Master Informatics Eng.

2020/21

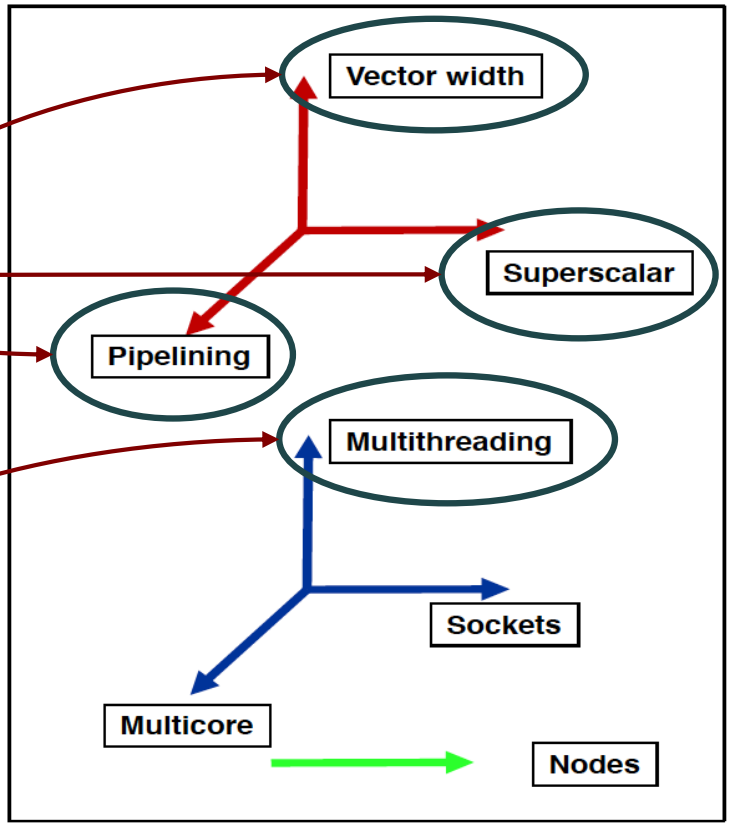
A.J.Proença

From ILP to Multithreading & Data Parallelism *(online)*

(most slides are borrowed)

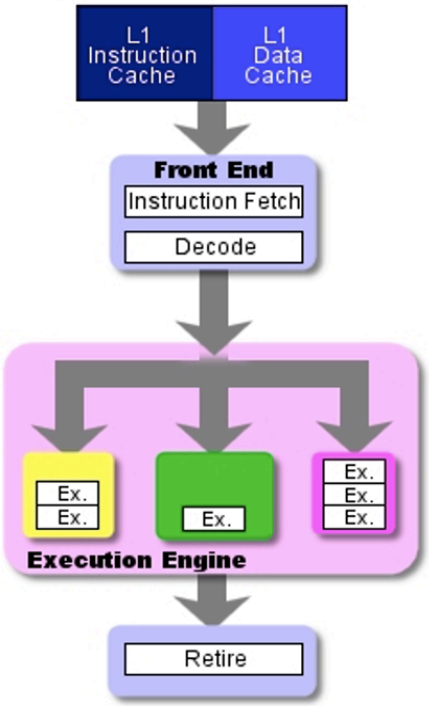
Key issues for parallelism in a single-core

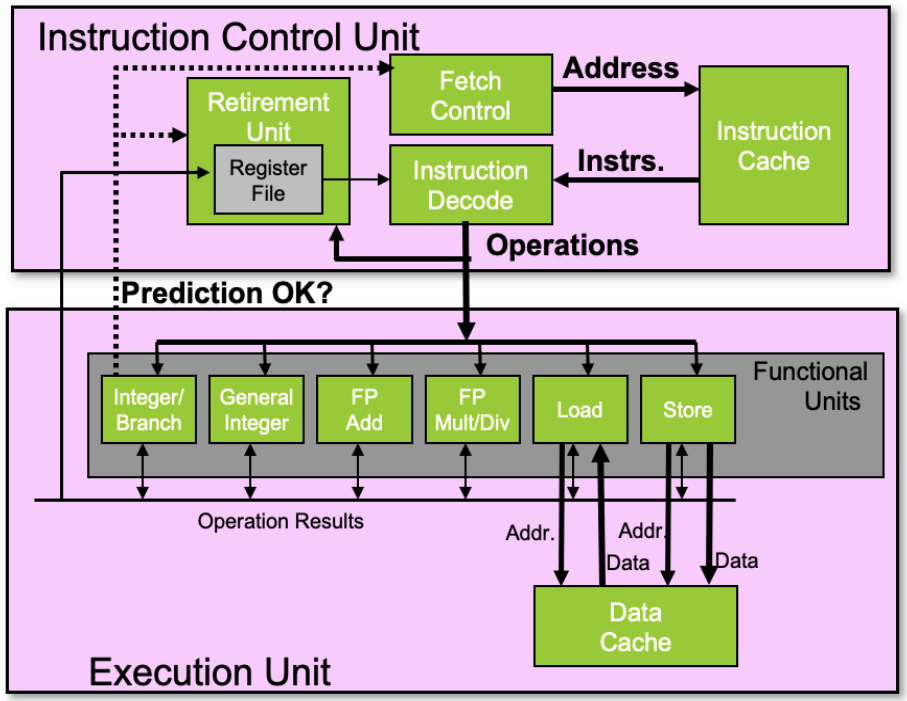
- **Currently under discussion:**
  - pipelining:
    reviewed in the `combine` example
  - superscalar:
    idem, but some more now
  - data parallelism:
    vector computers &
    vector extensions to scalar processors
  - multithreading:
    alternative approaches



Pipelining & superscalarity: a review

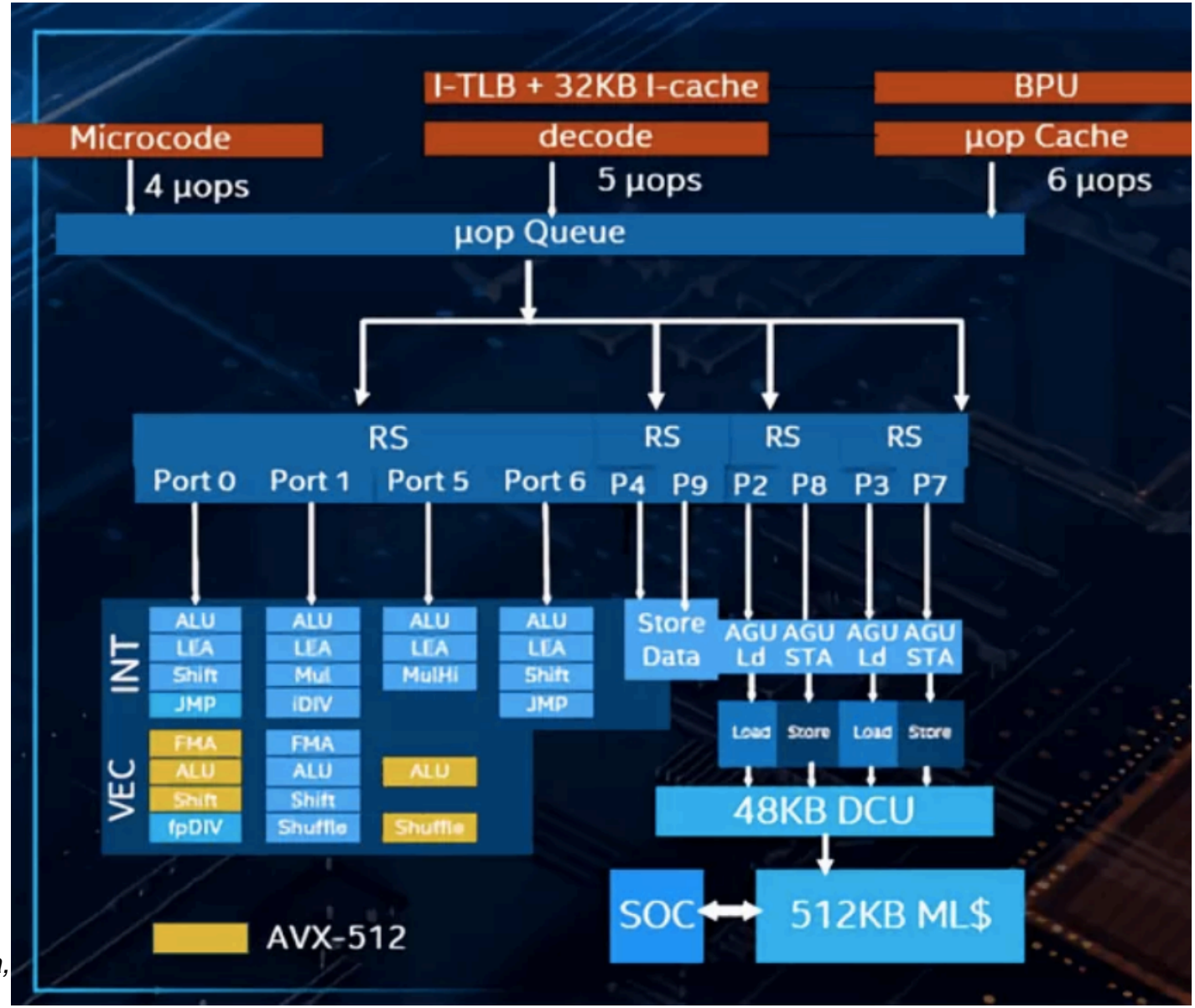
Topic addressed in the undergrad course through the **`combine`** example





- The analysed pipelines were only in the P6 **Execution Unit**, assuming that the **Instruction Control Unit** issues at each clock cycle all the required instructions for parallel execution
- The image suggests **(i)** a 3-way superscalar engine and **(ii)** an execution engine with 6 functional units

Intel Sunny Cove microarchitecture: 30 functional units



Comments to the slides on performance evaluation (1)

- **Assembly version for `combine4`**
  - data type: *integer* ; operation: *multiplication*

```
.L24:                                  # Loop:
  imull (%eax,%edx,4),%ecx             # t *= data[i]
  incl  %edx                           # i++
  cmpl  %esi,%edx                      # i:length
  jl    .L24                           # if < goto Loop
```

- **Translating 1st iteration into RISC-like instructions**

```
load  (%eax,%edx.0,4) ➔ t.1
imull t.1, %ecx.0     ➔ %ecx.1
incl  %edx.0          ➔ %edx.1
cmpl  %esi, %edx.1    ➔ cc.1
jl    -taken cc.1
```

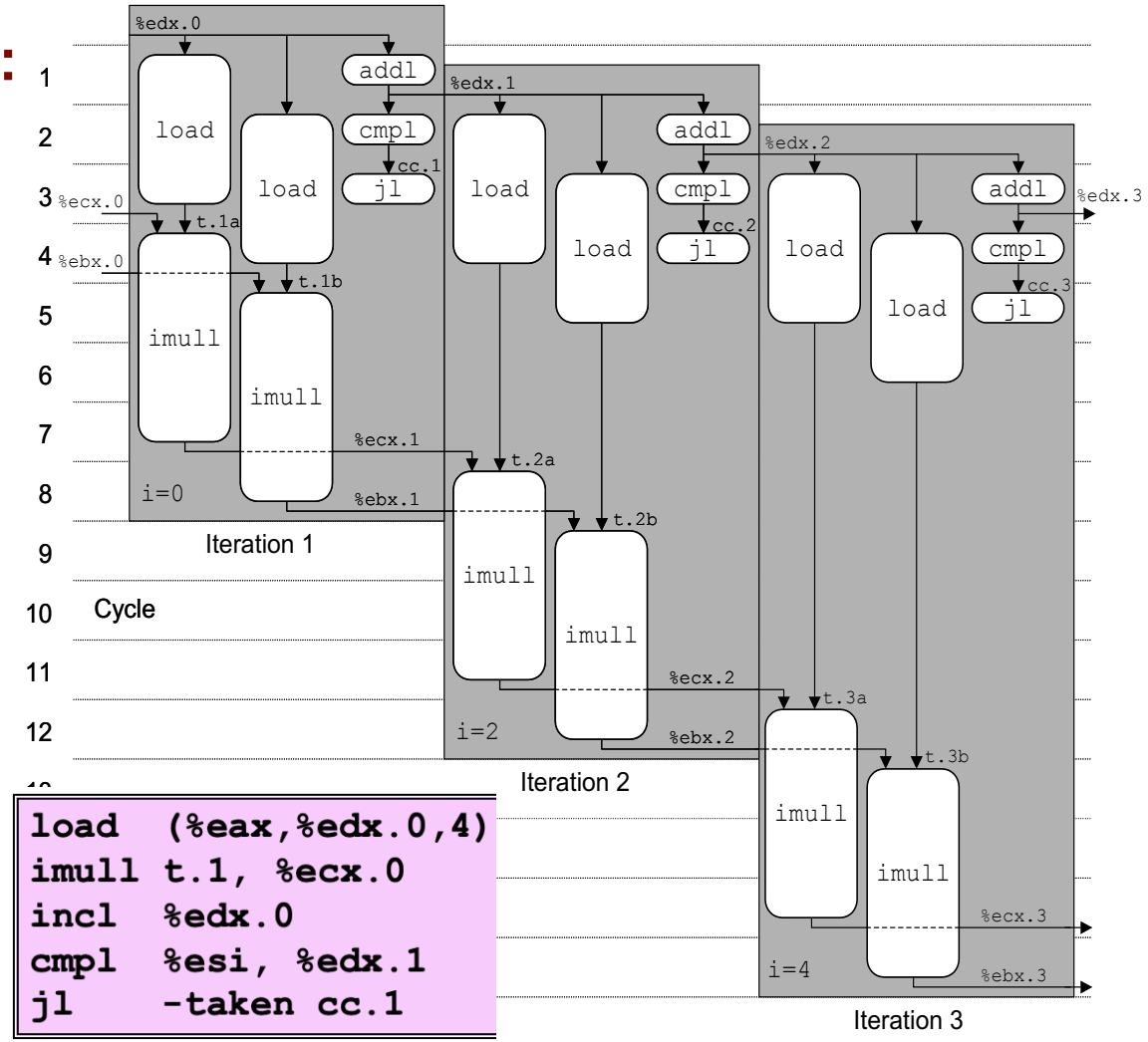
3+miss penalty?
+4
+1 Expected duration:
+1 10+ clock cycles
+1 per vector element

Timings in clock cycles

Comments to the slides on performance evaluation (2)

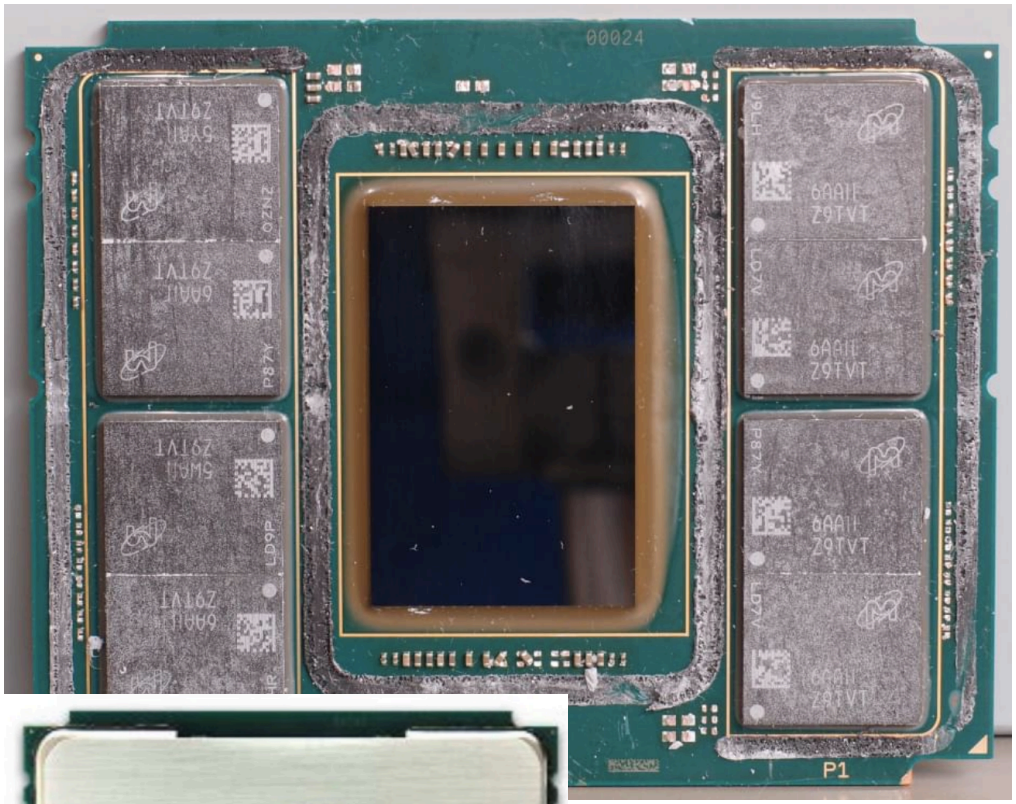
Features that lead to CPE=2:

- **in the hardware**
  - pipelined execution units with 1 clock-cycle/issue
  - more temporary registers
  - mem hierarchy with cache
  - out-of-order execution
  - at least 5-way superscalar
  - more 1 arithm & 1 load units
  - speculative jump
- **at the code level**
  - loop unroll 2x
  - 2-way parallelism



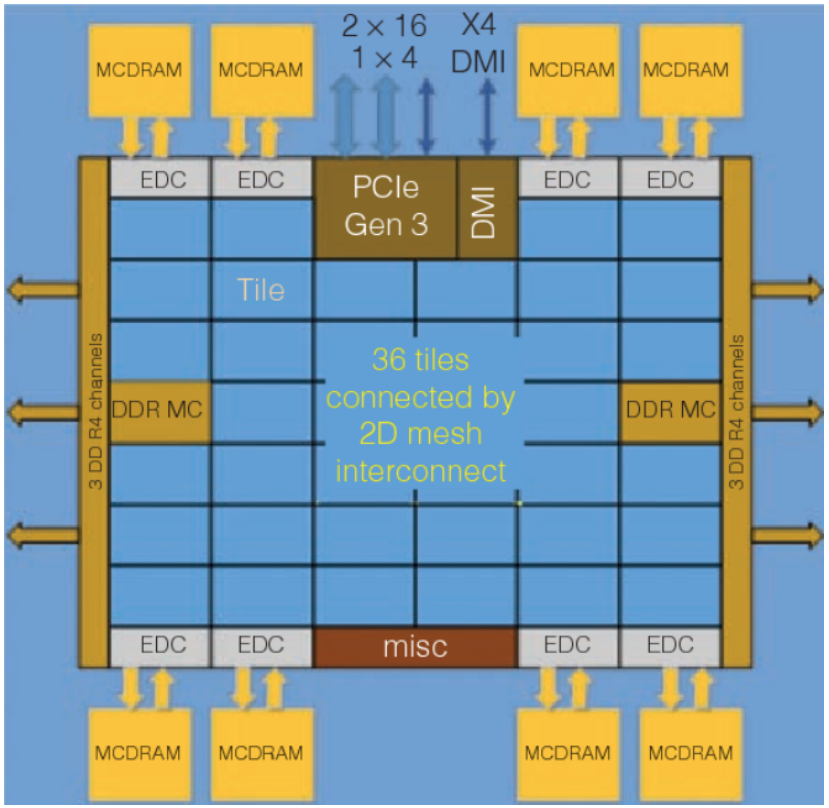

```
load   (%eax,%edx.0,4)
imull  t.1, %ecx.0
incl   %edx.0
cmpl   %esi, %edx.1
jl     -taken cc.1
```

Previous questions: max number of physical cores?

