TECHNISCHE UNIVERSITÄT DRESDEN

barkhausen institut

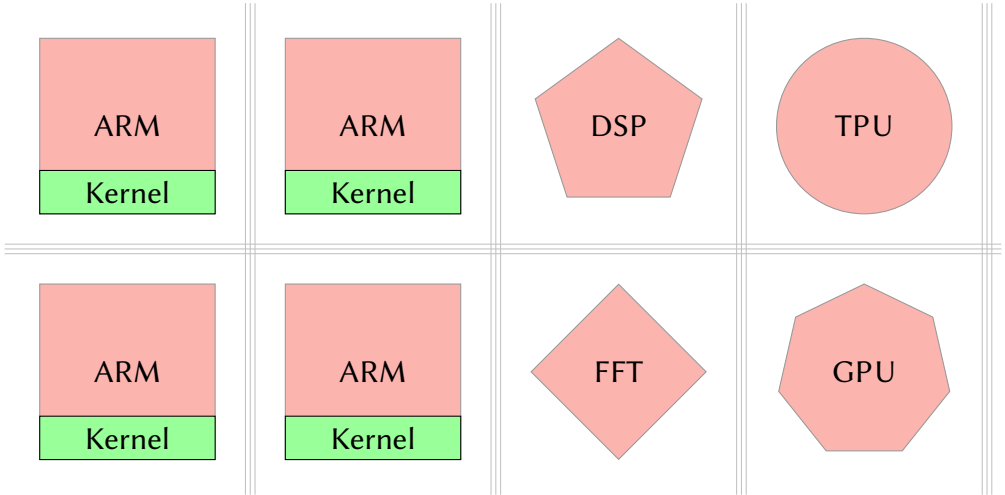
M³X: Autonomous Accelerators via Context-Enabled Fast-Path Communication

Nils Asmussen[†] [‡] Michael Roitzsch[†] [‡] Hermann Härtig[†] [‡]

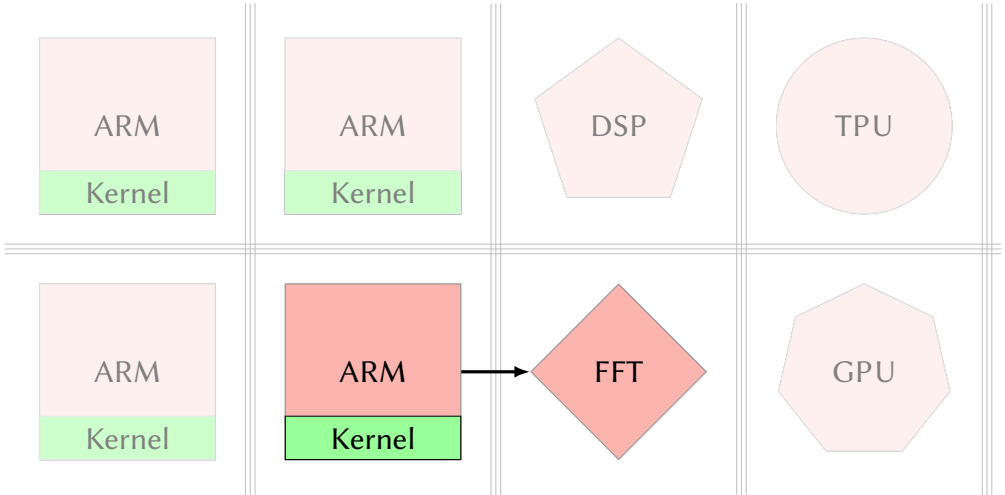
[†]Technische Universität Dresden [‡] Barkhausen Institut, Dresden

Renton, July 11th, USENIX ATC'19

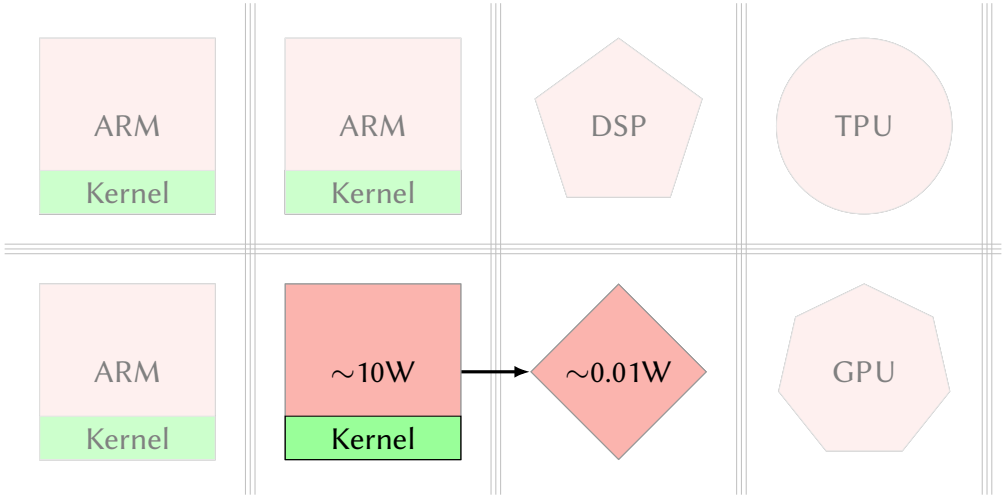
M³x: Autonomous Accelerators via Context-Enabled Fast-Path Communication



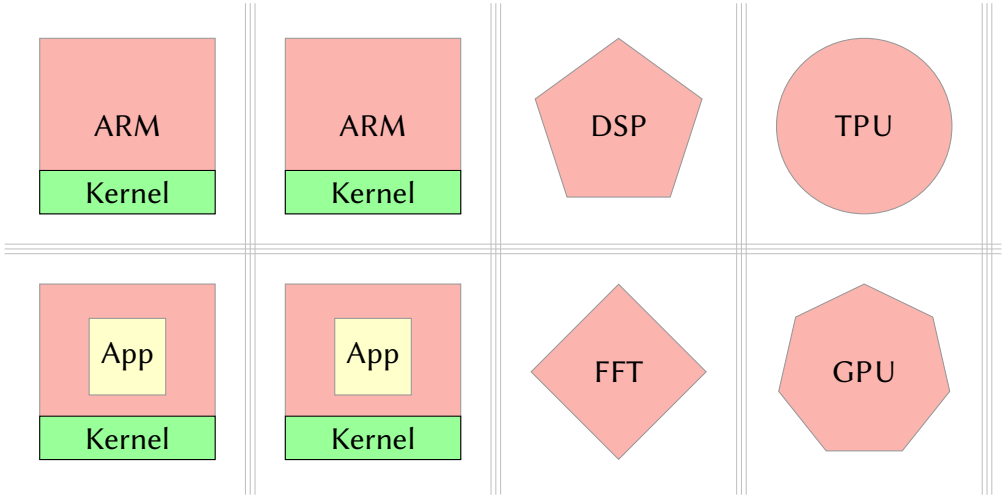
M³x: Autonomous Accelerators via Context-Enabled Fast-Path Communication



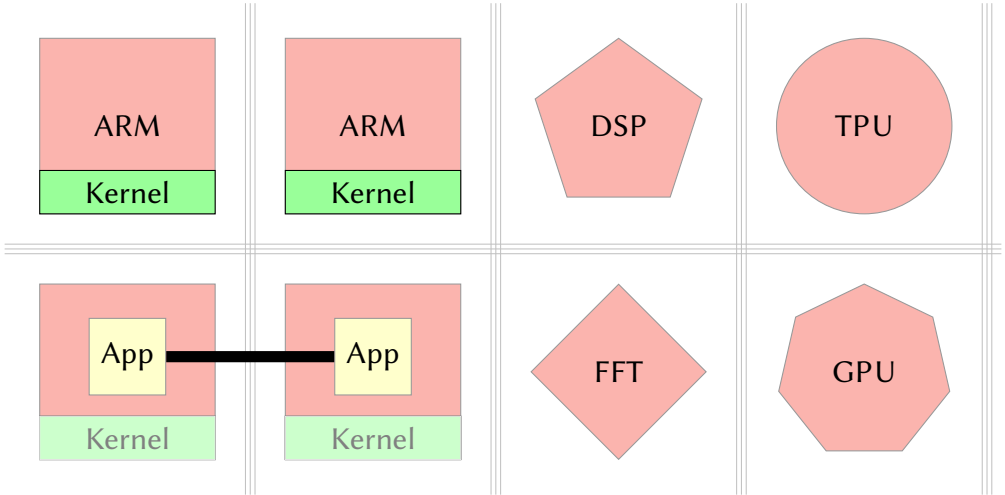
M³x: Autonomous Accelerators via Context-Enabled Fast-Path Communication



