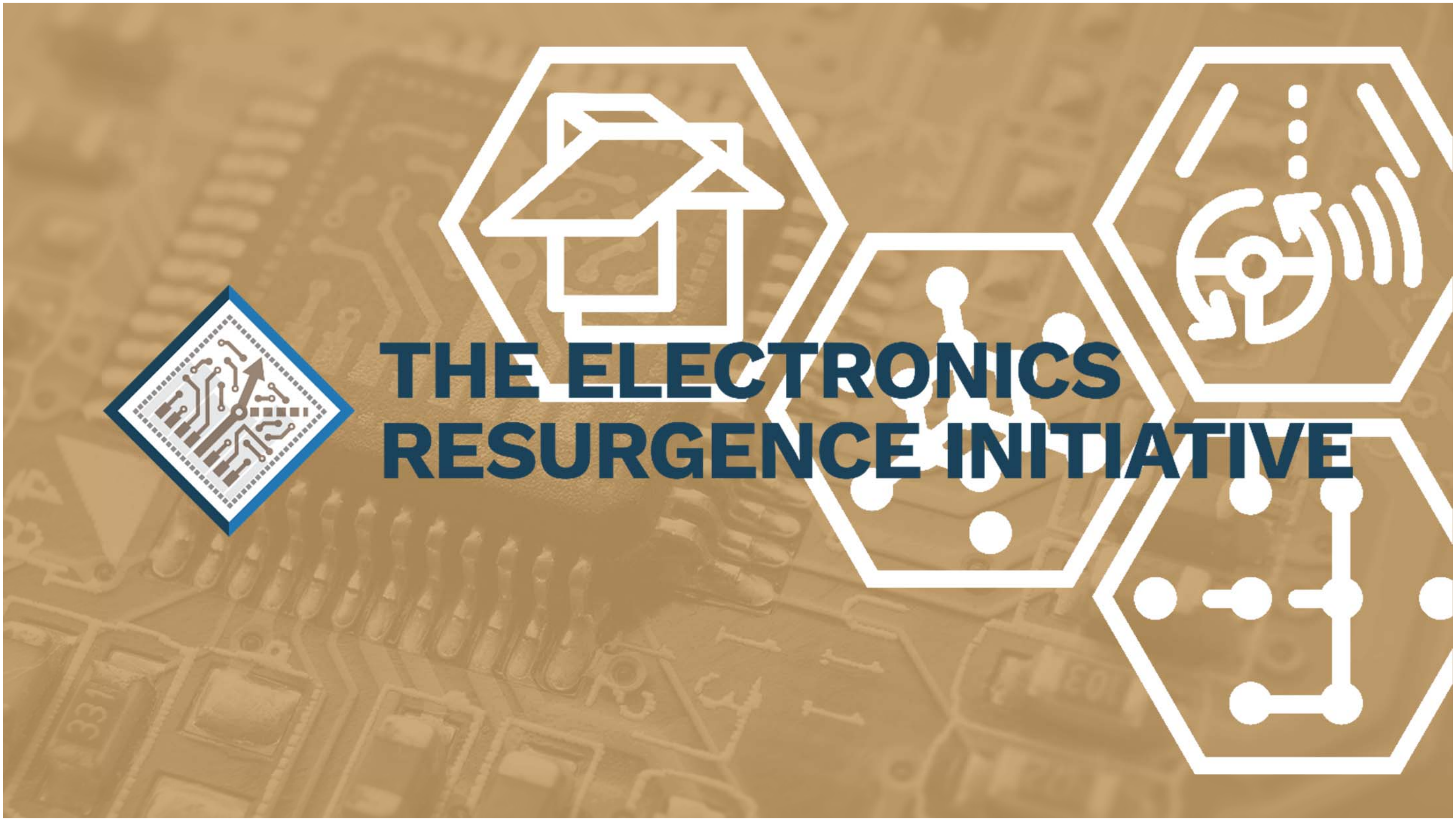
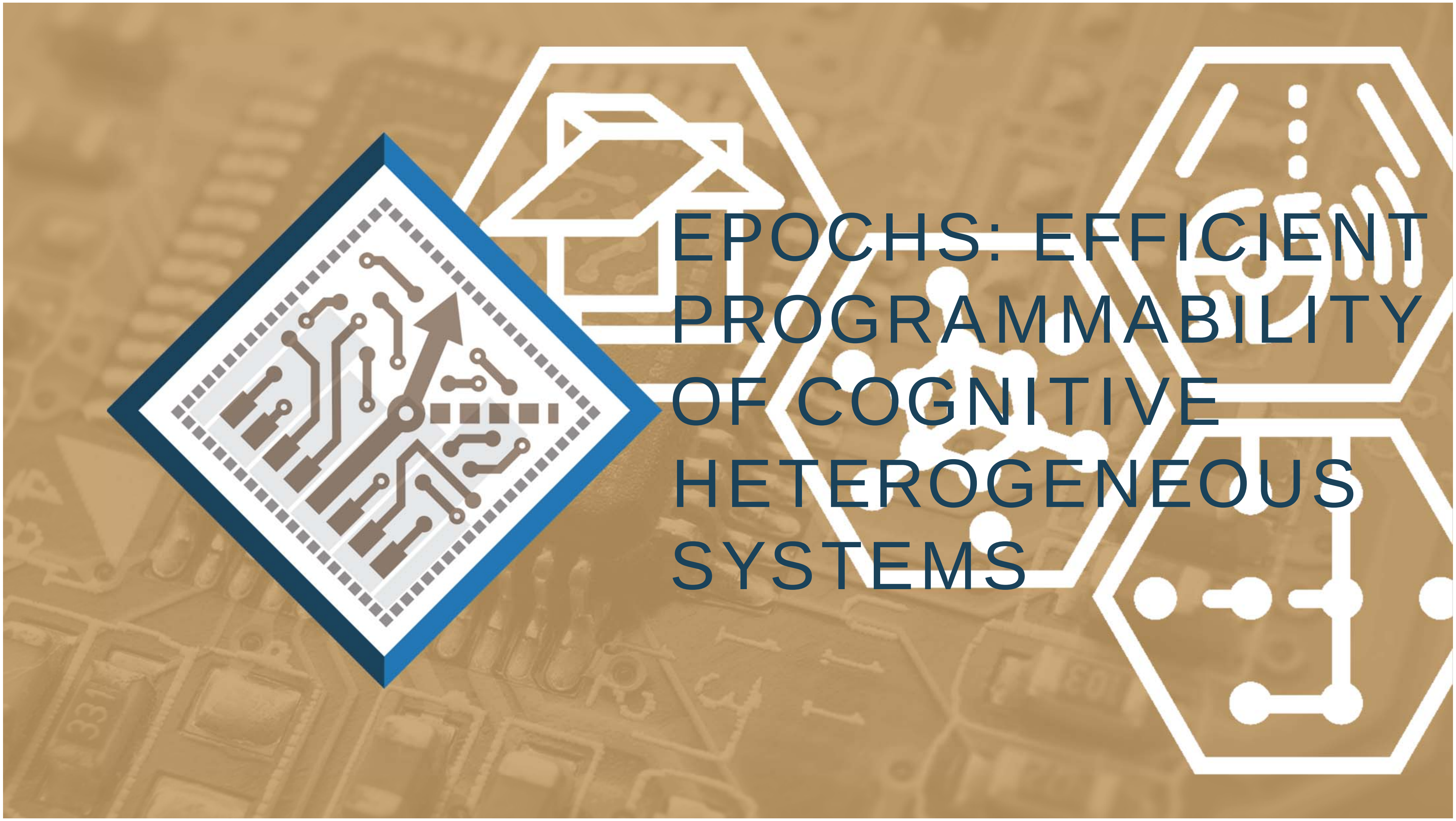


