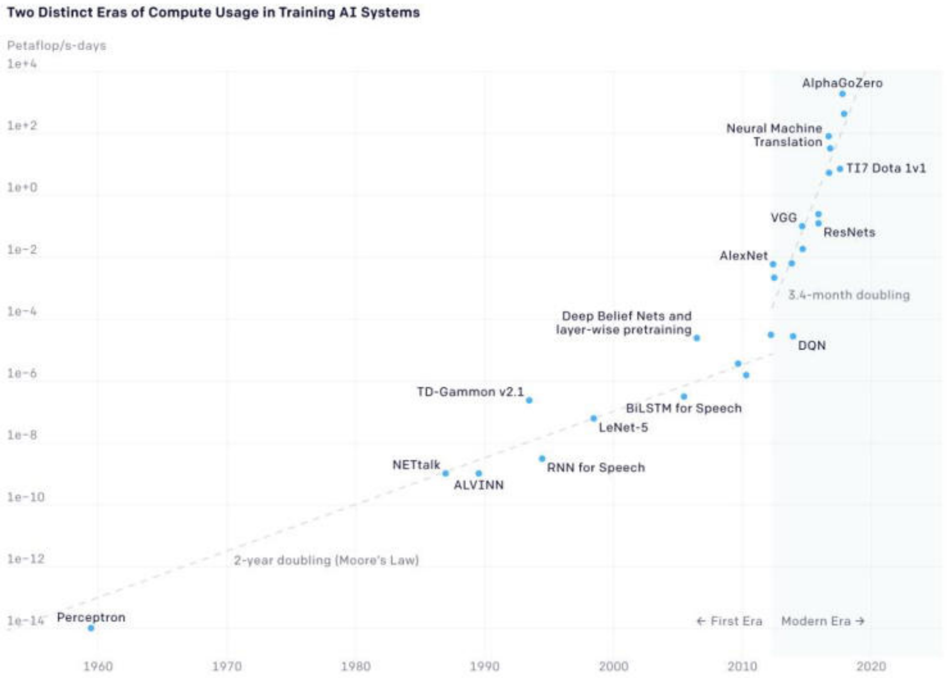
Full-Stack Optimizations for Next-Generation Deep-Learning Accelerators

Sophia Shao
ysshao@berkeley.edu
Electrical Engineering and Computer Sciences

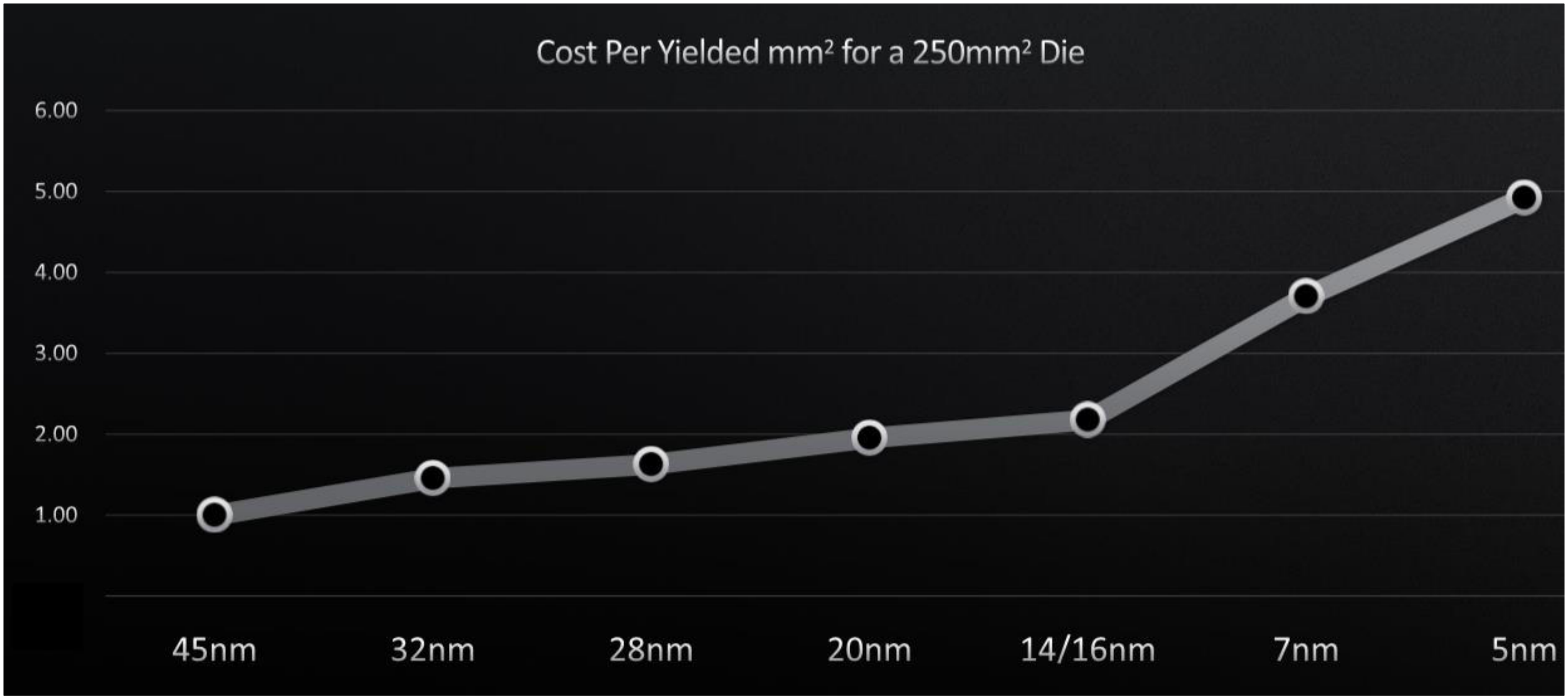
Berkeley
UNIVERSITY OF CALIFORNIA

Growing Demand in Computing



Two Distinct Eras of Compute Usage in Training AI Systems

OpenAI



Slowing Supply in Computing

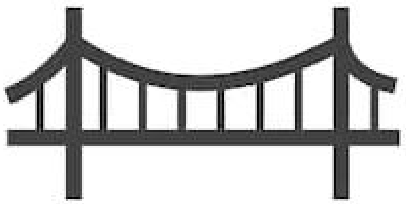
AMD, HotChips, 2019

Growing Demand in Computing + Slowing Supply in Computing = ?