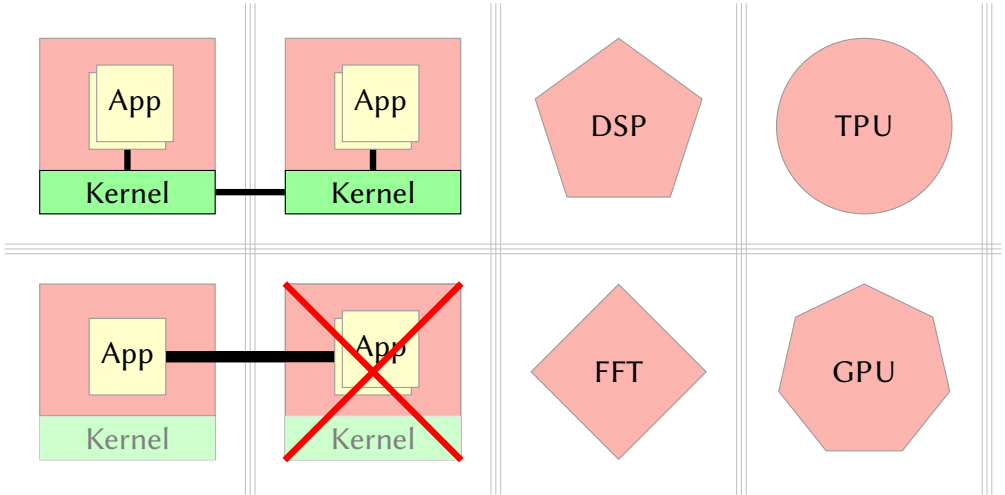
M³x: Autonomous Accelerators via Context-Enabled Fast-Path Communication



M³x: Autonomous Accelerators via Context-Enabled Fast-Path Communication



M³x: Autonomous Accelerators via Context-Enabled Fast-Path Communication



Our Work

- Rethink system architecture based on M^3
  - Hardware/operating-system co-design for heterogeneous systems
  - Simulation based on gem5
- Not built upon cache coherency
  - Costs (area, power, complexity, performance) increases with system size
  - More challenging for heterogeneous systems
  - Unclear whether future systems will be (globally) coherent
- Focus on fixed-function accelerators
  - Most difficult to support as “first-class citizens”
  - Provide none of the features OSes need

Related Work

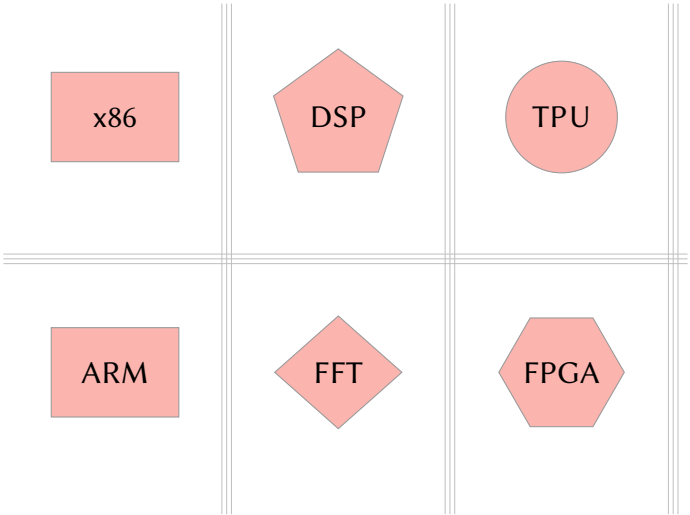
One specific accelerator

- GPUfs, GPUnet, Chimera
- ReconOS, BORPH

OSes for heterogeneous systems

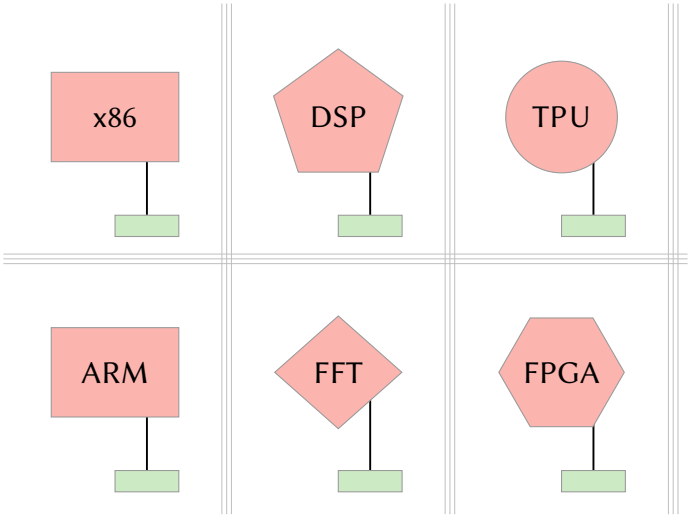
- Barrelfish
- Popcorn Linux, K2
- Helios

Approach



Key techniques:

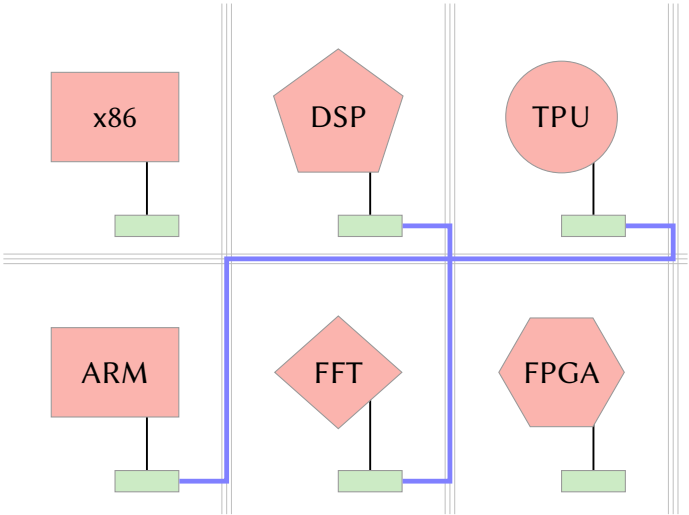
Approach



Key techniques:

- Add uniform interface

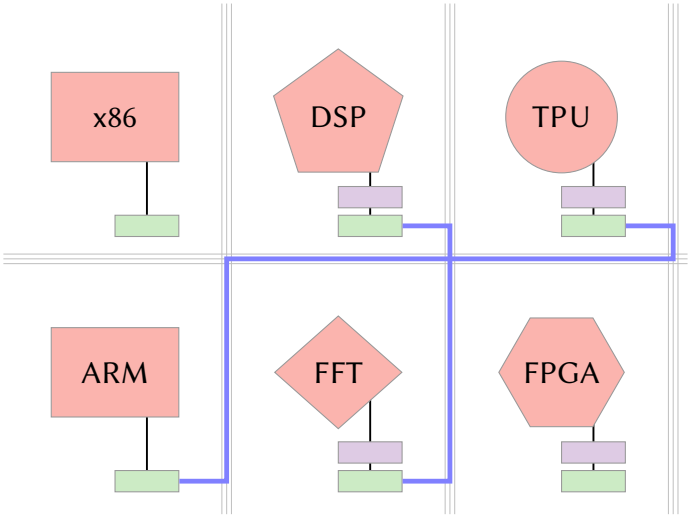
Approach



Key techniques:

- Add uniform interface
- Simple+generic protocols

Approach



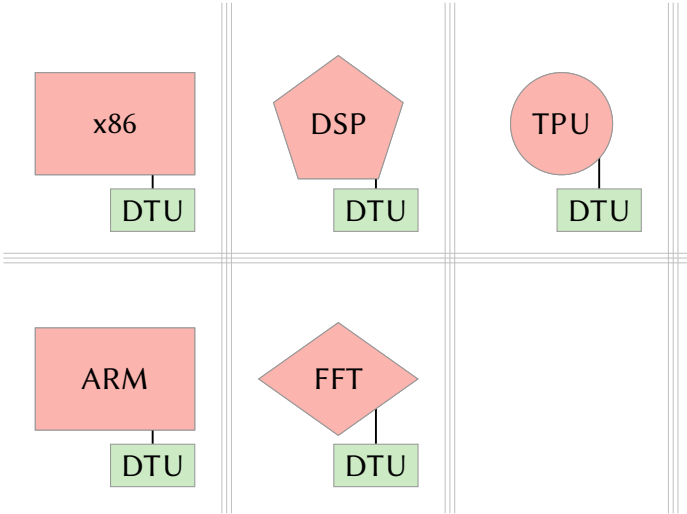
Key techniques:

- Add uniform interface
- Simple+generic protocols
- Don’t expect accelerators to run an OS
- Add lightweight component externally