THE ELECTRONICS RESURGENCE INITIATIVE

Distribution Statement A. Approved for Public Release



EPOCHS: EFFICIENT PROGRAMMABILITY OF COGNITIVE HETEROGENEOUS SYSTEMS

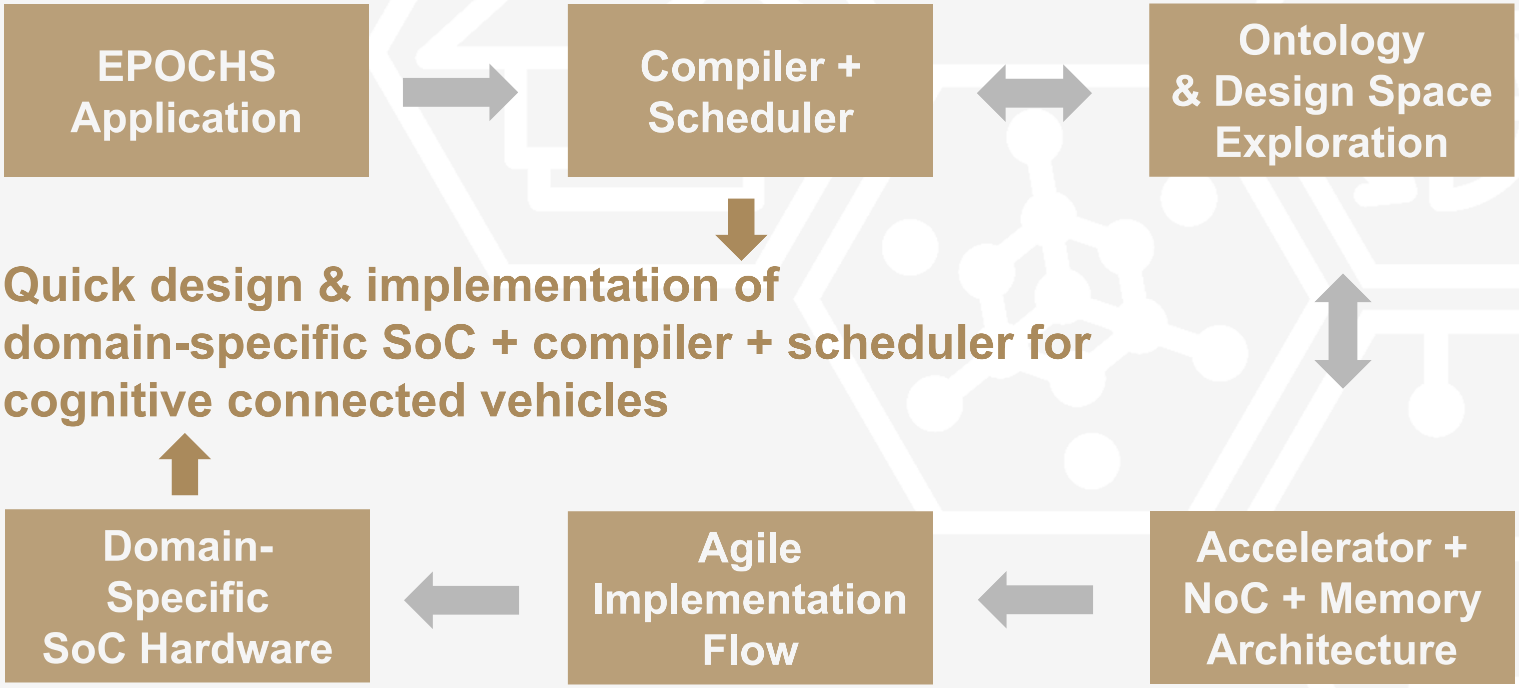
Distribution Statement A. Approved for Public Release

SARITA ADVE

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN
SADVE@ILLINOIS.EDU

EPOCHS OVERVIEW

S. Adve, V. Adve, P. Bose, D. Brooks, L. Carloni, S. Misailovic, V. Reddi, K. Shepard, G-Y Wei



EPOCHS Application

Compiler + Scheduler

Ontology & Design Space Exploration

Accelerator + NoC + Memory Architecture

Agile Implementation Flow

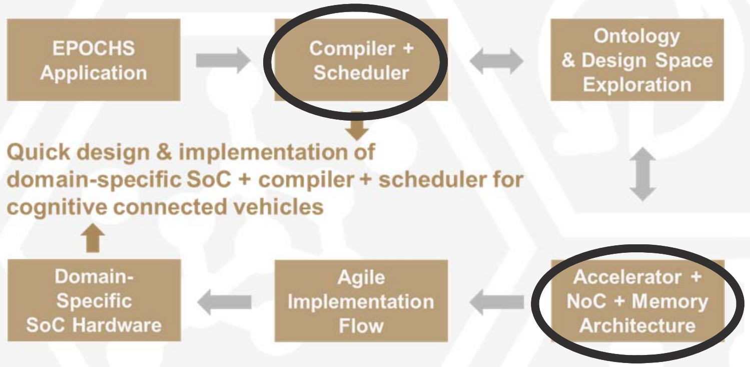
Domain-Specific SoC Hardware

Quick design & implementation of domain-specific SoC + compiler + scheduler for cognitive connected vehicles

KEY: HARDWARE & SOFTWARE INTERFACES

Key to heterogeneous system design: Interfaces for hardware & software

- Specializable
- Programmable
- Efficient


EPOCHS Application
Compiler + Scheduler
Ontology & Design Space Exploration
Quick design & implementation of domain-specific SoC + compiler + scheduler for cognitive connected vehicles
Domain-Specific SoC Hardware
Agile Implementation Flow
Accelerator + NoC + Memory Architecture

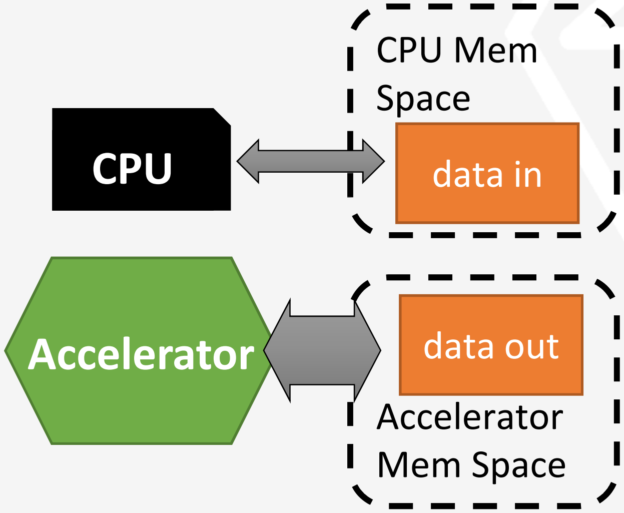
This talk: Two interfaces

- Spandex: Accelerator communication interface
- Heterogeneous Parallel Virtual Machine (HPVM): Hardware-software interface

Enable specialization with programmability and efficiency

HETEROGENEOUS SYSTEMS COMMUNICATION

Traditional heterogeneity:



CPU Mem Space

CPU

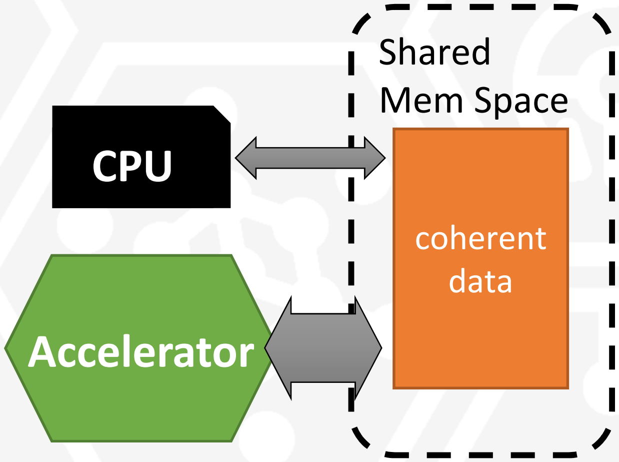
data in

Accelerator

data out

Accelerator Mem Space

Coherent shared memory:



Shared Mem Space

CPU

coherent data

Accelerator

- ✗ Wasteful data movement
- ✗ No fine-grain synchronization (atomics)
- ✗ No irregular access patterns

- ✓ Implicit data reuse
- ✓ Fine-grain synchronization (atomics)
- ✓ Irregular access patterns