Growing Demand in Computing + Slowing Supply in Computing = 💡

Domain-Specific Accelerators

Growing Demand in Computing

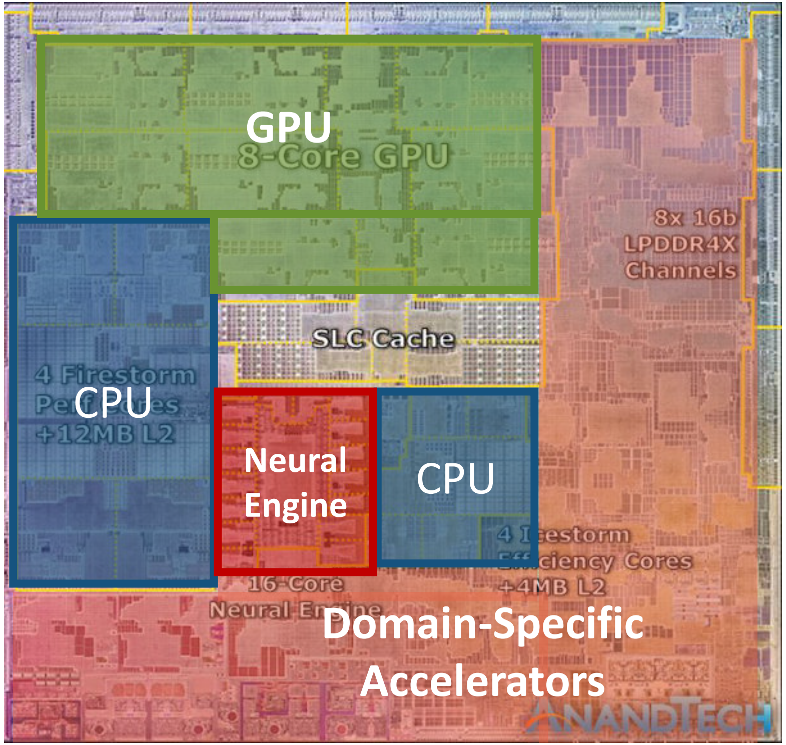


Slowing Supply in Computing

Domain-Specific Accelerators

- Customized hardware designed for a domain of applications.

Apple M1 Chip
2020



* AnandTech

Full-Stack Optimization for DL Accelerators

Design of Accelerators

- MAGNet [ICCAD’2019]
- Simba [MICRO’2019, VLSI’2019]

Integration of Accelerators

- Chipyard [IEEE Micro’2020]
- Gemmini [DAC’2021]

Scheduling of Accelerators

- CoSA [ISCA’2021]

Full-Stack Optimization for DL Accelerators

Design of Accelerators

Integration of Accelerators

Scheduling of Accelerators

Scalable Inference Accelerators

Motivation

- Need for fast and efficient inference accelerators from mobile to datacenter.

Challenge

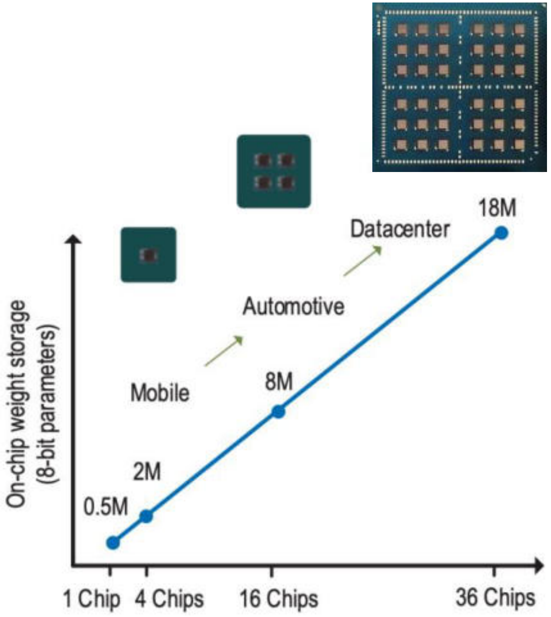
- High design cost of building unique hardware for each design target.

Opportunities

- Deep learning inference is intrinsically scalable with abundant parallelism.
- Recent advances in package-level integration for multi-chip-module-based designs.

The Multi-Chip-Module Approach

- Advantages:
  - Build systems larger than reticle limit
  - Smaller chips are cheaper to design
  - Smaller chips have higher yield
  - Faster time-to-market
- Challenges:
  - Area, energy, and latency for chip-to-chip communication



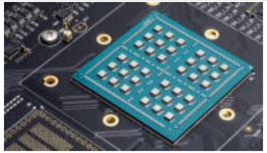
Ref: Zimmer et al., VLSI 2019

Simba: Scaling Inference with MCM-based Architecture

Best Paper Award at MICRO’2019, CACM Research Highlights

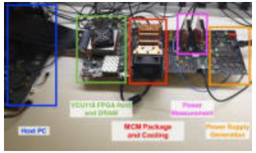
Simba Testchip:

- Package and chiplet architecture
- Processing element design
- Baseline uniform tiling across chiplets and PEs



