Spatial Accelerator at the Edge

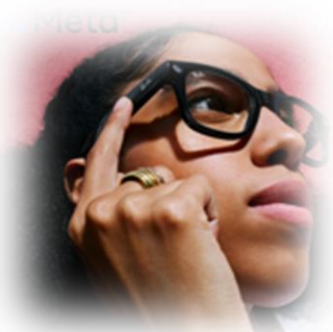
Tulika Mitra

Department of Computer Science
National University of Singapore

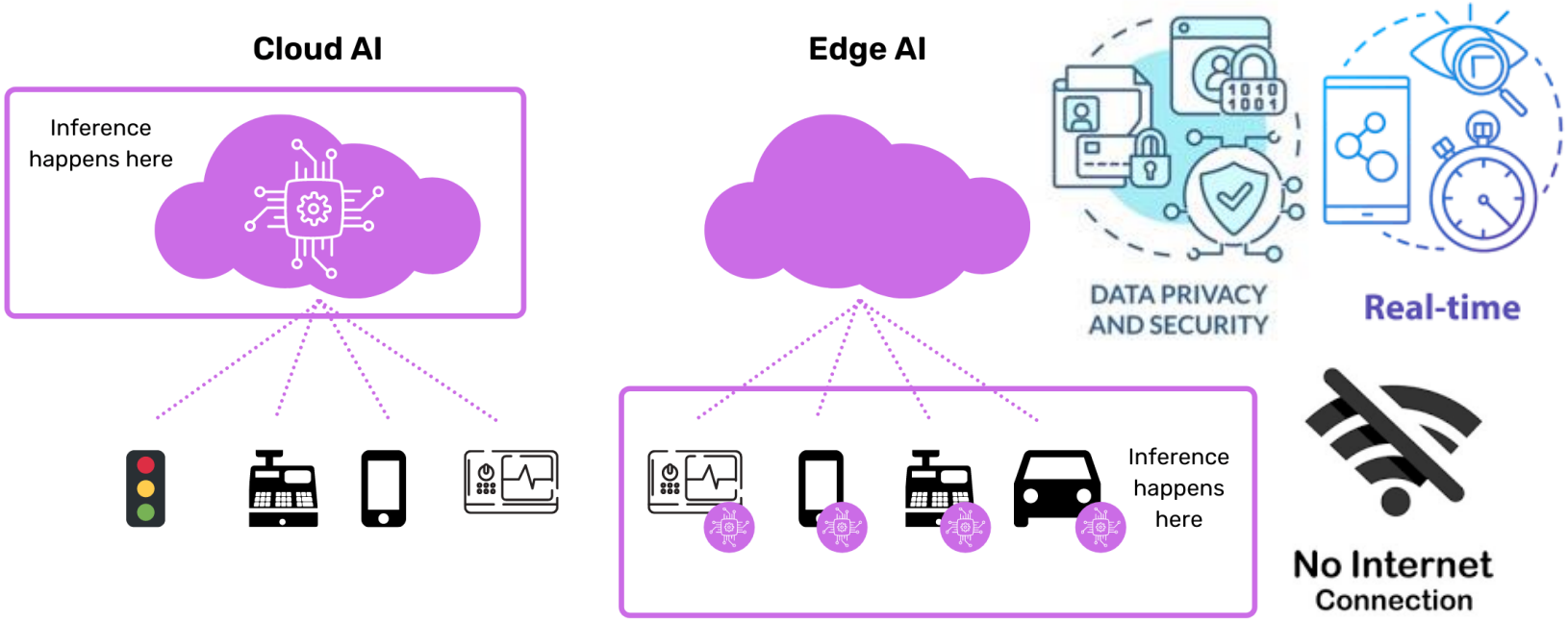
NANDA Workshop, September 2024

Edge Intelligence

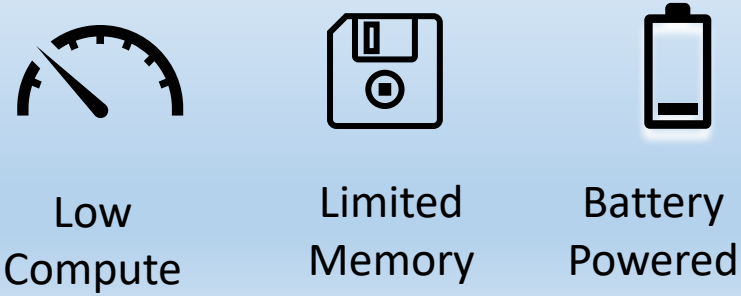




Real-time data processing and response locally on-device at the network's edge



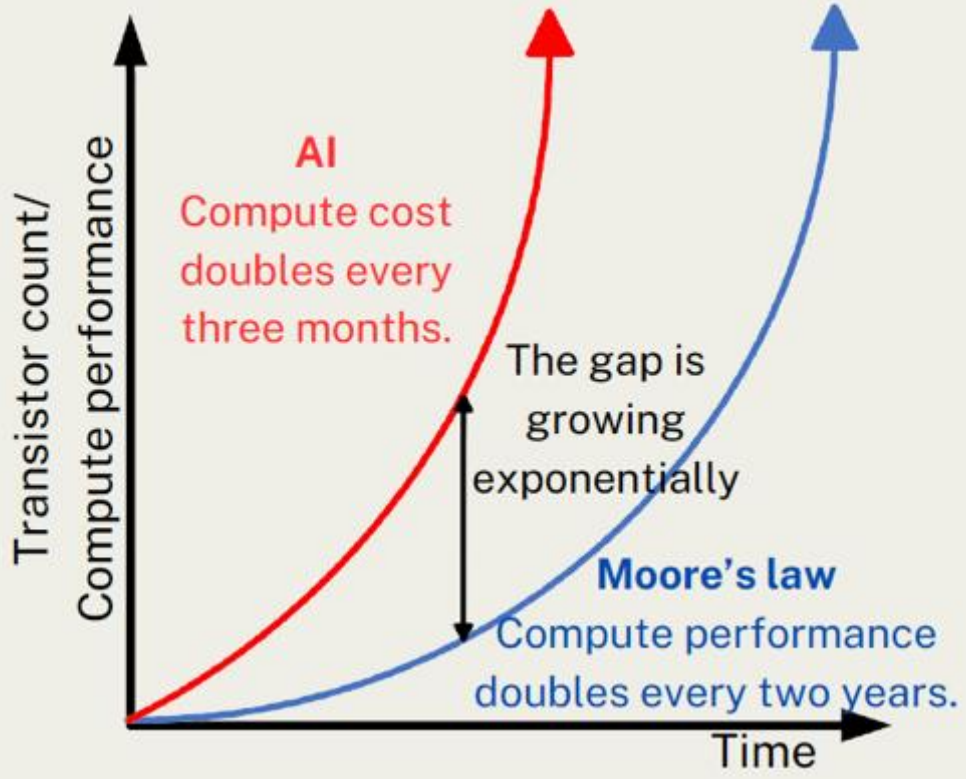
Edge Devices are resource constrained



Edge Intelligence Challenge



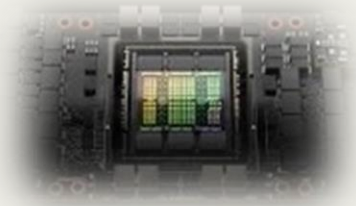
AI compute gap



Short battery life

Power-hungry applications drain battery life very quickly.

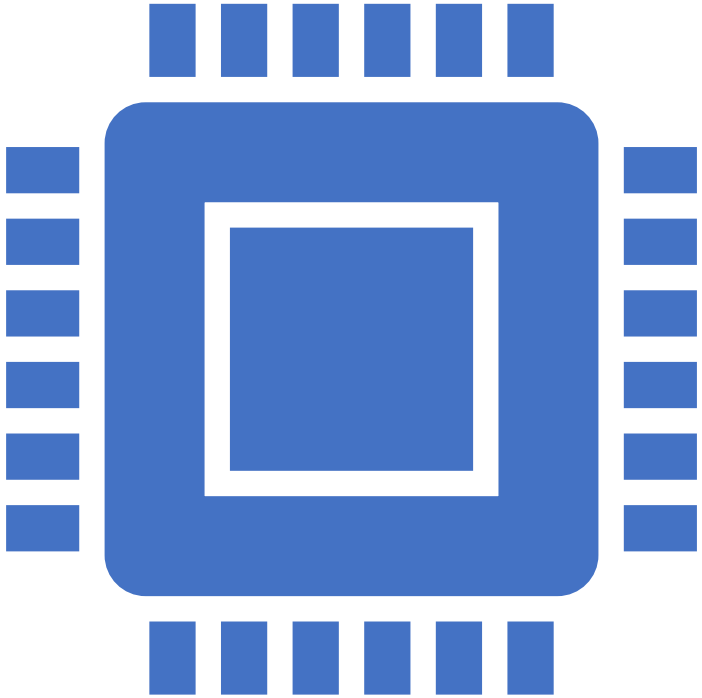

AI power gap


GPU
700W


Brain
20W

Watch
1W

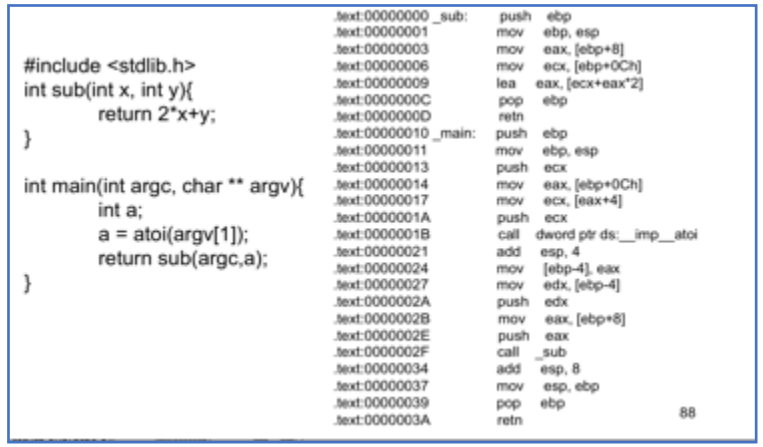
NUS National University of Singapore



Architecture-Compiler Codesign for < 1W Power

Microprocessor: von Neumann Model of Computation

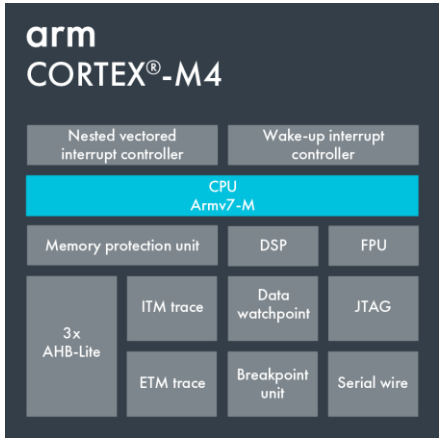
- High programmability: Supports all application
- **Inherently sequential:** Burden is on hardware to extract parallelism
  - High energy overhead and limited parallelism
- 1980-2005: Unprecedented performance growth of microprocessors due to Moore's Law, Dennard Scaling, and micro-architectural innovations
- **Hardware-driven parallelism extraction is too costly for edge devices**



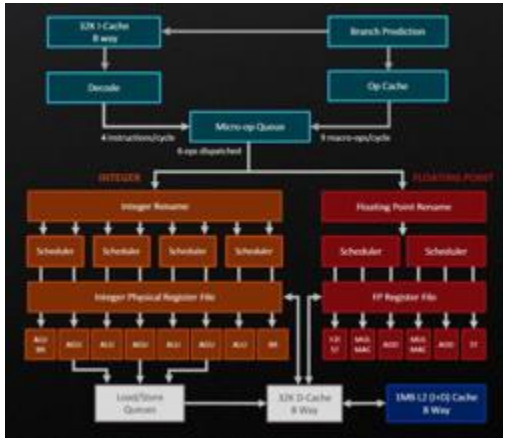
```
#include <stdlib.h>
int sub(int x, int y){
        return 2*x+y;
}

int main(int argc, char ** argv){
        int a;
        a = atoi(argv[1]);
        return sub(argc,a);
}
```

```
.text:00000000 _sub:    push   ebp
.text:00000001          mov    ebp, esp
.text:00000003          mov    eax, [ebp+8]
.text:00000006          mov    ecx, [ebp+0Ch]
.text:00000009          lea    eax, [ecx+eax*2]
.text:0000000C          pop    ebp
.text:0000000D          retn
.text:00000010 _main:   push   ebp
.text:00000011          mov    ebp, esp
.text:00000013          push   ecx
.text:00000014          mov    eax, [ebp+0Ch]
.text:00000017          mov    ecx, [eax+4]
.text:0000001A          push   ecx
.text:0000001B          call   dword ptr ds:__imp__atoi
.text:00000021          add    esp, 4
.text:00000024          mov    [ebp-4], eax
.text:00000027          mov    edx, [ebp-4]
.text:0000002A          push   edx
.text:0000002B          mov    eax, [ebp+8]
.text:0000002E          push   eax
.text:0000002F          call   _sub
.text:00000034          add    esp, 8
.text:00000037          mov    esp, ebp
.text:00000039          pop    ebp
.text:0000003A          retn
```

88



~0.8 mW @100MHz

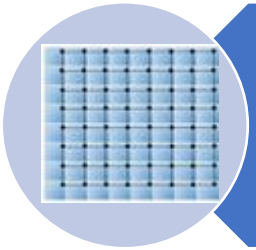


AMD Ryzen 7000
~65W, 3.6 GHz
12 cores, 24 threads

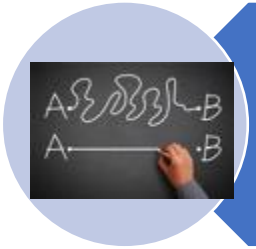
NUS National University of Singapore

Spatial Accelerators: Scaling Parallelism

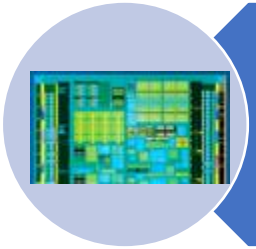
Software-driven parallelism extraction



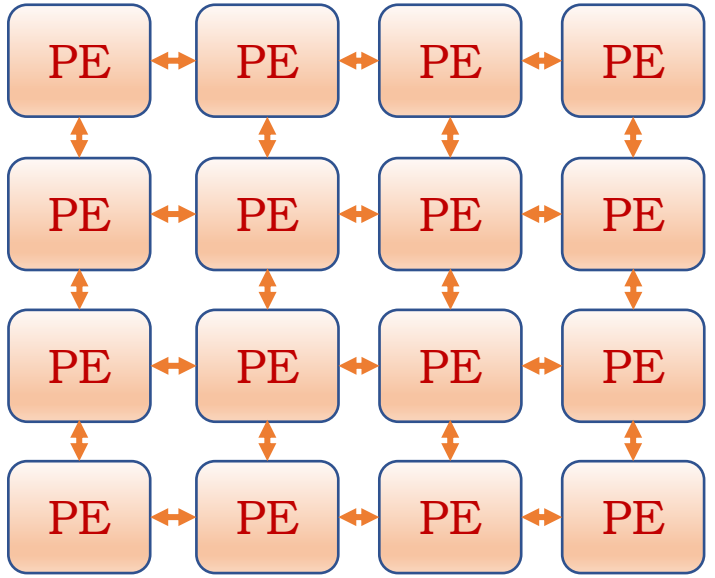
Embrace Parallelism



Keep Hardware Simple



Expose to Software



NUS National University of Singapore

What can we do with 1W power?

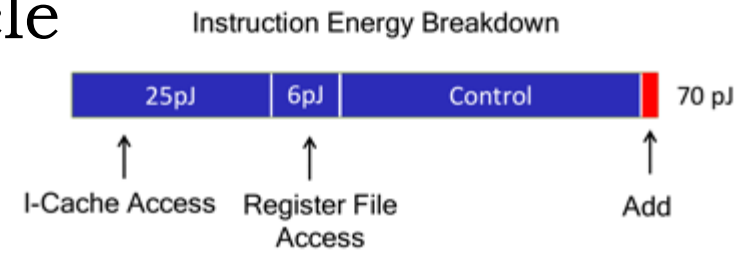
Integer	
Add	
8 bit	0.03pJ
32 bit	0.1pJ
Mult	
8 bit	0.2pJ
32 bit	3.1pJ

FP	
FAdd	
16 bit	0.4pJ
32 bit	0.9pJ
FMult	
16 bit	1.1pJ
32 bit	3.7pJ

Memory	
Cache	(64bit)
8KB	10pJ
32KB	20pJ
1MB	100pJ
DRAM	1.3-2.6nJ

At 100MHz, 0.9V, 45nm

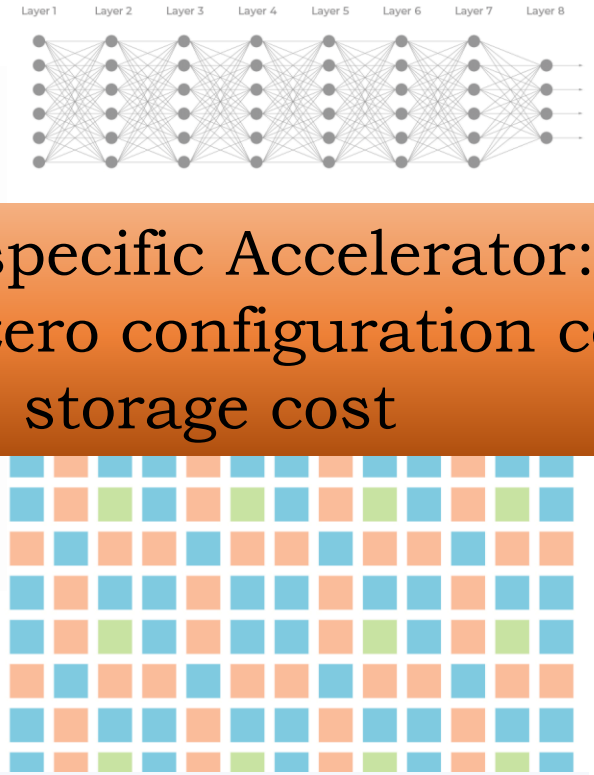
- Compute only: ~43K MAC units
  - 4.3 TOPS/W, 8-bit integer
- Storage only: ~100MB, 100 access per cycle
- How much storage for data?
- How much storage for configurations?



