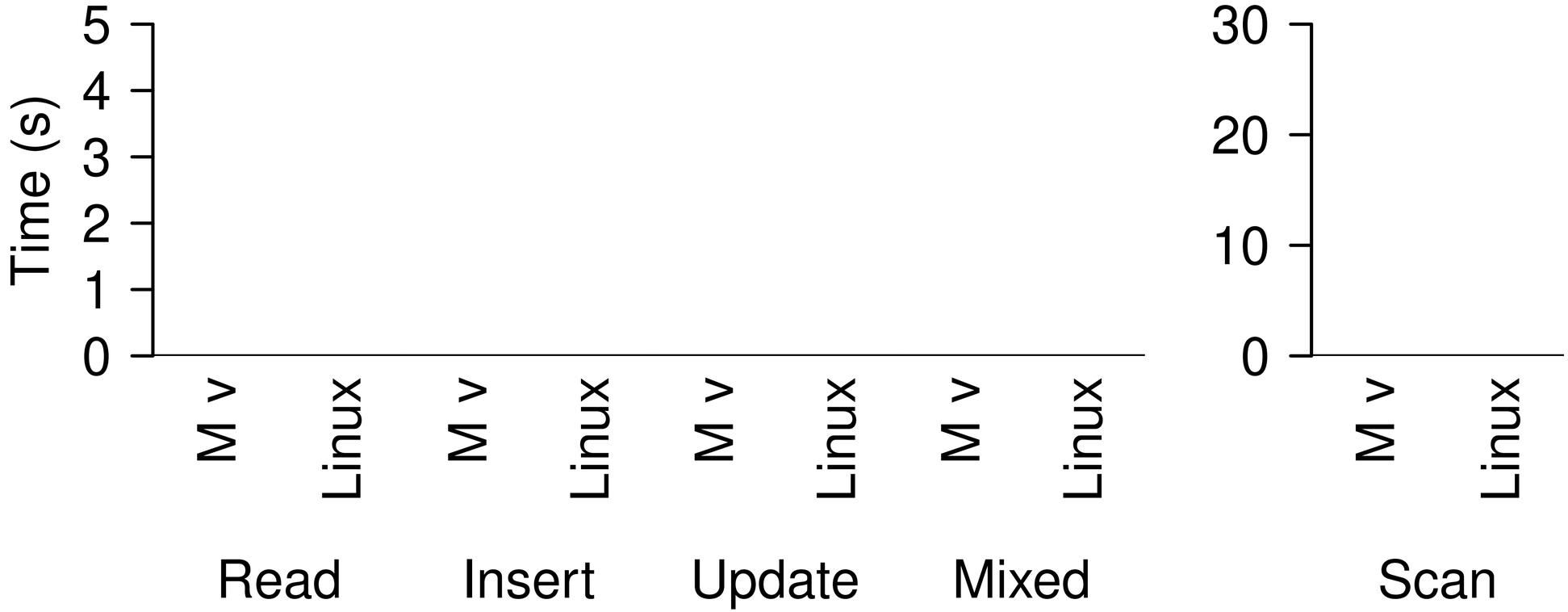
	LUTs [k]	FFs [k]	BRAMs
BOOM	143.8	71.8	159
Rocket	46.6	22.0	152
NoC router	3.4	2.2	0
vDTU	15.2	5.8	0.5
Control Unit	10.3	3.3	0.5
NoC CTRL	3.2	1.5	0
CMD CTRL	7.1	2.8	0.5
Unpriv. IF	6.2	2.5	0.5
Priv. IF	0.9	0.3	0
Register file	2.0	1.0	0
Memory mapper + PMP	0.6	0.2	0
I/O FIFOs	2.3	0.3	0

Performance Comparison with Linux

- LevelDB receives requests from remote machine, sends results back
- requests generated by YCSB, different shares of read/insert/update/scan
- single BOOM core runs: LevelDB, pager, file system, network stack