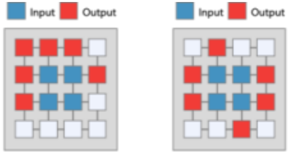
Simba Characterization:

- Comparison with GPUs
- NoP bandwidth sensitivity
- NoP latency sensitivity



Simba NoP-Aware Tiling:

- Non-uniform work partitioning
- Communication-aware data placement
- Cross-layer pipelining



Simba: Scalable MCM-Based Architecture

Package and chiplet spec

6mm^2 chiplet in TSMC 16nm

36 chiplets/package

Chip-to-chip interconnect

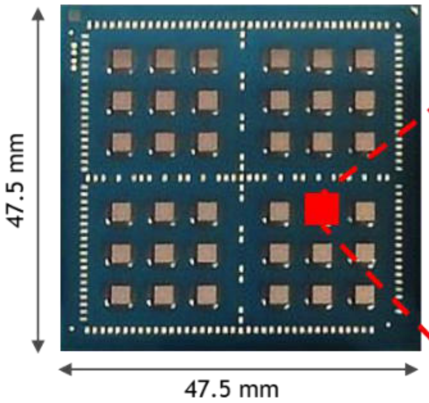
Ground-Referenced Signaling

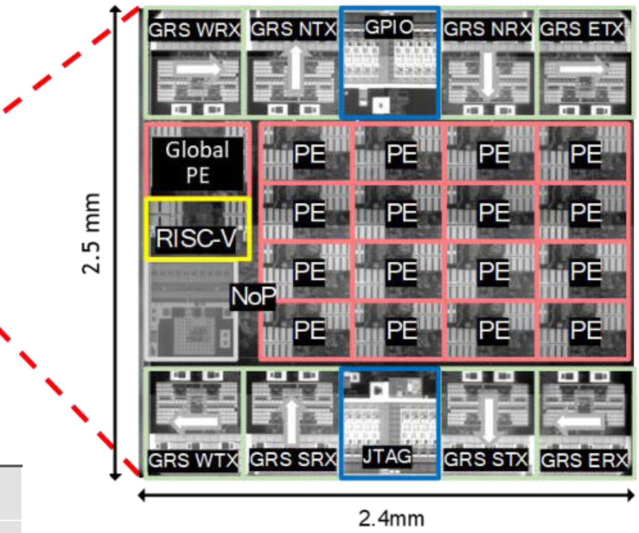
Efficient compute tiles

128 TOPS

0.11 pJ/Op

8-bit integer datapath



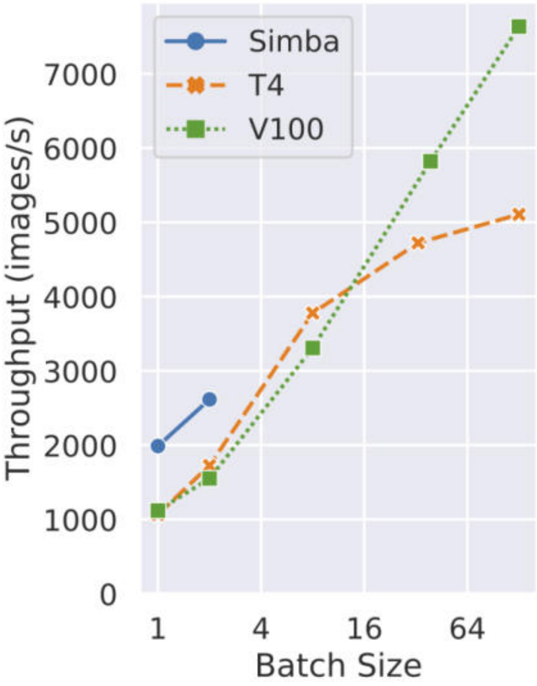


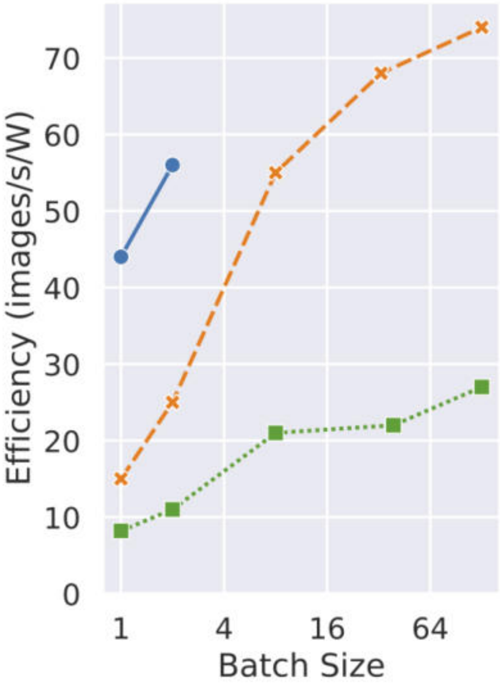
Ref: Zimmer et al., VLSI 2019

Core area	111.6 mm^2
Voltage	0.52-1.1 V
Frequency	0.48-1.8 GHz
SRAM	624KB/chip 23MB/package