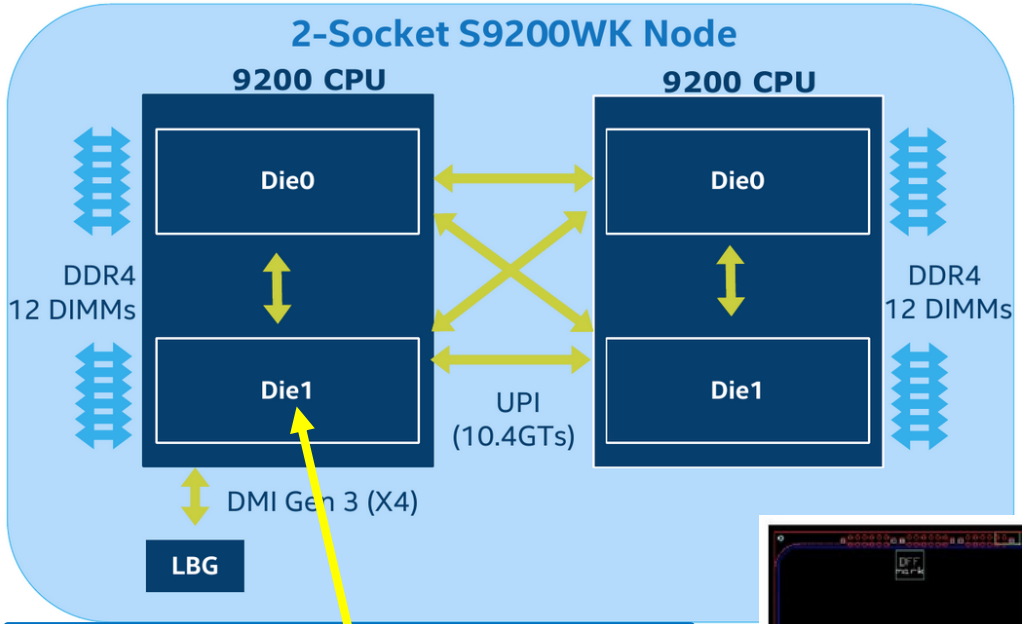


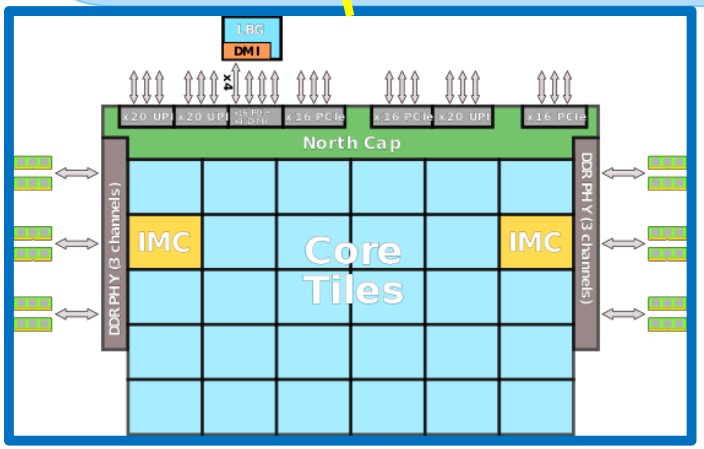
Intel

Xeon Phi package:
up to 72 cores
(discontinued in 2018)



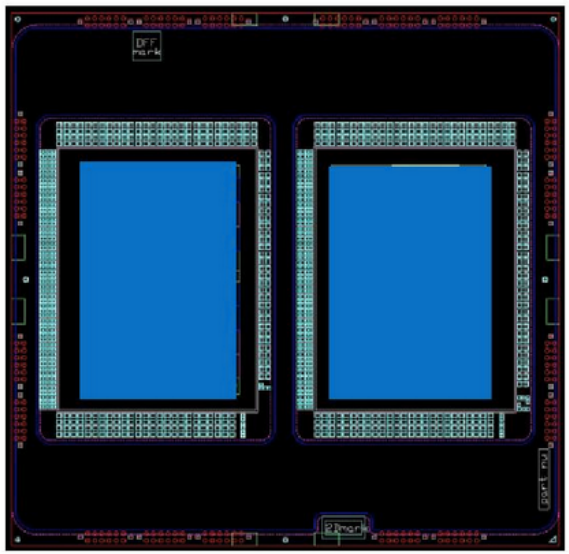
Previous questions: max number of physical cores?



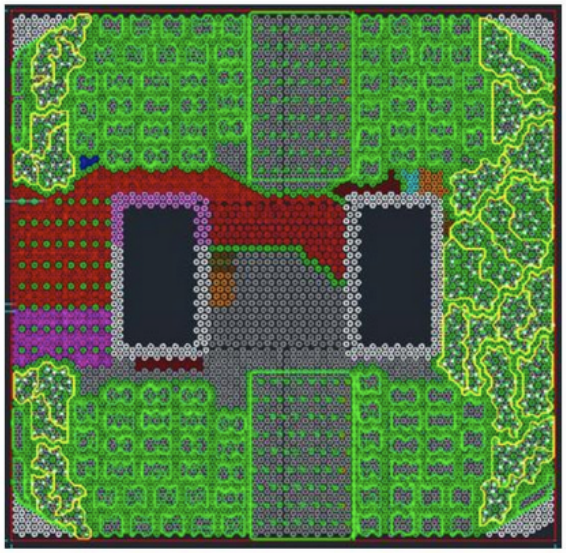


Intel

Xeon Platinum 9282 package:
56 cores
2-socket node: 112 cores

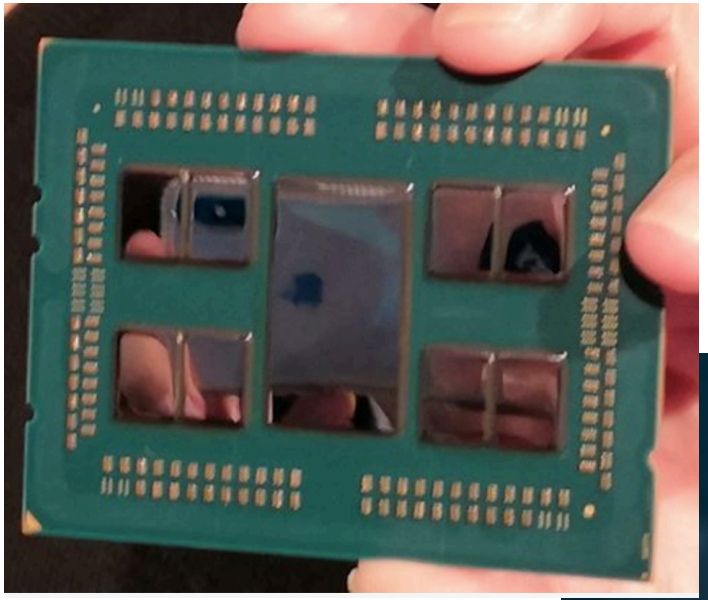


Package Top side view



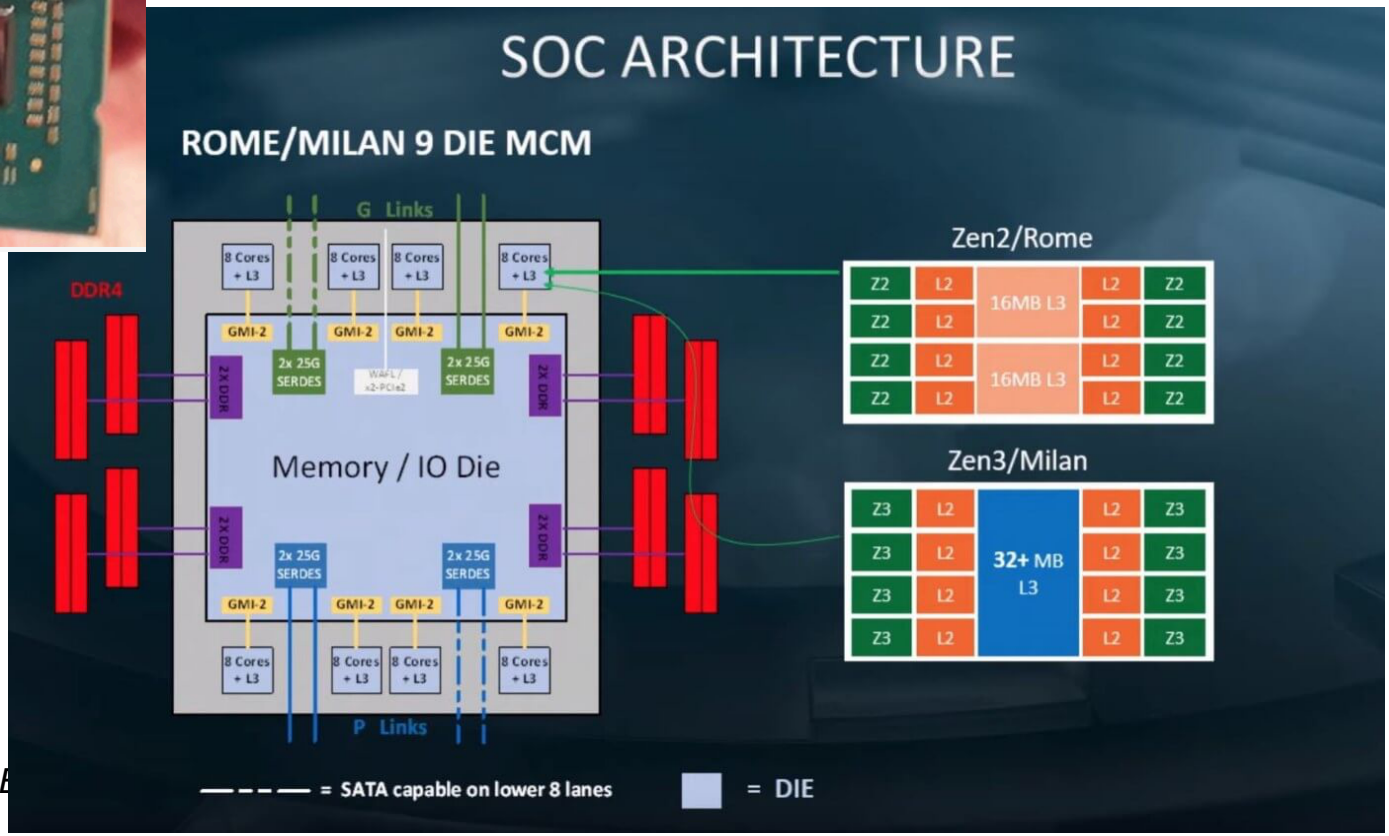
Package bottom side view

Previous questions: max number of physical cores?



AMD

Epyc Rome & Milan: 64 cores



Previous questions: max number of physical cores?

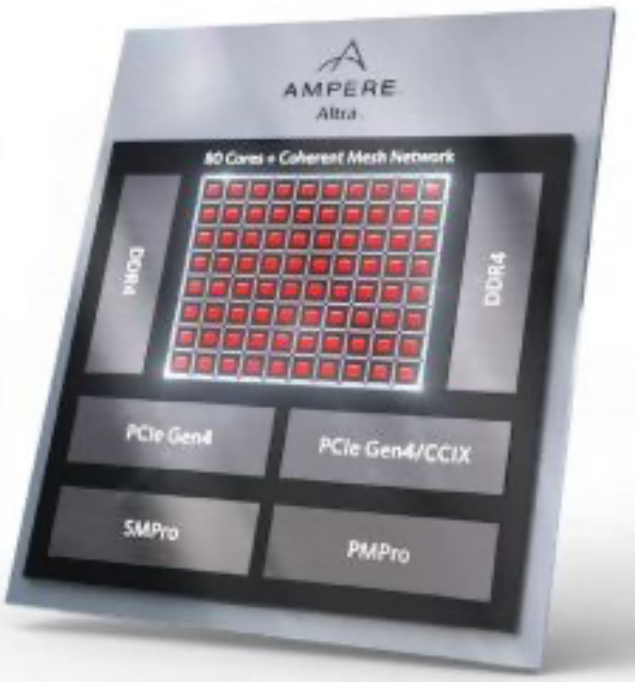
Ampere™ Altra™ processor complex

ARM

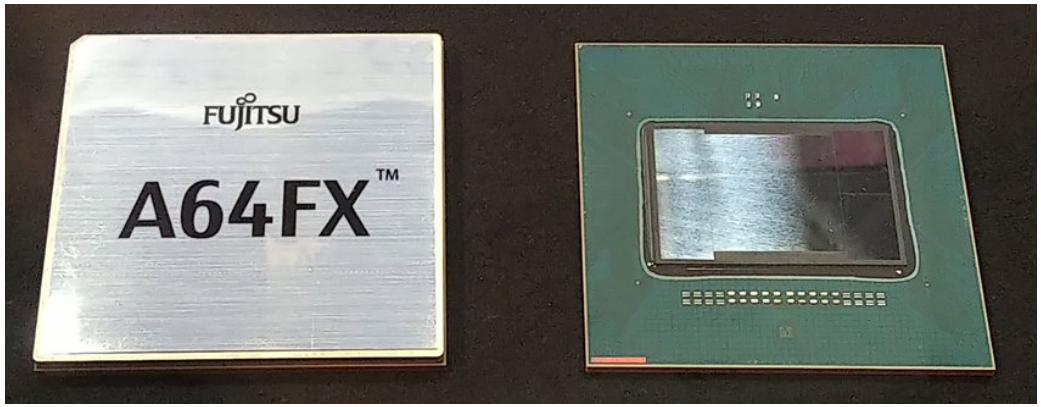
Ampere Altra: 80 cores

80 64-bit Arm CPU cores @ 3.0 GHz Turbo

- 4-Wide superscalar aggressive out-of-order execution
- Single threaded cores for performance and security isolation



Previous questions: max number of physical cores?

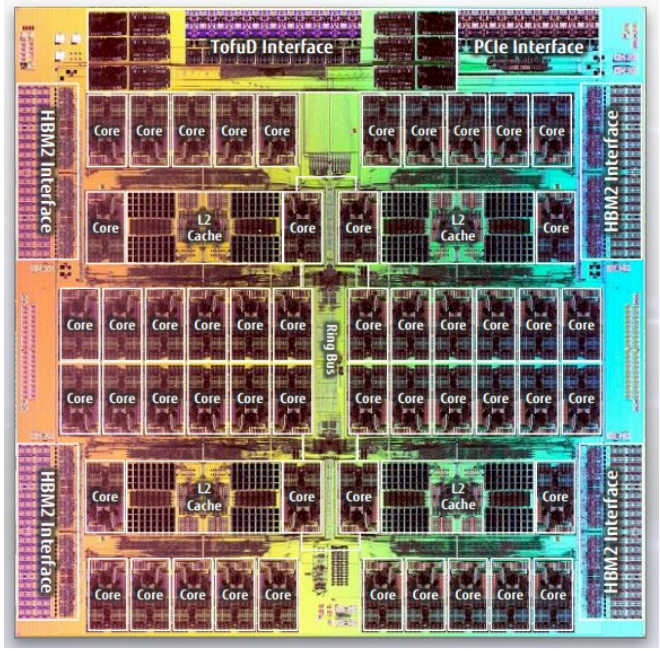


ARM

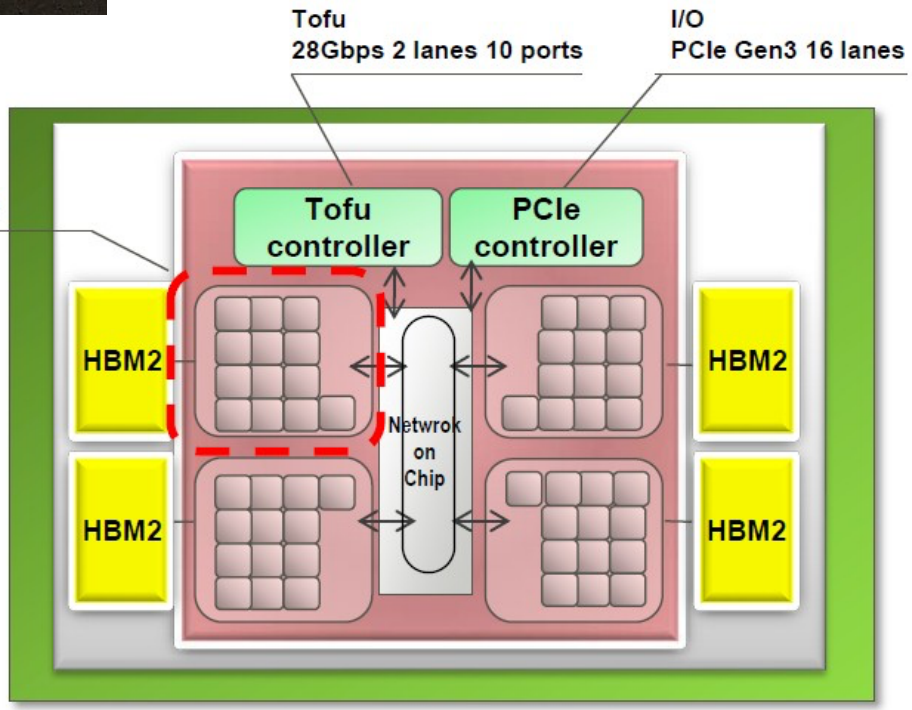
Fujitsu A64FX Arm Chip:

48+4 cores

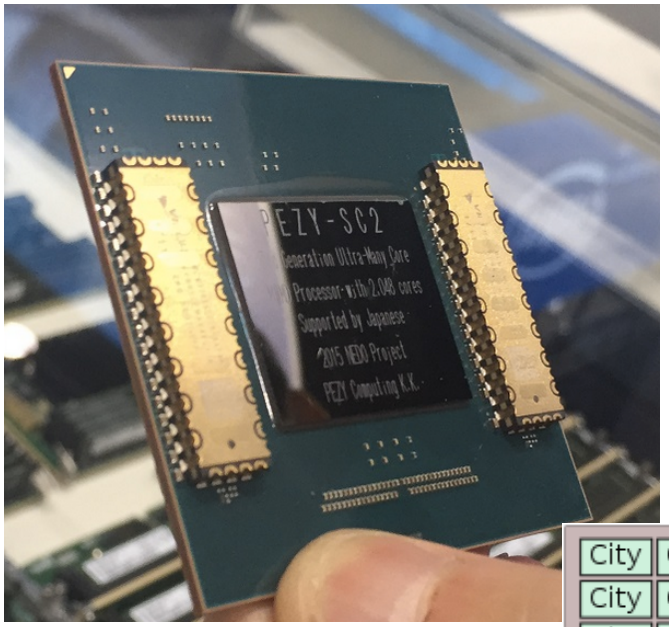
(in #1 TOP500, June 2020)



CMG specification
13 cores
L2$ 8MiB
Mem 8GiB, 256GB/s



Previous questions: max number of physical cores?

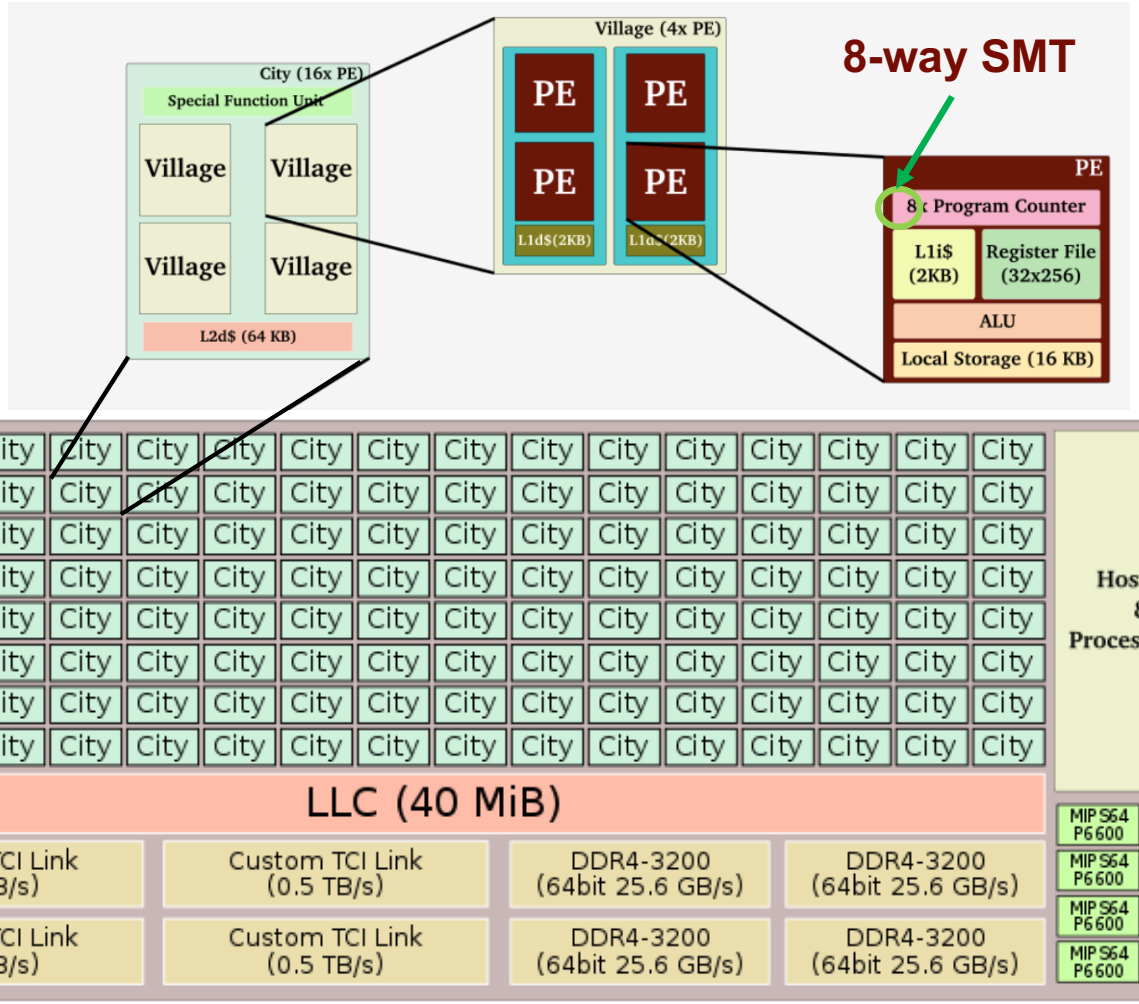


PEZY-SC2: **2048 cores**
+ 8x MIPS cores *(2017)*

PEZY-SC3: 8192 cores
(due in 2019, but…)

PEZY-SC4: 16384 cores
(due in 2020, but…)

Japan



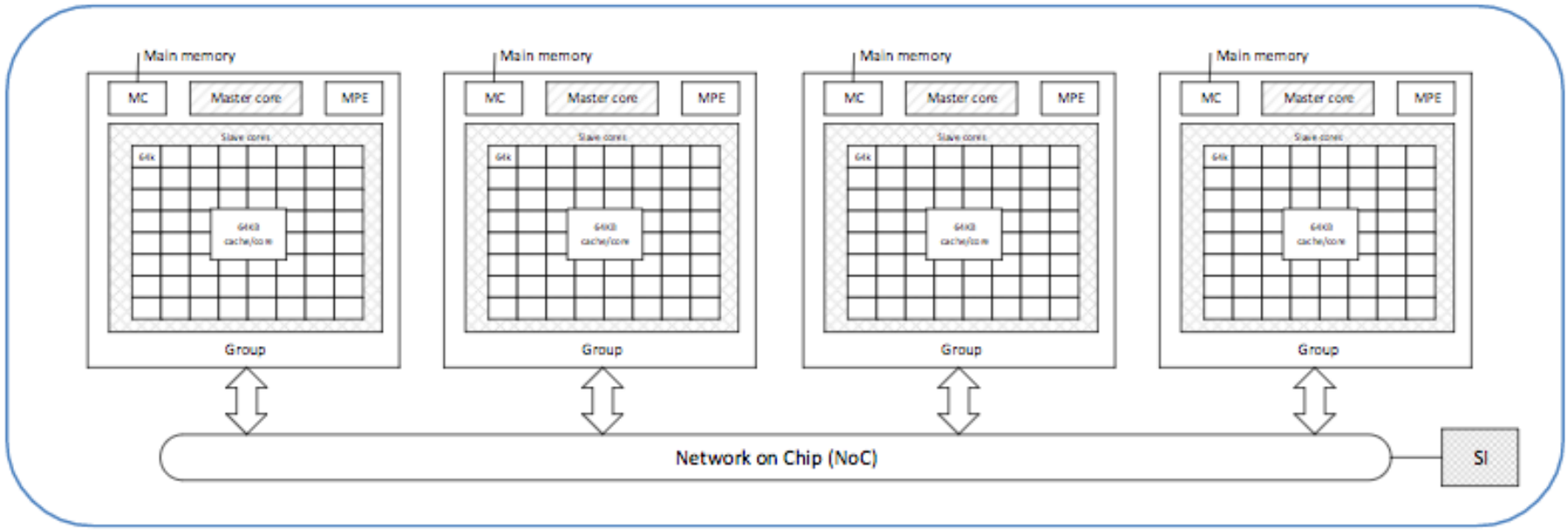
Previous questions: max number of physical cores?