Current solutions: complex and inflexible

HETEROGENEOUS DEVICES HAVE DIVERSE MEMORY DEMANDS

Temporal locality

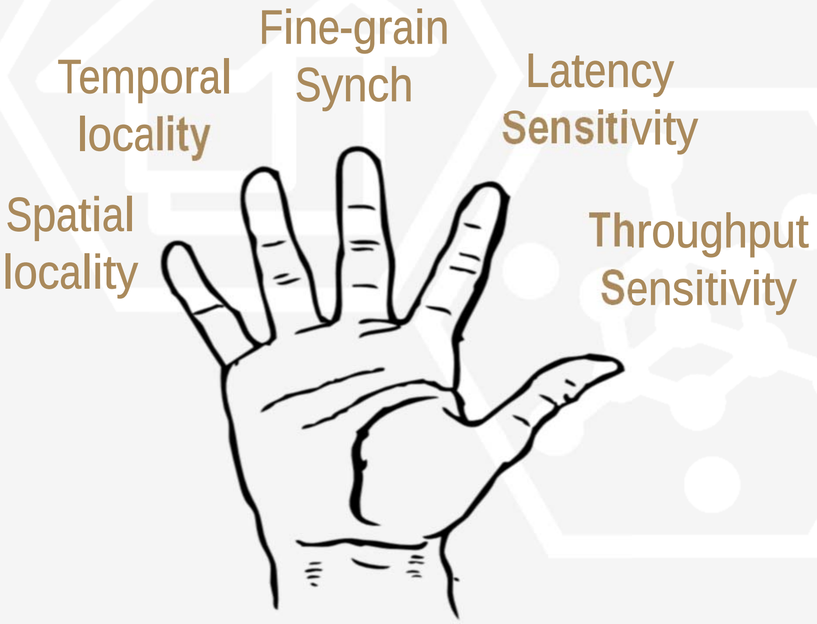
Fine-grain Synch

Latency Sensitivity

Spatial locality

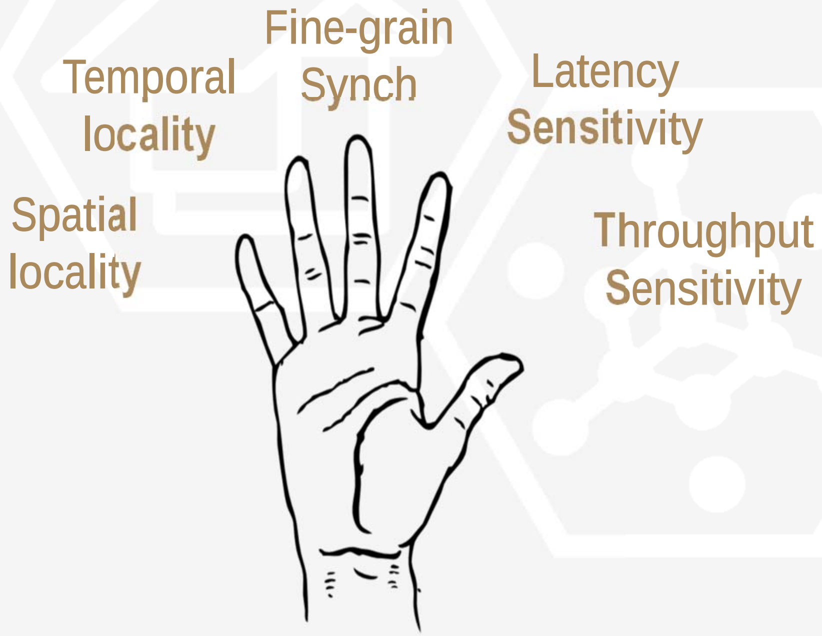
Throughput Sensitivity

HETEROGENEOUS DEVICES HAVE DIVERSE MEMORY DEMANDS



Typical CPU workloads: fine-grain synchronization, latency sensitive

HETEROGENEOUS DEVICES HAVE DIVERSE MEMORY DEMANDS



Typical GPU workloads: spatial locality, throughput sensitive

KEY PROPERTIES OF COHERENCE PROTOCOLS

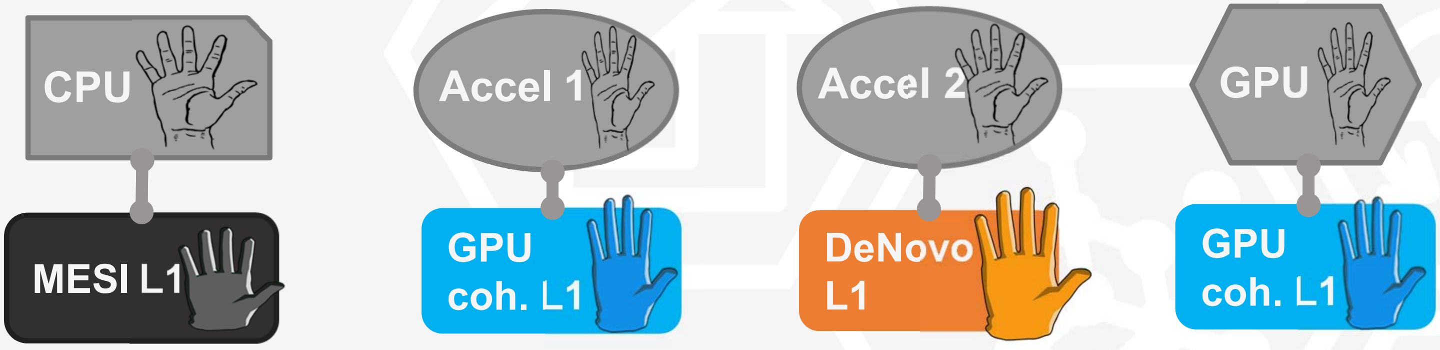
Properties	
Granularity	
Invalidation	
Updates	

KEY PROPERTIES OF COHERENCE PROTOCOLS

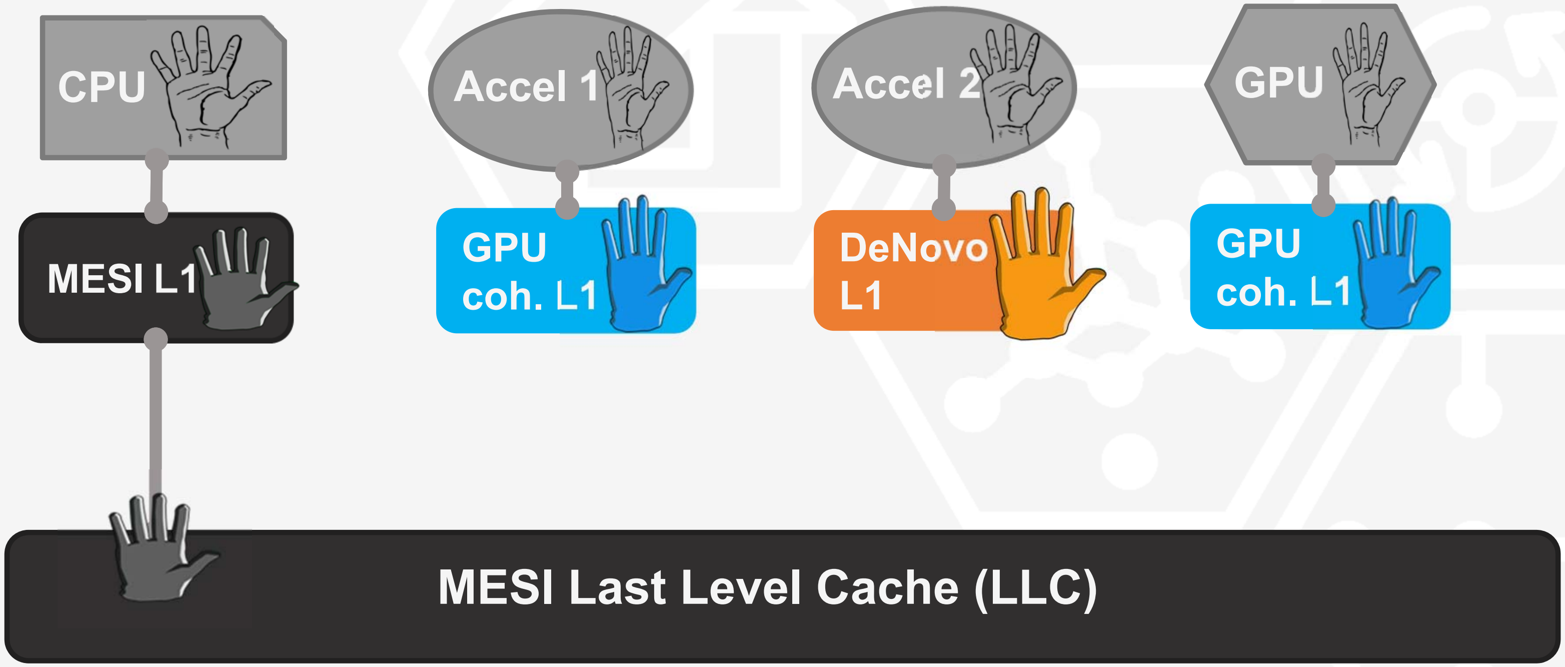
Properties	CPU	GPU	DeNovo (for CPU or GPU)
Granularity	Line	Reads: Line Writes: Word	Reads: Flexible Writes: Word
Invalidation	Writer-invalidate	Self-invalidate	Self-invalidate
Updates	Ownership	Write-through	Ownership

How to integrate different accelerators with different protocols?

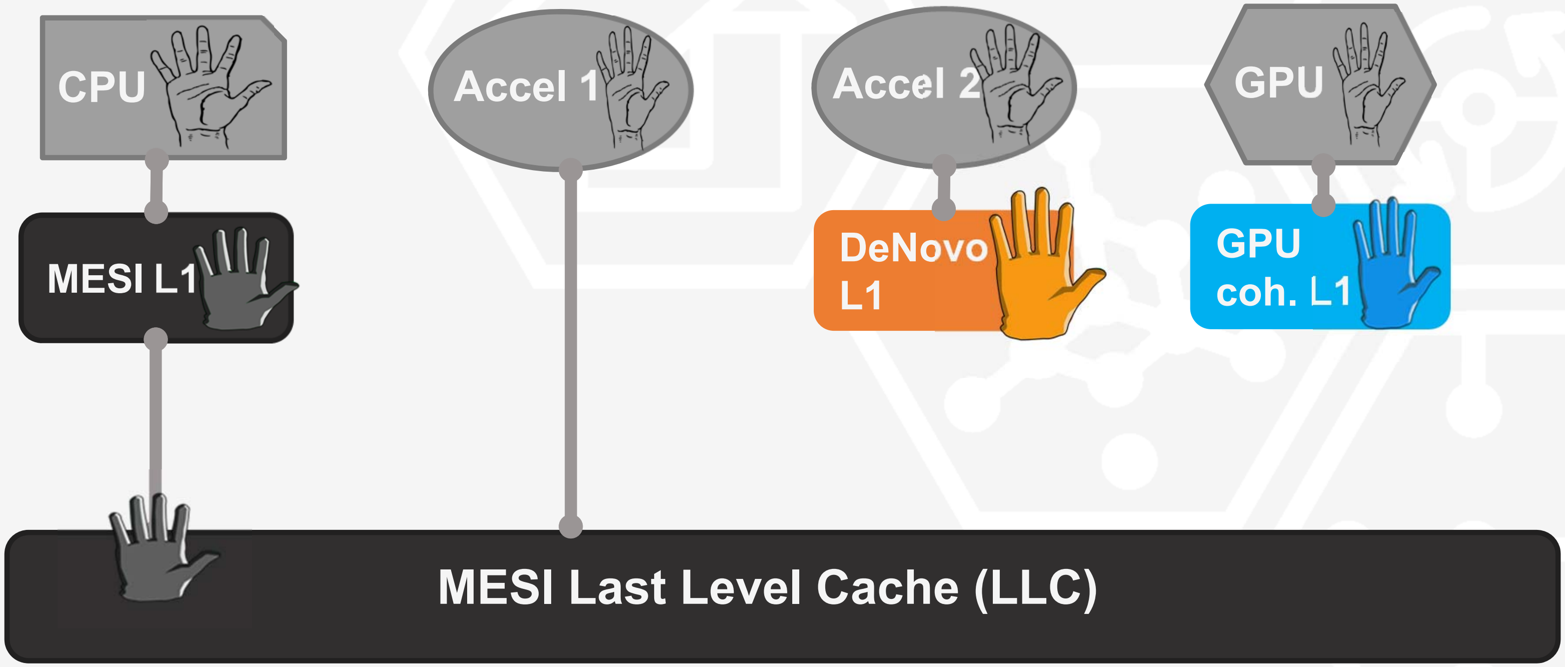
CURRENT SOLUTIONS: INFLEXIBLE, INEFFICIENT