Mark Horowitz. ISSCC 2014

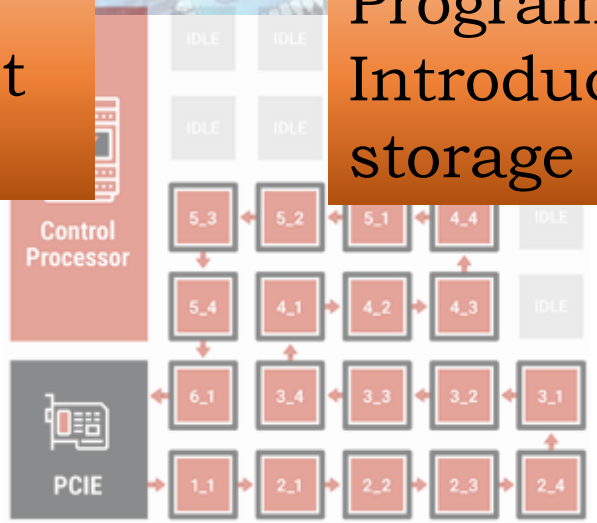
Accelerator design choices are tradeoff between on-chip compute, data storage, and configuration storage

Custom Edge AI Accelerators



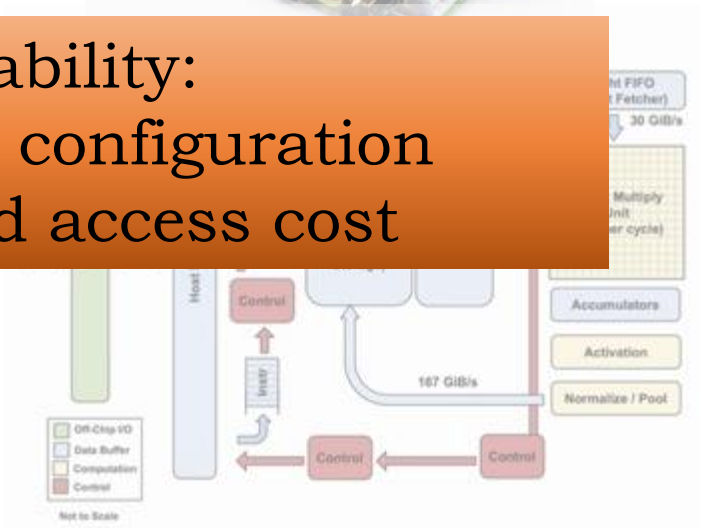
Domain-specific Accelerator:
Close to zero configuration cost
Very little storage cost

Hailo
26 TOPS at 3 TOPS/W



Programmability:
Introduces configuration storage and access cost

Mythic:
35 TOPS at 4 TOPS/W



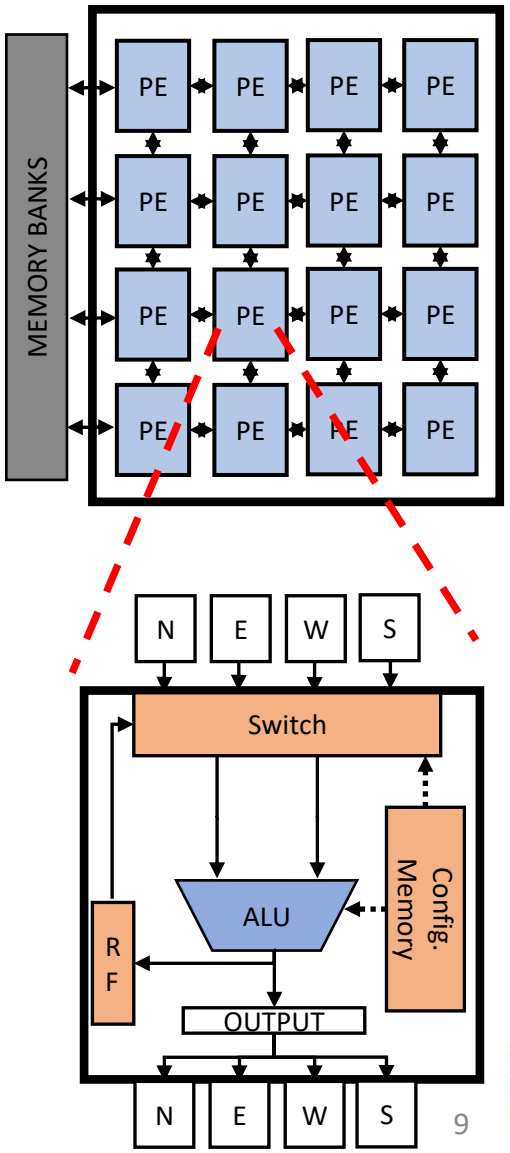
Edge TPU:
4 TOPS at 2 TOPS/W

Image Credit: Hailo, Mythic, Google

NUS National University of Singapore

CGRA Spatial Accelerator: Efficiency with Programmability

- Parallelism: Array of simple processing elements (PE)
- Operation of a PE and the data transfer between PEs can be configured per cycle
- Exposed to software: Compiler sets the configuration according to the dataflow of the application
- Simple hardware and hence low power: During execution, PE array and switches simply follow the pre-defined configurations
- As number of configurations is limited, CGRA can only support recurring computation, i.e., loops



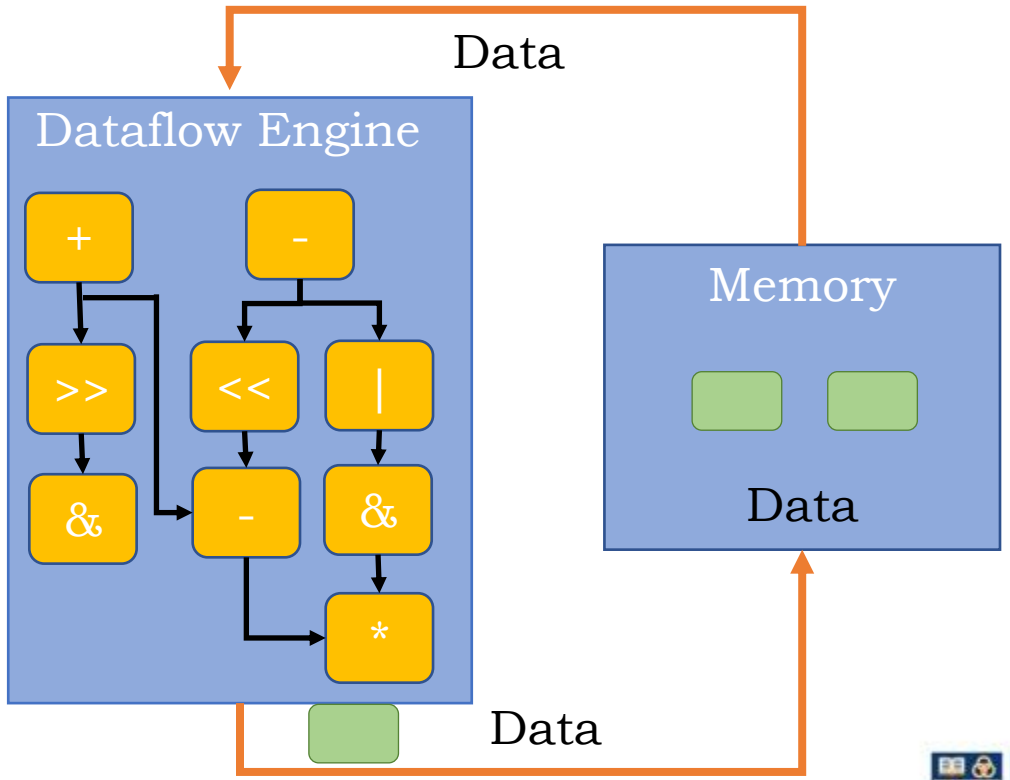
Dataflow Processor

- Expose instruction-level parallelism in dataflow graph at compile time
- Dataflow graph is directly mapped onto hardware
- Operation is triggered when input data is ready
- Dataflow computing dominates most accelerator designs today

A Preliminary Architecture for a Basic Data-Flow Processor*

Jack B. Dennis and David P. Misunas
Project MAC
Massachusetts Institute of Technology

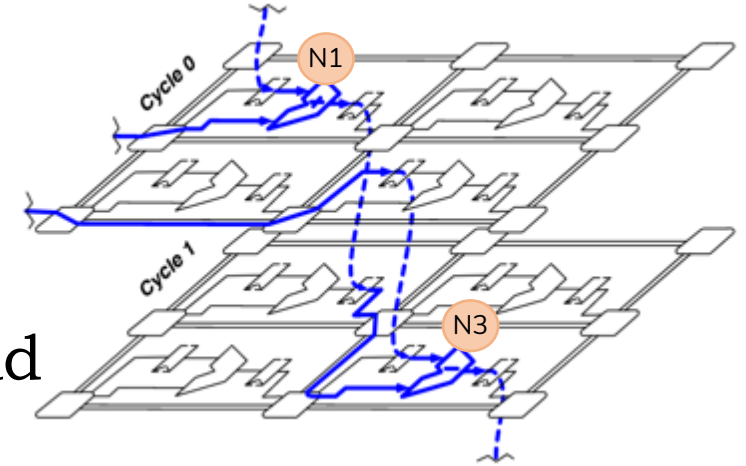
ISCA 1975



Dataflow synthesis on CGRA

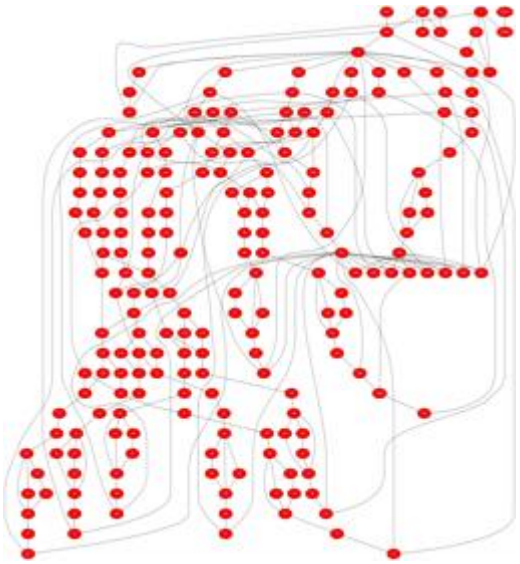
Mapping the dataflow graph (DFG) of the loop body

- **Placement:** assign DFG operations to PEs
- **Routing:** map data dependencies through links between PEs rather than through cache or scratchpad

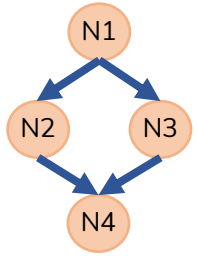


Placement and routing

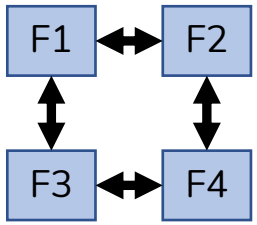
Loop kernel c code



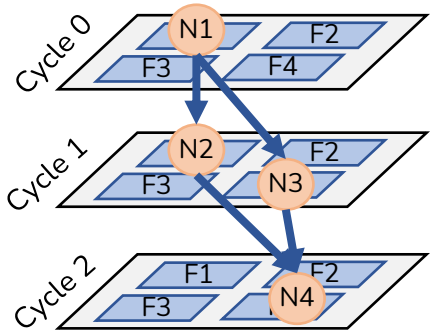
Dataflow graph (DFG) of the loop body



DFG



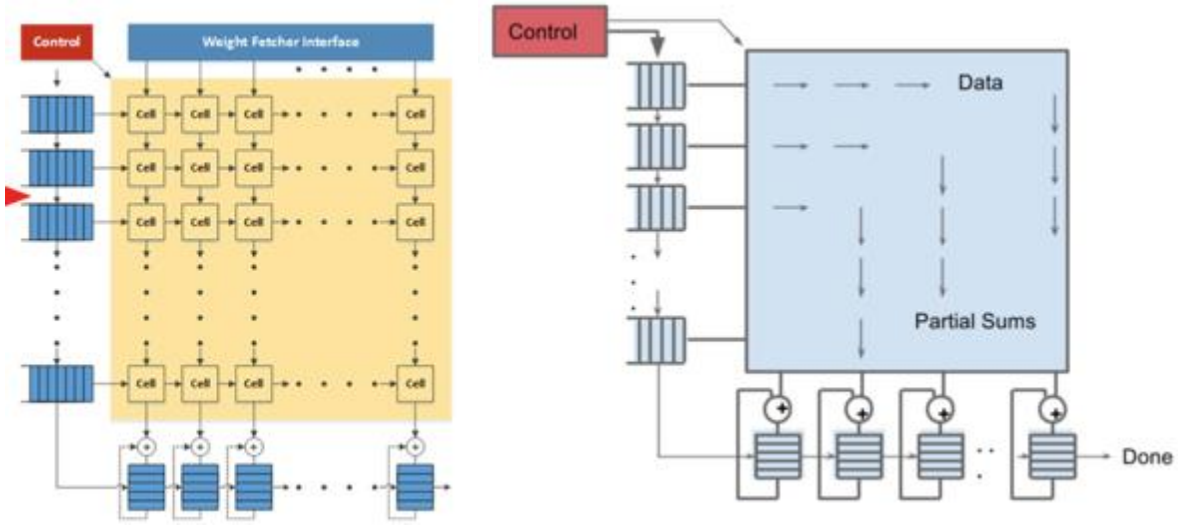
2x2 CGRA



Spatio-temporal mapping

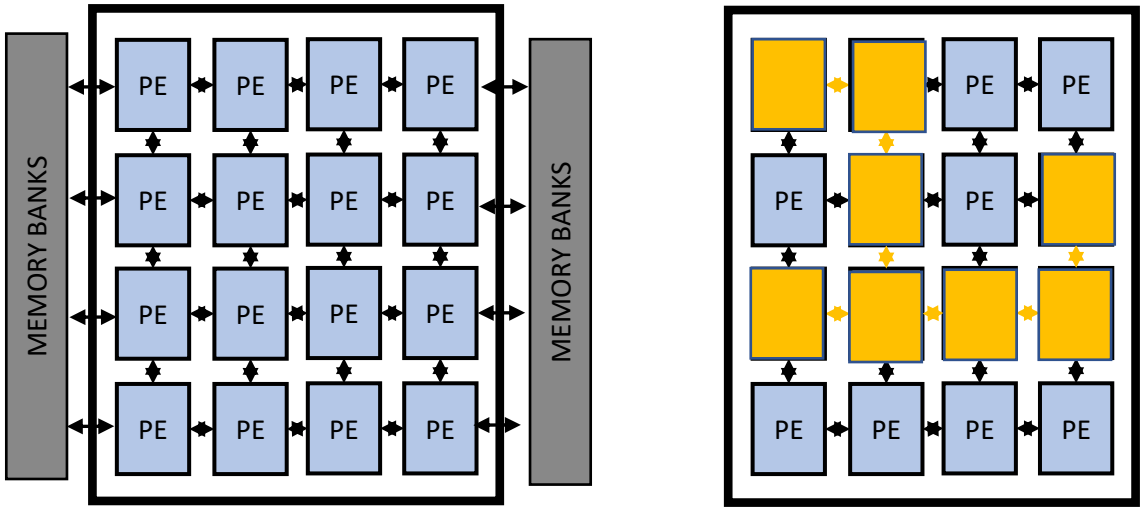
CGRA Spatial Accelerator: Efficiency + Programmability

Instantiate different task-specific virtual or logical accelerators on the same hardware fabric through software-directed reconfiguration



Google TPU: Specialized

Hardware architecture is designed specially to support matrix multiplication dataflow

