中山大學 SUN YAT-SEN UNIVERSITY

国家超级计算广州中心 NATIONAL SUPERCOMPUTER CENTER IN GUANGZHOU

Computer Architecture

计 算 机 体 系 结 构

第22讲：WSC & Interconnect (3)

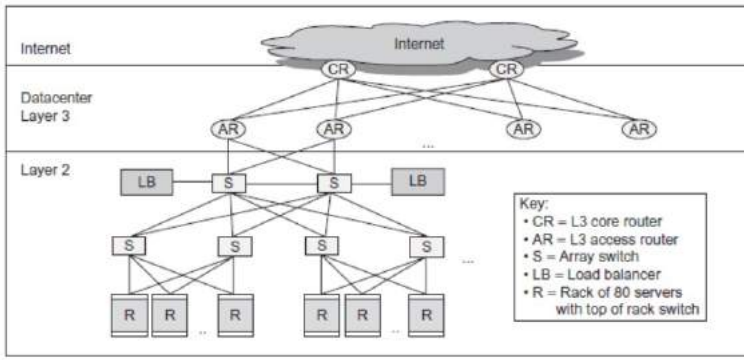
张献伟

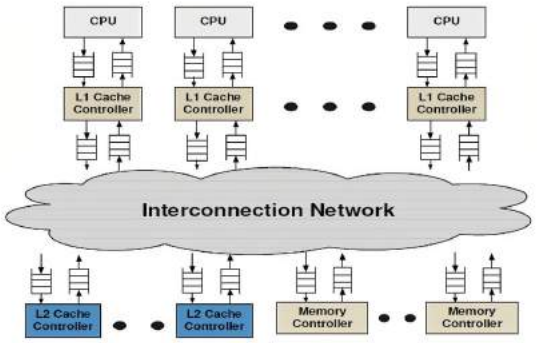
xianweiz.github.io

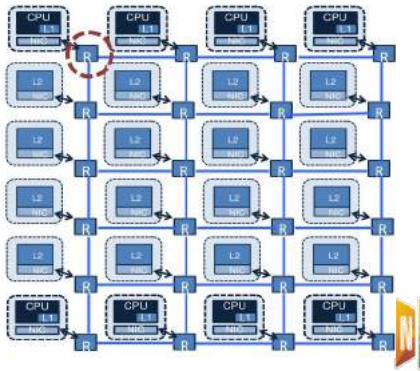
DCS3013, 12/19/2022

Review

- Warehouse scale computer
  - Request level parallelism
  - Server-rack-array-WSC
  - Load variance, fault tolerance
  - Power Usage Effectiveness (PUE)
- Interconnection network
  - Classification: based on number and proximity of devices to be connected
  - Four domains: LAN, WAN, SAN, On-chip
    - We are concerned with system-area and on-chip networks





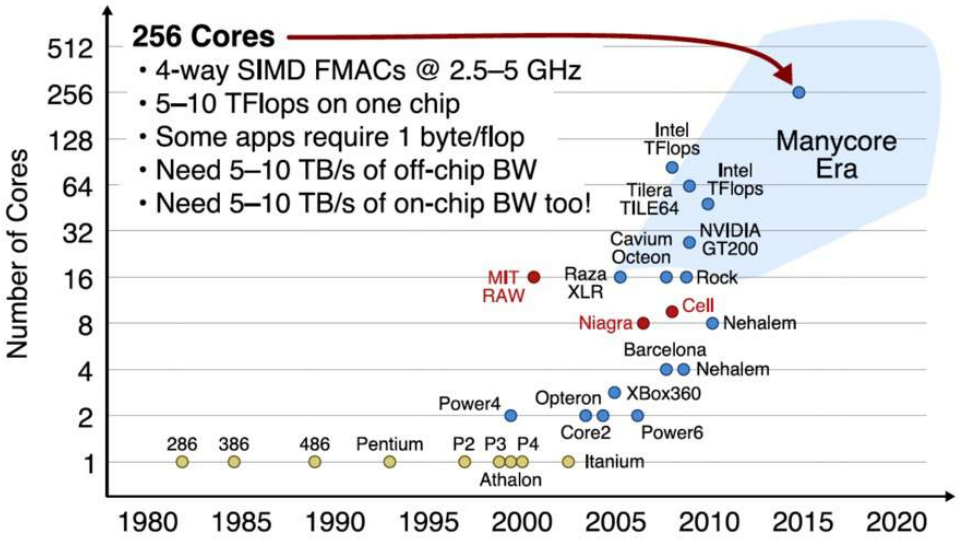


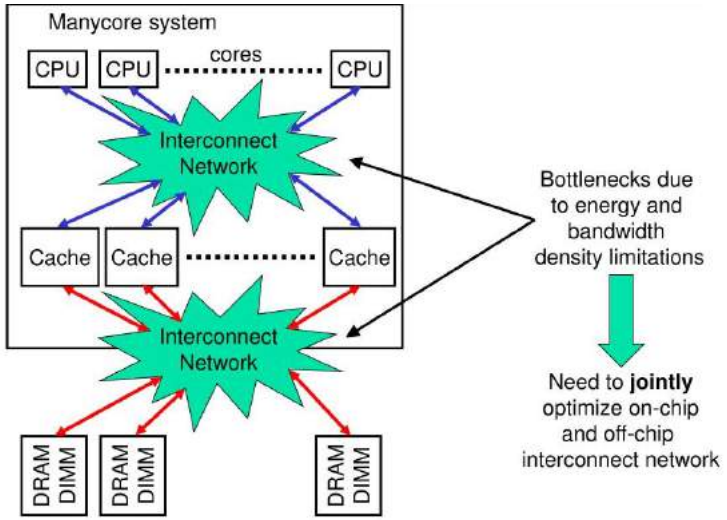
Why Study Interconnects?

- They provide external connectivity from system to outside world
  - Also, connectivity within a single computer system at many levels
    - I/O units, boards, chips, modules and blocks inside chips
- Interconnection networks should be well designed
  - To transfer the maximum amount of information
  - Within the least amount of time (and cost, power constraints)
  - So as not to bottleneck the system
- Application: managing communication can be critical to performance

Why Study Interconnects? (cont.)

- Trends: high demand on communication bandwidth
  - Increased computing power and storage capacity
  - Switched networks are replacing buses
- Computer architects/engineers must <u>understand interconnect problems and solutions</u> in order to more effectively design and evaluate systems



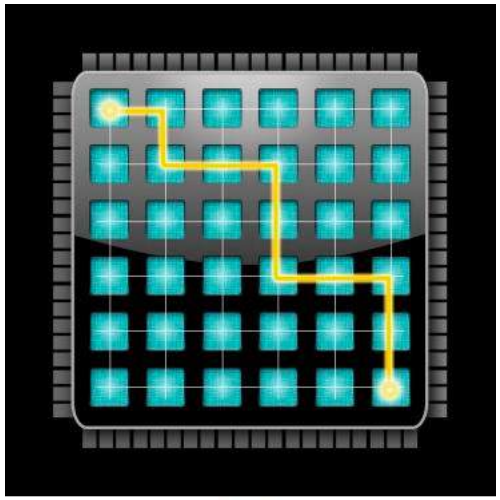


Basic Definitions[基本定义]

- An interconnection network is a graph of ***nodes*** interconnected using ***channels***
- ***Node****[节点]*: a vertex in the network graph
  - ***Terminal*** nodes: where messages originate and terminate
  - ***Switch*** (***router***) nodes: forward messages from in ports to out ports
  - ***Switch degree***: number of in/out ports per switch
- ***Channel****[信道]*: an edge in the graph
  - i.e., an ordered pair (x,y) where x and y are nodes
  - Channel = link (transmission medium) + transmitter + receiver
  - ***Channel width***: w = number of bits transferred per cycle
  - ***Phit*** (physical unit or digit): data transferred per cycle
  - ***Signaling rate***: f = number of transfer cycles per second
  - ***Channel bandwidth***: b = w × f

Basic Definitions (cont.)

- ***Path*** (or ***route***): a sequence of channels, connecting a source node to a destination node
- ***Minimal Path***: a path with the minimum number of channels between a source and a destination
  - Rxy = set of all minimal paths from x to y
- ***Network Diameter***: longest minimal path over all (source, destination) pairs



Source: MIT

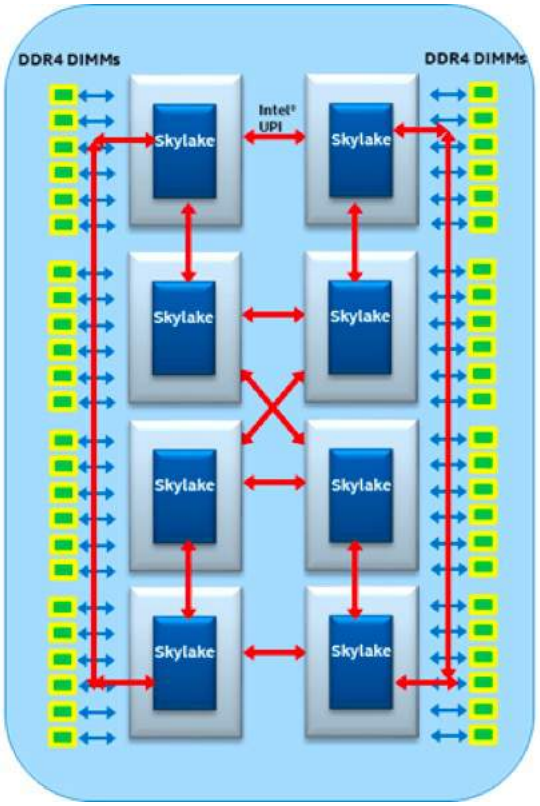
https://compas.cs.stonybrook.edu/~nhonarmand/courses/fa15/cse610/slides/07-icn.pdf

ICN Design Considerations[设计考虑]

- Application requirements
  - Number of terminals or ports to support
  - Peak bandwidth of each terminal
  - Average bandwidth of each terminal
  - Latency requirements
  - Message size distribution
  - Expected traffic patterns
  - Required quality of service
  - Required reliability and availability
- Job of an interconnection network is to transfer information from source node to dest. node in support of network transactions that realize the application
  - Latency as small as possible
  - As many concurrent transfers as possible
  - Cost as low as possible

ICN Design Considerations (cont.)