Outline

Outline

M³ System Architecture

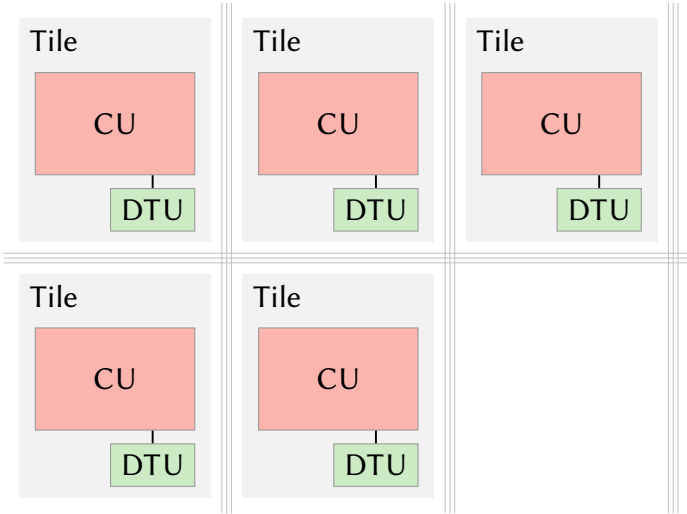


Key Ideas of M³ [1]:

- DTU as uniform interface

[1] M³: A Hardware/Operating-System Co-Design to Tame Heterogeneous Manycores, ASPLOS 2016

M³ System Architecture

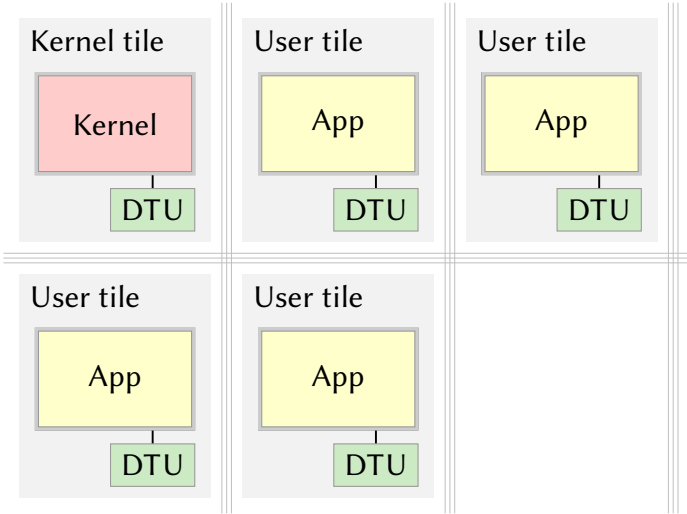


Key Ideas of M³ [1]:

- DTU as uniform interface

[1] M³: A Hardware/Operating-System Co-Design to Tame Heterogeneous Manycores, ASPLOS 2016

M³ System Architecture

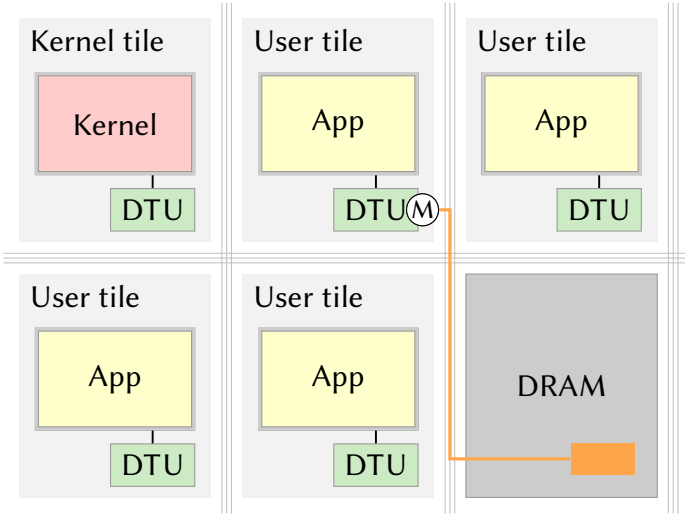


Key Ideas of M³ [1]:

- DTU as uniform interface
- Kernel controls user tiles remotely

[1] M³: A Hardware/Operating-System Co-Design to Tame Heterogeneous Manycores, ASPLOS 2016

M³ System Architecture: Fast-Path Communication

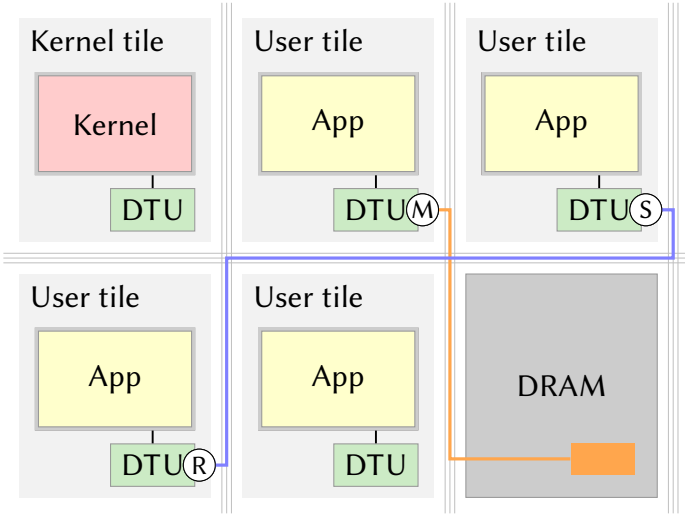


DTU provides *endpoints* to:

- Access memory (contiguous range, byte granular)

[1] M³: A Hardware/Operating-System Co-Design to Tame Heterogeneous Manycores, ASPLOS 2016

M³ System Architecture: Fast-Path Communication



DTU provides *endpoints* to:

- Access memory (contiguous range, byte granular)
- Receive messages into a receive buffer
- Send messages to a receiving endpoint

[1] M³: A Hardware/Operating-System Co-Design to Tame Heterogeneous Manycores, ASPLOS 2016

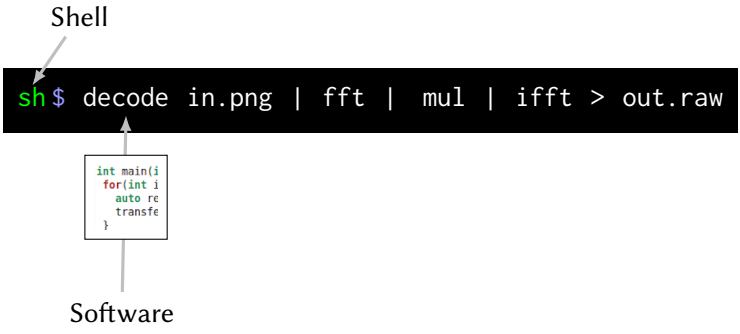
Outline

OS Service Access for all CUs

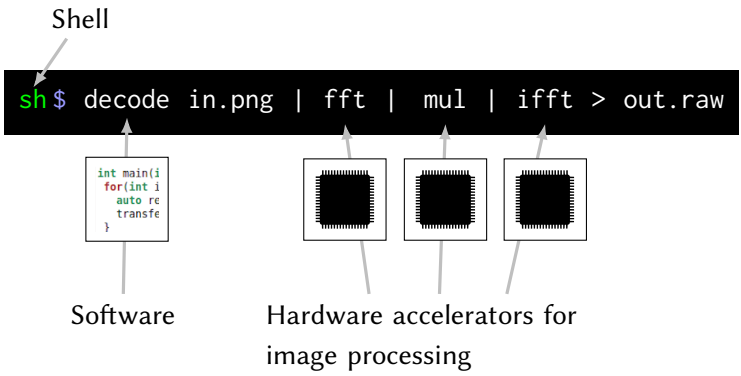
```
sh $ decode in.png | fft | mul | ifft > out.raw
```

