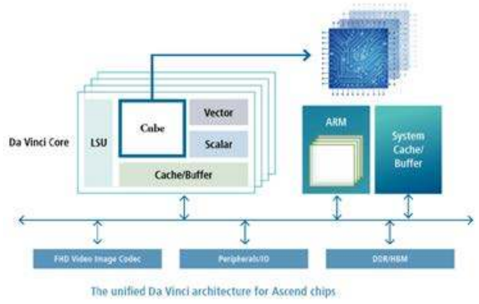
DaVinci: A Scalable Architecture for Neural Network Computing

Heng Liao, Jiajin Tu, Jing Xia, Xiping Zhou

2019-07

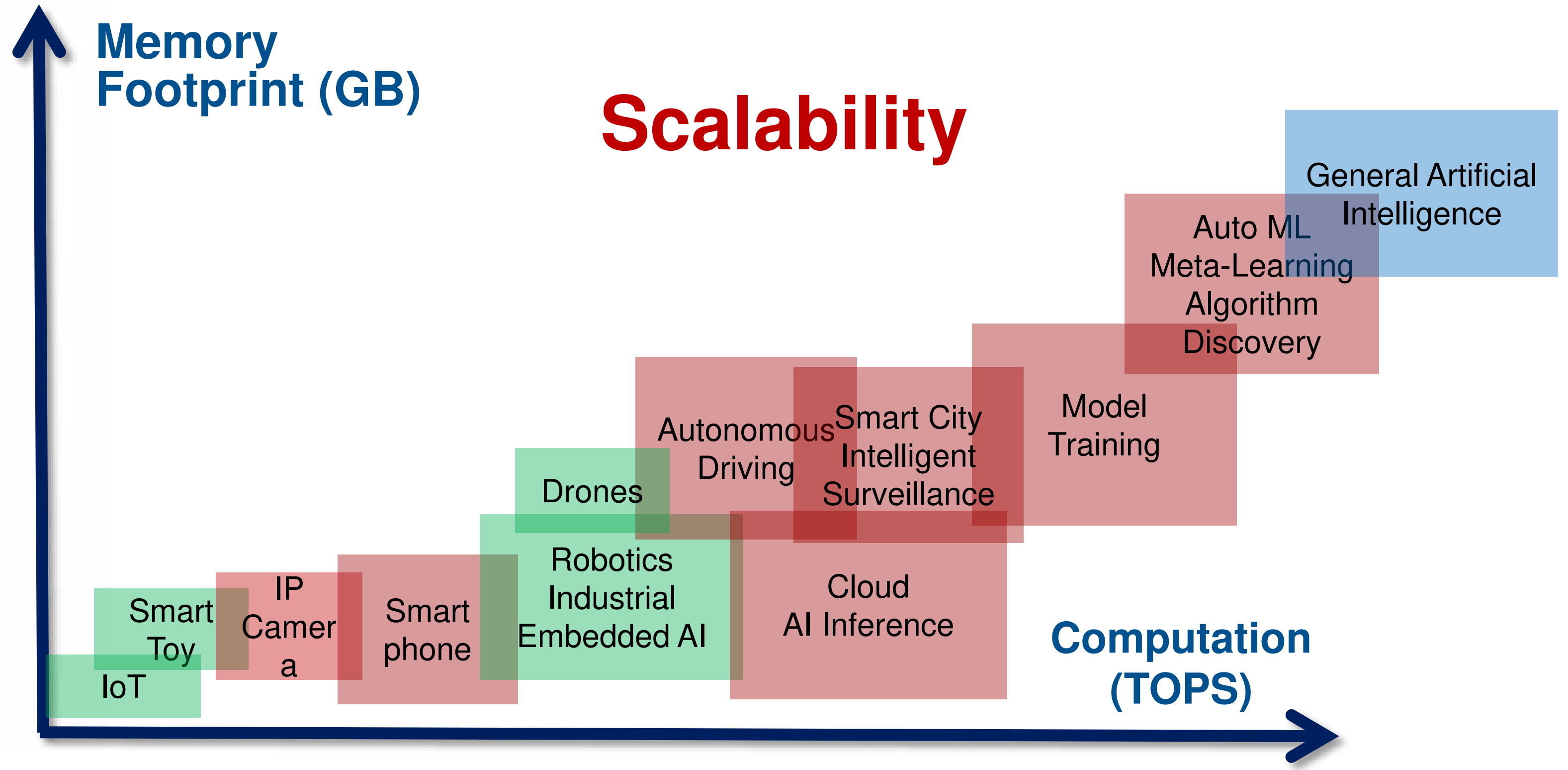
HUAWEI

Key element to enable intelligence in physical devices



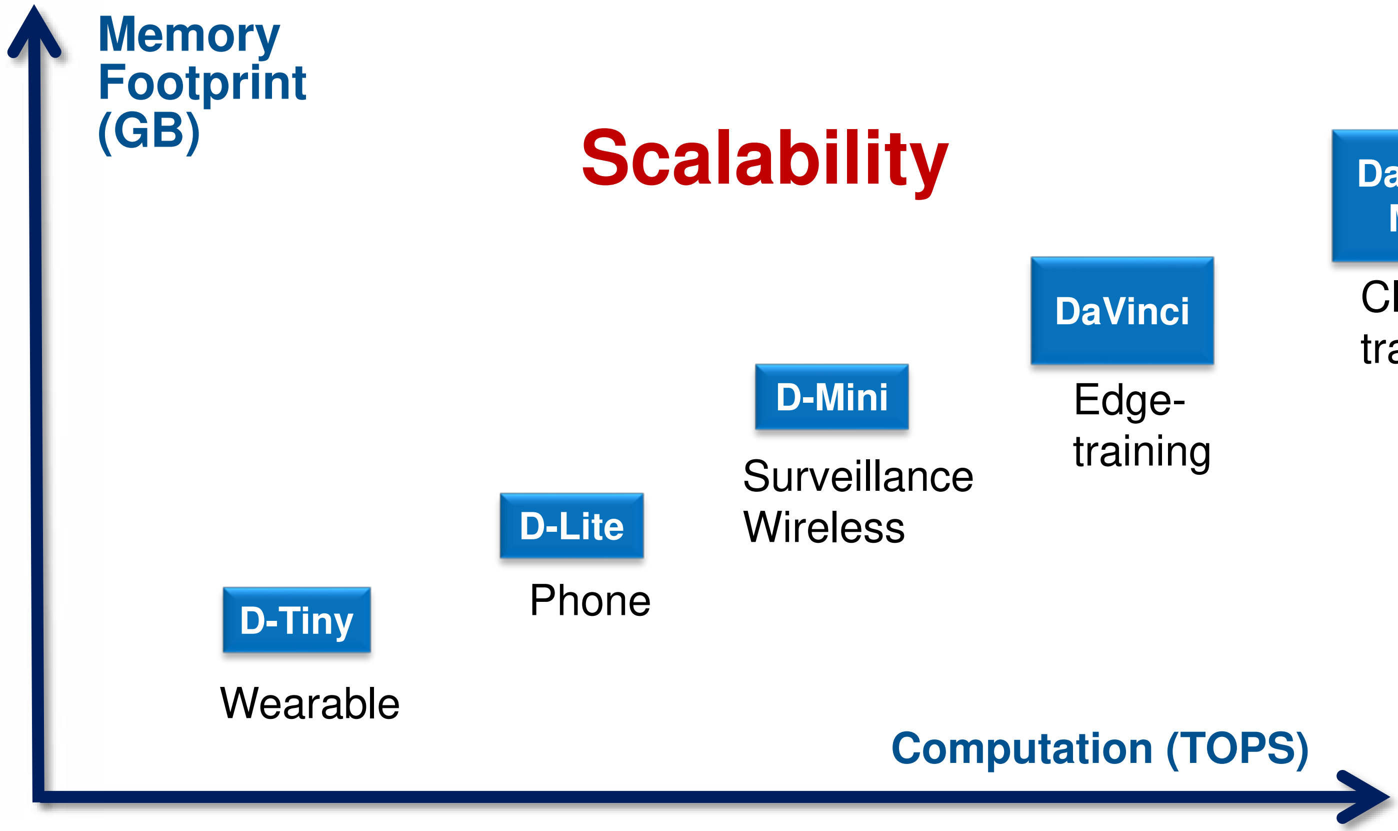
Control, Process, Transfer, Store ……

Ubiquitous AI computation

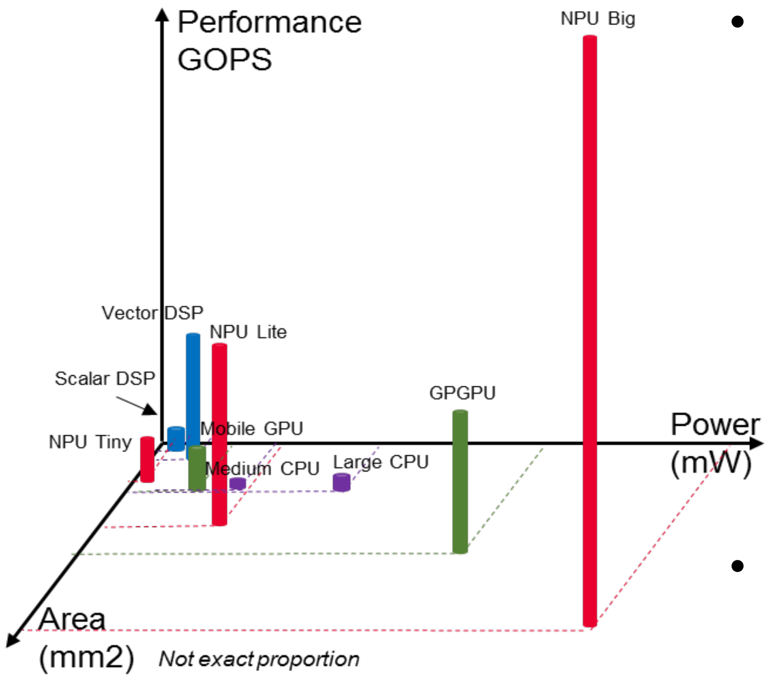

Memory Footprint (GB)
Scalability
General Artificial Intelligence
Auto ML Meta-Learning Algorithm Discovery
Model Training
Autonomous Driving
Smart City Intelligent Surveillance
Drones
Robotics Industrial Embedded AI
Cloud AI Inference
Smart phone
IP Camera
Smart Toy
IoT
Computation (TOPS)