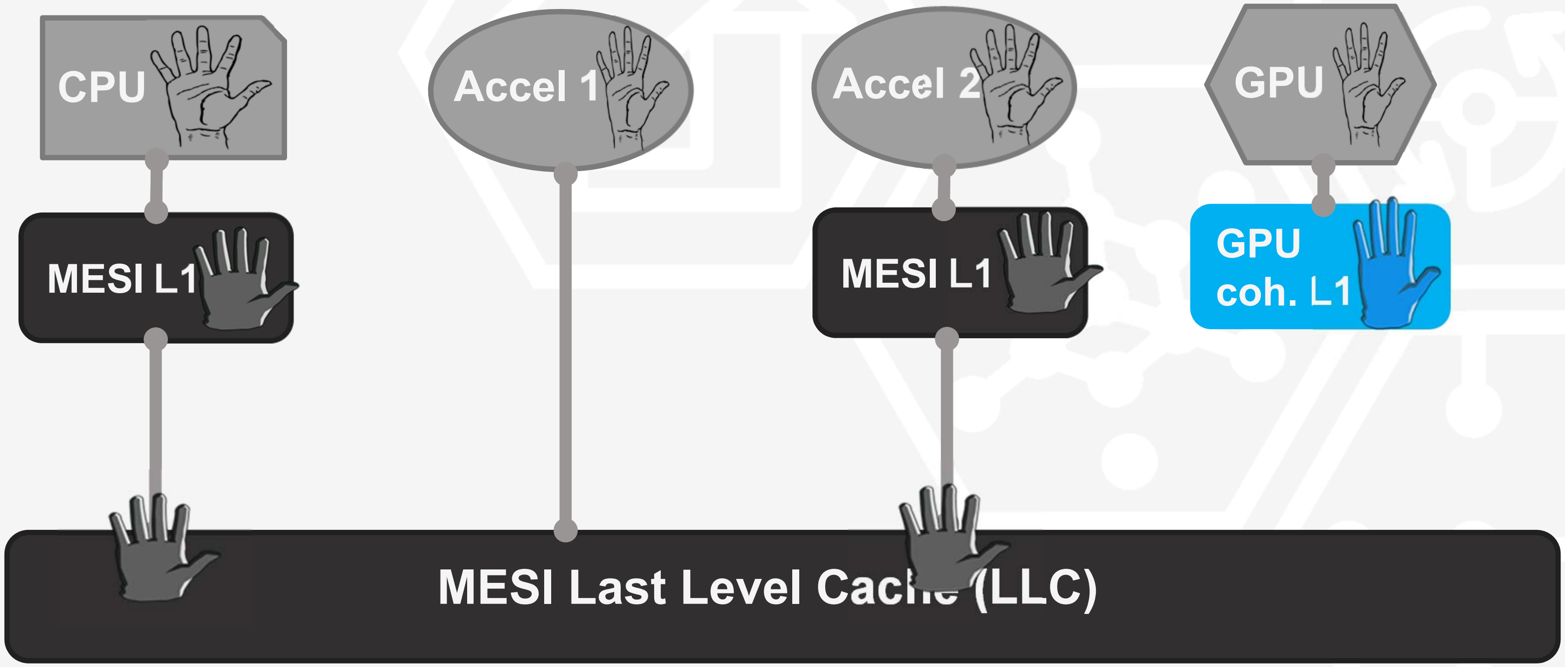
CURRENT SOLUTIONS: INFLEXIBLE, INEFFICIENT



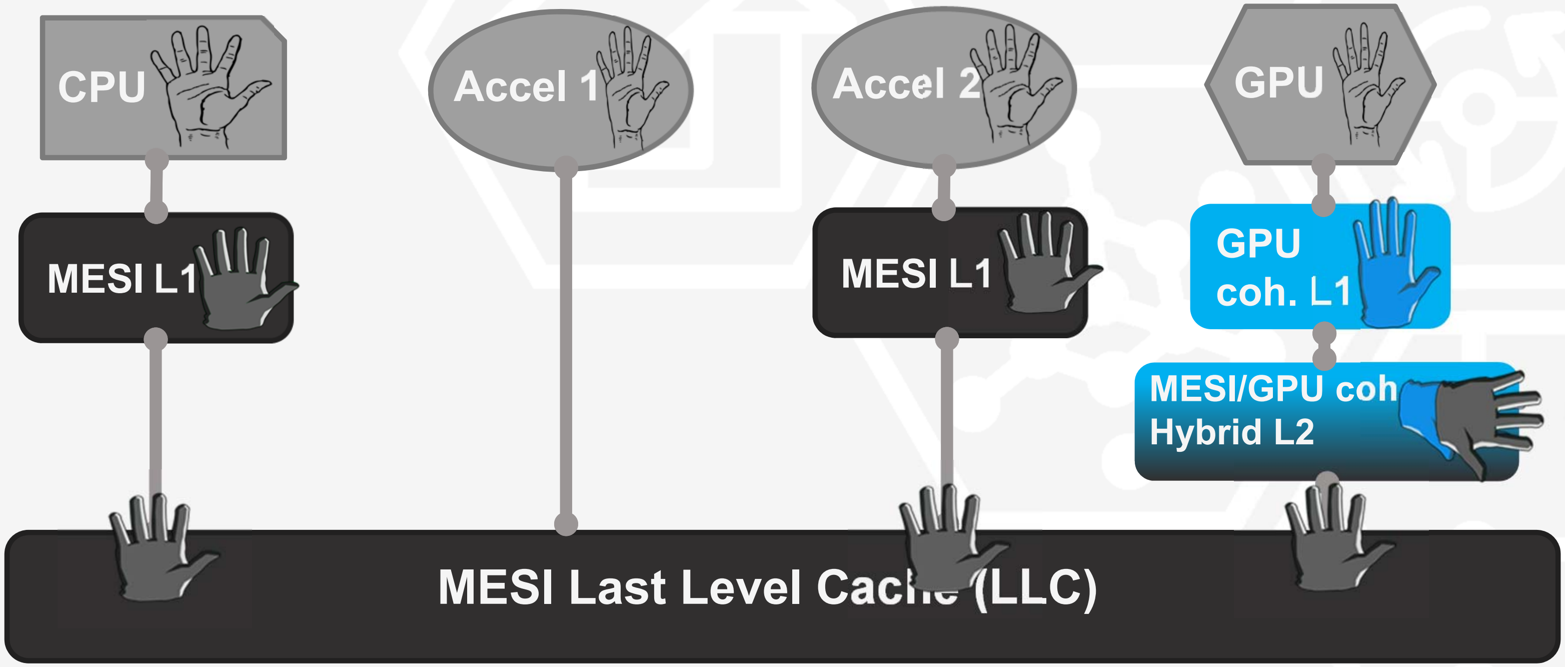
CURRENT SOLUTIONS: INFLEXIBLE, INEFFICIENT



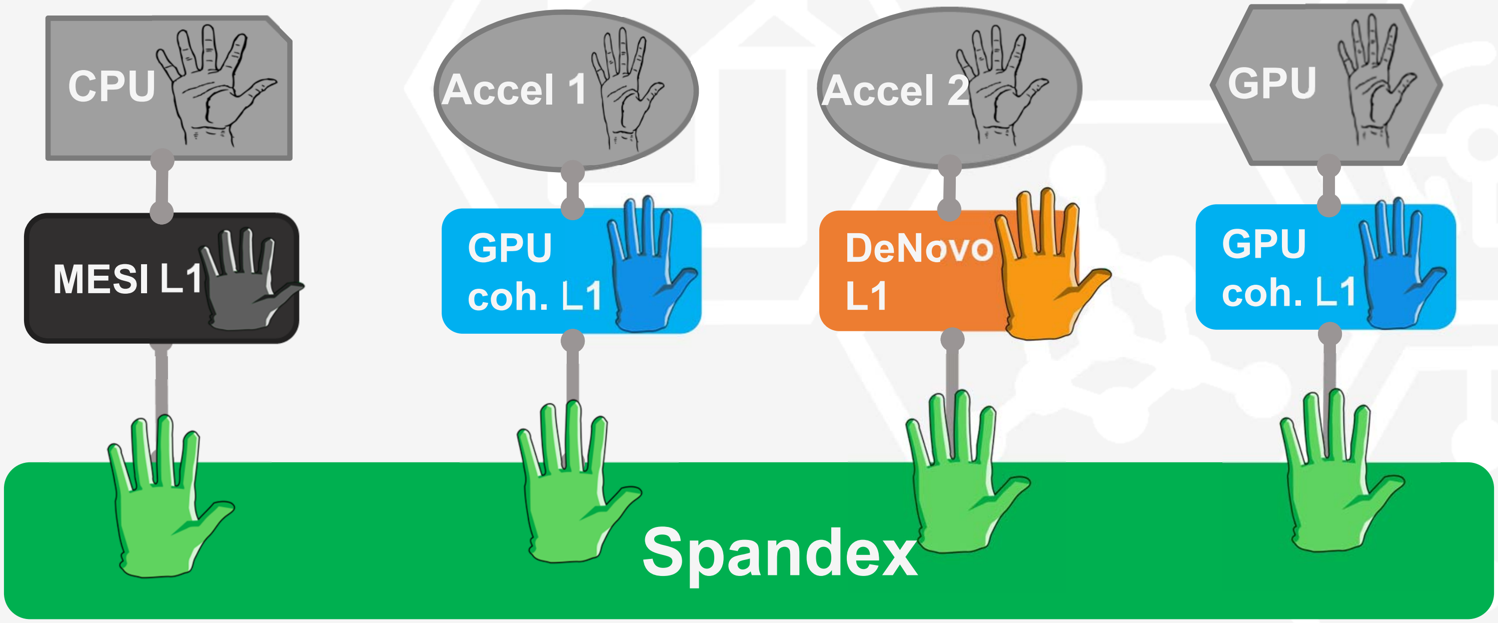
CURRENT SOLUTIONS: INFLEXIBLE, INEFFICIENT