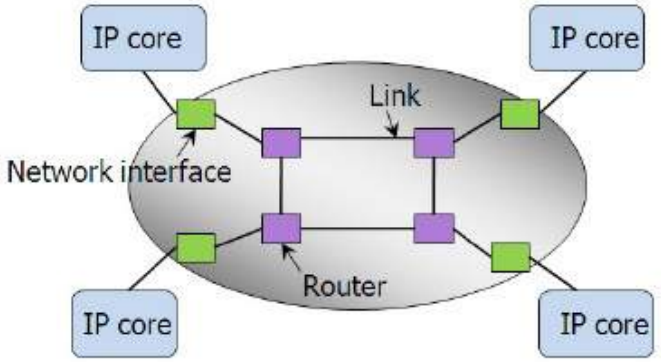
- Example requirements for a coherent processor-memory interconnect
  - Processor ports 1-2048
  - Memory ports 1-4096
  - Peak BW 8 GB/s
  - Average BW 400 MB/s
  - Message latency 100 ns
  - Message size 64 or 576 bits
  - Traffic pattern arbitrary
  - Quality of service none
  - Reliability no message loss
  - Availability 0.999 to 0.99999



Intel® Ultra Path Interconnect (Intel® UPI)

ICN Design Considerations (cont.)

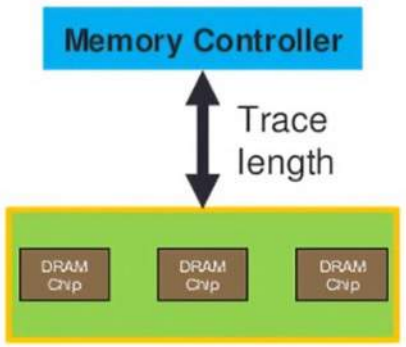
- Technology constraints
  - Signaling rate
  - Chip pin count (if off-chip networking)
  - Area constraints (typically for on-chip networking)
  - Chip cost
  - Circuit board cost (if backplane boards needed)
  - Signals per circuit board
  - Signals per cable
  - Cable cost
  - Cable length
  - Channel and switch power constraints
  - …

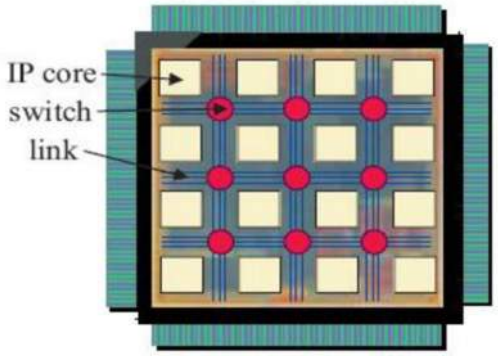


https://compas.cs.stonybrook.edu/~nhonarmand/courses/fa15/cse610/slides/07-icn.pdf

Off-chip vs. On-chip ICNs[片外 vs. 片上]

- Off-chip: I/O bottlenecks
  - Pin-limited bandwidth
  - Inherent overheads of off-chip I/O transmission
- On-chip
  - Wiring constraints
    - Metal layer limitations
    - Horizontal and vertical layout
    - Short, fixed length
    - Repeater[中继器] insertion limits routing of wires
      - Avoid routing over dense logic
      - Impact wiring density
  - Power
    - Consume 10-15% or more of die power budget
  - Latency
    - Different order of magnitude
    - Routers consume significant fraction of latency



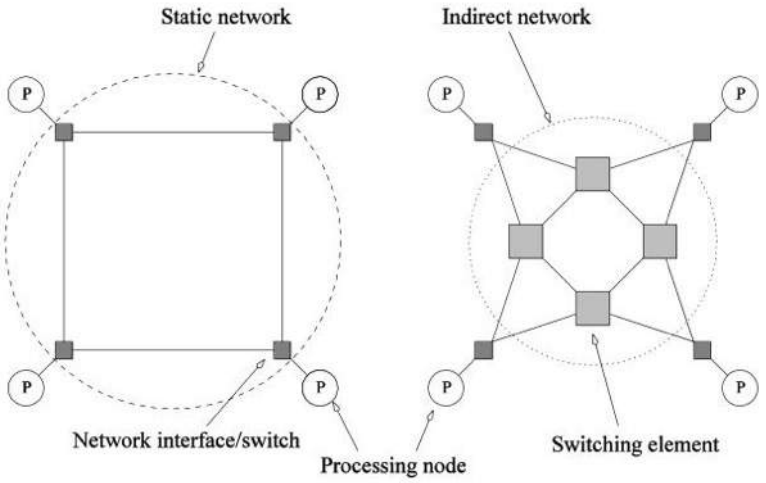


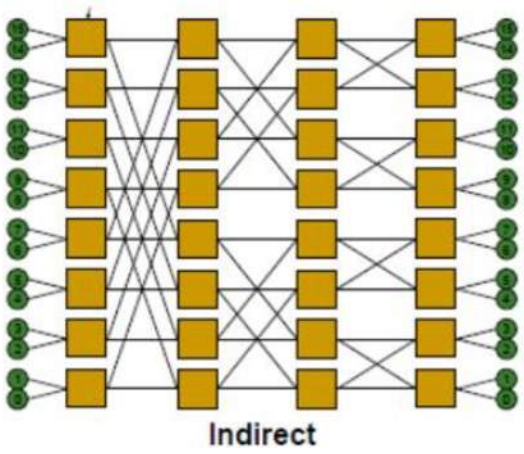
Main Aspects of an ICN[主要因素]

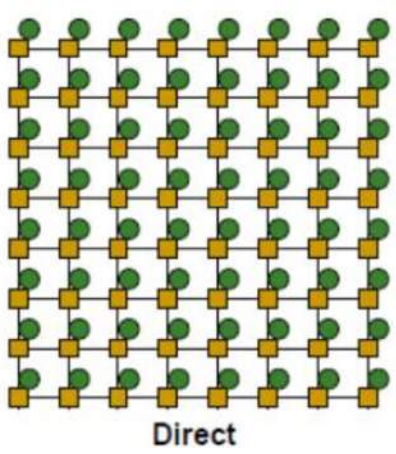
- **Topology**[拓扑]
  - How switches are connected via links
    - Affects routing, throughput, latency, complexity/cost of implementation
- **Routing**[寻路]
  - How a message gets from its source to its destination in the network
- **Flow control**[流控]
  - Allocating network resources (channels, buffers, etc.) to packets and managing contention
- **Switch microarchitecture**[交换机微架构]
  - Internal architecture of a network switch
- **Network interface**[网络接口]
  - How to interface a terminal with a switch
- **Link architecture**[链接架构]
  - Signaling technology and data representation on the channel

Types of Topologies[拓扑类型]

- Direct[直接拓扑结构]
  - Each router is associated with a terminal node
  - All routers are sources and destinations of traffic
- Indirect[非直接拓扑结构]
  - Routers are distinct from terminal nodes
  - Terminal nodes can source/sink traffic
  - Intermediate nodes switch traffic between terminal nodes

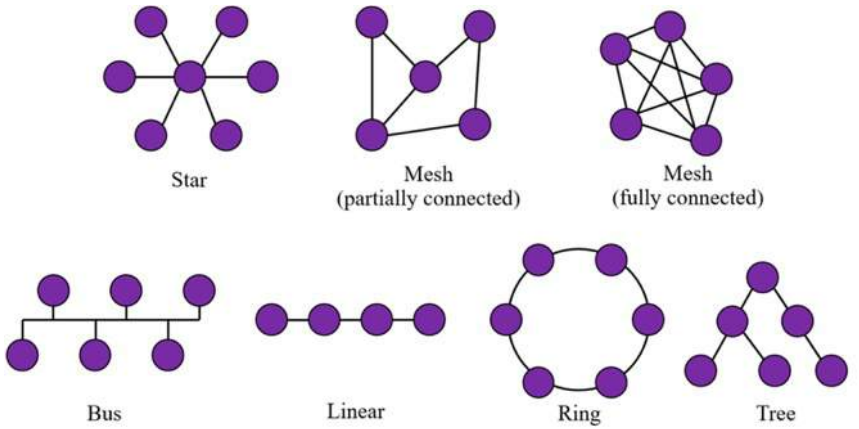






Network Topologies[拓扑]