Simba Characterization

- Comparison with GPUs running ResNet-50





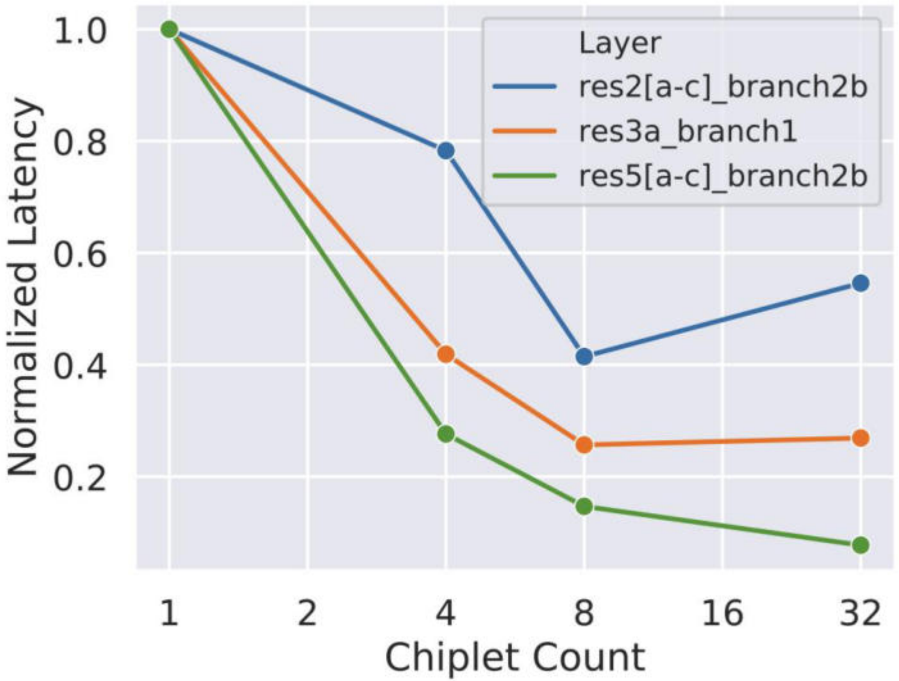
Simba Characterization

- Layer Sensitivity

- Running three ResNet-50 layers across different number of chiplets.

- Increasing the number of active chiplets does not always translate to performance gains.

- The cost of communication hinders the ability to exploit parallelism.



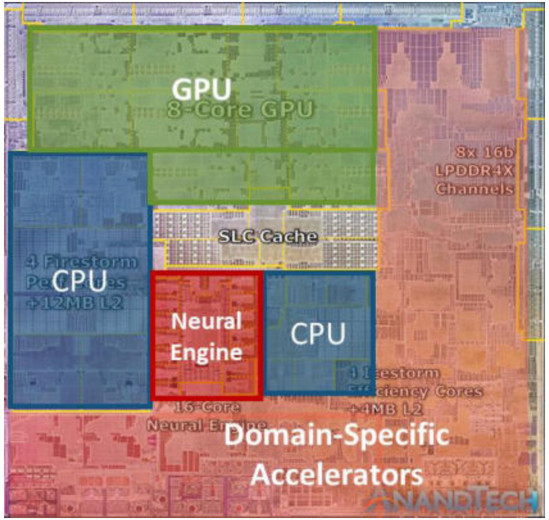
Full-Stack Optimization for DL Accelerators

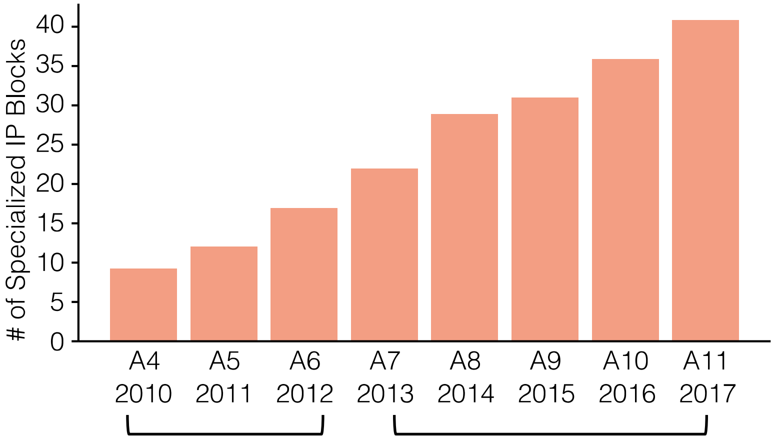
Design of Accelerators

✓ Integration of Accelerators

Scheduling of Accelerators

Accelerators don’t exist in isolation.


GPU
8-Core GPU
8x 16b LPDDR4X Channels
SLC Cache
4 Firestorm Performance Cores +12MB L2
CPU
Neural Engine
CPU
4 Icestorm Efficiency Cores +4MB L2
16-Core Neural Engine
Domain-Specific Accelerators
ANANDTECH