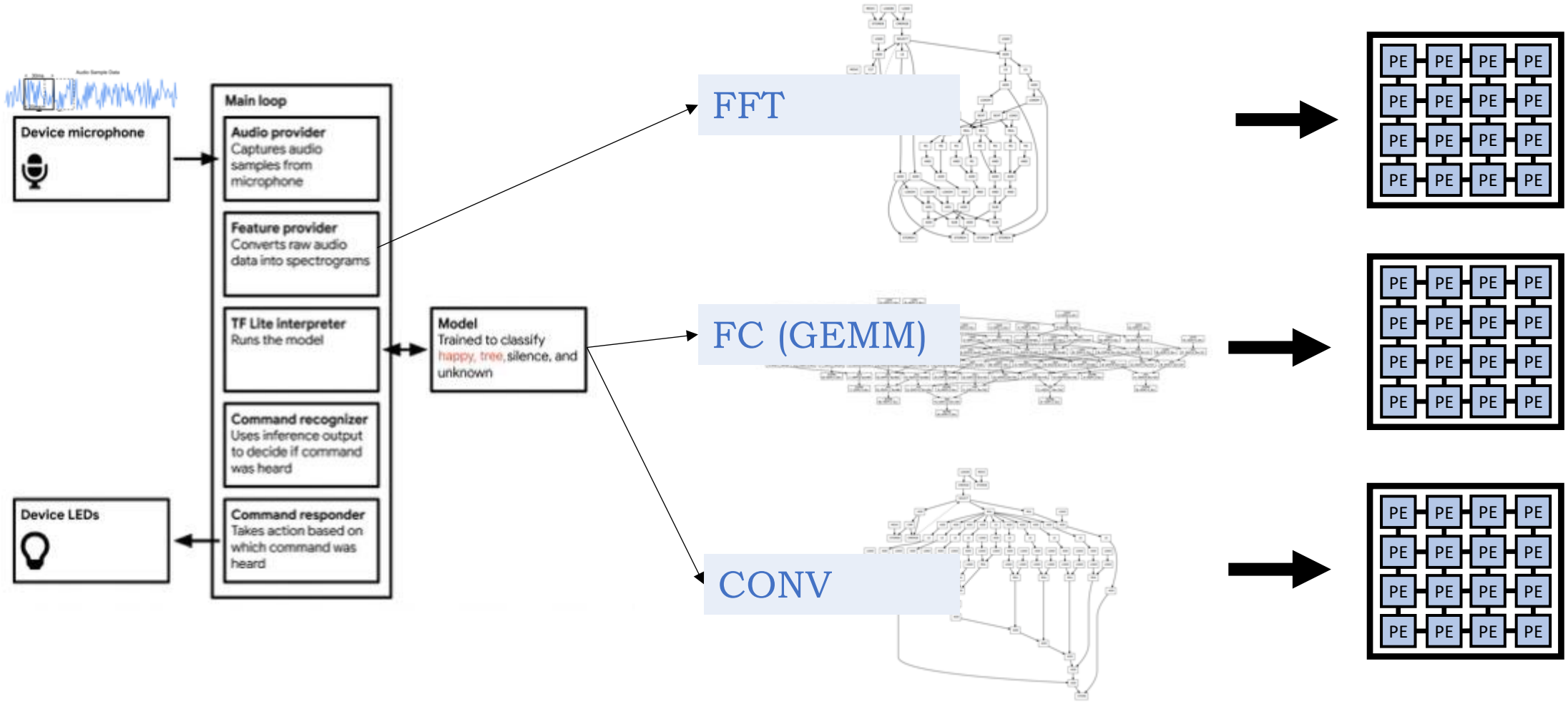


CGRA: Generalized

Hardware architecture is general and reconfigured to support different dataflows

NUS National University of Singapore

CGRA: Mapping Diverse Dataflows



Speech Recognition Model

Application Kernels

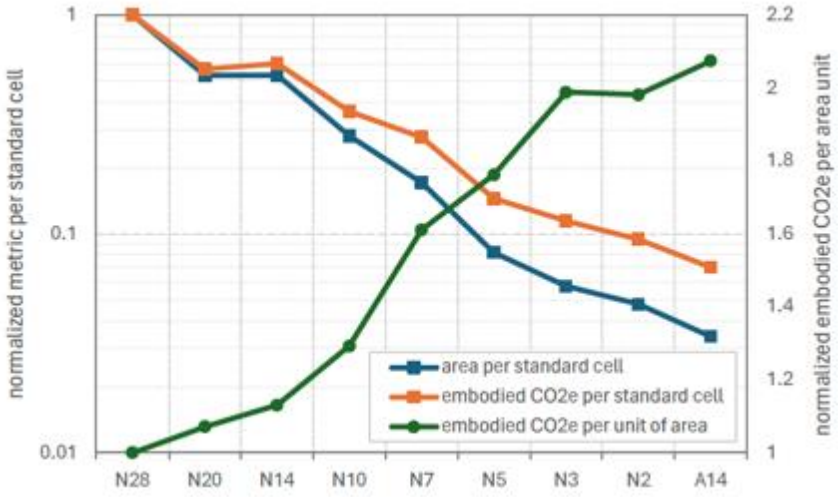
Different Dataflows

NUS National University of Singapore

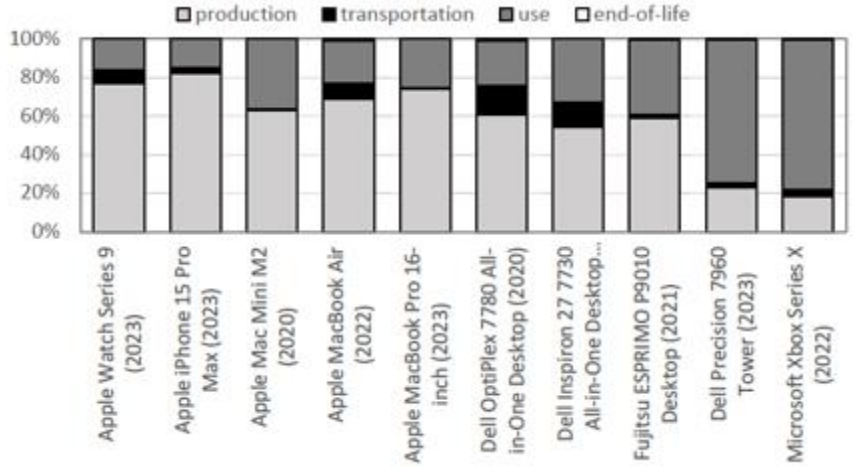
CGRA for Sustainability

Embodied footprint: Raw material extraction, manufacturing, assembly, transportation, end-of-life processing

Operational footprint: Device use during its entire lifetime



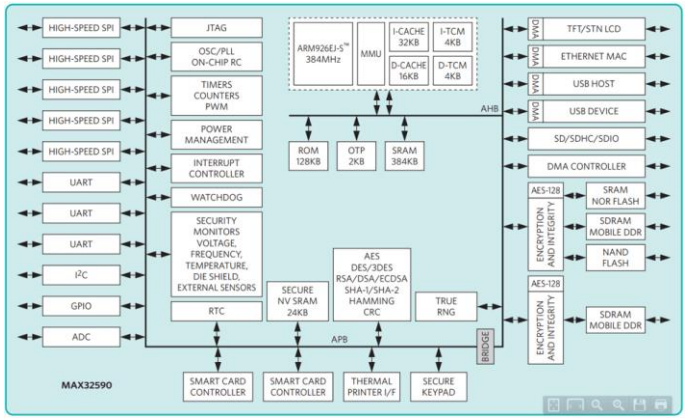
Embodied footprint per unit chip area is increasing with technology nodes



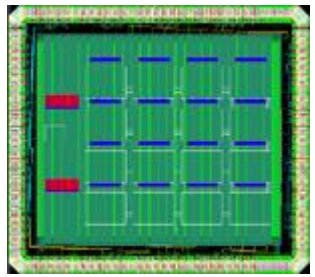
Embodied footprint dominated most consumer computing devices

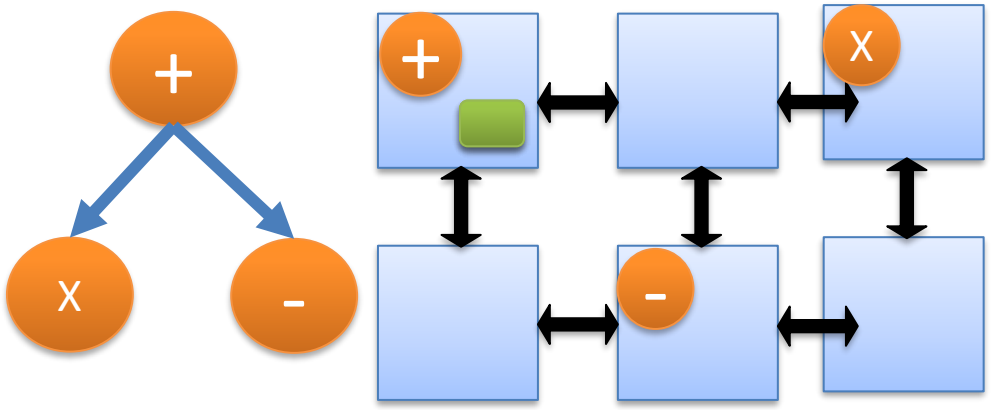
Concurrency (n)	Carbon Footprint Improvement	
	Avg Util ($n' < n$)	100% Util ($n' = n$)
1	-	7.60×
2	6.10×	3.84×
3	4.12×	2.59×
4	3.12×	1.97×
5	2.53×	1.59×

Carbon footprint saving by replacing SoC IP blocks with CGRA

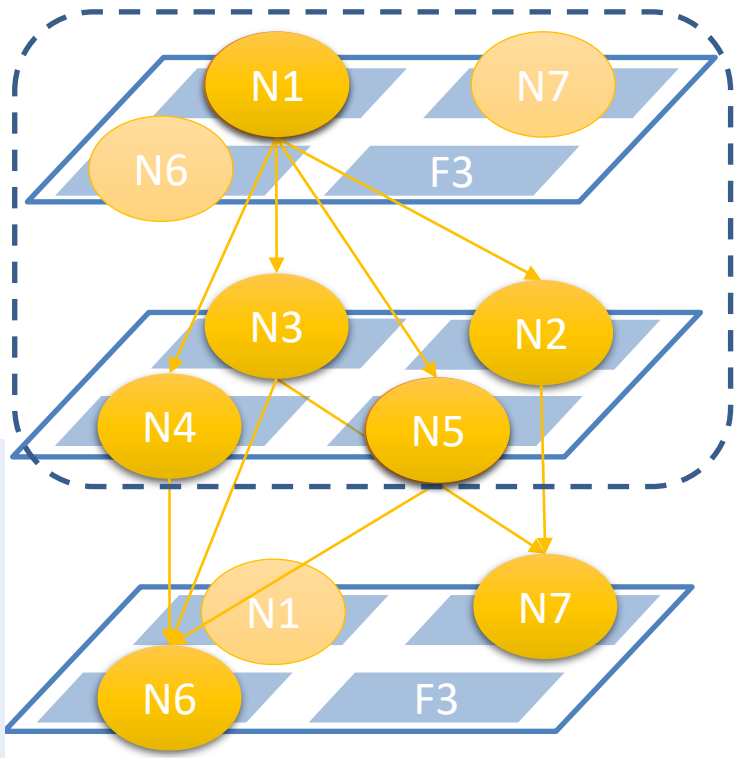


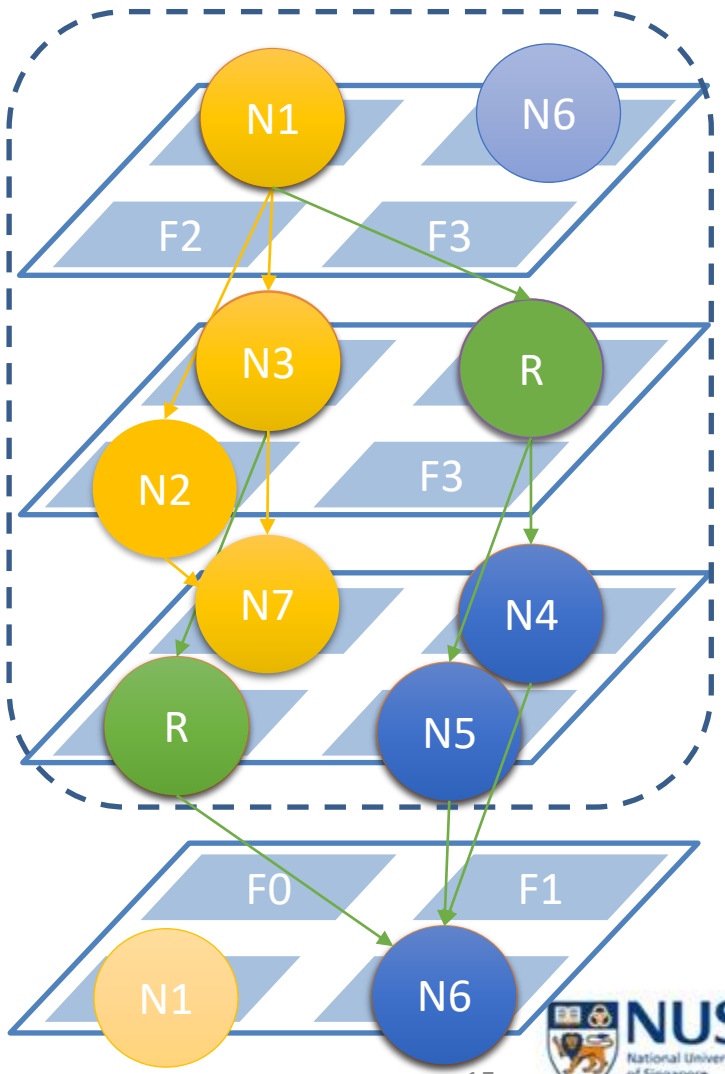
HyCUBE CGRA @ NUS





- Reconfigurable single-cycle multi-hop interconnect
- Creates large dynamic neighborhood reachable within single-cycle through asynchronous bypassing of intermediate PEs

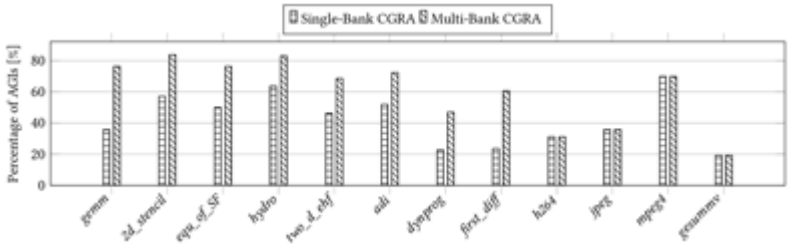


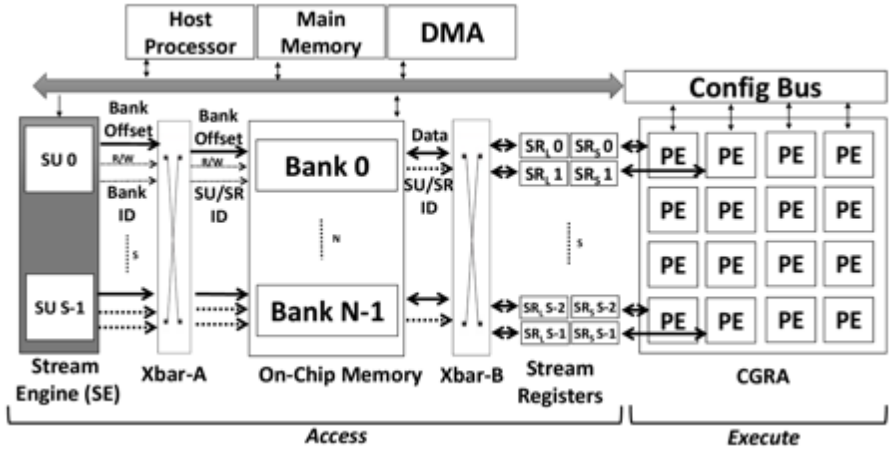


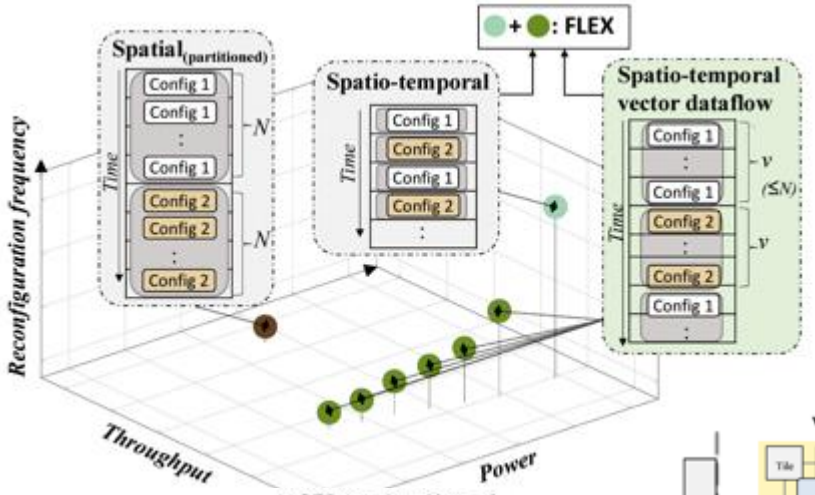
HyCUBE: a 0.9V 26.4 MOPS/mW, 290 pJ/cycle, Power Efficient Accelerator for IoT Applications ASSCC'19

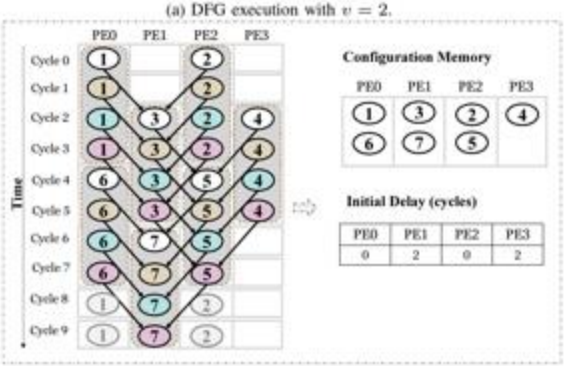
HyCUBE : A CGRA with Reconfigurable Single-cycle Multi-hop Interconnect DAC 2017

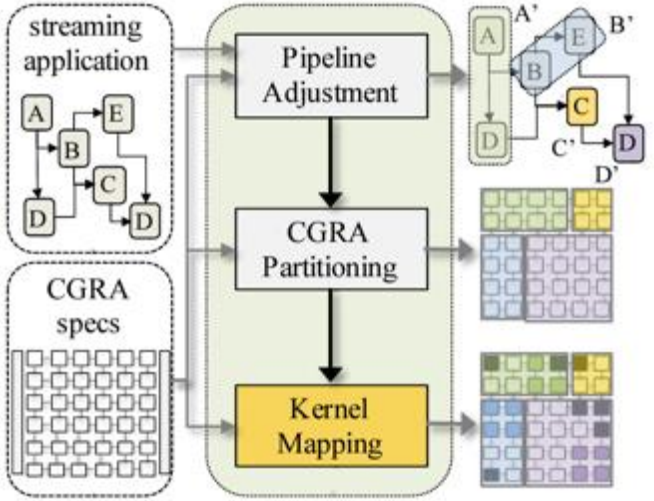
CGRA Architectural Innovations

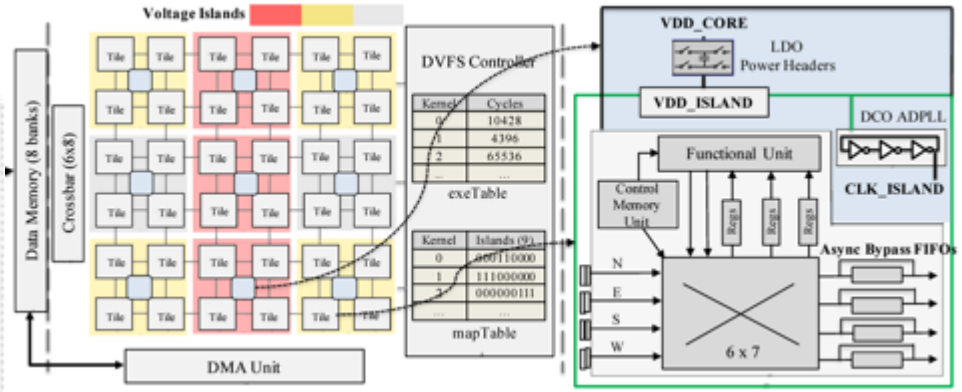












CASCADE
Decoupled Access-Execute CGRA
3x speedup, 2.2x performance/watt over iso-area CGRA

FLEX:
Spatio-temporal vector dataflow
45% less energy. 1.9× power efficiency at same performance over CGRA