- Blocking vs. Non-Blocking
  - If connecting any permutation of sources & destinations is possible, network is non-blocking; otherwise network is blocking
- A variety of network topologies have been proposed and implemented
  - These topologies tradeoff performance for cost
  - Commercial machines often implement hybrids of multiple topologies for reasons of packaging, cost, and available components



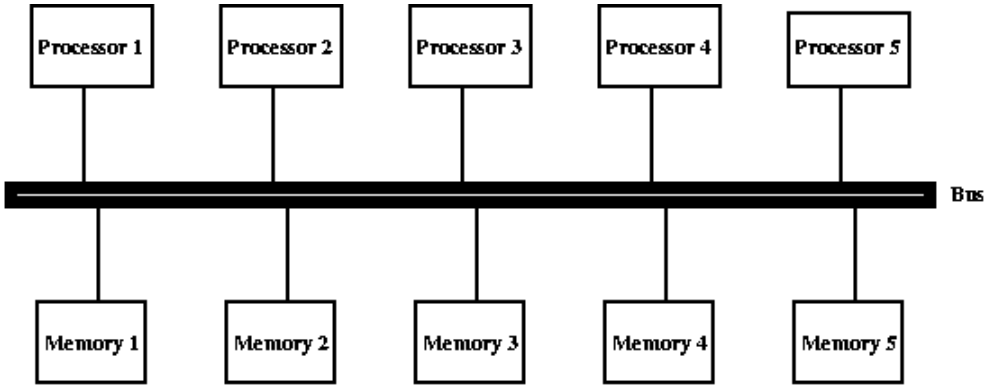
http://www.cs.csi.cuny.edu/~gu/teaching/courses/csc76010/slides/Interconnection%20Network.pdf

Metrics for Comparing Topologies[指标]

- **Switch degree**[交换度]
  - Proxy for switch complexity
- **Hop count**[跳数] (average and worst case)
  - Proxy for network latency
- **Maximum channel load**[最大通道负载]
  - A proxy for hotspot load
- **Bisection bandwidth**[对半带宽]
  - Proxy for maximum traffic a network can support under a uniform traffic pattern
- **Path diversity**[路径多样性]
  - Provides routing flexibility for load balancing and fault tolerance
  - Enables better congestion avoidance

Topologies: Buses[总线]

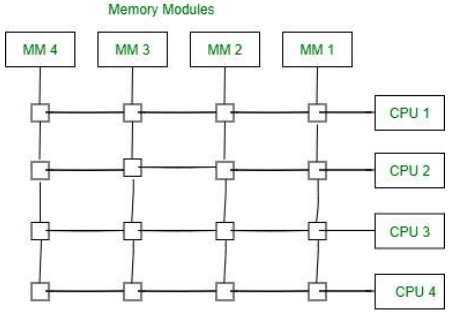
- All processors access a common bus for exchanging data
- The distance between any two nodes is O(1) in a bus. The bus also provides a convenient broadcast media
- However, the bandwidth of the shared bus is a major bottleneck
- Typical bus based machines are limited to dozens of nodes
  - Sun Enterprise servers and Intel Pentium based shared-bus multiprocessors are examples of such architectures

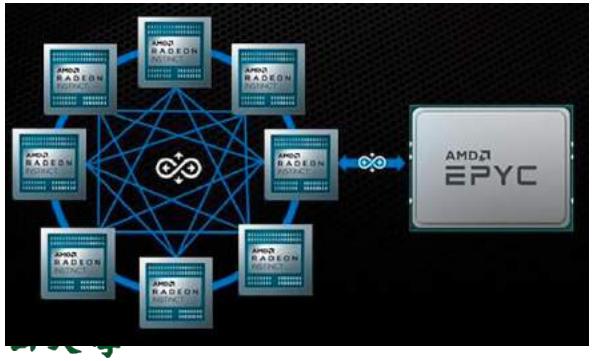


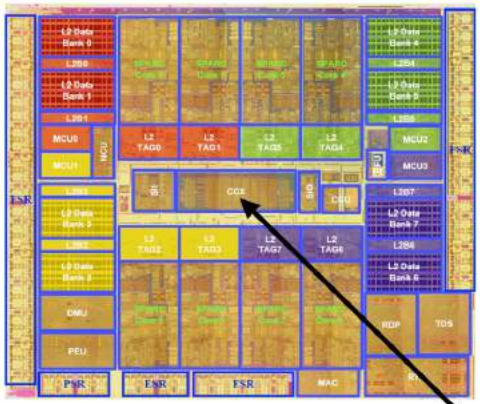
http://www.cs.csi.cuny.edu/~gu/teaching/courses/csc76010/slides/Interconnection%20Network.pdf

Topologies: Crossbars[交叉]

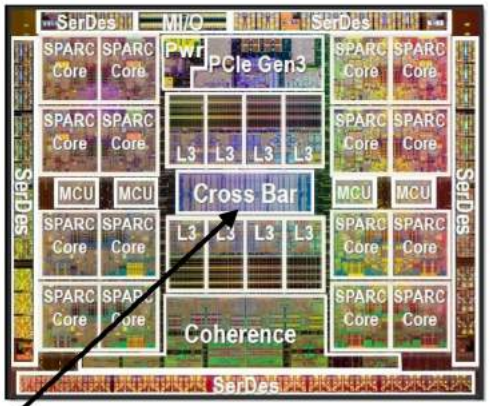
- A crossbar network uses an *p*×*m* grid of switches to connect *p* inputs to *m* outputs in a non-blocking manner
- The cost of a crossbar of *p* processors grows as O(p2)
  - This is generally difficult to scale for large values of *p*
  - Examples of machines that employ crossbars include the Sun Ultra HPC 10000 and the Fujitsu VPP500







Sun SPARC T2 (8 cores, 8 L2 cache banks)



Oracle SPARC T5 (16 cores, 8 L3 cache banks)

Note that crossbar (CCX) occupies about the same chip area as a core

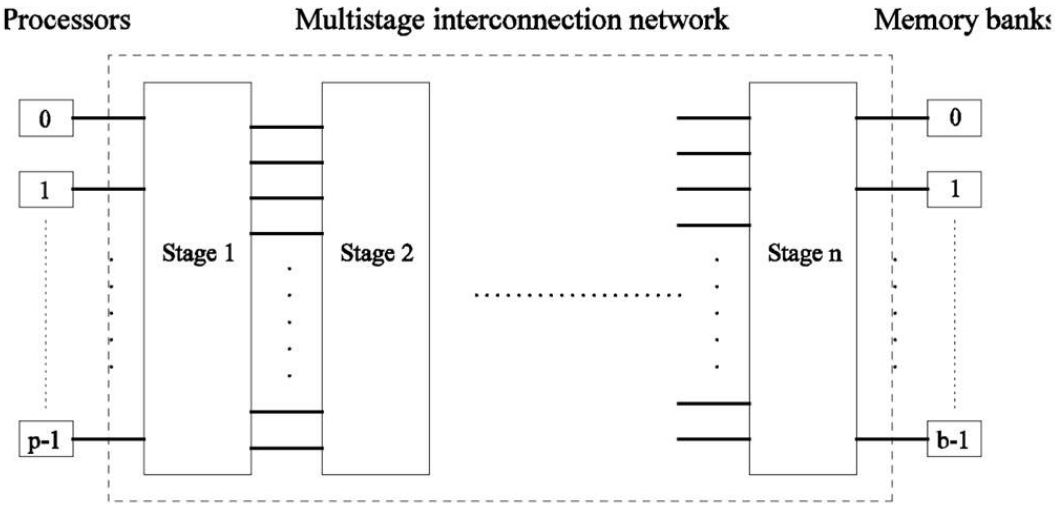
http://15418.courses.cs.cmu.edu/spring2016content/lectures/15_interconnects/15_interconnects_slides.pdf

SUN YAT-SEN UNIVERSITY

http://www.cs.csi.cuny.edu/~gu/teaching/courses/csc76010/slides/Interconnection%20Network.pdf

Topologies: Multistage[多级]

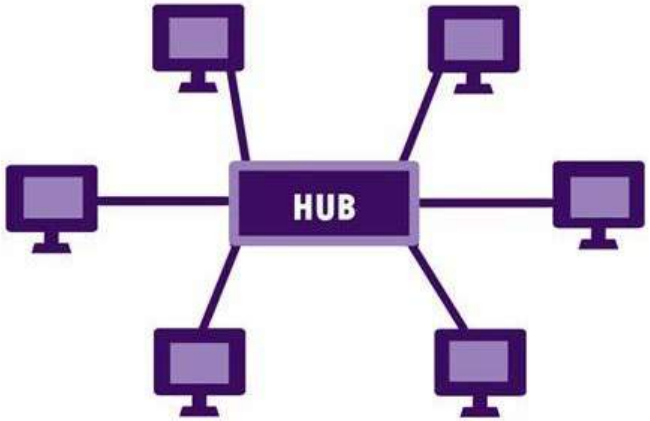
- Indirect network with multiple switches between terminals
- Multistage interconnects strike a compromise between Buses and Crossbars
  - Crossbars have excellent performance scalability but poor cost scalability
  - Buses have excellent cost scalability, but poor performance scalability