Applications across ~10^6 performance range

Ubiquitous AI computation

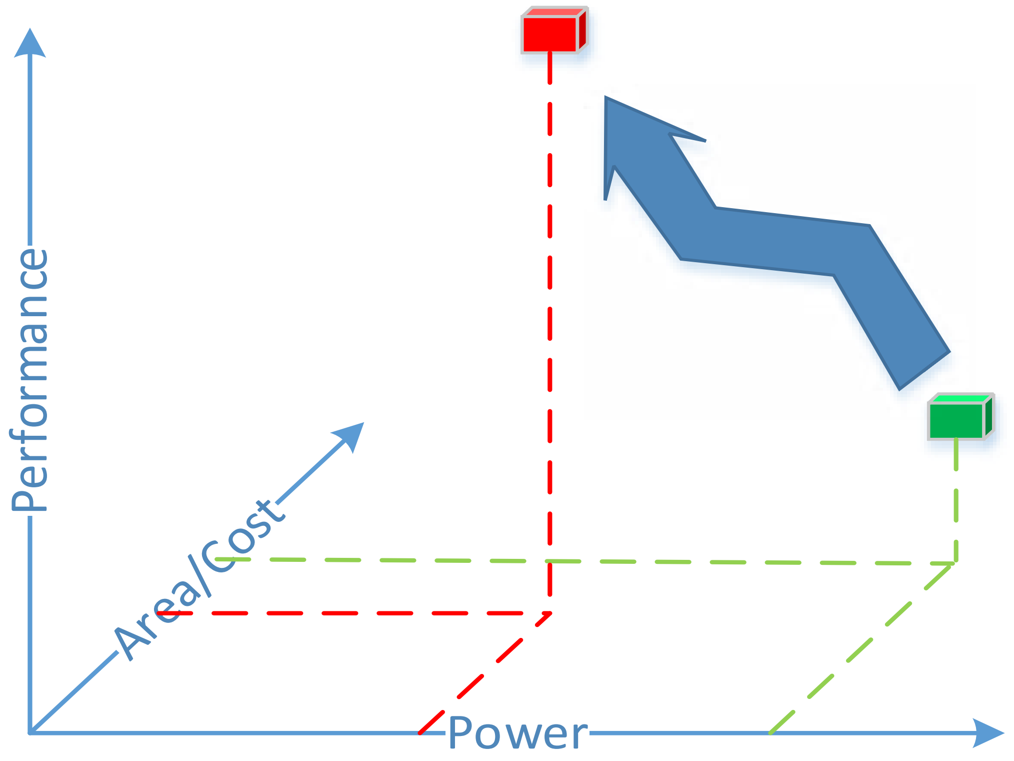


Rich Variety of Computing architectures in Huawei Portfolio



- Wide range of performance & efficiency
  - CPU: General purpose
  - GPU: Graphics
  - NPU: DNN
  - ISP: Camera sensor pipeline
  - DSP: Camera post processing, AR
  - VPU: Vision Processing Unit
  - NP: Network Processor
- Each category represents a different PPA curve

Target: Search for Optimal PPA in Design Space

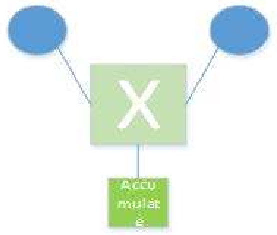


Architecture Overview of DaVinci

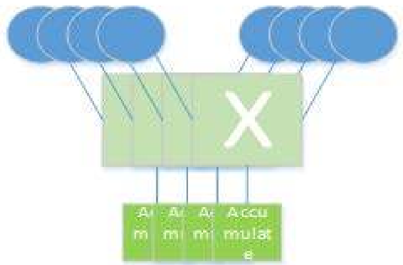
Building Blocks and their Computation Intensity

1D Scalar Unit + **2D Vector Unit** + **3D Matrix Unit**

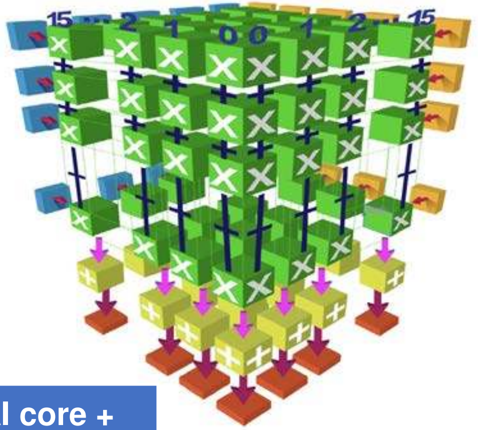
Full flexibility



Rich & efficient operations



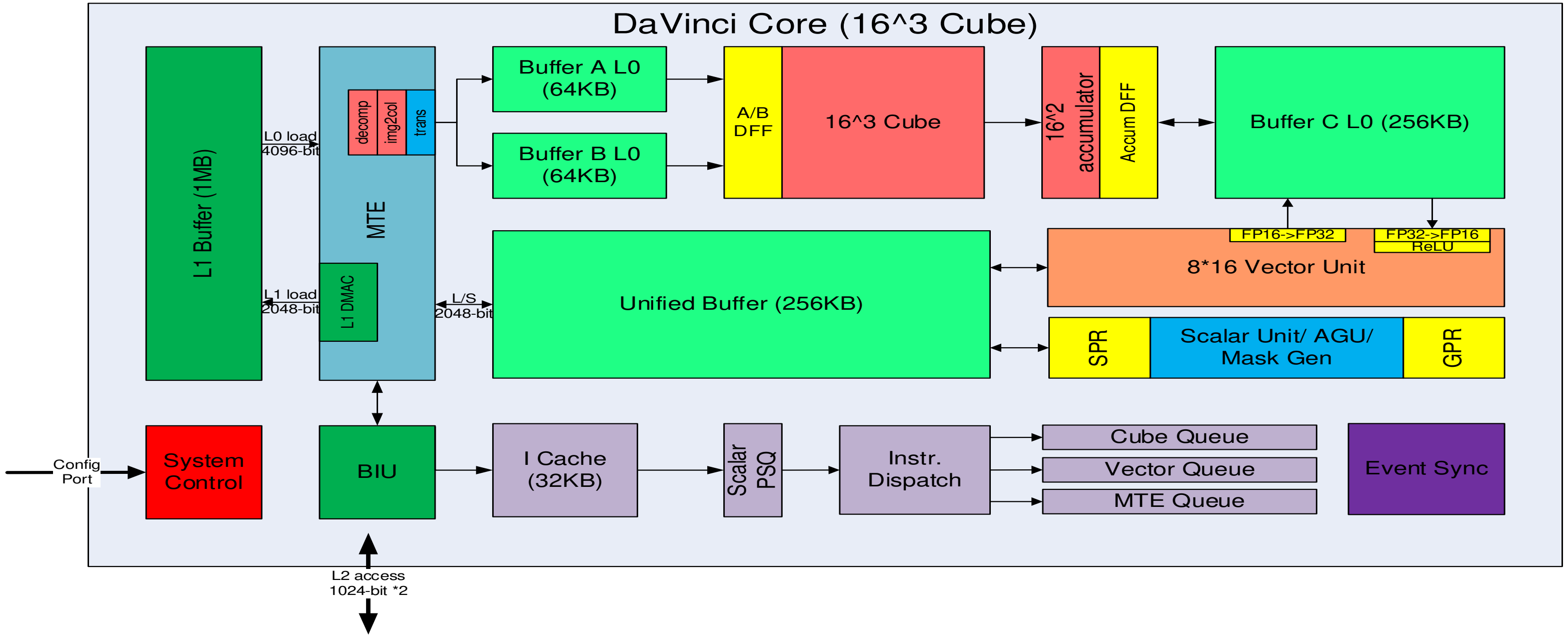
High intensity



N	N^2	N^3
1	1	1
2	4	8
4	16	128
8	64	512
16	**256**	**4096**
32	1024	32768
64	4096	262144

	GPU + Tensor core	AI core + SRAM
Area (normalized to 12 nm)	5.2mm^2	13.2mm^2
Compute power	1.7Tops fp16	8Tops fp16

DaVinci Core

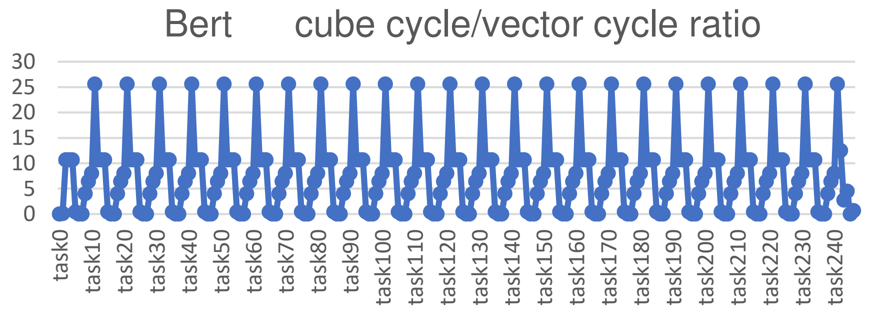


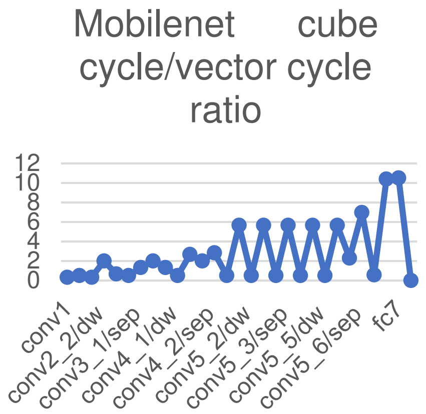
- **Cube**: 4096(16^3) FP16 MACs + 8192 INT8 MACs
- **Vector**: 2048bit INT8/FP16/FP32 vector with special functions (activation functions, NMS- Non Minimum Suppression, ROI, SORT)
- Explicit memory hierarchy design, managed by MTE

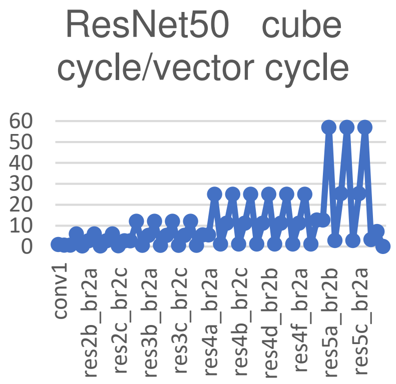
Micro Architecture Configurations

Core Version	Cube Ops/cycle	Vector Ops/Cycle	L0 Bus width	L1 Bus Width	L2 Bandwidth
Davinci Max	8192	256	Match Execution Units Not bottleneck	A:8192 B:2048	910: 3TB/s ÷32 610: 2TB/s ÷8 310: 192GB/s÷2
Davinci Lite	4096	128		A:8192 B:2048	38.4GB/s
Davinci Tiny	512	32		A:2048 B:512	None
	Set the performance baseline	Minimize vector bound		Ensure this is not a bound	Scarce, limited by NoC, avoid bound where possible

Resource Matching ---- Vector

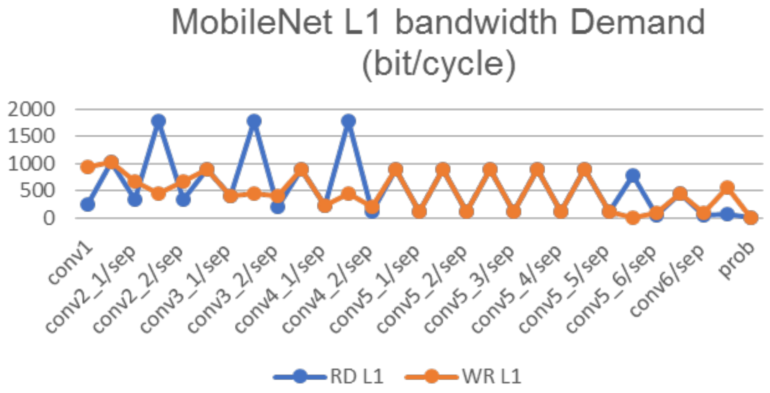


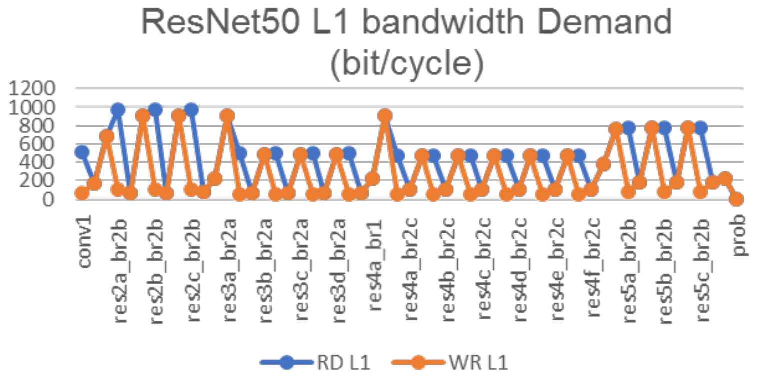




- Balance Computation Power between CUBE vs Vector by overlapping its computation time with Vector
- Carefully allocated the number of MACs in CUBE and Vectors
- Support multiple matrices multiply vector operations in CUBE.
- Expand the width of data bus between L1 feature map buffer and CUBE

Resource Matching ---- Memory Hierarchy





Davinci carefully balance the memory hierarchy design to avoid bandwidth become bottleneck at key locations.