OS Service Access for all CUs

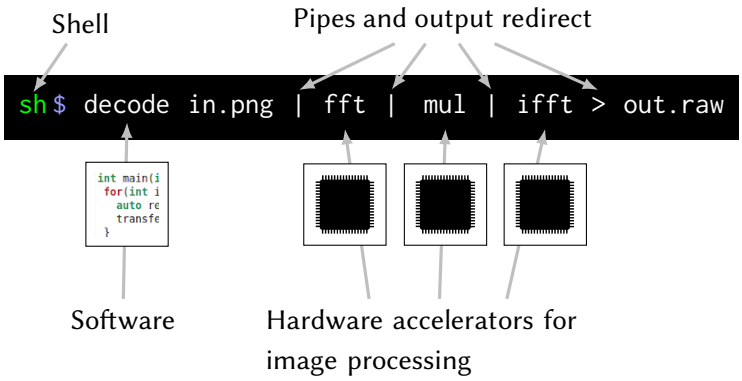
OS Service Access for all CUs



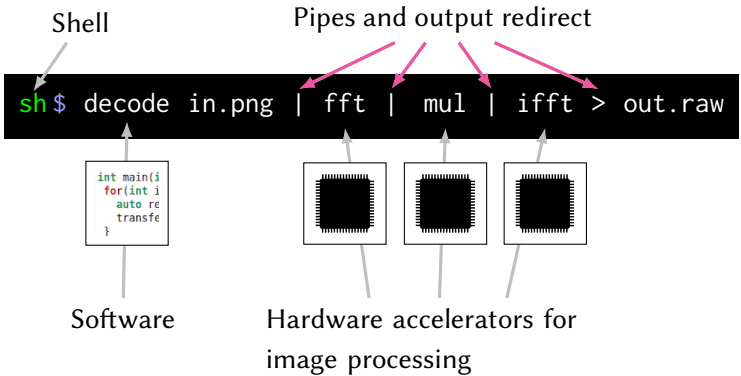
OS Service Access for all CUs



OS Service Access for all CUs



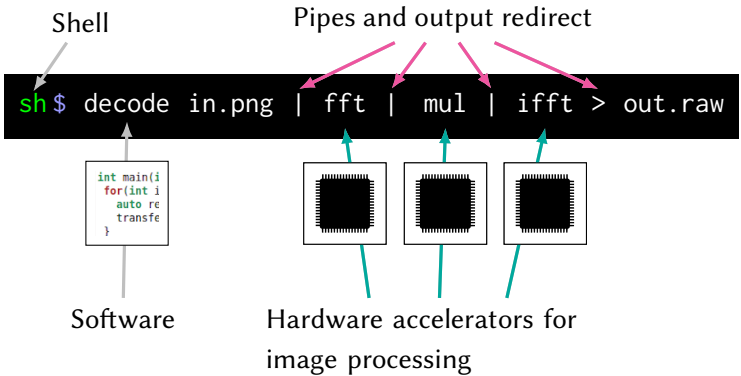
OS Service Access for all CUs



Challenges:

- OS must provide generic protocols

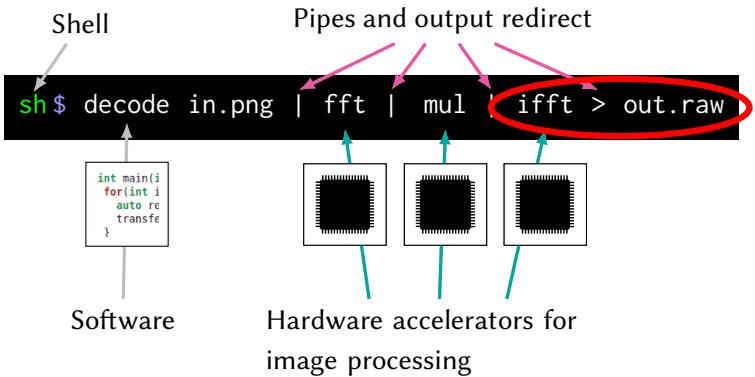
OS Service Access for all CUs



Challenges:

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

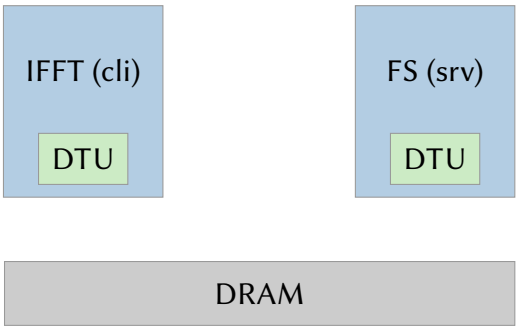
OS Service Access for all CUs



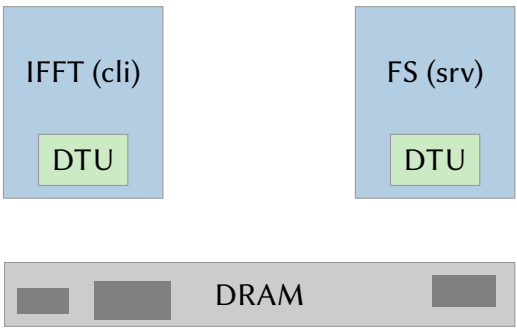
Challenges:

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

Generic Protocol



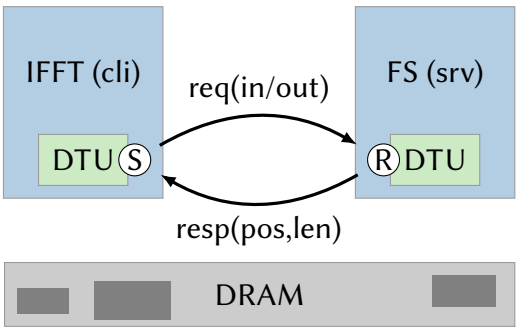
Generic Protocol



File protocol:

- Data in memory

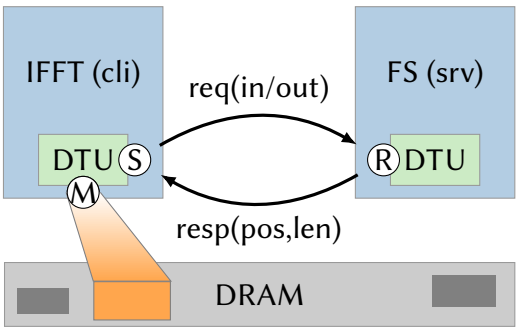
Generic Protocol



File protocol:

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

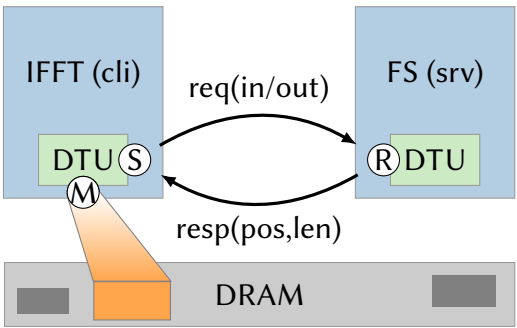
Generic Protocol



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

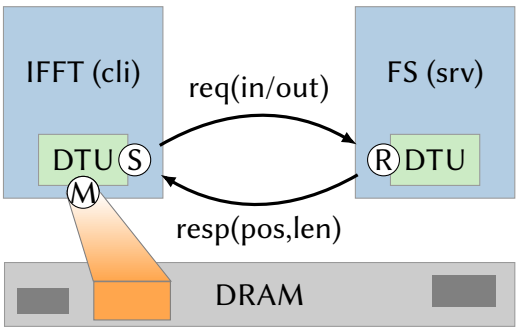
Generic Protocol



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

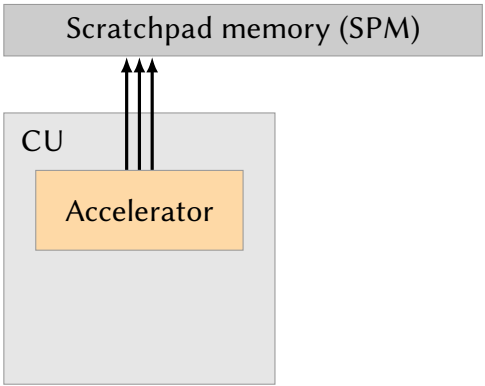
Generic Protocol



File protocol:

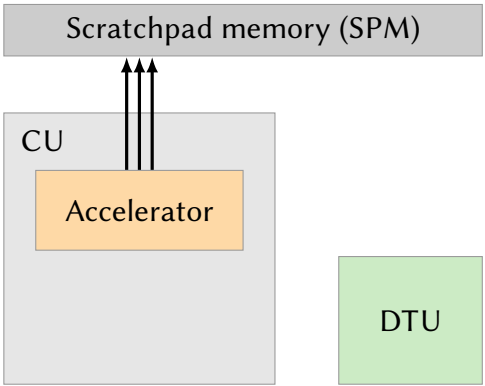
- Data in memory
- Msg channel between client and server
  - `req(in)` for next input piece
  - `req(out)` for next output piece
- Server configures client's memory EP
- Client accesses data via DTU
- Used by *all* CUs

Additions to Accelerator



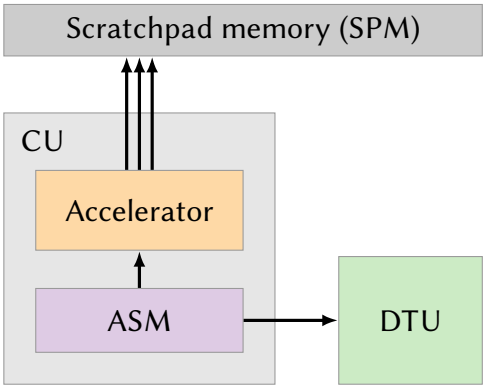
Off-the-shelf accelerators

Additions to Accelerator



Off-the-shelf accelerators

Additions to Accelerator

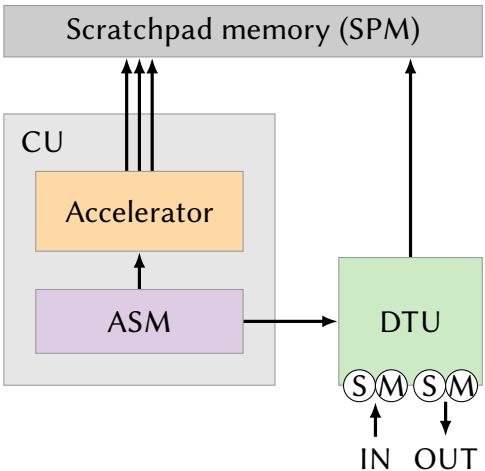


Off-the-shelf accelerators

Accelerator Support Module (ASM):