CURRENT SOLUTIONS: INFLEXIBLE, INEFFICIENT



SPANDEX: FLEXIBLE HETEROGENEOUS COHERENCE INTERFACE [ISCA'18]

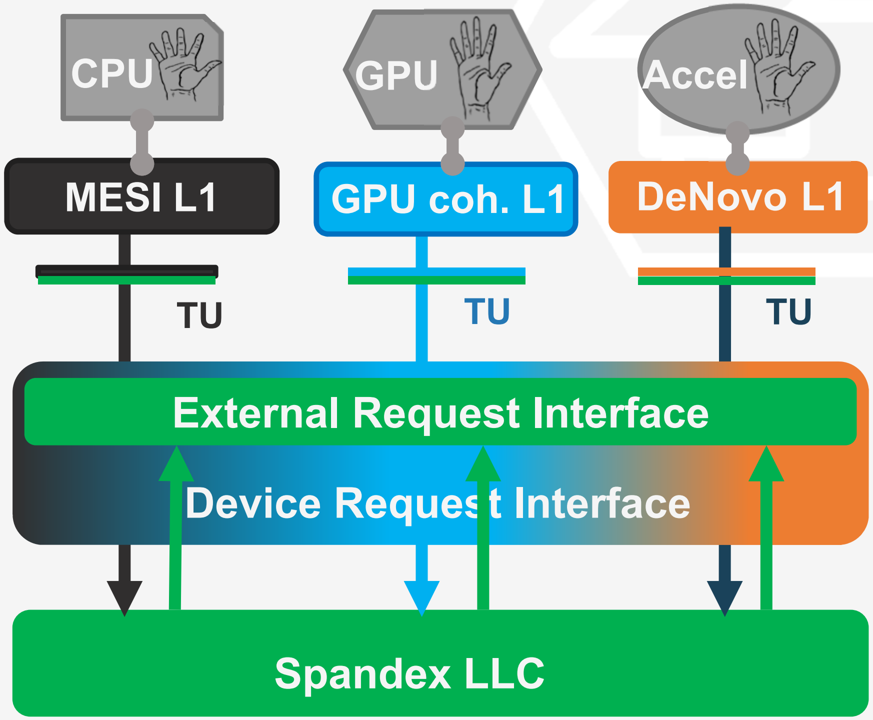


- Adapts to exploit individual device's workload attributes
- Better performance, lower complexity

⇒ Fits like a glove for heterogeneous systems!

Supported by ADA JUMP and C-FAR STARNET centers

SPANDEX OVERVIEW

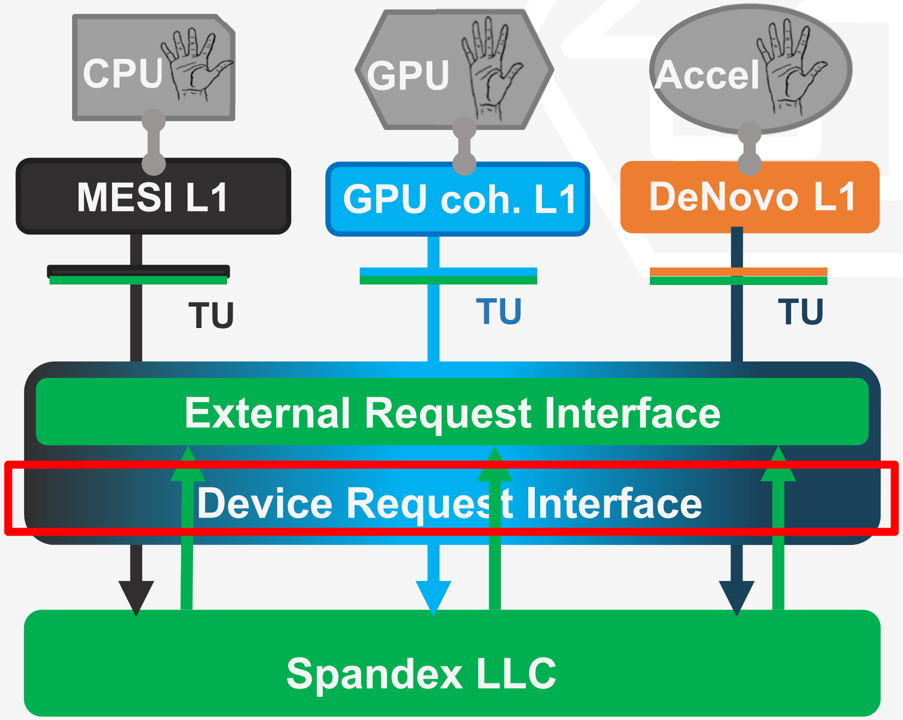


Key Components

- Flexible device request interface
- DeNovo-based LLC
- External request interface

Device may need a translation unit (TU)

SPANDEX OVERVIEW



Key Components

- Flexible device request interface

- DeNovo-based LLC

- External request interface

Device may need a translation unit (TU)

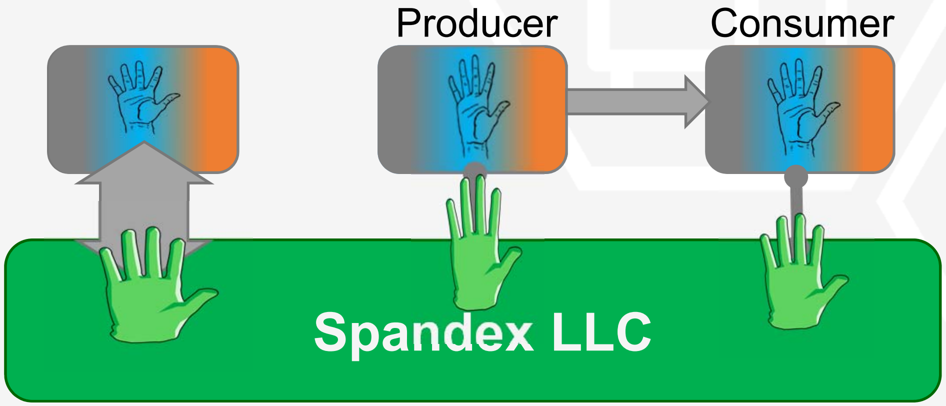
SPANDEX DEVICE REQUEST INTERFACE

Action	Request	Indicates
Read	ReqV	Self-invalidation
	ReqS	Writer-invalidation
Write	ReqWT	Write-through
	ReqO	Ownership only
Read+ Write	ReqWT+data	Atomic write-through
	ReqO+data	Ownership + Data
Writeback	ReqWB	Owned data eviction

- Requests also specify granularity and (optionally) a bitmask

DYNAMIC & NEW COHERENCE SPECIALIZATIONS

- Spandex flexibility & simplicity enables dynamic & new coherence specializations



Dynamic Spandex request selection

Producer-consumer forwarding

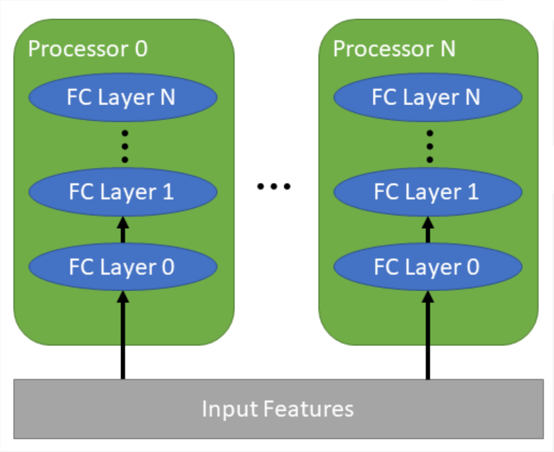
Extended granularity flexibility

- Directed by compiler or runtime
- Applied to neural networks: enables new programming patterns