Examples:

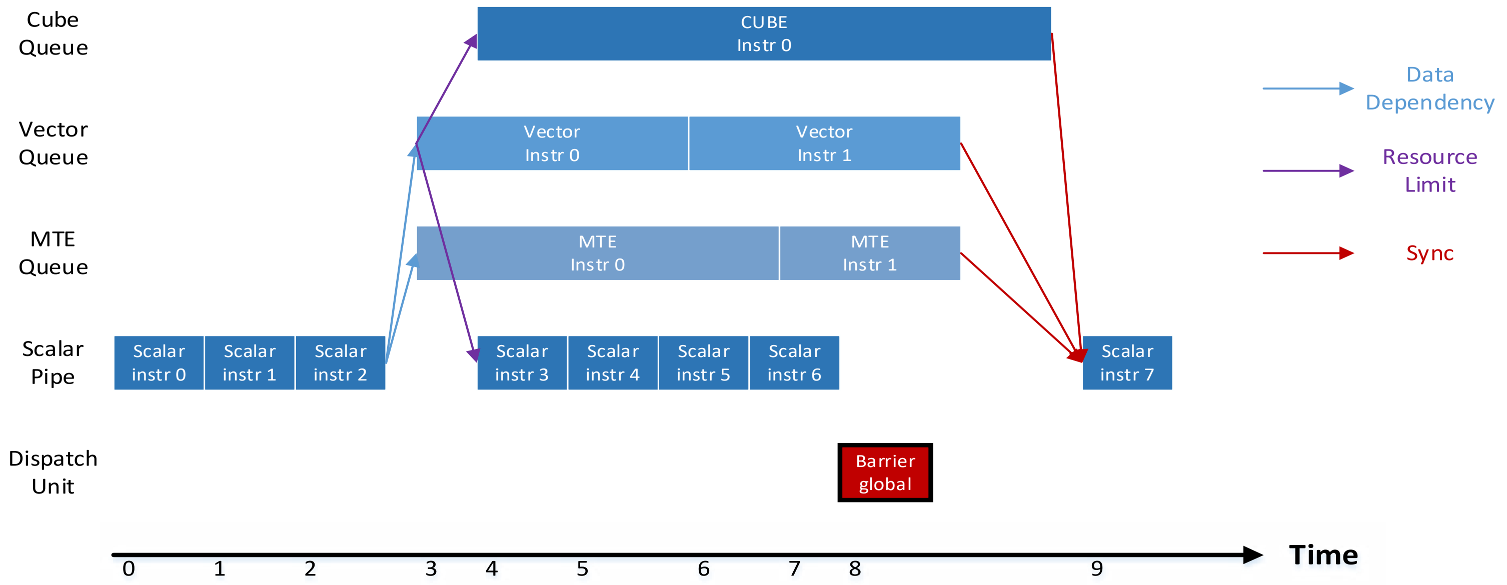
- Reduce the DDR bandwidth requirement by reusing data within L1, L0A, L0B.
- Asymmetric bandwidth provided according to the nature of computation
  - L1 -> L0A bandwidth >> L1->L0B bandwidth, because W*H could be much bigger than output channel number

More Challenges of DaVinci

Overview of the DSA Developer Stack

Level 3 Library (written by novice programmer)		TBE LIB
Level 3 Compiler (mathematical programming model)	TVM/XLA	TBE
Level 2 Library (written by skilled programmer)	CudaNN/ CuBLAS	TIK LIB
Level 2 Compiler (parallel/kernel programming model)	Cuda/OpenCL	TIK
Level 1 Library (written by expert)		CCE Lib
Low Level 1 Compiler (Intrinsic C) (Architecture defined programming)		CCE C
Instruction Set Architecture	GPU	NPU

Challenge 1: How to Enable Parallelism with Single Thread

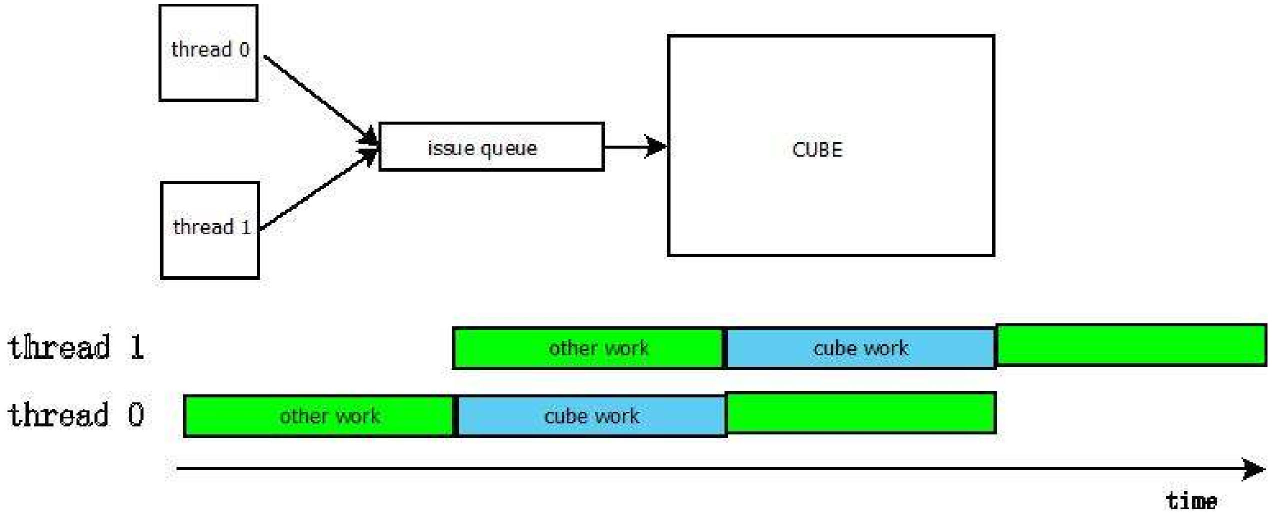


- Programmer is comfortable with the sequential code
- Davinc’s C like programming interface (CCE) let programmer to control the parallelism explicitly .

Solution with Multi-thread?

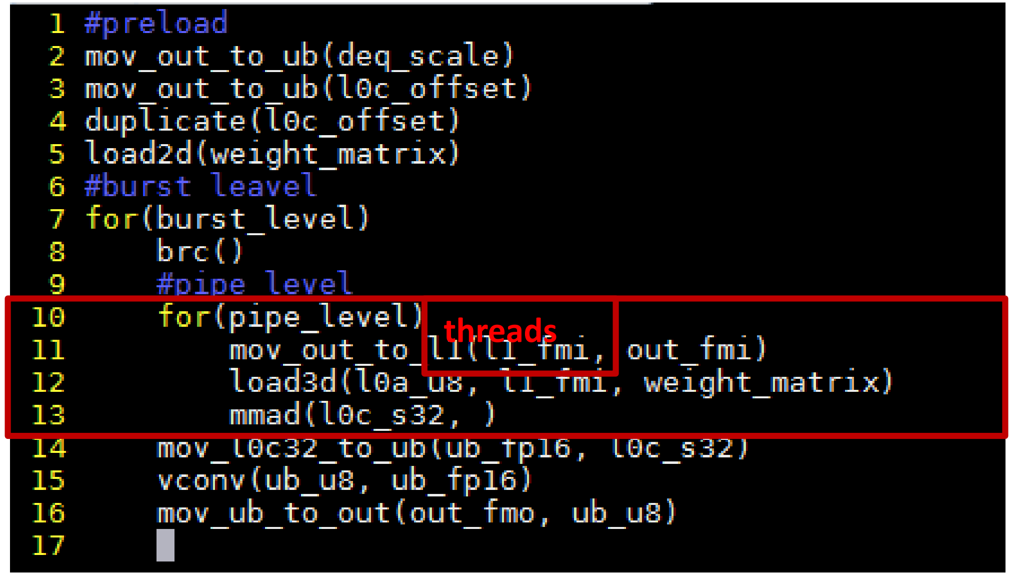
How about support hardware multi-thread feature?

- The code in each thread is sequential
- CUBE is a share resource between threads
- It has hardware cost



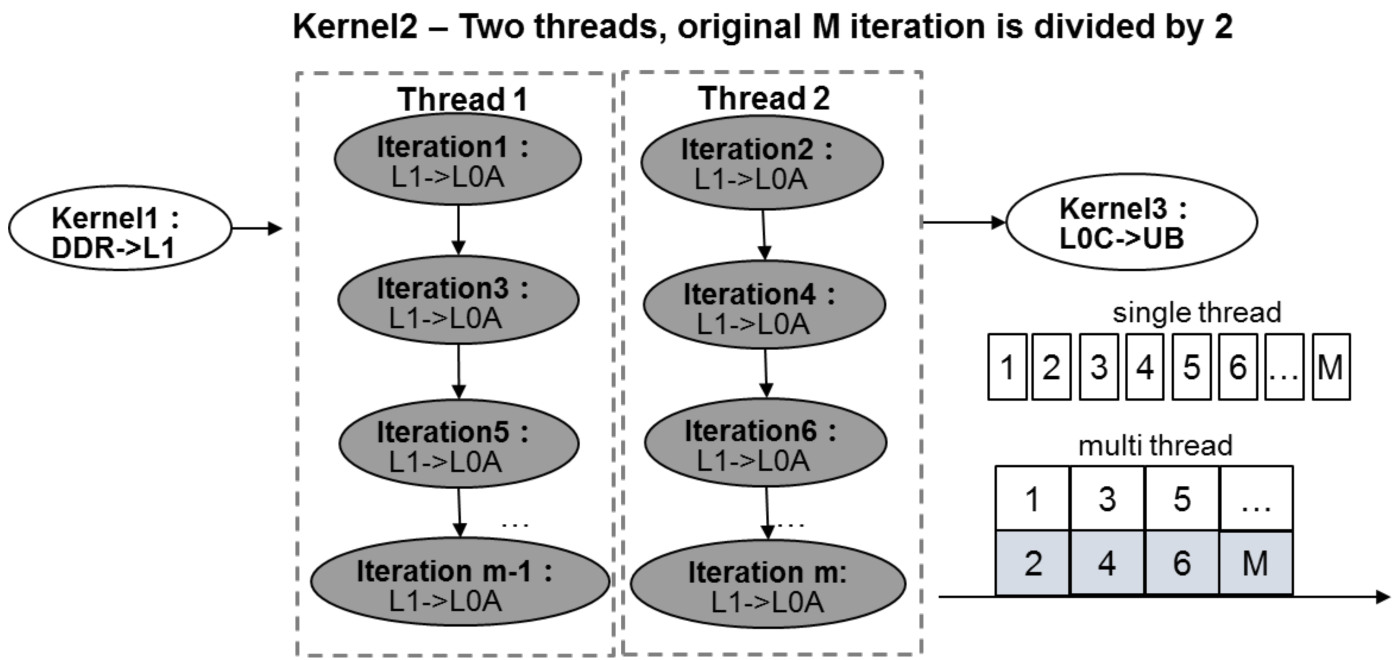
How does it work - TIK

- Typical sequential Davinci **c**ode is a combination of nested FOR loops
- Software multi-thread can be added to any FOR loop body (iterator kernel).