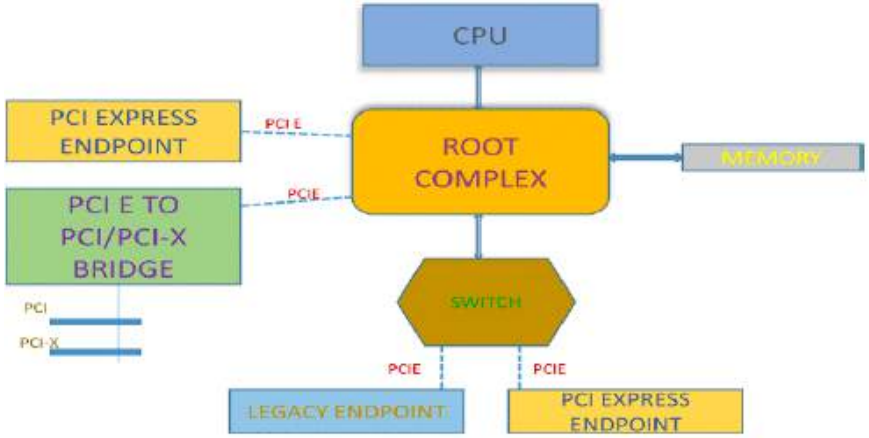


http://www.cs.csi.cuny.edu/~gu/teaching/courses/csc76010/slides/Interconnection%20Network.pdf

Topologies: Star Connected[星型]

- Every node is connected only to a common node at the center
- Distance between any pair of nodes is O(1)
- However, the central node becomes a bottleneck
- In this sense, star connected networks are static counterparts of buses

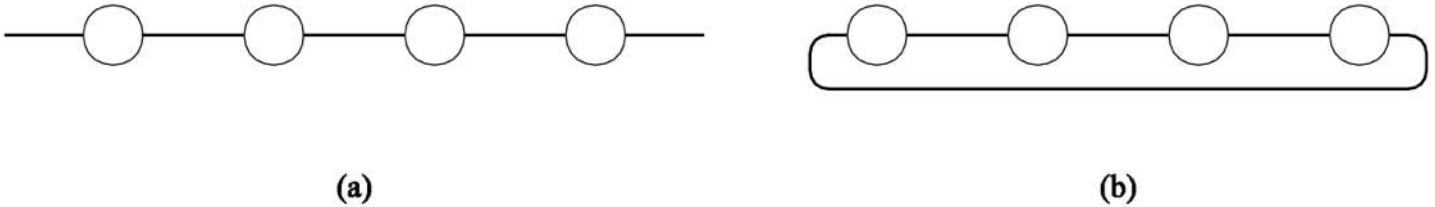




http://www.cs.csi.cuny.edu/~gu/teaching/courses/csc76010/slides/Interconnection%20Network.pdf

Topologies: Linear Arrays, Meshes, …

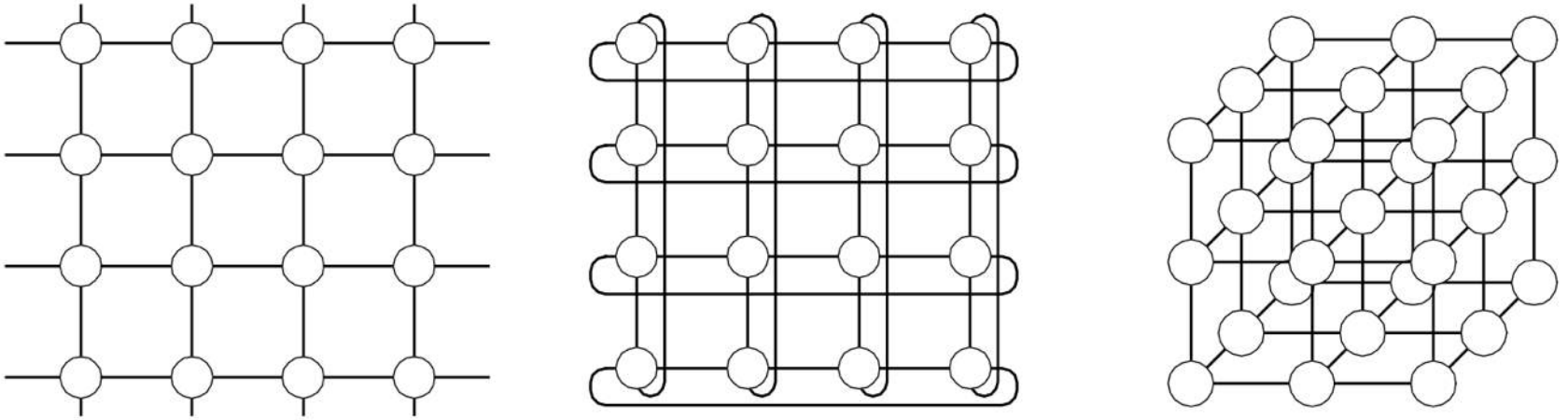
- In a **linear array**, each node has two neighbors, one to its left and one to its right
  - If the nodes at either end are connected, we refer to it as a 1-D torus or a ring
- A generalization to 2 dimensions has nodes with 4 neighbors, to the north, south, east, and west
- A further generalization to *d* dimensions has nodes with *2d* neighbors
  - A special case of a d-dimensional mesh is a hypercube. Here, d = log p, where p is the total number of nodes



(a) with no wraparound links; (b) with wraparound link.

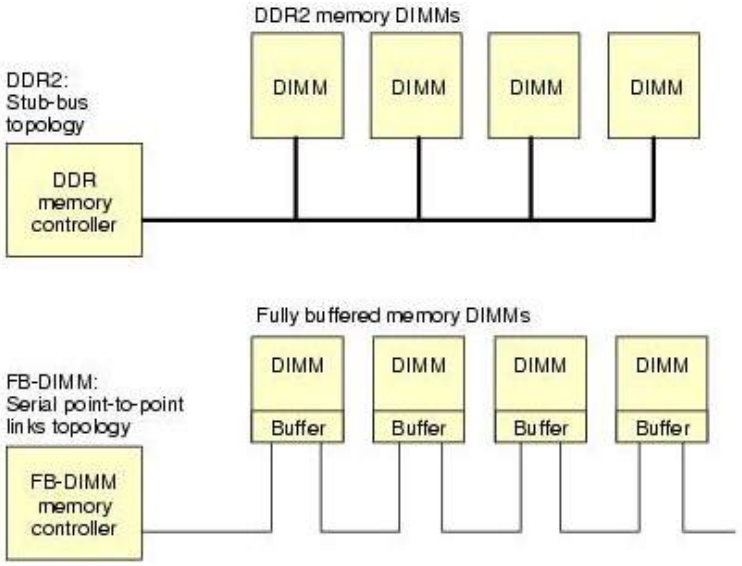
Topologies: Mesh[网格]

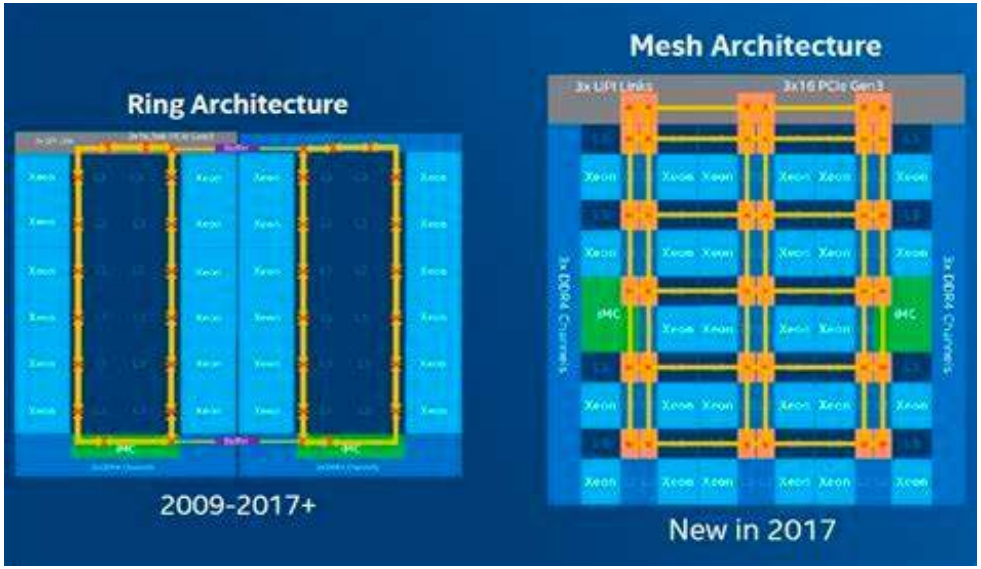
- Two and three dimensional meshes
  - (a) 2-D mesh with no wraparound
  - (b) 2-D mesh with wraparound link (2-D torus)
    - Mesh is not symmetric on edges: performance very sensitive to placement of task on edge vs. middle
    - Torus avoids this problem
  - (c) a 3-D mesh with no wraparound



http://www.cs.csi.cuny.edu/~gu/teaching/courses/csc76010/slides/Interconnection%20Network.pdf