China

Sunway SW 26010:

256+4 cores

(in #1 TOP500, June 2016)



Previous questions: max number of physical cores?

Cerebras Wafer Scale Engine (WSE):
the largest chip ever built)

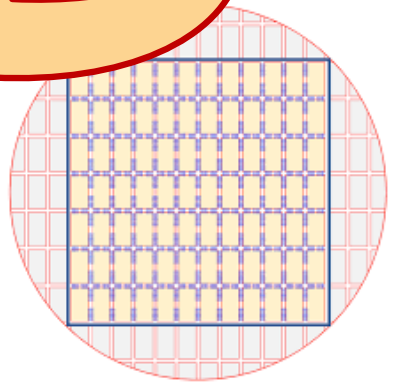
Worldwide

46,225 mm² chip
56x larger than the biggest GPU ever made

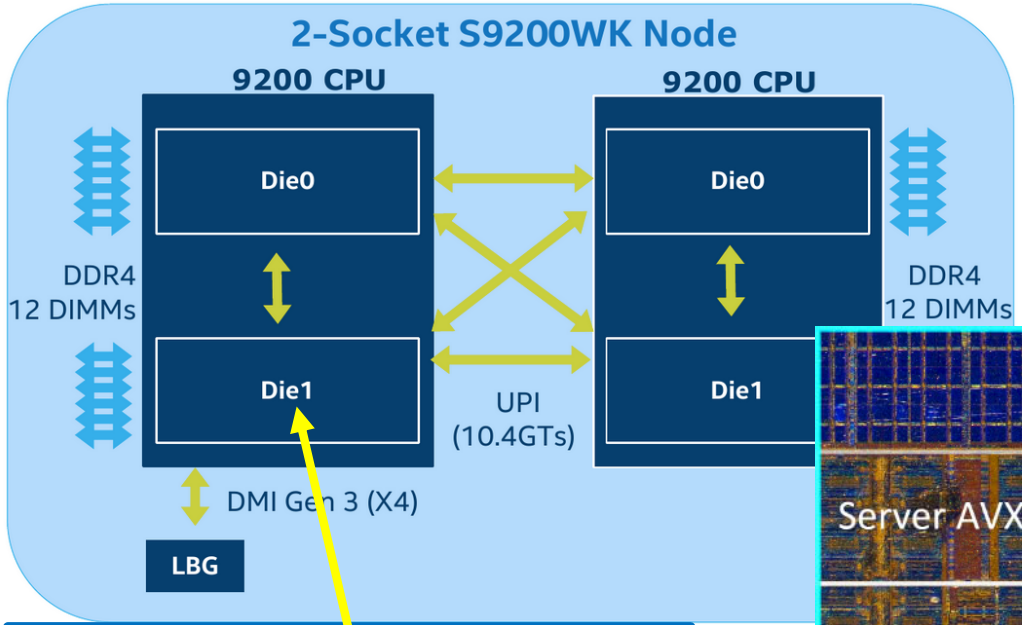
400,000 core
more cores

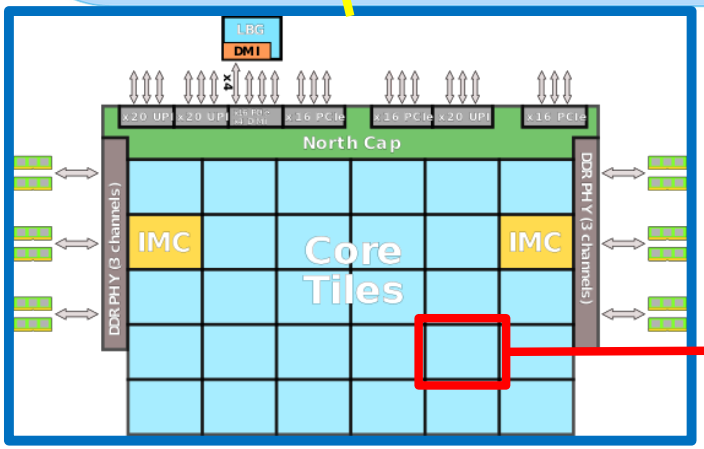
18 GB on-chip SRAM
3000x more on-chip memory

100 Pb/s interconnect
33,000x more bandwidth

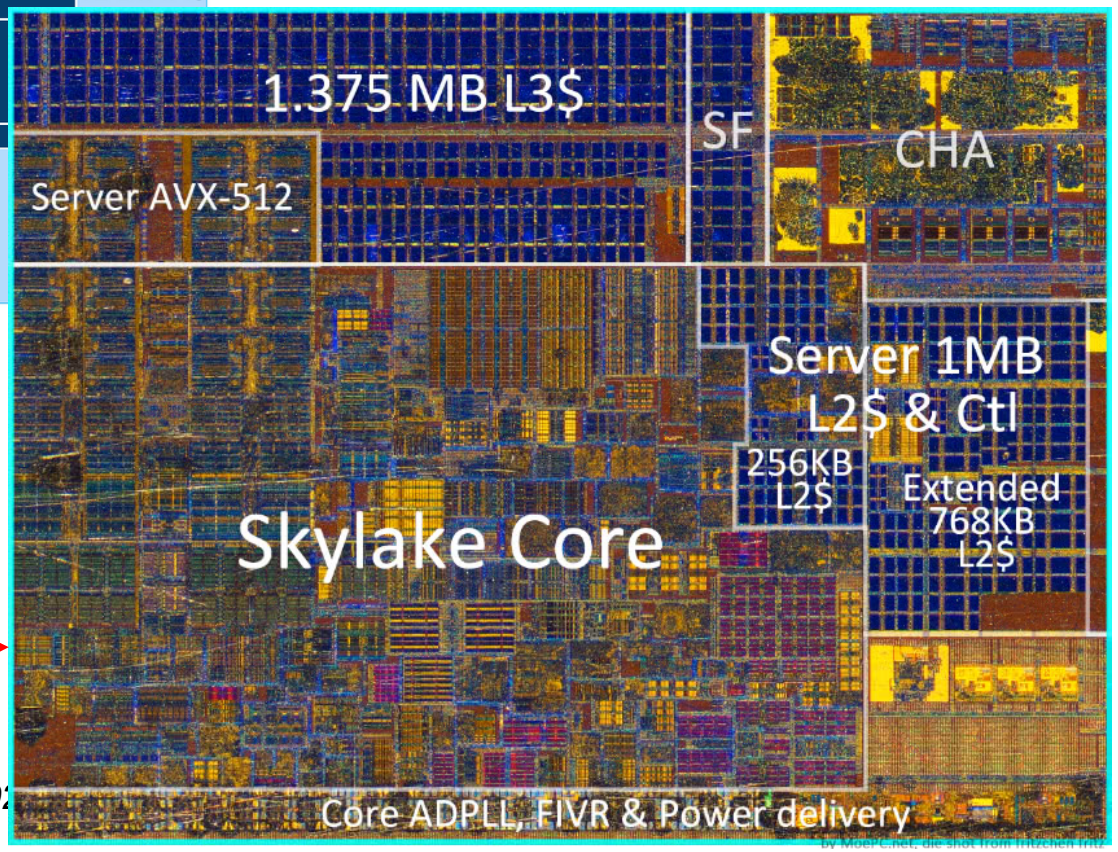


What is needed to increase the #cores in a chip?

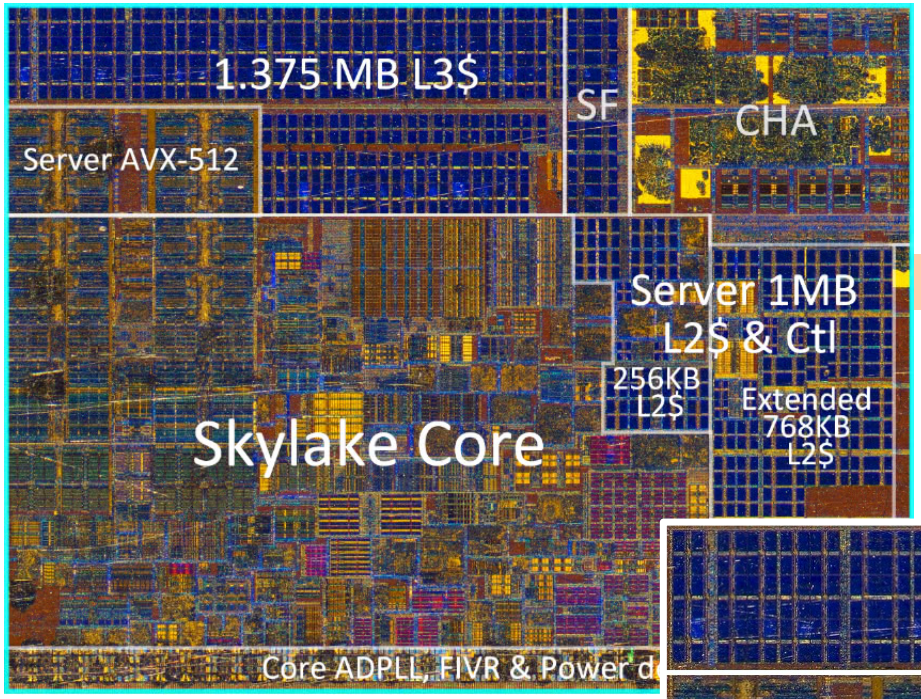




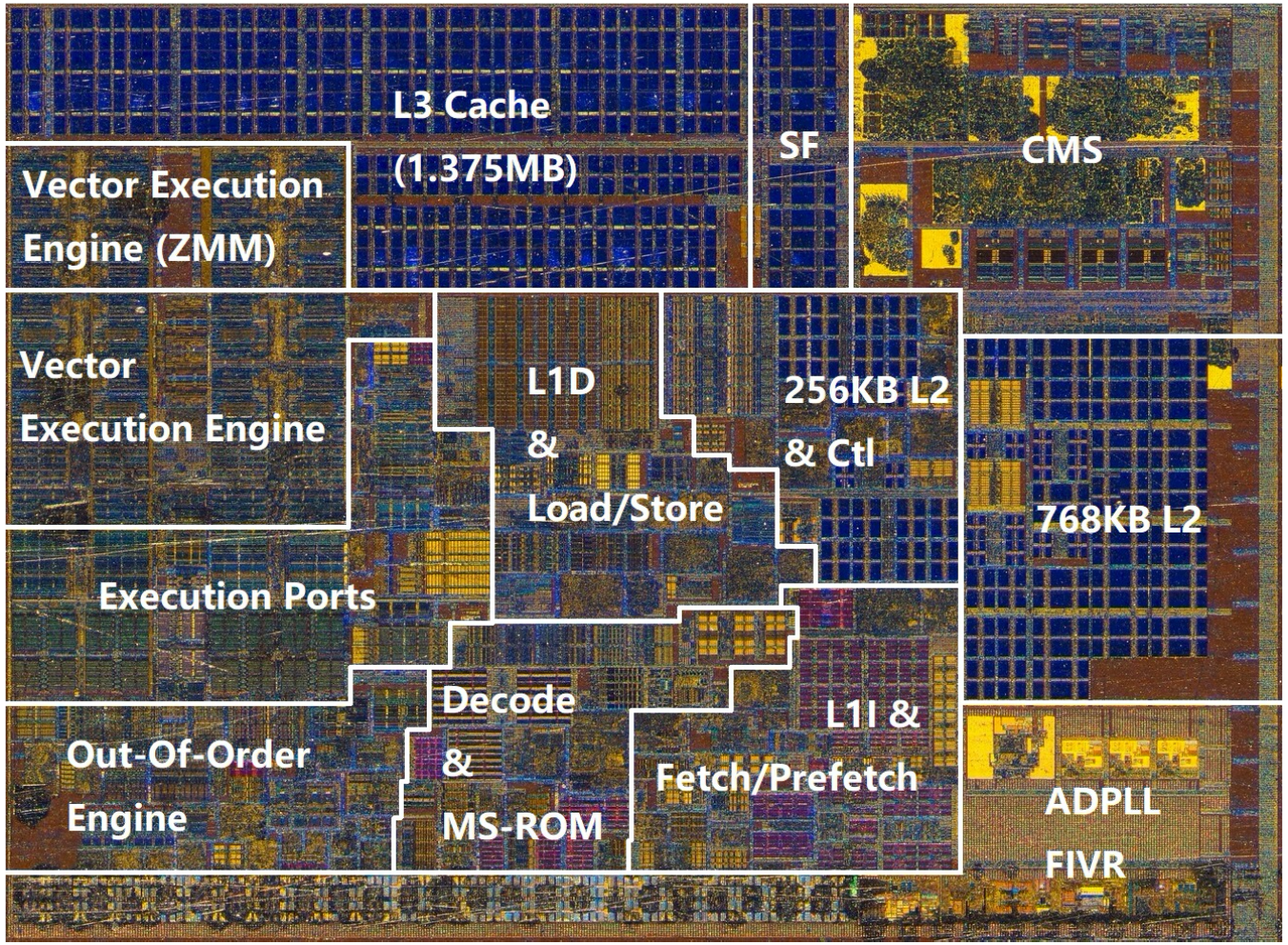
Layout of one core



What is needed to increase the #cores in a chip?



Using the same microelectronics technology, **remove** parts from the core



Which parts?

- L3 cache
- AVX-512
- reduce L2 cache
- in-order exec
- less functional units
- …

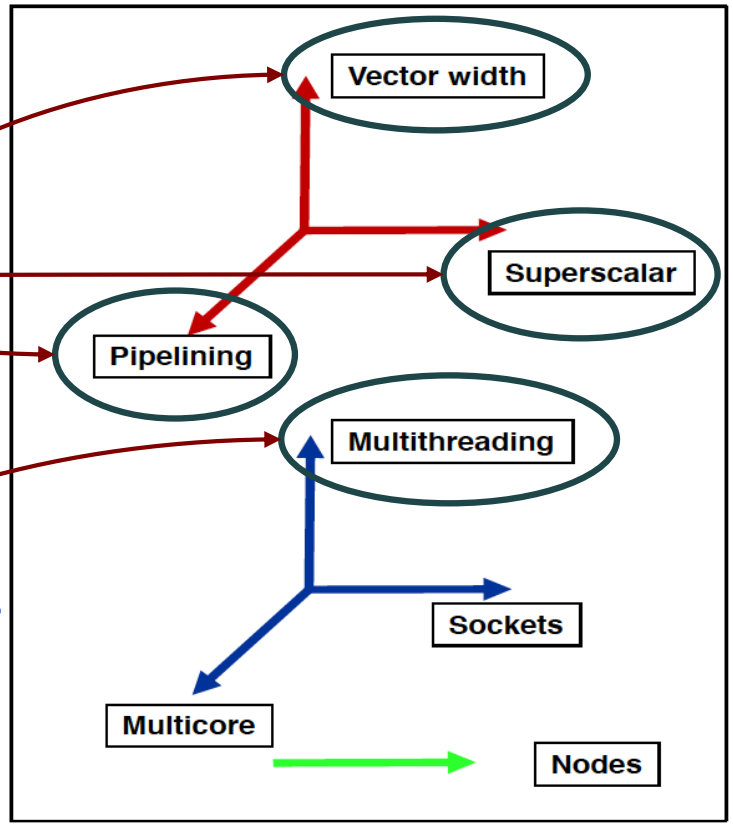
SMT in architectures designed by other companies

For each manufacturer identify the max hw support for SMT at each core (how many ways):

- Intel Xeon
- AMD Epyc
- Fujitsu Arm64FX
- IBM Power 9
- Sunway SW2610
- Apple A14

Key issues for parallelism in a single-core

- **Currently under discussion:**
  - pipelining:
    reviewed in the `combine` example
  - superscalar:
    idem, but some more now
  - data parallelism:
    vector computers &
    vector extensions to scalar processors
  - multithreading:
    alternative approaches



Some questions…

1. Is AVX-512 so much better than AVX2, why AMD does not plan to support AVX-512?
2. Each core in the Xeon Phi KNL (presented in next slides) has two AVX-512 units, but later chips designed by Intel did not follow the same path. Why?
3. Fujitsu used the ARM64 design to build the A64FX chip and took advantage of the SVE approach, which could operate on vectors up to 2048 bits; however, it opted to support only 512-bits long vectors. Why?

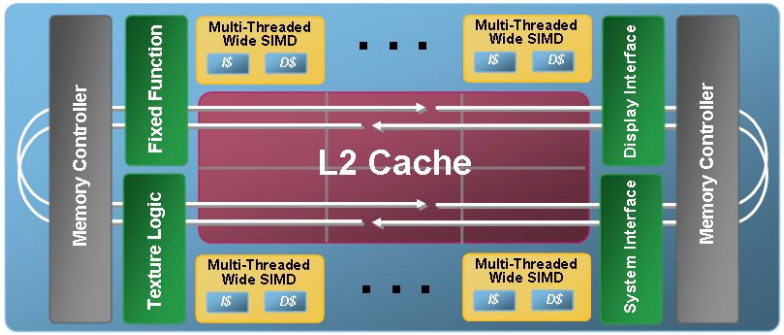
Suggestion for questions 2 & 3:

- compute the required bandwidth to access RAM when all cores perform a vector operation at each clock cycle and
- compare this value with the max bandwith of the embedded RAM.