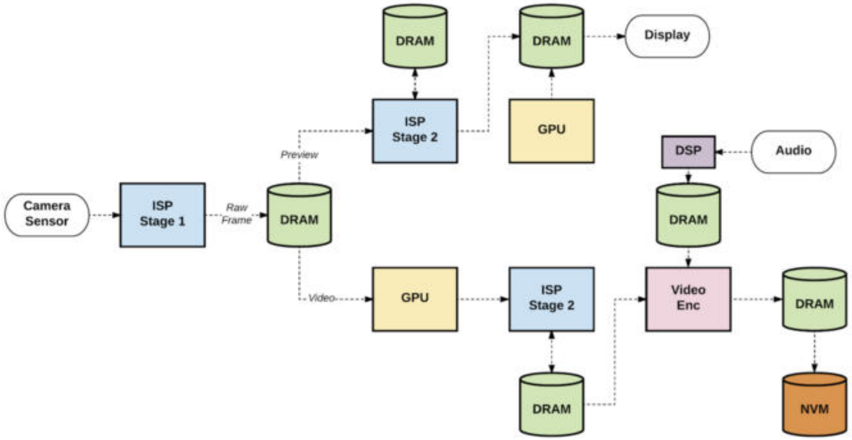
Maltiel consulting estimates

Shao et al. IEEE Micro 2015

http://vlsiarch.eecs.harvard.edu/research/accelerators/die-photo-analysis/

Mobile SoC Usecase

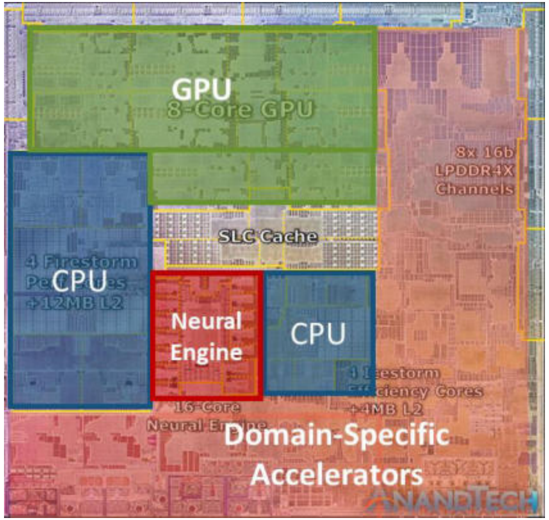
- Mainstream architecture has long focused on general-purpose CPUs and GPUs.
- In an SoC, multiple IP blocks are active at the same time and communicate frequently with each other.
- Example:
  - Recording a 4K video
  - Camera -> ISP
    - “Preview stream” for display
    - “Video stream” for storage
  - DRAM for data sharing

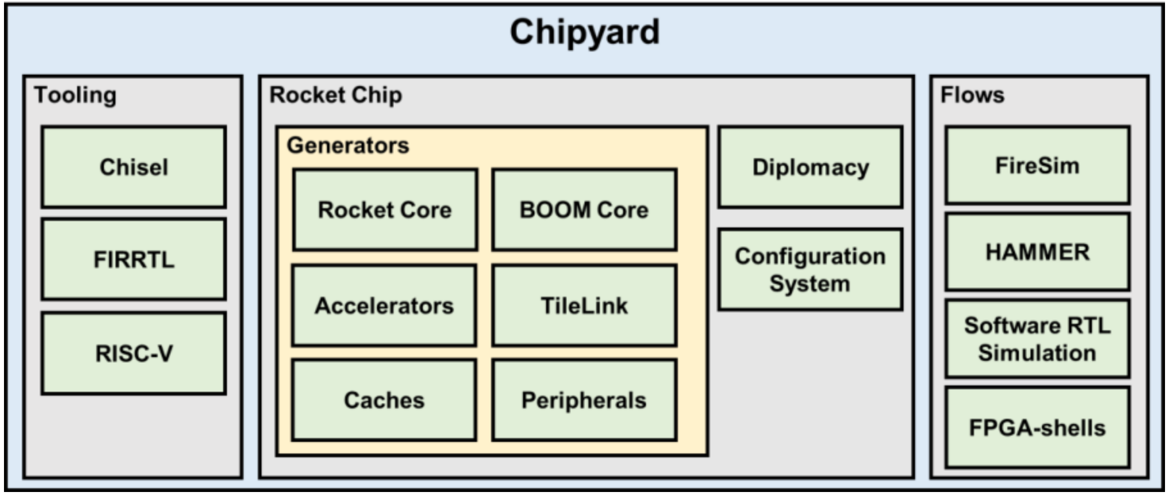


Two Billion Devices and Counting: An Industry Perspective on the State of Mobile Computer Architecture, IEEE Micro’2018

CHIPYARD SoC Framework

- Integrated **design**, **simulation** and **implementation** environment for specialized SoCs.





Chipyard

Tooling
- Chisel
- FIRRTL
- RISC-V

Rocket Chip
- Generators
  - Rocket Core
  - BOOM Core
  - Accelerators
  - TileLink
  - Caches
  - Peripherals
- Diplomacy
- Configuration System

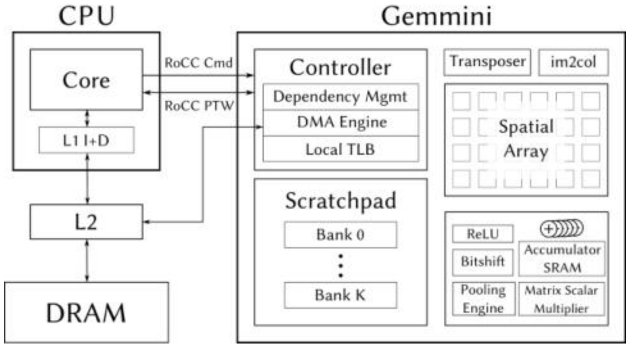
Flows
- FireSim
- HAMMER
- Software RTL Simulation
- FPGA-shells

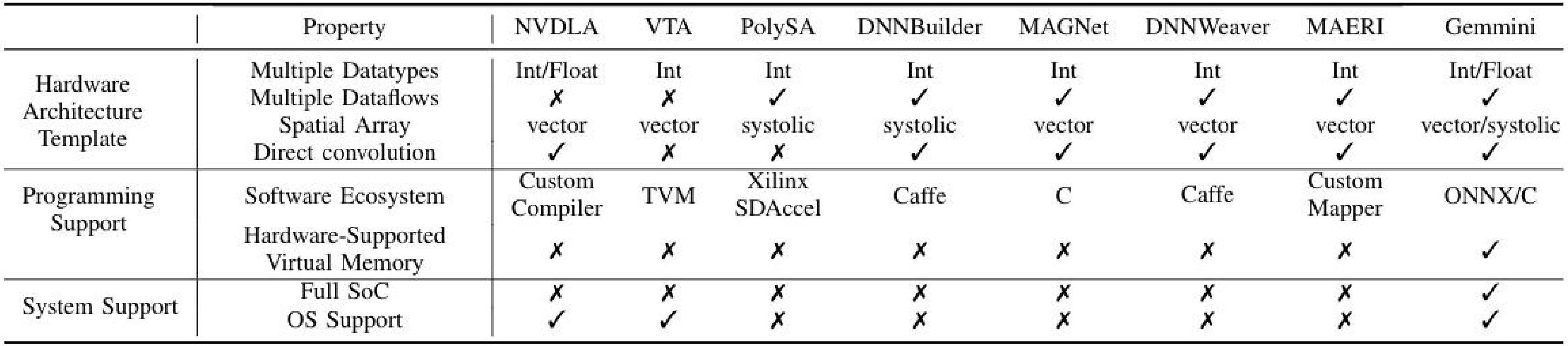
https://github.com/ucb-bar/chipyard

[IEEE Micro’2020]