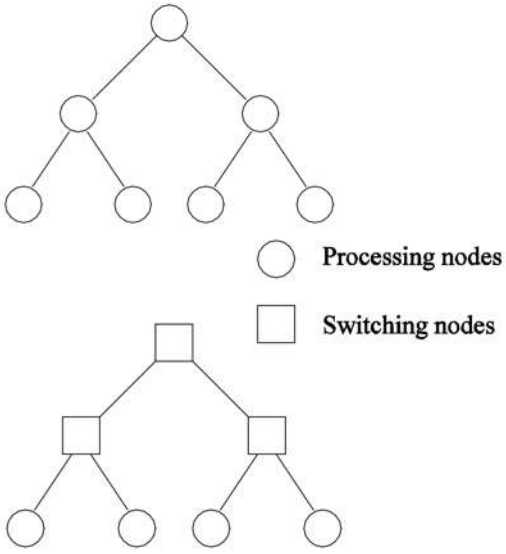
Examples

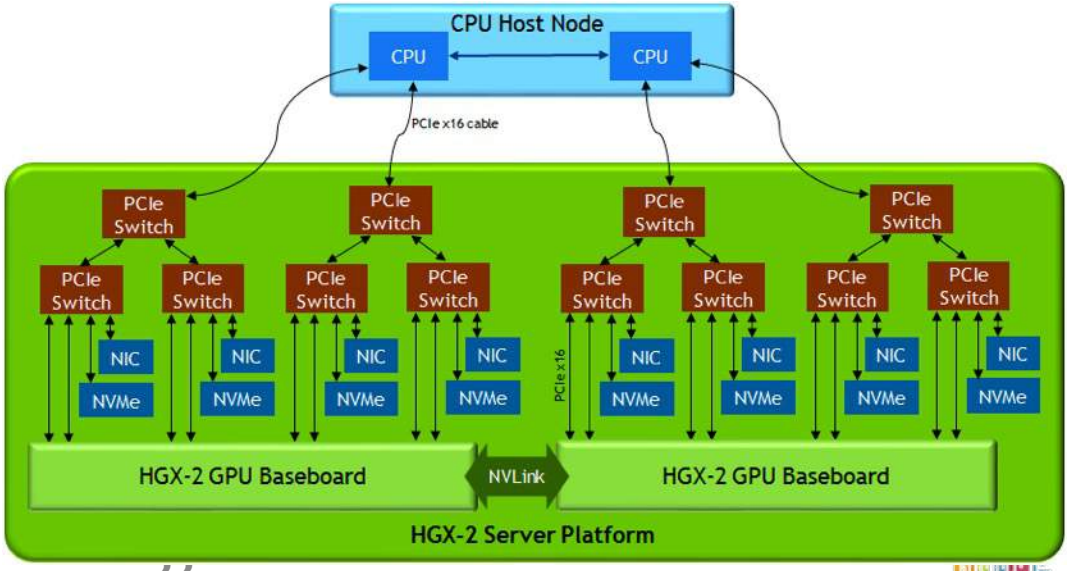

DDR2 memory DIMMs
DDR2:
Stub-bus
topology
DIMM
DIMM
DIMM
DIMM
DDR
memory
controller
Fully buffered memory DIMMs
FB-DIMM:
Serial point-to-point
links topology
DIMM
DIMM
DIMM
DIMM
Buffer
Buffer
Buffer
Buffer
FB-DIMM
memory
controller


Ring Architecture
2009-2017+
Mesh Architecture
New in 2017

Topologies: Tree[树型]

- Diameter and average distance logarithmic
  - *k*-ary tree, height = log*k* N
  - Address specified *d*-vector of radix *k* coordinates describing path down from root
  - Route up to common ancestor and down
- Trees can be laid out in 2D with no wire crossings
  - This is an attractive property of trees

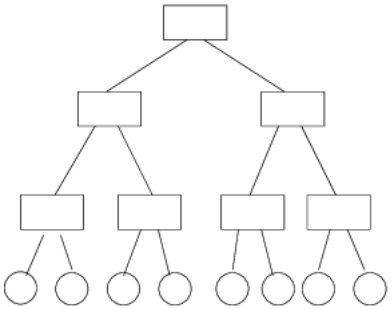




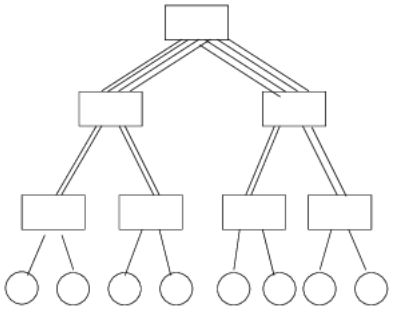
http://www.cs.csi.cuny.edu/~gu/teaching/courses/csc76010/slides/Interconnection%20Network.pdf

Topologies: Fat-Tree[胖树]

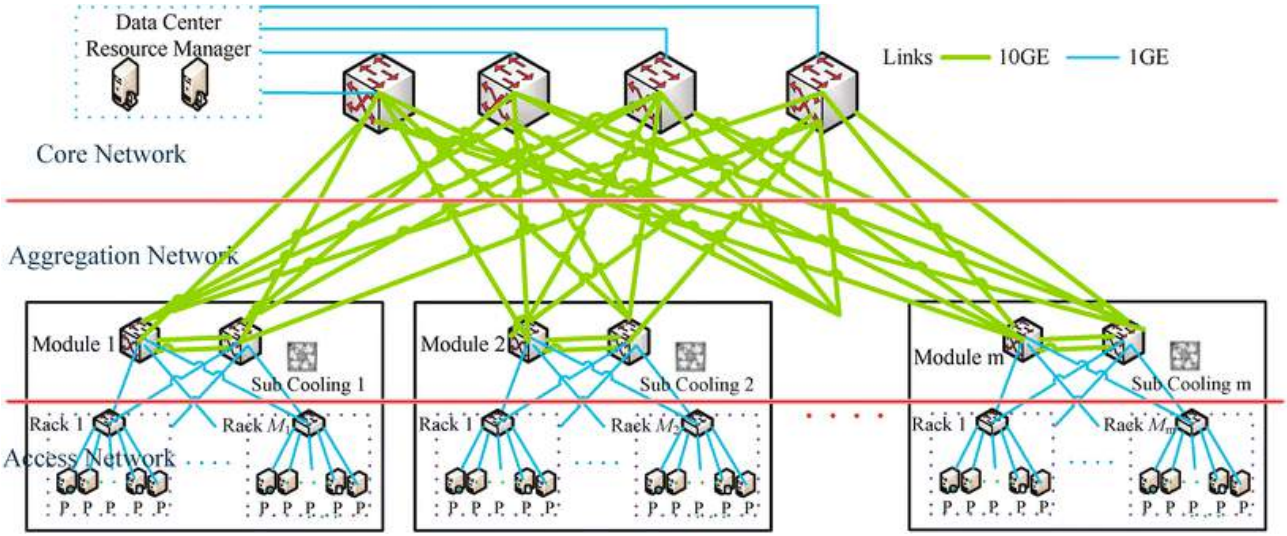
- Links higher up the tree potentially carry more traffic than those at the lower levels
- For this reason, a variant called a **fat-tree**, fattens the links as we go up the tree



(a) Binary tree



(b) Binary fat–tree



http://www.cs.csi.cuny.edu/~gu/teaching/courses/csc76010/slides/Interconnection%20Network.pdf

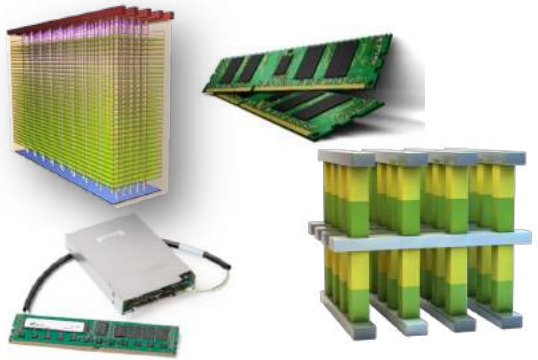
And Other Topologies …

- Many other topologies with different properties discussed in the literature
  - Clos Networks
  - Omega networks
  - Benes networks
  - Bitonic networks
  - Flattened Butterfly
  - Dragonfly
  - Cube-connected cycles
  - HyperX
  - …
- However, these are typically special purpose and not used in general purpose hardware

System Evolution

- Computing
  - Scalar → vector
  - Homogeneous → heterogeneous: CPU + GPU/accelerator/FPGA/DPU