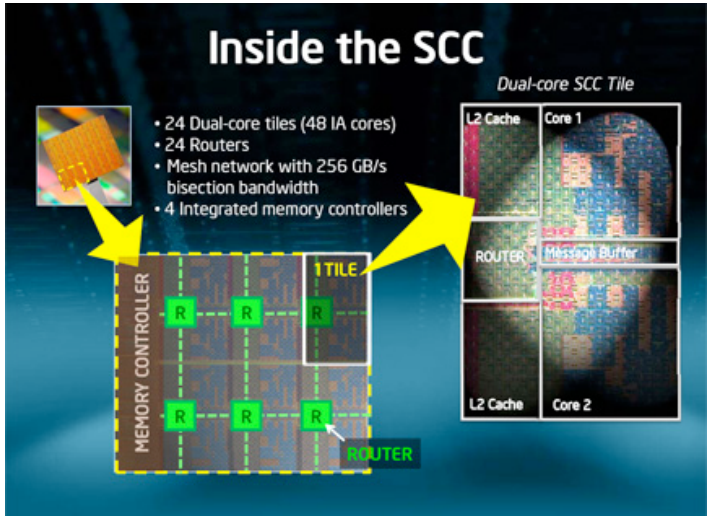
Intel MIC: Many Integrated Core

Intel evolution, from:

- Larrabee (80-core GPU) & SCC

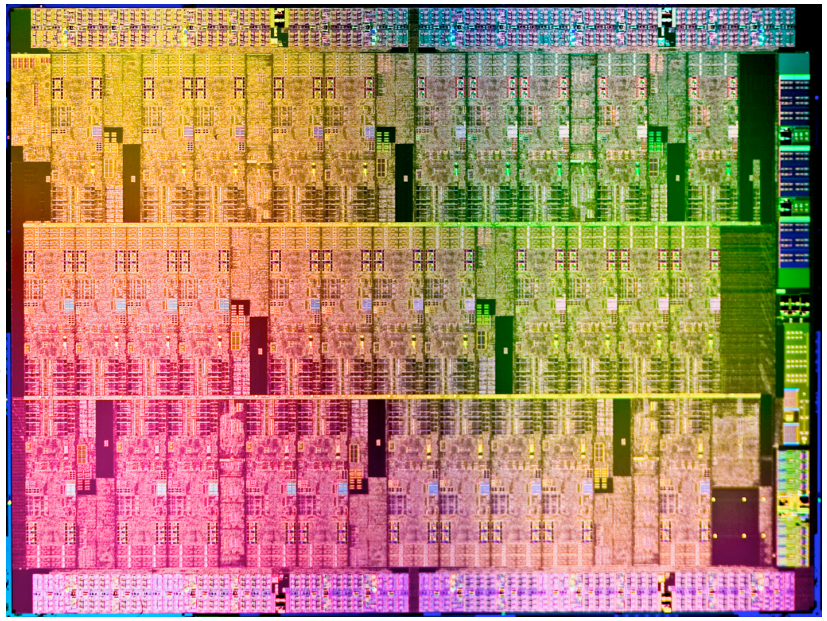


Single-chip
Cloud
Computer,
24x
dual-core tiles



to MIC:

- Knights Ferry (pre-production, Stampede)
- Knights Corner
  Xeon Phi co-processor up to 61 Pentium cores
- Knights Landing (~~& Knights Mill...~~)
  Xeon Phi full processor up to 36x dual-core Atom tiles



Launched in June 2016
Discontinued in July 2018

INTRODUCTION TO THE INTEL® XEON PHI™ PROCESSOR (CODENAME "KNIGHTS LANDING")

Dr. Harald Servat - HPC Software Engineer
Data Center Group – Innovation Performing and Architecture Group

Summer School in Advanced Scientific Computing 2016
~~February~~ 21st, 2016 – Braga, Portugal
June

INTEL® XEON PHI™ PROCESSOR FAMILY ARCHITECTURE OVERVIEW

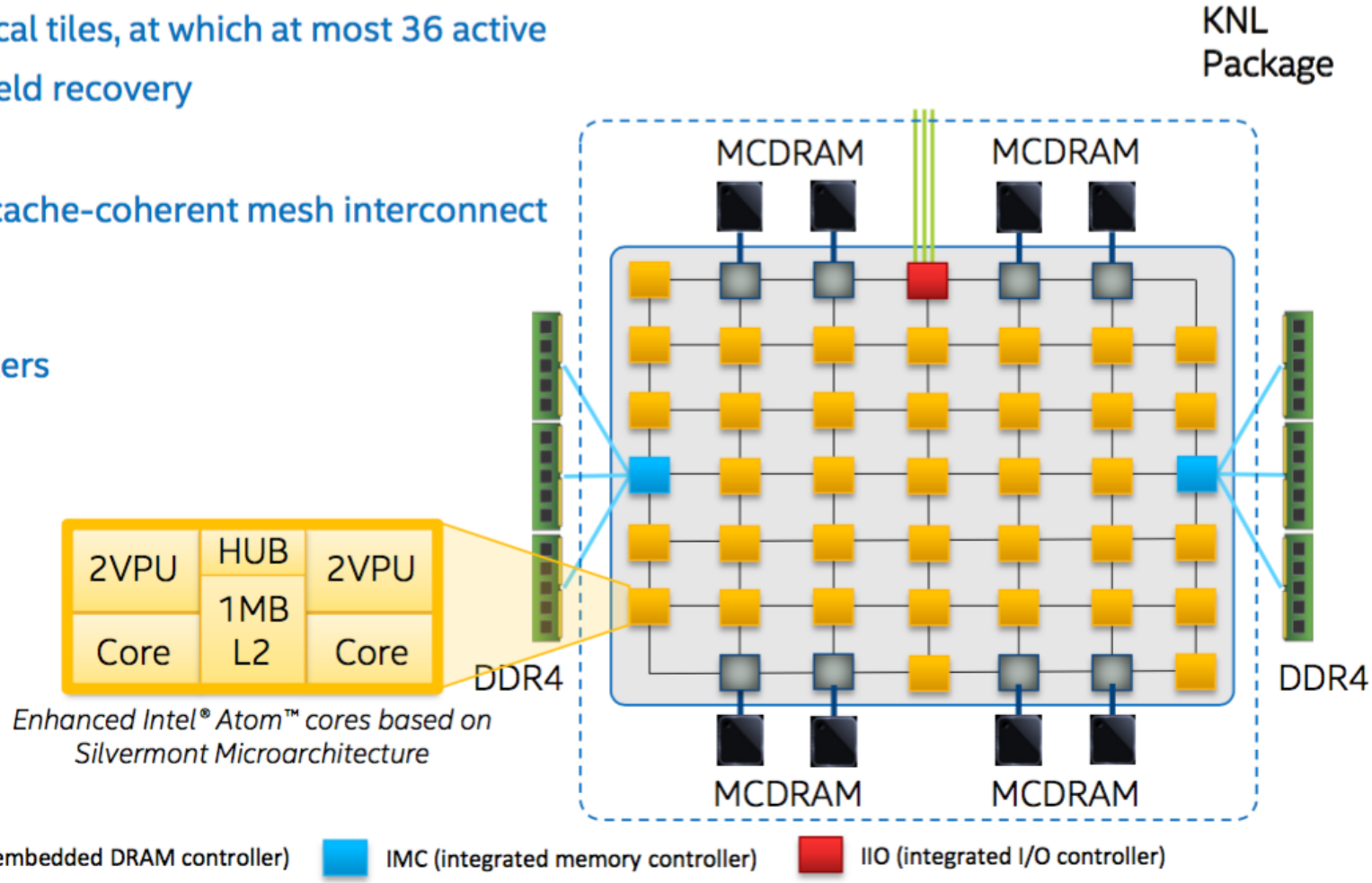
Codenamed "Knights Landing" or KNL

Comprises 38 physical tiles, at which at most 36 active

- Remaining for yield recovery

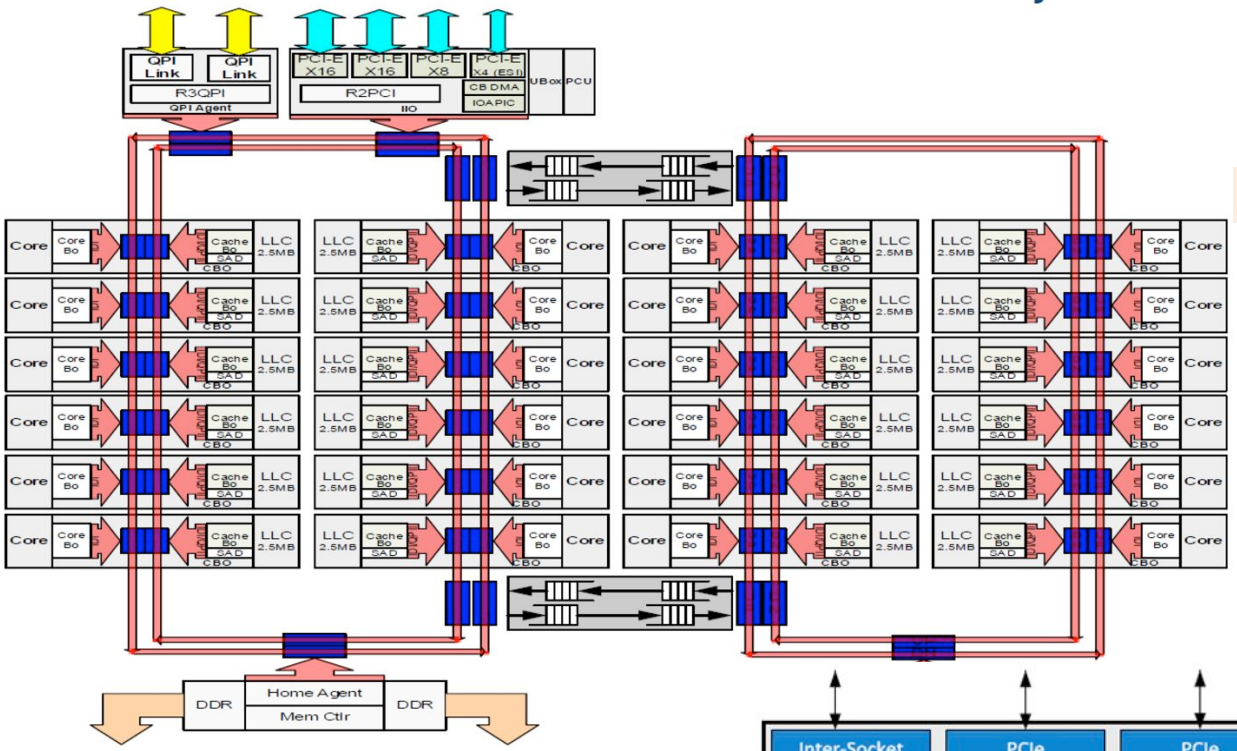
Introduces new 2D cache-coherent mesh interconnect (Untile)

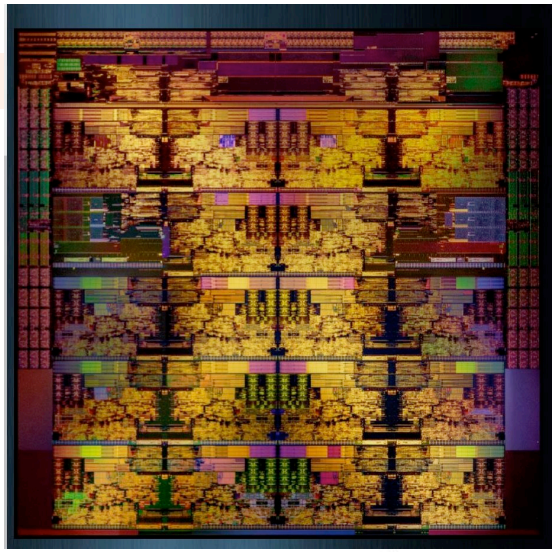
- Tiles
- Memory controllers
- I/O controllers
- Other agents

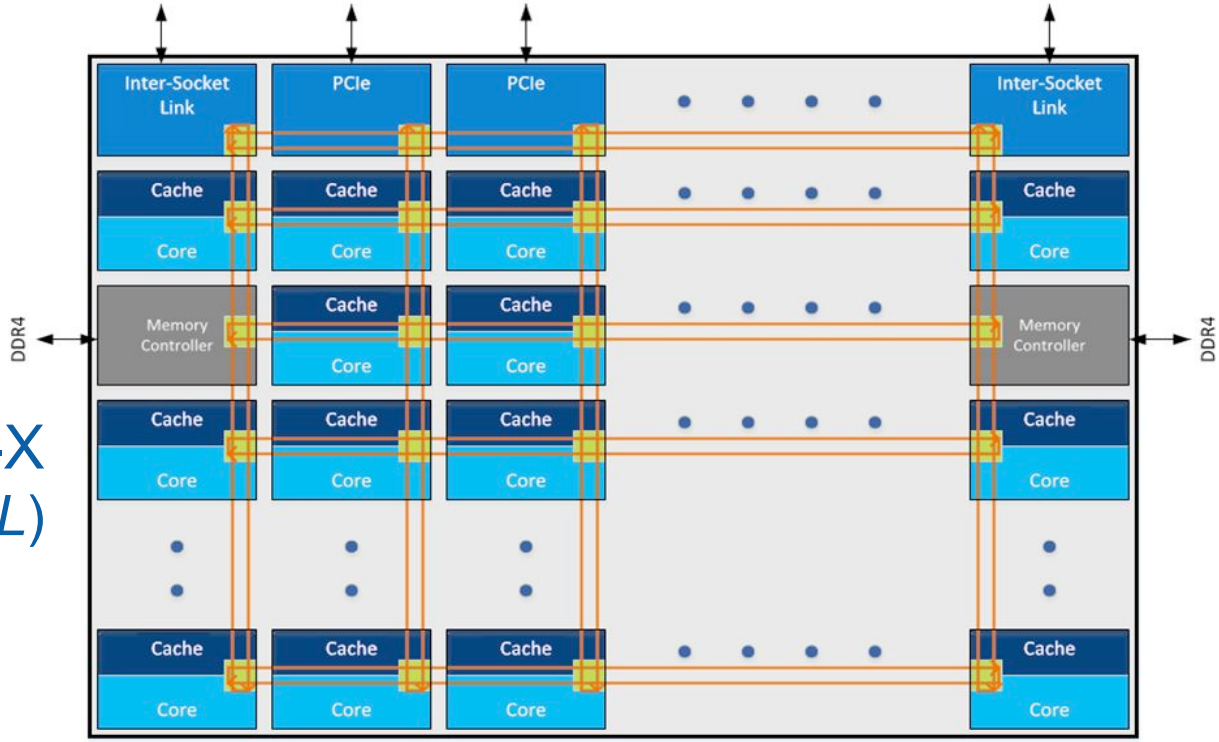


Evolution of on-chip Intel interconnect

Intel® Xeon® Processor E5 v4 Product Family HCC



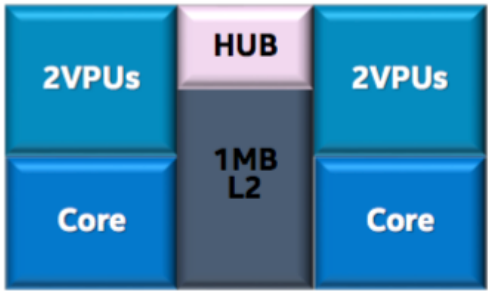




Intel 18-core Skylake-X
(follows KNL)

http://www.tomshardware.com/news/intel-mesh-architecture-skylake-x-hedt,34806.html

KNL PROCESSOR TILE



Tile

- 2 cores, each with 2 vector processing units (VPU)
- 1 MB L2-cache shared between the cores

Core

- Binary compatible with Xeon
- Enhanced Silvermont (Atom)-based for HPC w/ 4 threads
- Out-of-order core
- 2-wide decode, 6-wide execute (2 int, 2 fp, 2 mem), 2-wide retire

2 VPU

- 512-bit SIMD (AVX512) 32SP/16DP per unit
- Legacy X87, SSE, AVX and AVX2 support

TAKING BENEFIT OF THE CORE

OpenMP

From Wikipedia, the free encyclopedia

Thread affinity [edit]

Some vendors recommend setting the processor affinity on OpenMP threads to associate them with particular processor cores.[33][34][35] This minimizes thread migration and context-switching cost among cores. It also improves the data locality and reduces the cache-coherency traffic among the cores (or processors).

Threading

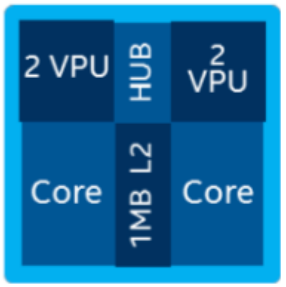
- Ensure that thread affinities are set.
- Understand affinity and how it affects your application (i.e. which threads share data?).
- Understand how threads share core resources.
  - An individual thread has the highest performance when running alone in a core.
  - Running 2 or 4 threads in a core may result in higher per core performance but lower per thread performance.
  - Due to resource partitioning, 3 thread configuration will have fewer aggregative resources than 1, 2 or 4 threads per core. 3 threads in a core is unlikely to perform better than 2 or 4 threads.

Vectorization

- Prefer AVX512 instructions and avoid mixing SSE, AVX and AVX512 instructions.
- Avoid cache-line splits; align data structures to 64 bytes.
- Avoid gathers/scatters; replace with shuffles/permutes for known sequences.
- Use hardware trascendentals (fast-math) whenever possible.
- AVX512 achieves best performance when not using masking
- KNC intrinsic code is unlikely to generate optimal KNL code, recompile from HL language.

DATA LOCALITY: NESTED PARALLELISM

- Recall that KNL cores are grouped into tiles, with two cores sharing an L2.
- Effective capacity depends on locality:
  - 2 cores sharing no data => 2 x 512 KB
  - 2 cores sharing all data => 1 x 1 MB
- Ensuring good locality (e.g. through blocking or nested parallelism) is likely to improve performance.




```
#pragma omp parallel for num_threads(ntiles)
for (int i = 0; i < N; ++i)
{
      #pragma omp parallel for num_threads(8)
      for (int j = 0; j < M; ++j)
      {
            …
      }
}
```

KNL PROCESSOR UNTILE

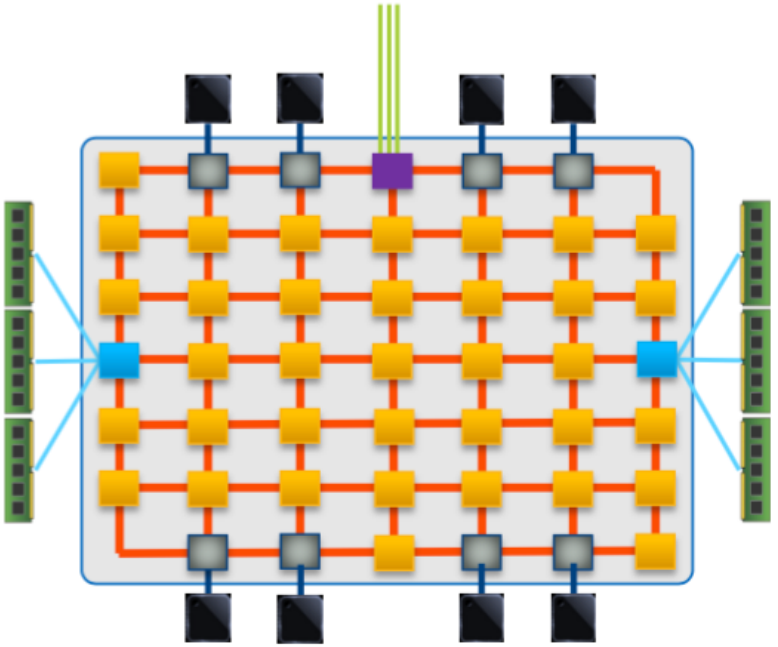
Comprises a mesh connecting the tiles (in red) with the MCDRAM and DDR memories.

