



SNAX

An Open-Source HW-SW Codesign
Framework for Heterogeneous Multi-Accelerator
Compute Clusters

*Joren Dumoulin, Ryan Antonio, Xiaoling Yi, Josse Van Delm, Chao Fang,
Guilherme Paim and Marian Verhelst*

joren.dumoulin@kuleuven.be





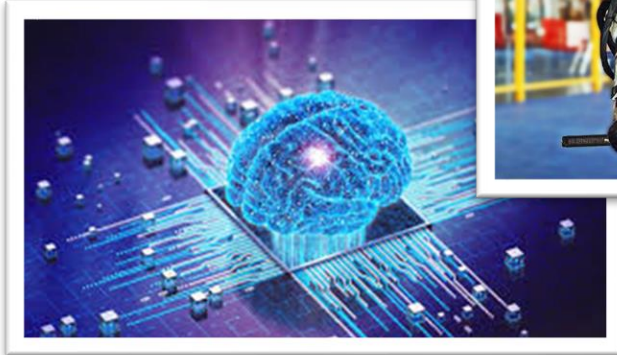
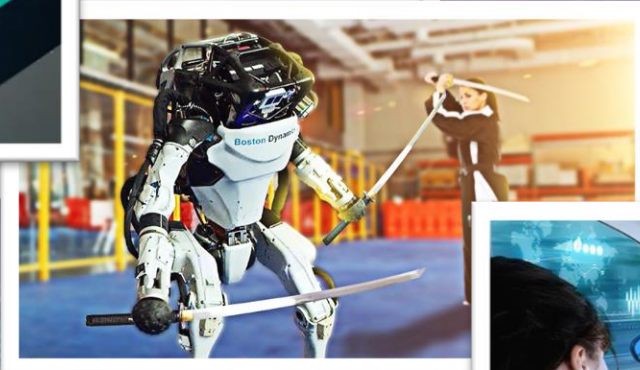
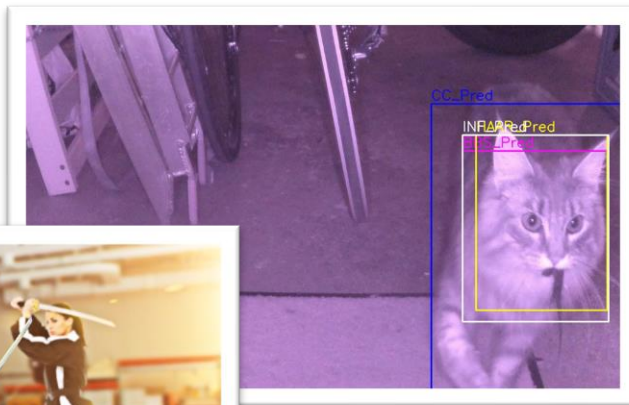
Outline

- **Introduction**
- SNAX Framework
 - SNAX Cluster
 - SNAX-MLIR
- Early Results
- Conclusion





Obligatory "AI is taking over the world" slide



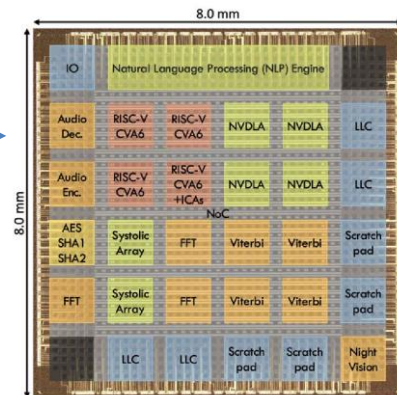
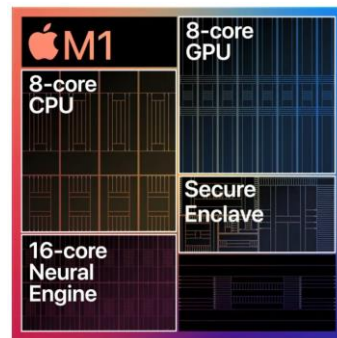