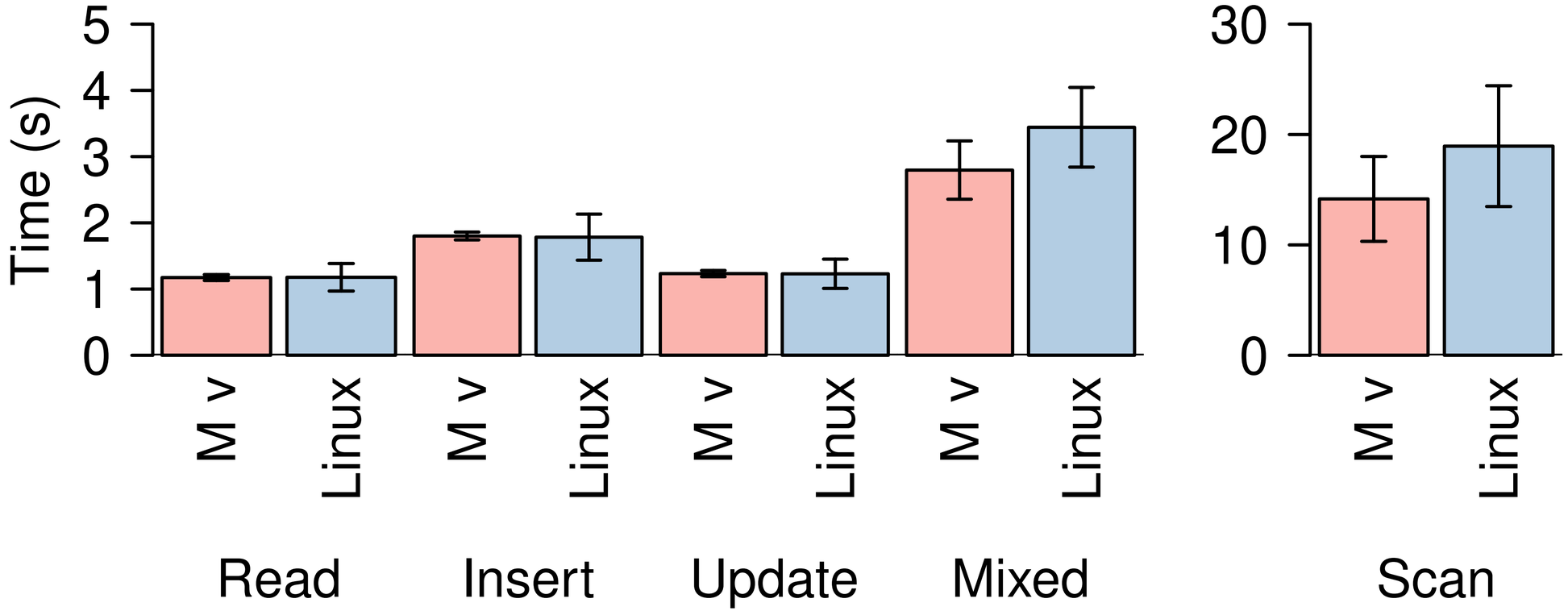
Performance Comparison with Linux

- LevelDB receives requests from remote machine, sends results back
- requests generated by YCSB, different shares of read/insert/update/scan
- single BOOM core runs: LevelDB, pager, file system, network stack



Performance Comparison with Linux

- LevelDB receives requests from remote machine, sends results back
- requests generated by YCSB, different shares of read/insert/update/scan
- single BOOM core runs: LevelDB, pager, file system, network stack



Conclusion

Conclusion

- **M³** explores a system architecture with a new per-tile hardware component

Conclusion

- **M³** explores a system architecture with a new per-tile hardware component
- **M³x** adds autonomous accelerator chains without core orchestration

Conclusion

- **M³** explores a system architecture with a new per-tile hardware component
- **M³x** adds autonomous accelerator chains without core orchestration
- **M³v** shows how general-purpose cores can be multiplexed efficiently

Conclusion

- **M³** explores a system architecture with a new per-tile hardware component
- **M³x** adds autonomous accelerator chains without core orchestration
- **M³v** shows how general-purpose cores can be multiplexed efficiently
- hardware implementation demonstrates modest additional cost

Conclusion

- **M³** explores a system architecture with a new per-tile hardware component
- **M³x** adds autonomous accelerator chains without core orchestration
- **M³v** shows how general-purpose cores can be multiplexed efficiently
- hardware implementation demonstrates modest additional cost
- tape-out in custom silicon underway