- Interacts with DTU and accelerator

Additions to Accelerator

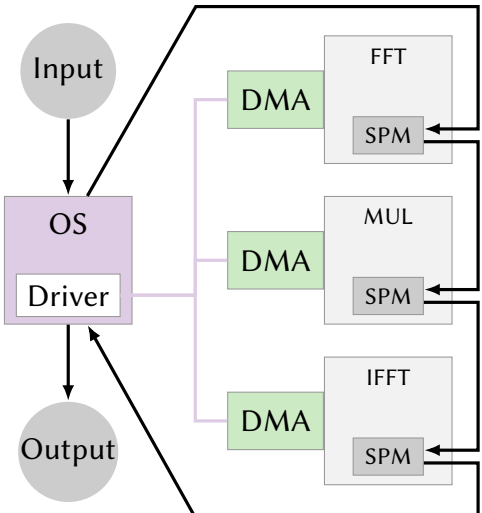


Off-the-shelf accelerators

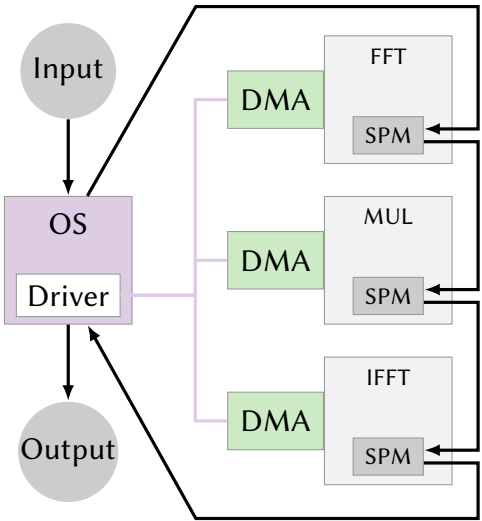
Accelerator Support Module (ASM):

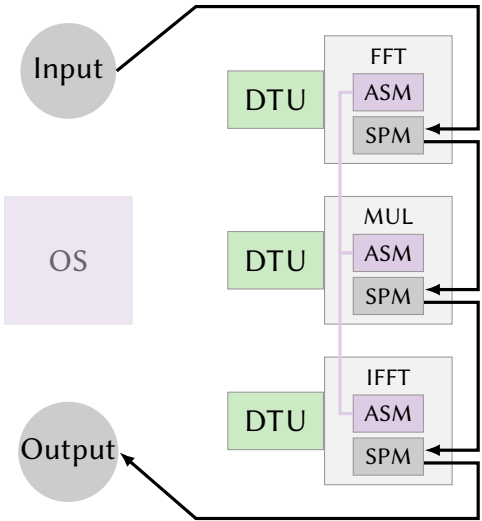
- Interacts with DTU and accelerator
- Implements file protocol for input and output channel