Software is getting very complex





Hardware is getting specialized and heterogeneous

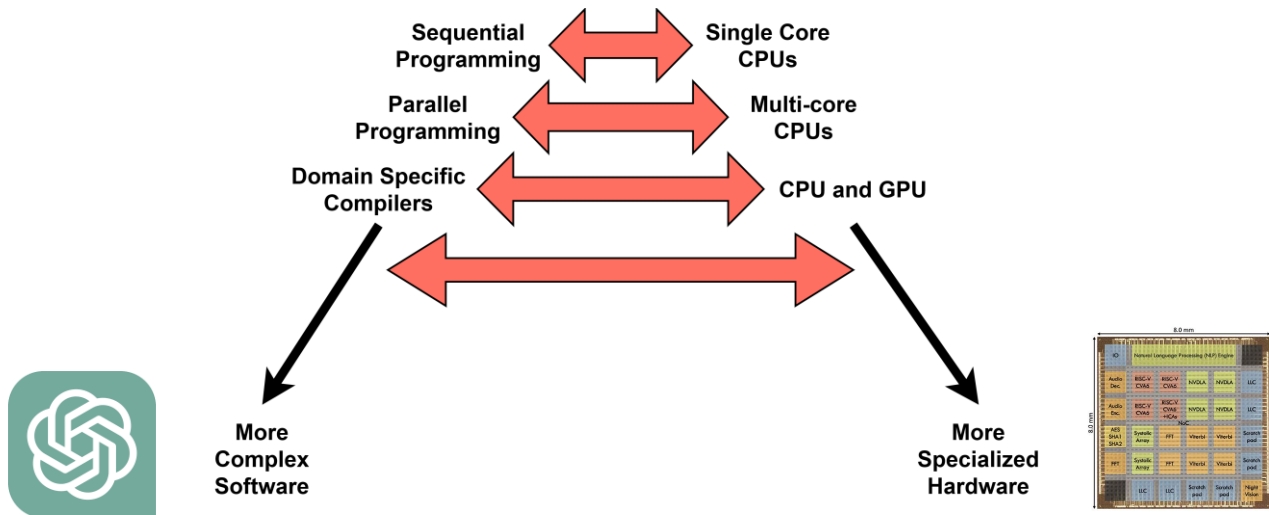


- From single-core CPU
- To multi-core CPU
- To CPU + GPU
- To heterogeneous CPU + GPU + NPU
- To 34-tile 14 accelerator type SoCs! 

M. C. Dos Santos *et al.*, "14.5 A 12nm Linux-SMP-Capable RISC-V SoC with 14 Accelerator Types, Distributed Hardware Power Management and Flexible NoC-Based Data Orchestration," *2024 IEEE International Solid-State Circuits Conference (ISSCC)*, San Francisco, CA, USA, 2024, pp. 262-264



Deploying complex software on specialized hardware



How to bridge this gap?

Open-Source Design Frameworks!



Existing frameworks

High Level SW
Abstractions

SW-HW Design Spectrum


Low Level
HW Design

 **Bifurcation !**

Software Frameworks

- ↳ *Compilation to Primitives*
- ↳ *Hardware Agnostic*
- ↳ *GPU & CPU Centric*

Hardware Frameworks

- ↳ *Non Unified Control Interface*
- ↳ *C-based Programming*
- ↳ *inline .asm magic* 



IREE



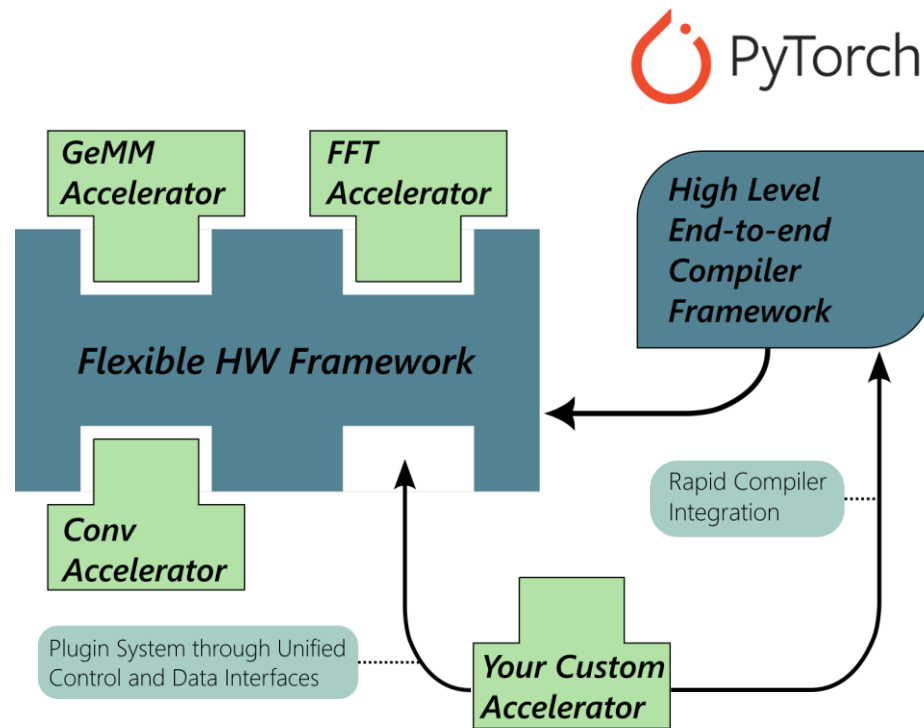
OpenXLA





What is Missing?