DATA PARALLEL VS. PIPELINED NEURAL NETWORKS

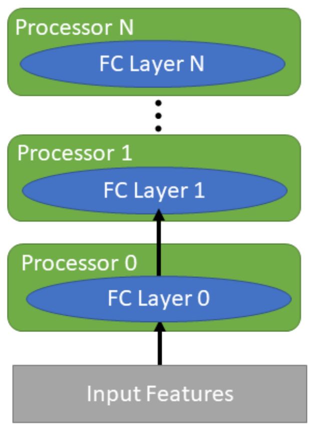
Fully connected DNN

Data Parallel



- Each core has independent image
- Each core processes all network layers
- No communication/synch overhead
- But need weights of all layers for reuse

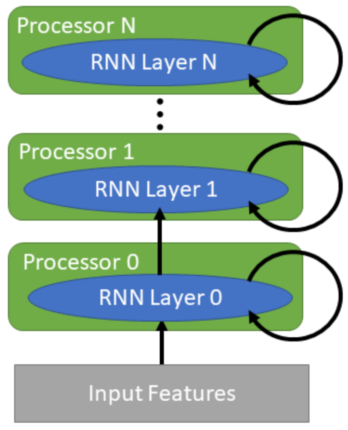
Pipelined



- Each core processes one layer
- Communication/synch overhead
- But need weights of only one layer for reuse

Recurrent neural nw (RNN)

Pipelined (no simple data parallel)

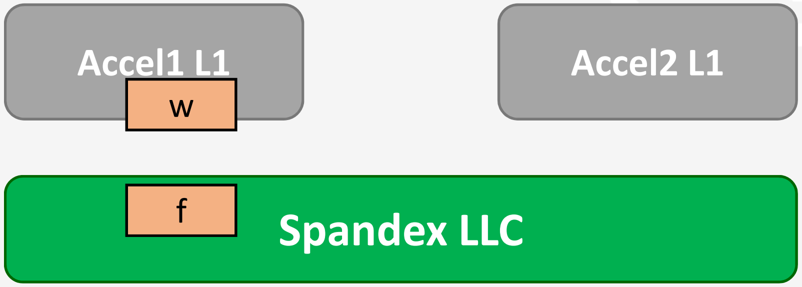


- No simple communication-free data parallel version since computation depends on previous input

Spandex flexibility enables fine-grained communication and efficient pipelines

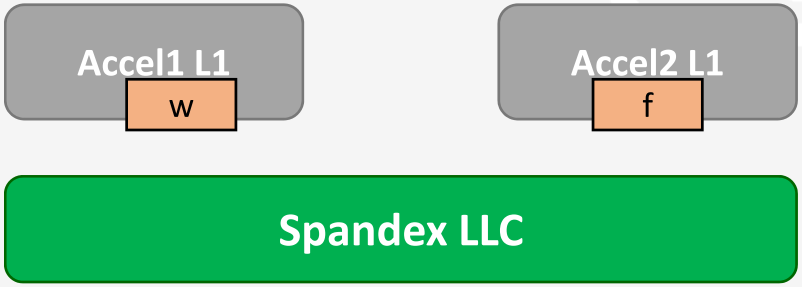
COHERENCE SPECIALIZATION FOR PIPELINED NEURAL NETWORKS

- Dynamically select coherence strategy for different data
  - Use ownership for weight data (reuse), writethrough for input feature data



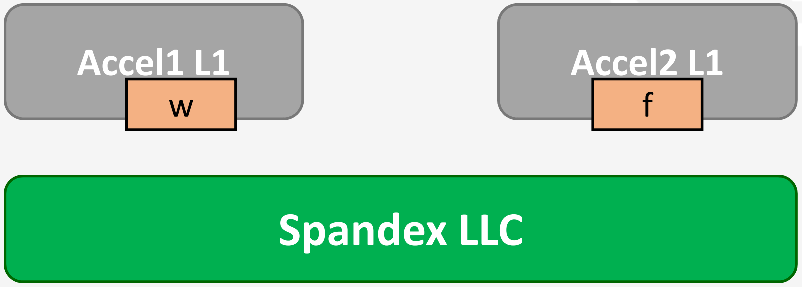
COHERENCE SPECIALIZATION FOR PIPELINED NEURAL NETWORKS

- Dynamically select coherence strategy for different data
  - Use ownership for weight data (reuse), writethrough for input feature data
- Optimization1: Producer → Consumer forward via last level cache (LLC)
  - LLC forwards feature output to next consumer



COHERENCE SPECIALIZATION FOR PIPELINED NEURAL NETWORKS

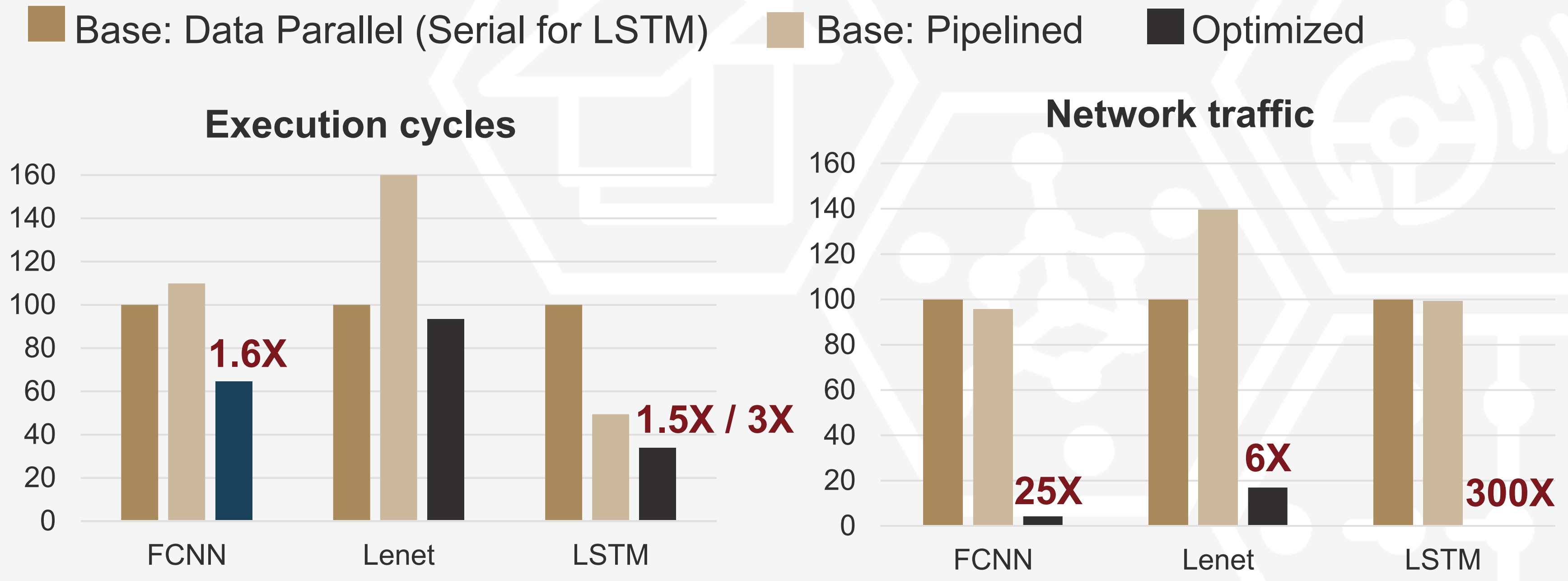
- Dynamically select coherence strategy for different data
  - Use ownership for weight data (reuse), writethrough for input feature data
- Optimization1: Producer → Consumer forward via last level cache (LLC)
  - LLC forwards feature output to next consumer
- Optimization2: Direct Producer → Consumer forward w/ owner prediction
  - Producer directly forwards feature data to consumer cache, with no LLC lookup



EVALUATION METHODOLOGY

- Baseline system: CPU + Multiple GPU compute cores on a network on chip
- Neural network (NN) computation on different compute cores of GPUs
- NNs based on standard networks or information from literature
- Cycle accurate architectural simulation: GEMS+GPGPUSim+Garnett+Spandex

RESULTS FOR NEURAL NETWORKS



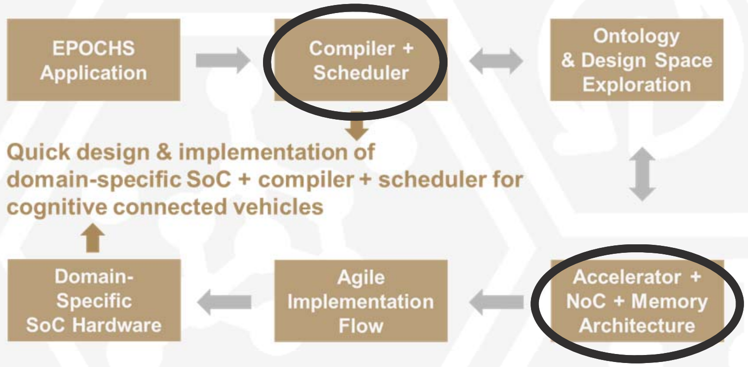
Large reduction in execution time and/or network traffic

Next steps for Spandex: Apply to EPOCHS application and integrate with compiler

RECAP SO FAR

Key to heterogeneous system design: Interfaces for hardware & software

- Specializable
- Programmable
- Efficient



EPOCHS Application

Compiler + Scheduler

Ontology & Design Space Exploration

Quick design & implementation of domain-specific SoC + compiler + scheduler for cognitive connected vehicles