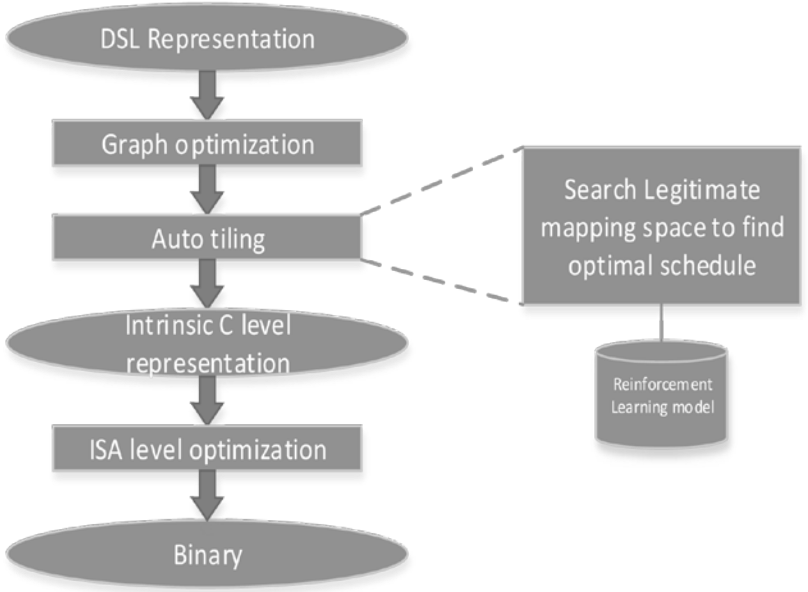
```
#preload
mov_out_to_ub(deq_scale)
mov_out_to_ub(l0c_offset)
duplicate(l0c_offset)
load2d(weight_matrix)
#burst leavel
for(burst_level)
    brc()
    #pipe level
    for(pipe_level)
        mov_out_to_l1(l1_fmi, out_fmi)
        load3d(l0a_u8, l1_fmi, weight_matrix)
        mmad(l0c_s32, )
    mov_l0c32_to_ub(ub_fp16, l0c_s32)
    vconv(ub_u8, ub_fp16)
    mov_ub_to_out(out_fmo, ub_u8)
```

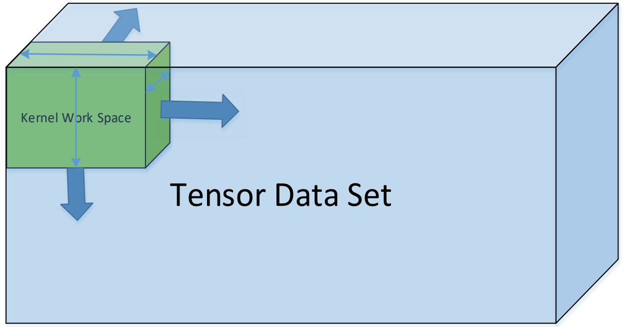
Programmer view of multi-thread



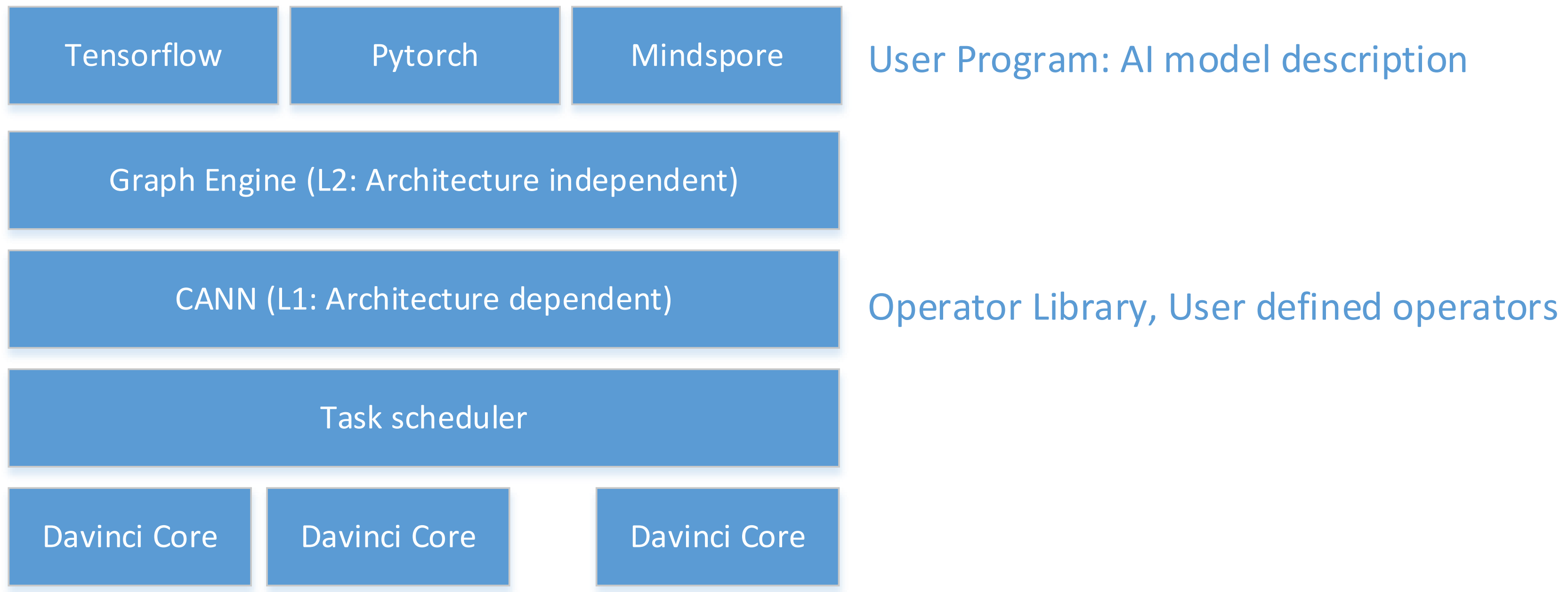
Advanced Compiler Techniques

- Architecture independent DSL→ C → Binary lowering process
- Traversal order determines data reuse factor
- Millions of legitimate mappings
- Find optimal mapping to
    - bridge the 2,000x memory bandwidth gap


DSL Representation
Graph optimization
Auto tiling
Intrinsic C level representation
ISA level optimization
Binary
Search Legitimate mapping space to find optimal schedule
Reinforcement Learning model



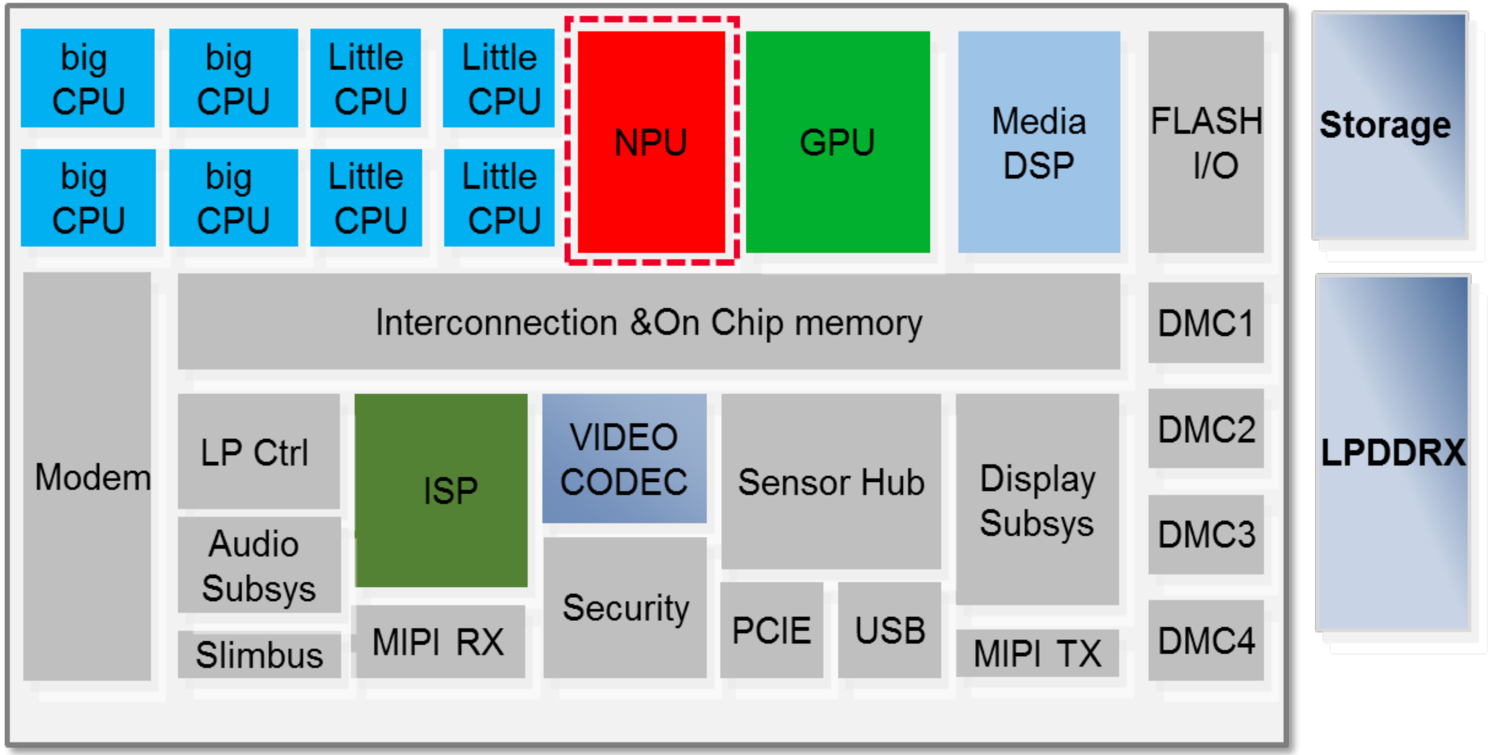
Putting All This Together


Tensorflow
Pytorch
Mindspore
User Program: AI model description
Graph Engine (L2: Architecture independent)
CANN (L1: Architecture dependent)
Operator Library, User defined operators
Task scheduler
Davinci Core
Davinci Core
Davinci Core