Gemmini: Full-System Co-Design of Hardware Accelerators

- **Full-stack**
  - Includes OS
  - End-to-end workloads
  - “Multi-level” API
- **Full-SoC**
  - Host CPUs
  - Shared memory hierarchies
  - Virtual address translation



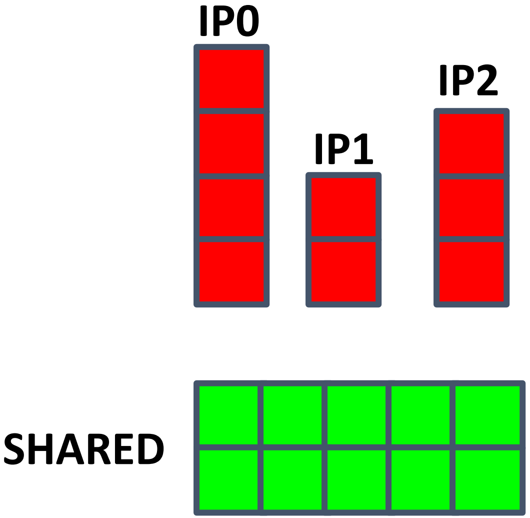


	Property	NVDLA	VTA	PolySA	DNNBuilder	MAGNet	DNNWeaver	MAERI	Gemmini
Hardware Architecture Template	Multiple Datatypes	Int/Float	Int	Int	Int	Int	Int	Int	Int/Float
	Multiple Dataflows	✗	✗	✓	✓	✓	✓	✓	✓
	Spatial Array	vector	vector	systolic	systolic	vector	vector	vector	vector/systolic
	Direct convolution	✓	✗	✗	✓	✓	✓	✓	✓
Programming Support	Software Ecosystem	Custom Compiler	TVM	Xilinx SDAccel	Caffe	C	Caffe	Custom Mapper	ONNX/C
	Hardware-Supported Virtual Memory	✗	✗	✗	✗	✗	✗	✗	✓
System Support	Full SoC	✗	✗	✗	✗	✗	✗	✗	✓
	OS Support	✓	✓	✗	✗	✗	✗	✗	✓

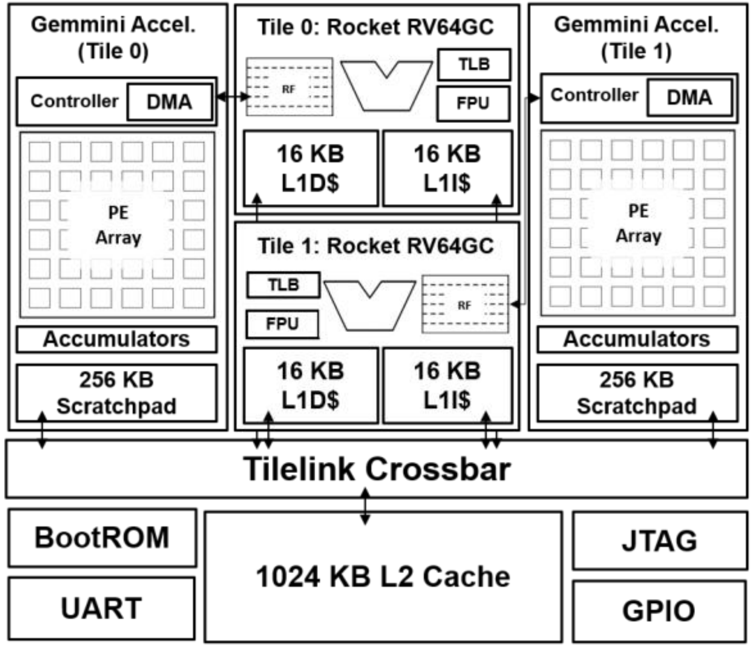
https://github.com/ucb-bar/gemmini

[DAC’2021]

Gemmini Case Study: Allocating on-chip SRAM



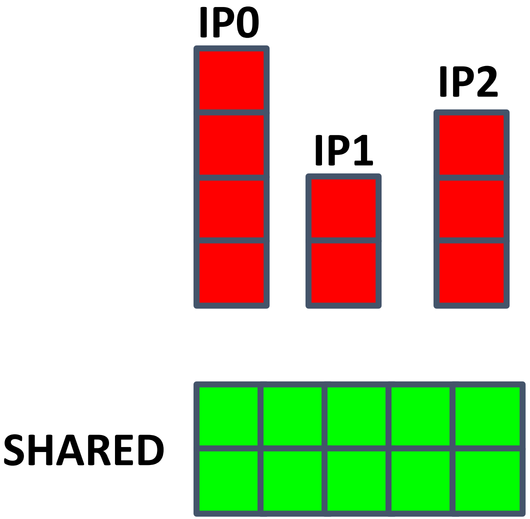
- **Where to allocated SRAM?**
  - Private within each IP
  - Shared



https://github.com/ucb-bar/gemmini

[DAC'2021]