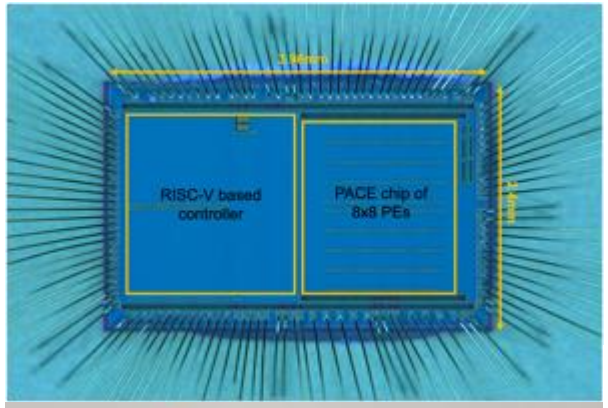
ICED:
DVFS-aware CGRA
1.32x energy-efficiency over CGRA at same performance

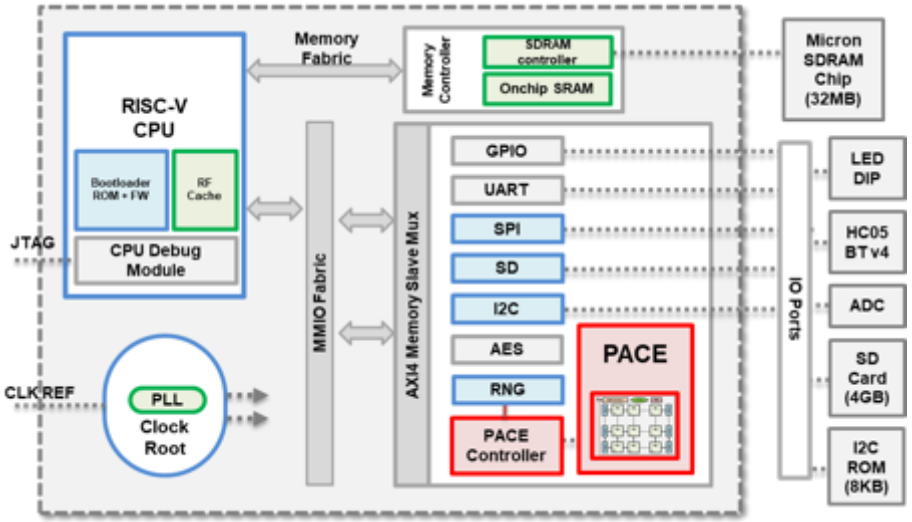
CASCADE: High Throughput Data Streaming via Decoupled Access/Execute CGRA. CASES 2019
FLEX : Introducing FLEXible Execution on CGRA with Spatio-Temporal Vector Dataflow. ICCAD 2023
ICED: An Integrated CGRA Framework Enabling DVFS-Aware Acceleration. MICRO 2024

NUS National University of Singapore

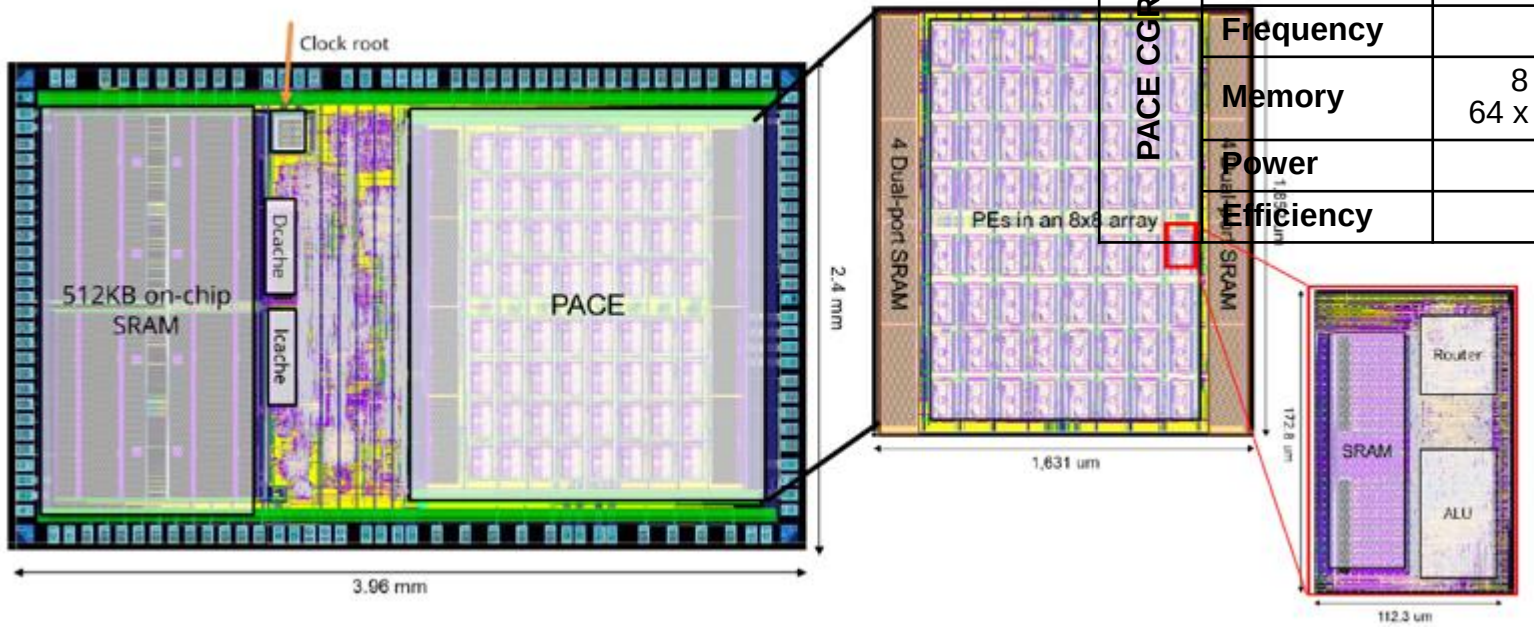
PACE CGRA+RISC-V SoC: 2023

8x8 CGRA: 1.1 mW at 10MHz
582 GOPS/W at 0.45V, 40nm ULP
Estimated 1 TOPS/W at 22nm





PACE SoC Specifications (Simulation)		
Tech. Node		UMC 40nm ULP
Area		3.96mm x 2.4mm (Die) 3.80mm x 2.0mm (Core)
Voltage		1V/3.3V
RISC-V based controller	**Memory**	512 KB on-chip memory 16KB ICache 16KB DCache
	Frequency	200/100 MHz
	Power	14mW @ 1.0V
PACE CGRA	**#PEs**	8x8 (17 instructions, including NOP)
	Frequency	10 MHz
	Memory	8 x 8KB data memory 64 x 0.25KB config memory
	Power	1.1 mW @ 0.45V
	Efficiency	582 TOPS/W



PACE: A Scalable and Energy Efficient CGRA in a RISC-V SoC for Edge Computing Applications. Hot Chips 2024

CGRA Comparison

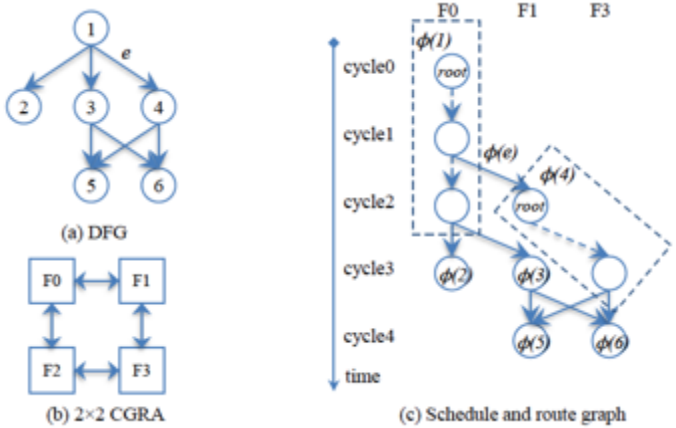
PACE has highest power efficiency even at 40nm

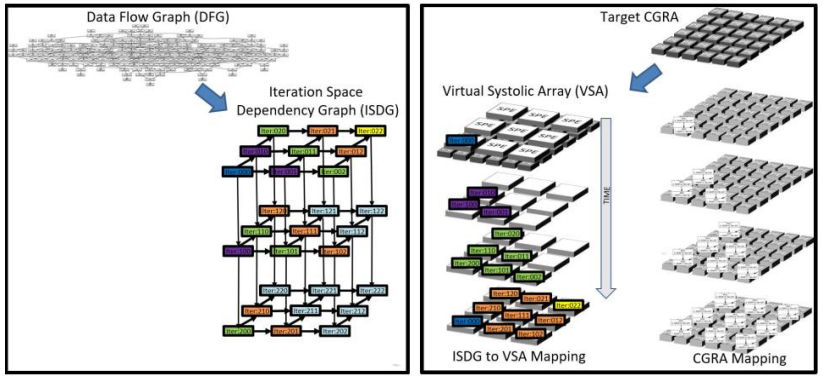
	Academic						Commercial		
	Post synthesis simulation/Post P&R				Silicon		Post synthesis simulation/Post P&R	Silicon	
CGRA	**UE_CGRA HPCA'21**	**SNAFU ISCA'21**	**TRANSPIRE DATE'20**	**RIPTIDE MICRO'22**	**AMBER VLSI'22**	**PACE**	**ULP-SRP FPT'12**	**Renesas DRP VLSI'18**	**Cardinal SN10 (Improved version of Plasticine for ML training) ISSCC'22**
PE array	8x8	6x6		6x6	384 PEs	**8x8**	3x3	96 PEs	640 PMU 640 PCU
Node	28nm TSMC	Intel 22FFL	28nm FDSOI	Intel 22FFL	16nm FINFET	**40nm UMC**	40nm TSMC	28 nm	7nm TSMC
Voltage (V)	-	-	0.6	-	1.29	**0.45**	1.1	-	-
Frequency (MHz)	750	50	50	50	955	**10**	100	333 MHz	-
Area (mm^2)	0.25[1]	0.27[1]	0.27	0.25[1]	20.1	**3.94**	-	-	-
Performance (MOPS)	625	71	-	62	367000	**640**	467[2]	960 gigaflops	2.6 petaflops
Power (mW)	14	0.54	-	0.24	682	**1.1**	11.3	-	-
Power efficiency (GOPS/W)	45	134	224	254	538	**582**	23	143[3]	-

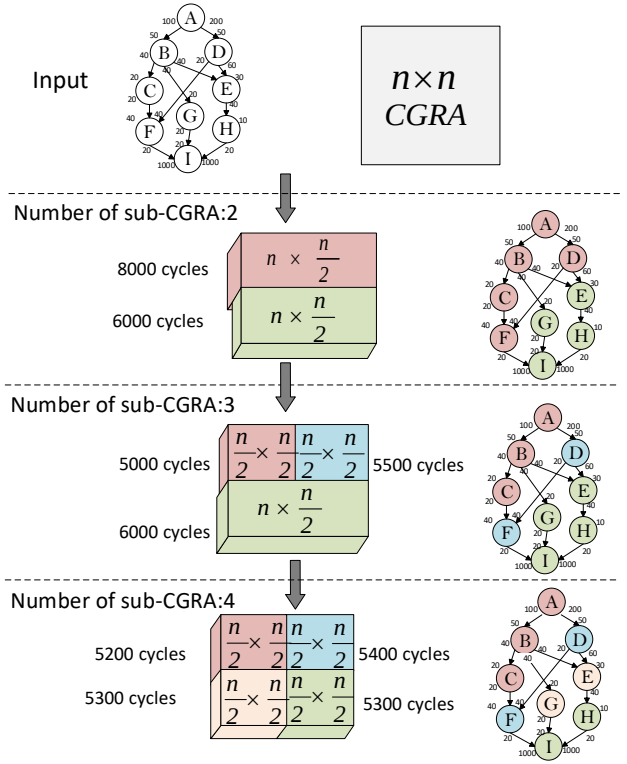
[1] Only the PE array included
[2] Estimated from published 467 MIPS
[3] Estimated value from experiments

Compilation: Dataflow Synthesis on CGRA

- Dataflow synthesis on CGRA provides a fertile ground for research
- Graph Minor (2012): Theoretically optimal based on graph minor modeling
- ChordMap (2020): Streaming dataflow application synthesis
- HiMap (2021), Panorama (2022): Scalable mapping through hierarchical abstractions
- LISA (2022): Automated compiler synthesis using GNN





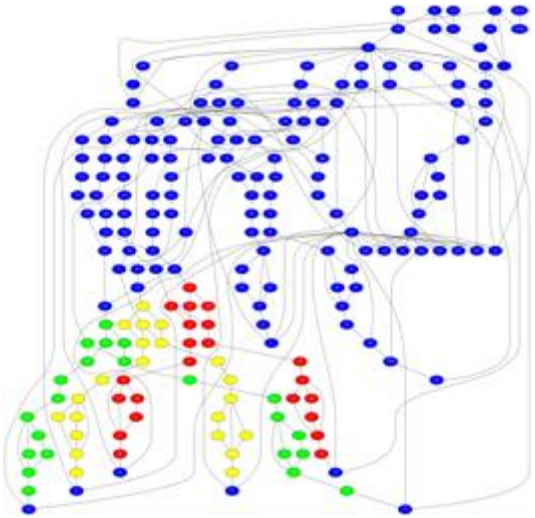


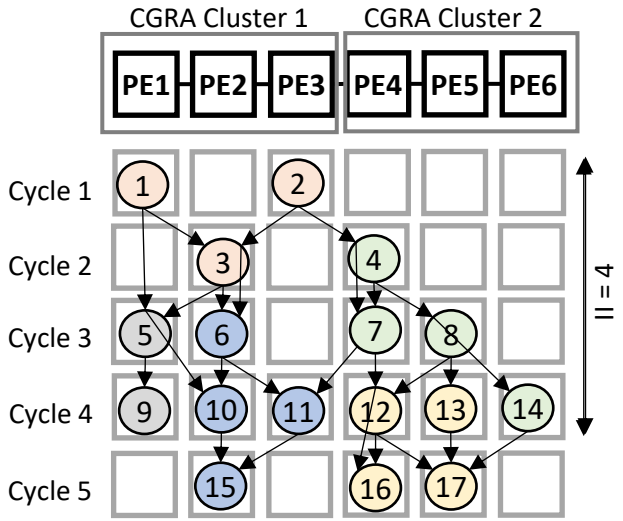
Graph Minor Approach for Application Mapping on CGRAs. FPT 2012 TRETS 2014 Best paper award
ChordMap: Automated Mapping of Streaming Applications onto CGRA. TCAD 2021
HiMap: Fast and Scalable High-Quality Mapping on CGRA via Hierarchical Abstraction. DATE 2021 TCAD 2022
Panorama: Divide-and-Conquer Approach for Mapping Complex Loop Kernels on CGRA. DAC 2022 Publicity Paper
LISA: Graph Neural Network based Portable Mapping on Spatial Accelerators. HPCA 2022 Distinguished Artifact

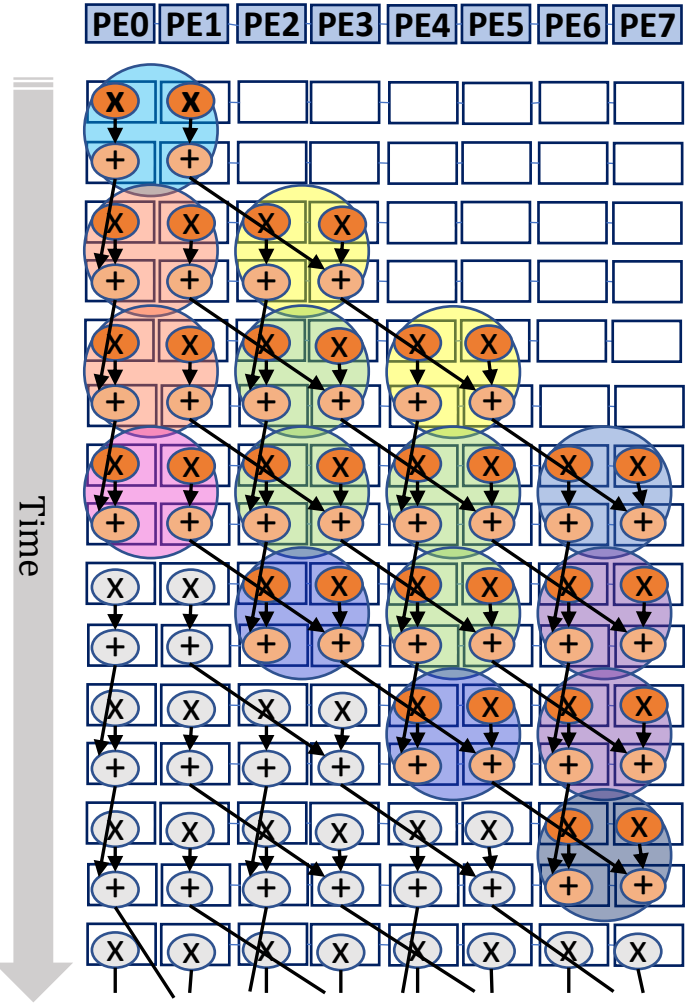
NUS National University of Singapore

Scaling Dataflow Synthesis on CGRA

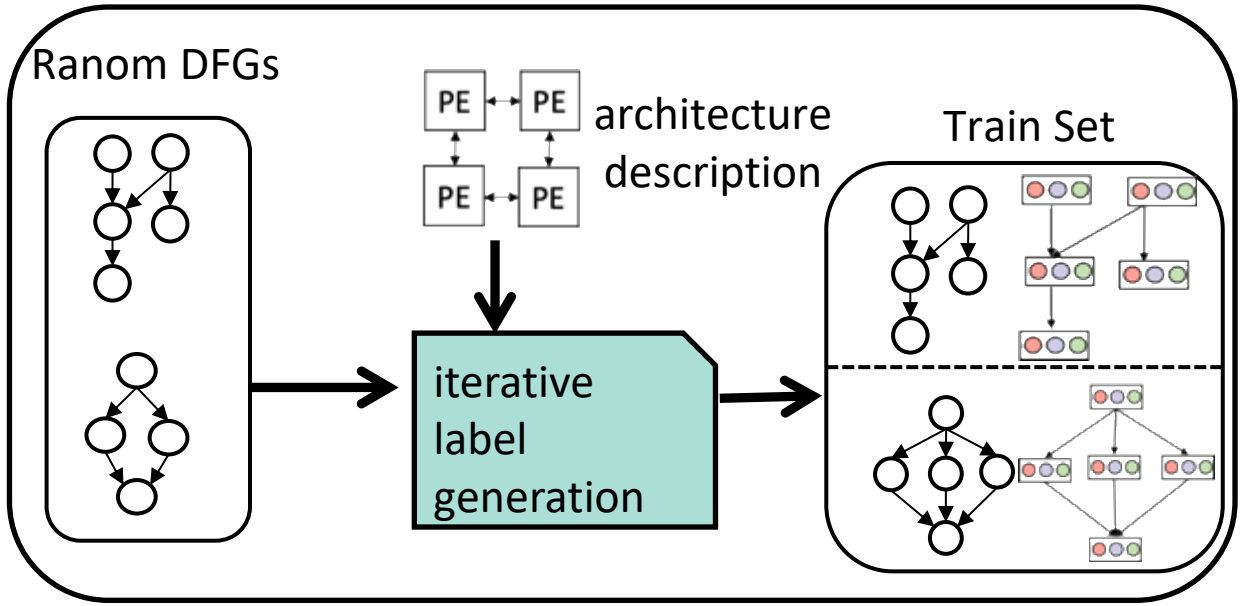
- Multiple abstractions to divide and conquer
- Clustering at both dataflow graph and architecture level
- Dramatic Improvement
  - 17x performance, 5x energy-efficiency
  - Reduce compilation time from days to15-min for 64x64 CGRA