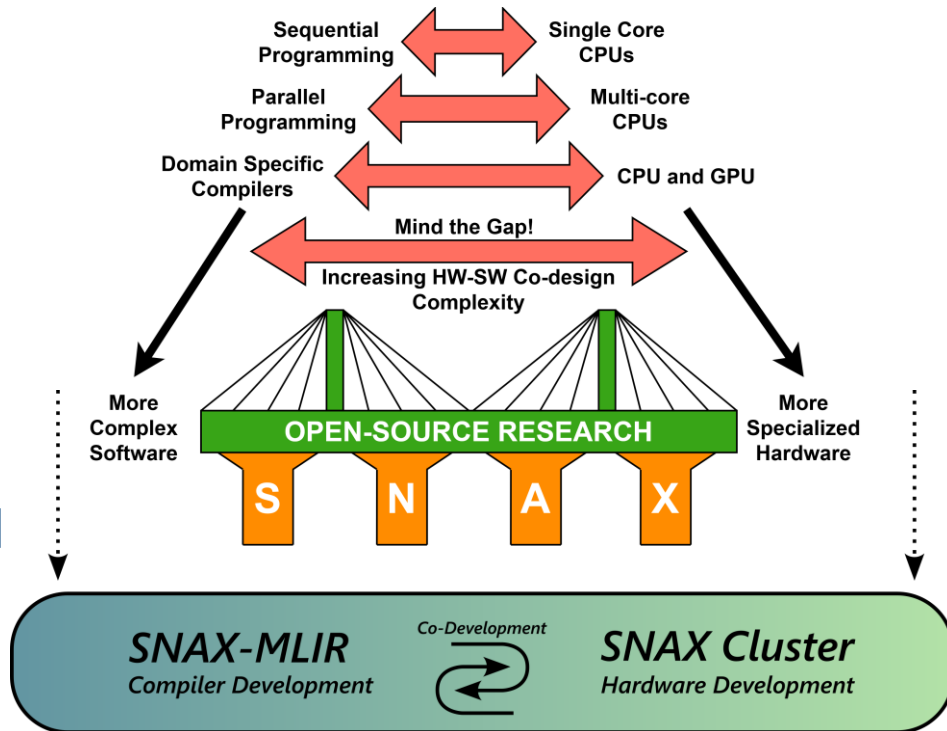
- Integrated HW & SW framework with end-to-end support
- Rapid plugin system in both Hardware and Software





SNAX To The Rescue

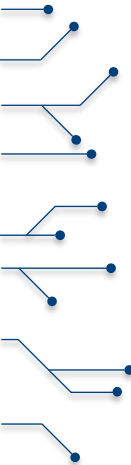
- SNAX Framework
 - Open-Source Design Framework
 - Spanning the entire HW-SW spectrum
 - Co-Development of Hardware Framework and Compiler Design





Outline

- Introduction
- SNAX Framework
 - **SNAX Cluster**
 - SNAX-MLIR
- Early Results
- Conclusion

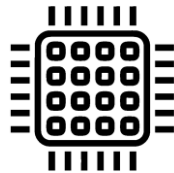




SNAX Hardware – What are we looking for

SNAX Cluster Design Requirements

1 Single-Core
to Manycore



2 Heterogeneous
Systems



3 High
Efficiency



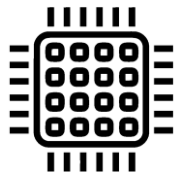
4 Uniformly
Programmable



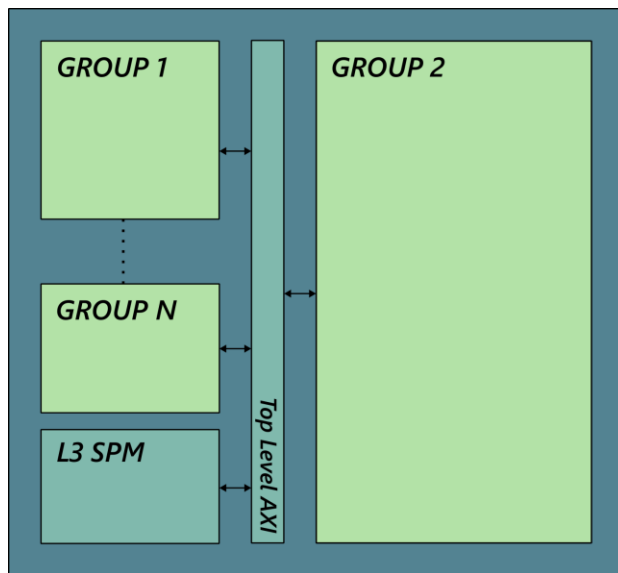


Realizing the SNAX Cluster

1 Single-Core
to Manycore



Achieving Flexible, Scalable
Manycore systems with PULP!
➔ starting from snitch cluster