Assisted vs. Autonomous

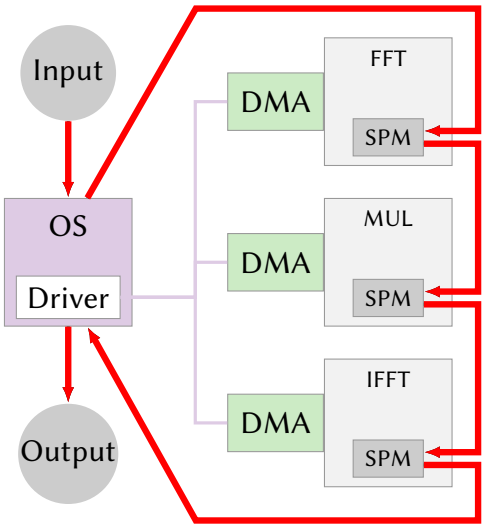


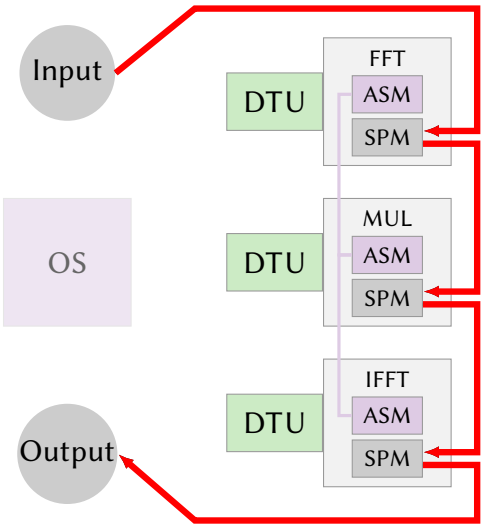
Assisted vs. Autonomous



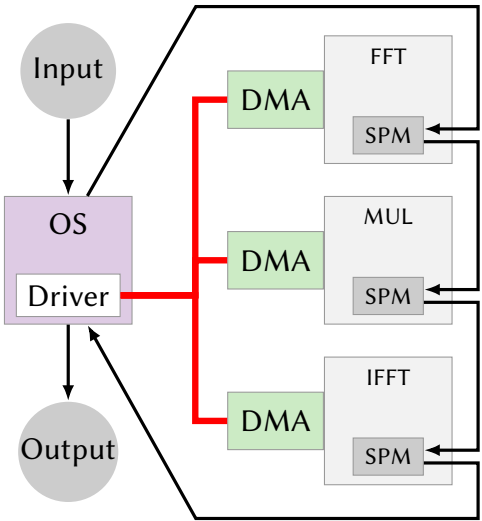


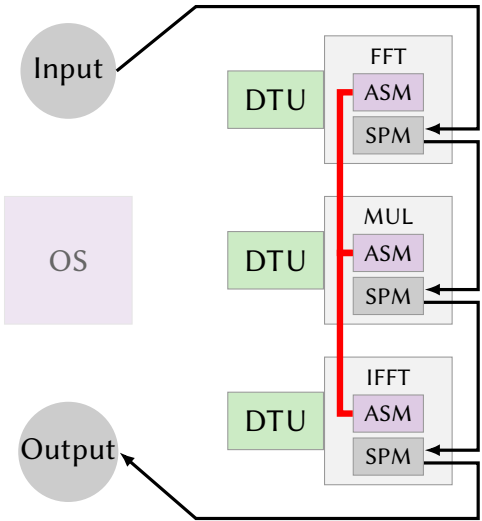
Assisted vs. Autonomous





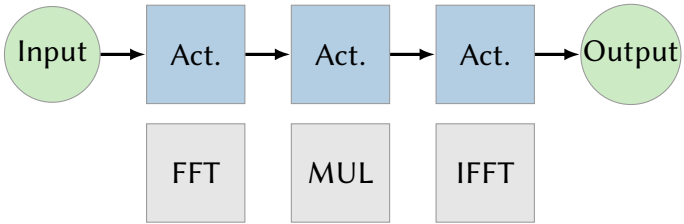
Assisted vs. Autonomous



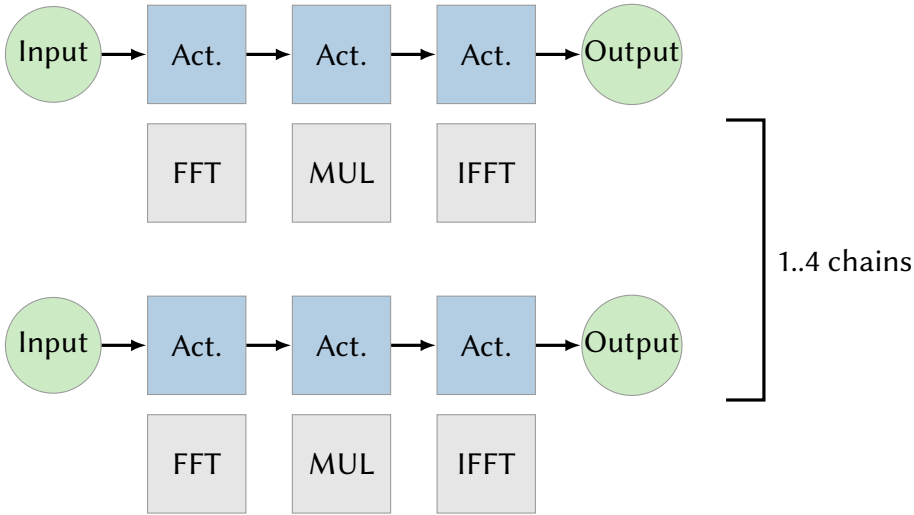


Accelerator Chains: Evaluation

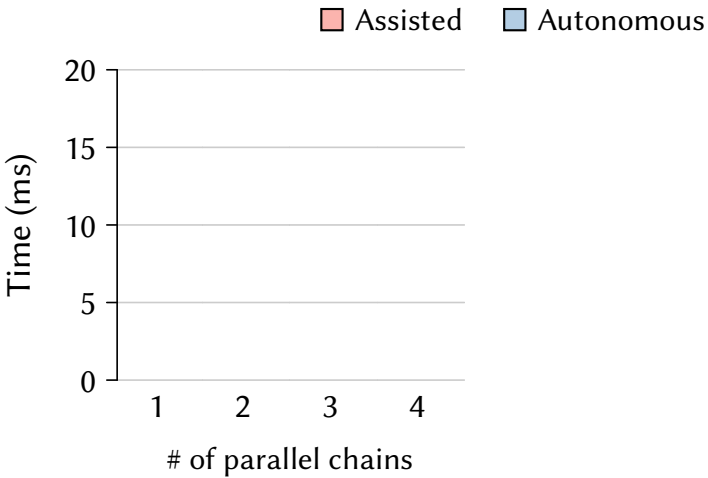
Accelerator Chains: Evaluation



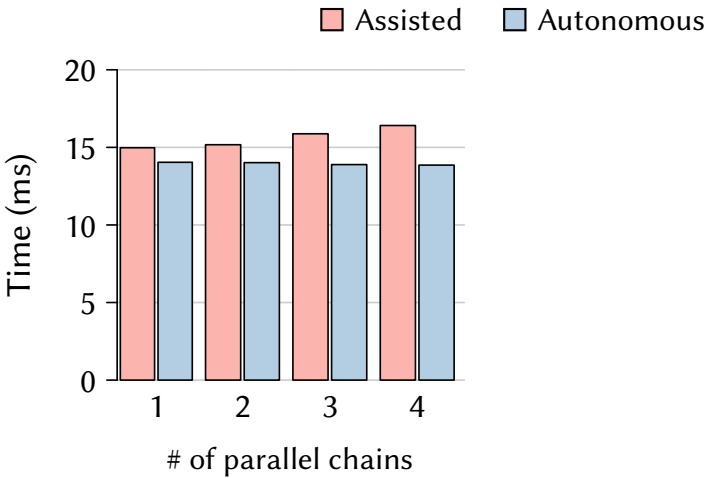
Accelerator Chains: Evaluation



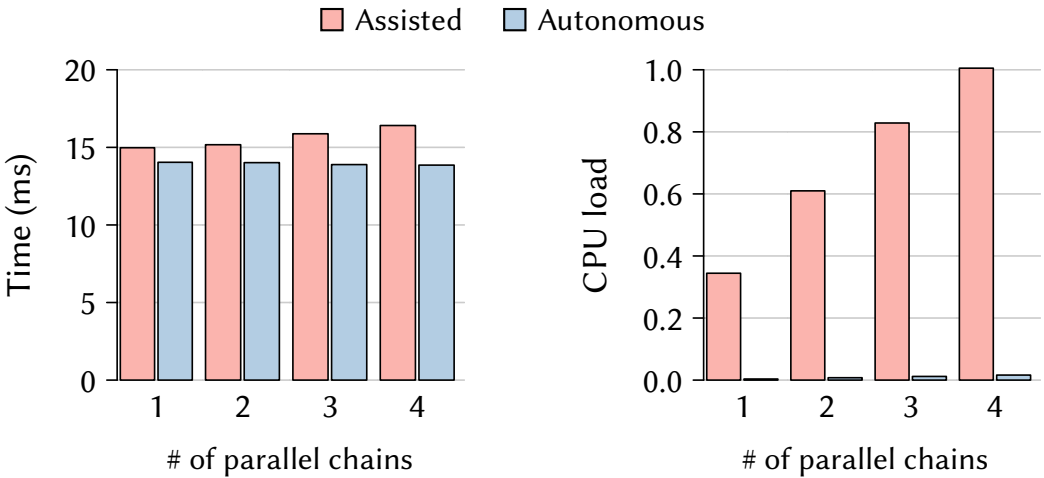
Accelerator Chains: Results



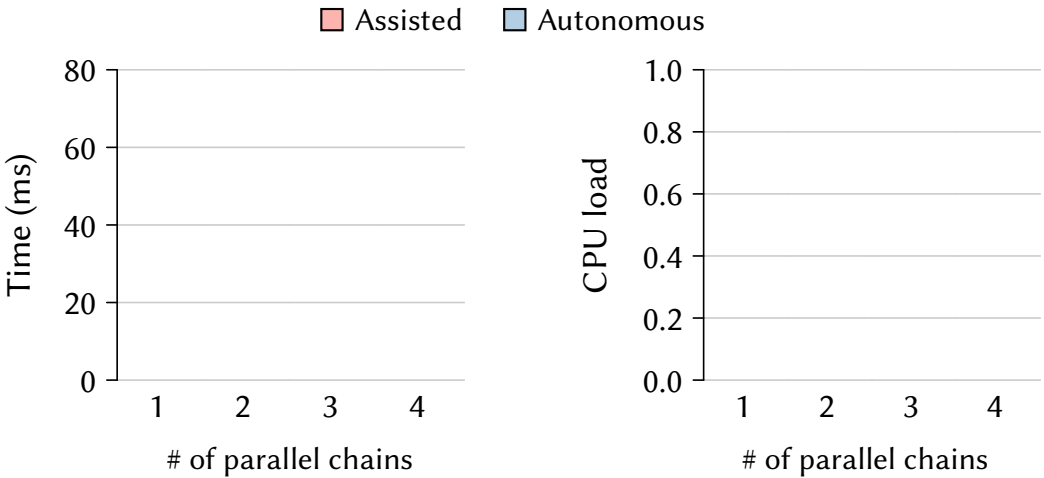
Accelerator Chains: Results



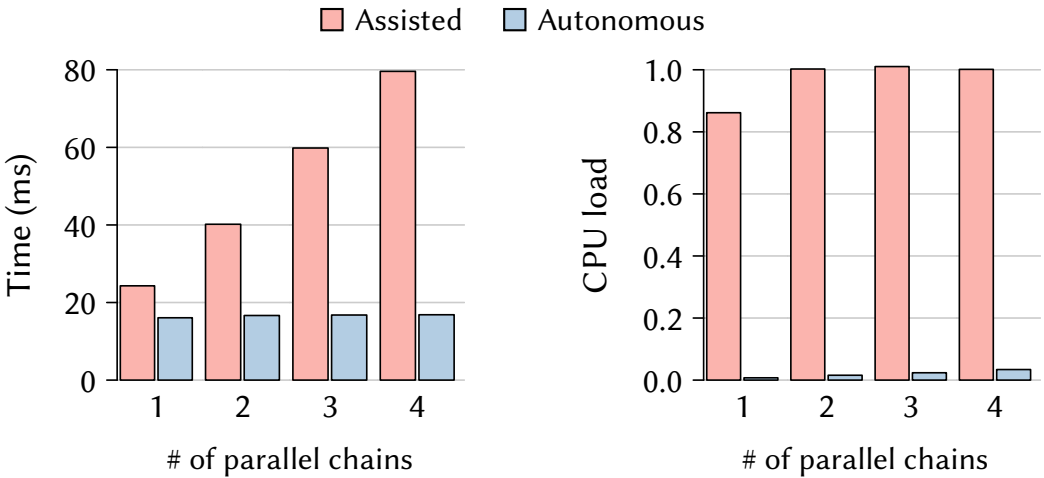
Accelerator Chains: Results



Accelerator Chains: Results (PCIe-like Latency)

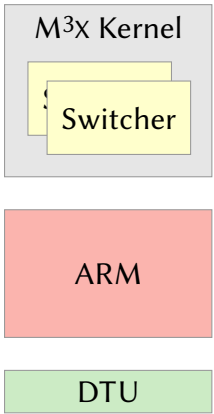


Accelerator Chains: Results (PCIe-like Latency)



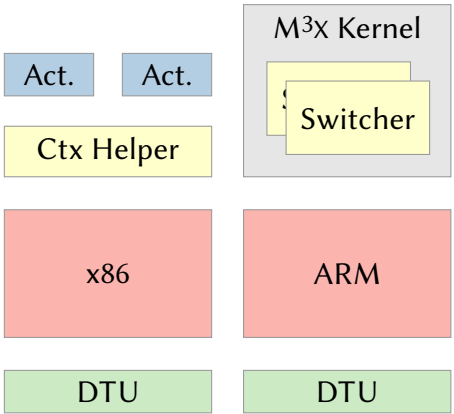
Outline

Context Switching Overview



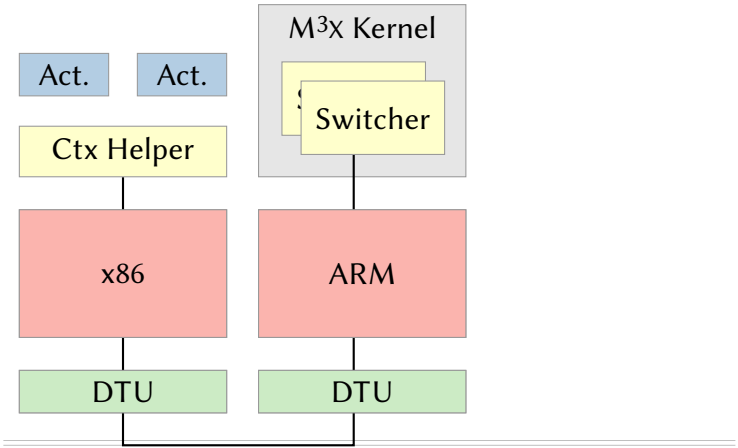
- Kernel handles complex part
  - Schedules/migrates activities
  - Initiates context switches