Domain-Specific SoC Hardware

Agile Implementation Flow

Accelerator + NoC + Memory Architecture

This talk: Two interfaces

✓ Spandex: Accelerator communication interface

➡ Heterogeneous Parallel Virtual Machine (HPVM): Hardware-software interface

Enable specialization with programmability and efficiency

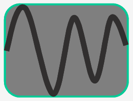
CURRENT INTERFACE LEVELS

App. productivity	***Domain-specific prog. language***
App. performance	***General-purpose prog. language***
Language innovation	***Language-level Compiler IR***
Compiler investment	***Language-neutral Compiler IR***
Object-code portability	***Virtual ISA***
Hardware innovation	***"Hardware" ISA***

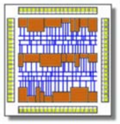
CPUs + Vector SIMD Units

GPU

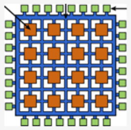
…



DSP

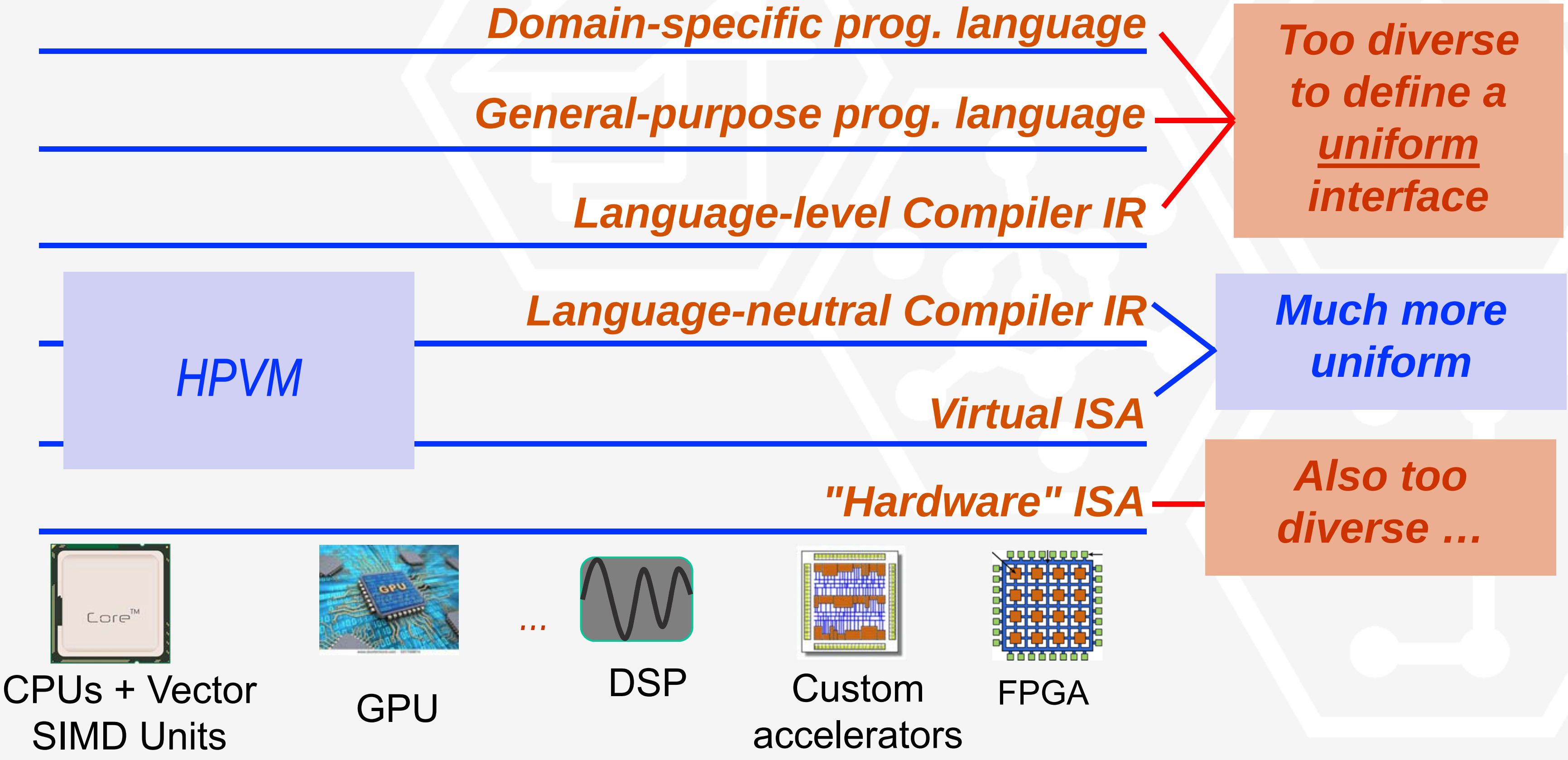


Custom accelerators



FPGA

WHERE SHOULD WE ABSTRACT HETEROGENEITY?



Domain-specific prog. language

General-purpose prog. language

Language-level Compiler IR

Too diverse to define a __uniform__ interface

HPVM

Language-neutral Compiler IR

Virtual ISA

Much more uniform

"Hardware" ISA

Also too diverse …

CPUs + Vector SIMD Units

GPU

…

DSP

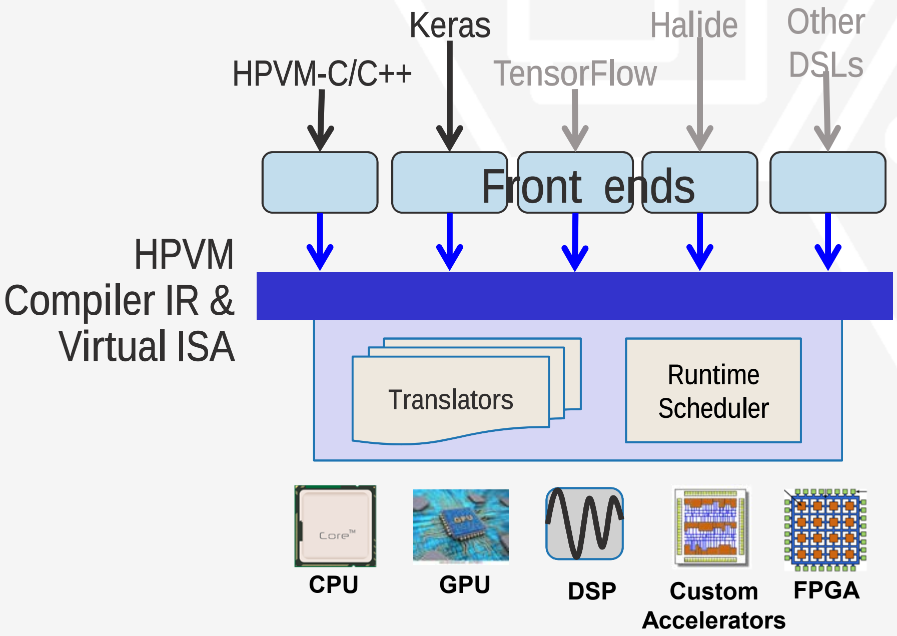
Custom accelerators

FPGA

HETEROGENEOUS PARALLEL VIRTUAL MACHINE

Key to Programmability:

Common abstractions for heterogeneous parallel hardware



Use HPVM for:

1. Portable *object* code
2. Retargetable parallel compiler IR and system
3. Run-time scheduling

Kotsifakou et al., PPOPP 2018

ABSTRACTION OF PARALLEL COMPUTATION

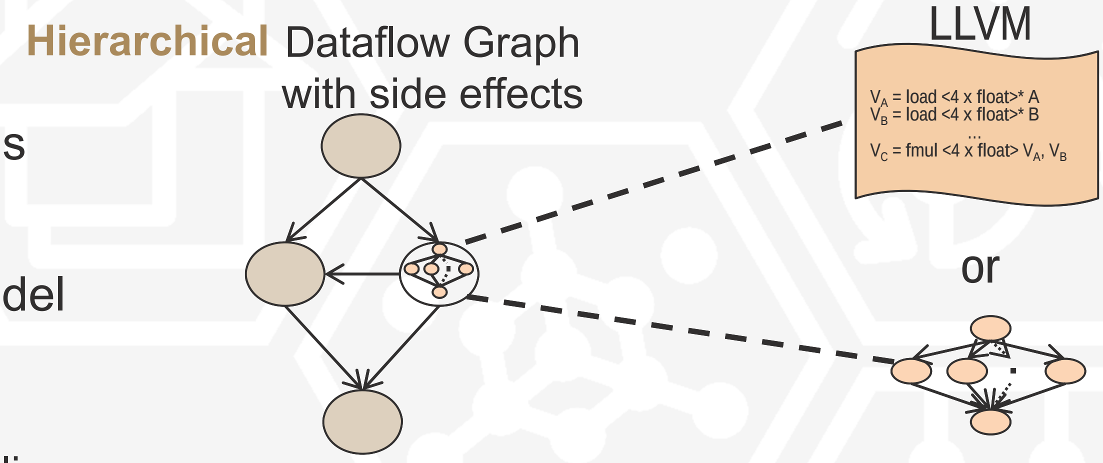
N different parallelism models

Single unified parallelism model

Captures

- coarse grain task parallelism
- streams, pipelined parallelism
- nested parallelism
- SPMD-style data parallelism
- fine grain vector parallelism