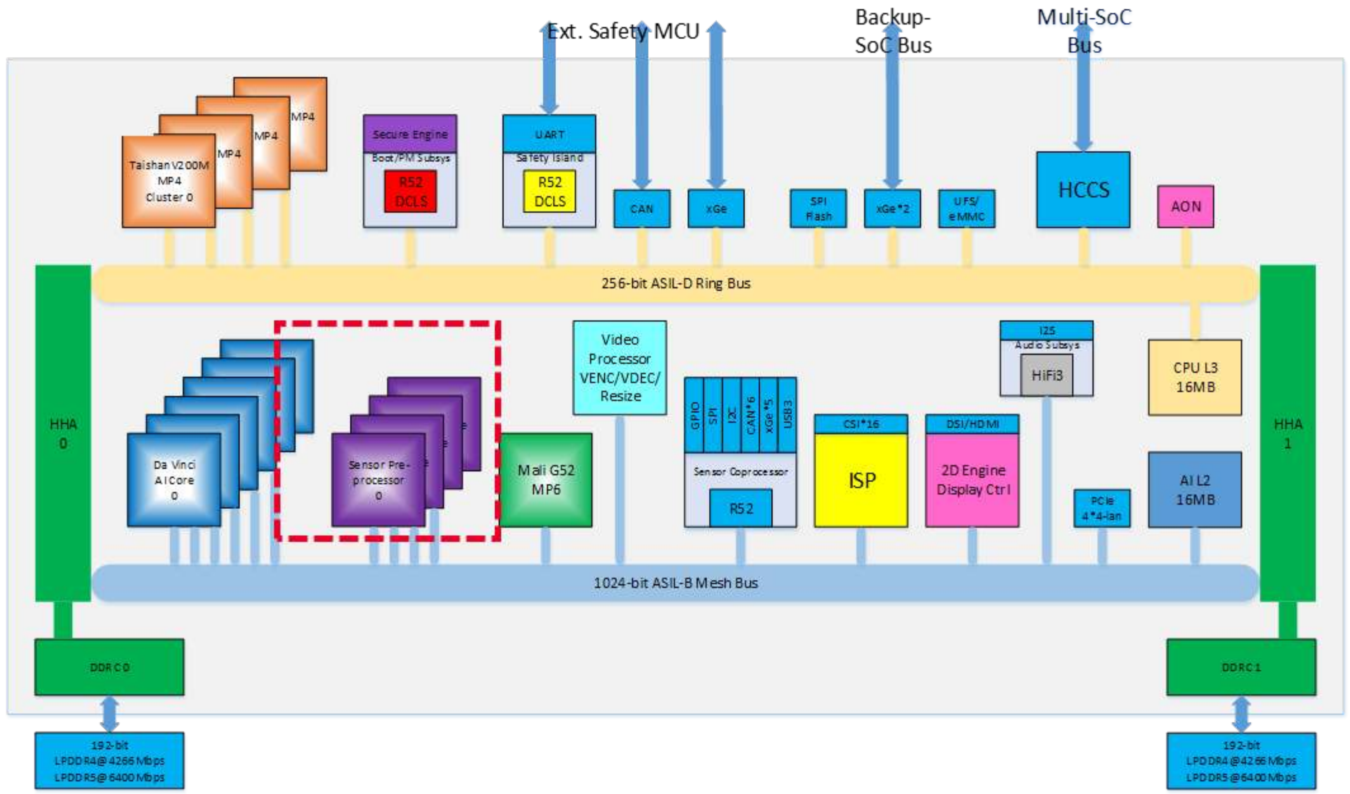
- User program AI model using familiar frameworks
- Extends operator library when necessary
- The tasks are executed in a single node, or over a network cluster