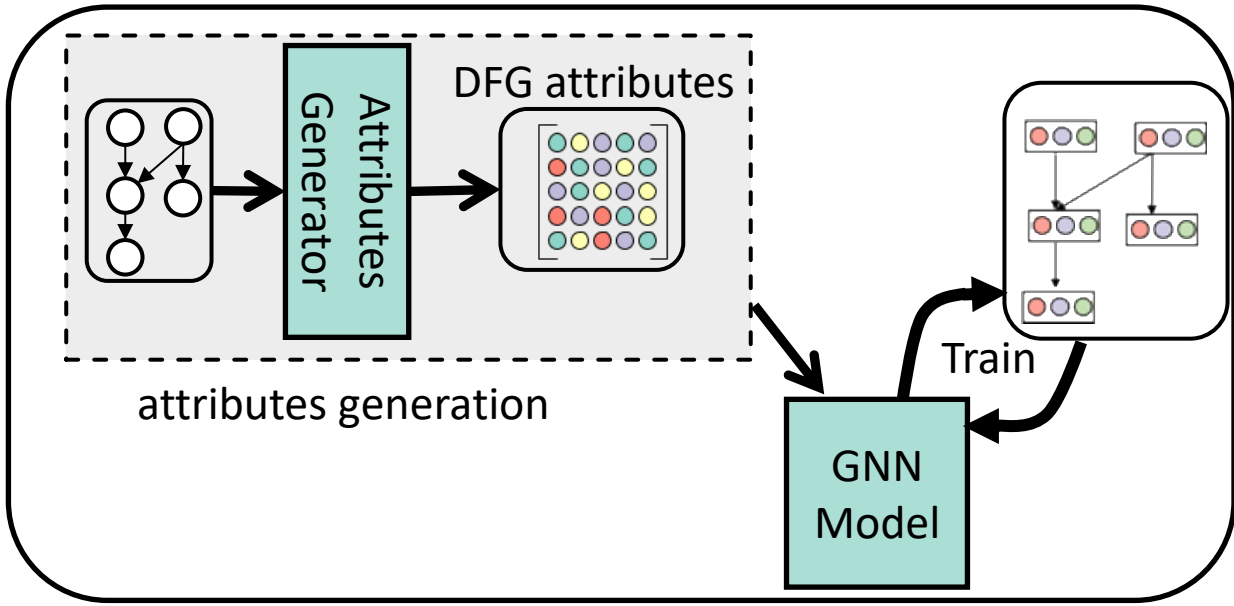




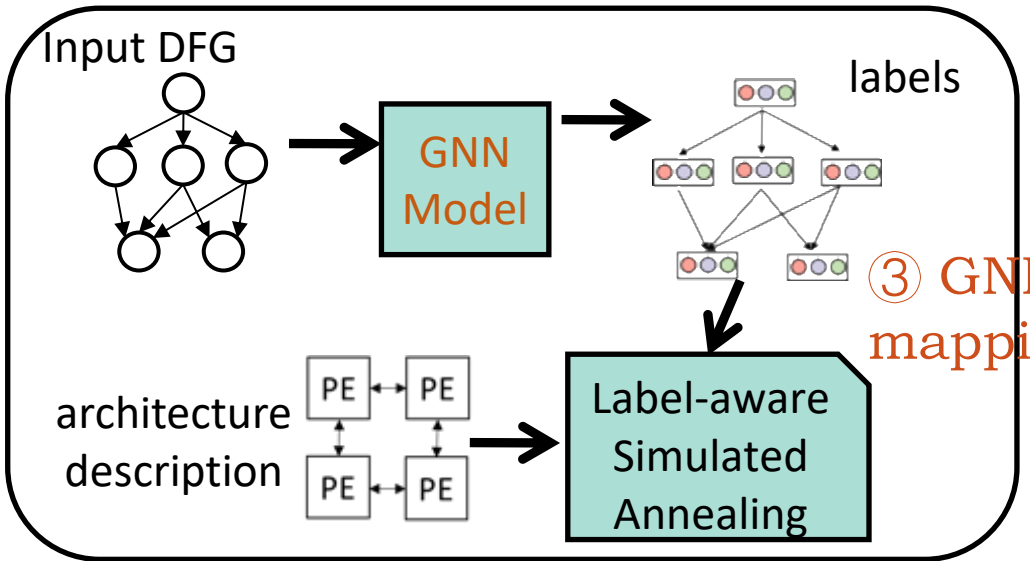
Automated GNN-based Mapping



① Generate training data



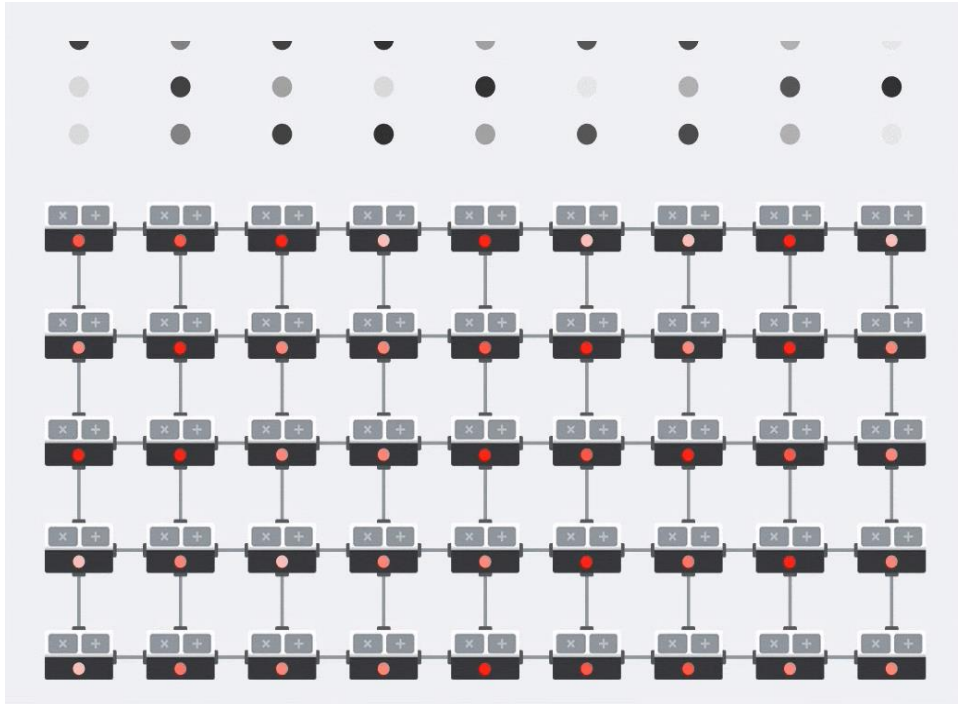
② Build GNN model



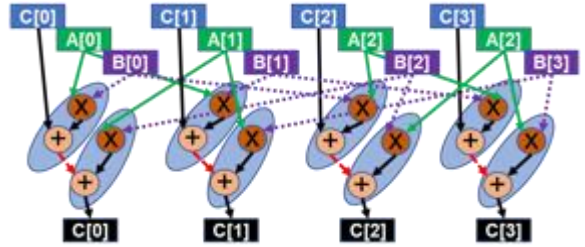
③ GNN-based mapping

- Generates best quality mapping in record time on diverse accelerators
- Achieves 17x compilation time reduction
- For 71 combinations of accelerators, kernels:
  LISA can map 70 combinations
  ILP maps only 23
  Simulated Annealing maps 50

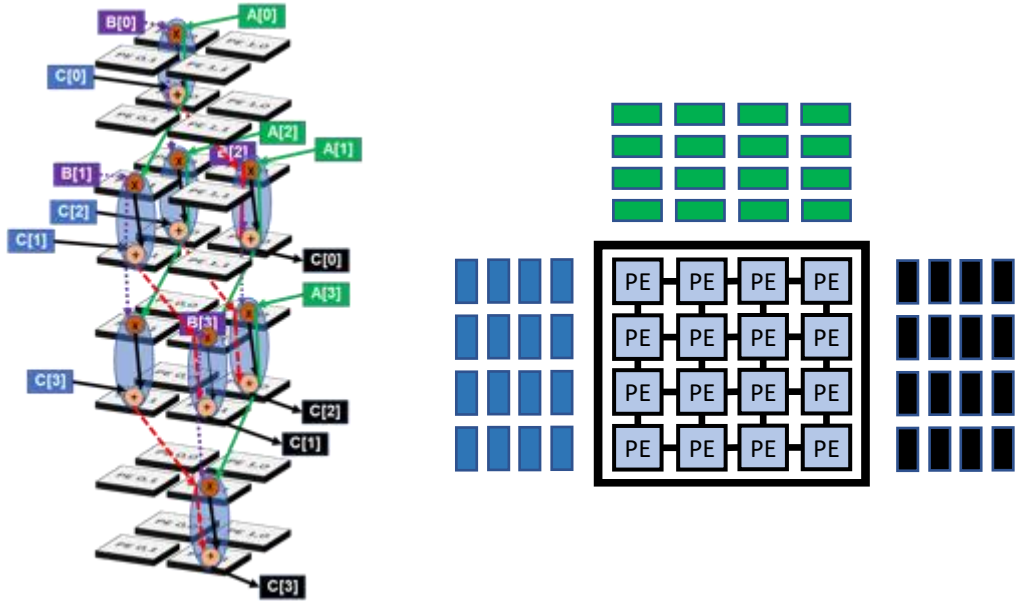
Instantiating TPU-like dataflow for GEMM