- Also with I/O controllers and other agents

Caching Home Agent (CHA) holds portion of the distributed tag directory and serves as connection point between tile and mesh

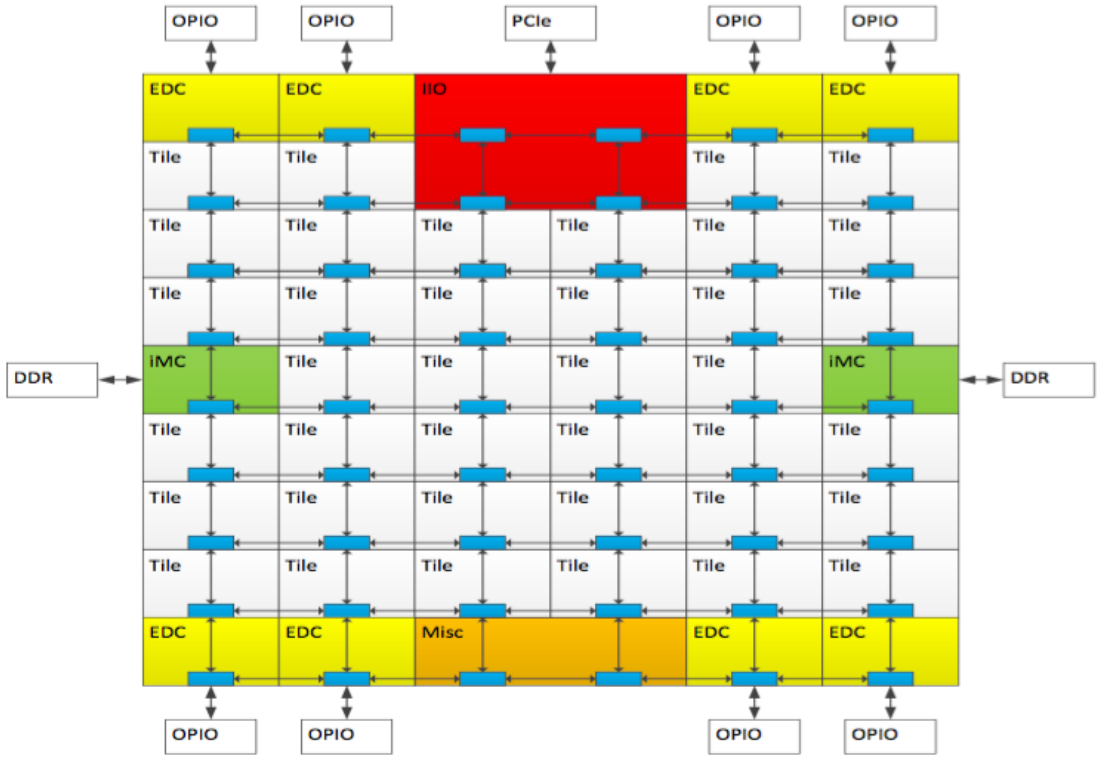
- No L3 cache as in Xeon

Cache coherence uses MESIF protocol (Modified, Exclusive, Shared, Invalid, Forward)



Tile EDC (embedded DRAM controller) IMC (integrated memory controller) IIO (integrated I/O controller)

KNL MESH INTERCONNECT



Mesh of Rings

- Every row and column is a ring
- YX routing: Go in Y → Turn → Go in X
  - 1 cycle to go in Y, 2 cycles to go in X
- Messages arbitrate at injection and on turn

Mesh at fixed frequency of 1.7 GHz

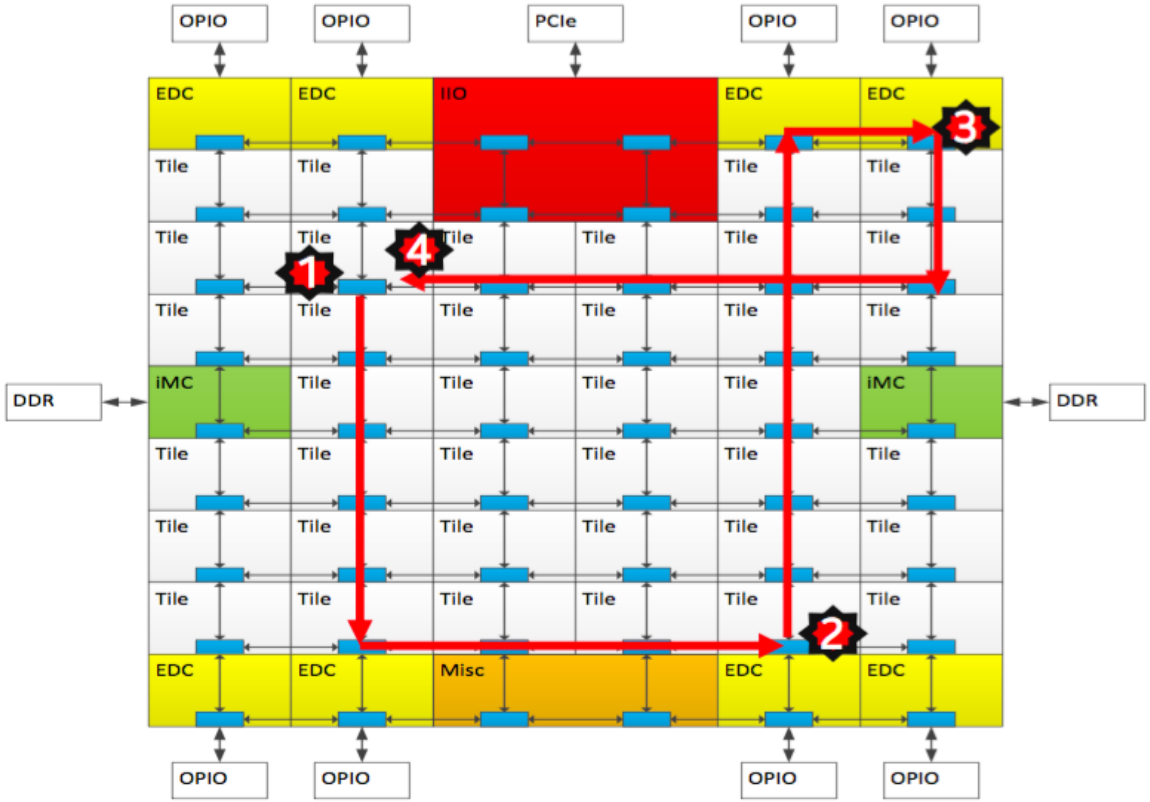
Distributed Directory Coherence protocol

KNL supports Three Cluster Modes

1) All-to-all
2) Quadrant
3) Sub-NUMA Clustering

Selection done at boot time.

CLUSTER MODE: ALL-TO-ALL



Address uniformly hashed across all distributed directories

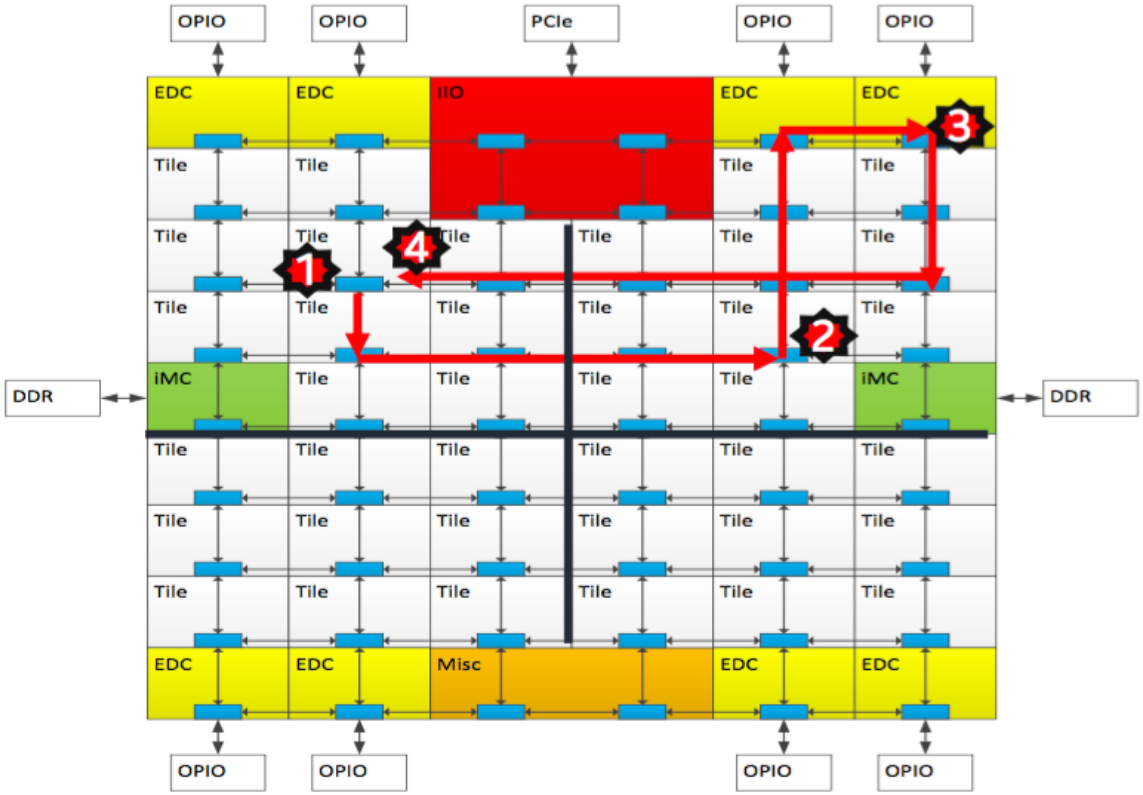
No affinity between Tile, Directory and Memory

Lower performance mode, compared to other modes. Mainly for fall-back

Typical Read L2 miss

1. L2 miss encountered
2. Send request to the distributed directory
3. Miss in the directory. Forward to memory
4. Memory sends the data to the requestor

CLUSTER MODE: QUADRANT



Chip divided into four Quadrants

Affinity between the Directory and Memory

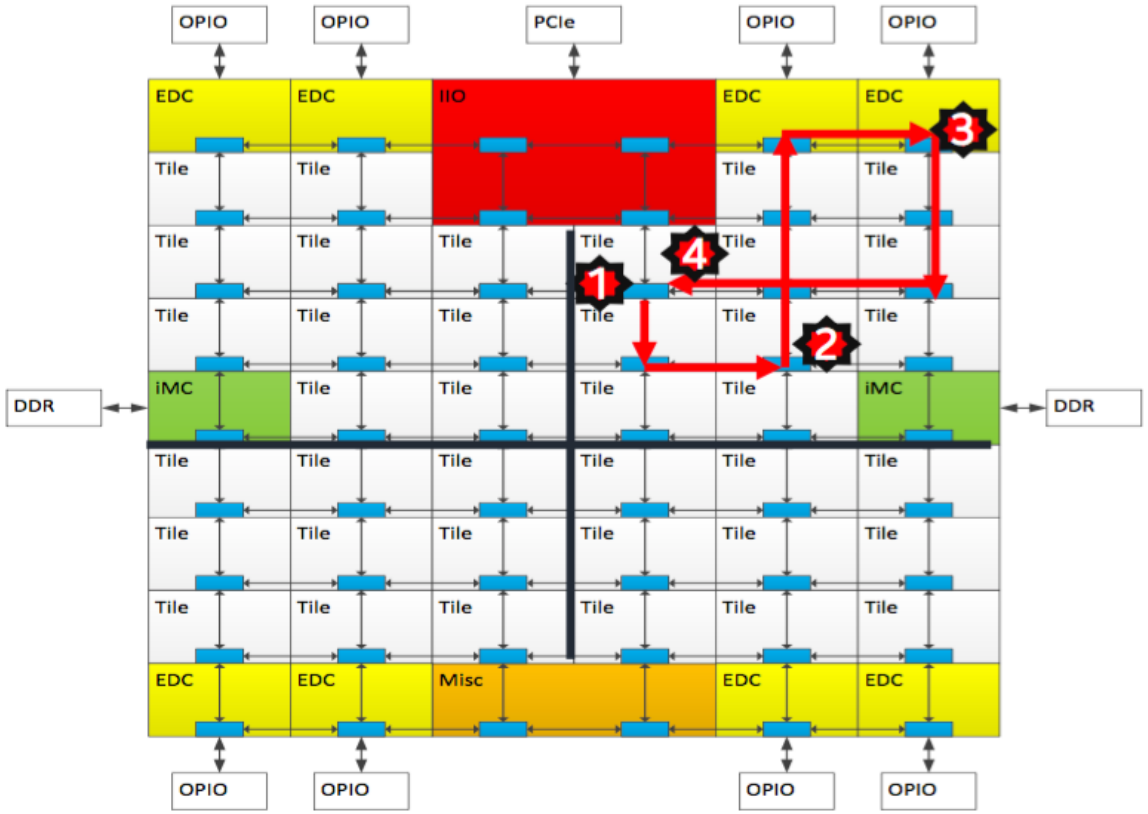
Lower latency and higher BW than all-to-all

SW Transparent

Typical Read L2 miss

1. L2 miss encountered
2. Send request to the distributed directory
3. Miss in the directory. Forward to memory
4. Memory sends the data to the requestor

CLUSTER MODE: SUB-NUMA CLUSTERING (SNC4)



Each Quadrant (Cluster) exposed as a separate NUMA domain to OS

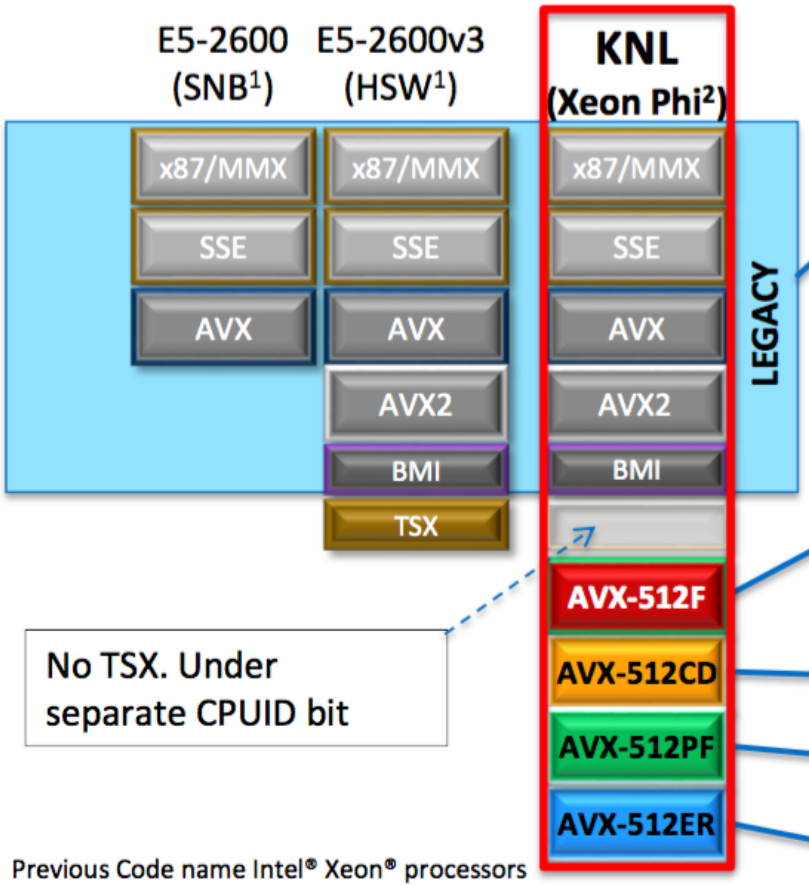
Analogous to 4-socket Xeon

SW Visible

Typical Read L2 miss

1. L2 miss encountered
2. Send request to the distributed directory
3. Miss in the directory. Forward to memory
4. Memory sends the data to the requestor

KNL HARDWARE INSTRUCTION SET



1. Previous Code name Intel® Xeon® processors
2. Xeon Phi = Intel® Xeon Phi™ processor

KNL implements all legacy instructions

- Legacy binary runs w/o recompilation
- KNC binary requires recompilation

KNL introduces AVX-512 Extensions

- 512-bit FP/Integer Vectors
- 32 registers & 8 mask registers
- Gather/Scatter

Conflict Detection: Improves Vectorization

Prefetch: Gather and Scatter Prefetch

Exponential and Reciprocal Instructions

GUIDELINES FOR WRITING VECTORIZABLE CODE

Prefer simple “for” or “DO” loops

Write straight line code. Try to avoid:

- function calls (unless inlined or SIMD-enabled functions)
- branches that can’t be treated as masked assignments.

Avoid dependencies between loop iterations

- Or at least, avoid read-after-write dependencies

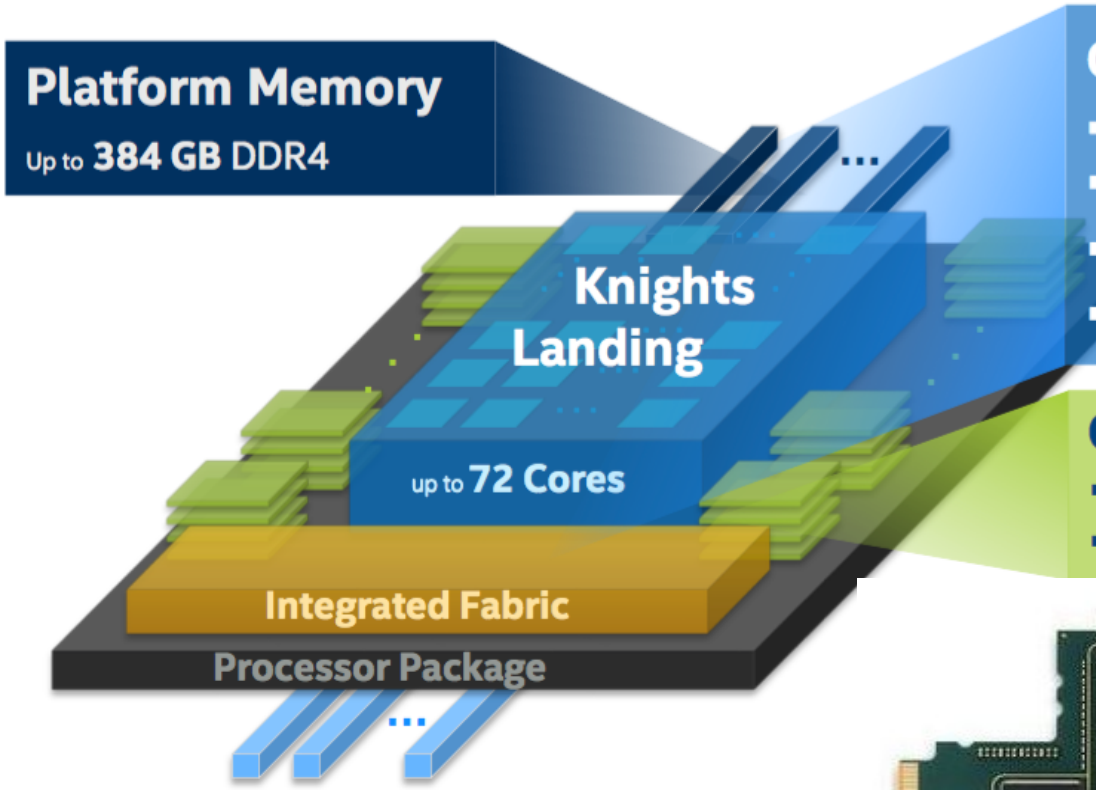
Prefer arrays to the use of pointers

- Without help, the compiler often cannot tell whether it is safe to vectorize code containing pointers.
- Try to use the loop index directly in array subscripts, instead of incrementing a separate counter for use as an array address.
- Disambiguate function arguments, e.g. -fargument-noalias

Use efficient memory accesses

- Favor inner loops with unit stride
- Minimize indirect addressing a[i] = b[ind[i]]
- Align your data consistently where possible (to 16, 32 or 64 byte boundaries)

INTEL® XEON PHI™ X200 PROCESSOR OVERVIEW



Compute

- Intel® Xeon® Processor Binary-Compatible
- 3+ TFLOPS, 3X ST (single-thread) perf. vs KNC
- 2D Mesh Architecture
- Out-of-Order Cores

On-Package Memory (MCDRAM)

- Up to 16 GB *at launch*
- Over 5x STREAM *vs. DDR4 at launch*

MCDRAM MODES

Cache mode

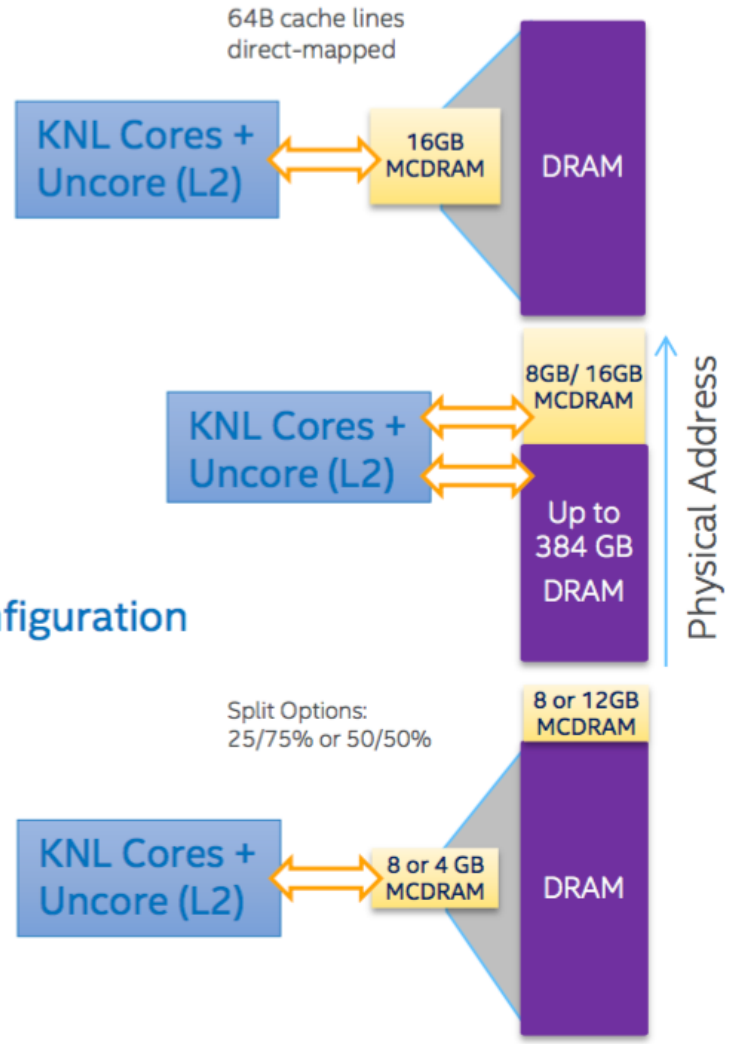
- Direct mapped cache
- Inclusive cache
- Misses have higher latency
  - Needs MCDRAM access + DDR access
- No source changes needed to use, automatically managed by hw as if LLC