Davinci AI Core in SoCs

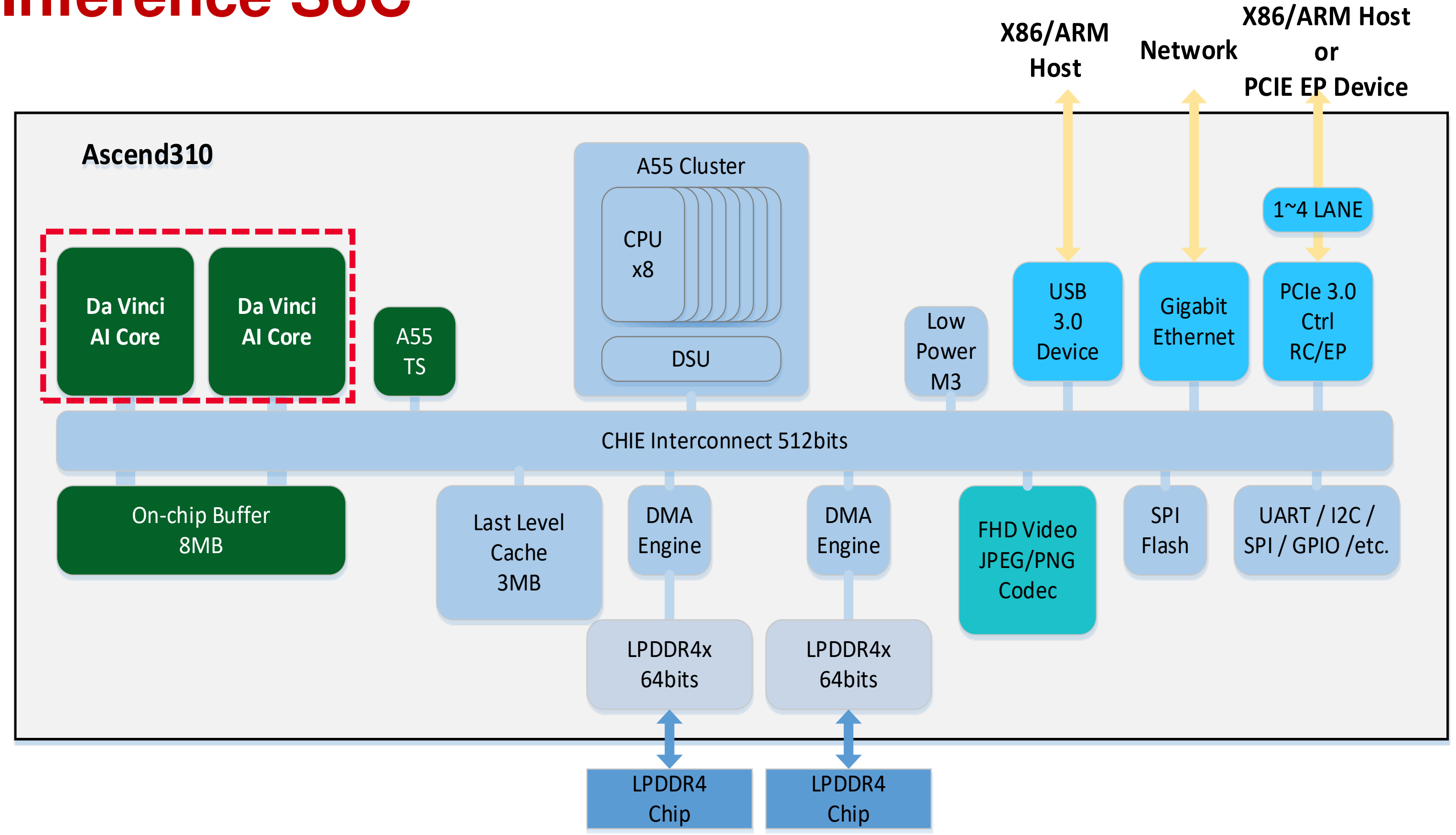
Mobile AP SoC



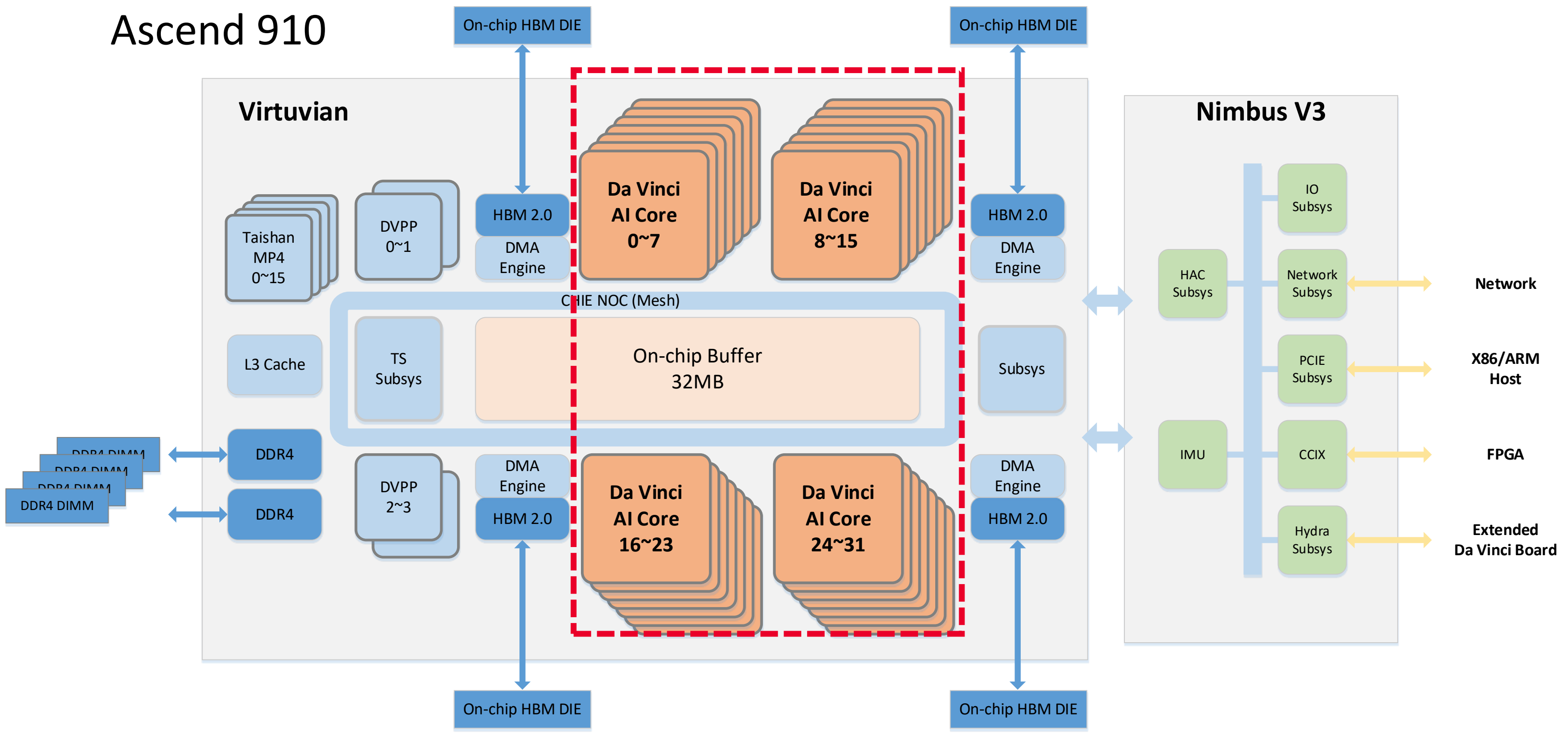
Automotive SoC



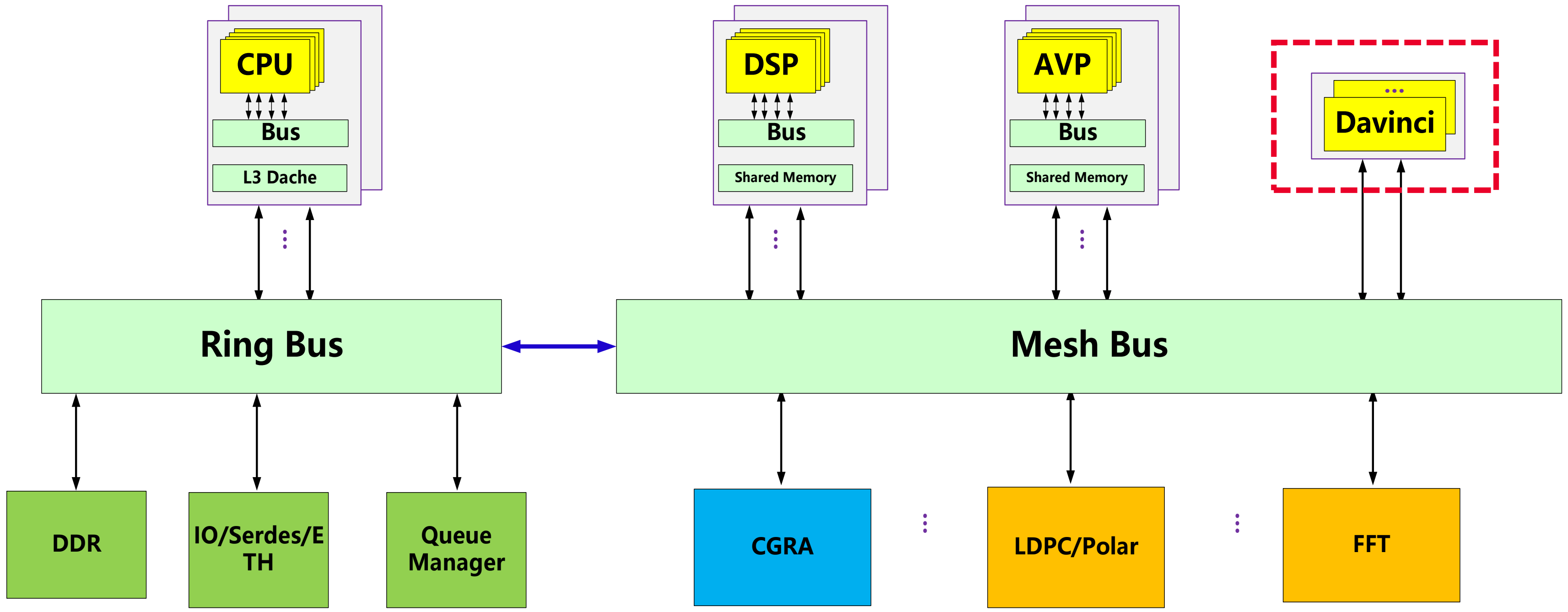
AI Inference SoC



AI Training SoC



Wireless SoC

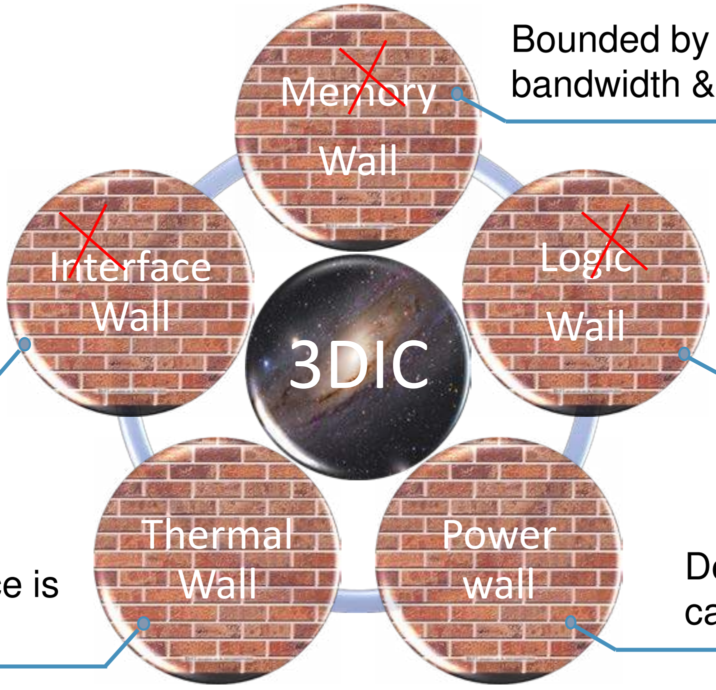


Alleviate Memory Wall and Slow down of Moore’s Law

Memory Wall & I/O Wall

	Bandwidth @	Bandwidth	Bandwidth Ratio	Challenges
Compiler Innovation	Execution Engine (512TOPS)	2048T Byte/Sec	1	Can build faster EU, but no way to feed data
	L0 Memory	2048T Byte/Sec	1/1	Very wide datapath, hard to do scatter-gather Inner-loop data reuse
	L1 Memory	200T Byte/Sec	1/10	Intra-kernel data reuse
Cluster Innovation	L2 Memory	20T Byte/sec	1/100	Inter-kernel data reuse
	HBM Memory	1T Byte/sec	1/2000	HBM size limits memory footprint
	Intra Node bandwidth	50G Byte/sec	1/40000	Scale-up node increase memory footprint, but severely bandwidth constrained
	Inter Node bandwidth	10G Byte/sec	1/200000	Model parallelism across nodes,severely bandwidth constrained

Technology challenges — Why do we need 3DIC



Bounded by memory bandwidth & capacity

Process scaling can not satisfy the need for more die area (many chips have reached full reticle size in 7+ process)

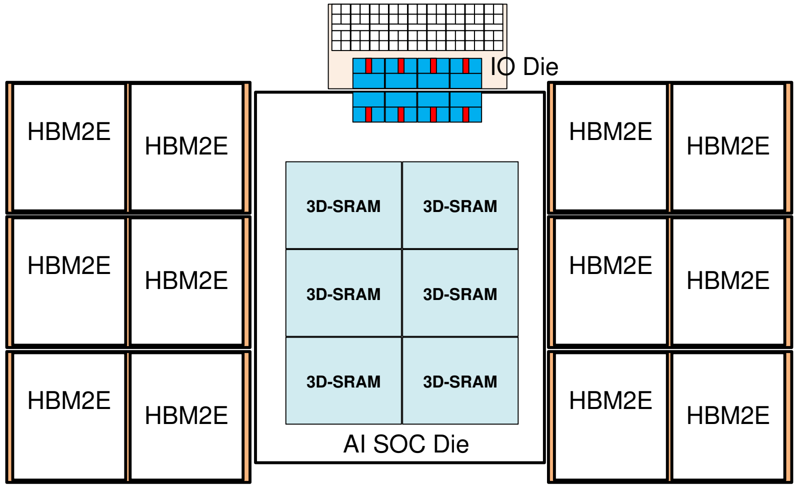
Interface IO speed does not scale with process node

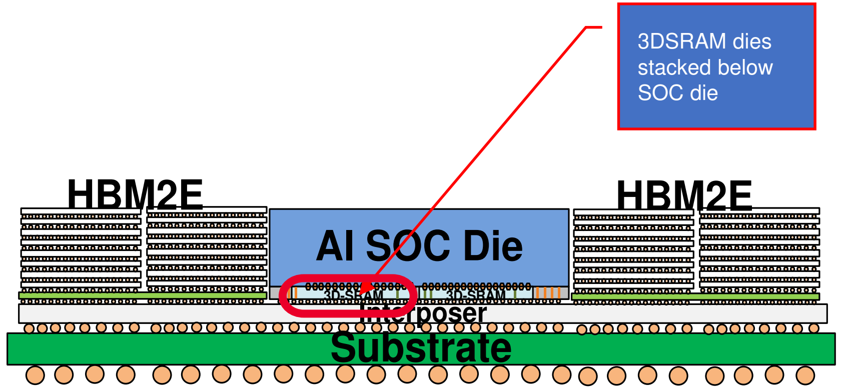
Ultimate system performance is thermal limited

Denard slow down causing power limits

- 3DIC can help alleviating memory wall, IO wall and logic wall.
- The search for new transistor to help power and thermal all.
- Architecture innovation to reach new grounds despite of all challenges

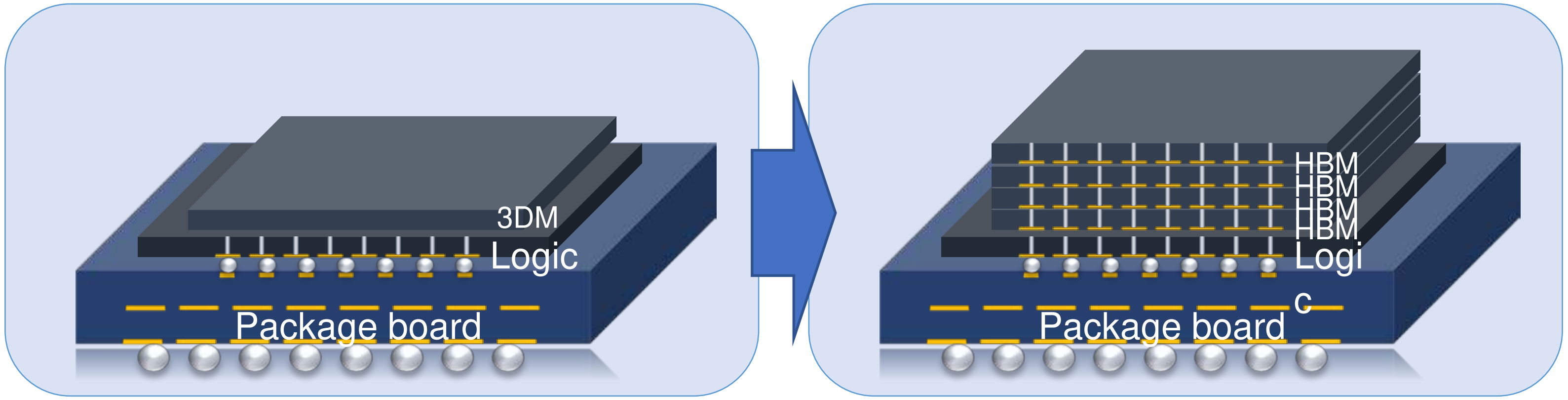
AI Training SoC: Logic + 3DSRAM + 12 HBM





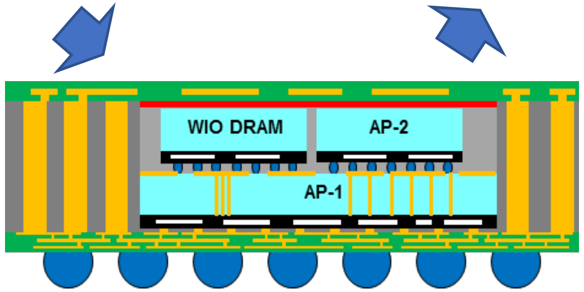
- Customized HBM2E with two Stacks to increase HBM bandwidth
- Large 3D-SRAM as AI core cache

Mobile AP: LoL + MoL



Step 1

- One logic die + 3D DRAM
- 3DM+POP LPDDR



Step 2

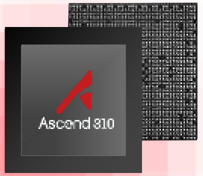
- Two logic die + 3D DRAM
- POP LPDDR

Step 3:

- Multi-layer 3D DRAM (remove POP LPDDR)
- Multi-layer Logic die

Physical Design of Davinci AI Chips

Ascend Architecture



Ascend 310

High Power Efficiency

Ascend-Mini
Architecture: DaVinci

FP16： 8 TeraFLOPS

INT8 ： 16 TeraOPS

16 Channel Video Decode – H.264/265

1 Channel Video Encode – H.264/265

Power： 8W

Process: 12nm

Ascend 910

High Computing Density

Ascend-Max
Architecture: DaVinci

FP16: 256 TeraFLOPS

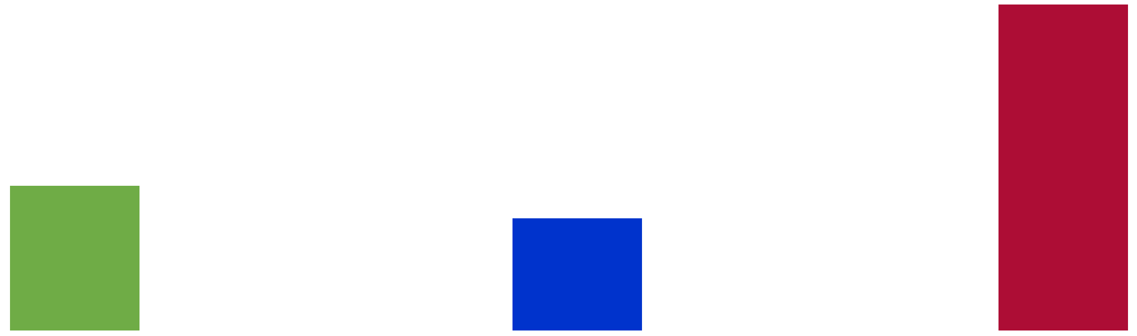
INT8: 512 TeraOPS

128 Channel Video Decode – H.264/265

Power: 350W

Process: 7+ nm EUV

Comparison of Computing Density



Tesla V100 (12 nm)

- 416mm²
- 120TFlops fp16

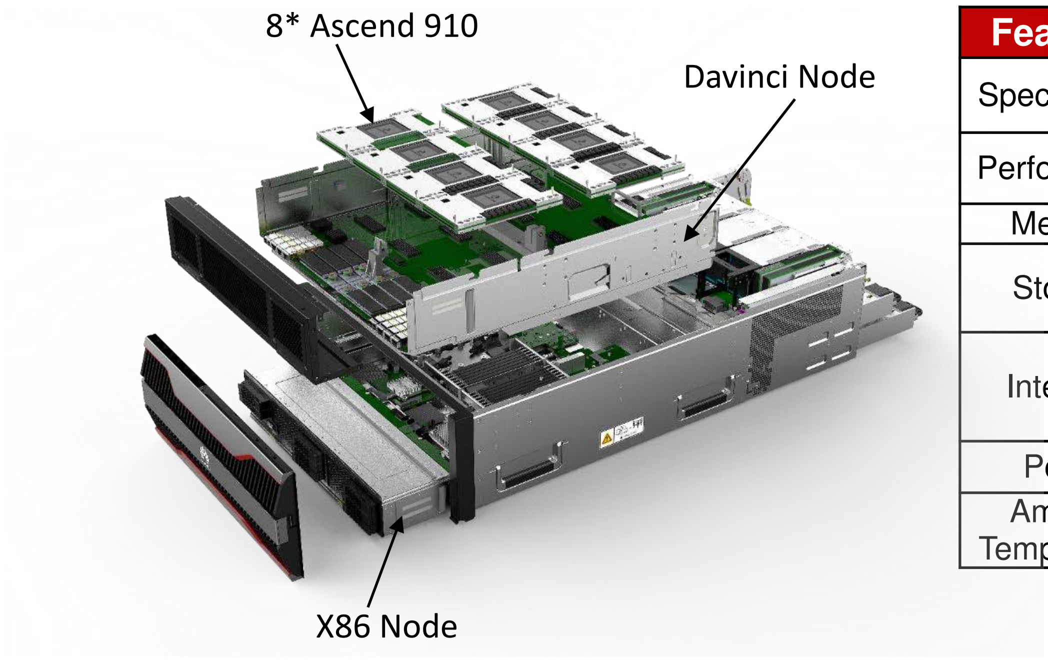
Google TPUv3 (?nm)

- (undisclosed)
- 105Tflops bfloat16

Asend910 (7+ nm)

- 182.4 mm²
- 256TFlops fp16

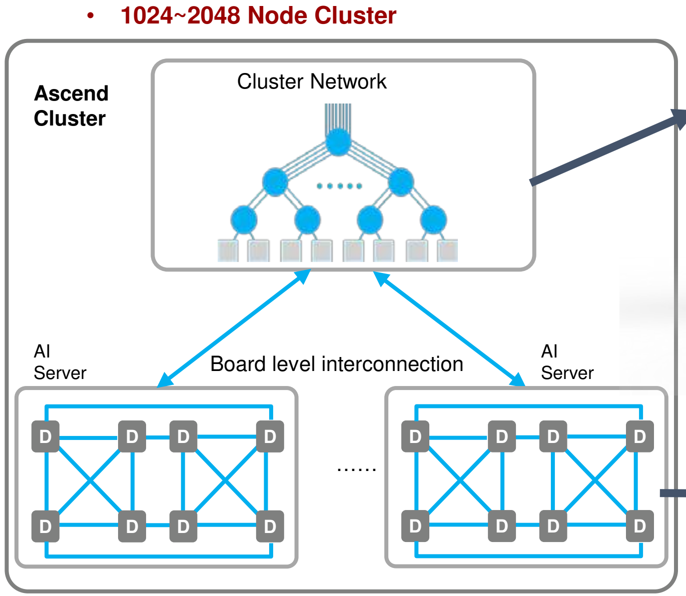
Ascend 910 AI Server



Features	AI Server SPEC.
Specification	8 * Davinci 2 * Xeon CPU + 24 DIMM
Performance	2PFops/Chassis ,256T/AI Module
Memory	24DIMM， Up to 1.5TB
Storage	6 * 2.5inch， NVME； 24TB 2 * 2.5inch， SAS/SATA， Raid1
Interface	8*100G Fiber 4 * PCIe IO
Power	6000W
Ambient Temperature	5~35℃

Ascend 910 Cluster

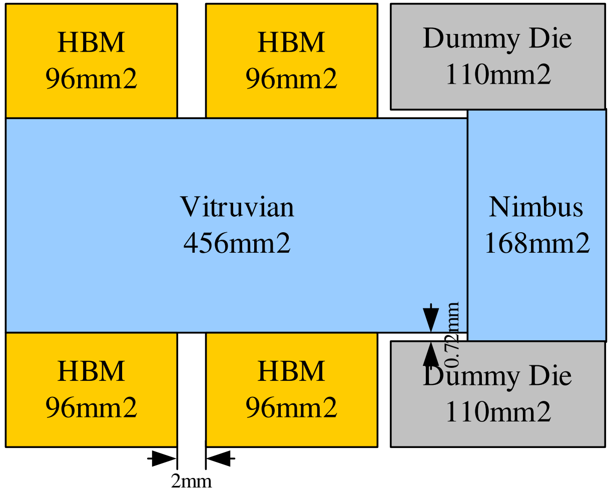
- 2048 Node x 256TFlops = 512 Peta Flops

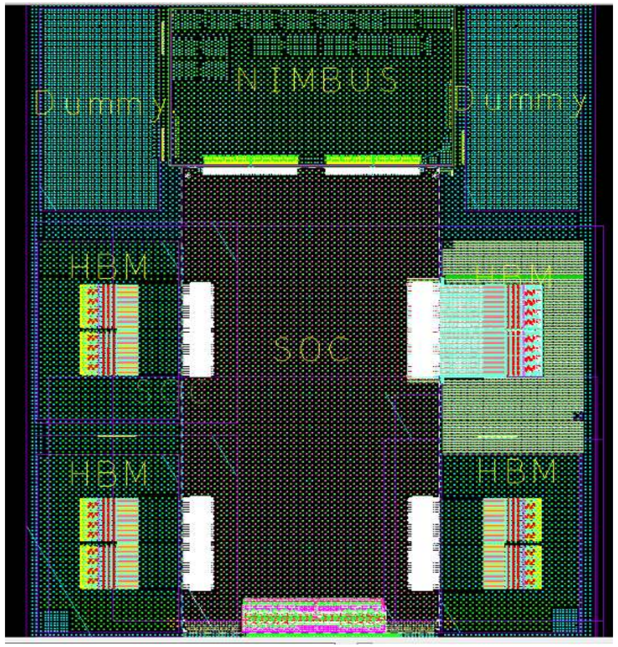


Ascend910 Board

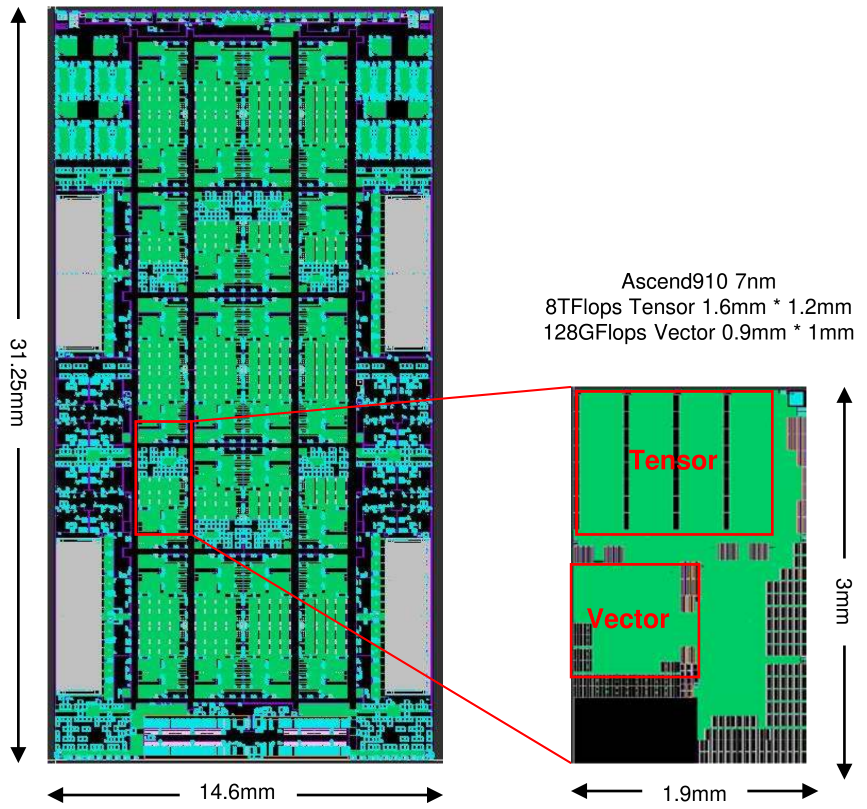
Ascend910 Die Shot

- Total 8 Dies integrated
  - Two dummy dies are added to ensure mechanical uniformity
- Total size：
$456+168+96x4+110x2=1228mm^2$