Google TPU Matrix Multiply Unit (MXU) Dataflow



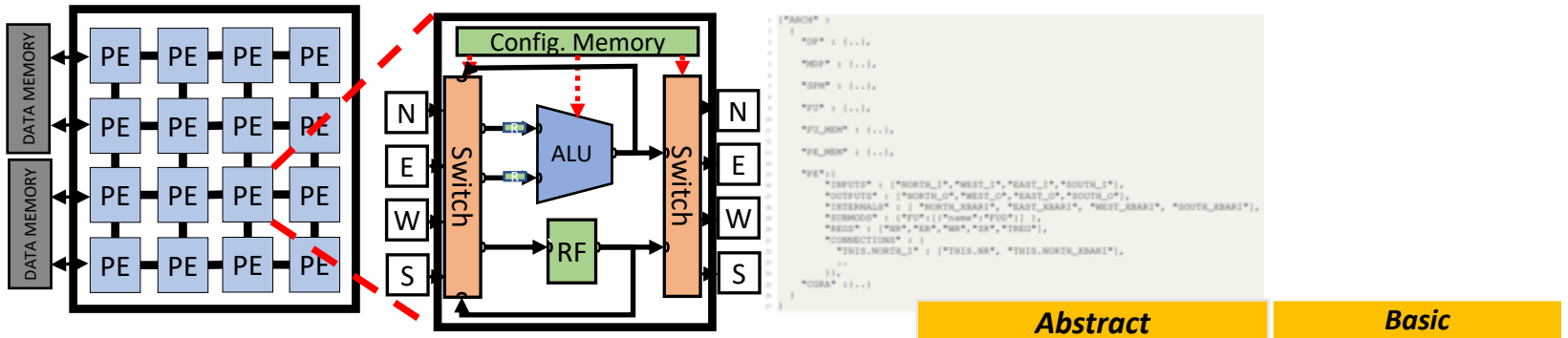
GEMM DFG

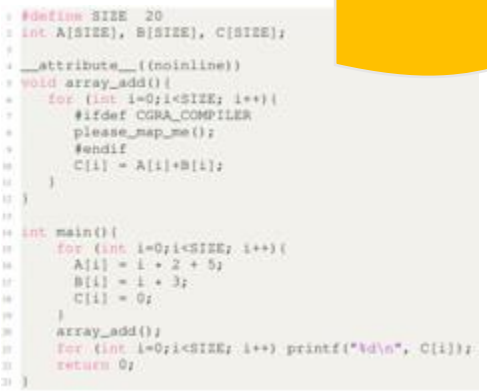


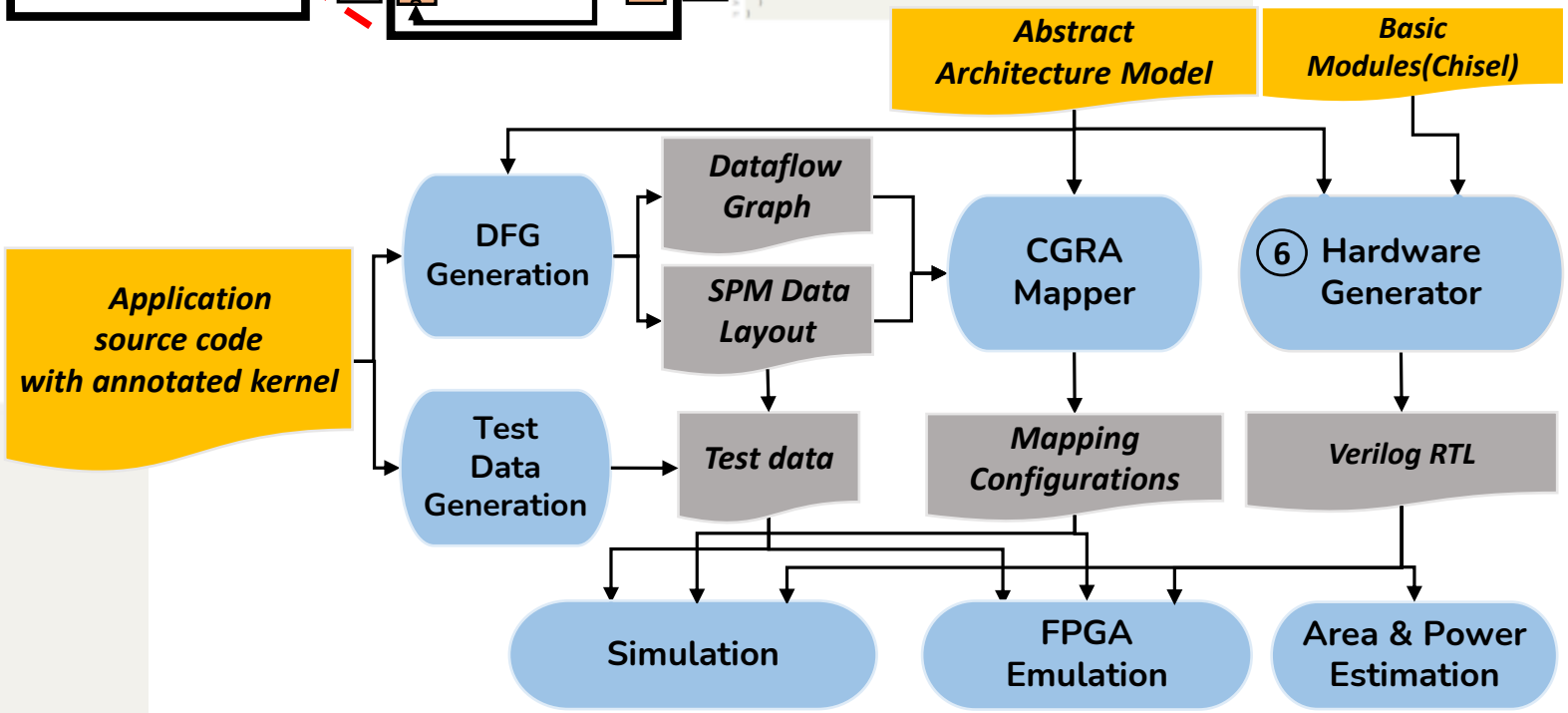
HiMap GEMM Schedule & Dataflow on CGRA

NUS National University of Singapore

NUS Morpher Open-Source CGRA Toolchain





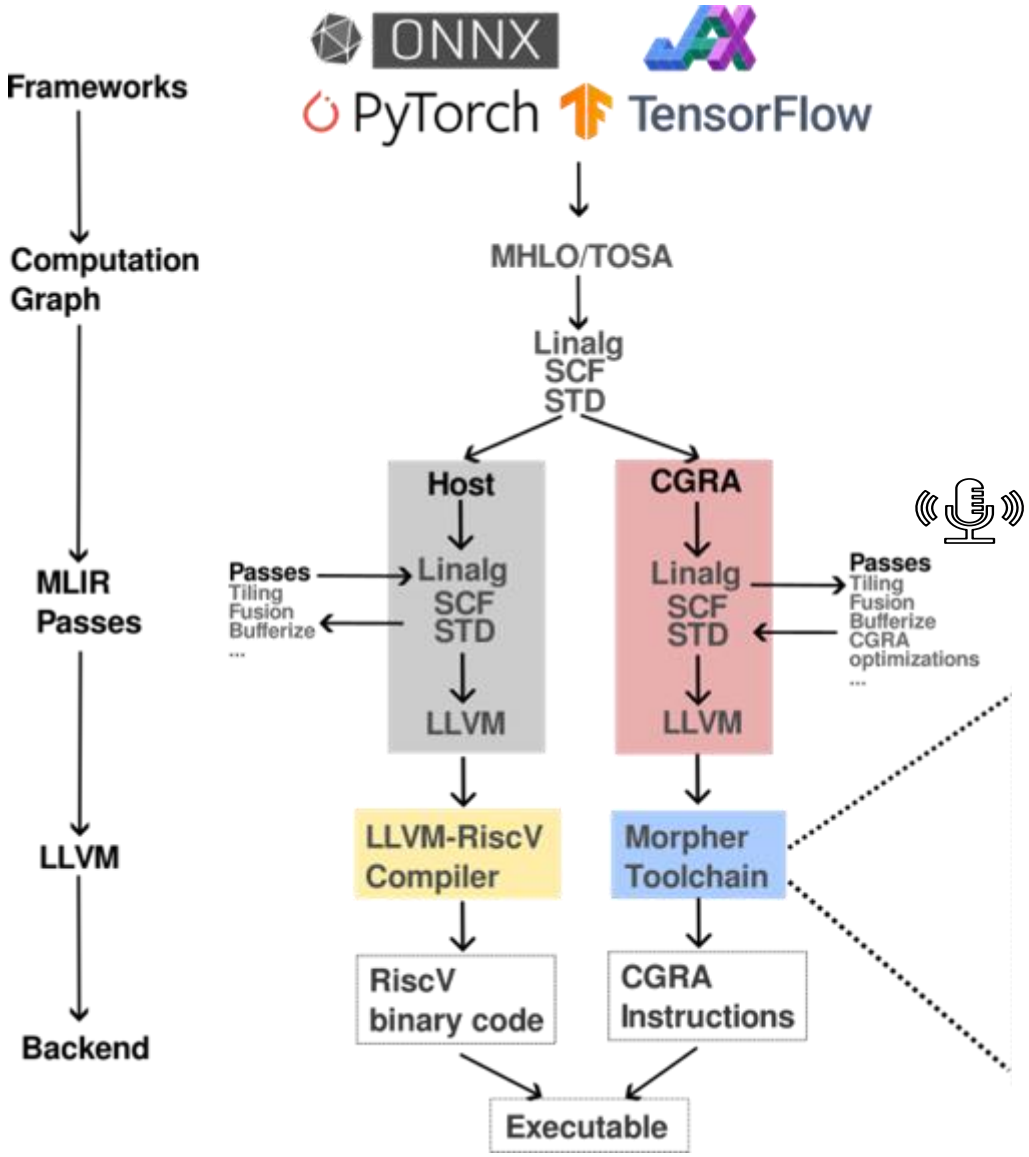


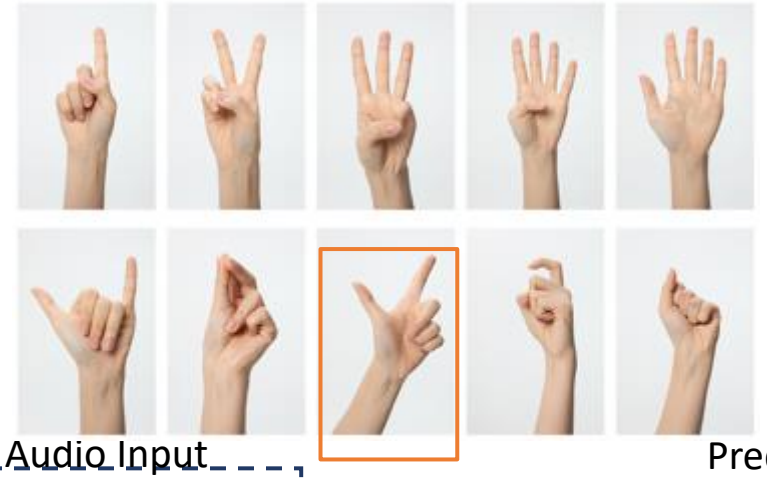
SCAN ME

https://github.com/ecolab-nus/morpher

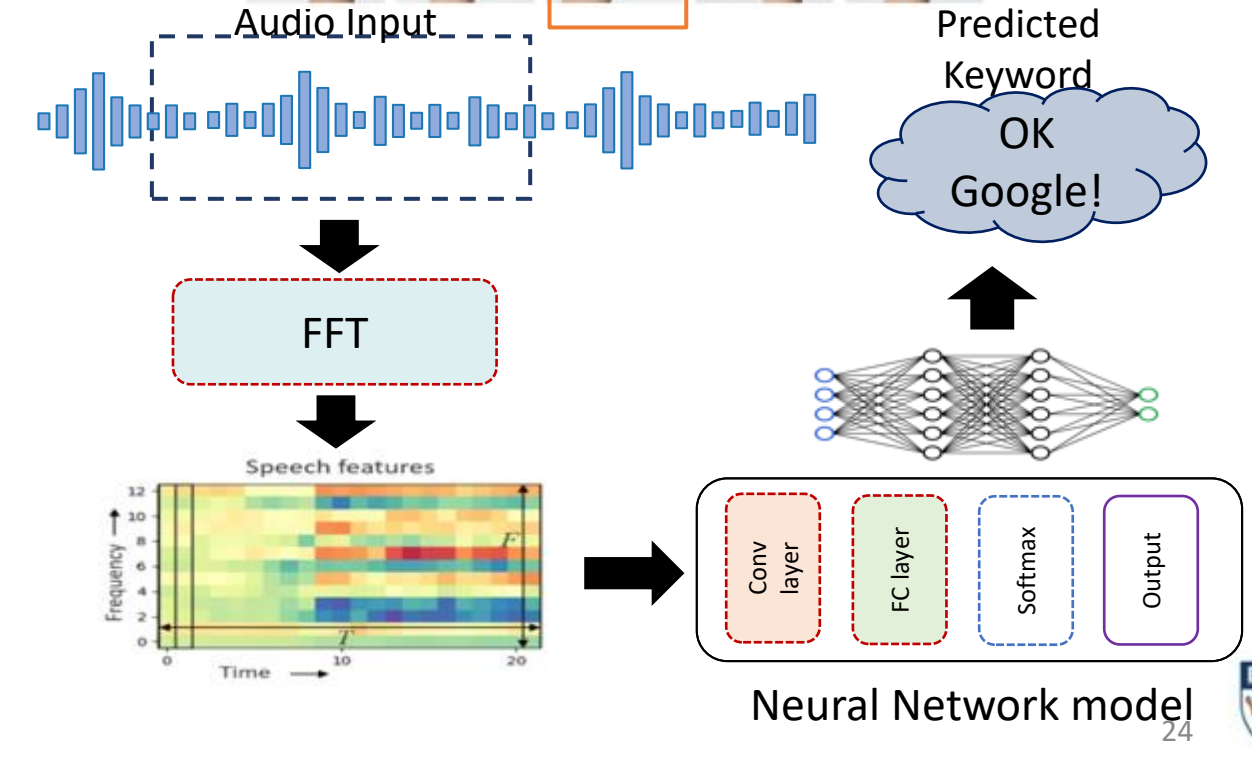
NUS National University of Singapore

NUS MLIR-Based End-to-End CGRA Compiler

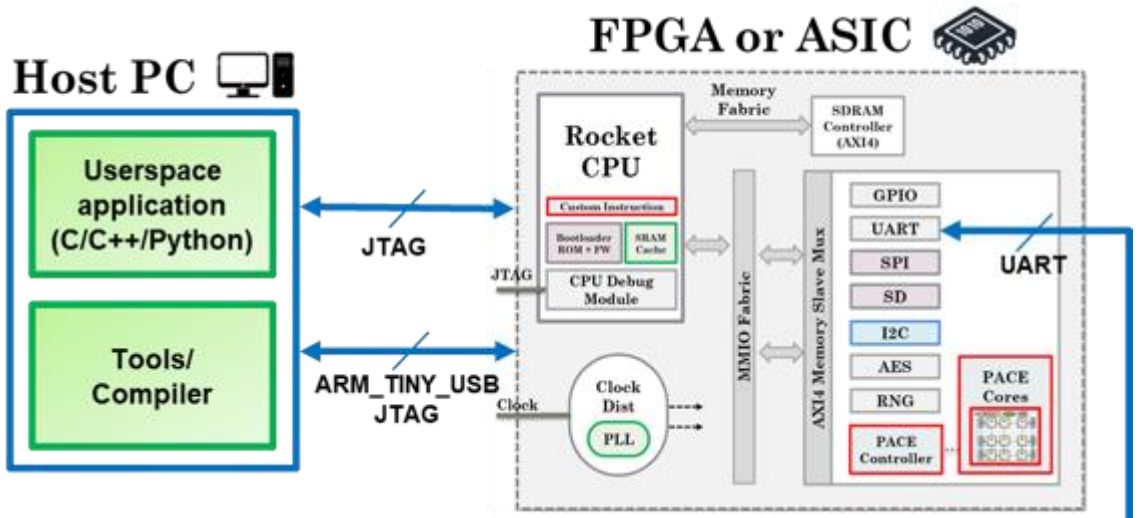




Gesture and voice recognition

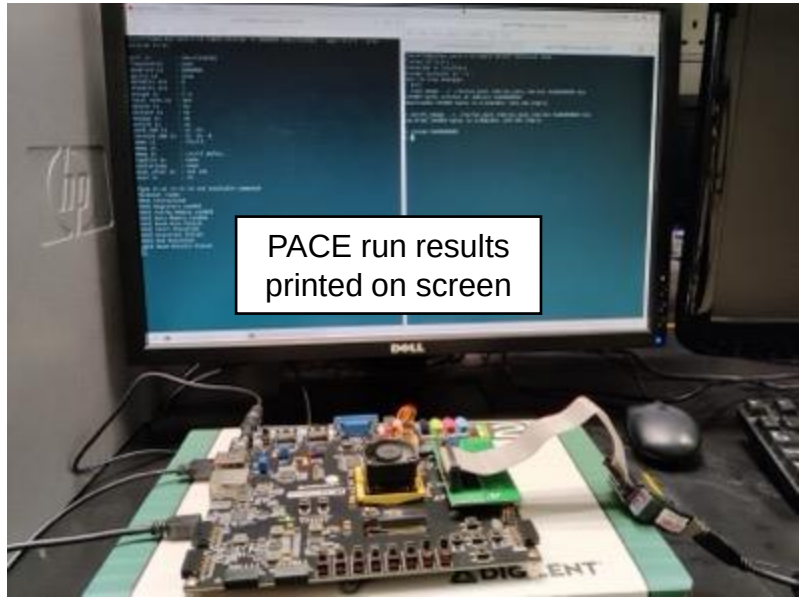


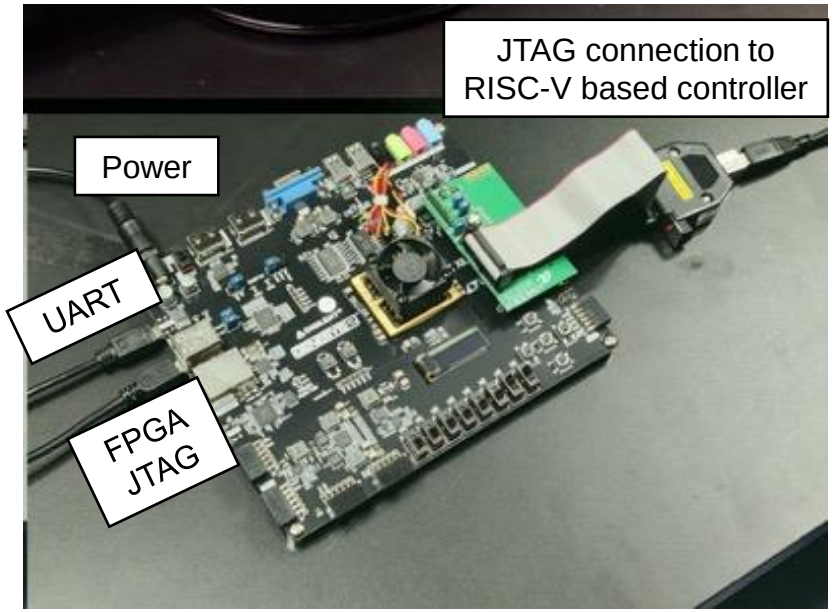
CGRA SoC Emulation on FPGA




```
Terminal ready
PACE Initialized
PACE Registers Loaded
PACE Config Memory Loaded
PACE Data Memory Loaded
PACE Read Data Finish
PACE Start Execution
PACE Execution Finish
PACE End Execution
PACE Read Results Finish
```

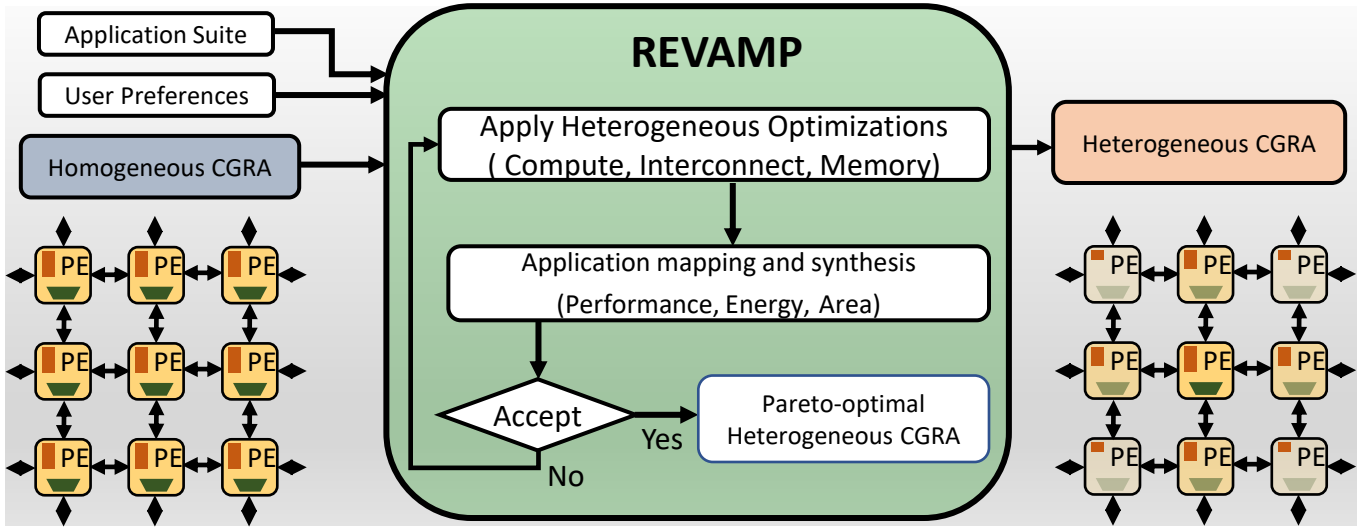
VCOM PORT

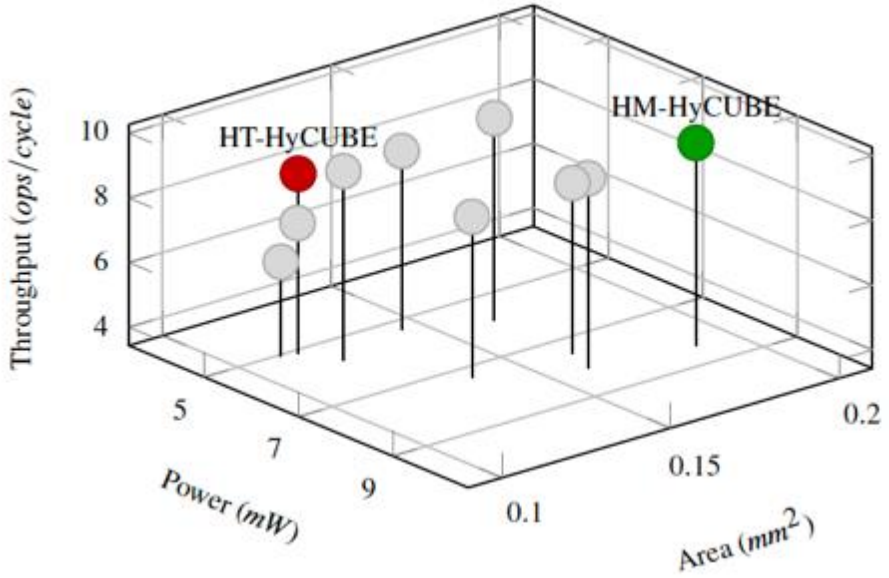




Domain-Specific CGRA

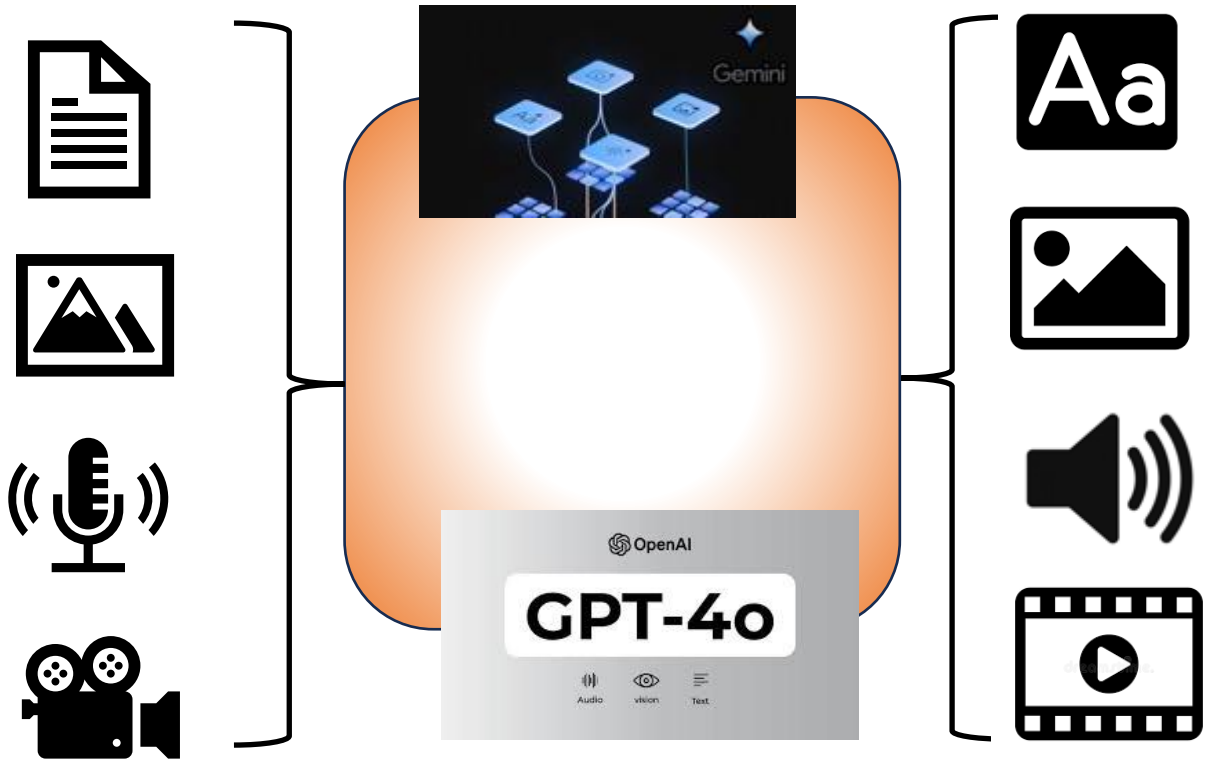
- Task-specific CGRA design
- RVAMP: Automated exploration of heterogeneous CGRA (compute, memory, network) design space for an application domain
- Architectural specification and compiler generation