- Memory
  - DRAM, GDDR, HBM
  - HDD, SSD, NVM



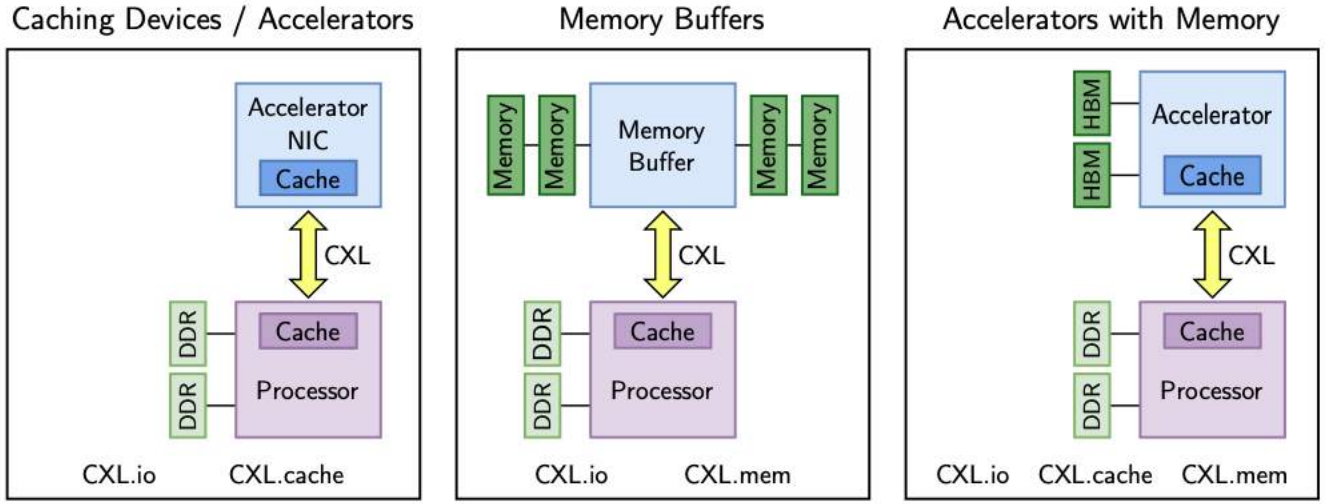
- Interconnect
  - Intra-node: PCI-e, CAPI/OpenCAPI, CCIX, UPI/QPI, NVLink, Gen-Z
  - Inter-node: Ethernet, InfiniBand, Omnipath, HPC Ethernet/Slingshot

Compute Express Link (CXL)

- Unveiled in March 2019
- Open industry standard processor interconnect
  - Unified, coherent memory space between the CPU and any memory attached CXL device.
  - High-bandwidth, low-latency connection between host and devices including accelerators, memory expansion, and smart I/O devices.
  - Utilizes PCI Express 5.0 physical layer infrastructure and the PCIe alternate protocol.
  - Designed to meet demanding needs of HPC work in AI, ML, communication systems through enablement of coherency and memory semantics across heterogeneous processing and memory systems.

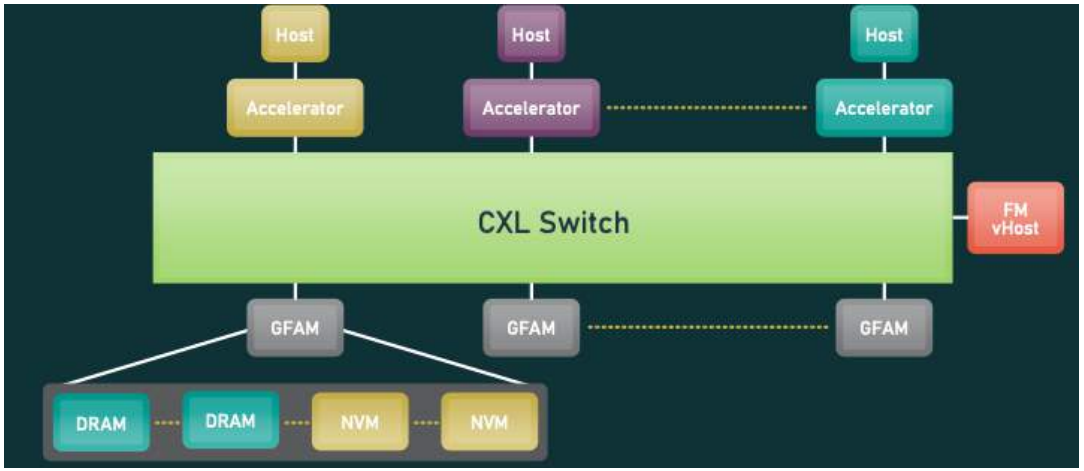
Compute Express Link (cont.)

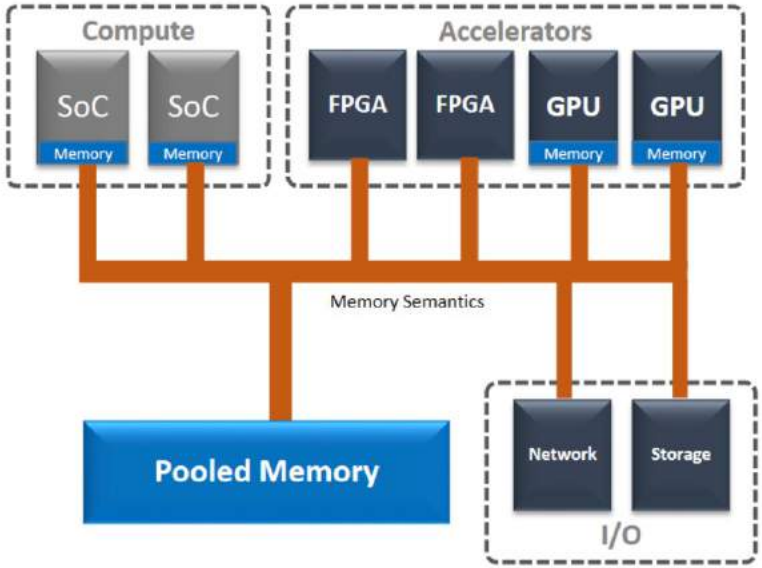
- The CXL transaction layer is comprised of three dynamically multiplexed sub-protocols on a single link
  - CXL.io: functionally equivalent to the PCIe 5.0 protocol
  - CXL.cache: for devices to cache data from the CPU memory
  - CXL.memory: for processor to access the memory of attached devices



CXL-based Memory

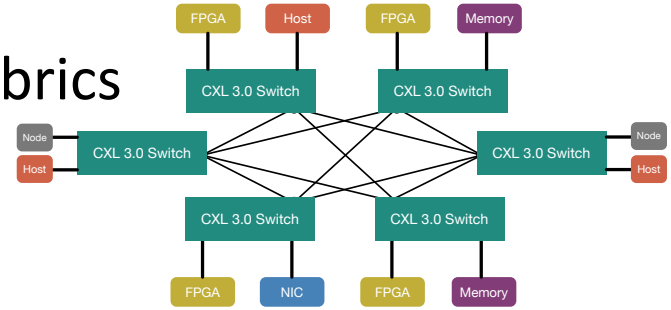
- Unified sharing memory between host and devices
  - No longer use host memory as an intermediary for communication
- Scalable to provide more memory types and capacity
  - Just attached via CXL fabric
  - GFAM: Global Fabric Attached Memory

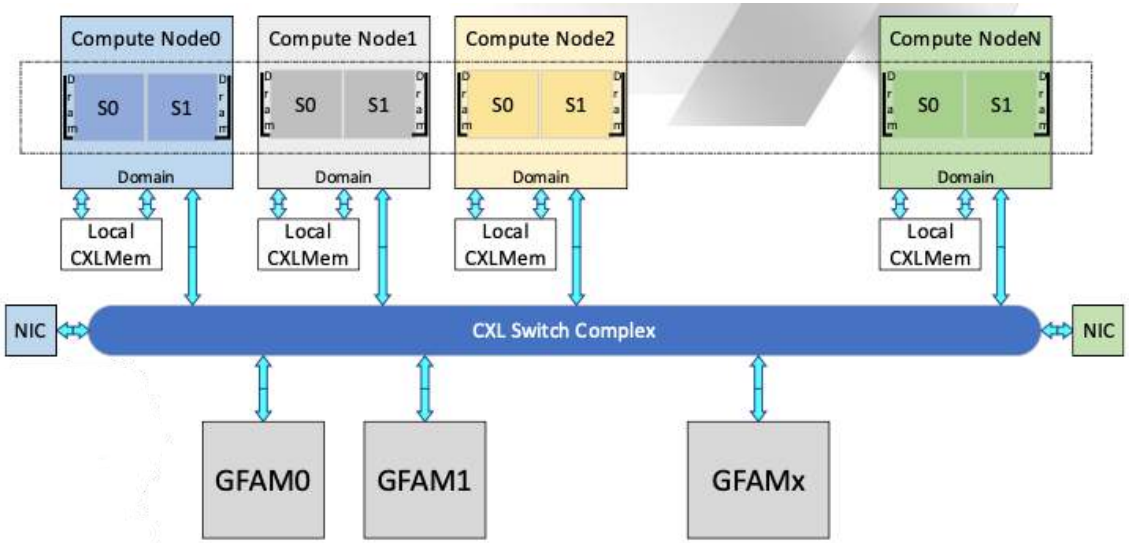


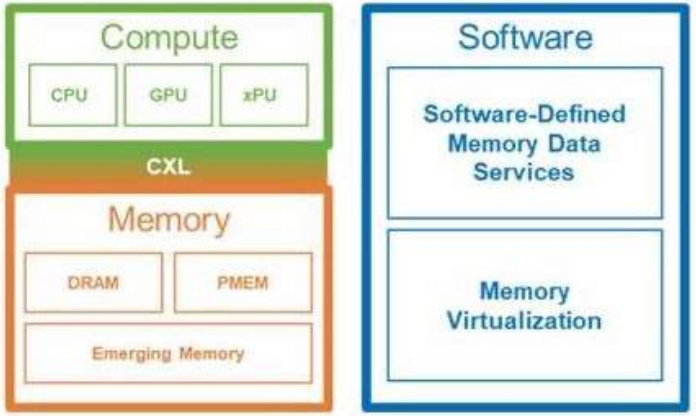


CXL-based System

- Multi-tiered switching
  - Enables the implementation of switch fabrics
  - Switches can connect to other switches
- Rack-scale memory fabric
  - Fine-grained resource sharing across multiple domains