Context Switching Overview



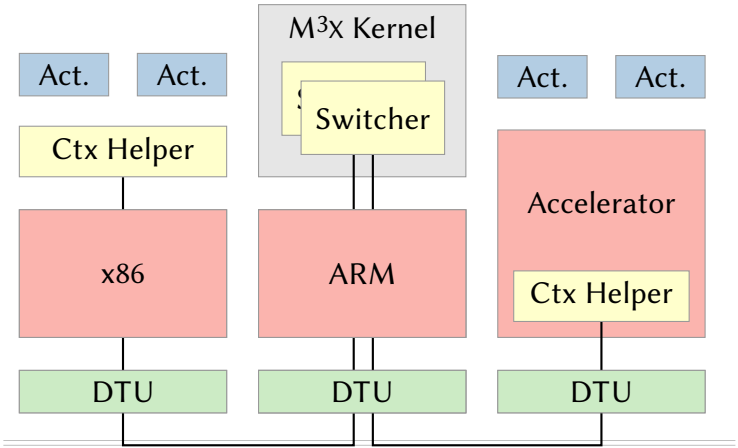
- Kernel handles complex part
  - Schedules/migrates activities
  - Initiates context switches
- Helper on user tiles implements save/restore
  - General purpose tiles: Software helper

Context Switching Overview



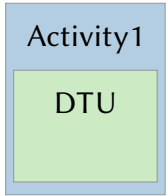
- Kernel handles complex part
  - Schedules/migrates activities
  - Initiates context switches
- Helper on user tiles implements save/restore
  - General purpose tiles: Software helper

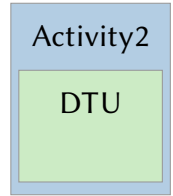
Context Switching Overview



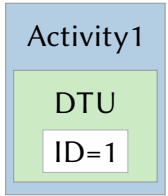
- Kernel handles complex part
  - Schedules/migrates activities
  - Initiates context switches
- Helper on user tiles implements save/restore
  - General purpose tiles: Software helper
  - Accelerator tiles: Helper implemented in hardware as part of ASM

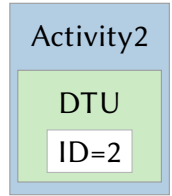
Combining Fast-Path Communication with Context Switching



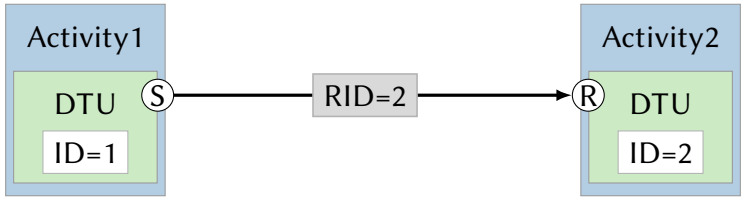


Combining Fast-Path Communication with Context Switching

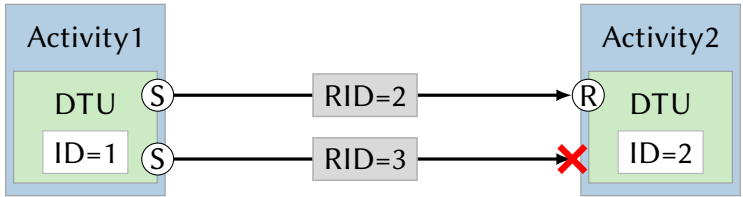




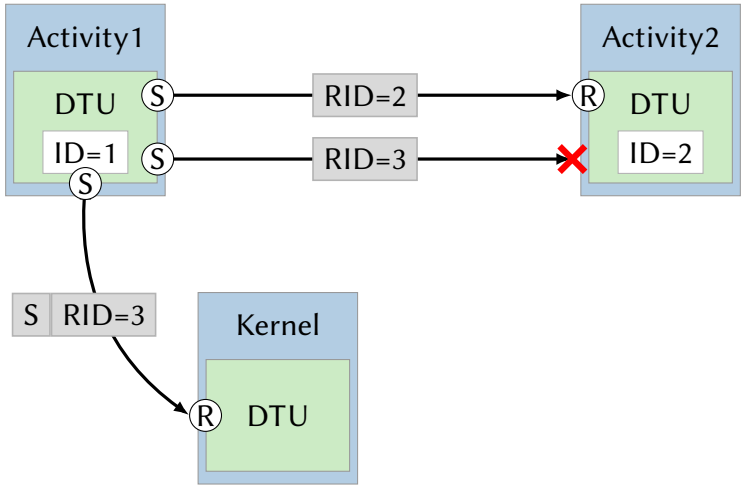
Combining Fast-Path Communication with Context Switching



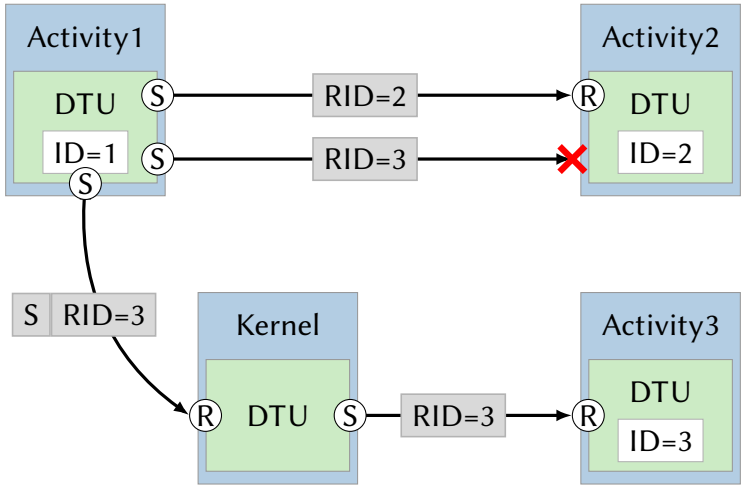
Combining Fast-Path Communication with Context Switching



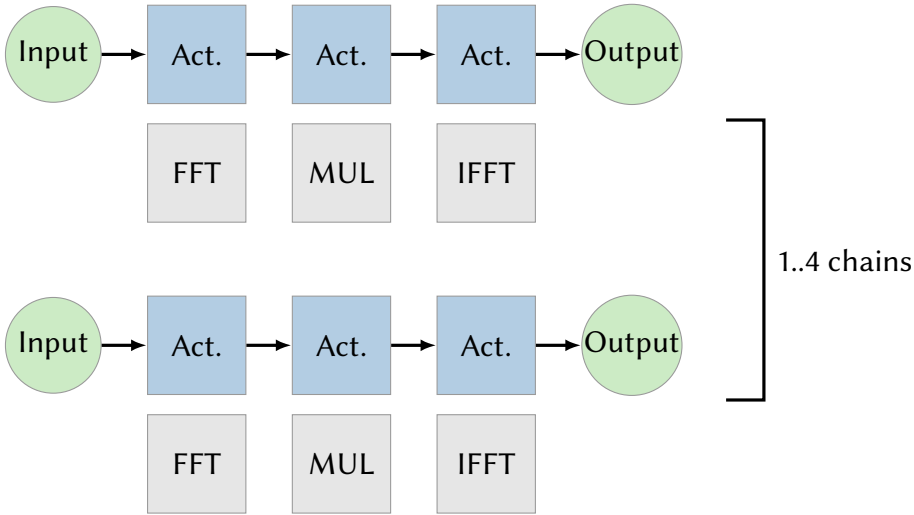
Combining Fast-Path Communication with Context Switching



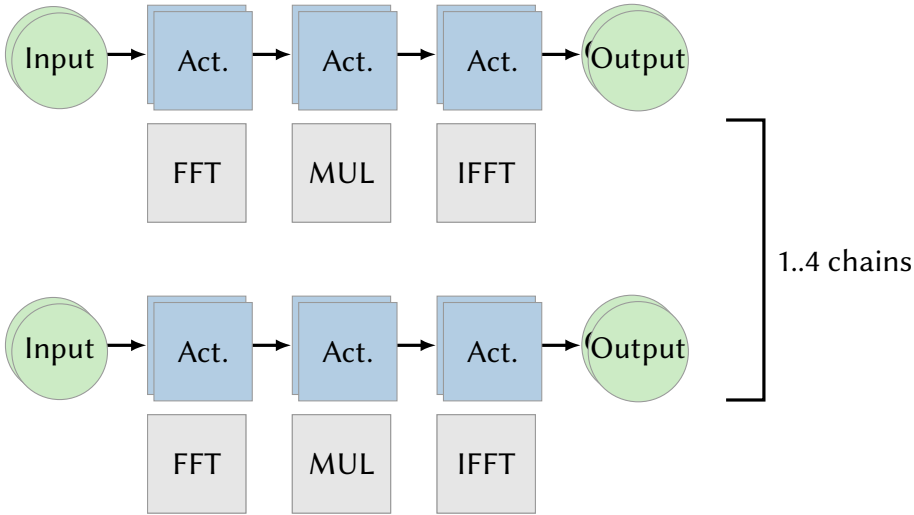
Combining Fast-Path Communication with Context Switching