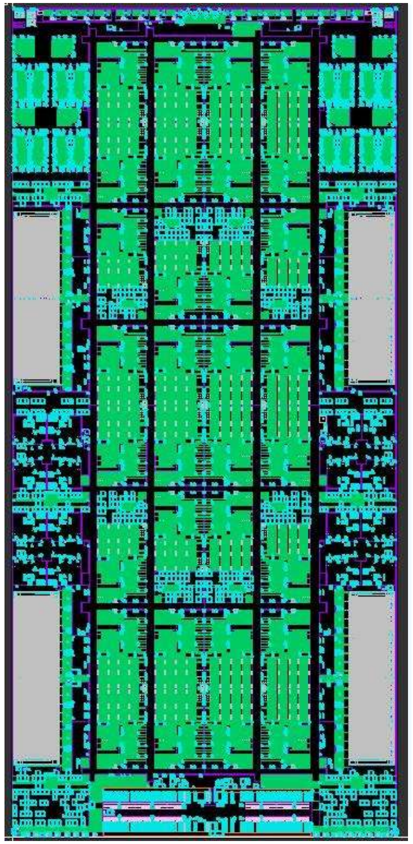


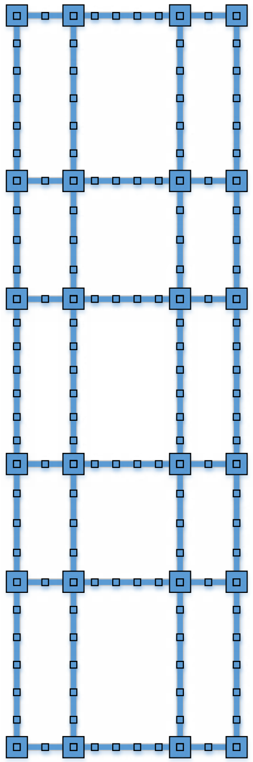
Ascend910 Floorplan



- Mesh NoC connects 32 Davinci Cores in the Ascend 910 Compute Die
- NoC provides Read Bandwidth of 128GBps + Write Bandwidth of 128GBps per core
- Inter-chip connections
  - 3x 240Gbps HCCS ports – for NUMA connections
  - 2x100Gbps RoCE interfaces for networking

Ascend910 NoC

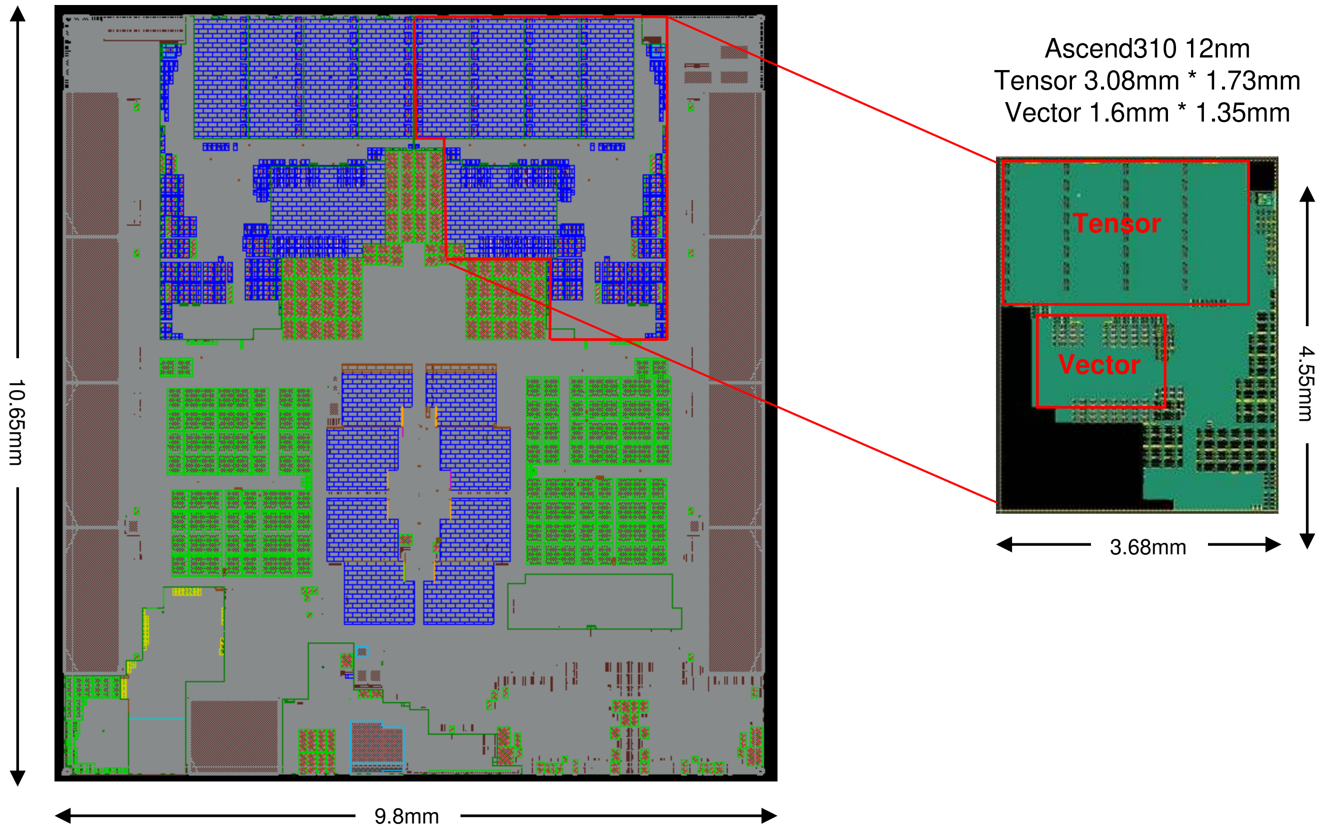




1024bits 2GHz NoC Mesh

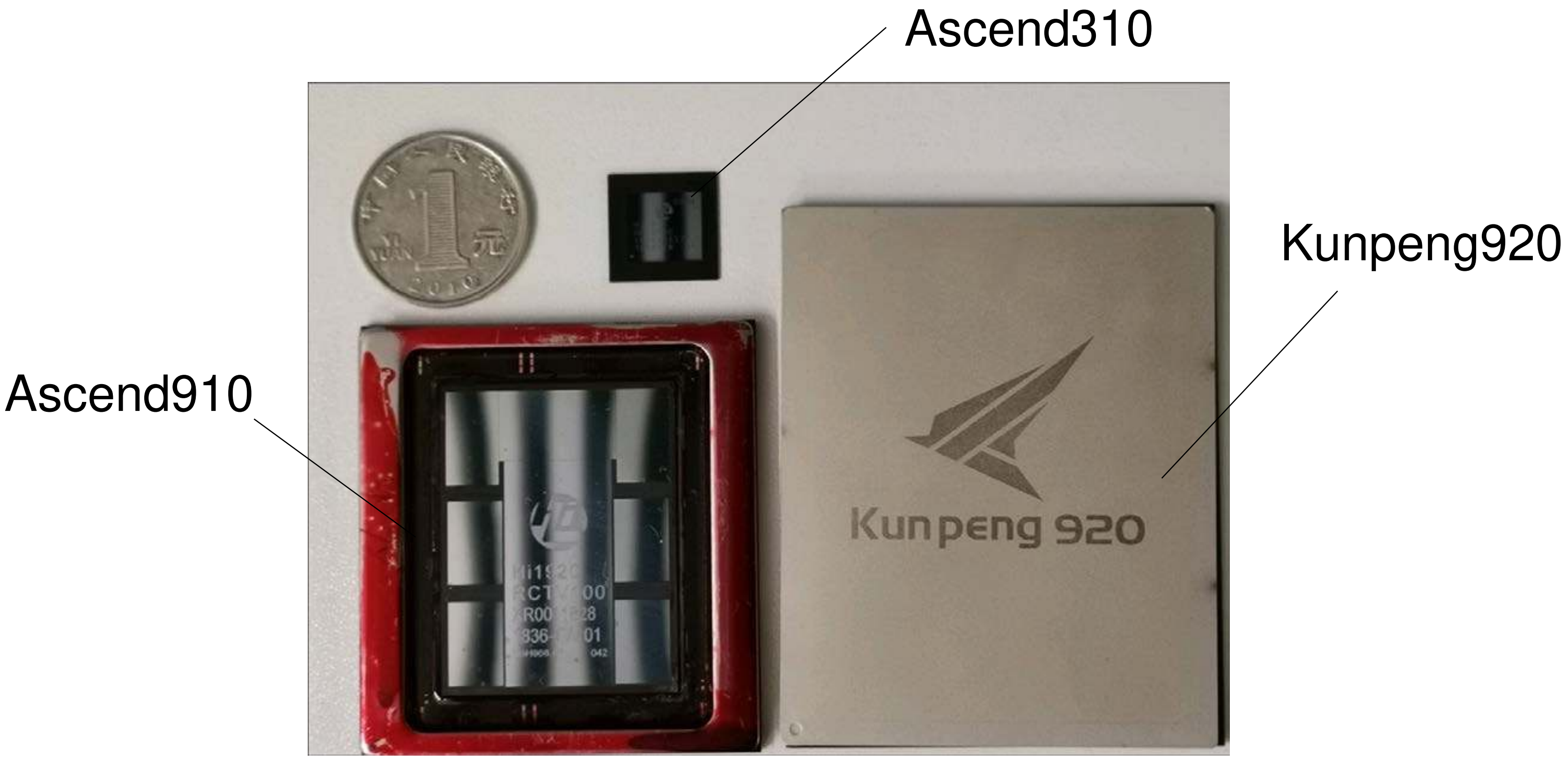
- Topology : 6 Rows x 4 Columns
- Access Bandwidth to on-chip L2 Cache: 4 TByte/s
- Access Bandwidth to offchip HBM: 1.2 TByte/s
- NoC bandwidth fairly shared among the Davinci Cores

Ascend310 Die Shot



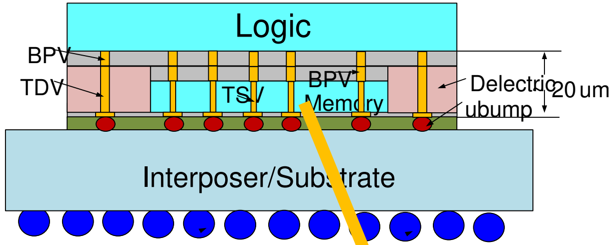
Ascend310 12nm
Tensor 3.08mm * 1.73mm
Vector 1.6mm * 1.35mm

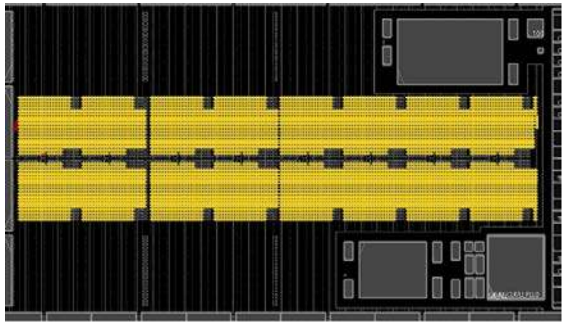
Kunpeng Vs Ascend

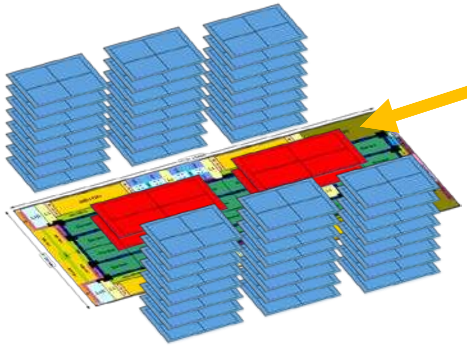


Future: Davinci 3DSRAM Floorplan

3D SRAM Architecture Diagram







More Challenges…

- Generalized Auto ML
- Efficiency for Re-enforcement Learning, GNN?
- Generalized method for Data/Model/Pipeline parallelism
- How to unify data precision?
- Finding the sweet spot architecture
  - ✓ Big chip vs Small chip
  - ✓ Dense vs Sparse
  - ✓ Out of memory, near memory, in memory

Thank you.

把数字世界带入每个人、每个家庭、
每个组织，构建万物互联的智能世界。

Bring digital to every person, home and organization for a fully connected, intelligent world.

HUAWEI