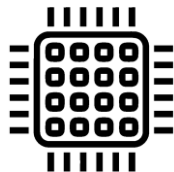


https://github.com/pulp-platform/snitch_cluster



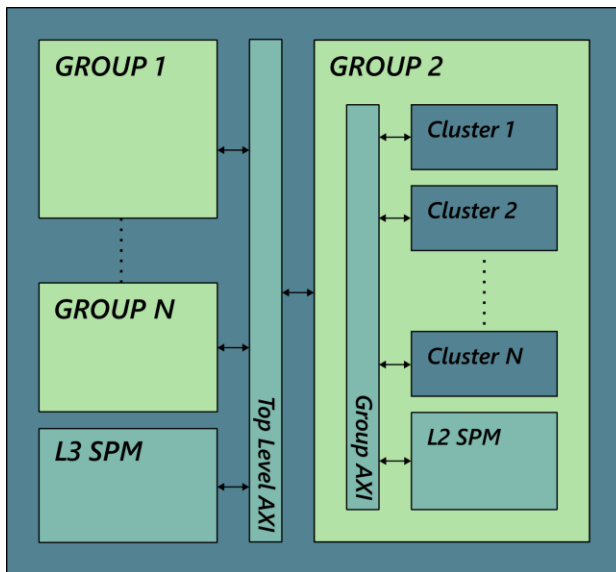
Realizing the SNAX Cluster

1 Single-Core to Manycore



Achieving Flexible, Scalable Manycore systems with PULP!

➔ starting from snitch cluster

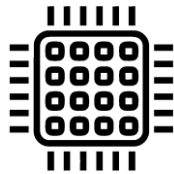


https://github.com/pulp-platform/snitch_cluster



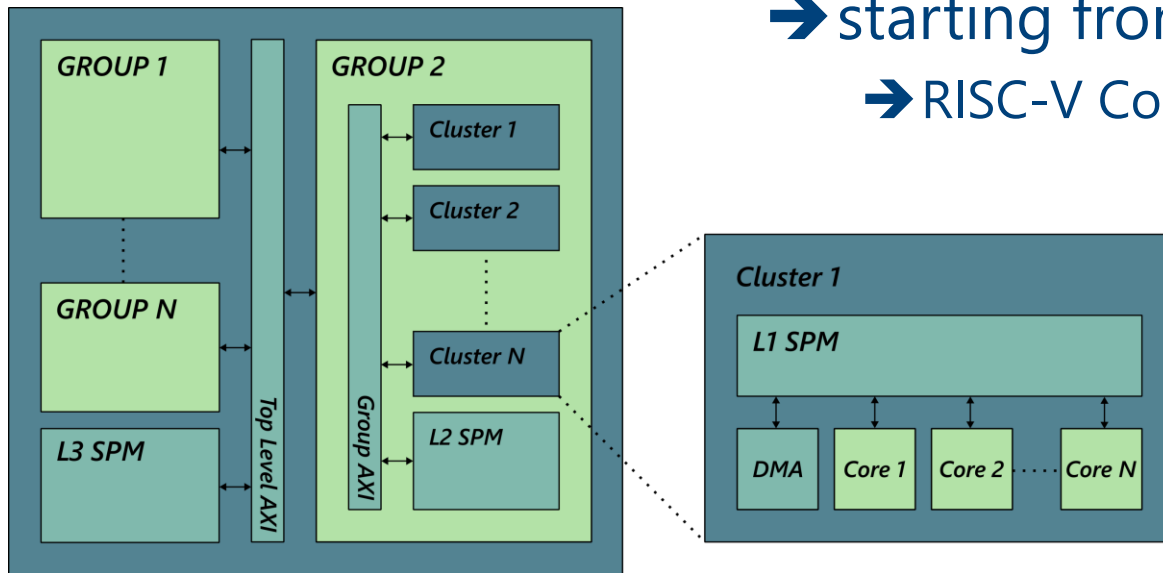
Realizing the SNAX Cluster

1 Single-Core to Manycore



Achieving Flexible, Scalable Manycore systems with PULP!