Flat mode

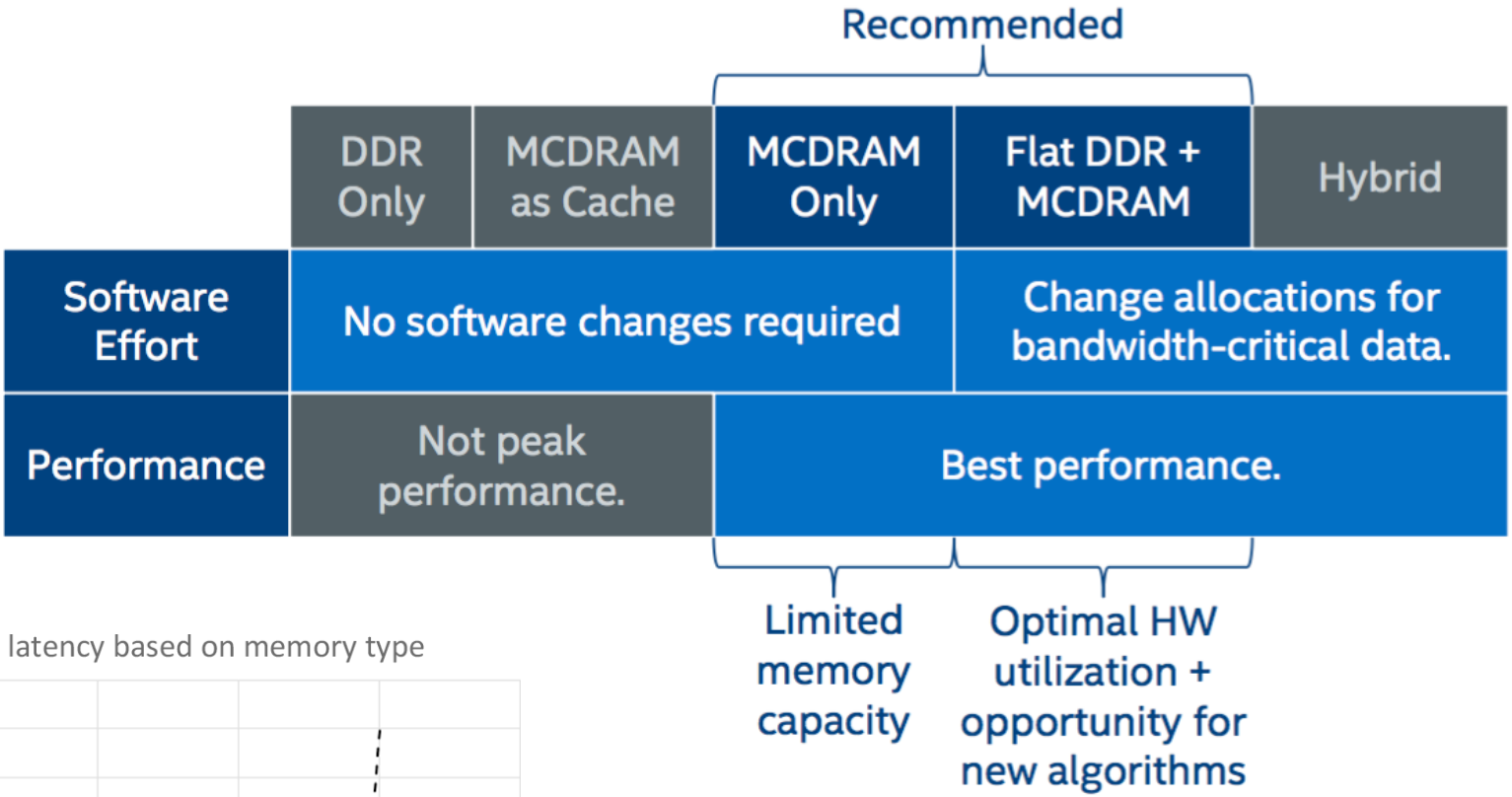
- MCDRAM mapped to physical address space
- Exposed as a NUMA node
  - Use `numactl --hardware, lscpu` to display configuration
- Accessed through memkind library or numactl

Hybrid

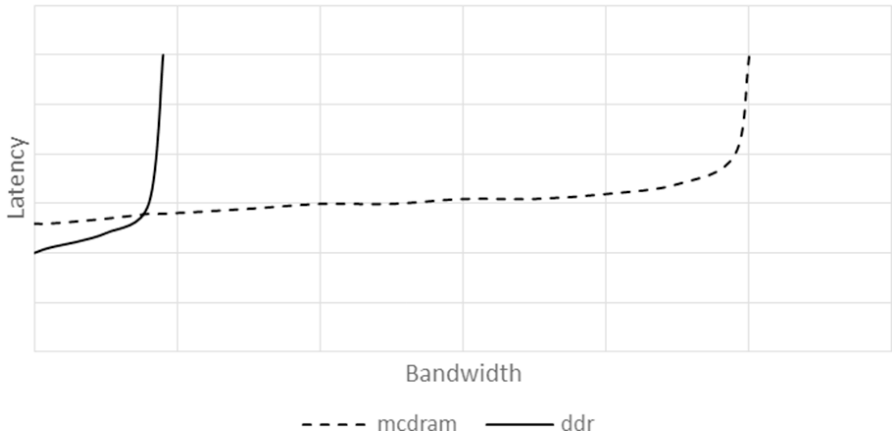
- Combination of the above two
  - E.g., 8 GB in cache + 8 GB in Flat Mode



TAKE AWAY MESSAGE: CACHE VS FLAT MODE

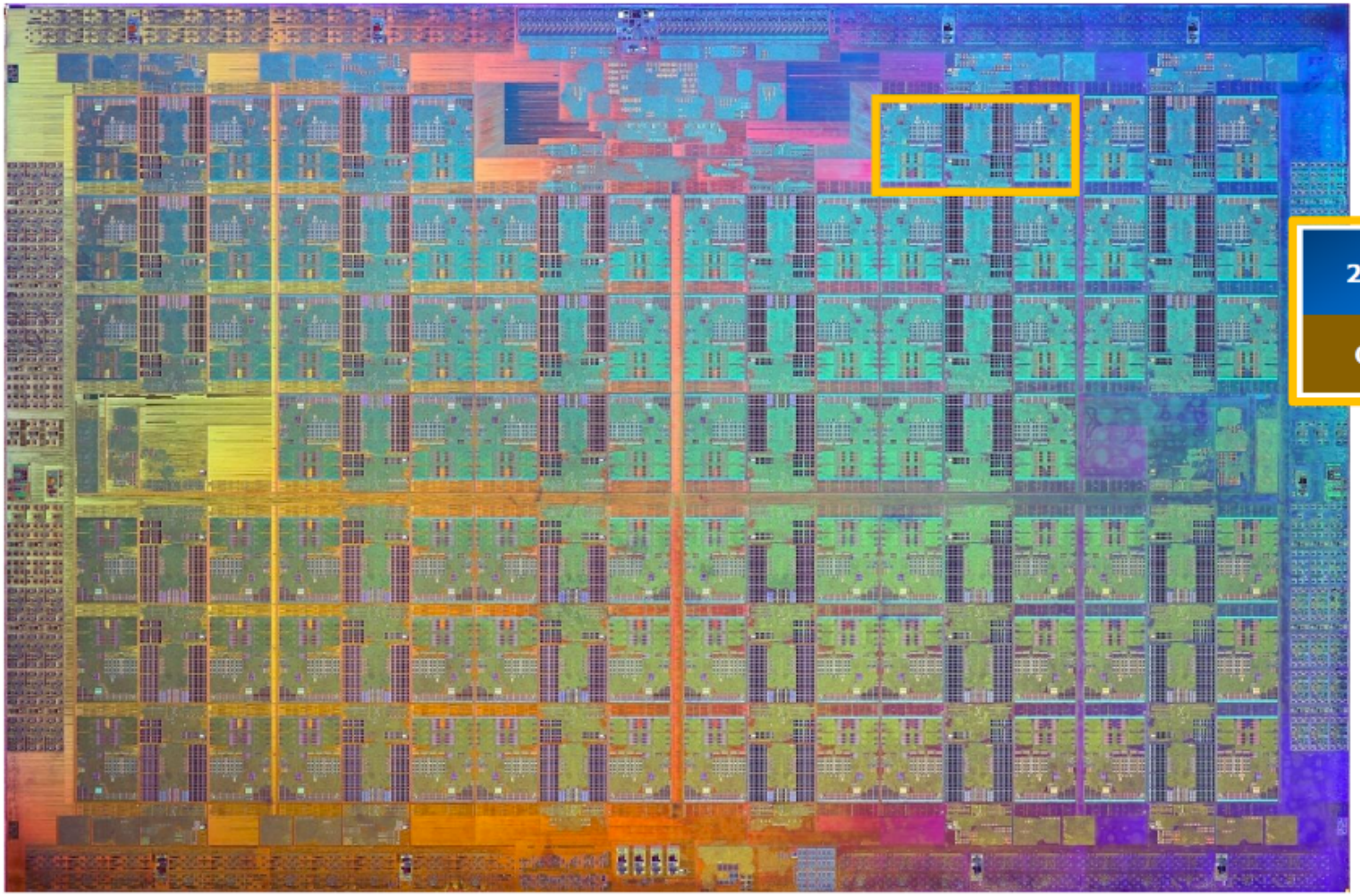


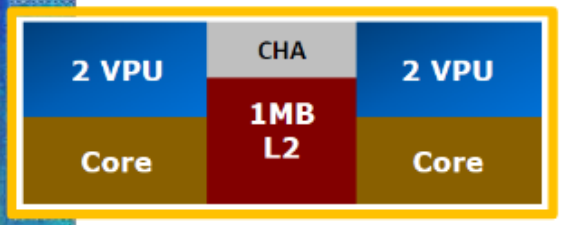
Bandwidth versus latency based on memory type



BAYNCORE

Intel® Knights Landing die





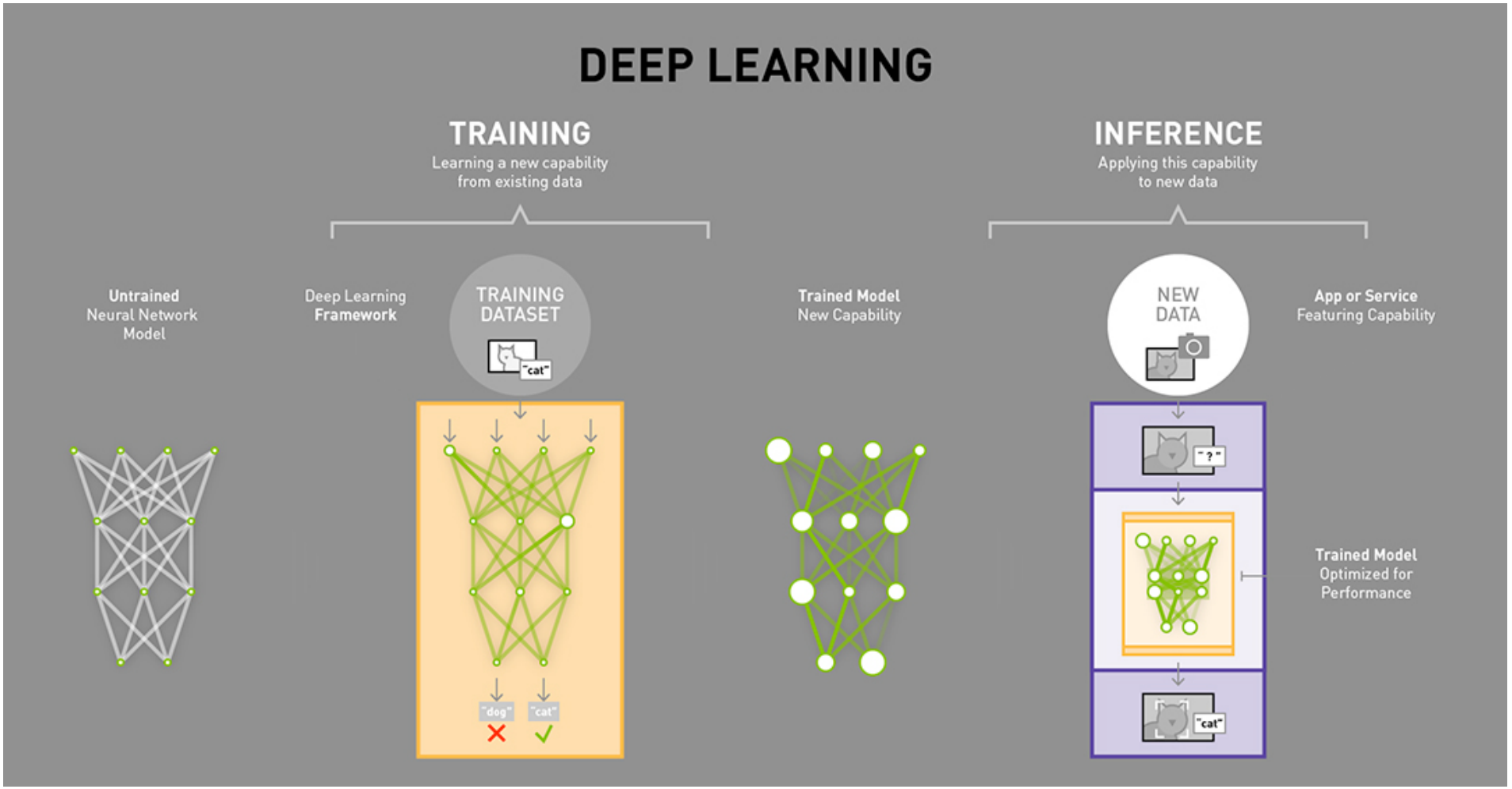
Notes:

- 38 tiles, 76 cores
- max announced:
  36 tiles
- max on sale:
  34 tiles

Beyond vector extensions

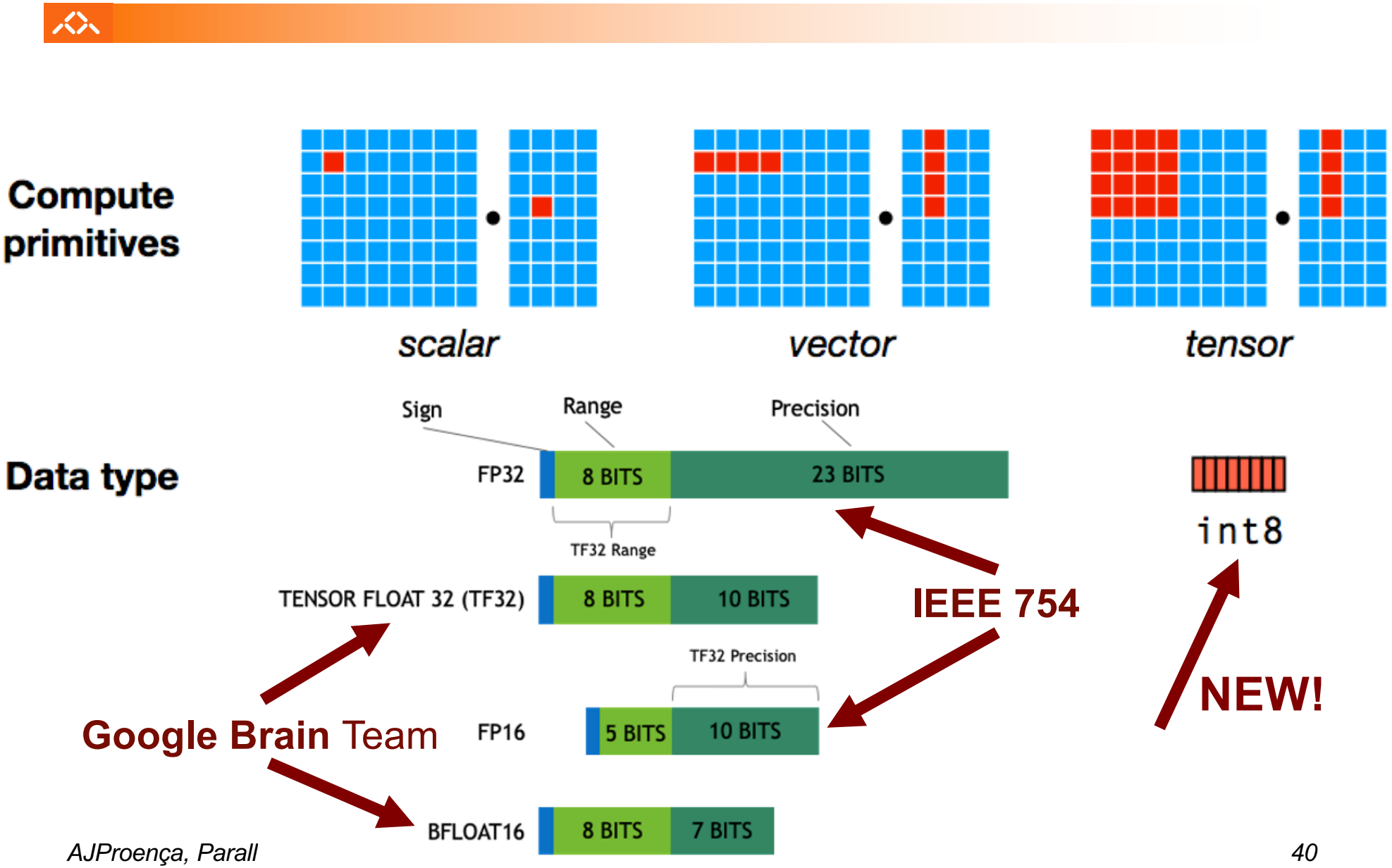
- Vector/SIMD-extended architectures are hybrid approaches
  - mix (super)**scalar + vector** op capabilities on a single device
  - **highly pipelined** approach to reduce memory access penalty
  - **tightly-closed access to shared memory**: lower latency
- Evolution of vector/SIMD-extended architectures
  - **computing accelerators optimized for number crunching (GPU)**
  - **add support for matrix multiply + accumulate operations**; **why?**
    - most scientific, engineering, AI & finance applications use matrix computations, namely the dot product: multiply and accumulate the elements in a row of a matrix by the elements in a column from another matrix
    - manufacturers typically call these extension **Tensor Processing Unit** (TPU)
  - **support for half-precision FP & 8-bit integer; why?**
    - **machine learning using neural nets** is becoming very popular; to compute the model parameter during training phase, intensive matrix products are used and with very low precision (is adequate!)

Machine learning w/ neural nets & deep learning...



Key algorithms to train & classify use matrix dot products,
but require lower precision numbers!

Required hardware operations & data types to train & classify neural nets



NVidia Volta Architecture: the new Tensor Cores

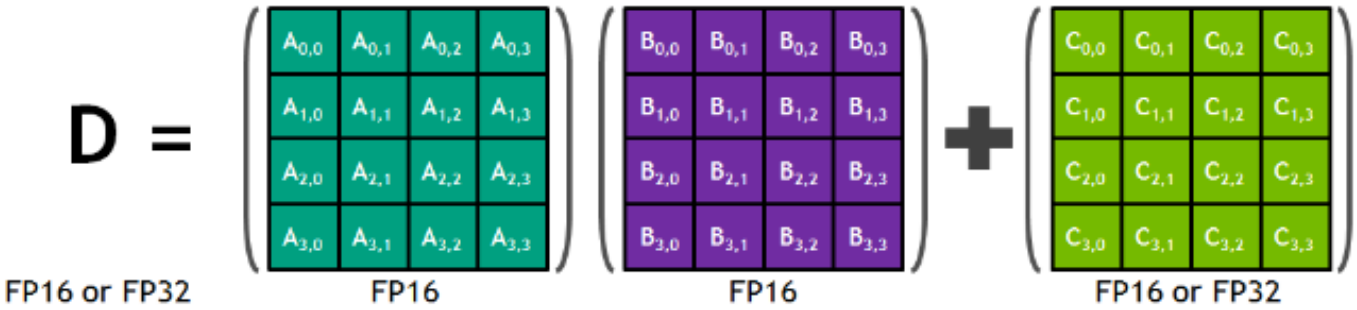


Figure 8. Tensor Core 4x4 Matrix Multiply and Accumulate

For each SM:
8x 64 FMA ops/cycle
1k FLOPS/cycle!