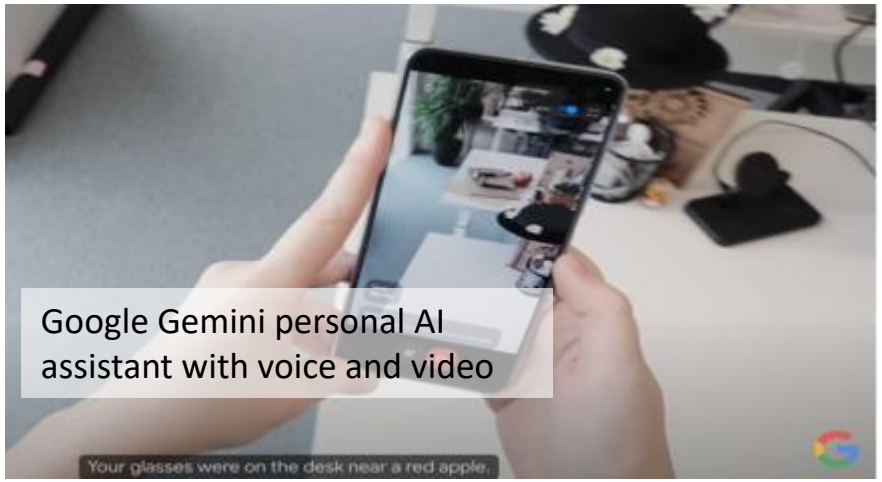




REVAMP: A Systematic Framework for Heterogeneous CGRA Realization. ASPLOS 2022
Toolchain: https://zenodo.org/record/5848404#.YgyrPTFByUk

Emergence of Native Multimodal LLMs
Unifying AI across modalities





- Seamless interaction with physical world via integration of multimodal data
- Process and generate text, images, audio, and video within a single model
- Support complex tasks requiring multi-modal reasoning
- Perfect opportunity for Edge AI

Mind the Gap: From Smaller to True Edge AI

- Emergence of more efficient small language models that are quite capable
- Model parameters: > 1 trillion in GPT-4 (2023) to 6.6 billion in Phi-3.5 (2024)
  Quality score: 86.4 in GPT-4 to 78.9 in Phi-3.5
- But…..even these smaller models are too big for edge AI

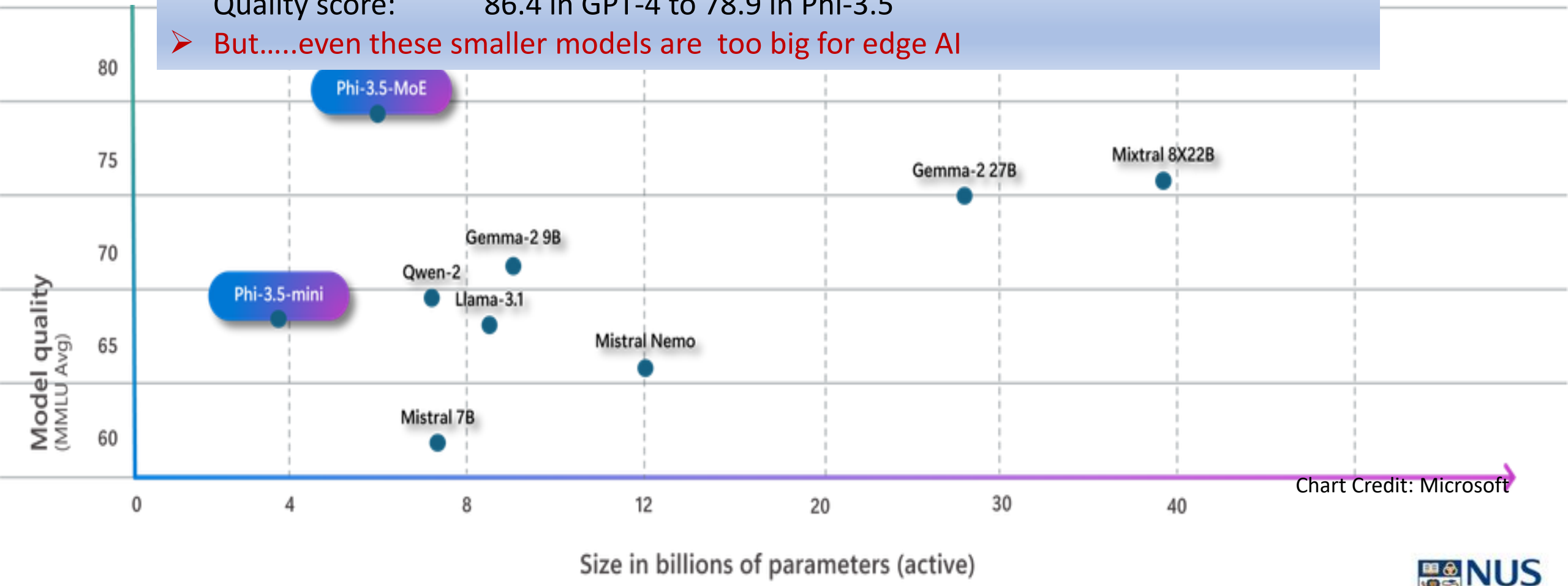
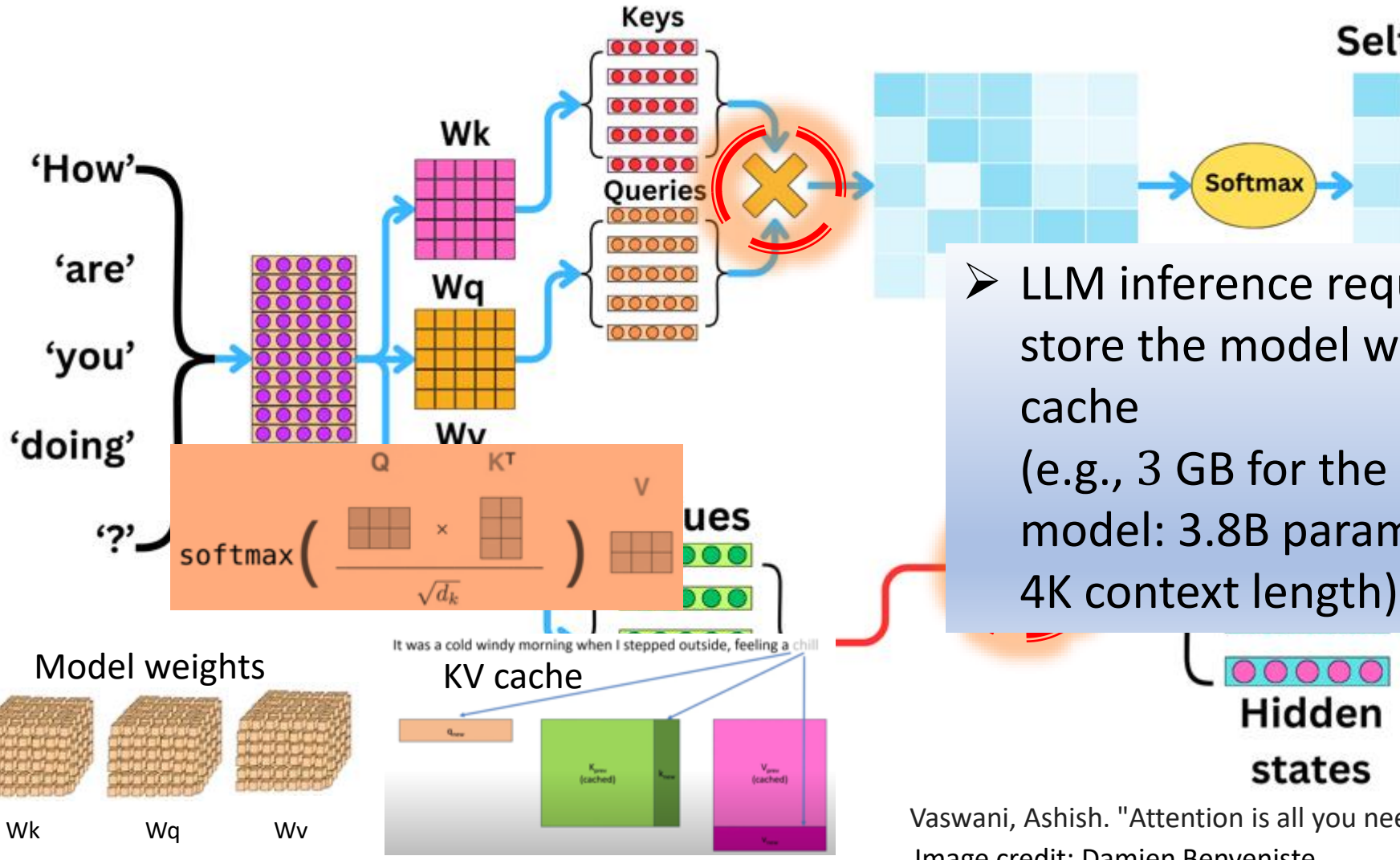


Chart Credit: Microsoft

MatMult is all you need

Attention is the cornerstone of LLMs enabling them to intelligently focus on the most relevant parts of any input



- LLM inference requires **huge memory** to store the model weights and key-value cache
  (e.g., 3 GB for the smallest Phi-3 Mini model: 3.8B parameters, 4 bit quantized, 4K context length)

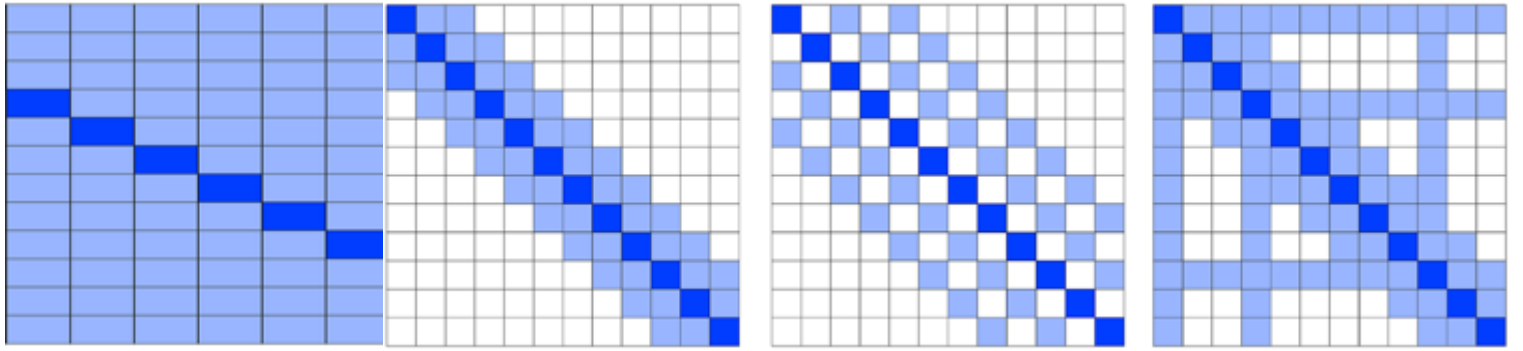
Vaswani, Ashish. "Attention is all you need." NeurIPS 2017

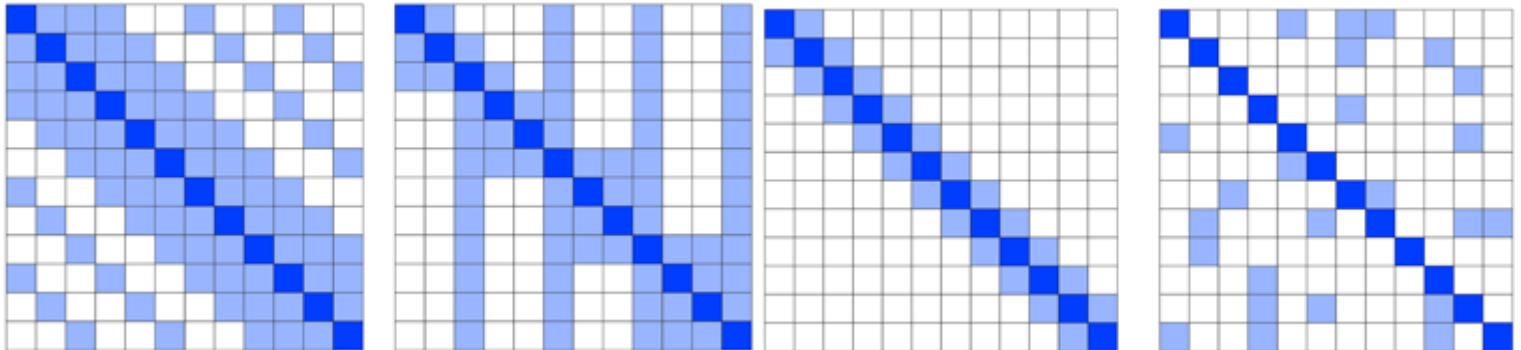
Image credit: Damien Benveniste

A Diverse Landscape of Sparse Attentions

Attention computation scales quadratically with sequence length, requiring $O(n^2)$ operations for n tokens

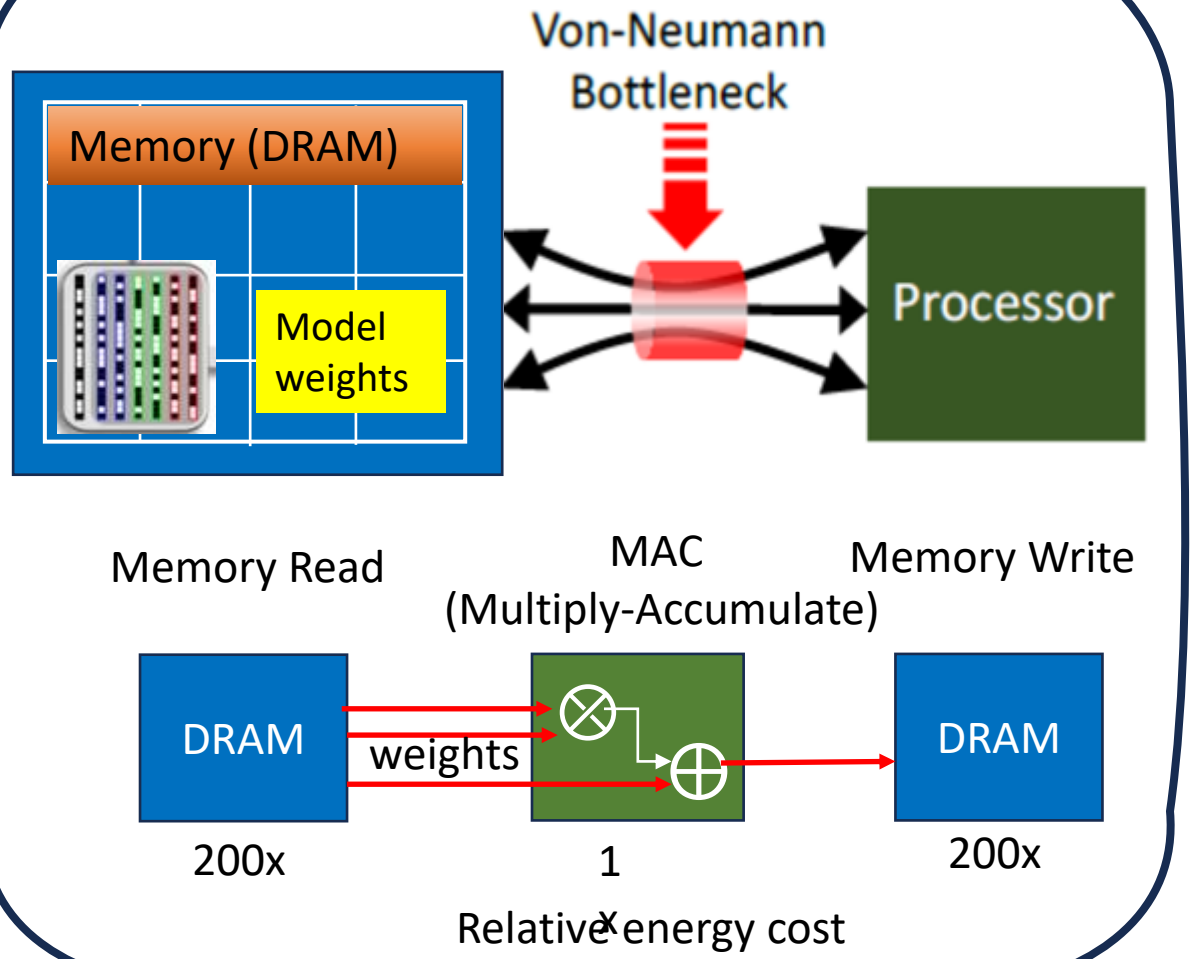
- Advanced attention mechanisms leverage diverse **sparse matrix computations** with specialized patterns for enhanced efficiency
- Each type of attention mechanism requires **unique dataflow**