- starting from snitch cluster
- RISC-V Cores



https://github.com/pulp-platform/snitch_cluster

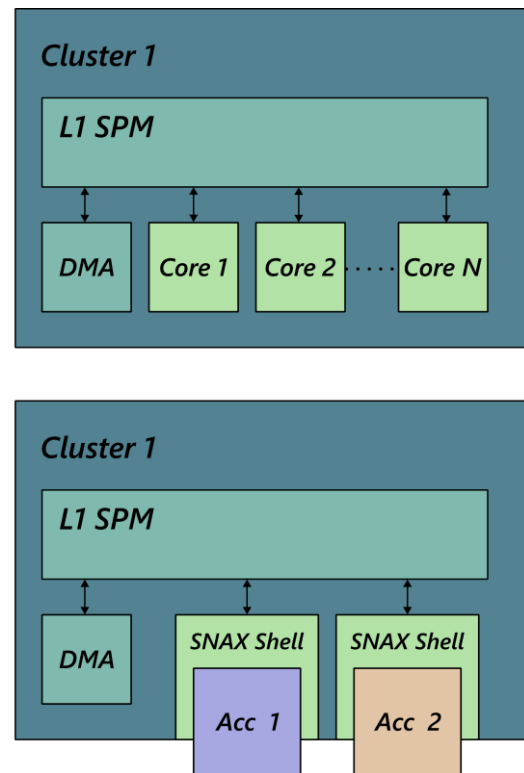


Realizing the SNAX Cluster

2 Heterogeneous Systems



- From Homogenous RISC-V Cores To Heterogeneous Accelerators
- Efficiency and Programmability through SNAX Shell
- SNAX = **Sn**itch **A**ccelerator **eX**tension!



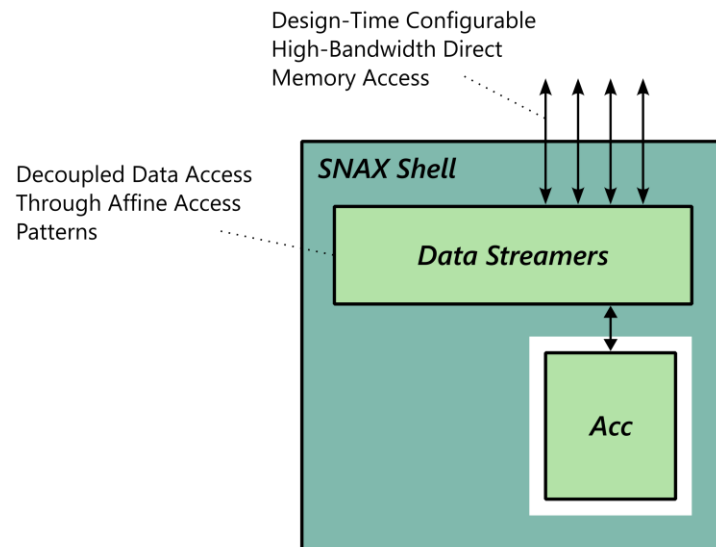


Realizing the SNAX Cluster

Efficiency through tight data coupling:

- Efficiency follows from efficient data movement & access.
- Decoupled direct data access through standard data streamers
- Connection to memory through configurable high-bandwidth connections

3 High Efficiency



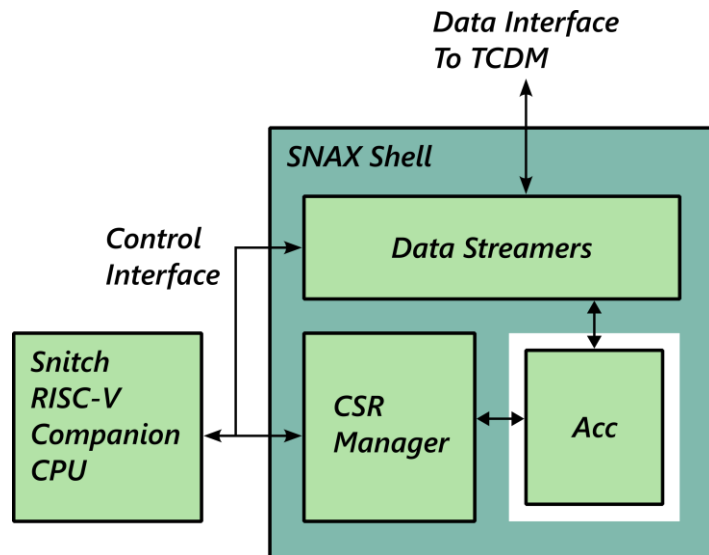


Realizing the SNAX Cluster

Programmability through loose control coupling:

- Snitch Companion CPU
- Decoupled Control through CSR Manager
- Decoupled Access
- Uniform, reusable components

4 Uniformly Programmable





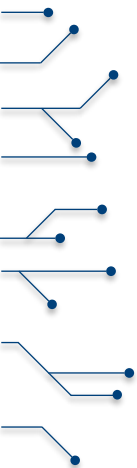
Outline

- Introduction
- SNAX Framework
 - SNAX Cluster
 - **SNAX-MLIR**
- Early Results
- Conclusion