中山大學 SUN YAT-SEN UNIVERSITY

国家超级计算广州中心 NATIONAL SUPERCOMPUTER CENTER IN GUANGZHOU

Computer Architecture

计 算 机 体 系 结 构

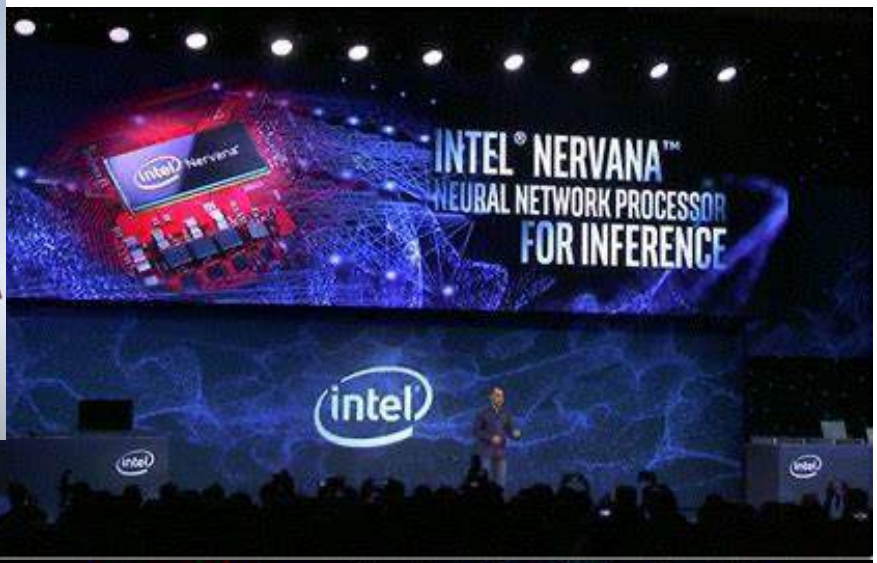
第22讲：Domain Specific Arch (1)

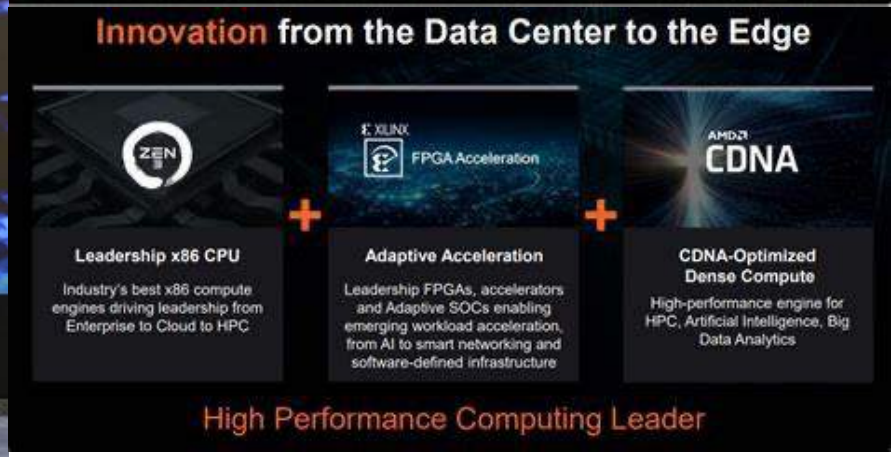
张献伟

xianweiz.github.io

DCS3013, 12/19/2022

HW Companies Building Custom Chips



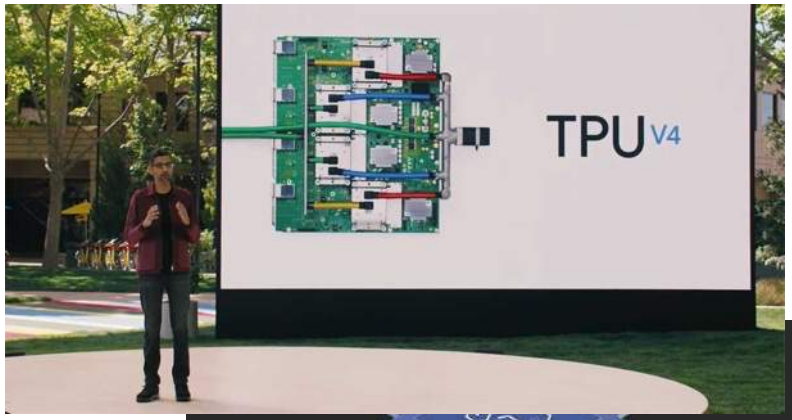


SW Companies are Building HW

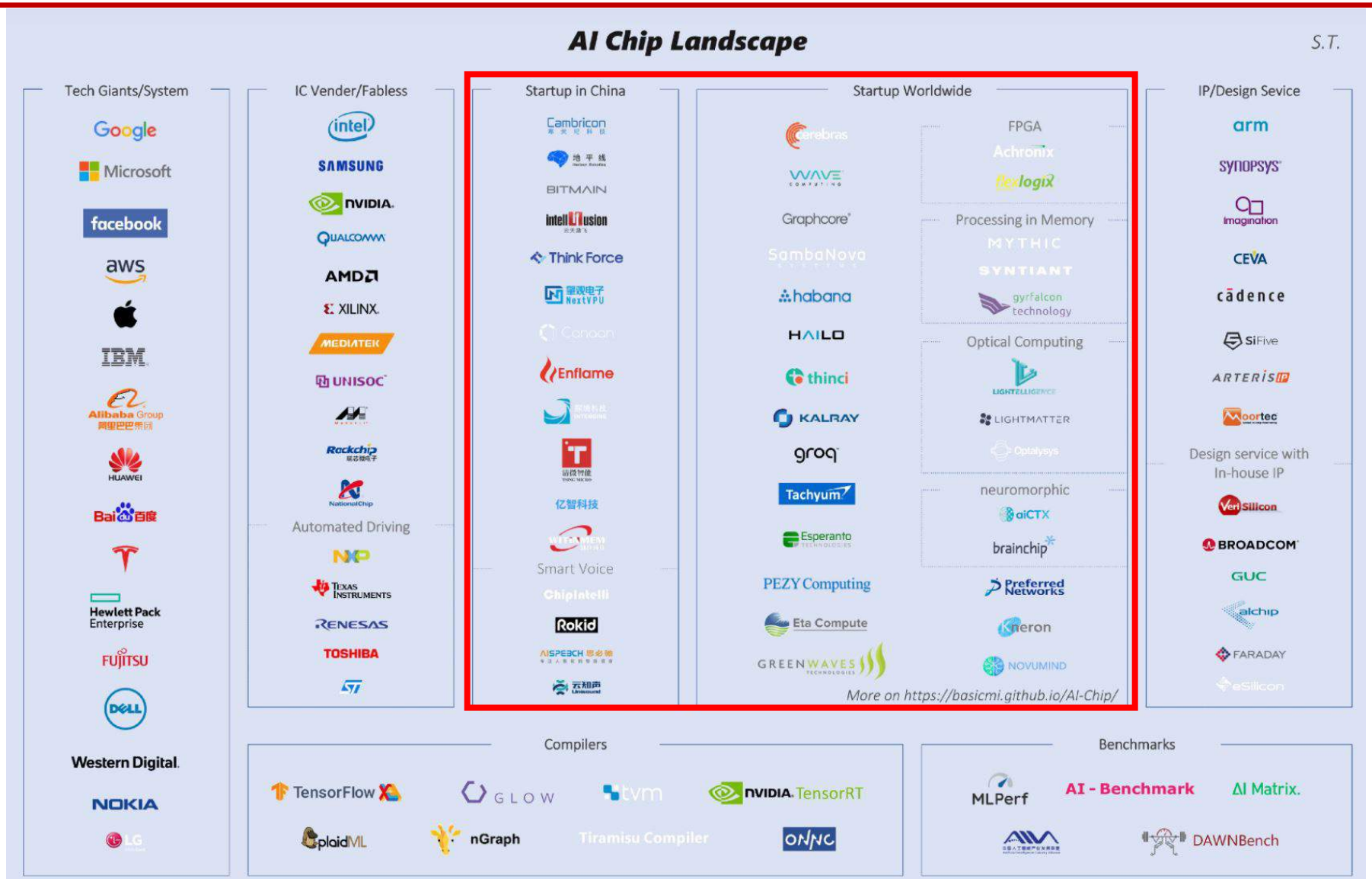
Chips Off the Old Block: Computers Are Taking Design Cues From Human Brains
(September 16, 2017)

After training a speech-recognition algorithm, for example, Microsoft offers it up as an online service, and it actually starts identifying commands that people speak into their smartphones. **G.P.U.s are not quite as efficient during this stage of the process. So, many companies are now building chips specifically to do what the other chips have learned.**

Google built its own specialty chip, a Tensor Processing Unit, or T.P.U. Nvidia is building a similar chip. **And Microsoft has reprogrammed specialized chips from Altera**, which was acquired by Intel, so that it too can run neural networks more easily.