Gemmini Case Study: Allocating on-chip SRAM


IP0

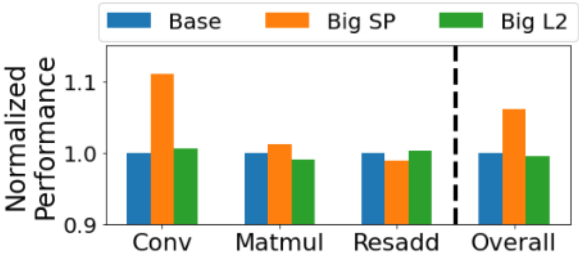
IP1

IP2

SHARED

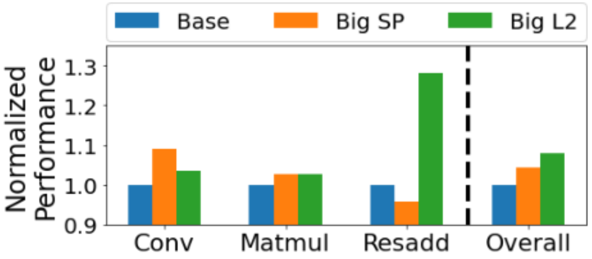
- **Where to allocated SRAM?**
  - Private within each IP
  - Shared

- **Application dependent.**



Single-Core SoC

- **SoC configuration dependent.**



Dual-Core SoC

https://github.com/ucb-bar/gemmini

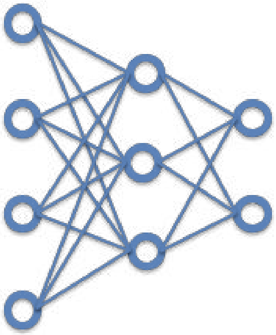
[DAC’2021]

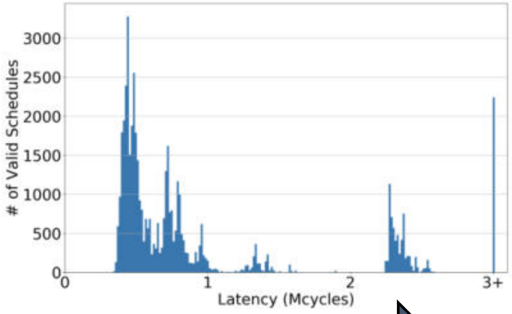
Full-Stack Optimization for DL Accelerators

- Design of Accelerators
- Integration of Accelerators
- Scheduling of Accelerators

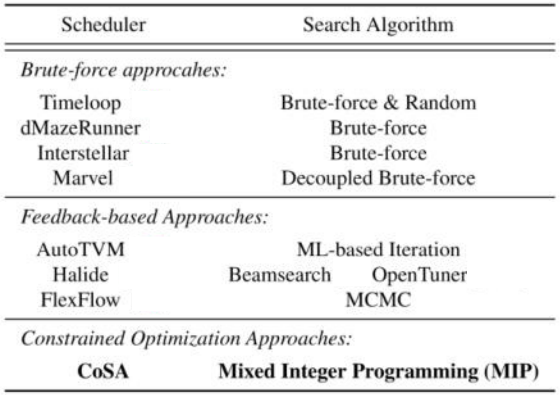
Large Space of Mapping Algorithms to ML Hardware