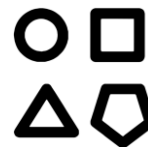




SNAX-MLIR Design Requirements



1 Diverse Accelerator Support



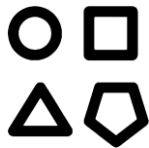
2 End-to-end Capabilities





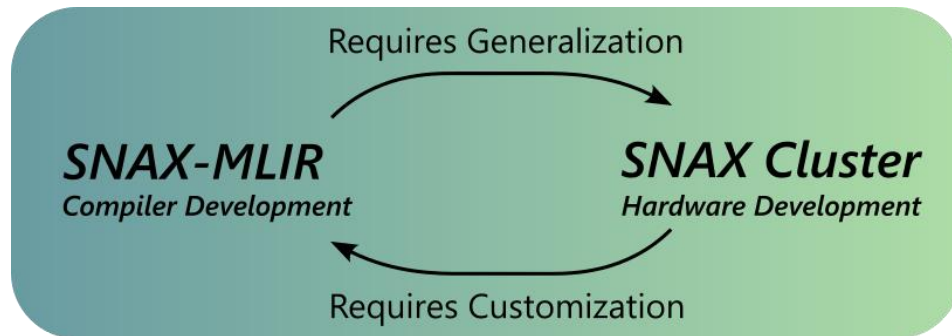
SNAX-MLIR For Wide Accelerator Support (draft figures)

1 Diverse Accelerator Support



To Achieve Diverse Accelerator Support:

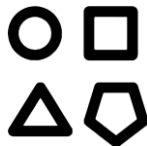
- Generalization
- Customization





SNAX-MLIR For Wide Accelerator Support (draft figures)

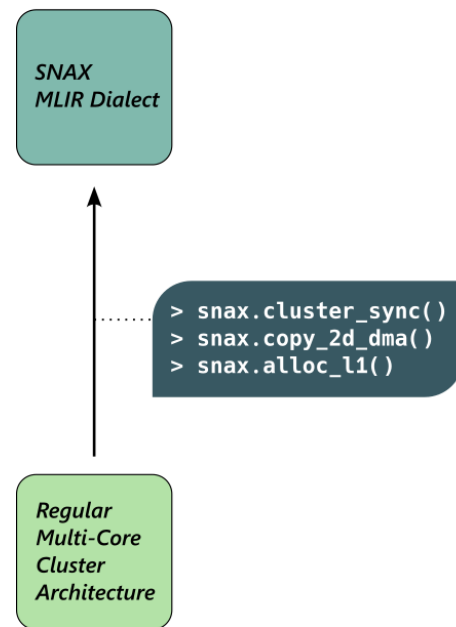
1 Diverse Accelerator Support



1. Raise Generalizations to the Compiler Abstractions

- Multi-Core Cluster Architecture

SNAX-MLIR Abstraction

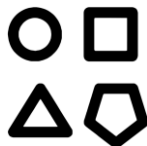


SNAX Cluster Generalization



SNAX-MLIR For Wide Accelerator Support (draft figures)

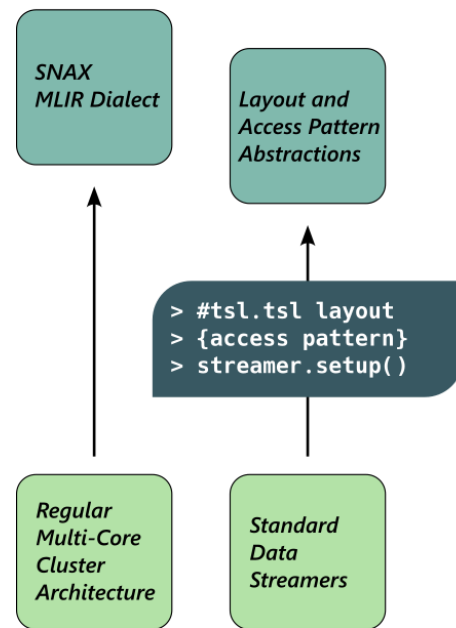
1 Diverse Accelerator Support



1. Raise Generalizations to the Compiler Abstractions

- Multi-Core Cluster Architecture
- SNAX Shell Components:
 - Data Streamers

SNAX-MLIR Abstraction

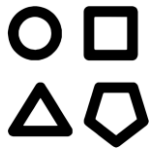


SNAX Cluster Generalization



SNAX-MLIR For Wide Accelerator Support (draft figures)

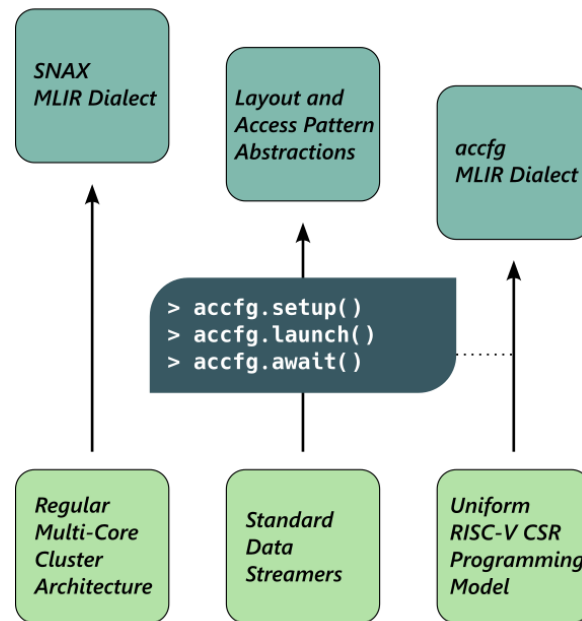
1 Diverse Accelerator Support



1. Raise Generalizations to the Compiler Abstractions

- Multi-Core Cluster Architecture
- SNAX Shell Components:
 - Data Streamers
 - RISC-V CSR Controller