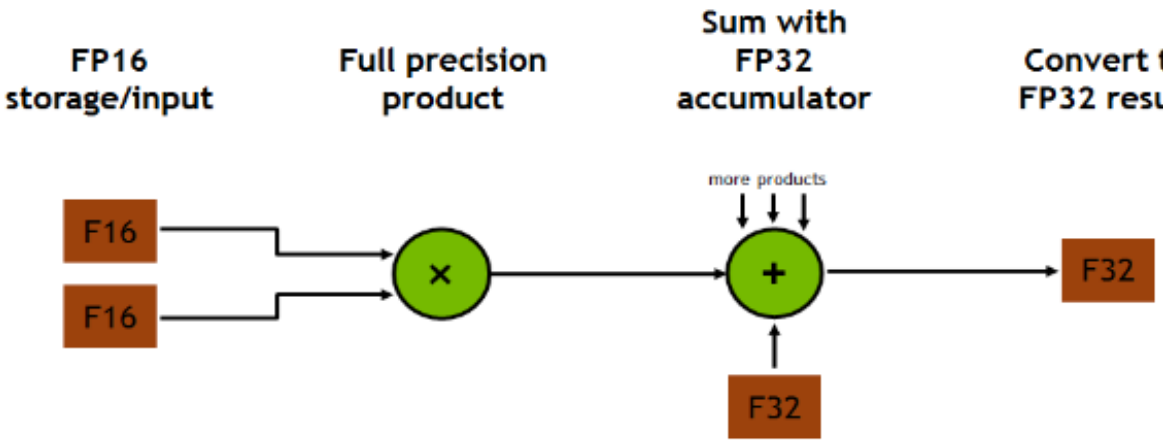
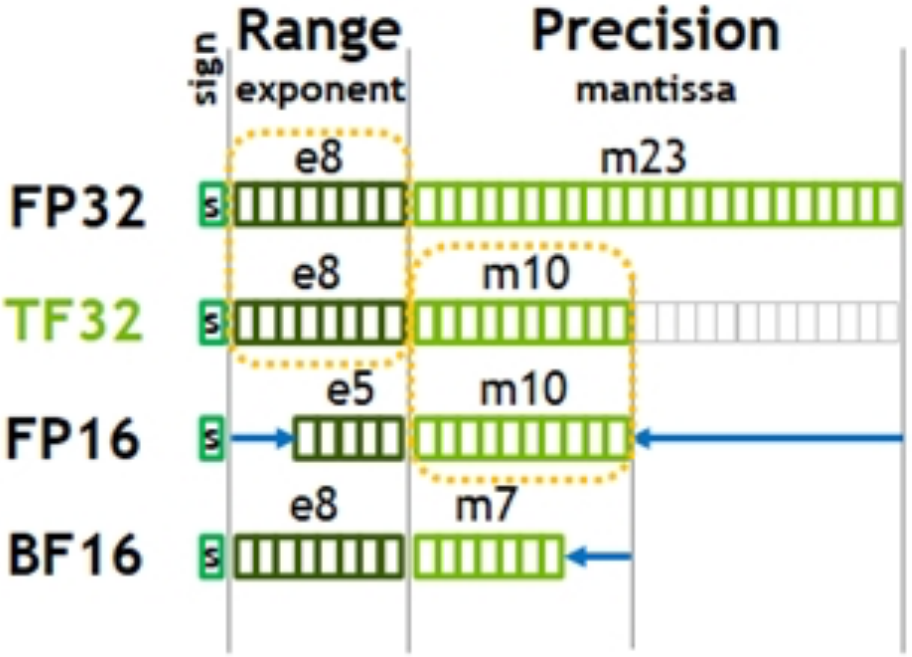
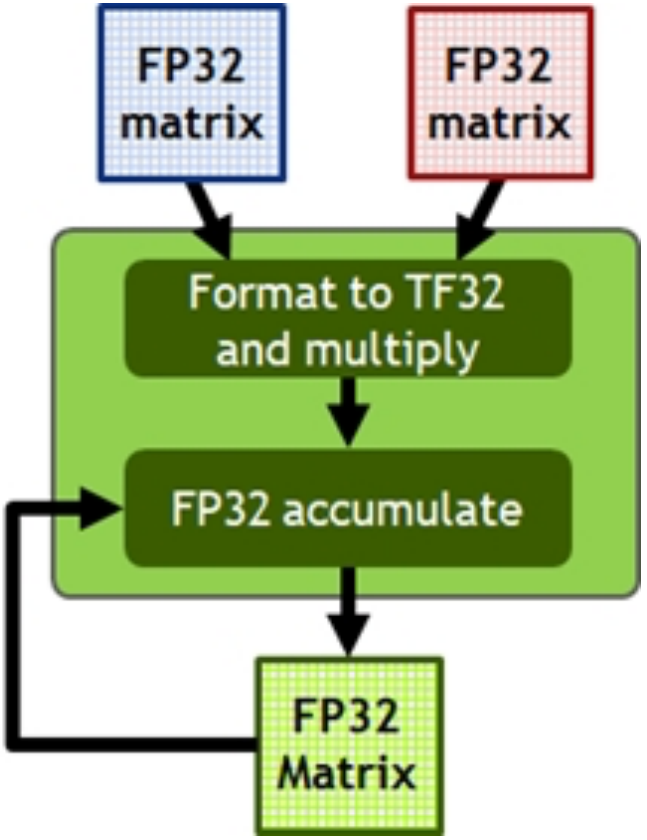


Figure 9. Mixed Precision Multiply and Accumulate in Tensor Core

http://www.nvidia.com/content/gated-pdfs/Volta-Architecture-Whitepaper-v1.1.pdf

Tensor Cores in NVidia Ampere



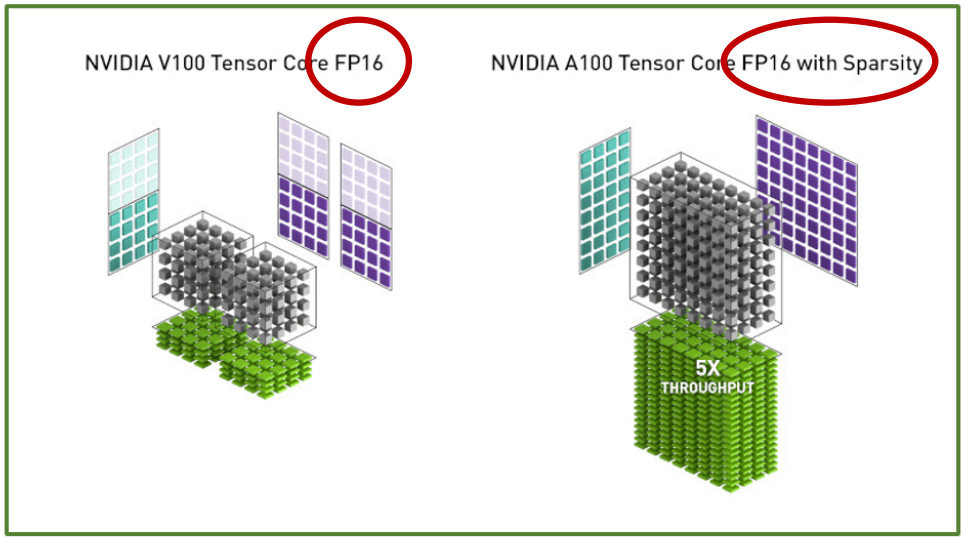


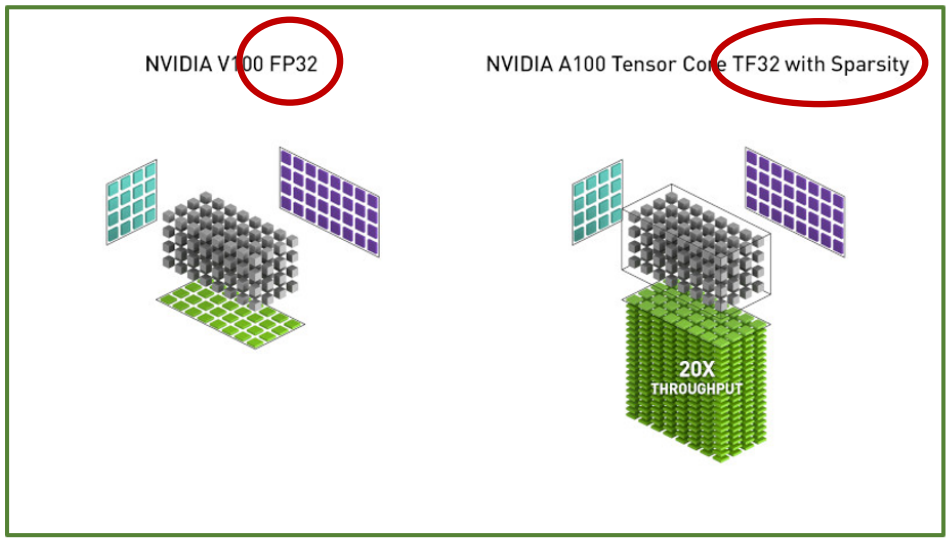
TF32: same range as FP32 and same precision as FP16

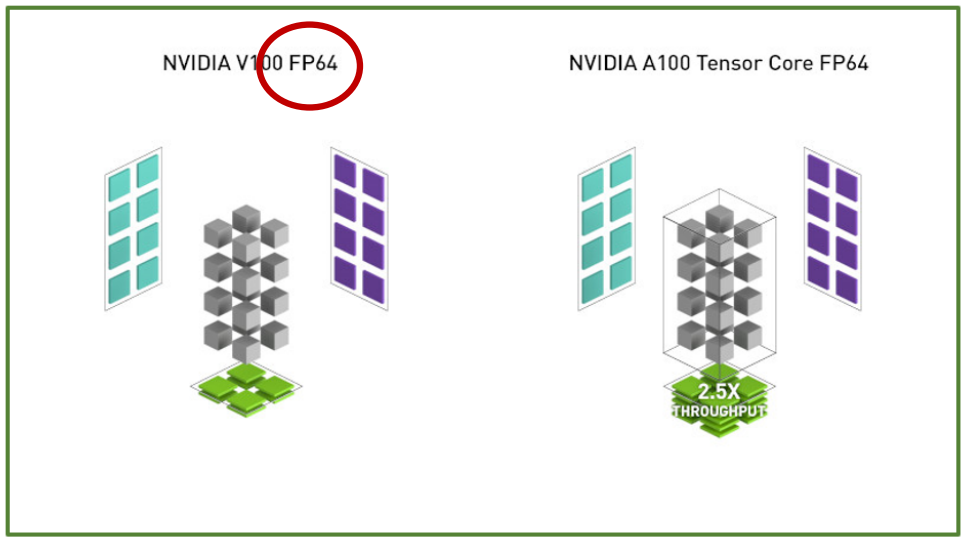
The FP multiplier scales with the square of the mantissa width ($8^2/11^2 \approx 0.5$)

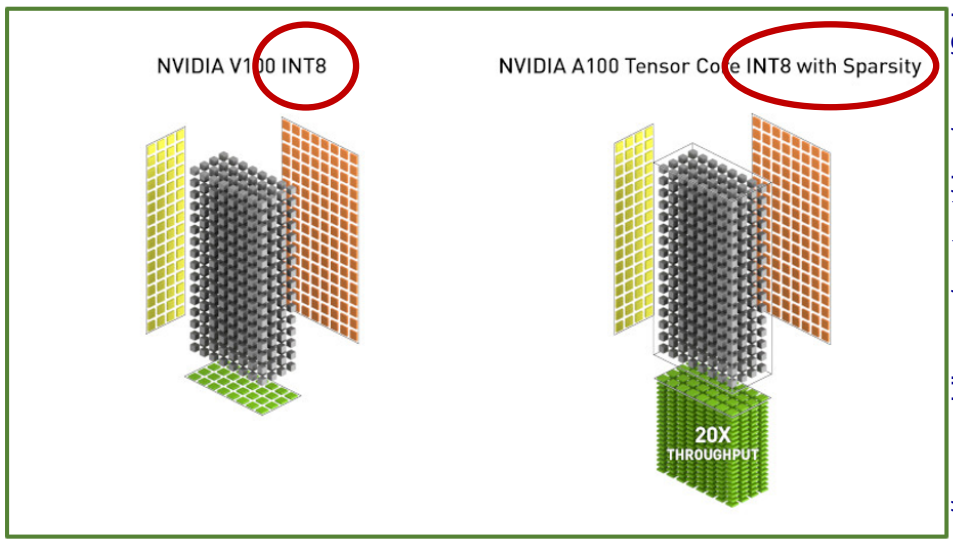
https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/nvidia-ampere-architecture-whitepaper.pdf

Tensor Cores: Volta vs. Ampere









https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/nvidia-ampere-architecture-whitepaper.pdf

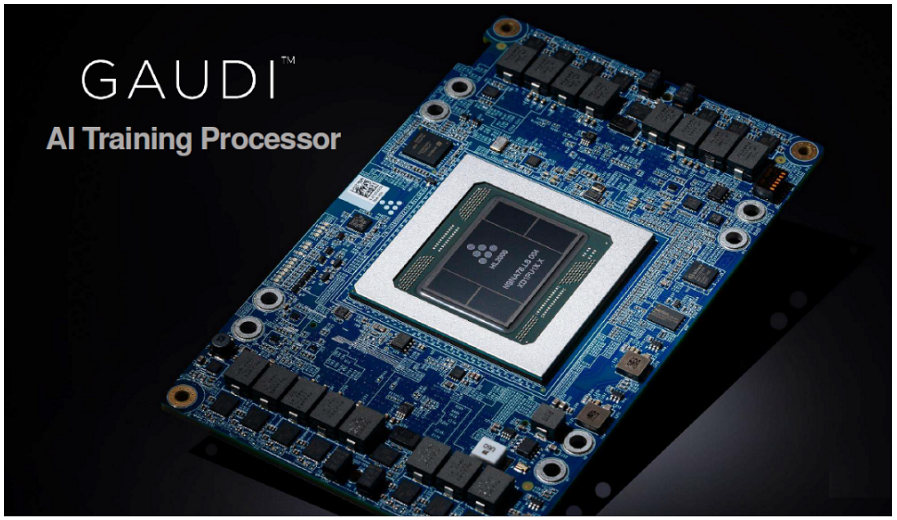
NVidia competitors with neural net features: Intel with Nervana & later Habana

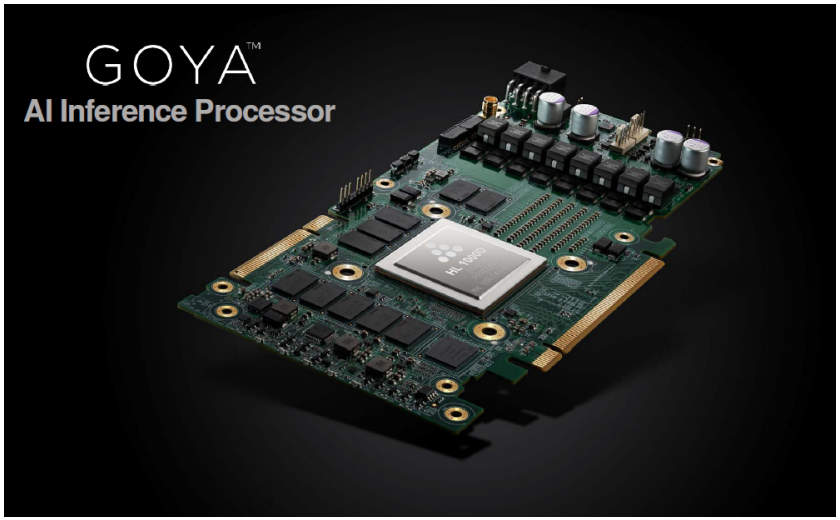
History

- Intel buys **Nervana** Engine *(Aug 2016)*
- Intel launches Nervana NNP (Neural Net Processor) *(Oct 2017)*
- Key features: matrix multiplication & convolution *(for neural nets)*
- Intel discontinues Nervana NNP *(Jan 2020…)*

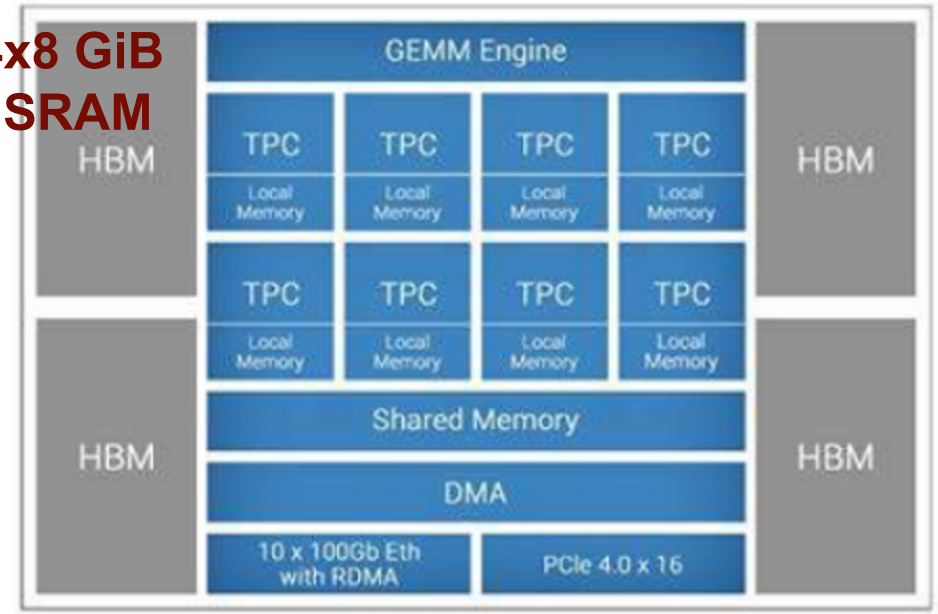
- Intel buys the Israel chipmaker **Habana** Labs *(Dec 2019)*
  - Habana training chip **Gaudi**, with support to FP32, INT32, **BF16**, INT16, INT8, UINT32, UINT16, UINT8
  - Habana inference chip **Goya**, with support to FP32, INT32, INT16, INT8, UINT32, UINT16, UINT8