Supports high-level optimizations as graph transformations

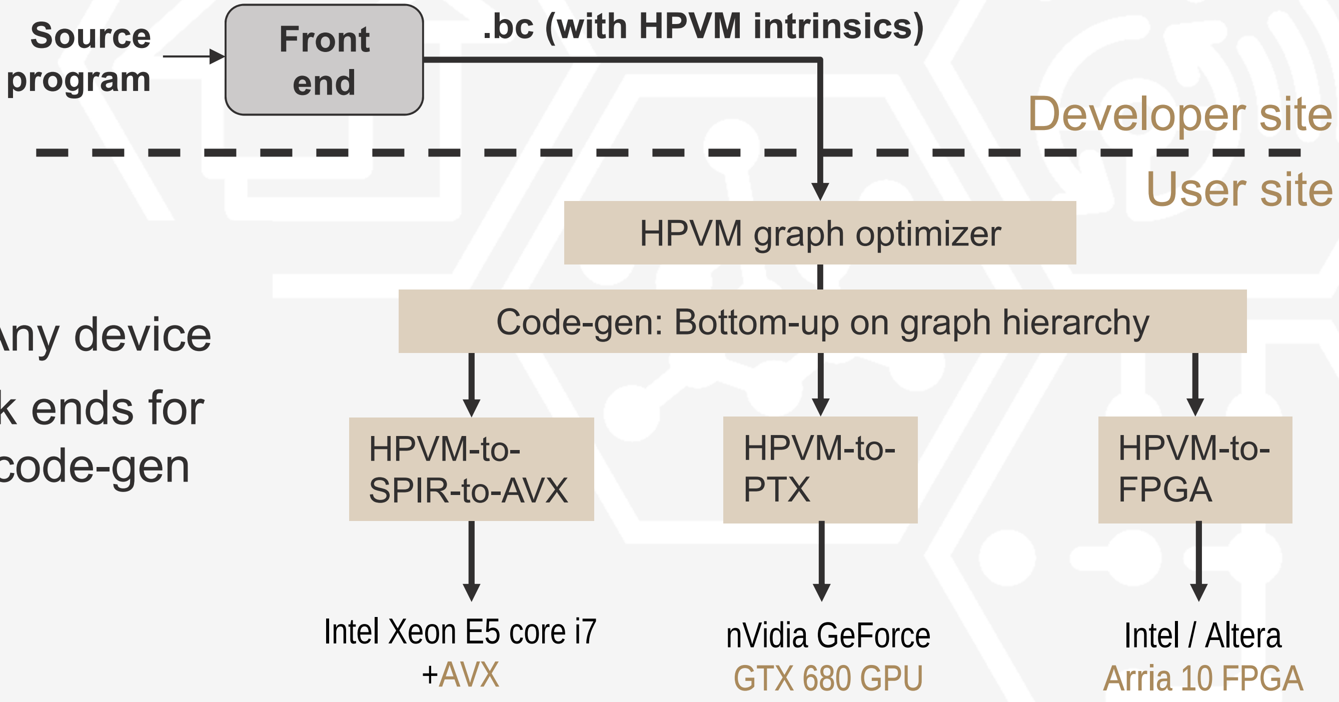


Hierarchical Dataflow Graph with side effects

LLVM

V_A = load <4 x float>* A
V_B = load <4 x float>* B
...
V_C = fmul <4 x float> V_A, V_B

HPVM BOTTOM-UP CODE GENERATION



Source program
Front end
.bc (with HPVM intrinsics)
Developer site
User site
HPVM graph optimizer
Code-gen: Bottom-up on graph hierarchy
HPVM-to-SPIR-to-AVX
HPVM-to-PTX
HPVM-to-FPGA
Intel Xeon E5 core i7 +AVX
nVidia GeForce GTX 680 GPU
Intel / Altera Arria 10 FPGA

Two key aspects

1. Any node ⟷ Any device
2. Reuse vendor back ends for high performance code-gen

APPROX-HPVM: ACCURACY AWARE OPTIMIZATION

Relaxing accuracy can enable higher efficiency (2X to 50X in our work)

Can we make these techniques easier to use?

Goal 1: Applications should only specify high-level accuracy goals

- Maximum acceptable loss in quality, e.g., inference error, PSNR
- End-to-end metrics, not per function or pipeline stage or …

Goal 2: (Often) Want object-code portability

- Approximation choices are highly system-dependent
- Can make orders-of-magnitude difference in performance, energy

ApproxHPVM: Accuracy-aware representation and optimizations

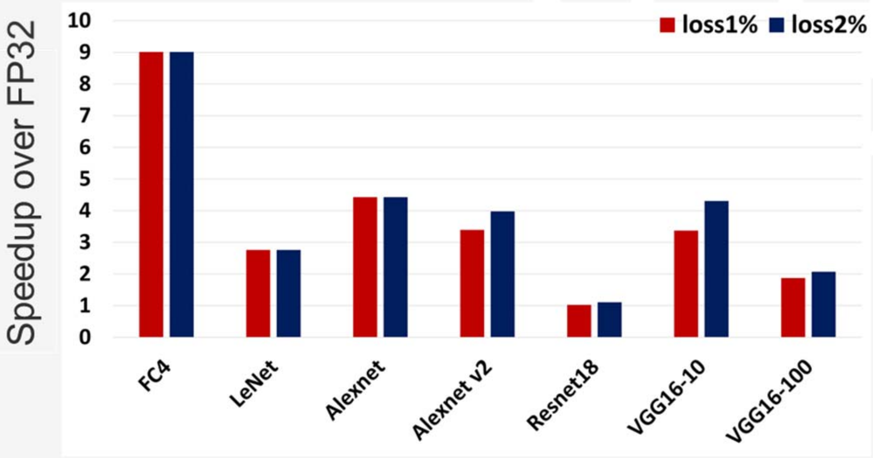
Implemented for Keras: Neural networks in TensorFlow

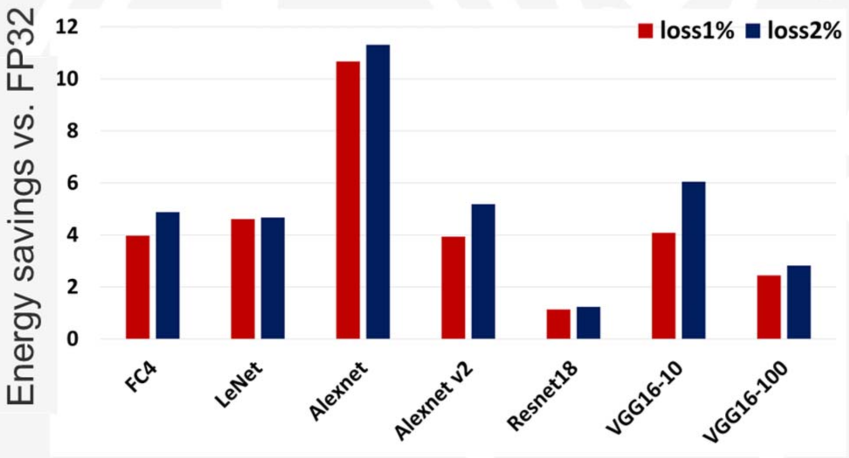
Sharif et al., OOPSLA 2019, accepted with revisions

DNN SPEEDUP AND ENERGY SAVINGS

Target system: NVIDIA Tegra TX2 + PROMISE ML accelerator

TX2: FP32 or FP16, PROMISE: 7 voltage levels in SRAM bitlines





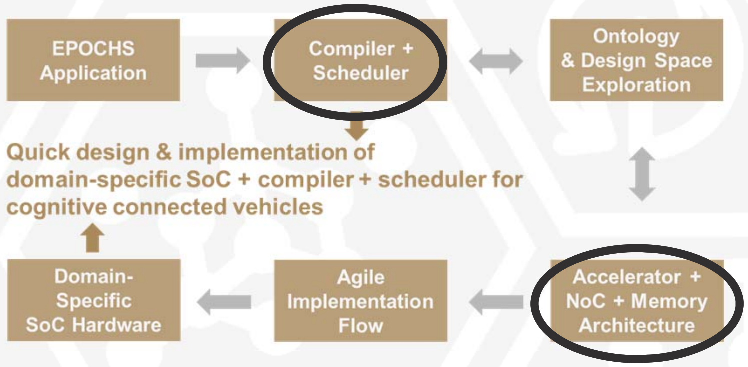
1-2% loss of inference accuracy gives

2X-9X speedup, 2X-11X energy savings in most networks

SUMMARY: HARDWARE & SOFTWARE INTERFACES

Key to heterogeneous system design: Interfaces for hardware & software

- Specializable
- Programmable
- Efficient


EPOCHS Application
Compiler + Scheduler
Ontology & Design Space Exploration
Quick design & implementation of domain-specific SoC + compiler + scheduler for cognitive connected vehicles
Domain-Specific SoC Hardware
Agile Implementation Flow
Accelerator + NoC + Memory Architecture

This talk: Two interfaces

- Spandex: Accelerator communication interface
- Heterogeneous Parallel Virtual Machine (HPVM): Hardware-software interface

Enable specialization with programmability and efficiency

ERI ELECTRONICS RESURGENCE INITIATIVE

SUMMIT

2019 | DETROIT, MI | **JULY 15-17**