SNAX-MLIR Abstraction

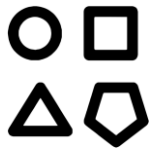


SNAX Cluster Generalization



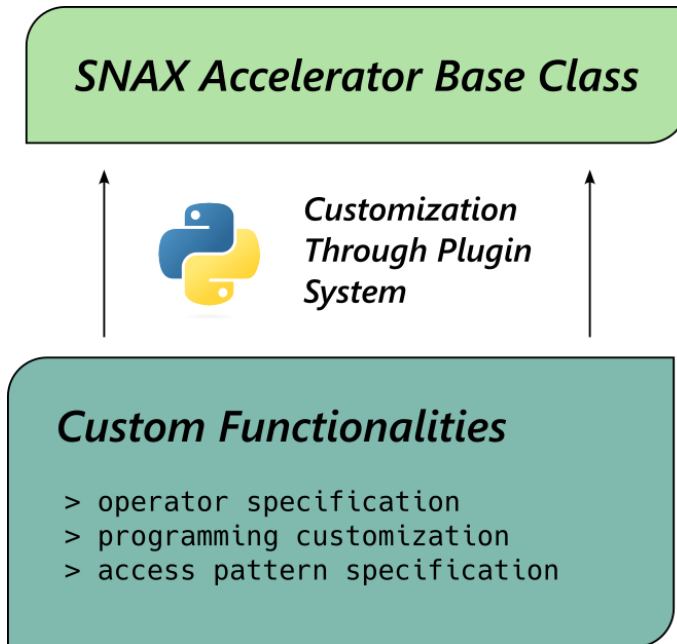
SNAX-MLIR For Wide Accelerator Support (draft figures)

1. Diverse Accelerator Support



2. Customization through Plugin Systems

- SNAX Accelerator Abstraction
- Hardware Developer Friendly Plugin System for Rapid Customization



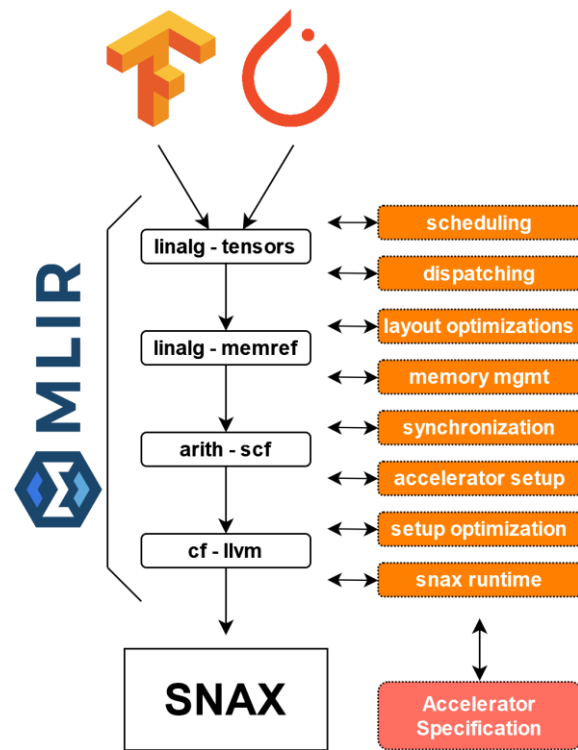


SNAX-MLIR For End-To-End ML Model Deployment on Accelerator Hardware

1 End-to-end Capabilities

MLIR Compiler Framework:

- **End-to-end**: available frontends for many languages and ML frameworks
- **Extensible** through custom dialects
- **Multi-level** for effective high-level transformations