Habana chips

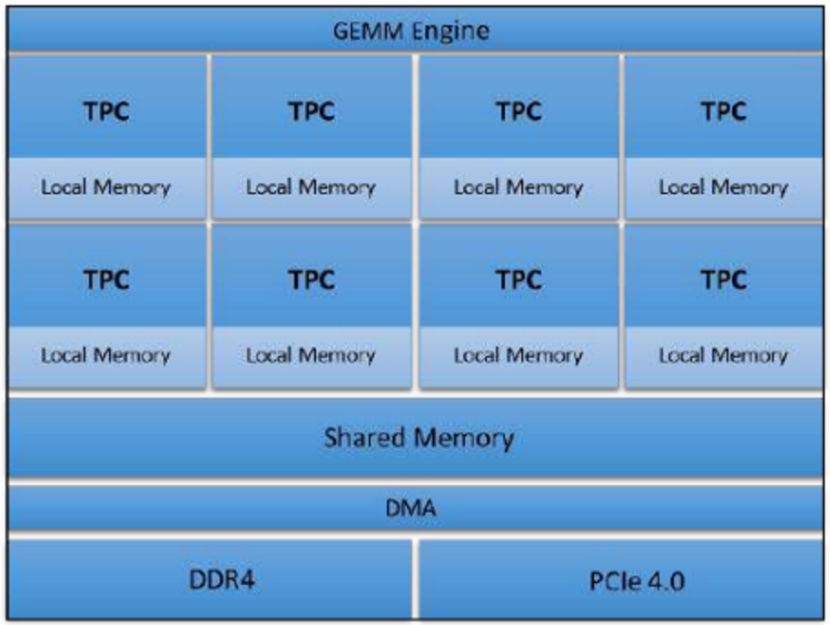




4x8 GiB SRAM



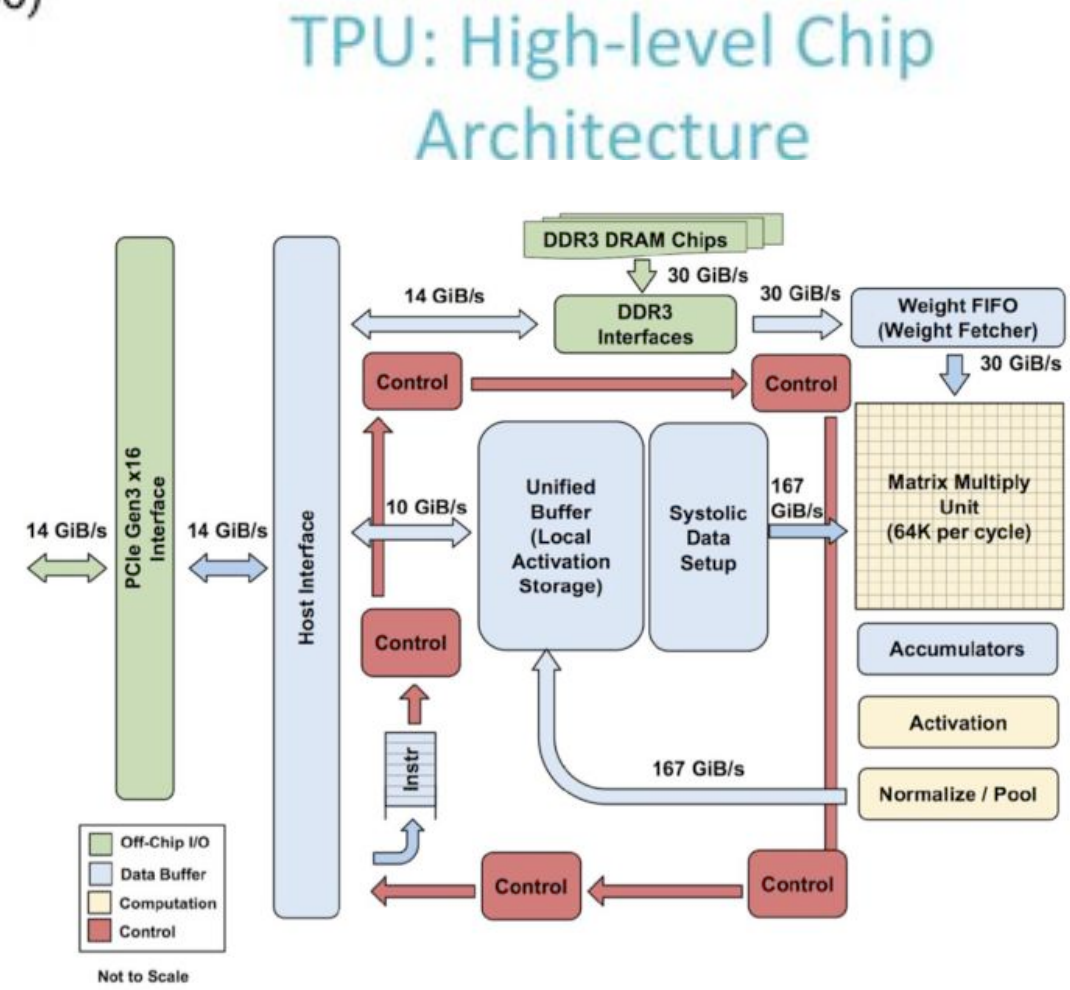
TSMC – 16nm



TSMC – 16nm

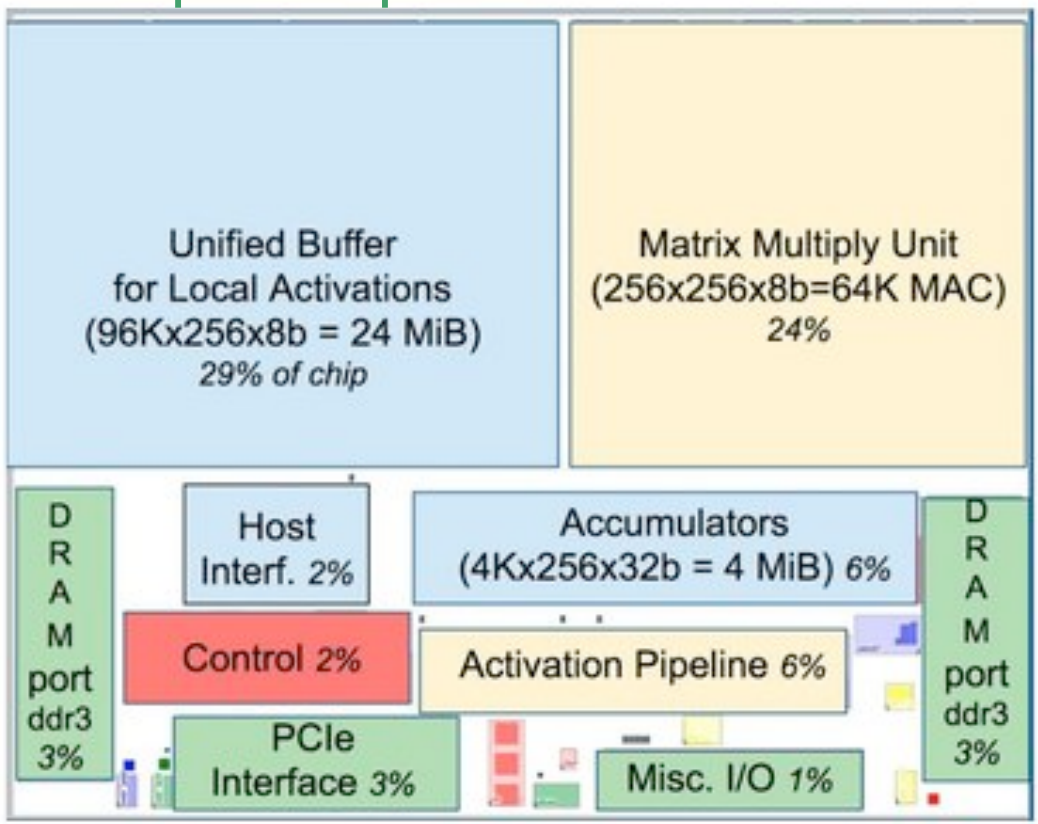
Google Tensor Processing Unit, TPU (May'16)

- The Matrix Unit: 65,536 (256x256) 8-bit multiply-accumulate units
- 700 MHz clock rate
- Peak: 92T operations/second
  - 65,536 * 2 * 700M
- >25X as many MACs vs GPU
- >100X as many MACs vs CPU
- 4 MiB of on-chip Accumulator memory
- 24 MiB of on-chip Unified Buffer (activation memory)
- 3.5X as much on-chip memory vs GPU
- Two 2133MHz DDR3 DRAM channels
- 8 GiB of off-chip weight DRAM memory

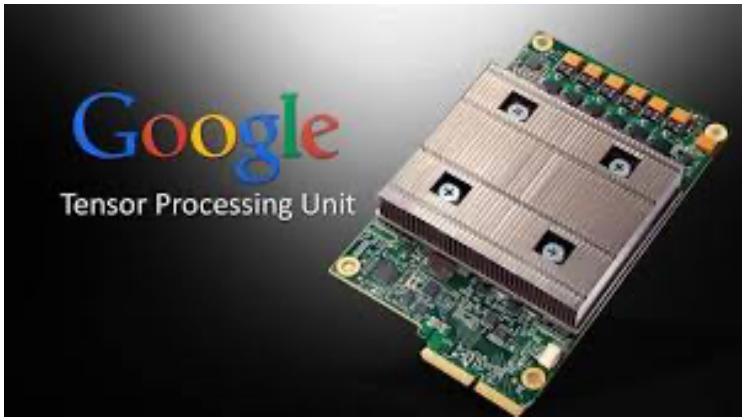


Google Tensor Processing Unit, TPU (May'16)

Chip floor plan



TPU: a Neural Network Accelerator Chip



TPUs are intensively used by Google, namely in Google Photos, RankBrain, StreetView & Google Translate

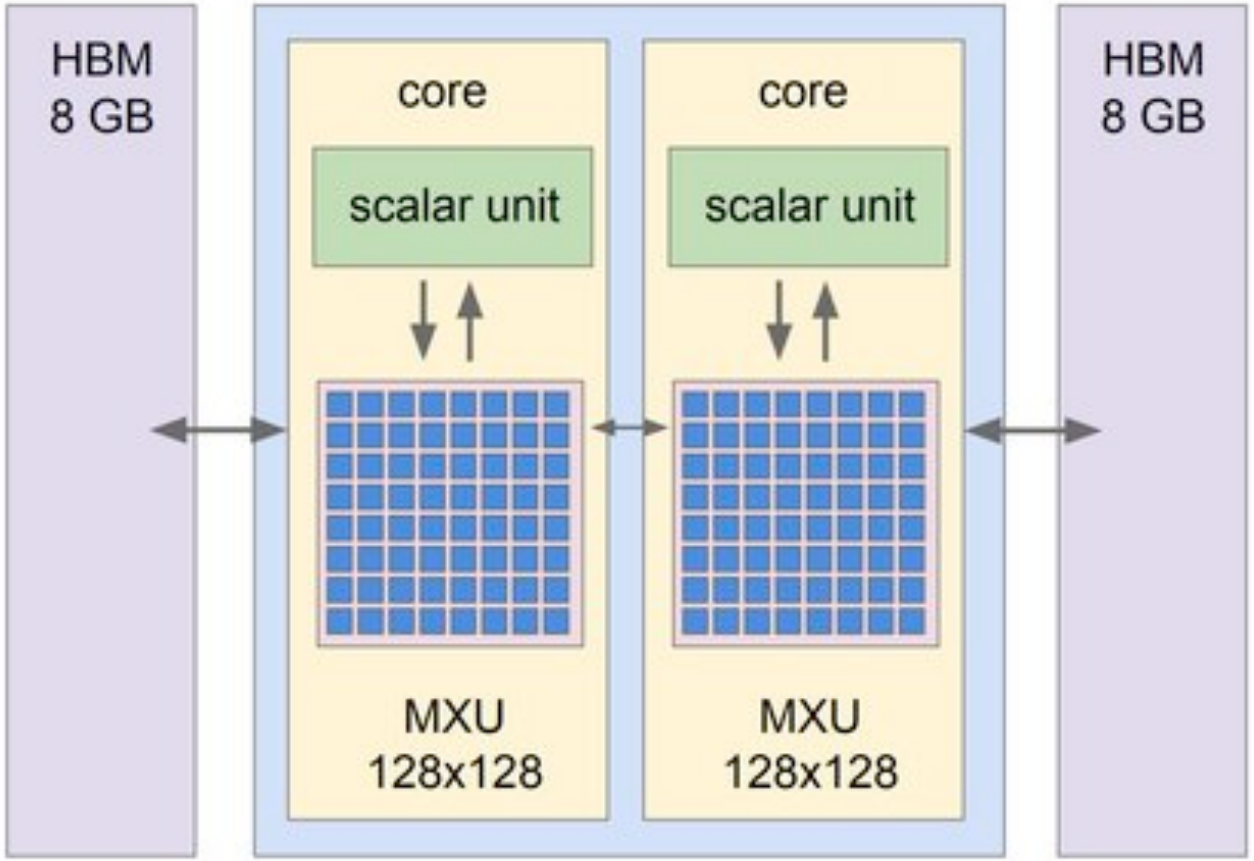
https://cloud.google.com/blog/big-data/2017/05/an-in-depth-look-at-googles-first-tensor-processing-unit-tpu

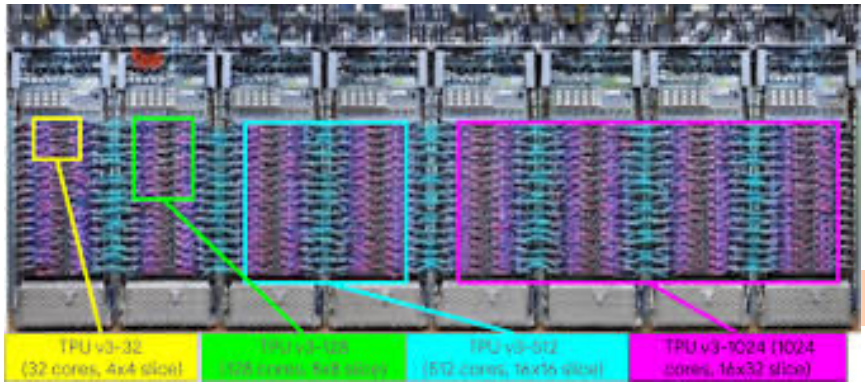
Google TPUv2 (May'17)

TPUv2 Chip

- 16 GB of HBM
- 600 GB/s mem BW
- Scalar unit: 32b float
- MXU: 32b float accumulation but reduced precision for multipliers
- 45 TFLOPS

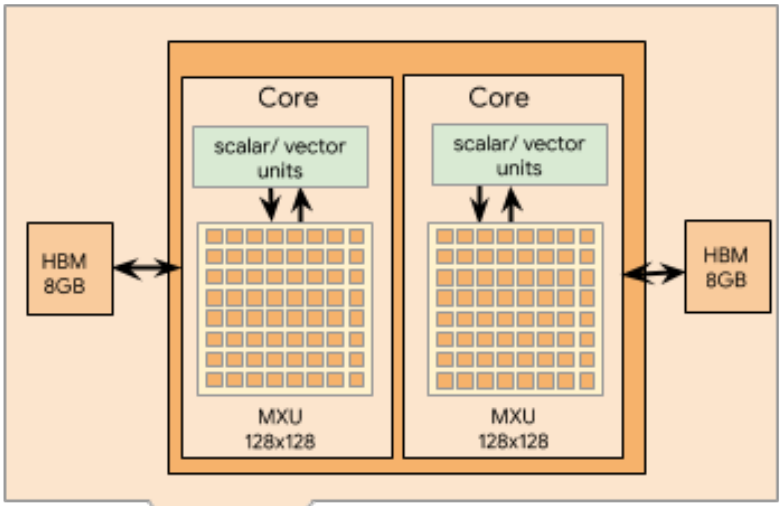
bfloat

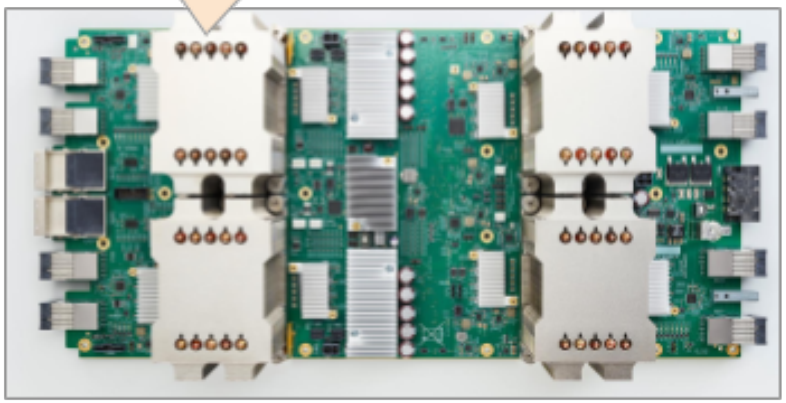




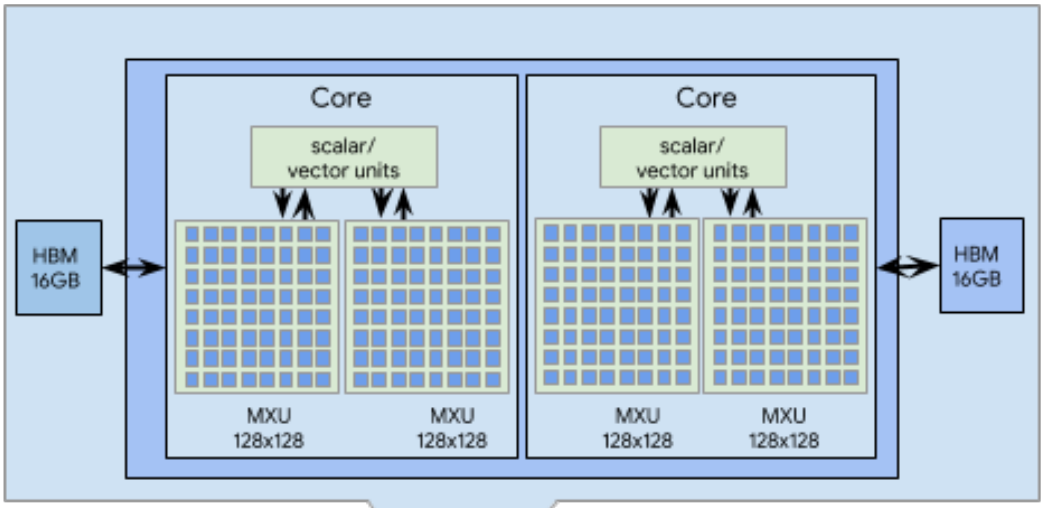
Google TPUv3 (May'18)

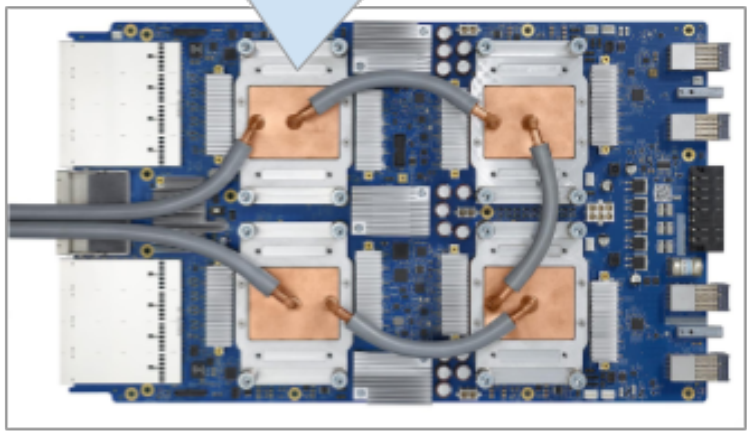
TPUv4 released Jun 2020
but no data available yet…





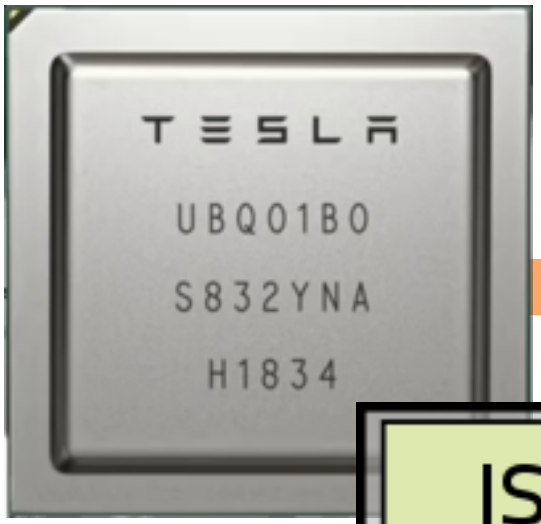
TPU v2 - 4 chips, 2 cores per chip

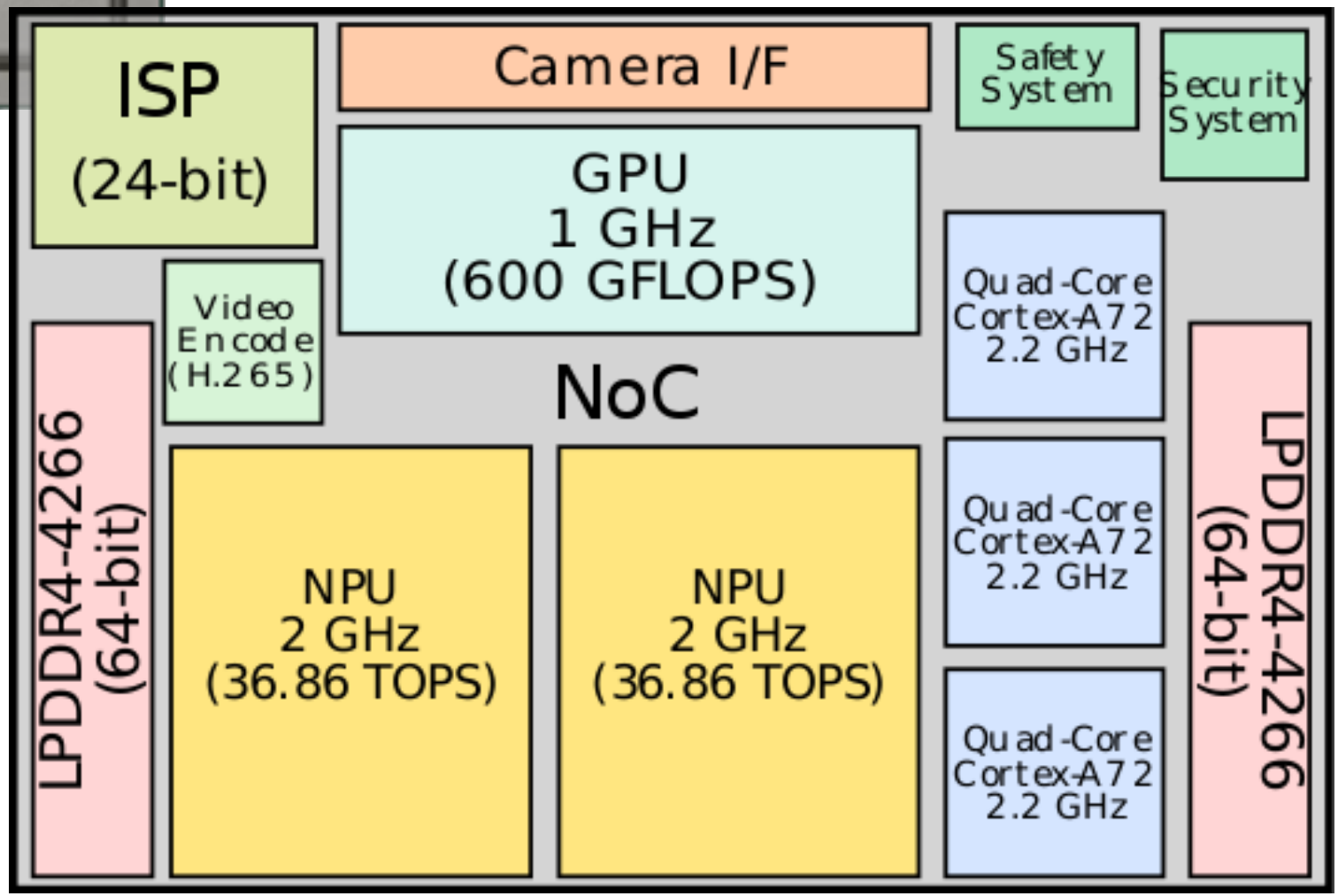




TPU v3 - 4 chips, 2 cores per chip

Tesla Full Self-Driving chip (FSD)





The Neural Processing Unit in FSD