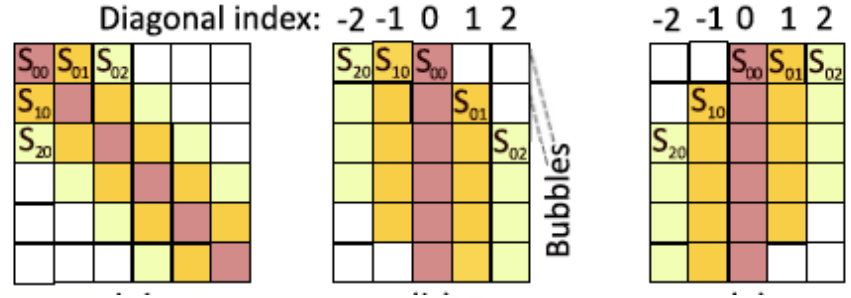


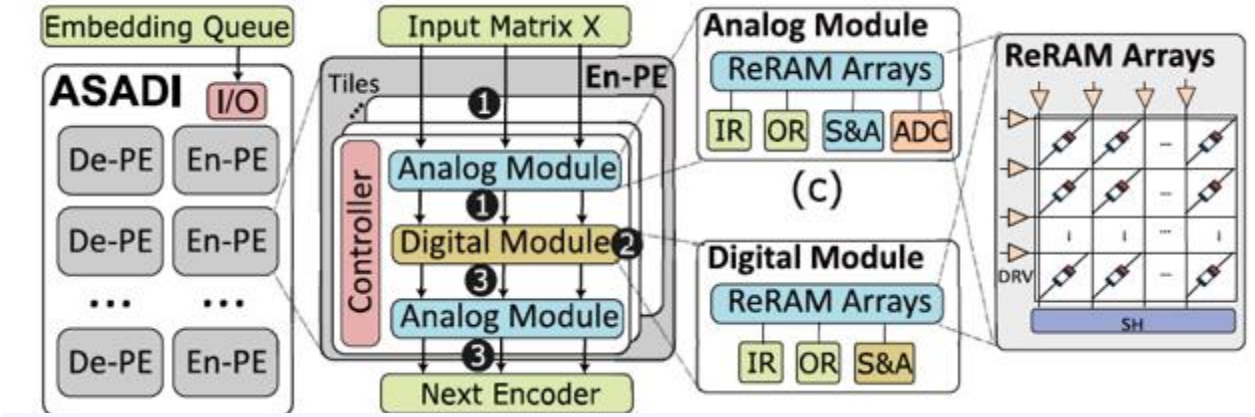
Memory transfer requires 200x more energy than compute

Von Neumann Architecture (Decoupled memory-compute)

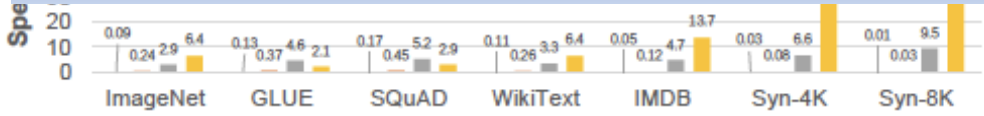


In-memory Computing (Integrated memory-compute)



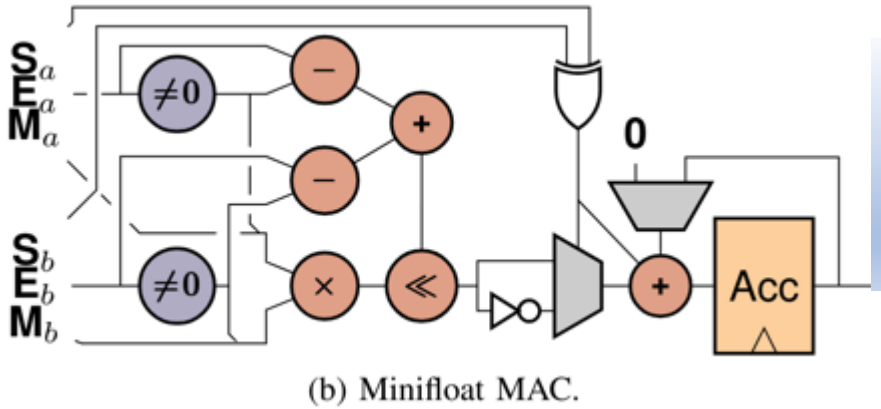


ASADI: ReRAM in-situ computing for diagonal attention can achieve 13x – 2470x speedup over GPU



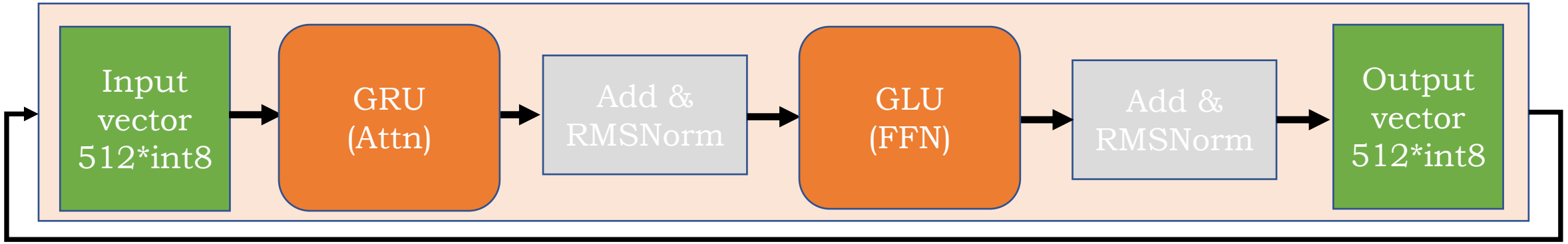
ASADI: Accelerating Sparse Attention using Diagonal-based In-situ Computing. HPCA 2024

Quantized Attention: Trim Bits, Enable Edge AI



(b) Minifloat MAC.

- 4-bit Mini-Floats provide comparable accuracy to 32-bit floating point in transformer models



Complete on-chip storage (BRAM + URAM) of weights on U280 FPGA

MatMult-Free Ternary LLM (1.58bits): *Work-in-progress*

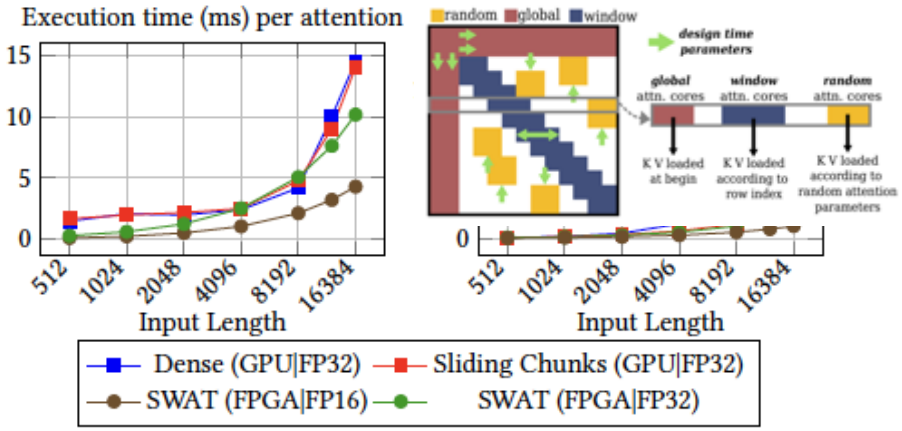
370M parameters: ~5K tokens/sec @ 21W vs. ~100 tokens/sec @ 200W

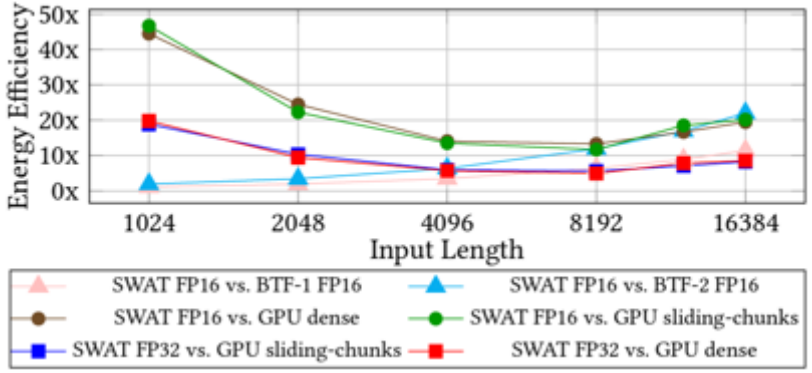
Shedding the Bits: Pushing the Boundaries of Quantization with Minifloats on FPGAs. FPL 2024

* Scalable MatMul-free Language Modeling, Rui-Jie Zhu et al. arXiv 2024

NUS National University of Singapore

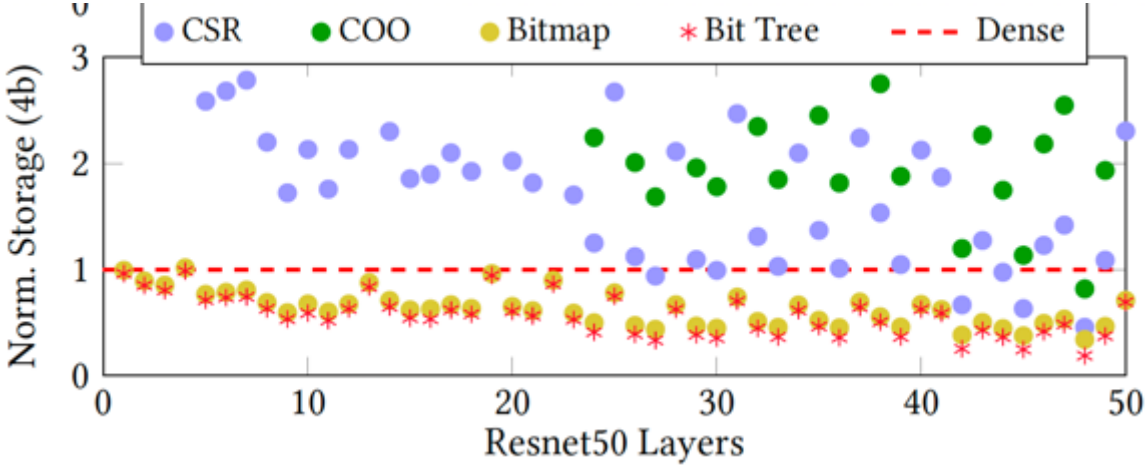
FPGA Acceleration of Sliding-Window Attention

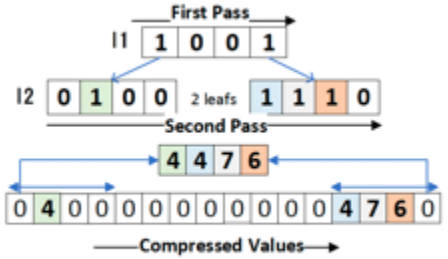




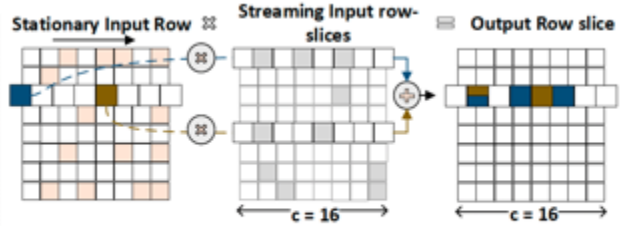
SWAT: FPGA Sliding-Window Attention Design achieves 15x energy-efficiency over GPU

ASIC accelerator for Sparse Matrix ML Workload

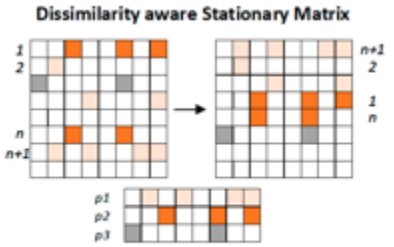




(a) Bit-Tree structure storage and traversal



(b) Row-Wise Product to Produce one row-slice of a Chunk



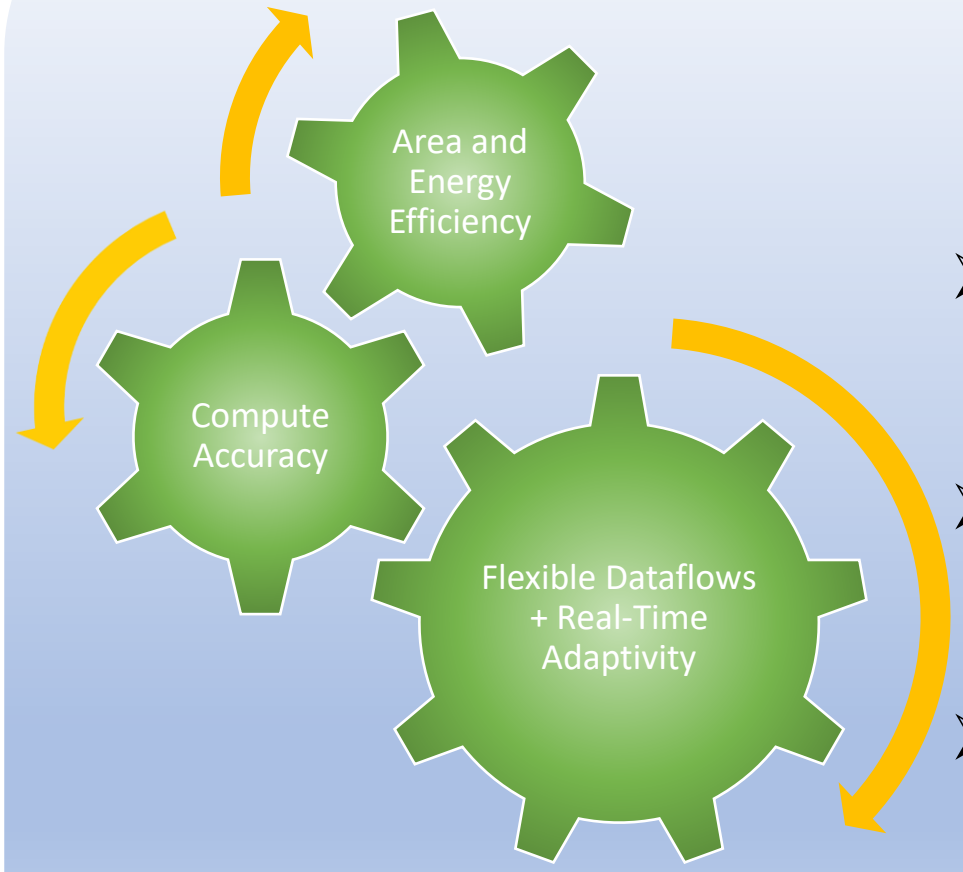
(c) Streaming Memory Access Patterns

ZeD: ASIC accelerator achieves 3.2x performance-per-area improvement over SOTA for sparse ML workloads

SWAT: Scalable and Efficient Window Attention-based Transformers Acceleration on FPGAs. DAC 2024
ZeD: A Generalized Accelerator for Variably Sparse Matrix Computations in ML. PACT 2024

NUS National University of Singapore

Challenge: CGRA for Arbitrary Sparse Dataflows



- Minimalist reconfigurable array that adapts to diverse attentions without high area, energy, speed costs
- Smart algorithm that maps arbitrary attention dataflows on the fabric
- Maintain high parallel processing while handling complex dataflow dependencies
- Achieve software-level accuracy with low-precision hardware technology

Acknowledgments

- **Li-Shiuan Peh**
- Anh Tuan Do
- Chen Liang
- Tan Cheng
- Manupa Karunaratne
- Wang Bo
- Aditi Kulkarni
- Dhananjaya Wijerathne
- Zhaoying Li
- Thilini Kaushalya Bandara
- V Vanchinathan
- Dan Wu
- Vishnu Paramasivam
- Chong Yi Sheng
- Pranav Dangi
- Zhenyu Bai
- Huize Li

National Research Foundation
Prime Minister's Office Singapore

Ministry of Education
Singapore

A*STAR Institute of Microelectronics

AMD

XILINX

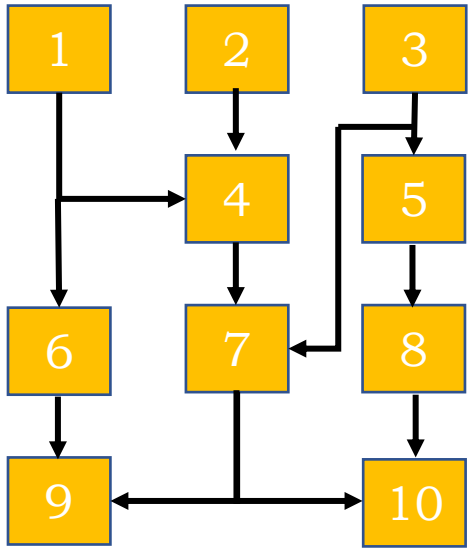
Black Sesame Technologies

Thank You

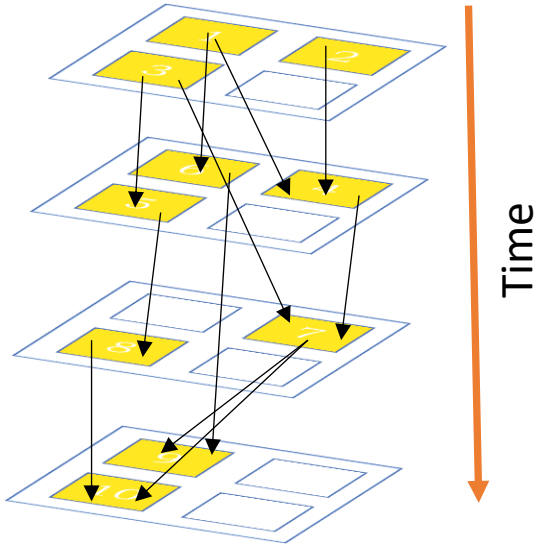
NUS National University of Singapore

CGRA versus FPGA

- CGRA offers word-level reconfiguration while FPGA offers bit-level reconfiguration
  - Reduces reconfiguration overhead providing better performance and lower power
- Built-in arithmetic functions improves energy-efficiency/frequency, makes mapping simpler
  - Closer to compilation for VLIW architecture than HLS for ASIC/FPGA
  - Easier programmability for software developers compared to FPGA
- CGRA offers spatio-temporal mapping while FPGA offers only spatial mapping
  - Reduces area cost, which is important for edge devices



Spatial Mapping



Spatio-temporal Mapping

CGRA vs FPGA

		Normalized w.r.t. FPGA		
CGRA	**Application**	**Power**	**Perf.**	**Power Efficiency**
Plasticine (ISCA'17)	CNN	1.20	95.1	76.9
	GEMM	1.40	33.0	24.4
	BFS	0.60	7.3	11.4
Amber (VLSI'22)	Resnet Conv2_x	0.11	1.0	9.2
	Resnet Conv3_1	0.11	1.0	9.1