Outline

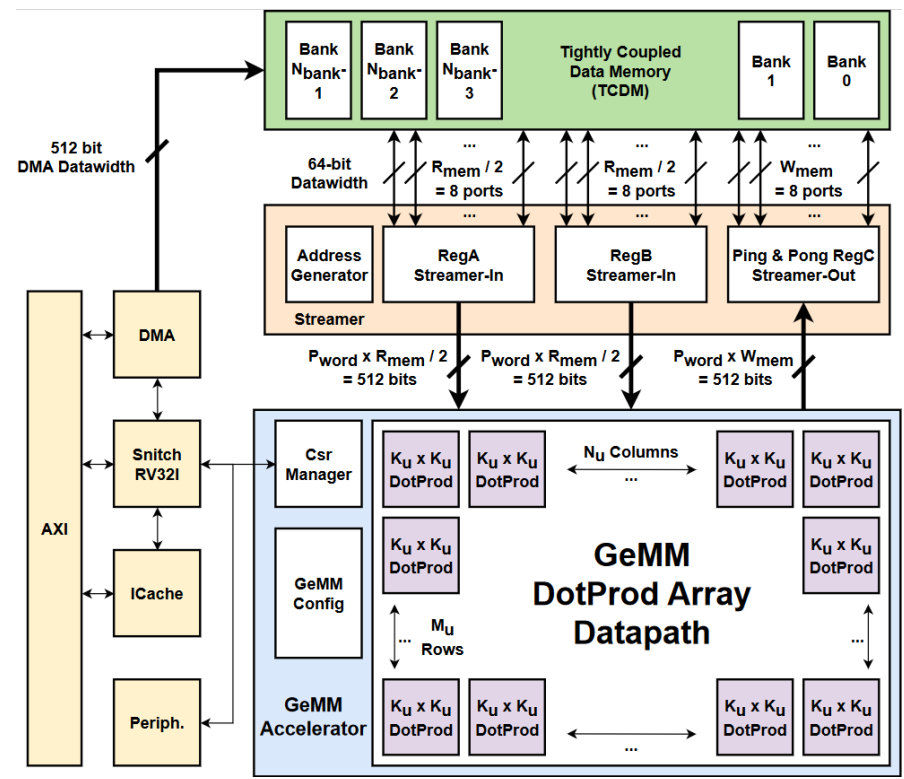
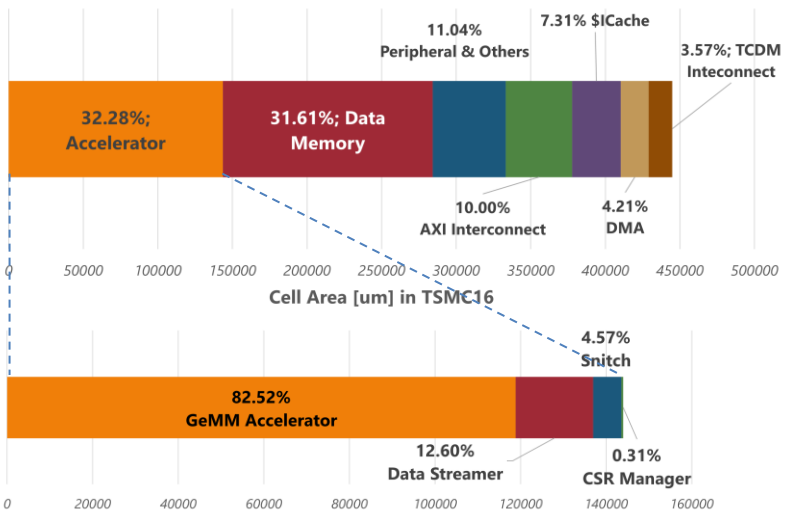
- Introduction
- SNAX Framework
 - SNAX Cluster
 - SNAX-MLIR
- **Early Results**
- Conclusion





GeMM Accelerator Case Study

Implementation of GeMM Accelerator into the SNAX Cluster through the SNAX Shell





GeMM Accelerator Case Study

Compilation from high-level representation

Automatically handled:

- Cluster Data Movements
- Multi-Core Synchronization
- Data Layout Optimizations
- On-the-fly layout transformations

=> >90% utilization on wide variety of matrix dimensions

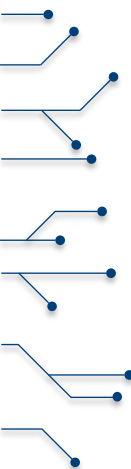
```
func.func @simple_matmul (  
    %arg0: memref<64x64xi8>,  
    %arg1: memref<64x64xi8>,  
    %arg2: memref<64x64xi8>  
) {  
    linalg.matmul ins(%arg0, %arg1) outs(%arg2) -> ()  
}
```





Outline

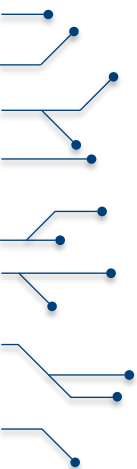
- Introduction
- SNAX Framework
 - SNAX Cluster
 - SNAX-MLIR
- Early Results
- **Conclusion**





Conclusion Slide

- SNAX aims to be a **design framework** for **heterogeneous multi-core** accelerator clusters of the **future**.
- Through **co-development** of the HW and SW stack, we aim to create an **end-to-end extendable** framework, allowing for small contributions to be integrated into a full design.





Come and have a look!



[*https://github.com/kuleuven-micas/snax_cluster*](https://github.com/kuleuven-micas/snax_cluster)

[*https://github.com/kuleuven-micas/snax-mlir*](https://github.com/kuleuven-micas/snax-mlir)