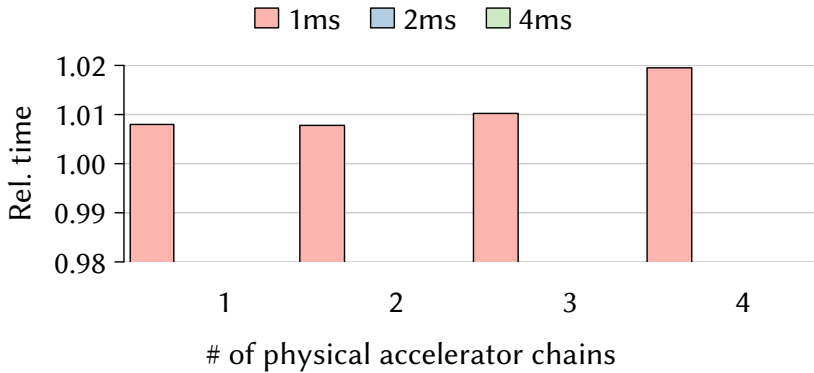
Results: Accelerator Sharing



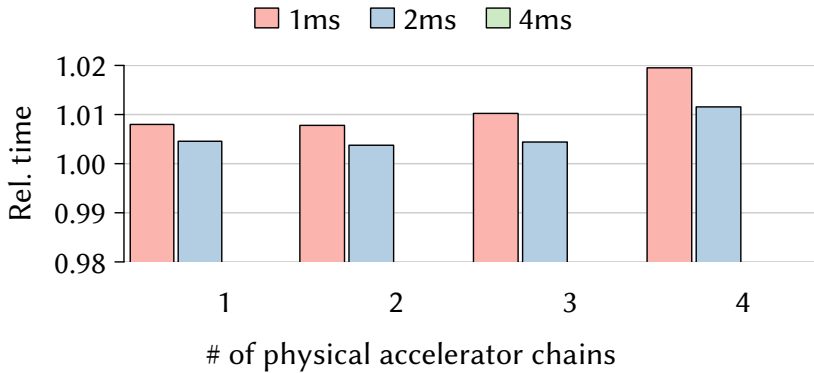
Results: Accelerator Sharing



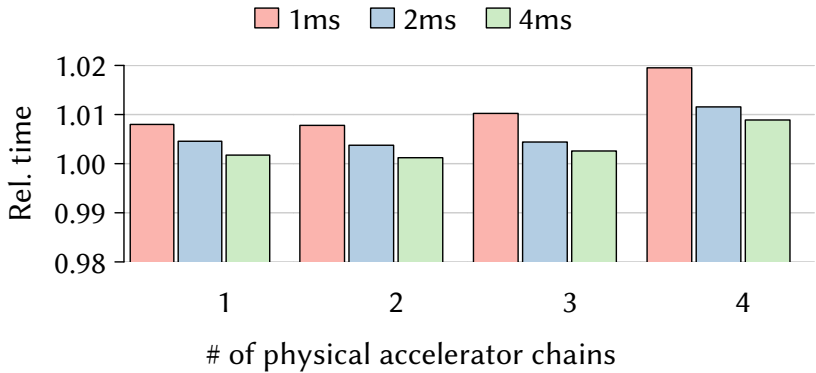
Results: Accelerator Sharing



Results: Accelerator Sharing



Results: Accelerator Sharing



Conclusion

- M³X enables autonomous accelerators and combines fast-path communication with context switching:
  - Adding uniform interface to compute units
  - Using simple and generic protocols
  - Adding lightweight component to accelerators
- Reduces CPU load by a factor of 30
- Retains advantages of fast-path communication
- Uses hardware resource efficiently

Conclusion

- M³X enables autonomous accelerators and combines fast-path communication with context switching:
  - Adding uniform interface to compute units
  - Using simple and generic protocols
  - Adding lightweight component to accelerators
- Reduces CPU load by a factor of 30
- Retains advantages of fast-path communication
- Uses hardware resource efficiently

- Visit the other talk on M³:

  SemperOS: A Distributed Capability System

  Exotic Kernel Features #2, 5:35 PM

Backup Slides

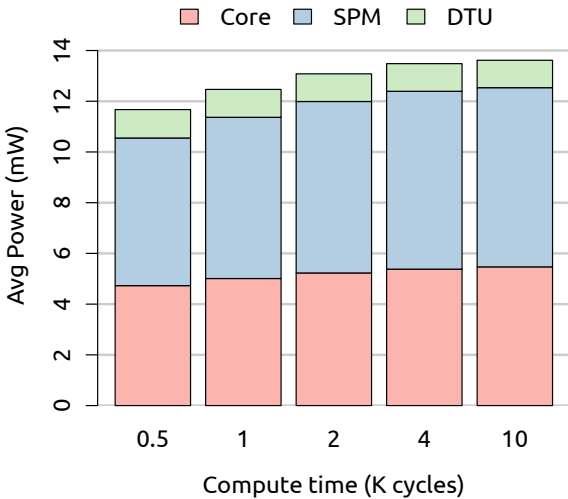
DTU Size

Module	**Area** (mm^2)
PE	0.476 864
├ SPM (32 KiB)	├ 0.211 805
├ Xtensa LX5	├ 0.185 259
└ DTU	└ 0.0798
├ Memory (8 EPs)	├ 0.060 991
└ Logic	└ 0.018 809

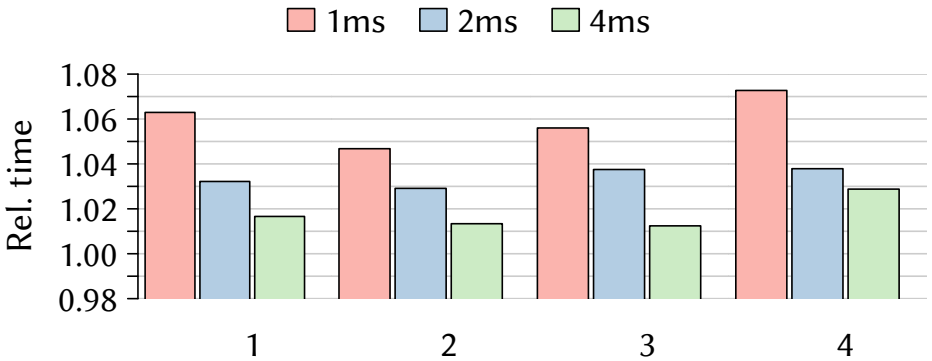
Comparison:

- Single Xtensa core has
  $\sim$ 50000 gates
- Single x86 core (haswell) has
  $\sim$ 100 Million gates

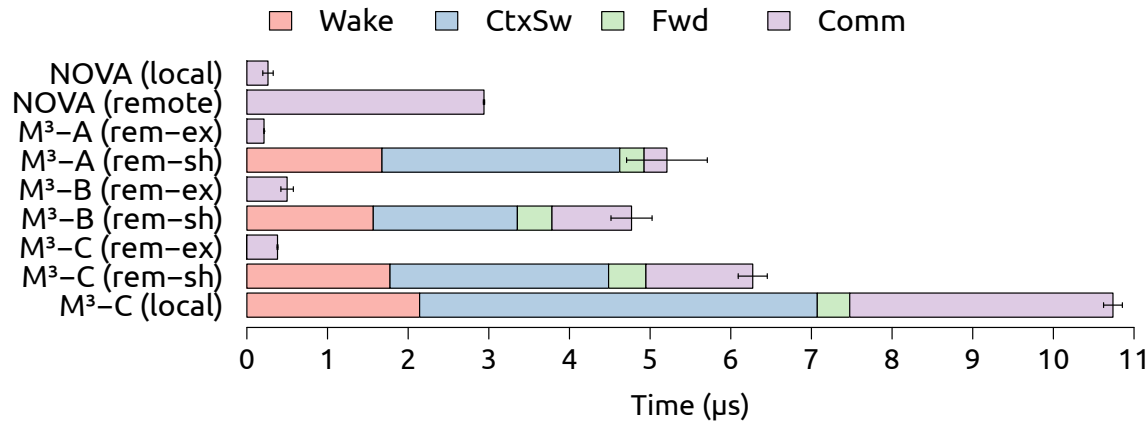
DTU Power Consumption



Accelerator Sharing (PCIe)



Context Switching Microbenchmark



Stream Processing ASM