Conclusion

- **M³** explores a system architecture with a new per-tile hardware component
- **M³x** adds autonomous accelerator chains without core orchestration
- **M³v** shows how general-purpose cores can be multiplexed efficiently
- hardware implementation demonstrates modest additional cost
- tape-out in custom silicon underway
- competitive performance to Linux with context-switch-heavy workloads

Conclusion

- **M³** explores a system architecture with a new per-tile hardware component
- **M³x** adds autonomous accelerator chains without core orchestration
- **M³v** shows how general-purpose cores can be multiplexed efficiently
- hardware implementation demonstrates modest additional cost
- tape-out in custom silicon underway
- competitive performance to Linux with context-switch-heavy workloads
- complete hardware/software stack available as open source:

Conclusion

- **M³** explores a system architecture with a new per-tile hardware component
- **M³x** adds autonomous accelerator chains without core orchestration
- **M³v** shows how general-purpose cores can be multiplexed efficiently
- hardware implementation demonstrates modest additional cost
- tape-out in custom silicon underway
- competitive performance to Linux with context-switch-heavy workloads
- complete hardware/software stack available as open source:

https://github.com/Barkhausen-Institut/M3

Lessons Learned

Lessons Learned

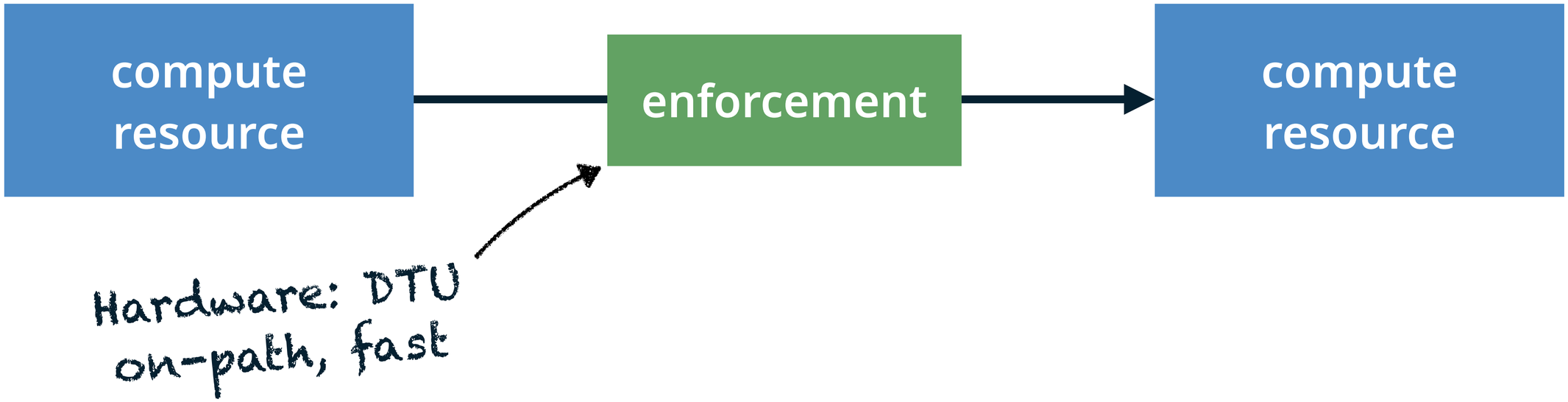
compute resource

compute resource

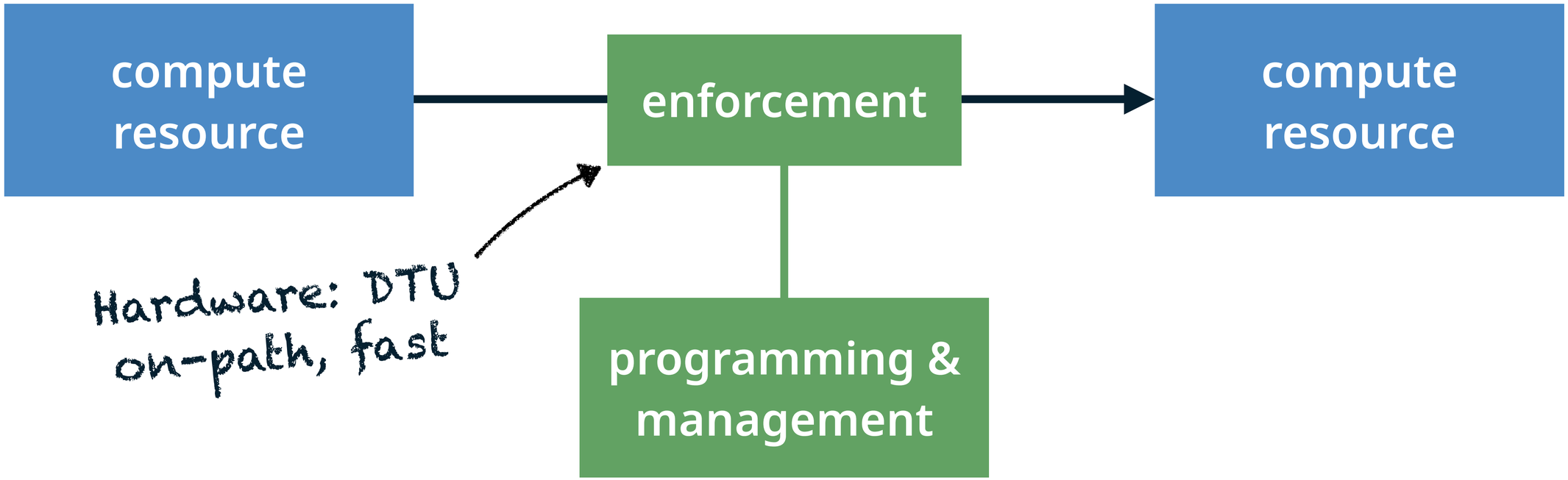
Lessons Learned

Lessons Learned

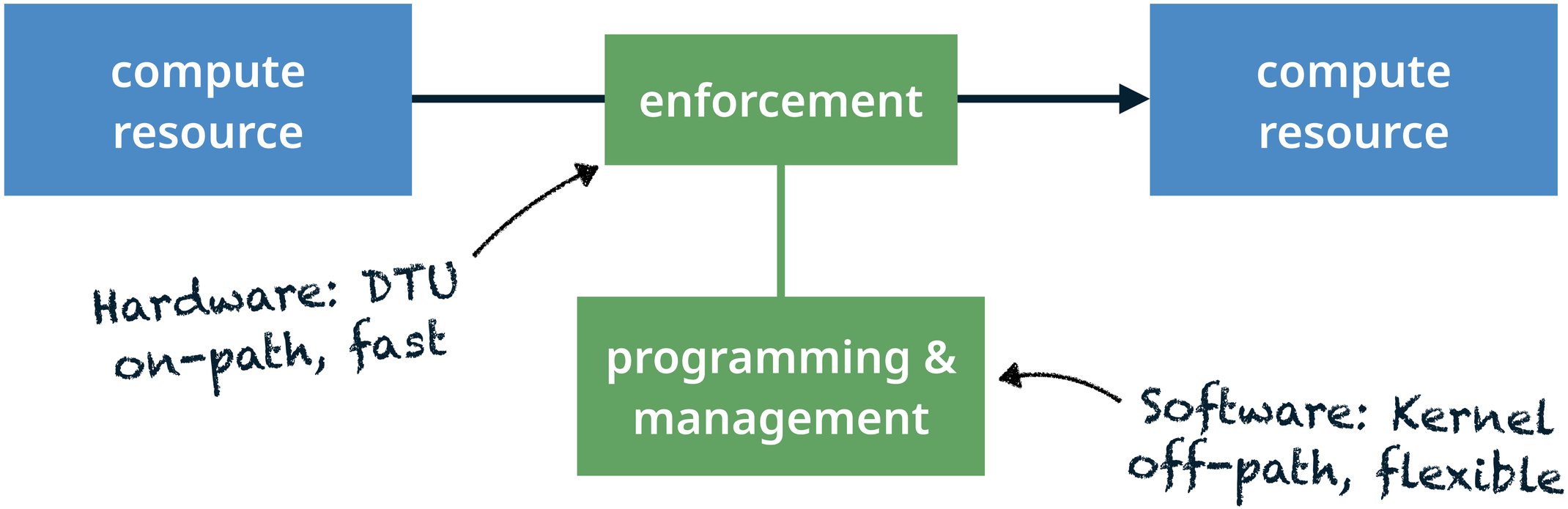
Lessons Learned



Lessons Learned



Lessons Learned



Lessons Learned