Algorithm





Scheduling

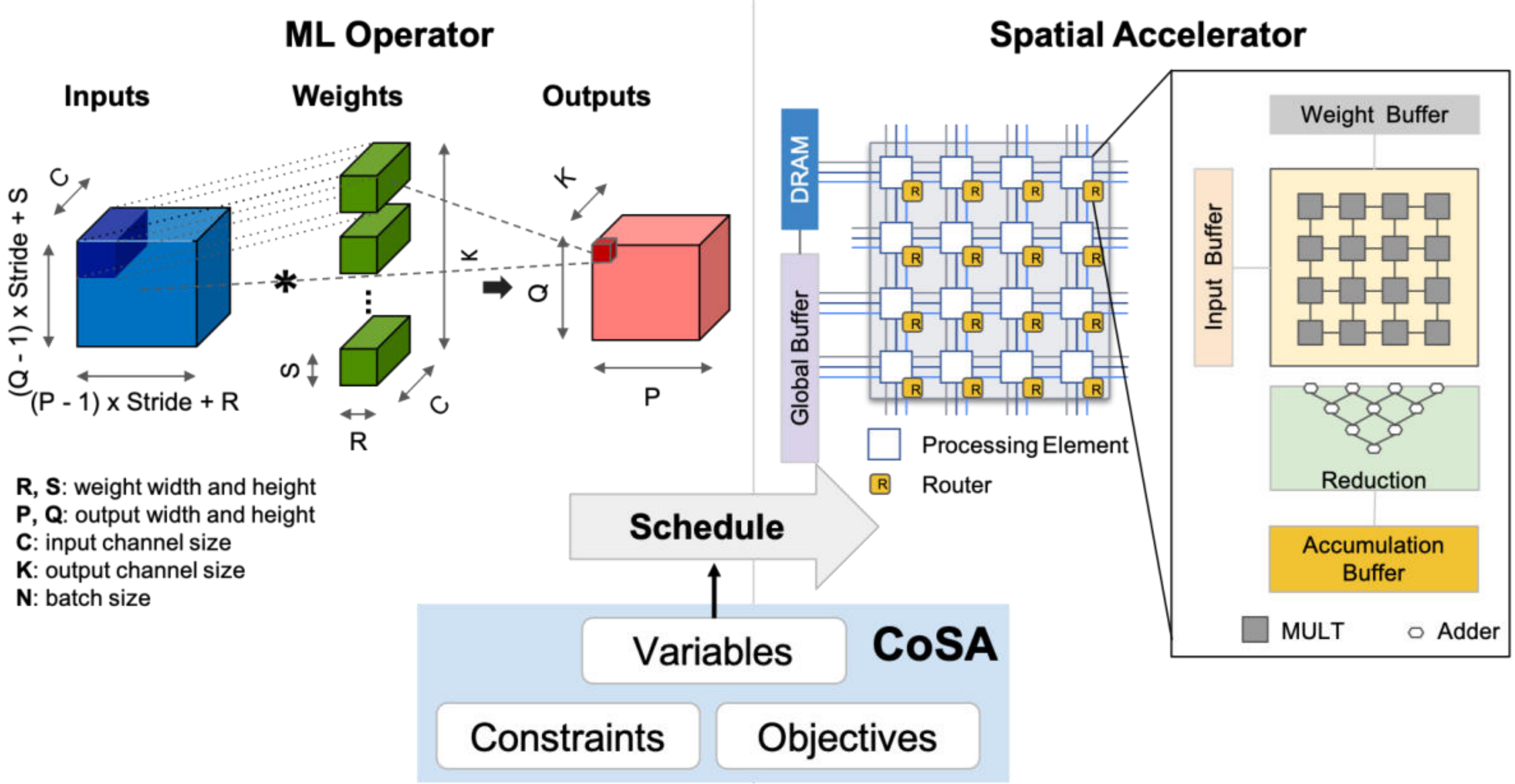


Scheduler	Search Algorithm
Brute-force approcahes:	
Timeloop	Brute-force & Random
dMazeRunner	Brute-force
Interstellar	Brute-force
Marvel	Decoupled Brute-force
Feedback-based Approaches:	
AutoTVM	ML-based Iteration
Halide	Beamsearch OpenTuner
FlexFlow	MCMC
Constrained Optimization Approaches:	
CoSA	**Mixed Integer Programming (MIP)**

Hardware

[ISCA'2021]

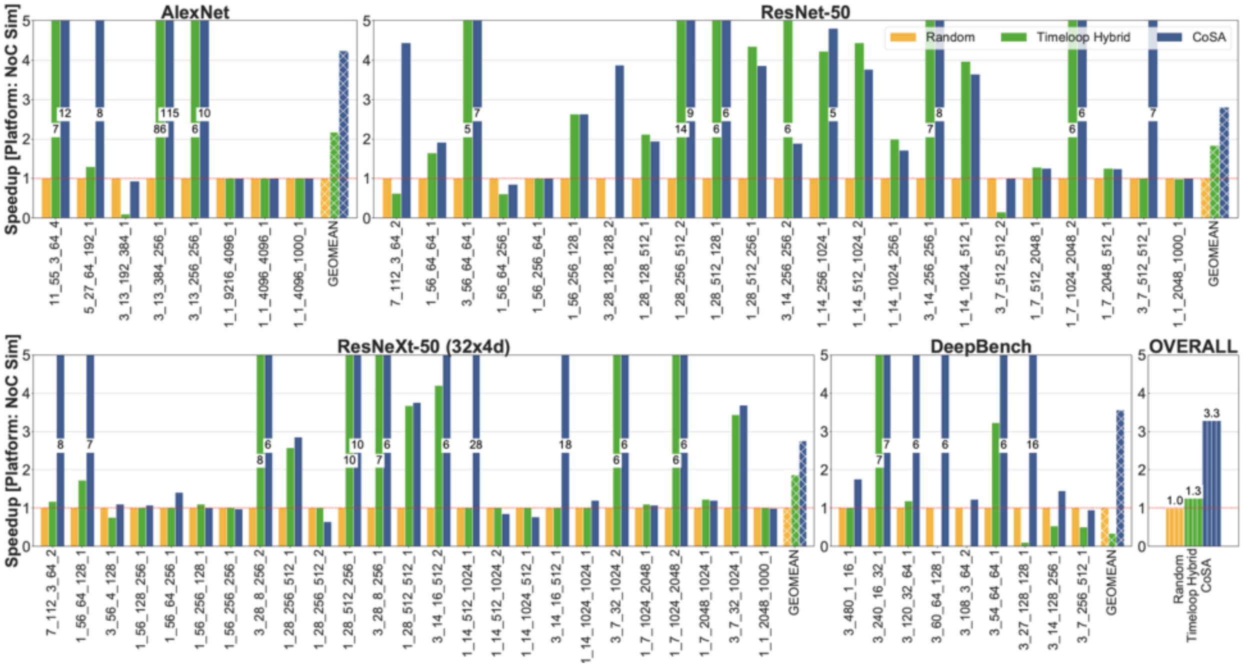
CoSA: Constrained-Optimization for Spatial Architecture



R, S: weight width and height
P, Q: output width and height
C: input channel size
K: output channel size
N: batch size

[ISCA'2021]

CoSA: Constrained-Optimization for Spatial Architecture



2.5x speedup compared to SoTA with 90x faster time-to-solution.

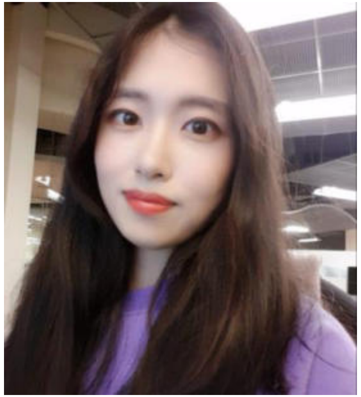
	CoSA	Random (5×)	Timeloop Hybrid
Avg. Runtime / Layer	4.2s	4.6s	379.9s
Avg. Samples / Layer	1	20K	67M
Avg. Evaluations / Layer	1	5	16K

[ISCA'2021]

Acknowledgement

Hasan Genc

Jenny Huang



Seah Kim

- Thanks collaborators from UC Berkeley and NVIDIA!

Full-Stack Optimization for DL Accelerators

- Design of Accelerators
- Integration of Accelerators
- Scheduling of Accelerators