Startups Building Custom Hardware

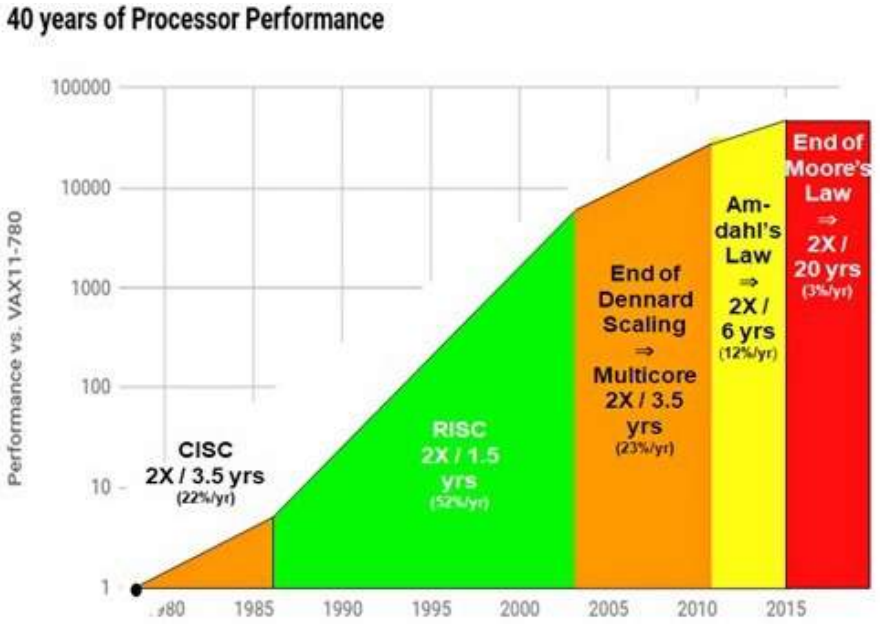


All information contained within this infographic is gathered from the internet and periodically updated, no guarantee is given that the information provided is correct, complete, and up-to-date.

https://www.servethehome.com/dell-emc-talks-deep-learning-and-ai-q3-2019/

Past General-Purpose[通用]

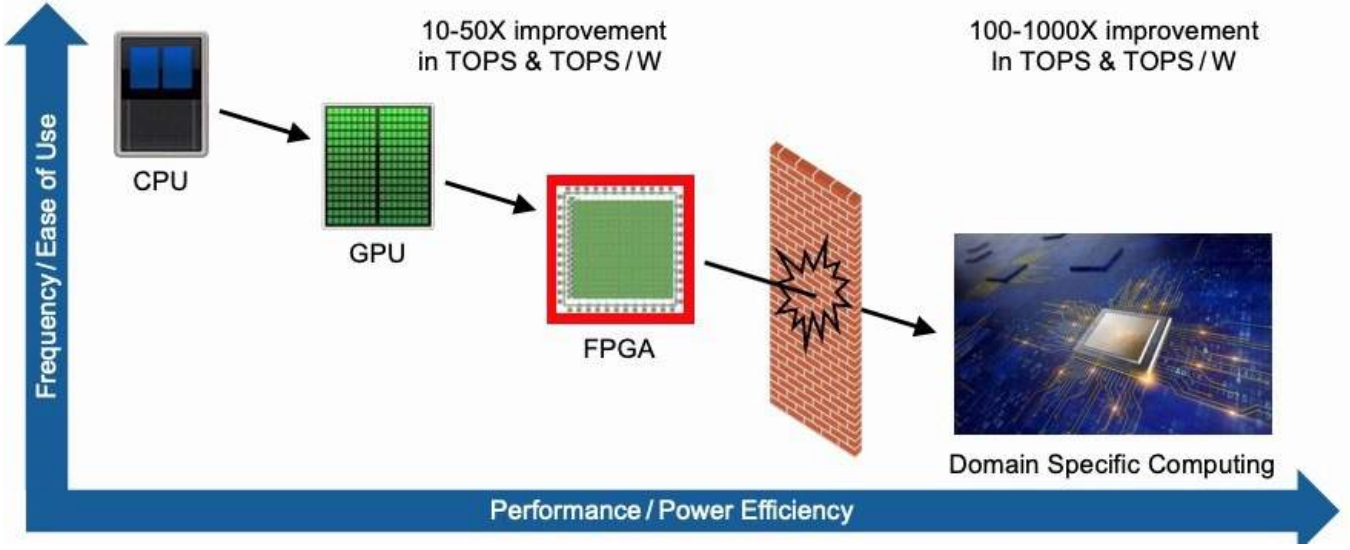
- Moore's Law enabled:
  - Deep memory hierarchy
  - Wide SIMD units
  - Deep pipelines
  - Branch prediction
  - Out-of-order execution
  - Speculative prefetching
  - Multithreading
  - Multiprocessing



- The sophisticated architectures targeted general-purpose code
  - Architects treated code as black boxes
  - Extract performance from software that is oblivious to architecture

Domain-Specific Architecture[领域专用]

- Hard to keep improving performance
  - More transistors means more power
  - Energy budget is limited: higher performance → lower energy/operation
  - Enhancing existing cores may only boost 10% performance
- Need factor of 100 improvements in number of operations per instruction
  - Requires domain specific architectures



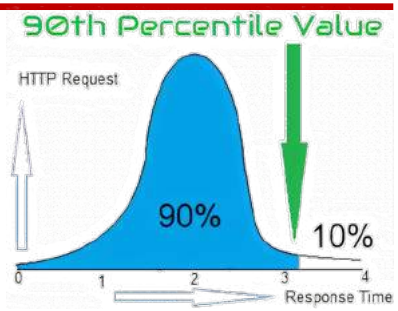
Domain-Specific Architecture (cont.)

- Computers will be much more heterogeneous[异构]
  - Standard processors to run conventional large programs
    - E.g., operating system
  - Domain-specific processors doing only a narrow range of tasks
    - But they do them extremely well
- DSA opportunities[机遇]
  - Preceding architecture from the past may not be a good match to some domains
    - E.g., caches are excellenet general-purpose architectures but not necessarily for DSAs
  - Domain-specific algorithms are almost always for small compute-intensive kernels of larger systems
    - DSAs should focus on the subset and not plan to run the entire program

DSA Challenges[挑战]

- Architects must expand their areas of expertise
  - Must now learn application domains and algorithms
- *Nonrecurring engineering* (NRE) costs[一次性工程成本]
  - Find a target whose demand is large enough to justify allocating dedicated silicon on an SOC or even a custom chip
    - The costs are amortized over the number of chips manufactured, so unlikely to make economic sense if you need only 1000 chips
  - For smaller volume applications, use reconfigurable chips such as FPGAs
    - Several different applications may reuse the same reconfigurable hardware to amortize costs
    - However, the hardware is less efficient than custom chips, so the gains from FPGAs are more modest
- Port software[移植软件]
  - Programming languages and compilers

DSA Design Guidelines[设计准则]



- Why guidelines?
  - Lead to increased area and energy efficiency
  - Provide two valuable bonus effects
    - Lead to simpiler designs, reducing the cost of NRE of DSAs
    - For user-facing apps, better match the 99th-percentile response-time deadlines

Guideline	TPU	Catapult	Crest	Pixel Visual Core
Design target	Data center ASIC	Data center FPGA	Data center ASIC	PMD ASIC/SOC IP
1. Dedicated memories	24 MiB Unified Buffer, 4 MiB Accumulators	Varies	N.A.	Per core: 128 KiB line buffer, 64 KiB P.E. memory
2. Larger arithmetic unit	65,536 Multiply-accumulators	Varies	N.A.	Per core: 256 Multiply-accumulators (512 ALUs)
3. Easy parallelism	Single-threaded, SIMD, in-order	SIMD, MISD	N.A.	MPMD, SIMD, VLIW
4. Smaller data size	8-Bit, 16-bit integer	8-Bit, 16-bit integer 32-bit Fl. Pt.	21-bit Fl. Pt.	8-bit, 16-bit, 32-bit integer
5. Domain-specific lang.	TensorFlow	Verilog	TensorFlow	Halide/TensorFlow

DSA Design Guidelines (cont.)

- *Use dedicated memories to minimize the distance over which data is moved*
  - Hardware cache → software-controlled scratchpad
    - Compiler writers and programmers of DSAs understand their domain
    - Software-controlled memories are much more energy efficient
- *Invest the resources saved from dropping advanced u-arch optimizations into more arithmetic units or bigger memories*
  - Owing to the superior understanding of the execution of programs
- *Use the easiest form of parallelism that matches the domain*
  - Target domains for DSAs almost always have inherent parallelism
    - How to utilize that parallelism and how to expose it to the software?
  - Design the DSA around the natural granularity of the parallelism and expose that parallelism simply in the programming model
    - SIMD > MIMD (i.e., DLP > TLP), VLIW > OoO